STÉPHANO WALLACE SARAIVA E SILVA

AVALIAÇÃO DA ADEQUAÇÃO DA BIBLIOTECA JQUERY UI COM AS RECOMENDAÇÕES DE ACESSIBILIDADE PARA APLICAÇÕES DE INTERNET RICAS

LAVRAS – MG

2013

STÉPHANO WALLACE SARAIVA E SILVA

AVALIAÇÃO DA ADEQUAÇÃO DA BIBLIOTECA JQUERY UI COM AS RECOMENDAÇÕES DE ACESSIBILIDADE PARA APLICAÇÕES DE INTERNET RICAS

Monografia de Graduação apresentada ao Departamento de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação

Orientador
Prof. DSc. André Pimenta Freire

LAVRAS – MG
2013

STÉPHANO WALLACE SARAIVA E SILVA

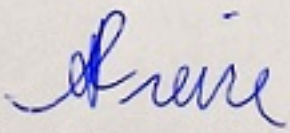
AVALIAÇÃO DA ADEQUAÇÃO DA BIBLIOTECA JQUERY UI COM AS RECOMENDAÇÕES DE ACESSIBILIDADE PARA APLICAÇÕES DE INTERNET RICAS

Trabalho de Conclusão de Curso de Graduação apresentado ao Colegiado do Curso de Bacharelado em Ciência da Computação, para obtenção do título de Bacharel.

APROVADA em 25 de novembro de 2014.

Dr. André Grützmann

Dr. José Monserrat Neto



Dr. André Pimenta Freire (Orientador)

LAVRAS-MG
Novembro/2014

RESUMO

Aplicações de Internet Ricas (RIA) são aplicações Web que possuem características de interatividade comuns com aplicações desktop convencionais. Em muitos dos casos, existem problemas em tais aplicações no tocante à acessibilidade para usuários com algum tipo de deficiência. jQuery é uma biblioteca projetada para simplificar a maneira de se usar JavaScript na criação de websites. Esta possui uma biblioteca voltada especialmente para elementos de interface de usuário chamada jQuery UI, que possibilita por exemplo a criação de widgets e efeitos visuais. O objetivo do trabalho apresentado nesta monografia foi de avaliar a biblioteca jQuery UI e verificar se ela se adequa às especificações da WAI-ARIA (Iniciativa de Acessibilidade Web - Aplicações de Internet Ricas Acessíveis). Para efetuar a avaliação, foi implementado um protótipo de aplicação Web utilizando as linguagens HTML, CSS e componentes de interface da biblioteca jQuery UI. Feito o protótipo, foi verificado que os widgets analisados se comportaram de forma diferente dependendo do navegador e do leitor de tela utilizados. Com os resultados obtidos constatou-se que a biblioteca jQuery UI ainda necessita de melhorias na acessibilidade, sendo que grande parte destas já foram implementadas por terceiros (via alteração do código-fonte ou inserção de plugins) e apenas necessitam ser inseridas à biblioteca principal. Devido a estes resultados, desenvolvedores devem atentar a alguns detalhes ao utilizar a biblioteca jQuery UI, como por exemplo, se o elemento inserido pode ser utilizado via teclado e se fornece *feedback* sonoro ao modificar seu estado.

Palavras-Chave: jQuery; Acessibilidade; Rich Internet Applications.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

1.1 Contexto

Muito progresso tem sido feito nas últimas décadas, devido em grande parte aos avanços da Internet. Criada com inspiração militar na década de 60, mal se sabia naquela época da evolução que aquele sistema de transmissão de pacotes sofreria. Pessoas são capazes de acessar conteúdo proveniente de qualquer parte do planeta a qualquer momento. Atualmente, a internet é utilizada não somente para acessar informações, mas para possibilitar que pessoas contribuam com conteúdo próprio na web, por meio de redes sociais, fóruns, blogs e diversos outros meios, que foram possíveis graças à web. Anteriormente, com a "Web 1.0", o usuário final era um consumidor "passivo" da informação que foi definida por um Webmaster (CORMODE; KRISHNAMURTHY, 2008). Atualmente, com a "Web 2.0", os usuários não somente definem o conteúdo da Web (através de notícias, vídeos, podcasts, imagens), mas também decidem qual conteúdo é mais interessante (PRIMO, 2007).

O crescente envolvimento dos usuários finais com a Web vem encorajando o desenvolvimento de novas tecnologias que melhorem a experiência de navegação na Web. Esta evolução tecnológica trouxe um novo paradigma chamado Aplicações Ricas para Internet (RIA - do inglês *Rich Internet Applications*). RIAs são aplicações Web desenvolvidas para entregar ao usuário características e funcionalidades similiares às que normalmente são associadas com aplicações para desktop. Elas normalmente são executadas em um navegador Web e, na maioria das vezes, não exige instalação de nenhum software adicional no lado cliente para funcionar corretamente (THIESSEN; HOCKEMA, 2010). Com esta introdução da RIA à Web, desenvolvedores precisavam de uma forma de desenvolver aplicações interativas e de poder disponibilizá-las na web. A tecnologia AJAX (do inglês, *Asynchronous JavaScript And XML*), Macromedia Flash e Java preenchem exatamente

essas necessidades. O primeiro destes, AJAX, é uma combinação de JavaScript, XML (do inglês, *EXtensible Markup Language*), XSLT (do inglês, *EXtensible Stylesheet Language*) e DOM (do inglês, *Document Object Model*) que juntas tornam as aplicações Web mais ricas em interatividade. O termo "assíncrono" em seu nome vem da maneira como são feitas as requisições e respostas HTTP entre cliente e servidor (NODA; HELWIG, 2005). Chamadas assíncronas permitem que cliente e servidor se comuniquem sem a necessidade de que a página Web seja recarregada. Outras vantagens que o uso de RIAs proporcionou são:

- Cria uma experiência rica para o usuário, possibilitando que o software possua rolagem suave de página, efeito de zoom, sombreamentos e interfaces com bordas arredondadas;
- Facilita interação física (através de interface multi-toque por exemplo) e áudio/visual;
- Reduz a latência, fazendo com que as aplicações Web respondam de forma mais rápida e ágil;
- Gera equilíbrio entre Cliente/Servidor, pois a carga de processamento entre estes torna-se mais equilibrada, visto que o servidor não necessita realizar todo o processamento e enviar para o cliente. Esta diminuição da carga permite que o servidor possa lidar com mais sessões de clientes concomitantemente.

1.2 Motivação

Com a recente evolução tecnológica, a Internet se tornou uma ferramenta para o entretenimento e para o mundo corporativo. Ela também facilitou atividades do dia-a-dia de seus usuários: é possível realizar transferências bancárias, pagamentos de contas, compras de diversos tipos de produto (inclusive compras em supermer-

cados) sem sair de casa. Apesar desta evolução ter trazido novos usuários, pessoas com deficiência não conseguem usufruir desta tecnologia de forma adequada devido à falta de acessibilidade, seja porque os desenvolvedores não deram a devida atenção, ou porque as ferramentas que não dão o devido suporte.

Segundo Tim Berners-Lee, um dos criadores da Web,"o poder da web está na sua universalidade. O acesso por todos, independentemente da deficiência é um aspecto essencial". Esta universalidade tem se tornado cada vez mais importante, tendo em vista que grande parte da sociedade possui algum tipo de deficiência. Segundo dados do Censo demográfico 2010 (IBGE, 2010), divulgado pelo Instituto Brasileiro de Geografia e Estatística (IBGE), 23,9% da população brasileira possui algum tipo de deficiência, o que resulta em mais de 45,6 milhões de brasileiros. Das pessoas entrevistadas no Censo, a deficiência visual foi a que mais apareceu entre as respostas, chegando a 35,7 milhões de pessoas, como mostra o gráfico na Figura 1.1.

Dentre as pessoas com deficiência visual, tem-se que 506.377 delas não conseguem enxergar de modo algum, 6.056.533 possuem grande dificuladade em enxergar e 29.211.482 possuem alguma dificuldade em enxergar, sendo que neste último caso a pessoa pode não necessitar utilizar algum tipo de tecnologia assistiva para que consiga navegar em páginas Web.

Cada uma das deficiência apresentadas na Figura 1.1 impacta de formas diferentes na maneira com que o usuário navega pelas páginas Web. Por exemplo, pessoas com deficiência visual, necessitam de um software leitor de telas para navegar ou um terminal braille. Pessoas com deficiência auditiva, necessitam de auternativas textuais para conteúdos que possuem áudio. Pessoas com deficiência motora, caso não consigam utizar o mouse como mecanismo principal de navegação precisam recorrer a alternativas como teclado ou ponteiras de cabeça.

A população que reside em áreas rurais também possuem dificuldades que limitam a utilização de páginas Web, como por exemplo, pessoas que desabilitam

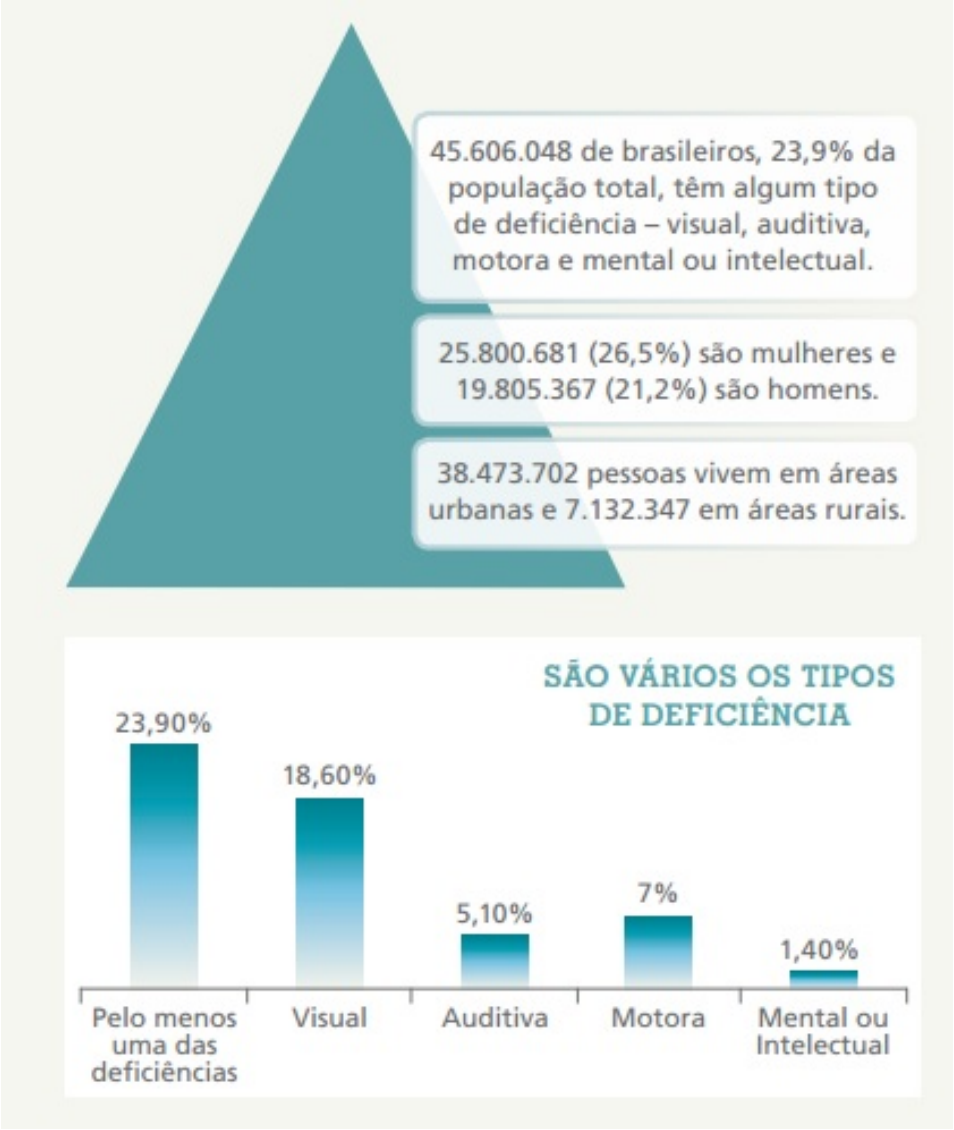


Figura 1.1: População com deficiência no Brasil. Fonte: Extraído da Cartilha do Censo 2010 - Pessoas com Deficiência

imagens nas páginas devido à baixa largura de banda precisam de uma alternativa textual para tais conteúdos.

Visando melhorar a experiência que estas pessoas com deficiência (visual, motora, auditiva) tem com a navegação web, o *World Wide Web Consortuim* (W3C) criou a WCAG (do inglês, *Web Content Accessibility Guidelines*),

que possui recomendações para acessibilidade do conteúdo Web, e a WAI-ARIA (do inglês, *Web Accessibility Initiative - Accessible Rich Internet Applications*) (W3C, 2013) que também especifica como melhorar a acessibilidade de páginas Web, mas voltada para aquelas que utilizam RIA em sua estrutura. Dentro das recomendações WAI-ARIA estão:

- Regras para descrever o tipo de *widget* apresentado, como por exemplo, menu, barra de progresso e árvores de conteúdo;
- Regras para descrever a estrutura da página web, tais como cabeçalhos, tabelas e regiões;
- Propriedades para descrever o estado em que um *widget* se encontra;
- Propriedades para definir regiões dinâmicas da página que são susceptíveis a atualizações, bem como uma política de interrupção para estas (por exemplo, mostrar atualizações críticas em uma caixa de diálogo);
- Propriedades para clicar-e-arrastar, descrevendo a fonte de onde foi clicado e o destino para onde foi arrastado;
- Maneiras de fornecer navegação via teclado.

Estas definições ajudam na navegação de páginas com leitores de tela e outros dispositivos de assistência.

Neste contexto de páginas cada vez mais interativas e com conteúdo dinâmico, foi criada a biblioteca chamada jQuery (THE JQUERY FOUNDATION, 2013), que tem por finalidade simplificar a maneira de se usar javaScript na criação de websites. Essa biblioteca está sendo muito empregada, desde pequenos até grandes Websites, tais como Wordpress (AUTOMATTIC INCORPORATION, 2005) e Wikipedia (WIKIMEDIA FOUNDATION, 2001).

