UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
RESOLUÇÃO DE PROBLEMAS II - ACH 0042

BRUNO IMPOSSINATO MUROZAKI, RODRIGO GUERRA

Acessibilidade de paginas WEB para usuários com deficiências visuais e auditivas

São Paulo
2016

BRUNO IMPOSSINATO MUROZAKI, RODRIGO GUERRA

Acessibilidade de paginas WEB para usuários com deficiências visuais e auditivas

Projeto de pesquisa apresentado à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo como requisito para a disciplina Resolução de Problemas II - ACH 0042.

Orientador: Profa. Dra. Sarajane Marques Peres

São Paulo

2016

Resumo

Dentro do mundo da Internet, muito se discute sobre elementos que tornam a vida do usuário comum mais fácil, com páginas desenhadas de forma acessível. Mas existem ainda problemas relacionados a usuários que possuem inabilidades específicas. Traremos neste relatório as dificuldades e os problemas que pessoas com deficiências visuais e auditivas têm ao acessar o conteudo na WEB, bem como os passos para o desenvolvimento de aplicações que facilitam o uso da internet para estas pessoas.

Palavras-chaves: Internet. Deficiência. Auditiva. Visual. Acesso. Conteúdo. Acessibilidade.

Abstract

Write here the English version of your “Resumo”. Example text, example text.

Keywords: Keyword1. Keyword2. Keyword3. Etc.

Lista de figuras

Figura 1 – Arquitetura geral do sistema 18
Figura 2 – Layout da página protótipo 20

Lista de algoritmos

Algoritmo 1 – Exemplo de menu em um modelo HTML anterior ao HTML5 13
Algoritmo 2 – Exemplo de menu em HTML5 13
Algoritmo 3 – Exempo de chamada simples feita à API ResponsiveVoiceJS 16
Algoritmo 4 – Exempo de hierarquia ideal de uma página HTML5 18
Algoritmo 5 – Exempo de seleção simples de conteúdo principal de um artigo WEB . . . 19
Algoritmo 6 – Seleção dos elementos HTML de acordo com o dominio da página 20

Sumário

1 Introdução 8

2 Objetivos e Hipótese 10

3 Ferramentas de Acessibilidade para Usuários com Deficiência Visual e Auditiva 11

3.1 Linguagens de Sinais 11

3.2 Simplifcação de Texto utilizando Processamento de Linguagem Natural 11

4 Padronização WEB 12

4.1 Coesão textual e visual 12

4.2 Padrões HTML5 12

5 Acessibilidade para Usuários com Deficiência Visual 15

5.1 Uso do text-to-speech 15

5.1.1 Ferramentas utilizadas 15

5.1.2 Servidor NodeJS 16

5.2 Arquitetura do protótipo 17

5.2.1 Seleçao do Conteúdo 18

5.2.2 Dificuldades na seleção do conteúdo 19

5.2.3 Melhorias na aplicação 19

6 Mais uma outra seção primária 22

6.1 Uma seção secundária 22

6.1.1 Uma seção terciária 22

6.1.2 Outra seção terciária 22

6.1.3 Mais uma seção terciária 22

6.2 Outra seção secundária 22

6.3 Mais uma seção secundária 22

7 Conclusão 23

Referências[1] 24

[1] De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

1 Introdução

O estudo de acessibilidade de ferramentas WEB é muito amplo. Existem diversos estudos e padrões definidos dentro da literatura que definem o que é ou não acessível. A entidade reguladora de padrões WEB, a *World Wide WEB Consortium* (W3C) especifica em seu guia diversos itens que facilitam o acesso do usuário, através a Iniciativa de Acessibilidade WEB W3C (2005c).

Mas há ainda na literatura uma falta de análise mais profunda quanto a questão de acessibilidade para com os usuários com algum tipo de deficiência ou inabilidade específica. A própria W3C, através da mesma Iniciativa de Acessibilidade WEB (WCAG) traz alguns pontos sobre como pessoas com inabilidades utilizam a WEB W3C (2005b).

