

CENTRO UNIVERSITÁRIO SERRA DOS ÓRGÃOS – UNIFESO  
CENTRO DE CIÊNCIAS E TECNOLOGIA – CCT  
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

**TESTES AUTOMATIZADOS DE ACESSIBILIDADE NA WEB: UM ESTUDO DE  
CASO NO AMBIENTE VIRTUAL DE APRENDIZAGEM DA UNIFESO**

Lucas da Silva Cruz

Teresópolis/ RJ  
Maio, 2021

CENTRO UNIVERSITÁRIO SERRA DOS ÓRGÃOS – UNIFESO  
CENTRO DE CIÊNCIAS E TECNOLOGIA – CCT  
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

**TESTES AUTOMATIZADOS DE ACESSIBILIDADE NA WEB: UM ESTUDO DE  
CASO NO AMBIENTE VIRTUAL DE APRENDIZAGEM DA UNIFESO**

Lucas da Silva Cruz

Trabalho de Conclusão de Curso apresentado ao Centro  
Universitário Serra dos Órgãos como requisito  
obrigatório para obtenção do título de Bacharel em  
Ciência da Computação.

Orientador(a): Tiago Resende

Teresópolis/ RJ  
Maio, 2021

CENTRO UNIVERSITÁRIO SERRA DOS ÓRGÃOS – UNIFESO  
CENTRO DE CIÊNCIAS E TECNOLOGIA – CCT  
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

**TESTES AUTOMATIZADOS DE ACESSIBILIDADE NA WEB: UM ESTUDO DE  
CASO NO AMBIENTE VIRTUAL DE APRENDIZAGEM DA UNIFESO**

Lucas da Silva Cruz

Trabalho de Conclusão de Curso aprovado como requisito parcial para obtenção do título de Bacharel em Ciência da Computação no Centro Universitário Serra dos Órgãos pela banca examinadora:

---

Nome do(a) Orientador(a) por extenso - titulação abreviada

---

Nome do(a) Convidado(a) por extenso - titulação abreviada

---

Nome do(a) Convidado(a) por extenso - titulação abreviada

Teresópolis/ RJ  
Maio, 2021

“A criatividade pode ser uma contribuição social, mas apenas na medida em que a sociedade é livre para usar os resultados.”

Richard Stallman

*Aos meus pais, que me deram todas as oportunidades e  
incentivos para estudar que eles mesmos não tiveram.*

## **AGRADECIMENTOS**

Antes de mais nada quero agradecer principalmente a minha família por todo apoio, participação e incentivo durante todo meu processo de formação. À minha companheira de longo prazo, Sarah de Almeida Ferreira, por sempre me ajudar e socorrer nos momentos mais difíceis da minha vida. Agradeço a todos os meus professores e professoras desde a escola à graduação, ao Centro Universitário Serra dos Órgãos (Unifeso) por todas as oportunidades, experiências e pelo ambiente acolhedor que se tornou a minha segunda casa. Um agradecimento especial ao meu orientador Tiago Resende pela sua impecável disponibilidade e paciência na execução de todo este trabalho.

A todos que, direta ou indiretamente, colaboraram para a realização deste trabalho.

## **RESUMO**

As Tecnologias de Informação e Comunicação (TICs) são elementos fundamentais e determinantes para criação de práticas mais democráticas. No âmbito da educação, os Ambientes Virtuais de Aprendizagem (AVA) têm se mostrado um importante instrumento na propagação de conhecimento para aqueles que historicamente não teriam acesso ao ensino, como as pessoas portadoras de alguma deficiência. No entanto, quando o acesso a essas tecnologias não se dá de forma uniforme é criado um fenômeno de marginalização informacional. O presente trabalho discorre sobre a utilização de testes automatizados de acessibilidade como forma de validar requisitos fundamentais recomendados pelas Diretrizes de Acessibilidade para o Conteúdo da Web (WCAG), realizando um estudo de caso do AVA usado no Centro Universitário Serra dos Órgãos (Unifeso) e considerando que os métodos e resultados documentados nesta auditoria possam ser reutilizados em outros e-Learnings, popularizados devido à digitalização intensificada pelo confinamento – principal meio de controle do coronavírus.

Palavras-chave: Acessibilidade, e-Learning, Testes automatizados, WCAG, AVA.

## **ABSTRACT**

The present work discusses the informational marginalization that a person with a disability have to deal on internet and the importance of manual and automated end-to-end tests to guarantee fundamental requirements recommended by the Web Content Accessibility Guidelines (WCAG). This study researches the main technologies and stacks involved in automating tests and discuss about the importance of integrating them into the test suite that is normally built during the process of developing a web application, applying the researched procedures to a study of case of the Virtual Learning Environment (VLE) of the Serra dos Órgãos University Center (Unifeso), considering that the methods and results developed and documented in this acessibility check can be reused in platforms of the same type, which became popular due to change in routine resulting from confinement – the main way of controlling coronavirus.

Keywords: Accessibility, Tests, WCAG, Digital Exclusion, ICTs.



## LISTA DE ILUSTRAÇÕES

	Página
Figura 1 – [digite o nome aqui].....	14
Figura 2 – [digite o nome aqui].....	15
Figura 3 – [digite o nome aqui].....	15

## LISTA DE TABELAS

	Página
Tabela 1 – jhgjgg.....	14
Tabela 2 – [digite o nome aqui].....	15
Tabela 3 – [digite o nome aqui].....	15

## **LISTA DE ABREVIATURAS E SIGLAS**

CAPTCHA: Completely Automated Public Turing Tests to tell Computers and Humans Apart

API: Application Programming Interface

OCR: Optical Character Recognition

## SUMÁRIO

	Página
<b>1 INTRODUÇÃO.....</b>	<b>13</b>
1.1 Justificativa.....	13
1.1 Motivação.....	13
1.1 Objetivos Gerais.....	13
1.2 Objetivos Específicos.....	13
<b>2 TRABALHOS RELACIONADOS.....</b>	<b>13</b>
1.1 NOME1.....	13
1.2 NOME2.....	13
1.3 Testes automáticos de acessibilidade em aplicações móveis.....	13
<b>4 FUNDAMENTAÇÃO TEÓRICA.....</b>	<b>13</b>
4.1 WCAG.....	13
4.2 Testes Automatizados na Web.....	13
4.2.1 WebDriver e Selenium.....	13
4.3 Testes automatizados de acessibilidade.....	13
4.4 Ferramentas automatizadas para testes de acessibilidade.....	13
4.4.1 aXe.....	13
4.4.2 Pa11y.....	13
4.4.3 Google Lighthouse.....	13
4.3 Testes manuais de acessibilidade (caso sobre tempo depois do MVP).....	13
4.3.1 Foco e Tab Order.....	13
4.3.2 Leitores de Tela.....	13
<b>3 METODOLOGIA E DESENVOLVIMENTO .....</b>	<b>13</b>
3.1 Ferramentas utilizadas na avaliação.....	13
3.2 Escopo e páginas da auditoria.....	13
<b>5 RESULTADOS E DISCUSSÃO .....</b>	<b>13</b>
5.1 Asserts que falharam.....	13
5.2 Sugestões de melhoria.....	13
<b>6 CONCLUSÕES.....</b>	<b>13</b>
<b>7 TRABALHOS FUTUROS.....</b>	<b>13</b>
<b>8 REFERÊNCIAS.....</b>	<b>13</b>

## 1. INTRODUÇÃO

Vivemos em um tempo de explosão informacional onde as Tecnologias de Informação e Comunicação (TICs) se tornaram elementos fundamentais e determinantes para a criação de práticas mais democráticas. Nesse contexto contemporâneo, o direito à comunicação e à informação, bem como a democratização das TICs, são fundamentais [MORIGI, V. J., et al. 2016].

Por outro lado, quando o acesso às tecnologias não se dá de forma uniforme é criado um fenômeno de "marginalização informacional", como afirma Mattelart (2002). Para João Pissarra Esteves (2011), essa exclusão social compreende uma nova versão do “digital divide”, que evidencia como as diferenças de acesso refletem as reais desigualdades sociais, políticas e econômicas entre a população incluída e excluída digitalmente.

Apesar do tema Inclusão Social e Digital vir sendo objeto de debates no meio acadêmico, governamental e empresarial, a questão da inclusão de pessoas portadoras de necessidades especiais, em todos os recursos da sociedade, ainda é muito incipiente no Brasil. [MACIEL, M. R. C., et al. 2010]. Uma das formas de exclusão digital está associada justamente à falta de acessibilidade nos serviços e informações da Web, embora ela tenha sido projetada para ser utilizada por qualquer pessoa. As Diretrizes de Acessibilidade para Conteúdo Web (WCAG) visam fornecer recomendações que tornem o conteúdo acessível a um maior número de pessoas com deficiência, que incluem:

“acomodações para cegueira e baixa visão, surdez e baixa audição, limitações de movimentos, incapacidade de fala, fotossensibilidade e combinações destas características, e alguma acomodação para dificuldades de aprendizagem e limitações cognitivas; mas não abordará todas as necessidades de usuários com essas deficiências. Seu conteúdo da Web também ficará mais acessível aos usuários em geral ao seguir estas diretrizes.”  
(WCAG 2.1, 2018, Abstract)

Nos dias atuais, é uma prática intrínseca ao desenvolvimento de um Sistema Web garantir a sua usabilidade, que pode ser definida como o fator que assegura que um produto ou serviço seja fácil de usar, eficiente e agradável a partir do ponto de vista do usuário [PREECE; ROGERS; SHARP, 2007]. Porém, em relação a acessibilidade, que considera a diversidade de seus possíveis usuários e as peculiaridades da interação dessas pessoas com o produto [TORRES; MAZZONI, 2004], muitas das vezes os desenvolvedores e especialistas

em Garantia de Qualidade (QA) não consideram as diretrizes e padrões da WCAG no design e na test suite (conjunto de casos de teste) das aplicações, gerando uma série de problemas para o usuário com alguma deficiência.

Por causa disso surgiram as engines como Axe e Pa11y, que implementam as regras da WCAG e rodam testes end-to-end (e2e) de acessibilidade nas páginas de uma aplicação da web. Elas foram arquitetadas e desenvolvidas para integrar com qualquer ambiente de teste já existente em um projeto, de forma que as organizações possam automatizar os testes de acessibilidade junto dos seus testes regulares.

Durante este estudo, foi realizado um levantamento das principais tecnologias e stacks utilizadas na construção de testes e2e de acessibilidade em aplicações web e como integrar essas stacks ao projeto, além de realizar um estudo de caso que evidencia as principais formas de se testar a acessibilidade e as soluções para os problemas mais comuns acusados por esses testes.

## **1.1. OBJETIVO GERAL**

Os objetivos gerais desta monografia são conscientizar os desenvolvedores de aplicações web e especialistas em QAs da importância de se validar a acessibilidade do sistema através de testes automatizados e manuais; explicitar as principais ferramentas utilizadas para se fazer testes e2e de acessibilidade em aplicações web e como integrá-las a um sistema; além de fazer um estudo de caso da verificação de acessibilidade do próprio Ambiente Virtual de Aprendizagem (AVA) do Centro Universitário Serra dos Órgãos (AVA).

## **1.2. HIPÓTESES**

1.2.1. Quando o acesso às tecnologias não se dá de forma uniforme é criado um fenômeno de marginalização informacional.

1.2.2. Testes de acessibilidade não são considerados pelas empresas que desenvolvem software com a mesma importância de outros tipos de testes automatizados, como de carga e estresse, integração ou de performance.

1.2.3. O confinamento como medida de contenção do coronavírus intensificou o problema de marginalização informacional da pessoa portadora de deficiência uma vez que ferramentas digitais passaram a ser utilizadas com muito mais frequência e muitas das vezes de forma obrigatória.

