UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ANTONIO VANDERLEI ROSA DE OLIVEIRA

PEDRO HENRIQUE SOARES DE ALMEIDA

PETERSON ANDRADE SPRENGER

ESTUDO E IMPLEMENTAÇÃO DE WIDGETS ACESSÍVEIS PARA WEB

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2014

ANTONIO VANDERLEI ROSA DE OLIVEIRA

PEDRO HENRIQUE SOARES DE ALMEIDA

PETERSON ANDRADE SPRENGER

ESTUDO E IMPLEMENTAÇÃO DE WIDGETS ACESSÍVEIS PARA WEB

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. MSc. Willian Massami Watanabe.

PONTA GROSSA

2014

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Ponta Grossa

Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

ESTUDO E IMPLEMENTAÇÃO DE WIDGETS ACESSÍVEIS PARA WEB

por

ANTONIO VANDERLEI ROSA DE OLIVEIRA
PEDRO HENRIQUE SOARES DE ALMEIDA
PETERSON ANDRADE SPRENGER

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 29 de janeiro de 2014 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. MSc. Willian Massami Watanabe
Prof. Orientador

_____
Prof. MSc. Geraldo Ranthum
Membro titular

_____
Profª. MSc. Thalita Scharr Rodrigues Pimenta
Membro titular

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

RESUMO

SPRENGER, Peterson A.; ALMEIDA, Pedro H. S. de; OLIVEIRA, Antonio V. R. de. **Estudo E Implementação De Widgets Acessíveis Para Web**. 2014. 83 f. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2014.

Este trabalho apresenta uma abordagem ao estudo das normas e diretrizes relacionadas à acessibilidade na Web, definidas pelo órgão que regulamenta a Web, a W3C. Discute os conceitos de Ajax, *Widgets*, entre outros, bem como sua relação com as chamadas RIA – *Rich Internet Application*. Apresenta os conceitos de WCAG 2.0, ARIA – *Accessible Rich Internet Applications* e acessibilidade na Web, demonstrando como ela funciona. Discute sobre o estudo da especificação ARIA, bem como sua relação com o leitor de telas NVDA – *NonVisual Desktop Access* e com a *Widget Tabpanel*. Mostra o desenvolvimento realizado pelos integrantes deste trabalho de *Widgets Tabpanel* e os resultados obtidos com essas implementações. Apresenta os conceitos de jQuery, jQuery-UI e Github, bem como a contribuição feita para o jQuery-UI com correções em seu código-fonte relacionadas a ARIA, trazendo os resultados obtidos dessa implementação. Mostra um estudo realizado sobre os sites mais populares do mundo, que visa avaliar a acessibilidade de *Widgets* nos mesmos.

Palavras-chave: Acessibilidade na Web. WAI-ARIA. WCAG. *Widgets*.

ABSTRACT

SPRENGER, Peterson A.; ALMEIDA, Pedro H. S. de; OLIVEIRA, Antonio V. R. de. **Study And Implementation Of Accessible Widgets For Web**. 2014. 83 f. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2014.

This work presents an approach to the study of standards and guidelines related to Web accessibility, defined by the agency that regulates the Web, the W3C. Discusses the concepts of Ajax, Widgets, among others, as well as their relationship with the so-called RIA – Rich Internet Application. Presents the concepts of WCAG 2.0, ARIA – Accessible Rich Internet Applications and Web accessibility, demonstrating how it works. Discusses the study of the ARIA specification, as well as its relationship with the screen reader NVDA – NonVisual Desktop Access and the TabPanel Widget. Shows the development performed by members of this work of TabPanel Widgets and the results obtained with these implementations. Presents the concepts of jQuery, jQuery-UI and Github, as well as contribution made to the jQuery-UI with corrections in your source code related to ARIA, bringing the results of this implementation. Shows a study on the most popular websites in the world, which aims to assess the accessibility of Widgets in them.

Keywords: Web Accessibility. WAI-ARIA. WCAG. Widgets.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

LISTA DE SIGLAS

CSS	*Cascading Style Sheets*
DHTML	*Dynamic HyperText Markup Language*
HTML	*HyperText Markup Language*
NVDA	*NonVisual Desktop Access*
UAAG	*User Agent Accessibility Guidelines*
URL	*Uniform Resource Locator*
W3C	*World Wide Web Consortium*
WCAG	*Web Content Accessibility Guidelines*
XMLHTTPRequest	*eXtensible Markup Language HyperText Transfer Protocol*

LISTA DE ACRÔNIMOS

AJAX	*Asynchronous JavaScript and XML*
ATAG	*Authoring Tool Accessibility Guidelines*
DOM	*Document Object Model*
RIA	*Rich Internet Application*
WAI-ARIA	*Web Accessibility Initiative-Accessible Rich Internet Applications*

SUMÁRIO

1 INTRODUÇÃO

O poder da Web está em sua universalidade. A possibilidade de ser acessada por qualquer pessoa, independentemente de suas características e habilidades, é um aspecto essencial da Web (BERNERS-LEE, 1990).

Essa afirmação reflete em portadores de necessidades especiais, que são pessoas que possuem algum tipo de deficiência física ou cognitiva. Esses indivíduos têm dificuldades na realização de determinadas atividades diárias, conforme a limitação por eles manifestada.

A acessibilidade vem ao encontro das necessidades dessas pessoas, de interagir com a Web de maneira plena, buscando nas Tecnologias de Informação a realização de trabalhos, obtenção de conhecimento, relacionamentos e lazer. Nesse contexto, estudos na área de "Acessibilidade na Web" têm como objetivo desenvolver tecnologias que permitam que pessoas com deficiências visuais, auditivas, físicas, cognitivas ou neurológicas possam perceber, interagir e navegar na Web (HENRY, 2007).

Hoje no Brasil, 23,9% da população apresenta algum tipo de deficiência (IBGE, 2010). Esse índice confirma que existem ainda muitas pessoas excluídas digitalmente em nosso país, enfatizando a necessidade de utilização de padrões de acessibilidade para o desenvolvimento de aplicações Web.

A W3C – *World Wide Web Consortium* é responsável pela elaboração das normas ATAG[1] – *Authoring Tool Accessibility Guidelines*, UAAG[2] – *User Agent Accessibility Guidelines*, WCAG[3] – *Web Content Accessibility Guidelines* e WAI-ARIA[4] – *Accessible Rich Internet Applications*, todas diretrizes relacionadas à acessibilidade.

A norma ATAG fornece diretrizes para a criação de ferramentas de autoria Web. A norma UAAG fornece as diretrizes para o desenvolvimento de agentes de usuário (navegadores, por exemplo) de forma a reduzir as barreiras de acessibilidade. A norma WCAG é um guia de boas práticas para o desenvolvimento de aplicações Web acessíveis. Por fim, a norma WAI-ARIA trata da semântica de *Widgets*, estruturas e comportamentos a fim de permitir que o agente do usuário

[1] http://www.w3.org/TR/ATAG20/
[2] http://www.w3.org/TR/UAAG20/
[3] http://www.w3.org/TR/WCAG20/
[4] http://www.w3.org/TR/wai-aria/

possa transmitir e interpretar as informações para usuários que apresentem necessidades especiais.

Pelo fato de não serem normas obrigatórias e serem recentes, na maioria das vezes sua aplicação aparece, quando não de forma escassa, de forma errônea. (FREIRE, 2009).

Implementações de acordo com as normas de acessibilidade estabelecidas pela W3C devem ser realizadas em projetos Web, para permitir a acessibilidade em aplicações Web. Uma implementação correta dos padrões presentes nas normas facilita a interação entre as aplicações e as Tecnologias Assistivas[5], como por exemplo, o leitor de telas NVDA[6] – *NonVisual Desktop Access*. (W3C, 2008a).

Nesse contexto, se faz necessário um estudo aprofundado das especificações de acessibilidade da W3C para servir de base ao desenvolvimento de *websites* e aplicações acessíveis.

1.1 OBJETIVOS

1.1.1 Objetivos Gerais

Com este trabalho, propõe-se estudar as normas e os padrões de acessibilidade definidos pela W3C, estudando e desenvolvendo *Widgets* com as especificações de acessibilidade definidas pela WAI-ARIA, visando através destes estudos contribuir para um projeto de código-fonte aberto através da correção de falhas de acessibilidade.

1.1.2 Objetivos Específicos

- Ampliar os conhecimentos na área de desenvolvimento de aplicações Web, estudando de forma aprofundada JavaScript, HTML – *HyperText Markup Language* e outros padrões Web;
- Implementar padrões de acessibilidade para a *Widget TabPanel*;
- Testar a acessibilidade de uma *Widget*;

[5] Conjunto de ferramentas, serviços, estratégias e práticas aplicadas para minimizar os problemas encontrados por pessoas com necessidades especiais.
[6] http://www.nvaccess.org/

- Estudar os conceitos de RIA – *Rich Internet Application* que utilizam de padrões Ajax, HTML, CSS – *Cascading Style Sheets* e JavaScript para entregar interações ricas e complexas aos usuários;
- Estudar e descrever os padrões de acessibilidade relacionados à RIA;
- Estudar e contribuir para um projeto de *software* livre na implementação de requisitos de acessibilidade relacionados à ARIA;
- Avaliar a acessibilidade de *Widgets* em sites populares.

1.2 JUSTIFICATIVA

Visando garantir a universalidade, ou seja, a possibilidade de que qualquer indivíduo possa acessar o conteúdo da Web sem muitas dificuldades, o estudo dos padrões de acessibilidade, bem como a sua implementação, se torna algo de fundamental importância para garantir a inclusão digital de milhares de usuários com necessidades especiais.

Com o estudo e implementação das normas de acessibilidade definidas pela W3C e com as correções criadas para a biblioteca jQuery-UI espera-se melhorar a acessibilidade nas aplicações que fazem uso dessa biblioteca. Espera-se também que o referencial teórico gerado a partir dessas implementações sirva de base para consultas e futuras implementações de *Widgets* que façam uso de padrões de acessibilidade WAI-ARIA.

1.3 ESTRUTURA DO TRABALHO

O presente trabalho encontra-se dividido em sete (7) capítulos. No capítulo seguinte (Capítulo 2) são apresentados conceitos que se encaixam no escopo do trabalho, como Web 2.0 e RIA. No Capítulo 3 são abordados conceitos relacionados especificamente à acessibilidade na Web. No Capítulo 4 são apresentados conceitos e implementações de *Widgets* acessíveis para Web, além do modelo de interação ao leitor de tela NVDA para a *Widget Tabpanel*, frequentemente utilizado como Tecnologia Assistiva por usuários que apresentam deficiência visual. No Capítulo 5 são descritas as ferramentas Github e jQuery-UI, além da contribuição realizada a esse projeto de *software* livre com correções de requisitos de acessibilidade

relacionados à ARIA – *Accessible Rich Internet Applications*. No capítulo 6 é apresentado um estudo que objetiva avaliar a acessibilidade de *Widgets* nos cem (100) sites mais populares do mundo, trazendo gráficos gerados a partir dos resultados obtidos. No Capítulo 7 as considerações finais do trabalho são apresentadas, juntamente com sugestões para trabalhos futuros.

2 WEB 2.0 COM AJAX E *WIDGETS*

A popularização e melhorias do acesso à internet fizeram com que fossem criados novos recursos de interatividade e o usuário passou a poder interagir não mais apenas como um agente passivo que só acessava o conteúdo, como também passou a produzi-lo.

As experiências de uso na Web se tornaram semelhantes às observadas em aplicações *desktop* e são resultado do emprego de um novo conjunto de tecnologias (DHTML, CSS, XMLHTTPRequest, DOM Events, etc) que permitem o desenvolvimento de interações complexas na arquitetura Web, nas chamadas RIA. Essas aplicações desenvolvem o conceito de separação de processamento, implementando interações por meio de *Widgets* no lado cliente da arquitetura Web, e comunicação assíncrona com o servidor por meio do objeto XMLHTTPRequest (MUNSON e PIMENTEL, 2008). O desenvolvimento dessas funcionalidades nas aplicações Web caracteriza o movimento Ajax (GARRETT, 2005), que por sua vez caracteriza a Web 2.0.

2.1 WEB 1.0

Nesse modelo de interação os *websites* contam apenas com elementos do tipo *link* e *input* para promover interatividade. Cada vez que o usuário manipula um componente da página Web, ele deve esperar a página recarregar por completo para perceber o resultado de suas ações (WATANABE, 2012).

Tim O'Reilly[7] considera que sites estáticos, com pouca ou nenhuma interatividade, e aplicativos que os usuários podem baixar mas não são autorizados a alterá-los ou ver como eles funcionam fazem parte da filosofia da Web 1.0.

A Web 1.0 atendeu aos seus usuários com as limitações tecnológicas de sua época, porém atualmente com os avanços nos meios de comunicação, que levaram a uma maior velocidade de acesso a Internet, fez com que novos recursos pudessem ser desenvolvidos e uma maior interatividade fosse criada, dando início a uma nova geração da Web, a Web 2.0.

[7] Creditado como criador da expressão Web 2.0. É também fundador da O'Reilly Media e entusiasta de movimentos de apoio ao *software* livre e código livre.

2.2 WEB 2.0

O termo Web 2.0 é utilizado para descrever a segunda geração da *World Wide Web*, que é caracterizada pelo fato de os usuários participarem no processo de autoria de conteúdo e pelo uso do conceito de troca de informações e colaboração com sites e serviços virtuais.

Em contraste ao modelo da Web 1.0, a ideia é criar ambientes *on-line* que facilitem e tornem mais dinâmico para os usuários colaborarem para a criação e organização de conteúdo. Anteriormente, eles apenas realizavam buscas e consultas nos conteúdos disponibilizados, enquanto que agora podem participar, inclusive, da autoria e disponibilização dos mesmos (WAMELEN e KOOL, 2008).

Alguns exemplos são a enciclopédia Wikipedia[8], que permite aos seus usuários disponibilizar e editar o conteúdo de forma colaborativa; e a oferta de serviços *on-line* oferecidos pelo Windows Live, que integra ferramenta de busca, de *e-mail*, comunicador instantâneo e programas de segurança, entre outros.

Esses tipos de sites e serviços que exploram a Web 2.0 crescem com rapidez e ganham cada vez mais adeptos.

2.3 AJAX

O acrônimo AJAX - *Asynchronous JavaScript and XML* atualmente tem seu significado estendido para muitas tecnologias além de JavaScript e XML. Ajax é definido como um estilo arquitetural composto por diferentes tecnologias e ideias relacionadas ao desenvolvimento Web (MAHEMOFF, 2006). Esse estilo define uma nova forma de interação de *websites* que pode ser observado em aplicações da Web 2.0 (WATANABE, 2012).

Com Ajax, as aplicações Web estreitam a diferença existente entre elas e as aplicações tradicionais de *desktop*. As aplicações Ajax habilitam interações ricas por meio da codificação de construções de lógica para adaptação do documento utilizando linguagens de *script* como o JavaScript (SCHMIDT et al., 2008).

O uso do Ajax resulta no total controle sobre os componentes da aplicação, permitindo definir rotinas para tratar as diferentes formas de interação do usuário

[8] http://en.wikipedia.org/wiki/Main_Page

com eles. O Ajax também implementa uma interface de controle sobre requisições remotas com o servidor (W3C, 2012a), sendo possível atualizar apenas partes de uma página Web, sem a necessidade de recarregar completamente a aplicação.

São várias as possibilidades existentes com a união desses recursos ao uso de linguagens de *script* para desenvolvimento de interações no lado cliente de uma aplicação Web.

No lado cliente da aplicação, são utilizadas tecnologias de apresentação de informação e linguagens de *script* para controle da interação, enquanto o servidor é utilizado para receber dados referentes à interação e transmitir atualizações de dados ao cliente (MUNSON e PIMENTEL, 2008).