O gráfico da Figura 1.2 (Q-SUCCESS WEB-BASED SERVICES, 2014) compara a utilização da biblioteca jQuery e de outras bibliotecas JavaScript. Se-

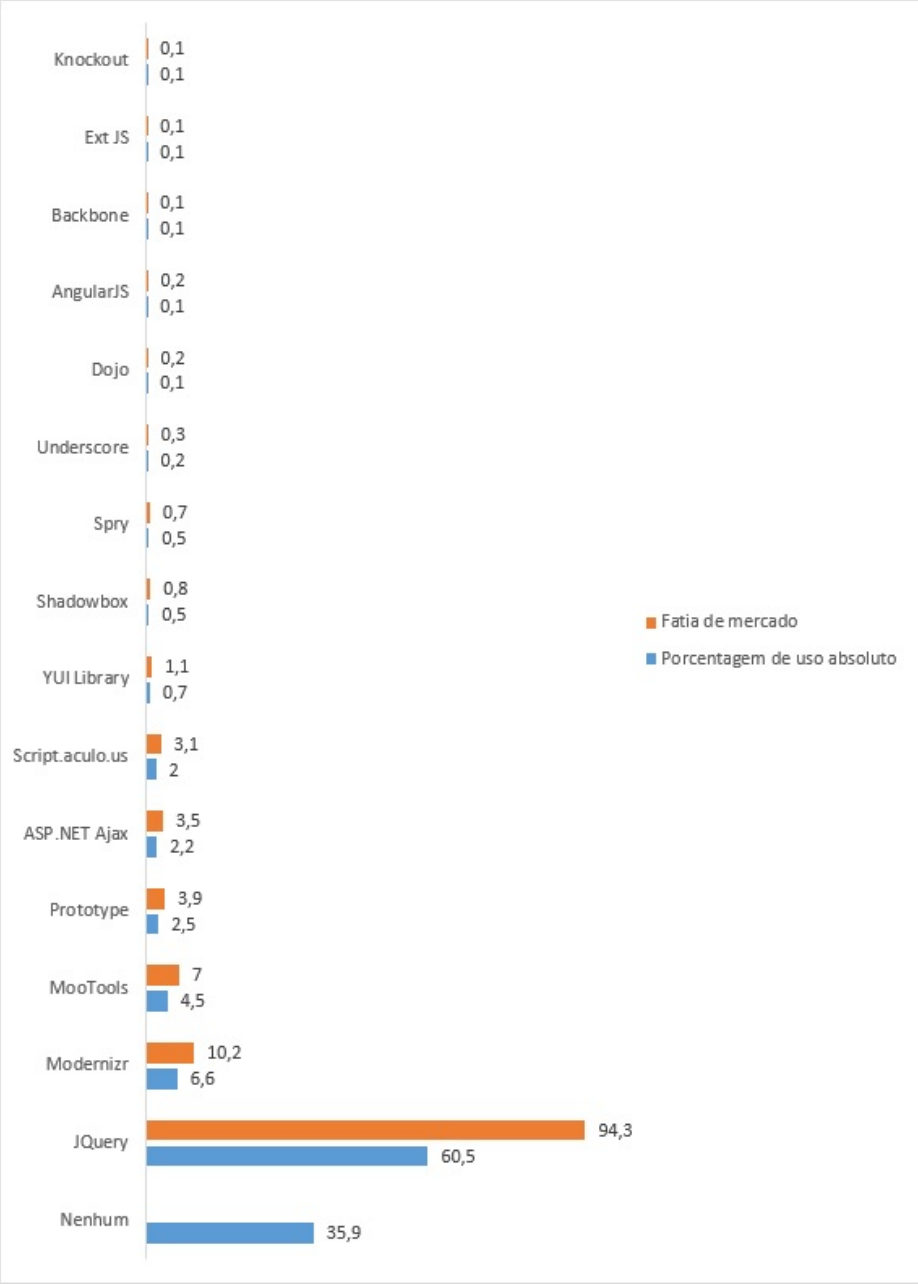


Figura 1.2: Porcentagem de Websites que utilizam diversas bibliotecas JavaScript. Fonte: Extraído do Website w3techs.com

gundo este, 60,5% de todos os websites monitorados utilizam a biblioteca jQuery, o que representa 94,3% da fatia de mercado da biblioteca JavaScript.

A análise desta biblioteca, verificando o quanto ela se adequa às especificações da WAI-ARIA, ajudaria a identificar possíveis falhas e formas de corrigí-las. Também será possível verificar se especificações da WAI-ARIA não são atendidas e como incorporá-las à biblioteca. Este tipo de análise se torna cada vez mais importante tendo em vista a quantidade de usuários com deficiência que podem ter acesso à internet e a importância que jQuery tem exercido sobre conteúdos dinâmicos em páginas Web.

1.3 Objetivos

A proposta do presente trabalho foi a de analisar a adequação de componentes de interface gerados pela biblioteca jQuery UI quanto aos requisitos de acessibilidade, por meio da implementação de um protótipo de aplicação RIA e análise do seu funcionamento com softwares leitores de tela utilizados por usuários cegos, seguidos de avaliações de acordo com *guidelines* do W3C.

Os objetivos específicos a serem alcançados nesta monografia são:

- Efetuar um estudo sobre acessibilidade em RIA;
- Análise da biblioteca jQuery UI, identificando elementos ARIA na implementação de seus componentes;
- Análise dos leitores de tela NVDA e Jaws, verificando qual o tipo de suporte que estes oferecem para Aplicações de Internet Ricas Acessíveis e como eles se comportam com tais componentes em um cenário real;
- Construção de um protótipo como prova de conceito utilizando componentes da biblioteca jQuery UI;
- Efetuar testes da aplicação implementada com leitores de tela para usuários cegos;
- Analisar a implementação de acordo com critérios do W3C-ARIA.

2 REFERENCIAL TEÓRICO

2.1 Web 2.0 e Aplicações Ricas para Internet - Conceitos e Tecnologias

O termo Web 2.0 foi criado em um *brainstorming* em uma conferência entre Tim O'Reilly e Daly Dougherty em junho de 2004. Essa foi definida como sendo *"os padrões de design e modelos de negócio para a próxima geração de software"* (VOSSEN; HAGEMANN, 2007). Com o passar dos anos, fica cada vez mais claro o quanto a forma como a Web é utilizada tem mudado. Atualmente, a Web é vista como uma plataforma onde seus usuários colaboram com conteúdo e o consomem. Um grande grupo de pessoas pode criar um trabalho coletivo cujo valor ultrapassa significativamente aquele provido por qualquer um de seus colaboradores individualmente. A Web tornou-se uma imensa fonte de conteúdo. Existem mercados como eBay e Mercado Livre onde usuários podem comprar e vender praticamente qualquer tipo de produto, coleções de mídia como YouTube e Flickr onde usuários compartilham vídeos, áudios e fotos, e redes sociais como Facebook e Twitter onde pessoas interagem entre si através de mensagens de texto, vídeos e fotos. Minsk *et al.* (2007) definem a Web 2.0 através de Conceitos e Tecnologias, algumas destas são apresentados na Tabela 2.1.

Tabela 2.1: Web 2.0: Conceitos e Tecnologias

Conceitos Web 2.0	**Tecnologias Web 2.0**
Aplicações Ricas para Internet (RIA)	Ajax
Software como um serviço	RSS
Inteligência Coletiva	Microsoft Ajax
	Blogs
	Wikis

2.1.1 Aplicações Ricas para Internet (RIA)

OReilly e Battelle (2009) apresentaram uma visão mais abrangente sobre a Web 2.0, colocando-a de forma mais específica através das Aplicações Ricas para Internet. Nesta, a experiência de navegação do usuário se torna mais dinâmica, rica e interativa, tudo isto graças à utilização de clientes "gordos". A utilização destes faz com que parte da computação seja transferida para o lado do cliente, reduzindo significativamente o processamento de dados no lado servidor.

Com a chegada das RIAs, a utilização de *scripts* tornou-se cada vez mais comum. AJAX (do inglês, *Asynchronous JavaScript And XML*), por exemplo, possibilita a realização de atualizações parciais da tela e comunicação assíncrona. Este modelo separa a interação do usuário com o servidor, atualizando apenas os elementos da interface que contêm novas informações (ROLLETT *et al.*, 2007). Esta arquitetura mais eficiente elimina a espera para que os usuário continuem trabalhando. Além disto, o consumo de largura de banda da rede é reduzido, juntamente com a sobrecarga do servidor, o que melhora a performance e escalabilidade do sistema. A Figura 2.1 ilustra as arquiteturas com e sem utilização de AJAX.

Do ponto de vista da arquitetura de software, a diferença mais significativa entre uma aplicação AJAX e uma aplicação Web HTML clássica é a introdução de uma *engine* no lado cliente. Esta *engine*, que é executada dentro do navegador Web, funciona como um intermediário entre a aplicação de interface do usuário e o servidor. Atividades do usuário levam a chamada à engine do lado cliente ao invés de uma solicitação da página para o servidor. Da mesma forma, a transferência de dados ocorre entre o servidor para a engine do lado cliente, ao invés de diretamente para o navegador Web.

A *engine* AJAX é a chave do seu modelo de aplicação. Sem ela, todo evento gerado pela atividade do usuário deve voltar para o servidor para processamento.

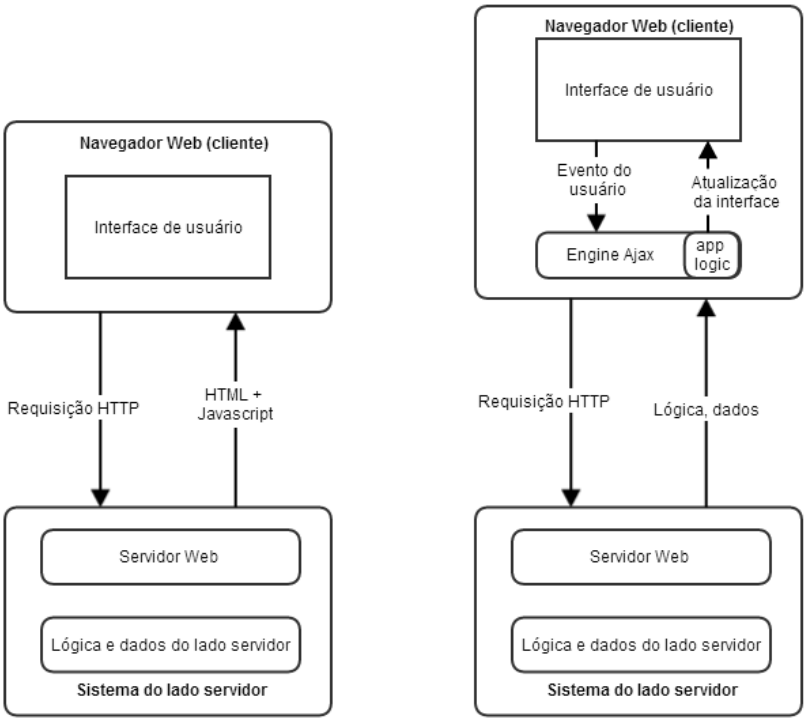


Figura 2.1: Arquitetura clássica de uma aplicação Web (à esquerda) e arquitetura AJAX (à direita)

O conteúdo da página é modificado utilizando-se a tecnologia DOM (HÉGARET, 2005) (do inglês, *Document Object Model*) e Javascript baseados no resultado dos processos de *download* e *upload*. DOM é uma representação em forma de árvore para documentos HTML (PEMBERTON, 2007) e XML (QUIN, 2003), onde tags são nós internos da árvore, e textos ou hiperlinks para outras árvores são os nós folha (LIN; HO, 2002).

Alguns exemplos de sites populares que utilizam AJAX são:

- Sugestão de pesquisa do Google;
- Chat do Facebook;
- Ferramenta de email Gmail;

- Mensageiro instantâneo Meebo.

A criação do AJAX é de grande importância para a elaboração de Aplicações de Internet Ricas. A nova forma com que o usuário interage com cada um destes sites tornou a navegação Web muito mais prática e intuitiva. Diversas novas tecnologias surgiram após a criação das RIAs - o que antes era um ambiente limitado por conteúdo estático e com pouca interação por parte do usuário, tornou-se algo muito mais dinâmico e interativo e inúmeras possibilidades agora são possíveis, a exemplo de:

- **RSS** (do inglês, *Really Simple Syndication*): Tem sido amplamente utilizado por grandes companias de mídia através de um método baseado em assinatura, onde o cliente recebe notícias em tempo real através de *feeds*.

- **Software como um Serviço**: Trata-se um modelo de negócios que visa entregar o software ao usuário final como um serviço ao invés de um pacote de software, que requer uma instalação local. Sua ideia é reduzir a necessidade de atualizações de software ou manutenção no lado cliente, deixando essas atividades para o servidor, o que torna tais operações transparentes para o usuário final. Uma de suas grandes vantagens é evitar a necessidade do usuário final ter que comprar um pacote de software completo, pagando por funcionalidades que não serão usadas, neste o usuário paga pelo que utiliza ou é gratuito.

- **Inteligência Coletiva**: Baseado na geração de conteúdo através da contribuição do conhecimento de diversas pessoas a respeito de determinado assunto. Isto pode ser definido como uma rede de inteligência coletiva, *um agregado de diversas iterações de contribuições individuais através de uma rede de computadores* (LINDEN; FENNA; DRAKOS, 2005). Existem na Web diversas tecnologias que aplicam o conceito de inteligência coletiva,

como por exemplo blogs e Wikis, onde o usuário é livre para expressar sua opinião e analisar produtos e serviços.

2.2 Acessibilidade Web

A Web fornece a seus usuários uma extensa quantidade de conteúdo, e sua população de usuários se torna cada vez mais diversa, sendo esta composta por pessoas de todas as idades, níveis educacionais e níveis de experiência no manuseio de computadores (SHNEIDERMAN, 2004). Muitos destes usuários possuem algum tipo de deficiência, seja ela motora, perceptual e/ou cognitiva. Para que a navegação Web seja possível, usuários com deficiência utilizam diversos tipos de recursos de Tecnologia Assistiva (hardware e/ou software), que podem ser:

- Equipamentos de entrada e saída (síntese de voz, Braille);
- Auxílios alternativos de acesso (ponteiras de cabeça, de luz);
- Teclados modificados ou alternativos;
- Softwares especiais (por exemplo, os de reconhecimento de voz).

Dentre os recursos de Tecnologia Assistiva mais utilizados por pessoas com deficiência visual estão os leitores de telas, que são softwares que identificam e interpretam o que está sendo enviado para a saída padrão. Esta interpretação é então enviada para um sintetizador de voz ou para um terminal Braille e reapresentada para o usuário cego.

Dentre as tarefas que podem ser realizadas por um leitor de tela via teclado estão: ler um documento, navegar em páginas web, abrir e fechar arquivos, entre outros. Leitores de tela estão disponíveis para os principais sistemas operacionais utilizados atualmente (Windows - ex.: NVDA, Jaws, Serotek, Window Eyes, Virtual Vision | Linux - ex.: Orca e Emacspeak | Mac OS - ex.: Apple VoiceOver).

Acessibilidade Web tem por foco principal fazer com que pessoas sejam capazes de adquirir e utilizar conteúdo Web. Independente de se o usuário final possua uma deficiência ou não, páginas Web devem ser desenhadas de tal forma que estes usuários consigam utilizá-las da forma que lhes for mais conveniente. Para que estas páginas sejam realmente consideradas acessíveis, elas devem ser flexíveis o suficiente para serem utilizadas por qualquer tipo de tecnologia assistiva (SLATIN; RUSH, 2002). O usuário deve ter uma experiência de navegação agradável, sem barreiras o impedindo de alcançar seu objetivo, seja ele qual for.

O foco principal da acessibilidade Web é o acesso por pessoas com algum tipo de deficiência. Porém, abrangendo o escopo da acessibilidade, pessoas sem deficiências também são beneficiadas (HENRY, 2006). O que para algumas pessoas é visto como um diferencial que melhora a experiência de navegação, para outras é uma necessidade essencial, que as tornam capazes de acessar o conteúdo da página Web.

2.2.1 Leitores de tela

Um leitor de tela é um software que possibilita pessoas com diferentes tipos de deficiência visual a utilizar computadores. Leitores de tela trabalham juntamento com o sistema operacional do computador para fornecer informações sobre ícones, menus, caixas de diálogo, arquivos e pastas.

Existem duas maneiras com que um leitor de tela pode fornecer *feedback* ao usuário:

- Fala - Leitores de tela utilizam uma *engine* texto-para-fala (TTS) para traduzir a informação contida na tela em um discurso, que pode ser ouvido através de fones de ouvido ou alto-falantes;

- Braille - Um hardware externo, conhecido como Display Braille é necessário para isto. Um display Braille contém uma ou mais linhas de células. Cada célula é constituída por uma série de pontos similares a um layout de peça

de dominó. À medida que a informação na tela do computador se altera, os caracteres no Display Braille também se alteram, fornecendo informações atualizadas diretamente do computador.

Devido ao fato de que a maioria dos usuários de leitores de tela não utilizam mouse, os leitores de tela utilizam uma extensa variedade de comandos via teclado para realizar diferentes tarefas (por exemplo ler documentos, navegar em páginas Web, abrir e fechar arquivos ou ouvir músicas).