### **1.3. OBJETIVOS ESPECÍFICOS**

Os objetivos específicos do trabalho são apresentar os testes automatizados e diferenciar as principais abordagens de automatização; realizar uma análise e descrição dos diferentes tipos de testes automatizados; discorrer sobre a importância dos testes e para automação de testes de acessibilidade, fazer um levantamento das principais técnicas e ferramentas para fazer testes e de acessibilidade; avaliar comparativamente as principais tecnologias usadas no desenvolvimento desses testes e de acessibilidade; e, por fim, exemplificar alguns dos principais problemas de acessibilidade e mostrar como adaptar o sistema de acordo com as especificações da WCAG para resolver os asserts dos testes automatizados.

## 2. REVISÃO DA LITERATURA

Não foram encontrados na literatura estudos que correlacionassem testes automatizados à verificação de acessibilidade segundo as Diretrizes de Acessibilidade para o Conteúdo da Web (WCAG) especificamente no campo do desenvolvimento de Aplicações Web. No entanto, alguns estudos analisam as potencialidades das Tecnologias de Informação e Comunicação (TICs) como ferramenta imprescindível à democracia. O próprio estudo d'O reencantamento do mundo e acesso à informação [MORIGI, V. J., et al. 2016] admite uma quebra do antigo modelo de comunicação, fundamentalmente marcado pela transmissão da informação pelo denominado centro (emissor) à periferia (receptor). Com as TICs, é possível uma melhor distribuição dos emissores, uma vez que a nova forma de interação social democratiza a produção da informação por diversos agentes da informação.

Por outro lado, apesar do potencial democrático da internet, estudos como A História da sociedade da informação de Mattelart (2002) evidencia um fenômeno conhecido como “marginalização informacional”, quando o acesso às tecnologias não se dá de forma uniforme. Isso pode ser relacionado ao trabalho de João Pissarra Esteves (2011), no seu estudo Novos media e deliberação: sobre redes, tecnologia, informação e comunicação, que associa essa exclusão social a um novo tipo de digital divide para além da pobreza, compreendendo também usuários que apesar de se conectarem à internet não têm acesso às informações.

Podemos ver no estudo Portadores de deficiência: a questão da inclusão social [MACIEL, M. R. C., et al. 2010] que apesar do tema Inclusão Social e Digital vir sendo objeto de debates no meio acadêmico, governamental e empresarial, a questão da inclusão de pessoas portadoras de necessidades especiais, em todos os recursos da sociedade, ainda é muito incipiente no Brasil. Porém, a garantia da qualidade de sistemas na Web é um tema cada vez mais discutido, além da usabilidade e outras peculiaridades de interação das pessoas com o produto digital. Essa visão, baseada na perspectiva e necessidades dos usuários, abrem margem para integração e normalização das verificações de acessibilidade nos testes automatizados, que também fazem parte da qualidade de um produto e além de melhorariam a experiência dos usuários portadores de alguma deficiência também faz com que todos os tipos de usuário sejam beneficiados com as melhorias de acessibilidade, como enfatizado pelas Diretrizes de Acessibilidade para Conteúdo Web (WCAG) .



## 4. FUNDAMENTAÇÃO TEÓRICA

Em 2019 a Educação a Distância (EaD) tinha mais de 3,5 milhões de pessoas matriculadas em cursos superiores, segundo o relatório do Censo Nacional da Educação Superior publicado em 2020 (MEC, 2019). Para oferta dos cursos a distância ou até mesmo como forma de apoiar o ensino presencial as universidades têm adotado o modelo de e-Learning, comumente denominado de Ambiente Virtual de Aprendizagem (AVA). Com isso, os AVAs têm se mostrado um importante instrumento na propagação de conhecimento para aqueles que historicamente não teriam acesso a educação seja por questões financeiras, físicas ou de tempo [SHARMA, 2014].

Apesar disso, embora a EaD seja uma proposta de democratização do ensino, e as pessoas com deficiência estejam amparadas pelas leis para acesso à educação, no Brasil, a prática da acessibilidade ainda é reduzida (OLIVEIRA; SILVA, 2019).

[incluir considerações dos impactos do covid-2019 e digitalização forçada das universidades, ead temporário e tudo mais que fez usarmos mais AVAs e evidenciar o problema com deficientes em AVAs não muito acessíveis]

Para fornecer recomendações que tornem o conteúdo nos serviços e informações da Web mais acessível a um maior número de pessoas com deficiência foram criadas as Diretrizes de Acessibilidade para Conteúdo Web (WCAG). Assim, podemos testar se os AVAs foram desenvolvidos de forma a incluir os mais diversos tipos de usuário, seja com deficiência visual, auditiva, física, de fala, intelectual, de linguagem, de aprendizagem e neurológica (WCAG 2.0, 2014).

### 4.1. WCAG

As Diretrizes de Acessibilidade para Conteúdo Web (WCAG) são um conjunto de recomendações para deixar a Web mais acessível. Seguir as diretrizes deixa não apenas mais usável para pessoas com alguma deficiência como também para usuários no geral. No documento é descrito vários critérios de sucesso sob a forma de declarações testáveis, especificados de forma que não dependem de nenhuma tecnologia específica e com informações de implementação e interpretação de cada critério de sucesso em documentos separados.

Para satisfazer as necessidades de diferentes grupos e situações foram definidos os chamados três níveis de conformidade: A, AA e AAA. O nível A atinge um nível mínimo de acessibilidade. Para atingir esse nível todos os critérios de sucesso do Nível A devem ser cumpridos e assim por diante nos outros níveis de conformidade. O nível AAA é o nível mais elevado e, em relação ao nível AA, esse é o nível considerado ideal, no qual o sistema atinge um nível fundamental de acessibilidade de modo que se torna acessível para a maioria das pessoas e na maior parte das situações, usando a maioria das tecnologias.

#### **4.2. Testes Automatizados na Web**

Testar um software consiste em executá-lo de acordo como foi especificado, para determinar se ele irá comportar como esperado no ambiente para o qual ele foi projetado [IZABEL, 2014]. Como a execução de todos os testes de forma manual é dispendiosa e requer muito tempo, é normal adotar abordagens automatizadas para isso.

Testes automáticos são executados por intermédio de um programa informático ou script, que é responsável pela comparação entre os resultados atuais e os esperados [SHARMA, 2014]. Essa comparação é conhecida como assert: quando o resultado é diferente do esperado, o assert falha e seu valor é definido como falso; quando é igual, o teste passa e consideramos que a validação foi verdadeira.

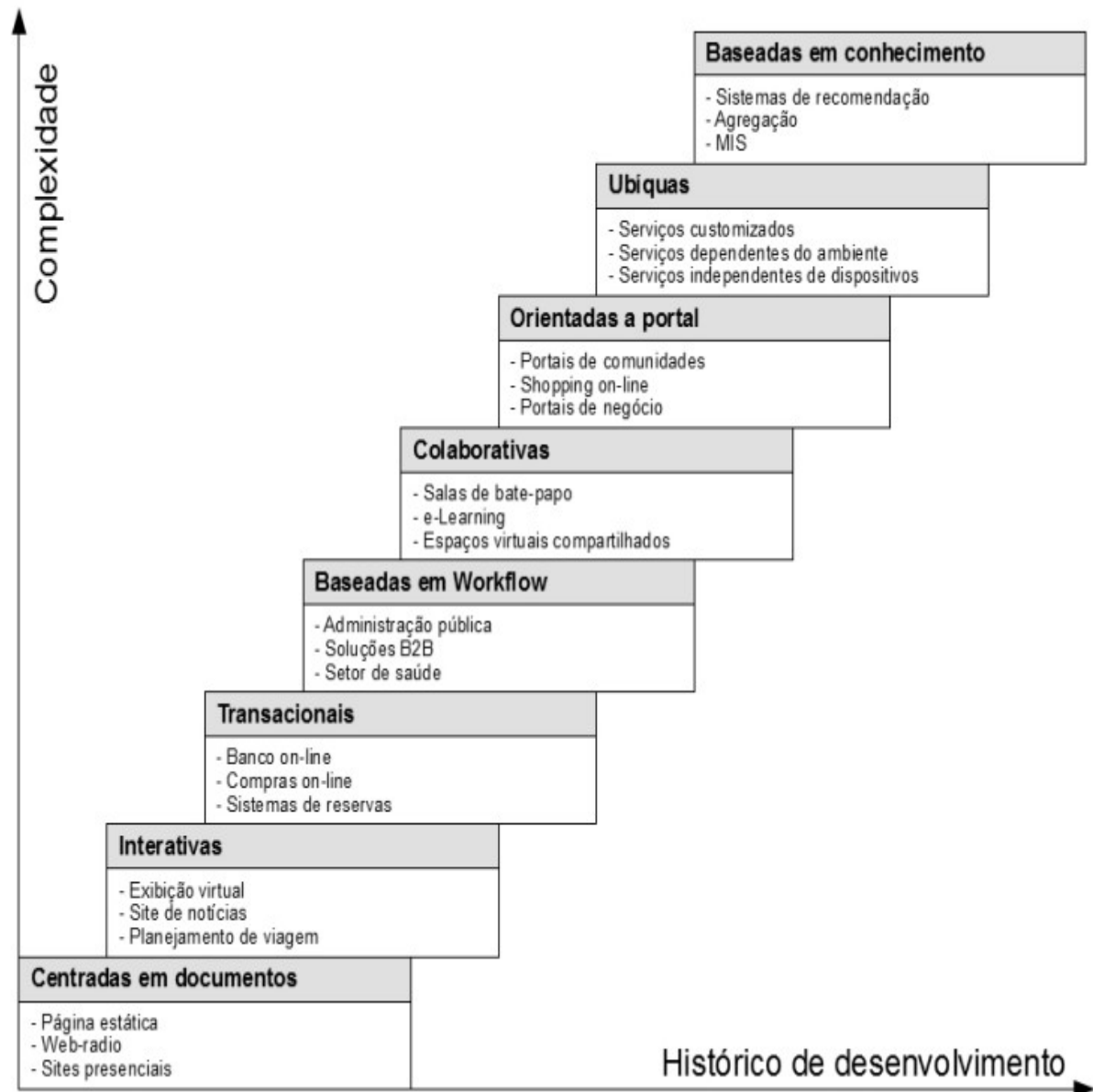
No entanto, os testes automatizados para os AVAs requerem técnicas diferentes das aplicações tradicionais. Isto porque as técnicas tradicionais foram construídas pensando nas aplicações para desktop, uma vez que a Web foi inicialmente concebida como forma de publicação de hipertextos estáticos. Isso foi destacado em 2007 por MANSOUR, afirmando que essas técnicas não consideram características das aplicações web tais como a sua natureza multi-camada, estrutura baseada em hiperlink e dirigida a evento.

Além disso, ao longo dos anos a própria Web sofreu sucessivas evoluções e modificações, apoiando aplicações de pequena e larga escala, desenvolvidas por equipes multidisciplinares com habilidades diversas e com

emprego de novas e variadas tecnologias (MENDES et al., 2006), que aliadas à rápida evolução tecnológica proporciona novos desafios para as técnicas utilizadas no desenvolvimento de software [ISABEL, 2011].

Em 2004, KAPPEL representou essas modificações categorizando as aplicações da web, de forma a compreender melhor suas características e particularidades a partir de um referencial histórico e por níveis de complexidade, onde as categorias mais novas são consideradas mais complexas.

Figura 20 – Categorias de Aplicações Web



Fonte: SANTA ISABEL, S. L., (2011). (Adaptado de KAPPEL, 2004)

Pensando nessas características, com o intuito de evitar erros comuns nos sistemas da web, foram criados frameworks como o Selenium.

#### 4.2.1. WebDriver e Selenium

WebDriver é uma API e protocolo que define uma interface de linguagem neutra para manipular o comportamento dos navegadores. Cada

implementação WebDriver específica de um navegador é chamada de driver – componente responsável por delegar ao navegador e pela comunicação do próprio navegador com o Selenium. Já o Selenium pode ser definido como um conjunto de diferentes ferramentas de software, onde cada uma possui um propósito específico para auxiliar no processo de automação de testes [Documentação do Selenium, 2019].