2.4 *WIDGETS*

Widgets são aplicativos exclusivamente do lado cliente, que são criados usando padrões Web como HTML e “embalados” para distribuição. Normalmente eles são instalados em uma máquina cliente ou em um dispositivo onde são executados como aplicativos autônomos, mas eles também podem ser incorporados em páginas da Web e executados em navegadores (*browsers*[9]). Os exemplos vão desde relógios simples, cotações de ações, rodízios de notícias, jogos e meteorologias, até aplicações complexas que obtêm dados de várias fontes para serem organizados e apresentados ao usuário de uma forma interessante e útil (W3C, 2012b).

2.5 APLICAÇÕES RICAS PARA INTERNET – RIA

As características geradas pelo uso de Ajax permitem uma experiência na Web semelhante à observada em aplicações *desktop*, e as aplicações que implementam esse novo modelo de interatividade são denominadas RIA – *Rich Internet Application* (VELASCO et al., 2008).

As RIAs disponibilizam mecanismos de interação que estão diretamente ligados aos princípios de usabilidade. A separação de tarefas entre o lado servidor e o lado cliente da aplicação Web é provavelmente uma das principais razões de seu sucesso.

[9] Programa que habilita a obtenção e apresentação de páginas web

Utilizar a Web como servidor para a aplicação assegura todas as vantagens de um ambiente de baixo custo, multiplataforma e arquitetura sem a necessidade de instalação de aplicações (MUNSON e PIMENTEL, 2008); enquanto o processamento computacional alocado no cliente permite o desenvolvimento de interações ricas semelhantes às existentes em aplicações *desktop* modernas (FRATERNALI et al., 2010).

As RIAs apresentam diferenças que afetam a interação do usuário e aspectos arquiteturais da aplicação (LINAJE et al., 2009), o que dificulta seu desenvolvimento, pois as metodologias utilizadas devem ser adaptadas aos novos aspectos de interação do usuário e a separação de processamento entre cliente e servidor.

As ferramentas disponíveis no mercado ainda focam as complexidades de implementação, seguindo um paradigma arquitetural em evolução e imaturo (FRATERNALI et al., 2010).

3 FUNDAMENTAÇÃO TEORICA

3.1 ACESSIBILIDADE NA WEB

Acessibilidade é definida como um requisito relacionado à qualidade das aplicações Web (WATANABE, 2012). O conceito de acessibilidade na Web pode ser descrito como uma forma de permitir que determinada tecnologia seja disponibilizada e utilizada por qualquer pessoa, oferecendo a cada usuário interfaces que atendam suas necessidades (YUNG, 2012). Ela envolve diferentes áreas: acessibilidade no computador, acessibilidade no navegador e acessibilidade no planejamento de páginas Web.

A acessibilidade no computador utiliza-se de *softwares* para uso genérico que visam garantir o acesso aos computadores e seus periféricos, podendo ser utilizados para o acesso a Web. A acessibilidade no navegador engloba *softwares* genéricos ou específicos, que podem facilitar o acesso a diferentes tipos de usuários. Na acessibilidade no planejamento de páginas Web, se faz de fundamental importância a escolha de ferramentas que possam atender as suas composições: conteúdo, estrutura e formato.

Esses conceitos estão diretamente relacionados, respectivamente, às normas ATAG[10] – *Authoring Tool Accessibility Guidelines*, UAAG[11] – *User Agent Accessibility Guidelines* e WCAG[12] – *Web Content Accessibility Guidelines*, que possuem uma série de especificações a serem seguidas se o intuito é promover a acessibilidade.

Com a utilização da norma WCAG, por exemplo, os desenvolvedores de conteúdo Web tornam suas aplicações compreensíveis e navegáveis. Essa especificação considera não só a utilização de uma linguagem clara e simples do conteúdo textual, mas também a apresentação de meios compreensíveis para proceder à navegação entre páginas e no interior delas. A inclusão de ferramentas de navegação e orientação nas páginas é um fator promotor da acessibilidade e da facilidade de uso (W3C, 1999).

[10] http://www.w3.org/TR/ATAG20/
[11] http://www.w3.org/TR/UAAG20/
[12] http://www.w3.org/TR/WCAG20/

3.2 FUNCIONAMENTO DA ACESSIBILIDADE NA WEB

Todos os usuários, independentemente de suas limitações físicas ou cognitivas, devem ter acesso equivalente à Web. Essa afirmação, ou esse modelo de acessibilidade à Web, está representado na figura a seguir (Figura 1):

Figura 1 - Modelo de Acessibilidade na Web

Fonte: Autoria Própria

Para atender aos usuários com necessidades especiais, é necessário analisar o seu tipo de necessidade especial. No quadro a seguir (Quadro 1) é possível encontrar alguns tipos de necessidades especiais separados por categoria:

Categoria	Necessidade Especial
Visão	Cegueira, miopia, acuidade visual fraca, visão embaraçada.
Audição	Surdez.
Motora	A incapacidade de usar o *mouse*, diminuição do tempo de resposta, controle motor com limitações.

Categoria	Necessidade Especial
Cognitiva	Incapacidade do Aprendizado, distração, incapacidade de memorização ou de assimilar grandes quantidades de informações.

Quadro 1 - Tipos de Necessidades Especiais

Fonte: Autoria Própria

Para cada uma dessas categorias, se faz necessário atender a certos tipos de adaptações no projeto do conteúdo de um *website*.

A WCAG[13] fornece um conjunto de diretrizes internacionais relacionadas à acessibilidade. Essas diretrizes são a base para a maioria das leis de Acessibilidade para a Web no mundo. A versão 2.0 dessas diretrizes é baseada em quatro princípios: perceptível, operável, compreensível e robusto. Esses princípios serão detalhados na sequência.

3.3 WCAG 2.0

A WCAG 2.0[14], assim como sua precursora (WCAG 1.0), define como tornar o conteúdo Web mais acessível às pessoas com deficiência. Ela é desenvolvida através do processo da W3C em colaboração com pessoas e organizações em todo o mundo, com o objetivo de fornecer um padrão comum para a acessibilidade do conteúdo da Web que atenda às necessidades de indivíduos, organizações e governos a nível internacional (W3C, 2008a).

Embora suas diretrizes referenciem uma grande quantidade de cenários, elas podem não ser suficientes para garantir o acesso à informação por pessoas com todos os tipos, níveis e combinações de deficiências. Essas diretrizes também ajudam no desenvolvimento de um conteúdo com melhor usabilidade para os usuários em geral.

Diferentemente da WCAG 1.0, a WCAG 2.0 divide suas diretrizes em quatro princípios, que servem como características necessárias ao conteúdo Web para que o mesmo seja apresentado ao usuário (W3C, 2008b). São eles:

[13] http://www.w3.org/TR/WCAG/
[14] http://www.w3.org/TR/WCAG20/

Princípio	Descrição	Diretrizes
Perceptível	Os usuários devem ser capazes de perceber a informação sendo apresentada (o conteúdo não pode ser invisível a todos os seus sentidos).	**Alternativas textuais:** disponibilizar alternativas textuais para conteúdos não textuais. **Mídias temporais:** disponibilizar alternativas para mídias temporais. **Adaptabilidade:** criar conteúdo que possa ser disponibilizado de diferentes maneiras sem perder a informação ou estrutura. **Distinguíveis:** facilitar aos usuários ver e ouvir o conteúdo, apresentando foco ao conteúdo principal sendo disponibilizado.
Operável	Os usuários devem ser capazes de operar a interface (a interface não pode exigir interações com as quais um usuário não possa realizar).	**Acessível pelo teclado:** tornar todas as funcionalidades acessíveis pelo teclado. **Tempo suficiente:** disponibilizar tempo suficiente de leitura e utilização do conteúdo. **Apreensibilidade:** não estruturar o site com conteúdos que possam causar apreensão nos usuários. **Navegabilidade:** disponibilizar meios que auxiliem a navegação do usuário, busca por conteúdos e localização.

Princípio	Descrição	Diretrizes
Compreensível	Os usuários devem ser capazes de entender a informação e as operações da interface (o conteúdo e as operações não podem ir além do conhecimento do usuário).	**Legível e compreensível:** disponibilizar o conteúdo de forma legível e compreensível aos usuários. **Previsibilidade:** os sites devem aparecer e operar por meios previsíveis. **Assistência de entrada:** auxiliar os usuários a evitar e corrigir erros.
Robusto	Os usuários devem ser capazes de acessar o conteúdo conforme a tecnologia avança (mesmo com o avanço tecnológico e dos agentes de usuário, o conteúdo deve permanecer acessível).	**Compatibilidade:** maximizar a compatibilidade com agentes de usuário e tecnologias disponibilizadas atualmente ou no futuro.

Quadro 2 - Princípios da WCAG 2.0

Fonte: Autoria Própria

3.4 ARIA

A especificação ARIA – *Accessible Rich Internet Applications* é definida pela WAI[15] – *Web Accessibility Initiative* e apresenta um *framework* para melhorar a acessibilidade e interoperabilidade do conteúdo de aplicações Web (WATANABE, 2012).

A WAI-ARIA define uma maneira de tornar o conteúdo Web e aplicações Web mais acessíveis a pessoas com necessidades especiais. A especificação apresenta

[15] http://www.w3.org/WAI/

requisitos tecnológicos, especialmente para conteúdo dinâmico e com controle avançado de interface de usuário desenvolvido com Ajax, HTML, JavaScript, e tecnologias relacionadas. Atualmente algumas funcionalidades utilizadas em sites Web não estão disponíveis para alguns usuários com deficiência, especialmente as pessoas que contam com leitores de tela e as pessoas que não podem utilizar o *mouse*. A WAI-ARIA aborda esses desafios de acessibilidade, por exemplo, através da definição de novos caminhos para que a funcionalidade seja fornecida às Tecnologias Assistivas. Com WAI-ARIA, os desenvolvedores podem criar aplicações Web avançadas, acessíveis e utilizáveis para pessoas com necessidades especiais (W3C, 2011a).

Diferente da especificação WCAG 2.0, a WAI-ARIA considera o aspecto tecnológico envolvido no desenvolvimento de diretrizes de acessibilidade para o conteúdo Web. A WAI-ARIA apresenta recomendações e especificações de implementação para desenvolvedores de *Widgets* customizadas e outros componentes de aplicações Web. Basicamente, a especificação adiciona novas formas de identificar e habilitar funcionalidades dinâmicas através de propriedades nas *tags* HTML.

4 DESENVOLVIMENTO DE *WIDGETS TABPANEL* SEGUINDO A ESPECIFICAÇÃO ARIA

Antes de iniciar o desenvolvimento de uma *Widget* acessível, se faz necessário um estudo da especificação ARIA para *Widgets*.

A WAI-ARIA determina padrões específicos dependendo do tipo de *Widget*. *Widgets* implementadas com a especificação ARIA facilitam o acesso ao conteúdo por pessoas com necessidades especiais, além de auxiliar as Tecnologias Assistivas na interpretação do conteúdo disponível nas páginas.

Neste capítulo foi realizado o estudo aprofundado da especificação ARIA e o desenvolvimento de algumas *Widgets Tabpanel*, bem como uma validação dos requisitos de acessibilidade implementados utilizando uma ferramenta específica. Para realizar essa tarefa de validação, foi utilizada a ferramenta "aria-check"[16], a qual faz uma análise minuciosa de todos os *roles*, *states* e *properties* da *Widget Tabpanel*.

Neste capítulo também foi testada a utilização da ferramenta de Tecnologia Assistiva NVDA e foi avaliado o impacto que a implementação dos padrões ARIA acarreta para os usuários portadores de algum tipo de deficiência.

Este Capítulo está dividido em seis (6) Seções. A Seção seguinte (4.2) apresenta um estudo detalhado da especificação ARIA; a Seção 4.3 descreve a tecnologia assistiva NVDA e como ela deve ser utilizada para auxiliar os usuários que apresentam necessidades especiais a interagir com aplicações Web; a Seção 4.4 descreve a *Widget TabPanel* e seu funcionamento, bem como os detalhes de sua implementação seguindo a especificação ARIA; a Seção 4.5 apresenta algumas implementações da *Widget TabPanel* seguindo a especificação ARIA; a Seção 4.6 apresenta os resultados parciais obtidos com o desenvolvimento de *Widgets Tabpanel* seguindo a especificação ARIA.

4.1 ESTUDO DA ESPECIFICAÇÃO ARIA

A especificação adiciona novas formas de identificar e habilitar funcionalidades dinâmicas através de propriedades nas *tags* HTML. Essas propriedades são divididas em três categorias de atributos: *role*, *state* e *property*.

[16] https://github.com/watinha/aria-check

O atributo *role* (papel) permite que o desenvolvedor marque um determinado elemento HTML com informações semânticas sobre o seu comportamento como um componente de interface. Os atributos *state* (estado) e *property* (propriedade) são semelhantes, diferindo somente em relação à frequência de alteração de estados que ambos podem apresentar (WATANABE, 2012). Enquanto o atributo *state* pode ter seu valor alterado frequentemente durante uma interação com o usuário, o atributo *property* apresenta menor probabilidade de ter seu valor alterado na aplicação (W3C, 2011b).

A seguir são apresentadas algumas propriedades da especificação ARIA que devem ser levadas em consideração em uma implementação de uma *Widget* acessível (W3C, 2011c):

Propriedade	**Descrição**
aria-label	Define uma cadeia de valor que rotula o elemento atual. Fornece ao usuário um nome reconhecível de objeto.
aria-labelledby	Identifica o(s) elemento(s) que rotula(m) o elemento atual. O objetivo da *aria-labelledby* é o mesmo que o de *aria-label*.
aria-hidden	Indica que o elemento e todos os seus descendentes não são visíveis ou perceptíveis para qualquer usuário, como implementado pelo autor.
aria-controls	Identifica o(s) elemento(s) cujo conteúdo ou presença são controlados pelo elemento atual. Por exemplo: • Em tabelas de conteúdo de exibição de árvore, pode controlar o conteúdo de um painel de documentos vizinhos; • Um guia controla o visor do seu painel de aba associado.
aria-describedby	Identifica o(s) elemento(s) que descreve(m) o objeto. É muito semelhante à marcação de um objeto com *aria-labelledby*. Destina-se a fornecer detalhes adicionais que alguns usuários podem precisar.

Quadro 3 - Propriedades da Especificação ARIA

Fonte: Autoria Própria

4.2 NVDA

Devido a limitações sensoriais, cognitivas ou físicas, algumas pessoas são impossibilitadas de acessar os recursos de *hardware* ou *software* que o mundo digital oferece (HOGETOP e SANTAROSA, 2002).

Para auxiliar no processo da inclusão digital dessas pessoas, existem as Tecnologias Assistivas. Dentre essas tecnologias, podem-se citar como exemplo os *softwares* leitores de telas.

O NVDA – *NonVisual Desktop Access* é um leitor de telas para o sistema operacional Windows. Basicamente, o *software* tenta converter todo o conteúdo textual de uma página para uma saída de voz (*text-to-speech*). Atualmente, o NVDA suporta vários sintetizadores de voz e usa uma estrutura modular, que permite suportar outros sintetizadores que sejam programados. O NVDA é um programa experimental, de código aberto e está em constante desenvolvimento (SILVEIRA et al., 2007).

O NVDA é utilizado por meio do teclado, com as teclas *Tab* e *Shift* + *Tab* para mover-se para frente e para trás entre os controles e utilizando setas para navegar entre os menus. Com a utilização de um desses comandos, ou pelas teclas de atalho listadas abaixo, o programa anunciará o que estiver em foco:

- **INSERT + numpad5:** fala o objeto atual;
- **INSERT + numpad8:** move para o objeto pai (indo na direção da raiz da árvore);
- **INSERT + numpad4:** move para o objeto imediatamente anterior ao atual (no mesmo nível);
- **INSERT + numpad6:** move para o objeto imediatamente posterior ao atual (no mesmo nível);
- **INSERT + numpad2:** move para o primeiro objeto filho (o primeiro galho partindo do objeto atual);
- **INSERT + shift + numpad4:** move para o objeto anterior no curso (cruza a fronteira dos objetos pai e filho até que possa ir "voltando");
- **INSERT + shift + numpad6:** move para o próximo objeto no curso (cruza a fronteira dos objetos pai e filho até que possa ir adiante);

- **INSERT + numpadMenos:** move o foco para o objeto atual;
- **INSERT + numpadEnter:** ativa o objeto atual;
- **INSERT + numpadDivisão:** move o mouse para o objeto atual;
- **INSERT + numpadMultiplicação:** move para o mouse;
- **INSERT + shift + numpad5:** anuncia as dimensões do objeto atual em função da tela;
- **INSERT + numpadMais:** fala todos os objetos (começando do objeto atual).

