UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
SISTEMAS DE INFORMAÇÃO

TESTES AUTOMATIZADOS DE ACESSIBILIDADE PARA PÁGINAS *WEB*

ALAN SCHVEITZER

FLORIANÓPOLIS
2022

ALAN SCHVEITZER

TESTES AUTOMATIZADOS DE ACESSIBILIDADE PARA PÁGINAS *WEB*

Trabalho de Conclusão de Curso apresentado ao curso de Sistemas de Informação da Universidade Federal de Santa Catarina, como requisito parcial para a Obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Raul Sidnei Wazlawick

FLORIANÓPOLIS

2022

ALAN SCHVEITZER

TESTES AUTOMATIZADOS DE ACESSIBILIDADE PARA PÁGINAS *WEB*

Trabalho de Conclusão de Curso apresentado ao curso de Sistemas de Informação da Universidade Federal de Santa Catarina, como requisito parcial para a Obtenção do grau de Bacharel em Sistemas de Informação.

Florianópolis, SC

Banca Examinadora:

_____
Prof. Dr. Raul Sidnei Wazlawick
Orientador

_____
Profa. Dra. Patrícia Vilain
Avaliadora

_____
Prof. Dr. Jean Carlo Rossa Hauck
Avaliador

AGRADECIMENTOS

Agradeço primeiramente a minha esposa Danyella Junkes, pois é ela quem me deu o apoio necessário não só para ingressar na Universidade, como para me manter firme e me ajudar a terminá-la, sem ela nada disso seria possível. A minha família, quem sempre esteve ao meu lado apoiando as minhas decisões e dando o suporte necessário para não desistir.

Aos mestres, que durante todo o curso me trouxeram grandes aprendizados e sempre estavam disponíveis para me auxiliar nas dúvidas e desafios encontrados. Deixo um agradecimento especial ao professor Raul Sidnei Wazlawick que aceitou orientar o meu trabalho. Agradeço também aos professores da banca Jean Carlo Rossa Hauck e Patrícia Vilain pelas contribuições para a realização deste trabalho.

A todos os meus colegas que conheci no curso, que durante esse período me deram apoio e incentivo a não desistir, em especial agradeço ao Alexandre Augusto Anhaia, Fernando Silva Poerner e Camila dos Reis.

Deixo um agradecimento e uma singela homenagem ao professor Leandro José Komosinski, ele foi o primeiro a aceitar o meu convite para orientar este trabalho, foi com a ajuda dele e seus valiosos conselhos que consegui iniciá-lo e desenvolvê-lo, também lhe agradeço por toda a sua dedicação para com os alunos, sempre presente nas aulas e auxiliando a todos que precisavam, muito obrigado e descanse em paz.

RESUMO

Com um número crescente de acessos à Web e à diversificação das pessoas que a acessam, um tema que vem ganhando relevância é a acessibilidade, que tem por objetivo que pessoas com qualquer tipo e grau de deficiência possam interagir, entender e navegar na Web. Garantir que uma página da Web seja acessível não é nada simples, porém é essencial. Dada a relevância e importância que o tema possui, em decorrência do surgimento da *Internet*, também surgiram organizações, como a World Wide Web Consortium (W3C) que busca, por meio da sua iniciativa intitulada Web Accessibility Initiative, definir normas de acessibilidade para que desenvolvedores e produtores de conteúdo criem páginas Web acessíveis, sendo o seu principal guia o Web Content Accessibility Guidelines (WCAG). Nesse sentido, o presente trabalho tem como objetivo desenvolver um projeto para automação dos testes de acessibilidade em páginas Web, baseado nas normas definidas no WCAG 2.0 ou versões superiores. Para corroborar com a fundamentação dos principais conceitos, foi efetuada uma pesquisa bibliográfica acerca do tema de acessibilidade Web, as normas existentes e os tipos de testes de acessibilidade. Além disso, foi realizada uma pesquisa sobre ferramentas e trabalhos existentes relacionados ao projeto de automação de testes de acessibilidade para páginas Web, tendo como objetivo identificar o atual estado de tais ferramentas e analisar como o projeto proposto se posiciona em relação a elas. Por fim, foi desenvolvido um projeto de automação de testes de acessibilidade para páginas Web que busca validar se as normas estabelecidas pelo WCAG 2.0 ou versões superiores estão sendo cumpridas, criando uma ferramenta que irá auxiliar desenvolvedores e produtores de conteúdo a garantirem que suas páginas Web sejam mais acessíveis.

Palavras-chave: Acessibilidade *Web*. Testes de acessibilidade. Normas. WCAG.

ABSTRACT

With an increasing number of people accessing the Web and the diversification of people who access it, a topic that has been gaining relevance is accessibility, which aims to enable people with any type and degree of disability to interact, understand, and navigate the Web. Ensuring that a Web page is accessible is very important. Given the relevance and importance of the theme, along with the emergence of the *Internet*, organizations also emerged, such as the World Wide Web Consortium (W3C), which seeks, through its Web Accessibility Initiative, to define accessibility standards for developers and content producers to create accessible Web pages, and its main guide is the Web Content Accessibility Guidelines (WCAG). In this sense, this work aims to develop a project for automating accessibility tests on Web pages, based on the standards defined in WCAG 2.0 or higher versions. To corroborate the foundation of the main concepts, a bibliographic research was carried out about the theme of Web accessibility, the existing standards and the types of accessibility tests. Furthermore, a survey was conducted about existing tools and works related to the project of automating accessibility tests for Web pages, with the aim of identifying the current state of such tools and analyzing how the proposed project positions itself in relation to them. Finally, an accessibility test automation project for Web pages was developed that seeks to validate whether the standards established by WCAG 2.0 or higher versions are being met, creating a tool that will help developers and content producers to ensure that their Web pages are more accessible.

Keywords: Web accessibility. Accessibility testing. Standards. WCAG.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

IBGE	Instituto Brasileiro de Geografia e Estatística
W3C	World Wide Web Consortium
WAI	*Web* Accessibility Initiative
WCAG	*Web* Content Accessibility Guidelines
HTML	HyperText Markup Language
MIT	Massachusetts Institute of Technology
ERCIM	European Research Consortium for Informatics and Mathematics
XML	Xtensible Markup Language
CDC	Centers for Disease Control and Prevention
DARPA	Defense Advanced Research Projects Agency
CERN	European Organization for Nuclear Research
API	Application Programming Interface
APPLET	Small computer application that performs one particular task
CI/CD	Continuous Integration / Continuous Delivery.
IDE	Integrated Development Environment

SUMÁRIO

1 INTRODUÇÃO

Com a globalização, a tecnologia tem-se tornado cada vez mais presente na vida das pessoas. Os acessos à *Internet* e sistemas *Web* vêm crescendo no Brasil, conforme aponta o IBGE (2019), a *Internet* nos domicílios urbanos era utilizada em 83,8% das residências no ano de 2018. Já no ano de 2019, esse número subiu para 86,7%.

Observa-se, dessa forma, o crescimento constante nos últimos anos no acesso à rede mundial de computadores. É cada vez mais comum vincularmos as tarefas do cotidiano, sejam elas profissionais ou de lazer, ao uso de tecnologias e dispositivos conectados à *Internet*, como aponta a pesquisa sobre o acesso à *Internet* pelos brasileiros onde são listados os itens mais acessados pela população, sendo eles serviços, portais, entretenimento, busca/navegação, social media, noticiários e outros (COMSCORE, 2015).

Um recurso que é cada vez mais usado é a *Web*, ela tem se tornado importante e presente em vários aspectos da vida como: saúde, governo, comércio, educação, recreação e muito mais **(**HENRY, 2019). Muitos usuários com deficiência consideram que a *Web* seja uma das principais fontes de informação, emprego e entretenimento (HARPER; YESILADA, 2008), ou seja, páginas *Web* que contam com acessibilidade ganham uma grande relevância e abrangem cada vez mais um grupo maior de pessoas.

No último Censo, realizado em 2010, 46 milhões de brasileiros, cerca de 24% da população do país, declarou ter algum grau de dificuldade em pelo menos uma das habilidades investigadas (enxergar, ouvir, caminhar ou subir degraus), ou possuir deficiência intelectual (IBGE, 2010), portanto é possível concluir que uma parte significativa da população brasileira possui algum tipo de deficiência.

Segundo Henry (2019) “acessibilidade na *Web* significa que sites, ferramentas e tecnologias são projetados e desenvolvidos para que pessoas com deficiência possam usá-los”. Sendo assim, um sistema *Web* precisa ser desenvolvido buscando a acessibilidade como um requisito e não apenas como um recurso adicional para que as pessoas com deficiência possam acessá-las.

Com o passar do tempo, páginas da *Web* se tornaram cada vez mais sofisticadas e complexas, com novos e avançados recursos, como, por exemplo, arrastar e soltar (*drag and drop*), o que causou um aumento significativo no desafio de prover ao usuário com deficiência uma experiência acessível (C*WEB*, 2020). Neste contexto, surgem organizações internacionais

com intuito de superar esses desafios, como, por exemplo, a maior organização internacional de padrões para *Web*, a World Wide Web Consortium (W3C), que através da sua iniciativa intitulada *Web* Accessibility Initiative (WAI), procura desenvolver um conjunto de diretrizes para auxiliar o desenvolvimento de páginas *Web* acessíveis, sendo o seu principal guia o *Web* Content Accessibility Guidelines (WCAG).

Mesmo com a importância que a acessibilidade em sistemas *Web* possui e ainda que existam esforços por vários entes para regulamentar a acessibilidade *Web*, na prática, a atenção dada para este tema é precária, sendo essencial que entre desenvolvedores e pesquisadores o tema seja mais presente (LOUREIRO, 2014). Portanto, neste contexto, a presente monografia visa efetuar um estudo sobre normas de acessibilidade para páginas *Web*, uma análise do estado da arte e também o desenvolvimento de um projeto para automação de testes de acessibilidade em sistemas *Web*, utilizando frameworks e tecnologias atuais como o NodeJS, JavaScript, Cypeess e a biblioteca axe-core, a qual é uma parte fundamental da solução desenvolvida, pois ela representa o principal mecanismo utilizado pela ferramenta para efetuar a detecção automatizada de violações de acessibilidade. Visando assim a criação de uma ferramenta automatizada para indicar se os sistemas *Web* alvos estão de acordo com as diretrizes do WCAG na versão 2.0 ou superior e, por fim, visa a geração de um relatório apontando pontos de falha ou melhoria em relação à acessibilidade nas páginas testadas.

1.1 OBJETIVOS

Esta seção trata dos objetivos e metas a serem alcançados através da pesquisa e execução do trabalho de conclusão de curso.

1.1.1 Objetivo Geral

A presente monografia tem como objetivo apresentar pesquisa sobre acessibilidade em páginas *Web* e suas normas. Também visa o desenvolvimento de um projeto para automação de testes de acessibilidade para páginas *Web*, com a finalidade de analisar se essas páginas estão consoantes às normas estabelecidas por entidades internacionalmente reconhecidas com o intuito de auxiliar o desenvolvimento de páginas *Web* acessíveis.

1.1.2 Objetivos Específicos

Este trabalho busca atingir quatro objetivos específicos, sendo eles:

i) Apresentar normas de acessibilidade para páginas *Web* existentes;
ii) Identificar ferramentas e metodologias para automação de testes de acessibilidade para páginas *Web*;
iii) Elaborar um projeto com ferramentas de uso livre para automação de testes de acessibilidade em páginas *Web* e executá-lo;
iv) Compor um relatório com o resultado de execução dos testes e apontamentos para melhoria nas páginas testadas.

1.2 METODOLOGIA

Na primeira fase deste trabalho, será efetuada uma pesquisa exploratória que Wazlawick (2020, p. 18) define como: “aquela em que o autor não tem necessariamente uma hipótese ou objetivo definido em mente. Ela pode ser considerada, muitas vezes, o primeiro estágio de um processo de pesquisa mais longo”. Para efetuar a pesquisa exploratória será utilizada a técnica de pesquisa bibliográfica que, segundo o mesmo autor (2020, p. 19), “implica o estudo de artigos, teses, livros e outras publicações usualmente disponibilizadas por editoras e indexadas”. Também será utilizada a técnica de pesquisa documental que “consiste na análise de documentos ou dados que não foram ainda sistematizados e publicados. Podem-se examinar relatórios de empresas, arquivos obtidos em órgãos públicos, bancos de dados, correspondências [...]” (WAZLAWICK**,** 2020, p. 18).

Na segunda fase, será efetuada uma pesquisa e análise sobre ferramentas que se assemelham a ferramenta proposta neste trabalho, esta pesquisa envolverá as seguintes etapas:

- Pesquisar por ferramentas semelhantes à ferramenta proposta no projeto deste trabalho, descrevendo-as e elencando as suas principais características.
- Colher e analisar as características dessas ferramentas, como licença de uso, tipo de ferramenta, suporte a linguagens de programação, versões suportadas do WCAG e outras características relevantes.
- Comparar as ferramentas pesquisadas em relação ao projeto proposto elencando a viabilidade do projeto e o seu diferencial.

Na terceira fase deste trabalho, será desenvolvido um projeto que, conforme Espinha (2020), consiste em um ato especial que possui um início e um fim determinados, ou seja, é temporário, e possui um objetivo a ser atingido considerando os recursos destinados a ele, sejam recursos financeiros, humanos ou materiais. Buscando conceber a automação dos testes de acessibilidade para páginas *Web*, serão desenvolvidos *scripts* utilizando a linguagem JavaScript, os quais serão executados pela ferramenta de testes Cypress que validará se as normas de acessibilidade definidas pelo guia W3C/WCAG estão sendo seguidas, servindo assim de insumo para a geração de um relatório com os apontamentos de falhas e melhorias de acessibilidade que serão efetuadas nas páginas alvos dos testes. Também será realizado um experimento onde os resultados obtidos com a execução da ferramenta serão comparados com resultados obtidos ao se utilizar outra ferramenta já utilizada no mercado e meio acadêmico para este fim, buscando com isso validar se a solução proposta possui a mesma ou uma melhor capacidade na detecção de violações de acessibilidade.

1.2.1 Etapas metodológicas

Nesta seção, serão apresentadas as etapas que representam como será a abordagem metodológica, as quais servirão de apoio para o desenvolvimento do trabalho.

A Figura 1 apresenta as etapas metodológicas utilizadas na composição:

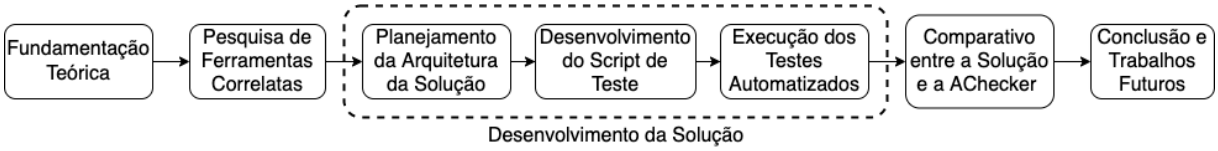


Figura 1 - Etapas metodológicas do trabalho (Autor 2021).

A primeira etapa consiste na fundamentação teórica, quando os conteúdos específicos relacionados ao tema do trabalho serão analisados. Dessa forma, serão apresentados conceitos fundamentais relacionados à acessibilidade em páginas *Web*, tipos de deficiência, boas práticas e guias para acessibilidade em páginas *Web*, testes de acessibilidade e automação de testes de acessibilidade.

Na segunda etapa será realizada uma pesquisa de ferramentas existentes que possuam relação com a ferramenta proposta no projeto deste trabalho. Esta etapa consiste em uma

descrição das ferramentas existentes, extraindo as suas principais características, e comparando com a ferramenta proposta.

Na terceira etapa do trabalho, será efetuado o desenvolvimento da solução para automação dos testes de acessibilidade usando como referência as normas do W3C/WCAG. Nesta etapa serão detalhadas as tecnologias e algoritmos escolhidos para a elaboração da solução e o desenvolvimento do *script* de teste. Por último, será efetuada a execução do projeto de testes automatizados e geração de um relatório com apontamentos de falhas encontradas e possíveis melhorias.

Na quarta etapa, com o intuito de demonstrar se o trabalho atingiu os objetivos estabelecidos, além de considerar a execução automatizada dos testes e geração do relatório com sucesso, será aplicado o método comparativo. Com isso, será efetuado um experimento em que os resultados apontados na execução automatizada dos testes da ferramenta desenvolvida neste trabalho serão comparados com os resultados obtidos através da execução de uma ferramenta amplamente reconhecida e usada no meio acadêmico e no mercado. O experimento será executado pelo próprio autor do trabalho que possui vasta experiência em testes de software. Ao fim, é esperado ratificar se o presente trabalho será útil para auxiliar no desenvolvimento de páginas *Web* mais acessíveis.

Na quinta e última etapa, será apresentada a conclusão do trabalho com os principais resultados obtidos, além de trabalhos que poderão ser desenvolvidos futuramente a partir deste estudo.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, apresentam-se os conceitos teóricos sobre os tipos de deficiências, acessibilidade na *Web*, normas estabelecidas pelo W3C, bem como tipos de testes manuais e automatizados de acessibilidade em páginas *Web*.

2.1 TIPOS DE DEFICIÊNCIA

O Centro de Controle e Prevenção de Doenças (2020), define que deficiência é qualquer condição do corpo ou da mente que dificulta que a pessoa com a condição realize certas atividades e interaja com o mundo ao seu redor (CDC, 2020). Já Maior (2021), traz uma definição mais ampla de deficiência, afirmando ser uma construção social em um processo constante de evolução, que depende de a sociedade assumir a sua responsabilidade nos processos de inclusão e envolvimento da pessoa com deficiência na vida da comunidade.

Segundo o CDC (2020), existem muitos tipos de deficiência que podem afetar uma pessoa, sendo elas:

- Visuais
- Movimento
- Raciocínio
- Memória
- Aprendizagem
- Comunicação
- Audição
- Relações sociais

A legislação brasileira, no decreto nº 5.296/2004, categoriza os diferentes tipos de deficiência como: “deficiência física, visual, auditiva, mental (funcionamento intelectual ou cognitiva significativamente inferior à média) e deficiência múltipla (associação de duas ou mais deficiências)” (BRASIL, 2004).

Para Brasil (2004), a deficiência física consiste em:

> Alteração completa ou parcial de um, ou mais segmentos do corpo humano, acarretando o comprometimento da função física, apresentando-se sob a forma de paraplegia, paraparesia,

> monoplegia, monoparesia, tetraplegia, tetraparesia, triplegia, triparesia, hemiplegia, hemiparesia, ostomia, amputação ou ausência de membro, paralisia cerebral, nanismo, membros com deformidade congênita ou adquirida, exceto as deformidades estéticas e as que não produzam dificuldades para o desempenho de funções (BRASIL, 2004).

A deficiência física engloba alguns impedimentos na movimentação, seja nos membros superiores, no caminhar ou no equilíbrio do corpo, o que pode comprometer o indivíduo em graus diferentes, podendo ser uma paralisia parcial ou total, ou ainda a falta de força. As pessoas com deficiência física podem apresentar limitações para executar determinados movimentos como levantar os braços, se virar e sair de um veículo. Elas também podem possuir limitações para ir e vir, obedecer a orientações para ficarem paradas, alterarem posições e se protegerem, caso necessitem. Para trazer qualidade de vida às pessoas com deficiência física e lhes proporcionar uma melhor funcionalidade, são utilizados alguns equipamentos como bengalas, estrutura para apoiar membros superiores, próteses, muletas e outros (MAIOR, 2021).

Brasil (2004) define deficiência auditiva como “perda bilateral, parcial ou total, de quarenta e um decibéis (dB) ou mais, aferida por audiograma nas frequências de 500Hz, 1.000Hz, 2.000Hz e 3.000Hz”. As pessoas que possuem deficiência auditiva podem ser enquadradas em dois grupos, o primeiro grupo é o de pessoas que antes ouviam e desenvolveram a comunicação oral. Caso elas tenham sido alfabetizadas, utilizam a língua portuguesa para escrever ou ler legendas e, assim, interagir com outras pessoas. Já o segundo grupo consiste em pessoas que já nasceram sem audição, ou que a perderam antes de aprender a falar. Estas pessoas usam a língua de sinais para se comunicar, podendo ou não falar e, caso consigam, é perceptível alguma alteração na forma que o fazem. Não é raro que pessoas do segundo grupo tenham a capacidade insuficiente de ler e escrever (MAIOR, 2021).

Para Brasil (2014), deficiência visual consiste em:

> Cegueira, na qual a acuidade visual é igual ou menor que 0,05 no melhor olho, com a melhor correção óptica; a baixa visão, que significa acuidade visual entre 0,3 e 0,05 no melhor olho, com a melhor correção óptica; os casos nos quais a somatória da medida do campo visual em ambos os olhos for igual ou menor que 60o; ou a ocorrência simultânea de quaisquer das condições anteriores (Brasil, 2004).

Ainda, sobre a definição de deficiência visual, Conde (2021) afirma que:

> Diversamente do que poderíamos supor, o termo cegueira não é absoluto, pois reúne indivíduos com vários graus de visão residual. Ela não significa, necessariamente, total incapacidade para ver, mas, isso sim, prejuízo dessa aptidão a níveis incapacitantes para o exercício de tarefas rotineiras (CONDE, 2021).

Já a deficiência visual costuma ser dividida em cegueira e baixa visão. Na baixa visão a pessoa se encontra em um meio-termo entre a cegueira e a possibilidade de enxergar completamente. A baixa visão se manifesta de muitas maneiras, ou seja, pessoas com essa condição, enxergam com percepção e intensidades distintas. Tudo depende da origem do comprometimento, de como ele se apresenta e de como a pessoa está acostumada a ele. Já na cegueira, o indivíduo pode ter a perda total da visão ou possuir alguma percepção baixa a luz ou vultos, dessa forma, algumas pessoas não conseguem enxergar absolutamente nada enquanto algumas pessoas conseguem distinguir vultos ou se a luz de um determinado ambiente está acesa ou apagada (IFPB, 2018).

Segundo Brasil (2004), a deficiência intelectual está relacionada com:

> Funcionamento intelectual significativamente inferior à média, com manifestação antes dos dezoito anos e limitações associadas a duas ou mais áreas de habilidades adaptativas, tais como: 1. comunicação; 2. cuidado pessoal; 3. habilidades sociais; 4. utilização dos recursos da comunidade; 5. saúde e segurança; 6. habilidades acadêmicas; 7. lazer; e 8. trabalho (BRASIL, 2004).

Quanto à deficiência intelectual, esta pode ser definida como um distúrbio envolvendo a presença de desenvolvimento mental incompleto ou interrompido. Ela é caracterizada primordialmente pela deterioração das funções mentais em cada estágio do desenvolvimento do indivíduo e que contribui para a composição de funções cognitivas, de linguagem, motoras e de socialização. Neste tipo de deficiência, a adaptação ao ambiente sempre é afetada. Para determinar o tipo e grau de deficiência intelectual são levados em conta sinais clínicos, comportamento adaptativo no meio cultural ao qual a pessoa está inserida, e também medidas psicométricas (KATZ; LAZCANO-PONCE, 2008).

Por fim, a deficiência múltipla consiste na associação de duas ou mais deficiências. Um exemplo que pode ser classificado como deficiência múltipla é a paralisia cerebral, onde o indivíduo possui limitações intelectuais e motoras que podem afetar a visão, a movimentação, a audição e as funções cognitivas. As pessoas com deficiência múltipla possuem níveis distintos

de autonomia, enquanto algumas têm uma grande independência, outras necessitam de cuidados permanentes (MAIOR, 2021).

Atualmente, os impedimentos, sejam eles mentais, intelectuais, físicos ou sensoriais, são considerados como intrínsecos à pluralidade do ser humano, de forma que a deficiência pode ser definida como um resultado da interação destes impedimentos com os obstáculos sociais, que possam dificultar a incorporação do indivíduo à sociedade (MPPR, 2021). Sendo assim, embora tenhamos uma definição dos tipos de deficiência, sejam em leis específicas ou em livros e artigos, apenas uma pessoa com deficiência é totalmente capaz de definir com exatidão quais são as suas limitações e classificar o tipo de deficiência que possui.

2.2 ACESSIBILIDADE NA *WEB*

Rachadel (2017), no seu trabalho intitulado Acessibilidade *Web*: uma análise sobre suas ferramentas e diretrizes, define a acessibilidade como:

> Acessibilidade é a qualidade atribuída àquilo que é acessível, ou seja, àquilo que é alcançável, àquilo que possui acesso fácil no trato e na aquisição. Nas últimas décadas ela está diretamente ligada às preocupações de arquitetos e urbanistas em prol do fornecimento de ensejos às pessoas portadoras de deficiência ou com mobilidade reduzida para o direito de aproveitamento dos espaços públicos com total segurança e autonomia.

Segundo a World Wide Web Consortium (W3C), acessibilidade na *Web* significa que sites *Web*, tecnologias e ferramentas sejam projetadas e desenvolvidas visando que pessoas com deficiência consigam usá-las. Sendo mais específico, elas devem ser desenhadas de forma que as pessoas com deficiência consigam navegar, interagir e contribuir com a *Web*. Nesse sentido, acessibilidade na *Web* pode ser definida como a possibilidade de qualquer indivíduo que possua alguma deficiência usá-la e interagir com ela (W3C, 2019).

Acessibilidade na *Web* refere-se à prática de tornar as páginas da *Web* acessíveis para todos os usuários, especialmente aqueles com deficiência. Embora uma *Web* acessível signifique um acesso sem precedentes à informação para pessoas com deficiência, pesquisas recentes sugerem que as melhores práticas em acessibilidade ainda não foram alcançadas (HARPER; YESILADA, 2008). Portanto, é possível afirmar a importância da acessibilidade na *Web* e o significado que ela possuiu para aqueles que dela dependem.

Para Cusin e Vidotti (2019), “a acessibilidade *Web* significa que pessoas com necessidades especiais podem usar a *Web*. Especificamente, significa que pessoas com necessidades especiais podem compreender, entender, navegar, interagir e contribuir com a *Web*”. A partir dessa afirmação, podemos chegar à conclusão que, para termos uma *Web* acessível, é necessário que pessoas com as mais variadas deficiências consigam encontrar páginas *Web* que as permitam interagir e contribuir.

2.3 WORLD WIDE WEB CONSORTIUM (W3C)

Em 1989, Tim Berners-Lee inventou e cunhou o termo World Wide *Web* e também desenvolveu o primeiro servidor da *WEB,* o CERN httpd. Além disso, foi responsável pelo desenvolvimento do primeiro programa cliente conhecido como navegador WorldWide*Web*, em outubro de 1990. Ele também escreveu a primeira versão do HyperText Markup Language (HTML), a linguagem de formatação de documentos com capacidade para links de hipertexto que se tornou o principal formato de publicação para a *Web* (W3C, 2021).

Em outubro de 1994, Tim Berners-Lee, conhecido como o pai da *Internet*, fundou o W3C[1] (World Wide Web Consortium). Sua fundação ocorreu no Instituto de Tecnologia de Massachusetts, no Laboratório de Ciência da Computação (MIT/LCS) em colaboração com o CERN, de onde a *Web* se originou com o apoio da DARPA e da Comissão Europeia (W3C, 2021).

O W3C é um consórcio internacional em que organizações filiadas, com uma equipe em tempo integral e o público em geral trabalham juntos para desenvolver padrões para a *Web*. Ele é liderado por Tim Berners-Lee e o CEO Jeffrey Jaffe, tendo como missão conduzir a *Web* para atingir todo seu potencial, desenvolvendo protocolos e diretrizes que garantam seu crescimento de longo prazo (W3C, 2021).

Atualmente, o W3C (2021) conta com cerca de 450 membros, entre os quais estão grandes empresas de tecnologia como Google, Facebook, Amazon, Oracle e outras. Isto demonstra a magnitude e relevância do consórcio para a criação e manutenção de protocolos e diretrizes que buscam fazer com que a *Web* atinja todo o seu potencial.

[1] https://www.w3.org/

2.4 NORMAS DE ACESSIBILIDADE DIGITAL

Por iniciativa de alguns indivíduos da diretoria do W3C, em 1996, a acessibilidade *Web* passou a ser considerada oficialmente como um projeto da organização. Em 1997 era lançado oficialmente pelo W3C o WAI[2] (*Web* Accessibility Initiative), iniciativa que visa desenvolver especificações técnicas, normas, e recursos de suporte que buscam apresentar soluções de acessibilidade transformando assim a *Web* em um lugar acessível (DARDAILLER, 2009).

Segundo Henry (2018), é essencial que vários componentes diferentes do desenvolvimento e interação da *Web* trabalhem juntos em sinergia para que a *Web* seja acessível a pessoas com deficiência. Estes componentes incluem:

- Conteúdo: Todas as informações contidas em uma página ou sistema *Web*, incluindo informações naturais como textos, imagens e sons, ou códigos que definem a estrutura da página e a sua apresentação;
- Navegadores de *Internet*, players de mídia, e outros "Agentes de usuário";
- Tecnologias assistivas, em alguns casos leitores de tela, teclados alternativos, software de escaneamento de texto etc.;
- Conhecimento do usuário, sua experiência, e em alguns casos as suas estratégias de adaptação para uso da *Web*;
- Desenvolvedores, incluindo designers, programadores, autores de conteúdo, também desenvolvedores com deficiência e usuários que contribuem no desenvolvimento etc.;
- Ferramentas de desenvolvimento usadas na criação dos sistemas *Web*;
- Ferramenta de avaliação como ferramentas para avaliação de acessibilidade, validação de HTML, validadores de CSS etc.

Ainda, conforme descreve Henry (2018), a WAI possui três principais guias que buscam trazer normas que possibilitem alcançar a acessibilidade *Web*:

- Normas para ferramentas de desenvolvimento de conteúdo para *Web* — Authoring Tool Accessibility Guidelines (ATAG);

[2] https://www.w3.org/WAI/

- Normas de acessibilidade para o agente do usuário — User Agent Accessibility Guidelines (UAAG);
- Normas de acessibilidade para conteúdo da *Web* — *Web* Content Accessibility Guidelines (WCAG).

Para o autor (2018), cada guia está ligada diretamente a um determinado componente, conforme descrito na figura 2:

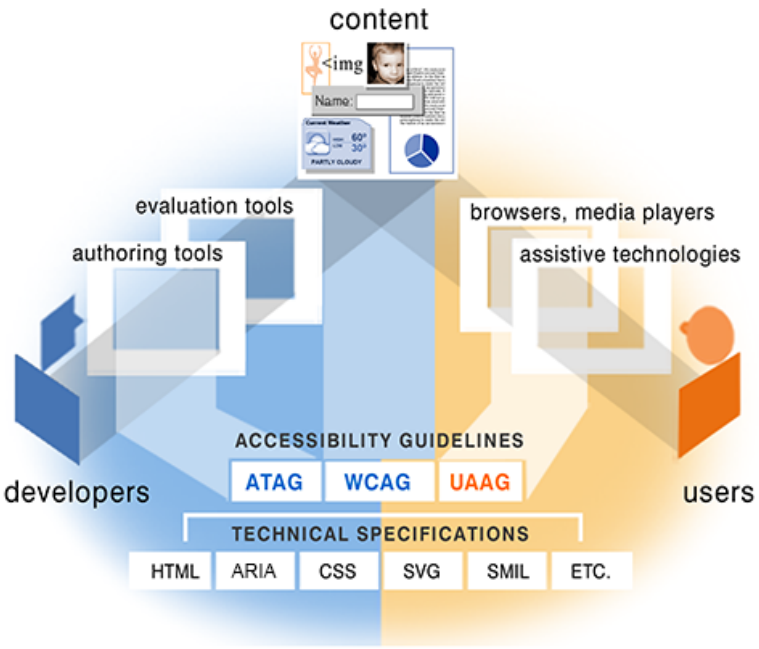


Figura 2 - Normas de acessibilidade e seus componentes (W3C, 2021)

As ATAG[3] são normas que devem ser absorvidas nas ferramentas de desenvolvimento *Web*, como, por exemplo, Content Management Systems (CMS), Integrated Development Environment (IDE) e outras, que visam facilitar a integração das normas nas fases de desenvolvimento.

Já as normas WCAG buscam verificar e guiar a acessibilidade ao conteúdo das páginas *Web*, seja um elemento textual ou audiovisual.

As normas UAAG são as normas que visam a acessibilidade nos navegadores, dos reprodutores multimídia ou qualquer outro componente que faça parte da renderização e exibição da página ou sistema *Web* (HENRY, 2018).

[3] https://www.w3.org/WAI/standards-guidelines/atag/

2.4.1 *Web* Content Accessibility Guidelines (WCAG)

Desenvolvido pelo W3C em cooperação com indivíduos e organizações em todo o mundo e publicado pela iniciativa WAI, o guia WCAG[4] é um conjunto de normas projetadas para tornar a *Web* mais acessível para pessoas com deficiência. O guia possui três versões, sendo que a terceira versão está em desenvolvimento. O WCAG é atualizado ao longo dos anos para considerar as mudanças nas tecnologias baseadas na *Web*, tecnologias assistivas, tendências de design, e o desenvolvimento e crescimento da *Web* em dispositivos móveis (BUREAU OF *INTERNET* ACCESSIBILITY, 2019).

Sobre as normas, o W3C (2008) afirma que:

> O cumprimento destas normas fará com que o conteúdo se torne acessível a um maior número de pessoas com incapacidades, incluindo cegueira e baixa visão, surdez e baixa audição, dificuldades de aprendizagem, limitações cognitivas, limitações de movimentos, incapacidade de fala, fotossensibilidade bem como as que tenham uma combinação destas limitações. Seguir estas diretrizes fará também com que o conteúdo *Web* se torne mais usável aos utilizadores em geral (W3C, 2008).

Fica evidente que a criação e cumprimento das normas estabelecidas no WCAG são essenciais para que uma página *Web* seja acessível e permita que pessoas com deficiência a utilizem. Assim, demonstra que as normas não têm apenas sentido teórico, mas também um sentido prático que deve ser considerado por todos os envolvidos no desenvolvimento e publicação de páginas *Web*.