Com o passar dos anos, algumas ferramentas isoladas permitem com que haja a inclusão por parte dos usuários com algumas inabilidades específicas. As ferramentas de leituras de texto (*Text To Speech*) por exemplo, dão ao usuário a chance de reproduzir o texto de uma forma que usuários com deficiência visual parcial ou total utilizem-se da WEB. Mas ainda há a necessidade de plataformas que possam resolver por inteiro o problema de acessibilidade do usuário, já que em muitos casos, não há apenas uma ferramenta que pode auxiliar a utilização do sistema.

Segundo Jensen1 e Øvad1 (2016), mais de um fator pode influenciar na acessibilidade por usuários com deficiência auditiva. Pela pesquisa feita em 2015, Jensen1 e Øvad1 (2016) cita que há entre os usuários com inabilidade auditiva, problemas de compreensão de textos extensos, além da produtividade interpretativa também ser afetada pela complexidade vocabular utilizada.

Nesta pesquisa, foram levantadas diversas hipóteses como o papel da imagem como facilitador de compreensão de texto, tempo gasto pelos usuários na completude de uma tarefa é maior em *websites* com poucas imagens e usuários com deficiência auditiva releem mais texto do que os demais usuários. Os resultados demonstram em linhas gerais que as imagens por so só não alteram o desempenho do usuário com inabilidades auditivas, mas que necessidades específicas sim podem a vir alterar o desempenho.

Dentre as análises feitas, em um website com textos muito extensos, usuários com deficiência auditiva tiveram um desempenho inferior ao de uma pessoa com audição normal, demorando mais de 300% do tempo para completar uma tarefa.

Para confirmar a hipótese de que um dicionário de palavras em uma linguagem de sinais poderia trazer benefícios ao usuário, a pesquisa montou um grupo de pessoas afim de avaliar esta hipótese. De acordo com classificações qualitativas (medição da dificuldade para completar tarefas), e quantitativas, a pesquisa chegou a conclusão que o usuário sim, encontra mais facilmente informações dentro do *website*, porém o tempo de execução da tarefa não modifica-se tanto, pelo trabalho a mais de se encontrar a palavra no dicionário de Sinais. Vale ressaltar, que a pesquisa levanta que este problema é meramente técnico, e refere-se a um aumento de tempo pela execução de uma tarefa a mais, incluindo-se o tempo de carregamento e fechamento da imagem. Como os tempos com e sem dicionário de sinais foi praticamente o mesmo, considerando-se a consulta como uma tarefa a mais, conclui-se que o tempo de execução da tarefa também diminui consideravelmente.

O Social4All (CRESPO; ESPADA; BURGOS, 2015) é uma aplicação que busca acessibilizar a WEB para diferentes tipos de pessoas. Seguindo o guia de desenvolvimento da WCAG, o Social4All indica os problemas de acessibilidade encontrados na página, assim como a importância de que este erro seja corrigido, e uma amostra do website de como a página fica seguindo o guia da WCAG. Além das adaptações automáticas, seguindo o Guia de Acessibilidade, o aplicativo também autoriza o usuário a inserir novos problemas encontrados na página, separando a experiência do usuário por perfis. Assim, quando o usuário utiliza-se do sistema, as alterações citadas se tornarão parte das alterações automáticas.

Através de um *gateway* WEB, a aplicação recupera qualquer website e adapta o código original, inserindo padrões definidos no Guia de acessibilidade. O aplicativo também possui um sistema de análise de performance do usuário, que mede o quão vantajosas foram as mudnaças para o usuário, e também mede o nível de acessibilidade já implementado pelo website.

2 Objetivos e Hipótese

Como há ainda na literatura poucos estudos sobre acessibilidade no meio digital para usuários com inabilidades, o objetivo central do projeto referencia-se ao estudo de formas acessíveis ao usuário deste tipo, com foco nas inabilidades visuais e auditivas. Criar e utilizar-se de ferramentas existentes e que já atuam em questões de acessibilidade no âmbito geral para facilitar o uso destes usuários no meio WEB. Ainda dentro do objetivo, temos a proposta de hipótese de que não existe alguma maneira universal de acessibilidade para um website.

3 Ferramentas de Acessibilidade para Usuários com Deficiência Visual e Auditiva

O termo acessibilidade não se refere apenas ao meio digital. Diversas ferramentas auxiliam as pessoas em suas tarefas diárias, facilitanto suas tarefas apesar de suas inabilidades. O uso destas ferramentas pode ser transplantado para o meio digital. É claro e evidente que nem todas as ferramentas podem ser utilizadas. Porém, em alguns casos, a computação pode e deve agir com estas ferramentas auxiliares, já que não há um meio comum a tecnologia que possa agir no sentido de acessibilidade por parte do usuário.

