Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Biblioteca Python para testes de acessibilidade em páginas com a base HTML

Autor: Romeu Carvalho Antunes
Orientador: Prof. Dr. Fábio Macedo Mendes

Brasília, DF
2020

Romeu Carvalho Antunes

Biblioteca Python para testes de acessibilidade em páginas com a base HTML

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Fábio Macedo Mendes

Coorientador: Prof. Dra. Carla Silva Rocha Aguiar

Brasília, DF

2020

Romeu Carvalho Antunes
Biblioteca Python para testes de acessibilidade em páginas com a base HTML/ Romeu Carvalho Antunes. – Brasília, DF, 2020-
30 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Fábio Macedo Mendes

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2020.

1. Acessibilidade. 2. Biblioteca Python. I. Prof. Dr. Fábio Macedo Mendes. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Biblioteca Python para testes de acessibilidade em páginas com a base HTML

CDU 02:141:005.6

Romeu Carvalho Antunes

Biblioteca Python para testes de acessibilidade em páginas com a base HTML

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 22 de dezembro de 2020 – Data da aprovação do trabalho:

Prof. Dr. Fábio Macedo Mendes
Orientador

Prof. Dra. Carla Silva Rocha Aguiar
Convidado 1

Prof. Dr. Renato Coral Sampaio
Convidado 2

Brasília, DF
2020

Resumo

No contexto atual da globalização tornou-se extremamente fácil e rápido o acesso à informação, lazer e cultura por meio da internet. Entretanto essa realidade não se aplica para pessoas que possuem alguma disfunção cognitiva ou física. Este trabalho tem como objetivo criar uma solução que auxilie o desenvolvimento Web de forma que seja possível garantir o mínimo de acessibilidade necessário para o acesso dessas pessoas. Visando auxiliar no desenvolvimento Web, será desenvolvido uma biblioteca utilizando *python3* que testará acessibilidade de páginas na internet de forma a garantir uma quantidade mínima de acessibilidade, a biblioteca poderá também ser integrada em outros frameworks tais como PyTest e Django.

Palavras-chave: *HTML*. Acessibilidade. Testes de acessibilidade. Biblioteca Python.

Abstract

In the current context of globalization, access to information, leisure and culture through the Internet has become extremely easy and fast. However, this reality does not apply to people who have some cognitive or physical dysfunction. This work aims to create a solution that helps Web development so that it is possible to guarantee the minimum accessibility necessary for the access of these people. In order to assist in Web development, a library will be developed using textit python3 that will test accessibility of pages on the internet in order to guarantee a minimum amount of accessibility, the library may also be integrated into other frameworks such as PyTest and Django.

Key-words: *HTML. Accessibility. Accessibility Tests. Python Library.*

Lista de ilustrações

Lista de quadros

Lista de abreviaturas e siglas

Branch	Ramificação do projeto dentro da plataforma *GitHub*.
Commit	Mensagem significativa de alguma alteração no projeto.
CSS	*Cascading Style Sheets*.
DevOps	Junção adotada pela comunidade das palavras desenvolvimento e operação
eMAG	Modelo de Acessibilidade em Governo Eletrônico.
HTML	Linguagem de Marcação de Hipertexto.
Issue	Funcionalidade da plataforma *GitHub* para a documentação do projeto.
JavaScript	Linguagem de programação interpretada.
Web	Nome pelo qual a rede mundial de computadores internet se tornou conhecida a partir de 1991.
OMS	Organização Mundial da Saúde.
ONU	Organização das Nações Unidas
Python4	*Python* é uma linguagem de programação de alto nível, interpretada.
PO	*Product Owner*.
SM	*Scrum Master*.
Sprint	Nome de um ciclo de desenvolvimento.
SEO	mecanismos de motores de buscas.
UX	*User Experience*.
VSC	Sistema de controle de versão.
WCAG	*Web Content Accessibility Guidelines*.
W3C	*World Wide Web Consortium*.

Sumário

1 Introdução

1.1 Contexto

No mundo contemporâneo uma parcela dos usuários da internet possui algum tipo de deficiência, seja ela sensorial ou motora, severa ou branda, por isso torna-se importante a considerações de medidas e soluções que visam melhorar a acessibilidade da internet, para não limitar o acesso dessas pessoas a internet.

De acordo com a Organização Mundial da Saúde (OMS), que é uma organização especializada subordinada a Organização das Nações Unidas (ONU), "Mais de um bilhão de pessoas, cerca de 15% da população mundial, tem algum tipo de deficiência, entre 110 milhões e 190 milhões de adultos têm dificuldades significativas no funcionamento."(OMS, 2019).

Com esses dados, fica claro perceber que tratar de acessibilidade não é nenhuma novidade, de forma que ela deve ser abordada com a mesma importância tanto no mundo digital como no físico. Vale ressaltar ainda que as questões de acessibilidade vão muito além de pessoas com deficiência. Atualmente o mundo caminha para o envelhecimento da população mundial, dificuldades na visão e motoras que são causadas pela chegada da idade, se tornam mais presentes no cotidiano das pessoas. Muitos idosos buscam a comodidade dos mecanismos de acessibilidade, porém as ferramentas não são adaptadas o suficiente.

A OMS (2019) aponta que, no mundo físico pessoas com deficiência têm menos acesso aos serviços de saúde e, portanto, experimentam necessidades não atendidas. Se esse fato já é realidade no mundo físico onde a falta de acessibilidade evidencia a exclusão desse grupo de pessoas. No mundo virtual torna-se muito mais difícil a percepção de tal exclusão visto que só acontece na individualidade de cada pessoa.

1.2 Problema

Devido a quantidade de pessoas que não são beneficiadas no meio digital, os engenheiros de software devem buscar formas de garantir com que os os usuários, que necessitam de tecnologias assistivas, possam ser contemplados com as mesmas funcionalidades que os usuários comuns.