2.4.1.1 WCAG 1.0

Em maio de 1999, era lançado o WCAG 1.0, a primeira versão do *Web* Content Accessibility Guidelines, com catorze normas, cada uma composta de um a dez *checkpoints* de verificação (W3C, 1999):

1. Fornecer alternativas equivalentes para conteúdo auditivo e visual.

[4] https://www.w3.org/WAI/standards-guidelines/wcag/

2. Garantir que conteúdos expressados em cores possam ser entendidos mesmo sem cores.
3. Quando houver uso de folhas de estilos e formatação, que sejam aplicadas corretamente.
4. Quando usada a linguagem natural nos documentos da *Web*, ela deve ser declarada.
5. Garantir que tabelas possam ser transformadas por navegadores acessíveis e outros agentes.
6. Assegurar que páginas sejam acessíveis mesmo com o surgimento de novas tecnologias ou quando determinadas tecnologias deixarem de ser utilizadas.
7. Certificar controle ao usuário sob conteúdos sensíveis ao tempo como movimento, *scrolling*, *update* automático, certificando que a página possa ser pausada ou parada conforme a necessidade.
8. Garantir acessibilidade diretamente seguindo os princípios do *design* acessível, utilizando o princípio de dispositivo independente.
9. *Design* para dispositivos independentes, que seja possível ao usuário interagir com os elementos da página via uma grande variedade de dispositivos de entrada.
10. Usar soluções internas para que tecnologias e navegadores acessíveis possam operar corretamente.
11. Utilizar os guias do W3C e suas tecnologias.
12. Prover a informação com contexto e orientação.
13. Providenciar mecanismos claros e consistentes para a navegação nas páginas.
14. Garantir que os documentos sejam claros e simples de modo a serem entendidos facilmente.

Cada uma das catorze normas listadas possui alguns *checkpoints* que servem como base para certificar o cumprimento das mesmas em adesão ao WCAG 1.0, sendo ao todo sessenta e cinco *checkpoints* que, por sua vez, são agrupados em três níveis de prioridade (W3C, 1999):

Nível 1: O desenvolvedor deve cumprir este *checkpoint*. Se o *checkpoint* não for cumprido será impossível que um ou mais grupo de pessoas tenha acesso à informação deste documento.

Os *checkpoints* pertencentes ao nível 1 são descritos no quadro abaixo:

Checkpoints do nível 1
Fornecer uma descrição em texto equivalente para cada elemento não textual (por exemplo, via "alt", "Longdesc" ou no conteúdo do elemento). Isso inclui: imagens, gráficos (incluindo símbolos), regiões do mapa de imagens, animações (por exemplo, GIFs animados), applets e objetos dinâmicos, arte ASCII, quadros, *scripts*, espaçadores, botões gráficos, sons (tocados com ou sem interação do usuário), arquivos de áudio, e vídeos.
Assegurar que todas as informações exibidas com cor também estejam disponíveis sem cor, por exemplo, marcação de texto.
Identificar nitidamente as alterações na linguagem natural do texto de um documento e quaisquer textos equivalentes (por exemplo, legendas).
Organizar documentos para poderem ser lidos sem folhas de estilo. Por exemplo, quando um documento HTML é renderizado sem as folhas de estilo associadas, ainda deve ser possível ler o documento.
Assegurar que os conteúdos equivalentes aos conteúdos dinâmicos sejam atualizados quando houver mudanças dos conteúdos dinâmicos originais.
Até que os agentes do usuário permitam que o usuário controle a tela, evitar causar movimento na tela.
Usar a linguagem mais clara e simples.
E se você usar imagens e mapas de imagem?
Fornecer links de texto redundantes para cada região ativa de um mapa de imagem.
Fornecer mapas de imagem do lado do cliente em vez de mapas de imagem do lado do servidor, exceto onde as regiões não podem ser definidas com uma forma geométrica disponível.
E se você usar tabelas?
Para tabelas de dados, identificar os cabeçalhos de linha e coluna.
Para tabelas de dados que possuem dois ou mais níveis lógicos de linhas ou cabeçalhos, usar a marcação para associar a células de dados e as células de cabeçalho.
E se você usar frames?
Dar um título a cada frame para facilitar a identificação e navegação do frame.
E se você usar applets e *scripts*?
Certificar que as páginas são utilizáveis quando *scripts*, applets ou outros objetos programáticos são desligados ou não suportados. Se isso não é possível, então fornecer informações sobre uma página alternativa que seja acessível.
E se você usar multimídia?
Até que os agentes do usuário possam ler automaticamente em voz alta o equivalente de texto de uma faixa visual, fornecer uma descrição auditiva das informações importantes da apresentação multimídia.
Para qualquer apresentação multimídia baseada em tempo (por exemplo, um filme ou animação), disponibilizar alternativas equivalentes, por exemplo, legendas ou descrições auditivas da apresentação multimídia.
E se tudo mais falhar?

Se, após os melhores esforços, você não conseguir criar uma página acessível, então forneça um link para uma página alternativa que use tecnologias W3C, que seja acessível, e possua informações equivalentes (ou funcionalidade equivalentes) enquanto a página original é atualizada.

Quadro 1 - Checkpoints do nível 1 (Adaptado de W3C, 2014).

Nível 2: O desenvolvedor tem que cumprir este *checkpoint*. Se o *checkpoint* não for cumprido um ou mais grupos terão dificuldades de acessar a informação contida no documento.

Os checkpoints pertencentes ao nível 2 são descritos a seguir:

Checkpoints do nível 2
Garantir que as combinações de cor de primeiro plano e cor de fundo oferecem contraste suficiente quando visto por alguém com déficits de cores ou quando visto em uma tela preta e branca. (Prioridade 2 para imagens, prioridade 3 para texto).
Quando existir uma linguagem de marcação apropriada, use a marcação em vez de imagens para transmitir informações.
Criar documentos que validam as gramáticas formais publicadas.
Usar folhas de estilo para controlar o layout e a apresentação.
Usar unidades relativas ao contrário de absolutas no atributo de idioma de marcação e nas propriedades de folha de estilo.
Utilizar elementos de cabeçalho para definir a estrutura do documento e então usar para especificação.
Marcar e listar itens de uma lista corretamente.
Marque as citações. Não usar a marcação de cotação para formatação como recuo.
Assegurar-se de que o conteúdo dinâmico seja acessível ou fornecer uma alternativa de apresentação ou página.
Até que os agentes do usuário permitam que os usuários controlem a atualização da página, evitar que a mesma seja atualizada.
Até que os agentes do usuário forneçam a capacidade de parar a atualização, não executar atualizações periódicas na página.
Até que os agentes do usuário forneçam a capacidade de interromper o redirecionamento automático, não redirecionar páginas automaticamente. Em vez disso, configure o servidor para executar os redirecionamentos.
Até que os agentes do usuário permitam que os usuários desativem o surgimento de novas janelas, não gerar pop-ups ou abrir outras janelas sem informar o usuário.
Use tecnologias W3C quando estiverem disponíveis e apropriadas para uma tarefa, procure usar as versões mais recentes quando suportada.
Evitar características depreciadas das tecnologias W3C.
Dividir grandes blocos de informações em grupos mais gerenciáveis.
Identificar visivelmente o alvo de cada link.
Fornecer metadados, adicionar informações semânticas a páginas e locais.
Fornecer informações sobre o layout geral de um site (por exemplo, um mapa ou índice).
Usar mecanismos de navegação de maneira consistente.
E se você usar tabelas ?

Não use tabelas para layout, a menos que a tabela faça sentido quando linearizado. Caso contrário, se a tabela não faz sentido, forneça uma alternativa equivalente (que pode ser uma versão linearizada).
Se uma tabela for usada para layout, não use nenhuma marcação estrutural para a criação da formatação visual.
E se você usar quadros?
Descreva o propósito dos quadros e como os quadros se relacionam uns aos outros caso não esteja óbvio pelo título dos quadros.
E se você usar formulários?
Até que os agentes do usuário permitam associações explícitas entre rótulos e controles de formulário, para todos os controles de formulário com etiquetas implicitamente associadas, certifique-se de que o rótulo esteja posicionado corretamente.
Associe as legendas explicitamente com o que elas significam.
E se você usar applets e *scripts?*
Para *scripts* e applets, certifique-se de que os manipuladores de eventos sejam inseridos independente do dispositivo.
Até que os agentes do usuário permitam aos usuários pausar o conteúdo em movimento, evite o movimento nas páginas.
Fazer elementos programáticos, como *scripts* e applets diretamente acessíveis ou compatível com tecnologias assistivas (Prioridade 1 se a funcionalidade é importante e não apresentada em outro lugar, caso contrário Prioridade 2.)
Certificar-se de que qualquer elemento que tenha sua própria interface possa ser operado de uma maneira independente do dispositivo.
Para *scripts*, especificar manipuladores de eventos lógicos em vez de manipuladores de eventos dependentes do dispositivo.

Quadro 2 - Checkpoints do nível 2 (Adaptado de W3C, 2014).

Nível 3: O desenvolvedor *Web* poderia cumprir esse *checkpoint*. Se o *checkpoint* não for cumprido um ou mais grupos por vezes terão dificuldades de acessar a informação contida no documento. Satisfazendo esse *checkpoint* a acessibilidade será melhorada. Os checkpoints pertencentes ao nível 3 são descritos no quadro abaixo:

Checkpoints do nível 3
Especificar a forma completa de cada abreviação ou acrônimo em um documento onde ela ocorra primeiro.
Identificar a linguagem natural primária de um documento.
Criar uma ordem de guia lógica através de links, controles de formulário e objetos.
Fornecer atalhos de teclado a links importantes (incluindo aqueles em mapas de imagem do lado do cliente), controles de formulário e grupos de controles de formulários.
Até que os agentes do usuário (incluindo tecnologias assistivas) renderizem links adjacentes distintamente, inclua caracteres imprimíveis (separados por espaços) entre os links adjacentes.
Fornecer informações para que os usuários possam obter documentos de acordo com suas preferências (por exemplo, linguagem, tipo de conteúdo, etc.)
Fornecer mecanismos como barras de navegação para destacar e dar acesso à navegação.

Se as funções de pesquisa forem fornecidas, ative diferentes tipos de pesquisas para diferentes níveis de habilidade e preferências.
Adicionar a informação distintiva no início dos títulos, parágrafos, listas, etc.
Forneça informações sobre coleções de documentos (isto é, documentos compreendendo várias páginas.).
Forneça um meio para pular arte ASCII com múltiplas linhas.
Complementar o texto com apresentações gráficas ou auditivas, onde possa facilitar a compreensão da página.
Criar um estilo de apresentação que seja consistente entre as páginas.
E se você usar imagens e mapas de imagem?
Até que os agentes do usuário renderizem equivalentes de texto para o mapa de imagem do lado do cliente, forneça links de texto redundantes para cada região ativa do mapa de imagem do lado do cliente.
E se você usar tabelas?
Fornecer resumos para tabelas.
Fornecer abreviaturas para etiquetas de cabeçalho.
Até que os agentes do usuário (incluindo tecnologias assistivas), renderizem textos *side-by-side* corretamente, forneça uma alternativa de texto linear (na página atual ou alguma outra).
E se você usar formulários?
Até que os agentes do usuário lidem com controles de campos vazios corretamente, incluir por padrão, *place-holding characters* nos campos com edição de texto ou áreas de digitação.

Quadro 3 - Checkpoints do nível 3 (Adaptado de W3C 2014).

2.4.1.2 WCAG 2.0

Em 11 de dezembro de 2008, era lançado pelo W3C o WCAG 2.0, a segunda versão do guia. O seu lançamento foi motivado pelas grandes mudanças na tecnologia a partir dos anos 2000. Ele foi uma evolução do seu antecessor, planejado para poder ser aplicado a basicamente qualquer produto digital incluindo documentos e aplicativos (BUREAU OF *INTERNET* ACCESSIBILITY, 2019). Ou seja, o guia que antes regia normas para páginas e conteúdo para *Web*, passa a considerar conteúdo para outras plataformas e tecnologias como aplicativos mobile, por exemplo. As normas do WCAG 2.0 foram desenvolvidas em colaboração com pessoas e organizações em um esforço mundial, tendo como alvo fornecer um padrão compartilhado referente à acessibilidade do conteúdo da *Web*, que seja aderente às necessidades das pessoas, das organizações e dos governos (W3C, 2014).

A versão 2.0 do guia manteve compatibilidade com a versão 1.0, isto significa que se determinada página *Web* correspondesse com a versão 1.0, então ela estaria em grande parte consoante a versão 2.0, sendo que a sua atualização para adequação a versão 2.0 do guia seria mínima. Mesmo com o lançamento da versão 2.0, a versão 1.0 do guia não foi substituída,

porém, é recomendado o uso da versão mais nova do guia, para que assim mais dispositivos e formas de acesso sejam disponibilizados (WAI EOWG, 2021).

Segundo C*WEB* (2020) o WCAG 2.0 "foi reconhecido em 2012 pela Organização Internacional para Padronização (ISO) como um padrão internacional para acessibilidade *Web*, a ISO/IEC 40.500:2012, com tradução para português do Brasil autorizada pelo W3C e publicada em 24 de outubro de 2014". Isso demonstra um alto nível de maturidade do guia além da sua qualidade e excelência como conjunto de normas que devem ser seguidas por aqueles que desejam que suas páginas *Web* sejam acessíveis.

CALDWELL *et al.*, 2014 afirma que:

> As Diretrizes de Acessibilidade para Conteúdo *Web* (WCAG) 2.0 definem a forma de como tornar o conteúdo da *Web* mais acessível para pessoas com deficiência. A acessibilidade abrange uma vasta gama de deficiências, incluindo visual, auditiva, física, de fala, intelectual, de linguagem, de aprendizagem e neurológica. Embora estas diretrizes cubram uma ampla diversidade de situações, elas não são capazes de abordar as necessidades das pessoas com todos os tipos, graus e combinações de deficiências. Estas diretrizes tornam também o conteúdo da *Web* mais acessível por pessoas idosas, cujas habilidades estão em constante mudança devido ao envelhecimento, e muitas vezes melhoram a usabilidade para usuários em geral.

Sendo assim, as normas contidas no guia buscam abranger uma vasta gama de deficiências, porém mesmo com uma ampla cobertura, é inviável que o guia consiga abranger todos os tipos, graus e combinações de deficiências. O guia também busca tornar o conteúdo da *Web* mais acessível para pessoas idosas e passa, assim, a não considerar apenas aspectos relacionados a deficiências, mas aspectos relacionados aos fatores limitantes que surgem advindo da idade do usuário, sejam esses fatores físicos ou intelectuais.

Ainda, conforme o W3C (2014) "o documento WCAG 2.0 foi concebido para satisfazer as necessidades daqueles que precisam de um padrão técnico referenciável e estável". O guia é organizado em quatro princípios, cada um dos seus princípios conta com recomendações. Para cada recomendação, há critérios de sucesso a serem seguidos. Para ser possível seguir os critérios de sucesso, são divulgadas técnicas específicas (EMAG, 2020).

Figura 3 - Estrutura do WCAG 2.0 (EMAG, 2020)

Segundo Fenner (2021), os padrões de acessibilidade definidos no WCAG 2.0 são divididos em 4 princípios, os quais contêm diretrizes com critérios de sucesso e técnicas específicas sendo eles:

a) **Perceptível:** este princípio define que em uma página o seu conteúdo deve ser apresentado em mais de uma forma. Para alcançar esse objetivo, o guia recomenda que em conteúdo multimídia como vídeos, imagens e outros, se tenha uma descrição alternativa, e que o código HTML da página possa ser compreendido e lido por ferramentas leitoras de tela.

b) **Operável:** este princípio demonstra que para que uma página seja operável, todos os usuários que a usarem devem ter a plena capacidade de efetuar todas as operações disponíveis sem haver qualquer barreira ou dificuldade de acesso. Para isto, o princípio sugere que a codificação do HTML esteja adequada, assim permitindo que ocorra a navegação com o uso do teclado, isto é, que o usuário possa operar a página por outro meio além do mouse. Outro ponto que o princípio sugere é em relação à velocidade da página, a mesma deve possuir equilíbrio onde a velocidade da página não seja muito alta ou muito baixa e também se faz necessário a ordenação do cabeçalho para que ele seja estruturado organizadamente. Por último, o princípio sugere que sejam evitados elementos que possam ser estímulos para ataques epiléticos, como, por exemplo, utilização demasiada de *pop-ups* ou excesso de iluminação e de cores.

c) **Compreensível:** este princípio auxilia para que o conteúdo da página seja compreensível, para isto o princípio sugere que no conteúdo da página as sentenças sejam escritas de forma clara e objetiva, que se evitem expressões muito específicas e seja realizada uma escolha adequada das fontes. Essas medidas têm como consequência

transformar a página acessível para pessoas com deficiências intelectuais, com dislexia e também acabam sendo úteis para pessoas que não fazem parte de um determinado contexto ou possuem limitações de leitura, sendo um exemplo, pessoas com escolaridade baixa.

d) **Robusto:** este princípio visa auxiliar na codificação da página *Web*. No guia WCAG, é determinado que o HTML da página tem que estar ordenado para que tecnologias assistivas consigam executar na página e também para que a página seja navegável utilizando apenas o teclado, sendo este último ponto essencial para que pessoas que possuam algum tipo de deficiência motora utilizem a página.

Cada um dos quatro princípios conta com algumas normas, mesmo que genéricas. Dessa forma, elas possuem critérios de sucesso que são objetivos e devem ser cumpridos para satisfazê-las. Com o intuito de ajudar os desenvolvedores a atingirem os critérios de sucesso, o guia disponibiliza uma grande gama de técnicas que podem ser utilizadas (CE*WEB*, 2020). Chantre (2015), afirma que com base nesses princípios, foram apresentados doze guias de acessibilidade. Para cada guia, existem os critérios de sucesso associados, sendo 61 ao todo, que definem a sua testabilidade, bem como, um conjunto de especificações técnicas que podem ser utilizadas para alcançar a conformidade com as normas definidas pelo WCAG 2.0, conforme Figura 4.

Figura 4 - A estrutura do WCAG 2 (CHANTRE, 2015).

Para cada um dos quatro princípios existem normas estabelecidas, que por sua vez possuem critérios de sucesso que atestam se as normas foram seguidas ou não e, por último, são listadas várias técnicas que podem ser empregadas para o alcance dos critérios de sucesso definidos (CHANTRE 2015). O quadro Princípios e suas normas, lista as normas estabelecidas para cada um dos quatro princípios.

1	**Perceptível**
1.1	Alternativas em Texto: Fornecer alternativas textuais para qualquer conteúdo não textual, para que possa ser transformado em outras formas de acordo com as necessidades dos usuários, tais como impressão com tamanho de fontes maiores, braille, fala, símbolos ou linguagem mais simples.
1.2	Fornecer alternativas para mídias baseadas em tempo.
1.3	Criar conteúdo que pode ser apresentado de diferentes maneiras (por exemplo, um layout simplificado) sem perder informação ou estrutura.
1.4	Facilitar a audição e a visualização de conteúdo aos usuários, incluindo a separação entre o primeiro plano e o plano de fundo.
2	**Operável**
2.1	Fazer com que toda funcionalidade fique disponível a partir de um teclado.
2.2	Fornecer aos usuários tempo suficiente para ler e utilizar o conteúdo.
2.3	Não criar conteúdo de uma forma conhecida por causar convulsões.
2.4	Fornecer maneiras de ajudar os usuários a navegar, localizar conteúdos e determinar onde se encontram.
3	**Compreensível**
3.1	Tornar o conteúdo de texto legível e compreensível.
3.2	Fazer com que as páginas *Web* apareçam e funcionem de modo previsível.
3.3	Ajudar os usuários a evitar e corrigir erros.
4	**Robusto**
4.1	Maximizar a compatibilidade entre os atuais e futuros agentes de usuário, incluindo tecnologias assistivas.

Quadro 4 - Princípios e suas normas (Caldewell, 2014).

Chantre (2015), ainda expõe que no WCAG 2.0, são definidos níveis de conformidade que se relacionam a cada critério de sucesso. Estes níveis são classificados conforme Figura 5:

Figura 5 - Descrição dos níveis de acessibilidade do WCAG 2.0 (CHANTRE, 2015)

O nível A corresponde ao menor nível. Ao atingir o nível A são alcançados os níveis mínimos desejados de acessibilidade, ou seja, atingir tal nível não garante que uma página seja acessível, porém pode significar a ausência de acessibilidade no geral.

O nível AA é o nível intermediário, o qual, quando alcançado, é garantido que a página seja bastante acessível para a maioria dos usuários, cobrindo uma gama grande de tecnologias. Ou seja, ao atingir o nível AA, a página poderá ser acessada com poucas dificuldades por pessoas com as mais variadas deficiências e usando os mais variados meios de acesso.

O nível AAA, também chamado nível triplo A, é o nível mais alto de conformidade que uma página pode alcançar. Este nível de acessibilidade tem como característica ser bastante específico e meticuloso. Manter conformidade com este nível é algo custoso e possui uma implementação difícil, pois a maioria dos critérios relacionados a ele são referentes a situações específicas. Ao obter este nível é garantido a acessibilidade total a página, independente da deficiência ou do meio de acesso de quem as consome (CE*WEB*, 2021).

Princípios	**Normas**	**Nível A**	**Nível AA**	**Nível AAA**
1. Perceptível	1.1 Alternativas em texto	1.1.1		
	1.2 Mídia baseada em tempo	1.2.1; 1.2.3	1.2.4; 1.2.5	1.2.6; 1.2.9
	1.3 Adaptável	1.3.1; 1.3.3		
	1.4 Discernível	1.4.1; 1.4.2	1.4.3; 1.4.5	1.4.6; 1.4.9
2. Operável	2.1 Acessível por teclado	2.1.1; 2.1.2		2.1.3
	2.2 Tempo suficiente	2.2.1; 2.2.2		2.2.3; 2.2.5
	2.3 Convulsões e reações físicas	2.3.1		2.3.2
	2.4 Navegável	2.4.1; 2.4.4	2.4.5 2.4.7	2.4.8; 2.4.10
3. Compreensível	3.1 Legível	3.1.1	3.1.2	3.1.3; 3.1.6

		3.2 Previsível		3.2.1; 3.2.2		3.2.3; 3.2.4		3.2.5
		3.3 Assistência de entrada		3.3.1; 3.3.2		3.3.3; 3.3.4		3.3.5; 3.3.6
4. Robusto		4.1 Compatível		4.1.1; 4.1.2				

Quadro 5 - Distribuição das normas e níveis de conformidade do WCAG 2.0 (Adaptado de W3C, 2014).

2.4.1.3 WCAG 2.1

Em 5 de junho de 2018, foi lançada a versão 2.1 do WCAG. Esta nova versão, não veio para substituir a versão anterior, mas sim para complementá-la, ou seja, o WCAG 2.1 mantém compatibilidade com WCAG 2.0. Isso significa que, caso a página esteja em conformidade com o WCAG 2.1, ela estará automaticamente em conformidade com a versão 2.0. A versão 2.1 do guia foi muito bem recebida, pois após uma década de avanços tecnológicos, a versão 2.0 ficou defasada. A versão 2.1 trouxe novos critérios de sucesso para melhorar a acessibilidade em dispositivos móveis, e também melhorar a acessibilidade para pessoas com baixa visão e deficiências cognitivas (BUREAU OF *INTERNET* ACCESSIBILITY, 2019).

No WCAG 2.1 foram adicionados novos critérios de sucesso:

Princípios		**Normas**		**Nível A**		**Nível AA**		**Nível AAA**
1. Perceptível		1.3 Adaptável				1.3.4 Orientação 1.3.5 Identificar o objetivo de entrada		1.3.6 Identificar o objetivo
		1.4 Discernível				1.4.10 Realinhar 1.4.11 Contraste não textual 1.4.12 Espaçamento de texto 1.4.13 Conteúdo em foco por mouse ou teclado		
2. Operável		2.1 Acessível por teclado		2.1.4 Atalhos de teclado por caractere				
		2.2 Tempo suficiente		2.2.1; 2.2.2				2.2.6 Limites de Tempo

		2.3 Convulsões e reações físicas						2.3.3 Animação de Interações
		2.5 Modalidades de entrada		2.5.1 Gestos de Acionamento 2.5.2 Cancelamento de Acionamento 2.5.3 Rótulo em Nome Acessível 2.5.4 Atuação em Movimento				2.5.5 Tamanho da área clicável 2.5.6 Mecanismos de Entrada Simultâneos
4. Robusto		4.1 Compatível				4.1.3 Mensagens de Status		

Quadro 6 - Novos critérios de sucesso no WCAG 2.1 (Adaptado de W3C, 2018).

2.5 TIPOS DE TESTES DE ACESSIBILIDADE *WEB*

Existem muitas formas para se testar a acessibilidade de páginas *Web*. Comumente, elas são divididas em dois grandes grupos, um grupo é composto por avaliações técnicas que são geralmente executadas por especialistas que possuem conhecimentos de *Web design* e acessibilidade. O outro grupo é composto pela avaliação da experiência do usuário que conta com a participação de usuários comuns da *Internet*. Os grupos são subdivididos em testes automatizados e testes manuais. A figura 6 demonstra como são divididos os tipos de testes de acessibilidade (HASSANZADEH; NAVIDI, 2010). A figura 6 demonstra como são divididos os tipos de testes de acessibilidade, nas próximas subseções, estes tipos de testes são descritos de forma mais detalhada.

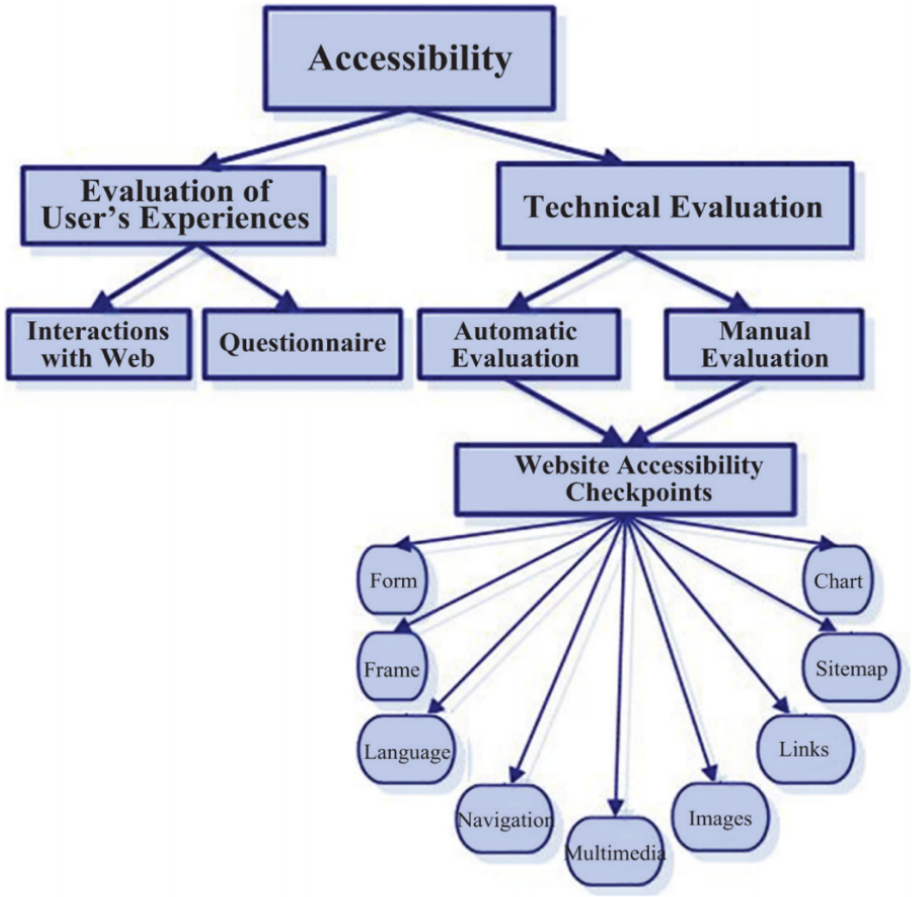


Figura 6 - Divisão dos testes de acessibilidade (Hassanzadeh e Navidi, 2010).

2.5.1 Testes Manuais De Acessibilidade *Web*

Os testes manuais podem ser executados por dois grupos distintos, sendo o primeiro deles composto por pessoas técnicas. Testes manuais executados por este grupo consistem na revisão de detalhes específicos de acessibilidade de páginas *Web*. Para efetuar os testes, é executado pelos especialistas um método que possui várias etapas que buscam examinar as páginas em acordo com os *checkpoints* de acessibilidade. Após os especialistas avaliarem a página, tamanhos de fonte, arquivos de áudio, imagens, carregamento de *frames* etc. é definido o quão acessível são as páginas avaliadas e o que precisa ser ajustado para que as páginas sejam acessíveis consoante as normas estabelecidas. Esse primeiro grupo possui como vantagem a possibilidade de identificação de um número considerável de problemas de acessibilidade para usuários com as mais variadas deficiências e possui um bom custo benefício. Todavia, como desvantagem, esse método de teste depende muito que o avaliador tenha grande conhecimento das normas e indicadores de acessibilidade e consiga sugerir melhorias e/ou correções. Além

disso, este método pode ser executado de uma forma não técnica ou profissional como deveria (HASSANZADEH; NAVIDI, 2010).

O segundo grupo de testes manuais consiste nos testes baseados na experiência do usuário. A execução desses testes por vezes conta com a participação de especialistas e consiste em uma série de testes executadas pelos usuários com acompanhamento de especialistas onde são apontados erros e dificuldades de acessibilidade no acesso dessas páginas. Esse método possui como vantagem uma alta acurácia na identificação dos problemas de acessibilidade enfrentados pelos usuários e também conta com o benefício de ter as páginas testadas por usuários que possuem deficiências variadas. No entanto, como desvantagem, esse método possui um custo elevado e leva muito tempo para ser executado, além disso, os resultados podem acabar sendo genéricos e determinados por um grupo específico de pessoas (HASSANZADEH; NAVIDI, 2010).

Para efetuar os testes manuais a WAI/W3C disponibiliza um documento intitulado Easy Checks[5] que auxiliam nos testes iniciais de acessibilidade em páginas *Web*. Este documento foi projetado para auxiliar na execução de testes rápidos e fáceis, cobrindo apenas alguns problemas de acessibilidade. Porém, já se demonstra suficiente enquanto um guia inicial, que ajuda a detectar problemas na acessibilidade das páginas testadas **(**HENRY, 2017).

2.5.2 Testes Automatizados De Acessibilidade *Web*

Hassanzadeh e Navidi (2010), definem testes automatizados de acessibilidade *Web* como:

> São testes executados por ferramentas que avaliam se as páginas determinadas estão de acordo com as normas estabelecidas pelo W3C ou pela regulamentação 508 do governo dos Estados Unidos da América. Essas ferramentas são muito convenientes, pois elas economizam tempo e esforço humano nos testes de acessibilidade.

As ferramentas de testes automatizados de acessibilidade podem ajudar na detecção rápida de problemas de acessibilidade nas páginas testadas. Elas podem ser usadas durante as fases de *design* e desenvolvimento de páginas *Web*. Não é possível que essas ferramentas consigam testar todos os aspectos de acessibilidade, sendo assim, é necessário o julgamento e discernimento de pessoas, pois estas ferramentas podem acabar produzindo resultados falsos ou enganosos (ABOU-ZAHRA, 2017).

[5] https://www.w3.org/WAI/test-evaluate/preliminary/

Atualmente, são encontradas demasiadas ferramentas para testes automatizados de acessibilidade *Web* (ABOU-ZAHRA, 2009). Algumas ferramentas são focadas em testes de acessibilidade em páginas *Web*, enquanto outras ferramentas também realizam validação da garantia de qualidade e usabilidade. Essas ferramentas possuem diversas formas de relatórios de execução, incluindo relatórios em formato HTML, em formato XML, arquivos enviados por *e-mail* e outras (VIGO et al., 2013).

As ferramentas para testes automatizados de acessibilidade em páginas *Web* são direcionadas para diferentes públicos, que vão desde autores de conteúdo, *designers*, testadores de *software*, desenvolvedores até mesmo usuários finais. Estas ferramentas oferecem vários recursos e funcionalidades que permitem ao usuário comparar e avaliar qual é a mais adequada para atender às suas necessidades (ABOU-ZAHRA, 2017).

O site da WAI/W3C mantém uma lista que conta com um total de 157 ferramentas para efetuar os testes automatizados de acessibilidade em páginas *Web*. Na lista são encontradas ferramentas de vários tipos e cobrem outras normas além das estabelecidas no WCAG. No quadro abaixo, é mostrado a quantidade de ferramentas disponíveis em relação aos guias que estas utilizam para os testes (EGGERT, 2016):

Guias	**Quantidade de ferramentas**
WCAG 2.1	75
WCAG 2.0	129
WCAG 1.0	43
BITV, German government standard	21
RGAA, French government standard	12
JIS, Japanese industry standard	18
EN 301 549, European accessibility standard	8
EPUB Accessibility 1.0	4
Irish National IT Accessibility Guidelines	16
MAAG 1.0 -Korea government standard	1
Section 508, US federal procurement standard	67
SI 5568, Israeli *Web* accessibility guidelines	7
Stanca Act, Italian accessibility legislation	11

Quadro 7 - Guias e quantidades de ferramentas disponíveis (Elaborado pelo autor. Adaptado de EGGERT (2016).)

Ao interpretar o Quadro 3 é possível constatar que a versão 2.0 do WCAG é suportada por um número muito grande de ferramentas, o que reforça ainda mais a importância e reconhecimento que o guia WCAG possui. Além disto, é possível constatar que existem outros guias de acessibilidade para *Web*, suportados por outras ferramentas de testes de acessibilidade em páginas *Web*, mostrando, assim, uma boa gama de guias que podem auxiliar no desenvolvimento de páginas *Web* acessíveis.