3.1 Linguagens de Sinais

3.2 Simplifcação de Texto utilizando Processamento de Linguagem Natural

4 Padronização WEB

Desde o inicio da internet, a padronização de mostragem de páginas na rede foi sofrendo alterações. Com o passar dos anos, os textos foram sendo substituidos por imagens e os layouts transformados para se tornarem mais agradáveis e acessiveis. O principal motivo para mudanças foi mudar a experiência do usuário de forma com que as tarefas pudessem ser executadas de maneira mais simples e prática.

4.1 Coesão textual e visual

4.2 Padrões HTML5

O *Hyper Text Marking Language* (HTML) é uma linguagem de programação padronizada utilizada para a construção de websites. Desde o ínicio da utilização massiva da internet, a padronização de conteúdo a ser mostrado dentro do HTML foi uma preocupação por parte dos desenvolvedores. Com as ferramentas de pesquisa, essa demanda por padronização se mostrou ainda mais necessária.

Como o padrão foi desenhado em uma época diferente, os propósitos de uma página WEB já não eram mais os mesmos, portanto foi visto a necessidade de mudar as marcações cobertas pela linguagem. Utilizando-se de conceitos previamente desenvolvidos, como *tableless* (W3C, 2005a) e *navigation*, permitindo ao programador reproduzir explicitamente as marcações de seções e navegações, respectivamente.

Mecanismos de busca visam procurar dentro dos websites por informações específicas. Preço de produtos, notícias, informações de redes sociais. A gama de funcionalidades e tipo de informações disponíveis em um website aumentou, e a padronização quanto a localidade destas informações se mostrou necessária.

Como as aplicações já fazem o uso desta padronização visando o agrupamento destas informações, uma aplicação que vise facilitar o uso da Internet por pessoas com inabilidades específicas pode se utilizar destes padrões.

Duas padronizações particularmente se mostram interessantes. A navegação do usuário, geralmente realizada por ancoras (*links*) dentro do website agora se mostra presente sempre dentro da marcação *nav*. Assim, ao mostrar as opções de navegação e redirecionamento dentro da página, a aplicação pode procurar imediatamente por

conteúdos existentes na marcação *nav*, e mostra-las ao usuário, seja em uma forma visual diferente (aumentando as letras, para usuários com dificuldades visuais), ou até mesmo formando alguma maneira de reproduzir em alguma mídia o seu conteúdo, e posteriormente redirecionar o usuário através de comandos simples, diferentes de comandos padrões de usuários (um clique, por exemplo). O Algoritimo 1 demonstra como se construia um menu, geralmente feito em um elemento de marcação *ul* (Acrônimo para *Unordered List*).

Algoritmo 1 Exemplo de menu em um modelo HTML anterior ao HTML5

```
<ul class="menu">
<li class="menuItem">Item 1</li>
<li class="menuItem">Item 2</li>
<li class="menuItem">Item 3</li>
<li class="menuItem">Item 4</li>
</ul>
```

Note que pelo conceito previamente definido, de manter cada marcação HTML exclusivamente trabalhando com questões para qual a marcação foi construida, utilizar uma lista não é a melhor forma de se manter um menu de navegação em uma página WEB. Também é importante perceber de que a classe CSS definida pode variar, de design para design. Portanto, um website pode definir sua *ul* de navegação com uma classe e outro website com uma classe de nome totalmente distinto. Isto prejudica a procura por padrões por parte das aplicações. O Algoritmo 2 mostra como a marcação *nav* permite uma padronização mais segura da navegação.

Algoritmo 2 Exemplo de menu em HTML5

```
<nav>
<div class="menuItem">Item 1</div>
<div class="menuItem">Item 2</div>
<div class="menuItem">Item 3</div>
<div class="menuItem">Item 4</div>
</nav>
```

Também pode-se perceber que não é mais necessário manter os itens do menu como subitens de uma lista. Isto implica uma redução de código CSS, já que cada item presente em uma lista tem um estilo automático gerado, com implicações em posicionamento da tela, identação, espaçamento de linhas, etc. A partir do momento em que cada item dentro da marcação de navegação é apenas uma *div*, esta estilização automática não existe por parte do browser.