Dessa forma este projeto visa auxiliar em uma parte miníma do desenvolvimento de produtos de software, focando na parte web que utiliza Linguagem de Marcação de Hipertexto (*HTML*), de forma que seja possível realizar testes, em ambiente de desenvol-

vimento para garantir que uma quantidade mínima de acessibilidade seja alcançada por este produto.

1.3 Objetivos

1.3.1 Objetivo Geral

Desenvolver uma biblioteca, em *Python*, simples e capaz de auxiliar os desenvolvedores na realização de testes de acessibilidade em páginas Web.

1.3.2 Funcionalidades

- Desenvolver classe responsável pelo retorno dos erros de acessibilidade;
- Criar funções que auxiliem a identificação a erros comuns de acessibilidade;
- Integrar a classe de teste a uma biblioteca de teste conhecida como PyTest;
- Desenvolver classe base para testes que utilizem o *Django* como tecnologia;
- Publicar o pacote e abrir o repositório para contribuição da comunidade;

2 Referencial Teórico

2.1 Considerações iniciais

Neste capítulo serão apresentados os conceitos e termos, da engenharia de Software que ajudarão o leitor a entender de melhor forma o trabalho.

Junto aos conceitos apresentados essa secção apresentará de forma simplificada alguns pontos das linguagens, *HTML*, *CSS* e *JavaScript* que serão alvos de estudo para elaboração da solução deste trabalho.

Para a parte de acessibilidade, alvo deste trabalho, serão introduzidos os principais padrões definidos pela comunidade de desenvolvimento WEB.

2.2 Evolução de Software

No contexto da evolução de um produto de software vários campos podem ser levados em consideração. A alteração ou falta dela no campos do software, da sua documentação, das suas propriedades ou na experiência do cliente podem ser classificados como evoluções de software. (CHAPIN et al., 2001). No entanto áreas como teste e *DevOps* são maneiras de realizar uma evolução no software de maneira mais sutil para aqueles que enxergam o desenvolvimento do produto como uma caixa preta.

2.3 Testes de Software

Segundo Myers, Sandler e Badgett (2011) teste de software é o processo de executar um programa com o objetivo único de encontrar defeitos.

Apesar de muita das vezes essa área do desenvolvimento de software ser deixada de lado dentro do processo de criação do produto, ela tornar-se um grande diferencial no momento de definição de sucesso ou fracasso da solução. Com a grande gama de subcampos dentro do contexto de testes de software e diversas ferramentas para auxiliar no processo de realizá-los.

Dentro dos subcampos podemos nos aprofundar nos testes automatizados de software, que não são nada mais que um conjunto de instruções e casos de testes pré definidos que serão responsáveis por testar o seu produto, de forma que a medida que o produto for incrementado os testes automatizados garantirão a integridade do produto após o incremento.

Alinhado com o conceito de testes automatizados podemos introduzir o conceito de integração contínua, que é uma prática do desenvolvimento de software que visa a automação das *builds* do produto, junto com a execução dos testes automatizados garantindo a integridade do produto que está sendo entregue a cada incrementação.

2.4 Acessibilidade

A palavra Acessibilidade é usada para descrever facilidades ou assistências às pessoas com necessidades especiais, como em acessível para cadeirantes. Isso pode se estender para o Sistema Braille, rampas para cadeirantes, sinais sonoros em passagens de pedestres, modelos de páginas e assim por diante (MOZILLA, 2019a).

O Consórcio World Wide Web (W3C) é um consórcio internacional no qual organizações filiadas, uma equipe em tempo integral e o público trabalham juntos para desenvolver padrões para a Web. Liderado pelo inventor da web Tim Berners-Lee e o CEO Jeffrey Jaffe, o W3C tem como missão Conduzir a World Wide Web para que atinja todo seu potencial, desenvolvendo protocolos e diretrizes que garantam seu crescimento de longo prazo. (W3C, 2019)

A Web é fundamentalmente projetada para funcionar para todas as pessoas, independentemente de máquina, programas, língua, cultura, localização ou capacidade física ou mental de seus utilizadores. Quando a web atende a esses requisitos, ela se torna acessível também a pessoas com dificuldades auditivas, motoras, visuais e cognitivas, de acordo com a W3C.

Para garantir acessibilidade no contexto mundial, As Diretrizes de Acessibilidade para Conteúdo da Web (WCAG) são desenvolvidas através do processo W3C em cooperação com indivíduos e organizações em todo o mundo, com o objetivo de fornecer um único padrão compartilhado para acessibilidade de conteúdo da Web que atenda às necessidades de indivíduos, organizações e governos internacionalmente, de acordo com W3C (2019)

Dentro do contexto brasileiro existe também o eMAG no qual suas recomendações permitem que a implementação da acessibilidade digital seja conduzida de forma padronizada, de fácil implementação, coerente com as necessidades brasileiras e em conformidade com os padrões internacionais. É importante ressaltar que o eMAG trata de uma versão especializada do documento internacional WCAG (*Web Content Accessibility Guidelines*) voltada para o governo brasileiro, o eMAG também não exclui qualquer boa prática de acessibilidade do WCAG como o Ministério da Economia deixa claro, (ECONOMIA, 2019).

2.4.1 Ferramentas de acessibilidade

Para navegar na web, existem diversas extensões e ferramentas nativas para garantir a acessibilidade. Essas tecnologias assistivas utilizam a navegação por *tab* entre outros mecanismos como *feedback* de voz para garantir que, os usuários que possuam alguma limitação se sintam confortáveis na navegação (MOZILLA, 2019a).

2.5 Linguagens

Esta parte do referencial teórico tem como objetivo familiarizar o leitor com os recursos das linguagens que possuem benefícios de acessibilidade e apresentar a forma de usá-los adequadamente em seus documentos da web.