3 ESTADO DA ARTE

Este capítulo tem o objetivo identificar o estado atual em que se encontram ferramentas, sejam elas comerciais ou gratuitas e pesquisas que se relacionem ao projeto deste trabalho. Ou seja, testes automatizados de acessibilidade para páginas *Web*, tendo como intenção analisar como essas ferramentas se posicionam em relação à solução proposta neste trabalho.

3.1 DEFINIÇÃO DO ESTUDO

Para efetuar a pesquisa sobre trabalhos e ferramentas, foram utilizadas algumas fontes de pesquisa além da própria *Web*, com intuito de localizar ferramentas disponíveis e artigos que se relacionam com as tecnologias utilizadas na composição deste trabalho. Entre as fontes de pesquisa utilizadas estão:

- ACM Digital Library[6]
- IEEE Xplore Digital Library[7]
- Research Gate[8]
- Medium[9]
- Google[10]

Baseado no tema deste trabalho e nas tecnologias utilizadas nele, foram empregadas diversas *strings* na pesquisa. Para cada termo definido também foram utilizados sinônimos em inglês. A tabela 1 demonstra alguns termos utilizados na pesquisa para a realização do estudo.

[6] https://dl.acm.org/
[7] https://ieeexplore.ieee.org/Xplore/home.jsp
[8] https://www.researchgate.net/
[9] https://medium.com/
[10] https://www.google.com.br/

Tabela 1 - *Strings* e termos da pesquisa

Termo	Sinônimo	Termo em Inglês
Testes de acessibilidade para páginas *Web*	Validação de acessibilidade para páginas *Web*	Accessibility testing for *Web* pages, Accessibility validation for *Web* pages
Ferramenta para testes de acessibilidade em páginas *Web*	Ferramentas de avaliação de acessibilidade da *Web*	Tool for accessibility testing of *Web* pages, *Web* Accessibility Assessment Tools
Testes automatizados de acessibilidade para páginas *Web*	Tecnologias para automação de testes de acessibilidade *Web*	Automated accessibility testing for *Web* pages, Technologies for automating *Web* accessibility testing

Fonte: O autor (2021)

Devido ao grande número de resultados e ferramentas encontrados, foram definidos critérios de inclusão e exclusão, buscando, assim, um resultado mais assertivo na pesquisa, onde foram considerados critérios de funcionalidade e semelhança tecnológica dessas ferramentas. Foram excluídas ferramentas do tipo *Plugin* de navegador, aplicativos *desktop*, aplicações mobile, ferramentas que contemplam apenas a versão 1.0 do WCAG ou que não contemplem o guia WCAG. Foram incluídas as ferramentas que mais apareceram na pesquisa e também que tenham sido citadas com mais frequência nos trabalhos relacionados.

3.2 FERRAMENTAS DISPONÍVEIS E TRABALHOS RELACIONADOS

3.2.1 Empirical Studies on *Web* Accessibility of Educational *Web*sites: A Systematic Literature Review

Campoverde-Molina et al. (2020) no seu trabalho intitulado Empirical Studies on *Web* Accessibility of Educational *Web*sites: A Systematic Literature Review, procura através de uma revisão sistemática da literatura analisar os métodos empíricos para validação de acessibilidade em páginas de sites educacionais e os possíveis erros nestas páginas.

Foram selecionados 25 estudos dos quais, foi constatado que 20 usaram ferramentas automatizadas para avaliar a acessibilidade dos sites educacionais, 2 estudos efetuaram uma avaliação de forma manual com usuários reais e outros 3 estudos usaram uma combinação de ferramentas automatizadas com usuários reais e especialistas na área.

Para atingir os objetivos do trabalho os autores elaboraram perguntas que deveriam ser respondidas pela pesquisa, uma das perguntas consistia em: “Qual o tipo de ferramenta online ou serviços que auxiliavam os usuários reais e especialistas na avaliação da acessibilidade?” Como resposta os autores obtiveram que 23 estudos usaram ferramentas automatizadas para efetuar a validação da acessibilidade nos sites alvos, sendo elas:

- AChecker
- Accessibility wizard
- Accessibility valet
- Accessibility colour wheel
- aXe
- Bobby
- Colour contrast analyser
- CynthiaSays
- Etre accessibility check
- EvalAccess
- EIII page checker
- Functional accessibility evaluator
- Fujitsu *Web* accessibility inspector
- FAE, HiSoftware compliance sheriff
- Magenta
- Ocawa
- Siteimprove
- TAW
- TENON
- Total validator
- WAVE, *Web*Acc checker
- *Web*page analyzer
- W3C markup validation service
- W3C CSS validation service

Os autores do trabalho então efetuaram uma contagem das ferramentas citadas mais de uma vez nos estudos selecionados e que foram utilizadas na avaliação de acessibilidade,

como resultado foi obtido o gráfico demonstrado na Figura 9 – Número de ferramentas manuais e automatizadas.

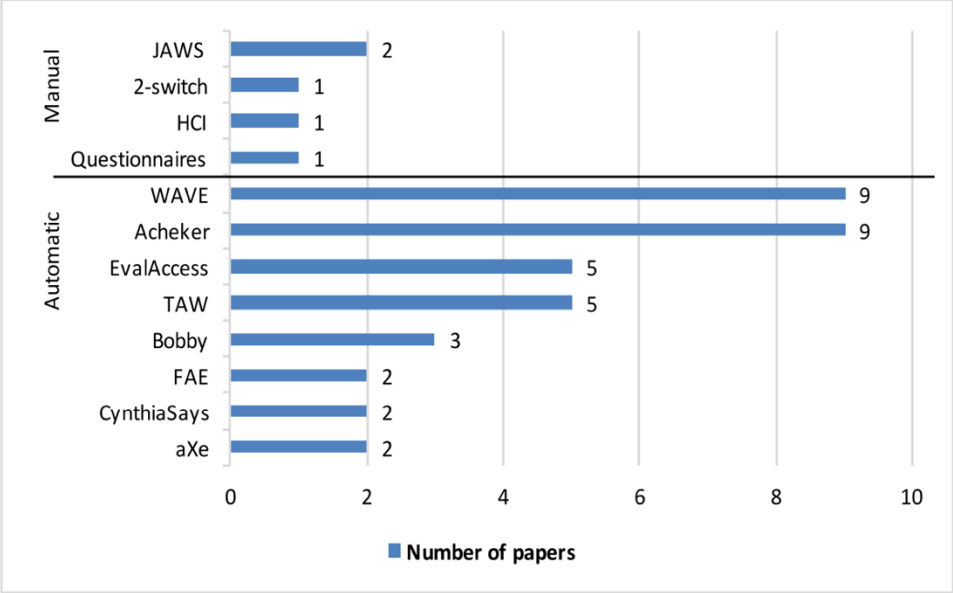


Figura 7 - Número de ferramentas manuais e automatizadas (Campoverde-Molina *et al.* 2020)

Através do gráfico é possível concluir que as quatro ferramentas automatizadas mais utilizadas nos estudos selecionados foram WAVE, Achecker, EvalAces e TAW. Sendo assim, estas ferramentas serão consideradas neste estudo junto a outras ferramentas que mais aparecem na pesquisa, ou serem mais citadas em trabalhos correlatos e que se encaixem nos critérios de inclusão.

3.2.2 Achecker

Achecker[11] é uma ferramenta *online* e gratuita utilizada por usuários e pesquisadores para verificar a acessibilidade de páginas *Web*. A ferramenta Achecker é a ferramenta mais utilizada para efetuar validações de acessibilidade baseadas no guia WCAG 2.0 (ALISMAIL; CHIPIDZA, 2021). Ela é capaz de detectar três tipos de problemas, sendo eles problemas conhecidos, ou seja, problemas de acessibilidade detectados com exatidão e são inegavelmente problemas concretos de acessibilidade; possíveis problemas, sendo estes problemas de acessibilidade que provavelmente existem, porém, necessitam de uma avaliação humana para

[11] https://achecker.achecks.ca/checker/index.php

confirmar; potenciais problemas, neste tipo de problema a detecção pela ferramenta não pode ser efetuada com exatidão, sendo totalmente necessário o julgamento de uma pessoa (KUMARI; VERMA, 2020).

Segundo Rodriguez, Y. S *et al*. (2020) a ferramenta Achecker é de código aberto e pertence à universidade OCAD que fica em Ontário no Canadá. Para utilizar a ferramenta é necessário que o usuário informe o endereço do site a ser validado, podendo também enviar o arquivo HTML da página para análise, ou então, o código HTML da página pode ser inserido no campo disponível no site para ser analisado.

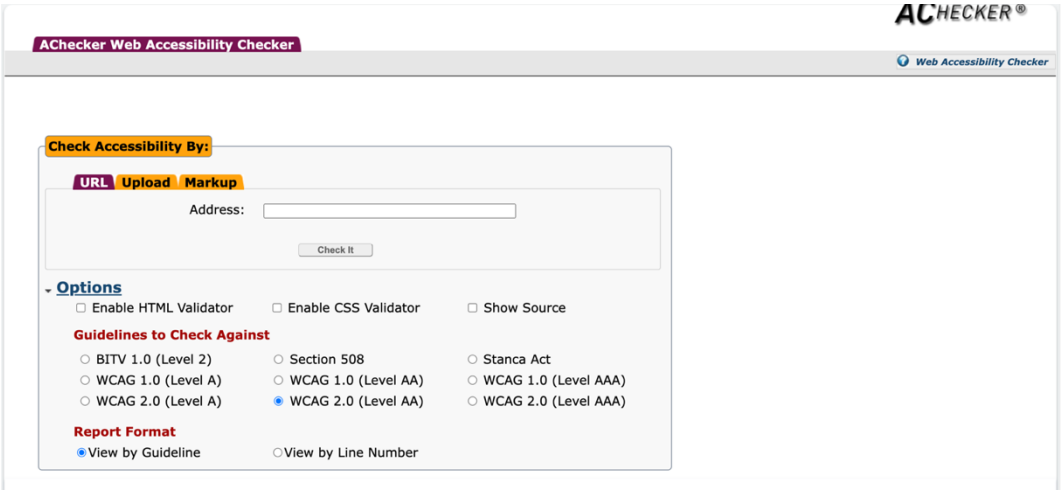


Figura 8 - Página inicial da ferramenta Achecker (IDRC, 2021)

A ferramenta possibilita ao usuário selecionar qual guia deseja usar como referência para a execução automatizada da validação dos problemas de acessibilidade. Após informar a versão do guia a ser utilizado e ao clicar no botão *check it*, então é apresentado um relatório consolidando todos os problemas de acessibilidade encontrados na página alvo da avaliação (IDRC, 2021).

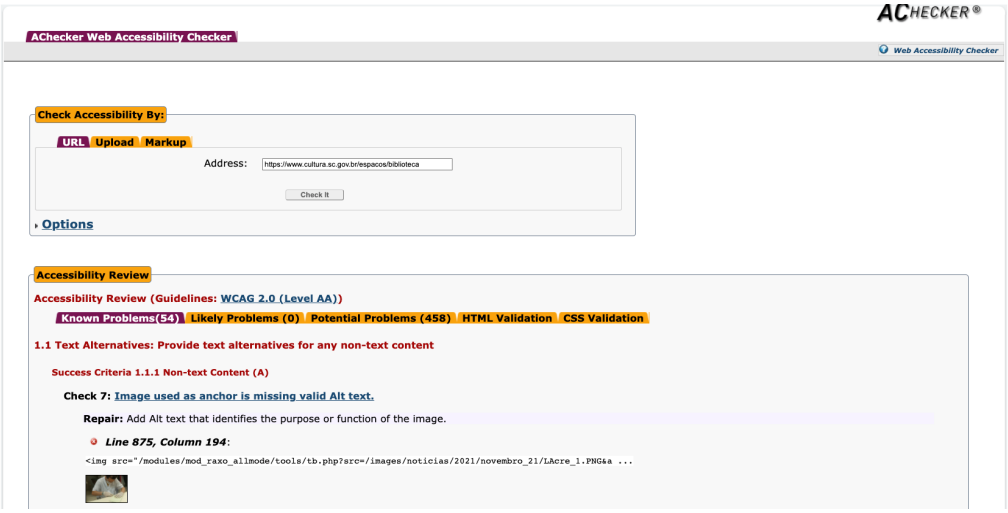


Figura 9 - Relatório gerado após a execução da ferramenta (IDRC, 2021)

3.2.3 accessMonitor

Segundo Oliveira et al. (2018), o accessMonitor[12] é um validador automático de acessibilidade, ele foi desenvolvido e é mantido pela Agência para a Sociedade do Conhecimento, do ministério da educação de Portugal. Ele permite verificar automaticamente violações de acessibilidade para uma determinada página *Web* baseado nas diretrizes do WCAG 2.1, a análise pode ser executada informando o endereço da página, fazendo o *upload* do arquivo com o código HTML ou inserindo o código HTML diretamente no editor de código disponível no site (IFRS, 2021).

[12] https://accessmonitor.acessibilidade.gov.pt/

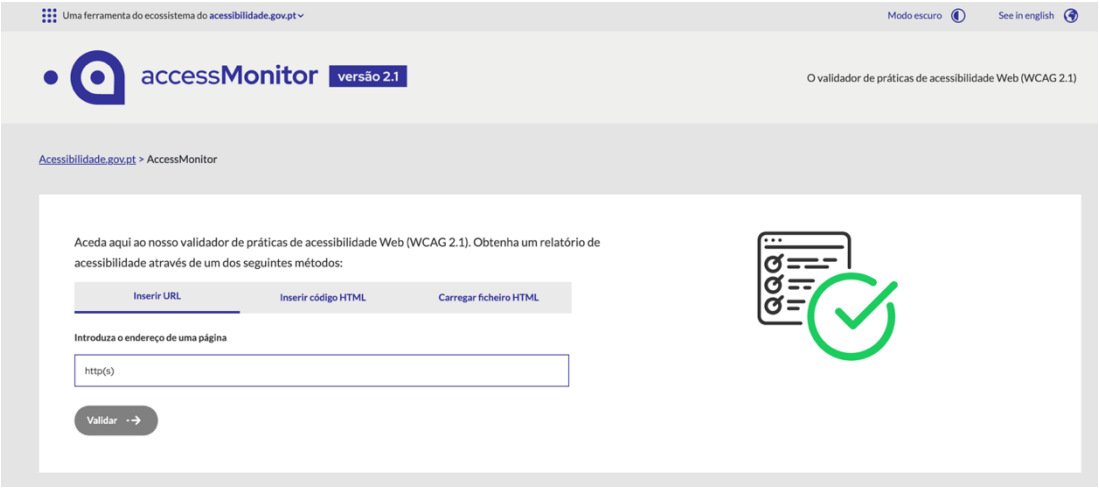


Figura 10 - Página inicial da ferramenta acessMonitor (AccessMonitor, 2021)

Após informar o endereço do site alvo e clicar em Validar, o accessMonitor gera um relatório com informações como pontuação atingida, o número de elementos verificados, o tamanho da página avaliada, um resumo das práticas encontradas em relação ao WCAG 2.1. Por fim é exibida uma ficha de avaliação detalhando as práticas encontradas e a sua relação com as diretrizes do WCAG 2.1.

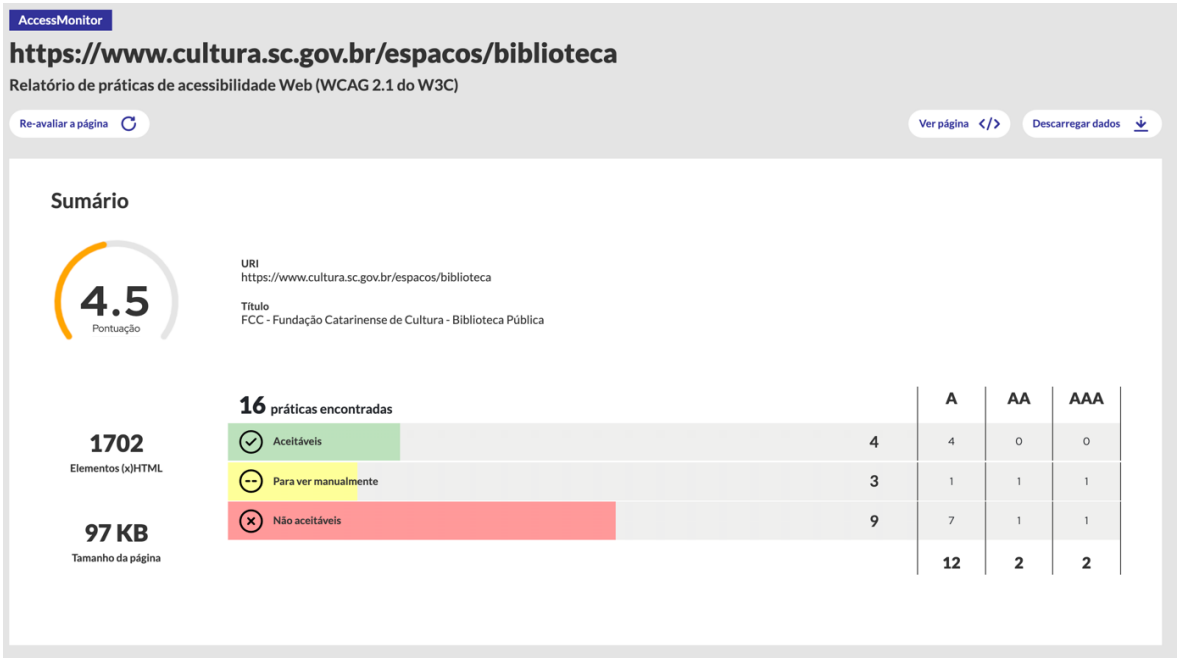


Figura 11 - Sumário com resultado da avaliação (AccessMonitor, 2021)

3.2.4 Axe DevTools

Axe DevTools[13] é uma ferramenta desenvolvida pela empresa Deque Systems, ela foi baseada na extensão de navegador gratuita chamada Axe - *Web* Accessibility Testing. O seu desenvolvimento teve como objetivo permitir que equipes de desenvolvimento atinjam a acessibilidade nas suas páginas desenvolvidas, utilizando técnicas de *machine learning* e outras tecnologias. A ferramenta consegue identificar entre 76 a 84 por cento de falhas de acessibilidade em uma página *Web*, representando uma das maiores taxas de detecção entre as ferramentas disponíveis no mercado (BATEMAN, 2021).

Atualmente a ferramenta possui três versões, sendo uma gratuita, porém limitada a uma extensão de navegador; uma segunda versão chamada Pro com custo de $40.00 dólares mensais; e uma versão completa chamada Enterprise, cujo valor pode ser obtido entrando em contato com a empresa. Na versão mais completa da ferramenta são oferecidos os seguintes recursos (DEQUE, 2021):

- Testes completamente automatizados
- Central de ajuda detalhada
- Guia inteligente dos testes
- Teste individual de componentes
- Exportar e compartilhar defeitos detectados
- Salvar o resultado dos testes
- Definir a regras de acessibilidade que serão testadas
- Integração com ferramentas de CI/CD
- Customização do relatório de execução
- Testes em *Android*[14] e *IOS*[15]
- Suporte *enterprise*

[13] https://www.deque.com/axe/devtools/
[14] https://www.android.com/intl/pt-BR_br/
[15] https://www.apple.com/br/

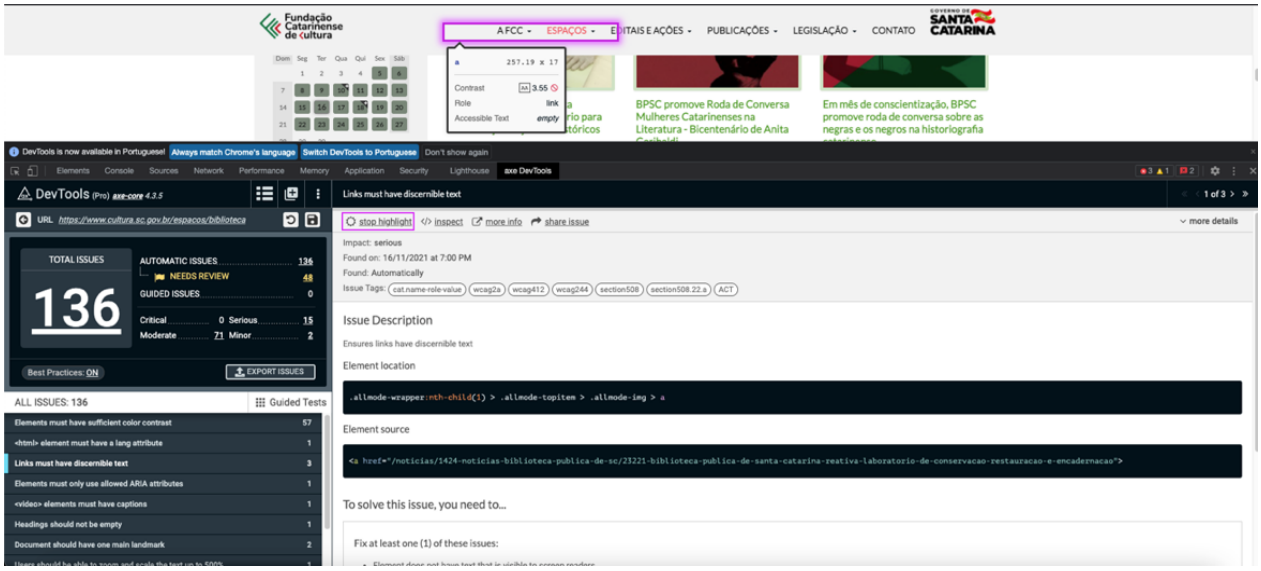


Figura 12 - Execução de testes com a ferramenta Axe DevTools (Deque, 2021)

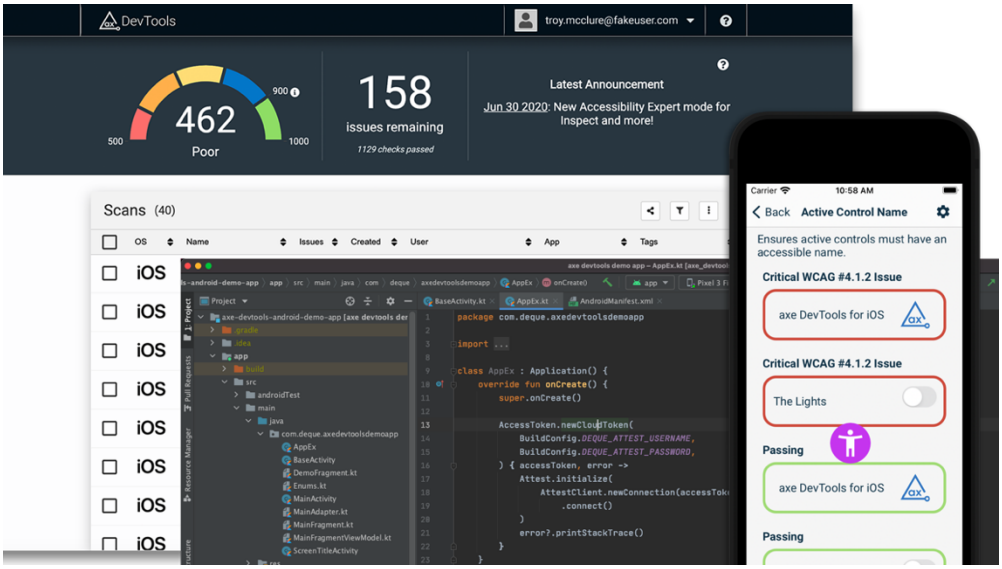


Figura 13 - Execução de testes com a ferramenta Axe DevTools em dispositivo mobile (Deque, 2021)

3.2.5 Cynthia Says

A Cynthia Says[16] é uma ferramenta gratuita que faz parte de um produto chamado Compliance Sheriff, criado pela empresa Appgate Solution no ano de 2003 (HENRY, 2021). Ela se destina ao uso pessoal, e tem como objetivo informar a comunidade sobre como construir uma *Web* acessível. Ela identifica falhas de acessibilidade seguindo os padrões estabelecidos pela Section 508 e pelo guia WCAG 2.0 (CRYPTZONE NORTH AMERICA, 2021).

[16] http://www.cynthiasays.com/

Segundo Tanowitz e Morz (2013) ela permite a usuários testar apenas páginas individuais através do seu site, ao fim da execução a ferramenta fornece *feedback* sobre os problemas encontrados em um formato de relatório que é claro e fácil de entender. Os relatórios identificam os problemas específicos no código para que desenvolvedores *Web* possam realizar as correções necessárias para garantir a acessibilidade.

A empresa Appgate Solution afirma que a Cynthia Says é apenas uma parte do pacote Compliance Sheriff que fornece uma solução mais ampla. Ela permite de forma automatizada e completa o monitoramento, auditoria e testes para garantir que todo conteúdo da *Web* permaneça em conformidade com os padrões mais recentes de acessibilidade (CRYPTZONE NORTH AMERICA, 2021).

Figura 14 - Relatório da execução de testes da ferramenta Cynthia Says (Cryptzone North America, 2021)

3.2.6 Google Lighthouse

Segundo Harvey (2021) o Lighthouse[17] foi criado e é mantido pela Google, ele é uma ferramenta projetada para testes de performance e acessibilidade tendo sido desenvolvido no

[17] https://github.com/GoogleChrome/lighthouse

navegador Google Chrome. É uma ferramenta gratuita e de código aberto que permite testar a performance e acessibilidade de páginas *Web* sob inúmeros aspectos. O Lighthouse possui um cliente de linha de comando chamado Lighthouse CI ou abreviado LHCI, este cliente de linha de comando permite que a ferramenta seja executada em ambiente de CI/CD.

O Lighthouse pode ser executado de forma programática utilizando a linguagem de programação JavaScript[18], permitindo assim, que a execução da ferramenta seja customizada. Isso possibilita a escolha de quais funcionalidades serão testadas, a coleta e análise dos resultados da execução e também a personalização das configurações para se adequar melhor as páginas *Web* que serão testadas. Embora executar a ferramenta em modo programático traga muitas opções, nem sempre pode ser a melhor escolha de execução, sendo encorajado uma análise da melhor forma de executar a ferramenta em relação à página a ser testada (BOWMAN, 2020).

O Google Lighthouse CI possui dois componentes principais, um cliente e um servidor. O servidor simplifica a forma como são gerados os resultados dos testes e a sua disponibilização via navegador em tempo real, isto torna a estrutura mais simples, pois o desenvolvedor não precisa se preocupar com o componente de servidor (HARVEY, 2021).

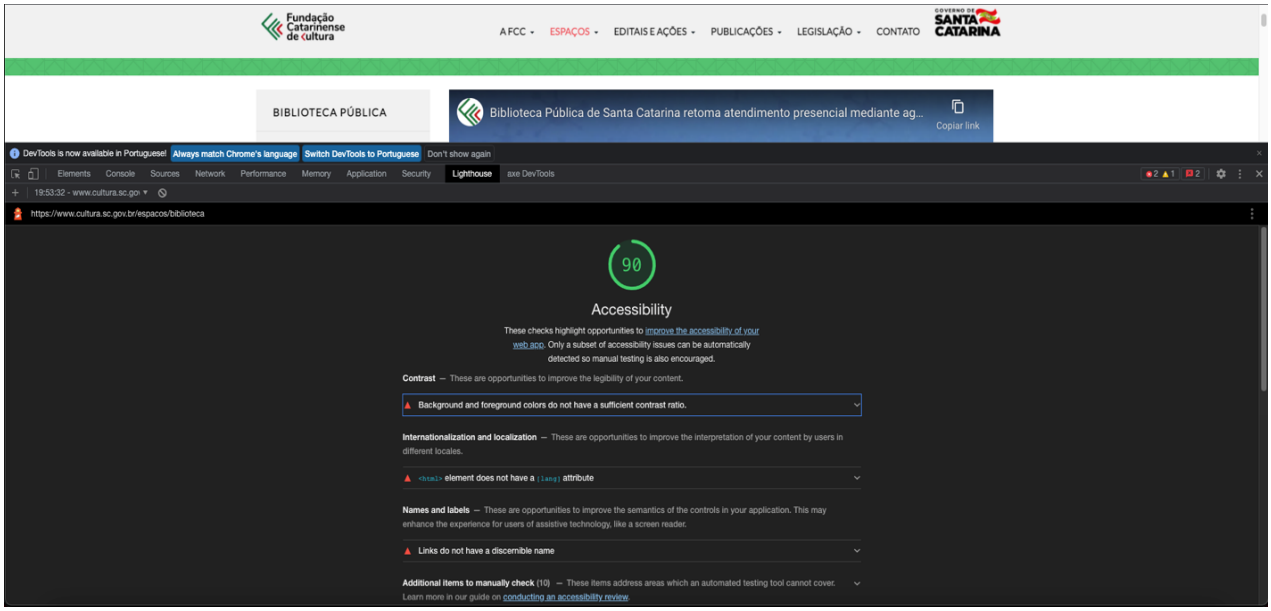


Figura 15 - Execução da ferramenta Lighthouse no navegador Google Chrome (O autor, 2021)

[18] https://developer.mozilla.org/pt-BR/docs/*Web*/JavaScript

3.2.7 Tenon.io

Tenon[19] é uma empresa fundada em 2014 que desenvolve ferramentas especializadas para testes de acessibilidade em páginas *Web*, tendo como seu principal produto a Test API, um serviço REST que permite efetuar testes automatizados de acessibilidade em páginas *Web* através de requisições HTTP. Além da API, no site da empresa é possível testar uma página *Web* ou analisar o código HTML em um campo específico para isto (TENON, 2021).

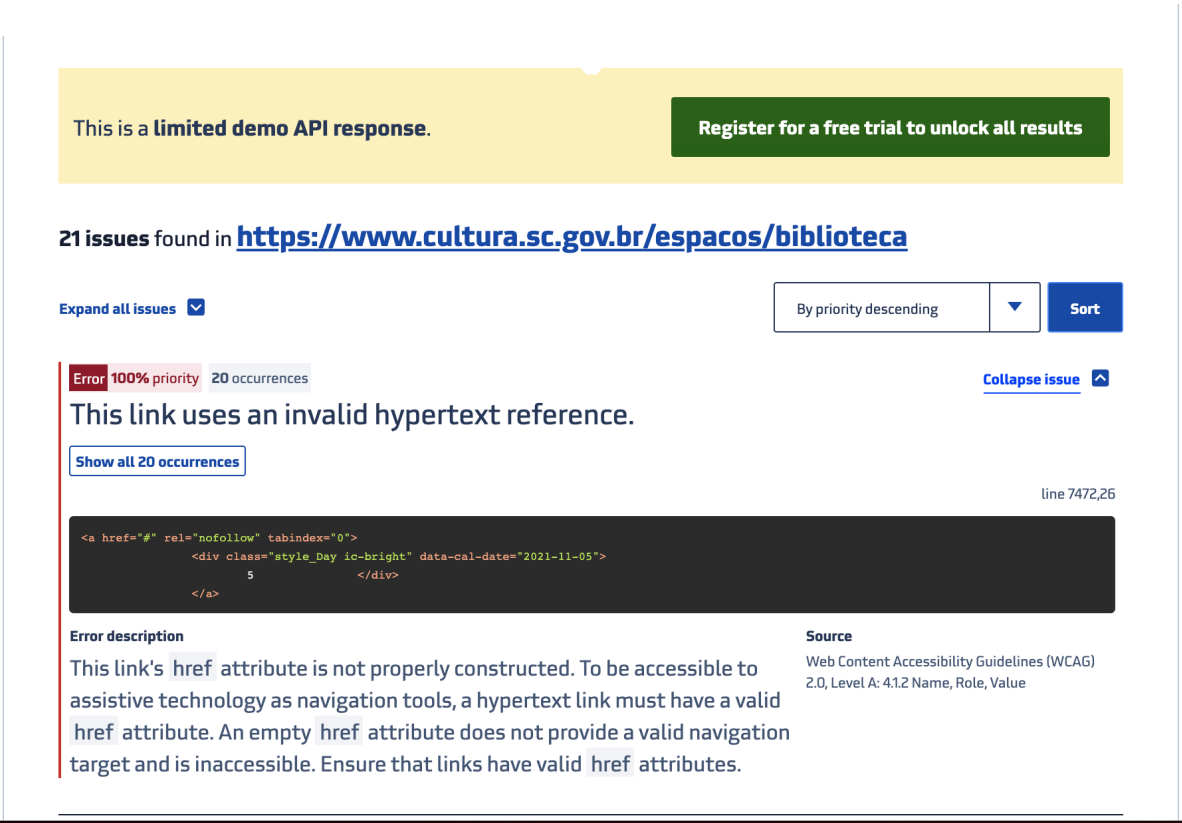


Figura 16 - Execução da versão *Web* da ferramenta Tenon (Tenon, 2021)

Entre alguns recursos disponíveis na ferramenta estão integrações com ferramentas de desenvolvimento como IDEs, Jira[20], Git[21] e outras, além de validação de problemas de acessibilidade em acordo com o guia WCAG 2.1. Ademais, há fácil integração com variados

[19] https://tenon.io/
[20] https://www.atlassian.com/software/jira
[21] https://git-scm.com/

ambientes de desenvolvimento, pois por se tratar de uma API, a ferramenta é facilmente integrada a várias linguagens de programação e ambientes de CI/CD. A ferramenta possui um custo que vai de $82.00 dólares por mês limitado a 3000 requisições na API, até um custo de $687.00 dólares por mês limitado a 30.000 requisições na API (TENON, 2021).

3.2.8 WAVE *Web* Accessibility Evaluation Tool

A *Web* Accessibility Evaluation Tool[22] (WAVE) foi originalmente criada pelo Dr. Lean Kasday da Universidade Temple, e desde então é mantida e desenvolvida pela empresa *Web*AIM (UNIVERSITY OF MINNESOTA, 2021).

A WAVE é uma ferramenta para automação de testes de acessibilidade em páginas *Web*, ela detecta erros de acessibilidade e outros problemas que possam existir em uma página *Web*. Com o uso da ferramenta, usuários, *designers* e desenvolvedores conseguem identificar problemas de acessibilidade assim que as páginas são desenvolvidas (SHARMA, 2021).

