UNIVERSIDADE SAGRADO CORAÇÃO

GUILHERME JOÃO LACERDA

PRÁTICAS DE MELHORIA DE PERFORMANCE EM APLICAÇÕES QUE UTILIZAM AS TECNOLOGIAS HTML5, JAVASCRIPT E CSS3

BAURU
2014

GUILHERME JOÃO LACERDA

PRÁTICAS DE MELHORIA DE PERFORMANCE EM APLICAÇÕES QUE UTILIZAM AS TECNOLOGIAS HTML5, JAVASCRIPT E CSS3

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, sob a orientação do Prof. Esp. Alex Setolin Beirigo.

BAURU
2014

Lacerda, Guilherme João.

L131p

Práticas de melhoria de performance em aplicações que utilizam as tecnologias HTML5, Javascript e CSS3 / Guilherme João Lacerda. -- 2014.

60f. : il.

Orientador: Prof. Esp. Alex Setolin Beirigo.

Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade do Sagrado Coração – Bauru – SP.

1. Front-end. 2. Javascript. 3. HTML5. 4. CSS3. 5. Performance. I. Beirigo, Alex Setolin. II. Título.

GUILHERME JOÃO LACERDA

PRÁTICAS DE MELHORIA DE PERFORMANCE EM APLICAÇÕES QUE UTILIZAM AS TECNOLOGIAS HTML5, JAVASCRIPT E CSS3.

Trabalho de Conclusão de Curso apresentado à Universidade Sagrado Coração, para a obtenção do título de bacharel em Ciência da Computação, sob a orientação do Prof. Alex Setolin Beirigo.

Banca examinadora:

_____
Prof. Esp. Alex Setolin Beirigo
Universidade Sagrado Coração

_____
Prof. Ms. Marcio Cardim
Universidade Sagrado Coração

_____
Prof. Esp. André Luiz Ferraz Castro
Universidade Sagrado Coração

Bauru, 28 de novembro de 2014

AGRADECIMENTOS

Agradeço aos meus pais Paulo e Nilza, pelo apoio e confiança que me deram na construção de toda minha vida escolar. Ao meu irmão Gustavo, que me estimulou a concentrar e transmitir de melhor forma idéias neste trabalho. Agradeço também ao meu orientador Alex Setolin Beirigo, por acreditar em mim e me dar forças para seguir em frente. Por fim, agradeço a Deus pela força que foi me dada durante esses anos.

RESUMO

As páginas e sistemas WEB estão evoluindo e tornando-se cada vez mais comuns, sendo acessadas de vários dispositivos, desde computadores pessoais até celulares. Muitos aplicativos que foram originalmente desenvolvidos como aplicações desktop estão sendo portados para WEB, exigindo do programador um conhecimento em tecnologias como Javascript, HTML5 e CSS3.

Com base neste contexto, este trabalho propõe a análise de sites como base para detectar problemas que afetem a performance, bem como aplicar práticas em um ambiente preparado para corrigi-los e melhorar o tempo de acesso.

Palavras-chave: Javascript, HTML5, CSS3, Front-end, Performance.

ABSTRACT

Websites and system WEB are envolving and becoming increasingly common, being accessed from several devices, from computers to cell phones. Many applications that were originally developed as applications desktop are being ported to WEB, requiring the programmer knowledge technologies as Javascript, HTML5 and CSS3. Based on this context, this work proposes the analysis of sites to detect problems that affect performance, and implement practices to correct them and improve access time.

Palavras-chave: Javascript, HTML5, CSS3, Front-end, Performance.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API - Application Programming Interface

AJAX - Asynchronous Javascript and XML

CSS – Cascading Style Sheets

HTML – Hypertext Markup Language

HTTP – Hypertext Transfer Pro

JS - Javascript

PHP – Personal Home Page

URL – Uniform Resource Locator

XML - Extensible Markup Language

W3C – World Wide Web Consortium

SUMÁRIO

1 INTRODUÇÃO

O HTML nasceu no ano de 1991, no laboratório de física de partículas *European Council for Nuclear Research* na Suíça. Foi criado pelo inglês Tim Berners-Lee, de 44 anos com o objetivo de interligar computadores do laboratório e outras instituições de pesquisas, para exibir documentos científicos de forma simples e de fácil acesso.

Com o passar do tempo, a utilização de ferramentas para autoria HTML aumentou, tendo a necessidade de tornar a sintaxe cada vez mais robusta. Logo foram lançadas novas versões do HTML, como o HTML 2.0 em 1994, o HTML 3.0 ainda neste mesmo ano, o HTML 3.2 em 1995, o HTML 4.0 em 1999 e o HTML5, que começou a ser desenvolvido em 2008 e sua versão final está prevista para o fim de 2014.

O HTML5 foi criado exatamente para dar significado semântico às páginas web, além de padronizar e facilitar o entendimento das sessões de um site pelos navegadores. (SIMONI, 2013).

A medida que o HTML foi se popularizando e evoluindo, começou a surgir a necessidade de se definir o layout dos documentos. No ano de 1994 foi criada a linguagem CSS por Hakon Wium Lie, justamente para resolver esse problema.
Segundo Pereira (2009), o CSS é uma folha de estilo composta por camadas e utilizada para definir a apresentação em páginas da internet que adotam para o seu desenvolvimento linguagens de marcação, como o HTML.

Em 1995 a proposta do CSS foi apresentada para a *W3C*, que estava acabando de nascer. A proposta foi aceita e foi criada uma equipe liderada por Hakon para continuar o desenvolvimento do CSS.

Para tornar a Internet mais dinâmica, Brendan Eirch da Netspace criou a linguagem de programação Javascript, no ano de 1995. JavaScript é uma linguagem de programação interpretada. Foi originalmente implementada como parte dos navegadores web para que scripts pudessem ser executados do lado do cliente e interagissem com o usuário sem a necessidade deste script passar pelo servidor, controlando o navegador, realizando comunicação assíncrona e alterando o conteúdo do documento exibido. (PRADO, 2014).

Foi rapidamente aceito como a principal linguagem de script client-side de páginas *WEB*, mas sua popularidade culminou com o advento do *AJAX*, um uso metodológico de tecnologias como Javascript e *XML*, que utilizam de solicitações assíncronas de informações, permitindo que sejam feitas requisições ao servidor sem precisar recarregar a página de novo.

2 OBJETIVOS

2.1 OBJETIVO GERAL

Analisar o desempenho em aplicações WEB que utilizam as tecnologias HTML5, Javascript e CSS3, para encontrar os principais problemas que possam prejudicar a performance, bem como apresentar práticas para corrigir estes problemas e melhorar a velocidade de acesso.

2.2 OBJETIVOS ESPECÍFICOS

- Identificar os três sites mais acessados no Brasil que utilizam recursos de HTML5, Javascript e CSS3.
- Através de ferramentas de diagnóstico de performance, verificar os principais problemas que estejam prejudicando a performance de cada um dos três sites escolhidos, utilizando diferentes navegadores.
- Demonstrar práticas para corrigir os problemas listados pelas ferramentas de diagnóstico de performance.

3 JUSTIFICATIVA

Apesar do Flash ainda ser muito utilizado em diversos sites para a criação de animações, jogos, infográficos ou para visualização de dados mais interessantes, esta tecnologia será descontinuada, pois diversos sistemas operacionais estão deixando de adotar o Flash por questões de segurança, desempenho e passando adotar o HTML5 como tecnologia principal.

Com a popularização do HTML5, surgiram novos desafios para os programadores *Front-End* como a necessidade de aprender a linguagem de programação Javascript e a linguagem de folhas de estilo CSS a fim de criar páginas mais atraentes e dinâmicas.

As linguagens Javascript e CSS possuem alto nível de complexidade. Existem várias formas de utilizá-las para se alcançar o resultado esperado, mas nem sempre são empregadas da maneira correta, comprometendo a performance.

Essas aplicações devem ser as mais otimizadas possíveis, devido ao processamento limitado e conexão lenta de dispositivos móveis.

Este trabalho acadêmico visa auxiliar os desenvolvedores *Front-End* a conseguirem melhorar a performance de suas aplicações WEB através de inúmeros testes e práticas abordados nesta monografia.

4 FUNDAMENTAÇÃO TEÓRICA

4.1 QUALIDADE DE SOFTWARE

Qualidade é um termo subjetivo, de difícil definição, relacionado diretamente às percepções de cada indivíduo, de acordo com sua cultura, região, formação, entre outros fatores que influem diretamente nesse conceito, (JURAN, 1998).

Segundo Juran (1998, p. 21) qualidade significa “aquelas características dos produtos que atendem as necessidades dos clientes e, assim, proporcionam a satisfação do mesmo”.

Ainda de acordo com Juran (1998, p. 22), qualidade pode ser definida como algo “livre de deficiências”, ou seja, livre de erros que requerem retrabalho, ou que resultem em falhas, insatisfação dos clientes, entre outros.

As palavras sobre qualidade de Juran também se encaixam na qualidade de software. A qualidade de software é determinada pela qualidade nos processos que foram utilizados para desenvolver esse software. Ou seja, para melhorar a qualidade de um software, deverá ser melhorada a qualidade dos processos. Pode também ser medida através do grau de satisfação em que as pessoas avaliam determinado produto ou serviço.

A qualidade se torna cada vez mais necessária hoje em dia para que um produto seja competitivo. Um dos problemas que afetam seriamente a qualidade de um software é a performance, o que será abordado nesta monografia. Problemas de performance geram filas nos servidores, causando lentidões e até mesmo perda de informações.

4.2 TESTE DE PERFORMANCE

Testes de performance são essenciais por avaliarem a capacidade de resposta de um sistema em determinados cenários e configurações e por permitirem planejar melhorias no ambiente para atender a demandas atuais e futuras. (Tomaz, 2013).

O teste de performance é um método de investigar as características de uma aplicação relacionadas a aspectos de qualidade, que podem impactar os usuários reais, submetendo a mesma a simulações baseadas em situações reais. (Barber, 2004).

No contexto das aplicações web, Pressman (2005) afirma que se um usuário tem de esperar muito tempo (para acesso, processamento do lado do servidor, para formatação ou exibição do lado do cliente), ele ou ela pode decidir ir para outro lugar.

O teste de desempenho é projetado para testar o desempenho do software durante a execução de uma aplicação, visando descobrir situações que levam a uma possível falha de software. É um tipo de teste que serve para determinar o tempo de resposta, a confiabilidade e a escalabilidade da aplicação sob uma determinada carga de trabalho.

O objetivo desses testes é eliminar possíveis gargalos e estabelecer uma base para os testes de regressão futuros. Em outras palavras, serve para tornar a aplicação mais estável e livre de bugs. A figura 1 mostra um gráfico sobre o tempo de resposta de aplicações e a aceitação dos usuários.

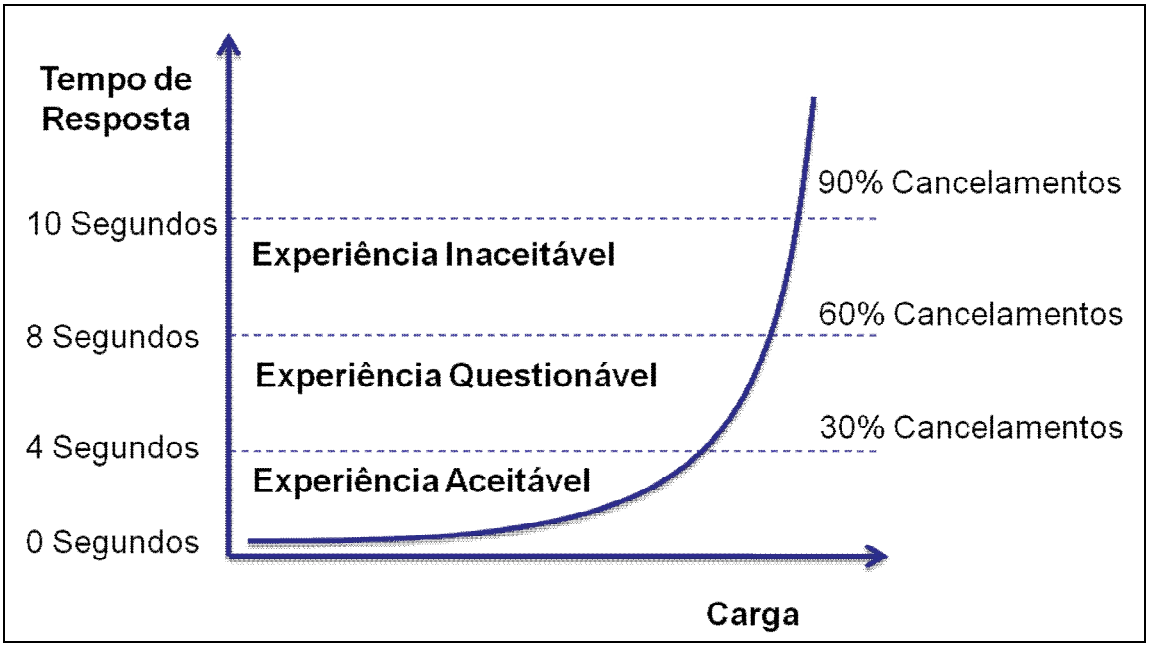


Figura 1 - Gráfico tempo resposta x aceitação usuários.

Fonte: CARRÉRA (2012).

Os resultados dos testes são usados para determinar um padrão de comportamento e capacidade da aplicação que está sendo testada. Esses limites e padrões são chamados de baseline e servem de referência para testes de carga e stress ou outros que estejam sendo planejados/executados no ambiente.

Logo, é fundamental que, desde o início do ciclo de vida da aplicação, a performance seja priorizada, para que todo um ambiente de testes seja preparado de maneira adequada para simular o ambiente de produção e desse modo auxiliar o desenvolvimento a fim de atingir o nível de qualidade desejado.

4.3 HTML5

Bozza (2011) afirma que o HTML5 é uma linguagem de marcação de hipertexto para apresentar e estruturar o conteúdo na web.

Um dos principais objetivos do HTML5 é facilitar a manipulação do elemento possibilitando o desenvolvedor a modificar as características dos objetos de forma não intrusiva e de maneira que seja transparente para o usuário final. (EIS; FERREIRA, 2012, p. 29).

