UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

NATAN DE SOUZA RAMOS

DESENVOLVIMENTO DE UM APLICATIVO PARA APOIO AO ENSINO E APRENDIZAGEM DE SISTEMAS ESTRUTURAIS

CRICIÚMA

2015

NATAN DE SOUZA RAMOS

DESENVOLVIMENTO DE UM APLICATIVO PARA APOIO AO ENSINO E APRENDIZAGEM DE SISTEMAS ESTRUTURAIS

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador: Prof. MSc. Luciano Antunes

Coorientador: Prof. MSc. Marcio Vito

CRICIÚMA

2015

NATAN DE SOUZA RAMOS

DESENVOLVIMENTO DE UM APLICATIVO PARA APOIO AO ENSINO E APRENDIZAGEM DE SISTEMAS ESTRUTURAIS

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Navegadores de Internet.

Criciúma, 22 de junho de 2015.

BANCA EXAMINADORA

Prof. MSc. Luciano Antunes - (UNESC) - Orientador

Prof. MSc. Marcio Vito - (UNESC) - Coorientador

Prof. MEng. Evânio Ramos Nicoleit - (UNESC)

Prof. Esp. Gilberto Vieira da Silva - (UNESC)

Dedico este trabalho a alguns familiares e amigos que não estão mais entre nós e que perdi durante esta caminhada. Foram pessoas que estiveram sempre em meus pensamentos e, quando ainda em vida, me motivavam a alcançar o mais alto lugar e meus objetivos, acreditando na minha pessoa.

AGRADECIMENTOS

Gostaria de agradecer a Deus por ter me dado sabedoria e capacidade para a construção deste trabalho e para que eu pudesse concluir mais essa fase importante da vida.

A minha mãe Mariléia Cunha de Souza Ramos e ao meu pai Adroaldo Fregulia Ramos que me apoiaram em todos os momentos e me deram força para que eu seguisse adiante, lutando para me ver feliz.

Ao meu irmão Luan de Souza Ramos, que sempre me ajudou, me compreendeu e permaneceu ao meu lado, compartilhando os bons momentos e me ajudando a superar as dificuldades e angústias.

Aos meus irmãos Vagner de Souza Ramos e Vitor de Souza Ramos, pois na pequenez de crianças, foram gigantes em relação ao incentivo e me compreenderam quando não pude lhes dar atenção ou ajudar em alguma atividade.

Também gostaria de agradecer a todos os meus familiares, pois me transmitiram força e alegria, colaborando muito para que eu permanecesse motivado e com fé.

Ao meu amigo e professor Luciano Antunes, pela orientação neste trabalho, onde não mediu esforços para que tudo fosse realizado da melhor maneira possível.

Agradeço ao professor Marcio Vito pela coorientação neste trabalho, o qual dispôs de seu tempo para me ajudar nas questões pertinentes à sua área de atuação.

Agradeço aos meus colegas e amigos que pude conquistar ao longo desses cinco anos. Com a união e perseverança, aprendendo e ensinando, compartilhando e adquirindo conhecimento, conseguimos conquistar essa graduação.

“A percepção do desconhecido é a mais fascinante das experiências. O homem que não tem os olhos abertos para o misterioso passará pela vida sem ver nada.”

Albert Einstein

RESUMO

Em algumas situações o entendimento de certos conteúdos se torna complicado não apenas pela complexidade dos assuntos, mas também pela metodologia empregada. A utilização de novos métodos pedagógicos, combinados com recursos computacionais, pode facilitar a compreensão por parte dos acadêmicos. Desta forma, o aplicativo desenvolvido neste trabalho tem como finalidade auxiliar os acadêmicos da área da engenharia, na compreensão de assuntos e conceitos relacionados aos sistemas estruturais. Para o desenvolvimento desta ferramenta, foram exploradas as linguagens *client side*, com destaque para o HTML5, visto que o seu novo elemento, o *canvas*, possibilita o desenvolvimento de animações. As tecnologias *client side* são executadas pelos próprios navegadores de internet que buscam, por sua vez, uma padronização para atender as exigências propostas pelo World Wide Web Consortium (W3C) e dar suporte ao maior número possível de *tags*. Contudo, os navegadores podem apresentar diferentes comportamentos, considerando que possuem motores de renderização e de execução de JavaScript distintos. Frente a isso, este trabalho também consiste na comparação de desempenho entre os navegadores Google Chrome, Mozilla Firefox e Internet Explorer. Para isto, foi utilizada a ferramenta *jsPerf*, um software para montagem de cenários de teste, analisando a execução do código fonte. Na análise dos resultados observou-se que o HTML5 apresenta-se como uma tecnologia promissora para o desenvolvimento de aplicações web interativas. Já o elemento *canvas* pode ser utilizado para a renderização de imagens e formas, além de realizar animações, sendo suportado pelos principais navegadores utilizados. Sobre a comparação entre os navegadores, constatou-se que o Google Chrome apresentou o melhor desempenho, seguido pelo Mozilla Firefox e Internet Explorer. Além disso, o aplicativo trará uma forma alternativa de apoio ao aprendizado, pois o comportamento dos elementos estruturais poderá ser observado de maneira mais clara.

Palavras-chave: HTML5. Canvas. Navegadores. Comparação.

ABSTRACT

In some situations the understanding of certain content becomes complicated not only by the complexity of the issues, but also the methodology. The use of new teaching methods, combined with computational resources, can facilitate understanding on the part of academics. Thus, the application developed in this work aims to assist academic engineering area, the understanding of issues and concepts related to structural systems. For the development of this tool, the languages were explored client side, especially the HTML5, as its new element, canvas, enables the development of animations. The client side technologies are implemented by own Internet browsers seeking, in turn standardization to meet the requirements proposed by the World Wide Web Consortium (W3C) and support the greatest number of tags. However, browsers may have different behaviors, considering they have a rendering engine and separate JavaScript execution. Because of that, this work is the performance comparison between Google Chrome, Mozilla Firefox and Internet Explorer. For this, it used the jsPerf tool, a software for mounting test scenarios, analyzing the execution of the source code. The analysis of the results was observed that HTML5 is presented as a promising technology for developing interactive web applications. Since the canvas element can be used for rendering images and shapes, and perform animations, being supported by the main browsers used. About the comparison between browsers, it was found that Google Chrome had the best performance, followed by Mozilla Firefox and Internet Explorer. Furthermore, the application will support an alternative form of learning, because the behavior of the structural elements can be seen more clearly.

Palavras-chave: HTML5. Canvas. Browsers. Comparison.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CLATES	Centro Latino-Americano de Tecnologia Educacional
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
CSS	Cascading Style Sheets
DOM	Document Object Model
EDUCOM	Educação com Computadores
EUA	Estados Unidos da América
FINEP	Financiadora de Estudos e Projetos
FTP	File Transfer Protocol
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
ITU	International Telecommunication Union
JSF	Java Server Faces
MEC	Ministério da Educação
NTE	Núcleos de Tecnologia Educacional
NUTES	Núcleo de Tecnologia Educacional para a Saúde
PROINFO	Programa Nacional de Informática na Educação
SEED	Secretaria de Educação à Distância
SEI	Secretaria Especial de Informática
SVG	Scalable Vectorial Graphics
UFMG	Universidade Federal de Minas Gerais
UFPE	Universidade Federal de Pernambuco
UFRGS	Universidade Federal do Rio Grande do Sul
UFRJ	Universidade Federal do Rio de Janeiro
UnB	Universidade de Brasília
Unicamp	Universidade Estadual de Campinas
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WHATWG	Web HyperText Application Technology WorkingGroup
WWW	World Wide Web

SUMÁRIO

1 INTRODUÇÃO

A tecnologia está presente em muitas atividades desempenhadas pelos seres humanos, nas mais diversas áreas do conhecimento. Esta, quando associada ao uso do computador na educação, implica diretamente na melhoria do processo de ensino e aprendizagem. Sua utilização deve ser planejada, visando coerência com estratégias, métodos e técnicas de ensino, aproveitando suas qualidades de potencial (GRZESIUK, 2008).

Atualmente nas instituições de ensino, em muitas ocasiões, é utilizado um modelo pedagógico baseado em aulas expositivas e dialogadas. O conteúdo é distribuído em transparências, onde o professor vai realizando a explicação e ao final de um determinado assunto, são aplicados exercícios de fixação.

Muitas vezes, essa metodologia não é suficiente para que o acadêmico compreenda alguns conceitos importantes e de fácil percepção. De acordo com o coorientador deste trabalho e outros professores de engenharia da UNESC, esse problema está presente entre os acadêmicos da área. Um exemplo disso é quando o aluno realiza os cálculos, mas não entende o comportamento dos elementos estruturais.

Nos sistemas estruturais, onde se trabalha com o equilíbrio dos corpos, a compatibilidade de deslocamentos e a resistência dos materiais, fenômenos simples em seus princípios, o aluno tem dificuldades em observar o lado qualitativo, não tendo a percepção do problema (REBELLO, 2000).

Tomando como base esse contexto, a presente pesquisa aprofundou os conhecimentos relacionados aos sistemas estruturais, a fim de desenvolver um aplicativo que possa ser utilizado para qualificar e melhorar o processo de ensino e aprendizagem sobre este assunto.

O desenvolvimento desse aplicativo baseia-se nas tecnologias *client side*, onde a execução é realizada pelo próprio navegador de internet. Atualmente existem muitos navegadores disponíveis no mercado, estando estes em constante disputa para conquistar mais espaço e adquirir novos usuários.

No caso dos navegadores, softwares que tem a função básica de receber um script e renderizar a página para o usuário, alguns fatores podem ser avaliados para se testar o desempenho. A velocidade e compatibilidade com HTML5, a capacidade ao trabalhar com efeitos gráficos, ou até mesmo a simples execução de

um código JavaScript (SANTOS, 2013).

Na busca pelo melhor navegador de internet, deve-se considerar a preferência do usuário, o qual se identifica com um determinado navegador. No entanto, isto não significa que ele seja o melhor em termos técnicos e computacionais. Sabe-se que o desempenho de um navegador pode ser influenciado por inúmeros motivos, como a configuração do computador utilizado, o sistema operacional, a quantidade de aplicativos abertos simultaneamente, a velocidade da internet e os complementos instalados junto ao *browser* (HAUTSCH, 2010).

Frente a isso, esse trabalho buscou definir as métricas de comparação de desempenho entre os principais navegadores de internet, além de identificar e compreender os novos recursos trazidos pela linguagem HTML5 empregados no desenvolvimento do aplicativo.

1.1 OBJETIVO GERAL

Analisar comparativamente o desempenho de um aplicativo desenvolvido em HTML5 em diferentes navegadores.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste trabalho são:

a) compreender os sistemas estruturais na engenharia;
b) desenvolver um aplicativo para apoio ao ensino e aprendizagem de sistemas estruturais por meio da tecnologia HTML5;
c) identificar as métricas para a comparação da aplicação desenvolvida com a tecnologia HTML5;
d) realizar uma análise comparativa da ferramenta desenvolvida, em diferentes navegadores.

1.3 JUSTIFICATIVA

Em meio ao avanço tecnológico, os computadores e demais recursos estão sendo utilizados em diversos setores da sociedade, estando cada vez mais presentes no processo de ensino e aprendizagem. A utilização de diversos softwares educacionais, em diferentes modalidades, comprova que esta ferramenta pode ser muito útil (GRZESIUK, 2008).

Os estudantes podem aprender melhor e mais depressa quando são estimulados por um alto nível de interatividade, podendo seguir um modo pessoal de aprendizado. Em particular, eles podem personalizar sua participação junto ao software didático, estabelecendo a quantidade de tempo gasto em cada tópico e determinando seus próprios passos de aprendizado (GONÇALVES; CANESIN, 2002).

Desta forma, fica evidente a importância da utilização do computador e de softwares educacionais para melhorar o processo de ensino e aprendizagem. O desenvolvimento da ferramenta irá proporcionar condições mais adequadas para o entendimento dos sistemas estruturais, sendo que os acadêmicos de engenharia terão uma alternativa para tornar o seu aprendizado mais dinâmico e interessante.

Sempre que temos a opção de escolha entre vários softwares que desempenham uma determinada função, nos deparamos com aspectos que influenciam na tomada de decisão. Essa situação não poderia ser diferente com os navegadores, onde existem fatores objetivos e subjetivos que determinam essa escolha. Os fatores objetivos são aqueles que podem ser medidos sem interferência pessoal, como os resultados de benchmarks e que levam a uma comparação mais precisa entre os softwares (SANTOS, 2013).

Como nem todos os navegadores utilizam o mesmo motor JavaScript e implementam a especificação do HTML5 da mesma maneira, acaba existindo diferença entre eles, principalmente no que diz respeito ao desempenho (HARBS, 2013).

Com as mudanças e evoluções da web, os navegadores não tem apenas a função de exibir conteúdos da internet, mas devem suportar um alto nível de interatividade, onde as páginas ganham animações e elementos visuais para diversificar a experiência do usuário. Desta forma, esse tipo de software passou a ter grande importância para os usuários e profissionais da área tecnológica. Devido à

necessidade de oferecer mecanismos de segurança da informação, estabilidade, interatividade e velocidade na exibição do conteúdo, a escolha de um navegador se tornou algo importante tanto para os usuários, quanto para os desenvolvedores (PRASS, 2011).

Tendo em vista que foram envolvidas linguagens atuais como HTML5 e JavaScript, a comparação entre os navegadores realizada neste trabalho, demonstra a situação de cada *browser* em relação à compatibilidade e desempenho na execução de algumas funcionalidades das páginas web. Esses resultados podem auxiliar na escolha para a adoção de um determinado navegador, ou até mesmo, indicar quais tecnologias deve-se utilizar para o desenvolvimento web, dependendo do que se deseja fazer.

1.4 ESTRUTURA DO TRABALHO

O trabalho possui uma estrutura composta por sete capítulos onde no primeiro capítulo é apresentada a introdução, o objetivo geral e os objetivos específicos, além da justificativa para o desenvolvimento do trabalho.

O segundo capítulo aborda os conhecimentos sobre informática na educação. Neste capítulo também é apresentada um pouco da história da informática na educação no Brasil, onde são destacados os principais acontecimentos ao longo dos anos.

O capítulo três é destinado ao estudo de sistemas estruturais. São estudados conceitos relacionados à área de engenharia como mecânica dos corpos rígidos, deformações dos elementos estruturais e resistência dos materiais. Também são abordados assuntos como compressão, tração e flexão, além de se apresentar de forma breve, a função de um engenheiro.

No capítulo quatro são apresentados conteúdos relacionados á área tecnológica. Inicialmente é apresentada a estrutura da web, seu surgimento e evolução, bem como os principais navegadores de internet. Posteriormente são demonstradas as tecnologias *cliente side,* além das principais novidades disponibilizadas pelo HTML5, com destaque para o elemento *canvas.*

O quinto capítulo apresenta os trabalhos correlatos, onde são demonstrados outros trabalhos que se assemelham ao trabalho desenvolvido. Para

cada trabalho correlato, é descrito o que é o trabalho, como foi desenvolvido e os resultados que foram alcançados.

O capítulo seis trata do desenvolvimento do trabalho. Este capítulo tem o objetivo de detalhar as etapas metodológicas necessárias para o desenvolvimento do trabalho. Inicialmente são apresentados os principais passos e procedimentos para a implementação do aplicativo. Na sequencia é descrita com mais detalhes a comparação de desempenho, deste o funcionamento do software utilizado, até a montagem dos cenários de teste. Ao final do capítulo são demonstrados os resultados obtidos com a comparação de desempenho entre os navegadores.

No último capítulo foi elaborada a conclusão onde são expostas as considerações finais sobre o que foi alcançado com o desenvolvimento do trabalho. Também são apresentadas as sugestões e oportunidades para elaboração de trabalhos futuros.

2 INFORMÁTICA NA EDUCAÇÃO

Um dos alicerces fundamentais de um país para a sustentação e qualidade do seu desenvolvimento, é a educação. Nas últimas décadas pesquisadores vem estudando conceitos e técnicas para o desenvolvimento de ferramentas a fim de auxiliar e melhorar o processo educacional em diversos níveis e áreas do conhecimento (GONÇALVES; CANESIN, 2002).

A tecnologia vem adquirindo cada vez mais espaço e importância no cenário educacional. A utilização de seus recursos vem sendo cada vez mais acentuada e crescendo de forma muito rápida, o que proporciona mudanças estruturais e funcionais no processo de educação. Os avanços tecnológicos nos influenciam não apenas no que fazemos, mas também em nosso comportamento, na maneira em que construímos nosso conhecimento e nos relacionamos com o mundo (WALNIER, 2006).

A internet, os smartphones, a televisão e demais dispositivos tecnológicos, estão em constante evolução e mudança, influenciando diretamente na maneira como nos relacionamos, recebemos e trocamos informações. As pessoas estão cada vez mais conectadas, aumentando assim as possibilidades de construção do conhecimento, facilitando a pesquisa, a aprendizagem e tarefas como compras, pagamentos e outros serviços. O mundo real e o virtual se completam e estão integrados, sendo que o acesso aos recursos tecnológicos deve ser considerado um requisito para uma cidadania plena (MORAN, 2007).

Com as possibilidades tecnológicas disponíveis nos dias atuais, o processo de formação do educador também é afetado, pois existe a necessidade de adaptação do mesmo em relação às novas ferramentas que podem ser empregadas no ensino. O professor, além de compreender sobre assuntos da sua área de atuação, deve se capacitar e identificar a forma mais eficaz de emprego dos recursos computacionais. De qualquer forma, não se deve responsabilizar por completo a pessoa do educador no que diz respeito à inserção da informática na educação, sendo que toda a comunidade escolar também deve se adaptar as evoluções tecnológicas (WALNIER, 2006).

No momento em que se pretende aplicar a informática na educação, é papel da entidade escolar proporcionar a infraestrutura adequada, além disso, incentivar o trabalho em equipe, integração de pessoas e a capacidade do aluno

para pensar e tomar decisões. O educador deve estar disposto a buscar novas formas de ensino e se adaptar a rápidas mudanças, ser dinâmico e flexível, estando ciente do seu papel de facilitador e coordenador do processo de ensino e aprendizagem (NASCIMENTO, 2007).

Tendo em vista essas mudanças no processo de ensino e aprendizagem em virtude das possibilidades de utilização de recursos tecnológicos combinados com as propostas pedagógicas dos professores, o Ministério da Educação (MEC) cita algumas características importantes para que o educador possua um perfil inovador (NASCIMENTO, 2007):

a) estar sempre disposto a adquirir novos conhecimentos;
b) trabalhar com temas emergentes no contexto de interesse dos alunos;
c) promover o desenvolvimento de projetos cooperativos com maior interação dos alunos, e até mesmo projetos interdisciplinares;
d) incentivar a reflexão, a depuração e o pensar sobre os assuntos abordados;
e) dominar recursos computacionais;
f) identificar as potencialidades de aplicações dos recursos computacionais na prática pedagógica.

A sociedade está mudando seu perfil para uma sociedade onde surgem novas formas de aprender, com novos participantes que são muito efetivos, e a busca do conhecimento é contínua. A educação escolar precisa auxiliar o aprendizado de forma mais afetiva e ética, explorando novos métodos e tecnologias com objetivo de construir cidadãos mais plenos, completos em diversas dimensões (MORAN, 2007).

Observando a grande contribuição trazida pelo emprego da informática na educação, representada pelo computador, o governo brasileiro tomou iniciativas para apoiar essa prática, com o objetivo de melhorar o processo de ensino e aprendizagem e garantir aos alunos o acesso a esses novos recursos.

2.1 INFORMÁTICA NA EDUCAÇÃO NO BRASIL

A informática na educação no Brasil deu seus primeiros passos no início da década de 1970, onde em 1971 na Universidade Federal de São Carlos, em São Paulo, foi promovido um seminário sobre o uso de computadores no estudo de

física, tendo a presença do especialista Elisha Rhoders Huggins, da Universidade de Dartmouth, Estados Unidos da América (EUA) (WALNIER, 2006).

Nesta mesma época outras três universidades brasileiras iniciaram alguns trabalhos utilizando a informática na educação. Em 1973 na Universidade Federal do Rio de Janeiro (UFRJ), o Núcleo de Tecnologia Educacional para a Saúde (NUTES) e o Centro Latino-Americano de Tecnologia Educacional (CLATES) utilizaram software de simulações para o ensino de química. No mesmo ano na Universidade Federal do Rio Grande do Sul (UFRGS), com auxílio de software, foi realizado o simulado de fenômenos de física com alunos de graduação. Em 1974 na Universidade Estadual de Campinas (Unicamp), foi desenvolvido por um aluno de iniciação científica, um software na linguagem BASIC, para facilitar o próprio ensino de fundamentos de programação BASIC, utilizado por Mestrandos de Ensino de Ciência e Matemática desta universidade (WALNIER, 2006).

Em 1981, no período de 25 a 27 de agosto, acontece na Universidade de Brasília (UnB), o I Seminário Nacional de Informática na Educação, promovido pela Secretaria Especial de Informática (SEI), Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Financiadora de Estudos e Projetos (FINEP) e pelo MEC. Esse evento teve participação de especialistas nacionais e internacionais, sendo o primeiro fórum a discutir o tema no país, onde foi reconhecida a importância de pesquisas para a aplicação do computador na educação, e esclarecendo que o mesmo serve para ampliar as funções do educador e jamais substituí-lo (NASCIMENTO, 2007).