Para que o NVDA possa funcionar de forma eficiente, a correta definição de alguns elementos de apoio sugeridos pela norma WCAG são de fundamental importância. Na figura a seguir (Figura 2), é ilustrada uma situação comum na construção de códigos HTML:

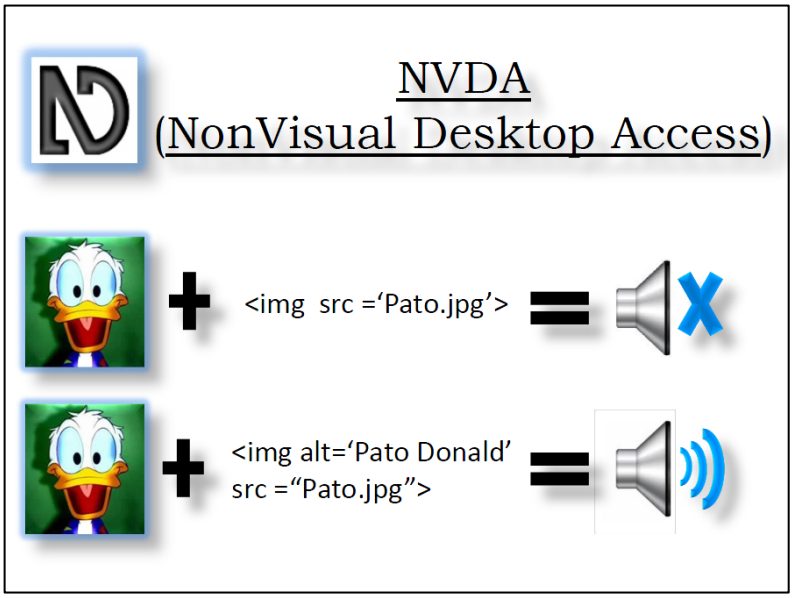

Figura 2 - Exemplo de Funcionamento do NVDA

Fonte: Autoria Própria

No primeiro caso, a imagem (elemento não textual) não possui como propriedade uma alternativa textual, o que torna sua interpretação pelo NVDA impossível. No segundo caso, onde existe a alternativa textual, o NVDA consegue interpretar esse conteúdo e transformá-lo em som.

Além desses elementos de apoio, a implementação correta de *roles*, *states* e *properties* especificados pela WAI-ARIA também auxiliam em um melhor funcionamento do NVDA.

4.3 *WIDGET TABPANEL* COM ARIA

Um *Tabpanel* é um recipiente para recursos associados a uma guia (aba). É um conjunto de páginas em camadas, onde apenas uma página é exibida de cada vez. O aspecto geral é semelhante a uma pasta de arquivo com uma "guia" que contém o título da pasta. As abas estão dispostas ao longo de uma das bordas do conteúdo, mas são encontradas mais frequentemente no topo da página (W3C, 2013). No quadro a seguir (Quadro 4) são encontrados alguns termos que facilitam o entendimento da *Widget Tabpanel*:

Termo	**Descrição**
tabbed interface component	Grupo de guias e paineis de tabulação associados.
tabpanel	Área de conteúdo associada a uma aba.
tab	Título do *tabpanel*
tablist	Grupo de abas.

Quadro 4 - Termos da *Widget Tabpanel*

Fonte: Autoria Própria

Quando o usuário ativa uma aba, o conteúdo associado a ela é disponibilizado para visualização. Essa aba permanece ativa até que outra seja ativada. Apenas a aba ativa deve entrar na ordem de tabulação[17]. Por padrão, quando um *Tabpanel* é inicializado, uma aba deve ser ativada automaticamente. Um *Tabpanel* é considerado uma *Widget* complexa, pelo fato de ser possível ativar/desativar ações e mover o ponto de relação do usuário dentro do conteúdo.

A WAI-ARIA especifica o comportamento que uma *Widget Tabpanel* deve ter em uma interação com o teclado. Para cada tecla de atalho acionada, espera-se uma resposta específica. Isso é melhor exemplificado no quadro a seguir (Quadro 5):

[17] Ordem em que os componentes da interface recebem o foco quando a tecla de tabulação (*Tab*) é pressionada.

Tecla de Atalho	Comportamento Esperado
Tab	Seta o foco somente na aba que está ativa.
Seta para esquerda ou seta para cima	Com o foco em alguma aba, esse atalho ativa e seta o foco na aba imediatamente anterior.
Seta para direita ou seta para baixo	Com o foco em alguma aba, esse atalho ativa e seta o foco na aba imediatamente posterior.
Ctrl + seta para cima	Com o foco em qualquer parte do tabpanel, esse atalho seta o foco em sua aba associada.
Ctrl + PageUp	Com o foco em qualquer parte do tabpanel, esse atalho seta o foco na aba imediatamente anterior em relação a sua aba associada.
Ctrl + PageDown	Com o foco em qualquer parte do tabpanel, esse atalho seta o foco na aba imediatamente posterior em relação a sua aba associada.

Quadro 5 - Teclas de Atalho da *Widget Tabpanel*

Fonte: Autoria Própria

A WAI-ARIA também especifica os papeis (*roles*), estados (*states*) e propriedades (*properties*) que devem ser utilizados no desenvolvimento de uma *Widget Tabpanel*. São eles (W3C, 2013):

- O *tabbed interface component* contém guias e seus paineis de conteúdo associados;
- O painel de conteúdo usa o *role* *tabpanel*;
- Um elemento com *role* *tab* é usado como um rótulo de agrupamento, fornecendo um *link* para a seleção do *tabpanel* ser passada ao usuário;
- Atribuir a relação *aria-controls* de uma aba para a identificação do seu *tabpanel*;
- Desenvolvedores devem gerenciar o estado de cada aba, alterando de acordo com o necessário a propriedade *aria-selected*;
- Um *tablist* é o *role* “raíz” para um conjunto de elementos com o atributo *role* definido como *tab*.

4.4 IMPLEMENTAÇÕES DA *WIDGET TABPANEL* COM ARIA

Abaixo seguem as implementações da *Widget TabPanel* realizadas por cada um dos integrantes deste trabalho. Serão apresentadas apenas as partes mais relevantes dos códigos fonte de HTML e JavaScript, de forma a não prejudicar o entendimento dos mesmos.

4.4.1 Implementação HTML (Antonio)

APÊNDICE A – Implementação HTML da *Widget Tabpanel* (Antonio).

4.4.2 Implementação JavaScript (Antonio)

APÊNDICE B – Implementação JavaScript da *Widget Tabpanel* (Antonio).

4.4.3 Implementação HTML (Pedro)

APÊNDICE C – Implementação HTML da *Widget Tabpanel* (Pedro).

4.4.4 Implementação JavaScript (Pedro)

APÊNDICE D – Implementação JavaScript da *Widget Tabpanel* (Pedro).

4.4.5 Implementação HTML (Peterson)

APÊNDICE E – Implementação HTML da *Widget Tabpanel* (Peterson).

4.4.6 Implementação JavaScript (Peterson)

APÊNDICE F – Implementação JavaScript da *Widget Tabpanel* (Peterson).

4.5 RESULTADOS PARCIAIS

Os estudos realizados sobre as especificações WAI-ARIA e WCAG serviram de base para o desenvolvimento das *Widgets Tabpanel*. Após várias correções no código fonte e avaliações com a ferramenta “aria-check”[18], constatou-se que a implementação dos padrões de acessibilidade não é uma tarefa simples. Tendo em vista a utilização da linguagem de programação JavaScript, que é fracamente tipada, bem como a complexidade das normas, a ocorrência de erros se torna frequente e o tempo de desenvolvimento se torna elevado.

Após a implementação dos *Tabpanels* foi utilizada a ferramenta de Tecnologia Assistiva NVDA para avaliar o impacto da utilização dos padrões de acessibilidade para os portadores de necessidades especiais. Foi possível perceber que, de fato, a utilização dos padrões de acessibilidade melhora a usabilidade do *software* e facilita o acesso à informação por pessoas com algum tipo de deficiência.

[18] https://github.com/watinha/aria-check

5 CONTRIBUINDO PARA O PROJETO JQUERY-UI

A biblioteca jQuery-UI foi desenvolvida com o objetivo de melhorar a interação dos usuários com as interfaces de sistemas que façam uso dessa biblioteca. Um dos focos da biblioteca é garantir uma melhor usabilidade para usuários com necessidades especiais.

Neste capítulo foi realizado o estudo da biblioteca JavaScript de código fonte aberto jQuery-UI, visando implementar de forma correta os padrões de acessibilidade definidos pela WAI-ARIA para a *Widget Tabpanel*, além de realizar uma contribuição ao repositório oficial do jQuery-UI com essas correções.

Para realizar essa contribuição, foi necessário efetuar um cadastro no site de repositórios de código fonte com controle de versão Github, que aloja dentre seus repositórios o repositório oficial do jQuery-UI.

O presente Capítulo está dividido em cinco (5) Seções. A Seção seguinte (5.2) apresenta os conceitos de JavaScript, jQuery e jQuery-UI; a Seção 5.3 apresenta a forma de utilização da comunidade para o desenvolvimento colaborativo de código aberto Github; a Seção 5.4 apresenta a contribuição realizada para a biblioteca jQuery-UI, corrigindo falhas na implementação dos padrões de acessibilidade segundo a especificação ARIA; a Seção 5.5 apresenta os resultados parciais obtidos com o estudo da plataforma Github e as correções realizadas na biblioteca jQuery-UI.

5.1 JQUERY-UI – PROJETO QUE ESTENDE O JQUERY

Para entender o que é jQuery e para que serve, primeiro é necessário entender o que é JavaScript e para que serve. Sendo JavaScript uma linguagem de programação que roda diretamente no navegador (lado cliente), sua utilização torna-se essencial para o desenvolvimento de páginas dinâmicas.

JavaScript foi criada para atender as necessidades de interação com a página no lado cliente, tornando desnecessário o reenvio de informações para o servidor já que o tratamento dessas informações é realizado diretamente no navegador. Pode-se citar como principal exemplo a validação de formulários, que pode ser feita no lado cliente sem que haja necessidade de enviar os dados ao servidor para serem

verificados. Para facilitar o trabalho dos desenvolvedores, foi criada uma biblioteca JavaScript chamada jQuery.

jQuery é uma biblioteca JavaScript gratuita que pode ser utilizada tanto para desenvolver projetos pessoais como comerciais. Seu lema é “escrever menos e fazer mais”, proporcionando ao desenvolvedor um conjunto de funcionalidades necessárias para atribuição de eventos, definição de efeitos, alteração ou criação de elementos na página, dentre outras infinidades de ações que visem aprimorar a usabilidade, a acessibilidade e o *design*, enriquecendo a experiência do usuário.

A biblioteca jQuery-UI proporciona abstrações de baixo nível de interação e animação, avançados efeitos especiais e de alto nível construídos em cima da biblioteca jQuery, podendo ser utilizada para construir aplicações web altamente interativas (ROSALES-MORALES et al., 2011).

Segundo John Resign[19], o foco principal da biblioteca jQuery é a simplicidade, evitando aos desenvolvedores o martírio de escrever longos e complexos códigos para criar simples efeitos.

5.2 GITHUB

O Github[20] é um repositório de código fonte com base no sistema de controle de versão Git[21]. Esse site integra uma série de características sociais; os desenvolvedores criam perfis que podem ser preenchidos com informações de identificação, incluindo uma imagem, o seu nome, *e-mail*, organização, localização e *webpage*. O perfil de um desenvolvedor é visível para outros usuários e exibe todos os repositórios que a pessoa está trabalhando e uma lista de suas atividades mais recentes no site.

O Github abriga atualmente mais de seis milhões de repositórios de código, e tem mais de 3,5 milhões de contribuintes registrados (PRESTON-WERNER, 2013). Enquanto a maioria dos projetos no Github são de desenvolvimento individual, muitos são os projetos ativos de multidesenvolvedores em escalas significativas que vêm ocorrendo há algum tempo. Cada repositório no Github tem uma página dedicada de projeto que hospeda os arquivos de código fonte, histórico de *commits*

[19] Criador da biblioteca JavaScript jQuery
[20] http://github.com/
[21] http://git-scm.com/

(alterações no código), questões abertas e outros dados associados ao projeto. Os desenvolvedores podem criar URLs – *Uniform Resource Locators* permanentes de ligação para linhas específicas dentro de um arquivo de código. Essa funcionalidade permite que as informações sobre os artefatos dentro do site fluam da comunidade Github para a Web em geral.

As ações no Github ocorrem quando uma pessoa muda um artefato ou interage com outra pessoa pelo site. Essas ações podem ser relacionadas ao código, comunicação ou inscrição. Ações em código ou associadas ao código incluem *commit*, *fork* e submeter um *pull request*. Os proprietários de projetos podem fazer *commits* alterando diretamente o conteúdo de arquivos de código. Desenvolvedores sem direito a alterações no código devem fazer um *fork* do projeto, ou seja, criar uma cópia pessoal do código que eles podem alterar livremente. Eles podem então apresentar algumas ou todas as alterações no projeto original, mediante a emissão de um *pull request*. O proprietário do projeto ou outro membro com direitos de alterações pode mesclar em suas mudanças. Os desenvolvedores também podem se comunicar por ações relacionadas com o código, através da apresentação de um comentário em um *commit*, um problema, ou um *pull request*.

O registro de todas as informações ativas combinado às inscrições do usuário permite que atualizações de atividade fluam pelo site. Os desenvolvedores podem "seguir" (*following*) outros desenvolvedores e visualizar (*watching*) outros repositórios, inscrevê-los em um *feed* de ações e comunicações dos desenvolvedores ou projetos com atualizações frequentes para projetos ativos.

Ações em artefatos também se tornam os próprios artefatos, como o histórico das ações do usuário sobre artefatos de código é registrado ao longo do tempo. O *feed* apresenta um histórico recente de *following*, *watching*, *commits*, problemas e *pull requests* de comentários. Visualizações no site, tais como a da rede, fornecem acesso ao histórico de *commits* ao longo do tempo em todos os *forks* de um determinado projeto (DABBISH et al., 2012).

5.3 CONTRIBUINDO PARA O JQUERY-UI

Milhares de programadores espalhados pelo mundo contribuem para o repositório de código fonte oficial do jQuery-UI, presente na plataforma Github, de forma a aprimorá-lo cada vez mais.

Abaixo seguem, respectivamente, o código fonte original e o código fonte com as correções relacionadas à ARIA, que corresponde à contribuição feita pelos integrantes deste trabalho ao jQuery-UI.

5.3.1 Código-fonte Original do jQuery-UI

APÊNDICE G – Código-fonte Original do jQuery-UI.

5.3.2 Código-fonte do jQuery-UI Após Correções.

APÊNDICE H – Código-fonte do jQuery-UI Após Correções.

5.4 RESULTADOS PARCIAIS

Após a realização de testes com a ferramenta “aria-check” em *Tabpanels* implementados com o código original da biblioteca jQuery-UI, foi detectado que quatro (4) padrões de acessibilidade da norma WAI-ARIA não haviam sido implementados corretamente. São eles:

- O foco deve mudar para a aba imediatamente posterior quando o usuário está dentro de um *tabpanel* e pressiona as teclas ctrl + pagedown;
- O foco deve mudar para a aba imediatamente anterior quando o usuário está dentro de um *tabpanel* e pressiona as teclas ctrl + pageup;
- Ao pressionar as teclas Ctrl + Home o foco deve mudar para a primeira aba e seu conteúdo deve ser exibido;
- Ao pressionar as teclas Ctrl + End o foco deve mudar para a última aba e seu conteúdo deve ser exibido.