Eis; Ferreira (2012, p. 30) afirmam que o HTML5 também cria novas tags e modifica a função de outras. As versões antigas do HTML não tinham um padrão universal para a criação de seções comuns e específicas como rodapé, cabeçalho, sidebar, menus e etc.

O HTML5 modifica a forma de como é escrito o código e são organizadas as informações na página, de uma forma mais semântica e simples. Com base na definição anterior, as Figuras 2 e 3 mostram uma comparação da estrutura de uma página em HTML4 e HTML5.

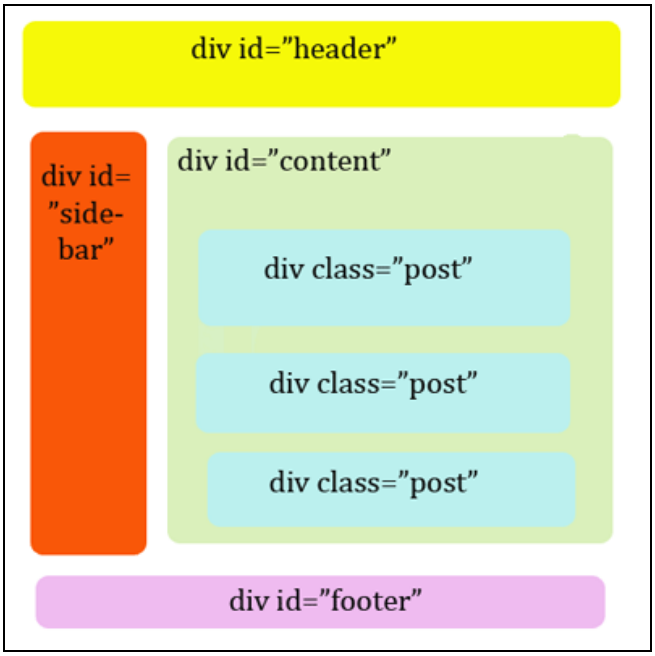


Figura 2 - Estrutura do HTML4

Fonte: LAWSON (2011).

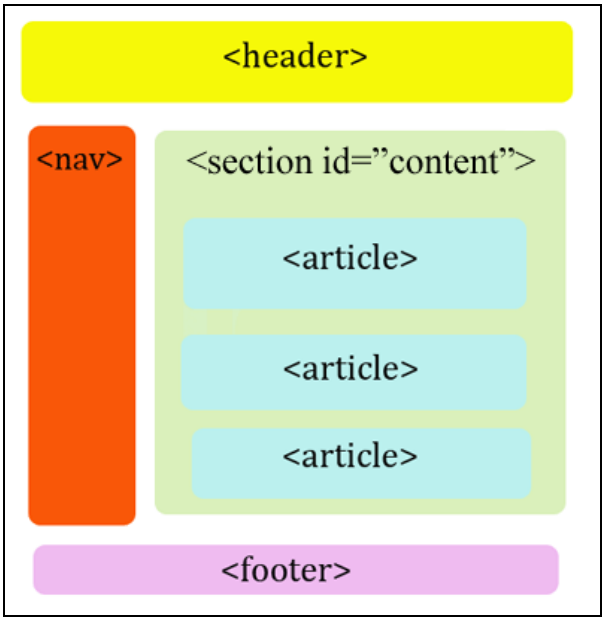


Figura 3 - Estrutura do HTML5.

Fonte: LAWSON (2011).

O HTML5 possui um conjunto de novas funcionalidades encapsuladas em APIs que podem ser acessadas via Javascript. Está sendo desenvolvido com o objetivo de trabalhar de forma independente, ou seja, não depende de outros plugins como por exemplo o Flash para exibir animações. A Figura 4 mostra a estrutura básica do HTML5.

Figura 4 - Código do HTML5.

Fonte: Elaborado pelo autor.

4.4 CSS 3

É a mais nova versão das Cascading Style Sheets (CSS), que são folhas de estilo responsáveis em apresentar o visual dos sites. (RODRIGUES, 2012).

Silva (2012, p. 25) afirma que “o CSS têm por finalidade devolver à marcação HTML/XML o propósito inicial da linguagem”. O HTML foi criado para ser uma linguagem exclusivamente de marcação e estruturação de conteúdo, ou seja, não cabe ao HTML fornecer informações ao agente de usuário sobre a apresentação dos elementos, como por exemplo: cores de fontes, tamanhos de texto, apresentação dos elementos.

O objetivo inicial do CSS era separar o conteúdo da forma. Entretanto, o objetivo de oferecer um controle total aos designers sobre os elementos da página foi mais difícil de cobrir, pois os desenvolvedores ainda continuavam a usar truques diversos para conseguir efeitos comuns, como por exemplo a borda arredonda.

Segundo Alvarez (2008), a novidade mais importante que o CSS 3 traz para os desenvolvedores consiste na incorporação de novos mecanismos para manter um maior controle sobre o estilo com o qual se mostram os elementos das páginas, sem ter que recorrer a truques ou hacks, que muitas vezes complicavam o código.

A figura 5 mostra uma comparação entre o CSS 2 usando hacks e o CSS 3 para arredondar as bordas de um elemento.

```
3   /* CSS2 utilizando hack para borda arredonda */
4   #borda {
5     position:relative;
6     behavior: url(border-radius.htc);
7     -webkit-border-radius: 10px;
8     -moz-border-radius: 10px;
9     border-radius: 10px;
10  }
11
12  /* CSS3 com borda arredonda nativa */
13  #borda {
14      border-radius: 10px;
15  }
16
```

Figura 5 - Código de comparação entre CSS2 e CSS3.

Fonte: Elaborado pelo autor.

4.5 JAVASCRIPT

Segundo Silva (2010, p. 23), “a linguagem de scripts Javascript foi criado pela Netscape em parceria com a Sun Microsystems, com a finalidade de fornecer um meio de adicionar interatividade a uma página web”.

Silva (2010, p. 29) ressalta que “linguagem de script é uma linguagem de programação usada para manipular, personalizar e automatizar as funcionalidades de um sistema existente”. Essas funcionalidades já se encontram disponíveis por meio de uma interface do usuário e a linguagem de script provê um mecanismo para acessá-las.

> Silva (2010, p. 23) explica que “o Javascript é uma linguagem desenvolvida para rodar no lado do cliente, isto é, a interpretação e o funcionamento da linguagem dependem de funcionalidades hospedadas no navegador do usuário. Isso é possível porque existe um interpretador JavaScript hospedado no navegador”.

A figura 6 mostra um exemplo de código de Javascript para mostrar uma caixa de alerta para o usuário.

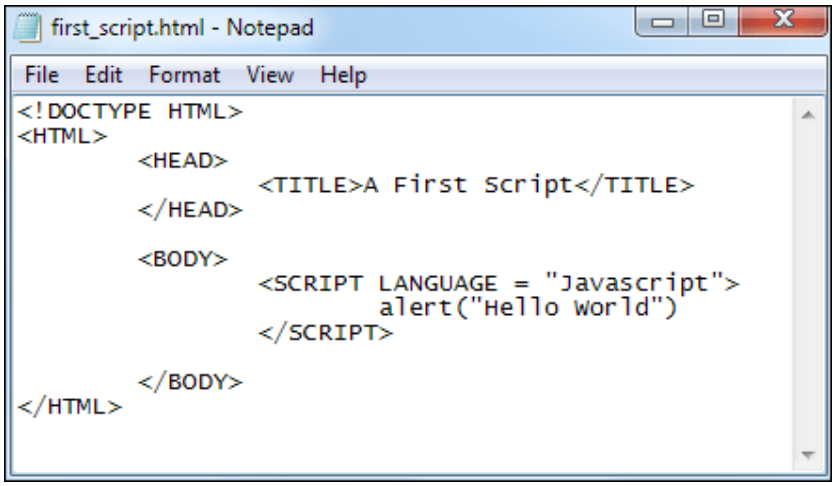


Figura 6 - Exemplo de código de Javascript.

Fonte: Elaborado pelo autor.

A figura 7 mostra o resultado do código da figura 05 sendo executado pelo navegador.

Figura 7 - Resultado do código Javascript.

Fonte: Elaborado pelo autor.

Também é possível usar Javascript do lado do servidor. Tanto a Netscape como a Microsoft desenvolveram interpretadores Javascript para serem hospedados no servidor. Porém, essa prática não é muito comum, e na maioria das vezes o Javascript é rodado no navegador do usuário.

Um dos recursos mais utilizados do Javascript hoje em dia é o AJAX, um uso metodológico de tecnologias como o Javascript e o XML.

Para Niederauer (2013, p. 17), “AJAX vem da expressão *Asynchronous JavaScript and XML”*. É o uso sistemático de Javascript e XML, entre outras

tecnologias, para tornar o navegador mais interativo com o usuário, utilizando-se solicitações assíncronas de informações. Assim é possível fazer uma solicitação ao servidor WEB sem que seja necessário recarregar a página acessada.

Esta tecnologia é muito utilizada por grandes sites, como Facebook e Gmail, permitindo assim que todo o conteúdo do site seja mostrado em apenas uma única página, a qual recebe o nome de *Single-Page-Application* e dispensa o uso de âncoras para redirecionamento.

A figura 8 mostra um diagrama de como é feita uma requisição AJAX para um servidor web.

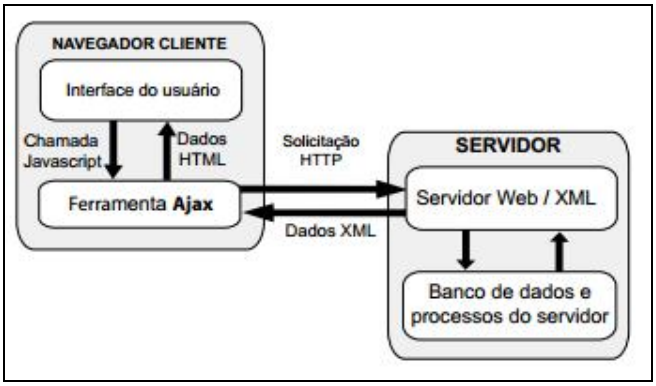


Figura 8 - Diagrama requisição AJAX.

Fonte: NIEDERAUER (2007, p. 17).

Em uma requisição normal sem o uso da tecnologia AJAX, a resposta do servidor é feita diretamente para a interface do usuário, através de uma linguagem de programação *back-end*, como o PHP.

4.6 HISTÓRIA DOS NAVEGADORES

Segundo Olhar Digital, (2009 citado por CORREIA, 2009) com o advento da Internet, ampliou-se o campo das informações e, para utilizarmos todos os recursos disponíveis, são necessários softwares que possibilitem essa busca, conhecidos como navegadores.

O browser é um dos principais softwares de um computador. Vários sistemas hoje em dia são acessados via navegador, e cada vez mais os sistemas via software estão deixando de existir.

O primeiro navegador foi criado por Tim Berners-Lee em 1990, chamado WorldWideWeb. Desde então, o desenvolvimento dos navegadores tem sido intrisecamente ligado ao desenvolvimento da própria Web.

O navegador WorldWideWeb não fez muito sucesso por ser um navegador de modo texto. Entretanto, o que fez a Web ficar realmente popular foi a introdução do navegador NCSA Mosaic, um navegador gráfico rodado originalmente no Unix, que logo foi portado para o Apple Macintosh e Microsoft Windows. A primeira versão do Mosaic foi lançada em 1993 por Marc Andreesen, o líder do projeto.

Abaixo, um pouco da história dos principais navegadores, que são: Chrome, Firefox, Intert Explorer, Mosaic, Netscape, Opera, Safari e WWW.

4.6.1 WWW (1991)

Foi o primeiro navegador web. Foi desenvolvido por Tim Berners-Lee e rodava no sistema operacional NeXTEP. No início sua interface era bem simples, a maior parte das informações eram no formato de texto, com poucas imagens. Mais tarde, para não ser confundido com a própria rede, teve seu nome alterado para Nexus.

4.6.2 Mosaic (1993)

O Mosaic foi criado por uma equipe de estudantes do Centro Nacional de Aplicações de Supercomputação (NCSA), na Universidade de Illinois. Foi primeiro navegador capaz de exibir texto e imagens na mesma página. Isso tornou a internet mais atraente e foi determinante para a abertura da web para o público em geral. Marc Andreesen, o líder do time que desenvolveu o Mosaic saiu da NCSA, e junto com Jim Clark, fundou a Mosaic Communications, que depois foi renomeada para Netscape Communications, onde foi desenvolvido o navegador Netscape.

4.6.3 Netscape (1994)

O Netscape foi um navegador muito popular na sua época, pois trazia todas as características que um browser moderno oferece nos dias atuais, como por exemplo, a navegação por abas, o bloqueio de pop ups, suporte a cookies e histórico de visitas, entre outros.

Durante muitos anos foi o navegador mais utilizado, mas em 2002 já restavam poucos usuários devido ao fato da Microsoft passar a incluir o Internet Explorer junto com o sistema operacional Windows.

4.6.4 Internet Explorer (1995)

Em 1995, a Microsoft entra na briga dos navegadores com seu Internet Explorer 1.0, parte integrante do pacote adicional do Windows 95. Com isso, teve início a “guerra dos navegadores”.

Segundo Olhar Digital (2009 citado por CORREIA, 2009), a guerra dos naveadores Web é o nome dado a um período de quatro anos (de 1995 a 1999) no qual a empresa Netscape, produtora do software browser homônimo, perde a sua liderança absoluta no mercado de navegadores para a Microsoft.

Este período resultou em uma reversão total no uso de um software para outro, além de gerar projetos como o Mozilla e o Opera.

O motivo da rápida popularidade do Internet Explorer foi devido à interligação ao sistema operacional Windows. Os usuários automaticamente o utilizavam ao invés de fazer o download de outro navegador. Porém, o Internet Explorer é alvo de inúmeras falhas de segurança e arquitetura. Também é bastante criticado pelos desenvolvedores Web devido a sua complexidade para renderizar HTML e CSS corretamente.

A versão atual do Internet Explorer é a 11, e se encontra bem mais madura e compatível com as últimas funcionalidades do HTML5. No entanto, seu legado acaba prejudicando sua imagem, e por isso ainda é bastante ignorado pela comunidade de desenvolvedores.