No ano de 1982, é promovido no mês de agosto, pela SEI, CNPq e MEC, o II Seminário Nacional de Informática na Educação, com participação de especialistas das áreas de educação, psicologia, informática e sociologia. Neste encontro foi reforçada a ideia de que o computador seria apenas um recurso auxiliar no processo educacional e não o determinador único das atividades a serem desenvolvidas. Ainda foi proposto que os recursos computacionais não fossem aplicados apenas no 2º grau, de acordo com o objetivo inicial do governo federal, mais em outros níveis e também devia assumir um perfil de interdisciplinaridade (NASCIMENTO, 2007).

Com essas iniciativas realizadas no início dos anos 80, em 1983, foi criado o Projeto EDUCOM, que consistia na implantação de centros-piloto nas universidades públicas do país. Neste mesmo ano, 26 universidades apresentaram

projetos com o intuito de implantar o centro-piloto, no entanto apenas cinco universidades foram aprovadas, sendo elas a Universidade Federal de Pernambuco (UFPE), Universidade Federal de Minas Gerais (UFMG), UFRJ, UFRGS e Unicamp (TAVARES, 2008).

O Projeto EDUCOM tinha como principais objetivos pesquisar sobre o uso de informática educacional e realizar uma melhor formação de recursos humanos, ou seja, capacitar os professores. Os centros-piloto também investiram na produção de softwares educacionais e realizaram pesquisas na área de educação especial (TAVARES, 2008).

Com o passar dos anos o Projeto EDUCOM foi sendo ampliado e passou a estar presente em outras regiões do país, além disso, esse projeto serviu de inspiração para uma das principais ações do governo federal em relação à informática na educação no Brasil, sendo criado no ano de 1997 o Programa Nacional de Informática na Educação (PROINFO), idealizado pela Secretaria de Educação à Distância (SEED) e pelo MEC (TAJRA, 2000).

Esse projeto tem como finalidade promover o uso de recursos tecnológicos na educação pública de ensino fundamental e médio, implantando de forma descentralizada os Núcleos de Tecnologia Educacional (NTE), os quais possuem uma infraestrutura adequada prestando apoio à informatização das escolas no que diz respeito ao planejamento do emprego de novas tecnologias, suporte técnico e capacitação de professores (FNDE, 2014).

Os principais objetivos do PROINFO são (TAJRA, 2000):

a) elevar a qualidade do processo de ensino e aprendizagem;
b) disponibilizar ambientes de aprendizagem inovadores, com a infraestrutura adequada, garantindo o acesso a novas tecnologias;
c) estimular uma educação voltada ao desenvolvimento científico e tecnológico;
d) capacitar os estudantes para uma cidadania global numa sociedade tecnologicamente desenvolvida.

Pode-se observar que ao longo da história o computador foi sendo inserido na educação de nosso país, e com o passar dos anos foi ganhando mais espaço no cenário educacional, motivando a organização de encontros de especialistas para a discussão do tema, fazendo com que o governo intervisse com a criação de programas para apoiar essas iniciativas.

3 SISTEMAS ESTRUTURAIS

A mecânica dos corpos rígidos forma uma base para o projeto e análise de componentes estruturais, mecânicos e elétricos que são estudados pela engenharia. A mecânica pode ser definida como um ramo dedicado ao estudo dos corpos, seu estado de repouso ou movimento em função de forças que a eles são aplicadas (HIBBELER, 1999).

A mecânica dos corpos rígidos é divida em duas partes, sendo estática, onde é tratado o equilíbrio dos corpos que estão em repouso ou que sofrem algum movimento em velocidade constante, e a dinâmica, que estuda os corpos que se movem com uma velocidade variável (HIBBELER, 1999).

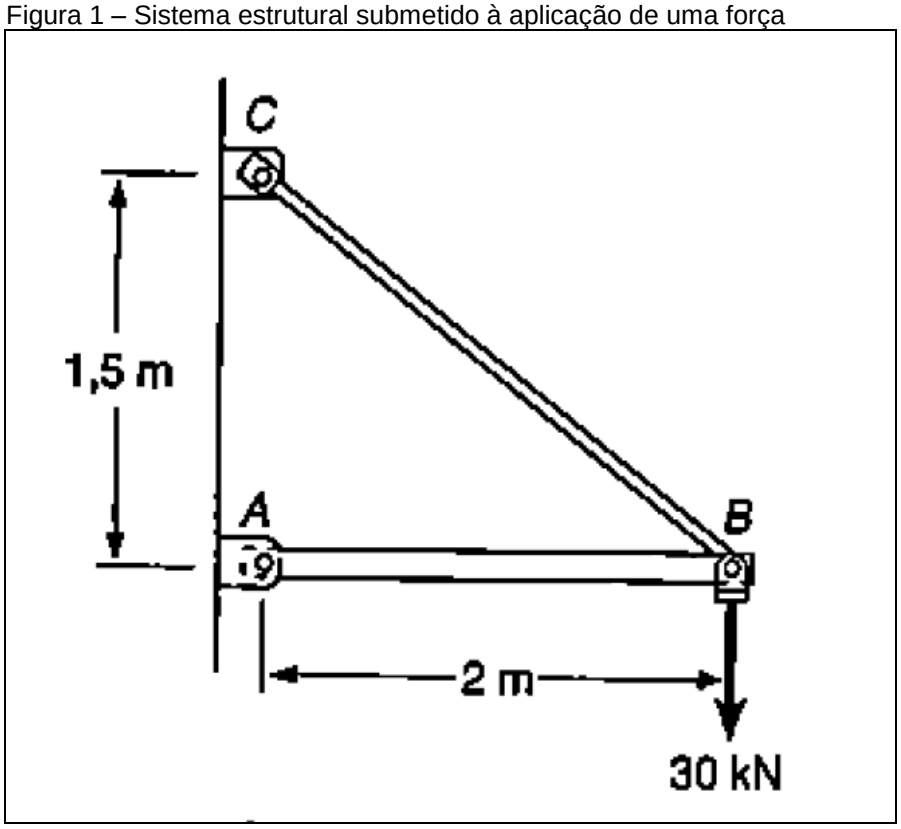
Quando uma força é aplicada a um corpo, ela pode causar alterações na forma e dimensão deste corpo. Essas alterações podem ser chamadas de deformações, onde em alguns casos são perceptíveis a olho nu, mais em outros exigem o uso de equipamentos de medida muito precisos para serem identificadas. A deformação de um elástico ao ser tracionado é facilmente enxergada. Por outro lado, quando existem várias pessoas em movimento dentro de um prédio, os elementos estruturais sofrem deformações milimétricas. Alguns corpos também podem sofrer deformações quando submetidos a certa temperatura, pois o material que o compõem pode sofrer alterações em suas propriedades (HIBBELER, 2000).

A resistência de um material está diretamente ligada à sua capacidade de suportar a ação de uma carga sem apresentar deformações indevidas ou falhas que podem comprometer a sua função em um sistema estrutural. Essa propriedade é uma característica particular, apresentando variações de um material para outro, podendo estar sujeito a cargas estáticas, cíclicas, de grande duração ou impulsivas. Para determinar essa resistência devem ser feitos experimentos de forma padronizada para se obter os resultados de cada tipo de material. Um dos testes mais importantes é o ensaio de tração ou compressão. Ele é utilizado principalmente para determinar a relação entre a tensão normal média e a deformação normal média de muitos materiais da engenharia como metais, cerâmicas, entre outros (HIBBELER, 2000).

As deformações originadas pela aplicação de forças podem causar diferentes comportamentos dos elementos estruturais. Dentre algumas das ações sofridas por esses elementos estão a compressão, tração, torção e flexão. Na figura

1 é demonstrada uma estrutura onde é aplicada uma força de 30 KN no ponto *B.* Desta forma a barra entre os pontos *C* e *B* tende a ser tracionada, ou seja, esticada. Enquanto isso a barra compreendida entre os pontos *A* e *B* sofre uma compressão, pois a força exercida no ponto *B* empurra a barra contra o seu apoio no ponto *A* (BEER; JOHNSTON JUNIOR, 1995).

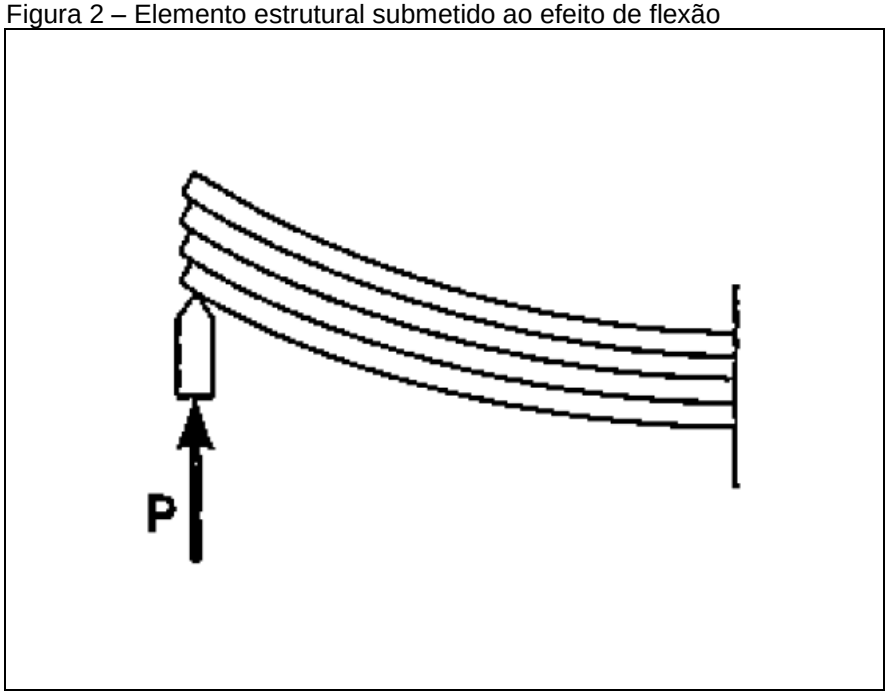
Figura 1 – Sistema estrutural submetido à aplicação de uma força



Fonte: Beer; Johnston Junior (1995).

A figura 2 apresenta uma barra que está fixada em uma base. Esta barra sofre a ação de uma força indicada pelo deslocamento vertical que a impulsiona para cima. Como a barra está presa à base, ela não cede ao esforço e acaba sendo flexionada, ou seja, tende a se curvar. Caso a barra não estivesse totalmente presa a uma base fixa, ela iria ser impulsionada no sentido da aplicação da força, mas não iria se curvar, mantendo a sua forma original (BEER; JOHNSTON JUNIOR, 1995).

Figura 2 – Elemento estrutural submetido ao efeito de flexão



Fonte: Beer; Johnston Junior (1995).

Os engenheiros têm como função analisar as estruturas e máquinas já existentes, as quais devem estar planejadas com o objetivo de suportar os mais diversos carregamentos a que são submetidas. Também cabe a esses profissionais, projetar novas estruturas de maneira que exerçam sua função de forma correta e segura, além de definir os componentes estruturais necessários, isto baseados nas diferentes propriedades de cada material (BEER; JOHNSTON JUNIOR, 1995).

A introdução e entendimento de programação de computadores nos cursos de engenharia, assim como a disponibilidade de computadores pessoais e terminais de rede nos campus das universidades, proporcionam aos acadêmicos desta área, utilizarem destes recursos para a resolução de muitos problemas encontrados no seu campo de atuação (BEER; JOHNSTON JUNIOR, 1995).

O quarto capítulo fala sobre a web, seu surgimento e seu funcionamento, trazendo informações históricas. Também são abordadas as tecnologias *client side* bem como os navegadores, destacando os três principais navegadores utilizados atualmente. O capítulo também trata do HTML, desde a sua criação, até a versão cinco. São apresentadas as principais mudanças e novas funcionalidades trazidas pelo HTML5, onde o elemento *canvas* e o *SVG* recebem uma atenção especial.

4 APLICAÇÕES WEB

A Internet pode ser considerada um dos meios mais importantes de comunicação do mundo atual, aproximando pessoas, organizações e sistemas. A evolução e crescimento de aplicações para o mundo virtual têm sido cada vez mais impressionantes. É possível notar que, alguns anos atrás era complicado o desenvolvimento de uma aplicação web que exigisse a troca de informações com um banco de dados. Atualmente isso é muito comum (SOARES, 2006).

A Word Wide Web (WWW) foi proposta inicialmente em 1989 pelo físico britânico Tim Berners-Lee, o qual trabalhava no CERN[1]. O propósito original da web era o compartilhamento de informações de forma automática entre cientistas em universidades e instituições de todo o mundo (CERN, 2014a, tradução nossa).

A proposta feita por Tim Berners-Lee não foi muito aceita inicialmente. No entanto ele persistiu com a ideia e em outubro de 1990 definiu três tecnologias fundamentais que são a base da internet até hoje, sendo o HyperText Markup Language (HTML), o Uniform Resource Identifier (URI) e o HyperText Transfer Protocol (HTTP). Os anos foram passando e cada vez mais pessoas começaram a entender e utilizar a web, colaborando muito para o seu crescimento. Em abril de 1993 o CERN anunciou que a Wold Wide Web estaria disponível para qualquer um usar em uma base livre de royalties (WORLD WIDE WEB FOUNDATION, 2014, tradução nossa).

O primeiro site da web foi dedicado a sua própria criação, sendo hospedado no computador NeXT de Tim Berners-Lee. O site apresenta informações sobre como utilizar a web, acessar diversos documentos, como se comunicar com outras pessoas e até como criar seu próprio servidor. A máquina NeXT, o servidor web original, ainda está no CERN, e no ano de 2013 o primeiro site foi reintegrado ao seu endereço original: http://info.cern.ch/hypertext/WWW/TheProject.html (CERN, 2014a, tradução nossa).

Tim Berners-Lee e demais pessoas perceberam que para um maior aproveitamento do potencial da web, as tecnologias envolvidas deveriam ser

[1] O CERN é uma organização que realiza pesquisas nucleares, onde são estudados os componentes básicos da matéria, como as partículas interagem, e as leis fundamentais da natureza. Fundado em 1954, o laboratório CERN fica localizado próximo à cidade de Genebra na Suíça. O nome CERN vem do francês *Conseil Européen pour la Recherche Nucléaire,* ou Conselho Europeu de Pesquisa Nuclear (CERN, 2014b, tradução nossa).

padronizadas, implementadas da mesma forma em todo o mundo, então fundou no ano de 1994 o World Wide Web Consortium (W3C). O W3C é um ambiente onde as partes interessadas compartilham informações e conhecimento a fim de chegar a um consenso em torno da especificação e diretrizes que garantem a disponibilização mundial da internet, bem como sua evolução de forma responsável e inovadora (WORLD WIDE WEB FOUNDATION, 2014, tradução nossa).

Um navegador, também conhecido pelos termos ingleses *web browser* ou simplesmente *browser,* é um programa de computador que possibilita aos usuários interagirem com os documentos da Internet, ou páginas web, as quais podem ser elaboradas com o uso de HTML, ASP, PHP e CSS, entre outras tecnologias, ficando hospedadas em um servidor web (SOUSA, 2011).

A tabela 1 mostra uma relação dos principais navegadores utilizados atualmente, demonstrando o percentual de uso de cada um deles.

Tabela 1 – Estatística de utilização dos principais *browsers*

2014	Chrome	Internet Explorer	Firefox	Safari	Opera
Outubro	60.4%	9.5%	23.4%	3.9%	1.6%
Setembro	59.6%	9.9%	24.0%	3.6%	1.6%
Agosto	60.1%	8.3%	24.7%	3.7%	1.8%
Julho	59.8%	8.5%	24.9%	3.5%	1.7%
Junho	59.3%	8.8%	25.1%	3.7%	1.8%
Maio	59.2%	8.9%	24.9%	3.8%	1.8%
Abril	58.4%	9.4%	25.0%	4.0%	1.8%
Março	57.5%	9.7%	25.6%	3.9%	1.8%
Fevereiro	56.4%	9.8%	26.4%	4.0%	1.9%
Janeiro	55.7%	10.2%	26.9%	3.9%	1.8%

Fonte: W3SCHOOLS (2014a)

O *browser* se comunica geralmente com servidores web por meio do protocolo de transferência HTTP, mas pode ter suporte a outros protocolos desta natureza, como o File Transfer Protocol (FTP), para transferência de arquivos, e o HyperText Transfer Protocol Secure (HTTPS), uma versão criptografada do HTTP. A principal função de um *browser* é realizar o pedido de um determinado conteúdo da web e providenciar a exibição do mesmo, tendo um papel de cliente, pois faz o pedido e aguarda a sua entrega (SOUSA, 2011).

Pode-se observar que os *browsers* mais utilizados atualmente são o Google Chrome, Mozilla Firefox e Internet Explorer. Na realização deste trabalho

serão adotados justamente estes três navegadores (W3SCHOOLS, 2014a, tradução nossa).

O Internet Explorer, também conhecido como IE, é um navegador de internet de licença proprietária desenvolvido pela Microsoft, lançado em agosto de 1995, sendo disponibilizado junto ao sistema operacional Windows 95. Neste momento a principal concorrente da Microsoft na disputa dos navegadores era a Netscape que possuía o *browser* Netscape Navigator. O IE continuou sendo disponibilizado junto ao sistema operacional da Microsoft e à medida que o Windows se popularizava, o IE seguia o mesmo caminho, conquistando o posto de navegador mais utilizado no mercado (RUSSO, 2010).

Recentemente a Microsoft vinha trabalhando em um novo projeto de navegador chamado Project Spartan, e no início de 2015 anunciou que o Internet Explorer será substituído pelo navegador Microsoft Edge, resultante do projeto mencionado acima. Esse novo navegador será adicionado ao Windows 10, o qual será lançado em 2015. O Microsoft Edge terá um motor de renderização novo chamado EdgeHTML além de possuir uma interface minimalista, seguindo a tendência do concorrente Google Chrome (HIGA, 2015).

O Mozilla Firefox é um navegador de internet desenvolvido pela Fundação Mozilla, ou Mozilla Foundation, cujo lançamento se deu no dia 9 de novembro de 2004, período em que o navegador predominante era o Internet Explorer. Inicialmente o *browser* se chamou Phoenix, mas o nome precisou ser alterado devido a conflitos legais. Com isso passou a se chamar Mozilla Firebird, no entanto o nome coincidiu com o de outro programa já existente. Desta forma foi adotada a nomenclatura Mozilla Firefox, homenageando uma espécie de urso panda (LANDIM, 2011).

Inicialmente o Firefox não apresentou muitas mudanças em relação ao Internet Explorer, tendo destaque o gerenciamento de senhas. Com o passar do tempo o navegador começou a ser utilizado pelo fato de ser uma alternativa ao predominante Internet Explorer. Na sua segunda versão, o Firefox lançou algumas novidades que foram de extrema importância para a sua inserção no mercado de navegadores de internet. O *browser* da Mozilla foi o pioneiro na utilização de abas, não sendo mais necessária a abertura de várias janelas durante a navegação. Até o momento os fabricantes não se preocupavam muito com a disponibilização em diversos idiomas, o que foi aprimorado pelo Firefox que disponibilizou algumas

versões em diferentes idiomas (LANDIM, 2011).

O navegador também inseriu a funcionalidade para restauração de páginas, onde ao reiniciar o navegador, as páginas anteriores são carregadas novamente, garantindo que o usuário não perca seu histórico de navegação quando o *browser* é fechado de maneira inesperada. A possibilidade de adição de extensões como plug-ins, por exemplo, é uma característica marcante do Mozilla Firefox, permitindo que novas funcionalidades sejam acopladas ao navegador (LANDIM, 2011).

O Mozilla Firefox utiliza o motor de renderização Gecko e o motor de execução de JavaScript chamado SpiderMonkey, sendo que a performance em relação à execução de JavaScript geralmente é criticada pelos usuários e desenvolvedores. À medida que o navegador foi se consolidando e agregando funções, também aumentaram a preocupação com a segurança e sempre se buscou o aperfeiçoamento, garantindo o funcionamento de tecnologias como HTML5 e CSS3 (LANDIM, 2011).

O Google Chrome é um navegador de internet desenvolvido pela Google, lançado em setembro de 2008. A Google percebeu que os navegadores da época não supriam as novas necessidades da web, sendo que eram destinados a manipular e organizar páginas repletas de textos. A criação do Chrome levou em consideração aplicações mais interativas e dinâmicas, onde os usuários podem ouvir música, assistir a vídeos, realizar conversas e até jogar games online (AQUINO, 2011).

O Google Chrome foi desenvolvido utilizando o motor de renderização WebKit e o motor V8 para a execução de JavaScript. A combinação dessas tecnologias resultou em um navegador muito veloz, com um desempenho satisfatório em relação aos *browsers* já existentes. Outra característica importante do Chrome, é que ele trabalha com o conceito de operações independentes, ou seja, cada página é tratada como um novo processo, garantindo que, caso uma página apresente erro, as outras não sejam afetadas (AQUINO, 2011).