2.6 *HTML*

2.6.1 Semântica e estruturação

Dentro da linguagem *HTML* inúmeras são as formas de utilização dos componentes para se obter uma página renderizada desejada, no entanto a utilização do que chamamos de *HTML* semântico (às vezes chamado de *POSH* ou *HTML* Semântico Antigo Simples) é um boa pratica e isso quer dizer utilizar os elementos e componentes corretos para obter a página renderizada desejada.

A utilização correta dos elementos semânticos do *HTML* implicam em uma série de vantagens que são:

- Facilidade no desenvolvimento do projeto;
- Otimização funcional de maneira mais leve dentro de aparelhos móveis ocupando menos espaço de memória;
- Mecanismos de motores de buscas (SEO) dão mais importância às palavras-chave dentro dos cabeçalhos, links, etc, para que seus documentos sejam mais acessíveis aos clientes;

Devemos utilizar o *HTML* correto para o trabalho e isso não pode ser ignorado, pois é um dos principais locais em que a acessibilidade é muito prejudicada se não for tratada adequadamente, segundo Mozilla (2019f)

É trabalho do desenvolvedor atuar utilizando as melhores práticas da comunidade, fazendo isso já estará ajudando a melhorar a acessibilidade do produto.

Um exemplo de um bom *HTML* semântico:

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>TCC ROMEU</title>
  </head>
  <body>
    <main>
      <h1>Primeiro Título</h1>
      <p>Primeiro parágrafo do documento</p>
      <p>Segundo parágrafo do documento</p>
      <ol>
        <li>Primeiro Elemento da Lista</li>
        <li>Segundo Elemento da Lista</li>
        <li>Terceiro Elemento da Lista</li>
      </ol>
      <h2>Primeiro Subtitulo</h2>
      <p>Primeiro parágrafo do Subtitulo</p>
      <h2>Segundo Subtitulo</h2>
      <p>Primeiro parágrafo do segundo Subtitulo</p>
      <table border="1">
        <caption>Exemplo de Tabela</caption>
        <thead>
          <tr>
            <th scope="col">Primeira Coluna</th>
            <th scope="col">Segunda Coluna</th>
            <th scope="col">Terceira Coluna</th>
            <th scope="col">Quarta Coluna</th>
          </tr>
        </thead>
        <tbody>
        <tr>
          <th scope="row">Primeira Linha</th>
          <td>1.1</td>
          <td>1.2</td>
          <td>1.3</td>
        </tr>
      </tbody>
    </table>
    <img src="paisagem_pequena.jpg"
    alt="Avião sobrevoando um céu rosa com uma lua minguante
    a sua esquerda e embaixo o por do sol sobre duas montanhas pretas">
    <nav>
      <a href="#">Página inicial</a>
    </nav>
  </main>
</body>
</html>
```

Figura 1 – Exemplo de um bom *HTML* semântico.

Com um *HTML* escrito da forma correta, algumas vantagens para os usuários das ferramentas de acessibilidade se tornam evidentes como:

- A leitura da tela lê cada cabeçalho à medida que avança no conteúdo, notificando o que é um cabeçalho, o que é um parágrafo, etc;

- A sequência de leitura após cada elemento, permite que você siga o ritmo que for mais confortável para você;
- A alternância para o cabeçalho seguinte / anterior em muitos leitores de tela. dentro dos cabeçalhos, links, etc, para que seus documentos sejam mais acessíveis aos clientes;
- A exibição de uma lista de todos os títulos em muitos leitores de tela, permitindo usá-los como um índice prático para encontrar conteúdo específico;

Um exemplo de um *HTML* ruim:

