INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
CAMPUS TERESINA CENTRAL
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

LEONARDO LAMBERTO DE SANTANA SALMENTO

NOWILL - PLUGIN PARA AUXÍLIO NA CRIAÇÃO DE APLICAÇÕES VUE ACESSÍVEIS

TERESINA, PIAUÍ

2020

LEONARDO LAMBERTO DE SANTANA SALMENTO

NOWILL - PLUGIN PARA AUXÍLIO NA CRIAÇÃO DE APLICAÇÕES VUE ACESSÍVEIS

Trabalho de Conclusão de Curso (monografia) apresentado como exigência parcial para obtenção do diploma do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Teresina Central.

Orientador: Prof. Dr. Thiago Alves Elias da Silva
Coorientador: Prof. M^{e}. Fernando Castelo Branco Gonçalves Santana

TERESINA, PIAUÍ

2020

LEONARDO LAMBERTO DE SANTANA SALMENTO

NOWILL - PLUGIN PARA AUXÍLIO NA CRIAÇÃO DE APLICAÇÕES VUE ACESSÍVEIS

Trabalho de Conclusão de Curso (monografia) apresentado como exigência parcial para obtenção do diploma do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Teresina Central.

Orientador: Prof. Dr. Thiago Alves Elias da Silva

Aprovado pela banca examinadora em 09 de outubro de 2020.

BANCA EXAMINADORA:

Prof. Dr. Thiago Alves Elias da Silva
Instituto Federal de Educação, Ciência e Tecnologia do Piauí - IFPI

Prof. M^{e}. Fernando Castelo Branco Gonçalves Santana
Instituto Federal de Educação, Ciência e Tecnologia do Piauí - IFPI

Prof. Dr. Ricardo Martins Ramos
Instituto Federal de Educação, Ciência e Tecnologia do Piauí - IFPI

Prof. M^{e}. Ely da Silva Miranda
Instituto Federal de Educação, Ciência e Tecnologia do Piauí - IFPI

TERESINA, PIAUÍ

2020

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD

Salmento, Leonardo Lamberto de Santana

S171n Nowill : plugin para auxílio na criação de aplicações vue acessíveis / Leonardo Lamberto de Santana Salmento. - 2020.
45 f.: il. color.

Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) - Instituto Federal do Piauí, Campus Teresina Central, 2020.
Orientador : Prof. Dr. Thiago Alves Elias da Silva.
Coorientador : Prof. Me. Fernando Castelo Branco Gonçalves Santana.

1. Acessibilidade. 2. Acessibilidade na web. 3. Visual Studio Code (VSCode). 4. Lint. I.Título.

CDD - 004.21

Elaborado por Sindya Santos Melo CRB 3/1085

AGRADECIMENTOS

Após uma longa caminhada, estamos chegando a um fim. Não foi uma trajetória fácil e tão pouco rápida, até passei mais tempo do que devia, mas vale o que aprendemos no decorrer dela. E não teria conseguido chegar até aqui sozinho, dessa forma, gostaria de agradecer aos que me ajudaram a chegar até aqui.

Aos meus pais, que sempre me apoioram, incentivaram a estudar e por terem me possibilitado chegar aonde eu cheguei. Aos professores do IFPI, que me ensinaram a gostar do curso, a entender como ele funciona e perceber cada vez mais que estou na area certa.

Aos Professores Thiago e Fernando, que me convidaram para o Lims onde tive a oportunidade de participar de vários projetos, e ainda aceitaram ser meus orientadores para mais esse desafio na minha vida, e mais uma vitória está a caminho.

E é claro, aos meus amigos, que estiverem sempre ao meu lado, me apoiando, fazendo trabalhos, estudando para provas, nas correrias, nas horas boas e ruins. Sempre tive dificuldades em escrever textos, e se eu consegui fazer um TCC, um texto desse tamanho, com certeza a ajuda deles foi fundamental :)

Obrigado por tudo povo!

"Crianças, vocês não podem se apegar ao passado. Porque não importa o quanto você segurar...
ele já se foi."
(How I Met Your Mother)

“Nada consta”
(Autor)

RESUMO

A *web* foi criada em 1991 por Tim Berners-Lee. Conforme o crescimento da rede, Berners percebeu que ela só chegaria ao seu potencial se pudesse ser usada por qualquer um, em qualquer lugar. Além disso, várias pessoas estavam sendo prejudicadas com a falta de acessibilidade na *web*, portanto foi criado a *World Wide Web Consortium* (W3C) com a proposta de padronizar essa rede na busca de atingir todo o seu potencial. Por conta disso, foi criada a *Web Content Accessibility Guidelines* (WCAG), uma cartilha de acessibilidade buscando guiar programadores no desenvolvimento de páginas acessíveis. Mesmo após a implementação de tais diretrizes, em 2016, apenas 30% dos sites eram acessíveis no Reino Unido. No Brasil, o governo desenvolveu o Modelo de Acesso em Governo Eletrônico (eMAG), buscando padronizar os modelos de acessibilidade para as necessidades brasileiras. Alguns dos problemas vistos eram representados pelo tempo perdido para estudar a fundo as questões de acessibilidade e pela complexidade de algumas regras. Pensando nisso, este trabalho propõe desenvolver uma ferramenta para o *Visual Studio Code*, que auxilie o programador no desenvolvimento de aplicações *Vue* acessíveis baseando-se no modelo do eMAG.

Palavras-chaves: Acessibilidade, Lint, Acessibilidade na web, Visual Studio Code.

ABSTRACT

The web was created in 1991 by Tim Berners-Lee. As the network grew, Berners realized that it would only reach its potential if it could be used by anyone, anywhere. In addition, several people were being harmed by the lack of accessibility on the web, so the World Wide Web Consortium (W3C) was created with the proposal to standardize this network in the search to reach its full potential. Because of this, the Web Content Accessibility Guidelines (WCAG) was created, an accessibility guide seeking to guide programmers in the development of accessible pages. Even after the implementation of such guidelines, in 2016, only 30% of websites were accessible in the United Kingdom. In Brazil, the government developed the Electronic Government Access Model (eMAG), seeking to standardize accessibility models for Brazilian needs. Some of the problems seen were represented by the time lost to study the accessibility issues in depth and the complexity of some rules. With this in mind, this work proposes to develop a tool for Visual Studio Code, which assists the programmer in the development of accessible Vue applications based on the eMAG model.

Key-words: Accessibility, lint, web accessibility, Visual Studio Code.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

VSCode	*Visual Studio Code*
CERN	Organização Europeia para a Pesquisa Nuclear
W3C	*World Wide Web Consortium*
TTS	*Text-to-Speech*
XP	*Extreme Programming*
WCAG	*Web Content Accessibility Guidelines*
IDE	*Integrated Development Environment*
eMAG	Modelo de Acessibilidade em Governo Eletrônico
NPM	*Node Package Manager*
SPA	*Single Page Application*

SUMÁRIO

1 INTRODUÇÃO

No ano de 1989, Tim Berners-Lee tendo como ideia inicial de atender à demanda por compartilhamento automatizado de informações em universidades e institutos em todo o mundo, criou a *Web*. No final de 1990, a primeira página *Web* já funcionava na Organização Europeia para Pesquisa Nuclear (CERN), seu local de trabalho. Em 1991, pessoas de fora da CERN foram convidadas a participar da comunidade *Web* e conforme essa rede começou a crescer, Berners percebeu que ela só chegaria ao seu potencial máximo se pudesse ser usada por qualquer um, em qualquer lugar, sem precisar pagar uma taxa ou pedir permissão (FOUNDATION, 2019) e (CERN, 2019).

A facilidade de utilizar a Web permitiu que hoje as informações fossem compartilhadas em quantidade e velocidade superiores aos daquela época. Entretanto, com tantas informações em distribuição, muitas pessoas foram prejudicadas pela falta de acessibilidade em sites, uma vez que as mesmas possuíam algum tipo de deficiência ou incapacidade para percepção dessas informações. Segundo pesquisa realizada em 2018, existem cerca de 1 bilhão de pessoas com algum tipo de deficiência em todo mundo (BANK, 2018). No Brasil, esse grupo corresponde a cerca de 23,9% da população total sendo 18,8% dessa proporção portadores de deficiência visual (IBGE, 2010).

Diante disso, em outubro de 1994 foi criada a *World Wide Web Consortium* (W3C), dirigida por Berners, buscando padronizar a internet de modo a facilitar o seu uso para as pessoas com deficiências (PACIELLO, 2000) propondo levar a internet ao seu potencial máximo, desenvolvendo protocolos e diretrizes para o crescimento a longo prazo da *Web*. Dessa forma, foi criado a *Web Content Accessibility Guidelines* (WCAG), uma cartilha de acessibilidade para guiar desenvolvedores no desenvolvimento de páginas acessíveis. Em pesquisa realizada em fevereiro de 2019, constatou-se que atualmente existem cerca de 1,4 bilhões de *websites* hospedados ao redor do mundo (NETCRAFT, 2019). No Reino Unido, entretanto, mesmo após a implementação das diretrizes, segundo o IEEE em 2016, cerca de 70% dos *websites* do estado soberano não eram acessíveis, ainda que o mesmo contasse com 12 milhões de pessoas com alguma deficiência (DAVIS, 2017).

Em 2016, no Brasil, foi criado a Lei Brasileira de Inclusão (LBI), que estipula que todos os sites brasileiros devem ser acessíveis (BRASIL, 2015) e (FABRO, 2019). O governo brasileiro desenvolveu o Modelo de Acesso em Governo Eletrônico (eMAG), baseando-se no WCAG, buscando padronizar os modelos de acessibilidade digital para as necessidades brasileiras (BRASIL, 2014). Mesmo com a lei e com o modelo criado pelo governo, a (BIGDATACORP, 2019) mostrou que dos 14 milhões de sites brasileiros, menos de 1% passou nos testes de acessibilidade. E se tratando de sites governamentais, esse percentual cai para 0,34%. A grande maioria dos sites, 93,7%, apresentou falha em algum dos testes realizados, mas pontuou bem em

outros.