A outra diz respeito à forma com que o conteúdo principal da página é mostrado. Como dito anteriormente, o conceito de *tabless* surgiu antes mesmo do HTML5. Esta prática iniciou-se pois uma diagramação simples das páginas se utilizavam basicamente de tabelas. Porém, o objetivo inicial de uma tabela não condiz com um desenho de uma página por inteiro. Assim, dentro do guia de desenvolvimento padrão de uma página WEB, era inaceitável a utilização de tabelas para a diagramação do *layout* de uma página, sendo indicável a utilização de divisões, ou dentro do *layout*.

Mas este conceito ainda era insuficiente, mesmo para necessidades básicas de informações no HTML. O próprio exemplo do menu de opções de navegação já demonstra isto. Outro exemplo da ineficiência de utilizar-se apenas divisões é a de páginas com conteúdo escrito denso, ou mesmo páginas com diferentes tipos de conteúdo.

Páginas de jornais, portais de noticias são exemplos de utulização massiva da marcação *article*. A ideia é manter o conteúdo principal escrito desta página dentro desta marcação.

Outro exemplo refere-se a páginas que possuem várias seções de conteudo diferentes. Muitas páginas procuram dar a experiência do usuário de uma forma corrida e extensa, em uma mesma página, porém, com muitas funcionalidades e conteúdos diferentes. Dai a utilizaçao da marcação *section*, que separa por seções as partes de uma página web.

Com estas divisões bem feitas, facilita-se o trabalho de mecanismos de buscas de conteúdo dentro da página WEB. Os padrões e as indicações feitas pela W3C (1994) vão muito além destas aqui citadas. Porém, pouco acrescentam nas questões específicas de detecção de conteúdo para eventuais mudanças para usuários com inabilidades.

5 Acessibilidade para Usuários com Deficiência Visual

Com os estudos realidados na área de acessibilidade para Deficientes Visuais, o grupo de pesquisa desenvolveu um protótipo simples para demonstrar alguns pontos os quais podem ser úteis para a melhoria da experiência na Internet por parte destes usuários. Algumas tecnologias já existentes no mercado (com conceitos até fora do meio digital) puderam ser aplicados para um exemplo pequeno desta aplicação. Descorreremos através desta unidade, os passos de desenvolvimento desta aplicação protótipo.

5.1 Uso do text-to-speech

O conceito de reproduzir textos em áudio não restringe-se ao meio digital. Já há alguns anos, pessoas com deficiência visual tem acesso a ferramentas do dia-a-dia que reproduzem em forma de áudio seu conteúdo, como relógios, agendas eletrônicas, etc. Existem até aplicações *desktop* que realizam este tipo de trabalho para dentro do ambiente do Sistema Operacional. Um exemplo disto é o Vision (1997) que funciona dentro do Sistema Operacional trazendo o áudio para o usuário das palavras escritas na tela.

Existem disponíveis na Internet, ferramentas WEB que reproduzem áudios sobre o conteúdo encontrado no navegador. Estas aplicações (assim como as que funcionam no *desktop*) são chamadas de *text-to-speech*, que como o próprio nome já diz, simplesmente geram áudios do texto definido.

Porém, ainda é necessário analisar qual conteúdo deve ser reproduzido. As ferramentas disponíveis na WEB possuem apenas o formato de biblioteca (ou APIs). Portanto, cabe ao aplicativo que consome estas APIs que reproduzem textos em forma de áudio, qual conteúdo reproduzir.

5.1.1 Ferramentas utilizadas

Tentando desenvolver uma aplicação exatamente para suprir esta demanda, o grupo fez o uso de uma API escrita em *JavaScript* que realiza chamadas simples à biblioteca de sons do Google disponível em seu módulo tradutor (GOOGLE, 2006).

Esta API (ResponsiveVoiceJS (1994)) se encarrega de fazer as requisições diretamente aos servidores do Google, deixando a iteração com a aplicação em si transparente e de fácil compreensão. O Algorítmo 3 apresenta um trecho simples de código que inicia já a aplicação, reproduzindo o conteúdo HTML descrito em um simples elemento HTML. O protótipo final utiliza-se de uma chamada à API idêntica à esta, com diferenciações apenas no momento, e nos elementos em que a chamada é feita.