```
<font size="7">Primeiro Titulo</font>
<br><br>
Primeiro paragrafo
<br><br>
Segundo paragrafo
<br><br>
1. Primeiro item lista
<br><br>
2. Segundo item lista
<br><br>
3. Terceiro item lista
<br><br>
<font size="5">Primeiro Subtitulo</font>
<br><br>
Primeiro paragrafo do subtitulo
<br><br>
<font size="5">Segundo subtitulo</font>
<br><br>
Primeiro paragrafo do segundo subtitulo
```

Figura 2 – Exemplo de *HTML* ruim.

Quando o *HTML* é mal estruturado, ou não totalmente escrito de maneira correta as vantagens atingidas anteriormente são perdidas de forma parcial ou até mesmo total, dificultando ao invés de facilitar com a navegação do usuário.

Uma linguagem clara do idioma também é uma boa prática segundo o Mozilla (2019f) sugerindo que:

- Não se use traços se puder evitá-lo. Em vez de escrever 5–7, escreva 5 a 7;
- Expanda abreviações - em vez de escrever Jan, escreva janeiro;

- Expanda siglas, pelo menos uma ou duas vezes. Em vez de escrever *HTML* na primeira instância, escreva *Hypertext Markup Language*;

A utilização das tecnologias mais novas do *HTML* (componentes do *HTML5*) é uma boa prática, visto que evitando soluções alternativas que podem confundir os leitores de tela, os componentes nativos tornam a leitura muito mais eficiente e prática.

O Mozilla (2019f) indica que além de seu conteúdo ter boa semântica, ele deve fazer sentido lógico em sua ordem de origem - você sempre pode colocá-lo onde deseja usar o *CSS* mais tarde, mas deve começar com a ordem de origem, assim, o que os usuários de leitores de tela receberem para eles fará sentido.

2.6.2 Controle de interface de usuário

Entende-se como controles da interface do usuário, as principais partes dos documentos da web com as quais os usuários interagem - geralmente botões, links e controles de formulário, (MOZILLA, 2019f).

Os diferentes navegadores permitem a navegação nas páginas web utilizando o teclado o que é um aspecto importante da acessibilidade dos controles da interface do usuário. Essa navegação se torna ainda mais fluida utilizando as boas práticas do *HTML* semântico como observado na Figura 1.

Apesar de não ser ideal, é possível acrescentar elementos que facilitem a navegação pelas ferramentas de leitura, apesar do trabalho extra, que será necessário para corrigir esse erro, não é considerado uma boa prática visto que existe a possibilidade de utilização do *HTML* semântico e seus componentes, sendo que essa solução de correção deverá ser utilizada em última instância.

Outro ponto que vale a pena ser ressaltado é que existe a possibilidade de se corrigir a navegação por "*tab*", caso a construção da página tenha sido realizada de forma imprópria. Além da marcação de uma "*tag*"especifica para que a solução possa funcionar será necessário também o desenvolvimento de um mecanismo em *JavaScript* para que a função nativa dos elementos em *HTML* semântico como "Enter", que funciona para ativação do componente, possam funcionar.

Os links e botões devem ser significativos. Em vez de utilizar um simples "Clique Aqui", utilize "Clique para saber mais sobre este TCC". A mesma ação vale para rótulos de formulários e entradas de texto. Utilizando essas práticas a navegação e o acerto do usuário torna-se mais efetivo.

2.6.3 Imagens

Para os conteúdos de imagens os leitores de tela não podem ajudar muito, no entanto numa tentativa de auxiliar o usuário, que possuem a visão comprometida, eles acabam lendo o nome da imagem, que está escrita no código. Por outro lado, muitas vezes esses nomes são criados de forma automática fazendo com que a solução paliativa do leitor de tela não ajude em nada.

Pelo fato da ferramenta não conseguir "ler" o que está dentro da imagem é importante lembrar que jamais se deve colocar texto em formato de imagem, pois isso impossibilita a leitura da ferramenta.

A boa prática utilizada para sanar a dificuldade, segundo o Mozilla (2019f) é adicionar a "*tag*", "*alt*", embaixo da declaração da imagem contendo uma descrição assertiva e impessoal sobre o que a imagem retrata. Isso fará com que o usuário consiga imaginar a figura através da descrição utilizada, que será lida pela ferramenta de assistência à acessibilidade. Nota-se que a descrição da imagem deverá ser utilizada apenas quando a imagem tiver importância significativa para o contexto, caso contrário deverá ser colocada uma "*tag*"vazia para indicar que a imagem não deverá ser descrita pela ferramenta.

2.6.4 Links

Os links (o <a> elemento com um "*href*"atributo), dependendo de como são usados, podem ajudar ou prejudicar a acessibilidade. Por padrão, os links são acessíveis na aparência. Eles podem melhorar a acessibilidade, permitindo que o usuário navegue rapidamente nas seções de um documento. Eles também podem prejudicar a acessibilidade se o estilo acessível for removido ou se o *JavaScript* fizer com que eles se comportem de maneira inesperada, em conformidade com o Mozilla (2019f)

É recomendado ainda que além da cor diferenciada entre os links utilize também distinguir de forma significativa do plano de fundo por meio de contraste com o requisito mínimo de 3:1 do texto para o link e de 4,5:1 com relação as cores definidas pela comunidade do Mozilla (2019b)

2.6.5 *CSS*

O *CSS* é responsável por conseguir alterar o estilo de determinado componente estrutural que está aplicado, mudando configuração de fonte, negrito, itálico, espaçamento entre linhas e letras, sombras projetadas e outros recursos de texto, segundo Mozilla (2019d).

Apesar de não possuir o mesmo peso que o *HTML*, o mecanismo para adicionar estilo *CSS*, quando usado corretamente, utilizando as melhores práticas, pode promover

melhores experiências no quesito da acessibilidade. O oposto também é válido caso seja utilizado de maneira equivocada pode prejudicar de forma significativa a acessibilidade do usuário.