Pessoas com deficiência utilizam de vários tipos de ferramentas para ter acesso às informações, como softwares de reconhecimento de voz, *plugins* ou extensões para traduzir o conteúdo da página para leitores de tela, libras ou trabalhar em conjunto com um display de braille (ISOCIAL, 2017). Esses artefatos são implementados por programadores, que trabalham na maioria das vezes utilizando ferramentas que os auxiliam no processo de desenvolvimento. Um exemplo dessas ferramentas de auxílio são os *lints*, que analisam o código em busca de possíveis divergências de padronização que vão contra determinadas normas, possibilitando ao desenvolvedor focar em algo que precise de uma atenção maior.

O *Visual Studio Code* (VSCode) é uma IDE relativamente nova, criada em 2015 pela *Microsoft*. Possui suporte a uma grande variedade de linguagens, multiplataforma, gratuita, instala novas extensões falcimente, personalizável, tem integração com o git, e por essas razões vem ganhando cada vez mais usuários (EDUONIX, 2019) e (VSCODE, 2020).

Considerando a dificuldade que os deficientes sofrem ao acessar páginas *Web* não acessíveis e a facilidade que os programadores ganham ao trabalhar com ferramentas *lint*, este trabalho apresenta um plugin do tipo *lint* para o VSCode, que auxilia o desenvolvedor na criação de aplicações Vue acessíveis, uma vez que a IDE possui poucos *plugins* relacionados a acessibilidade no seu *marketplace*.

1.1 JUSTIFICATIVA

Assim como descrito por (CARRER, 2018) e (W3C, 2005), a acessibilidade na *Web* tem como objetivo tornar essa rede inteligível para todos, permitindo o entendimento, a percepção, a navegação e a interação, e não só isso, permitir que possam contribuir com a mesma. Vale lembrar também que isso não vale apenas para pessoas com deficiências, mas para pessoas com incapacidades provisórias, idosos e até mesmo pessoas com conexões lentas.

Embora a ausência de acessibilidade na *Web* atinja muitas pessoas, o fato da maioria dos usuários não serem afetados pode acabar não despertando tanto interesse dos desenvolvedores a torná-la parte padrão de seus projetos. Por muitas vezes, ao tentar inserir acessibilidade no projeto, ele pode acabar se tornando complexo pela quantidade de regras que devem ser aplicadas, e pelo fato delas serem subdivididas em outros grupos para estudá-las e aplicá-las corretamente, isso necessitaria de muito tempo e esforço por parte dos programadores (SCHEE, 2019).

Dessa forma, uma ferramenta que minimize o tempo dos desenvolvedores e os ajude a aderir às recomendações de acessibilidade do eMAG se mostra de suma importância na tentativa de tornar a acessibilidade na Web cada vez mais utilizada e mais fácil de se desenvolver.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O objetivo geral deste trabalho é desenvolver uma ferramenta para o VSCode, que auxilie o programador no desenvolvimento de aplicações em Vue acessíveis, baseando-se nas recomendações do eMAG.

1.2.2 Objetivos Específicos

- Identificar quais recomendações podem ser validadas através de um plugin;
- Utilizar dessas recomendações segundo o eMAG;
- Sugerir ao programador alterações no código ao desenvolver uma aplicação em Vue.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 ACESSIBILIDADE

Segundo (IPED, 2020), a acessibilidade é um conjunto de medidas voltadas a garantir a possibilidade de acesso a um ou mais lugares para pessoas que possuam necessidades especiais. A acessibilidade deve cuidar para que as pessoas com necessidades especiais consigam não apenas acessar lugares, mas também que eles se adaptem as condições de cada um, possibilitando, entre outras coisas, que estas pessoas consigam romper barreiras importantes para suas vidas.

Enquanto, para a legislação brasileira, a acessibilidade é descrita como sendo a possibilidade e condição de alcance para utilização, com segurança e autonomia, dos espaços, mobiliários, equipamentos urbanos, edificações, transportes, informação e comunicação, inclusive seus sistemas e tecnologias, bem como de outros serviços e instalações abertos ao público, de uso público ou privados de uso coletivo, tanto na zona urbana como na rural, por pessoa com deficiência ou com mobilidade reduzida (BRASIL, 2000).

A Figura 1 mostra um exemplo de acessibilidade com o uso de piso tátil e corrimão em uma rampa.

Figura 1 – Rampa com piso tátil

FONTE: (VIVADECORA, 2020).

Porém, no contexto deste trabalho, abordaremos o conceito de acessibilidade na *Web*.

2.1.1 WCAG

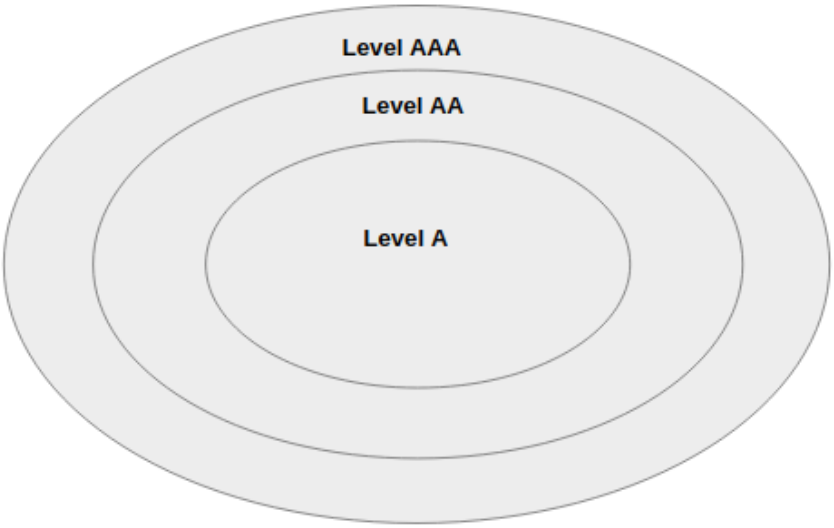
A *Web Content Accessibility Guidelines* (WCAG), foi desenvolvida com a finalidade de tornar o conteúdo da *Web* mais acessível. Isso significa que sites, ferramentas e tecnologias são

desenvolvidas para que pessoas com alguma deficiência consigam perceber, entender, navegar e interagir com a mesma (W3C, 2005). Para isso, baseando-se nas diretrizes da WCAG, a *Web* pode ser dívidida em 3 níveis de conformidade (W3C, 2019), que vão desde os mais simples até os mais complexos (WCAG, 2014), sendo eles:

- **Nível A -** Os critérios de sucesso no nível inicial são projetados para serem os mais fáceis de atender, sem muito impacto no design ou na estrutura do site. Por exemplo, no nível A, você não tem permissão para identificar algo apenas por cor, como "Pressione o botão verde para continuar".
- **Nível AA -** No nível atual, requer um pouco mais de comprometimento . Continuando com o tema da cor, você deve garantir que todo o texto atenda aos requisitos de contraste de cores. O requisito difere um pouco com base no tamanho do texto, mas na verdade é bastante rigoroso.
- **Nível AAA -** Nesse nível, o requisito é aprimorado com um requisito de contraste de cores ainda mais rigoroso para o texto. Essencialmente, você só pode usar cores muito escuras em um fundo muito claro e vice-versa. Quase todo o texto colorido falha.

Vale lembrar que para um site afirmar que é compatível com o nível AA, ele tem que atender todos os critérios do nível A, e do nível AA. Como a Figura 2 demonstra, para afirmar que um site é compatível com algum nível de acessibilidade, ele deve atender a todos os critérios do nível atual e do nível anterior (MYACCESSIBLESITE, 2020).

Figura 2 – Níveis de acessibilidade



FONTE: Elaborado pelo autor.

2.1.2 eMAG

O eMAG tem o papel de guiar no desenvolvimento e adaptação de conteúdos digitais do governo. As recomendações eMAG foram baseadas nas diretrizes de acessibilidade da WCAG, adaptando-as para as necessidades brasileiras, elas permitem que a adesão da acessibilidade digital seja realizada de forma padronizada e de fácil implementação.

Para facilitar essa implementação, as recomendações no eMAG são separadas por seções:

- 1. Marcação;
- 2. Comportamento (*Document Object Model - DOM*);
- 3. Conteúdo/Informação;
- 4. Apresentação/Design;
- 5. Multimídia;
- 6. Formulário.

Diferente da WCAG, no eMAG as recomendações de acessibilidade não são divididas por níveis de prioridade e sim por área. Por se tratar de recomendações para páginas de governo, todas as recomendações necessárias para determinada situação devem ser seguidas. Assim, se a página é a área de contato, as recomendações de formulário (além das de marcação, conteúdo, etc) devem ser seguidas, se apresentar vídeo, atenção especial deve ser dada as recomendações de multimídia (BRASIL, 2020). No anexo A é possível verificar todas as recomendações eMAG com as respectivas explicações.

2.2 ELEMENTOS DA PROGRAMAÇÃO

2.2.1 Linguagem de Programação

A linguagem de programação permite aos programadores escreverem um conjunto de ordens, ações consecutivas, dados e algoritmos, através de uma série de instruções, para criar programas que controlam o comportamento físico e lógico de uma máquina. Em outras palavras, a linguagem de programação é uma forma estruturada de comunicação, composto por conjuntos de símbolos, paravras-chave, regras semânticas e sintáticas, que fazem a comunicação entre o programador e o computador (ROCKCONTENT, 2020).

2.2.2 *Plugin*

Um *plugin* é um código que funciona dentro de um programa, geralmente dando funcionalidades extra ao *software*. Ele pode ser ativado ou desativado sem precisar alterar qualquer parte do código. Em algumas linguagens, os *plugins* são implementados como bibliotecas compartilhadas e são carregadas pelo programa principal (VANDEVOORDE, 2006) e (MÅRTENSSON, 2013).