4.6.5 Mozilla (1998)

O navegador Mozilla Firefox é um navegador livre e multi-plataforma que foi desenvolvido pela Mozilla Foundation com a ajuda de centenas de colaboradores. Nasceu do código fonte do Netscape.

O Firefox é um dos maiores rivais do Internet Explorer, e com sua criação foi reativada a chama Guerra dos Navegadores.

A comunidade Mozilla é bastante focada na parte de desenvolvimento. Por isso, o Firefox é um navegador recomendado para desenvolvedores Front-End, com inúmeras ferramentas para análise de código fonte e testes de performance.

Em comparação com o Internet Explorer, a versão mais recente do Firefox é bem mais estável para rodar aplicações HTML5. Seu suporte para as últimas funcionalidades do CSS 3 também é maior, e sua velocidade para processar Javascript é superior.

4.6.6 Opera (2000)

O Opera é um navegador criado em 1994 pela empresa estatal de telecomunicações da Noruega e foi a primeira alternativa leve para os usuários.

Segundo Olhar Digital (2009 citado por CORREIA, 2009), recentemente o Opera perdeu seu posto de “navegador alternativo” para o Mozilla Firefox, mas conta ainda com uma fiel comunidade de usuários. Diversos dos recursos mais modernos existentes entre os navegadores vieram do Opera e foram copiados para os demais.

Atualmente, o Opera usa o motor de renderização WebKit, por isso sua performance para executar aplicações HTML5 é praticamente a mesma que a do navegador Chrome e Safari, que também usam este motor.

4.6.7 Safari

Até 2003, a plataforma Mac usava navegadores Netscape. Em 2003, a Apple anunciou seu próprio navegador, o Safari, incluindo-o como navegador padrão a partir do sistema operacional Mac OS X v10.3.

Com uma interface simples, suas funções são básicas: Abas, bloqueador de pop-ups, baixador de arquivos, leitor de notícias RSS e modo privado, o qual evita que terceiros monitorem sua navegação.

O público principal do Safari são usuários com dispositivos da própria apple. Não é muito comum encontrar usuários do Windows ou Linux utilizando Safari em seus computadores.

O Safari, assim como o Chrome e o Opera, utiliza o motor de renderização WebKit.

4.6.8 Google Chrome

O Google se lançou no mercado de navegadores em setembro de 2008 com o Chrome, um navegador projetado do zero e com a promessa de ser mais rápido, seguro e estável que os concorrentes.

Entre seus pontos altos destaca-se a estrutura de processamento do programa, em que cada aba roda um processo paralelo, o que, segundo o Google, pouparia recursos do sistema e preveniria vazamentos de memória e travamentos do computador. Na prática, o Chrome é o navegador que mais consome memória RAM do computador, apesar de ser o mais rápido para executar Javascript.

4.7 FERRAMENTAS DE DIAGNÓSTICO DE PERFORMANCE

Segundo Tomaz (2013), testes de performance ou desempenho são essenciais por avaliarem a capacidade de resposta de um sistema em determinados cenários e configurações e por permitirem planejar melhorias no ambiente para atender a demandas atuais e futuras.

Uma ferramenta diagnóstico tem como objetivo escanear uma página web em busca de problemas ou más práticas relacionadas à performance. Ao encontrar um destes itens, a ferramenta dá algumas sugestões de como contornar o problema. Neste trabalho acadêmico são abordados o YSlow, PageSpeed e WebPageTest, que são as principais ferramentas de diagnóstico existentes.

4.7.1 YSlow

O YSlow foi a primeira ferramenta ferramenta de diagnóstico de performance, criada em 2007 pela Yahoo! como uma extenção para o navegador Mozilla Firefox. O brasileiro Marcel Duran pouco antes de sair do Yahoo! Conseguiu transformar o projeto YSlow em uma iniciativa de código aberto, expandindo para outras plataformas como Google Chrome, Safari e Opera. A ferramenta, que depende do complemento FireBug para funcionar, analisa uma página web de acordo com 23 critérios, avalia com uma nota geral de A a F, um score de 0 a 100, e avalia cada um dos 23 critérios individualmente com uma nota de A a F, facilitando o usuário a identificar em quais pontos a página precisa melhorar.

A figura 9 mostra uma visão geral do YSlow fazendo a verificação do site Yahoo.

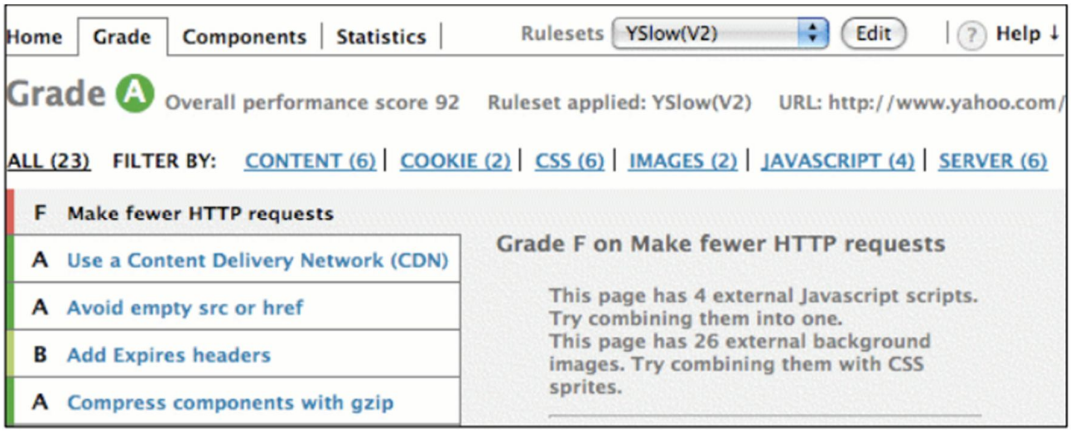


Figura 9 - Resultado de um diagnóstico do YSlow.

Fonte: Elaborado pelo autor.

4.7.2 PageSpeed

O PageSpeed é uma ferramenta desenvolvida pela Google. Foi lançada em 2010, inicialmente implementada como uma extensão do navegador Google Chrome, e depois lançada para o Firefox. Uma vantagem em relação ao seu concorrente YSlow é que possui uma versão online, livre de instalaçãoes, a qual

analisa a performance dos sites independente do navegador que estiver sendo utilizando.

A ferramenta analisa a página web baseada em 31 critérios, e avalia com uma nota geral de 0 a 100, como pode ser visto na figura 10

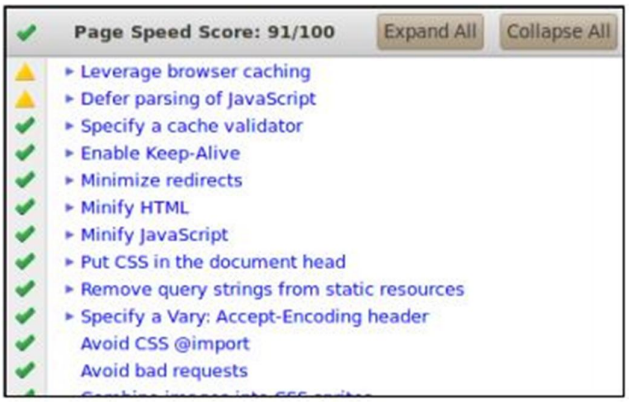


Figura 10 - Resultado de um diagnóstico do PageSpeed.

Fonte: Elaborado pelo autor.

4.7.3 WebPageTest

Diferente das ferramentas YSlow e PageSpeed, o WebPageTest somente tem uma versão de diagnósticos online. Também tem uma opção para analisar sites pelo celular ou tablet. Conforma ilustrado na figura 11, o usuário digita uma URL e depois escolhe o navegador e local no qual deseja que a página seja carregada.

O diferencial desta ferramenta consiste na escolha geográfica de onde será realizada a requisição, permitindo que uma determina página que esteja hospedada no Brasil seja acessada por alguém de outro país.

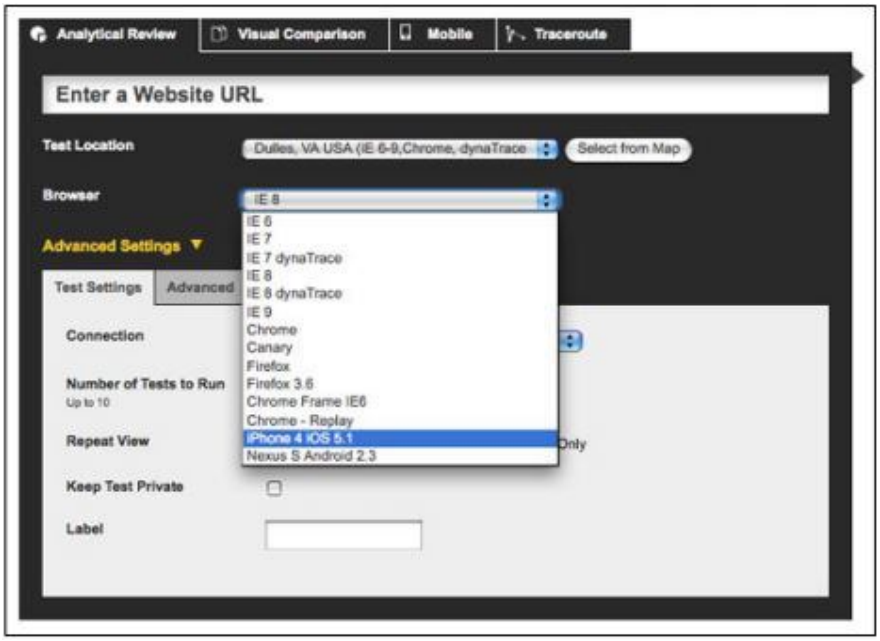


Figura 11 - Interface do WebPageSpeed.

Fonte: Elaborado pelo autor.

4.8 FERRAMENTAS DE MELHORIA DE PERFORMANCE

Uma ferramenta de melhoria de performance tem como objetivo melhorar os pontos críticos de um sistema os quais são revelados através das ferramentas de diagnóstico de performance. Estas ferramentas ajudam o desenvolvedor a realizar tarefas de otimização que seriam inviáveis de se fazer sem o auxílio de um software, pois daria muito trabalho e gastaria muito tempo.

Neste trabalho acadêmico são abordados as seguintes ferramentas: Grunt, UglifyJs e UglifyCSS, SpritePad e Kraken.IO, que são as mais utilizadas para melhoramento de performance.

4.8.1 Grunt

Segundo Santana (2013), O Grunt é uma aplicação de linha de comando que tem como objetivo automatizar tarefas, principalmente tarefas em aplicações JavaScript. O Grunt sozinho não consegue realizar nenhuma ação, ele depende de plugins como o UglifyJs e UglifyCSS para realizar uma determinada tarefa. O site

oficial do Grunt conta com uma enorme base de plugins, cada um específico para uma determinada ação, cabendo ao usuário baixá-los e usá-los.

Por ser um automatizador de tarefas, é possível configurar o Grunt para realizar tarefas a partir de certos eventos, como por exemplo, juntar vários arquivos Javascript em apenas um único arquivo e minificar a cada vez que o desenvolvedor fizer alguma alteração em um desses arquivos.

A figura 12 mostra a interface de linha de comando do Grunt configurado para rodar cinco tarefas usando vários plugins.

```
var config = require('./config.json')
  , getImgs = require('./get_imgs')
  ;

module.exports = function(grunt) {

  // Project configuration.
  grunt.initConfig({ ...
  });

  // Load plugins
  grunt.loadNpmTasks("grunt-contrib-jshint");
  grunt.loadNpmTasks('grunt-contrib-concat');
  grunt.loadNpmTasks('grunt-contrib-uglify');
  grunt.loadNpmTasks('grunt-contrib-cssmin');
  grunt.loadNpmTasks('grunt-couchapp');

  // sync images from config.img_dir
  var description = "Gets images from img_dir, timestamps and writes to attachments/img";
  grunt.registerTask('sync', description, getImgs(grunt, config));

  // Default task(s).
  grunt.registerTask('default', [
    'jshint'
  , 'concat'
  , 'uglify'
  , 'cssmin'
  , 'sync'
  , 'couchapp'
  ]);

};
```

Figura 12 - Interface Grunt.

Fonte: Elaborado pelo autor.

4.8.2 UglifyJS

O UglifyJS é uma ferramenta para o Grunt que tem como objetivo concatenar vários arquivos Javascript em um único arquivo, diminuindo o número de requisições HTTP que o usuário faz para o servidor. Outra importante função do UglifyJS é minificar arquivos Javascript, ou seja, remover qualquer tipo de caracter que não faça diferença para o navegador interpretá-los corretamente, como espaços em

brancos desnecessários e comentários, reduzindo consideravelmente o tamanho do arquivo.

Utilizando o Grunt corretamente é possível configurar o UglifyJS para minificar e concatenar arquivos Javascript a cada vez que o desenvolvedor fizer alguma alteração no arquivo, eliminando qualquer tarefa manual por parte dos desenvolvedores.

A figura 13 mostra um exemplo do Grunt rodando o UglifyJS para minificar o arquivo script.js.

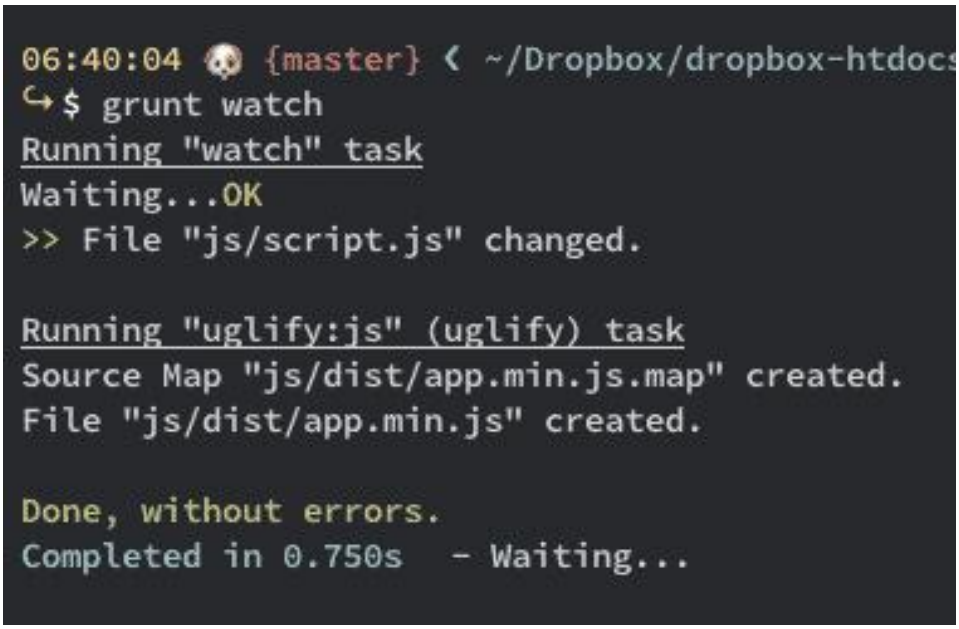


Figura 13 - Exemplo de UglifyJS.