Ainda segundo Sharma (2021), a ferramenta pode ser utilizada diretamente pelo site informando o endereço da página que será testada. Também é possível utilizar extensões para os navegadores Firefox e Google Chrome. A ferramenta WAVE utiliza o WCAG 2.0 como guia de referência. Embora seja uma ferramenta que auxilia na detecção de erros de acessibilidade, ela não substitui uma checagem completa das páginas testadas utilizando outras ferramentas e técnicas.

Conforme Pope Tech (2021), a WAVE também possui uma API que permite executar os testes de acessibilidade integrando com ferramentas de CI/CD, permitindo também a implementação de relatórios personalizados e integração com outras ferramentas. A ferramenta na sua versão API possui uma licença anual, o custo desta licença vai de $4,000.00 dólares no plano mais básico até $12,000.00 dólares no plano mais completo chamado *Enterprise*. A versão API da ferramenta possui como recursos:

- Possibilidade de criação de *scripts* para interação com elementos das páginas alvo dos testes.
- Teste de páginas protegidas por senha e páginas intranet, suportando as formas mais comuns de autenticação.

[22] https://wave.*Web*aim.org/

- Salva o DOM da página após ele ser carregado. Ele pode ser utilizado em conjunto com valores de seletores como CSS Selector ou Xpath para associar erros específicos a elementos específicos da página.
- Efetua captura de telas.
- Permite definir o tamanho da tela, assim possibilitando simular dispositivos móveis.
- Integração com serviços como browlerss.io possibilitando testes off-line das páginas.

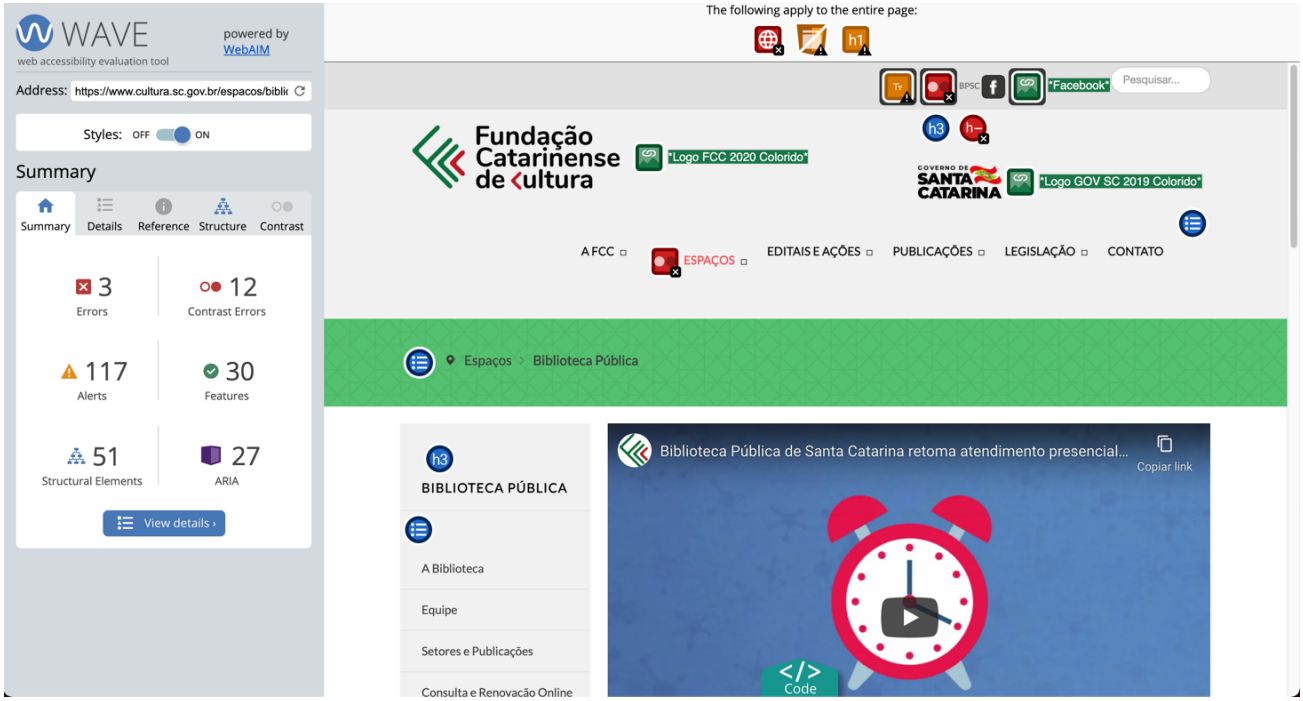


Figura 17 - Uso da ferramenta WAVE através do seu site (Wave, 2021)

3.2.9 TAW – Test de Acessibilidade *Web*

Desenvolvida pelo CIT[23] (*Centro de Technologico de la Informacion y la Comunicación)* é uma ferramenta gratuita para testar acessibilidade de sites *Web*, ela está disponível em versões online, aplicativos de desktop e extensão para o navegador Firefox (ISMAIL; KUPPUSAMY, 2019).

[23] https://www.ctic.uni.edu.pe/

Segundo Roig-Vila *et al.* (2014) a ferramenta TAW[24] permite que sejam escolhidos entre os três níveis de acessibilidade definidos no WCAG 2.0, nível A, AA ou AAA, qual deles será usado para efetuar a avaliação da página *Web*. Além disso, a ferramenta classifica as falhas de acessibilidade encontradas em três tipos, o primeiro tipo chamado "*Problems"*, problemas detectados automaticamente e não necessitam de uma checagem manual, se trata de um problema que sem qualquer margem de dúvidas infringe alguma diretriz do WCAG 2.0. O segundo tipo é chamado "*Warnings*", problemas deste tipo significam erros de acessibilidade que foram detectados, porém, necessitam de uma revisão manual para confirmação. O último tipo é chamado "*Not reviewed*", problemas deste tipo necessitam obrigatoriamente de uma revisão manual para serem confirmados como problemas de acessibilidade.

Figura 18 - Página inicial da ferramenta TAW (Taw, 2021)

Após informar o endereço da página a ser testada, a ferramenta gera um relatório composto de um sumário com um resumo dos tipos e quantidades de problemas de acessibilidade detectados. Além disso, ela traz uma lista detalhada com cada uma das diretrizes do guia WCAG 2.0, informando quais diretrizes foram violadas, quais necessitam de uma validação parcialmente manual, quais necessitam de uma avaliação totalmente manual e por último quais não foram violadas pelo site alvo do teste (TAW, 2021).

[24] https://www.tawdis.net/

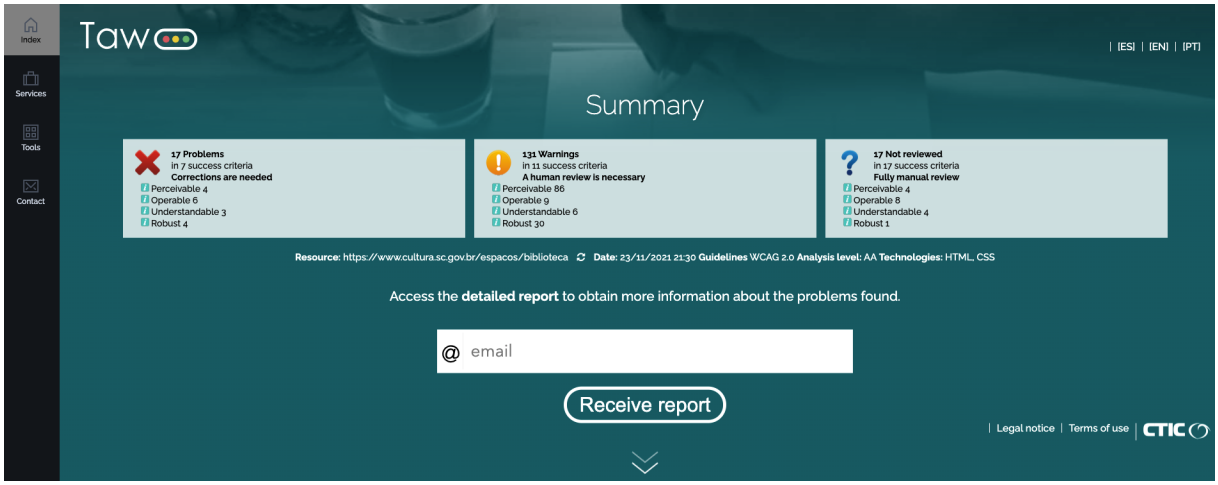


Figura 19 - Sumário do teste executado pela ferramenta TAW (Taw, 2021)

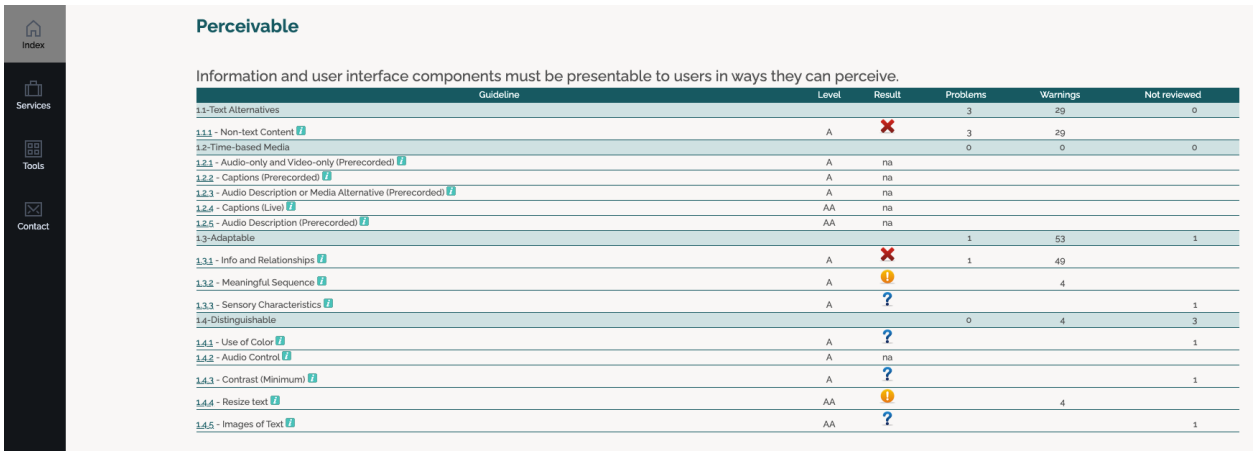


Figura 20 - Resultado detalhado da execução da ferramenta TAW (Taw, 2021)

3.3 COMPARAÇÃO DAS FERRAMENTAS

Nesta parte do trabalho é efetuada uma comparação entre as ferramentas apresentadas na pesquisa da sessão anterior e a solução proposta neste trabalho. Seu objetivo é comparar as características das ferramentas pesquisadas, assim justificando a necessidade e o diferencial da solução que será desenvolvida.

Para realizar a comparação foram considerados os aspectos mais importantes das ferramentas em relação ao objetivo de validar automaticamente problemas de acessibilidade em páginas *Web*. Também foram considerados aspectos que sejam relacionados aos requisitos funcionas e não funcionais definidos para a solução desenvolvida neste trabalho. A tabela abaixo detalha a comparação entre as ferramentas, ela foi desenvolvida pelo autor concatenando

todas as características das ferramentas estudadas anteriormente, além disto, também são demonstradas e comparadas as características da solução desenvolvida neste trabalho.

Tabela 2 - Comparação das ferramentas

Ferramenta	Custo anual ($)	Código aberto	CI/CD	*Scriptable*	Possui cliente	Apenas Online	Aceita Plugins
Achecker	Gratuito	Sim	Não	Não	Não	Sim	Não
accessMonitor	Gratuito	Sim	Não	Não	Não	Sim	Não
Axe Devtools	$480.00	Não	Sim	Sim	Sim	Não	Sim
Cynthia Says	Gratuito	Não	Não	Não	Não	Sim	Não
Lighthouse	Gratuito	Sim	Sim	Sim	Sim	Não	Não
Taw	Gratuito	Não	Não	Não	Não	Sim	Não
Tenon	$984.00 a $8,244.00	Não	Sim	Sim	Sim	Não	Não
Wave	$4,000.00 a $ 12,000,00	Não	Sim	Sim	Não	Não	Não
Solução Desenvolvida	Gratuito	Sim	Sim	Sim	Sim	Não	Sim

Fonte: O autor (2021)

- **Custo anual**: A maioria das ferramentas pesquisadas são gratuitas, porém algumas têm custo anual, sendo que entre as ferramentas pagas, duas possuem uma anuidade significativa como Tenon e Wave. Isto pode tornar o uso dessas ferramentas inviável dependendo dos recursos da companhia ou pessoas que irão utilizá-las.
- **Código aberto**: As ferramentas Achecker, AccessMonitor e Lighthouse possuem o seu código aberto, assim permitindo que outros desenvolvedores possam corrigir possíveis problemas nas ferramentas e também adicionar novos recursos a elas.
- **CI/CD**: As ferramentas Axe Devtools, Lighthouse, Tenon e Wave permitem a sua utilização em processos de integração e entrega contínuas. Essas ferramentas podem ser executadas durante o desenvolvimento das páginas *Web*, dando, assim, a possibilidade de as páginas serem validadas quanto a problemas de acessibilidade ainda no início do seu desenvolvimento.
- ***Scriptable***: No caso das ferramentas Axe Devtools, Lighthouse, Tenon e Wave, elas podem ser estendidas ou automatizadas por *scripts*, o que possibilita que estas ferramentas sejam utilizadas através de linguagens de programação, permitindo, assim, uma maior personalização no seu uso e adaptação ao meio de desenvolvimento ao qual a ferramenta está inserida.

- **Possui cliente**: Apenas as ferramentas Axe Devtools, Lighthouse e Tenon possuem clientes para execução da ferramenta via linha de comando. Isto significa a possibilidade da inserção dessas ferramentas em ambientes que executam programas via comandos em terminais de texto, os quais não necessitam de uma *interface* gráfica para serem executados, possibilitando que estas ferramentas sejam executadas de uma forma mais personalizada.
- **Apenas online**: Ferramentas executadas apenas *online* não permitem outro modo de execução, sendo esse um fator limitante, pois para utilizá-las é necessário acesso à *Internet* e um navegador *Web*. Também é necessário que as páginas que serão validadas estejam hospedadas em algum servidor *Web*, impedindo que os testes sejam executados no início do desenvolvimento dessas páginas. As ferramentas Achecker, accessMonitor, Cynthia Says e Taw possuem essa característica que pode limitar o seu uso.
- **Aceita plugins:** Apenas a ferramenta Axe Devtools permite o uso de *plugins*, assim estendendo as funcionalidades da ferramenta e permitindo uma maior personalização no seu uso.

Após finalizar a comparação entre as ferramentas pesquisadas, fica evidente que há espaço no mercado para uma ferramenta que seja gratuita e auxilie desenvolvedores e criadores de conteúdo no desenvolvimento de páginas *Web* mais acessíveis. Embora a ferramenta Lighthouse preencha a maioria dos requisitos, sendo eles, ser uma ferramenta gratuita, de código aberto, automatizada via *script*, que pode ser executada em processo de CI/CD e que possui um cliente para execução via linha de comando, ela se limita quanto à questão de conhecimento técnico necessário para a sua execução de forma automatizada, pois exige conhecimento em programação. Além disso, o Lighthouse não é extensível e personalizável com adição de *plugins* o que acaba sendo um fator limitante. A ferramenta proposta neste trabalho conta com os recursos de todas as ferramentas pesquisadas anteriormente, sejam elas pagas ou gratuitas. Ela também é gratuita, de código aberto, tendo como diferencial a possibilidade de adição de *plugins,* considerando as suas funcionalidades, o uso de tecnologias atuais e amplamente usadas no mercado, uma fácil execução e a possibilidade de personalização e melhoria pela comunidade ou os seus usuários. Todos esses aspectos serão demonstrados com mais detalhes no próximo capítulo.

4 A FERRAMENTA PROPOSTA

Neste capítulo são apresentados o levantamento de requisitos, a arquitetura da solução, as tecnologias utilizadas no seu desenvolvimento e a implementação da ferramenta.

4.1. LEVANTAMENTO REQUISITOS

O levantamento de requisitos advém da análise de outras ferramentas existentes e a documentação delas. Os requisitos funcionais foram elaborados partindo do principio de quais são os recursos mínimos necessários para a execução de testes automatizados de acessibilidade para páginas Web. Os requisitos não funcionais são baseados no conhecimento do autor sobre tais ferramentas.

4.1.1 Requisitos Funcionais

RF-1: Definir endereço das páginas: O usuário poderá definir o endereço das páginas que serão validadas pela ferramenta.

RF-2: Seleção do guia de referência: O usuário poderá selecionar quais guias de referência deseja utilizar nos testes automatizados.

RF-3: Adicionar *plug-ins*: O usuário poderá adicionar puglins na ferramenta possibilitando a extensão das suas funcionalidades.

RF-4: Gerar relatório: O usuário poderá ao fim da execução da ferramenta visualizar um relatório da execução.

4.1.2 Requisitos Não Funcionais

RNF-1: A ferramenta deve ser executa utilizando um ambiente com o Node.js na versão 16.13 ou superior e o NPM na versão 8.1 ou superior.

RNF-2: Será possível a execução da ferramenta em ambiente de CI/CD.

RNF-3: O código da ferramenta deverá ser aberto e disponível em algum repositório de códigos na internet.

4.2 ARQUITETURA DA FERRAMENTA

O usuário, através de uma *interface* de linha de comando[25], interage com a ferramenta informando o(s) endereço(s) ou a localização do(s) arquivo(s) HTML das páginas que deseja submeter a validação de acessibilidade baseadas nas diretrizes do guia WCAG ou outros. Após

[25] https://www.hostinger.com.br/tutoriais/o-que-e-cli

informar tais dados e executar o comando, o *test runner* Cypress[26] executa o *script* de validação, que por sua vez irá executar uma instância de um navegador *Web*, onde serão carregadas as páginas que terão a sua estrutura HTML avaliada em relação às diretrizes estabelecidas no guia selecionado. Ao finalizar a execução, ainda na *interface* de linha de comando, são retornados os dados demonstrando se alguma diretriz foi violada, um identificador da violação, a quantidade de violações, a criticidade da violação, uma breve descrição, o número de nodos[27] afetados e também mais detalhes sobre cada violação. A Figura 21 – Arquitetura geral da solução, demonstra uma visão geral da arquitetura descrita anteriormente.

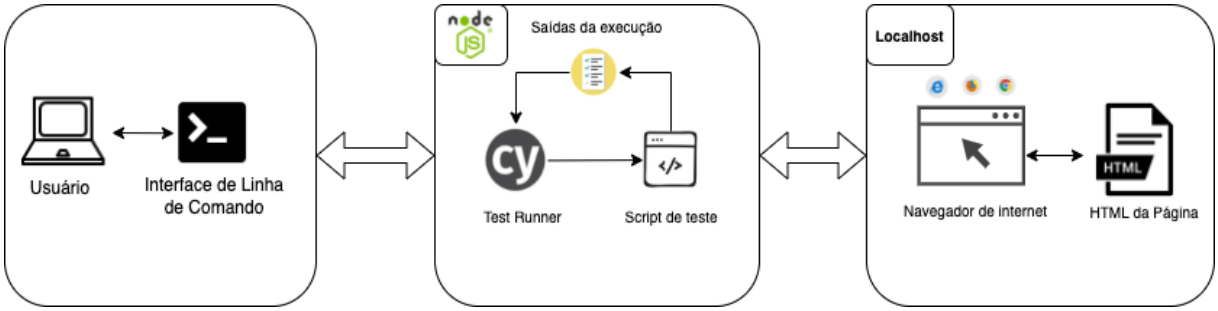


Figura 21 - Arquitetura geral da solução (O Autor, 2021)

4.3 TECNOLOGIAS UTILIZADAS

4.3.1 JavaScript

Segundo Flanagan (2013), JavaScript é uma linguagem de alto nível, ela é utilizada na maioria dos sites mais modernos, nos navegadores mais atuais, em console de jogos, *tablets* e *smartphones*, o que a torna a linguagem de programação mais onipresente da história. Ela é uma linguagem dinâmica, não tipada, interpretada, podendo ser usada em programação funcional e orientada a objetos.

Atualmente a linguagem não se limita apenas à validação de dados simples como era nos seus primórdios, ela interage com praticamente todas as funcionalidades disponíveis em um navegador. Ela é conhecida atualmente como uma linguagem de programação completa capaz de implementar funções complexas, efetuar cálculos, e outros. A linguagem tornou-se uma parte importante da *Web*, incluída em navegadores *Web* comuns até navegadores

[26] https://www.cypress.io/
[27] http://www.w3big.com/pt/htmldom/htmldom-nodes.html

alternativos, como navegadores executados em telefones celulares projetados para pessoas com deficiência. Até mesmo empresas que possuem as suas próprias linguagens adotaram o uso do JavaScript, a exemplo da Microsoft que criou uma implementação de JavaScript e o colocou no seu navegador, o *Internet* Explorer (FRISBIE, 2019).

O que tornou a linguagem JavaScript tão popular e usada, algo que muitas linguagens orientadas a objetos não conseguiram, não é apenas que a linguagem é simples e poderosa, mas ela conseguiu encontrar um nicho sendo um motor *Web* tanto como cliente quanto como servidor (BENSON JUNIOR, 1999).

Fica evidente que o JavaScript é uma linguagem simples, porém poderosa, pois abre um grande leque de possibilidades. Para a solução que será desenvolvida neste trabalho ela se demonstra a linguagem ideal, pois vai ao encontro das características desejadas quanto ao uso de tecnologias atuais e amplamente utilizadas no mercado e quanto a simplificação da sintaxe do código que será desenvolvido.

4.3.2 NODE.JS

De acordo com Moraes (2021), o Node.js[28] foi apresentado pelo seu criador Ryan Dahl no ano de 2009 na JSConf Europa[29]. Sendo uma plataforma poderosa, ela possibilita a construção fácil e rápida de aplicações de rede escaláveis. Ela utiliza a *engine* JavaScript de código aberto e alta performance do navegador Google Chrome, conhecida como motor V8[30], essa *engine* é escrita em C++ com a adição de alguns módulos, como a libuv[31].

O Node.js é uma tecnologia inovadora, sua arquitetura é totalmente orientada ao que é chamado de *non-blocking thread* (não bloqueante). Aplicações que trabalham com a realização de muitas operações de entrada e saída em processamento de arquivos, encontram no Node.js uma arquitetura que garante uma boa performance em relação ao consumo de memória, pois ela utiliza de uma forma mais eficiente o poder de processamento dos servidores (PEREIRA, 2014). Segundo Moraes (2021), esta arquitetura também serve perfeitamente para solucionar problemas de tráfego intenso de aplicações e redes em tempo real, que geralmente representam os maiores desafios nas aplicações *Web* mais modernas.

---

[28] https://nodejs.org/en/
[29] https://www.youtube.com/watch?v=ztspvPYybIY
[30] https://v8.dev/
[31] https://libuv.org/

O Node.js é de alto nível e altamente escalável, pois para programar usando a plataforma não é necessário um grande conhecimento de diversos protocolos de rede e *Internet*, também não é necessário o uso de bibliotecas para acessar os recursos do sistema operacional, uma vez que o Node.js utiliza a linguagem de programação JavaScript e é executado utilizando o motor V8 (PEREIRA, 2014).

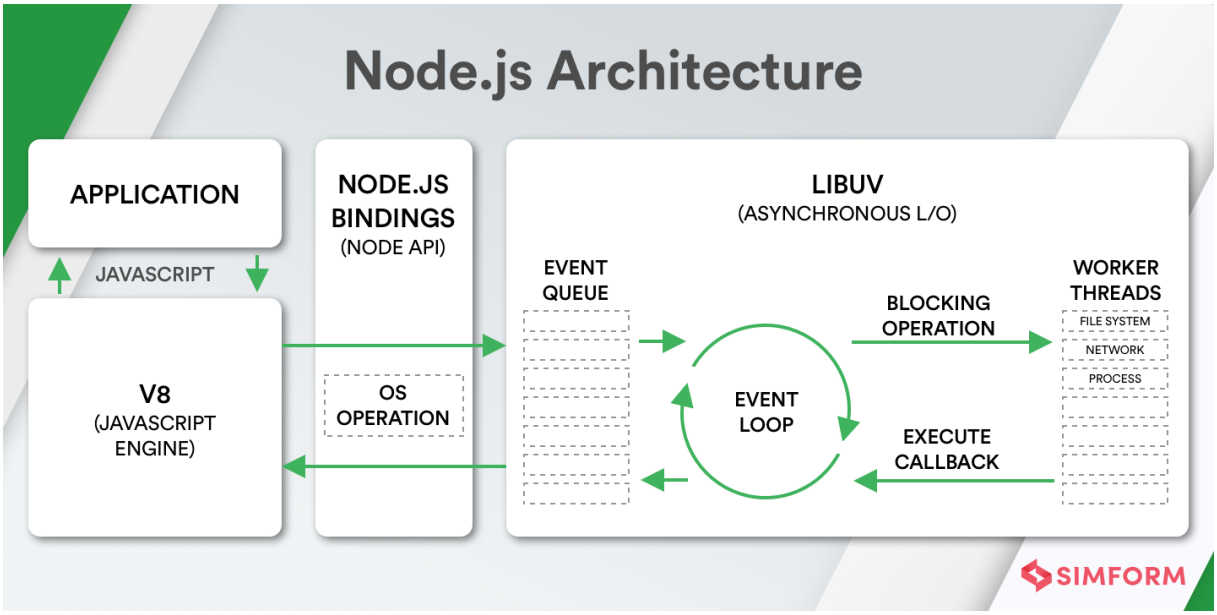


Figura 22 - Arquitetura do node (Kaneriya, 2020)

4.3.2.1 Node Package Manager (NPM)

O NPM[32] é um repositório onde são publicados projetos de código aberto desenvolvidos utilizando Node.js, ela também é uma ferramenta de linha de comando utilizada para interagir com repositórios onde ficam esses projetos de código aberto, auxiliando na instalação do pacote de códigos, no gerenciamento de suas versões e das suas dependências. Muitas aplicações baseadas em Node estão publicadas no NPM, e muitas outras são publicadas todos os dias (OPEN JS FOUNDATION, 2011). Segundo Tal e Maple (2019), no ano de 2019 foi alcançada a marca de um milhão de pacotes registrados no NPM, significando um grande número de projetos de código abertos, disponíveis para os mais variados usos, criados e mantidos por milhares de desenvolvedores, tornando o NPM uma fonte ilimitada de recursos para o desenvolvimento de novos projetos.

[32] https://www.npmjs.com/

Além da grande importância em números e projetos disponíveis, outra característica relevante do NPM é permitir que as dependências dos projetos desenvolvidos em Node sejam gerenciadas de forma simples, pois toda vez que um projeto em Node.js é criado, também é criado um arquivo chamado *package.json,* arquivo que guarda a lista de dependências (Pacotes) utilizados no projeto e a versão de cada uma delas, permitindo que sejam instaladas ou atualizadas com comandos simples, economizando tempo e simplificando o desenvolvimento de aplicações (OPEN JS FOUNDATION, 2011).

4.3.3 Cypress

Cypress é um *framework*[33] para testes *end-to-end*[34] escrito por desenvolvedores para desenvolvedores e analistas de qualidade. Possui um foco em teste de aplicações *Web*, suportando atualmente apenas a linguagem JavaScript para a escrita dos testes (MWAURA, 2021).

Ainda de acordo com Mwaura (2021), o Cypress foi desenvolvido especificamente para times que utilizam a linguagem JavaScript no desenvolvimento dos seus produtos que necessitam, de uma forma simples e rápida, escrever os testes automatizados sem a necessidade de lidar com as complicações e informações intrínsecas, quando se está estruturando um *framework* para testes automatizados.

Ele não é apenas um *framework* amigável para quem está iniciando, é também completo, já que conta com tudo o que é necessário para o início do desenvolvimento de testes automatizados. Não é necessário para o início do desenvolvimento dos testes que os envolvidos possuam conhecimento prévio de configurações de *setup*, processos de asserções e configuração de executores de testes. Para isso, o Cypress conta com um *browser* próprio já configurado, um executor de testes e um *framework* chamado chai[35] já configurado para efetuar as asserções (MWAURA, 2021).

Conforme afirma Khetarpal (2021), o *framework* Cypress possui algumas funcionalidades que facilitam o dia a dia dos desenvolvedores e analistas de testes, além dos diferenciais perante outros *frameworks* para testes automatizados existentes, sendo eles:

[33] https://rockcontent.com/br/blog/framework/
[34] https://blog.cedrotech.com/teste-end-to-end
[35] https://www.chaijs.com/

- **Viagem no tempo**: São gerados *snapshots*[36] do site que está sendo validado durante a execução dos testes, isso permite a visualização da execução de cada comando, possibilitando assim, uma visão clara do que está ocorrendo durante a execução de cada etapa do teste.
- **Esperas automáticas**: Diferente de outros *frameworks* para testes automatizados, no Cypress não é necessário especificar pausas ou limite de tempo para a execução dos comandos ou asserções, isto é efetuado automaticamente pelo próprio *framework*.
- **Controle do tráfego de rede**: Utilizando o Cypress é possível criar uma pequena rede para interceptar todo o tráfego, assim o usuário é capaz de customizar e validar requisições e respostas de API's conforme necessário.
- **Consistente nos resultados**: O Cypress é executado diretamente no navegador, sem a necessidade de *drivers* adicionais ou ferramentas de terceiros, isso garante uma execução mais veloz e consistente, além de testes menos suscetíveis a falhas.

O Cypress ainda conta com um grande benefício por estar inserido no universo da linguagem JavaScript e do Node.JS: a opção da adição de *plugins*. Atualmente, existe uma infinidade de *plugins* com as mais diversas funcionalidades que podem ser adicionados ao *framework* via NPM, isto não apenas estende a funcionalidade do *framework*, como também adiciona novos comandos, opções de personalização e muito mais (HRIC, 2020).

Baseado no exposto, é possível concluir que o uso do *framework* Cypress traz muitos benefícios no desenvolvimento de testes automatizados. Além do descrito anteriormente, o *framework* conta com uma excelente documentação, uma comunidade ativa e é amplamente usado por grandes empresas de tecnologia, como GoDaddy, Siemens, Slido e outras, conforme indica a *Featured Customers* (2021). Sendo assim, ele se demonstra ideal para o desenvolvimento da solução proposta, indo ao encontro dos objetivos citados anteriormente quanto ao uso de tecnologias modernas e extensíveis com *plugins.*

[36] https://www.controle.net/faq/o-que-e-snapshot

4.3.4 Axe-core

Conforme afirma Berry (2020) o axe-core é um mecanismo para testar a acessibilidade em sites e outras páginas HTML, ele é seguro e leve, projetado para poder ser integrado em qualquer ambiente de desenvolvimento e testes, possibilitando a automação dos testes de acessibilidade com os demais testes funcionais.

O mecanismo foi criado com o objetivo de que só haverá uma *Web* acessível se formos capazes de empoderar os desenvolvedores para poderem assumir a responsabilidade de testar a acessibilidade e as boas práticas de código. Nesse sentido, o axe-core reflete como a *Web* é desenvolvida, podendo ser executado em navegadores modernos e ferramentas de testes e sendo integrado a outras etapas de testes e qualidade no processo de desenvolvimento (AXE-CORE, 2021).

Isto significa que o axe-core pode ser usado para validar a estrutura HTML das páginas em busca de falhas de acessibilidade. Para tanto, ele utiliza um sistema baseado em regras, onde é permitido selecionar um conjunto de regras ou guias de referência conforme a necessidade dos testes. Essas regras estão de acordo com os guias WCAG, Section 508 e outros. Cada regra conta com a informação sobre o impacto ao usuário caso ela seja violada, classificado como pequeno, moderado, sério ou crítico. Além disto, cada regra se relaciona diretamente a um ou mais critérios de sucesso definidos no guia WCAG e outros guias. A Tabela 3 – Exemplos de regras do axe-core relacionadas ao guia WCAG em inglês, lista algumas dessas regras validadas pelo axe-core que cobrem uma ou mais versões do guia WCAG (AXE-CORE, 2021).

Tabela 3 – Exemplos de regras do axe-core relacionadas ao guia WCAG em inglês

ID axe-ore	Description	WCAG Version
area-alt	Ensures <area> elements of image maps have alternate text	wcag2a
aria-progressbar-name	Ensures every ARIA progressbar node has an accessible name	wcag2a
aria-toggle-field-name	Ensures every ARIA toggle field has an accessible name	wcag2a
aria-tooltip-name	Ensures every ARIA tooltip node has an accessible name	wcag2a
aria-valid-attr-value	Ensures all ARIA attributes have valid values	wcag2a

aria-valid-attr	Ensures attributes that begin with aria- are valid ARIA attributes	wcag2a
audio-caption	Ensures <audio> elements have captions	wcag2a
blink	Ensures <blink> elements are not used	wcag2a
button-name	Ensures buttons have discernible text	wcag2a
color-contrast	Ensures the contrast between foreground and background colors meets WCAG 2 AA contrast ratio thresholds	wcag2aa
definition-list	Ensures <dl> elements are structured correctly	wcag2a
document-title	Ensures each HTML document contains a non-empty <title> element	wcag2a
duplicate-id-active	Ensures every id attribute value of active elements is unique	wcag2a
duplicate-id-aria	Ensures every id attribute value used in ARIA and in labels is unique	wcag2a
duplicate-id	Ensures every id attribute value is unique	wcag2a

Fonte: Elaborado pelo autor. Adaptado de Axe-Core (2021).

Para o desenvolvimento da solução apresentada neste trabalho, a biblioteca axe-core tem um grande papel, pois é graças a esta biblioteca e o seu algoritmo que a solução é capaz de detectar as violações de acessibilidade de forma automatizada, embora seja possível utilizar outras bibliotecas para tal fim, a biblioteca axe-core se mostrou durante o desenvolvimento do trabalho a que mais se adequa aos objetivos do mesmo, sendo ela uma parte central da solução desenvolvida.

