WAE - *Web Accessibility Engine*: Uma Ferramenta de Transpilação Acessível e Estruturada para Programadores Cegos ou de Baixa Visão

Douglas Sousa Jorge
Instituto de Informática
UFG
Goiânia, Brasil
douglassousa@discente.ufg.br

Maurício Lima
Instituto de Informática
UFG
Goiânia, Brasil
k20x@outlook.com

Elisângela Silva Dias
Instituto de Informática
UFG
Goiânia, Brasil
elisangelasd@ufg.br

Augusto César Falcão
Instituto de Informática
UFG
Goiânia, Brasil
augustocesar.ffalcao@gmail.com

***Abstract*—This study focuses on the development of the WAE - Web Accessibility Engine, a tool designed to facilitate the prototyping and development of static web pages by blind or visually impaired programmers. The research included a literature review, which led to the selection of the accessible coding paradigm "Grid Coding" and the YAML language as the foundations for the project. To evaluate the accessibility of the pages generated by WAE, the Accessibility Evaluator and Simulator for Websites (ASES) was used. The results showed that most components created by WAE met high accessibility standards, highlighting the tool's potential to contribute significantly to inclusive web development.**

***Keywords*—Accessibility; visually impaired; ASES.**

***Resumo*—Este estudo foca no desenvolvimento do *WAE - Web Accessibility Engine*, uma ferramenta projetada para facilitar a prototipação e o desenvolvimento de páginas *web* estáticas por programadores cegos ou com baixa visão. A pesquisa incluiu uma revisão da literatura, que resultou na escolha do paradigma de codificação acessível "*Grid Coding*" e da linguagem YAML como bases para o projeto. Para avaliar a acessibilidade das páginas geradas pelo WAE, utilizou-se o *Avaliador e Simulador de Acessibilidade em Sítios* (ASES). Os resultados mostraram que a maioria dos componentes criados pelo WAE atendeu a altos padrões de acessibilidade, evidenciando o potencial da ferramenta para contribuir de forma significativa para o desenvolvimento *web* inclusivo.**

***Palavras-chave*—Acessibilidade; Deficiência Visual; ASES.**

I. Introdução

As Tecnologias Assistivas (TAs) desempenham um papel importante na promoção da inclusão e da acessibilidade, especialmente para pessoas com deficiência (PcDs) [1]. Essas tecnologias são fundamentais para garantir que todos, independentemente de suas capacidades, possam participar plenamente da sociedade digital [2]. No entanto, apesar dos avanços significativos, ainda existem áreas em que a acessibilidade precisa ser aprimorada, particularmente no campo da programação e do desenvolvimento *web* [3].

Um dos principais desafios enfrentados no cenário tecnológico é a inclusão de desenvolvedores cegos. Esses indivíduos frequentemente se deparam com ambientes de programação que não são totalmente acessíveis, o que cria barreiras significativas para suas carreiras profissionais e seu desenvolvimento pessoal [4]. A escassez de ferramentas de programação projetadas com foco na acessibilidade limita não apenas as oportunidades para programadores cegos, mas também priva o campo da tecnologia de talentos valiosos e de perspectivas únicas [5].

Além disso, o desenvolvimento *web*, área de importância crescente dentro da programação [6], representa uma oportunidade significativa para a promoção da inclusão. À medida que mais aspectos da vida cotidiana e do trabalho migram para o ambiente online, torna-se imperativo que o desenvolvimento *web* seja acessível a todos, incluindo as PcDs. Por isso, este estudo apresenta o desenvolvimento de uma ferramenta denominada *WAE - Web Accessibility Engine*, cuja finalidade é a transpilação de código de uma linguagem de alto nível para *HyperText Markup Language (HTML)*, com foco especial na acessibilidade. Essa ferramenta foi concebida para facilitar a prototipação e o desenvolvimento de páginas *web* por indivíduos cegos ou com baixa visão. O WAE incorporará o paradigma *Grid-Coding*, conforme descrito por [7], e se adaptará à estrutura da linguagem YAML [8], com o objetivo de criar uma interface de desenvolvimento *web* que seja tanto acessível quanto intuitiva para os usuários.