Com o passar dos anos a Google investiu muito em seu navegador, realizando melhorias contínuas e disponibilizando novas versões, onde também eram corrigidos os erros e falhas de segurança tornando o *browser* cada vez mais consistente. Todo esse investimento fez com que o Google Chrome se tornasse um dos navegadores mais importantes e utilizados pelos usuários da web. Um aspecto

muito importante e bem recebido pelos usuários foi a interface simples e compacta oferecida por este navegador (AQUINO, 2011).

A partir do lançamento do Mozilla Firefox, no ano de 2004, o Internet Explorer começou a perder espaço no mercado de navegadores. Em 2008, com o surgimento do Google Chrome, o IE, perdeu ainda mais usuários, assim como o Firefox, pois o navegador da Google teve uma ótima recepção pelos utilizadores da web. Sendo assim, hoje os navegadores de internet mais utilizados são o Google Chrome, Mozilla Firefox e Internet Explorer, seguindo essa ordem de utilização.

4.1 CLIENT SIDE: TECNOLOGIAS EXECUTADAS PELO BROWSER

Apesar de ser a principal tecnologia que matem a estrutura das páginas da internet, o HTML não dispõe de todos os recursos e funcionalidades que são necessários para o desenvolvimento de aplicações web. Por estas razões são utilizadas junto a ele, outras linguagens que possibilitam a construção de páginas com grande interatividade e com funcionalidades mais complexas, a fim de atender as exigências da web (RODRIGUES; PRADO, 2014).

As tecnologias *client side* são aquelas executadas no lado do cliente, ou seja, pelo *browser* que está sendo utilizado pelo usuário. Sua utilização é muito importante, pois permite a criação de páginas web mais sofisticadas, ao invés de apenas a exibição do HTML que é retornado pelo servidor. Além disso, com essas tecnologias é possível a criação de alguns procedimentos básicos da web sem a necessidade de acessar o servidor. Um exemplo é a validação das informações de um formulário quando o mesmo é submetido na aplicação (LOBO FILHO, 2010).

Dentre as tecnologias mais utilizadas junto ao HTML, estão o CSS e o JavaScript, todas elas *client side*. O HTML é responsável por fornecer a estrutura da página, a construção do conteúdo. O CSS permite a formatação e customização da informação, podendo definir aspectos do leiaute como cores, por exemplo. O JavaScript tem um papel fundamental, pois realiza a parte lógica do programa, possibilita a execução das funções e animações da página (RODRIGUES; PRADO, 2014).

A figura 3 faz uma demonstração das responsabilidades assumidas por cada tecnologia citada acima, no desenvolvimento web.

Figura 3 – Demonstração das funções do HTML, CSS e JavaScript

Fonte: Serra (2011).

As tecnologias *client side* são muito importantes no desenvolvimento web, pois proporcionam a criação de páginas bem formatadas, com acessibilidade, usabilidade e arquitetura de informação. Além disso, uma implementação *client side* bem realizada, possibilita o acesso de conteúdos por diferentes usuários e por diferentes dispositivos (TABLELESS, 2009).

4.2 HTML5

No início da década de 1990, quando recentemente havia sido proposta a World Wide Web (WWW), enfrentavam-se problemas para a publicação de conteúdos devido à diversidade de formato, estrutura e leiaute dos documentos. Com o objetivo de padronizar a estrutura dos documentos textuais foi desenvolvida a HyperText Markup Language (HTML) e também o protocolo de comunicação HTTP, que possibilitaram a disseminação em massa dos conteúdos hipertextos na rede mundial de computadores (DIAS, 2007).

O HTML é uma linguagem de marcação e construção de conteúdos para a web, como textos, imagens, áudio, vídeo e animações. O HTML foi desenvolvido inicialmente por Tim Berners-Lee no início da década de 1990, quando também se iniciava a World Wide Web, com o intuito de publicar conteúdos e poder interliga-los eletronicamente (SILVA; SANTOS, 2013).

No período de 1993 a 1995 surgiram as versões HTML+, HTML 2.0 e HTML 3.0, mantidas pelo Internet Engineering Task-Force (IETF), as quais apresentavam algumas mudanças com o objetivo de enriquecer a linguagem, no entanto durante esses anos o HTML não era reconhecido pelo W3C como um padrão de desenvolvimento para a internet. Esse reconhecimento veio em 1997,

quando a entidade passou a trabalhar com a versão 3.2 que foi padronizada e definida como prática comum entre os desenvolvedores (SILVA; SANTOS, 2013).

No final do ano de 1997 foi publicado o HTML 4.0 que era uma recomendação do W3C e trouxe vários elementos e atributos diferentes, mais ao mesmo tempo retirou algumas marcações proprietárias e em desuso dando lugar as folhas de estilo. Em 1999 foi publicado o HTML 4.01 também sendo uma recomendação do W3C, possuindo algumas melhorias em relação ao HTML 4.0. No dia 12 de maio de 2001 foi publicada uma errata para o HTML 4.01, sendo esta a revisão mais recente do HTML até o início dos trabalhos com a versão 5 (SOUSA, 2011).

Com o passar dos anos a internet foi evoluindo, assim como a necessidade por páginas mais sofisticadas, que apresentassem uma maior interatividade, formando aplicações mais complexas. Para que isso fosse possível, novas tecnologias eram necessárias, além de superar as limitações do próprio HTML, dos navegadores e a baixa capacidade da banda de transmissão da internet (LALLI; BUENO; ZACHARIAS, 2008).

Com essas novas exigências que surgiam no desenvolvimento web, os navegadores não puderam acompanhar as novas tecnologias, e ao invés de modificarem os *browsers*, os idealizadores da web propuseram o uso de plug-ins. Os plug-ins são programas que trabalham junto ao *browser* e executam aplicações por meio de arquivos específicos que o navegador não poderia executar sozinho. Normalmente os plug-ins são fáceis de serem baixados e instalados (HARRIS, 2014).

Com o passar dos anos, foram surgindo alguns problemas na utilização de plug-ins, pois não são nativos aos navegadores, dependendo dos usuários para a instalação e atualização dos mesmos. Além disso, não possuem uma padronização apurada e não tem sua codificação aberta, impedindo adaptações. O HTML5 visa minimizar esses problemas além de agregar novas funcionalidades para atender as exigências da web atual (SCHROEDER, 2012).

Após o lançamento do HTML 4.01, o W3C começou a especificação de uma nova forma de desenvolvimento para a web, o XHTML2, com o intuito de suprir as falhas do HTML 4.01. Essas novas ideias exigiam um novo aprendizado por parte dos desenvolvedores, além da necessidade de reconstrução dos sites que já

existiam. Essas dificuldades fizeram com que o mercado não fosse muito receptivo ao XHTML2 (SCHROEDER, 2012).

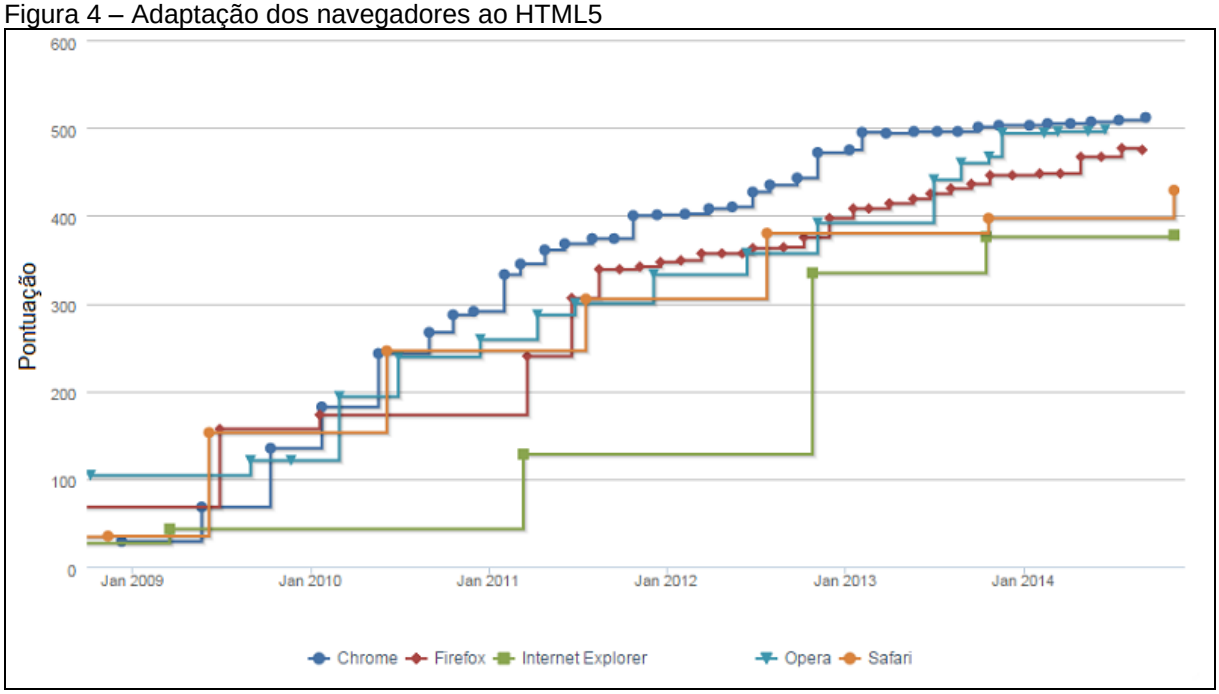
Cansados de esperar pelo W3C, em 2004, representantes da Mozilla, Apple e Opera, se uniram e fundaram o Web HyperText Application Technology WorkingGroup (WHATWG) e começaram a desenvolver uma alternativa ao XHTML2, que consistia na compatibilidade com sites antigos e na adição de novas tags ao HTML. Outra proposta feita por esse grupo era a especificação detalhada das tags já existentes no HTML para que pudessem ser implementadas com padrão pelos diferentes navegadores. Esse conjunto de inovações seria conhecido como HTML5. Observando as conquistas do WHATWG, o W3C com o apoio de Tim Berners-Lee, também começou a trabalhar com o HTML5 em outubro de 2006, descontinuando o XHTML2 no ano de 2009 (SCHROEDER, 2012).

Após alguns anos de evolução e amadurecimento, recentemente em 28 de outubro de 2014, o W3C inclui o HTML5 em sua lista de recomendações para o desenvolvimento de aplicações web. Com esta iniciativa pretende incentivar mais empresas e programadores independentes a utilizarem a tecnologia no desenvolvimento de suas aplicações, tanto para desktop, bem como em dispositivos móveis (ALVES, 2014).

O W3C demorou alguns anos para recomendar o uso do HTML5 devido a uma série de testes e melhorias. O grupo de desenvolvimento da entidade trabalhou com aproximadamente 60 empresas para aprimorar a linguagem, tendo solucionado cerca de quatro mil erros desde o começo dos trabalhos. Como resultado, o HTML5 se confirma como alternativa ideal para o desenvolvimento de conteúdos web (ALVES, 2014).

A figura 4 mostra uma pontuação que determina a adaptação dos navegadores em relação às funcionalidades disponibilizadas pelo HTML5. O site que realiza a avaliação está em constante evolução e atualiza os testes sempre que ocorre alguma mudança na especificação da linguagem. A pontuação máxima que pode ser atingida pelo *browser* é de 555 pontos. Para a avaliação são definidas pontuações para cada funcionalidade da linguagem, onde as mais complexas tem um peso maior em relação às tags mais comuns e simples (HTML5TEST... 2014).

Figura 4 – Adaptação dos navegadores ao HTML5



Fonte: HTML5TEST (2014).

A seguir são identificadas as principais mudanças e novas funcionalidades trazidas pelo HTML5. O texto busca destacar as novas tags e ferramentas que foram adicionadas a linguagem, realizando uma breve explicação de cada uma delas.

4.3 EVOLUÇÕES EM RELAÇÃO ÀS REVISÕES ANTERIORES

Uma das grandes deficiências da web atual é a falta de reprodução nativa de áudio e vídeo, sendo necessária a utilização de plug-ins, como o *Flash Player* para contornar o problema.

O HTML5 fornece o suporte à reprodução nativa de áudio e vídeo por meio das tags *audio* e *video,* respectivamente. A tabela 2 mostra os principais atributos para a tag *audio*, realizando uma explicação de cada atributo e identificando os valores admitidos em cada um deles.

Tabela 2 – Atributos, valores e funcionamento da tag *audio*

Atributo	Valor	Descrição
autoplay	autoplay	Determina que a reprodução do áudio inicie assim que o mesmo estiver sido carregado.
controls	controls	Especifica quais controles de áudio serão exibidos. Ex: play, pause, volume.
loop	loop	Determina que o áudio se repita novamente quando chegar ao fim.
muted	muted	Especifica que a saída de áudio deve ser silenciada.
preload	auto metadata none	Define se o áudio deve ser carregado junto com a página.
src	URL	Especifica o caminho do arquivo a ser reproduzido.

Fonte: W3SCHOOLS (2014b)

A tag *video* também possui alguns atributos principais, os quais podem ser observados na tabela 3.

Tabela 3 – Atributos, valores e funcionamento da tag *video*

Atributo	Valor	Descrição
autoplay	autoplay	Determina que a reprodução do vídeo inicie assim que o mesmo estiver sido carregado.
controls	controls	Especifica quais controles de vídeo serão exibidos. Ex: play, pause, volume.
height	pixels	Define a altura do player de vídeo.
loop	loop	Determina que o vídeo se repita novamente quando chegar ao fim.
muted	muted	Especifica que o vídeo será silenciado.
poster	URL	Especifica o caminho de uma imagem a ser exibida enquanto o vídeo é carregado.
preload	auto metadata	Especifica se o vídeo deve ser carregado junto com a página.
src	URL	Especifica o caminho do arquivo a ser reproduzido.
width	pixels	Define a largura do player de vídeo.

Fonte: W3SCHOOLS (2014c)

Na internet os usuários muitas vezes são ao mesmo tempo os produtores e consumidores dos conteúdos publicados. Hoje em dia, apesar da imensa quantidade de informações, muitas páginas não estão estruturadas da melhor forma, dificultando o trabalho dos buscadores de conteúdos. O HTML5 trás algumas tags como *article, header, footer,* as quais podem melhorar a indexação das páginas, pois com esses novos elementos é possível definir dentro da página qual a informação mais relevante, ou seja, qual informação deve ser mais visível aos buscadores (SILVA; SANTOS, 2013).

A capacidade de geolocalização também é um novo recurso trazido pelo HTML5 e consiste na ação de obter a localização geográfica do usuário. Existem três formas de conseguir essa informação (FERREIRA; EIS, 2010):

a) através do IP: é o método mais utilizado pelos navegadores web em computadores. Através de consultas e serviços, é possível identificar a possível localização de onde determinado IP está acessando. Pode determinar a cidade ou região da localização do usuário, não sendo muito preciso;
b) triangulação GPRS: dispositivos sem o recurso de GPS ou com o mesmo estando desativado, podem utilizar a triangulação das antenas GPRS próximas para obter a localização. Possui uma precisão maior que a localização por IP, podendo determinar o bairro em que o usuário está;
c) GPS: é o método mais preciso, utilizando o apoio de satélites. Pode ter uma margem de erro de cinco metros, sendo um ótimo resultado.

No caso da geolocalização, por questões de segurança, é um padrão do HTML5 que o usuário seja consultado, podendo optar por bloquear ou permitir a execução da funcionalidade. O desenvolvedor não pode definir qual a forma de geolocalização que será utilizada, apenas é possível ativar ou desativar o modo de alta precisão que, de acordo com os recursos e capacidades do dispositivo, irá escolher a forma mais otimizada de busca (FERREIRA; EIS, 2010).

Outro ponto importante que foi melhorado pelo HTML5 trata o armazenamento local de dados pelo próprio *browser*, anteriormente implementado através de *cookies,* os quais apresentam alguns problemas, principalmente a capacidade de armazenamento que gira em torno de 4KB cada, não sendo o suficiente para a web atual que trabalha com uma quantidade de informações muito grande (SERRA, 2011).

Para isto o HTML5 apresenta a *Web Storage,* uma API com capacidade de armazenamento local no *browser*, suportando de 5 a 10MB de informações. Apesar do servidor não ter acesso direto aos dados, é possível através de JavaScript que ele envie pedidos de leitura e escrita quando necessário. Existem duas formas de trabalhar com a *Web Storage* (SERRA, 2011):

a) *localStorage:* as informações podem ser acessadas novamente mesmo após o fechamento do *browser*;

b) *sessionStorage:* as informações persistem apenas enquanto a determinada sessão está aberta no *browser*.

O HTML5 também permiti que a aplicação execute em modo *off-line,* mesmo sem a conexão com a internet. Isso é possível devido à definição de um arquivo manifesto, cuja interpretação e execução são de responsabilidade do navegador. Nesse arquivo são definidas as páginas que o navegador do usuário deve armazenar em *cache*, além de configurações que permitem a execução *off-line* da aplicação web. Esse cacheamento é realizado automaticamente pelo *browser* no momento do primeiro acesso a página (LOBO FILHO, 2010).

A Web SQL Databases é uma API *client side* oferecida pelo HTML5 que fornece armazenamento e acesso a dados. É possível a criação de um banco de dados local, onde podem ser criadas tabelas e realizada a inserção, edição e exclusão de dados, além de possibilitar consultas SQL. Na criação do banco de dados é possível definir o seu tamanho, mas fica a cargo do navegador aprovar esse limite de armazenamento. Contudo é recomendado que seja criado um banco com o menor tamanho possível, apenas com a capacidade necessária para que o desempenho da aplicação não seja prejudicado em função das limitações do dispositivo do usuário (ELEMAR JUNIOR, 2010).

Até esse momento foram apresentadas algumas das principais funcionalidades que foram agregadas ao desenvolvimento web com a chegada do HTML5. A seguir serão abordados com mais detalhes os elementos *canvas* e o *Scalable Vectorial Graphics (SVG),* ambos com muita importância e capacidade para o desenvolvimento visual da web, manipulando imagens e criando animações.

4.4 CANVAS E SVG

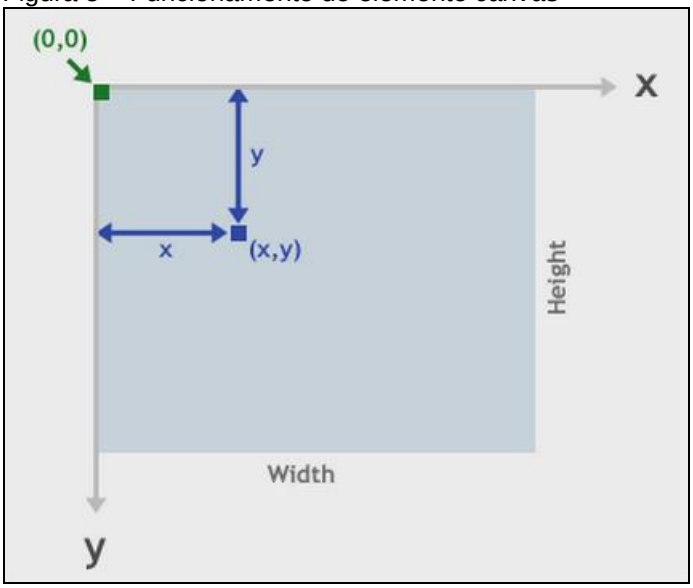
Além de novas funcionalidades e recursos de multimídia, uma das características mais esperadas do HTML5, é o elemento para a criação e manipulação de imagens, o *canvas.* Para que seja possível trabalhar com o *canvas,* são necessários o CSS e principalmente o JavaScript, tendo estes, um papel muito importante no desenvolvimento web com HTML5.

O elemento *canvas* é uma API que possibilita a definição de uma área onde podem ser renderizados elementos visuais em tempo de execução, permitindo

por meio de JavaScript, que esses elementos sejam manipulados, utilizando o conceito de bitmap, pixel a pixel (SEBBEN; GUEDES, 2010).

A figura 5 mostra o conceito de coordenadas (x,y) que é utilizado pelo elemento *canvas.*

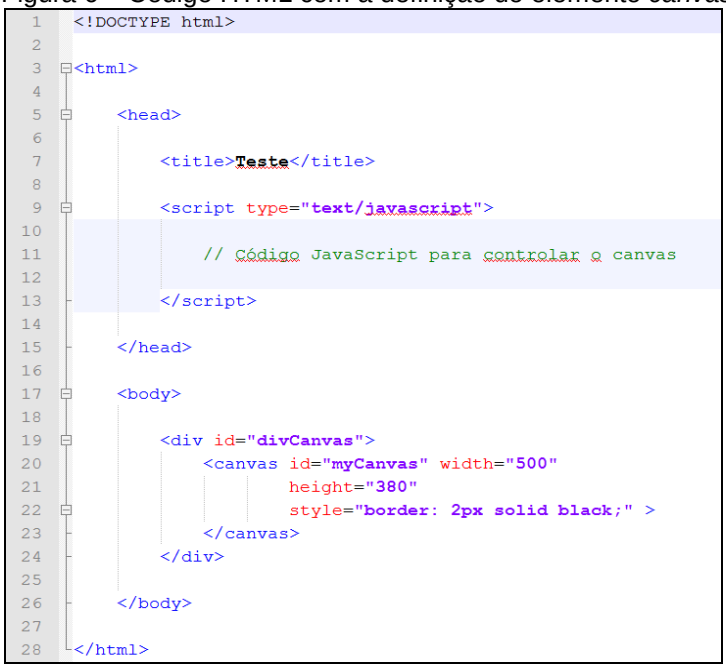
Figura 5 – Funcionamento do elemento *canvas*



Fonte: Alvarez (2010).