Fonte: Elaborado pelo autor.

4.8.3 UglifyCSS

O UglifyCSS é uma ferramenta bastante similar ao UglifyJS, porém com o objetivo de minificar e concatenar arquivos CSS. Utilizando o UglifyCSS é possível reduzir em até 60% o tamanho original do arquivo, resultando em requisições bem mais rápidas ao servidor.

4.8.4 SpritePad

O SpritePad é uma ferramente online para concatenar várias imagens em uma única imagem. Segundo Rocha (2012), a concatenação de imagens, também conhecida como CSS Sprites, é uma técnica que se baseia em combinar diversas imagens em uma só, em busca de diminuir o número de requisições HTTP para o servidor. Após as imagens terem sido agrupadas em uma única imagem, ela é carregada pelo CSS usando a propriedade background-image, e em seguida é posicionado o ícone desejado pela propriedade background-position, conforme as figuras 14 e 15.

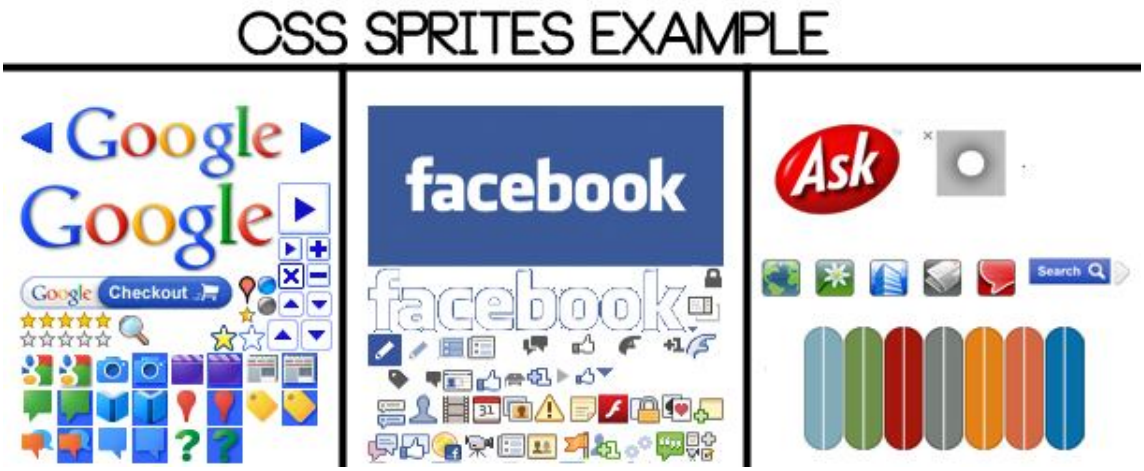


Figura 14 - Agrupamento de imagens em um Sprite.

Fonte: Fonte desconhecida.

```
.icon-foo {
  background-image: url('mySprite.png');
  background-position: -10px -10px;
}

.icon-bar {
  background-image: url('mySprite.png');
  background-position: -5px -5px;
}
```

Figura 15 - Posicionamento de um Sprite por CSS.

Fonte: Elaborado pelo autor.

Para usar o SpritePad, basta arrastar os ícones para a ferramenta, posicioná-las e clicar em Download para gerar o Sprit com todas as imagens juntas. A imagem 16 mostra a interface da ferramenta SpritePad.

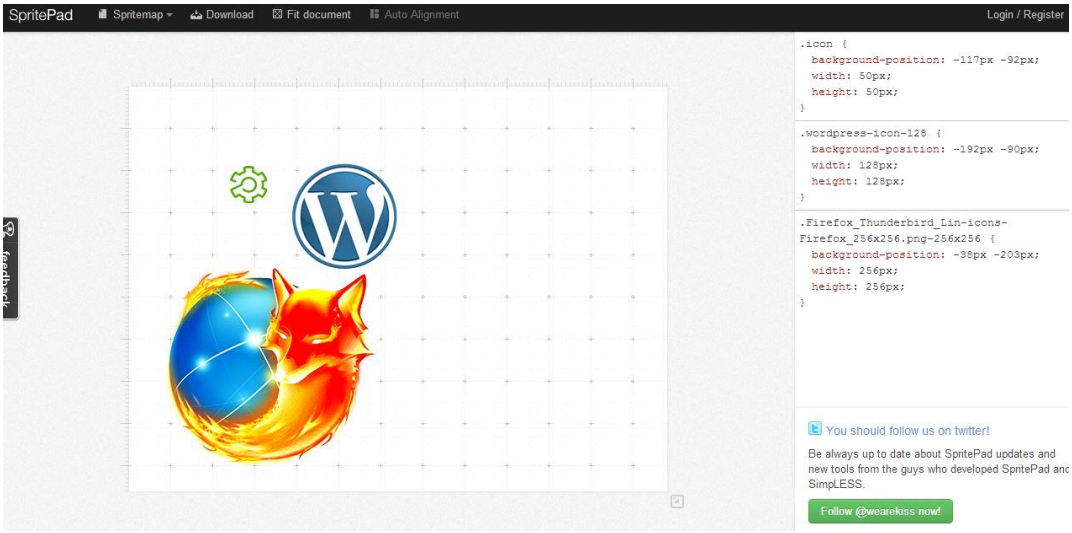


Figura 16 - Interface SpritePad.

Fonte: Elaborado pelo autor.

4.8.5 KRAKEN.IO

O Kraken.io é uma ferramenta de otimização de imagem que consegue comprimi-las sem que a mesma tenha perda de informações, ou seja, o resultado

final não terá diferenças visuais em relação ao original, mesmo que o tamanho do arquivo seja reduzido. Esta técnica, chamada de Lossless, remove metadados não visivéis ao usuário como data de criação da imagem, aplicativo que foi utilizado para criá-la, etc. A ferramenta também tem a opção de otimização de imagem com perda de informações, chamada Lossy. Essa opção consegue imagens mais leves do que a sem perda de informações, porém o resultado final é um pouco inferior à imagem original, sendo pouco perceptível ao olho humano. A figura 17 mostra uma comparação de uma imagem original, da técnica Lossless e da técnica Lossy.

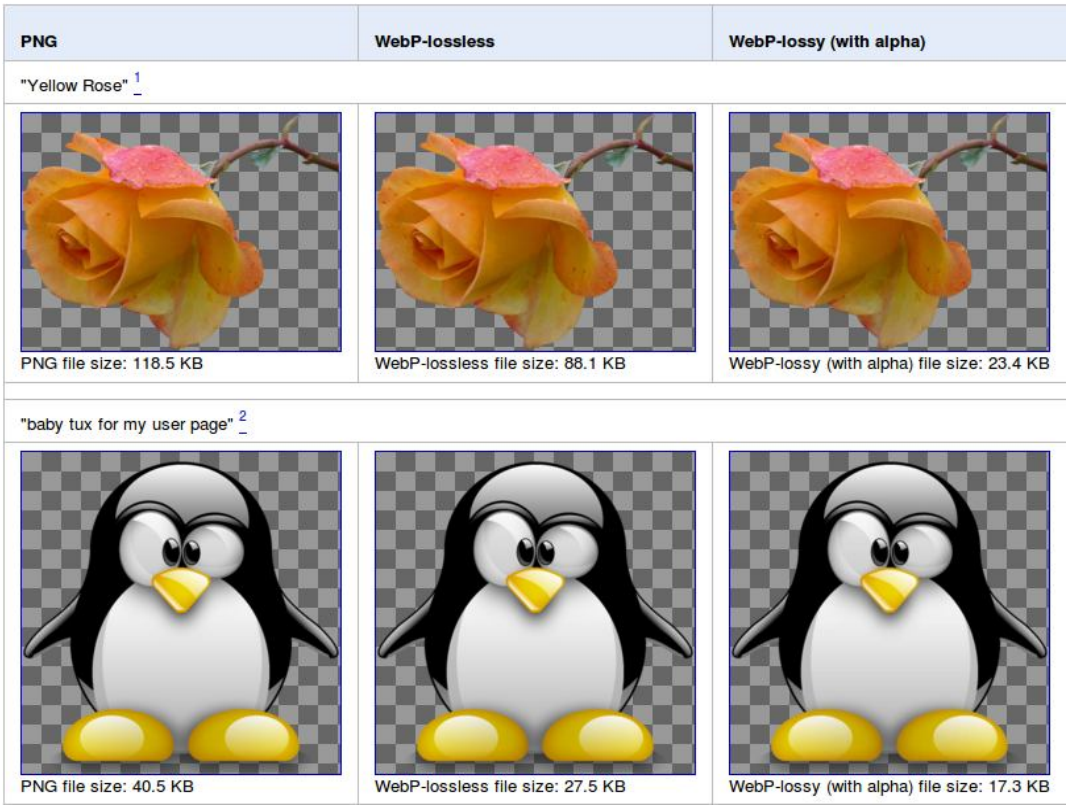


Figura 17 - Comparação Lossless e Lossy.

Fonte: PARFENI (2011).

Além de poder escolher qual o tipo de otimização, é possível redimensionar a imagem, escolhendo qual a altura e a largura com que a imagem otimizada será criada. A figura 18 mostra a interface da ferramenta Kraken.io.

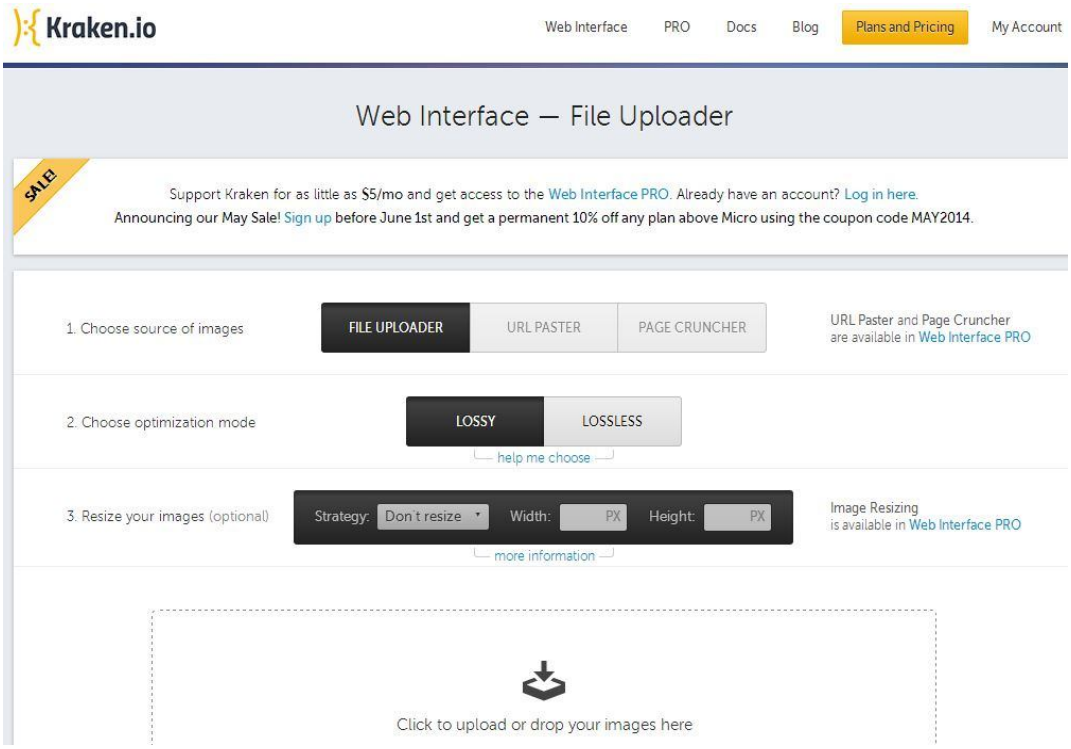


Figura 18 - Interface Kraken.io.

Fonte: Elaborado pelo autor.

5 METODOLOGIA

Este trabalho acadêmico visou demonstrar uma série de etapas que devem ser seguidas para melhorar a velocidade de acesso e carregamento de uma página web.

Para sua realização, foram escolhidas as três páginas mais acessadas no Brasil que utilizam recursos do HTML5, CSS3 e Javascript. Após a escolha das páginas, foram utilizadas ferramentas de diagnóstico de performance para encontrar possíveis causas que estejam prejudicando a velocidade de acesso.

Posteriormente, ao serem levantados todos os pontos que estes sites devem melhorar, foram demonstradas algumas práticas para corrigi-los através de ferramentas de melhoria de performance.

Foi utilizado o serviço de hospedagem gratuita de sites Hostinger para realizar os testes. O serviço oferece um espaço de 2GB de disco, além de hospedar páginas PHP com banco de dados MySQL. Com os testes publicados na internet foi possível medir o tempo de resposta antes e depois dos testes. A conexão com a internet utilizada foi de 30 MB

5.1 CRITÉRIO DEFINIDO PARA A SELEÇÃO DE PÁGINAS

Para definir quais seriam as três páginas avaliadas em diferentes critérios ao longo da monografia, foi utilizada a ferramenta Alexa, que é um site contendo o ranking das páginas mais acessadas de cada país. A figura 19 mostra a interface da ferramenta Alexa listando os sites mais visitados do Brasil.