O *CSS*, por ser muito versátil, pode ser utilizado para tornar qualquer componente do *HTML* parecido com outro, mas como foi explicado ao longo deste referencial teórico a ideia seria utilizar o *HTML* semântico para obter o tipo de comportamento desejado, evitando assim que confunda tanto o usuário comum quanto o usuário que faz uso de tecnologias assistivas.

De acordo com Mozilla (2019e) utilizar uma hierarquia estrutural correta, com a semântica adequada é fundamental para a acessibilidade, e caso o desenvolvedor deseje alterar algum estilo de um componente como: tamanhos de fonte, alturas de linha, espaçamento entre letras, etc. sensíveis, para tornar seu texto lógico, legível e confortável de ler, destacar de melhor forma o título, mudar o contraste da página, essas alterações podem e devem ser feitas utilizando o *CSS*, (MOZILLA, 2019c).

É recomendado seguir as convenções definidas para a utilização do *CSS* para não prejudicarmos a acessibilidade, no entanto existem algumas "*tags*"que podem danificar o fluxo pelos leitores de tela, a "*tag*" "*display:nome*"ou "*visibility: hidden*" Não devem ser usadas se desejar que o conteúdo seja lido por um leitor de tela. Mas use-as para conteúdos em que você não deseja ser lida pelos leitores de tela, em conformidade com o WebAIM (2019).

De acordo com o Mozilla (2019c) os usuários podem substituir seus estilos por seus próprios estilos personalizados e isso pode acontecer por vários motivos. Um usuário com deficiência visual parcial pode querer aumentar o texto em todos os sites visitados, ou um usuário com severa deficiência de cor pode querer colocar todos os sites em cores de alto contraste que sejam fáceis de visualizar. Qualquer que seja a necessidade, O programador deve se sentir confortável com isso e tornar seus designs flexíveis o suficiente para que essas alterações funcionem em seu design. Como exemplo, convém garantir que sua área de conteúdo principal possa lidar com texto maior (talvez ela comece a rolar para permitir que tudo seja visto) e não oculte apenas ou quebre completamente.

2.6.6 *JavaScript*

A linguagem de programação interpretada e estruturada também pode atrapalhar ou ajudar no quesito de acessibilidade conforme é utilizada, apesar de ser uma ferramenta poderosa nem sempre sua utilização será 100% acessível, no entanto é uma boa prática tentar torná-la o mais acessível possível.

Um Jogo 3D utilizando *JavaScript* por exemplo não é fácil de se tornar totalmente acessível, e não seria razoável esperar que o programador o torne 100% acessível a pes-

soas cegas, mas você pode implementar controles de teclado para que seja utilizável por usuários que não são usuários de mouse e fazer com que o esquema de cores seja suficientemente contrastante para ser usado por pessoas com deficiências de cores, segundo Mozilla (2019c).

Uma quantidade excessiva de *JavaScript* dentro da página também não é ideal pois dessa forma o desenvolvedor estará deixando de utilizar o *HTML* semântico e suas funcionalidades nativas de acessibilidade, o *JavaScript* deve ser usado de forma discreta, para garantir uma melhor acessibilidade, isso implica que deve ser utilizado para aprimorar funcionalidades não criá-las do 0, conforme o Mozilla (2019c) alguns bons exemplos de uso de *JavaScript* são:

- Fornecer validação de formulário do lado do cliente, que alerta os usuários para problemas com suas entradas de formulário rapidamente, sem ter que esperar o servidor verificar os dados. Se não estiver disponível, o formulário ainda funcionará, mas a validação poderá ser mais lenta.
- Fornecer controles personalizados para *HTML5* vídeos acessíveis a usuários que usam apenas teclado, além de um link direto para o vídeo que pode ser usado para acessá-lo se o *JavaScript* não estiver disponível (os <video>controles padrão do navegador não são acessíveis pelo teclado na maioria dos navegadores).

Eventos de mouse tratados com *JavaScript* não serão acessíveis, e deverão ser contornados, utilizando duplicação de eventos que podem ser acessado de maneiras diferentes pelas tecnologias assistivas.

3 Metodologia

3.1 Metodologias de Desenvolvimento

Para o desenvolvimento do projeto dentre as várias metodologias e práticas que existem dentro do mundo de projetos de engenharia de software, o que se fez mais pertinente para o desenvolvimento desse projeto, tendo em vista que não seria realizado por uma equipe e sim por um único desenvolvedor e o orientador de TCC foram algumas práticas ágeis dentre elas:

- Reuniões de 15 em 15 dias, agendadas de acordo com a viabilidade do orientador e do orientado;
- Programação pareada com o orientador do trabalho;
- *Review*, onde o código desenvolvido foi apresentado e avaliado pelo orientador;
- *Planning* onde em cima da avaliação da review pontos de melhoria e incrementação foram propostos;

3.1.1 Levantamento das Funcionalidas

No projeto, pelo seu objetivo de tentar encontrar de forma automática alguns dos defeitos comuns de acessibilidade, tornou-se necessário o levantamento de algumas ferramentas que realizam a identificação desses erros de forma automática ou manual.

Para a criação da lista de possíveis funcionalidades desta solução tomei como base algumas extensões do Google Chrome que auxiliam o desenvolvedor no quesito de acessibilidade, após o teste individual de cada uma das extensões realizei o levantamento de cada uma das funcionalidades, sendo as ferramentas analisadas:

- Extensão AXE - Web Accessibility Testing;
- HTML Heading Highlighter;
- WAVE Evaluation Tool;
- Accessibility Insights for Web;
- Web Developer;

Para a extensão *AXE - Web Accessibility Testing* foram levantadas as seguintes funcionalidades:

Quadro 1 – AXE - Web Accessibility Testing, fucionalidades.

Extensão AXE - Web Accessibility Testing	
Funcionalidade	**Testável com HTML**
Permite a identificação se os links possuem textos descritivos	Sim
Permite a identificação se a página possui um bom contraste nas colorações de fontes, de acordo com a WCAG 2 AA de limiares de relação de contraste	Não