Algoritmo 3 Exempo de chamada simples feita à API ResponsiveVoiceJS

```
function ReproduzirTexto(elemento)
    responsiveVoice.speak(elemento.innerHTML);
```

Como já citado, a API se encarrega apenas de encapsular chamadas feitas à API do Google Tradutor, e retornar sua resposta. Portanto, a cada conversão *text-to-speech* feita pela API, há uma chamada JQuery (2006a) feita ao servidor Google. A resposta é automaticamente interpretada pelo navegador como reprodução do arquivo de som recebido, e imediatamente se inicia a reprodução do áudio.

Para fins técnicos, vale ressaltar que a requisição ao browser é feita por pedaços. Portanto, dependendo do tamanho do texto, mais de uma requisição será feita ao servidor do Google, bem como mais de uma resposta será recebida. O áudio não necessariamente é recebido por inteiro pelo navegador.

5.1.2 Servidor NodeJS

O NodeJS (2009) é uma ferramenta que permite com que os desenvolvedores *JavaScript* utilizem-se da aplicação a fim de criar uma interface do lado Servidor, toda ela escrita na mesma linguagem *JavaScript*. Do lado do desenvolvedor WEB (sobretudo desenvolvedores mais focados em *front-end*), ficou mais fácil se desenvolver aplicações do lado servidor (ou aplicações *back-end*). Também, pela arquitetura assíncrona e diretamente ligada à funções antes tidas apenas do lado cliente, há também a facilidade maior em lidar com requisições HTTP e objetos de transmissão de dados tal qual o JSON (1999)

O papel principal deste servidor é fornecer a interface inicial ao navegador, e buscar o conteúdo da página WEB desejada. Explicaremos na seção 5.2 qual é exatamente o papel do servidor, e como analisamos o conteúdo da página web desejada.

Optamos por utilizar esta ferramenta pela facilidade do grupo em utiliza-la. Porém, não há nela qualquer fator exclusivo que permita a execução do projeto apenas com o uso desta. Alguns fatores como a facilidade de se trabalhar com o objeto JSON deixam o desenvolvimento mais natural. Mas caso a equipe de desenvolvimento opte por utilizar outro tipo de servidor WEB, não há nenhum impedimento.

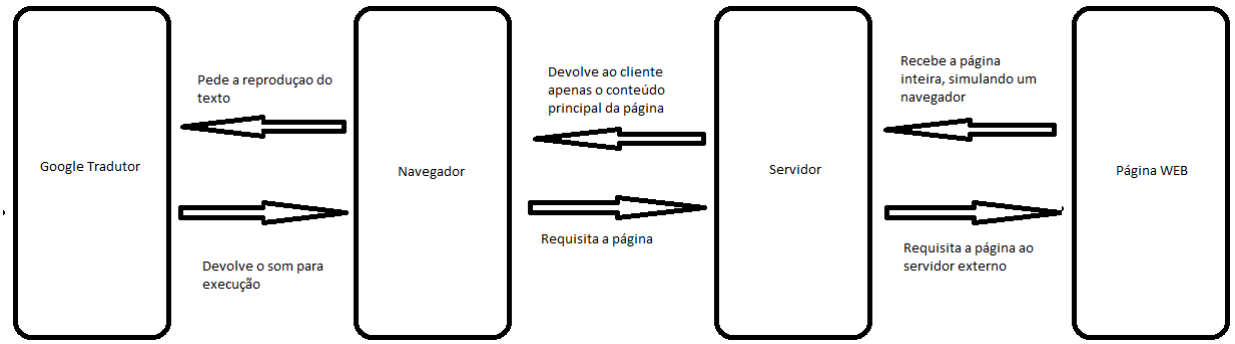
5.2 Arquitetura do protótipo

Detalharemos aqui a arquitetura geral do protótipo. Vale salientar que esta arquitetura não é de fato a melhor para a resolução do problema. Conceitos de comunicação entre cliente/servidor WEB devem ser levados em conta e melhor tradados durante a aplicação. A aplicação não se encontra *RESTfull* (REST,), que seria a melhor abordagem para o desenvolvimento. Porém, como o objetivo do protótipo era apenas provar conceitos relativos à padronização WEB e a reprodução por meio de áudio de conteúdos escritos, acreditamos que a arquitetura para se utilizar estas ferramentas não fosse importante, como desenvolvimento de prova de conceito. Mas fica claro que em um futuro desenvolvimento de um produto mais robusto, uma arquitetura WEB melhor deve ser adotada.