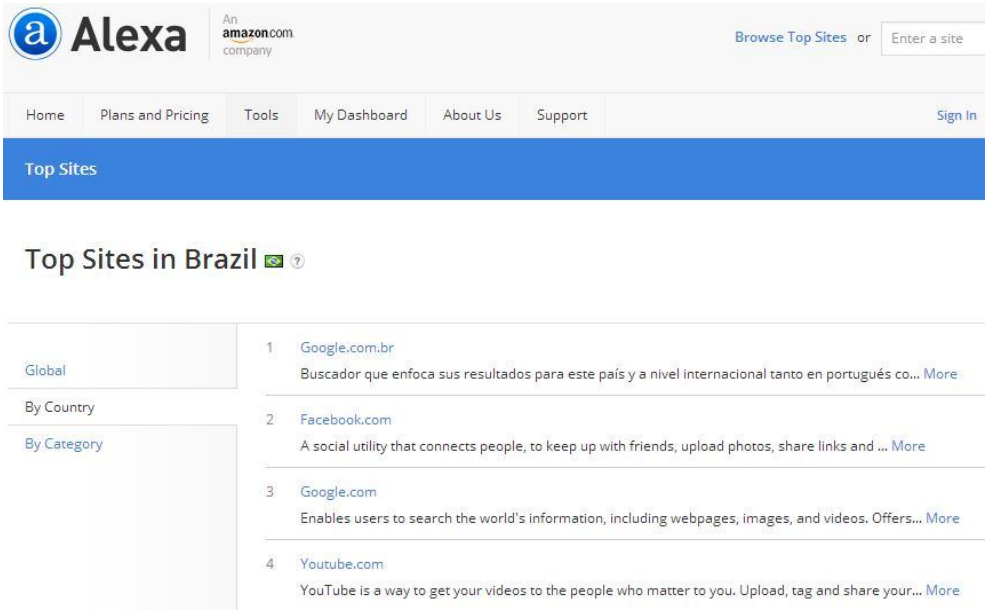


Figura 19 - Interface Alexa.

Fonte: Elaborado pelo autor.

Após a consulta dos sites mais visitados do Brasil, foi realizado um filtro para selecionar apenas os sites que utilizam tecnologias como HTML5, Javascript e CSS3. A identificação destes sites é feita através de uma análise do nível de complexidade da página. Por exemplo, o site mais acessado do Brasil é o Google, porém, é considerado um site simples para ser usado neste trabalho, sem grandes funcionalidades e tantos conteúdos dinâmicos quanto o segundo colocado, o Facebook. Logo o site Facebook se utiliza de recursos HTML5, Javascript e CSS3 bem mais que o site do Google.

A lista dos três sites mais acessados, segundo a ferramenta Alexa, que utilizam essas tecnologias, segundo pesquisa realizada em 10 de maio de 2014, encontra-se na tabela 1.

Rank	Website	URL	Ranking Alexa
1	Facebook	facebook.com	2
2	Youtube	youtube.com	4
3	Mercado Livre	mercadolivre.com.br	9

Tabela 1 - Três sites escolhidos para análise.

Fonte: Elaborado pelo autor.

5.2 TESTES DE DIAGNÓSTICO DE PERFORMANCE

Os três sites apresentados na tabela 1 foram testados com as principais ferramentas de diagnóstico de performance: PageSpeed, YSlow e WebPageTest. A tabela 2 mostra as notas obtidas por cada ferramenta, em uma escala de 0 à 100.

Rank	Website	PageSpeed	Yslow	WebPageTest	Média
1	Facebook	89 (Chrome) 84 (Firefox) 84 (IE)	82 (Chrome) 72 (Firefox)	70	78
2	Youtube	89 (Chrome) 84 (Firefox) 83 (IE)	81 (Chrome) 73 (Firefox)	69	77
3	Mercado Livre	88 (Chrome) 82 (Firefox) 82 (IE)	86 (Chrome) 80 (Firefox)	65	77

Tabela 2 - Resultado dos testes de performance.

Fonte: Elaborado pelo autor.

De acordo com os testes realizados, é possível observar que os três sites escolhidos obtiveram resultados bastante parecidos. Todos apresentaram uma performance acima da média, o que já era esperado, pois os domínios testados têm como foco uma boa performance. Em todos os testes o navegador Chrome obteve uma nota superior em relação aos seus concorrentes, enquanto o Firefox e o Internet Explorer obtiveram notas bem parecidas.

A tabela 3 mostra as principais sugestões de melhoria de performance que as ferramentas sugeriram.

Website	Melhorias
Facebook	- Compactar HTML - Concatenar Javascript - Concatenar Css - Combinar imagens com CSS Sprites
Youtube	- Compactar HTML - Otimizar imagens
Mercado Livre	- Compactar CSS - Compactar Javascript

Tabela 3 - Lista de melhorias sugeridas pelas ferramentas.

Fonte: Elaborado pelo autor.

Com base nas melhorias sugeridas pelas ferramentas de diagnóstico de performance, neste capítulo serão abordado técnicas para reduzir o número de requisições e o tamanho dos arquivos através da concatenação de código (Seção 5.3), concatenação de imagem (Seção 5.4), compressão de código (Seção 5.5) e compressão de imagem (Seção 5.6).

5.3 CONCATENAÇÃO DE CÓDIGO

Muitos arquivos Javascript e CSS podem comprometer a performance de um site. Para cada arquivo, uma requisição HTTP é feita para o servidor e depois armazenado em cache. A tabela 4 mostra a quantidade de arquivos que as páginas testadas contêm.

Website	Arquivos Javascript	Arquivos CSS
Facebook	52	17
Youtube	12	7
Mercado Livre	5	2

Tabela 4 - Número de requisições JS e CSS.

Fonte: Elaborado pelo autor.

JavaScript e CSS são utilizados na maioria dos sites hoje em dia. Engenheiros de Frontend precisam escolher entre colocar seu código JavaScript e

CSS "inline" (incorporado no documento HTML) ou incluí-los em um script e folha de estilo externos. Em geral, utilizar scripts e folhas de estilo externos são melhores para performance. Entretanto, se você seguir a recomendação da engenharia de software e modularizar seu código quebrando-o em diversos arquivos pequenos, você piora a performance porque cada arquivo resulta em uma requisição HTTP adicional (SOUDERS, 2007, p. 33).

Conforme os dados apresentadod na tabela 4, o site Facebook contêm uma grande quantidade de arquivos Javascript e CSS comparado aos outros sites testados. Foram necessárias 69 requisições HTTP para carregar todos os seus arquivos. Uma solução para melhorar a performance do site Facebook seria concatenar os arquivos Javascript e CSS em apenas um único arquivo, diminuindo consideravelmente o número de requisições HTTP para o servidor e, consequentemente, deixando o acesso ao site mais rápido.

5.3.1 Concatenação de Javascript e CSS

Para realizar a concatenação de arquivos Javascript e CSS foi utilizada a ferramenta Grunt e os plugin UglifyJS e UglifyCSS. Para realizar os testes, foi desenvolvida uma página HTML simples contendo 10 arquivos Javascript e 10 arquivos CSS. A figura 20 mostra o Grunt configurado para concatenar os arquivos Javascript.

```
1  module.exports = function( grunt ) {
2
3    grunt.initConfig({
4
5      concatenar : {
6        options : {
7          mangle : false
8        },
9
10       my_target : {
11         files : {
12           'concatenado/arquivos.js' : [ 'js/*.js' ]
13         }
14       }
15     } // concatenar
16
17   });
18
19   // Plugins do Grunt
20   grunt.loadNpmTasks( 'grunt-contrib-uglify' );
21
22
23   // Tarefas que serão executadas
24   grunt.registerTask( 'default', [ 'concatenar' ] );
25
26 };
```

Figura 20 - Grunt configurado para concatenar

Fonte: Elaborado pelo autor

Na linha 5 começa a ser configurado a tarefa “concatenar”. Na linha 11 é definido o caminho através do qual será gerado o arquivo com os scripts concantedos, no caso ‘concatenado/arquivos.js’. Logo em seguida é definido o caminho de onde estão os arquivos a serem concatenados, no caso o diretório js. A expressão “*.js” procura todos os arquivos com a extensão .js no diretório. Na linha 20 é dito ao Grunt para carregar o plugin Uglify, e na linha 24 é executada a tarefa uglify como tarefa principal.

Após configurado o Grunt para concatenar arquivos, é executado no terminal de comandos do Grunt a tarefa para concatenar conforme figura 21.

```
C:\Users\Guilherme\Desktop\tcc\concatenacao_codigo>grunt concatenar
Running "uglify:my_target" (uglify) task
>> 1 file created.

Done, without errors.
```

Figura 21 - Grunt rodando tarefa concatenar

Fonte: Elaborado pelo autor

Ao rodar a tarefa “concatenar”, é gerado o arquivo “arquivos.js” no diretório “concatenado” contendo todos os scripts juntos, conforme mostrado na figura 21.

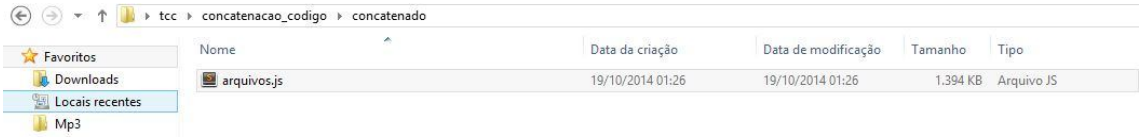


Figura 22 - Arquivo concatenado

Fonte: Elaborado pelo autor

O mesmo processo foi feito com os arquivos CSS, para gerar um único “arquivos.css”.

Após serem gerados os arquivos concatenados, foi alterado na página HTML para carregar somente os arquivos “arquivos.js” e “arquivos.css”. A tabela 5 mostra uma comparação do tempo de resposta e número de requisições ao servidor antes e depois do arquivos serem concatenados.

	Número de requisições	Tempo de resposta
Antes	21	5,59s
Depois	3	5,03s

Tabela 5 - Arquivos concatenados antes e depois

Fonte: Elaborado pelo autor

Com base nos dados apresentados na tabela 5 é possível perceber uma redução significativa, de 85%, no número de requisições HTTP ao servidor de 85%, poupando processamento. Neste caso, não houve ganhos significativos no tempo de resposta, apenas 10%.

5.4 CONCATENAÇÃO DE IMAGEM

De acordo com o teste de performance realizado nos três sites, o único que teve a concatenação de imagens sugerida foi o Facebook. É possível observar na figura 22 que o site Facebook não faz o uso de Sprites para carregar seus ícones,

resultando em mais requisições desnecessárias. Foi utilizado o Console do navegador Chrome para capturar as imagens carregadas do Facebook.

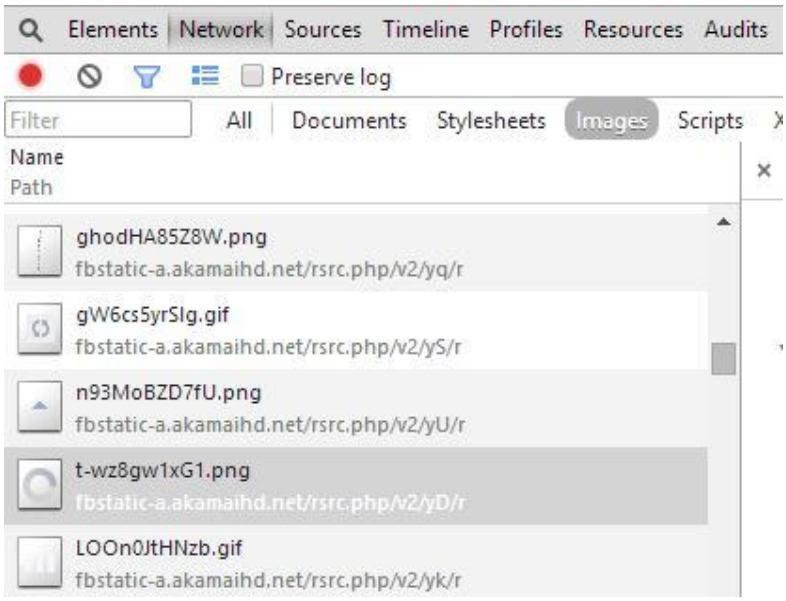


Figura 23 - Ícones separados Facebook.

Fonte: Elaborado pelo autor.

Para resolver este problema, é utilizada a ferramenta SpritePad que junta várias imagens em uma só, sendo feito o posicionamento da mesma através do CSS.

5.4.1 Concatenação de Ícones em uma única imagem

Para realizar a concatenação de imagens foi utilizada a ferramenta SpritePad. Para realizar os testes, foi desenvolvida uma página HTML simples com 20 ícones separados. Os ícones foram baixados pelo site FindIcons, um site de pacote de ícones gratuitos.

Após a realização do download dos ícones foi feito o upload dos mesmos na ferramenta online SpritePad. Então, eles foram colocados lado a lado, conforme mostra a figura 23.

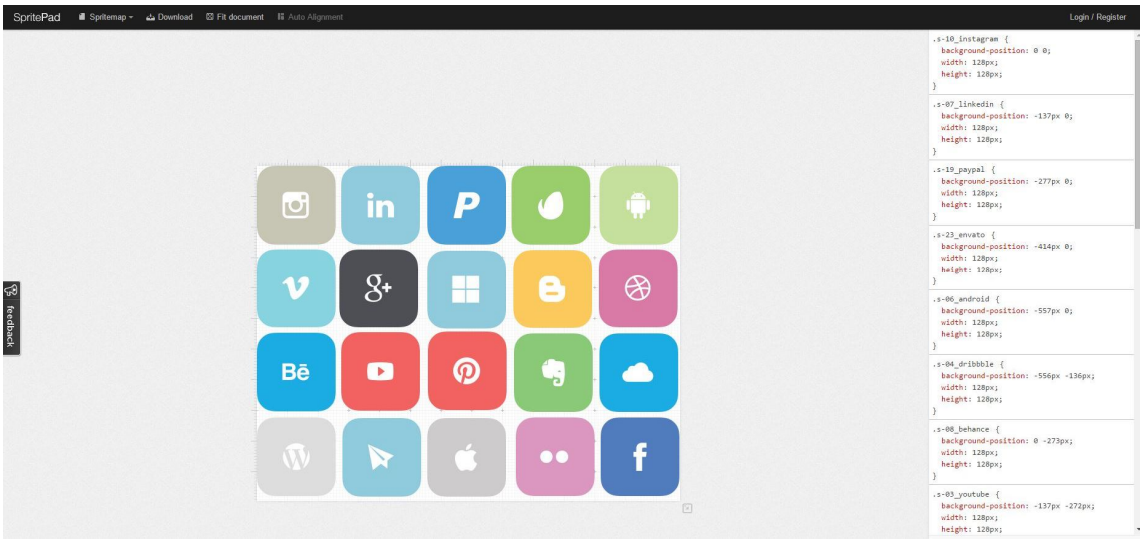


Figura 24 - Imagens carregadas no Spritepad

Fonte: Elaborado pelo autor

Feito isto, após clicar em Download, é gerado uma única imagem no formato PNG contendo todos os ícones juntos, conforme figura 24.