Este estudo tem como objetivo geral estabelecer um fluxo completo de prototipação de páginas *web* estáticas que seja plenamente acessível a programadores cegos ou com baixa

visão, unificando as etapas de edição, transpilação e verificação de acessibilidade em uma solução única. Para atingir esse propósito, pretende-se desenvolver um transpilador capaz de converter descrições em linguagem de alto nível em código *HTML* semântico acompanhado de *CSS* estruturado, reduzindo o esforço cognitivo na criação de interfaces. Essa ferramenta será integrada ao *Grid Editor*, uma *IDE* baseada no paradigma *Grid-Coding*, a fim de oferecer uma experiência de escrita contínua e acessível, especialmente no tocante à estilização e à organização da página. Por fim, as páginas geradas serão avaliadas com o Avaliador e Simulador de Acessibilidade em Sítios (ASES) [9], assegurando conformidade com as diretrizes *WCAG* e garantindo que os protótipos possam ser utilizados de maneira eficaz por pessoas com deficiência visual.

O restante deste estudo está organizado da seguinte forma. A Seção 2 revisa os trabalhos correlatos que fundamentam a pesquisa. A Seção 3 apresenta a fundamentação teórica, detalhando conceitos de acessibilidade na *web*, o paradigma *grid-coding*, a transpilação de linguagens de alto nível para HTML, o avaliador ASES, além de protótipos de aplicações e páginas estáticas. A Seção 4 descreve a metodologia adotada, contemplando a pesquisa bibliográfica, a pesquisa exploratória e a análise de acessibilidade das páginas geradas pelo processo de transpilação. A Seção 5 expõe o desenvolvimento do transpilador. A Seção 6 trata da definição e implantação da arquitetura, especificando o transpilador, o componente de configuração, os componentes estruturais, as diretrizes WCAG incorporadas e o desenvolvimento do *Grid Editor*. A Seção 7 apresenta e discute os resultados obtidos. A Seção 8 aborda os conceitos e considerações éticas do estudo, inclusão e justiça social, autonomia e agência do usuário, privacidade e segurança de dados, transparência, responsabilidade e sustentabilidade, ética em pesquisas com populações vulneráveis e mitigação de limitações e vieses. Por fim, a Seção 9 oferece as conclusões e sugere direções para trabalhos futuros.

II. Trabalhos Correlatos

Ehtesham et al. [7] desenvolveram o *Grid Editor*, um ambiente que organiza o código em uma grade lógica e oferece *feedback* auditivo e visual sincronizado. Em um experimento com 12 programadores cegos ou com baixa visão, o *Grid Editor* reduziu significativamente o tempo de edição e o número de erros em relação a um editor textual convencional, demonstrando o impacto positivo de pistas multimodais na navegação estrutural do código.

Baker et al. [5] complementam essa perspectiva ao mapear as barreiras encontradas por estudantes cegos em cursos de ciência da computação. Entre os obstáculos relatados estão a falta de material acessível, a escassez de exemplos adaptados e a interação limitada com docentes. Esses fatores geram isolamento e diminuem a motivação, indicando que a disponibilidade de ferramentas inclusivas, embora necessária, não é suficiente sem o suporte pedagógico adequado.

Schanzer et al. [10] avançam no problema ao propor um *toolkit* que transforma navegadores em ambientes de programação acessíveis. O sistema fornece descrições auditivas detalhadas da estrutura do programa e permite navegação por atalhos de teclado, resultando em melhor orientação espacial no código sem aumento relevante no tempo total de execução das tarefas.

Embora relevantes, os três trabalhos atuam em pontos isolados,*editor*, contexto educacional e *framework* de navegador, sem oferecer um fluxo produtivo integrado. O presente estudo aborda essa lacuna ao combinar a interface do *Grid Editor* às propriedades de legibilidade do YAML [8] em um transpilador que gera HTML acessível. Essa integração cria uma cadeia completa de prototipagem: escrita em alto nível, edição guiada por multimodalidade e saída diretamente renderizável na *web*. Assim, reduzimos a carga cognitiva identificada por Baker et al. [5] e ampliamos o alcance prático das soluções de Ehtesham et al. [7] e Schanzer et al. [10], fornecendo não apenas um *editor* ou *toolkit*, mas um ecossistema coeso que torna a programação *web* mais acessível a pessoas com deficiência visual.

III. Fundamentação Teórica

Nesta seção, serão explorados os conceitos de Acessibilidade na *web*, o Paradigma *Grid-Coding*, a importância da transpilação de linguagens de alto nível para HTML, o Avaliador e Simulador de Acessibilidade em Sítios (ASES), os protótipos de aplicações e as páginas *web* estáticas, que são fundamentais para a compreensão deste estudo.