Para a utilização do elemento *canvas,* é necessário que ele seja definido na estrutura da página HTML, como pode ser observado na figura 6, onde é declarado o *canvas* com o identificador myCanvas, sendo adicionadas propriedades de largura, altura e borda para o elemento.

Figura 6 – Código HTML com a definição do elemento *canvas*



```
<!DOCTYPE html>

<html>

    <head>

        <title>Teste</title>

        <script type="text/javascript">

            // Código JavaScript para controlar o canvas

        </script>

    </head>

    <body>

        <div id="divCanvas">
            <canvas id="myCanvas" width="500"
                    height="380"
                    style="border: 2px solid black;" >
            </canvas>
        </div>

    </body>

</html>
```

Fonte: Do autor.

Estando o *canvas* definido na página HTML, através de JavaScript, é possível manipular o seu contexto, permitindo que sejam criadas e manipuladas formas e imagens. Na figura 7 está representado um trecho de JavaScript que obtém o contexto 2D do *canvas* e desenha na tela uma um retângulo na vertical que está posicionado sobre uma linha na horizontal.

Figura 7 – Código JavaScript que manipula o *canvas*

```
<script type="text/javascript">

    // Código JavaScript para controlar o canvas

    var can = function(){

        var canvas;
        var context;

        return {

            init: function(){

                canvas = document.getElementById("myCanvas");
                context = canvas.getContext("2d");

                context.fillStyle = "#98713D";
                context.fillRect(100,85,30,220);

                context.fillStyle = "black";
                context.fillRect(10,306,215,2);
            }
        };

    }();

    window.onload = can.init;

</script>
```

Fonte: Do autor.

O HTML5 também dá suporte ao *Scalable Vectorial Graphics (SVG),* sendo uma tecnologia aberta e recomendada pelo W3C. O *SVG* é uma grande aposta para revolucionar o desenvolvimento de interfaces para a internet (SEBBEN; GUEDES, 2010).

O *SVG* é uma linguagem para criação de gráficos vetoriais baseada em XML e acessível via Document Object Model (DOM). Ele possibilita a geração de gráficos escaláveis que não perdem resolução ou são danificados quando a tela é redimensionada (FERREIRA; EIS, 2010).

O surgimento do *canvas* tem impulsionado o mercado de jogos, pois é possível a confecção de games com a utilização de linguagens abertas e livres de licenças proprietárias. A utilização do JavaScript que controla o elemento *canvas* que está embutido no HTML, proporciona o desenvolvimento de aplicações que podem ser executadas diretamente pelo navegador de maneira leve e prática (HENRIQUES et al, 2011).

5 TRABALHOS CORRELATOS

Neste capítulo são apresentados outros trabalhos que realizaram estudos semelhantes a este trabalho que está sendo desenvolvido, apresentando dados que identificam a origem e os autores de cada trabalho, além de identificar o que é o trabalho, qual o principal objetivo e os assuntos abordados. Também são apresentados aspectos de metodologia, ou seja, como o trabalho foi desenvolvido, quais ferramentas utilizadas, além de apresentar uma breve descrição dos resultados obtidos.

5.1 HTML5: UMA ANÁLISE DE COMPATIBILIDADE DOS ELEMENTOS ÁUDIO, VÍDEO E CANVAS COM OS PRINCIPAIS NAVEGADORES

Este artigo foi desenvolvido por Cleberson Paulo de Andrade Silva e Marilde Terezinha Prado Santos, sendo publicado no ano de 2013 pela Revista TIS, Tecnologias, Infraestrutura e Software. O trabalho consiste na apresentação das novas funcionalidades do HTML5, além da análise da compatibilidade das novas funcionalidades trazidas por essa linguagem nos diferentes navegadores, com destaque para reprodução de áudio, vídeo e animações com o uso da tag *canvas* (SILVA; SANTOS, 2013).

Para a realização do trabalho foram consultados artigos científicos, sites e livros que possuem conteúdo semelhante ao que foi proposto na pesquisa, além da especificação do HTML5 disponibilizada pelo W3C. Para a realização das análises de compatibilidade, foi desenvolvido o projeto E-Delivery, que consiste em uma aplicação web com o objetivo de facilitar as relações entre os estabelecimentos comerciais e os consumidores, sendo um meio alternativo ao uso do telefone para a realização da compra de produtos e serviços. Junto a esse projeto foram desenvolvidos protótipos que utilizaram os componentes de áudio, vídeo e o *canvas,* permitindo que as comparações nos diferentes navegadores fossem realizadas (SILVA; SANTOS, 2013).

Como resultado deste trabalho foi possível concluir que apesar de o HTML5 ser uma inovação recente e estar em fase de amadurecimento, muitos componentes já são suportados pelos navegadores disponíveis no mercado, sendo que o Safari, Opera e Google Chrome apresentaram uma compatibilidade maior.

Dos componentes avaliados, a reprodução de vídeo apresentou os maiores problemas entre os navegadores. O Mozzila Firefox teve um desempenho intermediário, suportando todos os componentes, tendo problemas apenas com alguns formatos de áudio e vídeo. Já o Internet Explorer teve a pior compatibilidade com os elementos avaliados, inclusive não suportando a tag *canvas* (SILVA; SANTOS, 2013).

5.2 DESENVOLVIMENTO DE JOGO EDUCACIONAL SOBRE ECOTOXICOLOGIA UTILIZANDO HTML5

Este artigo foi elaborado por Diogo Moreira Bispo, Márcio Silvatti Zabeu, Gisela Aragão Umbuzeiro e Marcos Augusto Francisco Borges, sendo publicado no ano de 2012 pela Revista Brasileira de Informática na Educação. (BISPO et al, 2012).

O trabalho consiste no desenvolvimento de um jogo educacional chamado *Daphnia World*, para auxiliar os alunos de Saneamento Ambiental da Faculdade de Tecnologia da Unicamp no ensino de ecotoxicologia. Para o desenvolvimento do trabalho foram realizados estudos com o intuito de compreender os termos *Daphnia* e *Toxicologia Ambiental,* além dos jogos educativos, que também foram alvo de estudo (BISPO et al, 2012).

O jogo foi criado para ser jogado na internet com o uso de um navegador, explorando a capacidade de fácil distribuição e acessibilidade, sendo acessível a diversos dispositivos que possuam um navegador web e conexão com a internet, como computadores e dispositivos móveis (BISPO et al., 2012).

O jogo foi desenvolvido utilizando-se principalmente o HTML5, combinado com outras tecnologias como Cascading Style Sheets (CSS), JavaScript e Document Object Model (DOM). Para a edição de imagens foram utilizados dois softwares livres, o Inkscape, um editor gráfico vetorial, com capacidade similar ao Illustrator e CorelDraw. O outro software se chama Gimp, já conhecido por fazer parte de algumas distribuições do Linux, com tarefas de retoques e composição de imagens (BISPO et al, 2012).

Entre os resultados e conclusões do trabalho, foi destacada a capacidade de desenvolvimento oferecida pelo HTML5, mostrando-se adequado para o desenvolvimento de jogos educativos. Foi destacado o uso do elemento *canvas,* que

possui uma API para o desenvolvimento de animações em 2D, e também a linguagem JavaScript. O jogo desenvolvido atendeu aos objetivos que haviam sido propostos no trabalho. O jogo previa o uso principalmente por jovens, no entanto as crianças também acolheram de forma muito produtiva esta atividade (BISPO et al, 2012).

5.3 AVALIAÇÃO E COMPARAÇÃO DE RECURSOS UTILIZANDO TECNOLOGIAS HTML5 E FLASH PARA CRIAÇÃO DE APLICAÇÕES WEB

Este trabalho de conclusão de curso foi desenvolvido pelo acadêmico Tiago de Carvalho, sendo apresentado no ano de 2013 para obtenção do grau de bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense (UNESC) (CARVALHO, 2013).

O trabalho consiste na avaliação e comparação de alguns recursos das tecnologias HTML5 e Flash, buscando auxiliar os desenvolvedores na tomada de decisão em relação a qual tecnologia utilizar para o desenvolvimento de aplicações web. Foram estudadas as duas tecnologias citadas acima a fim de adquirir o conhecimento necessário para a realização do trabalho, sendo que os principais estudos se concentraram na reprodução de áudio e vídeo, além do consumo de processamento e memória (CARVALHO, 2013).

Para possibilitar a comparação dos recursos, foram desenvolvidos pequenos protótipos nas duas linguagens, realizando a implementação de forma mais semelhante possível. Também foram definidos os métodos de comparação entre os recursos, baseando-se principalmente em recomendações da International Telecommunication Union (ITU), e preparado o material de áudio e vídeo com as mesmas propriedades para o HTML5 e para o Flash. Os testes foram realizados utilizando-se os navegadores Internet Explorer, Mozilla Firefox e Google Chrome (CARVALHO, 2013).

Em relação aos resultados obtidos foi possível constatar que o uso do HTML5 é muito superior em relação ao consumo de recursos de hardware, sendo um aspecto favorável, visto a utilização da web por aparelhos como *tablets* e celulares. Quanto à qualidade de reprodução de áudio e vídeo, as duas tecnologias apresentaram um bom desempenho, com algumas superioridades em formatos de áudio e vídeo, e entre diferentes navegadores, mas nada que pode ser considerado

um problema na escolha de uma ou outra linguagem. Foi destacado que em caso de aplicações projetadas para computadores *desktop* o Flash seria mais aconselhado pelo fato de estar presente em boa parte dos computadores existentes (CARVALHO, 2013).

5.4 ESTUDO DE VIABILIDADE DO HTML5 PARA DESENVOLVIMENTO WEB

Este trabalho de conclusão de curso foi desenvolvido pelo acadêmico Daniel Fuverki Hey, sendo apresentado no ano de 2010 para obtenção do grau de bacharel no curso de Ciência da Computação da Universidade Estadual de Maringá (HEY, 2010).

O trabalho buscou compreender os novos recursos presentes na versão cinco do HTML, como elementos estruturais de marcação, apresentação de conteúdo e multimídia, baseando-se na sua mais recente especificação (HEY, 2010).

Para o desenvolvimento do trabalho, foram realizadas pesquisas em livros e na internet, a fim de obter o conhecimento a cerca dos temas a serem abordados, tomando como base outros trabalhos semelhantes que analisaram a compatibilidade dos novos elementos do HTML5 (HEY, 2010).

Foi realizada uma análise de compatibilidade dos novos recursos trazidos pelo HTML5 nos diferentes navegadores. Para isto foram estudados os principais motores de renderização e motores de execução de JavaScript dos browsers. Entre os principais motores de renderização estudados, destacam-se o Gecko, Presto, Webkit e Trident. O motores de execução JavaScript estudados foram o Spidermonkey e o V8 (HEY, 2010).

Foram estudadas outras tecnologias para a implementação de aplicações web complexas e interativas, com destaque para o Flash, Flex, JavaFX e Silverlight. Este estudo buscou compreender o funcionamento, as vantagens e desvantagens dessas tecnologias em relação uma as outras e com o HTML5 (HEY, 2010).

O trabalho concluiu que de acordo com as necessidades de uma determinada aplicação web, pode ser utilizada uma ou outra tecnologia, depende do que deseja fazer. Apesar de o HTML5 estar em evolução, apresentando problemas de compatibilidade em alguns elementos, já pode ser utilizado para o desenvolvimento web, ou seja, é viável a sua utilização (HEY, 2010).

5.5 MOTOR PARA JOGOS 2D UTILIZANDO HTML5

Este trabalho de conclusão de curso foi elaborado pelo acadêmico Marcos Harbs para obtenção do grau de bacharel no curso de Ciência da Computação, sendo submetido no ano de 2013 na Universidade Regional de Blumenau (HARBS, 2013).

O trabalho consiste no desenvolvimento de um motor para jogos em duas dimensões, onde o elemento *canvas* do HTML5 e o JavaScript são amplamente utilizados. Também foi desenvolvida uma ferramenta web que utiliza o motor criado para a criação e edição de jogos de forma visual, além de serem realizados testes de desempenho e memória nos principais navegadores de internet do mercado (HARBS, 2013).

O desenvolvimento do motor de jogos baseou-se na linguagem JavaScript e em elementos do HTML5, principalmente o *canvas*. Para o desenvolvimento do motor de jogos foi utilizado o editor de texto Sublime Text 2.0.2, o qual permite a manipulação de HTML5 e JavaScript. Para facilitar o trabalho com a detecção de colisões de objetos, foi acoplado junto ao motor de jogos, o motor de simulações físicas conhecido como Box2DJS. Também foram desenvolvidas interações com o Kinect por meio da biblioteca Zigfu e com joysticks utilizando a Gamepad API (HARBS, 2013).

Para o desenvolvimento do editor de jogos foi utilizada a linguagem Java, juntamente com o framework Java Server Faces 2.2 (JSF). Para a customização e adição de componentes visuais, foi utilizada a biblioteca do Primefaces 3.5. Como servidor de aplicação, adotou-se o JBoss 6.1.0 e na implementação foi utilizado o Eclipse IDE for Java EE Developers, já possuindo alguns plug-ins para desenvolvimento com JSF (HARBS, 2013).

Os navegadores utilizados durante o desenvolvimento foram o Google Chrome 32.0.1678.0 dev-m Aura, Internet Explorer 10.0.9200.16721, Opera 17.0 e Mozilla Firefox 25.0. A máquina utilizada foi um computador desktop com processador AMD FX(tm)-6200 Six-Core, memória de 8 GB DDR3 e uma placa de vídeo AMD Radeon HD 6850 (HARBS, 2013).

Após o desenvolvimento do motor de jogos e do editor, foram criados dois jogos simples, no intuito de testar as ferramentas desenvolvidas. Os jogos foram o Tangram e o Space Invaders. Os testes buscaram avaliar o desempenho, onde foi

criada uma cena de jogo com 200 objetos rígidos que reagem a colisões, coletando-se a média de quadros por segundo da execução da cena (HARBS, 2013).

Para realizar a análise de gargalos de desempenho utilizou-se a ferramenta Profiler, recurso disponível no Google Developer Tools. Ela permite analisar quais métodos estão exigindo maior tempo de processamento, o que auxilia na identificação de possíveis melhorias (HARBS, 2013).

Na avaliação do consumo de memória, também foi utilizada a ferramenta Profiler do Google Chrome, onde foi possível identificar que o consumo de memória aumenta à medida que são colocados mais corpos rígidos na cena do jogo reagindo a colisões (HARBS, 2013).

Por fim, pode-se concluir que o HTML5 é uma tecnologia recente e continua sendo aperfeiçoada, mas através do elemento *canvas*, juntamente com a linguagem JavaScript, é possível o desenvolvimento de jogos e aplicações interativas. Durante os testes de desempenho, constatou-se que o navegador Internet Explorer teve o melhor desempenho, e houve uma semelhança entre o desempenho do Google Chrome e do Opera, isso porque, ambos utilizam o motor de execução JavaScript desenvolvido pela Google (HARBS, 2013).

6 DESENVOLVIMENTO E COMPARAÇÃO DO APLICATIVO EM DIFERENTES NAVEGADORES

Este capítulo realiza a apresentação do que foi desenvolvido no trabalho. Serão detalhados os passos, e conhecimentos necessários, além dos aspectos de metodologia, mostrando como o trabalho foi desenvolvido. Por fim, é feita a análise e organização dos dados que foram obtidos como resultado.

O trabalho consiste no desenvolvimento e comparação de um aplicativo que será utilizado para auxiliar no processo de ensino e aprendizagem de sistemas estruturais, uma área relacionada à engenharia. O aplicativo foi desenvolvido em parceria com o curso de Engenharia Civil da UNESC, onde houve a orientação de um profissional da área.

O objetivo principal do trabalho é realizar uma comparação do aplicativo desenvolvido entre os principais navegadores de internet, analisando aspectos de desempenho e compatibilidade. Na comparação foi analisada a execução do código utilizado para a construção do aplicativo, sendo possível avaliar qual o desempenho de cada *browser.*

Após a conclusão deste trabalho, pretende-se disponibilizar o aplicativo desenvolvido para que outros usuários possam se beneficiar com a sua utilização, além de poderem colaborar com seu desenvolvimento.

6.1 METODOLOGIA

Para a realização do trabalho, com intuito de alcançar os objetivos propostos, foi necessário o cumprimento de algumas etapas e a compreensão de alguns conceitos, tecnologias para o desenvolvimento e formas de comparação de aplicações web.

Primeiramente foi realizado um levantamento bibliográfico buscando aprofundar o conhecimento sobre os sistemas estruturais e os seus elementos, bem como os conceitos de compressão, tração e flexão. Estes conhecimentos foram necessários, pois o aplicativo implementado trata desses assuntos que são da área da engenharia.

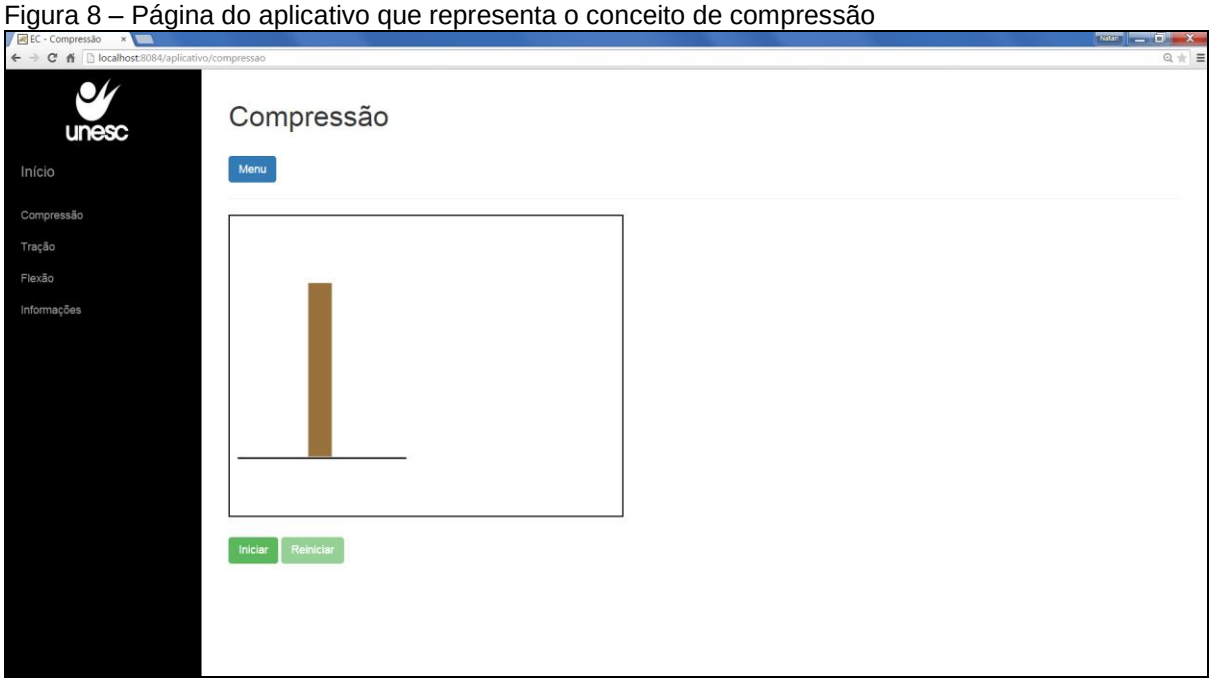
Também foi pesquisado sobre a estrutura da web, bem como os navegadores mais utilizados atualmente, apresentando características e informações

de cada um deles. A linguagem HTML5 também foi estudada, onde foram apresentadas as principais novidades da nova versão do HTML, realizando uma breve descrição da utilidade e funcionamento de cada uma delas.

Para o desenvolvimento do aplicativo foram utilizadas tecnologias *client side*, as quais são executadas pelo próprio navegador, envolvendo HTML5, CSS e JavaScript. Na definição do leiaute do aplicativo, foi utilizado um template do Twitter Bootstrap, chamado *Simple Sidebar.* Para a utilização deste template foi necessário entender a sua estrutura para que as páginas pudessem ser criadas e adaptadas de acordo com a necessidade.

O principal elemento utilizado foi o *canvas*, permitindo a renderização de imagens e formas, além de realizar animações em tempo de execução. A seguir será detalhada a estrutura e funcionamento de uma das páginas do aplicativo, possibilitando o entendimento da implementação com o *canvas*. A figura 8 mostra a página que aborda o conceito de compressão em seu status inicial, recém-carregada no navegador.

Figura 8 – Página do aplicativo que representa o conceito de compressão



Fonte: Do autor.

O código demonstrado a seguir foi adaptado para que sejam apresentadas apenas as informações mais importantes em relação ao desenvolvimento da página. A figura 9 mostra a estrutura HTML com a definição do elemento *canvas* no corpo da página, onde são definidas a largura, altura e

propriedades de borda para o elemento. Também é definido por meio do atributo *id,* o identificador *myCanvas* que será utilizado posteriormente para acessar as propriedades do *canvas*.