Com esses erros detectados, foi realizada uma rápida análise da estrutura da biblioteca jQuery-UI, sendo possível localizar qual arquivo deveria ser alterado: “jquery.ui.tabs.js”. Uma nova análise foi feita em cima desse arquivo para isolar os trechos de código que deveriam ser atualizados. Após isso, as correções foram realizadas, como é mostrado no item 5.4.2 do presente trabalho.

Com as correções feitas, um *bug* foi registrado junto ao repositório de código fonte oficial do jQuery-UI. O registro desse *bug* serviu para indicar aos

desenvolvedores que correções deveriam ser feitas no código fonte e para identificar sobre quais correções nossa contribuição se referia. Só então foi possível submeter o código fonte corrigido ao repositório oficial.

6 AVALIAÇÃO DA ACESSIBILIDADE DE *WIDGETS* EM SITES POPULARES

Neste capítulo é apresentado um estudo que visou avaliar a acessibilidade das *Widgets* dos cem (100) sites mais acessados do mundo. Essa lista de sites foi obtida pela empresa Alexa[22], que é responsável por gerar informações sobre a Web.

Foi avaliada, especificamente, a acessibilidade de cinco (5) *Widgets*: *Accordion*, *Auto Complete*, *Menu*, *Tabpanel* e *Tooltip*, que possuem uma especificação detalhada de acessibilidade na especificação ARIA. Para efetuar essas avaliações, foi realizada uma análise manual do código fonte das páginas, inspecionando os elementos e verificando se as *Widgets* encontradas atendiam à especificação ARIA. A forma de avaliação levou em conta a implementação correta de *roles*, *states* e *properties*, além da navegação pelo teclado; todos definidos pela norma WAI-ARIA de forma específica para cada *Widget*.

Este Capítulo está dividido em seis (6) Seções. Da Seção seguinte (6.2) até a Seção 6.5, são apresentados, através de gráficos, os resultados obtidos da verificação realizada nos sites da implementação ou não dos padrões ARIA para cada *Widget*; a Seção 6.6 apresenta os resultados parciais obtidos com as avaliações da acessibilidade de *Widgets* em sites populares.

6.1 *WIDGET AUTO COMPLETE*

Dos cem (100) sites avaliados, cinquenta e quatro (54) tem a *Widget Auto Complete*. Todos os gráficos a seguir foram feitos em cima desses cinquenta e quatro (54) sites.

O gráfico apresentado a seguir (Gráfico 1) mostra na primeira coluna a quantidade de sites que não apresentam os *roles*, *states* e *properties* definidos pela norma WAI-ARIA. A segunda coluna mostra a quantidade de sites que apresentam pelo menos um (1) desses padrões estabelecidos pela norma:

[22] http://www.alexa.com/

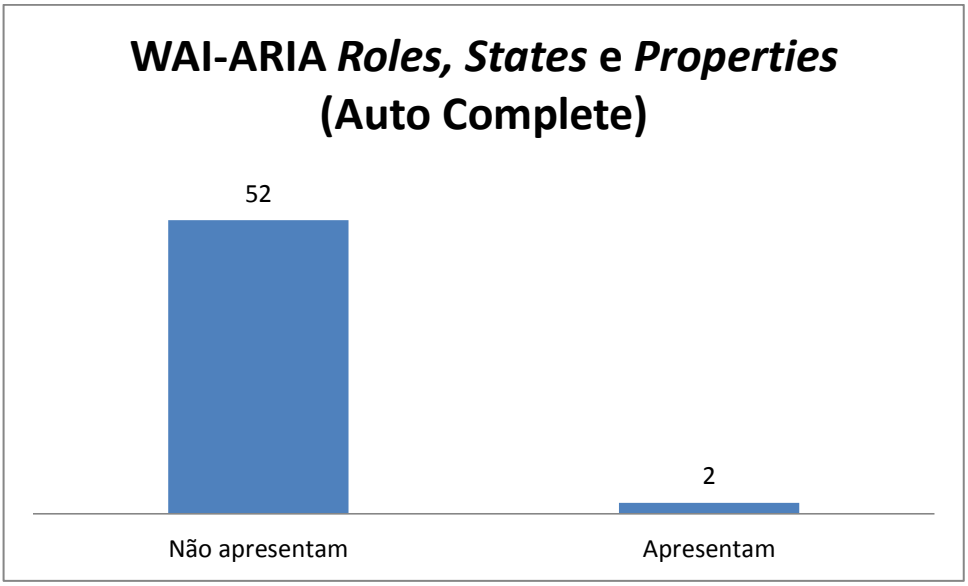

Gráfico 1 - Sites que não apresentam acessibilidade x sites que apresentam (*Auto Complete*)
Fonte: Autoria Própria

O gráfico a seguir (Gráfico 2) leva em conta a implementação de acessibilidade pelo teclado. A primeira coluna mostra a quantidade de sites classificados como inacessíveis, ou seja, que não implementam nenhuma tecla de atalho. A segunda coluna mostra a quantidade de sites classificados como parcialmente acessíveis, que são aqueles que implementam entre uma tecla de atalho até o máximo menos um (1)[23]. A terceira coluna mostra a quantidade de sites classificados como completamente acessíveis, ou seja, que implementam todas as teclas de atalho definidas pela WAI-ARIA:

[23] A *Widget Auto Complete* tem um total de quinze (15) teclas de atalho definidas pela WAI-ARIA

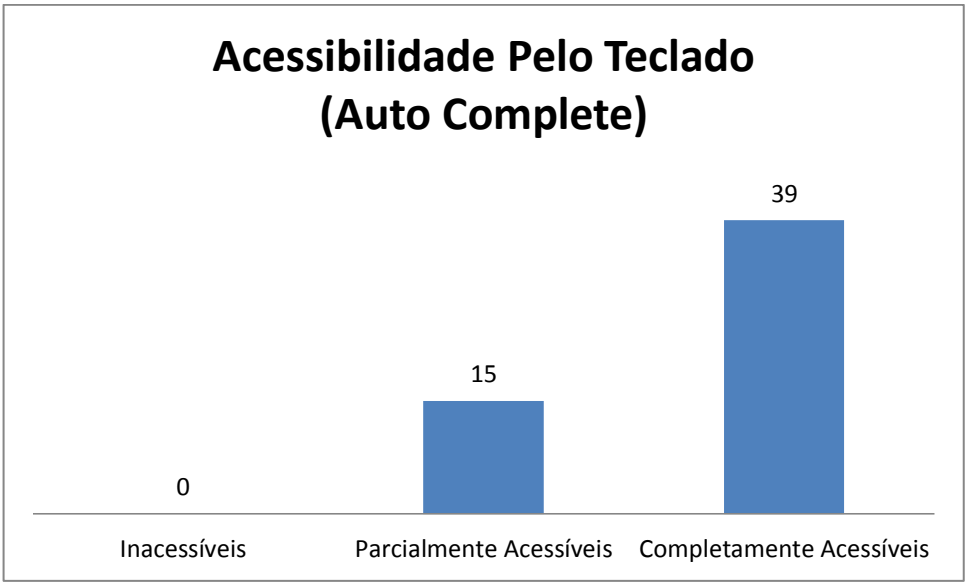

Gráfico 2 - Classificação de Acessibilidade por Teclado (*Auto Complete*)

Fonte: Autoria Própria

O gráfico a seguir (Gráfico 3) é uma representação mais detalhada do Gráfico 1. Cada coluna está baseada no total de sites, e não o conjunto delas (por isso o somatório das colunas ultrapassa os 100% (54 sites)).

A primeira coluna mostra a quantidade de sites que não apresentam os *roles*, *states* e *properties* definidos pela norma WAI-ARIA. A segunda e a terceira coluna mostram a quantidade de sites que implementam os *roles* específicos dessa *Widget* estabelecidos pela norma. A quarta coluna mostra a quantidade de sites que implementam os *states* e *properties* específicos dessa *Widget* estabelecidos pela norma. A quinta e última coluna mostra a quantidade de sites que implementam ambos os *roles*, *states* e *properties* definidos pela norma.

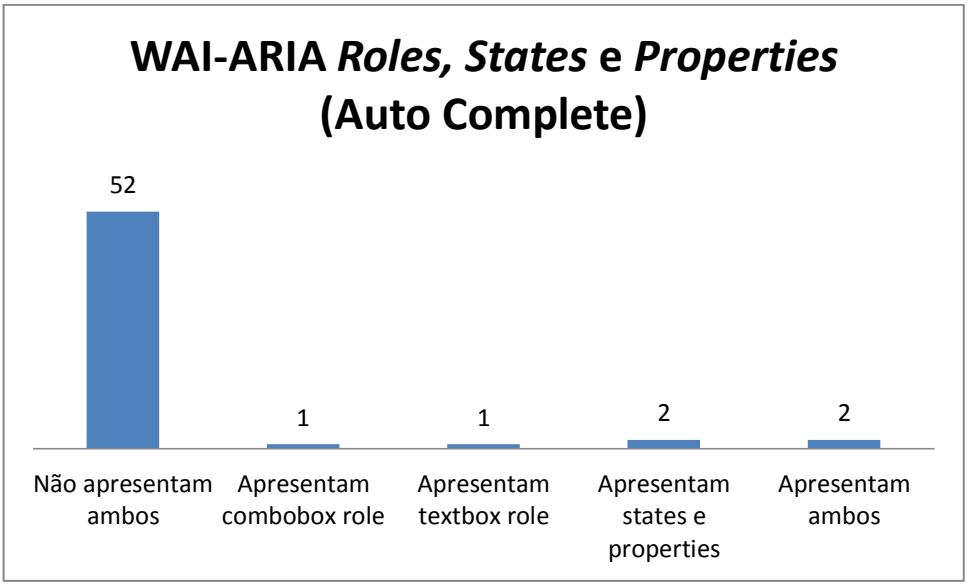

Gráfico 3 - WAI-ARIA *roles*, *states* e *properties* (*Auto Complete*)

Fonte: Autoria Própria

Baseando-se nos gráficos um (1), dois (2) e três (3), foi possível realizar uma classificação geral dos sites. Estabeleceram-se quatro níveis de acessibilidade: Inacessíveis, *Fallback*, Acessíveis e ARIA. Os sites classificados como Inacessíveis são aqueles que não apresentam nenhum dos padrões definidos pela norma WAI-ARIA. Os sites classificados como *Fallback* são aqueles que apresentam uma implementação pobre e mínima dos requisitos de acessibilidade. Os sites classificados como Acessíveis são aqueles que implementam requisitos de acessibilidade, porém não todos; e algumas vezes de forma errônea. Por fim, os sites classificados como ARIA são aqueles que implementam todos os requisitos de acessibilidade definidos pela WAI-ARIA de forma correta.

O gráfico a seguir (Gráfico 4) representa em números essas classificações:

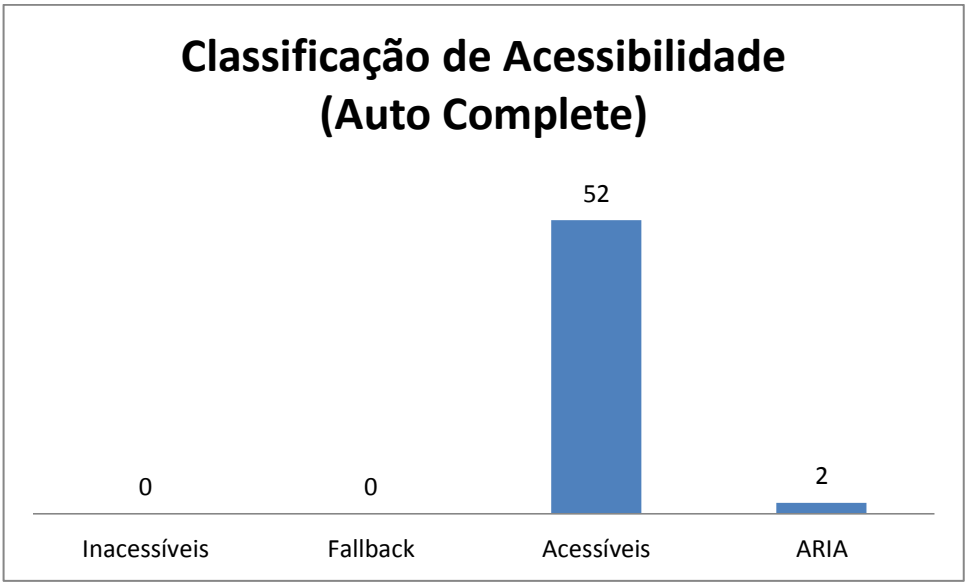

Gráfico 4 - Classificação geral de Acessibilidade (*Auto Complete*)

Fonte: Autoria Própria

6.2 *WIDGET MENU*

Dos cem (100) sites avaliados, trinta e dois (32) têm a *Widget Menu*. Todos os gráficos abaixo foram feitos em cima desses trinta e dois (32) sites.

O gráfico a seguir (Gráfico 5) mostra na primeira coluna a quantidade dos sites que não apresentam os *roles*, *states* e *properties* definidos pela norma WAI-ARIA. A segunda coluna mostra a quantidade de sites que apresentam pelo menos um (1) desses padrões estabelecidos pela norma:

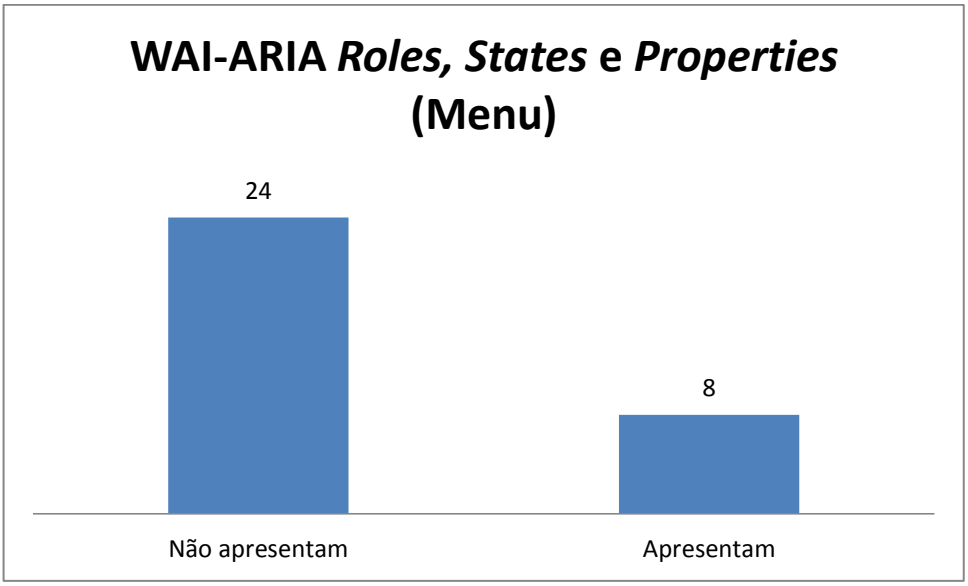

Gráfico 5 - Sites que não apresentam acessibilidade x sites que apresentam (*Menu*)
Fonte: Autoria Própria

O gráfico a seguir (Gráfico 6) leva em conta a implementação de acessibilidade pelo teclado. A primeira coluna mostra a quantidade de sites classificados como inacessíveis, ou seja, que não implementam nenhuma tecla de atalho. A segunda coluna mostra a quantidade de sites classificados como parcialmente acessíveis, que são aqueles que implementam entre uma tecla de atalho até o máximo menos um (1)[24]. A terceira coluna mostra a quantidade de sites classificados como completamente acessíveis, ou seja, que implementam todas as teclas de atalho definidas pela WAI-ARIA:

[24] A *Widget Menu* tem um total de dezesseis (16) teclas de atalho definidas pela WAI-ARIA

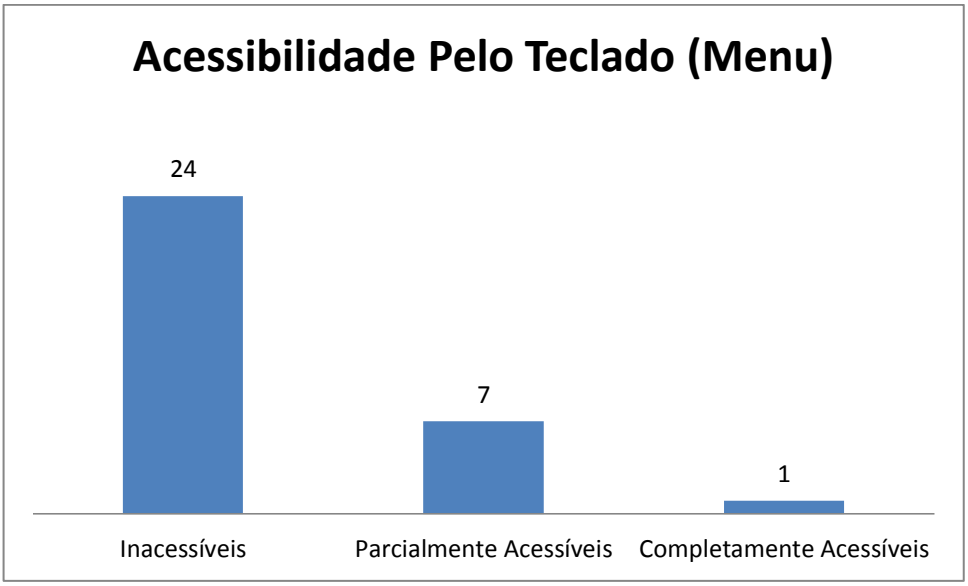

Gráfico 6 - Classificação de Acessibilidade pelo Teclado (*Menu*)

Fonte: Autoria Própria

O gráfico a seguir (Gráfico 7) é uma representação mais detalhada do Gráfico 5. Cada coluna está baseada no total de sites, e não o conjunto delas (por isso o somatório das colunas ultrapassa os 100% (32 sites)).