Na configuração do Selenium é definida e instalada as bibliotecas de linguagem – para linguagem escolhida do projeto de automação – e os binários dos drivers que executarão os testes no navegador. Graças a isso, além da possibilidade de ser controlado por diferentes linguagens de programação e frameworks de testes, o Selenium suporta a automação de praticamente todos os principais navegadores do mercado (Firefox, Google Chrome, Edge, Opera, Safari).

#### **4.3. Testes Automatizados de Acessibilidade**

Além dos frameworks voltados para testes automatizados na Web também foram criados métodos para garantir uma melhor experiência do usuário nessas aplicações. Por exemplo, levando em consideração a usabilidade, que pode ser definida como o fator que assegura que um produto ou serviço seja fácil de usar, eficiente e agradável a partir do ponto de vista do usuário [PREECE; ROGERS; SHARP, 2007] temos as heurísticas de Nielsen, baseada em 294 tipos de erro de usabilidade que Jakob Nielsen, Ph.D., diretor da Nielsen Norman Group, encontrava em suas análises.

Mas em relação a acessibilidade, que considera a diversidade de seus possíveis usuários e as peculiaridades da interação dessas pessoas com o produto [TORRES; MAZZONI, 2004], muitas das vezes os desenvolvedores e especialistas em Garantia de Qualidade (QA) não consideram as diretrizes e padrões da WCAG no design e na test suite (conjunto de casos de teste) das aplicações, gerando uma série de problemas para o usuário com alguma deficiência.

Nesse contexto que surgiram as engines como Axe e Pa11y, que verificam se as regras da WCAG foram devidamente implementadas e executam testes automatizados de acessibilidade nas páginas de uma aplicação

da web. Elas foram arquitetadas e desenvolvidas para integrar com qualquer ambiente de teste já existente em um projeto, de forma que as organizações possam automatizar os testes de acessibilidade junto dos seus testes regulares.

#### **4.4. Ferramentas automatizadas para testes de acessibilidade**

##### **4.4.1. aXe**

O aXe é uma engine de testes de acessibilidade open-source para validação das recomendações sugeridas pela WCAG. Foi projetada para integrar com qualquer ambiente de teste existente, possibilitando a automatização dos testes de acessibilidade em conjunto com os testes funcionais regulares no processo de desenvolvimento de um sistema web. O aXe oferece uma série de ferramentas além da API (axe-core), como utilitários de CLIs (Interfaces de linha de comando) e a sua extensão para o Google Chrome.

A axe-core integra-se com o Selenium e funciona basicamente como uma ferramenta que definirá como automatizar os testes da página seguindo regras para o WCAG 2.0 e 2.1 nos níveis A e AAA. Com ela, é possível encontrar em média 57% dos problemas previstos no WCAG automaticamente.

##### **4.4.2. Pa11y**

O Pa11y também possui uma ferramenta CLI de teste de acessibilidade automatizado, que carrega páginas da web e destaca todos os problemas de acessibilidade encontrados. Além disso, também disponibiliza um serviço de Dashboard que faz os testes diariamente e ilustra melhor seus resultados para não-desenvolvedores, com gráficos que rastreiam melhorias e regressões ao longo do tempo. Esse painel também conta com um Webservice (Pa11y Webservice) baseado em JSON que dá suporte à criação

de front-ends com o dashboard personalizado ou com redirecionamento dos dados.

#### **4.4.3. Google Lighthouse**

O Google Lighthouse é uma ferramenta automatizada de código aberto que utiliza a biblioteca do aXe (axe-core) para fornecer um conjunto de testes de acessibilidade. Ela roda todas as regras marcadas com os tipos wcag2a e wcag2aa, apesar de desabilitar alguns itens específicos. O Lighthouse foi adicionada ao Chrome DevTools a partir da versão 60 do Google Chrome e também pode ser usado como uma ferramenta de linha de comando. Ele foi criada para oferecer uma auditoria abrangente de todos os aspectos de Qualidade de um app da Web.

Figura 1 – Regras WCAG definidas no Lighthouse

```
22 function runAllChecks() {
23   // @ts-ignore axe defined by axeLibSource
24   return window.axe.run(document, {
25     elementRef: true,
26     runOnly: {
27       type: 'tag',
28       values: [
29         'wcag2a',
30         'wcag2aa',
31       ],
32     },
33     resultTypes: ['violations', 'inapplicable'],
34     rules: {
35       'tabindex': {enabled: true},
36       'accesskeys': {enabled: true},
37       'table-fake-caption': {enabled: false},
38       'td-has-header': {enabled: false},
39       'marquee': {enabled: false},
40       'area-alt': {enabled: false},
41       'blink': {enabled: false},
42       'server-side-image-map': {enabled: false},
43     },
44     // @ts-ignore
45   }).then(axeResult => {
46     // Augment the node objects with outerHTML snippet & custom path string
47     // @ts-ignore
48     axeResult.violations.forEach(v => v.nodes.forEach(node => {
```

Fonte: Código retirado do [repositório do Lighthouse](#) no Github. 2021.

Atualmente, o Lighthouse cobre, além dos testes de acessibilidade, testes de desempenho, progressividade, boas práticas e SEO. A ferramenta também pode fazer testes para desktop ou mobile, no último caso usando emuladores de celular durante a auditoria.



### **3. METODOLOGIA E DESENVOLVIMENTO**

Foi realizada uma pesquisa exploratória com o intuito de levantar procedimentos a serem adotados para se testar a acessibilidade de sistemas da Web. Com isso, o estudo foi realizado através das principais diretrizes e organizações de padronização que desenvolvem os pilares de tecnologias para se testar acessibilidade, de modo que antes de apurar as possíveis soluções sejam identificadas da melhor maneira possível as falhas de acessibilidade. Através das pesquisas nas WCAG também foi possível a identificação de ferramentas de avaliação automática para os testes de acessibilidade, que foram aplicadas em um estudo de caso do Ambiente Virtual de Aprendizagem utilizado pela Unifeso – Centro Universitário Serra dos Órgãos.

#### **3.1. Ferramentas utilizadas na avaliação**

[Escreva aqui]Os testes automatizados foram realizados usando o Google Lighthouse,

#### **3.2. Escopo e páginas da auditoria**

Foram analisadas 89 páginas do AVA a partir de um login com perfil de estudante. 178 testes automatizados foram executados – 1 por página – e encontradas, em um primeiro momento, um total de 600 falhas de acessibilidade. Ainda é preciso analisar as páginas com privilégio administrativo, disponíveis apenas no login de professores.

Em segunda instância, verificou-se que algumas páginas estavam sendo afetadas pelos mesmos componentes, o que causou duplicidades na análise ainda que os impactos na acessibilidade da plataforma afetem individualmente e repetitivamente a experiência do usuário com alguma deficiência. A pesquisa seguiu identificando em cada resultado por página, manualmente, os problemas de acessibilidade que já haviam sido acusados pelos testes em páginas anteriores. Foram consideradas duplicatas as análises sobre os mesmos componentes em duas páginas diferentes, que acusavam as mesmas vulnerabilidades. O mesmo tipo de falha de acessibilidade, encontrado em

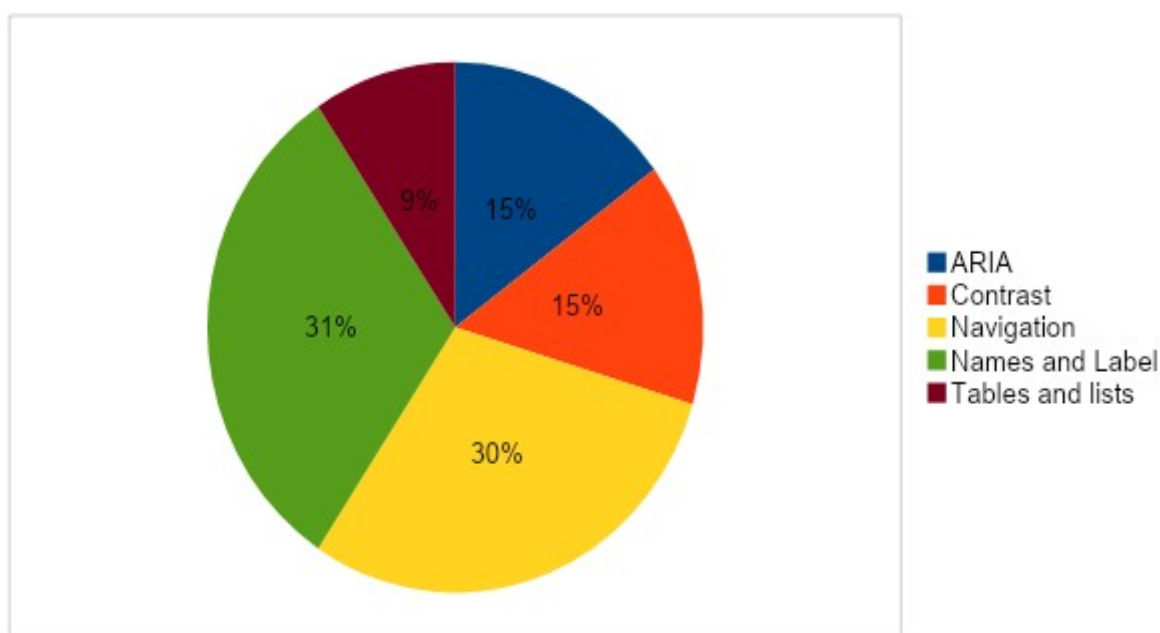
elementos distintos, também foi analisado separadamente considerando que apesar de ter a mesma solução ela deve ser aplicada em contextos diferentes, dependendo do elemento. Nessa fase, o total de falhas de acessibilidade passou a ser de 148 vulnerabilidades.

Durante a análise de duplicidade, os resultados também mostraram que a estrutura das páginas e seções é que continham os problemas de acessibilidade e não os dados ou valores taggeados. Com isso, foi revelado também que o mesmo tipo de página ou seção apresentavam os mesmos problemas de falha de acessibilidade ainda que fizesse parte de uma disciplina diferente. Por exemplo, a disciplina “2021/1 – DESENVOLVIMENTO DE APLICAÇÕES MÓVEIS” possuía as mesmas falhas de acessibilidade na seção Conteúdo das Aulas que a disciplina “2021/1 – COMPUTAÇÃO GRÁFICA E PROCESSAMENTO DE IMAGENS” na mesma seção. Ou seja, apesar dos testes automatizados terem sido executados em cima das disciplinas ofertadas para o 6º/7º período flex. das turmas semestrais de Ciência da Computação em 2021, os resultados cobrem todas as disciplinas do curso que apresentam as mesmas seções.

## 5. RESULTADOS E DISCUSSÕES

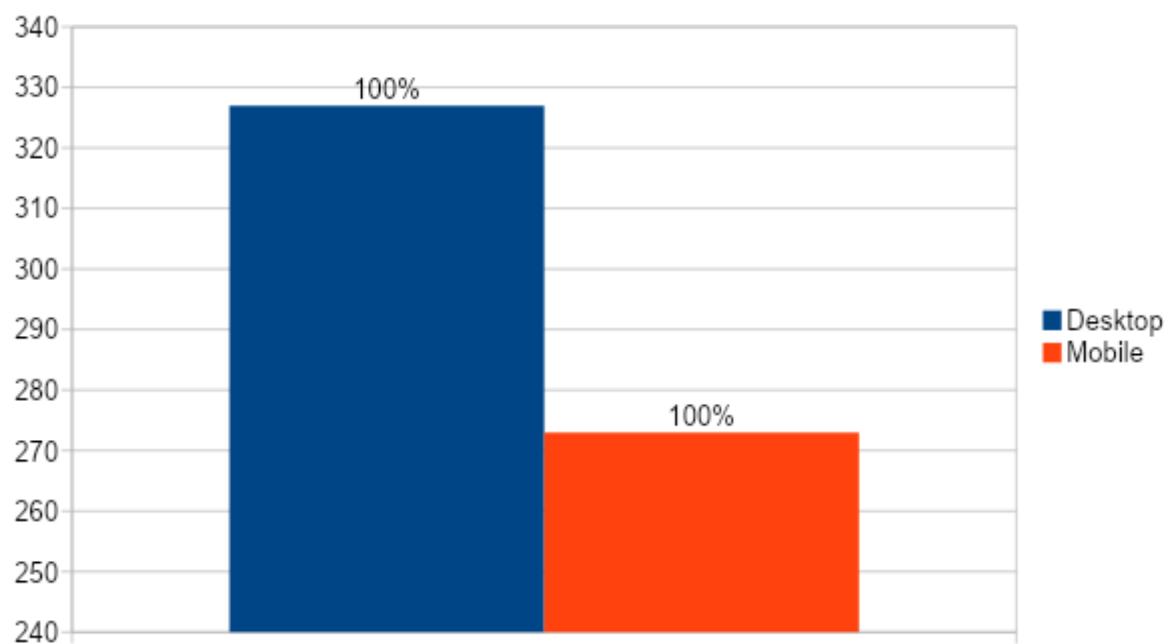
A avaliação identificou os problemas chaves de acessibilidade, mais frequentes e como estão distribuídos, que se relacionam com os fatores de pontuação e classificação de acessibilidade providos pelo próprio Google Lighthouse. A partir desse resultado, foram levantadas 12 propostas de soluções que resolveriam as falhas de acessibilidade e que devem ser aplicadas contextualmente a cada elemento acusado pelo Lighthouse. As soluções, quando implementadas, impactariam consideravelmente a pontuação de acessibilidade do AVA, sendo esperada uma mudança na sua classificação de laranja para verde – considerada ideal pela ferramenta do Google.