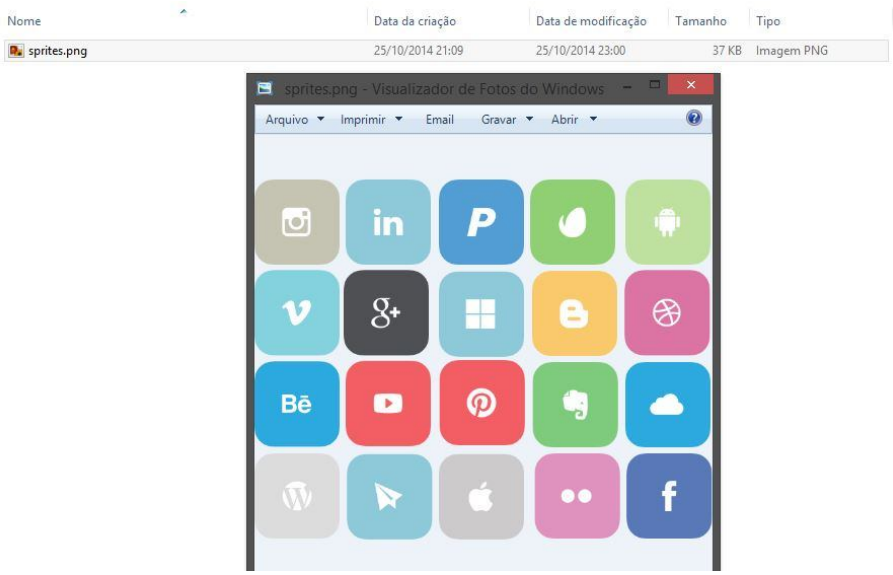


Figura 25 - Arquivo com a sprite

Fonte: Elaborado pelo autor

Depois de ser gerado a sprite foi feita a comparação utilizando os 20 ícones separados e apenas a Sprite com ícones juntos. A tabela 6 mostra os resultados obtidos.

	Número de requisições	Tempo de resposta
Antes	21	2,52s
Depois	2	2,15s

Tabela 6 - Sprites antes e depois

Fonte: Elaborado pelo autor

Houve uma redução de 90% no número de requisições. Isso ocorreu porque não foi mais preciso fazer uma requisição para cada um dos 20 ícones, e sim apenas uma requisição ao sprite.png, que contém todos os ícones unidos. Neste caso, não houve ganhos significativos no tempo de resposta, apenas 14%.

5.5 COMPRESSÃO DE CÓDIGO

Os desenvolvedores costumam escrever comentários e indentar o código para mantê-lo organizado e legível. Essas práticas não fazem diferença no momento em que o código é executado, portanto, trafegar esse volume de bytes é desnecessário. Uma boa prática para reduzir o tamanho dos arquivos é comprimi-los. Essa técnica consiste em remover comentários que os desenvolvedores escreveram, espaços em branco desnecessários, e substituir o nome das variáveis por nomes menores sem alterar a funcionalidade do código.

5.5.1 Compressão de Javascript e CSS

Foi utilizada a ferramenta Grunt junto com os plugins UglifyJS e UglifyCSS para comprimir arquivos Javascript e CSS. Para realizar os testes foi desenvolvida uma página simples HTML e nela foi adicionado um arquivo Javascript com 5,29 MB contendo alguns frameworks, como Jquery e AngularJS, e um arquivo CSS de 5,68

MB contendo alguns frameworks, como Bootstrap e Foundation. A figura 25 mostra o Grunt configurado para comprimir os arquivos Javascript.

```
1  module.exports = function( grunt ) {
2
3    grunt.initConfig({
4
5      uglify : {
6        options : {
7          mangle : false
8        },
9
10       my_target : {
11         files : {
12           'comprimido/arquivo.js' : [ 'js/arquivo.js' ]
13         }
14       }
15     } // uglify
16
17   });
18
19   // Plugins do Grunt
20   grunt.loadNpmTasks( 'grunt-contrib-uglify' );
21
22
23   // Tarefas que serão executadas
24   grunt.registerTask( 'comprimir', [ 'uglify' ] );
25
26 };
```

Figura 26 - Grunt configurado para comprimir

Fonte: Elaborado pelo autor

Na linha 12 é definido que o Grunt faça a compressão do arquivo.js do diretório e gere um arquivo arquivo.js dentro do diretório comprimido.

Após configurado o Grunt para concatenar arquivos é executado no terminal de comandos do Grunt a tarefa para comprimir, conforme figura 27.

```
C:\Users\Guilherme\Desktop\tcc\compressao_codigo>grunt comprimir
Running "uglify:my_target" (uglify) task
>> 1 file created.

Done, without errors.
```

Figura 27 - Grunt rodando tarefa comprimir

Fonte: Elaborado pelo autor

Ao rodar a tarefa comprimir é gerado o arquivo arquivos.js no diretório comprimido, contendo o arquivo comprimido, conforme mostrado na 26.

Figura 28 - Arquivo comprimido

Fonte: Elaborado pelo autor

O mesmo procedimento foi feito para comprimir o arquivo CSS. Após serem gerados os arquivos comprimidos foi alterado na página HTML para carregar estes arquivos ao invés dos originais. A tabela 7 mostra uma comparação do tamanho em, MB, das requisições e do tempo de resposta antes e depois dos arquivos serem comprimidos.

	Tamanho total da requisição	Tempo de resposta
Antes	10,8 MB	8,96s
Depois	3,1 MB	2,84s

Tabela 7 - Arquivo comprimido antes e depois

Fonte: Elaborado pelo autor

Com base nos dados apresentados na tabela 7, foi reduzido consideravelmente o tamanho total das requisições, levando em consideração que na compressão dos arquivos houve uma redução de mais de 71% do tamanho em MB, e como consequência, o tempo de resposta diminuiu em 68%.

5.6 COMPRESSÃO DE IMAGEM

Um dos tipos de arquivo que mais consomem carregamento em uma página web são as imagens. É preciso tomar cuidado ao escolher o formato e a resolução da imagem evitando formatos muito pesados, como o formato .BMP, que armazena fotos e gráficos nos pixels e torna o arquivo muito grande, bem como resoluções muito altas para imagens que serão pequenas na página, desperdiçando bytes.

De acordo com os testes de performance realizados nos sites, o único que teve como sugestão para comprimir as imagens foi o Youtube.

5.6.1 Compressão Lossless

Uma das formas de otimização de imagens é a compressão Lossless, sem perda de informação. Foi utilizada a ferramenta Kraken.io para fazer a compressão de imagens sem perda de qualidade. Esta ferramenta foi aplicada em uma imagem PNG de dimensões 1920x1080 pixels e 203 KB de tamanho, e depois feita a comparação para ver qual a porcentagem de economia de bytes.

Na figura 27 é mostrada a interface do Kraken.io. A ferramenta com a opção Lossless selecionada conseguiu reduzir uma quantidade de 59,33 KB, diminuindo 29,14% do tamanho original, passando para 144,27 KB.

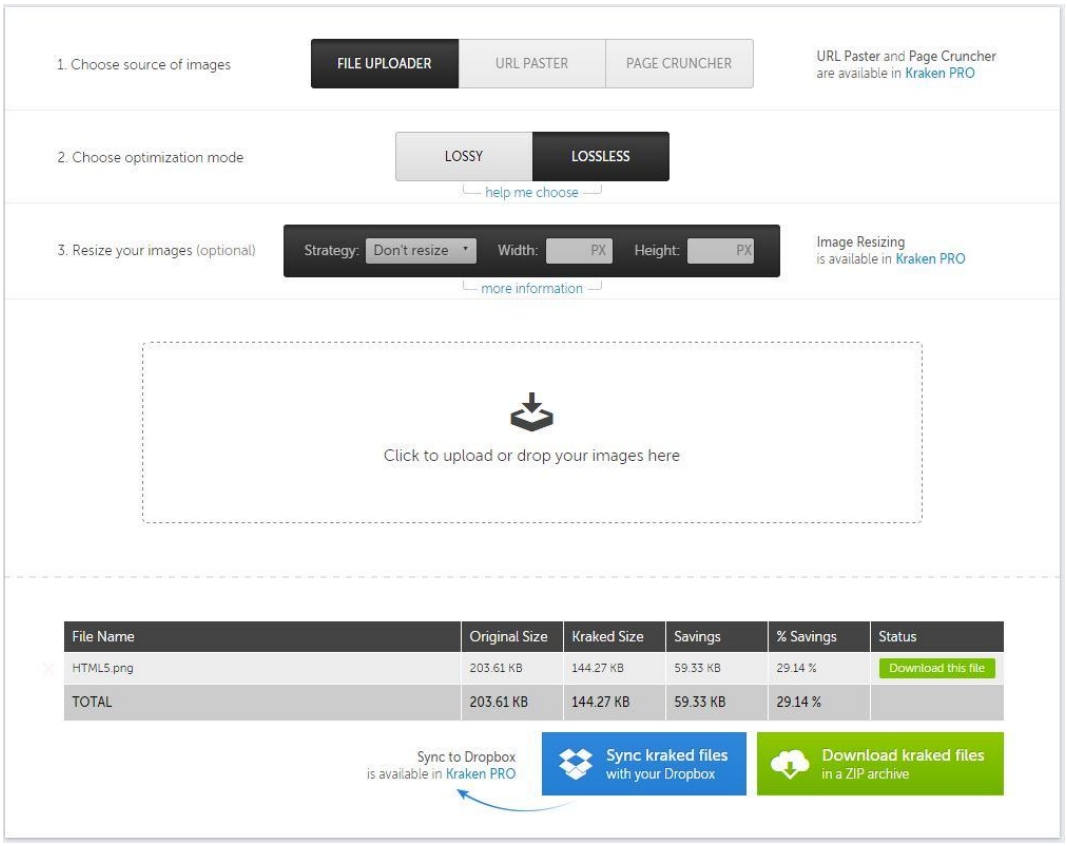


Figura 29 - Kraken.io utilizando lossless

Fonte: Elaborado pelo autor

A figura 28 mostra uma comparação visual antes e depois de aplicar a compressão. A imagem da esquerda é a original, enquanto que na da direita foi

aplicada a compressão Lossless. Analisando a comparação não é possível detectar perdas de detalhes nas imagens.

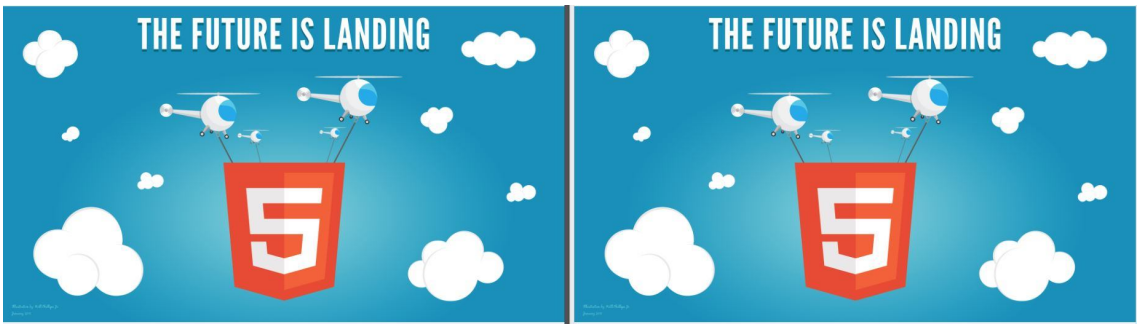


Figura 30 - Comparação lossless

Fonte: Elaborado pelo autor

5.6.2 Compressão Lossy

Outra forma de compressão de imagem é a chamada Lossy. Esta técnica resulta em imagens mais leves do que a técnica Lossless, porém a qualidade da imagem é reduzida, e mesmo que a diferença seja mínima, pode ser visível ao olho humano. Para fazer esta compressão também foi utilizada a ferramenta Kraken.io, aplicada na mesma imagem da compressão Lossless. Em seguida, foi feita a comparação para ver qual a porcentagem de economia de bytes.

Na figura 29 é mostrada a interface do Kraken.io. A ferramenta com a opção Lossy selecionada conseguiu diminuir uma quantidade de 125,10 KB, reduzindo 61,44% do tamanho original, passando para 78,51 KB.