A primeira coluna mostra a quantidade de sites que não apresentam os *roles*, *states* e *properties* definidos pela norma WAI-ARIA. A segunda e a terceira coluna mostram a quantidade de sites que implementam os *roles* específicos dessa *Widget* estabelecidos pela norma. A quarta coluna mostra a quantidade de sites que implementam os *states* e *properties* específicos dessa *Widget* estabelecidos pela norma. A quinta e última coluna mostra a quantidade de sites que implementam ambos os *roles*, *states* e *properties* definidos pela norma.

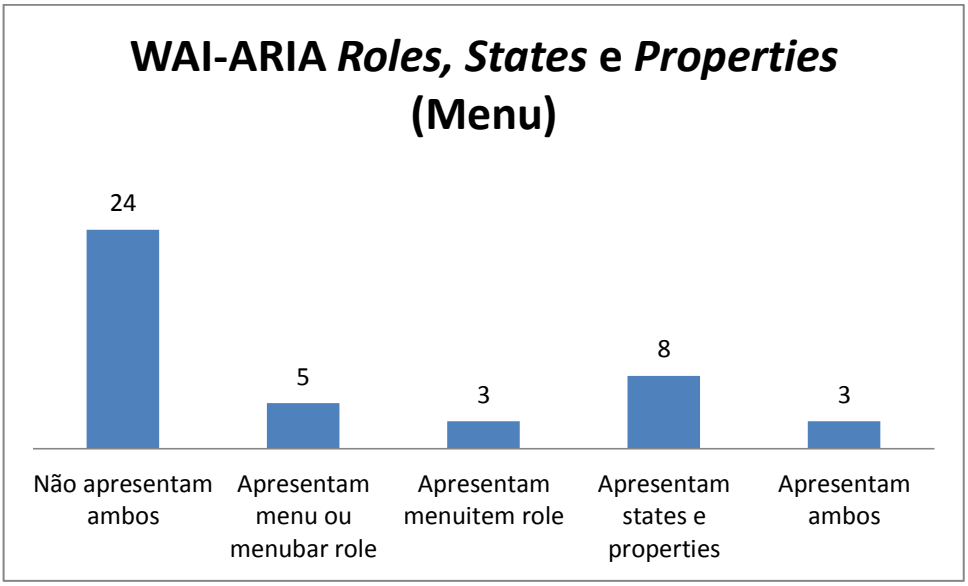

Gráfico 7 - WAI-ARIA *roles*, *states* e *properties* (*Menu*)

Fonte: Autoria Própria

Baseando-se nos gráficos cinco (5), seis (6) e sete (7), foi possível realizar uma classificação geral dos sites. Estabeleceram-se quatro níveis de acessibilidade: Inacessíveis, *Fallback*, Acessíveis e ARIA. Os sites classificados como Inacessíveis são aqueles que não apresentam nenhum dos padrões definidos pela norma WAI-ARIA. Os sites classificados como *Fallback* são aqueles que apresentam uma implementação pobre e mínima dos requisitos de acessibilidade. Os sites classificados como Acessíveis são aqueles que implementam requisitos de acessibilidade, porém não todos; e algumas vezes de forma errônea. Por fim, os sites classificados como ARIA são aqueles que implementam todos os requisitos de acessibilidade definidos pela WAI-ARIA de forma correta.

O gráfico a seguir (Gráfico 8) representa em números essas classificações:

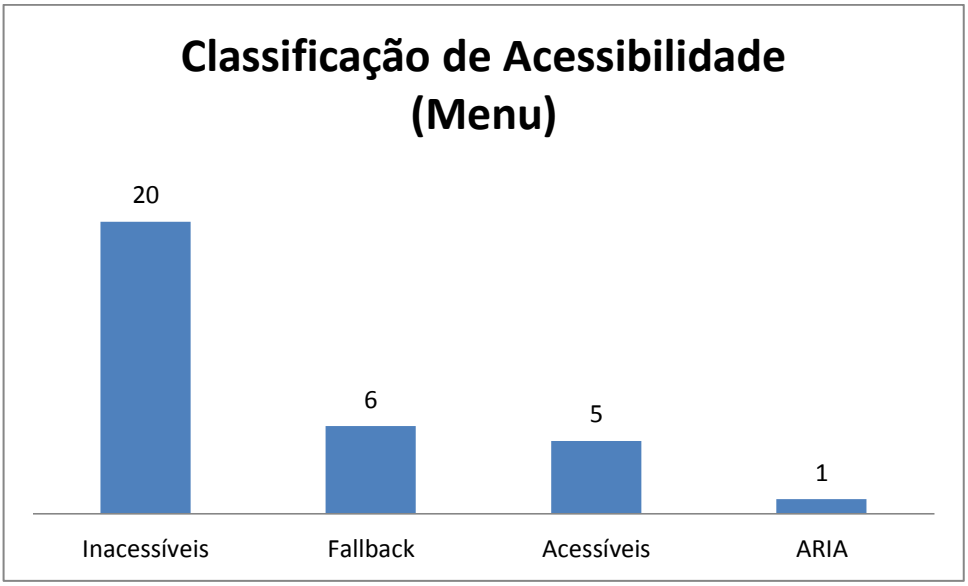

Gráfico 8 - Classificação Geral de Acessibilidade (*Menu*)

Fonte: Autoria Própria

6.3 *WIDGET TABPANEL*

Dos cem (100) sites avaliados, trinta (30) têm a *Widget Tabpanel*. Todos os gráficos abaixo foram feitos em cima desses trinta (30) sites.

O gráfico a seguir (Gráfico 9) mostra na primeira coluna a quantidade dos sites que não apresentam os *roles*, *states* e *properties* definidos pela norma WAI-ARIA. A segunda coluna mostra a quantidade de sites que apresentam pelo menos um (1) desses padrões estabelecidos pela norma:

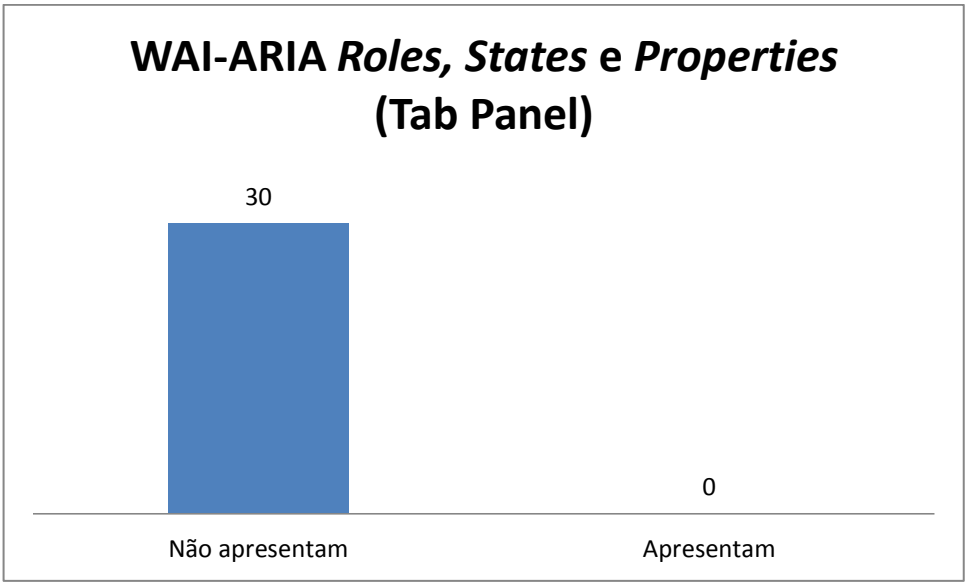

Gráfico 9 - Sites que não apresentam acessibilidade x sites que apresentam (*Tab Panel*)
Fonte: Autoria Própria

O gráfico a seguir (Gráfico 10) leva em conta a implementação de acessibilidade pelo teclado. A primeira coluna mostra a quantidade de sites classificados como inacessíveis, ou seja, que não implementam nenhuma tecla de atalho. A segunda coluna mostra a quantidade de sites classificados como parcialmente acessíveis, que são aqueles que implementam entre uma tecla de atalho até o máximo menos um (1)[25]. A terceira coluna mostra a quantidade de sites classificados como completamente acessíveis, ou seja, que implementam todas as teclas de atalho definidas pela WAI-ARIA:

[25] A *Widget Tabpanel* tem um total de quatorze (14) teclas de atalho definidas pela WAI-ARIA

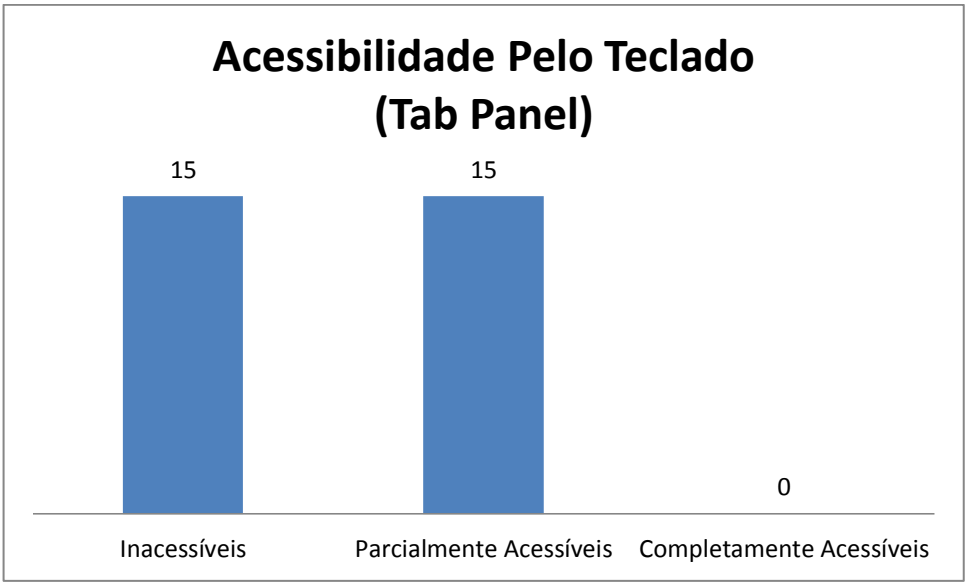

Gráfico 10 - Classificação de Acessibilidade pelo Teclado (*Tab Panel*)

Fonte: Autoria Própria

O gráfico a seguir (Gráfico 11) é uma representação mais detalhada do Gráfico 9. Cada coluna está baseada no total de sites, e não o conjunto delas (por isso o somatório das colunas ultrapassa os 100% (30 sites)).

A primeira coluna mostra a quantidade de sites que não apresentam os *roles*, *states* e *properties* definidos pela norma WAI-ARIA. A segunda e a terceira coluna mostram a quantidade de sites que implementam os *roles* específicos dessa *Widget* estabelecidos pela norma. A quarta coluna mostra a quantidade de sites que implementam os *states* e *properties* específicos dessa *Widget* estabelecidos pela norma. A quinta e última coluna mostra a quantidade de sites que implementam ambos os *roles*, *states* e *properties* definidos pela norma.

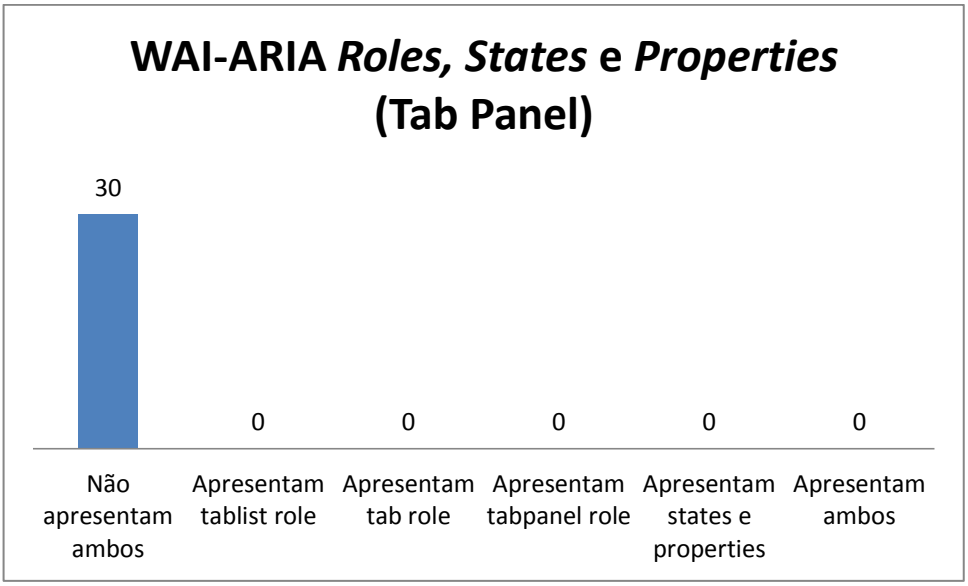

Gráfico 11 - WAI-ARIA *roles*, *states* e *properties* (*Tab Panel*)

Fonte: Autoria Própria

Baseando-se nos gráficos nove (9), dez (10) e onze (11), foi possível realizar uma classificação geral dos sites. Estabeleceram-se quatro níveis de acessibilidade: Inacessíveis, *Fallback*, Acessíveis e ARIA. Os sites classificados como Inacessíveis são aqueles que não apresentam nenhum dos padrões definidos pela norma WAI-ARIA. Os sites classificados como *Fallback* são aqueles que apresentam uma implementação pobre e mínima dos requisitos de acessibilidade. Os sites classificados como Acessíveis são aqueles que implementam requisitos de acessibilidade, porém não todos; e algumas vezes de forma errônea. Por fim, os sites classificados como ARIA são aqueles que implementam todos os requisitos de acessibilidade definidos pela WAI-ARIA de forma correta.