Figura 9 – Estrutura HTML da página que aborda compressão

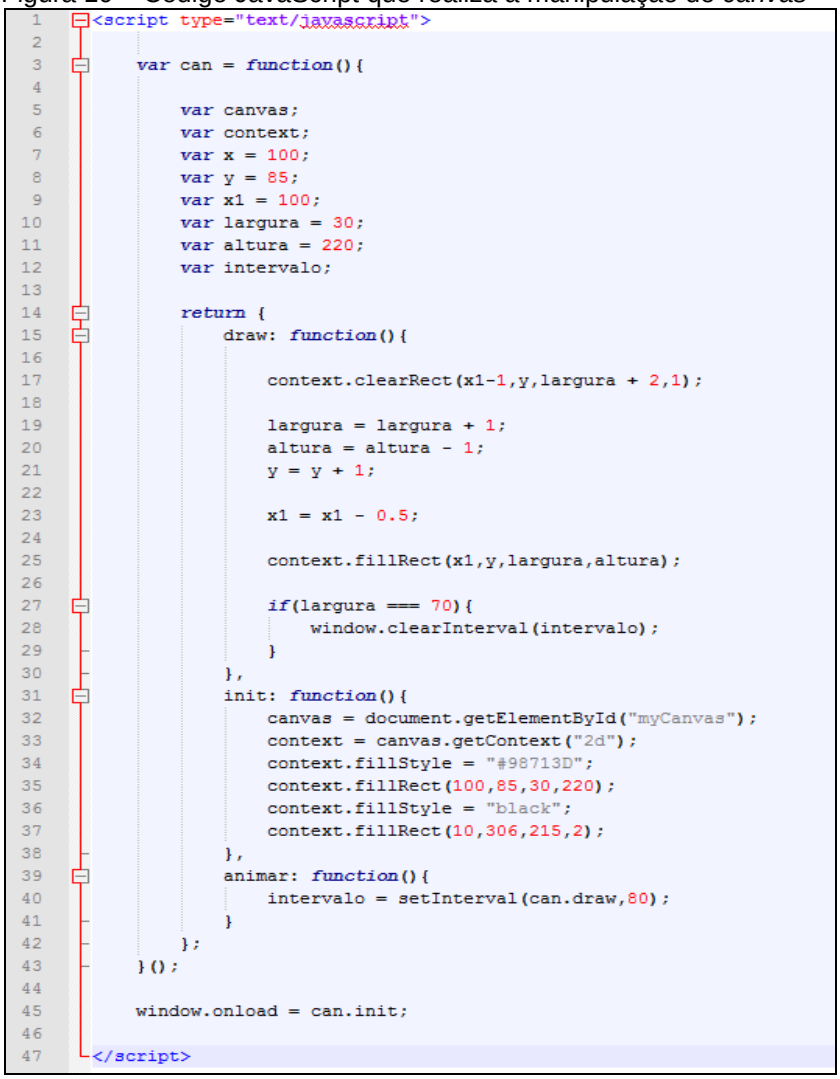
```
<!DOCTYPE html>

<html>

    <head>

        <title>EC - Compressão</title>

        <script type="text/javascript">

            // Código JavaScript para controlar o canvas

        </script>

    </head>

    <body>

        <div id="divCanvas">
            <canvas id="myCanvas" width="500"
                    height="380"
                    style="border: 2px solid black;" >
            </canvas>
        </div>

    </body>

</html>
```

Fonte: Do autor.

A figura 10 mostra o código JavaScript que controla o elemento *canvas.* No carregamento da página é executado o bloco *init* da função JavaScript, onde é obtido o elemento canvas contido no corpo da página e retornado o seu contexto 2D para que seja possível acessar as suas funções. Posteriormente, por meio da função *fillRect,* é desenhado o retângulo e a linha, demonstrados na página (figura 6). A função *fillRect* recebe os argumentos x e y que definem o posicionamento de início do desenho, além da altura e largura da forma a ser desenhada.

Ao clicar no botão *Iniciar,* demonstrado na figura 6, a animação é iniciada por meio do bloco *animar,* presente no script da figura 8. Essa parte do script chama a função *draw* em intervalos de 80 milissegundos, sendo reformulado o desenho do *canvas,* onde as variáveis de altura, largura, x e y são alteradas, fazendo com que o desenho seja refeito a cada chamada da função *draw*. Para encerrar a animação, é verificado o valor da variável largura, que determina a limpeza do intervalo que está sendo executado, finalizando a animação.

Figura 10 – Código JavaScript que realiza a manipulação do *canvas*



```
<script type="text/javascript">

    var can = function(){

        var canvas;
        var context;
        var x = 100;
        var y = 85;
        var x1 = 100;
        var largura = 30;
        var altura = 220;
        var intervalo;

        return {
            draw: function(){

                context.clearRect(x1-1,y,largura + 2,1);

                largura = largura + 1;
                altura = altura - 1;
                y = y + 1;

                x1 = x1 - 0.5;

                context.fillRect(x1,y,largura,altura);

                if(largura === 70){
                    window.clearInterval(intervalo);
                }
            },
            init: function(){
                canvas = document.getElementById("myCanvas");
                context = canvas.getContext("2d");
                context.fillStyle = "#98713D";
                context.fillRect(100,85,30,220);
                context.fillStyle = "black";
                context.fillRect(10,306,215,2);
            },
            animar: function(){
                intervalo = setInterval(can.draw,80);
            }
        };
    }();

    window.onload = can.init;

</script>
```

Fonte: Do autor.

Foi tomada como exemplo para a explicação uma das páginas desenvolvidas. No entanto, as outras páginas que realizam animações seguem a mesma estrutura e funcionamento, apenas sendo utilizadas outras funções do HTML5 *canvas* como a função *bezierCurveTo*, que possibilita o desenho de formas em curva.

Após o desenvolvimento do aplicativo, a fim de alcançar o objetivo computacional do trabalho, foi iniciada a comparação entre os navegadores. Nesta etapa foram pesquisados softwares e formas de comparação que poderiam ser utilizados.

Para a realização da comparação foram adotados os navegadores Google Chrome, Mozilla Firefox e Internet Explorer, sendo estes os mais utilizados atualmente, como pode ser visto ao longo da pesquisa bibliográfica. Foram utilizadas

as versões mais atuais destes navegadores, colocando-os nas mesmas condições de uso, sem nenhum tipo de bloqueio durante os testes, como execução de códigos JavaScript, por exemplo. As versões dos navegadores que foram utilizadas são apresentadas a seguir:

a) Google Chrome: 42.0.2311;

b) Mozilla Firefox: 37.0;

c) Internet Explorer 11: 11.0.9600.

Para realizar a comparação foi utilizado o software *jsPerf*, um sistema que possibilita a criação de cenários de teste, comparando o desempenho de trechos de código por meio de *benckmarks*. O *jsPerf* também permite o versionamento de cada teste elaborado, possibilitando a edição e execução de cada projeto em revisões diferentes (SMUS, 2013).

O software executa o código informado, o máximo de vezes possível durante um período de tempo. Ao final da execução, são geradas informações de desempenho. Estas informações são referentes à quantidade de operações por segundo que foram alcançadas com a execução do script, onde quanto mais operações por segundo, melhor o desempenho. Este desempenho pode ser avaliado entre os navegadores, pois o software permite que um mesmo teste seja executado em diferentes *browsers* (SMUS, 2013).

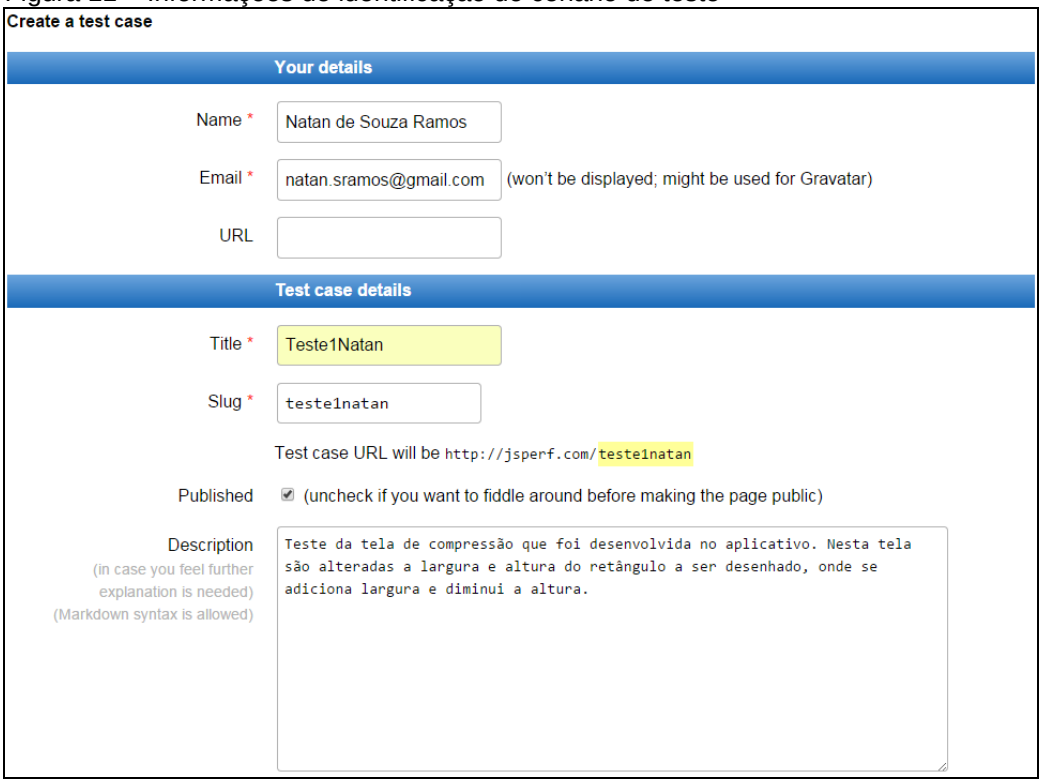
O *JsPerf* foi desenvolvido por John-David Dalton, um programador da Microsoft especializado em análise de desempenho de código. John-David Dalton tem participação na produção e manutenção do Internet Explorer, além de colaborar com outros projetos de código aberto, onde procura auxiliar os desenvolvedores para que produzam scripts mais otimizados e bem elaborados. A seguir serão detalhados os passos para criação e execução dos cenários de teste por meio do *jsPerf*.

Para a criação dos cenários de testes na ferramenta *jsPerf*, é necessário o preenchimento de algumas informações. Inicialmente devem ser fornecidos os dados do criador do cenário de teste, como nome e e-mail. Posteriormente deve-se definir o título e uma breve descrição, o que facilita a identificação do cenário de teste.

No momento em que é informado o título do cenário de teste, o campo *Slug* é alimentado automaticamente, onde é montada a URL que armazenará o teste. A URL é formada com o endereço jsPerf.com mais uma barra seguida do

conteúdo do campo *Slug.* A figura 11 apresenta o formulário do *jsPerf* para o preenchimento dos campos de identificação descritos acima.

Figura 11 – Informações de identificação do cenário de teste



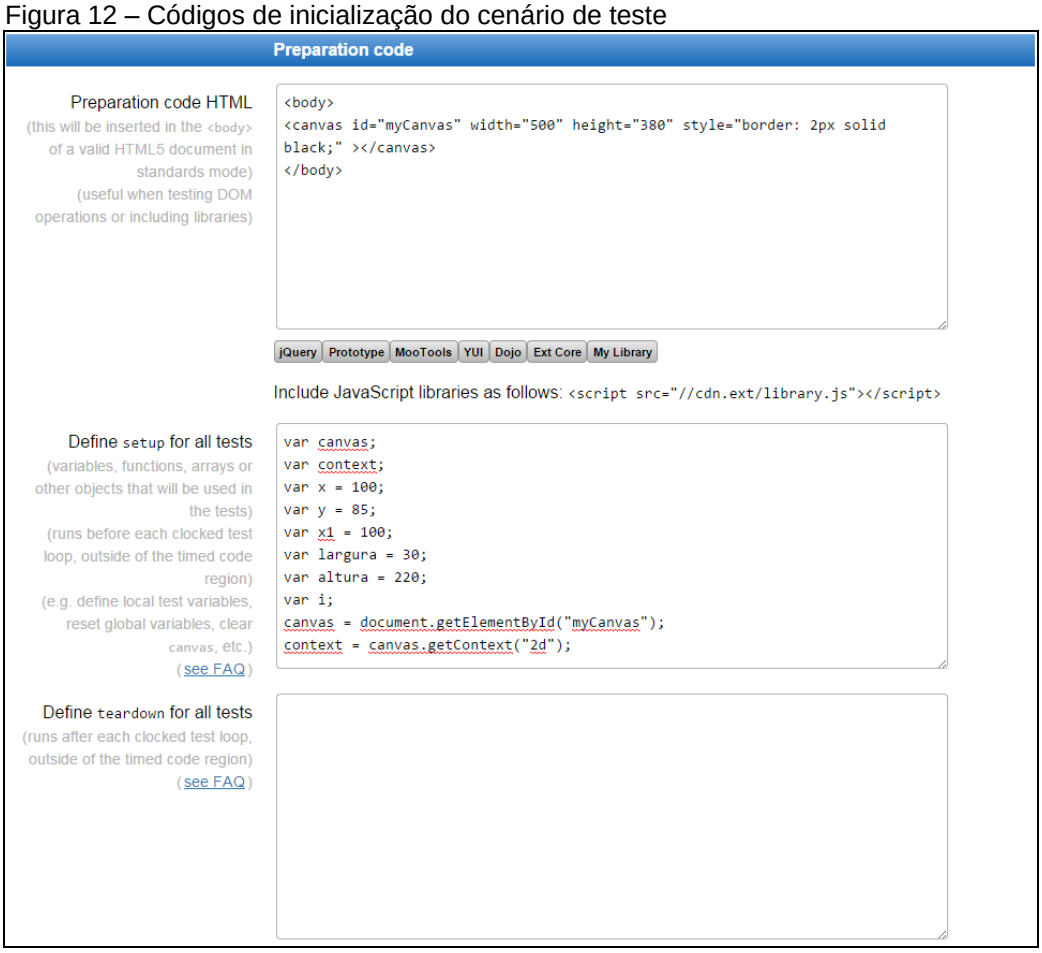
Fonte: Do autor.

Após a definição das informações de identificação do cenário de teste, devem ser informados os códigos que realizam a inicialização do teste. Na figura 12 são demonstrados os campos que possibilitam o preenchimento desses scripts.

No campo *Preparation code HTML,* deve ser informado o código HTML presente na estrutura da página, sendo necessário apenas o trecho contido no corpo da página, o qual se encontra dentro da tag *body.* Esse campo suporta elementos do HTML5, como o *canvas,* por exemplo, os quais podem ser manipulados ao longo da execução do teste.

O campo *Define setup for all tests* tem a função de armazenar a declaração de variáveis, funções ou objetos que serão utilizadas durante a execução. O último campo presente na preparação inicial dos scripts é o *Define teardown for all tests,* sendo executado após o término do teste. Entre outras funções, esse campo permite a reinicialização de variáveis ou o encerramento de uma conexão aberta durante a execução do cenário de teste. No entanto, esse campo geralmente não é utilizado, pois na maioria dos casos não é necessário.

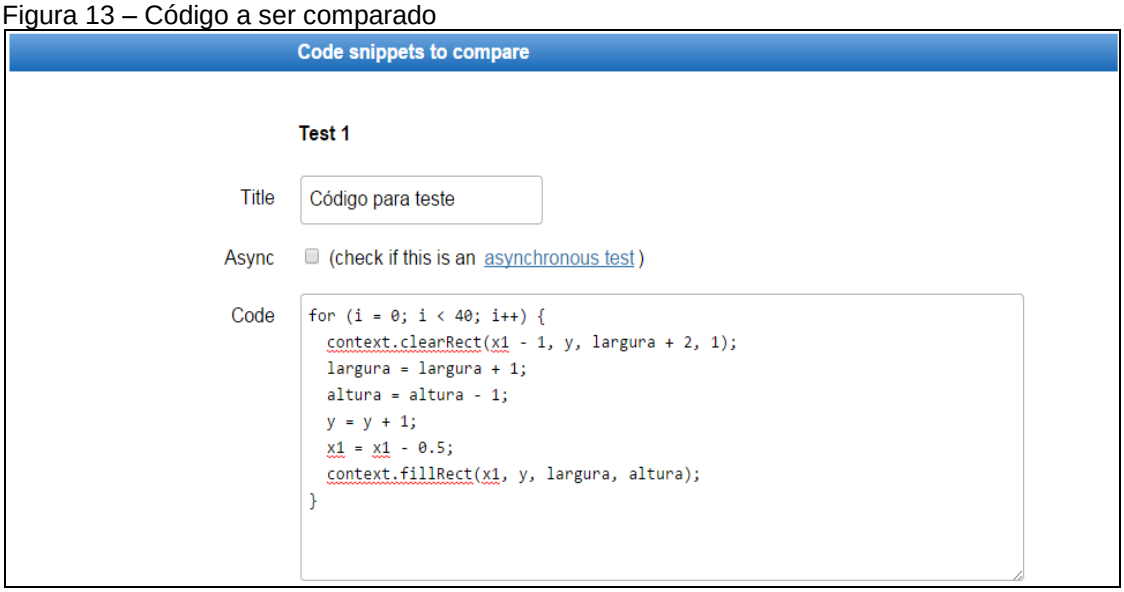
Figura 12 – Códigos de inicialização do cenário de teste



Fonte: Do autor.

Para concluir a montagem do cenário de teste, deve ser fornecido o código a ser comparado, que irá executar e manipular a página HTML ao longo da execução. Esse código altera as variáveis e elementos definidos nos campos de inicialização, realizando alterações na página. Na figura 13 pode ser observado o preenchimento deste campo.

Figura 13 – Código a ser comparado

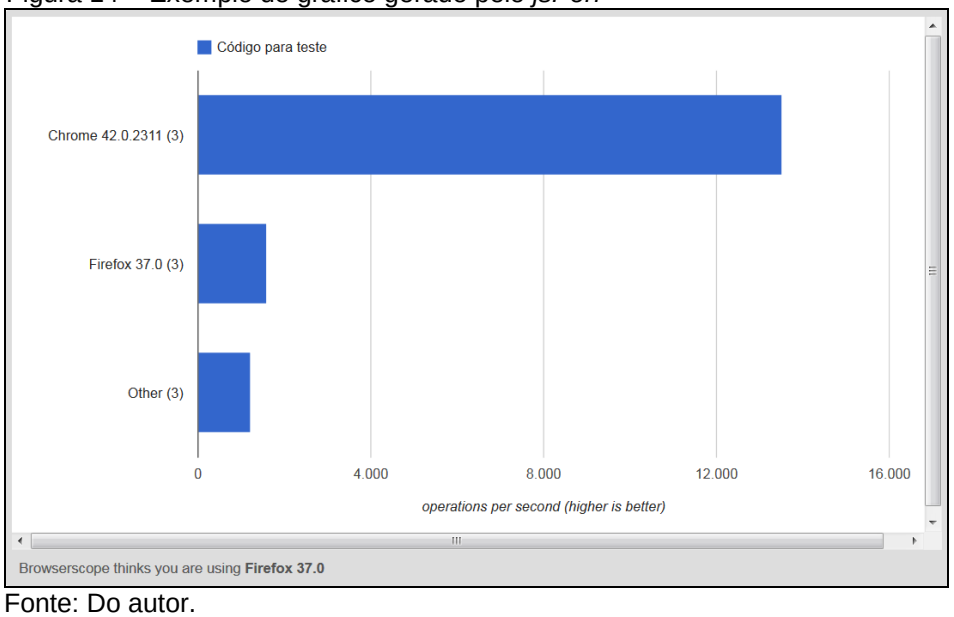


Fonte: Do autor.

Estando o cenário de teste totalmente elaborado, basta iniciar a sua execução por meio do botão *Run tests*. Após o término da execução são geradas de forma gráfica, a quantidade de operações por segundo obtidas com a execução do script. O sistema permite a visualização do gráfico de diversas formas, como barras, colunas, linhas, gráficos de pizza e em tabela, além de acumular as informações de cada navegador à medida que o teste vai sendo executado.

A figura 14 mostra o exemplo de um gráfico gerado pela ferramenta ao final da execução de um teste que foi executado no Google Chrome, Mozilla Firefox e Internet Explorer.

Figura 14 – Exemplo de gráfico gerado pelo *jsPerf*



Fonte: Do autor.

Estando compreendido o funcionamento do software *jsPerf*, foram montados três cenários de teste, contemplando as páginas do aplicativo desenvolvido que realizam animações. Esses cenários de teste serão detalhados a seguir, sendo apresentados os códigos utilizados na sua elaboração.

6.1.1 Cenário de teste 1

Este cenário de teste foi montado com base na tela do aplicativo que trabalha com o conceito de compressão. A seguir serão apresentados os scripts que foram utilizados na construção do teste. Para cada trecho de código demonstrado, será informada qual a sua função no cenário de teste, ou seja, em qual dos campos do *jsPerf,* este código foi inserido.

A figura 15 apresenta o código HTML utilizado para definir o elemento *canvas.* Esse trecho de código foi inserido no campo *Preparation Code HTML* presente na ferramenta.

Figura 15 – Código HTML utilizado no cenário 1

```
1  <body>
2      <canvas id="myCanvas"
3              width="500"
4              height="380"
5              style="border: 2px solid black;" >
6      </canvas>
7  </body>
```

Fonte: Do autor.