4.3.5 VSCode

O Visual Studio Code [37](VSCode) é um editor de código multiplataforma desenvolvido pela Microsoft, iniciado em 2011 com o nome Mônaco. A princípio, era um editor de código executado apenas no navegador de *Internet*, porém, anos depois, ele foi alterado para

[37] https://code.visualstudio.com/

ser executado como um aplicativo *desktop*, lançado oficialmente em 2015. Desde então ele é desenvolvido de forma contínua trazendo novas funcionalidades e uma melhor estabilidade (PLAINER, 2020).

Conforme Paluca (2021), o VSCode é baseado no *framework* Electron[38], ele utiliza muitas tecnologias que também são usadas na *Web*, isto o torna uma ferramenta fácil de ser executada em múltiplas plataformas. O VSCode vem ganhando uma grande popularidade desde da sua criação, e tem sido atualmente um dos editores de código mais utilizados. A Figura 23 demonstra a popularidade do editor entre os desenvolvedores.

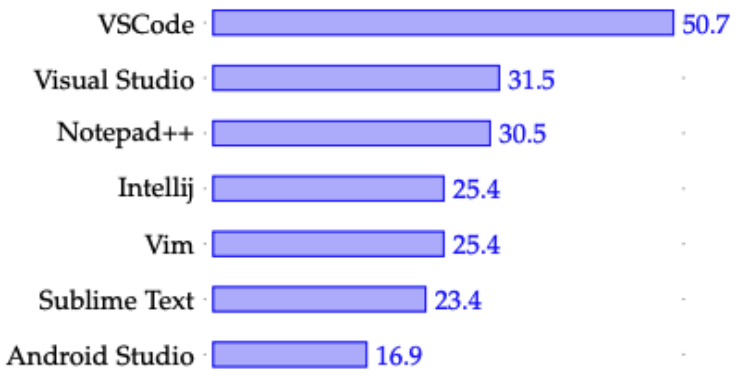


Figura 23 - Porcentagem de desenvolvedores que usam o editor (Paluca, 2021)

Ainda, conforme o autor, o editor possibilita a expansão das suas funcionalidades através do uso de extensões, as quais são desenvolvidas pela comunidade utilizando a linguagem Typescript e podem ser obtidas através da ferramenta presente no próprio editor, o Visual Code Extensions. Além das extensões, o editor já suporta por padrão funcionalidades para linguagens de programação, como auto completar, ir até uma função, checagem de erros etc.

4.4 IMPLEMENTAÇÃO DA FERRAMENTA

Para o desenvolvimento da solução foi utilizado como base o *framework* Cypress. Após efetuar a instalação do *framework* através do NPM, o Cypress foi executado utilizando uma *interface* de linha de comando. Após executar o comando *npx cypress open*, o *framework* gerou a estrutura inicial do projeto, incluindo as pastas e arquivos necessários para o

[38] https://www.electronjs.org/

desenvolvimento e execução dos testes. A figura 24 demonstra a estrutura de pastas e arquivos gerados pelo *framework* inicialmente.

O *script* de teste que será executado pelo *framework* foi armazenado na pasta *integration*, esta é a pasta padrão utilizada pelo *framework* para a execução dos *scripts* de testes, embora essa configuração seja personalizável, o autor optou por manter o padrão do *framework*.

O arquivo *cypress.json*[39] armazena as configurações de execução do *framework*. Nele são configurados parâmetros como tempo limite de execução, resolução de tela e outras configurações permitidas pelo *framework* para execução dos *scripts*. Já o arquivo *package.json*[40] é um tipo de manifesto do projeto. Nele são definidas as informações referentes ao projeto, como nome do autor, licença de uso e outras. É nele também que são registradas as dependências do projeto e definidos comandos personalizados que podem ser executados pelo NPM.

A pasta intitulada *plugins* é o local onde os *plugins* são armazenados. Por padrão, são implementados no arquivo index.js utilizando a linguagem JavaScript. Através deste arquivo é possível estender e personalizar as funcionalidades do *framework*. Qualquer pessoa com conhecimento na linguagem JavaScript pode implementar seus próprios *plugins*, além disto é possível encontrar na *Internet* uma infinidade de *plugins* desenvolvidos pela comunidade[41].

A pasta *support* contém o arquivo command.js[42], que permite que sejam implementados comandos personalizados. Estes comandos podem ser adicionados aos *scripts* simplificando a implementação dos testes e deixando o código mais legível.

Por fim na pasta *fixtures*[43] são armazenados arquivos que contém dados que serão utilizados nos testes, como credenciais de usuário para determinadas aplicações e outras informações.

---

[39] https://docs.cypress.io/guides/references/configuration#cypress-json
[40] https://www.luiztools.com.br/post/o-guia-completo-do-package-json-do-node-js/
[41] https://docs.cypress.io/plugins/directory
[42] https://docs.cypress.io/api/cypress-api/custom-commands
[43] https://docs.cypress.io/api/commands/fixture

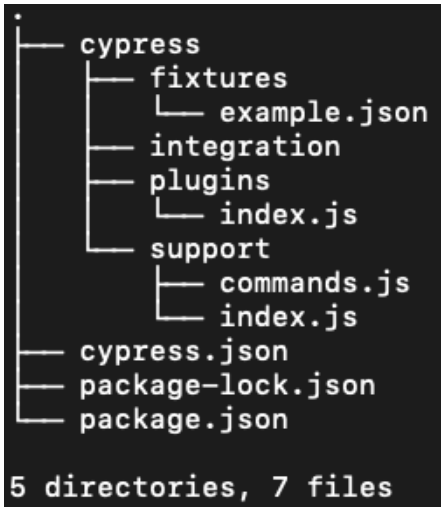


Figura 24 - Estrutura de diretórios e arquivos (O autor, 2021)

4.4.1 IMPLEMENTAÇÃO DO SCRIPT DE TESTE

Tendo como princípio a estrutura gerada pelo *framework*, iniciou-se o desenvolvimento do *script* de teste. Inicialmente foi criado o arquivo *pages.validator.spec.js*, que contém um *script* que segue a *interface* padrão encontrada em outro *framework* de testes chamado Mocha[44], através desta *interface*, é descrito o *script* de teste utilizando a chave "*describe()*" e também são definidos os testes que o *script* executará utilizando a chave "*it()*".

Para ser possível informar o endereço da página ou o caminho para os arquivos HTML das páginas que serão validadas, foi definida a variável *pagesToEvaluate* que recebe os dados informados na variável de ambiente chamada *pages*. Esta variável de ambiente é informada na execução da ferramenta na linha de comando. Tanto estas variáveis quanto a forma de execução são detalhados na seção Execução dos testes com a ferramenta.

Para cada página informada é efetuado pelo *script* um laço de repetição utilizando a função *forEach*[45] do JavaScript. Esta função garante que cada página informada seja validada pela lógica implementada no *script* de teste e permite ao usuário informar e validar mais de uma página com apenas um único comando.

[44] https://mochajs.org/#bdd
[45] https://developer.mozilla.org/pt-BR/docs/*Web*/JavaScript/Reference/Global_Objects/Array/forEach

O *script* inicia a validação através da execução da função *cy.visit*, que recebe como parâmetro o endereço da página. Assim o *script* inicia "visitando" a página alvo, que é completamente carregada pelo navegador de *Internet*. Em seguida, o *framework* executa a função *cy.inject*, injetando o axe-core *runtime* na página. Após a injeção do axe-core, é armazenada na variável *logFormat* a função da classe responsável por formatar e organizar os resultados dos testes. Logo em seguida, são armazenadas as configurações do axe-core na variável *wcagVersions*, através da qual é possível definir quais versões do guia WCAG serão utilizadas, possibilitando também nesta variável definir o uso de outros guias e configurações permitindo a personalização da validação de violações de acessibilidade[46].

Por fim, o *script* executa a função *cy.checkA11y*, recebendo os parâmetros dos elementos HTML que serão escaneados. No caso deste projeto, ele é informado com o valor *null*, para que assim sejam validados todos os elementos das páginas. Também é informado como parâmetro as configurações de execução do axe-core e as versões do guia WCAG que serão utilizadas na validação. Por último, é definido o parâmetro *logFormater* para que os resultados gerados pela função sejam formatados e tragam de forma organizada todas as violações detectadas. A figura 25 - Código do *script* de teste, demonstra o código implementado. Este e todos os códigos do projeto estão disponíveis no apêndice deste trabalho.

```
1  import { logHelper } from "../support/loghelper.js";
2
3  const pagesToEvaluate = Cypress.env('pages')
4
5  describe('Accessibility evaluation of web pages', () => {
6
7    pagesToEvaluate.forEach((page) => {
8
9      it('Evaluates accessibility for the eating page', () => {
10
11       cy.visit(page);
12
13       cy.injectAxe();
14
15       const logFormater = logHelper.logFormatter
16
17       const wcagVersions = { runOnly: { values: ['wcag2a', 'wcag2aa', 'wcag2aaa'] } }
18
19       cy.checkA11y(null, wcagVersions, logFormater);
20
21     })
22
23   });
24 })
25
```

Figura 25 - Código do *script* de teste (O autor, 2021)

[46] https://www.deque.com/axe/core-documentation/api-documentation/#options-parameter

5 EXECUÇÃO DOS TESTES COM A FERRAMENTA

Neste capítulo é apresentada a execução da ferramenta, demonstrando a configuração necessária do ambiente para a execução, o planejamento, a execução dos testes, uma análise dos resultados obtidos e um estudo comparativo, entre a ferramenta desenvolvida e outra reconhecida no mercado e no meio acadêmico.

5.1 CONFIGURAÇÃO DO AMBIENTE DE EXECUÇÃO

Como pré-requisito para a utilização da ferramenta, é obrigatório a instalação e configuração do Node.js na versão 16.13 ou superior, e do NPM na versão 8.1 ou superior. A instalação e configuração varia em relação ao sistema operacional utilizado. As instruções e arquivos para instalação são encontrados no site do Node.js[47]. Para o desenvolvimento deste trabalho foi utilizado um Macbook com sistema operacional macOS na versão 12.0 *(Monterey)*. Com este sistema operacional bastou efetuar o *download* do instalador disponibilizado no site do próprio Node.js, executar o instalador e seguir o passo a passo disponível.

Após o Node.js e o NPM instalados e configurados, para que se conclua a configuração do ambiente de execução da ferramenta, é necessário que o usuário esteja no diretório do projeto. Para instalar as dependências do projeto, deve-se executar o comando *npm install* em um cliente de terminal. Por fim, todas as dependências e configurações do projeto serão instaladas.

5.1.1 PLANEJAMENTO DA EXECUÇÃO DOS TESTES

Para executar a ferramenta, foram selecionadas três páginas *Web* de interesse público, as quais, na concepção do autor, deveriam possuir o mínimo ou nenhuma violação de acessibilidade que dificulte ou até mesmo impeça o acesso e o uso, para que pessoas com algum grau de deficiência possam utilizá-las de forma plena.

As análises dos sites selecionados, cujas páginas serão objeto dos testes executados pela ferramenta, foram baseadas na experiência de uso do autor, conforme exposto abaixo:

[47] https://nodejs.org/en/download/

- **Sábio – Sistema de automação de Bibliotecas**: Esta página pertence à biblioteca pública do estado de Santa Catarina, atualmente disponível no site da Fundação Catarinense de Cultura, acessada através da opção de menu "Consulta e Renovação Online" ou pelo endereço http://sabio.biblioteca.sc.gov.br/sabio/. É nesta página que os usuários podem consultar o acervo, efetuar reserva de livros, consultar horários de funcionamento e outras funções importantes da biblioteca pública do estado.
- **Portal Governo de Santa Catarina**: Página inicial do portal do governo do Estado de Santa Catarina, acessada pelo endereço https://www.sc.gov.br/. Neste portal é possível encontrar os serviços oferecidos pelo governo em áreas como educação, direitos sociais, segurança pública e outros serviços prestados ao cidadão. Também é possível acessar as principais notícias e oportunidades oferecidas pelo governo do estado como benefícios sociais.
- **Pergamum UFSC**: O Pergamum é o sistema de catálogo de livros da Biblioteca Universitária da Universidade Federal de Santa Catarina, no qual é possível consultar todo o acervo disponível, sejam obras físicas ou virtuais. Nesse sistema são encontradas obras, seus exemplares, informações sobre elas e outras funcionalidades. A página pode ser acessada pelo endereço https://pergamum.ufsc.br/pergamum/biblioteca/index.php

Cada página listada será alvo da execução da ferramenta e terão suas violações de acessibilidade descritas. Esta execução tem como objetivo avaliar o funcionamento da ferramenta e apontar as violações de acessibilidade encontradas nessas páginas. Na próxima seção serão descritas as violações de acessibilidade identificadas, os elementos HTML afetados e as possíveis soluções para tais.

5.2 EXECUÇÃO E ANÁLISE DOS RESULTADO

Para efetuar a execução dos testes de acessibilidade de forma automatizada utilizando a ferramenta desenvolvida, é necessário a execução de um único comando no cliente de linha de comando, composto pela execução da ferramenta npx[48], pelo *framework* Cypress recebendo o parâmetro de execução *run* e, também, pelo parâmetro *--env pagaes* que recebe os endereços

[48] https://blog.rocketseat.com.br/conhecendo-o-npx-executor-de-pacote-do-npm/

das páginas que serão alvos dos testes. Na Figura 26 – Comando para execução dos testes, é demonstrado o comando completo utilizado para executar a ferramenta.

```
~ % npx cypress run --env pages = '["http://sabio.biblioteca.sc.gov.br",
"https://www.sc.gov.br/","https://pergamum.ufsc.br/pergamum/biblioteca/index.php"]'
```

Figura 26 - Comando para execução dos testes (O Autor, 2021)

Após executar o comando descrito anteriormente, o *framework* Cypress inicia o acesso às páginas e a validação das possíveis violações de acessibilidade. Após finalizar a execução, caso sejam detectadas violações, é gerado na saída do cliente de linha de comando um relatório contendo um sumário das violações encontradas, detalhes sobre cada violação e a localização de cada violação na estrutura HTML da página. Também é exibido para cada violação um endereço *Web* que traz a descrição completa da regra violada, formas de corrigir o problema e outras informações.

```
Accessibility evaluation of web pages

    Number of violations: 6 types of Wcag violations detected.
```

(index)	Id of Axe rule violated	User impact	Description of rule	Number of nodes affected
0	'button-name'	'critical'	'Ensures buttons have discernible text'	'2'
1	'frame-title'	'serious'	'Ensures <iframe> and <frame> elements have an accessible name'	'2'
2	'html-has-lang'	'serious'	'Ensures every HTML document has a lang attribute'	'1'
3	'image-alt'	'critical'	'Ensures <img> elements have alternate text or a role of none or presentation'	'26'
4	'label'	'critical'	'Ensures every form element has a label'	'1'
5	'scrollable-region-focusable'	'moderate'	'Ensure elements that have scrollable content are accessible by keyboard'	'1'

```
        ######## Violation 1 of 6: 'button-name', whit User impact critical ########

        Buttons must have discernible text. Acess for more details: https://dequeuniversity.com/rules/axe/4.3/button-name?application=axeAPI

        2 instances of this violation:

        Violation 1 instance 1: <button class="x-btn-text" type="button" id="ext-gen226" style="background-image: url(&quot;http://sabio.biblioteca.sc.gov.br/sabio/modulo.sabioweb/imagens/previous.png&quot;);"> </button>

        Violation 1 instance 2: <button class="x-btn-text" type="button" id="ext-gen234" style="background-image: url(&quot;http://sabio.biblioteca.sc.gov.br/sabio/modulo.sabioweb/imagens/next.png&quot;);"> </button>

        ######## Violation 2 of 6: 'frame-title', whit User impact serious ########

        Frames must have an accessible name. Acess for more details: https://dequeuniversity.com/rules/axe/4.3/frame-title?application=axeAPI

        2 instances of this violation:

        Violation 2 instance 1: <iframe id="__gwt_historyFrame" style="width: 0; height: 0; border: 0"></iframe>
```

Figura 27 - Relatório de execução dos testes (O Autor, 2021)

5.2.1 Sábio - Sistema de automação de Bibliotecas

Após o autor acessar a página inicial do Sábio - Sistema de Automação de Bibliotecas, foi possível concluir que ele é o sistema que permite, entre outras funções, a consulta do acervo da Biblioteca Pública do Estado, a reserva de exemplares e o acompanhamento de empréstimos. A página também disponibiliza informações inerentes à biblioteca como o horário de funcionamento, dados de contato e outras informações, ou seja, esta página *Web* é um elemento

importante para a educação e inclusão da população, portanto é essencial que esse sistema conte com páginas *Web* acessíveis que permitam a inclusão de toda a população.

Figura 28 - Página inicial do Sábio (Sábio, 2021)

Após finalizar a execução dos testes automatizados na página inicial do Sábio[49] com a solução desenvolvida neste trabalho, a ferramenta detectou ao todo seis violações de acessibilidade, sendo três violações com impacto crítico, duas com impacto sério e uma com impacto moderado. As violações estão presentes em trinta e três nodos da estrutura HTML da página, o quadro abaixo resume os dados das violações encontradas.

Índice	Regra do Axe violada	Impacto ao Usuário	Descrição da regra	Nodos afetados	Critério de Sucesso WCAG
1	button-name	Crítica	Ensures buttons have discernible text	2	4.1.2
2	frame-title	Séria	Ensures <iframe> and <frame> elements have an accessible name	2	2.4.2
3	html-has-lang	Séria	Ensures every HTML document has a lang attribute	1	3.1.1
4	image-alt	Crítica	Ensures <img> elements have alternate text or a role of none or presentation	26	1.1.1

[49] http://sabio.biblioteca.sc.gov.br/sabio/

5	label	Crítica	Ensures every form element has a label	1	1.3.1
6	scrollable-region-focusable	Moderada	Ensure elements that have scrollable content are accessible by keyboard	1	2.4.7

Quadro 8 - Violações na página inicial do Sábio (O Autor, 2021)

A primeira violação de acessibilidade detectada pela ferramenta é identificada pela biblioteca Axe-core como *button-name*, ao consultar o site da Deque University, através do endereço fornecido no relatório de execução[50], é demonstrado que esta violação significa que os botões da página deveriam possuir uma descrição em texto de forma mais clara. Essa violação contraria o Critério de Sucesso 4.1.2: Nome, Função, Valor, que segundo descrito no site do WCAG[51], tem como objetivo garantir que as tecnologias assistivas, como leitores de tela, possam coletar as informações sobre a função desses botões e o seu estado atual. No caso da página do Sábio, há dois botões que por não contarem com uma descrição de texto, atrapalham ou até impedem o uso da aplicação por pessoas com algum grau de deficiência visual que dependam de leitores de tela.

A segunda violação detectada foi da regra *frame-title*, segundo o que consta no site da Deque[52], esta regra se refere ao Critério de Sucesso 2.4.2: Páginas com Título[53]. Ao violar essa regra, a página do Sábio prejudica tecnologias assistivas que utilizem o título dos quadros ou janelas para auxiliar o usuário. Esse critério de sucesso tem como objetivo auxiliar os usuários a entenderem de forma clara qual é a finalidade da página que estão acessando através de uma descrição no título da mesma. No caso da página do Sábio, temos dois quadros que não possuem um título claro.

A terceira violação detectada foi da regra *html-has-lang*. Conforme visto no site da Deque [54], ela faz referência direta a ferramentas que efetuam a leitura da tela. Essas ferramentas precisam identificar qual o idioma utilizado na página, para, então, utilizar o idioma definido pelo usuário. Por exemplo, a página é em português, mas o leitor de tela do usuário, por padrão, efetua a leitura em inglês, dessa forma, ele não conseguiria efetuar a leitura da tela, uma vez que não está definido o idioma, impossibilitando que a ferramenta possa efetuar a troca e operar

[50] https://dequeuniversity.com/rules/axe/4.3/button-name
[51] https://www.w3.org/WAI/WCAG21/Understanding/name-role-value.html
[52] https://dequeuniversity.com/rules/axe/4.3/frame-title
[53] https://www.w3.org/WAI/WCAG21/Understanding/page-titled
[54] https://dequeuniversity.com/rules/axe/4.3/html-has-lang

normalmente. Esta regra faz referência ao Critério de Sucesso 3.1.1: Idioma da Página[55], cujo objetivo é garantir que os desenvolvedores disponibilizem informações sobre o idioma da página para as tecnologias assistivas.

A violação mais comum encontrada na página inicial do Sábio foi da regra *image-alt*[56]. De acordo com as informações do site da Deque, quando essa regra é violada, ferramentas leitoras de tela não possuem uma forma de transcrever as imagens para os usuários. Tal regra violada faz referência ao Critério de Sucesso 1.1.1: Conteúdo Não Textual[57], que afirma que o conteúdo exibido ao usuário precisa de uma alternativa textual equivalente. A ausência deste critério pode prejudicar ou impedir o uso da página por usuários com algum grau de deficiência visual.

A regra intitulada *label*[58], refere-se ao Critério de Sucesso: 1.3.1 Informações e Relações, que tem como propósito que todo campo utilizado para inserção de texto seja acessível. Para tanto, é necessário que tal campo tenha uma etiqueta que o identifica, tornando-o passível de leitura por leitores de tela. Além disso, também elimina ambiguidades que possam confundir os usuários.

A última violação detectada pela ferramenta é da regra *scrollable-region-focusable*[59]. Ao violar esta regra, a página prejudica ou até mesmo impossibilita a navegação utilizando o teclado. Ela se relaciona com o Critério de Sucesso 2.4.7 Foco Visível[60], que permitirá que os usuários saibam qual elemento da página se relaciona a teclas do teclado, possibilitando a navegação na página com o uso deste.

Após analisar os resultados da execução na página inicial do Sábio e ao navegar pela página, fica evidente que a maior parte das violações de acessibilidade detectadas pela ferramenta são violações que prejudicam ou impedem na sua grande maioria o uso de ferramentas leitoras de tela. Assim, é possível deduzir que os usuários dependentes de tais ferramentas terão dificuldades para utilizar a página. Conforme o autor pôde constatar, os ajustes sugeridos pela empresa Deque e pelo próprio guia WCAG para as violações encontradas são tecnicamente simples. Dessa forma, caso houvesse o uso da ferramenta proposta neste

[55] https://www.w3.org/WAI/WCAG21/Understanding/language-of-page
[56] https://dequeuniversity.com/rules/axe/4.3/image-alt
[57] https://www.w3c.br/traducoes/wcag/wcag21-pt-BR/#non-text-content
[58] https://dequeuniversity.com/rules/axe/4.3/label
[59] https://dequeuniversity.com/rules/axe/4.3/scrollable-region-focusable
[60] https://www.w3.org/WAI/WCAG21/Understanding/focus-visible

trabalho ou equivalente, os ajustes necessários poderiam ser identificados e implementados ainda durante o desenvolvimento da página.

5.2.2 Portal do Governo de Santa Catarina

Após uma análise exploratória efetuada pelo autor, foi possível constatar que o Portal do Governo de Santa Catarina é uma página *Web* que contém informações, notícias, oportunidades e acesso a inúmeros serviços oferecidos pelo governo do Estado de forma *online*. É nela que o cidadão encontra informações importantes sobre a gestão pública do Estado e onde são disponibilizados acessos para serviços como direitos sociais e cidadania, saúde, educação, segurança pública e outros. Ao acessar a página fica claro que se trata de uma ferramenta de interesse público. Dado esse motivo, deveria possuir uma boa acessibilidade, permitindo que pessoas com deficiência possam usufruir de todos os recursos e informações contidas nela.

Figura 29 – Portal do Governo do Estado de Santa Catarina (Portal do Governo, 2022)

As validações na página do portal do governo do Estado de Santa Catarina foram executadas de forma automatizada com sucesso. Após a execução, foram detectadas cinco violações de acessibilidade. Destas cinco violações, quatro possuíam um impacto sério e uma

possuía impacto moderado. Ao todo dezessete nodos da estrutura HTML apresentaram alguma violação das regras de acessibilidade. No Quadro 9, é possível encontrar uma síntese das violações detectadas.

Quadro 9 - Violações detectadas no portal do governo de Santa Catarina

Índice	Regra do Axe violada	Impacto do Usuário	Descrição da regra	Nodos afetados	Critério de Sucesso WCAG
1	color-contrast	Sério	Ensures the contrast between foreground and background colors meets WCAG 2 AA contrast ratio thresholds	6	1.4.3
2	duplicate-id-active	Sério	Ensures every id attribute value of active elements is unique	1	4.1.1
3	duplicate-id	Pequeno	Ensures every id attribute value is unique	1	4.1.1
4	html-lang-valid	Sério	Ensures the lang attribute of the <html> element has a valid value	1	3.1.1
5	link-name	Sério	Ensures links have discernible text	10	2.4.4

Quadro 9 - Violações detectadas no portal do governo de Santa Catarina (O Autor, 2022)

Inicialmente, foi detectada pela ferramenta em seis nodos da estrutura HTML a violação da regra intitulada *color-contrast*. As informações disponíveis na página da Deque[61] afirmam que esta regra se relaciona ao critério de sucesso do WCAG 1.4.3 Contraste Mínimo[62]. Não respeitar essa regra e não atingir o critério de sucesso, significa que a página possui um ou mais textos que não dispõem de um contraste mínimo entre o texto e o fundo da página. Dessa forma, algumas pessoas com baixa visão ou que não enxergam algum espectro de cor, podem não conseguir ler tais textos.

Posteriormente, foi detectada a violação da regra *duplicate-id-active*. Esta regra, segundo a Deque[63], define que cada elemento da página deve possuir um identificador único e trata especificamente de elementos ativos na página e/ou passíveis de foco. Ela se relaciona ao critério de sucesso 4.1.1 Análise[64], que dispõe que a página, ao possuir identificadores duplicados em elementos ativos, pode impedir o uso de leitores de tela ou *scripts* do lado do

[61] https://dequeuniversity.com/rules/axe/4.3/color-contrast
[62] https://www.w3.org/WAI/WCAG21/Understanding/contrast-minimum.html
[63] https://dequeuniversity.com/rules/axe/4.3/duplicate-id-active
[64] https://www.w3.org/TR/UNDERSTANDING-WCAG20/ensure-compat-parses.html

cliente, que seriam utilizados por pessoas com baixa ou nenhuma visão para navegarem na página. A terceira regra violada é a *duplicate-id*, similar a regra citada anteriormente, porém, ao contrário desta, os elementos que possuem identificadores repetidos, são elementos inativos e não passíveis de foco. Dessa maneira, tem um impacto menor ao usuário, porém com as mesmas consequências da violação anterior.

A quarta violação detectada na página foi da regra *html-has-lang*, mesma regra violada na página do Sábio. Assim como ocorre na página do Sábio, na página do portal do governo do Estado de Santa Catarina, ao violar a regra, leitores de tela utilizados por pessoas com baixa visão são prejudicados, atrapalhando ou até mesmo impedindo o uso da página por tais usuários, pois, uma vez que não há definição do idioma da página, essas ferramentas não conseguem se adaptar para o pleno funcionamento.

A última violação detectada foi a *link-name*[65], que foi a violação mais presente na página afetando ao todo 10 nodos. A regra *link-name*, define que todo *link,* seja ele texto ou imagem, devem possuir um texto descritivo que permita aos leitores de tela transcrevam para pessoas com baixa ou nenhuma visão o seu conteúdo. Essa regra contempla o critério de sucesso 2.4.4. Finalidade do Link Em contexto[66]. O guia WCAG propõe uma série de técnicas para a correção desta violação, a mais comum é a adição de um texto descritivo no *link*, a fim de informar a sua finalidade.

5.2.3 Pergamum UFSC

Conforme acesso efetuado pelo autor e a sua experiência prévia utilizando o Pergamum, é possível afirmar que se trata do sistema utilizado pela Universidade Federal de Santa Catarina para consulta do acervo das suas bibliotecas e outras funções. A página inicial pode ser acessada através do site da biblioteca universitária ou pelo endereço https://pergamum.ufsc.br/. Através desta página, qualquer pessoa pode consultar as obras disponíveis no acervo, obtendo informações sobre tais obras, como autores, localização na biblioteca, versões virtuais e outras. Também é possível, através do Pergamum, a reserva de exemplares para futura retirada na biblioteca. O Pergamum consiste em um elemento importante para o funcionamento da biblioteca da Universidade, e representa para a população uma importante ferramenta para obtenção de conhecimento, principalmente para estudantes e

[65] https://dequeuniversity.com/rules/axe/4.3/link-name
[66] https://www.w3.org/WAI/WCAG21/Understanding/link-purpose-in-context.html

professores. Ou seja, é indispensável que a página seja acessível, possibilitando que todos que dela necessitem tenham o devido acesso.

Figura 30 - Página inicial do Pergamum (Pergamum UFSC, 2022)

As validações foram executadas na versão de alto contraste da página do Pergamum. Foram detectadas ao todo nove violações na página inicial que afetam vinte e um nodos da estrutura HTML. Além das violações já detectadas nas páginas validadas anteriormente, a ferramenta também detectou a violação das regras *object-alt*[67] e *select-name*[68], relacionadas respectivamente aos critérios de sucesso 1.1.1 Conteúdo Não Textual[69], e 4.1.2 Nome, Função, Valor, do guia WCAG[70]. Foi possível constatar que a maioria das violações possui um impacto crítico ou sério, ou seja, violações que se não corrigidas prejudicarão significativamente a acessibilidade. Assim como nas páginas avaliadas anteriormente, foram detectadas violações que possuem maior impacto para usuários com baixa ou nenhuma visão, o que na concepção do autor representa um grande desafio para estudantes, professores e toda a comunidade que necessita da página e dos serviços oferecidos por ela. O Quadro 10 mostra um resumo das violações detectadas.

[67] https://dequeuniversity.com/rules/axe/4.3/object-alt
[68] https://dequeuniversity.com/rules/axe/4.3/select-name
[69] https://www.w3c.br/traducoes/wcag/wcag21-pt-BR/#non-text-content
[70] https://www.w3.org/WAI/WCAG21/Understanding/name-role-value.html

Índice	Regra do Axe violada	Impacto do Usuário	Descrição da regra	Nodos afetados	Critério de Sucesso WCAG
1	color-contrast	Sério	Ensures the contrast between foreground and background colors meets WCAG 2 AA contrast ratio thresholds	1	1.4.3
2	duplicate-id-active	Sério	Ensures every id attribute value of active elements is unique	1	4.1.1
3	duplicate-id	Pequeno	Ensures every id attribute value is unique	1	4.1.1
4	html-has-lang	Sério	Ensures the lang attribute of the <html> element has a valid value	1	3.1.1
5	image-alt	Crítico	Ensures <img> elements have alternate text or a role of none or presentation	4	1.1.1
6	label	Crítico	Ensures every form element has a label	4	1.3.1
7	link-name	Sério	Ensures links have discernible text	2	2.4.4
8	object-alt	Sério	Ensures <object> elements have alternate text	1	1.1.1
9	select-name	Crítico	Ensures select element has an accessible name	6	4.1.2

Quadro 10 - Violações detectadas na página do Pergamum (O Autor, 2022)

No apêndice deste trabalho, é possível ler os relatórios de execução completos de todas as páginas analisadas, demonstrando os elementos da estrutura HTML afetados pelas violações, permitindo também a qualquer interessado identificar e corrigir as violações descritas.

5.3 ESTUDO COMPARATIVO ENTRE AS FERRAMENTAS

Para constatar se a solução desenvolvida neste trabalho alcançou o objetivo de detectar de forma automatizada falhas de acessibilidade em páginas *Web*, foi efetuado um estudo comparativo. Neste estudo, os resultados obtidos anteriormente com a execução dos testes pela ferramenta nos sites escolhidos, foram comparados com o resultado obtido ao efetuar os mesmos testes, porém utilizando a ferramenta Achecker. Como critério de comparação entre as ferramentas será utilizado a quantidade de violações detectadas por cada uma delas.

Como demonstrado anteriormente, a ferramenta AChecker é amplamente utilizada no mercado e no meio acadêmico para efetuar a validação automatizada de falhas de acessibilidade em páginas *Web*, o que vai ao encontro do objetivo do estudo comparativo, onde será aferido se a solução elaborada neste trabalho se equipara a uma ferramenta amplamente reconhecida e utilizada.

Ao submeter a página do Sábio - Sistema de Automação de Bibliotecas à análise da ferramenta AChecker, foram detectadas duas violações de acessibilidade. A primeira referente ao Critério de Sucesso 1.1.1: Conteúdo Não Textual[71], afetando um nodo da estrutura HTML. A segunda violação detectada se refere ao Critério de Sucesso 3.1.1: Idioma da Página[72], que segundo o relatório da ferramenta, afeta dois nodos da estrutura HTML. Ao se comparar os resultados obtidos pela ferramenta Achecker em relação aos resultados obtidos pela ferramenta desenvolvida neste trabalho, é possível constatar que para a página do Sábio, a solução desenvolvida neste trabalho conseguiu detectar as duas violações detectadas pela Achecker. Além disso, também conseguiu detectar violações de acessibilidade não detectadas pela Achecker, ou seja, a solução desenvolvida neste trabalho, quando aplicada a página inicial do Sábio, detectou uma quantidade maior de violações de critérios de sucesso do guia WCAG. O Quadro 11 demonstra quais foram os critérios de sucesso violados e detectados por cada uma das ferramentas.