O gráfico a seguir (Gráfico 12) representa em números essas classificações:

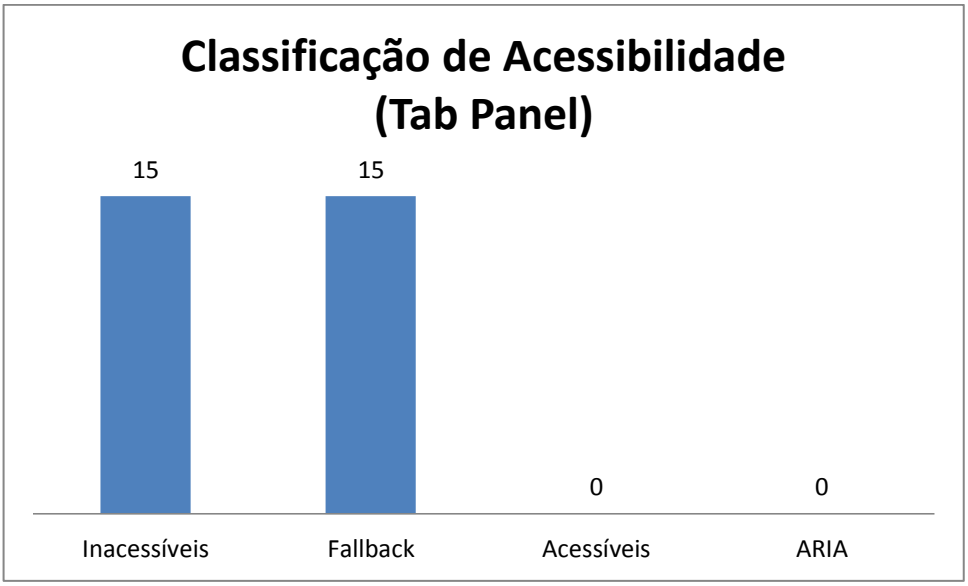

Gráfico 12 - Classificação Geral de Acessibilidade (*Tab Panel*)

Fonte: Autoria Própria

6.4 *WIDGET ACCORDION* E *WIDGET TOOLTIP*

Dos cem (100) sites avaliados, apenas um (1) tinha a *Widget Accordion* e nenhum tinha a *Widget Tooltip*. Isso leva a conclusão de que essas *Widgets* não são exploradas e/ou não são conhecidas.

A única *Widget Accordion* encontrada não apresentava nenhum dos *roles*, *states* e *properties* definidos pela norma WAI-ARIA, além de não apresentar o mínimo de acessibilidade por teclado, sendo classificada como inacessível.

6.5 RESULTADOS PARCIAIS

Após a seleção dos cem (100) sites mais visitados em toda a Web e a realização de um abrangente estudo complementar das normas WAI-ARIA especificamente para cada tipo de *Widget*, deu-se início a uma série de testes em cada site.

Efetuando uma análise detalhada do código HTML e JavaScript de cada página, foi possível constatar que a maioria dos sites não fazem a implementação dos padrões de acessibilidade, e quando a fazem, apenas uma pequena parte de suas funções é implementada seguindo os padrões da norma WAI-ARIA.

Como se pode notar, das cinco (5) *Widgets* procuradas para avaliação, duas (2) não foram encontradas em números significativos, o que mostra a falta de exploração e/ou falta de conhecimento sobre essas *Widgets*. As *Widgets* que foram comumente encontradas (*Auto Complete*, *Menu* e *Tabpanel*) têm como principal objetivo melhorar a usabilidade do site em geral, porém a maioria de suas implementações não levaram em consideração atender pessoas com necessidades especiais.

A seguir é apresentado um gráfico (Gráfico 13) gerado a partir da avaliação final das *Widgets Auto Complete*, *Menu* e *Tabpanel*, mostrando a presença dessas *Widgets* nos sites mais populares do mundo e como elas foram avaliadas de acordo com o nível de acessibilidade apresentado:

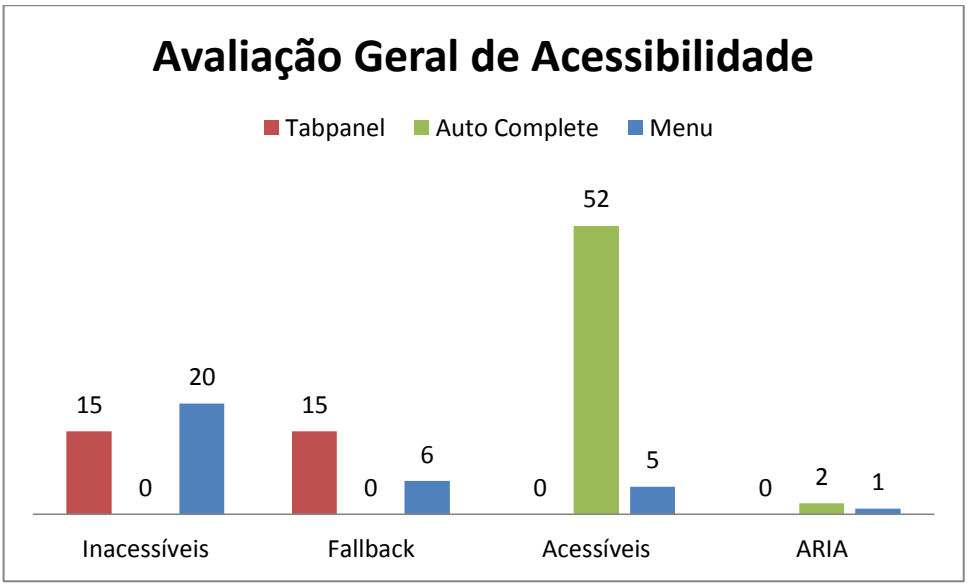

Gráfico 13 - Avaliação Geral de Acessibilidade

Fonte: Autoria Própria

É possível perceber a diferença de acessibilidade apresentada entre as *Widgets*. No *Auto Complete* a presença de acessibilidade é maior se comparada às outras *Widgets*. Isso é interessante para levantar algumas questões relacionadas ao assunto, que podem vir a serem utilizadas em pesquisas e trabalhos futuros.

7 CONSIDERAÇÕES FINAIS

Observou-se com a realização deste trabalho que a utilização das normas WCAG e WAI-ARIA é de fundamental importância para melhorar a acessibilidade e a navegação em aplicações Web, possibilitando ao usuário com necessidades especiais uma utilização mais agradável. Notou-se também que a implementação dessas normas facilita a utilização de programas leitores de tela, bem como o acesso a determinados objetos não textuais como imagens, dentre outros.

Com a utilização do NVDA foi possível simular na prática as condições que usuários com deficiência visual enfrentam ao acessar páginas Web e fazer um comparativo entre páginas que apresentam recursos de acessibilidade implementados e páginas que não apresentam esses recursos. Constatou-se que a forma de tratamento pelos leitores de tela depende de como a aplicação foi desenvolvida.

A implementação das *Widgets TabPanel* possibilitou entender, analisar e aprofundar conhecimentos referentes a navegação por setas e teclas de atalho, que para pessoas com necessidades especiais são de fundamental importância.

A contribuição para a biblioteca jQuery-UI permitiu conhecer o processo de desenvolvimento de aplicações de código aberto, bem como aprofundar conhecimentos da linguagem de programação JavaScript e ainda conhecer o funcionamento de sistemas de controle de versão Git.

Por fim, sugere-se como trabalhos futuros a publicação de um artigo tendo como base os resultados das avaliações realizadas no Capítulo 6 deste trabalho, contribuições para outros projetos de *software* livre implementando requisitos de acessibilidade de acordo com a especificação ARIA, estudos e implementações de outras *Widgets*, avaliações sobre outras *Widgets* em sites populares com relatórios das mesmas e continuação da implementação de requisitos de acessibilidade na biblioteca jQuery-UI para outras *Widgets*.

REFERÊNCIAS

BERNERS-LEE, Tim. **Information management: A proposal**. W3C archive, 1990. Disponível em: <http://www.nic.funet.fi/index/FUNET/history/internet/w3c/proposal.html>. Acesso em: 23 mai. 2013.

DABBISH, Laura; STUART, Colleen; TSAY, Jason; HERBSLEB, Jim. **Social Coding in Github: Transparency and Collaboration in an Open Software Repository**. 2012. Disponível em: <http://www.cs.cmu.edu/afs/cs/Web/People/xia/resources/Documents/cscw2012_Github-paper-FinalVersion-1.pdf>. Acesso em: 05 nov. 2013.

FRATERNALI, Piero; COMAI, Sara; BOZZON, Alessandro; CARUGHI, Giovanni T. **Engineering rich internet applications with a model-driven approach**. ACM Trans. Web, v. 4, n. 2, p. 1–47, bBBaixado, 2010. Disponível em: <http://portal.acm.org/citation.cfm?id=1734200.1734204&coll=Portal&dl=GUIDE&CFID=89708622&CFTOKEN=40265052>. Acesso em: 25 jun. 2013.

FREIRE, André P.; BITTAR, Thiago J.; FORTES, Renata P. M.. **An approach based on metrics for monitoring web accessibility in Brazilian municipalities web sites.** 2008. In *Proceedings of the 2008 ACM symposium on Applied computing* (SAC '08). ACM, New York, NY, USA, 2421-2425. DOI=10.1145/1363686.1364259 Disponível em: <http://doi.acm.org/10.1145/1363686.1364259>. Acesso em: 20 nov. 2013.

GARRETT, Jesse J. **Ajax: A new approach to web applications**. 2005. Disponível em: <http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/>. Acesso em: 20 mai. 2013.

HENRY, Shawn L. **Just Ask: Integrating Accessibility Throughout Design (Paperback)**. Madison, Lulu.com, 1430319526, 2007. Disponível em: <www.uiAccess.com/justask/>. Acesso em: 22 jul. 2013.

HOGETOP, Luisa; SANTAROSA, Lucila M. C. **Tecnologias Adaptiva/Assistiva Informáticas na Educação Especial: viabilizando a acessibilidade ao potencial**

individual. **Revista de Informática na Educação: Teoria, Prática** – Porto Alegre, Vol 5, n.2 (nov/2002), p.103-117. Disponível em: <http://www.educadores.diaadia.pr.gov.br/arquivos/File/artigos_edespecial/tecnologias_assistivas.pdf>. Acesso em: 15 ago. 2013.

IBGE. **Características gerais da população, religião e pessoas com deficiência**, 2010. Disponível em: <http://www.ibge.gov.br/home/estatistica/populacao/censo2010/caracteristicas_religiao_deficiencia/caracteristicas_religiao_deficiencia_tab_pdf.shm>. Acesso em: 22 mai. 2013.

LINAJE, Marino; PRECIADO, Juan C.; SANCHEZ-FIGUEROA, Fernando. **Domain-specific model for designing rich internet application user interfaces**. In: Computer-Aided Design of User Interfaces VI, Springer London, bBBaixado2 (pxx-linaje-f.pdf), 2009, p. 295–306. Disponível em: <http://www.springerlink.com/content/k611w3t86k730682/>. Acesso em: 07 nov. 2013.

MAHEMOFF, Michael. **Ajax design patterns**. O'Reilly Media, Inc., 2006.

MUNSON, Ethan V.; PIMENTEL, Maria G. **Specialized documents.** In: Web Accessibility, Springer London, 2008, p. 274–285 (Human-Computer Interaction Series, v.4). Disponível em: <http://www.springerlink.com/content/x226q248w51j6qg0/>. Acesso em: 20 set. 2013.

PRESTON-WERNER, Tom. **Five Years**. 2013. Disponível em: <https://github.com/blog/1470-five-years>. Acesso em: 14 dez. 2013.

ROSALES-MORALES, Viviana Y.; ALOR-HERNÁNDEZ, Giner; JUÁREZ-MARTÍNEZ, Ulises. **An Overview of Multimedia Support into JavaScript-based Frameworks for Developing RIAs**. 2011. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5749341>. Acesso em 14 out. 2013.

SCHMIDT, Kay-Uwe; DÖRFLINGER, Jörg; RAHMANI, Tirdad; SAHBI, Mehdi; STOJANOVIC, Ljiljana; THOMAS, Susan M. **An user interface adaptation architecture for rich internet applications**. In: ESWC'08: Proceedings of the 5th

European semantic web conference on The semantic web, Berlin, Heidelberg: Springer-Verlag, 2008, p. 736–750.

SILVEIRA, Clóvis; HEIDRICH, Regina O.; BASSANI, Patrícia B. S. **Avaliação das tecnologias de softwares existentes para a Inclusão Digital de Deficientes visuais através da utilização de Requisitos de qualidade.** Licenciatura em Computação – Centro Universitário Feevale. Novo Hamburgo, RS, 2007. Disponível em: <http://www.br-ie.org/pub/index.php/sbie/article/download/612/598>. Acesso em: 25 set. 2013.

VELASCO, Carlos A.; DENEV, Dimitar; STEGEMANN, Dirk; MOHAMAD, Yehya. **A web compliance engineering framework to support the development of accessible rich internet applications**. In: W4A '08: Proceedings of the 2008 international cross-disciplinary conference on Web accessibility (W4A), New York, NY, USA: ACM, 2008, p. 45–49.

W3C. **Web Content Accessibility Guidelines 1.0**. Recommendation 5-May-1999. Disponível em: <http://www.w3.org/TR/1999/WAI-WEBCONTENT-9990505/#context-and-orientation>. Acesso em: 25 nov. 2013.

W3C. **Web Content Accessibility Guidelines (WCAG) 2.0.** W3C Recommendation, 2008a. Disponível em: <http://www.w3.org/TR/WCAG20/>. Acesso em: 20 mai. 2013.

W3C. **A guide to understanding and implementing web content accessibility guidelines 2.0**. W3C Working Group Note, 2008b. Disponível em: <http://www.w3.org/TR/UNDERSTANDING-WCAG20/Overview.html>. Acesso em: 25 jun. 2013.

W3C. **WAI-ARIA Overview**, 2011a. Disponível em: <http://www.w3.org/WAI/intro/aria>. Acesso em: 27 jul. 2013.

W3C. **Accessible Rich Internet Applications - (WAI-ARIA) version 1.0**. W3C Candidate Recommendation, 2011b. Disponível em: <http://www.w3.org/TR/wai-aria/>. Acesso em: 28 jul. 2013.

W3C. **Accessible Rich Internet Applications (WAI-ARIA) 1.0**. Supported States and Properties, 2011c. Disponível em: <http://www.w3.org/TR/wai-aria/states_and_properties>. Acesso em: 28 ago. 2013.

W3C. **Xmlhttprequest**. W3C Working Draft, 2012a. Disponível em: <http://www.w3.org/TR/XMLHttpRequest/>. Acesso em: 28 set. 2013.

W3C. **Packaged Web Apps (Widgets) – Packaging and XML Configuration (Second Edition).** W3C Recommendation, 2012b. Disponível em: <http://www.w3.org/TR/widgets/>. Acesso em: 22 nov. 2013.

W3C. **An author's guide to understanding and implementing Accessible Rich Internet Applications**, 2013. Disponível em: <http://www.w3.org/TR/wai-aria-practices/#tabpanel>. Acesso em: 22 mai.2013.

WAMELEN, Johan; KOOL, Dennis. **Web 2.0: a basis for the second society?** In: Proceedings of the 2nd international conference on Theory and practice of electronic governance, New York, NY, USA: ACM, 2008, p. 349–354 (ICEGOV '08, v.1). Disponível em: <http://doi.acm.org/10.1145/1509096.1509169>. Acesso em: 24 jun. 2013.

WATANABE, Willian M. **Avaliação automática para aplicações ricas na web: um apoio aos testes de aceitação**. 2012. 62 f. Qualificação. Universidade de São Paulo. São Carlos, 2012.

YUNG, Oelinton. **Implementação dos Conceitos de Acessibilidade na Loja Ecommerce modelov2 da Empresa WebStorm Internet.** 2012. 42 f. Trabalho de Conclusão de Curso. Tecnologia em Analise e Desenvolvimento de Sistemas - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2012.