Partindo do princípio de que páginas Web são construídas utilizando código bem estruturado, leitores de tela são capazes de interagir com estas mais facilmente. Um leitor de tela lê o código da página e tornam disponíveis alguns comandos via teclado, estes comandos serão diferentes de acordo com a estrutura que o usuário está focando (por exemplo tabelas, listas ou widgets).

2.2.2 Modelos de Acessibilidade

Atualmente, desenvolvedores podem seguir *guidelines*, que consistem em conjuntos de recomendações a serem considerados para que o processo de acessibilidade de páginas Web seja conduzido de forma padronizada e de fácil implementação.

A WCAG 2.0 (Web Content Accessibility Guidelines), um dos conjuntos de guidelines mais relevantes desenvolvido pela Web Accessibility Initiative (WAI) (W3C, 2003) do World Wide Web Consortium (W3C) foi lançada em 2008, trazendo guias de acessibilidade atualizadas com as técnicas de desenvolvimentos da Web moderna. A WCAG 2.0 consiste em 4 princípios:

- **Princípio 1 -** Perceptível: Informação e componentes da interface de usuário devem ser apresentáveis aos usuários da forma que eles conseguem perceber;
- **Princípio 2 -** Operável: Componentes da interface de usuário e navegação devem ser operáveis;

- **Princípio 3 -** Compreensível: Informação e operação da interface de usuário devem ser compreensíveis;
- **Princípio 4 -** Robusto: Conteúdo deve ser robusto o suficiente para que possa ser interpretado de forma confiável por uma extensa variedade de agentes de usuário, incluindo tecnologias assistivas.

Estes princípios são quebrados em 12 guidelines, que seguem:

- ***Guideline* 1.1 -** Alternativas textuais: Prover alternativas textuais para qualquer conteúdo não-textual;
- ***Guideline* 1.2 -** Mídia baseada em tempo: Prover alternativas para mídias baseadas em tempo;
- ***Guideline* 1.3 -** Adaptável: Criar conteúdo que pode ser apresentado de diferentes maneiras;
- ***Guideline* 1.4 -** Distinguível: Facilitar a forma com que usuários vêem e ouvem ao conteúdo;
- ***Guideline* 2.1 -** Acessibilidade via teclado: Tornar todas as funcionalidades disponíveis via teclado;
- ***Guideline* 2.2 -** Tempo suficiente: Prover para o usuário tempo suficiente para ler e utilizar o conteúdo;
- ***Guideline* 2.3 -** Convulsões: Não desenhar a página Web de uma forma que já é conhecida por causar convulsões;
- ***Guideline* 2.4 -** Navegável: Prover formas de ajudar o usuário a navegar;
- ***Guideline* 3.1 -** Legível: Fazer com que o conteúdo textual seja legível e compreensível;

- ***Guideline* 3.2 -** Previsível: Fazer com que páginas Web apareçam e sejam operadas de formas previsíveis;
- ***Guideline* 3.3 -** Assistência de entrada: Ajudar usuários a evitar erros na entrada de dados e se caso erros sejam cometidos, ajudá-los a corrigí-los;
- ***Guideline* 4.1 -** Compatível: Maximizar a compatibilidade com agentes de usuários atuais e futuros.

Cada guideline é dividida em um número de critérios de sucesso. São 61 critérios de sucesso ao todo, sendo que cada um possui uma série de técnicas de sucesso que lhes são atribuídos. A WCAG 2.0 define três níveis de critérios de sucesso, que são:

- **Nível A**

  Exigência básica de acessibilidade, se não forem cumpridas, poderão fazer com que grupos de usuários sejam impedidos de acessar as informações do documento;

- **Nível AA**

  Equivale às normas e às recomendações de acessibilidade que garantem o acesso às informações do documento. Se não forem cumpridas, grupos de usuários terão dificuldades para navegar e acessar os conteúdos;

- **Nível AAA**

  Garantindo um nível de acessibilidade maior que os anteriores, caso este nível seja alcançado, o acesso aos documentos publicados na Web será facilitado. Se essas regras não forem cumpridas, grupos de usuários poderão enfrentar dificuldades para acessar as informações dos documentos publicados.

O Brasil também possui seu modelo de acessibilidade, chamado Modelo de Acessibilidade em Governo Eletrônico (e-MAG), o qual foi desenvolvido pela

equipe do Departamento de Governo Eletrônico (GOVERNO FEDERAL, 2005). Este, que já se encontra em sua versão 3.1, apoia-se na WCAG 2.0 e considera as novas pesquisas de acessibilidade à Web. Apesar de utilizar a WCAG como referência, o e-MAG 3.0 foi desenvolvido e pensado para as necessidades locais, visando atender as prioridades brasileiras.

Além do e-MAG, o governo brasileiro também fornece gratuitamente o software Avaliador e Simulador de Acessibilidade de Sítios (ASES), que se encontra em fase beta. O software permite avaliar, simular e corrigir a acessibilidade de páginas, Websites e portais, sendo de grande valia para os desenvolvedores e publicadores de conteúdo (GOVERNO FEDERAL, 2009).

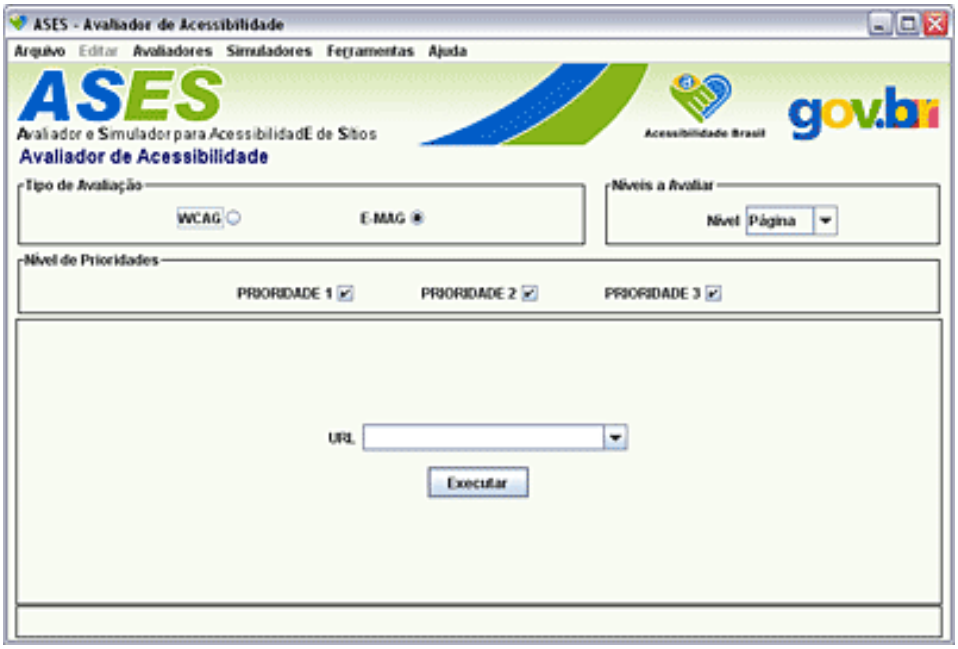


Figura 2.2: Tela inicial do software ASES

O software ASES possui as seguintes funcionalidades:

- Avaliador de acessibilidade (e-MAG e WCAG);
- Avaliador de CSS;
- Avaliador de HTML (4.01 e XHTML);

- Simuladores de leitor de tela (exibe o menor tempo necessário para leitura de qualquer texto em uma página) e Baixa visão (simula diversas doenças da visão, como: daltonismo, miopia, catarata);

- Ferramenta para selecionar o DocType, conteúdo alternativo, associador de rótulos, links redundantes, corretor de eventos e preenchimento de formulários.

Além do ASES, existem outras ferramentas que avaliam a acessibilidade de páginas Web. A W3C apresenta em sua página [1] uma extensa lista com diversas ferramentas para avaliação de acessibilidade. Dentre estas, algumas merecem destaque por conter o idioma português como opção, são estas:

- **Colour Contrast Analyser** (JUN; AUSTRALIA; FAULKNER, 2003)
  Verifica combinações do primeiro plano e plano de fundo da página para determinar se eles apresentam boa visibilidade de cores. Também possui a função de criar simulações de certas deficiências visuais, como o daltonismo. A relação de contraste ajuda a determinar se o contraste entre duas cores pode ser bem percebido por pessoas com deficiências visuais.

- **Hera** (BENAVíDEZ; RESTREPO; MCCATHIENEVILE, 2003)
  HERA é uma ferramenta para avaliar a acessibilidade das páginas Web de acordo com as recomendações das Diretrizes de Acessibilidade para o Conteúdo Web 1.0 (WCAG 1.0). O HERA efetua uma análise automática prévia da página e disponibiliza informações dos erros encontrados (detectáveis de forma automática) e quais os pontos de verificação que devem ser revistos manualmente. A ferramenta também disponibiliza um formulário que permite modificar os resultados automáticos, inserir comentários a cada um dos pontos de verificação e indicar o nome do revisor. Também é possível gerar

[1] `http://www.w3.org/WAI/ER/tools/complete`

um relatório final sobre a revisão, para imprimir ou descarregar, em diversos formatos: (XHTML, RDF e PDF).

- **daSilva** (ACESSIBILIDADE BRASIL, 2006)
  Primeiro avaliador de acessibilidade em português para Websites, detecta um código HTML e faz uma análise do seu conteúdo, verificando se este dentro ou não do conjunto de regras de acessibilidade do WCAG e e-GOV.

Ferramentas que avaliam a acessibilidade de Websites são muito úteis durante o processo de desenvolvimento de páginas Web, elas ajudam o desenvolvedor a encontrar erros e, às vezes, o mostra possíveis soluções ou dicas. Porém, apesar de úteis, estas ferramentas não são perfeitas, vários testes ainda devem ser realizados manualmente, como por exemplo, verificar se um texto alternativo descrito em um atributo 'alt' é equivalente e apropriado à imagem a qual ele referencia. Com isso, os selos de acessibilidade fornecidos por estas ferramentas de avaliação não são garantia de acessibilidade, problemas ainda podem ser encontrados pelos usuários. Do mesmo modo que a falta do selo não impede que determinada página Web seja acessível.

2.3 Acessibilidade em Aplicações Ricas para Internet

O desenvolvimento de novas tecnologias requerem mais do que simples vigilância para assegurar que aspectos de acessibilidade sejam incluídos. Novas tecnologias trazem novos paradigmas, algumas vezes de formas inesperadas, trazendo novos tipo de desafios para acessibilidade (COOPER, 2007). Interfaces de usuário RIA introduziram dois novos elementos às tradicionais aplicações Web (LINAJE *et al.*, 2010):

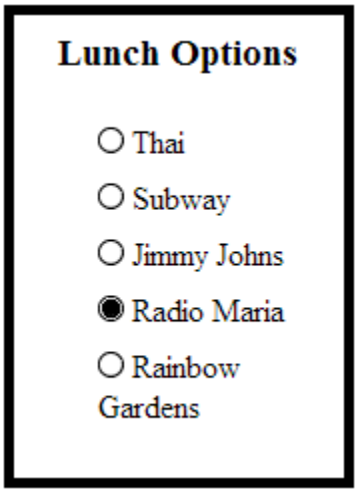
- Widgets, que são controles customizados que adicionam diversas novas operações dentro de uma página Web que antes não eram possíveis com o uso de HTML;

- Comunicação assíncrona, que permitem fazer requisições a um servidor sem a necessidade de recarregar a página Web por completo.

Como parte da solução para Web 2.0, a W3C Web Accessibility Initiative está desenvolvendo um conjunto de especificações para ARIA (Acessibilidade em Aplicações Ricas para Internet). Descrito em um roteiro para RIA (Aplicações Ricas para Internet) (W3C, 2013), são apresentadas tecnologias para mapeamento de controles, Ajax *live regions* e eventos para APIs de acessibilidade, incluindo controles customizados usados para RIA, assim como técnicas para marcações comuns para estrutura Web como menus, conteúdo primário, conteúdo secundário, etc. As especificações técnicas são apresentadas em duas partes: Papéis (*'roles'*) para RIA e Estados e Módulos de Propriedades para RIA. Marcações ARIA foram desenvolvidas para inserir no código HTML informações úteis para recursos de Tecnologia Assistiva. É uma forma de rotular controles e apresentar informações sobre seus estados:

- ARIA *tabindex*: Recursos de Tecnologias Assistivas não conseguem obter controle do *widget* ao navegar por abas (ou níveis diferentes)de conteúdo para permitir que usuários utilizem unicamente o teclado. Para resolver isto, o WAI-ARIA introduziu uma extensão ao uso da propriedade tabindex, que já estava disponível em HTML. Atribui-se o valor 0 para tabindex do descendente ativo em um widget enquanto o valor -1 é atribuido para os outros elementos filhos do *widget*, à medida que o usuário navega entre os itens, os itens antigos recebem o valor -1 para o atributo tabindex e o novo item recebe o valor 0 (exemplo na Figura 2.3);

- ARIA *live regions*: Quando parte da interface era atualizada (um div, uma área de tabela, um parágrafo, etc.), um recurso de Tecnologia Assistiva não conseguia capturar tal ação e por consequência, não conseguia notificar o usuário sobre a mudança. Em versões anteriores, softwares como leitores




```
 //trecho de código HTML
 //...
 <li id="r3"
tabindex="-1"
role="radio"
aria-checked="false">
    <img role="presentation" src="images/radio-unchecked.gif" alt=""/>
    Jimmy Johns
 </li>
 <li id="r4"
       tabindex="0"
       role="radio"
       aria-checked="true">
    <img role="presentation" src="images/radio-checked.gif" alt=""/>
    Radio Maria
 </li>
 <li id="r5"
       tabindex="-1"
       role="radio"
       aria-checked="false">
    <img role="presentation" src="images/radio-unchecked.gif" alt=""/>
    Rainbow Gardens
 </li>
</ul>
//...
```

Figura 2.3: Comando mostrando funcionamento do atributo tabindex, onde a opção “Radio Maria” está selecionada (com valor de tabindex = 0)

de tela só esperavam por novos conteúdos quando uma nova página era carregada. Para resolver isto, a WAI-ARIA introduziu as *live regions*, com

diversos atributos que podem ser associados a qualquer widget, especificando quando o widget está apto para atualizar (totalmente, parcialmente, ou nunca, dependendo do valor atribuido à propriedade específica);

- ARIA *labelledby*: Utilizado para indicar os IDs dos elementos que são rotulados para o objeto. É utilizado para estabelecer um relacionamento entre widgets ou grupos em seus rótulos. Usuários de Tecnologia Assistiva frequentemente navegam pelas áreas de uma página pressionando a tecla tab. Se um rótulo não for associado com um elemento de entrada, widget ou grupo, este então não será lido pelo leitor de tela.