Permite a identificação dos elementos de formulários que possuem ou não rótulos	Sim
Permite verificar se o atributo "role" do HTML está sendo usado corretamente de acordo com a ARIA	Sim
Permite a identificação da falta de *landmarks*	Sim
Permite a identificação de links idênticos	Sim
Permite a identificação de tags descritivas de imagens	Sim

Para a extensão *HTML Heading Highlighter* foram levantadas as seguintes funcionalidades:

Quadro 2 – HTML Heading Highlighter.

HTML Heading Highlighter	
Funcionalidade	**Testável com HTML**
Identificação dos títulos e subtítulos do HTML	Sim

Para a extensão *WAVE Evaluation Tool* foram levantadas as seguintes funcionalidades:

Quadro 3 – WAVE Evaluation Tool, funcionalidades.

Wave	
Funcionalidade	**Testável com HTML**
Permite a identificação de elementos estruturais	Sim
Permite a identificação de tags ARIA	Sim
Permite a identificação de links quebrados	Sim
Permite a identificação de links redundantes	Sim
Permite a identificação de links que redirecionam para pdf	Sim
Permite a identificação de títulos redundantes	Sim
Permite a identificação de accesskeys que podem entrar em conflito com as teclas de atalho que os usuários utilizam para navegação, em tecnologias assistivas, indicando que devem ser evitadas e implementadas com cuidado	Não
Permite a identificação de tags noscript	Não
Permite a identificação de imagens que possuem ou não textos alternativos	Sim
Permite a identificação de labels para auxiliar o leitor necessitado	Sim
Permite a identificação se a página possui um bom contraste nas colorações de fontes, de acordo com a WCAG 2 AA de limiares de relação de contraste	Não
Auxilia o desenvolvedor no desenvolvimento de tabelas para que os leitores de tela consigam ler de forma uniforme ao invés de serem valores jogados, dificultando para o usuário que utiliza tecnologias assistivas.	Não

Para a extensão *Accessibility Insights for Web* foram levantadas as seguintes funcionalidades:

Quadro 4 – Accessibility Insights for Web, funcionalidades.

Accessibility Insights for Web	
Funcionalidade	**Testável com HTML**
Permite a identificação se os botões possuem algum tipo de descrição	Sim
Permite a identificação se a página possui um bom contraste nas colorações de fontes, de acordo com a WCAG 2 AA de limiares de relação de contraste	Não
Permite a identificação da falta de elementos marcados com ARIA-labels	Sim
Permite a identificação de links que possuem textos descritivos	Sim
Permite verificar se as listas que existem na página estão na estrutura correta, para promover uma melhor acessibilidade	Sim

Para a extensão *Web Developer* não foram levantadas funcionalidades pois a interface que realiza a verificação da acessibilidade redireciona o usuário para a ferramenta *WAVE Evaluation Tool*.

Tendo em vista o levantamento das funcionalidades gerou-se um *Backlog* que será priorizado para este projeto:

Quadro 5 – Funcionalidades Levantadas.

Funcionalidades
Identificar se os links possuem textos descritivos
Identificar se a página possui um bom contraste nas colorações de fontes, de acordo com a WCAG 2 AA de limiares de relação de contraste
Identificar os títulos e subtítulos do HTML
Identificar os elementos de formulários que possuem ou não rótulos
Identificar as tags ARIA
Identificar se o atributo "role"do HTML está sendo usado corretamente de acordo com a ARIA
Identificar a falta de landmarks
Identificar os links redundantes
Identificar os links quebrados
Identificar tags descritivas de imagens
Identificar os elementos estruturais
Identificar links que redirecionam para pdf
Identificar accesskeys que podem entrar em conflito com as teclas de atalho que os usuários utilizam para navegação, em tecnologias assistivas, indicando que devem ser evitadas e implementadas com cuidado
Identificar títulos redundantes
Identificar labels para auxiliar o leitor necessitado
Auxiliar o desenvolvedor no desenvolvimento de tabelas para que os leitores de tela consigam ler de forma uniforme ao invés de serem valores jogados, dificultando para o usuário que utiliza tecnologias assistivas

A lista de funcionalidades servirá como guia do projeto onde tentarei implementar a maior quantidade possível de verificações afim de alcançar o maior valor final para os testes, entretanto a biblioteca poderá ser modificada pela comunidade pois está disponível segundo a licença GPL3. Cabe ressaltar também que algumas funcionalidades como verificação de contraste, visualmente podem ser realizadas de forma fácil entretanto a automação deste tipo de funcionalidade pode ser muito onerosa, aliado a este fato a tecnologia que foi selecionada para o desenvolvimento do projeto foi visando os testes que usam somente o HTML sem a utilização de estilo, cabe ainda ressaltar que os testes não

rodam no navegador, entretanto os testes podem ser realizados porém utilizando mais do que a tecnologia atual.

Além do fato da tecnologia escolhida para o desenvolvimento da biblioteca impossibilita alguns testes visto que eles não são rodados no navegador. Por isso serão implementadas primeiro, funcionalidades que podem ser automatizadas de forma mais rápida para que o MVP gerado ao final do semestre consiga agregar o máximo de valor.

Pelo fato de ser uma automação de uma ferramenta manual como extensões do Chrome a quantidade de erros que será captada poderá ser impactada, entretanto, o objetivo é tornar este impacto mínimo para garantir que a ferramenta de automação tenha algum valor para o desenvolvedor que deseja utiliza-lá , das funcionalidades levantadas apenas a verificação de títulos e subtítulos do HTML e a verificação de tags descritivas de imagens foram implementadas.

3.2 Problema

A abordagem para a solução dos problemas de acessibilidade encontrados no desenvolvimento web é baseada no princípio de dividir para conquistar, essa abordagem garante também a modularização do desenvolvimento para que este ocorra de forma constante durante todo o semestre letivo.