APÊNDICE A – Implementação HTML da *Widget Tabpanel* (Antonio)

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="pt-br">
        ...
        <script src = "teste.js"></script>
        ...
        <body onload="HOME()">
                ...
                <ul id="tablist" role="tablist"> <!-- rótulo-->
                        <li  role ="tab" aria-controls ="tab0" id ="K0" onclick="HOME()"
onfocus="HOME()" tabindex="0" onkeypress="onKeyPress(event,0)">HOME</li>
                        <li  role ="tab" aria-controls ="tab1" id ="K1" onclick="Semestre1()"
onfocus="Semestre1()" tabindex="-1" onkeypress="onKeyPress(event,1)">1° Semestre</li>
                        <li  role ="tab" aria-controls ="tab2" id ="K2" onclick="Semestre2()"
onfocus="Seemstre2()" tabindex="-1" onkeypress="onKeyPress(event,2)">2° Semestre</li>
                        <li  role ="tab" aria-controls ="tab3" id ="K3" onclick="Semestre3()"
onfocus="Semestre3()" tabindex="-1" onkeypress="onKeyPress(event,3)">3° Semestre</li>
                        <li  role ="tab" aria-controls ="tab4" id ="K4" onclick="Semestre4()"
onfocus="Semestre4()" tabindex="-1" onkeypress="onKeyPress(event,4)">4° Semestre</li>
                        <li  role ="tab" aria-controls ="tab5" id ="K5" onclick="Semestre5()"
onfocus="Semestre5()" tabindex="-1" onkeypress="onKeyPress(event,5)">5° Semestre</li>
                        <li  role ="tab" aria-controls ="tab6" id ="K6" onclick="Semestre6()"
onfocus="Semestre6()" tabindex="-1" onkeypress="onKeyPress(event,6)">6° Semestre</li>
                </ul>
                ...
                <img id ="foto" src="Imagem2.jpg" onclick = "mostrar()" alt="Vanderlei" />
                ...
                <div id="tab0" onkeypress="ctrlup(event,0)" role="tabpanel"  aria-labelledby="K0"> <!--
controlado pelo k0 HOme -->
                        ...
                </div>
                <div id="tab1" onkeypress="ctrlup(event,1)" role="tabpanel" aria-labelledby="K1">
                        ...
                        <li> <a href ="#"> Álgebra</a></li>
                        <li> <a href ="#"> Algoritmos</a></li>
                        ...
                </div>
                <div id="tab2" onkeypress="ctrlup(event,2)" role="tabpanel" aria-labelledby="K2">
                        ...
                        <li> <a href ="#"> Arquitetura</a></li>
                        <li> <a href ="#"> Comunicação</a></li>
                        ...
                </div>
                <div id="tab3" onkeypress="ctrlup(event,3)" role="tabpanel" aria-labelledby="K3">
                        ...
                        <li> <a href ="#"> Análise</a></li>
                        <li> <a href ="#"> IHC</a></li>
                        ...
                </div>
                <div id="tab4" onkeypress="ctrlup(event,4)" role="tabpanel" aria-labelledby="K4">
                        ...
                        <li> <a href ="#"> Análise e projeto</a></li>
                        <li> <a href ="#"> Estrutura de dados</a></li>
                        ...
                </div>
                <div id="tab5" onkeypress="ctrlup(event,5)" role="tabpanel" aria-labelledby="K5">
                        ...
                        <li> <a href ="#"> Gestão da Qualidade</a></li>
                        <li> <a href ="#"> Gestão de Sites Web</a></li>
                        ...
                </div>
```

```
                <div id="tab6" onkeypress="ctrlup(event,6)" role="tabpanel" aria-labelledby="K6">
                        ...
                        <li><a href ="#">  Análise e projeto</a></li>
                        <li><a href ="#">  Estrutura de dados</a></li>
                        ...
                </div>
                ...
                <div id="sair" class="overlay " onclick = "voltar()"></div>
        </body>
</html>
```

APÊNDICE B – Implementação JavaScript da *Widget Tabpanel* (Antonio)

```
function mostrar() {
	var imagem = document.getElementById("foto"), sair = document.getElementById("sair");
	imagem.className ="conteudo";
	sair.style.display ="block";
	sair.style.display ="imagem";
}
function voltar() {
	var imagem = document.getElementById("foto"), sair = document.getElementById("sair");
	imagem.className ="";
	sair.style.display ="none";
	sair.style.display ="";
}
function onKeyPress( event, cod ) {
	if( event.ctrlKey ) { // control
		if( event.keyCode == 36 )
			HOME();
		else
			if( event.keyCode == 35 )
				Semestre6();
	}
	if( event.keyCode == 40 || event.keyCode == 39 ) {
		if( cod == 0 )
			Semestre1();
		if( cod == 1 )
			Semestre2();
		if( cod == 2 )
			Semestre3();
		...
		if( cod == 6 )
			HOME();
	} else {
		if( event.keyCode == 37 || event.keyCode == 38 ) {
			if( cod == 0 )
				Semestre6();
			if( cod == 1 )
				HOME();
			if( cod == 2 )
				Semestre1();
			...
			if( cod == 6 )
				Semestre5();
		}
	}
}
function ctrlup( event, cod ) {
	if( event.ctrlKey ) {
		if( event.keyCode == 38 ) //up
			document.getElementById("K"+cod).focus();
		else {
			if( event.keyCode == 34 ) { //pgdn
				if( cod == 0 )
					Semestre1();
				if( cod == 1 )
					Semestre2();
				if( cod == 2 )
					Semestre3();
				...
				if( cod == 6 )
					HOME();
```

```
                        } else
                                if( event.keyCode == 33 ) { // pgup
                                        if( cod == 0 )
                                                Semestre6();
                                        if( cod == 1 )
                                                HOME();
                                        if( cod == 2 )
                                                Semestre1();
                                        ...
                                        if( cod == 6 )
                                                Semestre5();
                                }
                        }
                }
}
function HOME() {
        document.getElementById("tab0").style.display = "block";
        document.getElementById("tab1").style.display = "none";
        document.getElementById("tab2").style.display = "none";
        document.getElementById("tab3").style.display = "none";
        document.getElementById("tab4").style.display = "none";
        document.getElementById("tab5").style.display = "none";
        document.getElementById("tab6").style.display = "none";
        document.getElementById("imagem").style.display = "block";
        document.getElementById("K0").focus();
        for( var i = 0; i <= 6; i++ )
                document.getElementById("K"+i).tabIndex="-1";
        document.getElementById("K0").tabIndex="0";
}
function Semestre1() {
        document.getElementById("tab0").style.display = "none";
        document.getElementById("tab1").style.display = "block";
        document.getElementById("tab2").style.display = "none";
        document.getElementById("tab3").style.display = "none";
        document.getElementById("tab4").style.display = "none";
        document.getElementById("tab5").style.display = "none";
        document.getElementById("tab6").style.display = "none";
        document.getElementById("imagem").style.display = "none";
        document.getElementById("K1").focus();
        for( var i = 0; i <= 6; i++ )
                document.getElementById("K"+i).tabIndex="-1";
        document.getElementById("K1").tabIndex="0";
}
function Semestre2() {
        document.getElementById("tab0").style.display = "none";
        document.getElementById("tab1").style.display = "none";
        document.getElementById("tab2").style.display = "block";
        document.getElementById("tab3").style.display = "none";
        document.getElementById("tab4").style.display = "none";
        document.getElementById("tab5").style.display = "none";
        document.getElementById("tab6").style.display = "none";
        document.getElementById("imagem").style.display = "none";
        document.getElementById("K2").focus();
        for( var i = 0; i <= 6; i++ )
                document.getElementById("K"+i).tabIndex="-1";
        document.getElementById("K2").tabIndex="0";
}
...
function Semestre6() {
```

```
        document.getElementById("tab0").style.display = "none";
        document.getElementById("tab1").style.display = "none";
        document.getElementById("tab2").style.display = "none";
        document.getElementById("tab3").style.display = "none";
        document.getElementById("tab4").style.display = "none";
        document.getElementById("tab5").style.display = "none";
        document.getElementById("tab6").style.display = "block";
        document.getElementById("imagem").style.display = "none";
        document.getElementById("K6").focus();
        for( var i = 0; i <= 6; i++ )
                document.getElementById("K"+i).tabIndex="-1";
        document.getElementById("K6").tabIndex="0";
}
```

APÊNDICE C – Implementação HTML da *Widget Tabpanel* (Pedro)

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-br" lang="pt-br">
        ...
        <body onload="tab(0)">
                ...
                <div id="tabpanel_0" class="tabpanel">
                        <ul id="tablist_0" role="tablist">
                                <li id="tab_0" onkeypress="teclastablist(event, 0)" onfocus="tab(0)"
onclick="tab(0)" tabindex="0" aria-controls="panel_0" role="tab">Home</li>
                                <li id="tab_1" onkeypress="teclastablist(event, 1)" onfocus="tab(1)"
onclick="tab(1)" tabindex="-1" aria-controls="panel_1" role="tab">1º Semestre</li>
                                <li id="tab_2" onkeypress="teclastablist(event, 2)" onfocus="tab(2)"
onclick="tab(2)" tabindex="-1" aria-controls="panel_2" role="tab">2º Semestre</li>
                                <li id="tab_3" onkeypress="teclastablist(event, 3)" onfocus="tab(3)"
onclick="tab(3)" tabindex="-1" aria-controls="panel_3" role="tab">3º Semestre</li>
                                <li id="tab_4" onkeypress="teclastablist(event, 4)" onfocus="tab(4)"
onclick="tab(4)" tabindex="-1" aria-controls="panel_4" role="tab">4º Semestre</li>
                                <li id="tab_5" onkeypress="teclastablist(event, 5)" onfocus="tab(5)"
onclick="tab(5)" tabindex="-1" aria-controls="panel_5" role="tab">5º Semestre</li>
                                <li id="tab_6" onkeypress="teclastablist(event, 6)" onfocus="tab(6)"
onclick="tab(6)" tabindex="-1" aria-controls="panel_6" role="tab">6º Semestre</li>
                        </ul>
                        <div id="panel_0" aria-labelledby="tab_0" role="tabpanel"
onkeypress="teclastabpanel(event, 0)">
                                ...
                                <img class="minha-imagem" id="minha-imagem" border="1"
src="foto.jpg" alt="Pedro Henrique" onclick="show_image('minha-imagem')">
                                ...
                        </div>
                        <div id="panel_1" aria-labelledby="tab_1" role="tabpanel"
onkeypress="teclastabpanel(event, 1)">
                                ...
                                <p><a target="_blank"
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-
em-analise-de-sistemas/planos-de-ensino/1o-periodo/si31e-algebra/view">Álgebra</a></p>
                                <p><a target="_blank"
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-
em-analise-de-sistemas/planos-de-ensino/1o-periodo/si31c-algoritmos/view">Algoritmos</a></p>
                                ...
                        </div>
                        <div id="panel_2" aria-labelledby="tab_2" role="tabpanel"
onkeypress="teclastabpanel(event, 2)">
                                ...
                                <p><a target="_blank"
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-
em-analise-de-sistemas/planos-de-ensino/2o-periodo/si32d-arquitetura-de-
computadores/view">Arquitetura De Computadores</a></p>
                                <p><a target="_blank"
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-
em-analise-de-sistemas/planos-de-ensino/2o-periodo/si32f-comunicacao-de-dados/view">Comunicação
De Dados</a></p>
                                ...
                        </div>
                        <div id="panel_3" aria-labelledby="tab_3" role="tabpanel"
onkeypress="teclastabpanel(event, 3)">
                                ...
                                <p><a target="_blank"
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-
em-analise-de-sistemas/planos-de-ensino/3o-periodo/si33a-analise-de-sistemas/view">Análise De
Sistemas</a></p>
```

```
                                    <p><a target="_blank"
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-em-analise-de-sistemas/planos-de-ensino/3o-periodo/si33e-interface-humano-computador/view">Ihc Interface Humano Computador</a></p>
                                    ...
                                </div>
                                <div id="panel_4" aria-labelledby="tab_4" role="tabpanel"
onkeypress="teclastabpanel(event, 4)">
                                    ...
                                    <p><a target="_blank"
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-em-analise-de-sistemas/planos-de-ensino/4o-periodo/si34a-analise-e-projeto-orientado-a-objetos/view">Análise E Projeto Orientado A Objetos</a></p>
                                    <p><a target="_blank"
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-em-analise-de-sistemas/planos-de-ensino/4o-periodo/si34f-estrutura-de-dados/view">Estrutura De Dados</a></p>
                                    ...
                                </div>
                                <div id="panel_5" aria-labelledby="tab_5" role="tabpanel"
onkeypress="teclastabpanel(event, 5)">
                                    ...
                                    <p><a target="_blank"
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-em-analise-de-sistemas/planos-de-ensino/5o-periodo/si35e-gestao-da-qualidade/view">Gestão Da Qualidade</a></p>
                                    <p><a target="_blank"
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-em-analise-de-sistemas/planos-de-ensino/5o-periodo/si35b-gestao-de-sites-web/view">Gestão De Sites Web</a></p>
                                    ...
                                </div>
                                <div id="panel_6" aria-labelledby="tab_6" role="tabpanel"
onkeypress="teclastabpanel(event, 6)">
                                    ...
                                    <p><a href="">Atividades Complementares</a></p>
                                    <p><a target="_blank"
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-em-analise-de-sistemas/planos-de-ensino/6o-periodo/si36b-gerenciamento-de-sistemas-de-informacao/view">Gerenciamento De Sistemas De Informação</a></p>
                                    ...
                                </div>
                            </div>
                            ...
                            <div class="background_black hidden"></div>
                            <script src="tabpanel.js"></script>
                        </body>
</html>
```

APÊNDICE D – Implementação JavaScript da *Widget Tabpanel* (Pedro)

```
/* JS TabPanel */
function tab( panel_ativo ) {

	for ( var i = 0; i < 7; i++  ) {
		document.getElementById("panel_"+i).style.display = "none";
		document.getElementById("tab_"+i).className = "tabpanel";
		document.getElementById("tab_"+i).tabIndex = "-1";
	}
	document.getElementById("panel_"+panel_ativo).style.display = "block";
	document.getElementById("tab_"+panel_ativo).className = "selected";
	document.getElementById("tab_"+panel_ativo).tabIndex = "0";
	document.getElementById("tab_"+panel_ativo).focus();
}

/* JS Dialog */
var background_black = document.querySelector(".background_black"), image, old_class, first_click = true;

function show_image( id ) {
	image = document.getElementById(id);
	background_black.className = "background_black";
	if( first_click ) {
		old_class = image.className;
		first_click = false;
	}
	image.className = "space_image";
}

background_black.addEventListener("click", function () {
	background_black.className = "background_black hidden";
	image.className = old_class;
	first_click = true;
});

/* JS Teclas */
function teclastablist( event, id ) {

	if( event.ctrlKey ) { /* Control + ... */
		switch( event.keyCode ) {
			case 36:/* Home */
				tab( 0 );
				return;
			case 35:/* End */
				tab( 6 );
				return;
			default:
				return;
		}
	}
	else {
		switch( event.keyCode ) {
			case 39:/* Right */
			case 40:/* Down */
				if( id < 6 )
					tab( ++id );
				else
					tab( 0 );
				return;
			case 37:/* Left */
			case 38:/* Up */
```

```
                                        if( id > 0 )
                                                tab( --id );
                                        else
                                                tab( 6 );
                                        return;
                                default:
                                        return;
                        }
                }
}

function teclastabpanel( event, id ) {

                if( event.ctrlKey ) { /* Control + ... */
                        switch( event.keyCode ) {
                                case 38:/* Up */
                                        document.getElementById("tab_"+id).focus();
                                        return;
                                case 33:/* PageUp */
                                        if( id > 0 )
                                                tab( --id );
                                        else
                                                tab( 6 );
                                        return;
                                case 34:/* PageDown */
                                        if( id < 6 )
                                                tab( ++id );
                                        else
                                                tab( 0 );
                                        return;
                                default:
                                        return;
                        }
                }
}
```