A seguir a figura 16 mostra o código de inicialização que foi informado no campo *Define setup for all tests*. Este script declara as variáveis utilizadas ao longo do teste e faz a inicialização do elemento *canvas.*

Figura 16 – Código de inicialização utilizado no cenário 1

```
1  var canvas;
2  var context;
3  var x = 100;
4  var y = 85;
5  var x1 = 100;
6  var largura = 30;
7  var altura = 220;
8  var i;
9
10 canvas = document.getElementById("myCanvas");
11 context = canvas.getContext("2d");
```

Fonte: Do autor.

Por fim a figura 17 mostra o código que realiza as manipulações na página HTML. Este trecho de código foi informado no campo que armazena o script que é realmente comparado durante a execução do cenário de teste.

Figura 17 – Código utilizado na comparação do cenário 1

```
1  for (i = 0; i < 40; i++) {
2
3    context.clearRect(x1 - 1, y, largura + 2, 1);
4
5    largura = largura + 1;
6    altura = altura - 1;
7    y = y + 1;
8
9    x1 = x1 - 0.5;
10
11   context.fillRect(x1, y, largura, altura);
12
13 }
```

Fonte: Do autor.

6.1.2 Cenário de teste 2

O segundo cenário de teste foi montado a partir da tela que realiza a animação abordando o conceito de tração. A seguir serão disponibilizados os scripts utilizados para a elaboração deste cenário de teste, sendo descrita a função e utilização de cada um deles no *jsPerf.*

O trecho de código demonstrado na figura 18 apresenta a definição do HTML utilizado no segundo cenário de teste. Esse código é responsável pela declaração do elemento *canvas,* e foi adicionado ao campo *Preparation Code HTML* do *jsPerf.*

Figura 18 – Código HTML utilizado no cenário 2

```
<body>
    <canvas id="myCanvas"
            width="500"
            height="380"
            style="border: 2px solid black;" >
    </canvas>
</body>
```

Fonte: Do autor.

O script da figura 19 foi utilizado no campo *Define setup for all testes,* o qual tem a função de declarar as variáveis que serão utilizadas durante a execução do teste, bem como a obtenção do elemento *canvas* do corpo da página.

Figura 19 – Código de inicialização utilizado no cenário 2

```
var canvas;
var context;
var x = 100;
var y = 150;
var x1 = 100;
var largura = 50;
var altura = 180;
var xaux = 150;
var i;

canvas = document.getElementById("myCanvas");
context = canvas.getContext("2d");
```

Fonte: Do autor.

O código utilizado na comparação e que realiza a alteração na página HTML para o cenário de teste 2 pode ser observado na figura 20.

Figura 20 – Código utilizado na comparação do cenário 2

```
for (i = 0; i < 30; i++) {

    context.clearRect(x1 - 0.5,y,1,altura);
    context.clearRect(xaux - 0.5,y,1,altura);
    xaux = xaux - 0.5;

    largura = largura - 1;
    altura = altura + 1;
    y = y - 1;

    x1 = x1 + 0.5;

    context.fillRect(x1,y,largura,altura);

}
```

Fonte: Do autor.

6.1.3 Cenário de teste 3

O cenário de teste 3 foi elaborado baseando-se na implementação da página do aplicativo que realiza a animação relacionada ao conceito de flexão. A seguir são demonstrados os códigos utilizados na ferramenta *jsPerf* para a montagem do cenário de teste 3.

Na figura 21 pode ser visualizado o código HTML que foi inserido no campo *Preparation Code HTML* do *jsPerf.* Neste código HTML o elemento *canvas* é declarado e são definidas algumas propriedades para ele.

Figura 21 – Código HTML utilizado no cenário 3

```
<body>
    <canvas id="myCanvas"
            width="500"
            height="180"
            style="border: 2px solid black;" >
    </canvas>
</body>
```

Fonte: Do autor.

Para o cenário de teste 3 também foi definido o script para inicialização do teste, onde foram declaradas as vaiáveis necessárias e o elemento *canvas* foi inicializado, obtendo-se também o seu contexto 2D. O script presente na figura 22 foi inserido no campo *Define setup for all testes* e mostra as funções descritas acima.

Figura 22 – Código de inicialização utilizado no cenário 3

```
var canvas;
var context;
var curva = 95;
var i;

canvas = document.getElementById("myCanvas");
context = canvas.getContext("2d");
```

Fonte: Do autor.

Para conclusão da montagem do cenário de teste 3, foi adicionado o código responsável por realizar a manipulação do elemento *canvas* que foi definido na página HTML. A figura 23 permite a visualização do código.

Figura 23 – Código utilizado na comparação do cenário 3

```
for (i = 0; i < 75; i++) {

    context.clearRect(20,60,290,300);

    context.lineWidth = 15;
    context.beginPath();
    context.moveTo(33,95);
    context.bezierCurveTo(165,curva,165,curva,299,95);
    context.stroke();

    curva = curva + 1;

}
```

Fonte: Do autor.

Para os três cenários de teste foram definidas as informações de identificação na montagem utilizando o *jsPerf*. O autor deste trabalho, Natan de Souza Ramos, foi informado como responsável pela elaboração dos três cenários e no campo e-mail foi inserido o endereço natan.sramos@gmail.com. Cada cenário de teste também recebeu uma breve descrição que informa qual página do aplicativo está sendo abordada e a principal função do script que irá realizar as manipulações durante a execução.

Também foram informados os títulos Teste1Natan, Teste2Natan e Teste3Natan para os cenários de teste um, dois e três, respectivamente. Com o preenchimento do título para cada cenário, foram montadas as URLs que armazenam cada teste, as quais são demonstradas abaixo:

a) cenário de teste 1: jsperf.com/teste1natan/6;

b) cenário de teste 2: jsperf.com/teste2natan/2;

c) cenário de teste 3: jsperf.com/teste3natan.

Após a montagem dos três cenários de teste, foi realizada a execução de cada um deles a fim de obter os resultados de desempenho. Durante a execução foram tomadas algumas medidas para que não houvesse interferências que pudessem influenciar na precisão dos resultados.

Foi aberta no *browser* apenas a página do *jsPerf,* onde durante os testes não foi utilizada a barra de rolagem e a página não foi redimensionada pois estes procedimentos causam disparos de eventos no DOM e podem tornar os testes menos precisos (GIANNATTASIO, 2013).

Com o objetivo de validar os resultados e garantir um teste mais confiável, os cenários de teste foram executados três vezes cada um, e também em cada

navegador. O *jsPerf* faz uma média das operações por segundo obtidas em cada execução. Desta forma, como resultados finais, foram considerados os valores das últimas execuções, pois a ferramenta fez a média das três execuções realizadas fornecendo o gráfico com esses valores (GIANNATTASIO, 2013).

6.2 RESULTADOS OBTIDOS

Com a utilização do HTML5 foi possível desenvolver o que havia sido proposto para o aplicativo. O elemento *canvas* possibilita a elaboração de muitas funcionalidades, e no caso das animações, atendeu as expectativas, além de ser compatível com os navegadores e versões utilizados na comparação.

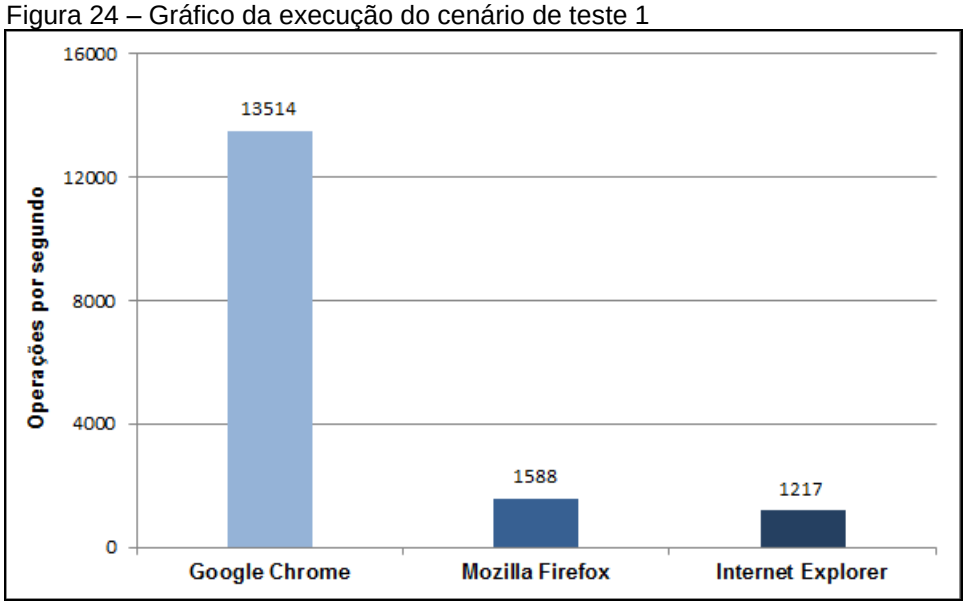
O aplicativo também atendeu as necessidades relacionadas ao ensino e aprendizagem. As animações realizadas pelo sistema foram capazes de ilustrar e exemplificar os conceitos de compressão, tração e flexão relacionados aos sistemas estruturais.

O HTML5 agrega novas funcionalidades para o desenvolvimento de aplicações, onde algumas das principais deficiências da web são atendidas. Como destaque surge a reprodução nativa de áudio e vídeo acabando com a necessidade de plug-ins e softwares de terceiros. Essas são algumas das novidades, mas o interessante é que o HTML5 facilita a adição de algumas funcionalidades nas páginas web, que antes eram complicadas de serem desenvolvidas.

Com a comparação do aplicativo através dos cenários de teste elaborados, foi possível obter os resultados de desempenho de cada navegador. Como descrito na metodologia, a ferramenta *jsPerf* fornece gráficos com os resultados de cada cenário de teste. Esses resultados referem-se à quantidade de operações por segundo executadas pelo navegador. Os gráficos gerados pela ferramenta foram adaptados para um melhor entendimento e são apresentados a seguir.

A figura 24 mostra o gráfico que demonstra o resultado de desempenho obtido com a execução do cenário de teste 1. Neste primeiro gráfico o Google Chrome aparece com um melhor desempenho entre os três navegadores. Em segundo lugar está o Mozilla Firefox seguido do Internet Explorer, o qual apresentou o pior resultado.

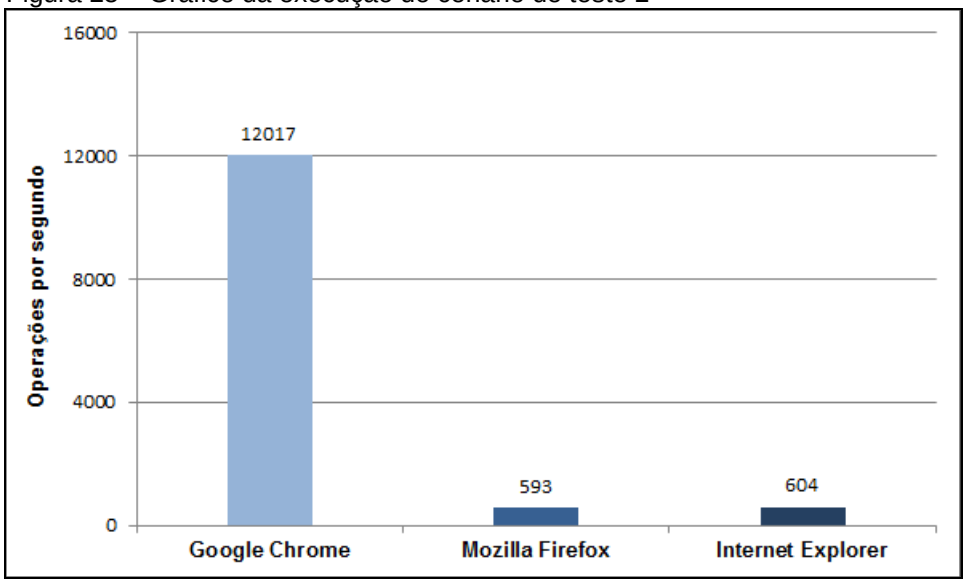
Figura 24 – Gráfico da execução do cenário de teste 1



Fonte: Do autor.

O gráfico do cenário de teste 2 é demonstrado na figura 25. Neste segundo gráfico pode-se notar que o Google Chrome permanece na frente em relação aos demais navegadores. No entanto, neste caso o Internet Explorer apresenta um desempenho melhor que o Mozilla Firefox.

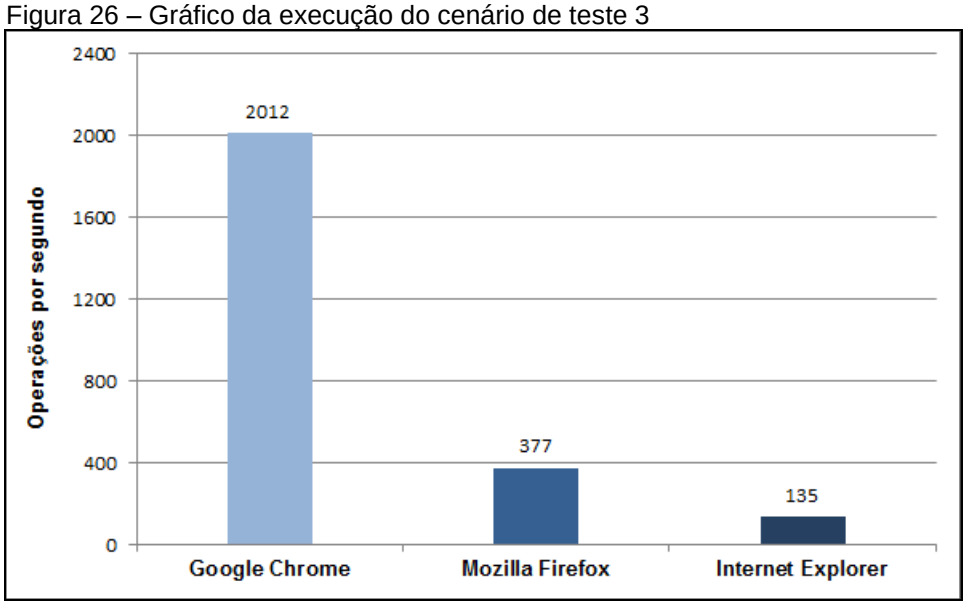
Figura 25 – Gráfico da execução do cenário de teste 2



Fonte: Do autor.

Na figura 26 está demonstrado o gráfico de execução do terceiro cenário de teste, onde o Google Chrome se confirma com o melhor desempenho, seguido do Mozilla Firefox, que volta a estar a frente do Internet Explorer, o qual apresenta novamente o pior desempenho.

Figura 26 – Gráfico da execução do cenário de teste 3



Fonte: Do autor.

De maneira geral, o Google Chrome apresentou o melhor desempenho entre os três navegadores, possuindo inclusive uma vantagem considerável em relação aos demais. Apesar de ser inferior ao Internet Explorer em um dos três testes, o Mozilla Firefox é o segundo colocado, ficando atrás apenas do Google Chrome. Por fim, o Internet Explorer aparece com o pior resultado, apesar de estar próximo do Firefox, ainda assim foi inferior.

Desta forma foram obtidos os resultados dos testes de comparação de desempenho entre os navegadores Google Chrome, Mozilla Firefox e Internet Explorer. Esses resultados fornecem dados que levam a conclusões a cerca da disputa que acontece entre os navegadores de internet, os quais estão sempre em busca do aperfeiçoamento das funcionalidades para se adaptarem as mudanças da web.

7 CONCLUSÃO

Com este trabalho pode-se concluir que a utilização de recursos tecnológicos para auxiliar o processo de ensino e aprendizagem é de grande importância e eficácia. O aplicativo oferece uma contribuição para o processo de aprendizagem de conceitos da área de engenharia, onde o aprendizado de sistemas estruturais se torna mais interativo e dinâmico, além de facilitar o entendimento.

A utilização do elemento *canvas* apresenta-se como uma boa alternativa para o desenvolvimento de animações e funcionalidades que envolvam efeitos visuais nas páginas web. Foi possível identificar que o uso deste elemento do HTML5 é de certa forma simples e oferece muitas possibilidades de desenvolvimento. Para a manipulação do *canvas* é necessário o conhecimento da linguagem JavaScript, pois é por meio dela que são efetuados os desenhos e efeitos no elemento. Além dos benefícios mencionados, vale destacar que o *canvas* foi suportado pelos navegadores e versões utilizadas na comparação, não apresentando problemas no funcionamento das animações desenvolvidas.

Mesmo em processo de evolução e dependendo da adaptação pelos navegadores, o HTML5 apresenta-se como uma tecnologia promissora para o desenvolvimento de aplicações web interativas, além de minimizar problemas relacionados à compatibilidade com o uso de alguns recursos tecnológicos.

Mesmo estando em constante evolução e aperfeiçoamento, os navegadores de internet podem apresentar diferenças na forma como suportam os elementos disponíveis para a construção de páginas web. Com a comparação realizada, constatou-se que o navegador Google Chrome está na frente em relação ao Mozilla Firefox e Internet Explorer no que diz respeito ao desempenho na execução de animações com o elemento *canvas*. Em segundo lugar está o Mozilla Firefox, apresentando um resultado inferior ao navegador da Google. Por fim, o Internet Explorer obteve os piores resultados nos testes de desempenho.

Ao longo do trabalho foi possível notar que o Google Chrome vem se sobressaindo em relação aos demais navegadores. Essa vantagem se estabelece não apenas pelo maior número de usuários, mas também por sua capacidade de renderizar as páginas de maneira rápida e eficaz. A comparação de desempenho realizada neste trabalho confirma a superioridade do Google Chrome.

Em aplicações maiores e mais complexas, como jogos, o desempenho e compatibilidade do navegador, ao trabalhar com certo componente HTML, podem influenciar no funcionamento da interação realizada pelo usuário.

Ao longo do desenvolvimento do trabalho foram encontradas algumas dificuldades. No entanto, o estudo e entendimento dos conceitos relacionados aos sistemas estruturais foram mais complicados, isso por se tratar de assuntos pertencentes à outra área do conhecimento.

Mesmo com as dificuldades, os objetivos foram atingidos, sendo que os resultados podem nortear os desenvolvedores sobre a utilização do HTML5, além de fornecer subsídios sobre o atual funcionamento, desempenho e utilização dos principais navegadores de internet.

Durante o desenvolvimento do trabalho, considerando os conhecimentos construídos acerca das tecnologias e conceitos estudados, surgem novas possibilidades para dar continuidade ao projeto, no intuito de auxiliar ainda mais os acadêmicos no ensino e aprendizagem de sistemas estruturais. Com a ampliação do trabalho poderão ser explorados os recursos computacionais oferecidos, além de obter mais detalhes sobre a função do navegador de internet. Desta forma, como trabalhos futuros, sugere-se:

a) desenvolver animações que abordem outros conceitos relacionados aos sistemas estruturais, como torção;
b) adicionar efeitos 3D em algumas animações para tornar a experiência do usuário mais próxima da realidade, facilitando o seu entendimento;
c) realizar outras comparações entre os navegadores, onde sejam analisados outros aspectos como consumo de memória;
d) adaptar o aplicativo para utilização em dispositivos móveis, onde as animações possam ser realizadas nos navegadores disponíveis sem afetar o seu funcionamento;
e) ampliar o desenvolvimento utilizando mais formas, exemplos clássicos e modelos, além de realizar integrações com applets e bibliotecas gráficas.

REFERÊNCIAS

ALVAREZ, Miguel Angel. **Entender a tela de canvas.** 2010. Disponível em: <http://www.criarweb.com/artigos/entender-tela-de-canvas.html> Acesso em: 23 nov. 2014.

ALVES, Paulo. **W3C finalmente recomenda o padrão HTML5 para a internet; entenda o que é.** Disponível em: <http://www.techtudo.com.br/noticias/noticia/2014/10/w3c-finalmente-recomenda-o-padrao-html5-para-internet-entenda-o-que-e.html> Acesso em: 29 out. 2014.

AQUINO, Fernando D'. **A evolução dos navegadores:** Google Chrome. 2011. Disponível em: <http://www.tecmundo.com.br/infografico/9485-a-evolucao-dos-navegadores-google-chrome-infografico-.htm>. Acesso em: 28 abr. 2015.

BEER, Ferdinand P.; JOHNSTON JR, E. Russell. **Resistência dos Materiais.** 3 ed. São Paulo: Pearson Makron Books, 1995.

BISPO, Diogo Moreira et al. Desenvolvimento de jogo educacional sobre ecotoxicologia utilizando HTML5. **Revista Brasileira de Informática na Educação.** Porto Alegre, v. 20, n. 1, p. 121-131, abr. 2012. Disponível em: <http://www.br-ie.org/pub/index.php/rbie/article/view/1366/1199> Acesso em: 9 set. 2014.

CARVALHO, Tiago de. **Avaliação e comparação de recursos utilizando tecnologias HTML5 e Flash para criação de aplicações web.** 2013. 92 f. TCC (Graduação) – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, 2013. Disponível em: <http://tcc.kironunesc.net.br/arquivos/trabalhos/368.pdf> Acesso em: 3 nov. 2014.

CERN. **The birth of the web.** 2014a. Disponível em: <http://home.web.cern.ch/topics/birth-web> Acesso em: 5 nov. 2014.

CERN. **About CERN.** 2014b. Disponível em: <http://home.web.cern.ch/about> 5 nov. 2014.