Critério de Sucesso / Ferramenta	1.1.1	1.3.1	2.4.2	2.4.7	3.1.1	4.1.2
Achecker	x				x	
Ferramenta do trabalho	x	x	x	x	x	x

Quadro 11 - Comparação entre detecção de violações pelas ferramentas no Sábio (O Autor, 2022)

Na página inicial do portal do Governo de Santa Catarina, a ferramenta Achecker detectou a violação de oito critérios de sucesso que afetam ao todo cinquenta e um nodos da estrutura HTML. A AChecker conseguiu detectar todas as violações que foram detectadas pela ferramenta desenvolvida neste trabalho. Foram detectadas as violações dos critérios de sucesso do guia WCAG 1.4.3 Contraste Mínimo, 2.4.4 Finalidade do Link Em Contexto, 3.1.1 Idioma das Partes e 4.1.1 Análise. Além de detectar todas as violações, tal qual a ferramenta deste trabalho, a Achecker ainda detectou outras violações na página, não detectadas pela solução elaborada pelo autor. Foram detectadas violações dos critérios de sucesso 1.1.1 Conteúdo Não

[71] https://www.w3c.br/traducoes/wcag/wcag21-pt-BR/#non-text-content
[72] https://www.w3.org/WAI/WCAG21/Understanding/language-of-page

visual[73], 1.3.1 Informações e Relações[74], 1.4.4 Redimensionar Texto[75] e 3.3.2 Rótulos ou Instruções[76]. É possível constatar que para a página do Governo do Estado de Santa Catarina, a ferramenta AChecker obteve um desempenho melhor ao detectar um número maior de violações quando comparado com a ferramenta deste trabalho, conforme observado abaixo.

Critério de Sucesso / Ferramenta	1.1.1	1.3.1	1.4.3	1.4.4	2.4.4	3.1.1	3.2.2	4.1.1
Achecker	x	x	x	x	x	x	x	x
Ferramenta do trabalho			x		x	x		x

Quadro 12 - Comparação entre detecção de violações pelas ferramentas no portal do governo (O Autor, 2022)

Para concluir o estudo comparativo entre as ferramentas, o site do Pergamum UFSC também foi submetido à análise da ferramenta AChecker. Como resultado, foi detectada a violação de cinco critérios de sucesso pela AChecker. Já a ferramenta deste trabalho detectou a violação de sete critérios de sucesso, o critério de sucesso 1.4.4 Redimensionar Texto foi o único detectado apenas pela ferramenta AChecker, demonstrando um resultado similar ao obtido ao avaliar o Sábio, onde a solução apresentada conseguiu detectar um número maior de violações na página quando comparado com a AChecker, conforme quadro a seguir.

Critério de Sucesso / Ferramenta	1.1.1	1.3.1	1.4.3	1.4.4	2.4.4	3.1.1	4.1.1	4.1.2
Achecker	x			x	x	x	x	
Ferramenta do trabalho	x	x	x		x	x	x	x

Quadro 13 - Comparação entre detecção de violações pelas ferramentas no Pergamum (O Autor,2022)

Após concluir a execução na ferramenta AChecker para todas as páginas e comparando com os resultados obtidos ao utilizar a solução proposta neste trabalho, constatou-se que a solução desenvolvida possui capacidade similar ou superior em número de violações detectadas em comparação com a Achecker. Ainda que para páginas como a do portal do governo de Santa Catarina o resultado divirja dos demais, isto é justificável devido ao fato da biblioteca do axe-core utilizada no desenvolvimento da solução cubra apenas 57% dos critérios de sucesso do

[73] https://www.w3c.br/traducoes/wcag/wcag21-pt-BR/#non-text-content
[74] https://www.w3c.br/traducoes/wcag/wcag21-pt-BR/#info-and-relationships
[75] https://www.w3c.br/traducoes/wcag/wcag21-pt-BR/#resize-text
[76] https://www.w3c.br/traducoes/wcag/wcag21-pt-BR/#labels-or-instructions

guia WCAG[77]. Ou seja, além da diferença entre o modo como as ferramentas detectam as violações, essa limitação de cobertura afetou o número de violações detectadas, pois para efetuar uma comparação justa, foi utilizado o único guia coberto pela AChecker, o WCAG na versão 2.1. Diferente da AChecker, a solução proposta neste trabalho permite também a detecção de outras violações com o uso de conjuntos de regras disponíveis[78] como a de boas práticas, Section508 e outros que aumentam consideravelmente o número de violações passíveis de detecção, por conseguinte a efetividade da ferramenta.

Além da questão de número de violações detectadas, pode-se considerar outros aspectos que posicionam a solução proposta neste trabalho acima da ferramenta AChecker, como a possibilidade da sua execução em um ambiente de CI/CD, sendo totalmente personalizável e estendível. Ademais, a solução deste trabalho também pode ser usada como complemento à Achecker e outras ferramentas similares.

Na seção de apêndices deste trabalho, no Apêndice B – Capturas de tela da execução na ferramenta Achecker, é possível obter as evidências que evidenciam os resultados descritos nesta seção.

[77] https://github.com/dequelabs/axe-core#the-accessibility-rules

[78] https://github.com/dequelabs/axe-core/blob/develop/doc/rule-descriptions.md

6 CONCLUSÃO

O tema da acessibilidade digital atinge toda a sociedade, tornando-se essencial para democratização e garantia do acesso digital. Considerando o grande aumento de acessos à Web nos últimos anos, cada vez mais pessoas dos mais variados grupos tiveram o seu cotidiano relacionado a ela, seja para lazer, educação ou informação. Um desses grupos é o de pessoas com deficiência, o qual necessita que as páginas Web sejam acessíveis e lhes permitam navegar sem dificuldades. O desenvolvimento de páginas Web acessíveis é algo complexo. Neste contexto, organizações como o W3C, buscam definir normas e princípios que visam auxiliar os desenvolvedores e criadores de conteúdo digital a desenvolverem páginas Web mais acessíveis.

O presente trabalho teve como proposta desenvolver uma solução para a automação dos testes de acessibilidade de páginas Web, de modo a auxiliar desenvolvedores e criadores de conteúdo no desafio de criarem páginas Web mais acessíveis. Para alcançar este objetivo, neste trabalho foi desenvolvida uma ferramenta capaz de detectar a violação de normas de acessibilidade de páginas Web, à luz das normas estabelecidas no guia WCAG e outros. Através da ferramenta, é possibilitado ao desenvolvedor ter uma visão clara se as páginas desenvolvidas possuem uma boa acessibilidade. Isto ocorre com a sinalização das violações de regras de acessibilidade detectadas de forma automatizada, onde são informadas a localização da violação na estrutura HTML, informações sobre as violações e como corrigí-las. É possível, através da ferramenta, validar a acessibilidade de qualquer página Web independente da tecnologia usada no seu desenvolvimento.

A ferramenta foi projetada e desenvolvida utilizando frameworks como NodeJs, Cyrpess e a biblioteca axe-core, a qual é um elemento central para o funcionamento e eficácia da ferramenta. Ela possui o intuito de ser genérica, gratuita e com código aberto, com o propósito de ser utilizada para validar a acessibilidade digital em qualquer página Web, apenas informando o endereço ou arquivo HTML desta. O projeto foi idealizado para ser executado através de um script, que pode ser facilmente reproduzido em outras linguagens e utiliza tecnologias modernas e amplamente difundidas. Dessa forma, a solução pode ser executada de maneira simples em ambientes de desenvolvimento de páginas Web.

Após finalizar o desenvolvimento da ferramenta, foram executadas validações em determinadas páginas Web. A execução com sucesso da ferramenta e a detecção de violações de acessibilidade, demonstrou que o objetivo deste trabalho foi alcançado. Além disso, também foi efetuada uma comparação com uma ferramenta utilizada no mercado, objetivando atestar se

a solução desenvolvida neste trabalho, possuía a mesma capacidade em número de detecções de violações detectadas. Ao fim da comparação dos resultados, pode-se atestar que a ferramenta desenvolvida neste trabalho possui uma capacidade similar ou superior quando comparada a outra ferramenta.

Por fim, a elaboração deste trabalho proporcionou ao autor uma grande oportunidade de conhecimento sobre assuntos como tipos de deficiência, acessibilidade Web e suas normas, o que está sendo estudado e desenvolvido sobre acessibilidade Web, testes manuais e automatizados de acessibilidade em páginas Web, ferramentas modernas de desenvolvimento, scripts e todas as tecnologias utilizadas para compor este projeto.

6.1 TRABALHOS FUTUROS

A ferramenta desenvolvida neste trabalho e os conceitos abordados podem ser incrementados e melhorados. Ficam propostos como trabalhos futuros, melhorias da solução desenvolvida e novos conhecimentos a serem abordados, conforme listado abaixo:

- Estudar e abordar outros guias de acessibilidade como o eMAG, Section508, *EN 301 549 European accessibility standard* e outros.
- Inserir a funcionalidade de paralelização dos testes na solução proposta permitindo a validação de várias páginas em simultâneo.
- Adicionar um mecanismo para efetuar *login* e a validação em páginas *Web* protegidas.
- Melhorar o relatório de execução com a geração de um arquivo no formato HTML, contendo estatísticas e mais informações dos resultados da execução.
- Inserir a possibilidade de validar páginas *Web* no formato mobile.
- Estender as regras e guias suportados pela solução.

REFERÊNCIAS

ABOU-ZAHRA, Shadi (ed.). **Selecting *Web* Accessibility Evaluation Tools**. 2017. Elaborado por W3C. Disponível em: <https://www.w3.org/WAI/test-evaluate/tools/selecting/>. Acesso em: 07 set. 2021.

ABOU-ZAHRA, Shadi. A Data Model to Facilitate the Automation of *Web* Accessibility Evaluations. **Electronic Notes In Theoretical Computer Science**, [S.L.], v. 157, n. 2, p. 3-9, maio 2006. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1571066106002398?via%3Dihub>. Acesso em: 07 set. 2021.

ACCESSMONITOR (Portugal). **Acesse Monitor.** Disponível em: https://accessmonitor.acessibilidade.gov.pt/. Acesso em: 24 nov. 2021.

ALISMAIL, Sarah; CHIPIDZA, Wallace. Accessibility evaluation of COVID-19 vaccine registration *Web*sites across the United States. **Journal Of The American Medical Informatics Association**, California, v. 28, n. 9, p. 1990-1995, 16 maio 2021. Oxford University Press (OUP). http://dx.doi.org/10.1093/jamia/ocab105. Disponível em: <https://academic.oup.com/jamia/article/28/9/1990/6276434>. Acesso em: 22 nov. 2021.

AXE-CORE. 2021. Disponível em: <https://github.com/dequelabs/axe-core>. Acesso em: 08 dez. 2021.

BATEMAN, Ryan. **Deque Systems Launches axe DevTools Pro**. 2021. Disponível em: <https://finance.yahoo.com/news/deque-systems-launches-axe-devtools-160000830.html>. Acesso em: 16 nov. 2021.

BENSON JUNIOR, Brent W. JavaScript. **Acm Sigplan Notices**, Burlington, v. 34, n. 4, p. 25-27, maio 1999. Disponível em: <https://dl.acm.org/doi/10.1145/312009.312023>. Acesso em: 02 dez. 2021.

BERRY, Aaron. **A11y testing with axe-core**. 2020. Disponível em: https://aaron-kt-berry.medium.com/a11y-testing-with-axe-core-eb074744e073. Acesso em: 08 dez. 2021.

BOWMAN, Katy. **An Introduction To Running Lighthouse Programmatically**. 2020. <Disponível em: https://www.smashingmagazine.com/2020/09/introduction-running-lighthouse-programmatically/>. Acesso em: 16 nov. 2021.

BRASIL. **Decreto Nº 5.296 de 2 de dezembro de 2004**. Disponível em: <http://www.planalto.gov.br/ccivil_03/_Ato2004-2006/2004/Decreto/D5296.htm> Acesso em: 15 Aug. 2021.

BUREAU OF *INTERNET* ACCESSIBILITY (Rhode Island). **History of the *Web* Content Accessibility Guidelines (WCAG)**. 2019. Disponível em: <https://www.boia.org/blog/history-of-the-*Web*-content-accessibility-guidelines-wcag>. Acesso em: 29 ago. 2021.

CAIADO, R. et al. **Metodologia de Revisão Sistemática da Literatura com aplicação do método de apoio multicritério à decisão smarter.** Anais do XII Congresso Nacional de Excelência em Gestão e III Inovarse. 2016. Disponível em: https://www.inovarse.org/sites/default/files/T16_002.pdf. Acesso em: 08 ago. 2021.

CALDWELL, Ben *et al* (ed.). ***Web*** **Content Accessibility Guidelines (WCAG) 2.0**. 2014. Disponível em: https://www.w3.org/Translations/WCAG20-pt-br/#programmaticallysetdef. Acesso em: 31 ago. 2021.

CAMPOVERDE-MOLINA, Milton *et al*. Empirical Studies on *Web* Accessibility of Educational *Web*sites: a systematic literature review. **Ieee Access**, Cuenca, v. 8, n. 1, p. 91676-91700, 14 maio de 2020. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/access.2020.2994288. Disponível em: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9092982>. Acesso em: 22 nov. 2021.

CDC. **Disability and Health Overview**. Centers for Disease Control and Prevention. Disponível em: https://www.cdc.gov/ncbddd/disabilityandhealth/disability.html. Acesso em: 15 Aug. 2021.

CHANTRE, José Rui Moneteiro. **Testes automáticos de acessibilidade em aplicativos móveis**. 2015. 90 f. Dissertação (Mestrado) - Curso de Engenharia da Informática, Universidade da Beira Interior, Covilhã - Portugal, 2015. Disponível em: <https://ubibliorum.ubi.pt/bitstream/10400.6/5775/1/4657_8856.pdf>. Acesso em: 31 ago. 2021.

COMSCORE. **Digital future in focus Brazil 2015**. 2015. Disponível em: https://www.comscore.com/por/Insights/Apresentacoes-e-documentos/2015/2015-Brazil-Digital-Future-in-Focus. Acesso em: 25 jul. 2021.

CONDE, Antônio João Menescal. **Definição de cegueira e baixa visão**. 2021. Disponível em: http://www.ibc.gov.br/images/conteudo/AREAS_ESPECIAIS/CEGUEIRA_E_BAIXA_VISAO/ARTIGOS/Def-de-cegueira-e-baixa-viso.pdf. Acesso em: 03 set. 2021.

CRYPTZONE NORTH AMERICA (Estados Unidos da América). **Compliance Sheriff Cynthia Say Portal**. 2021. Disponível em: http://www.cynthiasays.com/home.aspx. Acesso em: 11 nov. 2021.

CUSIN, Cesar Augusto; VIDOTTI, Silvana Aparecida Borsetti Gregorio. **Inclusão digital via acessibilidade Web**. **Liinc**, Rio de Janeiro, v. 5, n. 1, p. 45-65, mar. 2019. Disponível em: <http://revista.ibict.br/liinc/article/view/3189/2851>. Acesso em: 22 ago. 2021.

CWEB (comp.). **CARTILHA ACESSIBILIDADE NA *WEB***: w3c brasil. W3C BRASIL. 2020. Disponível em: https://ce*Web*.br/cartilhas/cartilha-w3cbr-acessibilidade-web-fasciculo-IV/. Acesso em: 25 jul. 2021.

DARDAILLER, Daniel. **WAI early days**. 2009. Disponível em: <https://www.w3.org/WAI/history>. Acesso em: 29 ago. 2021.

EGGERT, Eric. ***Web* Accessibility Evaluation Tools List**. 2016. Disponível em: <https://www.w3.org/WAI/ER/tools/>. Acesso em: 07 set. 2021.

ESPINHA, Roberto Gil. **Você realmente sabe o que é um projeto? Descubra o conceito, os principais tipos e as fases de um projeto**. 2020. Disponível em: https://artia.com/blog/o-que-e-um-projeto/. Acesso em: 08 ago. 2021.

FEATURED CUSTOMERS (Sunrise, FL). **21 Companies that are using Cypress**. 2021. Disponível em: <https://www.tawdis.net/index>. Acesso em: 23 nov. 2021.

FENNER, Priscila. **Entenda o WCAG 2.0 de forma simples e rápida**. 2021. Disponível em: <https://blog.handtalk.me/wcag-2-0/>. Acesso em: 20 ago. 2021.

FLANAGAN, David. **JavaScript**: o guia definitivo. 6. ed. S.I: O'Reilly Media, 2013.

FRISBIE, Matt. **Professional JavaScript for *Web* Developers**. 4. ed. S.I: Wrox Press, 2019.

HARPER, Simom; YESILADA, Yeliz. ***Web* Accessibility**: a foundation for research. University Of Manchester, Uk: Springer, 2008. 364 p.

HARVEY, Greg. **Automated accessibility testing with Google LHCI**. 2021. Disponível em: <https://www.codeenigma.com/blog/automated-accessibility-testing-google-lhci>. Acesso em: 16 nov. 2021.

HASSANZADEH, Mohammad; NAVIDI, Fatemeh. *Web* site accessibility evaluation methods in action. **The Electronic Library**, Tehran, v. 28, n. 6, p. 789-803, 16 nov. 2010. Disponível em: <https://www.emerald.com/insight/content/doi/10.1108/02640471011093499/full/pdf?title=*Web*-site-accessibility-evaluation-methods-in-action-a-comparative-approach-for-ministerial-*Web*-sites-in-iran>. Acesso em: 07 set. 2021.

HENRY, Brad. **CynthiaSays.com Accessibility *Web*site Scan Announcement**. 2021. Disponível em: <https://www.tpgi.com/cynthiasays-com-accessibility-*Web*site-scan-announcement/>. Acesso em: 11 nov. 2021.

HENRY, Shawn Lawton. **Easy Checks – A First Review of *Web* Accessibility**. 2018. Disponível em: <https://www.w3.org/WAI/fundamentals/components/#relate>. Acesso em: 29 ago. 2021.

HENRY, Shawn Lawton. **Essential Components of *Web* Accessibility**. 2017. Disponível em: < https://www.w3.org/WAI/test-evaluate/preliminary/>. Acesso em: 07 set. 2021.

IBGE. **Censo 2010**: pessoas com deficiência. Pessoas com deficiência. 2010. Disponível em: https://educa.ibge.gov.br/jovens/conheca-o-brasil/populacao/20551-pessoas-com-deficiencia.html. Acesso em: 25 jul. 2021.

IBGE. **Pesquisa mostra que 82,7% dos domicílios brasileiros têm acesso à *Internet***. 2019. Disponível em: https://www.gov.br/mcom/pt-br/noticias/2021/abril/pesquisa-mostra-que-82-7-dos-domicilios-brasileiros-tem-acesso-a-*Internet*. Acesso em: 25 jul. 2021.

IDRC (Canadá). **Achecker**. 2021. Disponível em: https://achecker.achecks.ca/checker/index.php. Acesso em: 22 nov. 2021.

IFPB. **Cegueira x baixa visão**. 2018. Disponível em: <https://www.ifpb.edu.br/assuntos/fique-por-dentro/cegueira-x-baixa-visao. Acesso em: 03 set>. 2021.

IFRS, **Avaliador automático de acessibilidade AccessMonitor**. 2021. Disponível em: <https://cta.ifrs.edu.br/avaliador-automatico-de-acessibilidade-accessmonitor/>. Acesso em: 11 nov. 2021.

ISMAIL, Abid; KUPPUSAMY, K.s.. *Web* accessibility investigation and identification of major issues of higher education *Web*sites with statistical measures: a case study of college *Web*sites. **Journal Of King Saud University - Computer And Information Sciences**, Arábia Saudita, v. 1, n. 1, p. 1-11, jan. 2019. Elsevier BV. http://dx.doi.org/10.1016/j.jksuci.2019.03.011. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1319157818312394?via%3Dihub>. Acesso em: 23 nov. 2021.

KANERIYA, Tejas. **What is Node.js?**: where, when & how to use it (with examples). Where, When & How To Use It (With Examples). 2020. Disponível em: https://www.simform.com/blog/what-is-node-js/#section4. Acesso em: 06 dez. 2021.

KATZ, Gregorio; LAZCANO-PONCE, Eduardo. Intellectual disability: definition, etiological factors, classification, diagnosis, treatment and prognosis. **Medigraphic**, Cuautitlán Izcalli, v. 50, n. 1, p. 132-141, 02 out. 2008. Disponível em: <https://www.medigraphic.com/pdfs/salpubmex/sal-2008/sals082e.pdf>. Acesso em: 03 set. 2021.

KHETARPAL, Aashish. **What is Cypress**: introduction and architecture. Introduction and Architecture. 2021. Disponível em: <https://www.toolsqa.com/cypress/what-is-cypress/>. Acesso em: 07 dez. 2021.

KITCHENHAM, Ba; CHARTERS, Stuart M.. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. 2007. Disponível em: https://www.researchgate.net/publication/302924724_Guidelines_for_performing_Systematic_Literature_Reviews_in_Software_Engineering. Acesso em: 08 ago. 2021.

KUMARI, Pratibha; VERMA, Shilpi. *Web*site Accessibility Evaluation of National Institutes under the DEPWD Ministry of Social Justice & Empowerment. **. Library Philosophy And Practice (Ejournal).** Lucknow, Índia, p. 4578-4578. 05 dez. 2020. Disponível em: <https://digitalcommons.unl.edu/libphilprac/4578/>. Acesso em: 22 nov. 2021.

LOUREIRO, Janaína Rolan. **Acessibilidade *Web* em Redes Sociais**. 2014. 212 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Universidade Federal de Mato Grosso do Sul, Campo Grande/Ms, 2014. Disponível em: https://repositorio.ufms.br/handle/123456789/2206. Acesso em: 01 ago. 2021.

MAIOR, Izabel. **História, conceito e tipos de deficiência**. 2021. Disponível em: <http://www.deficienciavisual.pt/txt-Historia_conceito_tipos_def.htm>. Acesso em: 16 ago. 2021.

MORAES, William Bruno. Introdução. In: MORAES, William Bruno. **Construindo aplicações com NodeJS**. 3. ed. São Paulo: Novatec, 2021. Cap. 1. p. 13-14. Disponível em: <https://www.google.com.br/books/edition/Construindo_aplica%C3%A7%C3%B5es_com_NodeJS_3%C2%AA/iMMqEAAAQBAJ>. Acesso em: 06 dez. 2021.
MPPR. **Conceitos de Deficiência**. Disponível em: <https://pcd.mppr.mp.br/pagina-41.html>. Acesso em: 15 ago. 2021.

RACHADEL, Rodrigo Augusto. **Acessibilidade *WEB***: uma análise sobre suas ferramentas e diretrizes. 2017. 82 f. TCC (Graduação) - Curso de Línguas Estrangeiras Aplicadas Ao Multilinguismo e À Sociedade da Informação, Fundação Universidade de Brasília, Brasília, 2017. Disponível em: <https://bdm.unb.br/bitstream/10483/19460/1/2017_RodrigoAugustoRachadel.pdf>. Acesso em: 22 ago. 2021.

VIGO, Markel *et al*. Benchmarking *Web* accessibility evaluation tools. **Proceedings Of The 10Th International Cross-Disciplinary Conference On *Web* Accessibility - W4A '13**, Rio de Janeiro, v. 1, n. 1, p. 1-10, maio 2013. Disponível em: <https://dl.acm.org/doi/10.1145/2461121.2461124. Acesso em: 07 set. 2021>.

W3C (Estados Unidos). ***Web* Content Accessibility Guidelines 1.0**. 1999. Disponível em: <https://www.w3.org/TR/WAI-*WEB*CONTENT/>. Acesso em: 29 ago. 2021.

W3C. **Conhecendo o W3C**. 2021. Disponível em: <https://www.w3c.br/Sobre/ConhecendoW3C>. Acesso em: 22 ago. 2021.

MWAURA, Waweru. Differences between Selenium *Web*Driver and Cypress: why choose cypress?. In: MWAURA, Waweru. **End-to-End *Web* Testing with Cypress**. Brimingham, Uk: Packt Publishing, 2021. Cap. 2. p. 16-17. Disponível em: <https://www.google.com.br/books/edition/End_to_End_*Web*_Testing_with_Cypress/Er0YEAAAQBAJ. Acesso em: 07 dez>. 2021.

OLIVEIRA, Cynthia *et al*. **Avaliação da usabilidade e da acessibilidade do portal da transparência do governo do estado de Pernambuco**. In: Encontro Nacional De Estudantes De Biblioteconomia, Documentação, Ciência e Gestão Da Informação (ENEDB), 41., 2018, Rio de Janeiro. **Anais [...] .** Rio de Janeiro: Unirio, 2018. p. 205-227. Disponível em: <https://app.uff.br/riuff/bitstream/handle/1/14469/205.pdf?sequence=1&isAllowed=y>. Acesso em: 11 nov. 2021.

OPENJS FOUNDATION. **What is npm?** 2011. Disponível em: https://nodejs.org/en/knowledge/getting-started/npm/what-is-npm/. Acesso em: 06 dez. 2021.

PEREIRA, Caio Ribeiro. E assim nasceu o Node.JS. In: PEREIRA, Caio Ribeiro. **Aplicações *Web* real-time com Node.js**. São Paulo: Casa do Código, 2014. Cap. 1. p. 6-7.

PLAINER, Michael. **Study of Visual Studio Code**. 2020. Disponível em: <https://www21.in.tum.de/teaching/osp/WS20/assets/pr-plainer-vscode.pdf>. Acesso em: 13 dez. 2021.

PLAUCA, Denis. **Isabelle/VSCode**: editor improvements and prover ide integrations. 2021. 32 f. TCC (Graduação) - Curso de Bachelor 's Thesis In Informatics, Department Of Informatics, Technische Universität München, Munich, 2021. Disponível em: <https://www21.in.tum.de/students/past/vscode_plugin_improvements/assets/Isabelle_VSCode_Thesis.pdf>. Acesso em: 13 dez. 2021.

RODRIGUEZ, Y. S *et al*. Estado de la accesibilidad *Web* de los portales de gobierno electrónico en América latina. **Bibliotecas. Anales de Investigación (Cuba)**, v. 16, p. 7-22, 2020. Disponível em: <http://hdl.handle.net/20.500.11959/brapci/159177>. Acesso em: 22 nov. 2021.

ROIG-VILA, Rosabel *et al*. Assessment of *Web* Content Accessibility Levels in Spanish Official Online Education Environments. **International Education Studies**, Alicante, v. 7, n. 6, p. 31-45, 20 maio 2014. Canadian Center of Science and Education. http://dx.doi.org/10.5539/ies.v7n6p31. Disponível em: <https://www.ccsenet.org/journal/index.php/ies/article/view/34547>. Acesso em: 23 nov. 2021.

SHARMA, Anita. *Web* Accessibility of Indian University Library *Web*site: An Evaluation with WAVE *Web*site Evaluation Tool. **Library Philosophy And Practice (Ejournal).** Motihari, Índia, p. 5844-5844. jul. 2021. Disponível em: <https://digitalcommons.unl.edu/libphilprac/5844/>. Acesso em: 21 nov. 2021.

TAL, Liran; MAPLE, Simon. **Npm passes the 1 millionth package milestone! What can we learn?** 2019. Disponível em: https://snyk.io/blog/npm-passes-the-1-millionth-package-milestone-what-can-we-learn/. Acesso em: 06 dez. 2021.
TAW (Canadá). **TAW**. 2021. Disponível em: <https://www.tawdis.net/index>. Acesso em: 23 nov. 2021.

TENON. **Accessibility as a Service**. 2021. Disponível em: https://tenon.io/. Acesso em: 16 nov. 2021.

UNIVERSITY OF MINNESOTA (USA). **Accessibility 102: The WAVE Accessibility Tool**. 2021. Disponível em: <https://www.d.umn.edu/itss/training/online/wave/>. Acesso em: 21 nov. 2021.

W3C. **Current members & testimonials**. 2021. Disponível em: <https://www.w3.org/Consortium/Member/List>. Acesso em: 22 ago. 2021.

W3C. **Facts about w3c**. 2021. Disponível em: <https://www.w3.org/Consortium/facts>. Acesso em: 22 ago. 2021.

W3C. **Introduction to *Web* Accessibility**. 2019. Disponível em: <https://www.w3.org/WAI/fundamentals/accessibility-introt>. Acesso em: 15 ago. 2021.

W3C. **Sobre o W3C**. 2021. Disponível em: <https://www.w3c.br/Sobre/>. Acesso em: 22 ago. 2021.

WAI EOWG. **How to Update Your *Web* Site from WCAG 1.0 to WCAG 2.0**. 2021. Disponível em: <https://www.w3.org/WAI/WCAG20/from10/*Web*sites.php>. Acesso em: 03 nov. 2021.

WAVE. **Wave**. Disponível em: https://wave.*Web*aim.org/. Acesso em: 24 nov. 2021.

WAZLAWICK, Raul Sidnei. **Metodologia de Pesquisa para Ciência da Computação**. 3. ed. Rio de Janeiro: Grupo Editorial Nacional S.A., 2020. 129 p.

APÊNDICE A – RELATÓRIOS DE EXECUÇÃO DOS TESTES

Página Sábio - http://sabio.biblioteca.sc.gov.br/sabio/

Accessibility evaluation of *Web* pages

Number of violations: 6 types of Wcag violations detected.

(Index)	Id of Axe rule violated	User impact	Description of rule	Number of nodes affected
1	button-name	critical	Ensures buttons have discernible text	2
2	frame-title	serious	Ensures <iframe> and <frame> elements have an accessible name	2
3	html-has-lang	serious	Ensures every HTML document has a lang attribute	1
4	image-alt	critical	Ensures <img> elements have alternate text or a role of none or presentation	26
5	label	critical	Ensures every form element has a label	1
6	scrollable-region-focusable	moderate	Ensure elements that have scrollable content are accessible by keyboard	1

######## Violation 1 of 6: 'button-name', whit User impact critical ########

Buttons must have discernible text. Access for more details: https://dequeuniversity.com/rules/axe/4.3/button-name?application=axeAPI

2 instances of this violation:

Violation 1 instance 1:
<button class="x-btn-text" type="button" id="ext-gen226" style="background-image: url("http://sabio.biblioteca.sc.gov.br/sabio/modulo.sabio*Web*/imagens/previous.png");"> </button>

Violation 1 instance 2:
<button class="x-btn-text" type="button" id="ext-gen234" style="background-image: url("http://sabio.biblioteca.sc.gov.br/sabio/modulo.sabio*Web*/imagens/next.png"); "> </button>

######## Violation 2 of 6: 'frame-title', whit User impact serious ########

Frames must have an accessible name. Access for more details: https://dequeuniversity.com/rules/axe/4.3/frame-title?application=axeAPI

2 instances of this violation:

Violation 2 instance 1:
<iframe id="__gwt_historyFrame" style="width: 0; height: 0; border: 0"></iframe>

Violation 2 instance 2:
<iframe src="javascript:"" id="modulo.sabio*Web*" tabindex="-1" style="position: absolute; width: 0px; height: 0px; border: none;"></iframe>

######## Violation 3 of 6: 'html-has-lang', whit User impact serious ########

<html> element must have a lang attribute. Acess for more details: https://dequeuniversity.com/rules/axe/4.3/html-has-lang?application=axeAPI

1 instance of this violation:

Violation 3 instance 1: <html>

######## Violation 4 of 6: 'image-alt', whit User impact critical ########

Images must have alternate text. Access for more details: https://dequeuniversity.com/rules/axe/4.3/image-alt?application=axeAPI

26 instances of this violation:

Violation 4 instance 1:

<img src="http://sabio.biblioteca.sc.gov.br/sabio/modulo.sabio*Web*/clear.cache.gif" class="x-form-trigger x-form-arrow-trigger" id="ext-gen275">

Violation 4 instance 2:
<img src="fotoobra/100430.JPG?showDefaultImage=true">

Violation 4 instance 3:
<img src="fotoobra/100431.JPG?showDefaultImage=true">

Violation 4 instance 4:
<img src="fotoobra/100428.JPG?showDefaultImage=true">

Violation 4 instance 5:
<img src="fotoobra/100432.JPG?showDefaultImage=true">

Violation 4 instance 6:
<img src="fotoobra/100424.JPG?showDefaultImage=true">

Violation 4 instance 7:
<img src="fotoobra/100429.JPG?showDefaultImage=true">

Violation 4 instance 8:
<img src="fotoobra/100427.JPG?showDefaultImage=true">

Violation 4 instance 9:
<img src="fotoobra/100425.JPG?showDefaultImage=true">

Violation 4 instance 10:
<img src="fotoobra/100426.JPG?showDefaultImage=true">

Violation 4 instance 11:
<img src="fotoobra/100422.JPG?showDefaultImage=true">

Violation 4 instance 12:
<img src="fotoobra/100420.JPG?showDefaultImage=true">

Violation 4 instance 13:
<img src="fotoobra/100423.JPG?showDefaultImage=true">

Violation 4 instance 14:
<img src="fotoobra/100421.JPG?showDefaultImage=true">

Violation 4 instance 15:
<img src="fotoobra/100419.JPG?showDefaultImage=true">

Violation 4 instance 16:
<img src="fotoobra/100418.JPG?showDefaultImage=true">

Violation 4 instance 17:
<img src="fotoobra/100414.JPG?showDefaultImage=true">

Violation 4 instance 18:
<img src="fotoobra/100416.JPG?showDefaultImage=true">

Violation 4 instance 19:
<img src="fotoobra/100417.JPG?showDefaultImage=true">

Violation 4 instance 20:
<img src="fotoobra/100415.JPG?showDefaultImage=true">

Violation 4 instance 21:
<img src="fotoobra/100411.JPG?showDefaultImage=true">

Violation 4 instance 22:
<img src="fotoobra/100409.JPG?showDefaultImage=true">

Violation 4 instance 23:
<img src="fotoobra/100412.JPG?showDefaultImage=true">

Violation 4 instance 24:
<img src="fotoobra/100413.JPG?showDefaultImage=true">

Violation 4 instance 25:
<img src="fotoobra/100410.JPG?showDefaultImage=true">

Violation 4 instance 26:
<img src="fotoobra/100408.JPG?showDefaultImage=true">

######## Violation 5 of 6: 'label', whit User impact critical ########

Form elements must have labels. Access for more details: https://dequeuniversity.com/rules/axe/4.3/label?application=axeAPI

1 instance of this violation:

Violation 5 instance 1:
<input type="text" size="24" autocomplete="off" id="ext-gen189" name="ext-gen189" class=" x-form-text x-form-field " style="width: 43px;">

######## Violation 6 of 6: 'scrollable-region-focusable', whit User impact moderate ########

Scrollable region must have keyboard access. Access for more details: https://dequeuniversity.com/rules/axe/4.3/scrollable-region-focusable?application=axeAPI

1 instance of this violation:

Violation 6 instance 1:

Página portal do Governo de Santa Catarina – https://www.sc.gov.br

Accessibility evaluation of *Web* pages

Number of violations: 5 types of Wcag violations detected.