A Figura 1 representa um pedido básico de reprodução de texto. Vale salientar que o inicio do processo se dá no navegador. É nele em que o usuário insere o endereço da página a ser adaptada. Assim, ao confirmar o endereço, a aplicação informa ao servidor (escrito em NodeJS) que esta página é a escolhida. Portanto, o servidor se encarrega de buscar esta página na Internet, simulando uma requisição simples de um navegador. A resposta naturalmente é a própria página, codificada em HTML. Daí, o trabalho do servidor é encontrar as informações necessárias, baseadas numa escolha de padrões dentro da página, como explicaremos na seção 5.2.1.

Ao selecionar este conteúdo, o servidor envia de volta para o browser. Recebido, aparece na tela, adicionado como texto simples em uma divisória da página construida, sem qualquer estilo ou *script*. Daí, permite-se ao usuário requisitar a execução do audio do texto. Ao requisitar esta funcionalidade, o navegador faz nova requisição AJAX, mas agora para o servidor do Google Tradutor, enviando o texto a ser reproduzido e a linguagem. O mesmo devolve ao navegador o conteúdo requisitado, e automaticamente o áudio começa a ser executado.

Figura 1 – Arquitetura geral do sistema



5.2.1 Seleçao do Conteúdo

Vale aqui uma breve explicação sobre a escolha do conteúdo feita pela aplicação. Como explicado anteriormente, a padronização por parte da marcação HTML é de extrema importância para que aplicações que buscam interpretar e utilizar parte da página WEB. A aplicação protótipo por nós desenvolvida visa trazer o conteúdo principal da página para ser reproduzido ao usuário, que não possui a capacidade visual de analisar o conteúdo.

Uma página comum de notícia ou blog deve manter seu conteúdo principal dentro das marcações *section* seguidas por uma marcação *article* como mostrada no Algoritmo 4.

Algoritmo 4 Exempo de hierarquia ideal de uma página HTML5

```
<section class="menuWrapper">
<nav class="menuItem">
...
...
</nav>
</section>
...
...
<section class="articleWrapper">
<article class="articleContent">
<p>
Conteudo principal do artigo...
</p>
</article>
</section>
```

Podemos ver que a marcação *section* separa as principais seções da página. E a marcação *article* contém o conteúdo principal. Assim, o servidor da aplicação desenvolvida

busca retornar ao navegador apenas o conteúdo dentro desta marcação *article*, com ou sem as marcações adicionais de parágrafo (conhecida como a marcação *p*).

O algoritmo 5 mostra simplificadamente como ficaria uma seleção por *JavaScript* deste conteúdo, retornando-o para um uso qualquer de um consumidor desta função.

Algoritmo 5 Exempo de seleção simples de conteúdo principal de um artigo WEB

```
function SelecionarCOnteudo(elemento)
    var contentArray = document.getElementsByTagName("article").
    var textContent = ;
    for var i = 0; i ¡ contentArray.length; i++ do
        textContent += contentArray[i].innerHTML;
    return return textContent
```

Este exemplo retorna todo o conteúdo dentro da marcação *article*, o que traz também a marcação *p* em formato texto, de forma indesejada. O protótipo desenvolvido utiliza JQuery (2006b) que facilita a retirada das marcações, e consegue selecionar apenas o texto. Outro item utilizado pela aplicação foi a seleção da marcação *h*, que guarda o título do texto, e é de vital importância para a compreensão do texto em áudio.

5.2.2 Dificuldades na seleção do conteúdo

Como o conteúdo principal de uma página de um artigo jornalístico por exemplo se refere apenas o conteúdo da reportagem e suas imagens. Para um usuário com deficiências visuais, a imagem em sí não interessa, tampouco os demais itens, como propagandas, iterações de redes sociais, etc. A seleção da marcação article já seria o suficiente em um mundo ideal, porém há um problema de padronização por parte dos portais de notícias do mundo. Muitos ainda não aderiram ao formato HTML5 de marcações, e também incluem conteúdos que não interessam ao usuário, como número de comentários em redes sociais, anchoras para compartilhamento em aplicativos de mensagem, etc. A solução temporária foi analisar padrões específicos de cada portal de noticia e realizar uma pesquisa pelas marcações existentes nesses padrões analisados.