Na questão de divisão dos problemas levantados que devem ser solucionados com a biblioteca a ser implementada temos:

- Análise estática do HTML, através do validador da W3C;
- Criação da classe de Exceções responsável por devolver ao desenvolvedor os erros cometidos no desenvolvimento web;
- Validador de acessibilidade;
- Distribuição da biblioteca;

Estes 3 itens visam contemplar uma biblioteca capaz de receber de forma dinâmica ou estática, código *HTML* e dessa forma realizar algumas verificações comuns relativas a acessibilidade.

A classe de exceções será projetada de forma que o desenvolvedor que utilize a biblioteca consiga obter nos testes automáticos uma lista de objetos que irá conter: O erro encontrado, a *tag* que está errada, linha e coluna do erro;

O validador de acessibilidade será a parte fundamental do código visto que ele receberá o HTML como uma *string* e realizará o parse das *tag* com seus respectivos

atributos e valores, uma vez que o código tenha sido quebrado em *tags*, executará uma validação nas *tags* que ocorrem erros de acessibilidade de forma mais comum, um exemplo é a utilização da *tag* de *<img>* sem a declaração do atributo "*alt*"este que é o responsável por dizer ao leitor de tela o que aquela imagem representa.

Por fim o pacote será distribuído de forma que sua importação se torne rápida e fácil para os desenvolvedores e que ele seja facilmente acoplado a outros frameworks que utilizam os pacotes do Python.

3.3 Cronograma

Para o cronograma de desenvolvimento da solução, o semestre letivo de 2020/1 será dividido em *sprints* quinzenais, em que o objetivo será realizar uma funcionalidade por quinzena. A medida em que o projeto for avançando e dificuldades forem surgindo cabe ao aluno solucioná-las, caso estejam ao seu alcance e, caso não estejam, buscar da melhor forma a ajuda para transpor as dificuldades:

Primeiro mês	**Segundo mês**	**Terceiro mês**	**Quarto mês**
Ajustes e revisões de texto	Implementação do pacote alvo de testes em django	Implementação da função busca h1	Ajustes e cache
Alinhamento da solução com o orientador	Implementação inicial do pacote de acessibilidade	Implementação da função hierarquia de título e de imagem	Correções textuais
Início das implementações do pacote	Implementação do parse HTML	Implementação da validação estática	Defesa do TCC

Quadro 6 – Cronograma do projeto.

4 Conclusão

Neste capítulo será descrita a forma como os problemas de implementação foram abordados, como foram criadas as soluções, as dependências necessárias para o projeto, e as ferramentas utilizadas no desenvolvimento.

4.1 Dependências

A biblioteca que foi implementada pode ser instalada por meio do pacote Pip (2020) que nada mais é que o gerenciador de pacotes para o *Python*, além disso a próprio biblioteca possui ainda algumas dependências obrigatórias que são:

- Beautifulsoup4 Soup (2020), responsável por realizar o parse das *tags HTML*, da página ou fragmento que será analisado ;
- PYW3C Byasov (2020), responsável por realizar a validação estática do código *HTML*, utilizando a própria W3C para validação online;
- Joblib developers (2020), responsável por realizar o cache da validação estática garantindo assim que não serão enviadas múltiplas requisições ao site da w3c;

A biblioteca além das dependências obrigatórias possui ainda uma dependência opcional que seria o pacote do Django na versão 2.2 ou superior, essa dependência é opcional pois caso o desenvolvedor web faça o uso deste *framework* ele poderá importar a classe do pacote criada especialmente para este *framework* de forma a facilitar os testes de acessibilidades na plataforma.

4.2 Funcionalidades

Nesta secção explicarei como os problemas levantados na secção de metodologia foram sanados, começaremos com a análise estática do *HTML*. O pacote que foi selecionado para auxiliar no desenvolvimento dessa funcionalidade é o pacote py_w3c este pacote funciona realizando requisições para o próprio validador disponibilizado de forma online e gratuita pela própria W3C, neste validador o usuário pode indicar uma página web, um fragmento de código ou um arquivo a ser validado.

O validador verifica a sintaxe de documentos da Web, escritos em formatos como (X) *HTML*. Ele compara seu documento *HTML* com a sintaxe definida de *HTML* e relata quaisquer discrepâncias . A lista de erros que são verificados pelo validador se encontra

disponível na documentação de erros do validadador da w3C no site:
https://validator.w3.org/docs/errors.html (W3C, 2020).

Como tentativa de aumentar a eficiência do projeto e diminuir possíveis gargalos, tendo em vista que o validador estático da w3c utiliza de recursos onlines fazendo requisições para o validador e afim de evitar requisições duplicadas foi utilizado o pacote do *joblib* que tem como objetivo dentro da solução armazenar a resposta da w3c, de forma que quando outra requisição idêntica aparente ser disparada ao invés de ir buscar o retorno da requisição no validador será buscada em cache, garantindo o tempo quase instantâneo de resposta e blindando o validador de requisições duplicadas.

Para o problema levantado da verificação da acessibilidade a biblioteca conseguiu provar sua eficiência utilizando em conjunto a dependência do *Beautifulsoup4* que é responsável por extrair os dados das paginas ou fragmentos *HTML*, uma vez que estejam extraídos o *Beautifulsoup4* consegue retornar uma arvore das *tags* presentes de forma que seja possível acessar valores, atributos e parentes da mesma, com esse comportamento foi possível desenvolver as seguintes funcionalidades para a biblioteca

- Verificação de *tag* de imagem: Nesta verificação é buscado o atributo *alt* que é responsável por descrever a imagem, tornando assim identificável para o leitor de tela;