2.2.2.1 *Lint*

Ferramentas de *lint* são *scritps* feitos para percorrer todo o código de um programa buscando erros de sintaxe, a ausência de ponto e vírgula ou, até mesmo, pontos considerados más práticas (JUSTEN, 2015) e (FERREIRA, 2012). Elas são ferramentas de análise de código estático que conseguem perceber erros antes mesmo do código ser executado (MEDEIROS, 2017). Um *lint* pode ser programado para verificar se o código está dentro de determinados estilos ou diretrizes (BELLAIRS, 2019). Alguns exemplos de tipos de lints:

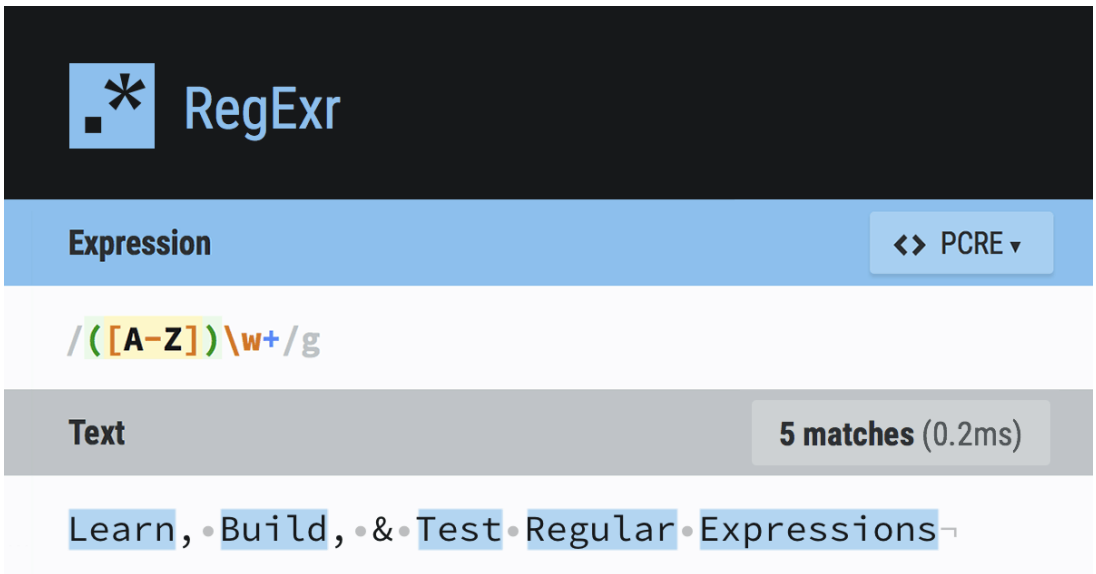
- **Tipo de Entrada** - Código fonte ou código binário/objeto;
- **Linguagens suportadas**;
- **Vulnerabilidades que são reportadas pelas ferramentas** - Problemas de estilo, nomenclatura, problemas de concorrência, problemas de segurança e etc;
- **Extensibilidade** - se a ferramenta pode ser melhorada com regras próprias;
- **Tipo de licença** - Código aberto, grátis ou comercial.

2.2.3 *Regex*

Expressões regulares ou *regex* é uma cadeia de caracteres que permite aos desenvolvedores combinar, localizar ou gerenciar um texto. Ele também é usado para encontrar texto dentro de um arquivo ou de uma string. Em JavaScript e TypeScript, as expressões regulares também são objetos (VAD, 2018) e (REGEXR, 2020).

Exemplo de regex demonstrado na Figura 3, onde um conjunto de simbolos corresponde as palavras em azul.

Figura 3 – Exemplo de regex



FONTE: (REGEXR, 2020).

2.2.4 *Vue*

Vue é um *framework* progressivo para a construção de interfaces de usuário. Isso quer dizer que ele pode ser adotado a projetos já existentes, pode ser conectado apenas em uma parte da aplicação que precisar de uma experiência mais rica e interativa, e integrar com outras bibliotecas. Além de que, Vue também é capaz de construir aplicações com o *Single-Page Applications (SPA)*, utilizando de ferramentas modernas e bibliotecas de apoio. (VUE, 2020) e (SILVA, 2018).

3 TRABALHOS RELACIONADOS

3.1 *WEB ACCESSIBILITY CHECKER*

O *Web Accessibility Checker* foi lançado em 2016 e teve sua última versão em 2018, é uma extensão para o navegador e utiliza o link da url para testar o *website* em execução, seguindo as diretrizes da WCAG, nos *levels* A e AA, além de apontar erros caso falte acessibilidade no site (KRISTENSEN, 2016). É possível adicionar, alterar ou excluir regras a serem analisadas no navegador.

O Web Acessibility Checker é uma extensão para o navegador. Assim, ele não analisa o código, mas analisa a estrutura da página depois que foi desenvolvida. E propõe uma maneira mais fácil de executar uma verificação de acessibilidade na web.

O Nowill, além de ter sido desenvolvido baseado no modelo de acessibilidade eMAG, ele também tem a intenção de auxiliar o programador no momento do desenvolvimento da aplicação, já o *Web Accessibility Checker* só faz a análise, depois do desenvolvimento completo.

3.2 *TENON HTML ACCESSIBILITY CHECKER*

Desenvolvida pela Microsoft em 2015, a Tenon HTML possuía duas vertentes, sendo elas uma extensão para o VSCode e uma extensão para o navegador. A versão para IDE foi descontinuada, enquanto a versão para o navegador trabalha com as diretrizes da WCAG nos *levels* A, AA e AAA (DEVLABS, 2016).

Em sua versão para o navegador, é uma ferramenta de teste automatizado de acessibilidade, verificando se o site está cumprindo com as normativas e diretrizes de acessibilidade da WCAG.

Mais uma vez, o Nowill se difere na finalidade da ferramenta, uma vez que, ela irá auxiliar o programador no momento do desenvolvimento, além de se basear nas recomendações do eMAG.

3.3 *WEB ACCESSIBILITY*

O *Web Accessibility* teve a sua primeira versão no final de 2018 e ainda se encontra em fase de desenvolvimento. Foram lançadas cerca de 12 versões, a última delas em março de 2019. Esta ferramenta destaca os elementos que o desenvolvedor deve alterar e dá dicas de como pode ser modificado, pensando nos critérios de acessibilidade (SCHEE, 2018). Essa ferramenta analisa códigos de páginas HTML, tem opção de corrigir a semântica da linguagem e possui também a possibilidade de limitar a quantidade máxima de erros a serem encontradas em uma

única página.

No quesito semelhança, tanto o presente trabalho quanto a ferramenta tratada neste tópico possuem o mesmo propósito de auxliar o programador no momento do desenvolvimento. No entanto, o *Web Accessibility* não explicita se foi baseado em algum conjunto de regras ou diretrizes, apesar de solucionar vários problemas trazidos pela WCAG. Já o Nowill, busca resolver os problemas brasileiros com base nas normas do eMAG.

4 NOWILL - PLUGIN PARA AUXÍLIO NA CRIAÇÃO DE APLICAÇÕES VUE ACESSÍVEIS

O Nowill é um plugin de análise de código estático para a IDE VSCode, que foi desenvolvido em *TypeScript* utilizando-se da própria IDE, e tem como finalidade auxiliar o programador no desenvolvimento de aplicações Vue acessíveis, validando a existência ou ausência de determinadas *tags* ou conjunto de regras, seguindo as recomendações do eMAG. A ferramenta faz a análise dessas *tags* e quando necessário, apresenta alguma informação ou sugere que o desenvolvedor faça alterações em seu código.

O nome da ferramenta tem a intenção de homenagear Dorina de Gouvêa Nowill. Ela foi a primeira aluna cega a frequentar um curso regular da Escola Normal Caetano de Campos, além de fazer uma especialização para educação de pessoas com deficiência visual nos Estados Unidos. Dorina se preocupava com a falta de livros em braille, diante disso, ela criou a fundação para o Livro do Cego no Brasil (NOWILL, 2019).

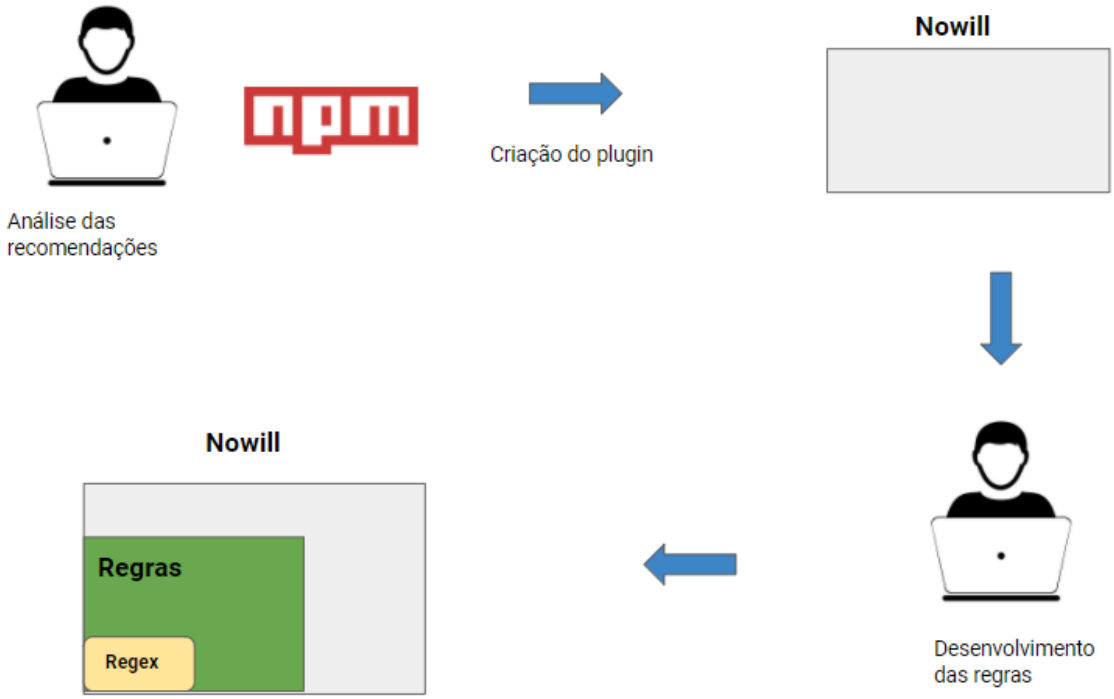
4.1 ELABORAÇÃO DO PROJETO

A elaboração do projeto ocorreu por meio das seguintes etapas:

- **Análise das recomendações:** Na fase inicial, foi feita uma análise a partir das recomendações, selecionando quais eram possíveis de se resolver com um analisador de código estático. E em quais *tags* elas se aplicam, ou seja, foram selecionadas as tags que podem influenciar na acessibilidade das páginas web;
- **Criação do plugin:** A criação do plugin é feita a partir do Gerenciador de Pacotes do Node (npm), criando alguns módulos automaticamente, porém, o plugin vem sem regras internas;
- **Desenvolvimento das regras:** Para desenvolver tais regras, foi utilizado o regex na identificação de padrões de escrita e aplicados no TypeScript. Enquanto o TypeScript é utilizado para criação das regras, o regex tem o papel de identificar padrões de escrita e verificar se o código necessita de mudanças para sugerí-las, caso necessário.