(Index)	Id of Axe rule violated	User impact	Description of rule	Number of nodes affected
0	color-contrast	serious	Ensures the contrast between foreground and background colors meets WCAG 2 AA contrast ratio thresholds	6
1	duplicate-id-active	serious	Ensures every id attribute value of active elements is unique	1
2	duplicate-id	minor	Ensures every id attribute value is unique	1
3	html-lang-valid	serious	Ensures <img> elements have alternate text or a role of none or presentation	1
4	link-name	serious	Ensures links have discernible text	10

######## Violation 1 of 5: 'color-contrast', whit User impact serious ########

Elements must have sufficient color contrast. Access for more details: https://dequeuniversity.com/rules/axe/4.3/color-contrast?application=axeAPI

6 instances of this violation:

Violation 1 instance 1:
Buscar

Violation 1 instance 2:
12 Janeiro 2022

Violation 1 instance 3:
Ciência e Tecnologia

Violation 1 instance 4:
12 Janeiro 2022

Violation 1 instance 5:
Desenvolvimento Econômico

Violation 1 instance 6:
Download

######## Violation 2 of 5: 'duplicate-id-active', whit User impact serious ########

IDs of active elements must be unique. Access for more details: https://dequeuniversity.com/rules/axe/4.3/duplicate-id-active?application=axeAPI

1 instance of this violation:

Violation 2 instance 1:
<input id="rsf_inp695" class="search-query input-medium" name="q" size="25" type="text" value="" placeholder="O que você procura: Exemplo: matrícula escolar, portal do servidor, IPVA...">

######## Violation 3 of 5: 'duplicate-id', whit User impact minor ########

id attribute value must be unique. Access for more details: https://dequeuniversity.com/rules/axe/4.3/duplicate-id?application=axeAPI

1 instance of this violation:

Violation 3 instance 1:
<form id="mod-finder-searchform" action="/buscar-tudo" method="get" class="form-search">

######## Violation 4 of 5: 'html-lang-valid', whit User impact serious ########

<html> element must have a valid value for the lang attribute. Access for more details: https://dequeuniversity.com/rules/axe/4.3/html-lang-valid?application=axeAPI

1 instance of this violation:

Violation 4 instance 1:
<html lang="*" dir="ltr" slick-uniqueid="3">

######## Violation 5 of 5: 'link-name', whit User impact serious ########

Links must have discernible text. Access for more details: https://dequeuniversity.com/rules/axe/4.3/link-name?application=axeAPI

10 instances of this violation:

Violation 5 instance 1:

Violation 5 instance 2:
<img src="" alt="" title="Encontre-nos no ">

Violation 5 instance 3:
<img src="" alt="" title="Encontre-nos no ">

Violation 5 instance 4:
<img src="" alt="" title="Encontre-nos no ">

Violation 5 instance 5:
<img src="" alt="" title="Encontre-nos no ">

Violation 5 instance 6:
<img src="" alt="Flickr" title="Encontre-nos no Flickr">

Violation 5 instance 7:
<a href="/noticias/temas/educacao-noticias/decreto-autoriza-aulas-100-presenciais-para-todos-os-estudantes-em-sc">

Violation 5 instance 8:

Violation 5 instance 9:
<a href="/noticias/temas/ciencia-e-tecnologia/programa-catarinense-de-inovacao-no-turismo-e-destaque-em-evento-internacional">

Violation 5 instance 10:
<a href="/noticias/temas/desenvolvimento-economico/brde-contrata-r-400-milhoes-para-micro-e-pequenos-negocios-de-santa-catarina">

Pergamum UFSC – https://pergamum.ufsc.br
Accessibility evaluation of *Web* pages

Number of violations: 9 types of Wcag violations detected.

(Index)	Id of Axe rule violated	User impact	Description of rule	Number of nodes affected

0	color-contrast	serious	Ensures the contrast between foreground and background colors meets WCAG 2 AA contrast ratio thresholds	1
1	duplicate-id-active	serious	Ensures every id attribute value of active elements is unique	1
1	duplicate-id	minor	Ensures every id attribute value is unique	1
3	html-has-lang	serious	Ensures every HTML document has a lang attribute	1
4	image-alt	critical	Ensures <img> elements have alternate text or a role of none or presentation	4
5	label	critical	Ensures every form element has a label	4
6	link-name	serious	Ensures links have discernible text	2
7	object-alt	serious	Ensures <object> elements have alternate text	1
8	select-name	critical	Ensures select element has an accessible name	6

######## Violation 1 of 9: 'color-contrast', whit User impact serious ########

Elements must have sufficient color contrast. Access for more details: https://dequeuniversity.com/rules/axe/4.3/color-contrast?application=axeAPI

1 instance of this violation:

Violation 1 instance 1:
<legend class="txt_acervo_legend" style="color: rgb(53, 120, 171);">Pesquisa geral</legend>

######## Violation 2 of 9: 'duplicate-id-active', whit User impact serious ########

IDs of active elements must be unique. Access for more details: https://dequeuniversity.com/rules/axe/4.3/duplicate-id-active?application=axeAPI

1 instance of this violation:

Violation 2 instance 1:

<input type="radio" name="tipo_pesquisa" id="tipo_pesquisa" value="Palavra" checked=""
onclick="document.pesquisas.codigo_autoridade.value=''; tipo_pesq('palavra','');">

######## Violation 3 of 9: 'duplicate-id', whit User impact minor ########

id attribute value must be unique. Access for more details:
https://dequeuniversity.com/rules/axe/4.3/duplicate-id?application=axeAPI

1 instance of this violation:

Violation 3 instance 1:
Fechar(X)

######## Violation 4 of 9: 'html-has-lang', whit User impact serious ########

<html> element must have a lang attribute. Access for more details:
https://dequeuniversity.com/rules/axe/4.3/html-has-lang?application=axeAPI

1 instance of this violation:

Violation 4 instance 1:
<html xmlns="http://www.w3.org/1999/xhtml">

######## Violation 5 of 9: 'image-alt', whit User impact critical ########

Images must have alternate text. Access for more details:
https://dequeuniversity.com/rules/axe/4.3/image-alt?application=axeAPI

4 instances of this violation:

Violation 5 instance 1:
<img border="0" src="../img/logo_empresa.jpg">

Violation 5 instance 2:
<img src="../img/logo_inf_meu_pergamum.png" border="0">

Violation 5 instance 3:
<img src="../img/logo_inf_barra.png" width="3" height="48">

Violation 5 instance 4:
<img src="../img/logo_inf_icap.png" border="0">

######## Violation 6 of 9: 'label', whit User impact critical ########

Form elements must have labels. Access for more details:
https://dequeuniversity.com/rules/axe/4.3/label?application=axeAPI

4 instances of this violation:

Violation 6 instance 1:
<input name="termo_para_pesquisa" type="text" class="pmu_campo" id="termo_para_pesquisa" value="" maxlength="255" autocomplete="off" onfocus="this.className='pmu_campo2';if(this.value.trim()=='Digite o termo para a pesquisa')this.value=";" onblur="this.className='pmu_campo';if(this.value.trim()==")this.value='Digite o termo para a pesquisa';">

Violation 6 instance 2:
<input type="radio" name="tipo_pesquisa" id="tipo_pesquisa" value="Palavra" checked="" onclick="document.pesquisas.codigo_autoridade.value="; tipo_pesq('palavra',");">

Violation 6 instance 3:
<input type="radio" name="tipo_pesquisa" id="tipo_pesquisa" value="Índice" onclick="document.pesquisas.codigo_autoridade.value="; tipo_pesq('indice',");">

Violation 6 instance 4:
<input name="ano_publicacao" type="text" class="pmu_campo3" id="ano_publicacao" size="3" maxlength="4">

######## Violation 7 of 9: 'link-name', whit User impact serious ########

Links must have discernible text. Access for more details: https://dequeuniversity.com/rules/axe/4.3/link-name?application=axeAPI

2 instances of this violation:

Violation 7 instance 1:

Violation 7 instance 2:
<img src="../img/logo_inf_icap.png" border="0">

######## Violation 8 of 9: 'object-alt', whit User impact serious ########

<object> elements must have alternate text. Access for more details: https://dequeuniversity.com/rules/axe/4.3/object-alt?application=axeAPI

1 instance of this violation:

Violation 8 instance 1:
<object id="BemaPrinter1" height="0" width="0" border="0" classid="clsid:310DBDAC-85FF-4008-82A8-E22A09F9460B" viewastext=""></object>

######## Violation 9 of 9: 'select-name', whit User impact critical ########

Select element must have an accessible name. Access for more details:
https://dequeuniversity.com/rules/axe/4.3/select-name?application=axeAPI

6 instances of this violation:

Violation 9 instance 1:
<select name="modo_ordenar" class="pmu_campo3" id="modo_ordenar">

Violation 9 instance 2:
<select name="osbairros2" class="pmu_campoSeletores">
<option>Todas</option>
</select>

Violation 9 instance 3:
<select name="codMat" id="codMat" class="pmu_campo3">
<option value="T">Título</option>
<option value="S">Assunto</option>
<option value="A">Autor</option>
<option value="L" selected="">Livre</option>
</select>

Violation 9 instance 4:
<select name="osbairros" class="pmu_campoSeletores" id="osbairros"
onclick="janela_obras('');">
<option id="id_filtro_obra_text">Todas</option>
</select>

Violation 9 instance 5:
<select name="n_registros_por_pagina" class="pmu_campo3" id="n_registros_por_pagina">
<option value="20">20</option>
<option value="30">30</option>
<option value="40">40</option>
<option value="50">50</option>

</select>

Violation 9 instance 6:

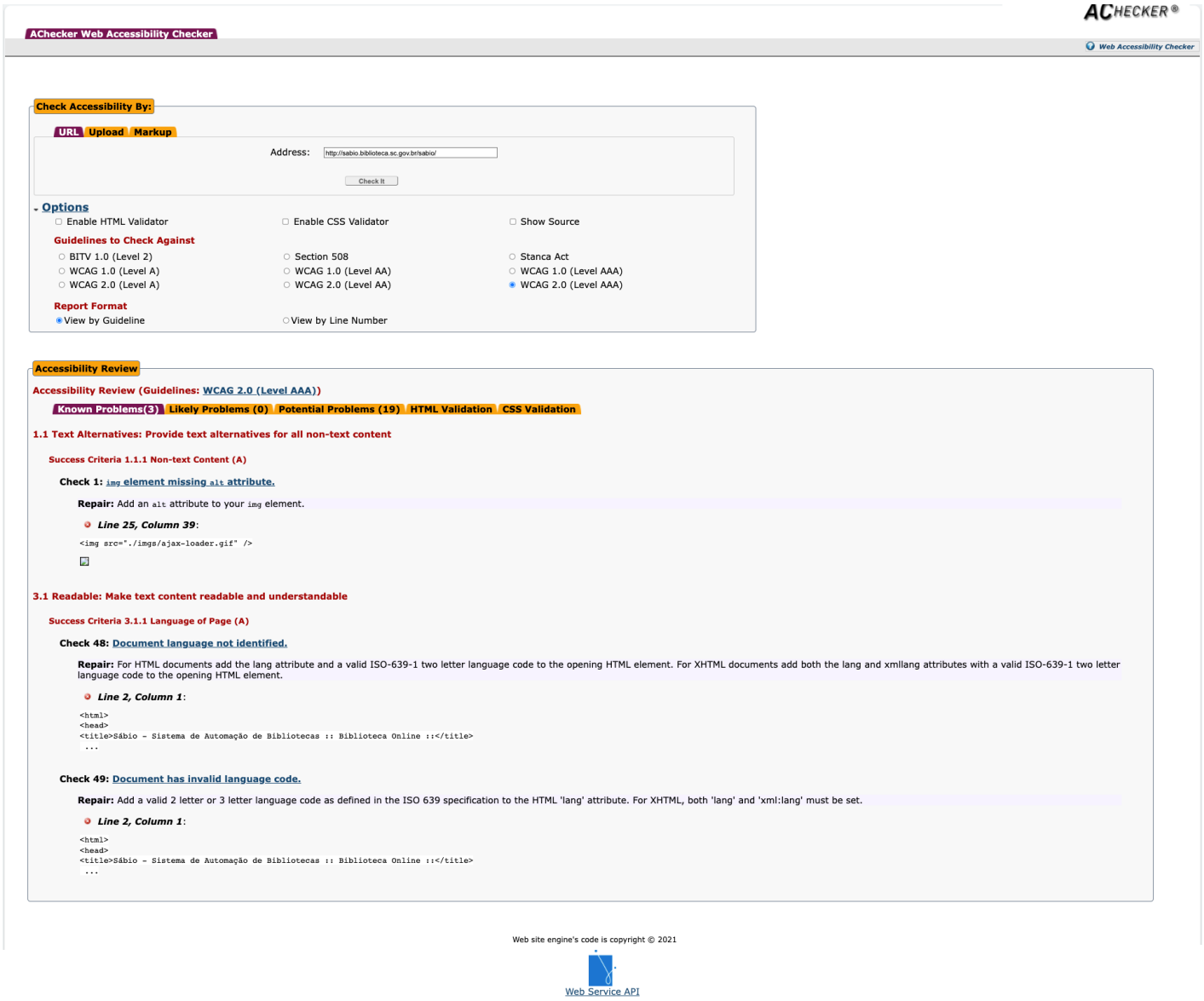
<select name="osbairros3" class="pmu_campoSeletores">

<option>Todas</option>

</select>

APÊNDICE B – CAPTURAS DE TELA DA EXECUÇÃO NA FERRAMENTA ACHECKER

Página Sábio - http://sabio.biblioteca.sc.gov.br/sabio/



Página portal governo de Santa Catarina - https://www.sc.gov.br/

AChecker Web Accessibility Checker

Web Accessibility Checker

Check Accessibility By:

URL Upload Markup

Address: https://www.sc.gov.br/

Check It

Options

Accessibility Review

Accessibility Review (Guidelines: WCAG 2.0 (Level AA))

Known Problems(51) Likely Problems (3) Potential Problems (289) HTML Validation CSS Validation

1.1 Text Alternatives: Provide text alternatives for any non-text content

Success Criteria 1.1.1 Non-text Content (A)

Check 7: Image used as anchor is missing valid Alt text.

Repair: Add Alt text that identifies the purpose or function of the image.

- *Line 127, Column 185*:

```
<img src="https://www.sc.gov.br/https://www.sc.gov.br/modules/mod_socialmedialinks/icons/default/siz ...
```

- *Line 127, Column 418*:

```
<img src="https://www.sc.gov.br/https://www.sc.gov.br/modules/mod_socialmedialinks/icons/default/siz ...
```

- *Line 127, Column 655*:

```
<img src="https://www.sc.gov.br/https://www.sc.gov.br/modules/mod_socialmedialinks/icons/default/siz ...
```

- *Line 127, Column 895*:

```
<img src="https://www.sc.gov.br/https://www.sc.gov.br/modules/mod_socialmedialinks/icons/default/siz ...
```

- *Line 317, Column 161*:

```
<img src="/modules/mod_raxo_allmode/tools/tb.php?src=/images/joomgallery/details/setoriais_6457/rodo ...
```

- *Line 355, Column 170*:

```
<img src="/modules/mod_raxo_allmode/tools/tb.php?src=/images/joomgallery/details/setoriais_6457/defe ...
```

- *Line 382, Column 168*:

```
<img src="/modules/mod_raxo_allmode/tools/tb.php?src=/images/joomgallery/details/setoriais_6457/gove ...
```

1.3 Adaptable: Create content that can be presented in different ways (for example simpler layout) without losing information or structure.

Success Criteria 1.3.1 Info and Relationships (A)

Check 213: `input` element, `type` of "text", has no text in `label`.

Repair: Add text to the `input` element's associated label that describes the purpose or function of the control.

- ***Line 261, Column 21***:

```
<input id="mod-finder-searchword" class="search-query input-medium" name="q" size="25" type="text" v ...
```

- ***Line 476, Column 3***:

```
<input type="text" name="q" id="mod-finder-searchword" class="search-query input-medium" size="20" v ...
```

- ***Line 482, Column 21***:

```
<input id="mod-finder-searchword" class="search-query input-medium" name="q" size="25" type="text" v ...
```

Check 57: `input` **element,** `type` **of "text", missing an associated label.**

Repair: Add a `label` element that surrounds the control's `label`. Set the `for` attribute on the `label` element to the same value as the `id` attribute of the control. And/or add a `title` attribute to the `input` element. And/or create a `label` element that contains the `input` element.

- ***Line 123, Column 3***:

```
<input type="text" name="q" id="mod-finder-searchword" class="search-query input-medium" size="25" v ...
```

- ***Line 261, Column 21***:

```
<input id="mod-finder-searchword" class="search-query input-medium" name="q" size="25" type="text" v ...
```

- ***Line 476, Column 3***:

```
<input type="text" name="q" id="mod-finder-searchword" class="search-query input-medium" size="20" v ...
```

- ***Line 482, Column 21***:

```
<input id="mod-finder-searchword" class="search-query input-medium" name="q" size="25" type="text" v ...
```

1.4 Distinguishable: Make it easier for users to see and hear content including separating foreground from background.

Success Criteria 1.4.3 Contrast (Minimum) (AA)

Check 304: The contrast between the colour of selected link text and its background is not sufficient to meet WCAG2.0 Level AA.

Repair: Use a colour contrast evaluator to determine if selected link text and background colours provide a contrast ratio of 4.5:1 for standard text, or 3:1 for larger text. Change colour codes to produce sufficient contrast. http://www.w3.org/TR/UNDERSTANDING-WCAG20/visual-audio-contrast-contrast.html#visual-audio-contrast-contrast-resources-head

- ***Line 573, Column 32***:

```
<a href="http://www.portaldecompras.sc.gov.br/" target="_blank">IR PARA O SITE</a>
```

Fixed size example: color contrast example

Real size example (10.61 points): color contrast example

CSS rules for the element:

- ***Line 588, Column 32***:

```
<a href="/oportunidades/invistaemsc?view=default" target="_self">EXPLORAR</a>
```

Fixed size example: color contrast example

Real size example (10.61 points): color contrast example

CSS rules for the element:

- ***Line 603, Column 32***:

```
<a href="/oportunidades/desenvolvimento/praquemvivenocampo?view=default" target="_self">EXPLORAR</a>
```

Fixed size example: color contrast example

Real size example (10.61 points): color contrast example

CSS rules for the element:

Check 301: **The contrast between the colour of text and its background for the element is not sufficient to meet WCAG2.0 Level AA.**

Repair: Use a colour contrast evaluator to determine if text and background colours provide a contrast ratio of 4.5:1 for standard text, or 3:1 for larger text. Change colour codes to produce sufficient contrast. http://www.w3.org/TR/UNDERSTANDING-WCAG20/visual-audio-contrast-contrast.html#visual-audio-contrast-contrast-resources-head

- ***Line 127, Column 4***:

```
<span style="color: #666666;">Conecte-se <div class="smile" style="text-align: center "> <a style="m ...
```

Fixed size example: color contrast example

Real size example (10.61 points): color contrast example

CSS rules for the element:

- ***Line 438, Column 394***:

```
<span class="rl_sliders-toggle-inner nn_sliders-toggle-inner"> Caminhos da Esperança: a rota da vac ...
```

Fixed size example: color contrast example

Real size example (10.61 points): color contrast example

CSS rules for the element:

- ***Line 572, Column 40***:

```
<span style="font-size: 18pt;">Portal de Compras</span>
```

Fixed size example: color contrast example

Real size example (18 points): color contrast example

CSS rules for the element:

- ***Line 602, Column 40***:

```
<span style="font-size: 18pt;">Desenvolvimento</span>
```

Fixed size example: color contrast example

Real size example (18 points): color contrast example

CSS rules for the element:

Success Criteria 1.4.4 Resize text (AA)

Check 117: **`i` (italic) element used.**

Repair: Replace your `i` elements with `em` or `strong`.

- ***Line 117, Column 145***:

```
<i class=""></i>
```

- ***Line 133, Column 169***:

```
<i class="fa fa-navicon"></i>
```

- ***Line 135, Column 63***:

```
<i class="fa fa-times"></i>
```

- ***Line 137, Column 4***:

```
<i class=""></i>
```

- ***Line 144, Column 35***:

```
<i class="fa fa-navicon"></i>
```

- ***Line 168, Column 35***:

```
<i class="fa fa-navicon"></i>
```

- ***Line 193, Column 32***:

- ***Line 193, Column 32***:

```
<i class="fa fa-navicon"></i>
```

- ***Line 194, Column 4***:

```
<i class=""></i>
```

- ***Line 209, Column 32***:

```
<i class="fa fa-navicon"></i>
```

- ***Line 261, Column 252***:

```
<i class="icon-search icon-white"></i>
```

- ***Line 476, Column 188***:

```
<i class="icon-search icon-white"></i>
```

- ***Line 482, Column 252***:

```
<i class="icon-search icon-white"></i>
```

2.4 Navigable: Provide ways to help users navigate, find content, and determine where they are.

Success Criteria 2.4.4 Link Purpose (In Context) (A)

Check 174: **Anchor contains no text.**

Repair: Add text to the `a` element or the `title` attribute of the `a` element or, if an image is used within the anchor, add Alt text to the image.

- ***Line 115, Column 5***:

```
<a href="/index.php" title=""></a>
```

- ***Line 127, Column 93***:

```
<a style="margin:3px;" rel="nofollow" href="https://facebook.com/GovernoSC" target="_blank"><img src ...
```

- ***Line 127, Column 331***:

```
<a style="margin:3px;" rel="nofollow" href="https://twitter.com/GovSC" target="_blank"><img src="htt ...
```

- ***Line 127, Column 564***:

```
<a style="margin:3px;" rel="nofollow" href="https://youtube.com/GovernoSC" target="_blank"><img src= ...
```

- ***Line 127, Column 801***:

```
<a style="margin:3px;" rel="nofollow" href="https://instagram.com/GovernoSC/" target="_blank"><img s ...
```

- ***Line 240, Column 7***:

```
<a href="#" onclick="javascript: JSNUtils.setTemplateAttribute('jsn_dona_pro_','mobile','no'); retur ...
```

- ***Line 243, Column 7***:

```
<a href="#" onclick="javascript: JSNUtils.setTemplateAttribute('jsn_dona_pro_','mobile','yes'); retu ...
```

- ***Line 317, Column 30***:

```
<a href="/noticias/temas/transportes-e-estradas/revitalizacao-da-rodovia-jorge-lacerda-deve-ficar-pr ...
```

- ***Line 355, Column 30***:

```
<a href="/noticias/temas/defesa-civil-e-bombeiros/defesa-civil-entrega-ordem-de-servico-para-constru ...
```

- ***Line 382, Column 30***:

```
<a href="/noticias/temas/educacao-noticias/governo-do-estado-entrega-mais-40-onibus-para-reforcar-o- ...
```

3.1 Readable: Make text content readable and understandable.

Success Criteria 3.1.1 Language of Page (A)

Check 49: **Document has invalid language code.**

Repair: Add a valid 2 letter or 3 letter language code as defined in the ISO 639 specification to the HTML 'lang' attribute. For XHTML, both 'lang' and 'xml:lang' must be set.

- ***Line 3, Column 1***:

```
<html lang="*" dir="ltr">
<head>
    <link href="/images/favicon.png" rel="apple-touch-icon" sizes=" ...
```

3.3 Input Assistance: Help users avoid and correct mistakes.

Success Criteria 3.3.2 Labels or Instructions (A)

Check 188: Label text is empty.

Repair: Add text to the `label` element.

- ***Line 123, Column 3***:

```
<input type="text" name="q" id="mod-finder-searchword" class="search-query input-medium" size="25" v ...
```

- ***Line 261, Column 21***:

```
<input id="mod-finder-searchword" class="search-query input-medium" name="q" size="25" type="text" v ...
```

- ***Line 476, Column 3***:

```
<input type="text" name="q" id="mod-finder-searchword" class="search-query input-medium" size="20" v ...
```

- ***Line 482, Column 21***:

```
<input id="mod-finder-searchword" class="search-query input-medium" name="q" size="25" type="text" v ...
```

4.1 Compatible: Maximize compatibility with current and future user agents, including assistive technologies.

Success Criteria 4.1.1 Parsing (A)

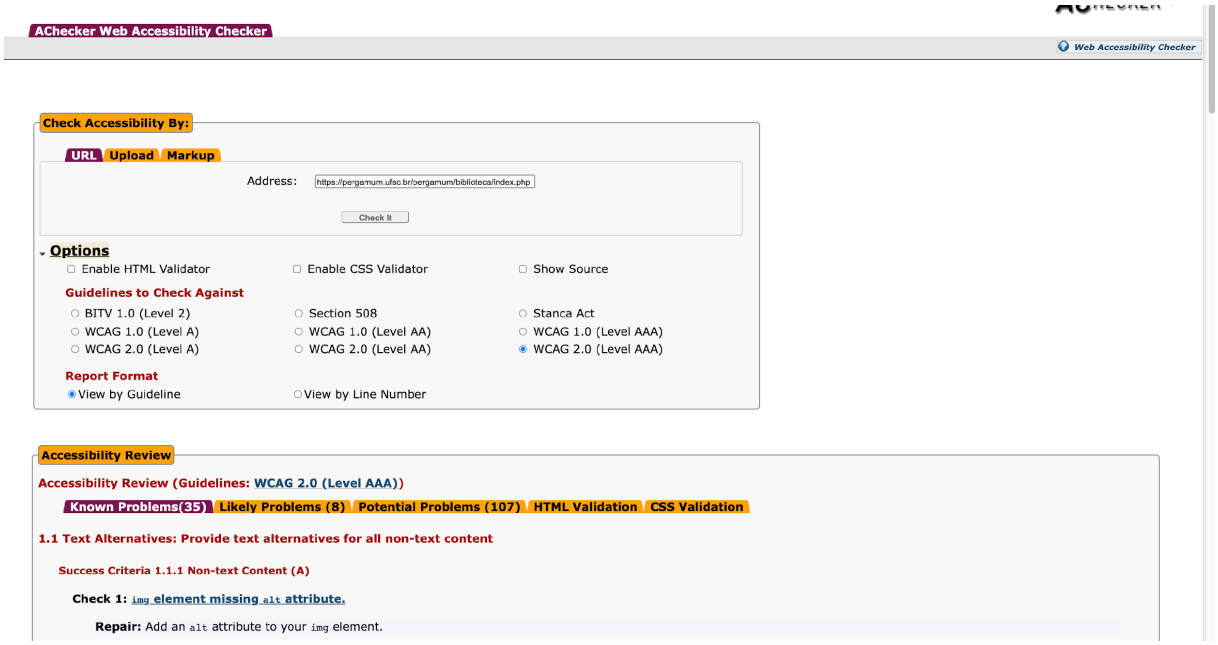
Check 185: `id` attribute is not unique.

Repair: Modify the `id` attribute value so it is unique.

- ***Line 260, Column 1***:

```
<body id="jsn-master" class="jsn-textstyle-business jsn-color-red jsn-direction-ltr jsn-responsive j ...(mod-finder-searchform)
```


Página inicial do Pergamum UFSC - https://pergamum.ufsc.br/



Line 6171, Column 2:

```
<img border="0" src="../img/banners/anima_00.png" />
```

Line 6176, Column 76:

```
<img border="0" src="../img/banners/anima_11.png" />
```

Line 6216, Column 28:

```
<img border="0" src="../img/logo_empresa.jpg"  />
```

1.4 Distinguishable: Make it easier for users to see and hear content including separating foreground from background.

Success Criteria 1.4.4 Resize text (AA)

Check 177: `font` used.

Repair: Remove the `font` element from the document.

Line 5903, Column 4:

```
<font class="link_aguarde">Aguarde...</font>
```

Line 5910, Column 4:

```
<font class="link_aguarde">Aguarde...</font>
```

Line 5918, Column 4:

```
<font class="link_aguarde">Aguarde...</font>
```

Line 5943, Column 4:

Line 5943, Column 4:

```
<font class="link_aguarde">Aguarde...</font>
```

Line 5950, Column 4:

```
<font class="link_aguarde">Aguarde...</font>
```

Line 5958, Column 4:

```
<font class="link_aguarde">Aguarde...</font>
```

Line 5965, Column 4:

```
<font class="link_aguarde">Aguarde...</font>
```

Line 5972, Column 4:

```
<font class="link_aguarde">Aguarde...</font>
```

Line 5979, Column 4:

```
<font class="link_aguarde">Aguarde...</font>
```

Line 5987, Column 4:

```
<font class="link_aguarde">Aguarde...</font>
```

Line 5994, Column 4:

```
<font class="link_aguarde">Aguarde...</font>
```

Line 6001, Column 4:

```
<font class="link_aguarde">Aguarde...</font>
```

Line 6008, Column 4:

```
<font class="link_aguarde">Aguarde...</font>
```

Line 6015, Column 4:

```
<font class="link_aguarde">Aguarde...</font>
```

Line 6022, Column 4:

```
<font class="link_aguarde">Aguarde...</font>
```

- ***Line 6030, Column 4***:

```
<font class="link_aguarde">Aguarde...</font>
```

- ***Line 6038, Column 4***:

```
<font class="link_aguarde">Aguarde...</font>
```

- ***Line 6045, Column 4***:

```
<font class="link_aguarde">Aguarde...</font>
```

- ***Line 6052, Column 4***:

```
<font class="link_aguarde">Aguarde...</font>
```

- ***Line 6059, Column 4***:

```
<font class="link_aguarde">Aguarde...</font>
```

- ***Line 6066, Column 4***:

```
<font class="link_aguarde">Aguarde...</font>
```

- ***Line 6074, Column 4***:

```
<font class="link_aguarde">Aguarde...</font>
```

- ***Line 6081, Column 4***:

```
<font class="link_aguarde">Aguarde...</font>
```

- ***Line 6090, Column 4***:

```
<font class="link_aguarde">Aguarde...</font>
```

- ***Line 6097, Column 4***:

```
<font class="link_aguarde">Aguarde...</font>
```

- ***Line 6106, Column 4***:

```
<font class="link_aguarde">Aguarde...</font>
```

- ***Line 6114, Column 4***:

```
<font class="link_aguarde">Aguarde...</font>
```

- ***Line 6216, Column 96***:

```
<font class="txt_silver_novo"></font>
```

2.4 Navigable: Provide ways to help users navigate, find content, and determine where they are.

Success Criteria 2.4.4 Link Purpose (In Context) (A)

Check 174: **Anchor contains no text.**

Repair: Add text to the `a` element or the `title` attribute of the `a` element or, if an image is used within the anchor, add Alt text to the image.

- ***Line 6176, Column 2***:

```
<a href="http://www.biblioteca.pucpr.br/pergamum/mobile/" target="_blank"><img border="0" src="../im ...
```

3.1 Readable: Make text content readable and understandable

Success Criteria 3.1.1 Language of Page (A)

Check 48: **Document language not identified.**

Repair: For HTML documents add the lang attribute and a valid ISO-639-1 two letter language code to the opening HTML element. For XHTML documents add both the lang and xmllang attributes with a valid ISO-639-1 two letter language code to the opening HTML element.

- ***Line 2, Column 1***:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Pragma" content="no-cache" /> ...
```

Check 49: **Document has invalid language code.**

Repair: Add a valid 2 letter or 3 letter language code as defined in the ISO 639 specification to the HTML 'lang' attribute. For XHTML, both 'lang' and 'xml:lang' must be set.

- ***Line 2, Column 1***:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Pragma" content="no-cache" /> ...
```

4.1 Robust - Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies.

Success Criteria 4.1.1 Parsing (A)

Check 185: `id` **attribute is not unique.**

Repair: Modify the `id` attribute value so it is unique.