- ARIA *landmark roles*: A utilização deste em Aplicações de Internet Ricas é mais complicada do que na Web tradicional no sentido de diferenciar seções de navegação na mesma página devido às diferenças em sua especificação de marcação. Visando isto, WAI-ARIA cria o atributo *landmark role*. Este atributo diferencia em uma página, por exemplo, o que nela é aplicação e o que nela é documento estático. Isto facilita a navegação do usuário pela página, apontando o foco inicial da página ao conteúdo que realmente é importante e permitindo localizar seções de forma mais rápida (exemplo na Figura 2.4);

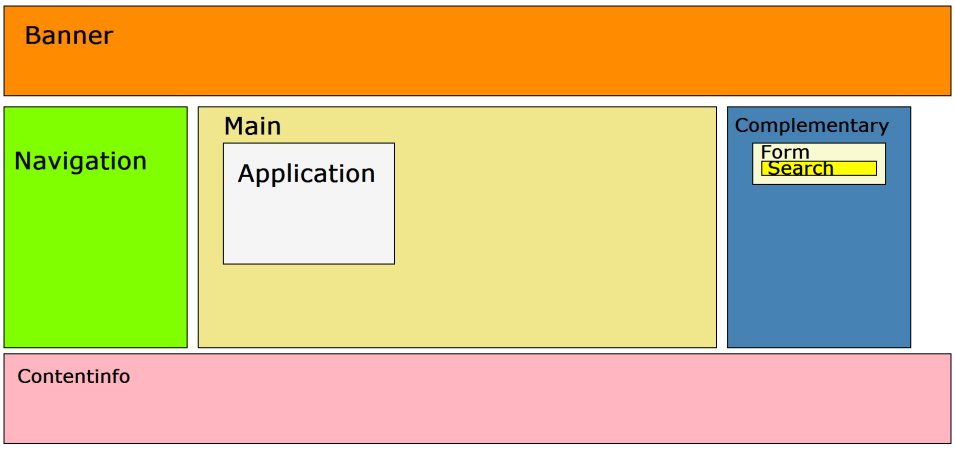


Figura 2.4: Exemplo de estrutura de uma página através de landmarks

- ARIA *states*: Apresenta qual o estado corrente dos elementos na página Web (por exemplo, se um *checkbox* está marcado ou não, ou se um *widget* está escondido ou desativado). Esses estados são inseridos no código através da inserção de atributos nos elementos da página, como por exemplo "*aria-busy*", "*aria-disabled*" ou "*aria-pressed*";

- ARIA *relationship*: Apresenta qual o relacionamento entre os elementos da página (por exemplo, se um elemento é parte de um grupo, ou se ele é rotulado por algum texto);

- ARIA *drag and drop*: Quando utilizado, recursos de Tecnologia Assistiva podem identificar se um elemento pode ser arrastado, se ele foi selecionado (segurado) ou se ele foi desselecionado (solto).

A suíte ARIA é especialmente desenvolvida como uma ponte entre tecnologias já estabelecidas e tecnologias emergentes. WAI-ARIA cataloga a semântica atualmente compreendida por tecnologias assistivas e fornece mecanismos para anexar novas semânticas ao conteúdo Web. Para isto, grupos de trabalho da *Web Accessibility Initiative* estão constantemente pesquisando por mudanças ocorridas na tecnologia atual, essa pesquisa é importante para descobrir problemas em novas tecnologias antecipadamente, e então desenvolver métodos efetivos de resolver estes problemas.

Tratando-se de leitores de tela, os navegadores Web desempenham um grande papel na interpretação destas marcações ARIA. A maioria dos navegadores suportam algum tipo de API (*Application Programming Interface*) de acessibilidade, e os recursos de Tecnologia Assistiva utilizam esta API para recolher informações sobre o que é apresentado na tela. Isto significa que o suporte de leitores de tela a ARIA depende de qual navegador Web está sendo utilizado.

Diversos leitores de tela já oferecem uma extensa lista de comandos via teclado para utilização de widgets, suporte para interpretação de *Live Regions, roles e states* ARIA.

2.4 A Biblioteca jQuery

A World Wide Web de hoje é um ambiente dinâmico, e seus usuários estão cada vez mais exigentes em relação às funções e ao estilo que um site deve possuir. Visando construir sites mais interessantes e interativos, desenvolvedores têm procurado por bibliotecas JavaScript como jQuery para simplificar tarefas que antes eram complicadas e automatizar tarefas comuns.

jQuery é uma biblioteca JavaScript de código aberto que simplifica a interação entre HTML e JavaScript, manipulação de eventos, animações e interações Ajax para um desenvolvimento Web rápido. Foi criada por John Resig em 2005 (RESIG, 2005) e lançada em janeiro de 2006.

A bilioteca jQuery tem se tornado cada vez mais popular devido à sua habilidade de auxiliar em uma extensa quantidade de tarefas. No ano de 2013, 69.6% dos 10.000 Websites mais visitados no mundo utilizavam a biblioteca. Em 2012, o resultado da pesquisa era de 54.7% e em 2011 o resultado ficou pouco abaixo dos 40% (BUILTWITH PTY LTD, 2013).

Chaffer e Swedberg (2011) apresentam algumas características que tornaram a bilioteca jQuery popular:

- **Acesso a elementos em um documento:** Sem a utilização de uma biblioteca JavaScript, desenvolvedores Web precisariam escrever diversas linhas de código para examinar a árvore DOM (que é uma multi-plataforma que representa como as marcações em HTML, XHTML e XML são organizadas e lidas pelo navegador Web) e localizar porções específicas da estrutura do documento HTML. jQuery possui um robusto e eficiente mecanismo seletor,

tornando mais fácil recuperar porções exatas do documento que necessitam ser inspecionadas e manipuladas. Figura 2.5;

```
<script>
   $( "p" ).find( "span" );
</script>
```

Figura 2.5: Código para busca de elemento em uma página

- **Modificar a aparência de uma página Web:** CSS oferece um método poderoso de alterar a forma com que um documento é renderizado, mas falha quando navegadores Web não dão suporte completo a seu padrão. jQuery conta com os mesmos padrões de suporte na maioria dos navegadores, além de poder alterar classes ou propriedades individuais de estilo aplicadas a uma porção do documento mesmo após a página ter sido renderizada. Figura 2.6;

```
<script>
   $( "p" ).last().addClass( "selected" );
</script>
```

Figura 2.6: Código para inserção de classe a determinado elemento (ou conjunto de elementos)

- **Alterar o conteúdo de um documento:** Textos podem ser alterados, imagens podem ser inseridas ou trocadas, listas podem ser reordenadas, ou uma estrutura inteira de um código HTML pode ser reescrita a extendida. Figura 2.7;

```
<script>
   $( "p" ).append( "<strong>Hello</strong>" );
</script>
```

Figura 2.7: Código para inserção de código HTML ao final do elemento

- **Responder a interações de usuário:** jQuery oferece uma forma elegante de interceptar uma extensa variedade de eventos, como o usuário clicar em um

link, sem a necessidade de desordenar o código HTML como manipuladores de evento. Ao mesmo tempo, sua API manipuladora de eventos remove inconsistências no navegador. Figura 2.8;

```
<script>
   $( "p" ).click(function() {
      $( this ).slideUp();
   });
</script>
```

Figura 2.8: Código que insere uma animação ao elemento clicado

- **Mudanças animadas sendo feitas em um documento:** jQuery oferece uma extensa variedade de efeitos, tais como desvanecer, limpar, esconder, alternar, entre outros (Figura 2.9);
- **Recuperar informação de um servidor:** esta atividade agora pode ser realizada sem a necessidade de atualizar a página completamente de forma rápida e fácil, permitindo que desenvolvedores foquem na funcionalidade do lado servidor;
- **Simplifica tarefas JavaScript comuns:** a biblioteca provê melhorias nas construções basicas JavaScript, tais como iterações e manipulações de listas.

2.4.1 jQuery UI

jQuery UI é um conjunto de interações de interface de usuário, efeitos, widgets e temas contruídas sobre a biblioteca JavaScript jQuery. A bilioteca jQuery UI foi criada a partir da união de diversos plugins da biblioteca jQuery voltado para gerenciamento da interface de usuário. Estes plugins ajudam a facilitar a interação com o usuário, e estas interações se tornaram mais simples de gerenciar utilizando a biblioteca jQuery UI.

Assim como a biblioteca jQuery, jQuery UI também é um *software* de código aberto distribuído pela *jQuery Foundation* sob a *MIT License*.

```
//trecho de código HTML
<div id="clickme">
   Click here
</div>
<img id="book" src="book.png" alt="" width="100" height="123">
//fim de trecho de código HTML

<script>
   $( "#clickme" ).click(function() {
      $( "#book" ).slideDown( "slow", function() {
      // Animation complete.
      });
   });
</script>
```

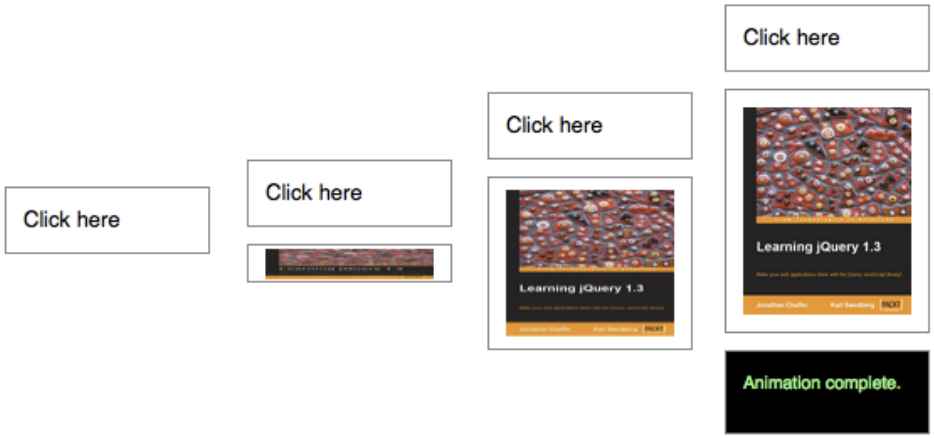


Figura 2.9: Comando que gera animação sobre a imagem quando o elemento é clicado.
Fonte: http://api.jquery.com/animate/

```
<script>
   $( "#result" ).load( "ajax/test.html" );
</script>
```

Os elemento contidos na biblioteca jQuery UI podem ser divididos em três categorias, que são:

- **Interações:** Fornece um conjunto de interações baseadas na utilização de mouse como “blocos” de interfaces ricas e widgets complexos. Dentre as principais interações estão as de mover widgets pela tela, arrastar e soltar

elementos da página e redimensionamento. Por estas interações serem feitas somente via mouse, acaba acarretando em um baixo nível de acessibilidade.

- **Widgets:** São recursos ricos que podem ser adicionados às páginas Web. Dentre os principais Widgets estão botões, caixas de diálogo, menus e barras de progresso. A biblioteca fornece um utilitário chamado *jQuery factory*, onde o desenvolvedor pode criar widgets customizados com o mesmo nível de abstração que qualquer outro elemento da biblioteca.

- **Efeitos:** Voltados para a parte visual da página, os efeitos fornecem diversos tipos de animações para os elementos desta, como mudança de cor, esconder, aparecer, dar destaque, entre outros.

2.4.2 jQuery e Acessibilidade

A Fundação jQuery possui um forte envolvimento com acessibilidade. Seu objetivo é fazer com que desenvolvedores consigam alcançar todos os usuários de conteúdo Web, incluindo aqueles com algum tipo de deficiência visual ou motora.

Por ser uma biblioteca de código aberto, usuários podem contribur com seu desenvolvimento, incluindo novas funcionalidades e/ou aprimorando outras, que são analisadas pela equipe principal, que decide se os novos trechos de código criados serão incluídos à biblioteca, gerando uma nova versão. Alguns widgets que já possuem WAI-ARIA em sua implementação são: efeitos de acordeão, caixas de diálogo e barras de progresso. Atributos ARIA estão sendo incorporados aos widgets da biblioteca desde sua versão 1.7, lançada em 2008.

Usuários também podem desenvolver plugins, que são distribuídos para toda comunidade e podem ser adquiridos no site oficial da biblioteca (JQUERY FOUNDATION, 2005).

Porém, apesar do envolvimento da Fundação jQuery, muito trabalho ainda precisa ser feito em se tratando de acessibilidade. É fácil encontrar plugins que tentam empregar melhor acessibilidade, mas não conseguem, possuindo diversos

bugs que acabam atrapalhando a experiência do usuário. Isto faz com que muitos desenvolvedores simplesmente desistam de tentar empregar acessibilidade a suas páginas Web e procurem por soluções mais simples e práticas.

2.5 Trabalhos Relacionados

Nesta seção, são apresentados trabalhos relacionados que investigaram a acessibilidade para Aplicações Ricas para Internet, mostrando seu funcionamento e o que ainda precisa ser melhorado a respeito.

Pfeiffer e Parker (2009) descrevem implementações existentes de acessibilidade para o elemento <video> em HTML5, utilizando também a biblioteca jQuery. Sabe-se que o elemento <video> está sendo cada vez mais implementado em navegadores Web, sendo que diversos navegadores já possuem suporte para ele. Porém, o mecanismo para inserção de dados para acessibilidade (como legendas simples e captions) ainda não possuem uma solução ideal definida.

O objetivo de seu trabalho foca em delinear casos de uso tanto para a inclusão de dados de acessibilidade dentro de um arquivo de vídeo, quanto para para a distribuição de dados de acessibilidade como um arquivo separado. Após serem apresentados alguns exemplos de implementação de legendas baseadas em tempo, conclui-se que a sincronização da legenda (formato SRT) com o <video> está bem implementada, porém ainda exitem diversas outras necessidades que ainda precisam ser melhor trabalhadas, tais como: opções de formatação, como trocar a cor das legendas, modificar sua posição durante a apresentação (por exemplo, passando para o topo da tela para não cobrir um texto in-video).

Abrangendo o escopo de análise, Clark *et al.* (2010) examinaram o nível de acessibilidade e usabilidade do framework Fluid Infusion (INCLUSIVE DESIGN RESEARCH CENTRE, 2010), com ênfase nos desafios, limitações e oportunidades apresentadas pela ARIA quando voltadas para as necessidades de usuários em um contexto real. O framework trabalha com a jQuery UI, combi-

nando JavaScript, CSS e HTML. Antes de apresentar o framework, são apresentados alguns desafios encontrados na acessibilidade de JavaScript, como é utilizada a navegação via teclado e como ARIA ajuda nesta navegação. Feito isto, são apresentadas características e componentes do framework e como ele se comporta com regiões vivas, reordenação de elementos na página, entre outros.

Ao final, conclui-se que somente a utilização de ARIA não é garantia de acessibilidade, logo desenvolvedores de toolkits devem também se focar na usabilidade geral de seus produtos por usuários com deficiência. Isto é exatamente o que o framework Fluid Infusion faz, combinando diversas técnicas para garantir a inclusão de uma experiência Web para todos.

Voltado para as Aplicações Ricas de Internet, Merayo (2011) introduz a criação e execução de aplicações Web acessíveis, explicando como funciona a comunicação cliente-servidor via XMLHttpRequest, como é executada uma aplicação Web, e mostrando quais requisitos esta necessita para ser de fato acessível. Após isto, Merayo realiza uma análise dos problemas de acessibilidade RIA (sendo estes divididos em 3 categorias: problemas de operabilidade, problemas semânticos de elementos interativos e Live-Regions) e as soluções que WAI-ARIA oferece para estes problemas. São utilizados exemplos de widgets criados pela biblioteca jQuery, dando algumas informações sobre seu suporte à acessibilidade. O trabalho de Merayo tem por conclusão que os principais problemas de acessibilidade podem ser resolvidos com WAI-ARIA, que é suportada pela maioria dos sistemas operacionais, navegadores e empresas de tecnologia assistiva. Aplicações não acessíveis conduzem à exclusão social, violando os direitos de muitos cidadãos, devido à sua deficiência. Cada organização é livre para escolher uma aplicação convencional ou RIA, mas qualquer que seja a escolha, o resultado deveria ser sempre acessível. Porque é possível, e necessário.

Focados na avaliação automatica de páginas Web utilizando técnicas da WCAG, Fernandes *et al.* (2013) apresentam um estudo comparativo para enten-

der a diferença entre as propriedades de acessibilidade da Web relacionando três diferentes perspectivas de avaliação: 1) antes do processamento do navegador; 2) após o processamento do navegador (carregamento dinâmico); 3) e, também após o processamento, considerando o disparo de eventos ocasionados pelas interações do usuário. Seus resultados mostram que para uma Aplicação de Internet Rica o número de resultados acessíveis varia consideravelmente entre estas três perspectivas. Neste trabalho é concluído que avaliar Aplicações de Internet Ricas sem considerar seus componentes dinâmicos geram uma percepção errônea de sua acessibilidade, ou seja, deve-se considerar todos os possíveis estados de uma página para se obter uma análise mais aprofundada.