Figura 2 – Gráfico representando os problemas de acessibilidade classificados por tipo



Fonte: Próprio autor. 2021.

Figura 3 – Gráfico representando os problemas de acessibilidade por dispositivo



Fonte: Próprio autor. 2021.

## 5.2. Sugestões de melhoria para resolver os asserts que falharam

### 5.2.1. ARIA input fields do not have accessible names

A falha de acessibilidade número 1 diz respeito aos elementos que não têm um valor de ARIA role apropriado e por isso não podem ser anunciados adequadamente aos usuários que utilizam leitores de tela. O Lighthouse tem algumas auditorias que cobrem um conjunto diferente de funções ARIA, dentre elas o conjunto conhecido como aria-input-field-name que cuida das roles combobox, listbox, searchbox, slider, spinbutton e textbox. Essa auditoria que, por exemplo, faz com que a div abaixo falhe no teste de acessibilidade.

```
<div class="carousel-inner" role="listbox">
```

O problema pode ser resolvido, conforme listado na seção 2.1 da Accessible Rich Internet Applications (WAI-ARIA) 1.1, subtópico listbox (role), adicionando o atributo aria-label ao elemento, que permite os leitores de tela e outras tecnologias assistivas anunciar seu valor para o usuário. Dessa forma, o problema acima poderia ser resolvido refatorando o exemplo da seguinte maneira:

```
<div class="carousel-inner" role="listbox" aria-label="Texto descritivo aqui">
```

### 5.2.2. Elements with an ARIA [role] that require children to contain a specific [role] are missing some or all of those required children.

A falha de acessibilidade número 2 acontece quando uma ARIA role é atribuída a um elemento com propósito de dizer aos leitores de tela e outras tecnologias assistivas qual o comportamento e os controles customizados que um componente da aplicação tem. Algumas dessas roles exigem que os filhos do elemento também tenham roles

específicas que trabalham em conjunto com a do pai. Por exemplo, a role tablist exige que os filhos tenham a role tab.

No exemplo abaixo, podemos ver que a div acusada com falha de acessibilidade possui uma role listbox, que por sua vez está associada as roles options.

Figura 4 – Bloco de código pertencente à página inicial do AVA mostrando a associação das roles listbox e options

```
▼ <main id="moodle-page" class="clearfix">
  ▼ <div id="page-header" class="clearfix ">
    ▼ <div id="snap-site-carousel" class="carousel slide" data-ride="carousel">
      ▼ <div class="carousel-inner" role="listbox"> == $0
        ▶ <div class="carousel-item carousel-slide_on e active">...</div>
        ▶ <div class="carousel-item carousel-slide_two ">...</div>
        ▶ <div class="carousel-item carousel-slide_three ">...</div>
        ::after
      </div>
      ▶ <div class="carousel-controls">...</div>
      ▶ <ol class="carousel-indicators js-only">...
    </ol> flex
  </div>
```

Fonte: Próprio autor. 2021.

A solução para este tipo de falha de acessibilidade também é encontrada na seção WAI-ARIA 1.1, no subtópico option (role), que afirma a necessidade de adicionar aos elementos filhos de uma div com a role listbox o atributo role com valor option. Caso contrário, a especificação adverte que os elementos não serão corretamente mapeados pela API de acessibilidade. Sendo assim, o problema poderia ser resolvido refatorando o case acima da seguinte maneira:

Figura 5 – Acréscimo das roles options para criar uma associação com a listbox

```
▼<main id="moodle-page" class="clearfix">
  ▼<div id="page-header" class="clearfix ">
    ▼<div id="snap-site-carousel" class="carousel s
      lide" data-ride="carousel">
        ▼<div class="carousel-inner" role="listbox"> == $0
          ▶<div class="carousel-item carousel-slide_on
            e active" role="option">...</div>
          ▶<div class="carousel-item carousel-slide_tw
            o " role="option">...</div>
          ▶<div class="carousel-item carousel-slide_th
            ree " role="option">...</div>
          ::after
        </div>
```

Fonte: Próprio autor. 2021.

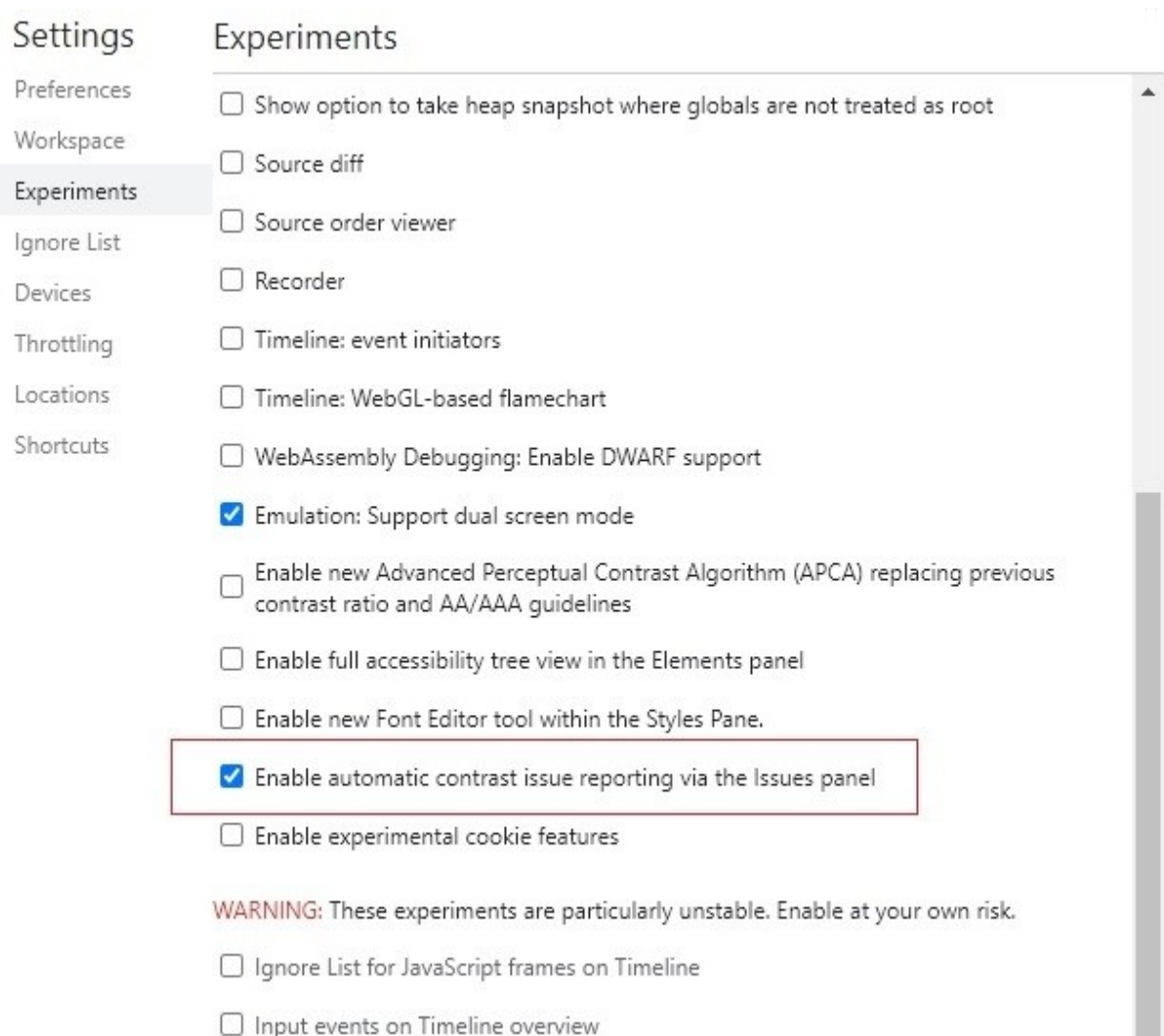
### 5.2.3. Background and foreground colors do not have a sufficient contrast ratio.

Textos que não têm contraste o suficiente, além de afetarem principalmente usuários com baixa visão, também dificultam a leitura de todos os tipos de usuários. Isso pode ser notado, por exemplo, ao tentar ler algo no celular a partir de um ambiente externo e iluminado. Para resolver isso os critérios de sucesso mínimos (Nível AA) da WCAG 2.1 exigem uma taxa de contraste de pelo menos 3:1 para textos grandes – também definidos pelas diretrizes como textos maiores que 18pt se não estiverem em negrito e 14pt se estiverem. Para os demais tamanhos de texto a taxa de contraste é de 4.5:1.

Na falha de acessibilidade número 3 o teste automatizado identificou, nos elementos apontados por ele, que a taxa de contraste é menor do que as taxas exigidas pela WCAG. Para resolver o problema, é recomendada a utilização de alguma ferramenta que mensure e redefina o contraste de um texto. Atualmente, existem várias ferramentas criadas com esse propósito, como a Chrome DevTools' Color Picker do próprio Google, a WCAG Color Contrast Checker e a Contrast Grid.

A ferramenta utilizada na pesquisa foi a Chrome DevTools' Color Picker pela integração com o Google Chrome, fazendo um link entre as análises automatizadas do Lighthouse e facilitando a identificação e apuração do elemento com falha na acessibilidade. Ela pode ser habilitada nas configurações do DevTools, na seção Experiments, como ilustrado mais abaixo na figura 6.