  Como demonstra a Figura 4, o projeto partiu das análises da recomendações, utlizando do NPM foi criado o plugin Nowill, onde foram desenvolvidas e aplicadas as regras para validar o uso das *tags*.

Figura 4 – Elaboração do Projeto



FONTE: Elaborado pelo autor.

4.2 IMPLEMENTAÇÃO

O VSCode utiliza o Yeoman para a criação de seus plugins, portanto foi necessário a instalação do mesmo através do npm para o desenvolvimento do Nowill. O Yeoman é utilizado para inicializar projetos de vários tipos e linguagens, criando módulos e configurações, poupando tempo aos desenvolvedores (YEOMAN, 2019).

Após a criação do plugin, foi utilizado regex para criar padrões de código para encontrar as *tags* HTML e assim identificar quais recomendações de acessibilidade do eMAG essa *tag* deve usar. Será feita a análise estática do código, e se for necessário, o *plugin* irá sugerir alterações que poderão ser realizadas pelo desenvolvedor.

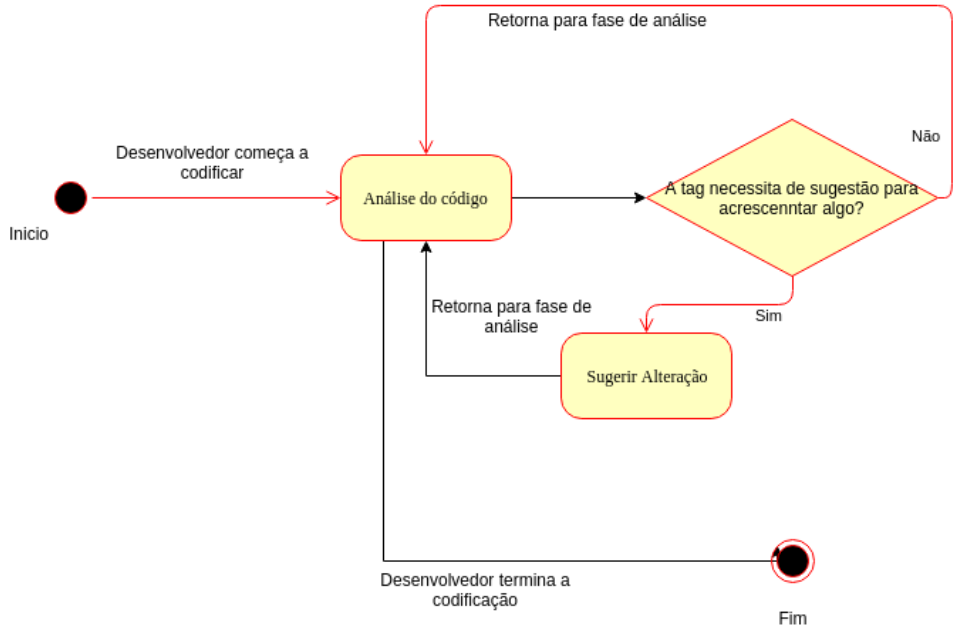
4.3 ARQUITETURA DO PROJETO

A arquitetura definida para o sistema se dá através da interação do programador com a IDE no momento do desenvolvimento, onde o plugin analisa o código estático para verificar se o mesmo está de acordo com suas regras. Essas regras serão definidas com base nas recomendações do eMAG explicitadas no tópico 4.6.

4.3.1 Diagrama de Estado

Assim que o desenvolvedor inicia a codificação, o plugin fica em estado de análise do código. Cada *tag* é analisada e é verificado a necessidade de ser feita alguma modificação. Caso seja necessário, o plugin sugere a alteração e depois volta a parte de análise do código. O estado mudaria para encerrado quando o desenvolvedor finaliza a codificação e fecha a IDE, assim como é demonstrado na Figura 5.

Figura 5 – Diagrama de Estado



FONTE: Elaborado pelo autor.

4.4 *TYPESCRIPT*

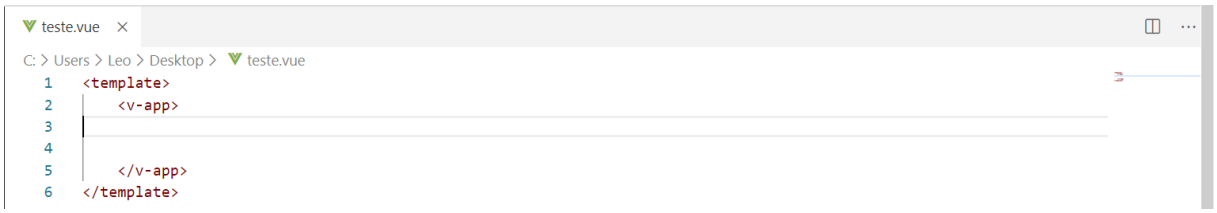
O plugin foi desenvolvido em TypeScript, uma linguagem desenvolvida pela microsoft. Ele é um superset tipado que é compilado para o JavaScript simples. Ele também já é utilizado por grandes *frameworks* como Angular e Ionic, desenvolvedores das duas empresas comentam que a linguagem deixa o código mais compreensível e fácil de trabalhar, do que o JavaScript puro (TYPESCRIPT, 2019).

4.5 FUNCIONAMENTO DA FERRAMENTA

Para melhor entendimento de como a ferramenta funciona, as Figuras 6, 7 e 8 mostram um exemplo prático do *plugin* desenvolvido.

A Figura 6 mostra uma página com apenas um código base em Vue onde o programador irá iniciar o seu projeto.

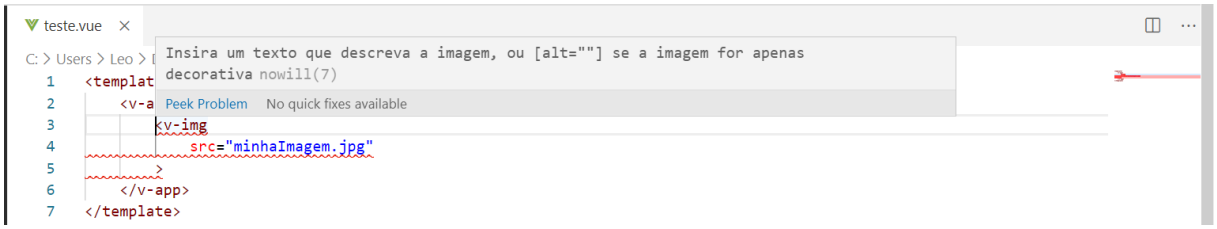
Figura 6 – Código base



FONTE: Elaborado pelo autor.

A Figura 7 mostra que o usuário criou a *tag* <v-img> para inserir uma imagem, porém ele não colocou o atributo alt, e recebe a mensagem de erro do plugin Nowill sinalizando toda a extensão da tag com um sublinhado vermelho.

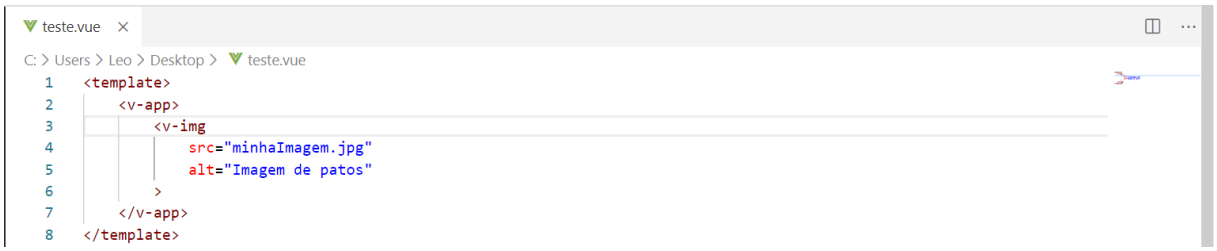
Figura 7 – Mensagem de erro



FONTE: Elaborado pelo autor.

Na Figura 8, após ele incluir o atributo alt, a mensagem de erro desaparece indicando que o código está livre dos erros de acessibilidades analisados pela ferramenta.

Figura 8 – Código corrigido



FONTE: Elaborado pelo autor.

4.6 RECOMENDAÇÕES DO EMAG

Nesta seção, serão apresentadas as recomendações nas quais a ferramenta se fundamentou e uma explicação das quais ela não está analisando.

4.6.1 Recomendação analisadas

O *plugin* foi desenvolvido para aplicações Vue, porém ressalta-se que devido as restrições do *framework*, algumas verificações não podiam ser realizadas em aquivos com extensão '.Vue'. Estas verificações foram aplicadas em arquivos HTML, com a intenção de fazer a verificação no arquivo 'index.html', arquivo que serve de base para receber as modificações feitas no Vue e gerar o arquivo HTML final.

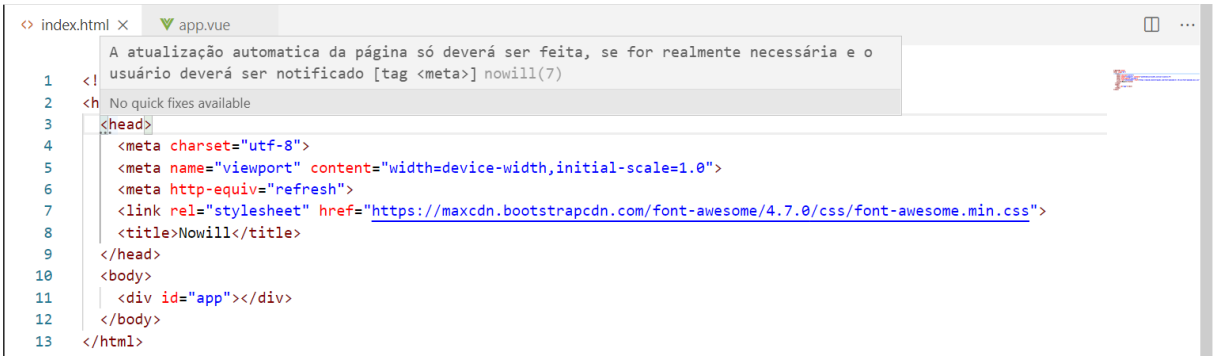
Abaixo segue a lista das recomendações que o plugin está analisando.