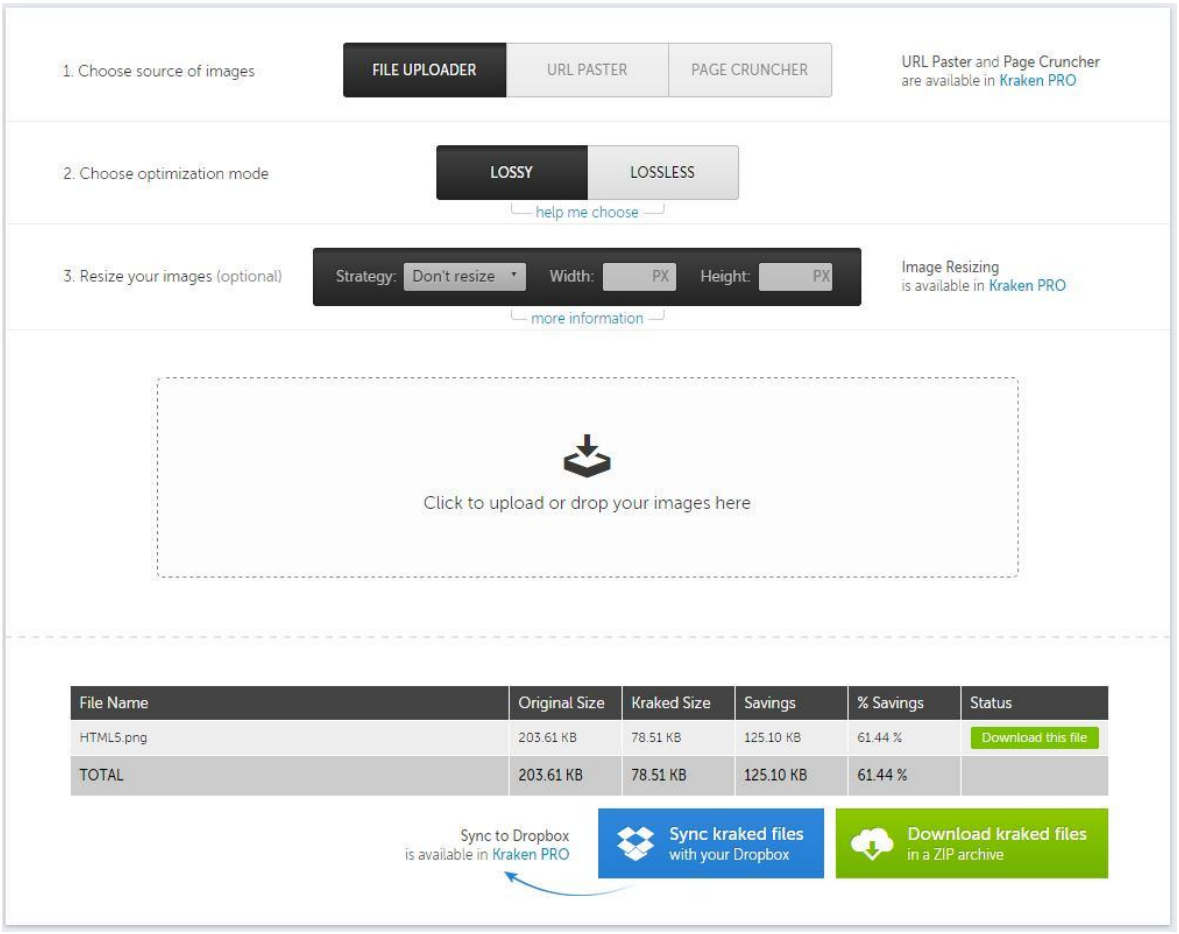


Figura 31 - Kraken.io utilizando lossy

Fonte: Elaborado pelo autor

A figura 30 mostra uma comparação visual antes e depois de aplicar a compressão. A imagem da esquerda é a original, enquanto que na da direita foi aplicada a compressão Lossy. Analisando a comparação, não é possível detectar perdas de detalhes nas imagens, mesmo que com a compressão ocorram perdas de informações. Se o foco do site da exibição de imagens, a compressão se torna a melhor opção para que os sites consigam diminuir o tamanho de MB trafegado.

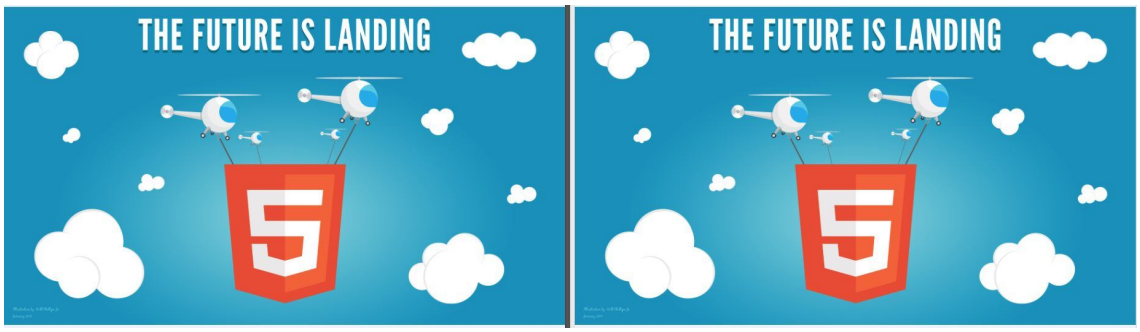


Figura 32 - Comparação lossy

Fonte: Elaborado pelo autor

6 CONSIDERAÇÕES FINAIS

Este trabalho teve como proposta a apresentação de técnicas para melhorar a performance de aplicações do lado do cliente, otimizando ao máximo as tecnologias HTML5, Javascript e CSS3. Também teve como objetivo demonstrar a importância destas três tecnologias no cenário atual, que exige páginas dinâmicas, com boa usabilidade e que possam ser acessadas em qualquer dispositivo.

Os testes foram feitos em sites com milhões de acessos para demonstrar que até grandes empresas como Google e Facebook apresentam algumas falhas de performance, as quais poderiam ser resolvidas aplicando as técnicas apresentados neste projeto.

Após identificar problemas que afetam a performance de páginas WEB e aplicar técnicas para corrigir estes problemas, foi provado que é possível otimizar a performance das páginas, diminuindo significativamente o tamanho em KB de arquivos e melhorando o tempo de acesso. Com os resultados obtidos, é possível chegar à conclusão de que nem sempre o problema de lentidão no carregamento de páginas ocorre somente devido à internet lenta, mas também por conta de arquivos Javascript e CSS mal escritos e excessivamente grandes.

Este projeto demonstrou apenas algumas ferramentas e técnicas que podem melhorar a performance de páginas. Portanto, em futuros projetos, essas técnicas podem ser aplicadas utilizando novas ferramentas com o objetivo de comparar, selecionar, ou ainda apresentar técnicas diferentes de melhoria de performance.

7 CRONOGRAMA

ETAPAS	MÊS DE EXECUÇÃO											
	1	2	3	4	5	6	7	8	9	10	11	12
Levantamento Bibliográfico: Fichamento de Livros, Periódicos, Vídeos, CD Rom, Pesquisas Internet etc.												
Pesquisa para elaboração teórica.												
Preparação e formatação da fundamentação teórica.												
Busca e teste em ferramentas para realização das propostas apresentadas.												
Redação preliminar do projeto de pesquisa.												
Considerações finais do projeto de pesquisa, revisão e entrega.												

Figura 33 - Cronograma

Fonte: Elaborado pelo autor

REFERÊNCIAS

SIMONI, R. HTML5 e SEO: mitos e verdades. **Tableless,** 2013. Disponível em: <http://tableless.com.br/html-5-e-seo-mitos-e-verdades/>. Acesso em: 10 maio 2014.

PEREIRA, A. P. O que é CSS?. **Tecmundo,** 2009. Disponível em: <http://www.tecmundo.com.br/programacao/2705-o-que-e-css-.htm>. Acesso em: 10 maio 2014.

PRADO, R. História do Javascript. **Romilsonprado,** 2014. Disponível em: <http://www.romilsonprado.com.br/historia-do-javascript/>. Acesso em: 10 maio 2014.

JURAN, J. M.; GODFREY, A. B. **Juran's Quality Handbook**. 5th ed. New York: McGraw-Hill, 1999. 1872 p.

BOZZA, C. O que é HTML5?. **Tecmundo,** 2013. Disponível em: <http://www.techtudo.com.br/artigos/noticia/2011/12/o-que-e-html5.html>. Acesso em: 12 maio 2014.

EIS, D.; Ferreira, E. **HTML5 e CSS3 com farinha e pimenta**. Novatec Editora, 2012 p. 278p

RODRIGUES, S. O que estudar para aprender CSS3? **LadoDesign,** 2012. Disponível em: <http://www.ladodesign.com.br/desenvolvimento-frontend/o-que-estudar-para-aprender-css3/ >. Acesso em: 12 maio 2014.

SILVA, M. S. **CSS3 Desenvolva aplicações Web profissionais com uso dos poderosos recursos de estilização das CSS3.** Novatec, 2012. 496p

SOULDERS, S. **High Performance Web Sites: Essential Knowledge for Front-End Engineers.** O'Reilly, 2007. 170p

ALVARES, M. A. Introdução a CSS3. **Criarweb,** 2008. Disponível em: <http://www.criarweb.com/artigos/introducao-a-css3.html>. Acesso em: 16 maio 2014.

SILVA, M. S. **Javascript Guia do Programador.** Novatec Editora, 2010. 608p.

NIEDERAUER, J. **WEB Interativa com AJAX E PHP**. Novatec Editora, 2007. 288p.

CORREIA, D. B. **Estudo de navegadores de acordo com os padrões W3C.** 2009. 61f. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) – Faculdade de Tecnologia da Zona Leste – Centro de Tecnologia da Zona Leste, São Paulo, 2009.

TOMAZ, R. Teste de desempenho – Conceitos importantes. **Crowstest,** 2013. Disponível em: <http://crowdtest.me/testes-desempenho-conceitos/>. Acesso em: 21 maio 2014.

SANTANA, V. Grunt: Você deveria estar usando. **Tableless**, 2013. Disponível em: <http://tableless.com.br/grunt-voce-deveria-estar-usando/>. Acesso em: 17 maio 2014.

ROCHA, Z. CSS Sprites. **Tableless**, 2012. Disponível em: <http://tableless.com.br/css-sprites/>. Acesso em: 18 maio 2014.

PRESSMAN, R. S. **Software Engineering: A practitioner's approach.** 6th ed. McGraw-Hill, 2005.

CARRÉRA, J. Entendendo os testes de performance. **Bytesdontbite**, 2012. Disponível em: <http://bytesdontbite.com/2012/04/02/entendendo-os-testes-de-performance/>. Acesso em: 18 maio 2014.

LAWSON, B. Fazendo o design de um blog com HTML5. **Imasters**, 2011. Disponível em: <http://imasters.com.br/artigo/20734/web-standards/fazendo-o-design-de-um-blog-com-html5/ >. Acesso em: 25 maio 2014.

PRÁTICAS DE MELHORIA DE PERFORMANCE EM APLICAÇÕES QUE UTILIZAM AS TECNOLOGIAS HTML5, JAVASCRIPT E CSS3

Guilherme João Lacerda[1], Alex Setolin Beirigo, André Castro, Marcio Cardim

[1]Centro de Ciências Exatas e Sociais Aplicadas – Universidade Sagrado Coração
Bauru – SP – Brasil

guilherme.lcd@gmail.com, setolin@gmail.com, andrecastro@usc.br, mcardim@usc.br

Abstract. *Websites and system WEB are envolving and becoming increasingly common, being accessed from several devices, from computers to cell phones. Many applications that were originally developed as applications desktop are being ported to WEB, requiring the programmer knowledge technologies as Javascript, HTML5 and CSS3. Based on this context, this work proposes the analysis of sites to detect problems that affect performance, and implement practices to correct them and improve access time.*

Resumo. *As páginas e sistemas WEB estão evoluindo e tornando-se cada vez mais comuns, sendo acessadas de vários dispositivos, desde computadores pessoais até celulares. Muitos aplicativos que foram originalmente desenvolvidos como aplicações desktop estão sendo portados para WEB, exigindo do programador um conhecimento em tecnologias como Javascript, HTML5 e CSS3. Com base neste contexto, este trabalho propõe a análise de sites para detectar problemas que afetem a performance, bem como aplicar práticas para corrigi-los e melhorar o tempo de acesso.*

1. Introdução

O HTML nasceu no ano de 1991, no laboratório de física de partículas *European Council for Nuclear Research* na Suíça. Foi criado pelo inglês Tim Berners-Lee, de 44 anos com o objetivo de interligar computadores do laboratório e outras instituições de pesquisas, para exibir documentos científicos de forma simples e de fácil acesso.

O HTML5 foi criado exatamente para dar significado semântico às páginas web, além de padronizar e facilitar o entendimento das sessões de um site pelos navegadores. (SIMONI, 2013).

A medida que o HTML foi se popularizando e evoluindo, começou a surgir a necessidade de se definir o layout dos documentos. No ano de 1994 foi criada a linguagem CSS por Hakon Wium Lie, justamente para resolver esse problema.

Em 1995 a proposta do CSS foi apresentada para a *W3C*, que estava acabando de nascer. A proposta foi aceita e foi criada uma equipe liderada por Hakon para continuar o desenvolvimento do CSS.

Para tornar a Internet mais dinâmica, Brendan Eirch da Netspace criou a linguagem de programação Javascript, no ano de 1995.

2. Objetivos

2.1 Objetivo Geral

Analisar o desempenho em aplicações WEB que utilizam as tecnologias HTML5, Javascript e CSS3, para encontrar os principais problemas que possam prejudicar a performance, bem como apresentar práticas para corrigir estes problemas e melhorar a velocidade de acesso.

2.2 Objetivos Específicos

- Identificar os três sites mais acessados no Brasil que utilizam recursos de HTML5, Javascript e CSS3.
- Através de ferramentas de diagnóstico de performance, verificar os principais problemas que estejam prejudicando a performance de cada um dos três sites escolhidos, utilizando diferentes navegadores.
- Demonstrar práticas para corrigir os problemas listados pelas ferramentas de diagnóstico de performance.

3. Justificativa

Com a popularização do HTML5, surgiram novos desafios para os programadores *Front-End* como a necessidade de aprender a linguagem de programação Javascript e a linguagem de folhas de estilo CSS a fim de criar páginas mais atraentes e dinâmicas.

As linguagens Javascript e CSS possuem alto nível de complexidade. Existem várias formas de utilizá-las para se alcançar o resultado esperado, mas nem sempre são empregadas da maneira correta, comprometendo a performance.

Essas aplicações devem ser as mais otimizadas possíveis, devido ao processamento limitado e conexão lenta de dispositivos móveis.

Este trabalho acadêmico visa auxiliar os desenvolvedores *Front-End* a conseguirem melhorar a performance de suas aplicações WEB através de inúmeros testes e práticas.

4. Revisão da Literatura

4.1. Testes de performance

Testes de performance são essenciais por avaliarem a capacidade de resposta de um sistema em determinados cenários e configurações e por permitirem planejar melhorias no ambiente para atender a demandas atuais e futuras. (Tomaz, 2013).

Os resultados dos testes são usados para determinar um padrão de comportamento e capacidade da aplicação que está sendo testada. Esses limites e

padrões são chamados de baseline e servem de referência para testes de carga e stress ou outros que estejam sendo planejados/executados no ambiente.

Logo, é fundamental que, desde o início do ciclo de vida da aplicação, a performance seja priorizada, para que todo um ambiente de testes seja preparado de maneira adequada para simular o ambiente de produção e desse modo auxiliar o desenvolvimento a fim de atingir o nível de qualidade desejado.

4.2 HTML5

Um dos principais objetivos do HTML5 é facilitar a manipulação do elemento possibilitando o desenvolvedor a modificar as características dos objetos de forma não intrusiva e de maneira que seja transparente para o usuário final. (EIS; FERREIRA, 2012, p. 29).