Figura 6 – Habilitação da Chrome DevTools' Color Picker no DevTools

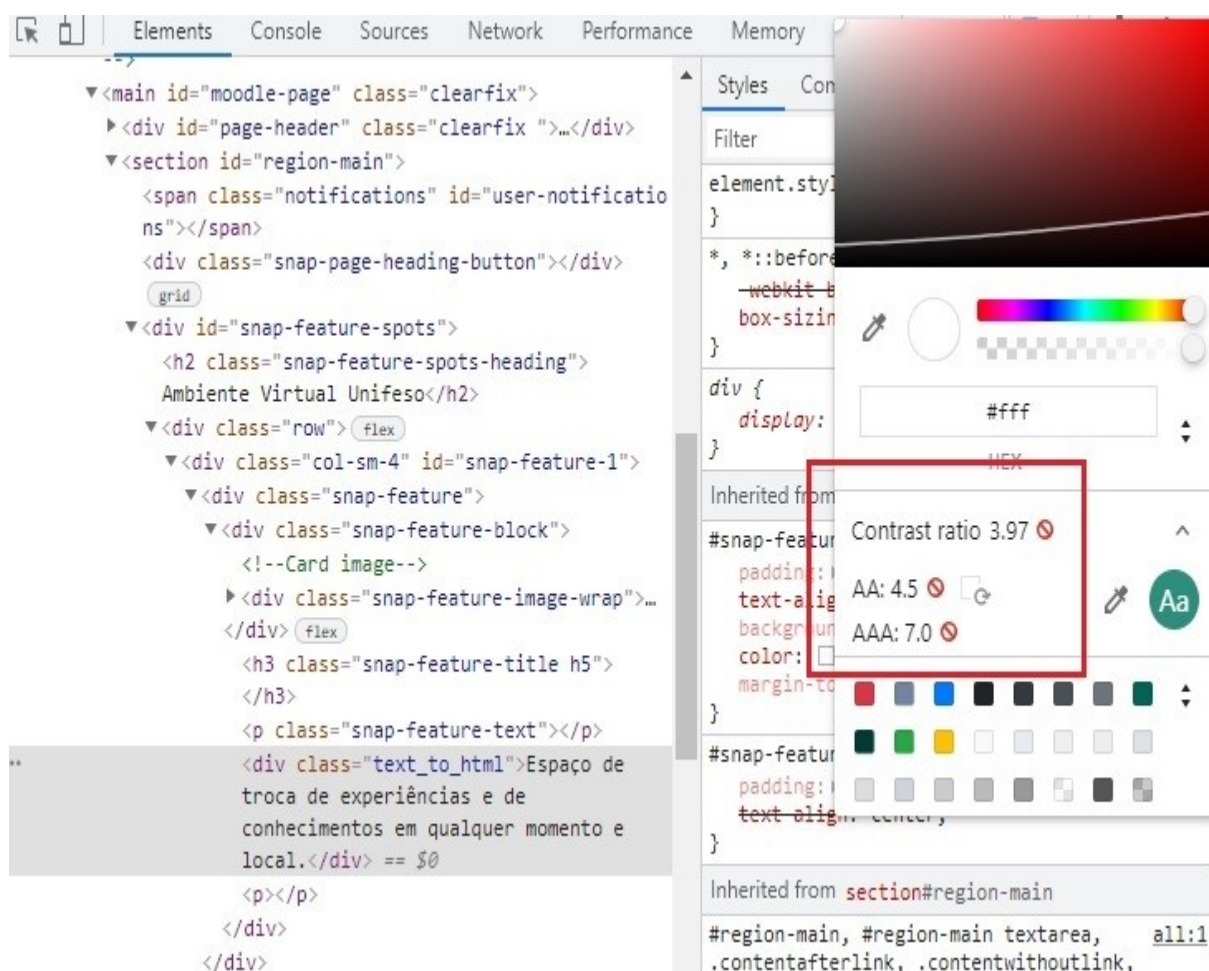


Fonte: Próprio autor, 2021.



Para utilização da ferramenta, depois de habilitada, basta inspecionar o elemento que terá a taxa de contraste apurada, identificar o valor de sua cor na guia Styles do DevTools e clicar na *thumbnail* à esquerda do valor. À direita do agrupamento Contrast ratio, poderá ser visualizada a taxa de contraste do elemento. Abaixo, o contraste sugerido pela WCAG no nível AA e no nível AAA. Este procedimento está ilustrado adiante na figura 7.

Figura 7 – Apuração da taxa de contraste com o Chrome DevTools' Color Picker



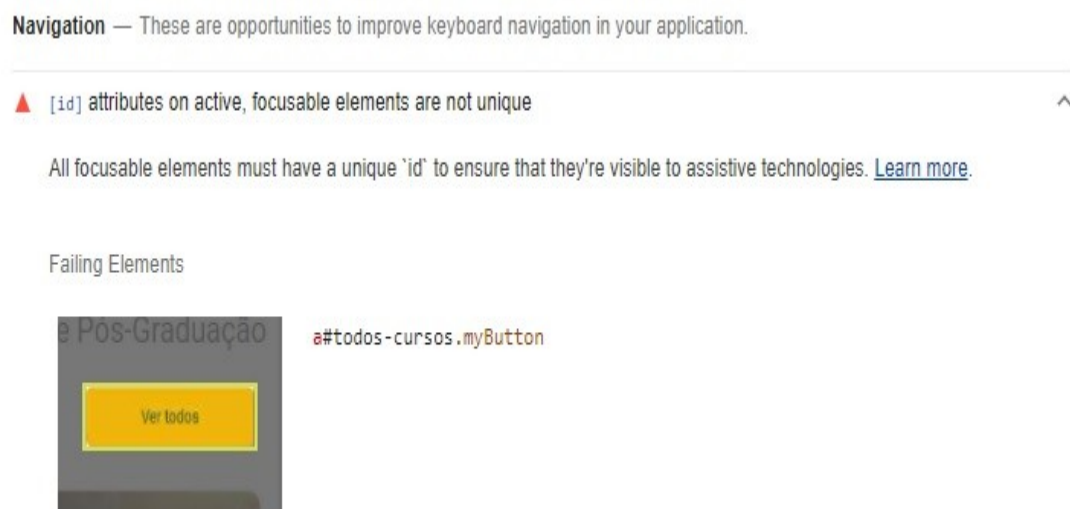
Fonte: Próprio autor. 2021.

#### 5.2.4. [id] attributes on active, focusable elements are not unique

O atributo id no HTML, além de ser utilizado para identificar o elemento em scripts e no CSS, também responde a leitores de tela e outras tecnologias assistivas. Essas tecnologias, por esperar que o id seja único por todo o documento, acabam anunciando somente o primeiro elemento que compartilha o id em casos onde o mesmo id é definido em mais de um elemento, como informado nas Técnicas para os Critérios de Sucesso 4.1.1 da WCAG 2.0. Dessa forma, os ids duplicados tornam apenas o primeiro elemento focalizável nas navegações por tab e shift + tab, fazendo com que o usuário não consiga acessar todas as funcionalidades da página.

A falha de acessibilidade número 4 indica que há mais de um elemento compartilhando o mesmo id, como na acusação de id redundante no exemplo abaixo.

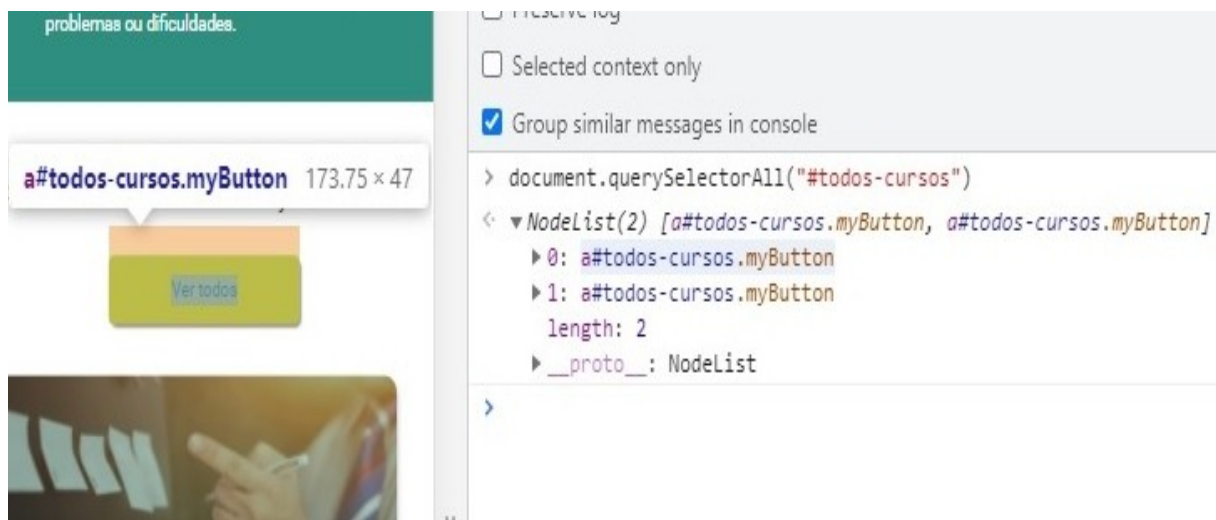
Figura 8 – Assert do Lighthouse acusando mais de um elemento com o mesmo Id



Fonte: Próprio autor, 2021.

Para resolver esse problema é preciso identificar quais elementos têm o mesmo id e modificá-los para que cada um tenha seu próprio id. Uma alternativa é cogitar retirar o id dos elementos e manipulá-los por suas classes.

Figura 9 – Identificação dos elementos compartilhando o mesmo Id

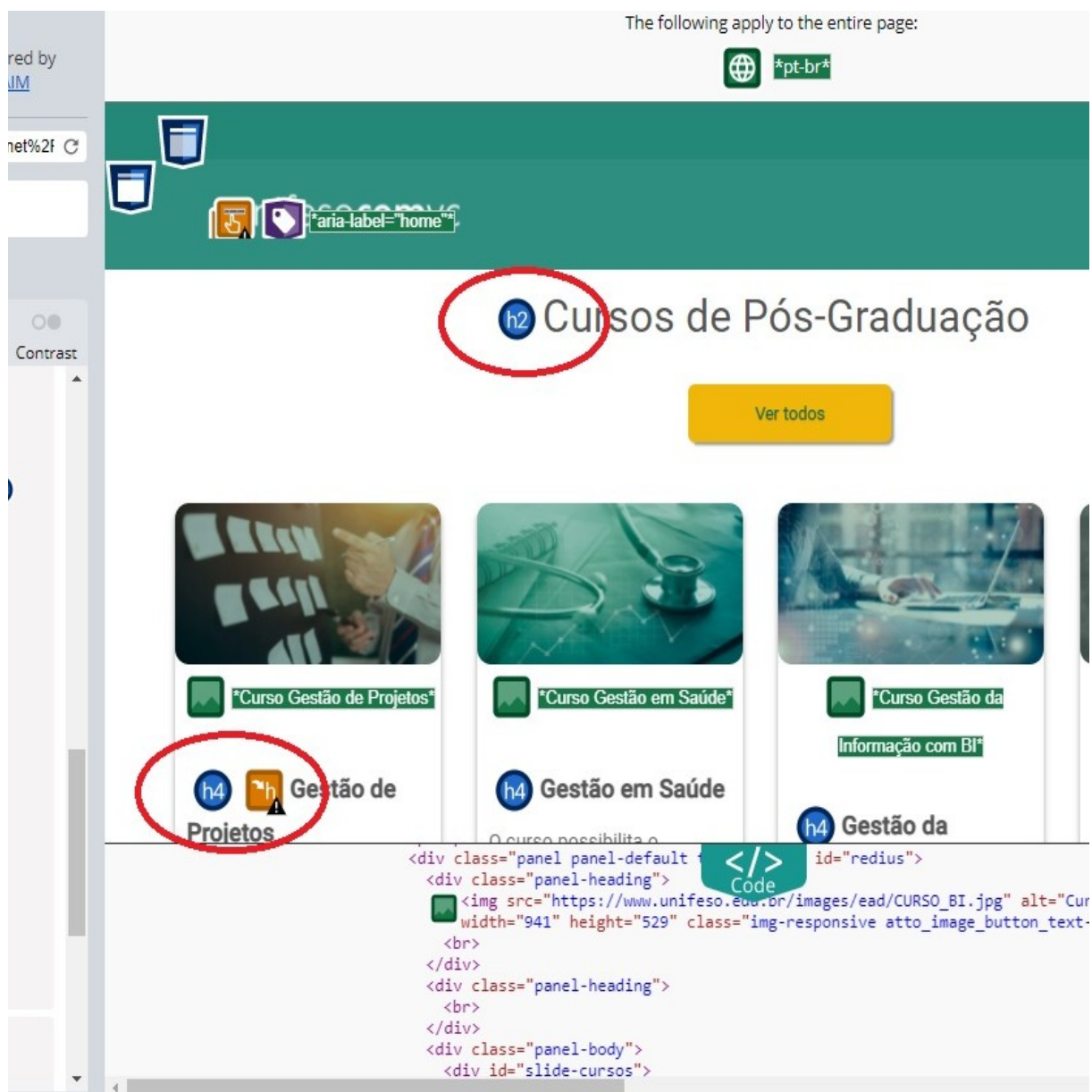


Fonte: Próprio autor, 2021.