4.6.1.1 Recomendações analisadas no HTML

- **2 - Comportamento**;
  - **2.3 - Não criar páginas com atualização automática periódica:** As páginas só devem ser atualizadas automaticamente se forem realmente necessárias, e o usuário deverá ser informado dessa atualização automática.

    A Figura 9 mostra que o desenvolvedor adicionou uma atualização automática na página, e o plugin sinaliza com a mensagem de informação ao usuário.

Figura 9 – Recomendação 2.3



FONTE: Elaborado pelo autor.

- **3 - Conteúdo/Informação**;
  - **3.1 - Identificar o idioma principal da página:** Deve-se identificar o idioma principal do documento. A identificação é feita através do atributo *lang*, da *tag* HTML.

    Na Figura 10 é demonstrado que o desenvolvedor não adicionou o idioma principal da página. O plugin sinaliza o erro e sugere a correção.

Figura 10 – Recomendação 3.1

FONTE: Elaborado pelo autor.

- **3.3 - Oferecer um título descritivo a página:** O título da página deve ser descritivo e informativo, devendo representar o conteúdo principal da página.

  A Figura 11 mostra que o desenvolvedor não colocou a *tag* <title> dentro do head. O plugin sinaliza o erro e sugere a correção.

Figura 11 – Recomendação 3.3

FONTE: Elaborado pelo autor.

- **4 - Apresentação/Design**;
  - **4.1 - Oferecer contraste mínimo entre plano de fundo e primeiro plano:** As cores do plano de fundo e do primeiro plano deverão ser suficientemente contrastantes para que possam ser visualizadas. Não deverão ser utilizadas imagens atrás do texto (background), pois acabam por dificultar a leitura e desviar a atenção do usuário.

    A Figura 12 mostra que, ao inserir um background na *tag* <body> o plugin informa as condições para o background ser aceito.

Figura 12 – Recomendação 4.1

FONTE: Elaborado pelo autor.

4.6.1.2 Recomendações analisadas no Vue

- **3 - Conteúdo/Informação**;
  - **3.5 - Descrição clara e sucinta de links:** Deve-se identificar claramente o destino de cada link, informando, inclusive, se o link remete a outro sítio. Além disso, é preciso que o texto do link faça sentido mesmo quando isolado do contexto da página.

    A Figura 13 mostra que o desenvolvedor criou um link com um texto que não não sinaliza para onde o levará. O plugin sinaliza e descreve o erro.

Figura 13 – Recomendação 3.5

FONTE: Elaborado pelo autor.

- **3.6 - Fornecer alternativa em texto para as imagens do sítio:** Deve ser fornecida uma descrição para as imagens da página.

  A Figura 14 mostra que o desenvolvedor adicionou uma imagem e não adicionou uma descrição. O plugin sinaliza o erro e informando como corrigí-lo.

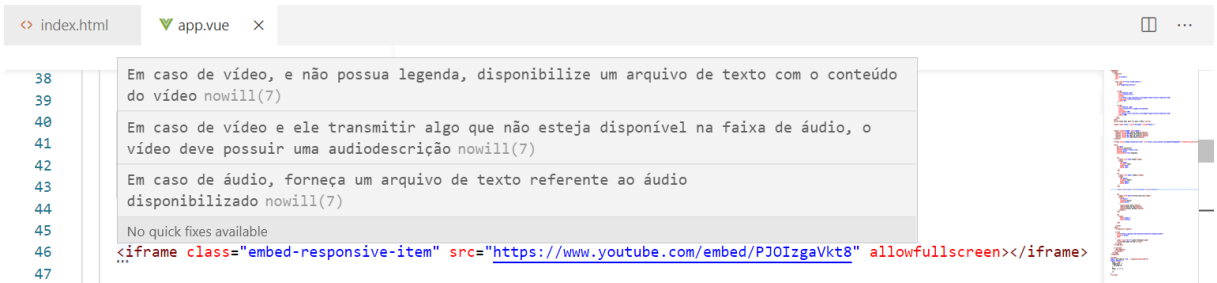
Figura 14 – Recomendação 3.6

FONTE: Elaborado pelo autor.

- **5 - Multimídia**;
  - **5.1 - Alternativa para vídeo:** Deve haver uma alternativa sonora ou textual para vídeos que não incluem faixas de áudio. Para vídeos que contêm áudio falado e no idioma natural da página, devem ser fornecidas legendas. Além de essencial para pessoas com deficiência visual, a alternativa em texto também é importante para usuários que não possuem equipamento de som, que desejam apenas realizar a leitura do material ou não dispõem de tempo para ouvir um arquivo multimídia.
  - **5.2 - Alternativa para aúdio:** Áudio gravado deve possuir uma transcrição descritiva. Além de essencial para pessoas com deficiência auditiva, a alternativa em texto também é importante para usuários que não possuem equipamento de som, que desejam apenas realizar a leitura do material ou não dispõem de tempo para ouvir um arquivo multimídia. Neste caso, também é desejável a alternativa em Libras.
  - **5.3 - Oferecer audiodescrição para vídeo pré-gravado:** Vídeos que transmitem conteúdo visual que não está disponível na faixa de áudio devem possuir uma audiodescrição.

    A Figura 15 mostra que ao criar uma *tag* <frame> para adicionar áudio ou vídeo, o plugin mostra as condições para cada tipo de uso.

Figura 15 – Recomendações 5.1-5.3



FONTE: Elaborado pelo autor.

- **6 - Formulários**;
  - **6.1 - Fornecer alternativa em texto para os botões de imagem de formulários:** Ao serem utilizados botões do tipo imagem (input type=”image”), que servem para o mesmo propósito do botão do tipo submit, deve ser fornecida uma descrição textual para o botão através do atributo alt. Já para outros tipos de botões (reset e button), é preciso substituir o botão pela imagem que se deseja utilizar através do CSS. Nesse caso, para que o botão seja acessível, ele deve possuir um value descritivo.

    A Figura 16 mostra que o usuário criou um botão de reset através da *tag* <input> e não colocou o atributo value. O plugin sinaliza o erro com a possível solução.

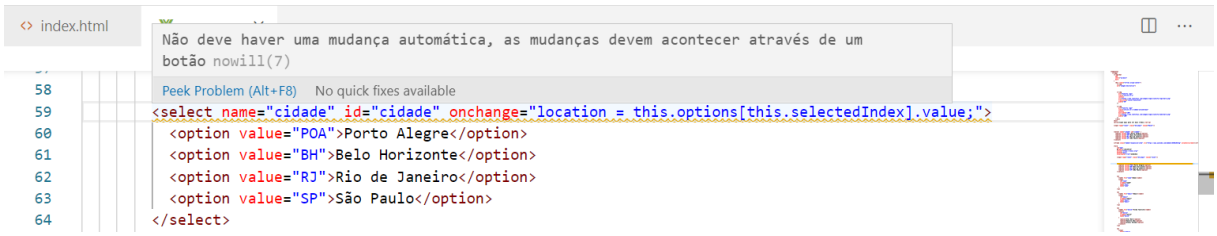
Figura 16 – Recomendação 6.1

FONTE: Elaborado pelo autor.

  - **6.4 - Não provocar automaticamente alteração no contexto:** Quando um elemento de formulário receber o foco, não deve ser iniciada uma mudança automática na página que confunda ou desoriente o usuário. Assim, as mudanças devem ocorrer através do acionamento de um botão.

    A Figura 17 mostra que o usuário criou um select utilizando do atributo onchange. O plugin sinaliza o erro e aponta uma possível solução.

Figura 17 – Recomendação 6.4



FONTE: Elaborado pelo autor.

4.6.2 Recomendação não analisadas

Serão apresentados os motivos pelo qual o plugin não está atingindo todas as recomendações do eMAG, detalhando por seções.

- **1 - Marcação:** Seção voltada para a organização do código. O plugin não conseguiu fazer nenhuma validação nesta seção, pois as recomendações presentes nelas são voltadas para como o programador organiza o seu código durante o desenvolvimento, como as recomendações (1.3 - Utilizar corretamente os níveis de cabeçalho, 1.4 – Ordenar de forma lógica e intuitiva a leitura e tabulação e 1.6 - Não utilizar tabelas para diagramação).
- **2 - Comportamento:** Esta seção é focada em como a página e o seu conteúdo se comportam em relação ao usuário, de modo que a acessibilidade não seja prejudicada. De modo que, seria preciso fazer uma análise se as funções da página podem ser acessadas via teclado, se scripts, conteúdos e outros elementos são sempre acessíveis, a existência de efeitos visuais piscantes e a existência de controle para as alterações temporais(slides). No momento a ferramenta está analisando apenas uma recomendação desta seção, a 2.3 - Não criar páginas com atualização automática periódica.
- **3 - Conteúdo / Informação:** Mesmo esta seção sendo a que contempla a maior quantidade de recomendações analisadas pelo plugin(quatro), ele não pode fazer a análise de todas as recomendações da mesma. O plugin não conseguiu analisar casos como o da mudança de idioma no conteúdo, informação ao usuário da sua localização na página, ou as que são relacionadas ao modo de escrita, como as recomendações (3.9 - Em tabelas, utilizar títulos e resumos de forma apropriada, 3.10 - Associar células de dados às células de cabeçalho, 3.11 - Garantir a leitura e compreensão das informações e 3.12 - Disponibilizar uma explicação para siglas, abreviaturas e palavras incomuns).
- **4 - Apresentação / Design:** Seção voltada para o visual. A única recomendação analisada pelo plugin dessa seção, foi utilizada apenas como um lembrete para o desenvolvedor do que deve ser feito. Por ser algo mais visual não foi possível verificar as demais recomendações.
- **5 - Multimídia:** Seção voltada para o uso de vídeos e aúdios na página. Nesta seção não foram possíveis fazer a análise de duas recomendações (5.4 - Fornecer controle de áudio para som e 5.5 - Fornecer controle de animação), pois não foi possível verificar todos os tipos de sons da página teriam um mecanismo para controlá-lo e se possuíam animações que iniciavam automaticamente.
- **6 - Formulários:** Seção voltada para a análise dos formulários. O plugin não conseguiu verificar todas as recomendações desta seção, pois possui algumas direcionadas para o modo de desenvolvimento do programador, a sua organização e oferecer formas alternativas ao captcha, como nas sequintes recomendações (6.2 – Associar etiquetas aos seus campos, 6.3 – Estabelecer uma ordem lógica de navegação, 6.7 –

Agrupar campos de formulário e 6.8 – Fornecer estratégias de segurança específicas ao invés de CAPTCHA).