5.2.3 Melhorias na aplicação

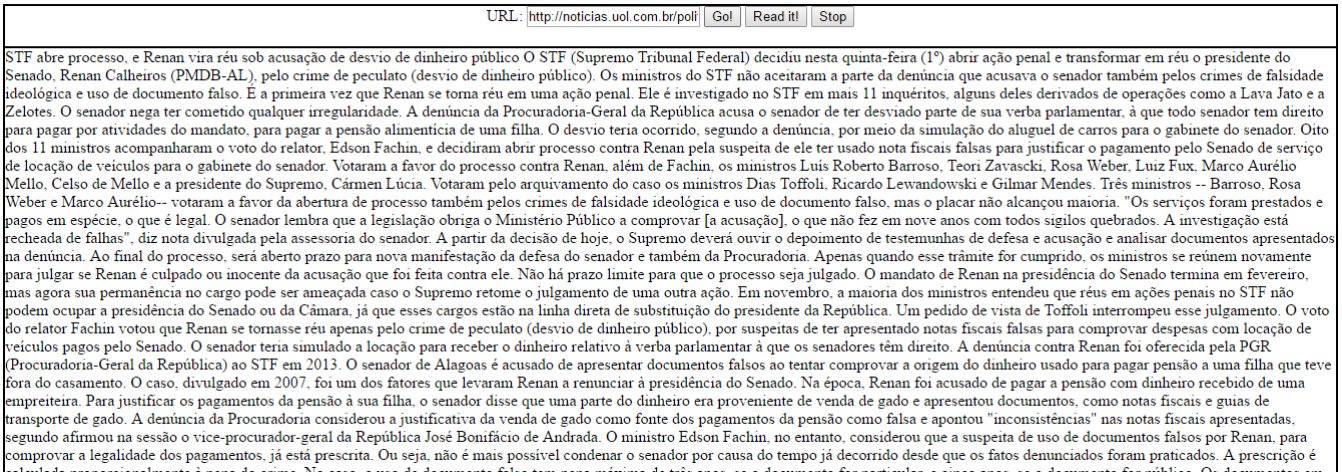
A Figura 2 demosntra a interface geral da aplicação. Página de simples diagramação, o objetivo principal é manter as funcionalidades de fácil acesso para um eventual auxílio

Algoritmo 6 Seleção dos elementos HTML de acordo com o dominio da página

```
var htmlPatterns = {
"folha.uol.com.br": "article>.content>p",
"uol.com.br": "article>header>h1, article>#texto>p, article>#texto>h3",
"estadao.com.br":"h2.titulo-principal, div.content>p",
".":"article"
};
function SelecionarConteudo {
    var contentArray = $(htmlPatterns[window.location.href]).
    var textContent = contentArray.text();
     return return textContent;
}
```

externo, ou pelo próprio comando do usuário. A aplicação permite que o usuário apenas use o teclado para disparar a ação de leitura do áudio, sem a necessidade da utilização do mouse.

Figura 2 – Layout da página protótipo



Mas há ainda muito espaço para melhorias. Como o protótipo foi criado apenas com o intuito de prova de conceito, há ainda muitos detalhes de acessibilidade que devem ser abordados. Traremos aqui alguns pontos que no desenvolvimento final do produto, podem apresentar melhorias substanciais para os usuários do sistema.

Fica claro na representação da aplicação que a utilização da aplicação é extremamente limitada ao usuário com deficiências auditivas. Fica necessário a presença de uma aplicação independente do navegador. O trabalho inicial que o usuário teria para apenas abrir o navegador seria gigantesco. Portanto, uma aplicação completa deve possuir um trabalho específico para que não haja a necessidade de um navegador desta maneira (pelo menos na maneira geral apresentada).