#### 5.2.5. Heading elements are not in a sequentially-descending order

Os chamados headings no HTML são definidos com as tags de <h1> até <h6> e representam os seis níveis de título de seção, assim especificado na seção 4.3.9 das recomendações do W3C para o HTML 5.2. Um erro muito comum é utilizar esses elementos para marcar slogans, títulos alternativos e subtítulos que não pretendem representar semanticamente o título de uma nova seção, além da utilização dos seus níveis para diminuir ou aumentar o tamanho da fonte do cabeçalho. Esses enganos costumam desviar a estrutura da ordem não-sequencial e descendente – ideal para que leitores de tela consigam navegar de heading a heading.

Figura 10 – Análise da acessibilidade dos headings na página inicial do AVA usando a extensão Web Developer



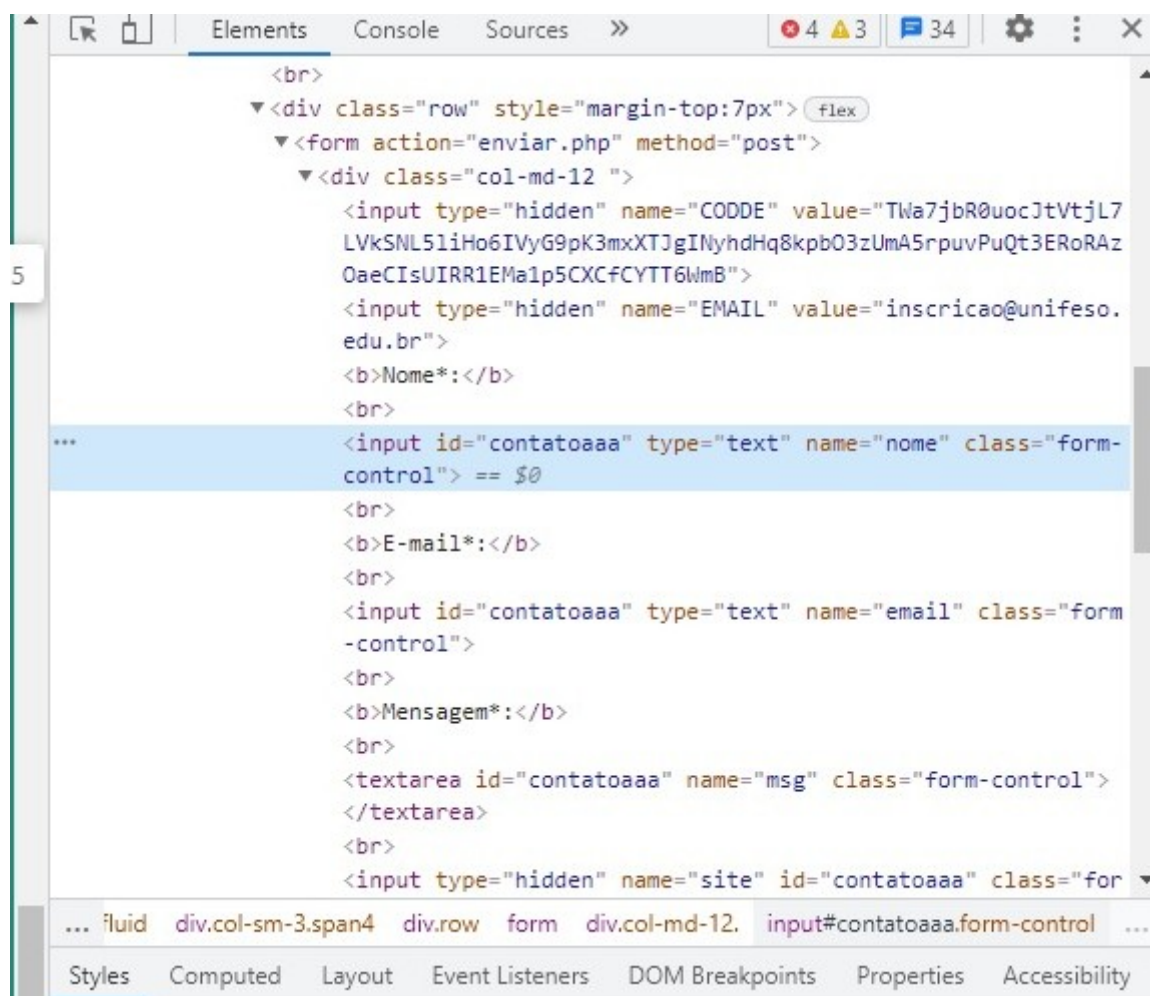
Fonte: Próprio autor. 2021.

Na figura acima, através da extensão Web Developer do Google Chrome, podemos visualizar que os elementos acusados pelo teste automatizado estão todos no nível h4, apesar dos headings anteriores estarem no nível h2. Para solucionar o problema, os headings devem ficar ordenados decrescente e sequencialmente, mudando os h4s da div com id “radius” para h3s e, assim, removendo o gap na sequência dos headings.

### 5.2.6. Form elements do not have associated labels

Esta falha de acessibilidade, representada como F68 nas técnicas para o WCAG 2.0, falha dos critérios de sucesso 1.3.1 e 4.1.2, acontece quando não há um label associado a cada um dos elementos de controles de um formulário. Os labels garantem que esses elementos sejam anunciados apropriadamente pelos leitores de tela, além de serem utilizados por outras tecnologias assistivas para que os usuários consigam navegar através do formulário. Na figura abaixo, por exemplo, podemos notar a ausência de um label para os inputs apresentados.

Figura 11 – Nenhum label associado aos inputs do form



Fonte: Próprio autor, 2021.

O método recomendado para a maioria das circunstâncias é usar o elemento `label` e uma associação explícita através dos atributos `for` e `id`.

#### 5.2.7. Image elements do not have [alt] attributes

O documento Técnicas do W3C para WCAG 2.0 contém orientações específicas sobre como atender aos critérios de sucesso das WCAG. Ele é atualizado periodicamente, cerca de duas vezes por ano, para cobrir as práticas recomendadas mais atuais e mudanças em tecnologias e ferramentas.

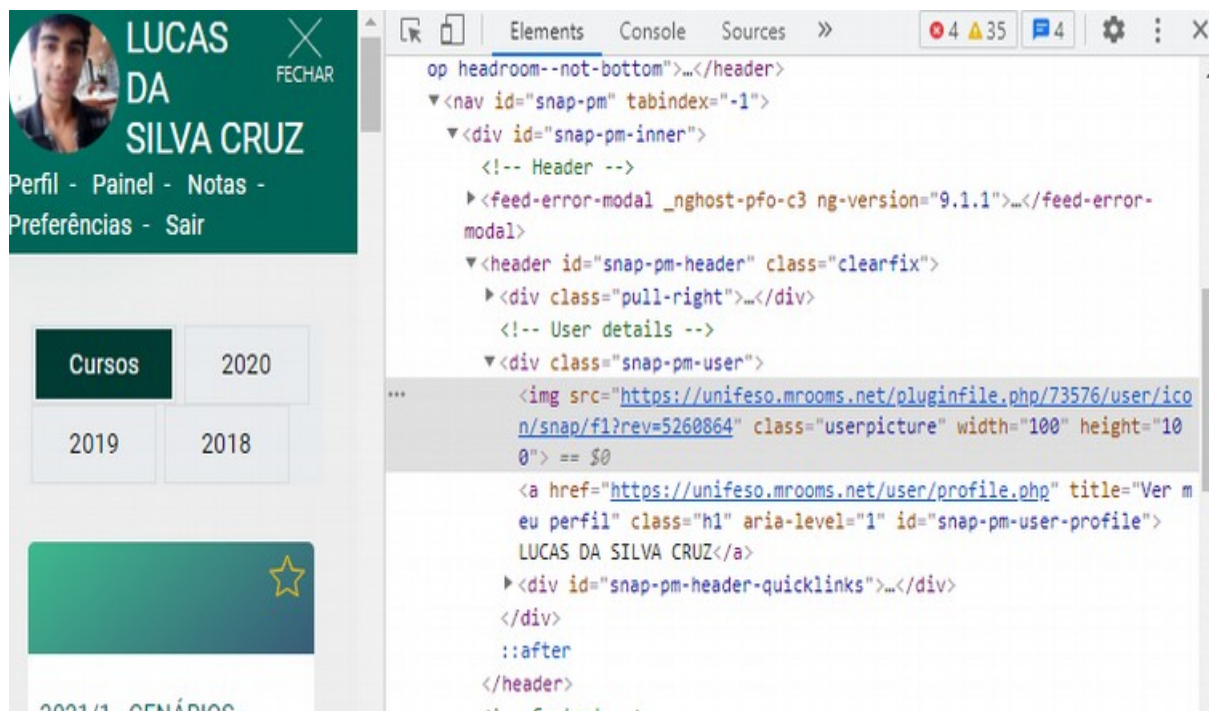
A falha de acessibilidade F68 desse documento, que acusa elementos do tipo `img` que não têm o atributo `alt`, justifica que sem esse atributo um texto alternativo não pode ser computado caso a imagem tenha algum problema para ser carregada. Ainda que algumas tecnologias assistivas tentem compensar a falta do texto alternativo lendo o nome do arquivo da imagem, ainda é insuficiente na medida em que nomes de arquivos geralmente não são descritivos (p. ex. `images/nav01.gif`).

Nos casos em que a imagem atua apenas como decoração e não fornece nenhum conteúdo útil de fato, ainda é uma boa prática atribuir o atributo `alt = ""` (vazio) para remover a falha da árvore de acessibilidade.

Embora existam os atributos WAI-ARIA que podem ser usados para fornecer um texto alternativo, desde que sejam compatíveis com a acessibilidade, ainda é recomendado pelas Técnicas do W3C para WCAG 2.0 o atributo `alt` como a forma preferida de resolver essa falha de acessibilidade.



Figura 12 – Ausência do atributo alt na tag img



Fonte: Próprio autor. 2021.

Na figura acima podemos perceber a ausência do atributo alt, aria-label ou aria-labelledby associado a algum id. Como também é um elemento contêdístico – que representa a imagem de perfil de algum usuário – é recomendado, para solucionar o problema, uma simples adição do atributo alt com um valor descritivo da imagem.

#### 5.2.8. <object> elements do not have [alt] text

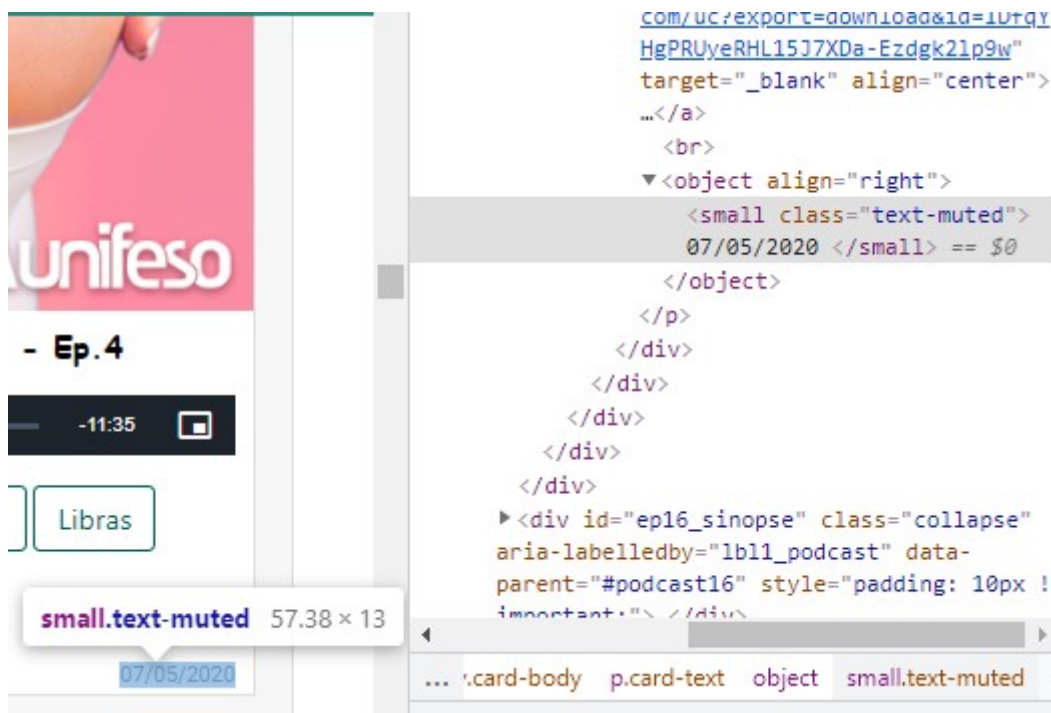
O elemento object representa um recurso externo que pode ser tratado como uma imagem, um contexto de navegação aninhado ou um recurso a ser provido por um plugin. [HTML elements reference, MDN Web Docs]. Dessa forma, a mídia desse elemento só está disponível para o usuário quando ela não é renderizada pelo agente do usuário – o agente pode não oferecer suporte à tecnologia de mídia ou o usuário o instruiu

a não renderizar essa tecnologia. [Falha de acessibilidade H53 das Técnicas do W3C para WCAG 2.0].