- Verificação da existência de um H1 na página *HTML*: Nesta verificação a biblioteca busca encontrar na árvore de *tags* a *tag* h1 que é responsável por indicar o título principal da página e é uma boa prática sempre existir;
- Verificação da hierarquia de títulos: Nesta verificação a biblioteca garante que uma vez que um título n+1 tenha aparecido, obrigatoriamente o seu anterior já deve ter ocorrido

Além disso a biblioteca permite ainda que o usuário escolha não realizar algumas das validações, passando na chamada da função de validação de acessibilidade uma lista daquelas verificações que deverão ser excluídas, caso não seja passada a lista ou seja vazia a biblioteca executará todas as funções

4.3 Disponibilização do pacote

Para o problema levantado relacionado a distribuição da biblioteca esta funcionará como um módulo que será empacotado e disponibilizado no Pypi. O PyPI é um repositório de software para a linguagem de programação Python que auxilia os desenvolvedores a encontrar e instalar software desenvolvido e compartilhado pela comunidade Python (INDEX, 2020).

Logo qualquer usuário de frameworks que possuem a possibilidade de importação de pacotes *Python* poderá utilizá-la. O objetivo com essa facilidade de importação é democratizar o acesso a ferramenta e ajudar os desenvolvedores da comunidade na produção de software cada vez mais acessíveis.

O pacote se encontra disponível no Pypi pelo nome de automated_accessibility_testing-1.0.1 e está disponível para colaboração e evolução no github no repositório automated_accessibility_testing

4.4 Trabalhos Futuros

Após a implementação da biblioteca fica claro perceber quão precário é o cenário de ferramentas automatizadas para testes de acessibilidade, e quão extenso é o caminho que podemos percorrer para melhorar a biblioteca e os testes existentes, inúmeras verificações podem ser contempladas dentro da própria biblioteca sem a necessidade de incorporarmos outra ferramenta para a verificação.

Além dos testes ainda é possível realizar como um trabalho futuro a integração da biblioteca com o pipeline de *DevOps* que é um esforço colaborativo e multidisciplinar dentro de uma organização para automatizar a entrega contínua de novas versões de software, garantindo ao mesmo tempo correção e confiabilidade (LEITE et al., 2019). De forma que o pipeline de desenvolvimento do software contenha este fluxo:

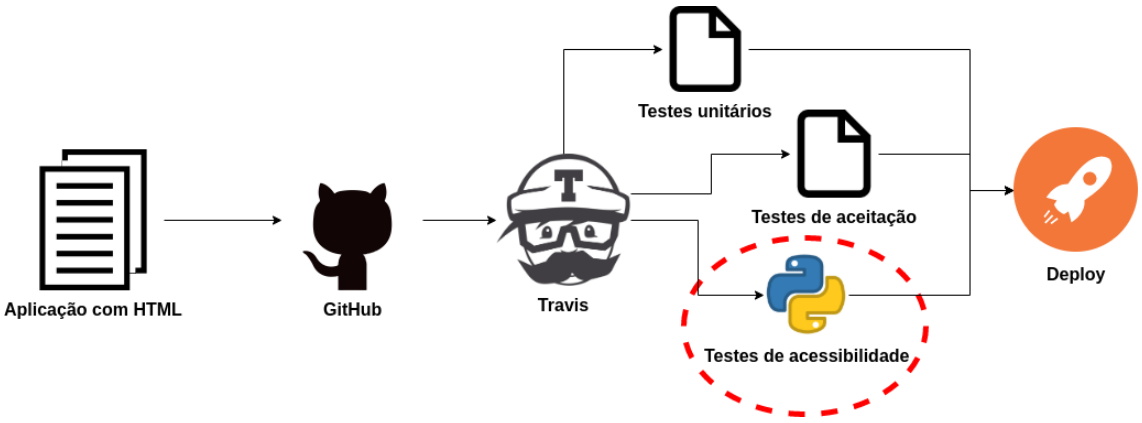

Figura 3 – *Pipeline DevOps* com solução acoplada. (Fonte(Antunes (2019))

Referências

ANTUNES, R. C. *Códigos HTML renderizados.* 2019. Disponível em: <https://github.com/RomeuCarvalhoAntunes/TCC1/tree/master/imagens>. Acesso em: Novembro, 12. 2019. Citado 2 vezes nas páginas 6 e 28.

BYASOV, K. *pyw3c.* 2020. Disponível em: <https://pypi.org/project/py_w3c/>. Acesso em: Dezembro, 03. 2020. Citado na página 26.

CHAPIN, N. et al. Types of software evolution and software maintenance. *Journal of software maintenance and evolution: Research and Practice*, Wiley Online Library, v. 13, n. 1, p. 3–30, 2001. Citado na página 12.

DEVELOPERS, J. *Joblib.* 2020. Disponível em: <https://joblib.readthedocs.io/en/latest/>. Acesso em: Dezembro, 03. 2020. Citado na página 26.

ECONOMIA, G. D. M. da. *eMAG - Modelo de Acessibilidade em Governo Eletrônico.* 2019. Disponível em: <http://emag.governoeletronico.gov.br/>. Acesso em: Outubro, 25. 2019. Citado na página 13.

INDEX, P. P. *Pypi.* 2020. Disponível em: <https://www.python.org/psf/>. Acesso em: Dezembro, 03. 2020. Citado na página 27.

LEITE, L. et al. A survey of devops concepts and challenges. *arXiv preprint arXiv:1909.05409*, 2019. Citado na página 28.

MOZILLA. *Acessibilidade.* 2019. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/Acessibilidade>. Acesso em: Outubro, 25. 2019. Citado 2 vezes nas páginas 13 e 14.

MOZILLA. *Color contrast.* 2019. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/Accessibility/Understanding_WCAG/Perceivable/Color_contrast>. Acesso em: Outubro, 25. 2019. Citado na página 18.

MOZILLA. *CSS and JavaScript accessibility best practices.* 2019. Disponível em: <https://developer.mozilla.org/en-US/docs/Learn/Accessibility/CSS_and_JavaScript>. Acesso em: Outubro, 25. 2019. Citado 2 vezes nas páginas 19 e 20.

MOZILLA. *Estilo de texto.* 2019. Disponível em: <https://developer.mozilla.org/en-US/docs/Learn/CSS/Styling_text>. Acesso em: Novembro, 12. 2019. Citado na página 18.

MOZILLA. *Fundamentos de texto HTML.* 2019. Disponível em: <https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/HTML_text_fundamentals>. Acesso em: Novembro, 12. 2019. Citado na página 19.

MOZILLA. *HTML: A good basis for accessibility.* 2019. Disponível em: <https://developer.mozilla.org/en-US/docs/Learn/Accessibility/HTML>. Acesso em: Outubro, 25. 2019. Citado 4 vezes nas páginas 14, 16, 17 e 18.

MYERS, G. J.; SANDLER, C.; BADGETT, T. *The art of software testing.* [S.l.]: John Wiley & Sons, 2011. Citado na página 12.

OMS. *Incapacidade e saúde*. 2019. Disponível em: <https://www.who.int/en/news-room/fact-sheets/detail/disability-and-health>. Acesso em: Novembro, 12. 2019. Citado na página 10.

PIP. *Pip*. 2020. Disponível em: <https://pip.pypa.io/en/stable/>. Acesso em: Dezembro, 03. 2020. Citado na página 26.

SOUP, B. *Beautiful Soup*. 2020. Disponível em: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>. Acesso em: Dezembro, 03. 2020. Citado na página 26.

W3C. *Visão geral das diretrizes de acessibilidade de conteúdo da Web (WCAG)*. 2019. Disponível em: <https://www.w3.org/WAI/standards-guidelines/wcag/>. Acesso em: Outubro, 25. 2019. Citado na página 13.

W3C. *W3C Validator*. 2020. Disponível em: <https://validator.w3.org/>. Acesso em: Novembro, 12. 2020. Citado na página 27.

WEBAIM. *CSS em ação - Conteúdo invisível apenas para usuários de leitores de tela*. 2019. Disponível em: <https://webaim.org/techniques/css/invisiblecontent/>. Acesso em: Novembro, 12. 2019. Citado na página 19.