4.7 DISTRIBUIÇÃO

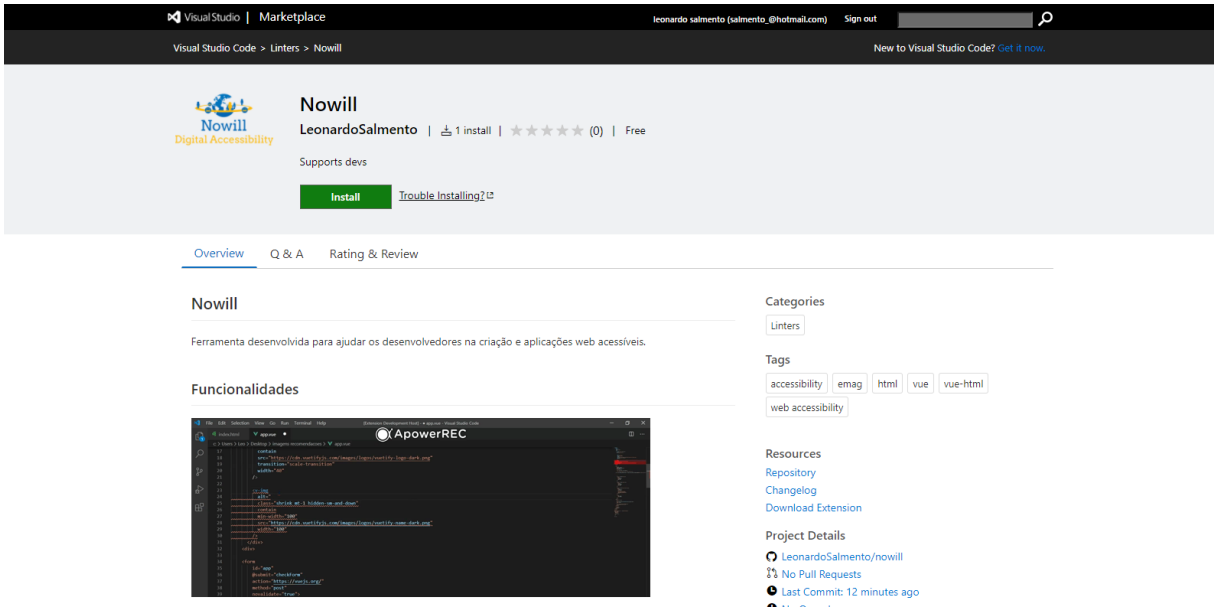
4.7.1 *Node Package Manager (NPM)*

O NPM é um repositório *online*, onde é possível fazer publicações de projetos de código aberto de Node.js. O VSCode utiliza do NPM para publicações de novos *plugins* desenvolvidos por qualquer desenvolvedor ou empresa no seu *marketplace*. Isso permite que essas ferramentas fiquem disponíveis para download para toda a comunidade. É necessário criar uma conta e gerar um token no site da IDE, para realizar o compartilhamento do plugin (NODEBR, 2016).

4.8 PUBLICAÇÃO

Assim como pode ser visto na Figura 18, o Plugin já foi publicado no marketplace do Visual Studio Code, e está disponível para baixar de forma gratuita na plataforma e na própria IDE.

Figura 18 – Publicação no marketplace do Visual Studio Code



FONTE: Elaborado pelo autor.

5 CONCLUSÃO

Este trabalho apresentou um plugin para a IDE VSCode intitulada "Nowill - Plugin para Desenvolvimento de Aplicações Vue Acessíveis". Desenvolvido em *TypeScript* e seguindo as normas do modelo eMAG, ela propõe auxiliar o programador no desenvolvimento de aplicações acessíveis em Vue.

Com base nos objetivos específicos definidos, foram encontradas recomendações das seções de Comportamento, Conteúdo, Apresentação, Multimídia e Formulários do eMAG que podem ser validadas através do plugin.

Com isso, a construção e objetivos do Nowill foram moldados em um plugin que sugere ao programador alterações no código durante o desenvolvimento de aplicações Vue tornando-as acessíveis. Espera-se que essa ferramenta contribua para que no futuro a acessibilidade na web torne-se uma prática comum e não um diferencial.

5.1 TRABALHOS FUTUROS

Para o futuro, espera-se aumentar o número de recomendações analisadas pelo plugin, além de expandí-lo, adicionando outros frameworks, assim deixando de ser apenas para o framework vue, mas para o desenvolvimento web de modo geral.

E com o código aberto disponível no github, espera-se que o sistema possa vir a receber atualizações de todos que possuem interesse em contribuir.

REFERÊNCIAS

BANK, T. W. *DISABILITY INCLUSION*. 2018. Disponível em: https://www.worldbank.org/en/topic/disability. Acesso em: 27 march. 2019.

BELLAIRS, R. *Why Is Linting Important? And How to Use Lint Tools*. 2019. Disponível em: https://www.perforce.com/blog/qac/why-linting-important-and-how-use-lint-tools. Acesso em: 14 april. 2019.

BIGDATACORP. *Acessibilidade dos sites brasileiros*. 2019. Disponível em: https://www.bigdatacorp.com.br/blog/index.php/2019/11/01/acessibilidade-dos-sites-brasileiros/.

BRASIL. *L10098*. 2000. Disponível em: http://www.planalto.gov.br/ccivil_03/LEIS/L10098.htm.

BRASIL. *eMAG - Modelo de Acessibilidade em Governo Eletrônico*. 2014. Disponível em: http://emag.governoeletronico.gov.br/.

BRASIL. *Lei Brasileira de Inclusão*. 2015. Disponível em: http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2015/lei/l13146.htm.

BRASIL. *Departamento de Governo Eletrônico*. 2020. Disponível em: https://www.gov.br/governodigital/pt-br/acessibilidade-digital/eMAGv31.pdf.

CARRER, A. *Acessibilidade na web. Por que isso é importante?* 2018. Disponível em: https://www.linkedin.com/pulse/acessibilidade-na-web-por-que-isso-é-importante-alexandre-carrer/. Acesso em: 10 june 2019.

CERN. *A short history of the Web*. 2019. Disponível em: https://home.cern/science/computing/birth-web/short-history-web. Acesso em: 19 April 2019.

DAVIS, A. *Most Developers Fail to Design Websites Accessible to People With Disabilities*. 2017. Disponível em: http://web.archive.org/web/20190130162212/http://theinstitute.ieee.org/ieee-roundup/blogs/blog/most-developers-fail-to-design-websites-accessible-to-people-with-disabilities.

DEVLABS, M. *Tenon HTML Accessibility Checker*. 2016. Disponível em: https://marketplace.visualstudio.com/items?itemName=jeffpetty.TenonHTMLAccessibilityChecker. Acesso em: 14 April 2019.

EDUONIX. *Visual Studio Code Is So Popular But Why?* 2019. Disponível em: https://code.visualstudio.com/docs/editor/whyvscode.

FABRO, N. *Menos de 1% dos sites brasileiros são acessíveis para pessoas com deficiência*. 2019. Disponível em: https://revistagalileu.globo.com/Tecnologia/noticia/2019/10/menos-de-1-dos-sites-brasileiros-sao-acessiveis-para-pessoas-com-deficiencia.html.

FERREIRA, D. *Assegurando a qualidade do seu código JavaScript*. 2012. Disponível em: https://tableless.com.br/qualidade-codigo-javascript/.

FOUNDATION, W. *History of the Web*. 2019. Disponível em: https://webfoundation.org/about/vision/history-of-the-web/. Acesso em: 20 April 2019.

IBGE. Características gerais da população, religião e pessoas com deficiência. Censo Demográfio 2010, 2010. Disponível em: https://biblioteca.ibge.gov.br/visualizacao/periodicos/94/cd_2010_religiao_deficiencia.pdf.

IPED. *O que é acessiblidade?* 2020. Disponível em: https://www.iped.com.br/materias/cotidiano/acessibilidade.html.

ISOCIAL. *Tecnologias Assistivas Para Pessoas Com Deficiência: O Que Há Disponível No Mercado?* 2017. Disponível em: https://blog.isocial.com.br/tecnologias-assistivas-para-pessoas-com-deficiencia-o-que-ha-disponivel-no-mercado/. Acesso em: 27 march. 2019.

JUSTEN, W. *Analisando seu código JS com um linter*. 2015. Disponível em: https://willianjusten.com.br/analisando-seu-codigo-js-com-linter/.

KRISTENSEN, M. *Web Accessibility Checker*. 2016. Disponível em: https://marketplace.visualstudio.com/items?itemName=MadsKristensen.WebAccessibilityChecker. Acesso em: 27 march. 2019.

MÅRTENSSON, E. *Design and development of a plugin-based architecture on an embedded system*. 2013.

MEDEIROS, J. E. R. d. *Estudo Comparativo de Ferramentas de Análise Estática de Código*. Dissertação (B.S. thesis) — Universidade Federal do Rio Grande do Norte, 2017.

MYACCESSIBLESITE. *WCAG Levels*. 2020. Disponível em: https://myaccessible.website/blog/wcaglevels/wcag-levels-a-aa-aaa-difference.

NETCRAFT. *February 2019 Web Server Survey*. 2019. Disponível em: https://news.netcraft.com/archives/2019/02/28/february-2019-web-server-survey.html. Acesso em: 26 march. 2019.

NODEBR. *O que é a NPM do Node.JS*. 2016. Disponível em: http://nodebr.com/o-que-e-a-npm-do-nodejs/.

NOWILL, F. D. *Dorina de Gouvêa Nowill*. 2019. Disponível em: https://www.fundacaodorina.org.br/a-fundacao/dorina-de-gouvea-nowill/.

PACIELLO, M. *Web accessibility for people with disabilities*. [S.l.]: CRC Press, 2000.

REGEXR. *RegExr: Learn, Build Test RegEx*. 2020. Disponível em: https://regexr.com/.

ROCKCONTENT. *O que é linguagem de programação?* 2020. Disponível em: https://rockcontent.com/blog/linguagem-de-programacao/.