DIAS, Cláudia. **Usabilidade na WEB: criando portais mais acessíveis.** 2 ed. Rio de Janeiro: Alta Books, 2007.

ELEMAR JUNIOR, **HTML5 parte 6:** armazenando dados com web sql databases. 2010. Disponível em: <http://elemarjr.net/2010/10/19/html-5-parte-6-armazendando-dados-com-web-sql-databases/> Acesso em: 16 nov. 2014.

FERREIRA, Elcio; EIS, Diego. **HTML5:** curso W3C escritório Brasil. 2010. Disponível em: <http://www.w3c.br/pub/Cursos/CursoHTML5/html5-web.pdf> Acesso em: 15 nov. 2014.

FNDE. **Apresentação.** Apresentação do Programa Nacional de Informática na Educação (Fundo nacional de desenvolvimento da educação). Disponível em: <http://www.fnde.gov.br/programas/programa-nacional-de-tecnologia-educacional-

proinfo> Acesso em: 7 out. 2014.

GIANNATTASIO, Gabriel R.. **TESTES DE DESEMPENHO PARA JAVASCRIPT COM JSPERF.** 2013. Disponível em: <http://gartz.com.br/testes-de-desempenho-para-javascript-com-jsperf/> Acesso em: 7 maio 2015.

GONÇALVES, Flávio A. S.; CANESIN, Carlos A. Java applets para um software educacional distribuído em eletrônica de potência. **Controle e Automação.** São Paulo, v. 13, n. 3, p. 314-326, set./out./nov./dez. 2002. Disponível em: <http://www.scielo.br/pdf/ca/v13n3/a10v13n3.pdf> Acesso em: 30 mar. 2014.

GRZESIUK, Diorgenes Felipe. **O uso da informática na sala de aula como ferramenta de auxílio no processo ensino-aprendizagem.** 2008. 48f. Monografia (Pós Graduação em Métodos e Técnicas de ensino, Modalidade de Ensino a Distância) – Universidade Tecnológica Federal do Paraná, Medianeira. Disponível em: <http://diorgenes.files.wordpress.com/2009/06/monografia_utfpr_diorgenes.pdf> Acesso em: 14 out. 2013.

HARBS, Marcos. **Motor para jogos 2D utilizando HTML5.** 2013. 78 f. TCC (Graduação) – Curso de Ciência da Computação, Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2013. Disponível em: <http://dsc.inf.furb.br/arquivos/tccs/monografias/2013_2_marcos-harbs_monografia.pdf> Acesso em: 29 maio 2015.

HARRIS, Tom. **Plug-ins.** Disponível em: <http://tecnologia.hsw.uol.com.br/animacoes-para-a-web4.htm> Acesso em: 15 nov. 2014.

HAUTSCH, Oliver. **Qual o melhor navegador?** 2010. Disponível em: <http://www.tecmundo.com.br/navegador/3591-fevereiro-de-2010-qual-o-melhor-navegador-.htm> Acesso em: 27 maio 2015.

HENRIQUES, Augusto et al. **O navegador como plataforma para jogos:** uma experiência extracurricular para desenvolvimento de software. 2011. Disponível em: <http://www.sbgames.org/sbgames2011/proceedings/sbgames/papers/comp/short/26-92300_2.pdf> Acesso em: 23 nov. 2014.

HEY, Daniel Fuverki. **Estudo de Viabilidade do HTML5 para Desenvolvimento Web.** 2010. 75 f. TCC (Graduação) – Curso de Ciência da Computação, Departamento de Informática, Universidade Estadual de Maringá, Maringá, 2010. Disponível em: <http://munif.com.br/munifold/arquivos/HTML5-FINAL.pdf?id=387> Acesso em: 15 nov. 2014.

HIBBELER, Russell Charles. **Mecânica Estática.** 8 ed. Rio de Janeiro: LTC, 1999.

HIBBELER, Russell Charles. **Resistência dos Materiais.** 3 ed. Rio de Janeiro: LTC, 2000.

HIGA, Paulo. **Adeus, Internet Explorer. Bem vindo, Microsoft Edge.** 2015. Disponível em: <https://tecnoblog.net/177528/microsoft-edge/> Acesso em: 1 maio

2015.

HTML5TEST – How well does your browser support HTML5? Disponível em: <http://html5test.com/results/desktop.html> Acesso em: 5 nov. 2014.

JSPERF. **jsPerf:** JavaScript performance playground. 2015a. Disponível em: <https://jsperf.com/> Acesso em: 10 maio 2015.

JSPERF. **Frequently asked questions.** 2015. Disponível em: <https://jsperf.com/faq> Acesso em: 10 maio 2015.

LALLI, Felipe Micaroni; BUENO, Felipe Franco; ZACHARIAS, Guilherme Keese. **Evolução da programação web.** 2008. 68 f. TCC (Graduação) – Curso de Ciência da Computação, Faculdade Comunitária de Campinas Unidade III, Campinas, 2008. Disponível em: <http://battisti.etc.br/unialfa/2011/pdi/evolucao-da-programacao-web.pdf> Acesso em: 15 nov. 2014.

LANDIM, Wikerson. **A Evolução dos Navegadores:** Mozilla Firefox. 2011. Disponível em: <http://www.tecmundo.com.br/infografico/9325-a-evolucao-dos-navegadores-mozilla-firefox-infografico-.htm> Acesso em: 30 abr. 2015.

LOBO FILHO, Roberto Jorge Haddock. **Integração entre HTML5 e JSF 2.0 em aplicações web off-line.** 2010. 82 f. TCC (Graduação) – Curso de Ciência da Computação, Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis, 2010. Disponível em: <https://projetos.inf.ufsc.br/arquivos_projetos/projeto_809/TCC+Artigo.pdf> Acesso em: 16 nov. 2014.

MORAN, José Manoel. **A educação que desejamos:** novos desafios e como chegar lá. Campinas: Papirus, 2007. 174 p.

NASCIMENTO, João Kerginaldo Firmino do. **Informática aplicada à educação.** 2007. 84 p. Universidade de Brasília. Disponível em: <http://portal.mec.gov.br/seb/arquivos/pdf/profunc/infor_aplic_educ.pdf> Acesso em: 3 maio 2014.

PRASS, Ronaldo. **Saiba mais sobre os principais navegadores de internet.** 2011. Disponível em: <http://g1.globo.com/tecnologia/noticia/2011/06/saiba-mais-sobre-os-principais-navegadores-de-internet.html> Acesso em: 04 jun. 2015.

REBELLO, Yopanan Conrado Pereira. **A concepção estrutural e a arquitetura.** São Paulo: Zigurate Editora, 2000. 271 p.

RODRIGUES, Lucas; PRADO, Antonio Francisco do. Desenvolvimento de aplicações móveis com serviços RESTfull e HTML5. **TIS, Tecnologias, Infraestrutura e Software.** São Carlos, v. 3, n. 2, p. 171-179, maio/ago. 2014. Disponível em: <http://revistatis.dc.ufscar.br/index.php/revista/article/view/88/81> Acesso em: 9 nov. 2014.

RUSSO, Rafael. **A História do Internet Explorer.** 2010. Disponível em:

<http://escreveassim.com.br/2010/11/18/a-historia-do-internet-explorer/> Acesso em: 1 maio 2015.

SANTOS, Jorge Guilherme S.. **Comparação de desempenho de navegadores.** 2013. Disponível em: <http://www.hardware.com.br/artigos/desempenho-navegadores/> Acesso em: 27 maio 2015.

SCHROEDER, Ricardo. HTML5, um novo desenvolvimento para a web. **Caminhos: Revista on-line de divulgação científica da UNIDAVI.** Rio do Sul, a. 3, n. 5, p. 25-39, jul./set. 2012. Disponível em: <http://www.caminhos.unidavi.edu.br/wp-content/uploads/2012/11/DG_2_2.pdf> Acesso em: 17 out. 2013.

SEBBEN, Naiara; GUEDES, Anibal Lopes. Gráficos vetoriais na criação de interfaces de sistemas para a internet. In: PÓS GRADUAÇÃO – ESPECIALIZAÇÃO EM DESENVOLVIMENTO DE SISTEMAS PARA A INTERNET, 1., 2010, São Miguel do Oeste. **Anais Eletrônicos.** São Miguel do Oeste, 2010. Disponível em: <http://www.naiarasebben.com.br/artigos/graficos_vetoriais_escalaveis_na_criacao_de_interfaces_de_sistemas_para_a_internet.pdf> Acesso em: 23 nov. 2014.

SERRA, Ricardo Jorge Maia e. **Interfaces tácteis baseadas em HTML5/CSS/JavaScript.** 2011. 89f. Dissertação (Mestrado) – Curso de Computação, Faculdade de Engenharia da Universidade do Porto, Porto (Portugal), 2011. Disponível em: <http://repositorio-aberto.up.pt/bitstream/10216/63293/1/000149242.pdf> Acesso em: 13 nov. 2014.

SILVA, Cleberson Paulo de Andrade; SANTOS, Marilde Terezinha Prado. HTML5: uma análise de compatibilidade dos elementos áudio, vídeo e canvas com os principais navegadores. **TIS, Tecnologias, Infraestrutura e Software.** São Carlos, v. 2, n. 1, p. 35-41, jan./fev./mar./abr 2013. Disponível em: <http://revistatis.dc.ufscar.br/index.php/revista/article/view/34/37> Acesso em: 9 set. 2014.

SMUS, Boris. **Melhoria do desempenho do canvas em HTML5.** 2013. Disponível em: <http://www.html5rocks.com/pt/tutorials/canvas/performance/> Acesso em: 7 maio 2015.

SOARES, Walace. **AJAX (Asynchronous JavaScript And XML):** guia prático para Windows. 1 ed. São Paulo: Érica, 2006.

SOUSA, Kassio. **HTML.** 2011. Disponível em: <http://www.connection.ufma.br/Apostila_html_e_xhtml.pdf> Acesso em: 12 nov. 2014.

TAJRA, Sanmya Feitosa. **Informática na educação:** novas ferramentas pedagógicas para o professor da atualidade. 2 ed. São Paulo: Érica, 2000.

TAVARES, Neide Rodriguez Barea. **História da informática educacional no Brasil observada a partir de três projetos públicos.** 2008. 16 p. LAPEQ – Laboratório de pesquisa em ensino de química e tecnologias educativas. Disponível em: <http://www.lapeq.fe.usp.br/textos/tics/ticspdf/neide.pdf> Acesso em: 15 out. 2014.

W3SCHOOLS. **Browser Statistics.** 2014a. Disponível em: <http://www.w3schools.com/browsers/browsers_stats.asp> Acesso em: 23 nov. 2014.

W3SCHOOLS. **HTML áudio tag.** 2014b. Disponível em: <http://www.w3schools.com/tags/tag_audio.asp> Acesso em: 23 nov. 2014.

W3SCHOOLS. **HTML vídeo tag.** 2014c. Disponível em: <http://www.w3schools.com/tags/tag_video.asp> Acesso em: 23 nov. 2014.

WALNIER, Juliano Miguel. **Informática na educação:** conceitos e tecnologias. 2006. 64 f. Monografia (Especialização em Didática e Metodologia do ensino Superior – Universidade do Extremo Sul Catarinense, Criciúma, 2006).

WILEY, D. **Connecting learning objects to instrucional design theory:** a definition, a metaphor, and a taxonomy. 2000. Disponível em: <http://www.reusability.org/read/chapters/wiley.doc> Acesso em: 30 out. 2014.

WORLD WIDE WEB FOUNDATION. **History of the Web.** Disponível em: <http://webfoundation.org/about/vision/history-of-the-web/> Acesso em: 5 nov. 2014.

APÊNDICE A - ARTIGO

Comparação de um aplicativo desenvolvido em HTML5, nos diferentes navegadores de internet

Natan de S. Ramos[1], Luciano Antunes[2], Marcio Vito[3]

[1]Acadêmico do Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias - Universidade do Extremo Sul Catarinense (UNESC) – Caixa Postal 3.167 – 88806-000 – Criciúma – SC – Brasil

[2] Professor do Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias – Universidade do Extremo Sul Catarinense (UNESC) – Caixa Postal 3.167 – 88806-000 – Criciúma – SC – Brasil

[3] Professor do Curso de Engenharia Civil – Unidade Acadêmica de Ciências, Engenharias e Tecnologias – Universidade do Extremo Sul Catarinense (UNESC) – Caixa Postal 3.167 – 88806-000 – Criciúma – SC – Brasil

natan.sramos@gmail.com, luc@unesc.net, marciovito@unesc.net

Abstract. *This article aims to demonstrate the performance comparison of an application developed in HTML5, held between different internet browsers. Internet browsers seek standardization to meet the requirements proposed by the World Wide Web Consortium (W3C) and support the greatest number of tags. They may show different behaviors since they are rendering engines and separate JavaScript execution.*

Resumo. *Este artigo tem como meta demonstrar a comparação de desempenho de um aplicativo desenvolvido em HTML5, realizada entre os diferentes navegadores de internet. Os navegadores de internet buscam uma padronização para atender as exigências propostas pelo World Wide Web Consortium (W3C) e dar suporte ao maior número possível de tags. Contudo, podem apresentar diferentes comportamentos, visto que possuem motores de renderização e de execução JavaScript distintos.*

1. Introdução

A tecnologia está presente em muitas atividades desempenhadas pelos seres humanos, nas mais diversas áreas do conhecimento. Esta, quando associada ao uso do computador na educação, implica diretamente na melhoria do processo de ensino e aprendizagem. Sua utilização deve ser planejada, visando coerência com estratégias, métodos e técnicas de ensino, aproveitando suas qualidades de potencial [GRZESIUK, 2008].

Muitas vezes, a metodologia não é a mais adequada e suficiente para que o acadêmico compreenda alguns conceitos importantes e de fácil percepção. De acordo com o professor Marcio Vito e outros professores de engenharia da UNESC, esse problema está presente entre os acadêmicos da área. Um exemplo disso é quando o aluno realiza os cálculos, mas não entende o comportamento dos elementos estruturais.

Nos sistemas estruturais, onde se trabalha com o equilíbrio dos corpos, a compatibilidade de deslocamentos e a resistência dos materiais, fenômenos simples em seus

princípios, o aluno tem dificuldades em observar o lado qualitativo, não tendo a percepção do problema [REBELLO, 2000].

O desenvolvimento do aplicativo tem como finalidade, auxiliar a aprendizagem dos sistemas estruturais, onde são representados os conceitos de compressão, tração e flexão. Para a implementação foram utilizadas as tecnologias *client side*, onde a execução é realizada pelo próprio navegador de internet. Atualmente existem muitos navegadores disponíveis no mercado, estando estes em constante disputa para conquistar mais espaço e adquirir novos usuários.

No caso dos navegadores, softwares que tem a função básica de receber um script e renderizar a página para o usuário, alguns fatores podem ser avaliados para se testar o desempenho. A velocidade e compatibilidade com HTML5, a capacidade ao trabalhar com efeitos gráficos, ou até mesmo a simples execução de um código JavaScript [SANTOS, 2013].

Na busca pelo melhor navegador de internet, deve-se considerar a preferência do usuário, o qual se identifica com um determinado navegador. No entanto, isto não significa que ele seja o melhor em termos técnicos e computacionais. Sabe-se que o desempenho de um navegador pode ser influenciado por inúmeros motivos, como a configuração do computador utilizado, o sistema operacional, a quantidade de aplicativos abertos simultaneamente, a velocidade da internet e os complementos instalados junto ao *browser* [HAUTSCH, 2010].

Com isso, esse trabalho buscou definir as métricas de comparação de desempenho entre os principais navegadores de internet, além de identificar e compreender os novos recursos trazidos pela linguagem HTML5 empregados no desenvolvimento do aplicativo.

2. Navegadores de internet e tecnologias *client side*

Um navegador, também conhecido pelos termos ingleses *web browser* ou simplesmente *browser,* é um programa de computador que possibilita aos usuários interagirem com os documentos da Internet, ou páginas web, as quais podem ser elaboradas com o uso de HTML, ASP, PHP e CSS, entre outras tecnologias, ficando hospedadas em um servidor web [SOUSA, 2011].

As tecnologias *client side* são aquelas executadas no lado do cliente, ou seja, pelo *browser* que está sendo utilizado pelo usuário. Sua utilização é muito importante, pois permite a criação de páginas web mais sofisticadas, ao invés de apenas a exibição do HTML que é retornado pelo servidor. Além disso, com essas tecnologias é possível a criação de alguns procedimentos básicos da web sem a necessidade de acessar o servidor. Um exemplo é a validação das informações de um formulário quando o mesmo é submetido na aplicação [LOBO FILHO, 2010].

Além de novas funcionalidades e recursos de multimídia, uma das características mais esperadas do HTML5, é o elemento para a criação e manipulação de imagens, o *canvas.* Para que seja possível trabalhar com o *canvas,* são necessários o CSS e principalmente o JavaScript, tendo estes, um papel muito importante no desenvolvimento web com HTML5.

O elemento *canvas* é uma API que possibilita a definição de uma área onde podem ser renderizados elementos visuais em tempo de execução, permitindo por meio de JavaScript, que esses elementos sejam manipulados, utilizando o conceito de bitmap, pixel a pixel [SEBBEN; GUEDES, 2010].

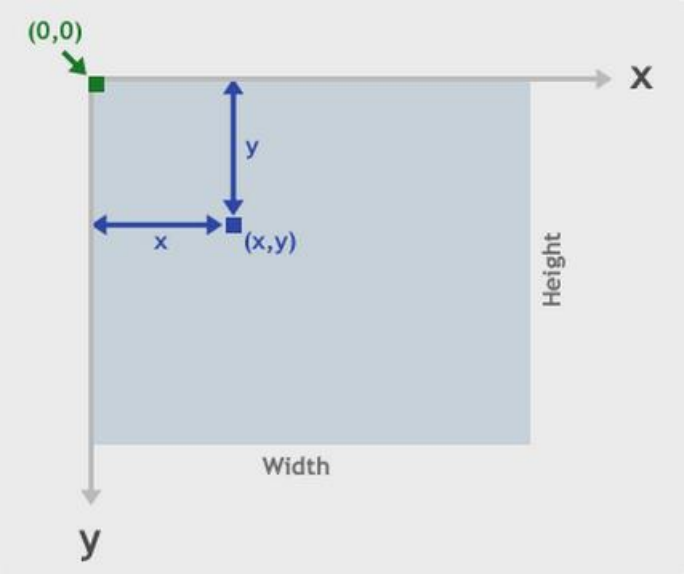


Figura 1. Funcionamento do elemento *canvas*

3. Desenvolvimento e comparação do aplicativo em diferentes navegadores

Está sessão tem como finalidade, detalhar o desenvolvimento do aplicativo, e principalmente a metodologia empregada para realizar a comparação de desempenho entre os navegadores de internet.

3.1 Metodologia

Para a realização do trabalho, com intuito de alcançar os objetivos propostos, foi necessário o cumprimento de algumas etapas e a compreensão de alguns conceitos, tecnologias para o desenvolvimento e formas de comparação de aplicações web.

Para o desenvolvimento do aplicativo foram utilizadas tecnologias *client side*, as quais são executadas pelo próprio navegador, envolvendo HTML5, CSS e JavaScript. Na definição do leiaute do aplicativo, foi utilizado um template do Twitter Bootstrap, chamado *Simple Sidebar*. Para a utilização deste template foi necessário entender a sua estrutura para que as páginas pudessem ser criadas e adaptadas de acordo com a necessidade.

O principal elemento utilizado foi o *canvas*, permitindo a renderização de imagens e formas, além de realizar animações em tempo de execução. O elemento *canvas* é declarado no corpo da página HTML, onde recebe um identificador para ser manipulado através do código JavaScript.

Após o desenvolvimento do aplicativo, a fim de alcançar o objetivo computacional do trabalho, foi iniciada a comparação entre os navegadores. Nesta etapa foram pesquisados softwares e formas de comparação que poderiam ser utilizados.

Para a realização da comparação foram adotados os navegadores Google Chrome, Mozilla Firefox e Internet Explorer, sendo estes os mais utilizados atualmente. Foram utilizadas as versões mais atuais destes navegadores, colocando-os nas mesmas condições de uso, sem nenhum tipo de bloqueio durante os testes, como execução de códigos JavaScript, por exemplo. As versões dos navegadores que foram utilizadas são apresentadas a seguir:

a) Google Chrome: 42.0.2311;
b) Mozilla Firefox: 37.0;
c) Internet Explorer 11: 11.0.9600.

Para realizar a comparação foi utilizado o software *jsPerf*, um sistema que possibilita a criação de cenários de teste, comparando o desempenho de trechos de código por meio de *benckmarks*. O *jsPerf* também permite o versionamento de cada teste elaborado, possibilitando a edição e execução de cada projeto em revisões diferentes [SMUS, 2013].

O software executa o código informado, o máximo de vezes possível durante um período de tempo. Ao final da execução, são geradas informações de desempenho. Estas informações são referentes à quantidade de operações por segundo que foram alcançadas com a execução do script, onde quanto mais operações por segundo, melhor o desempenho. Este desempenho pode ser avaliado entre os navegadores, pois o software permite que um mesmo teste seja executado em diferentes *browsers* [SMUS, 2013].