Nos casos em que os leitores de tela e outras tecnologias assistivas não conseguem interpretar o conteúdo do elemento object, ele deve oferecer um texto alternativo que permita transmitir o significado do elemento para os usuários. Essa falha H53 recomenda que o texto alternativo seja inserido no próprio corpo do elemento, como exemplificado abaixo:

```
<object type="application/pdf" data="/report.pdf">
  2019 Web Accessibility Report
</object>
```

Figura 13 – Ausência do atributo alt no corpo do elemento object



Fonte: Próprio autor. 2021.



Na figura acima pode ser notado que não há um texto alternativo inserido diretamente no corpo do elemento object, por isso ele é acusado com uma falha de acessibilidade. A solução recomendada para este tipo de problema é simplesmente adicionar um texto descritivo seu corpo.

#### 5.2.9. <frame> or <iframe> elements do not have a title

É demonstrada na técnica H64 do documento Techniques and Failures for Web Content Accessibility Guidelines 2.0. o uso do atributo title nos elementos frame e iframe: “O atributo title fornece um label para o frame e assim os usuários podem determinar qual frame querem entrar e explorar em detalhes.” (tradução nossa).

Os leitores de tela e outras tecnologias precisam desse atributo nos frames ou iframes para descrever o conteúdo deles. Sem o atributo, navegar por esses elementos pode se tornar muito rapidamente difícil e confuso para o usuário de alguma tecnologia assistiva.

Figura 14 – Ferramenta Lighthouse acusando elemento iframe de não ter atributo title

▲ <frame> or <iframe> elements do not have a title

Screen reader users rely on frame titles to describe the contents of frames. [Learn more.](#)

##### Failing Elements



iframe

```
<iframe src="https://player.vimeo.com/video/388218461" width="100%" height="auto"
frameborder="0" allow="autoplay; fullscreen" allowfullscreen="">
```

Fonte: Próprio autor. 2021.

#### 5.2.10. ARIA progressbar elements do not have accessible names.

“A role progressbar indica que a requisição do usuário foi realizada e o aplicativo está progredindo para concluir a ação solicitada.” (WAI-ARIA, 2017, v. 1.1, tradução nossa). Quando esse elemento não tem um nome acessível, os leitores de tela e outras tecnologias assistivas o anunciam com um nome genérico, inutilizável para o usuário que depende das tecnologias. Como por exemplo no elemento abaixo acusado pelo Lighthouse com essa falha:

Figura 15 – Lighthouse acusando elemento com nome inacessível para leitores de tela

##### Failing Elements



div

```
<div class="progress-bar bar" role="progressbar" aria-valuenow="35" style="width: 35%" aria-valuemin="0" aria-valuemax="100">
```

Fonte: Próprio autor. 2021.

Uma das formas de resolver esse problema, quando não queremos que o nome do elemento seja visível na página, é através do atributo `aria-label`, que tem justamente o propósito de informar um nome mais apropriado. O código acima poderia, então, ser refatorado da seguinte maneira:

```
<div class="progress-bar bar" role="progressbar" aria-label="nome-anunciado-pelas-tecs-assistivas" aria-valuenow="35" style="width: 35%" aria-valuemin="0" aria-valuemax="100">
```

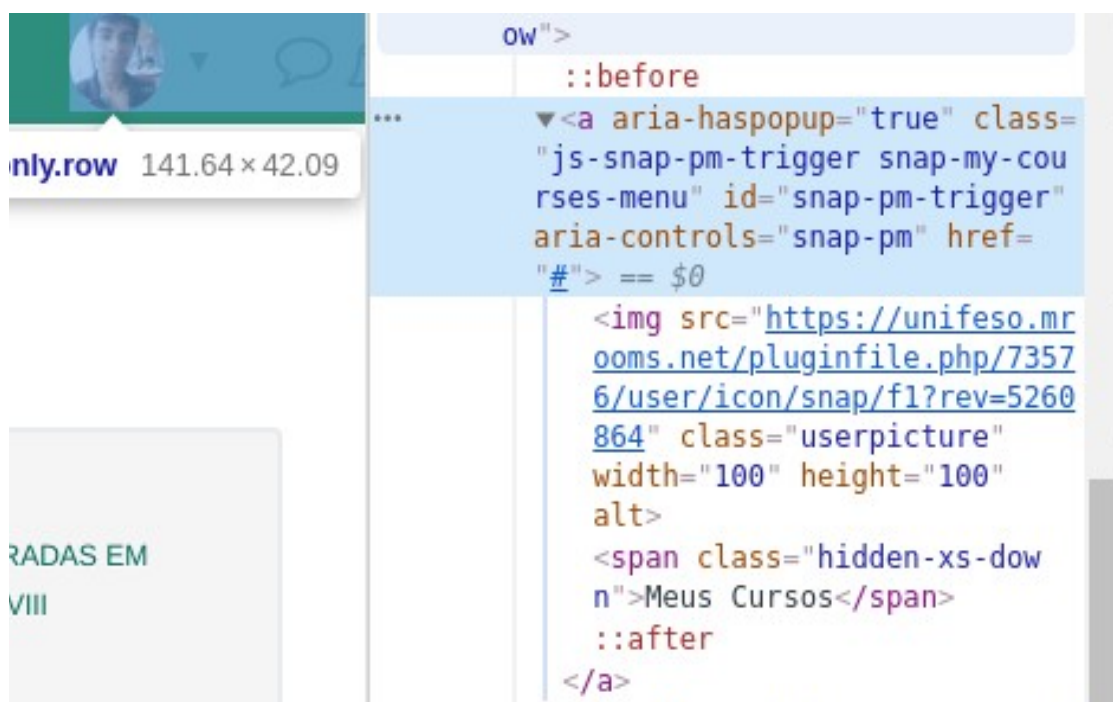
### 5.2.11. Links do not have a discernible name

Um link deve sempre prover no seu conteúdo um texto descritivo. “A descrição permite o usuário distinguir esse link de outros na página e também determinar se ele deve ou não seguir o link uma vez que a URI de destino geralmente não é suficientemente descritiva.” (Techniques and Failures for Web Content Accessibility Guidelines, 2016, v. 2.0, tradução nossa).

O documento também antecipa as situações onde uma imagem é o único conteúdo de um link. Nesse caso, ele diz que a alternativa em texto para imagem – atributo alt – descreve a função exclusiva do link.

A figura 16 exibe o trecho de código onde o Lighthouse identificou o problema. Como podemos ver, não há um texto descritivo contido diretamente no link de id “snap-pm-trigger”. Apesar de ter uma imagem, também não é fornecido um atributo alt que serviria como descrição da função do link.

Figura 16 – Um elemento de link que não fornece texto descritivo



Fonte: Próprio autor. 2021.

Para resolver esse problema basta adicionar algum valor ao atributo alt da tag img. Alternativamente, também é possível utilizar a aria-label, considerando que é uma situação muito próxima da analisada anteriormente no ponto 5.2.10, onde era necessário um texto descritivo para o elemento sem que ele fosse renderizado de fato na página. A figura abaixo representa a implementação dessa solução:

Figura 17 – Acréscimo de aria-label a um link sem texto descritivo

```
▼<a aria-haspopup="true" class="js-snap-pm-trigger snap-my-  
courses-menu" id="snap-pm-trigger" aria-controls="snap-pm"  
href="#" aria-label="nome-anunciado-pelas-tecs-assistivas">  
    
  <span class="hidden-xs-down">Meus Cursos</span>  
  ::after  
</a>  
►<div class="badge-count-container">...</div>
```

Fonte: Próprio autor. 2021.

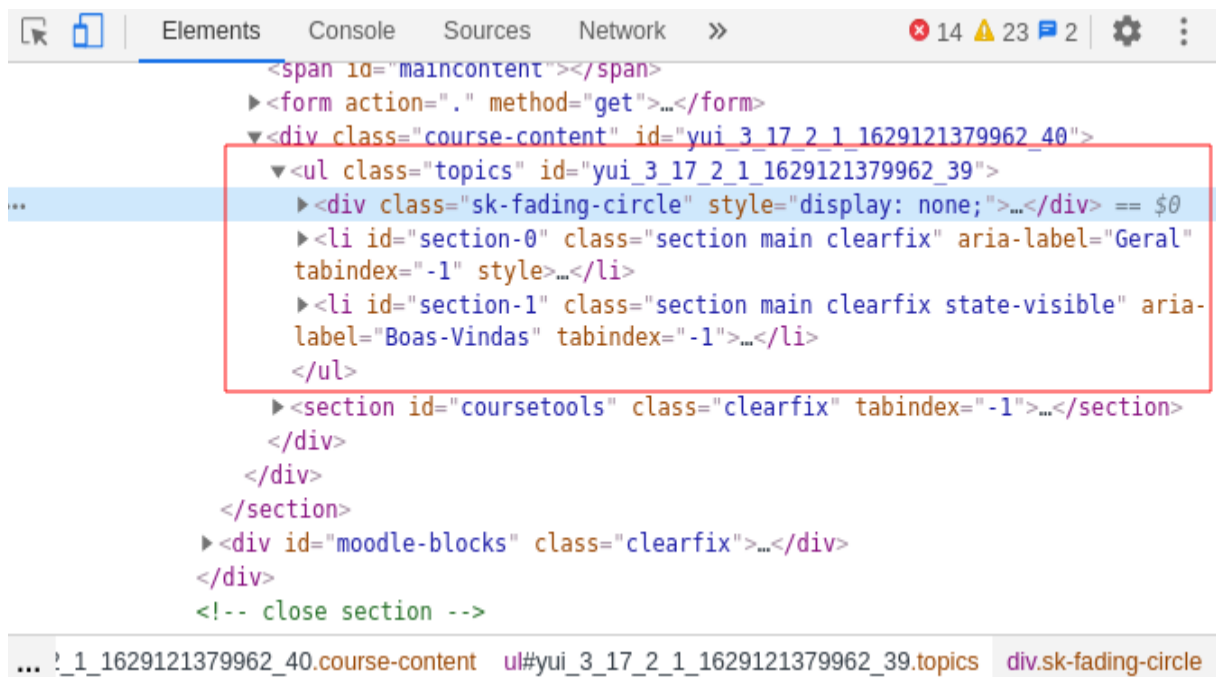
#### 5.2.12. Lists do not contain only <li> elements and script supporting elements (<script> and <template>)

Uma prática bastante comum na manipulação do HTML que infelizmente gera uma série de problemas de acessibilidade é a não utilização da semântica da linguagem (também conhecida como POSH, ou Plain Old Semantic HTML). Embora seja possível nas combinações de CSS com o JavaScript fazer com que qualquer elemento HTML se comporte da forma que quisermos, é muito provável que as propriedades built-in dos elementos – já embutidos com padrões de acessibilidade pelo teclado – não sejam consideradas na versão não-semântica do HTML. Um bom exemplo dessa prática é quando usam o elemento div para representar um botão sobrescrevendo os estilos padrão dele. Embora visualmente fique semelhante a um botão, as

propriedades padrão não serão as mesmas. Isso faz com que as tecnologias assistivas anunciem os elementos de forma errada para os usuários com alguma deficiência, além de encontrar problemas para navegar entre os elementos.