SCHEE, M. V. der. *Why the adoption of Web Accessibility keeps failing*. 2019. Disponível em: https://medium.com/@maxvanderschee/why-the-adoption-of-web-accessibility-keeps-failing-e78fe4c94149. Acesso em: 10 April 2019.

SCHEE, V. der. *Changelog*. 2018. Disponível em: https://marketplace.visualstudio.com/items?itemName=MaxvanderSchee.web-accessibility. Acesso em: 27 march. 2019.

SILVA, W. G. da. *Primeiros passos com o framework Vue.Js*. 2018. Disponível em: https://blog.cedrotech.com/primeiros-passos-com-o-framework-vue-js/#:~:text=Vue%20%C3%A9%20um%20framework%20acess%C3%ADvel,experi%C3%AAncia%20mais%20rica%20e%20interativa.

TYPESCRIPT. *TYpeScript - JavaScript tha scale*. 2019. Disponível em: https://www.typescriptlang.org/.

VAD, O. M. T. *How to create a chatbot using Tensorflow, Python and Recurrent Neural Networks*. Dissertação (B.S. thesis) — NTNU, 2018.

VANDEVOORDE, D. *Plugins in C++*. [S.l.], 2006.

VIVADECORA. *Aprenda a fazer cálculo de rampa de um jeito simples!* 2020. Disponível em: https://www.vivadecora.com.br/pro/arquitetura/calculo-de-rampa/.

VSCODE. *Why did we build Visual Studio Code?* 2020. Disponível em: https://code.visualstudio.com/docs/editor/whyvscode.

VUE. *Introdução*. 2020. Disponível em: https://br.vuejs.org/v2/guide/index.html.

W3C, W. W. W. C. *Introduction to Web Accessibility*. 2005. Disponível em: https://www.w3.org/WAI/fundamentals/accessibility-intro/. Acesso em: 14 April 2019.

W3C, W. W. W. C. *Understanding Conformance*. 2019. Disponível em: https://www.w3.org/TR/UNDERSTANDING-WCAG20/conformance.html#uc-levels-head. Acesso em: 21 April 2019.

WCAG, W. C. A. G. *Diretrizes de Acessibilidade para Conteúdo Web (WCAG) 2.0*. 2014. Disponível em: https://www.w3.org/Translations/WCAG20-pt-br/#intro. Acesso em: 20 march. 2019.

YEOMAN. What's yeoman? 2019. Disponível em: http://yeoman.io.

ANEXO A – EMAG - MODELO DE ACESSIBILIDADE EM GOVERNO ELETRÔNICO

A.1 1. MARCAÇÃO

A.1.1 Recomendação 1.1 – Respeitar os Padrões Web

Os Padrões Web são recomendações do W3C (World Wide Web Consortium), as quais são destinadas a orientar os desenvolvedores para o uso de boas práticas que tornam a web acessível para todos, permitindo assim que os desenvolvedores criem experiências ricas, alimentadas por um vasto armazenamento de dados, os quais estão disponíveis para qualquer dispositivo e compatíveis com atuais e futuros agentes de usuário (ex: navegadores).

Outro ponto importante no respeito aos Padrões Web é a separação de camadas. As camadas lógicas deverão ser separadas, de acordo com o objetivo para o qual elas foram desenvolvidas. Assim, para a camada de conteúdo devem ser utilizadas as linguagens de marcação, como HTML e xHTML. Para a camada de apresentação visual do conteúdo, utilizam-se as folhas de estilo css em qualquer uma de suas versões. Já para a camada que modifica o comportamento dos elementos, são utilizadas linguagens javascript e modelos de objeto (dom).

A.1.2 Recomendação 1.2 – Organizar o código HTML de forma lógica e semântica

O código HTML deve ser organizado de forma lógica e semântica, ou seja, apresentando os elementos em uma ordem compreensível e correspondendo ao conteúdo desejado. Cada elemento HTML deve ser utilizado para o fim que ele foi criado.

Assim, marcação semântica adequada deve ser utilizada para designar os cabeçalhos (h1, h2, h3), as listas (ul, ol, dl), texto enfatizado (strong), marcação de código (code), marcação de abreviaturas (abbr), marcação de citações longas (blockquote), etc. Dessa forma, as páginas poderão ser apresentadas e compreendidas sem recursos de estilização, tal como as folhas de estilo. Além disso, o código semanticamente correto é muito importante para usuários com deficiência visual, pois os leitores de tela descrevem primeiro o tipo de elemento e depois realizam a leitura do conteúdo que está dentro desse elemento.

A.1.3 Recomendação 1.3 – Utilizar corretamente os níveis de cabeçalho

Os níveis de cabeçalho (elementos HTML H1 a H6) devem ser utilizados de forma hierárquica, pois organizam a ordem de importância e subordinação dos conteúdos, facilitando a leitura e compreensão. Além disso, muitos leitores de tela utilizam a hierarquia de cabeçalhos como uma forma de navegação na página, pulando de um para outro, agilizando, assim, a navegação. Conceitualmente, existem seis níveis de títulos, sendo o H1 o mais alto, ou seja,

deverá corresponder ao conteúdo principal da página, assim é recomendável que toda página tenha apenas um H1. Já os níveis do H2 ao H6 poderão ser utilizados mais de uma vez na página, mas sem excesso e com lógica textual, obedecendo uma hierarquia. Para compreender melhor os níveis de título pode-se tomar como exemplo um sítio de um livro, onde o nome do livro é o H1, os capítulos são H2, os subcapítulos são H3 e assim por diante.

A.1.4 Recomendação 1.4 – Ordenar de forma lógica e intuitiva a leitura e tabulação

Deve-se criar o código HTML com uma sequência lógica de leitura para percorrer links, controles de formulários e objetos. Essa sequência é determinada pela ordem que se encontra no código HTML.

A.1.5 Recomendação 1.5 – Fornecer âncoras para ir direto a um bloco de conteúdo

Devem ser fornecidas âncoras, disponíveis na barra de acessibilidade, que apontem para links relevantes presentes na mesma página. Assim, é possível ir ao bloco de conteúdo desejado. Os links devem ser colocados em lugares estratégicos da página, como no início e fim do conteúdo e início de fim do menu. É importante ressaltar que o primeiro link da página deve ser o de ir para o conteúdo.

A.1.6 Recomendação 1.6 – Não utilizar tabelas para diagramação

As tabelas devem ser utilizadas apenas para dados tabulares e não para efeitos de disposição dos elementos na página. Para este fim, utilize as folhas de estilo.

A.1.7 Recomendação 1.7 – Separar links adjacentes

Links adjacentes devem ser separados por mais do que simples espaços, para que não fiquem confusos, em especial para usuários que utilizam leitor de tela. Para isso, é recomendado o uso de listas, onde cada elemento dentro da lista é um link. As listas podem ser estilizadas visualmente com CSS para que os itens sejam mostrados da maneira desejada, como um ao lado do outro.

A.1.8 Recomendação 1.8 – Dividir as áreas de informação

Áreas de informação devem ser divididas em grupos fáceis de gerenciar. As divisões mais comuns são “topo”, “conteúdo”, “menu” e “rodapé”. Nas páginas internas deve-se manter uma mesma divisão para que o usuário se familiarize mais rapidamente com a estrutura do sítio. É importante destacar, entretanto, que a página inicial pode ter uma divisão diferente das páginas internas, pois normalmente ela contém mais elementos.

A.1.9 Recomendação 1.9 – Não abrir novas instâncias sem a solicitação do usuário

A decisão de utilizar-se de novas instâncias – por exemplo abas ou janelas - para acesso a páginas e serviços ou qualquer informação deve ser de escolha do usuário. Assim, não devem ser utilizados:

- Pop-ups;
- A abertura de novas abas ou janelas;
- O uso do atributo target="_blank";
- Mudanças no controle do foco do teclado;
- Entre outros elementos, que não tenham sido solicitadas pelo usuário.

A.2 COMPORTAMENTO

A.2.1 Recomendação 2.1 - Disponibilizar todas as funções da página via teclado

Todas as funções da página desenvolvidas utilizando-se linguagens de script (javascript) devem ser programadas, primeiramente, para o uso com teclado. O foco não deverá estar bloqueado ou fixado em um elemento da página, para que o usuário possa mover-se pelo teclado por todos os elementos.

A.2.2 Recomendação 2.2 – Garantir que os objetos programáveis sejam acessíveis

Deve-se garantir que scripts e conteúdos dinâmicos e outros elementos programáveis sejam acessíveis e que seja possível sua execução via navegação. Além de proporcionar o uso por teclado, estratégias devem ser adotadas para proporcionar o acesso a todos independente de seu dispositivo.

A.2.3 Recomendação 2.3- Não criar páginas com atualização automática periódica

A atualização automática periódica – muito utilizada por canais de notícias - é comumente realizada através do uso do o atributo http-equiv com conteúdo "refresh" da elemento meta no HEAD do documento.

A.2.4 Recomendação 2.4 – Não utilizar redirecionamento automático de páginas

Não devem ser utilizadas marcações para redirecionar a uma nova página, como o uso do atributo http-equiv com conteúdo "refresh" do elemento META. Ao invés disso, deve-se configurar o servidor para que o redirecionamento seja transparente para o usuário.

A.2.5 Recomendação 2.5 – Fornecer alternativa para modificar limite de tempo

Em uma página onde há limite de tempo para realizar uma tarefa deve haver a opção de desligar, ajustar ou prolongar esse limite. Essa recomendação não se aplica a eventos em que o limite de tempo é absolutamente necessário.

Deve-se lembrar que, em ambos os casos, o limite de tempo deve ser informado.

A.2.6 Recomendação 2.6 – Não incluir situações com intermitência de tela

Não devem ser utilizados efeitos visuais piscantes, intermitentes ou cintilantes. Em pessoas com epilepsia fotosensitiva, o cintilar ou piscar pode desencadear um ataque epilético. A exigência dessa diretriz aplica-se também para propaganda de terceiros inserida na página.

A.2.7 Recomendação 2.7 – Assegurar o controle do usuário sobre as alterações temporais do conteúdo

Conteúdos como slideshows, que "se movem", rolagens, movimentações em geral ou animações não devem ser disparadas automaticamente sem o controle do usuário, mesmo em propagandas na página. Ao usuário deve ser repassado o controle sobre essas movimentações (quer seja por escolha de preferência de visualização da página, quer por outro método qualquer acessível a usuário com deficiência). Além disso, o usuário deve ser capaz de parar e reiniciar conteúdos que se movem, sem exceção.