O HTML5 possui um conjunto de novas funcionalidades encapsuladas em APIs que podem ser acessadas via Javascript. Está sendo desenvolvido com o objetivo de trabalhar de forma independente, ou seja, não depende de outros plugins como por exemplo o Flash para exibir animações.

4.3 CSS3

É a mais nova versão das Cascading Style Sheets (CSS), que são folhas de estilo responsáveis em apresentar o visual dos sites. (RODRIGUES, 2012).

Segundo Alvarez (2008), a novidade mais importante que o CSS 3 traz para os desenvolvedores consiste na incorporação de novos mecanismos para manter um maior controle sobre o estilo com o qual se mostram os elementos das páginas, sem ter que recorrer a truques ou hacks, que muitas vezes complicavam o código.

4.4 JAVASCRIPT

Segundo Silva (2010, p. 23), “a linguagem de scripts Javascript foi criado pela Netscape em parceria com a Sun Microsystems, com a finalidade de fornecer um meio de adicionar interatividade a uma página web”.

Silva (2010, p. 29) ressalta que “linguagem de script é uma linguagem de programação usada para manipular, personalizar e automatizar as funcionalidades de um sistema existente”. Essas funcionalidades já se encontram disponíveis por meio de uma interface do usuário e a linguagem de script provê um mecanismo para acessá-las.

4.5 FERRAMENTAS DE DIAGNÓSTICO DE PERFORMANCE

Segundo Tomaz (2013), testes de performance ou desempenho são essenciais por avaliarem a capacidade de resposta de um sistema em determinados cenários e configurações e por permitirem planejar melhorias no ambiente para atender a demandas atuais e futuras.

Uma ferramenta diagnóstico tem como objetivo escanear uma página web em busca de problemas ou más práticas relacionadas à performance. Ao encontrar um destes itens, a ferramenta dá algumas sugestões de como contornar o problema. Neste trabalho acadêmico são abordados o YSlow, PageSpeed e WebPageTest, que são as principais ferramentas de diagnóstico existentes.

4.6 FERRAMENTAS DE MELHORIA DE PERFORMANCE

Uma ferramenta de melhoria de performance tem como objetivo melhorar os pontos críticos de um sistema os quais são revelados através das ferramentas de diagnóstico de performance. Estas ferramentas ajudam o desenvolvedor a realizar tarefas de otimização que seriam inviáveis de se fazer sem o auxílio de um software, pois daria muito trabalho e gastaria muito tempo.

5. Metodologia

Este trabalho acadêmico visou demonstrar uma série de etapas que devem ser seguidas para melhorar a velocidade de acesso e sobre carregamento de uma página web.

Para sua realização, foram escolhidas as três páginas com maior número de acessos que utilizam recursos do HTML5, CSS3 e Javascript. Após a escolha das páginas, foram utilizadas ferramentas de diagnóstico de performance para encontrar possíveis que estão prejudicando a velocidade de acesso.

Posteriormente, ao serem levantados todos os pontos que estes sites devem melhorar, foram demonstradas algumas práticas para corrigi-los através de ferramentas de melhoria de performance.

Foi utilizado o serviço de hospedagem gratuita de sites Hostinger para realizar os testes. O serviço oferece um espaço de 2GB de disco, além de hospedar páginas PHP com banco de dados MySQL. Com os testes publicados na internet, foi possível medir o tempo de resposta antes e depois dos testes. A conexão com a internet utilizada foi de 30 MB.

Para definir quais seriam as três páginas avaliadas em diferentes critérios ao longo da monografia, foi utilizada a ferramenta Alexa, que é um site contendo o ranking das páginas mais acessadas de cada país. Os três sites escolhidos foram: Facebook, Youtube e Mercado Livre.

Os três sites apresentados foram testados com as principais ferramentas de diagnóstico de performance, que são: PageSpeed, YSlow e WebPageTest. A tabela a seguir mostra as principais sugestões de melhoria de performance que as ferramentas sugeriram.

Website	Melhorias
Facebook	- Compactar HTML - Concatenar Javascript - Concatenar Css - Combinar imagens com CSS Sprites
Youtube	- Compactar HTML - Otimizar imagens
Mercado Livre	- Compactar CSS - Compactar Javascript

Com base nas melhorias sugeridas pelas ferramentas de diagnóstico de performance, será abordado técnicas para reduzir o número de requisições e o tamanho dos arquivos através da concatenação de código, concatenação de imagem, compressão de código e compressão de imagem.

6. Resultados

6.1. CONCATENAÇÃO DE CÓDIGO

Para realizar a concatenação de arquivos Javascript e CSS foi utilizada a ferramenta Grunt e os plugin UglifyJS e UglifyCSS. Para realizar os testes, foi desenvolvida uma página HTML simples contendo 10 arquivos Javascript e 10 arquivos CSS.

Após serem gerados os arquivos concatenados, foi alterado na página HTML para carregar somente os arquivos “arquivos.js” e “arquivos.css”. A tabela xx mostra uma comparação do tempo de resposta e número de requisições ao servidor antes e depois do arquivos serem concatenados.

	Número de requisições	Tempo de resposta
Antes	21	5,59s
Depois	3	5,03s

Com base nos dados apresentados na tabela xx é possível perceber uma redução significativa no número de requisições HTTP ao servidor de 85%, poupando processamento. Neste caso não houve ganhos significativos no tempo de resposta, apenas 10%.

6.2. CONCATENAÇÃO DE IMAGEM

Para realizar a concatenação de imagens foi utilizada a ferramenta SpritePad. Para realizar os testes, foi desenvolvida uma página HTML simples com 20 ícones separados. Os ícones foram baixados pelo site FindIcons, um site de pacote de ícones gratuitos.

Após feito o download dos ícones, foi feito o upload deles na ferramenta online SpritePad, e colocados lado a lado.

Após ser gerado a sprite, foi feito a comparação utilizando os 20 ícones separados e utilizando apenas a sprite com os ícones juntos. A tabela a seguir mostra os resultados obtidos.

	Número de requisições	Tempo de resposta
Antes	21	2,52s
Depois	2	2,15s

Houve uma redução de 90% no número de requisições. Isto ocorreu porque não foi mais preciso fazer uma requisição para cada um dos 20 ícones, e sim apenas uma requisição ao sprite.png, que contém todos os ícones juntos. Neste caso não houve ganhos significativos no tempo de resposta, apenas 14%.

6.3. COMPRESSÃO DE CÓDIGO

Foi utilizada a ferramenta Grunt junto com os plugins UglifyJS e UglifyCSS para comprimir arquivos Javascript e CSS. Para realizar os testes, foi desenvolvido uma página simples HTML e nela foram adicionados um arquivo Javascript com 5,29 MB, contendo alguns frameworks como Jquery e AngularJS, e um arquivo CSS de 5,68 MB contendo alguns frameworks como Bootstrap e Foundation. A figura 25 mostra o Grunt configurado para comprimir os arquivos Javascript.

Após serem gerados os arquivos comprimidos, foi alterado na página HTML para carregar estes arquivos em vez do originais. A tabela a seguir mostra uma comparação do tamanho em MB das requisições e do tempo de resposta antes e depois do arquivos serem comprimidos.

	Tamanho total da requisição	Tempo de resposta
Antes	10,8 MB	8,96s
Depois	3,1 MB	2,84s

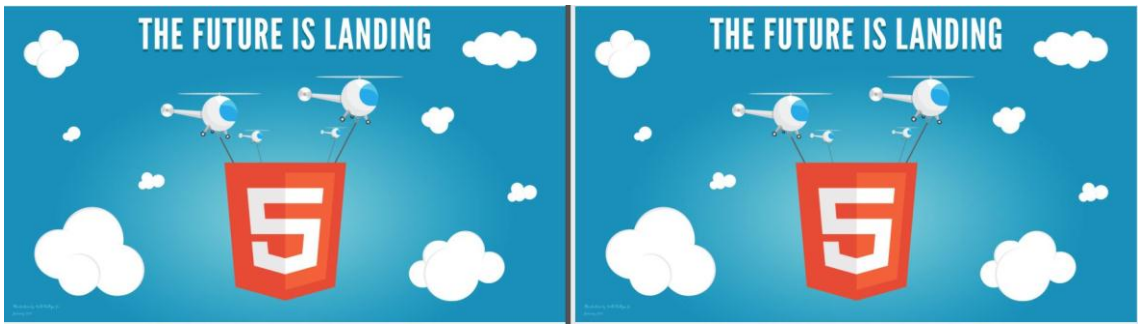
Com base nos dados apresentados na tabela xx, foi reduzido consideravelmente o tamanho total das requisições, levando em consideração que a compressão dos arquivos

houve uma redução de mais de 71% do tamanho em MB, e como consequência o tempo de resposta diminuiu em 68%.

6.4. COMPRESSÃO LOSSLESS

Uma das formas de otimização de imagens é a compressão Lossless, ou seja, quando não há perda de informação. Foi utilizada a ferramenta Kraken.io para fazer a compressão de imagens sem perda de qualidade. A ferramenta foi aplicada em uma imagem PNG de dimensões 1920x1080 pixels e 203 KB de tamanho, e depois feita a comparação para ver qual a porcentagem de economia de bytes.

A figura a seguir mostra uma comparação visual antes e depois de aplicar a compressão. A imagem da esquerda é a original, enquanto a da direita foi aplicada a compressão Lossless. Analisando a comparação, não é possível detectar perdas de detalhes nas imagens.

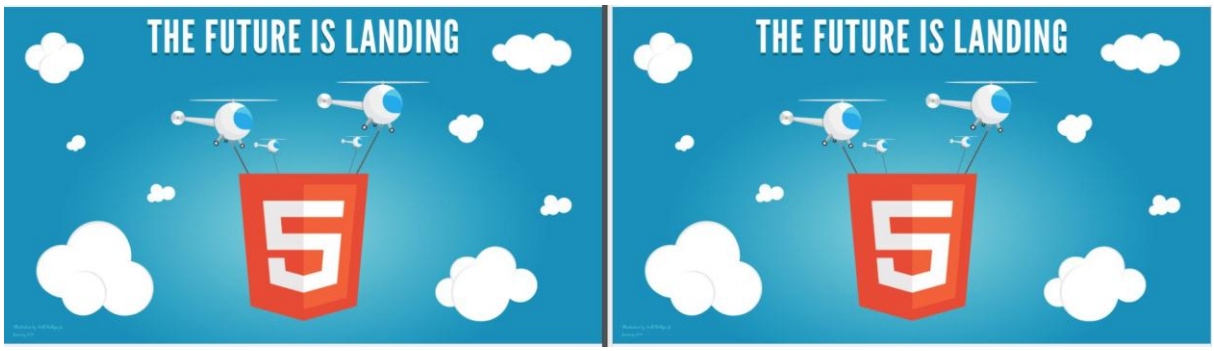


6.4. COMPRESSÃO LOSSY

Outra forma de compressão de imagem é a chamada Lossy. Esta técnica resulta em imagens mais leves que a técnica Lossless, porém a qualidade da imagem é reduzida e pode ser visível ao olho humano, mesmo que seja mínima a diferença. Para fazer esta compressão também foi utilizada a ferramenta Kraken.io. A ferramenta foi na mesma imagem da compressão Lossless, e depois feita a comparação para ver qual a porcentagem de economia de bytes.

A ferramenta com a opção Lossy selecionada conseguiu diminuir uma quantidade de 125,10 KB, diminuindo 61,44% do tamanho original, que agora ficou 78,51 KB.

A figura a seguir mostra uma comparação visual antes e depois de aplicar a compressão. A imagem da esquerda é a original, enquanto a da direita foi aplicada a compressão Lossy. Analisando a comparação, não é possível detectar perdas de detalhes nas imagens, mesmo que esta compressão exista perdas de informações. Esta compressão se torna a melhor opção para que os sites consigam diminuir o tamanho de MB trafegando, se o foco do site não ser exibição de imagens.



7. Conclusão

Este trabalho apresentou como proposta técnicas para melhorar a performance de aplicações do lado do cliente, otimizando ao máximo as tecnologias HTML5 Javascript e CSS3. Também teve como objetivo demonstrar a importância destas três tecnologias no cenário atual, que exige páginas dinâmicas, com boa usabilidade e que possam ser acessadas em qualquer dispositivo.

Os testes foram feitos em sites com milhões de acessos a fim de demonstrar que até grandes empresas como Google e Facebook apresentam algumas falhas de performance que poderiam ser resolvidas aplicando as técnicas apresentados neste projeto.

Após identificar problemas que afetam a performance de páginas WEB e aplicar técnicas para corrigir estes problemas, foi provado que é possível otimizar a performance das páginas, diminuindo significamente o tamanho em KB de arquivos e melhorando o tempo de acesso. Com os resultados obtidos, é possível chegar a conclusão que nem sempre o problema de lentidão no carregamento de páginas se dá graças a uma internet lenta, e sim graças a arquivos Javascript e CSS mal escritos e grandes de mais.

Este projeto demonstrou apenas algumas ferramentas e técnicas que podem melhorar a performance de páginas. Portanto, em futuros projetos, essas técnicas podem ser aplicadas utilizando novas ferramentas a fim de comparar qual é a melhor, ou apresentar técnicas diferentes de melhoria de performance.

8. Referências

SIMONI, R. HTML5 e SEO: mitos e verdades. Tableless, 2013. Disponível em: <http://tableless.com.br/html-5-e-seo-mitos-e-verdades/>. Acesso em: 10 maio 2014.

TOMAZ, R. Teste de desempenho – Conceitos importantes. Crowstest, 2013. Disponível em: <http://crowdtest.me/testes-desempenho-conceitos/>. Acesso em: 21 maio 2014.

EIS, D.; Ferreira, E. HTML5 e CSS3 com farinha e pimenta. Novatec Editora,2012 p. 278p

RODRIGUES, S. O que estudar para aprender CSS3? LadoDesign, 2012. Disponível em: <http://www.ladodesign.com.br/desenvolvimento-frontend/o-que-estudar-para-aprender-css3/ >. Acesso em: 12 maio 2014.

SILVA, M. S. CSS3 Desenvolva aplicações Web profissionais com uso dos poderosos recursos de estilização das CSS3. Novatec, 2012. 496p