O *JsPerf* foi desenvolvido por John-David Dalton, um programador da Microsoft especializado em análise de desempenho de código. John-David Dalton tem participação na produção e manutenção do Internet Explorer, além de colaborar com outros projetos de código aberto, onde procura auxiliar os desenvolvedores para que produzam scripts mais otimizados e bem elaborados. A seguir serão detalhados os passos para criação e execução dos cenários de teste por meio do *jsPerf*.

Para a criação dos cenários de testes na ferramenta *jsPerf*, é necessário o preenchimento de algumas informações. Inicialmente devem ser fornecidos os dados do criador do cenário de teste, como nome e e-mail. Posteriormente deve-se definir o título e uma breve descrição, o que facilita a identificação do cenário de teste.

No momento em que é informado o título do cenário de teste, o campo *Slug* é alimentado automaticamente, onde é montada a URL que armazenará o teste. A URL é formada com o endereço jsPerf.com mais uma barra seguida do conteúdo do campo *Slug*. A figura 2 apresenta o formulário do *jsPerf* para o preenchimento dos campos de identificação descritos acima.

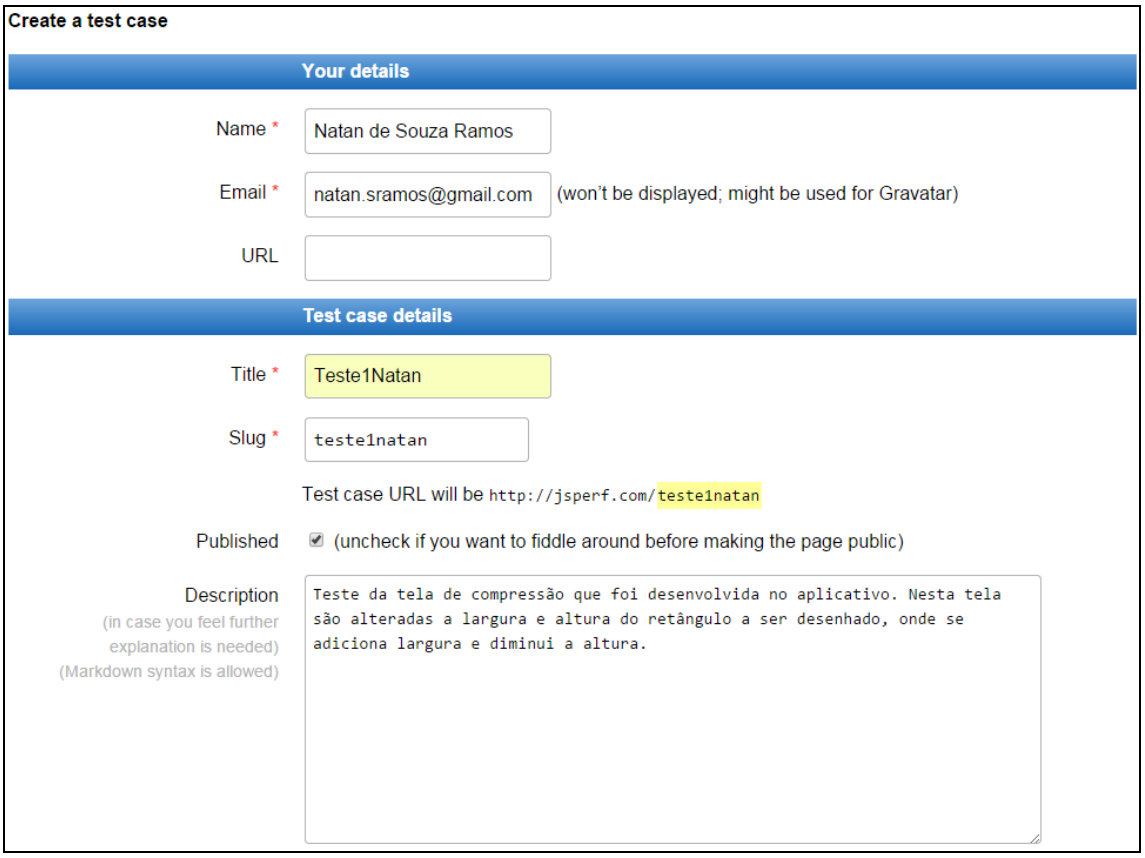


Figura 2. Informações de identificação do cenário de teste

Após a definição das informações de identificação do cenário de teste, devem ser informados os códigos que realizam a inicialização do teste. Na figura 3 são demonstrados os campos que possibilitam o preenchimento desses scripts.

No campo *Preparation code HTML,* deve ser informado o código HTML presente na estrutura da página, sendo necessário apenas o trecho contido no corpo da página, o qual se encontra dentro da tag *body*. Esse campo suporta elementos do HTML5, como o *canvas,* por exemplo, os quais podem ser manipulados ao longo da execução do teste.

O campo *Define setup for all tests* tem a função de armazenar a declaração de variáveis, funções ou objetos que serão utilizadas durante a execução. O último campo presente na preparação inicial dos scripts é o *Define teardown for all tests,* sendo executado após o término do teste. Entre outras funções, esse campo permite a reinicialização de variáveis ou o encerramento de uma conexão aberta durante a execução do cenário de teste. No entanto, esse campo geralmente não é utilizado, pois na maioria dos casos não é necessário.

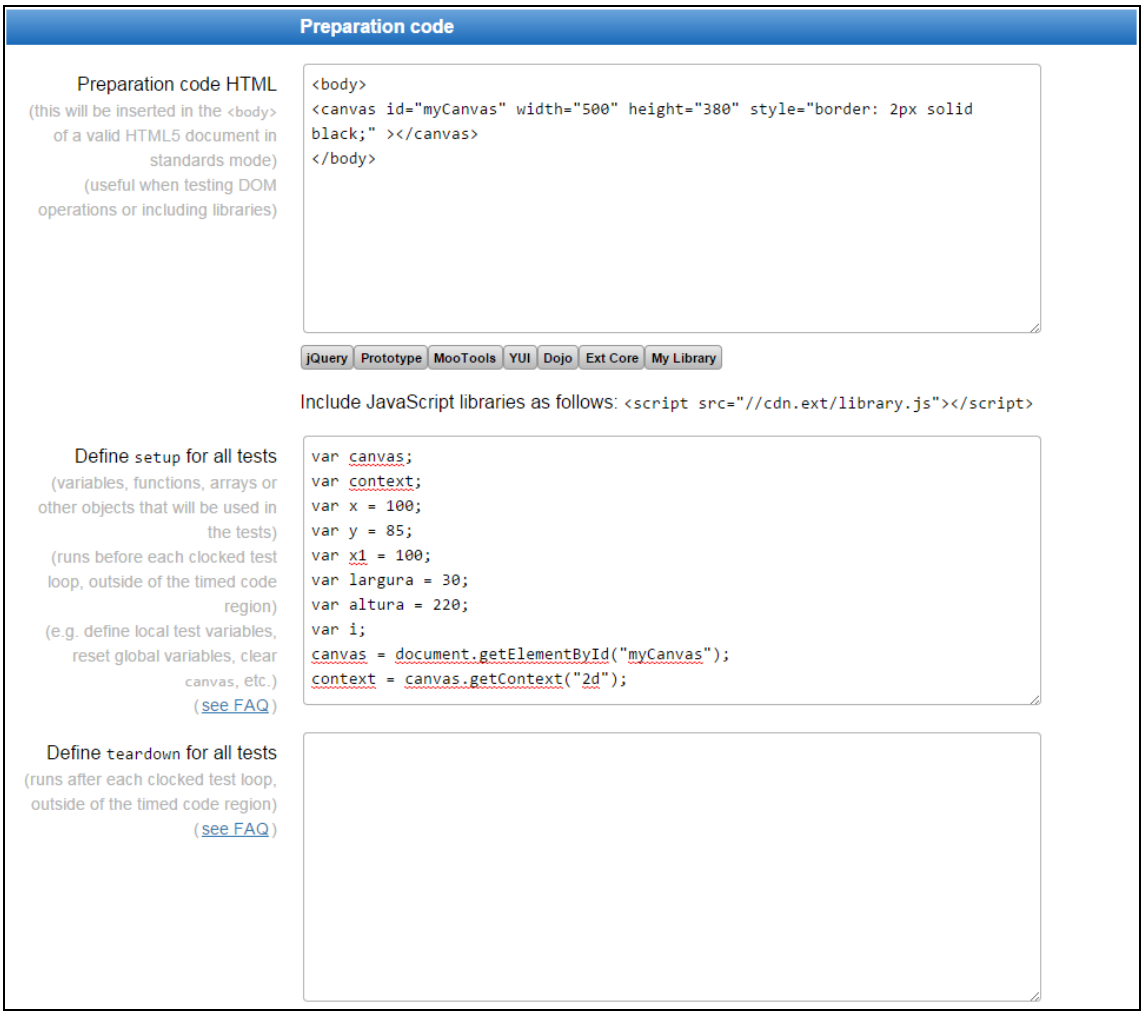


Figura 3. Códigos de inicialização do cenário de teste

Para concluir a montagem do cenário de teste, deve ser fornecido o código a ser comparado, que irá executar e manipular a página HTML ao longo da execução. Esse código altera as variáveis e elementos definidos nos campos de inicialização, realizando alterações na página. Na figura 4 pode ser observado o preenchimento deste campo.

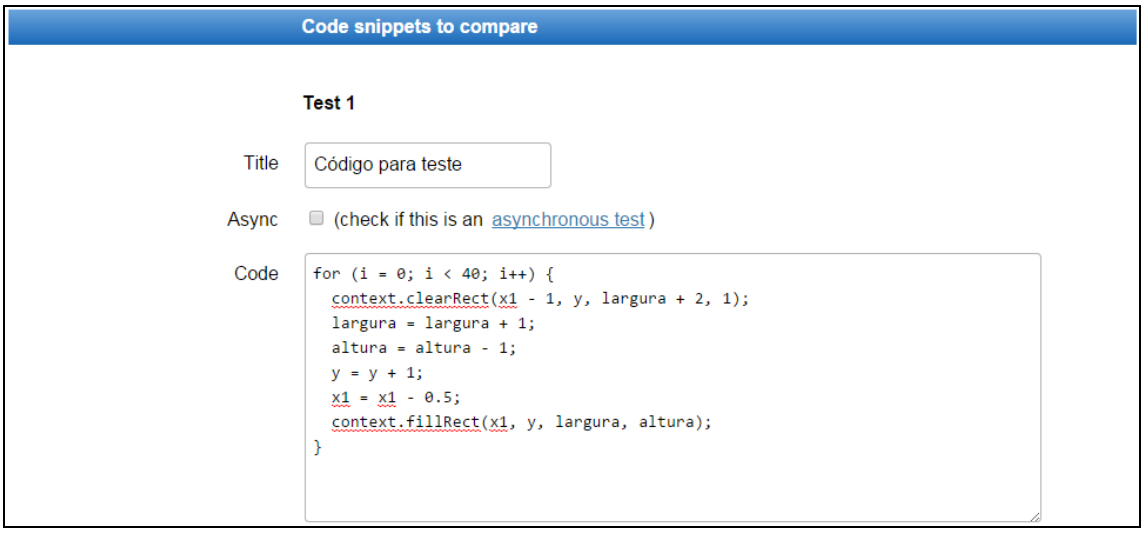


Figura 4. Código a ser comparado

Estando compreendido o funcionamento do software *jsPerf*, foram montados três cenários de teste, contemplando as páginas do aplicativo desenvolvido que realizam animações.

Para os três cenários de teste foram definidas as informações de identificação na montagem utilizando o *jsPerf*. O autor deste trabalho, Natan de Souza Ramos, foi informado como responsável pela elaboração dos três cenários e no campo e-mail foi inserido o endereço natan.sramos@gmail.com. Cada cenário de teste também recebeu uma breve descrição que informa qual página do aplicativo está sendo abordada e a principal função do script que irá realizar as manipulações durante a execução.

Também foram informados os títulos Teste1Natan, Teste2Natan e Teste3Natan para os cenários de teste um, dois e três, respectivamente. Com o preenchimento do título para cada cenário, foram montadas as URLs que armazenam cada teste, as quais são demonstradas abaixo:

a) cenário de teste 1: jsperf.com/teste1natan/6;
b) cenário de teste 2: jsperf.com/teste2natan/2;
c) cenário de teste 3: jsperf.com/teste3natan.

Após a montagem dos três cenários de teste, foi realizada a execução de cada um deles a fim de obter os resultados de desempenho. Durante a execução foram tomadas algumas medidas para que não houvesse interferências que pudessem influenciar na precisão dos resultados.

Foi aberta no *browser* apenas a página do *jsPerf,* onde durante os testes não foi utilizada a barra de rolagem e a página não foi redimensionada pois estes procedimentos causam disparos de eventos no DOM e podem tornar os testes menos precisos [GIANNATTASIO, 2013].

Com o objetivo de validar os resultados e garantir um teste mais confiável, os cenários de teste foram executados três vezes cada um, e também em cada navegador. O *jsPerf* faz uma média das operações por segundo obtidas em cada execução. Desta forma, como resultados finais, foram considerados os valores das últimas execuções, pois a ferramenta fez a média das três execuções realizadas fornecendo o gráfico com esses valores [GIANNATTASIO, 2013].

4. Resultados obtidos

Com a utilização do HTML5 foi possível desenvolver o que havia sido proposto para o aplicativo. O elemento *canvas* possibilita a elaboração de muitas funcionalidades, e no caso das animações, atendeu as expectativas, além de ser compatível com os navegadores e versões utilizados na comparação.

O aplicativo também atendeu as necessidades relacionadas ao ensino e aprendizagem. As animações realizadas pelo sistema foram capazes de ilustrar e exemplificar os conceitos de compressão, tração e flexão relacionados aos sistemas estruturais.

O HTML5 agrega novas funcionalidades para o desenvolvimento de aplicações, onde algumas das principais deficiências da web são atendidas. Como destaque surge a reprodução nativa de áudio e vídeo acabando com a necessidade de plug-ins e softwares de terceiros. Essas são algumas das novidades, mas o interessante é que o HTML5 facilita a adição de algumas funcionalidades nas páginas web, que antes eram complicadas de serem desenvolvidas.

Com a comparação do aplicativo através dos cenários de teste elaborados, foi possível obter os resultados de desempenho de cada navegador. Como descrito na metodologia, a ferramenta *jsPerf* fornece gráficos com os resultados de cada cenário de teste. Esses resultados referem-se à quantidade de operações por segundo executadas pelo navegador. Os gráficos gerados pela ferramenta foram adaptados para um melhor entendimento e os resultados são apresentados a seguir.

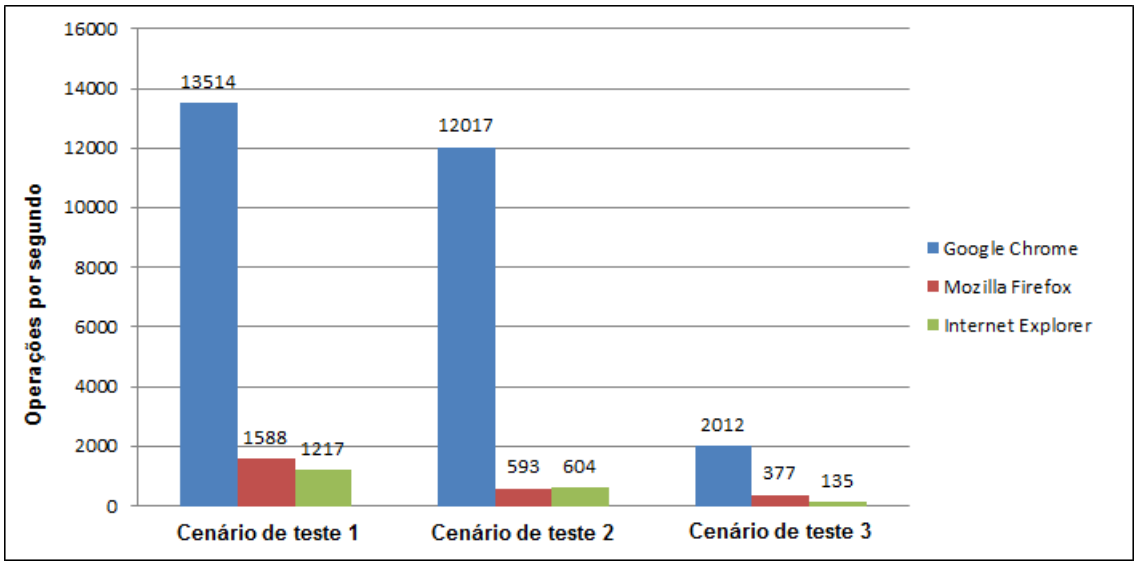


Figura 5. Resultados gerados pela ferramenta jsPerf

De maneira geral, o Google Chrome apresentou o melhor desempenho entre os três navegadores, possuindo inclusive uma vantagem considerável em relação aos demais. Apesar de ser inferior ao Internet Explorer em um dos três testes, o Mozilla Firefox é o segundo colocado, ficando atrás apenas do Google Chrome. Por fim, o Internet Explorer aparece com o pior resultado, apesar de estar próximo do Firefox, ainda assim foi inferior.

Desta forma foram obtidos os resultados dos testes de comparação de desempenho entre os navegadores Google Chrome, Mozilla Firefox e Internet Explorer. Esses resultados fornecem dados que levam a conclusões a cerca da disputa que acontece entre os navegadores

de internet, os quais estão sempre em busca do aperfeiçoamento das funcionalidades para se adaptarem as mudanças da web.

5. Conclusão

Mesmo em processo de evolução e dependendo da adaptação pelos navegadores, o HTML5 apresenta-se como uma tecnologia promissora para o desenvolvimento de aplicações web interativas, além de minimizar problemas relacionados à compatibilidade com o uso de alguns recursos tecnológicos.

Mesmo estando em constante evolução e aperfeiçoamento, os navegadores de internet podem apresentar diferenças na forma como suportam os elementos disponíveis para a construção de páginas web. Com a comparação realizada, constatou-se que o navegador Google Chrome está na frente em relação ao Mozilla Firefox e Internet Explorer no que diz respeito ao desempenho na execução de animações com o elemento *canvas*. Em segundo lugar está o Mozilla Firefox, apresentando um resultado inferior ao navegador da Google. Por fim, o Internet Explorer obteve os piores resultados nos testes de desempenho.

Ao longo do trabalho foi possível notar que o Google Chrome vem se sobressaindo em relação aos demais navegadores. Essa vantagem se estabelece não apenas pelo maior número de usuários, mas também por sua capacidade de renderizar as páginas de maneira rápida e eficaz. A comparação de desempenho realizada neste trabalho confirma a superioridade do Google Chrome.

Em aplicações maiores e mais complexas, como jogos, o desempenho e compatibilidade do navegador, ao trabalhar com certo componente HTML, podem influenciar no funcionamento da interação realizada pelo usuário.

Mesmo com as dificuldades, os objetivos foram atingidos, sendo que os resultados podem nortear os desenvolvedores sobre a utilização do HTML5, além de fornecer subsídios sobre o atual funcionamento, desempenho e utilização dos principais navegadores de internet.

Por fim, recomenda-se para trabalhos futuros:

a) adicionar efeitos 3D em algumas animações para tornar a experiência do usuário mais próxima da realidade;
b) realizar outras comparações entre os navegadores, onde sejam analisados outros aspectos como consumo de memória;
c) adaptar o aplicativo para utilização em dispositivos móveis, onde as animações possam ser realizadas nos navegadores disponíveis sem afetar o seu funcionamento;
d) ampliar o desenvolvimento utilizando mais formas, exemplos clássicos e modelos, além de realizar integrações com applets e bibliotecas gráficas.

6. Referências

Giannattasio, Gabriel R. (2013) “Testes de desempenho para JavaScript com jsPerf”, http://gartz.com.br/testes-de-desempenho-para-javascript-com-jsperf/, Maio.

Grzesiuk, Diorgenes Felipe. (2008) “O uso da informática na sala de aula como ferramenta de auxílio no processo de ensino-aprendizagem”, http://diorgenes.files.wordpress.com/2009/06/monografia_utfpr_diorgenes.pdf, Outubro.

Hautsch, Oliver. (2010) “Qual o melhor navegador?”, http://www.hardware.com.br/artigos/desempenho-navegadores/, Maio.

Lobo Filho, Roberto J. H. (2010) “Integração entre HTML5 e JSF 2.0 em aplicações web off-line”, https://projetos.inf.ufsc.br/arquivos_projetos/projeto_809/TCC+Artigo.pdf, Novembro.

Rebello, Yopanan C. P. (2000), A concepção estrutural e a arquitetura.

Santos, Jorge Guilherme S. (2013) “Comparação de desempenho de navegadores”, http://www.hardware.com.br/artigos/desempenho-navegadores/, Maio.

Sebben, Naiara, Guedes, Anibal L. (2010) “Gráficos vetoriais na criação de interfaces de sistemas para a internet”, http://www.naiarasebben.com.br/artigos/graficos_vetoriais_escalaveis_na_criacao_de_interfaces_de_sistemas_para_a_internet.pdf, Novembro.

Smus, Boris. (2013) “Melhoria do desempenho do canvas em HTML5”, http://www.html5rocks.com/pt/tutorials/canvas/performance/, Maio.

Sousa, Kassio. (2011) “HTML”, http://www.connection.ufma.br/Apostila_html_e_xhtml.pdf, Novembro.