Line 5887, Column 1:

```
<body onload="tipo_pesq('palavra','');verifica_erro();document.pesquisas.termo_para_pesquisa.focus() ...(fechar_2)
```


Web Service API

APÊNDICE C – CÓDIGO FONTE

O código completo pode ser encontrado no repositório acessando o endereço:
https://github.com/Schveitzer/automatic_accessibility_tests

#pages.validator.spec.js

```
import { logHelper } from "../support/loghelper.js";

const pagesToEvaluate = Cypress.env('pages')

describe('Accessibility evaluation of Web pages', () => {

  pagesToEvaluate.forEach((page) => {

    it('Evaluates accessibility for the eating page', () => {

      cy.visit(page);
      cy.wait(3000);

      cy.injectAxe();

      const logFormater = logHelper.logFormatter

      const wcagVersions = { runOnly: { values: ['wcag2a', 'wcag2aa', 'wcag2aaa'] } }

      cy.checkA11y(null, wcagVersions, logFormater);

    })

  });
})
```

#loghelper.js

```
/// <reference types="Cypress" />

class LogHelper {

  logFormatter(wcagViolations) {
    cy.task(
      'log',
      `
      Number of violations: ${wcagViolations.length} type${wcagViolations.length === 1 ? '' :
's'
      } of Wcag violation${wcagViolations.length === 1 ? '' : 's'} detected.`
    );
```

```
formatLoggedViolationsInATable(wcagViolations);

logEachViolationDetected(wcagViolations);

// -- logFormatter implementation details

function formatLoggedViolationsInATable(wcagViolations) {
  cy.task(
    'table',
    wcagViolations.map(
      // Table header
      ({ id, impact, description, nodes }) => ({
        'Id of Axe rule violated': id,
        'User impact': impact,
        'Description of rule': description,
        'Number of nodes affected': `${nodes.length}`,
      })
    )
  );
}

function logEachViolationDetected(wcagViolations) {
  wcagViolations.forEach((wcagFailure, indexInViolationsArray, array) => {
    const { id, impact, help, helpUrl, nodes } = wcagFailure;

    cy.task(
      'log',
      `
      ######## Violation ${indexInViolationsArray + 1} of ${array.length
      }: '${id}', whit User impact ${impact} ########`
    );

    cy.task(
      'log',
      `
      ${help}. Acess for more details: ${helpUrl}`
    );

    cy.task(
      'log',
      `
      ${nodes.length} instance${nodes.length > 1 ? 's' : ''} of this violation:`
    );

    nodes.forEach((node, indexInNodesArray) => {
      cy.task(
        'log',
        `
```

```
        Violation ${indexInViolationsArray + 1} instance ${indexInNodesArray + 1}:
${node.html
          }`
        );
      });
    });
  }
 }
}

export const logHelper = new LogHelper();
```

#suport/index.js

```
import './commands'
import 'cypress-axe'

Cypress.on("uncaught:exception", (err, runnable) => {
 // returning false here prevents Cypress from
 // failing the test
 return false;
});

Cypress.Server.defaults({
 delay: 500,
 force404: false,
 ignore: (xhr) => {
  // handle custom logic for whitelisting
  return true;
 },
});
```

#plugins/index.js

```
/// <reference types="cypress" />

module.exports = (on, config) => {
 on('task', {
  log(message) {
   console.log(message)

   return null
  },
  table(message) {
   console.table(message)
   return null
  }
```

```
 })
```

#cypress.json

```
{
   "video": false,
   "viewportWidth": 1366,
   "viewportHeight": 768,
   "defaultCommandTimeout": 10000
}
```

#package.json

```
{
 "version": "1.0.0",
 "description": "automatic accessibility tests for Web pages",
 "author": "Alan Schveitzer",
 "license": "MIT",
 "scripts": {
   "test:open": "npx cypress open",
   "test:chrome": "npx cypress run --config video=false --env allure=true --browser chrome",
   "test:chrome:headless": "npx cypress run --headless --config video=false --env allure=true --browser chrome"
 },
 "dependencies": {
   "axe-core": "^4.3.5",
   "cypress": "^8.7.0",
   "cypress-axe": "^0.13.0"
 }
}
```

APÊNDICE D – ARTIGO DA MONOGRAFIA

Testes automatizados de acessibilidade para páginas Web

Alan Schveitzer

Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)

Caixa Postal 476 – 88.010-970 – Florianópolis – SC – Brasil

alan.schveitzer@gmail.com

Abstract. *This article is a study of deficiencies, accessibility guides for web pages, research for automated accessibility testing tools for web pages, and the creation of a solution for automating accessibility tests on web pages, aiming to provide a tool that can help developers and content creators to create more accessible web pages. The developed solution uses current technologies. As a result, a command line execution tool was created. Through it, it is possible for the user to automatically detect violations rules of accessibility on Web pages at the beginning of their development.*

Resumo. *O presente trabalho tem como proposta o estudo sobre deficiências, guias de acessibilidade para página Web, pesquisa por ferramentas de testes automatizados de acessibilidade para páginas Web, e a criação de uma solução para automação de testes de acessibilidade em páginas Web, visando fornecer uma ferramenta que possa auxiliar desenvolvedores e criadores de conteúdo a criar páginas Web mais acessíveis. A solução desenvolvida utiliza tecnologias atuais, como resultado, uma ferramenta de execução via linha de comando foi criada. Através dela é possível ao usuário a detecção de forma automatizada de violações de regras de acessibilidade em páginas Web ainda no início de seu desenvolvimento.*

1. Introdução

Com a globalização, a tecnologia tem-se feito cada vez mais presente na vida das pessoas. Os acessos à *internet* e sistemas *Web* vêm crescendo no Brasil, conforme aponta o IBGE (2019), a Internet nos domicílios urbanos era utilizada em 83,8% dos domicílios no ano de 2018. Já no ano de 2019, esse número subiu para 86,7%.

É cada vez mais comum vincularmos as tarefas do cotidiano, sejam elas profissionais ou de lazer, ao uso de tecnologias e dispositivos conectados à internet, como aponta a pesquisa sobre o acesso à Internet pelos brasileiros onde são listados os serviços mais acessados pela população, sendo eles serviços, portais, entretimento, busca/navegação, social media, noticiários e outros (COMSCORE, 2015).

Conforme o Censo 2010, quase 46 milhões de brasileiros, cerca de 24% da população do país, declarou ter algum grau de dificuldade em pelo menos uma das habilidades investigadas (enxergar, ouvir, caminhar ou subir degraus), ou possuir deficiência intelectual (IBGE, 2010), portanto é possível concluir que uma parte significativa da população brasileira possui algum tipo de deficiência.

Segundo Henry (2019) “acessibilidade na *Web* significa que sites, ferramentas e tecnologias são projetados e desenvolvidos para que pessoas com deficiência possam usá-los”. Sendo assim, um sistema Web precisa ser desenvolvido buscando a acessibilidade como um

requisito e não apenas como um recurso adicional para que as pessoas com deficiência possam usá-las.

Com o passar do tempo, páginas da *Web* se tornaram cada vez mais sofisticadas e complexas, com novos e avançados recursos, como, por exemplo, arrastar e soltar (*drag and drop*), o que causou um aumento significativo no desafio de prover ao usuário com deficiência uma experiência acessível (CWEB, 2020). Neste contexto, surgem organizações internacionais com intuito de superar esses desafios, como, por exemplo, a maior organização internacional de padrões para Web a World Wide Web Consortium (W3C), que através da sua iniciativa intitulada *Web* Accessibility Initiative (WAI) procura desenvolver um conjunto de diretrizes para auxiliar o desenvolvimento de páginas web acessíveis, sendo o seu principal guia o Web Content Accessibility Guidelines (WCAG).

Portanto, neste contexto, o presente trabalho visa efetuar um estudo sobre normas de acessibilidade para páginas *Web*, além disso, o desenvolvimento de um projeto para automação de testes de acessibilidade em sistemas *Web*, visando a criação de uma ferramenta automatizada para indicar se os sistemas *Web* alvos estão de acordo com as diretrizes do WCAG na versão 2.0 ou superior, servido assim como uma ferramenta para auxiliar desenvolvedores a criarem páginas web mais acessíveis. Por fim, visa a geração de um relatório apontando pontos de falha ou melhoria em relação à acessibilidade nas páginas testadas.

2. Escopo

A solução desenvolvida neste trabalho constitui-se em uma aplicação executada via linha de comando, ela foi desenvolvida utilizando tecnologias como JavaScript, NodeJS, Cypress e a biblioteca axe-core, a qual dá a capacidade de detecção de violações de acessibilidade para a ferramenta. Ela permite que através de um único comando sejam executados testes automatizados em páginas *Web* com objetivo de detectar violações de regras de acessibilidade baseadas no guia WCAG e outros. Com isto ela permite aos desenvolvedores e criadores de conteúdo a detecção de falhas de acessibilidade nas páginas desenvolvidas ainda no seu início.

Como saída desta solução, é gerado na interface de linha de comando um relatório com informações demonstrando se foram detectadas violações de regras ou boas práticas de acessibilidade para páginas *Web*, a quantidade delas e informações detalhadas incluindo formas de correção das violações detectadas.

O presente trabalho se limita ao desenvolvimento de uma solução para execução de testes automatizados de acessibilidade para páginas *Web* através de um cliente de linha de comando, com o propósito de ser uma ferramenta que auxiliará no desenvolvimento de páginas *Web* mais acessíveis, podendo ser estendido e reaproveitado em novos trabalhos, haja visto que, o código desenvolvido é aberto e está disponível para todos.

3. Fundamentação Teórica

3.1 Tipos de deficiência

O Centro de Controle e Prevenção de Doenças (2020), define que deficiência é qualquer condição do corpo ou da mente que dificulta que a pessoa com a condição realize certas atividades e interaja com o mundo ao seu redor (CDC, 2020). Ainda segundo o autor existem muitos tipos de deficiência que podem afetar uma pessoa como visuais, motoras, de raciocínio, memória, comunicação e outras.

Brasil (2004), no decreto nº 5.296/2004, categoriza os diferentes tipos de deficiência como "deficiência física, visual, auditiva, mental (funcionamento intelectual ou cognitiva significativamente inferior à média) e deficiência múltipla (associação de duas ou mais deficiências)."

Já Maior (2021), traz uma definição mais ampla de deficiência, afirmando ser uma construção social em um processo constante de evolução, que depende de a sociedade assumir a sua responsabilidade nos processos de inclusão e envolvimento da pessoa com deficiência na vida da comunidade.

3.2 Acessibilidade na Web

Acessibilidade na *Web* refere-se à prática de tornar as páginas da *Web* acessíveis para todos os usuários, especialmente aqueles com deficiência. Embora uma *Web* acessível signifique um acesso sem precedentes à informação para pessoas com deficiência, pesquisas recentes sugerem que as melhores práticas em acessibilidade ainda não foram alcançadas (HARPER; YESILADA, 2008).

Para Cusin e Vidotti (2019), "a acessibilidade *Web* significa que pessoas com necessidades especiais podem usar a *Web*. Especificamente, significa que pessoas com necessidades especiais podem compreender, entender, navegar, interagir e contribuir com a *Web*".

3.3 *Web* Content Accessibility Guidelines (WCAG)

Desenvolvido pelo W3C (World Wide Web Consortium) em cooperação com indivíduos e organizações em todo o mundo e publicado pela iniciativa WAI, o guia WCAG é um conjunto de normas projetadas para tornar a *Web* mais acessível para pessoas com deficiência. O guia possui três versões, sendo que a terceira versão está em desenvolvimento. O WCAG é atualizado ao longo dos anos para considerar as mudanças nas tecnologias baseadas na *Web*, tecnologias assistivas, tendências de *design*, e o desenvolvimento, e crescimento da *Web* em dispositivos móveis (BUREAU OF *INTERNET* ACCESSIBILITY, 2019). Em 11 de dezembro de 2008, era lançado pelo W3C o WCAG 2.0, a segunda versão do guia. O seu lançamento foi motivado pelas grandes mudanças na tecnologia a partir dos anos 2000. Ele foi uma evolução do seu antecessor, planejado para poder ser aplicado a basicamente qualquer produto digital incluindo documentos e aplicativos (BUREAU OF *INTERNET* ACCESSIBILITY, 2019).

Segundo Fenner (2021), os padrões de acessibilidade definidos no WCAG 2.0 são divididos em 4 princípios, os quais contêm diretrizes com critérios de sucesso e técnicas específicas sendo eles Perceptível, Operável, Compreensível e robusto.

Para cada um dos quatro princípios existem normas estabelecidas, que, possuem critérios de sucesso que atestam se as normas foram seguidas ou não e, por último, são listadas várias técnicas que podem ser empregadas para o alcance dos critérios de sucesso definidos (CHANTRE 2015).

3.4 Tipos de Testes de Acessibilidade Web

Existem muitas formas para se testar a acessibilidade de páginas Web. Comumente, elas são divididas em dois grandes grupos, um grupo é composto por avaliações técnicas que são geralmente executadas por especialistas que possuem conhecimentos de Web *design* e acessibilidade. O outro grupo é composto pela avaliação da experiência do usuário que conta

com a participação de usuários comuns da Internet. Os grupos são subdivididos em testes automatizados e testes manuais (HASSANZADEH; NAVIDI, 2010).

Os testes manuais podem ser executados por dois grupos distintos, sendo o primeiro deles composto por pessoas técnicas. Testes manuais executados por este grupo consistem na revisão de detalhes específicos de acessibilidade de páginas *Web*. Para efetuar os testes, é executado pelos especialistas um método que possui várias etapas que buscam examinar as páginas em acordo com os *checkpoints* de acessibilidade (HASSANZADEH; NAVIDI, 2010).

Ainda conforme Hassanzadeh e Navidi (2010) o segundo grupo de testes manuais consiste nos testes baseados na experiência do usuário. A execução desses testes por vezes conta com a participação de especialistas e consiste em uma série de testes executadas pelos usuários com acompanhamento de especialistas onde são apontados erros e dificuldades de acessibilidade no acesso dessas páginas.

Para Hassanzadeh e Navidi (2010) testes automatizados de acessibilidade Web "São testes executados por ferramentas que avaliam se as páginas determinadas estão de acordo com as normas estabelecidas pelo W3C ou pela regulamentação 508 do governo dos Estados Unidos da América..."

As ferramentas para testes automatizados de acessibilidade em páginas *Web* são direcionadas para diferentes públicos, que vão desde autores de conteúdo, *designers*, testadores de *software*, desenvolvedores até mesmo usuários finais. Estas ferramentas oferecem vários recursos e funcionalidades que permitem ao usuário comparar e avaliar qual é a mais adequada para atender às suas necessidades (ABOU-ZAHRA, 2017).

4. Estado da Arte

Este capítulo tem o objetivo identificar o estado atual em que se encontram ferramentas, sejam elas comerciais ou gratuitas e pesquisas que se relacionem ao projeto deste trabalho.

4.1 Definição do estudo

Para efetuar a pesquisa sobre trabalhos e ferramentas, foram utilizadas algumas fontes de pesquisa além da própria web, com intuito de localizar ferramentas disponíveis e artigos que se relacionam com as tecnologias utilizadas na composição deste trabalho. Entre as fontes de pesquisa utilizadas estão:

- ACM Digital Library
- IEEE Xplore Digital Library
- Research Gate
- Medium
- Google

Baseado no tema deste trabalho e nas tecnologias utilizadas nele, foram empregadas diversas *strings* na pesquisa. Para cada termo definido também foram utilizados sinônimos em inglês. A tabela 1 alguns termos que foram utilizados na pesquisa para a realização do estudo.

Termo	Sinônimo	Termo em Inglês
Testes de acessibilidade para páginas web	Validação de acessibilidade para páginas web	Accessibility testing for web pages, Accessibility validation for web pages
Ferramenta para testes de acessibilidade em páginas web	Ferramentas de avaliação de acessibilidade da web	Tool for accessibility testing of web pages, Web Accessibility Assessment Tools
Testes automatizados de acessibilidade para páginas web	Tecnologias para automação de testes de acessibilidade web	Automated accessibility testing for web pages, Technologies for automating web accessibility testing

Tabela 1. *Strings* e termos da pesquisa

Foram definidos critérios de inclusão e exclusão, buscando assim, um resultado mais assertivo na pesquisa, onde foram considerados critérios de funcionalidade e semelhança tecnológica dessas ferramentas. Foram excluídas ferramentas do tipo *Plug-in* de navegador, aplicativos desktop, aplicações mobile, ferramentas que contemplam apenas a versão 1.0 do WCAG ou que não contemplem o guia WCAG. Foram incluídas as fermentas que mais apareceram na pesquisa e também que tenham sido citadas com mais frequência nos trabalhos relacionados.

4.2 Ferramentas disponíveis

Após finalizar a pesquisa e aplicar os critérios de inclusão exclusão, foram selecionadas 8 ferramentas, os seus nomes, e as suas características que foram consideradas mais relevantes pelo autor foram compiladas e comparadas na tabela 2.

Ferramenta	Custo anual ($)	Código aberto	CI/CD	*Scriptable*	Possui cliente	Apenas Online	Aceita Plug-ins
Achecker	Gratuito	Sim	Não	Não	Não	Sim	Não
acessMonitor	Gratuito	Sim	Não	Não	Não	Sim	Não
Axe Devtools	$480.00	Não	Sim	Sim	Sim	Não	Sim
Cynthia Says	Gratuito	Não	Não	Não	Não	Sim	Não
Lighthouse	Gratuito	Sim	Sim	Sim	Sim	Não	Não
Taw	Gratuito	Não	Não	Não	Não	Sim	Não
Tenon	$984.00 a $8,244.00	Não	Sim	Sim	Sim	Não	Não
Wave	$4,000.00 a $ 12,000,00	Não	Sim	Sim	Não	Não	Não

Tabela 2. Comparação das ferramentas

Custo anual, a maioria das ferramentas pesquisadas são gratuitas, porém temos algumas com custo anual, sendo que entre as ferramentas pagas, duas possuem um custo anual significativo, no caso as ferramentas Tenon e Wave, isto pode tornar o uso dessas ferramentas inviável dependendo dos recursos da companhia ou pessoas que irão utiliza-las.

Código aberto, as ferramentas Achecker, acessMonitor e Lighthouse possuem o seu código aberto, assim permitindo que outros desenvolvedores possam corrigir possíveis problemas nas ferramentas e também adicionar novos recursos nelas.

CI/CD, as ferramentas Axe Devtools, Lighthouse, Tenon e Wave permitem a sua utilização em processos de integração e entrega continuas, ou seja, essas ferramentas são capazes de serem executadas durante o desenvolvimento das páginas web, assim as páginas podem ser validadas quanto a problemas de acessibilidade ainda no inicio do seu desenvolvimento.

Scriptable, no caso das ferramentas Axe Devtools, Lighthouse, Tenon e Wave, elas podem ser estendidas ou automatizadas através de scripts, isto possibilita o uso destas ferramentas através de linguagens de programação, permitindo assim, uma maior personalização no seu uso e adaptação ao meio de desenvolvimento ao qual a ferramenta está inserida.

Possui cliente, apenas as ferramentas Axe Devtools, Lighthouse e Tenon possuem clientes para execução da ferramenta via linha de comando, isto significa ser possível a inserção dessas ferramentas em ambientes que executam programas via comandos em terminais de texto, ou seja, que não necessitam de uma interface gráfica para serem executados, assim permitindo que estas ferramentas sejam executadas de uma forma mais personalizada.

Apenas online, ferramentas que são executas apenas online, essas ferramentas não permitem outro modo de execução, sendo esse um fator limitante, pois para utilizar elas é necessário acesso a internet, também é necessário que as páginas que serão testadas estejam hospedadas em algum servidor web, impedindo assim, que os testes sejam executados no inicio do desenvolvimento dessas páginas.

Após finalizar a comparação entre as ferramentas pesquisadas, fica evidente que há a necessidade de uma ferramenta que seja gratuita e que auxilie desenvolvedores e criadores de conteúdo no desenvolvimento de páginas web mais acessíveis.

A ferramenta proposta neste trabalho conta com os recursos de ambas as ferramentas pesquisadas anteriormente, sejam elas pagas ou gratuitas. Ela também é uma ferramenta gratuita, de código aberto, tendo como diferencial a possibilidade de adição de plug-ins estendendo as suas funcionalidades, o uso de tecnologias atuais e amplamente usadas no mercado, uma fácil execução e a possibilidade de personalização e melhoria pela comunidade ou os seus usuários.

5. Desenvolvimento

5.1 Arquitetura da ferramenta

O usuário, através de uma *interface* de linha de comando, interage com a ferramenta informando o(s) endereço(s) ou a localização do(s) arquivo(s) HTML das páginas que deseja submeter a validação de acessibilidade baseados nas diretrizes do guia WCAG ou outros. Após informar tais dados e executar o comando, o *test runner* Cypress executa o *script* de validação, que então

irá executar uma instância de um navegador *Web*, onde serão carregadas as páginas que terão a sua estrutura HTML avaliada em relação às diretrizes estabelecidas no guia selecionado.

Ao finalizar a execução, ainda na *interface* de linha de comando, são retornados os dados demonstrando se alguma diretriz foi violada, um identificador da violação, a quantidade de violações, a criticidade da violação, uma breve descrição, o número de nodos afetados e também mais detalhes sobre cada violação.

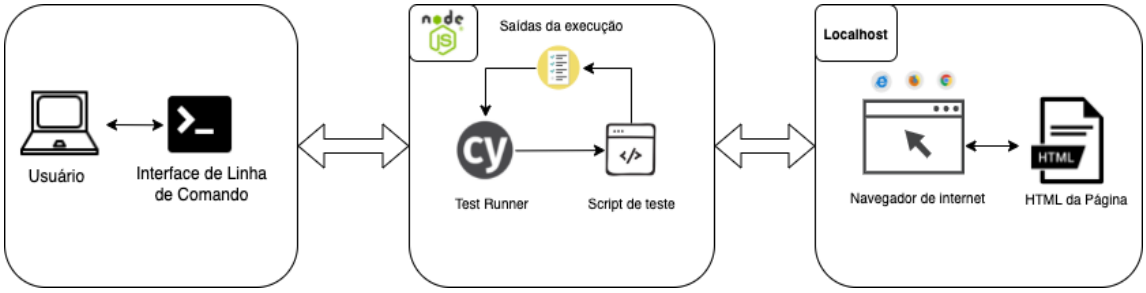


Figura 1. Arquitetura geral da solução

Para o desenvolvimento da solução foi utilizado como base o framework Cypress. Após efetuar a instalação do framework utilizando o NPM, o Cypress foi executado através uma interface de linha de comando. Após executar o comando *npx cypress open*, o framework gerou a estrutura inicial do projeto, incluindo as pastas e arquivos necessários para o desenvolvimento e execução dos testes.

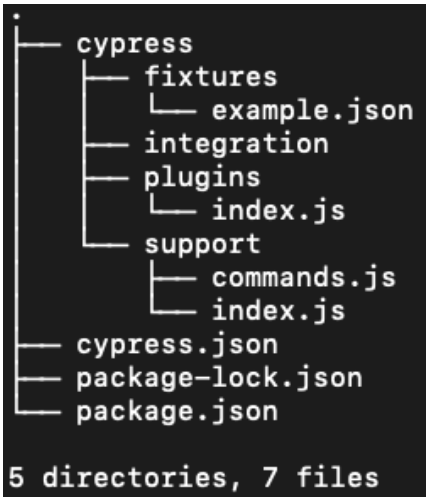


Figura 2. Estrutura de arquivos e diretórios

5.2 Implementação da ferramenta

Tendo como principio a estrutura inicial gerada pelo framework *Cypress*, iniciou-se o desenvolvimento do script de teste. Inicialmente foi criado o arquivo *pages.validator.spec.js*, ele contém um script que segue a interface padrão encontrada em outro framework de testes chamado Mocha, através desta interface é definida a descrição do script de teste utilizando a chave "*describe()*", também são definidos os testes que o script executará utilizando a chave "*it()*".

Para cada página informada através da variável *pagesToEvaluate* é efetuado pelo script um laço de repetição utilizando a função *forEach* do JavaScript, esta função garante que cada

página informada seja validada pela lógica implementada no script de teste, é esta função que permite ao usuário informar e validar mais de uma página com apenas um único comando.

Por fim o script executa a função *cy.checkA11y*, a função recebe como parâmetros os elementos HTML que serão escaneados, no caso deste projeto ele é informado com o valor *null*, para que assim sejam validados todos os elementos das páginas, também é informado como parâmetro as configurações de execução do axe-core, para este projeto são informadas as versões do guia WCAG que serão utilizadas na validação, e por último é informado o parâmetro o *logFormater* para que os resultados gerados pela função sejam formatados e tragam de forma organizada todas as violações detectadas.

```
1   import { logHelper } from "../support/loghelper.js";
2
3   const pagesToEvaluate = Cypress.env('pages')
4
5   describe('Accessibility evaluation of web pages', () => {
6
7     pagesToEvaluate.forEach((page) => {
8
9       it('Evaluates accessibility for the eating page', () => {
10
11        cy.visit(page);
12
13        cy.injectAxe();
14
15        const logFormater = logHelper.logFormatter
16
17        const wcagVersions = { runOnly: { values: ['wcag2a', 'wcag2aa', 'wcag2aaa'] } }
18
19        cy.checkA11y(null, wcagVersions, logFormater);
20
21      })
22
23    });
24  })
25
```

3. Código do script de teste

5.3 Execução da ferramenta

Para executar a ferramenta foram selecionadas três páginas Web, o critério utilizado para a seleção destas páginas foi o de que são páginas de interesse e uso público. Foram selecionadas as páginas do Sábio (Sistema de automação de Bibliotecas), página que pertence à biblioteca pública do estado de Santa Catarina, após o autor acessar a página, foi possível concluir que ele é o sistema que permite entre outras funções a consulta do acervo da biblioteca pública do estado, a reserva de exemplares, o acompanhamento de empréstimos. Outra página validada foi a do Portal do governo do Estado de Santa Catarina, ao acessar a página é possível constatar que se trata de uma página Web que contém informações, noticias, oportunidades e acesso a inúmeros serviços oferecidos pelo governo do estado de forma online. A última página validada foi a do Pergamum, conforme o autor conseguiu constatar, é possível afirmar que se trata do sistema utilizado pela Universidade Federal de Santa Catarina para consulta do acervo das suas bibliotecas e outras funções.

Após finalizar a execução dos testes automatizados na página inicial do Sábio com a solução desenvolvida neste trabalho, a ferramenta detectou ao todo seis violações de acessibilidade, sendo três violações com impacto ao usuário crítico, duas com impacto sério e

uma com moderado. As violações estão presentes em trinta e três nodos da estrutura HTML da página.

Ao finalizar os testes na página anterior então foram executados os testes na página do portal do governo do estado de Santa Catarina de forma automatizada com sucesso. Após o fim da execução foram detectadas cinco violações de acessibilidade na página, destas cinco violações quatro possuíam um impacto sério e uma possuía impacto moderado. Ao todo dezessete nodos da estrutura HTML apresentaram alguma violação das regras de acessibilidade.

Por último foram excetuados os testes na página do Pergamum. As validações foram executadas na versão de alto contraste da página. Foram detectadas ao todo nove violações na página inicial que afetam vinte e um nodos da estrutura HTML. Foi possível constatar que a maioria das violações possui um impacto crítico ou sério, ou seja, violações que se não corrigidas prejudicarão significativamente a acessibilidade. Assim como nas páginas avaliadas anteriormente, foram detectadas violações que possuem maior impacto para usuários com baixa ou nenhuma visão, o que na concepção do autor representa um grande desafio para estudantes, professores e toda a comunidade que necessita da página e dos serviços oferecidos por ela.

6. Estudo comparativo entre as ferramentas

Para constatar se a solução desenvolvida neste trabalho alcançou o objetivo de detectar de forma automatizada falhas de acessibilidade em páginas *Web*, foi efetuado um estudo comparativo. Neste estudo, os resultados obtidos anteriormente com a execução dos testes pela ferramenta nos sites escolhidos, foram comparados com o resultado obtido ao efetuar os mesmos testes, porém utilizando a ferramenta Achecker. Como critério de comparação entre as ferramentas será utilizado a quantidade de violações detectadas por cada uma delas.

As páginas do Sábio (Sistema de Biblioteca do Governo do Estado de Santa Catarina), Portal do Governo do Estado de Santa Catarina e Pergamum da UFSC, foram submetidas a validação automatizada com a ferramenta Achecker, utilizando o guia WCAG na versão 2.1 como referência. Após concluir a execução na ferramenta AChecker para todas as páginas e comparando com os resultados obtidos ao utilizar a solução proposta neste trabalho, constatou-se que a solução desenvolvida possui capacidade similar ou superior em número de violações detectadas em comparação com a Achecker. Ainda que para páginas como a do portal do governo de Santa Catarina o resultado divirja dos demais, isto é justificável devido ao fato da biblioteca do axe-core utilizada no desenvolvimento da solução cubra apenas 57% dos critérios de sucesso do guia WCAG. Ou seja, além da diferença entre o modo como as ferramentas detectam as violações, essa limitação de cobertura afetou o número de violações detectadas, pois para efetuar uma comparação justa, foi utilizado o único guia coberto pela AChecker, o WCAG na versão 2.1.

Além da questão de número de violações detectadas, pode-se considerar outros aspectos que posicionam a solução proposta neste trabalho acima da ferramenta AChecker, como a possibilidade da sua execução em um ambiente de CI/CD, sendo totalmente personalizável e estendível. Ademais, a solução deste trabalho também pode ser usada como complemento à Achecker e outras ferramentas similares.

7. Conclusão

O tema da acessibilidade digital atinge toda a sociedade, tornando o tema essencial para democratização e garantia do acesso digital. Considerando o grande aumento de acessos à Web nos últimos anos, cada vez mais pessoas dos mais variados grupos tiveram o seu cotidiano

relacionado a ela, seja para lazer, educação ou informação. Um desses grupos é o de pessoas com deficiência, o qual necessita que as páginas Web sejam acessíveis e lhes permitam navegar sem dificuldades. O desenvolvimento de páginas Web acessíveis é algo complexo. Neste contexto, organizações como o W3C, buscam definir normas e princípios que visam auxiliar os desenvolvedores e criadores de conteúdo digital a desenvolverem páginas Web mais acessíveis.

O presente trabalho teve como proposta desenvolver uma solução para a automação dos testes de acessibilidade de páginas Web, de modo a auxiliar desenvolvedores e criadores de conteúdo no desafio de criarem páginas Web mais acessíveis. Para alcançar este objetivo, neste trabalho foi desenvolvida uma ferramenta capaz de detectar a violação de normas de acessibilidade de páginas Web, à luz das normas estabelecidas no guia WCAG e outros.

A ferramenta foi projetada e desenvolvida com o intuito de ser genérica, gratuita e com código aberto, com o propósito de ser utilizada para validar a acessibilidade digital em qualquer página Web, apenas informando o endereço ou arquivo HTML desta.

Após finalizar o desenvolvimento da ferramenta, foram executadas validações em determinadas páginas Web. A execução com sucesso da ferramenta e a detecção de violações de acessibilidade, demonstrou que o objetivo deste trabalho foi alcançado. Além disso, também foi efetuada uma comparação com uma ferramenta utilizada no mercado, objetivando atestar se a solução desenvolvida neste trabalho, possuía a mesma capacidade em número de detecções de violações detectadas. Ao fim da comparação dos resultados, pode-se atestar que a ferramenta desenvolvida neste trabalho possuí uma capacidade similar ou superior quando comparada a outra ferramenta.

Ao fim o projeto alcançou com êxito o seu objetivo, ser uma ferramenta que auxilia desenvolvedores e criadores de conteúdo a tornar as páginas desenvolvidas mais acessíveis, de forma gratuita e com código aberto. Além disto a elaboração deste trabalho proporcionou ao autor uma grande oportunidade de conhecimento sobre assuntos como tipos de deficiência, acessibilidade Web e suas normas, o que está sendo estudado e desenvolvido sobre acessibilidade Web, testes manuais e automatizados de acessibilidade em páginas Web, ferramentas modernas de desenvolvimento, *scripts* e todas as tecnologias utilizadas para compor o projeto.

Referências

ABOU-ZAHRA, Shadi (ed.). Selecting *Web* Accessibility Evaluation Tools. 2017. Elaborado por W3C. Disponível em: <https://www.w3.org/WAI/test-evaluate/tools/selecting/>. Acesso em: 07 set. 2021.

BRASIL. Decreto Nº 5.296 de 2 de dezembro de 2004. Disponível em: <http://www.planalto.gov.br/ccivil_03/_Ato2004-2006/2004/Decreto/D5296.htm> Acesso em: 15 Aug. 2021.

BUREAU OF *INTERNET* ACCESSIBILITY (Rhode Island). History of the *Web* Content Accessibility Guidelines (WCAG). 2019. Disponível em: <https://www.boia.org/blog/history-of-the-*Web*-content-accessibility-guidelines-wcag>. Acesso em: 29 ago. 2021.

CDC. Disability and Health Overview. Centers for Disease Control and Prevention. Disponível em: https://www.cdc.gov/ncbddd/disabilityandhealth/disability.html. Acesso em: 15 Aug. 2021.

CHANTRE, José Rui Moneteiro. Testes automáticos de acessibilidade em aplicativos móveis. 2015. 90 f. Dissertação (Mestrado) - Curso de Engenharia da Informática, Universidade da Beira Interior, Covilhã - Portugal, 2015. Disponível em: <https://ubibliorum.ubi.pt/bitstream/10400.6/5775/1/4657_8856.pdf>. Acesso em: 31 ago. 2021.

COMSCORE. Digital future in focus Brazil 2015. 2015. Disponível em: https://www.comscore.com/por/Insights/Apresentacoes-e-documentos/2015/2015-Brazil-Digital-Future-in-Focus. Acesso em: 25 jul. 2021.

CUSIN, Cesar Augusto; VIDOTTI, Silvana Aparecida Borsetti Gregorio. Inclusão digital via acessibilidade Web. Liinc, Rio de Janeiro, v. 5, n. 1, p. 45-65, mar. 2019. Disponível em: <http://revista.ibict.br/liinc/article/view/3189/2851>. Acesso em: 22 ago. 2021.

CWEB (comp.). CARTILHA ACESSIBILIDADE NA *WEB*: w3c brasil. W3C BRASIL. 2020. Disponível em: https://ce*Web*.br/cartilhas/cartilha-w3cbr-acessibilidade-web-fasciculo-IV/. Acesso em: 25 jul. 2021.

FENNER, Priscila. Entenda o WCAG 2.0 de forma simples e rápida. 2021. Disponível em: <https://blog.handtalk.me/wcag-2-0/>. Acesso em: 20 ago. 2021.

HARPER, Simom; YESILADA, Yeliz. *Web* Accessibility: a foundation for research. University Of Manchester, Uk: Springer, 2008. 364 p.

HASSANZADEH, Mohammad; NAVIDI, Fatemeh. *Web* site accessibility evaluation methods in action. The Electronic Library, Tehran, v. 28, n. 6, p. 789-803, 16 nov. 2010. Disponível em: <https://www.emerald.com/insight/content/doi/10.1108/02640471011093499/full/pdf?title

=*Web*-site-accessibility-evaluation-methods-in-action-a-comparative-approach-for-ministerial-*Web*-sites-in-iran>. Acesso em: 07 set. 2021.

HENRY, Shawn Lawton. Essential Components of *Web* Accessibility. 2017. Disponível em: < https://www.w3.org/WAI/test-evaluate/preliminary/>. Acesso em: 07 set. 2021.

IBGE. Censo 2010: pessoas com deficiência. Pessoas com deficiência. 2010. Disponível em: https://educa.ibge.gov.br/jovens/conheca-o-brasil/populacao/20551-pessoas-com-deficiencia.html. Acesso em: 25 jul. 2021.

IBGE. Pesquisa mostra que 82,7% dos domicílios brasileiros têm acesso à *Internet*. 2019. Disponível em: https://www.gov.br/mcom/pt-br/noticias/2021/abril/pesquisa-mostra-que-82-7-dos-domicilios-brasileiros-tem-acesso-a-*Internet*. Acesso em: 25 jul. 2021.

MAIOR, Izabel. História, conceito e tipos de deficiência. 2021. Disponível em: <http://www.deficienciavisual.pt/txt-Historia_conceito_tipos_def.htm>. Acesso em: 16 ago. 2021.