Watanabe, Geraldo e Fortes (2014) apresentam uma investigação sobre como a acessibilidade via teclado é entregue em Aplicações de Internet Ricas. Eles conduziram uma avaliação em 32 página Web que continham Tab Widgets, esta avaliação consistia em verificar se os Widgets implementavam as recomendações ARIA, como a utilização dos atributos *role/state* e apresentação de estratégias de interação via teclado. O resultado de seu trabalho mostrou que, apesar de as especificações ARIA terem conseguido o status de *W3C Candidate Recommendation* em 2011, poucas páginas Web implementam Tab Widgets seguindo suas recomendações. O estudo também identificou mecanismos alternativos de navegação via teclado que são acessíveis para usuários de tecnologias assistivas, apesar das desvantagens que eles podem representar.

Giraud *et al.* (2011) descrevem os problemas de acessibilidade encontrados por pessoas cegas que utilizam Aplicações de Internet Ricas. Graças a seus testes com usuários, diversos problemas foram encontrados, não somente problemas de acessibilidade, mas também de consistência e organização dos dados. Neste trabalho são sugeridas recomendações para cada um dos problemas encontrados. Eles concluem que as recomendações da W3C fornecem recomendações para os problemas encontrados, porém muitas das vezes essas recomendações são ambí-

guas, complexas de se entender e de aplicar em situações particulares. A distinção entre acessibilidade e usabilidade para pessoas cegas é superficial na prática. De uma maneira lógica, usuários cegos não deveriam se adaptar à interface mas a interface deveria se adaptar aos usuários cegos.

A maneira com que o funcionamento da Acessibilidade para Aplicações Ricas para Internet e o que ainda precisa ser melhorado nestas são apresentados nestes trabalhos, serviram como base para o desenvolvimento desta monografia. Porém, apesar destes trabalhos citarem exemplos da biblioteca jQuery, poucos têm apresentado um estudo mais aprofundado, deixando de mostrar implementações e pontos fortes e fracos da biblioteca no quesito acessibilidade, que é o foco desta monografia. Ainda será realizada uma análise dos programas implementados com a biblioteca através da utilização de leitores de tela, para uma avaliação mais voltada ao usuário.

3 METODOLOGIA

3.1 Caracterização da Pesquisa

Este trabalho é, por natureza, uma pesquisa aplicada pois há uma finalidade de aplicação prática, ele teve o objetivo de buscar conhecimento e identificar problemas. Quanto aos objetivos, é um Estudo de Caso, com procedimento de avaliação de interface de caráter formativo, pois tem por propósito o aperfeiçoamento da biblioteca jQuery. É utilizada uma abordagem qualitativa, visto que durante a análise da biblioteca, foram qualificados os diferentes tipos de suporte à acessibilidade Web que esta oferece.

Quanto ao procedimento técnico utilizado na pesquisa, trata-se de um estudo de caso, que tem por objetivo obter o maior número possível de informações sobre a biblioteca e seu suporte para acessibilidade.

3.2 Procedimentos Metodológicos

3.2.1 Desenho

O trabalho passou pelas seguintes etapas:

1. **Análise da biblioteca jQuery:** o trabalho começou com a análise dos componentes da biblioteca jQuery, visando levantar dados sobre a presença de elementos de ARIA em suas implementações;

2. **Análise de leitores de tela:** foi realizada uma análise do suporte que estes oferecem para ARIA;

3. **Construção de protótipos:** foi implementado um protótipo utilizando os componentes da biblioteca para auxiliar na análise e verificar se a acessibilidade deste funciona de forma adequada juntamente com os leitores de tela.

3.2.2 Análise de Componentes da Biblioteca

Foi desenvolvido um protótipo que implementa diversas funcionalidades da biblioteca a fim de verificar sua adequação com as especificação WAI-ARIA. Segue alguns itens que foram testados:

- **Efeitos:** A biblioteca fornece várias técnicas para adicionar animações a uma página Web. Estas incluem animações simples, animações padrão (que são utilizadas com frequencia) e a capacidade de criar efeitos personalizados;
- **Eventos:** Métodos utilizados para registrar comportamentos a ter efeito quando o usuário interage com o navegador, e para manipular ainda mais esses comportamentos registrados;
- **Formulários:** Métodos e manipuladores de eventos que lidam com formulários e seus vários elementos;
- **Manipulação:** Métodos que manipulam DOM de alguma maneira. Alguns podem simplesmente modificar atributos de um elemento, enquanto outros definem propriedades de estilo de um elemento. Ainda existem os que modificam elementos (ou grupos de elementos) por completo;
- **Ajax:** A biblioteca possui um conjunto completo de recursos Ajax. As funções e métodos permitem carregar dados do servidor, sem uma atualização de página do navegador

3.2.3 Análise de Leitores de Tela

Durante o processo de avaliação das páginas criadas utilizando a biblioteca jQuery, foram analisados também os suportes que leitores de tela dão para Aplicações Ricas de Internet. Os leitores utilizados foram:

- Jaws (FREEDOM SCIENTIFIC, 2014): Software pago (com demonstração gratuita) para o sistema operacional Microsoft Windows. Desenvolvido pela Freedom Scientific, é líder de mercado em tecnologia para leitores de tela;
- NVDA (NV ACCESS, 2014): Leitor de tela gratuito, de código aberto para o sistema operacional Microsoft Windows. Desenvolvido pela NV Access juntamente com uma comunidade global de contribuidores.

3.2.4 Implementação de Protótipo de Aplicação RIA Utilizando jQuery

Na implementação do protótipo, foram utilizados diversos métodos da biblioteca, sendo os principais aqueles que já passaram por atualizações e que oferecem suporte a ARIA. Foram analisados também métodos que não oferecem suporte a ARIA, sendo que neste caso, foram apresentadas possíveis soluções para que tal método passe a oferecer suporte.

3.2.5 Inspeção de Acessibilidade dos Protótipos Utilizando Leitores de Tela

Feito o protótipo, este foi testado por meio de análises pelo investigador utilizando softwares leitores de tela. A partir disto, foi verificado o comportamento do protótipo em um contexto real, como por exemplo, se *landmarks* funcionam de maneira adequada utilizando as teclas de atalho do leitor de tela. Este protótipo também foi testado em diversos navegadores (Google Chrome, Microsoft Internet Explorer, Mozilla FireFox e Opera), a fim de verificar o quão robusto ele é.

4 DESENVOLVIMENTO DE PROTÓTIPO UTILIZANDO A BIBLIOTECA JQUERY UI

Para avaliação da biblioteca foi criado um website que utiliza efeitos e widgets da biblioteca jQuery UI. Foi avaliada a adequação destes com as recomendações de acessibilidade para aplicações de internet ricas.

O protótipo é um exemplo simples de uma página Web de uma companhia aérea fictícia, esta página possui uma tela principal com um formulário para o usuário se inscrever no *feed* de notícias da empresa, uma tela de cadastro onde o usuário insere dados pessoais, uma tela para reserva de passagens, uma tela para pesquisa de satisfação e uma com detalhes sobre a empresa. Todas estas telas são acessadas através de um menu flutuante lateral. Apesar de simples, são implementados na página diversos componentes da biblioteca jQuery UI, os quais foram criados com o objetivo de "materializar" a implementação dos componentes de interface para permitir uma avaliação mais detalhada.

4.1 Descrição dos widgets e efeitos utilizados

Esta seção descreve cada componente utilizado no protótipo, apresentando seu funcionamento, seu *layout* e suas opções de customização. Tais componentes são listados a seguir:

- ***Accordion:*** (Figura 4.1) Apresenta painéis de conteúdo dobráveis para apresentação de informação em uma quantidade limitada de espaço. Ele converte um par de cabeçalhos e painéis de conteúdo em um acordeão. Algumas customizações que podem ser realizadas neste widget são:
  - Minimizar conteúdo: Por padrão, acordeões sempre mantêm uma seção aberta. Para permitir que todas seções sejam minimizadas, basta alterar a opção *collapsible* para true;

- Ícones customizados: Customiza os ícones do cabeçalho com a opção *icons*;
- Preencher espaço: Devido ao fato de o acordeão ser composto de elementos em nível de bloco, por padrão sua largura preenche o espaço horizontal disponível. Para preencher o espaço vertical alocado pelo seu recipiente, basta definir a opção *heightstyle* para *fill* e o script irá definir automaticamente as dimensões do acordeão à altura de seu recipiente pai;
- Ordenável: Arraste os cabeçalhos para reordenar os painéis.

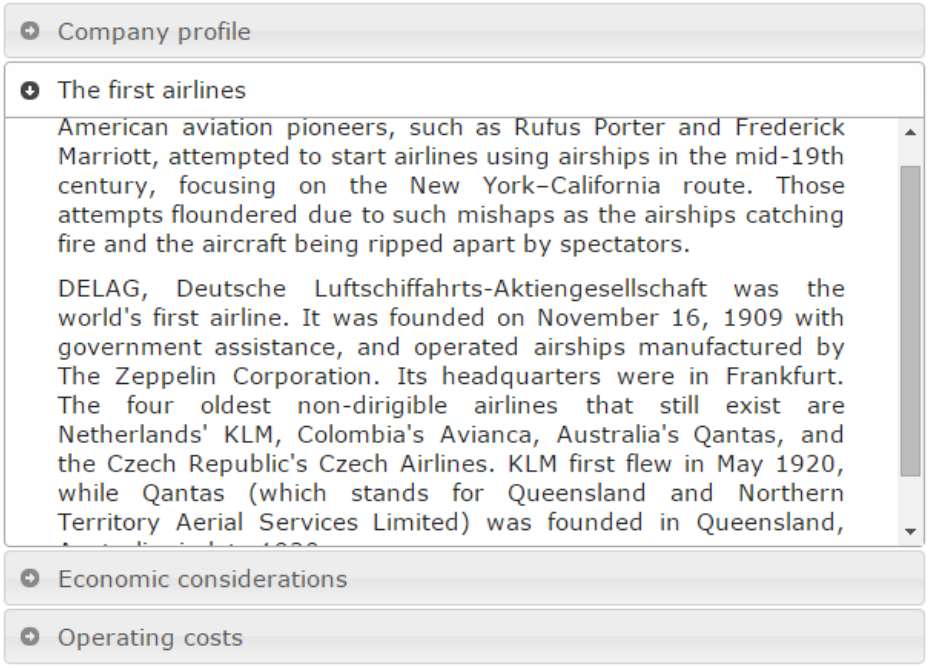


Figura 4.1: *Accordion* implementado no protótipo

- ***Autocomplete:*** (Figura 4.2) Possibilita ao usuário rapidamente encontrar e selecionar de uma lista pré-povoada de valores à medida que ele digita, otimizando assim a busca. Algumas customizações que podem ser realizadas neste widget são:

- Acentuação flexível: O campo de preenchimento automático utiliza uma opção de fonte personalizada que irá corresponder a caracteres que têm caracteres acentuados, mesmo quando o campo de texto não os contém;
- Categorias: Divide o resultado da busca em categorias;
- Múltiplos valores: Permite autocompletar diversos valores em um único campo de busca;
- Fonte de dados remota: Pode se especificar uma fonte de dados através de uma simples URL para a opção *source*.

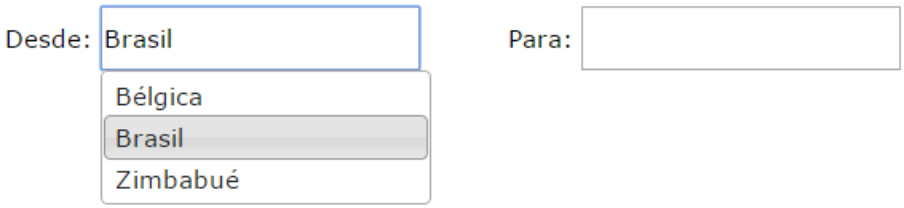


Figura 4.2: *Autocomplete* implementado no protótipo

- ***Datepicker:*** (Figura 4.3) Seleciona uma data a partir de um calendário (que pode ou não ser mostrado através de um pop-up). Algumas customizações que podem ser realizadas neste widget são:
  - Datas em outros meses: Pode-se apresentar datas de outros meses além do mês principal que está sem apresentado ao usuário. Estas outras datas também são selecionáveis;
  - Apresentar barras com botões: Apresenta um botão para "data de hoje" e um botão "Feito" para fechar o calendário. O texto dos botões são customizáveis;
  - Apresentar a semana do ano: O cálculo padrão segue a definição da ISO 8601: a semana começa no domingo e a primeira semana do ano contém a primeira quarta-feira do ano.

Figura 4.3: *Datepicker* implementado no protótipo

- ***Dialog:*** (Figura 4.4) Abre determinado conteúdo em uma janela interativa. Algumas customizações que podem ser realizadas neste widget são:
  - Modal básico: previne que o usuário interaja com o restante da página até que a caixa de diálogo seja fechada;
  - Formulário modal: Requer que o usuário entre com dados durante o processo de diversos passos.

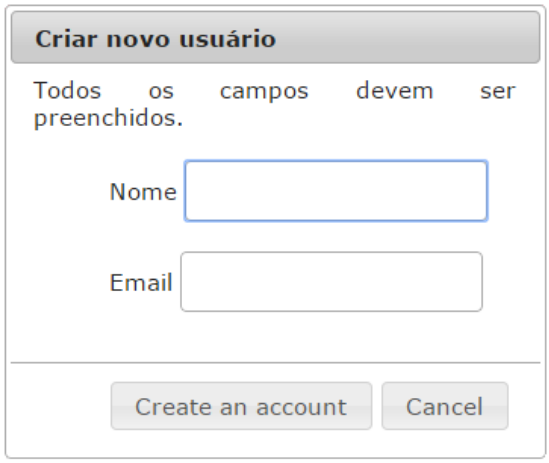


Figura 4.4: *Dialog* implementado no protótipo

- ***Menu:*** (Figura 4.5) Menu com interações de mouse e teclado para navegação. Apresenta a possibilidade de alteração de temas e inclusão de ícones.

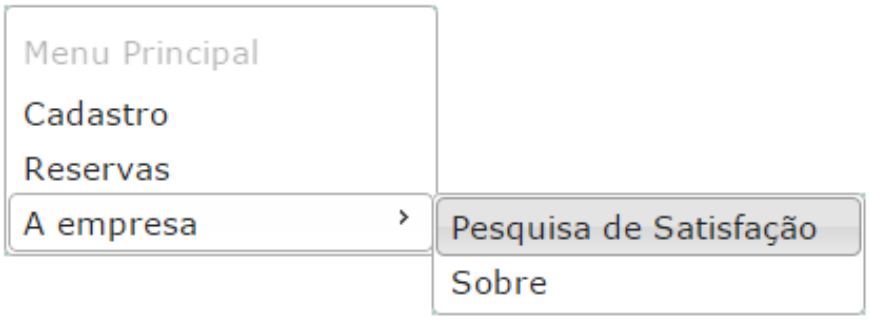


Figura 4.5: *Menu* implementado no protótipo

- ***Progressbar:*** (Figura 4.6) Apresenta o estado de um processo (determinado ou indeterminado). Possibilita inclusão e customização de rótulos.

Figura 4.6: *Progressbar* implementado no protótipo

- ***Spinner:*** Aprimora uma entrada de texto para inserir valores numéricos, com suporte para manipulação via setas do teclado e botões cima/baixo. Algumas customizações que podem ser realizadas neste widget são:
  - Utilização de valores decimais: Por padrão, são utilizados valores decimais;

- Mapa: Integração com Google Maps, utilizando o Spinner para alterar latitude e longitude;
- Overflow: Para uma restrição de -10 a 10, para qualquer valor acima de 10, este será alterado para -10 e vice versa.