Duas alternativas podem existir para a resolução deste problema. A primeira é adaptar uma aplicação de navegação WEB para este propósito. A outra é na verdade nem utilizar o navegador. A reprodução do áudio pode ficar a cargo do Sistema Operacional e de suas ferramentas, desde que haja um cliente WEB suficientemente inteligente para separar o áudio recebido na requisição ao servidor do Google Tradutor da mensagem, e executar o comando no Sistema Operacional.

No protótipo, existem apenas estudos de padrões para três portais WEB. Seria necessário em um primeiro momento, dar suporte para mais páginas WEB, de uso tradicional pela sociedade. A *priori*, este comportamento deveria poder ser generalizado pela padronização WEB ideal. Porém, em um meio tão diversificado de desenvolvimento, se mostra praticamente impossível uma pesquisa por elementos genérica para todo e qualquer página WEB.

Possívelmente, uma análise sobre inserção de novos padrões pelo próprio usuário, poderia trazer uma melhor abrangência do sistema sobre as páginas WEB dispostas na Internet, sob um modelo semelhante ao usado por Crespo, Espada e Burgos (2015). A inserção destes padrões deve ser feita de forma fácil ao usuário comum, de forma que este não possui necessariamente conhecimentos técnicos sobre programação *front-end*.

Também fica claro que este modelo não permite nenhuma nevegação por parte do usuário com inabilidade visual. Este problema requer um estudo mais aprofundado sobre ferramentas já existentes que facilitam a utilização do computador pelo usuário deficiente visual. Mas um caminho a ser seguido é o de comandos pelo teclado, algo brevemente implementado no protótipo.

Os controles de inicio e término da ação de leitura são indicios de que o usuário também pode usar o teclado para navegação. Daí a importância da padronização da marcação *nav*, já que seria nela que os elementos de menu estariam dispostos. Assim, seria possivel que uma aplicação recupere os elementos ali presentes juntamente com as anchoras necessárias, e portanto ao comando do teclado, o usuário poderia escolher para onde quer ser redirecionado.

6 Mais uma outra seção primária

6.1 Uma seção secundária

6.1.1 Uma seção terciária

6.1.2 Outra seção terciária

6.1.3 Mais uma seção terciária

6.2 Outra seção secundária

6.3 Mais uma seção secundária

Texto de exemplo, texto de exemplo.

Texto de exemplo, texto de exemplo.

7 Conclusão

Referências[1]

CRESPO, R. G.; ESPADA, J. P.; BURGOS, D. Social4all: Definitions od specific adaptations in web applications to improve accessibility. *Computer Standards and Interfaces*, v. 48, p. 1–9, 2015. Citado 2 vezes nas páginas 9 e 21.

GOOGLE. 2006. Disponível em: ⟨https://cloud.google.com/translate/docs/⟩. Citado na página 15.

JENSEN1, S. S.; ØVAD1, T. Optimizing web-accessibility for deaf people and the hearing impaired utilizing a sign language dictionary embedded in a browser. *Cogn Tech Work*, v. 18, p. 717–731, 2016. Citado na página 8.

JQUERY. 2006. Disponível em: ⟨https://api.jquery.com/category/ajax/⟩. Citado na página 16.

JQUERY. 2006. Disponível em: ⟨https://jquery.com/⟩. Citado na página 19.

JSON. 1999. Disponível em: ⟨http://www.json.org/⟩. Citado na página 16.

NODEJS. 2009. Disponível em: ⟨https://nodejs.org/⟩. Citado na página 16.

RESPONSIVEVOICEJS. 1994. Disponível em: ⟨https://responsivevoice.org/⟩. Citado na página 16.

REST. Disponível em: ⟨http://www.restapitutorial.com/⟩. Citado na página 17.

VISION, V. 1997. Disponível em: ⟨http://www.virtualvision.com.br/⟩. Citado na página 15.

W3C. 1994. Disponível em: ⟨https://www.w3.org/⟩. Citado na página 14.

W3C. 2005. Disponível em: ⟨https://www.w3.org/2002/03/csslayout-howto.html.en⟩. Citado na página 12.

W3C. *Web Accessibility Initiative*. 2005. Disponível em: ⟨https://www.w3.org/WAI/intro/people-use-web/Overview.html⟩. Citado na página 8.

W3C, W. W. W. C. *Web Accessibility Initiative*. 2005. Disponível em: ⟨https://www.w3.org/WAI/intro/accessibility.php⟩. Citado na página 8.

[1] De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.