APÊNDICE E – Implementação HTML da *Widget Tabpanel* (Peterson)

```
<!DOCTYPE HTML>
<html lang="pt-br">
        ...
        <code><script type="text/javascript" src="script.js"></script></code>
        ...
        <body>
                ...
                <div id="overlay" class="overlay" onclick ="overlay()"></div>
                <div id="conteudo" class="conteudo"><img src="foto.jpg" alt="Peterson" title="Peterson"
role="image"/></div>
                ...
                <ul id="menu" role="tablist" aria-labelledby="label_menu">
                        <li ><a href="#" onClick="tab(0)" onKeyDown="teclado(event, 0)" id="tab1"
tabindex="0"  role="tab" aria-controls="tabpanel0"  title="Home">Home</a></li>
                        <li ><a href="#" onClick="tab(1)" onKeyDown="teclado(event, 1)" id="tab2"
tabindex="-1" role="tab" aria-controls="tabpanel1"  title="1º Semestre">1º Semestre</a></li>
                        <li ><a href="#" onClick="tab(2)" onKeyDown="teclado(event, 2)" id="tab3"
tabindex="-1" role="tab" aria-controls="tabpanel2"  title="2º Semestre">2º Semestre</a></li>
                        <li ><a href="#" onClick="tab(3)" onKeyDown="teclado(event, 3)" id="tab4"
tabindex="-1" role="tab" aria-controls="tabpanel3"  title="3º Semestre">3º Semestre</a></li>
                        <li ><a href="#" onClick="tab(4)" onKeyDown="teclado(event, 4)" id="tab5"
tabindex="-1" role="tab" aria-controls="tabpanel4"  title="4º Semestre">4º Semestre</a></li>
                        <li ><a href="#" onClick="tab(5)" onKeyDown="teclado(event, 5)" id="tab6"
tabindex="-1" role="tab" aria-controls="tabpanel5"  title="5º Semestre">5º Semestre</a></li>
                        <li ><a href="#" onClick="tab(6)" onKeyDown="teclado(event, 6)" id="tab7"
tabindex="-1" role="tab" aria-controls="tabpanel6"  title="6º Semestre">6º Semestre</a></li>
                </ul>
                ...
                <div id="tabpanel0" aria-labelledby="tab1" role="tabpanel"
onKeyDown="tecladoPanel(event, 0)">
                        ...
                        <img id="foto" src="foto.jpg" alt="Peterson" aria-label="Foto Peterson"
onclick="conteudo()"/>
                        ...
                </div>
                <div id="tabpanel1" aria-labelledby="tab2" role="tabpanel"
onKeyDown="tecladoPanel(event, 1)">
                        <ul >
                                <li role="option" aria-posinset="1"><a
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-
em-analise-de-sistemas/planos-de-ensino/1o-periodo/si31a-fundamentos-da-
computacao/at_download/file">Fundamentos Da Computação</a></li>
                                <li role="option" aria-posinset="2"><a
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-
em-analise-de-sistemas/planos-de-ensino/1o-periodo/si31b-informatica-
instrumental/at_download/file">Informática Instrumental</a></li>
                                ...
                        </ul>
                </div>
                <div id="tabpanel2" aria-labelledby="tab3" role="tabpanel"
onKeyDown="tecladoPanel(event, 2)">
                        <ul >
                                <li role="option" aria-posinset="1"><a
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-
em-analise-de-sistemas/planos-de-ensino/2o-periodo/si32a-redes-de-
computadores/at_download/file">Redes De Computadores</a></li>
                                <li role="option" aria-posinset="2"><a
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-
```

```
em-analise-de-sistemas/planos-de-ensino/2o-periodo/si32b-sistemas-
operacionais/at_download/file">Sistemas Operacionais</a></li>
                                        ...
                                </ul>
                        </div>
                        <div id="tabpanel3" aria-labelledby="tab4" role="tabpanel"
onKeyDown="tecladoPanel(event, 3)">
                                <ul >
                                        <li role="option" aria-posinset="1"><a
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-
em-analise-de-sistemas/planos-de-ensino/3o-periodo/si33a-analise-de-
sistemas/at_download/file">Análise De Sistemas</a></li>
                                        <li role="option" aria-posinset="2"><a
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-
em-analise-de-sistemas/planos-de-ensino/3o-periodo/si33b-tecnologia-orientada-a-
objetos/at_download/file">Tecnologia Orientada A Objetos</a></li>
                                        ...
                                </ul>
                        </div>
                        <div id="tabpanel4" aria-labelledby="tab5" role="tabpanel"
onKeyDown="tecladoPanel(event, 4)">
                                <ul>
                                        <li role="option" aria-posinset="1"><a
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-
em-analise-de-sistemas/planos-de-ensino/4o-periodo/si34a-analise-e-projeto-orientado-a-
objetos/at_download/file">Análise E Projeto Orientado A Objetos</a></li>
                                        <li role="option" aria-posinset="2"><a
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-
em-analise-de-sistemas/planos-de-ensino/4o-periodo/si34b-gerencia-de-bancos-de-
dados/at_download/file">Gerência De Banco De Dados</a></li>
                                        ...
                                </ul>
                        </div>
                        <div id="tabpanel5" aria-labelledby="tab6" role="tabpanel"
onKeyDown="tecladoPanel(event, 5)">
                                <ul >
                                        <li role="option" aria-posinset="1"><a
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-
em-analise-de-sistemas/planos-de-ensino/5o-periodo/si35a-programacao-orientada-a-objetos-para-
internet/at_download/file">Programação Orientada A Objetos Para Internet</a></li>
                                        <li role="option" aria-posinset="2"><a
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-
em-analise-de-sistemas/planos-de-ensino/5o-periodo/si35b-gestao-de-sites-
web/at_download/file">Gestão De Sites WEB</a></li>
                                        ...
                                </ul>
                        </div>
                        <div id="tabpanel6" aria-labelledby="tab7" role="tabpanel"
onKeyDown="tecladoPanel(event, 6)">
                                <ul >
                                        <li role="option" aria-posinset="1"><a
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-
em-analise-de-sistemas/planos-de-ensino/6o-periodo/si36a-informatica-na-
sociedade/at_download/file">Informática Na Sociedade</a></li>
                                        <li role="option" aria-posinset="2"><a
href="http://www.utfpr.edu.br/pontagrossa/cursos/tecnologias/Ofertados-neste-Campus/tecnologia-
em-analise-de-sistemas/planos-de-ensino/6o-periodo/si36b-gerenciamento-de-sistemas-de-
informacao/at_download/file">Gerenciamento De Sistemas De Informação</a></li>
                                        ...
```

```
                    </ul>
                </div>
        ...
        </body>
</html>
```

APÊNDICE F – Implementação JavaScript da *Widget Tabpanel* (Peterson)

```
function tab( op ) {
	for( var i = 0; i <= 6; i++ ) {
		var tab = "tabpanel"+i;
		if( op == i ) {
			document.getElementById(tab).style.display = "block";
			document.getElementsByTagName("nav")[0].getElementsByTagName("li")[i].id
= "tab_ativo";
			document.getElementsByTagName("nav")[0].getElementsByTagName("a")[i].set
Attribute('tabindex','0');
		}
		else {
			document.getElementById(tab).style.display = "none";
			document.getElementsByTagName("nav")[0].getElementsByTagName("li")[i].id
= "";
			document.getElementsByTagName("nav")[0].getElementsByTagName("a")[i].set
Attribute('tabindex','-1');
		}
	}
}
//***********LightBox*************************************************************
function overlay() {
	document.getElementById('conteudo').style.display='none';
	document.getElementById('foto').style.display='block';
	document.getElementById('overlay').style.display='none';
}

function conteudo() {
	document.getElementById('foto').style.display='none';
	document.getElementById('conteudo').style.display='block';
	document.getElementById('overlay').style.display='block';
}
//***********Teclas Atalho Tabs***************************************************
function teclado( event, t ) {

	if( ( event.ctrlKey ) && ( event.keyCode == 33 ) ) { //Ctrl + PageUP
		t = t-1;
		if( t == -1 )
			t = 6;
		tab(t);
		document.getElementsByTagName("nav")[0].getElementsByTagName("a")[t].focus();
	}
	if( ( event.ctrlKey ) && ( event.keyCode == 34 ) ) { //Ctrl + PageDown
		t = t+1;
		if( t == 7 )
			t = 0;
		tab(t);
		document.getElementsByTagName("nav")[0].getElementsByTagName("a")[t].focus();
	}
	if( ( event.keyCode == 37 ) || ( event.keyCode == 38 ) ) { //Seta Esquerda ou Para Cima
		t = t-1;
		if( t == -1 )
			t = 6;
		tab(t);
		document.getElementsByTagName("nav")[0].getElementsByTagName("a")[t].focus();
	}
	if( ( event.keyCode == 39 ) || ( event.keyCode == 40 ) ) { //Seta Direita ou Para Baixo
		t = t+1;
		if( t == 7 )
			t = 0;
```

```
                tab(t);
                document.getElementsByTagName("nav")[0].getElementsByTagName("a")[t].focus();
        }
        if( ( event.ctrlKey ) && ( event.keyCode == 36 ) ) { //Ctrl + Home
                tab(0);
                document.getElementsByTagName("nav")[0].getElementsByTagName("a")[0].focus();
        }
        if( ( event.ctrlKey ) && ( event.keyCode == 35 ) ) { //Ctrl + End
                tab(6);
                document.getElementsByTagName("nav")[0].getElementsByTagName("a")[6].focus();
        }
}
//*************Teclas De Atalho Panels*************************************************************
function tecladoPanel( event, t ) {

        if( ( event.ctrlKey ) && ( event.keyCode == 33 ) ) { //Ctrl + PageUP
                t = t-1;
                if( t == -1 )
                        t = 6;
                tab(t);
                document.getElementsByTagName("nav")[0].getElementsByTagName("a")[t].focus();
        }
        if( ( event.ctrlKey ) && ( event.keyCode == 34 ) ) { //Ctrl + PageDown
                t = t+1;
                if( t == 7 )
                        t = 0;
                tab(t);
                document.getElementsByTagName("nav")[0].getElementsByTagName("a")[t].focus();
        }
        if( ( event.ctrlKey ) && ( event.keyCode == 38 ) ) { //Ctrl + SetaPara Cima
                tab(t);
                document.getElementsByTagName("nav")[0].getElementsByTagName("a")[t].focus();
        }
}
```

APÊNDICE G – Código-fonte Original do jQuery-UI

```
...
_tabKeydown: function( event ) {
	var focusedTab = $( this.document[0].activeElement ).closest( "li" ),
		selectedIndex = this.tabs.index( focusedTab ),
		goingForward = true,

	if ( this._handlePageNav( event ) ) {
		return;
	}

	switch ( event.keyCode ) {
		case $.ui.keyCode.RIGHT:
		case $.ui.keyCode.DOWN:
			selectedIndex++;
			break;
		case $.ui.keyCode.UP:
		case $.ui.keyCode.LEFT:
			goingForward = false;
			selectedIndex--;
			break;
		case $.ui.keyCode.END:
			selectedIndex = this.anchors.length - 1;
			break;
		case $.ui.keyCode.HOME:
			selectedIndex = 0;
			break;
		case $.ui.keyCode.SPACE:
			// Activate only, no collapsing
			event.preventDefault();
			clearTimeout( this.activating );
			this._activate( selectedIndex );
			return;
		case $.ui.keyCode.ENTER:
			// Toggle (cancel delayed activation, allow collapsing)
			event.preventDefault();
			clearTimeout( this.activating );
			// Determine if we should collapse or activate
			this._activate( selectedIndex === this.options.active ? false : selectedIndex );
			return;
		default:
			return;
	}

	// Focus the appropriate tab, based on which key was pressed
	event.preventDefault();
	clearTimeout( this.activating );
	selectedIndex = this._focusNextTab( selectedIndex, goingForward );

	// Navigating with control key will prevent automatic activation
	if ( !event.ctrlKey ) {
		// Update aria-selected immediately so that AT think the tab is already selected.
		// Otherwise AT may confuse the user by stating that they need to activate the tab,
		// but the tab will already be activated by the time the announcement finishes.
		focusedTab.attr( "aria-selected", "false" );
		this.tabs.eq( selectedIndex ).attr( "aria-selected", "true" );

		this.activating = this._delay(function() {
			this.option( "active", selectedIndex );
		}, this.delay );
```

```
		}
},

_panelKeydown: function( event ) {

		if ( this._handlePageNav( event ) ) {
			return;
		}

		// Ctrl+up moves focus to the current tab
		if ( event.ctrlKey && event.keyCode === $.ui.keyCode.UP ) {
			event.preventDefault();
			this.active.focus();
		}
},

// Alt+page up/down moves focus to the previous/next tab (and activates)
_handlePageNav: function( event ) {
		if ( event.altKey && event.keyCode === $.ui.keyCode.PAGE_UP ) {
			this._activate( this._focusNextTab( this.options.active - 1, false ) );
			return true;
		}
		if ( event.altKey && event.keyCode === $.ui.keyCode.PAGE_DOWN ) {
			this._activate( this._focusNextTab( this.options.active + 1, true ) );
			return true;
		}
},
...
```

APÊNDICE H – Código-fonte do jQuery-UI Após Correções

```
...
_tabKeydown: function( event ) {
	var focusedTab = $( this.document[0].activeElement ).closest( "li" ),
		selectedIndex = this.tabs.index( focusedTab ),
		goingForward = true;

	if ( this._handleTabNav( event ) ) {
		return;
	}

	switch ( event.keyCode ) {
		case $.ui.keyCode.RIGHT:
		case $.ui.keyCode.DOWN:
			selectedIndex++;
			break;
		case $.ui.keyCode.UP:
		case $.ui.keyCode.LEFT:
			goingForward = false;
			selectedIndex--;
			break;
		case $.ui.keyCode.SPACE:
			// Activate only, no collapsing
			event.preventDefault();
			clearTimeout( this.activating );
			this._activate( selectedIndex );
			return;
		case $.ui.keyCode.ENTER:
			// Toggle (cancel delayed activation, allow collapsing)
			event.preventDefault();
			clearTimeout( this.activating );
			// Determine if we should collapse or activate
			this._activate( selectedIndex === this.options.active ? false : selectedIndex );
			return;
		default:
			return;
	}

	// Focus the appropriate tab, based on which key was pressed
	event.preventDefault();
	clearTimeout( this.activating );
	selectedIndex = this._focusNextTab( selectedIndex, goingForward );

	// Navigating with ctrl key will prevent automatic activation (except for ctrl + home/end keys)
	if ( !event.ctrlKey ) {
		// Update aria-selected immediately so that AT think the tab is already selected.
		// Otherwise AT may confuse the user by stating that they need to activate the tab,
		// but the tab will already be activated by the time the announcement finishes.
		focusedTab.attr( "aria-selected", "false" );
		this.tabs.eq( selectedIndex ).attr( "aria-selected", "true" );
			this.activating = this._delay(function() {
			this.option( "active", selectedIndex );
		}, this.delay );
	}
},

_panelKeydown: function( event ) {
	if ( event.ctrlKey ) {
		switch ( event.keyCode ) {
			case $.ui.keyCode.UP:
```

```
                                        // Ctrl+up moves focus to the current tab
                                        event.preventDefault();
                                        this.active.focus();
                                        return;
                                case $.ui.keyCode.PAGE_UP:
                                        // Ctrl+pgup moves focus to the previous tab (and activates)
                                        this._activate( this._focusNextTab( this.options.active - 1, false ) );
                                        return;
                                case $.ui.keyCode.PAGE_DOWN:
                                        // Ctrl+pgdn moves focus to the next tab (and activates)
                                        this._activate( this._focusNextTab( this.options.active + 1, true ) );
                                        return;
                                default:
                                        return;
                        }
                }
},

// Ctrl+ home/end moves focus to the first/last tab (and activates)
_handleTabNav: function( event ) {
        if ( event.ctrlKey && event.keyCode === $.ui.keyCode.HOME ) {
                this._activate( this._focusNextTab( 0, true ) );
                return true;
        }
        if ( event.ctrlKey && event.keyCode === $.ui.keyCode.END ) {
                this._activate( this._focusNextTab( this.anchors.length - 1, true ) );
                return true;
        }
},
...
```