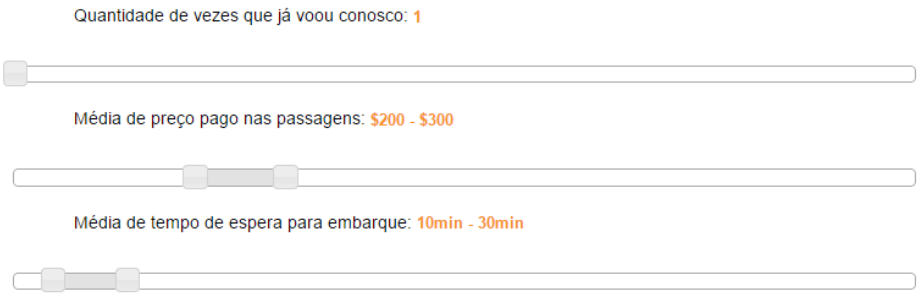


Figura 4.7: *Spinners* implementados no protótipo

- ***Tooltip:*** Pode ser atribuído a qualquer elemento. Ao passar o mouse sobre este elemento, o atributo title é mostrado em uma pequena caixa próxima ao elemento. Diferente de um tooltip nativo, este pode ser customizado.

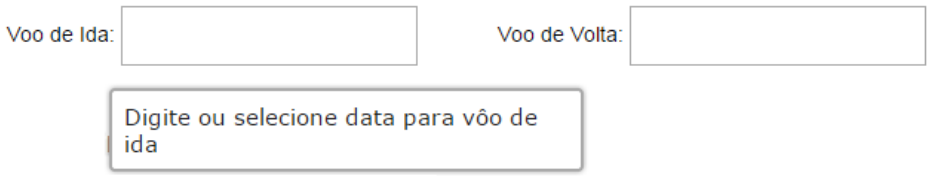


Figura 4.8: *Tooltip* implementado no protótipo

4.2 Desenvolvimento

Para cada página HTML desenvolvida, foi necessário:

1. Criação de um link para o arquivo jQuery UI CSS:

```
<link rel="stylesheet" href="../development-bundle/themes/base/jquery.ui.all.css">
```

2. Criação de uma tag <script> que aponta para a jQuery UI:

```
<script src="../js/jquery-ui-1.10.4.custom.js"></script>
```

Após isto, foi criado um arquivo JavaScript, no qual eram inseridos códigos da biblioteca jQuery UI para poder assim identificar quais elementos serão alterados pela biblioteca e como estas alterações serão realizadas. Segue a explicação de como foi desenvolvido o código para cada um dos elementos:

4.2.1 Accordion

No código JavaScript da Figura 4.9 foi criada uma variável *icons* com ícones customizados para os cabeçalhos ativo e não ativo. Feito isto, foi definido que o id "accordion" seria um widget jQuery do tipo acordeão e que este utilizaria os ícones da variável *icons*.

```
$(function() {
   var icons = {
      header: "ui-icon-circle-arrow-e",
      activeHeader: "ui-icon-circle-arrow-s"
   };
   $( "#accordion" ).accordion({
      icons: icons
   });
});
```

Figura 4.9: Código para criação de widget acordeão

No código HTML foi criado um div com id "accordion" e dentro deste div continham cabeçalhos h3 com o título de cada tópico e div's com o conteúdo correspondente de cada cabeçalho.

4.2.2 Autocomplete

No código HTML foi criada a classe "tags", representando o input onde o usuário inseria os locais de origem e destino do vôo. No código JavaScript da Figura

4.10, é primeiramente criado um vetor de Strings contendo países. Feito isto, foi definido que objetos da classe “tags” seriam widgets jQuery auto-completar, contendo como fonte de busca o array criado anteriormente.

```
var availableTags = [
   "África do Sul",
   "Alemanha",
   "Argentina",
   "Bélgica",
   "Brasil",
   "Canadá",
   "China",
   "Dinamarca",
   "Espanha",
   "Estados Unidos",
   "Grécia",
   "Haiti",
   "Itália",
   "Japão",
   "Macau",
   "Mongólia",
   "Paquistão",
   "Rússia",
   "Singapura",
   "Taiwan",
   "Uruguai",
   "Zimbabué"
];
$( ".tags" ).autocomplete({
   source: availableTags
});
```

Figura 4.10: Código para criação de widget auto-completar

4.2.3 Datepicker

No código HTML foram criados dois inputs com as id’s “from” e “to”. No código JavaScript da Figura 4.11, estas id’s foram definidas como sendo jQuery widgets datepicker, possuindo como opções: possibilidade de alterar o mês através do widget (changeMonth: true), alterar para 1 o número de meses que aparece assim que o usuário acessa o widget (numberOfMonths: 1) e, como as datas estão dentro de um intervalo, foram criadas funções em que se o usuário seleciona uma data na

opção de ida, na opção de volta a data escolhida anteriormente será a data mínima para escolha, e vice-versa.

```
$( "#from" ).datepicker({
   changeMonth: true,
   numberOfMonths: 1,
   onClose: function( selectedDate ) {
      $( "#to" ).datepicker( "option", "minDate", selectedDate );
   }
});

$( "#to" ).datepicker({
   changeMonth: true,
   numberOfMonths: 1,
   onClose: function( selectedDate ) {
      $( "#from" ).datepicker( "option", "maxDate", selectedDate );
   }
});
```

Figura 4.11: Código para criação de widget datepicker

4.2.4 Dialog

Como mostrados na Figura 4.12, primeiramente são criadas as variáveis que serão utilizados dentro do código JavaScript.

```
var name = $( "#name" ),
   email = $( "#email" ),
   allFields = $( [] ).add( name ).add( email ),
   tips = $( ".validateTips" );
```

Figura 4.12: Código para criação de widget dialog

Feito isto, é criada a função (Figura 4.13) que atualiza as dicas para inserção de dados. Esta recebe um texto que aparecerá para o usuário em destaque, após algum tempo este texto deixará de ficar em destaque.

A terceira função (Figura 4.14), é criada para realizar a verificação do tamanho da string inserida pelo usuário dentro do formulário. Caso o tamanho esteja fora do limite imposto, é apresentada uma dica, dizendo qual o intervalo de tamanho necessário para a string.

```
function updateTips( t ) {
   tips
      .text( t )
      .addClass( "ui-state-highlight" );
      setTimeout(function() {
         tips.removeClass( "ui-state-highlight", 1500 );
      }, 500 );
}
```

Figura 4.13: Código para criação de widget dialog

```
function checkLength( o, n, min, max ) {
   if ( o.val().length > max || o.val().length < min ) {
      o.addClass( "ui-state-error" );
      updateTips( "Length of " + n + " must be between " +
      min + " and " + max + "." );
      return false;
   } else {
      return true;
   }
}
```

Figura 4.14: Código para criação de widget dialog

A quarta função (Figura 4.15) serve para a checagem de expressões regulares, ou seja, se a string inserida pelo usuário possui caracteres que estejam de acordo com a entrada exigida no formulário. Também neste caso, se o usuário inserir um valor de entrada errado, é apresentada uma dica e aquele valor é adicionado à classe “ui-state-error”.

```
function checkRegexp( o, regexp, n ) {
   if ( !( regexp.test( o.val() ) ) ) {
      o.addClass( "ui-state-error" );
      updateTips( n );
      return false;
   } else {
      return true;
   }
}
```

Figura 4.15: Código para criação de widget dialog

A Figura 4.16 apresenta a função principal de criação do caixa de diálogo. Nela estão inseridas as opções do método *.dialog* como: não deixar que a caixa de diálogo abra automaticamente e sim quando o usuário selecionar um botão específico (autoOpen: false), altura e largura das caixas de texto onde o usuário irá inserir os dados, quais serão os botões de interação e quais serão suas funções, especificar qual o formato de expressão regular exigida na entrada e especificar qual será a mensagem de erro apresentada.

```
$( "#dialog-form" ).dialog({
   autoOpen: false,
   height: 300,
   width: 350,
   modal: true,
   buttons: {
      "Create an account": function() {
         var bValid = true;
         allFields.removeClass( "ui-state-error" );

         bValid = bValid && checkLength( name, "username", 3, 16 );
         bValid = bValid && checkLength( email, "email", 6, 80 );

         bValid = bValid && checkRegexp( name, /^[a-z]([0-9a-z_])+$/i,
                        "Padrão deve ser a-z 0-9" );
         bValid = bValid && checkRegexp( email, /^((([a-z]|\d|[!#\$...);

         if ( bValid ) {
            $( "#users tbody" ).append( "<tr>" +
            "<td>" + name.val() + "</td>" +
            "<td>" + email.val() + "</td>" +
            "</tr>" );
            $( this ).dialog( "close" );
            alert("Thank you!");
         }
      },
      Cancel: function() {
         $( this ).dialog( "close" );
      }
   },
      close: function() {
         allFields.val( "" ).removeClass( "ui-state-error" );
      }
});
```

Figura 4.16: Código para criação de widget dialog

4.2.5 Menu

No código HTML foi criada a classe “menu”, esta por sua vez, foi definida como widget menu(), como apresenta o código javaScript da Figura 4.17. O código dentro do método faz com que, ao ser selecionado um item, o usuário seja transferido para a página correspondente ao mesmo. Na Figura 4.11, ui.item retorna o elemento da lista e .children retorna a tag ‘a’, de onde é extraída o atributo *href*.

```
$(".menu").menu({
   select: function(event, ui) {
      window.location = ui.item.children().attr('href');
   }
});
```

Figura 4.17: Código para criação de widget menu

4.2.6 Progressbar

No código HTML foi criada uma div com id “progressDialog” e um button com id “progressTrigger”. Ao ser clicado, o botão apresenta ao usuário uma caixa de diálogo contendo a barra de progresso, que é preenchida de dez em dez por cento até alcançar a marca de cem, como apresentado no código JavaScript da Figura 4.18.

4.2.7 Spinner

No código HTML foram criados inputs com nome de class “spinner” e estes por sua vez foram definidos como spinners através do método jQuery, como mostrado no código JavaScript da Figura 4.19. Neste código é criado um evento “spin” que irá funcionar como um loop, sendo que quando o valor da entrada do usuário for maior que 30, o valor do spinner será modificado automaticamente para 0, e vice-versa. Dentro de cada condição é retornado o valor booleano falso, para que o evento seja sempre executado.

```
$(function() {
      $("#progressTrigger").button().click(function() {
            $("#progressMsg").remove();
            var progressBar = $("#sampleProgressBar")
            .progressbar({
                  value: 0,
                  labelledBy: "progressMsg"
             });
            var progressDialog = $("#progressDialog")
            .append("<p id='progressMsg' aria-live='true'>Aguarde...</p>")
            .dialog({autoOpen : true,
                  modal : true,
                  title :  "Progresso",
                  resizable : false,
                  draggable : false,
                  dialogClass : "noCloseBtn",
                  width : 500,
                  beforeClose : function() {
                     if ($("#sampleProgressBar").progressbar('value') != 100)
                           return false;
                  }
            })

            var progressUpdater;

            setTimeout(function() {
                  $("#sampleProgressBar").progressbar('value', 0);
                  progressUpdater = setInterval(function() {
                        if ($("#sampleProgressBar").progressbar('value') == 100) {
                              clearInterval(progressUpdater);
                              $("#progressDialog").dialog("close");
                              $('#progressTrigger').focus();
                        }
                        $("#sampleProgressBar").progressbar('value',
                        $("#sampleProgressBar").progressbar('value') + 10);
                        }, 250);
            }, 100);
      });
});
```

Figura 4.18: Código para criação de widget progressbar

4.2.8 Tooltip

Como mostrado no código JavaScript da Figura 4.20, todos os atributos “title” contidos no código HTML serão transformados em widgets jQuery tooltip.

```
$( ".spinner" ).spinner({
   spin: function( event, ui ) {
      if ( ui.value > 30 ) {
         $( this ).spinner( "value", 0 );
         return false;
      } else if ( ui.value < 0 ) {
         $( this ).spinner( "value", 30 );
         return false;
      }
   }
});
```

Figura 4.19: Código para criação de widget Spinner

```
$( "[title]" ).tooltip();
```

Figura 4.20: Código para criação de widget Tooltip

5 RESULTADOS DA AVALIAÇÃO E DISCUSSÃO

Após a criação do protótipo, todos os widgets implementados foram analisados utilizando os principais navegadores Web e leitores de tela livres e comerciais, o sistema operacional utilizado foi o Microsoft Windows 7 Ultimate 64 bits. A partir destes resultados foi possível analisar como a biblioteca jQuery UI se comporta em cada situação, ou seja, se ela apresenta compatibilidade com as diferentes APIs de acessibilidade utilizadas pelos navegadores e com os leitores de tela.

Para análise de resultados foram utilizados os seguintes navegadores:

- Google Chrome v.34
- Microsoft Internet Explorer v.11
- Mozilla Firefox v.29
- Opera v.21

Os widgets analisados se comportaram de formas diferentes dependendo do navegador e do leitor de tela utilizados. As Tabelas 5.1 e 5.2 representam tais dados, preenchendo o campo com o valor "sim" se o leitor obtém a resposta esperada do widget em determinado navegador e "não" em caso contrário.

Tabela 5.1: Funcionamento de widgets utilizando leitor de tela Jaws

Widget/Navegador	*Chrome*	*InternetExplorer*	*Firefox*	*Opera*
Accordion	Sim	Sim	Sim	Sim
Autocomplete	Sim	Sim	Sim	Não
Datepicker	Não	Não	Não	Não
Dialog	Sim	Sim	Sim	Sim
Menu	Sim	Sim	Sim	Não
Progressbar	Não	Sim	Sim	Não
Slider	Não	Não	Não	Não
Spinner	Não	Sim	Sim	Não
Tooltip	Sim	Sim	Sim	Sim

Para o leitor de tela Jaws, apenas os widgets *accordion, dialog* e *tooltip* se comportaram de forma adequada em todos os navegadores, não apresentando

nenhum comportamente fora do esperado. Os widgets *datepicker* e *slider* foram os únicos que não funcionaram corretamente em nenhum dos navegadores.

Os navegadores Internet Explorer e Firefox foram os que se comportaram de melhor forma com o leitor de tela Jaws, apenas dois widgets não funcionaram adequadamente em cada.

Tabela 5.2: Funcionamento de widgets utilizando leitor de tela NVDA

Widget/Navegador	*Chrome*	*InternetExplorer*	*Firefox*	*Opera*
Accordion	Sim	Sim	Sim	Sim
Autocomplete	Não	Não	Sim	Não
Datepicker	Não	Não	Não	Não
Dialog	Sim	Sim	Sim	Sim
Menu	Sim	Não	Sim	Sim
Progressbar	Não	Sim	Sim	Não
Slider	Não	Não	Não	Não
Spinner	Sim	Sim	Sim	Sim
Tooltip	Sim	Sim	Sim	Sim

Para o leitor de tela NVDA, apenas os widgets *accordion, dialog, spinner* e *tooltip* se comportaram de forma adequada em todos os navegadores. Os widgets *datepicker* e *slider* novamente foram os únicos que não funcionaram corretamente em nenhum dos navegadores.

O navegador Firefox foi o que se comportou de melhor forma com o leitor de tela NVDA, apenas dois widgets não funcionaram adequadamente.

5.1 Inserção de Código Feita Pelo JavaScript

Utilizando da mesma sintaxe que CSS para selecionar elementos em uma página, jQuery torna possível “caminhar” pela árvore DOM facilmente, possuindo comandos específicos para acessar nós da árvore, por exemplo, o nó pai de determinado elemento, o nó filho, nó folha, entre outros.

Por meio desta árvore DOM a biblioteca jQuery também pode inserir código adicional à página. Para isto são utilizados os métodos *before* e *after*, que inserem código HTML **antes** e **depois** do elemento especificado, respectivamente.