A.3 CONTEÚDO / INFORMAÇÃO

A.3.1 Recomendação 3.1 – Identificar o idioma principal da página

Deve-se identificar o principal idioma utilizado nos documentos. A identificação é feita por meio do atributo lang do HTML e, para documentos XHTML, é utilizado o xml:lang. Ele deve ser declarado em todas as páginas, pois além de auxiliar na acessibilidade do conteúdo, também permite melhor indexação pelos motores de busca

A.3.2 Recomendação 3.2 – Informar mudança de idioma no conteúdo

Se algum elemento de uma página possuir conteúdo em um idioma diferente do principal, este deverá estar identificado pelo atributo lang. Essa recomendação não se aplica para nomes próprios ou termos técnicos que sejam compreendidos no contexto.

A.3.3 Recomendação 3.3 – Oferecer um título descritivo e informativo à página

O título da página deve ser descritivo e informativo, devendo representar o conteúdo principal da página, já que essa informação será a primeira lida pelo leitor de tela, quando o usuário acessar a página. O título é informado pelo elemento TITLE e deve preferencialmente

seguir a estrutura recomendada pelo ePWG, que é [assunto principal da página] – [nome do sítio ou sistema] sem palavras extras, ou recursos estilísticos.

A.3.4 Recomendação 3.4 – Informar o usuário sobre sua localização na página

Deverá ser fornecido um mecanismo que permita ao usuário orientar-se dentro de um conjunto de páginas, permitindo que ele saiba onde está no momento. Assim, poderá ser utilizado o recurso de "migalha de pão" (breadcrumbs), que são links navegáveis em forma de lista hierárquica os quais permitem que o usuário saiba qual o caminho percorrido até chegar à página em que se encontra no momento.

A.3.5 Recomendação 3.5 – Descrever links clara e sucintamente

Deve-se identificar claramente o destino de cada link, informando, inclusive, se o link remete a outro sítio. Além disso, é preciso que o texto do link faça sentido mesmo quando isolado do contexto da página.

A.3.6 Recomendação 3.6 – Fornecer alternativa em texto para as imagens do sítio

Deve ser fornecida uma descrição para as imagens da página, utilizando-se, para tanto o atributo alt.

A.3.7 Recomendação 3.7 – Utilizar mapas de imagem de forma acessível

Um mapa de imagens é uma imagem dividida em áreas selecionáveis definidas por elemento AREA . Cada área é um link para outra página Web ou outra seção da página atual. É um recurso em desuso, mas pode ser útil na acessibilidade de infográficos, por exemplo.

A.3.8 Recomendação 3.8 – Disponibilizar documentos em formatos acessíveis

Os documentos devem ser disponibilizados preferencialmente em HTML. Também podem ser utilizados arquivos para download no formato ODF, tomando-se os cuidados para que sejam acessíveis. Se um arquivo for disponibilizado em PDF, deverá ser fornecida uma alternativa em HTML ou ODF.

A.3.9 Recomendação 3.9 – Em tabelas, utilizar títulos e resumos de forma apropriada

O título da tabela deve ser definido pelo elemento CAPTION e deve ser o primeiro elemento utilizado após a declaração do elemento TABLE. Em casos de tabelas extensas, deve ser fornecido um resumo de seus dados através do atributo summary que deve ser declarado no elemento TABLE.

A.3.10 Recomendação 3.10 – Associar células de dados às células de cabeçalho

Em tabelas de dados simples, o uso apropriado do elemento TH para os cabeçalhos e do elemento TD para as células de dados é essencial para torná-las acessíveis. Para incrementar a acessibilidade, deve-se utilizar os elementos THEAD, TBODY e TFOOT, para agrupar as linhas de cabeçalho, do corpo da tabela e do final, respectivamente, com exceção de quando a tabela possuir apenas o corpo, sem ter seções de cabeçalho e rodapé.

A.3.11 Recomendação 3.11 – Garantir a leitura e compreensão das informações

O texto de um sítio deve ser de fácil leitura e compreensão, não exigindo do usuário um nível de instrução mais avançado do que o ensino fundamental completo. Quando o texto exigir uma capacidade de leitura mais avançada, devem ser disponibilizadas informações suplementares que expliquem ou ilustrem o conteúdo principal. Outra alternativa é uma versão simplificada do conteúdo em texto.

A.3.12 Recomendação 3.12 – Disponibilizar uma explicação para siglas, abreviaturas e palavras incomuns

Recomenda-se que na primeira ocorrência de siglas, abreviaturas ou palavras incomuns (ambíguas, desconhecidas ou utilizadas de forma muito específica), deve ser disponibilizada sua explicação ou forma completa.

A.4 APRESENTAÇÃO / DESIGN

A.4.1 Recomendação 4.1 - Oferecer contraste mínimo entre plano de fundo e primeiro plano

As cores do plano de fundo e do primeiro plano deverão ser suficientemente contrastantes para que possam ser visualizadas, também, por pessoas com baixa visão, com cromodeficiências ou que utilizam monitores de vídeo monocromático.

A.4.2 Recomendação 4.2 – Não utilizar apenas cor ou outras características sensoriais para diferenciar elementos

A cor ou outras características sensoriais, como forma, tamanho, localização visual, orientação ou som não devem ser utilizadas como o único meio para transmitir informações, indicar uma ação, pedir uma resposta ao usuário ou distinguir um elemento visual.

A.4.3 Recomendação 4.3 – Permitir redimensionamento sem perda de funcionalidade

A página deve continuar legível e funcional mesmo quando redimensionada para até 200%. Assim, é preciso garantir que, quando a página for redimensionada, não ocorram

sobreposições nem o aparecimento de uma barra horizontal.

A.4.4 Recomendação 4.4 – Possibilitar que o elemento com foco seja visualmente evidente

A área que recebe o foco pelo teclado deve ser claramente marcada, devendo a área de seleção ser passível de ser clicada.

A.5 MULTIMÍDIA

A.5.1 Recomendação 5.1 – Fornecer alternativa para vídeo

Deve haver uma alternativa sonora ou textual para vídeos que não incluem faixas de áudio. Para vídeos que contêm áudio falado e no idioma natural da página, devem ser fornecidas legendas. Além de essencial para pessoas com deficiência visual, a alternativa em texto também é importante para usuários que não possuem equipamento de som, que desejam apenas realizar a leitura do material ou não dispõem de tempo para ouvir um arquivo multimídia.

A.5.2 Recomendação 5.2 – Fornecer alternativa para áudio

Áudio gravado deve possuir uma transcrição descritiva. Além de essencial para pessoas com deficiência auditiva, a alternativa em texto também é importante para usuários que não possuem equipamento de som, que desejam apenas realizar a leitura do material ou não dispõem de tempo para ouvir um arquivo multimídia. Neste caso, também é desejável a alternativa em Libras.

A.5.3 Recomendação 5.3 – Oferecer audiodescrição para vídeo prégravado

Vídeos que transmitem conteúdo visual que não está disponível na faixa de áudio devem possuir uma audiodescrição.

A.5.4 Recomendação 5.4 – Fornecer controle de áudio para som

Deve ser fornecido um mecanismo para parar, pausar, silenciar ou ajustar o volume de qualquer som que se reproduza na página.

A.5.5 Recomendação 5.5 – Fornecer controle de animação

Para qualquer animação que inicie automaticamente na página devem ser fornecidos mecanismos para que o usuário possa pausar, parar ou ocultar tal animação.

A.6 FORMULÁRIOS

A.6.1 Recomendação 6.1 – Fornecer alternativa em texto para os botões de imagem de formulários

Ao serem utilizados botões do tipo imagem (input type=”image”), que servem para o mesmo propósito do botão do tipo submit, deve ser fornecida uma descrição textual para o botão através do atributo alt, conforme o exemplo a seguir.

A.6.2 Recomendação 6.2 – Associar etiquetas aos seus campos

As etiquetas de texto (elemento LABEL) devem estar associadas aos seus campos (elementos INPUT, SELECT e TEXTAREA, à exceção do elemento BUTTON) correspondentes no formulário, através dos atributos for do label e id do input, os quais deverão ter o mesmo valor.

A.6.3 Recomendação 6.3 – Estabelecer uma ordem lógica de navegação

Os elementos do formulário devem ser distribuídos corretamente através do código HTML, criando, assim, uma sequência lógica de navegação. Assim, os formulários devem primeiro ser codificados considerando a ordem lógica de navegação para depois serem organizados visualmente via CSS.

A.6.4 Recomendação 6.4 – Não provocar automaticamente alteração no contexto

Quando um elemento de formulário receber o foco, não deve ser iniciada uma mudança automática na página que confunda ou desoriente o usuário. Assim, as mudanças devem ocorrer através do acionamento de um botão.

A.6.5 Recomendação 6.5 – Fornecer instruções para entrada de dados

Para conteúdo que exigir entrada de dados por parte do usuário, devem ser fornecidas quando necessário, instruções de preenchimento juntamente com as etiquetas (elemento LABEL).

A.6.6 Recomendação 6.6 – Identificar e descrever erros de entrada de dados e confirmar o envio das informações

Quando um erro de entrada de dados for automaticamente detectado, o item que apresenta erro deve ser identificado e descrito ao usuário por texto.

A.6.7 Recomendação 6.7 – Agrupar campos de formulário

É recomendado que os campos com informações relacionadas sejam agrupadas utilizando o elemento FIELDSET, principalmente em formulários longos. O agrupamento deverá ser feito

de maneira lógica, associando o elemento LEGEND explicando claramente o propósito ou natureza dos agrupamentos.

A.6.8 Recomendação 6.8 – Fornecer estratégias de segurança específicas ao invés de CAPTCHA

CAPTCHAs são utilizados para impedir que softwares automatizados, conhecidos como bots, executem ações que degradem a qualidade do serviço de um sistema, provocando danos em áreas e e-serviços de sítios em um curto espaço de tempo, podendo sobrecarregar servidores e deixar sítios indisponíveis por um dado período.

Recomenda-se uma combinação de diferentes estratégias para serviços mais seguros e acessíveis para substituir o uso de CAPTCHA, como por exemplo:

- Limites de conexão;
- Monitoramento;;
- Consistência nas políticas de segurança;
- Uso de técnicas de desenvolvimento de serviços e formulários seguros.