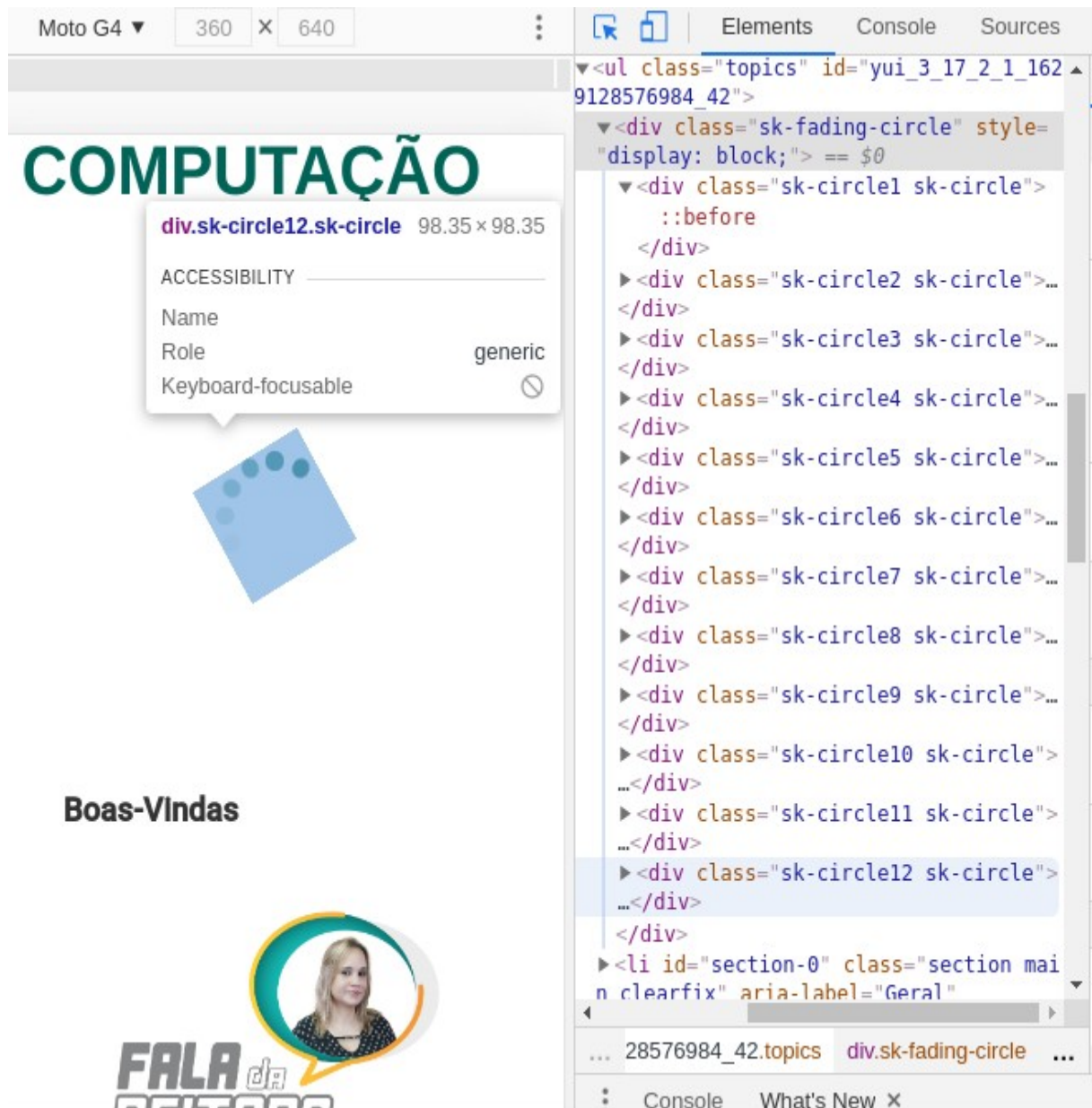
Uma lista pode ser de definição, ordenada ou desordenada. Quando a lista é ordenada (elemento ol) ou desordenada (elemento ul) todos os seus filhos devem ser um list item (elemento li). É por isso que, como afirmado na técnica H48 do documento Techniques and Failures for Web Content Accessibility Guidelines 2.0, “quando uma marcação é usada para formatar itens visualmente como uma lista, mas não representam o relacionamento da lista, os usuários (de alguma tecnologia assistiva) têm dificuldade para navegar pelas informações” (tradução nossa).

Figura 18 – Estrutura do AVA acusado pelo Lighthouse com problema de acessibilidade: um elemento ul mantendo uma div como filha além dos elementos lis apropriados



Fonte: Próprio autor. 2021.

Figura 19 – Div representando um item visual de carregamento



Fonte: Próprio autor. 2021.

Nas figuras acima podemos perceber, analisando os trechos de código denunciados pela ferramenta Lighthouse, que a div contida dentro da ul, na verdade, uma animação de carregamento para o usuário a espera de alguma requisição. Como não é semanticamente adequada a uma lista, a solução recomendada é de mover a div acima ou abaixo dela e usar CSS para reposicionar o elemento, de forma que não altere o visual atual da página mas entre em conformidade com o POSH.

## 6. CONCLUSÕES

Na métrica de desempenho e boa experiência de usuário com a acessibilidade de um site ou sistema da web, provida pelo Google Lighthouse, a classificação do AVA da Unifeso está em laranja – pontuação de 50 a 89 – que define o nível de acessibilidade do ambiente como “precisa de melhoria”. O Lighthouse também recomenda a coloração verde, que classifica o sistema com “bom nível de acessibilidade”, para uma experiência de usuário satisfatória e esperada da Web – projetada para ser utilizada por qualquer pessoa e que fornece diretrizes de acessibilidade que tornam o conteúdo na internet acessível a um maior número de pessoas com deficiência quando implementadas.

Este trabalho foi um estudo de caso com o intuito de tornar o AVA da Unifeso mais acessível, identificando suas vulnerabilidades e propondo melhorias nos pontos-chaves que mais afetam as páginas do sistema. Foi realizado um levantamento das principais tecnologias e stacks envolvidas na realização de testes automatizados de acessibilidade, além dos métodos e das principais abordagens utilizadas para testar a acessibilidade manualmente – complementando a análise com aspectos que somente os testes automatizados não conseguem identificar. A principal ferramenta escolhida para fazer os testes automatizados foi o próprio Lighthouse, que recentemente incorporou o axe-core como uma de suas bibliotecas e atualmente está integrado ao Chrome DevTools e disponível no Google Chrome a partir da versão 60.

Além das estatísticas que a pesquisa levantou, que ajudam a identificar os problemas-chaves de acessibilidade, mais frequentes e como estão distribuídos, também foram encontradas 12 propostas de soluções que impactariam consideravelmente na pontuação de acessibilidade do AVA, inserindo a plataforma na classificação verde – considerada ideal pelo Google Lighthouse.

## 7. TRABALHOS FUTUROS

**1.1.1.** Checagem de SEO com Lighthouse e impactos da acessibilidade no ranqueamento do Google: Estatísticas do antes e depois da melhoria de acessibilidade do AVA da Feso

Com as estatísticas de acessibilidade do AVA levantadas e documentadas antes das propostas de melhorias sugeridas nesse trabalho terem sido implementadas, sabendo-se que o Google Lighthouse também faz auditoria de SEO, uma nova análise automatizada após a realização dessas melhorias demonstraria a relação do nível de acessibilidade com o *ranqueamento* das páginas pelo Google.



## REFERÊNCIAS

MORIGI, V. J., et al. O reencantamento do mundo e acesso à informação: as potencialidades das Tecnologias de Informação e Comunicação (TICs) na construção e reforço da democracia Mídia, cidadania e utopia no Brasil. In: SOUSA, C. M., org. Um convite à utopia [online]. Campina Grande: EDUEPB, 2016. Um convite à utopia collection, vol. 1, pp. 303-339. ISBN: 978-85-7879-488-0. Available from: doi: 10.7476/9788578794880.0009. Also available in ePUB from: <http://books.scielo.org/id/kcdz2/epub/sousa-9788578794880.epub>.

MATELLART, A. História da sociedade da informação. São Paulo: Loyola, 2002.

ESTEVES, João Pissarra. Novos media e deliberação: sobre redes, tecnologia, informação e comunicação. Revista Media & Jornalismo, v. 18, n. 10, 2011.

ALONSO, Luiza Beth Nunes; FERNEDA, Edilson; SANTANA, Gislane Pereira. Inclusão digital e inclusão social: contribuições teóricas e metodológicas. Barbaroi, Santa Cruz do Sul, n. 32, p. 154-177, jun. 2010. Disponível em <[http://pepsic.bvsalud.org/scielo.php?script=sci\\_arttext&pid=S0104-65782010000100010&lng=pt&nrm=iso](http://pepsic.bvsalud.org/scielo.php?script=sci_arttext&pid=S0104-65782010000100010&lng=pt&nrm=iso)>. acessos em 12 jan. 2021.

MACIEL, MARIA REGINA CAZZANIGA. Portadores de deficiência: a questão da inclusão social. São Paulo Perspec., São Paulo, v. 14, n. 2, p. 51-56, June 2000. Available from <[http://www.scielo.br/scielo.php?script=sci\\_arttext&pid=S0102-88392000000200008&lng=en&nrm=iso](http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0102-88392000000200008&lng=en&nrm=iso)>. access on 12 Jan. 2021. <https://doi.org/10.1590/S0102-88392000000200008>.

TORRES, Elisabeth Fátima; MAZZONI, Alberto Angel. Conteúdos digitais multimídia: o foco na usabilidade e acessibilidade. Ci. Inf., Brasília, v. 33, n. 2, p. 152-160, Aug. 2004. Available from <[http://www.scielo.br/scielo.php?script=sci\\_arttext&pid=S0100-19652004000200016&lng=en&nrm=iso](http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0100-19652004000200016&lng=en&nrm=iso)>. access on 04 Feb. 2021. <https://doi.org/10.1590/S0100-19652004000200016>.

W3C Working Draft. Web Content Accessibility Guidelines 2.1 (WCAG 2.1). B. Caldwell, W. Chisholm, G. Vanderheiden, J. White, eds. World Wide Web Consortium (MIT, ERCIM, Keio). Disponível em: <<https://www.w3.org/TR/WCAG21>>. Acesso em: 13 jan. 2021.

BERNARDO, P. C.; KON, F. A importância dos testes automatizados. Engenharia de Software Magazine, 3. ed., 2008

ALMEIDA, Luís Fernando Magnanini de et al. Avaliação do desempenho em agilidade na gestão de projetos. Prod., São Paulo, v. 26, n. 4, p. 757-770, dez. 2016. Disponível em <[http://www.scielo.br/scielo.php?script=sci\\_arttext&pid=S0103-65132016000400757&lng=pt&nrm=iso](http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0103-65132016000400757&lng=pt&nrm=iso)>. acessos em 04 fev. 2021. Epub 10-Nov-2015. <https://doi.org/10.1590/0103-6513.116213>.

FUNDAÇÃO EDUCACIONAL SERRA DOS ÓRGÃOS - FESO  
CENTRO UNIVERSITÁRIO SERRA DOS ÓRGÃOS - UNIFESO  
CENTRO DE CIÊNCIA E TECNOLOGIA - CCT  
CURSO DE BACHARELADO EM XXXXX

**DECLARAÇÃO DE PRÓPRIA AUTORIA**

Teresópolis, XX/XX/XXXX

Eu, Nome do discente, declaro para fins de conclusão do Curso de Bacharelado em XXXX do UNIFESO, que este Trabalho de Conclusão de Curso é de minha própria autoria, estando ciente das consequências disciplinares a que estarei sujeito caso seja comprovada fraude ou má-fé.

Sem mais, subscrevo-me,

Atenciosamente,

---

Nome do discente