A posição em que o código é inserido em uma página possui papel importante na forma com que um leitor de tela irá interpretá-lo. Por exemplo, se um código HTML for inserido no final da página (sendo sua posição na tela alterada através de CSS), usuários de leitores de tela encontrarão dificuldade para ter acesso a este conteúdo adicional, pois apesar de o conteúdo aparecer visualmente na posição correta, o leitor de tela não consegue ter acesso imediato a este conteúdo, fazendo com que o usuário necessite procurar pela página onde o novo código foi inserido.

Não somente inserir, código HTML também pode ser removido da página através do uso da biblioteca jQuery. No protótipo, foi implementado um código que exemplifica a situação:

```
$(".formBut").click(function(){
   $(".formulario1").detach();
   $("<form class= 'formulario2' role='form' aria-labelledby='form'>\
         <h2>Logon</h2>\
         <label for='email'>Email: </label>\
               <input type='text' placeholder='email adress' ><br>\
               <label for='pw'>Password: </label>\
               <input type='password' placeholder='Password' ><br>\
               <button class='formBut2' type='button'>Finish</button>\
      </form>").insertAfter('div#main');

   $(".formBut2").click(function(){
      alert("Complete!");
   });
});
```

Figura 5.1: Código para remoção e inserção de código HTML

No código da Figura 5.1, todo código contido dentro da classe "formulario1" foi removido da árvore DOM e em seu lugar foi inserido o código especificado dentro da função jQuery. A Figura 5.2 apresenta como a alteração é feita no código HTML.

```
▼<div id="main" role="main" aria-labelledby="main">
 ▼<form class="formulario1" role="form" aria-labelledby="form">
    <h2>Dados Pessoais</h2>
    <label for="first">Nome: </label>
    <input type="text" placeholder="Digite seu primeiro nome" autofocus>
    <br>
    <label for="last">Sobrenome: </label>
    <input type="text" placeholder="Digite seu sobrenome">
    <br>
    <label for="dob">Data de nascimento: </label>
    <input type="date" placeholder="Digite sua data de nascimento">
    <br>
    <label for="adress">Endereço: </label>
    <input type="text" placeholder="Digite seu endereço">
    <br>
    <button class="formBut" type="button">Next</button>
  </form>
 </div>
```

```
 <div id="main" role="main" aria-labelledby="main"></div>
▼<form class="formulario2" role="form" aria-labelledby="form">
   <h2>Logon</h2>
   <label for="email">Email: </label>
   <input type="text" placeholder="email adress">
   <br>
   <label for="pw">Password: </label>
   <input type="password" placeholder="Password">
   <br>
   <button class="formBut2" type="button">Finish</button>
 </form>
```

Figura 5.2: Antes e depois de o script ter sido executado para substituir código HTML

Existem outras formas de modificar o conteúdo presente em uma página HTML, por exemplo utilizando-se os métodos jQuery *hide* e *show*. Apesar de visualmente terem o mesmo resultado do exemplo mostrado anteriormente, estes não modificam a árvore DOM, eles simplesmente ocultam do usuário o conteúdo, o que não o torna uma boa solução, pois leitores de tela conseguem ter acesso a este conteúdo escondido.

5.2 Comportamento com NVDA e Jaws

Através dos exemplos criados para o protótipo, foi feita uma análise de como cada componente da página adicionado através da biblioteca jQuery se comporta na utilização de leitores de tela.

5.2.1 Accordion

- Utilização via teclado: Não obteve problema durante a navegação, as setas são utilizadas para mover o foco entre os cabeçalhos do acordeão, sendo cada um expandido utilizando a tecla *space* ou *enter*.

- Comportamento com leitor de tela:

  - NVDA:

    Pontos positivos:

    * Informa o estado atual do item em foco (retraído ou expandido);
    * Lê o título presente no cabeçalho;

    Pontos negativos:

    * Impossibilidade de acessar o conteúdo apresentado ao selecionar qualquer cabeçalho, este conteúdo só foi lido pelo leitor ao passar o mouse sobre o conteúdo.

  - Jaws:

    Pontos positivos:

    * Informa o estado atual do item em foco (retraído ou expandido);
    * Lê o título presente no cabeçalho;
      Acessa o conteúdo apresentado ao selecionar o cabeçalho;

    Pontos negativos:

    * Nenhum;

5.2.2 Autocomplete

- Utilização via teclado: Não obteve problema durante a navegação, após o usuário começar a digitar determinada palavras, aparecem opções para que este possa selecioná-las, as setas são utilizadas para mover o foco entre estas opções e a tecla *enter* para selecionar a desejada.

- Comportamento com leitor de tela:

  - NVDA e Jaws:

    Pontos positivos:

    * Ao passar pelos itens da lista, cada opção foi lida para o usuário;

    Pontos negativos:

    * Ao começar a digitar, não é informado ao usuário que existe uma lista com as opções possíveis;
    * Não é apresentada a quantidade de elementos que foram gerados na lista;

5.2.3 Datepicker

- Utilização via teclado: Não obteve problema durante a navegação, para navegação ele utiliza as seguintes teclas:

  - Ctrl + Setas: Mover entre os dias do mês apresentado
  - Enter: Seleciona o mês focado
  - Page Up: Mudar para o mês anterior
  - Page Down: Mudar para o próximo mês

- Comportamento com leitor de tela:

  - NVDA e Jaws: Não conseguiram identificar o widget, cada dia do calendário foi lido como um campo “em branco”;

5.2.4 Dialog

- Utilização via teclado: Não obteve problema durante a navegação, as caixas de texto do formulário contidas dentro da caixa de diálogo puderam ser facilmente acessadas através da tecla *tab*, os botões presentes na caixa de diálogo também puderam ser acessados da mesma forma, além da possibilidade de serem selecionados através da tecla *enter* ou *space*.

- Comportamento com leitor de tela:

  - NVDA e Jaws:

    Pontos positivos:

    * Foco é transferido automaticamente para a caixa de diálogo criada;
    * Todos os botões presentes na caixa de diálogo foram identificados de forma correta, apresentando ao usuário o texto presente em cada;

    Pontos negativos:

    * Para o formulário presente dentro da caixa de diálogo, não é dado um feedback ao usuário quando algum campo está em destaque;
    * Quando é apresentado um texto adicional dentro da caixa de diálogo, uma dica por exemplo, o usuário não é informado de sua inserção e também não é possível acessar este conteúdo via teclado;

5.2.5 Menu

- Utilização via teclado: Não obteve problema durante a navegação, as setas são utilizadas para mover o foco entre os itens do menu, os botões *enter*/*space* são utilizados para selecionar um item ou abrir um submenu e a tecla *escape* é utilizada para fechar um submenu.

- Comportamento com leitor de tela:
  - NVDA e Jaws:

    Pontos positivos:

    * Todos os itens presentes no menu foram identificados corretamente tendo seus títulos lidos à medida que o foco era alterado;
    * Itens com sub-menu informados ao usuário;

    Pontos negativos:

    * Estado do item não informado, por exemplo, se um item não puder ser selecionado;

5.2.6 Progressbar

- Utilização via teclado: Não possui interação via teclado.
- Comportamento com leitor de tela:
  - NVDA:

    Pontos positivos:

    * O texto contido na barra de progresso é lido corretamente;
    * À medida que a barra de progresso aumenta de tamanho, um *feedback* sonoro é dado ao usuário, começando com um beep mais grave e terminando com um beep mais agudo;

    Pontos negativos:

    * Nenhum;
  - Jaws:

    Pontos positivos:

    * O texto contido na barra de progresso é lido corretamente;

    * À medida que a barra de progresso aumenta de tamanho, um *feedback* sonoro é dado ao usuário, dizendo a porcentagem em que a barra de progresso se encontra;

  Pontos negativos:

    * Nenhum;

5.2.7 Spinner

- Utilização via teclado: Não obteve problema durante a navegação, setas cima/baixo são utilizados para acrescentar ou diminuir em uma unidade o valor corrente, enquanto as teclas *Page Up* e *Page Down* são utilizadas para acrescentar ou diminuir o valor corrente em dez.

- Comportamento com leitor de tela:

  – NVDA e Jaws:

    Pontos positivos:

    * Todos os números são lidos à medida que o usuário aperta os botões para acrescentar o diminuir (independente da quantidade) o valor corrente;

    Pontos negativos:

    * Nenhum;

5.2.8 Tooltip

- Utilização via teclado: Não possui interação via teclado.

- Comportamento com leitor de tela:

  – NVDA e Jaws:

    Pontos positivos:

    * Todo texto lido corretamente (campo *text* HTML);

Pontos negativos:

* Nenhum;

5.3 Avaliação das páginas com técnicas WCAG 2.0 ARIA

Além dos testes com leitores de tela, foi feita uma verificação das técnicas ARIA. Estas técnicas são de extrema importância para aumentar a acessibilidade de páginas Web pois adicionam a seus componentes diversas informações semânticas, fazendo assim com que o usuário saiba exatamente como a página está estruturada e como ela se comporta ao receber seus comandos.

Foi adicionado código que utiliza técnicas da WCAG 2.0 às páginas HTML para que estas melhorassem sua usabilidade. A seguir é feita uma verificação, se estas alterações modificam o comportamento dos widgets implementados na biblioteca jQuery UI.

5.3.1 ARIA1: Utilizar a propriedade *aria-describedby* para criar um rótulo que descreva controles da interface de usuário

- Radio Button - Como apresentado na Figura 5.3, foi acrescentada uma descrição utilizando a propriedade *aria-describedby* para acrescentar informações mais detalhadas a um botão. No exemplo, a informação foi adicionada a um widget do tipo *radio button*, este continuou funcionando de forma correta e o leitor de tela leu a informação adicional corretamente.

```
<label for="radio2">Bom</label>
<span id="tip4_2">Você está satisfeito com o serviço</span>
<input type="radio" id="radio2" checked="checked" aria-describedby="tip4_2">
```

Figura 5.3: Utilização da propriedade aria-describedby

5.3.2 ARIA4: Utilizar a propriedade WAI-ARIA *role* para mostrar os tipos dos componentes da interface de usuário

- Accordion - Por padrão, ele utiliza a *role* “tablist”, não sendo necessário inserí-las no código HTML pois o leitor de tela já as localiza corretamente.
- Button - Por padrão, ele utiliza a *role* “button”, o que se torna errado quando criados botões do tipo “radio” (onde deveria ser role=“radio”). Não foi possível alterar a *role* e o leitor de tela continuava a interpretar o elemento como um *role* do tipo “button”.
- Dialog - Por padrão, já utiliza as *roles* “dialog”, não sendo necessário inserí-las no código HTML pois o leitor de tela já as localiza corretamente.
- Menu - Por padrão, já utiliza as *roles* “menu” e “menuitem”, não sendo necessário inserí-las no código HTML pois o leitor de tela já as localiza corretamente.

5.3.3 ARIA5: Utilizar a propriedade WAI-ARIA *state* para mostrar o estado dos componentes da interface de usuário

- Slider - Um widget com *role* do tipo “slider” representa o valor corrente e o alcance de possíveis valores através do tamanho do “slider” e a posição do manipulador. Estas propriedades podem ser representadas por atributos *aria-valuemin*, *aria-valuemax* e *aria-valuenow*. Por padrão, o widget não possui marcações ARIA e ao inserir estas, o widget deixou de funcionar corretamente, sendo inacessível para usuários de leitores de tela.

5.3.4 ARIA6: Utilizar a propriedade *aria-label* para criar rótulos para objetos

- Radio Button - Testado no widget de tipo *radio button*, funcionou de forma correta, sendo esta a melhor opção para criar rótulos do gênero pois garante

que leitores de tela conseguirão reconhecer tal campo. No caso do widget *radio button*, também pode-se utilizar *spans* juntamente com a propriedade *aria-describedby* para descrevê-lo.

5.3.5 ARIA19: Utilizar "role=alert" para identificar erros

- HTML <p> - No protótipo utilizou-se a propriedade *role* com valor *alert* para identificar erros em um formulário, onde no caso o usuário deixava um ou mais campos vazios. A propriedade foi utilizada em um elemento <p> já presente no código HTML, mas que não continha texto algum, ao usuário clicar no botão para submeter o formulário, era feita uma verificação se todos os campos necessários foram preenchidos, em caso negativo, o elemento <p> era preenchido com código HTML contendo o texto de alerta. Este texto de alerta foi lido corretamente pelos leitores de tela. Exemplo na Figura 5.4.

```
<script>
$(document).ready(function(e) {
   $('#signup').submit(function() {
      $('#errors').html('');
      if ($('#first').val() === '') {
         $('#errors').append('<p>Por favor insira seu primeiro nome</p>');
      }
      return false;
   });
});
</script>

<form name="signup" id="signup" method="post" action="">
  <p id="errors" role="alert" aria-atomic="true"></p>
  <p>
    <label for="first">Primeiro nome*</label><br>
    <input type="text" name="first" id="first">
  </p>
</form>
```

Figura 5.4: Utilização da propriedade role=alert

5.4 Discussão

Apesar de existirem diversas técnicas que melhoram a acessibilidade em aplicações ricas para internet, fica claro como sua utilização ainda não possui a prioridade desejada se tratando do desenvolvimento de Websites ou até mesmo bibliotecas. Diversos usuários continuam encontrando problemas ao navegar pelas páginas Web, principalmente as que possuem elementos interativos. Tais elementos possuem, na maioria vezes, um excelente *feedback* visual e um *feedback* sonoro de qualidade inferior ou até mesmo inexistente.

A inserção de elementos interativos às páginas trouxe diversas formas diferentes de se navegar na Web, porém muitas delas acabaram aumentando a dificuldade com que pessoas com algum tipo de deficiência (visual ou motora) navegassem, seja para clicar e arrastar um item, selecionar uma opção ou saber se um novo elemento apareceu na página. Isto acaba levantando a seguinte questão: até que ponto a utilização de elementos dinâmicos e/ou interativos é benéfica para a navegação Web? Muitas vezes elementos interativos que necessitam a utilização de mouse podem ser substituídos facilmente por outra opção mais simples, como a navegação via teclado. Existem também casos em que não é necessária a utilização de scripts para se obter o mesmo resultado.

Existem listas de recomendações que ajudam desenvolvedores a criar páginas Web mais acessíveis. Apesar de melhorar bastante a experiência do usuário, somente seguir as recomendações não é o suficiente para que uma página seja acessível, diversos outros fatores devem ser considerados, como por exemplo a linguagem utilizada, a forma com que os elementos estão distribuídos pela página ou onde novos elementos estão sendo inseridos.

Desenvolvedores muitas vezes temem que a criação de páginas acessíveis sejam mais caras e demandem mais tempo do que as páginas não acessíveis. Este medo deve ser deixado. Os benefícios de prover acessibilidade para uma grande

população de usuários quase sempre superam o tempo requerido por um desenvelvedor que conheça como implementar esta acessibilidade. Um desenvolvedor pode aprender o básico sobre acessibilidade Web em poucos dias, mas como qualquer outra habilidade técnica, são necessários alguns meses para que este conhecimento seja internalizado.

Abaixo estão listados os componentes da biblioteca jQuery UI que estiveram ou não de acordo com cada uma das *Guidelines* da WCAG 2.0. Apenas serão apresentados problemas de não-conformidade com as guidelines, se a guideline não foi citada para determinado componente é porque este está de acordo com suas recomendações:

- ***Accordion***: Obteve problema de compatibilidade com o leitor de tela NVDA, não estando de acordo com a *guideline* 4.1.

- ***Autocomplete***: Não apresenta a quantidade de elementos presentes na lista de opções apresentada, o mesmo também não informa ao usuário que uma lista foi gerada. Assim, ele não está de acordo com as *guidelines* 1.1, 1.4 e 2.4. O Componente também não auxilia o usuário com a entrada de dados, não estando de acordo com a *guideline* 3.3.

- ***Datepicker***: Não operou de forma correta em nenhum dos leitores testados, estando de acordo somente com as *guidelines* 2.2, 2.3 e 3.1.

- ***Dialog***: Não fornece *feedback* ao usuário a respeito de conteúdos em destaque ou novos conteúdos que surgiram. Logo, não está de acordo com as *guidelines* 3.1 e 3.3.

- ***Menu***: Não informa o estado do item (se pode ser selecionável ou não). Logo não está de acordo com as guidelines 1.1, 1.4, 2.4 e 3.3.

- ***Progressbar***: Operou de forma correta em todos os leitores testados.

- ***Spinner***: Operou de forma correta em todos os leitores testados.

- ***Tooltip***: Operou de forma correta em todos os leitores testados.

Devido ao fato da biblioteca jQuery UI ser de código aberto, diversos usuários/desenvolvedores podem colaborar com o projeto, corrigindo eventuais falhas ou até mesmo acrescentando novas funcionalidades à biblioteca. A exemplo disto, o grupo Pacciello (THE PACIELLO GROUP, AOL, AEGIS, 2012), um grupo de consultoria envolvido diretamente com acessibilidade, fez modificações na biblioteca jQuery UI original, acrescentando a esta melhores interações com leitores de tela. Alguns dos problemas nos Widgets descritos anteriormente foram resolvidos com estas modificações, mostrando ser uma questão de tempo até que os problemas de acessibilidade presentes na biblioteca sejam resolvidos. Não somente modificação do código original, mas também acrescentando conteúdo adicional, existem diversos plugins para a biblioteca que acrescentam acessibilidade à mesma, estes plugins podem ser encontrados na página oficial da biblioteca (THE JQUERY FOUNDATION, 2013).

Com relação aos leitores de tela NVDA e Jaws, o segundo apresentou menos não-conformidades, reconhecendo um número maior de elementos e dando um *feedback* melhor, com informações mais claras. Porém, por se tratar de um leitor de tela gratuito, o *software* NVDA obteve um resultado satisfatório, conseguindo reconhecer cerca de sessenta por cento dos elementos inseridos na página. A diferença de funcionamento mais perceptível era quando se alterava o navegador Web utilizado, isso ocorre devido ao fato de que cada navegador utiliza uma API de acessibilidade diferente (que também se altera dependendo do sistema operacional utilizado). Se fosse utilizada uma API de acessibilidade unificada, este problema poderia ser contornado de forma mais fácil.

Criar componentes interativos acessíveis não é fácil, e dependendo dos navegadores ou sistemas operacionais utilizados durante a fase de desenvolvimento, os resultados podem variar de acordo com os leitores de tela utilizados. Logo, a melhor forma de garantir o maior nível de acessibilidade para a maior porcen-

tagem de usuários que utilizam leitores de tela, é criar componentes interativos complexos utilizando a maior variedade de leitores de tela possível para testar sua funcionalidade em sistemas operacionais onde eles são mais amplamente utilizados.

6 CONCLUSÃO E TRABALHOS FUTUROS

A proposta do trabalho foi de analisar a adequação de componentes de interface gerados pela biblioteca jQuery UI quanto aos requisitos de acessibilidade, por meio da implementação de aplicações RIA e análise do seu funcionamento com softwares leitores de tela. Durante a etapa de implementação, componentes selecionados da biblioteca jQuery UI foram analisados de acordo com as recomendações ARIA. Para cada componente obteve-se resultados distintos, que variavam não somente com a utilização de leitores de tela diferentes, mas também com a utilização de navegadores Web.

Com relação à utilização via teclado, apenas o widget de calendário obteve problema de utilização. Este tipo de navegação é bastante importante para a acessibilidade, não somente para usuários com deficiência visual, mas também para aqueles com algum tipo de deficiência motora e que, por esta razão, não conseguem utilizar o mouse como principal mecanismo de navegação. Entretanto, os widgets da biblioteca jQuery UI também precisam interagir de forma correta com leitores de tela, passando para o usuário todas as informações que ele necessita para poder navegar pela página de forma clara, sem problemas. O fato de a biblioteca não ter apresentado o resultado esperado, em que o usuário conseguisse utilizar os componentes criados recebendo feedback correto e sem limitações na utilização, mostra que muito trabalho ainda precisa ser feito nesta para se alcançar o resultado ideal.

Desenvolvedores que utilizam a biblioteca e desejam que suas páginas Web sejam acessíveis, geralmente procuram por soluções alternativas, como plugins ou até versões modificadas da biblioteca principal (não oficial) que resolvem tais problemas. Estes plugins são facilmente inseridos ao código principal, onde na maioria das vezes necessita apenas de importar o código javascript e o código CSS.

O presente trabalho levantou questões relacionadas ao motivo pelo qual alguns problemas na acessibilidade de alguns componentes que já foram resolvidos por terceiros ainda não foram inseridos à biblioteca principal, ou qual seria a forma de se fazer com que os componentes da biblioteca funcionssem de forma adequada em todos os navegadores.

Para a realização de um trabalho futuro, sugere-se a análise de mais elementos da biblioteca para se obter um resultado mais aprofundado a respeito da acessibilidade da mesma, realização de uma análise mais aprofundada dos elementos da biblioteca (verificando o código fonte), realização de testes utilizando o protótipo criado com usuários de leitores de tela, utilizar uma variedade maior de leitores de tela e sistemas operacionais (por exemplo, Linux e Mac OS) e por último, analizar o funcionamento da biblioteca em smartphones e tablets, que são plataformas cada vez mais utilizadas para acesso à páginas Web.

REFERÊNCIAS BIBLIOGRÁFICAS

ACESSIBILIDADE BRASIL. *daSilva*. 2006. Disponível em: http://www.dasilva.org.br/. Acesso em: 19/12/2013.

AUTOMATTIC INCORPORATION. *WordPress.com: The best place for your personal blog or business site*. 2005. Disponível em: http://wordpress.com/. Acesso em: 26/05/2014.

BENAVíDEZ, C.; RESTREPO, E. G. y; MCCATHIENEVILE, C. *HERA*. 2003. Disponível em: http://www.sidar.org/hera/index.php.pt?ini=info. Acesso em: 19/12/2013.

BUILTWITH PTY LTD. *BuiltWith: Trends, Intelligence and Internet Research*. Suite 206 46-48 East Esplanade Manly NSW 2095 Australia, 2013. Disponível em: http://trends.builtwith.com/javascript/jQuery. Acesso em: 15/01/2014.

CHAFFER, J.; SWEDBERG, K. *Learning jQuery*. 3. ed. [S.l.]: Packt Publishing Ltd., 2011.

CLARK, C.; OBARA, J.; MITCHELL, J.; RICHARDS, J. Usable aria: The fluid infusion component set and the relationship between aria and usability. In: INCLUSIVE DESIGN RESEARCH CENTRE. *AEGIS - ACCESSIBLE Projects*. [S.l.]: Aegis - Open Accessibility Everywhere, 2010.

COOPER, M. Accessibility of emerging rich web technologies: Web 2.0 and the semantic web. In: *Proceedings of the 2007 International Cross-disciplinary Conference on Web Accessibility (W4A)*. New York, NY, USA: ACM, 2007. (W4A '07), p. 93–98. ISBN 1-59593-590-8. Disponível em: http://doi.acm.org/10.1145/1243441.1243463.

CORMODE, G.; KRISHNAMURTHY, B. Key differences between web 1.0 and web 2.0. *First Monday*, v. 13, n. 6, 2008. ISSN 13960466. Disponível em: http://firstmonday.org/ojs/index.php/fm/article/view/2125.

FERNANDES, N.; BATISTA, A. S.; COSTA, D.; DUARTE, C.; CARRIçO., L. Three web accessibility evaluation perspectives for ria. In: *In Proceedings of 10th International Cross-Disciplinary Conference on Web Accessibility*. ACM, New York, NY, USA: [s.n.], 2013. p. 9. Disponível em: http://doi.acm.org/10.1145/2461121.2461122.

FREEDOM SCIENTIFIC. *Jaws: The World's Most Popular Windows Screen Reader*. 2014. Disponível em: http://www.freedomscientific.com/products/fs/jaws-product-page.asp. Acesso em: 27/01/2014.

GIRAUD, S.; COLOMBI, T.; RUSSO, A.; THÉROUANNE, P. Accessibility of rich internet applications for blind people: A study to identify the main problems and solutions. In: *Proceedings of the 9th ACM SIGCHI Italian Chapter International Conference on Computer-Human Interaction: Facing Complexity*. New York, NY, USA: ACM, 2011. (CHItaly), p. 163–166. ISBN 978-1-4503-0876-2. Disponível em: http://doi.acm.org/10.1145/2037296.2037335.

GOVERNO FEDERAL. *e-Mag - Modelo de Acessibilidade em Governo Eletrônico*. 2005. Portal de Governo Eletrônico do Brasil. Disponível em: http://www.governoeletronico.gov.br/acoes-e-projetos/e-MAG. Acesso em: 18/02/2013.

GOVERNO FEDERAL. *ASES - Avaliador e Simulador para a Acessibilidade de Sítios*. 2009. Disponível em: http://www.governoeletronico.gov.br/acoes-e-projetos/e-MAG/ases-avaliador-e-simulador-de-acessibilidade-sitios. Acesso em: 18/12/2013.

HENRY, S. L. *Web Accessibility: Web Standards and Regulatory Compliance*. 1. ed. [S.l.]: Apress, 2006. 1-51 p.

HéGARET, P. L. *Document Object Model (DOM)*. 2005. Disponível em: http://www.w3.org/DOM/. Acesso em: 17/12/2013.

IBGE. *Censo Demográfico 2010: Características gerais da população, religião e pessoas com deficiência*. 2010. Disponível em: ftp://ftp.ibge.gov.br/Censos/Censo_Demografico_2010/Caracteristicas_Gerais_Religiao_Deficiencia/tab1_3.pdf. Acesso em: 30/11/2013.

INCLUSIVE DESIGN RESEARCH CENTRE. *Fluid: Designing software that work - for everyone*. 2010. Disponível em: http://fluidproject.org/products/infusion/. Acesso em: 26/01/2014.

JQUERY FOUNDATION. *The jQuery Plugin Registry*. 2005. Disponível em: http://plugins.jquery.com/. Acesso em: 16/01/2014.

JUN; AUSTRALIA, V.; FAULKNER, S. *Colour Contrast Analyser*. 2003. Disponível em: http://www.visionaustralia.org/digital-access-cca. Acesso em: 19/12/2013.

LIN, S.-H.; HO, J.-M. Discovering informative content blocks from web documents. In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2002. (KDD '02), p. 588–593. ISBN 1-58113-567-X. Disponível em: http://doi.acm.org/10.1145/775047.775134.

LINAJE, M.; LOZANO-TELLO, A.; PEREZ-TOLEDANO, M. A.; PRECIADO, J. C.; RODRIGUEZ-ECHEVERRIA, R.; SANCHEZ-FIGUEROA, F. Providing ria user interfaces with accessibility properties. *Journal of Symbolic Computation*, v. 46, p. 207–217, 2010.

LINDEN, A.; FENNA, J.; DRAKOS, N. *Networked collective intelligence represents a new paradigm of work*. 2005. Disponível em: https://www.gartner.com/doc/485541/networked-collective-intelligence-represents-new. Acesso em: 17/12/2013.

MERAYO, R. V. *Rich Internet Applications (RIA) and Web Accessibility*. 2011. Hipertext.net. Disponível em: http://www.upf.edu/hipertextnet/en/numero-9/ria-and-web-accessibility.html. Acesso em: 26/01/2014.

MINSK, G. C.; POH, L. S.; WEI, H.; SIEW, T. P. Web 2.0 concepts and technologies for dynamic b2b integration. In: *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*. [S.l.: s.n.], 2007. p. 315–321.

NODA, T.; HELWIG, S. *Rich Internet Applications - Technical Comparison and Case Studies of AJAX, Flash, and Java based RIA*. 2005.

NV ACCESS. *Home of the free NVDA screen reader*. Ferny Grove, Qld 4055, Australia, 2014. Disponível em: http://www.nvaccess.org/. Acesso em: 27/01/2014.

OREILLY, T.; BATTELLE, J. *Web Squared: Web 2.0 Five Years On*. 2009. Disponível em: http://www.web2summit.com/web2009/public/schedule/detail/10194. Acesso em: 17/01/2014.

PEMBERTON, S. *HyperText Markup Language*. 2007. Disponível em: http://www.w3.org/MarkUp/. Acesso em: 17/12/2013.

PFEIFFER, S.; PARKER, C. Accessibility for the html5 <video> element. In: *Proceedings of the 2009 International Cross-Disciplinary Conference on Web Accessibililty (W4A)*. New York, NY, USA: ACM, 2009. (W4A '09), p. 98–100. ISBN 978-1-60558-561-1. Disponível em: http://doi.acm.org/10.1145/1535654.1535679.

PRIMO, A. O aspecto relacional das interações na web 2.0. *E- Compós (Brasília)*, v. 9, p. 1–21, 2007.

Q-SUCCESS WEB-BASED SERVICES. *W3Techs - Web Technology Surveys*. 2014. Disponível em: http://w3techs.com/. Acesso em: 22/10/2014.

QUIN, L. *Extensible Markup Language*. 2003. Disponível em: http://www.w3.org/XML/. Acesso em: 17/12/2013.

RESIG, J. *John Resig - JavaScJava Programmer*. 2005. Disponível em: http://ejohn.org/about/. Acesso em: 24/01/2014.

ROLLETT, H.; LUX, M.; STROHMAIER, M.; DöSINGER, G.; TOCHTERMANN, K. The web 2.0 way of learning with technologies. *Int. J. Learning Technology*, v. 3, p. 87–107, 2007.

SHNEIDERMAN, B. Universal usability: Pushing human-computer interaction research to empower every citizen. In: *Media Access: Social and Psychological Dimensions of New Technology Use*. [S.l.]: Psychology Press, 2004. p. 255.

SLATIN, J. M.; RUSH, S. *Maximum Accessibility: Making Your Web Site More Usable for Everyone*. [S.l.]: Addison-Wesley Professional, 2002. 588 p.

THE JQUERY FOUNDATION. *jQuery - Write less, do more*. 2013. Disponível em: http://jquery.com/. Acesso em: 30/11/2013.

THE PACIELLO GROUP, AOL, AEGIS. *Accessible jQuery-ui Components Demonstration*. 2012. Disponível em: http://access.aol.com/aegis/#goto_slider. Acesso em: 15/05/2014.

THIESSEN, P.; HOCKEMA, S. Wai-aria live regions: Ebuddy im as a case example. In: *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A)*. New York, NY, USA: ACM, 2010. (W4A '10), p. 33:1–33:9. ISBN 978-1-4503-0045-2. Disponível em: http://doi.acm.org/10.1145/1805986.1806030.

VOSSEN, G.; HAGEMANN, S. *Unleashing Web 2.0 - From Concepts to Creativity*. 1. ed. [S.l.]: Morgan Kaufmann, 2007. 368 p.

W3C. *Web Accessibility Initiative (WAI)*. 2003. Disponível em: http://www.w3.org/WAI/. Acesso em: 18/12/2013.

W3C. *WAI-ARIA Overview*. 2013. Disponível em: http://www.w3.org/WAI/intro/aria. Acesso em: 30/11/2013.

WATANABE, W. M.; GERALDO, R. J.; FORTES, R. P. de M. Keyboard navigation mechanisms in tab widgets: An investigation on aria's conformance. In: *Proceedings of the 29th Annual ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2014. (SAC '14), p. 721–726. ISBN 978-1-4503-2469-4. Disponível em: http://doi.acm.org/10.1145/2554850.2554947.

WIKIMEDIA FOUNDATION. *Wikipedia - The Free Encyclopedia*. 2001. Disponível em: http://www.wikipedia.org/. Acesso em: 26/05/2014.