A. Acessibilidade na web

A acessibilidade na *web* é um princípio que visa garantir que *websites* e aplicações *web* sejam utilizáveis por todas as pessoas, independentemente de suas habilidades ou deficiências [11]. Este conceito é vital para o desenvolvimento de soluções tecnológicas inclusivas, particularmente para pessoas com deficiências visuais, auditivas, motoras ou cognitivas. Uma das principais referências neste contexto são as Diretrizes de Acessibilidade para Conteúdo *web* (*Web Content Accessibility Guidelines* - WCAG) [12], que fornecem recomendações para tornar o conteúdo da *web* mais acessível. As WCAG abrangem uma ampla gama de recomendações, incluindo a disponibilização de alternativas textuais para conteúdo não textual, a facilidade de navegação e a compatibilidade com assistentes de tecnologia, como leitores de tela. A conformidade com essas diretrizes não é apenas uma prática de inclusão, mas também pode melhorar a experiência geral do usuário

e otimizar o SEO (*Search Engine Optimization*) dos *websites* [13].

B. Paradigma Grid-Coding

O paradigma *Grid-Coding* [7] representa uma abordagem na programação especialmente projetada para atender às necessidades de programadores cegos ou com baixa visão. Esse paradigma estrutura o código em uma forma de grade, na qual cada elemento é posicionado em células específicas, facilitando a navegação e a compreensão do código por meio de leitores de tela e outras tecnologias assistivas.

O *Grid-Coding* permite que programadores com deficiência visual compreendam melhor a estrutura e o fluxo do código, superando muitos dos desafios associados à programação convencional. Além disso, essa abordagem pode ser integrada ao desenvolvimento *web*, tornando a escrita de código mais acessível e intuitiva, e possibilitando que desenvolvedores com deficiência visual participem mais ativamente neste processo de criação.

C. Transpilação de Linguagens de Alto Nível para HTML

A transpilação é um processo crítico no desenvolvimento de *software*, envolvendo a conversão de código-fonte escrito em uma linguagem para outra [14]. No contexto do desenvolvimento *web*, essa prática permite que os desenvolvedores utilizem linguagens mais abstratas e expressivas, que são posteriormente convertidas em HTML, a linguagem de marcação de texto padrão para a criação de páginas *web*.

Essa abordagem oferece diversos benefícios, como a simplificação dos processos de prototipação e desenvolvimento, permitindo que os programadores se concentrem na lógica e no design, ao invés de nos detalhes específicos da linguagem. Para programadores com deficiência visual, a transpilação de linguagens de alto nível para HTML é especialmente vantajosa, pois possibilita a utilização de linguagens mais compatíveis com tecnologias assistivas e paradigmas de programação acessíveis, como o *Grid-Coding*, facilitando sua participação no desenvolvimento *web*.

D. Avaliador e Simulador de Acessibilidade em Sítios (ASES)

Avaliadores de acessibilidade são ferramentas fundamentais no desenvolvimento de aplicações *web*, utilizadas para garantir que o conteúdo online seja acessível a todos os usuários, incluindo aqueles com deficiências [15]. Esses avaliadores analisam a conformidade de um site com as *Web Content Accessibility Guidelines* (WCAG, Diretrizes de Acessibilidade para Conteúdo *Web*) e outras normas relevantes, identificando e sugerindo correções para barreiras de acessibilidade [16].

O Avaliador e Simulador de Acessibilidade em Sítios (ASES) [9] é um exemplo dessa categoria de ferramentas, desenvolvido pelo governo brasileiro. O ASES é projetado especificamente para avaliar sites e aplicações *web* em relação às WCAG e à legislação brasileira sobre acessibilidade digital. Ele realiza uma análise detalhada das páginas *web*, fornecendo relatórios que destacam problemas de acessibilidade e oferecem recomendações para melhorias. Essa ferramenta desempenha um papel importante para tornar o conteúdo da *web* mais inclusivo e acessível a um espectro mais amplo de usuários.

E. Protótipos de Aplicações

Protótipos de páginas *web* são ferramentas essenciais no processo de design e desenvolvimento de aplicações [17]. Eles funcionam como modelos preliminares, permitindo aos desenvolvedores experimentar diferentes *layouts*, elementos de design e interações de usuário.

Essa fase de prototipação possui grande importância para testar ideias, identificar problemas de usabilidade e garantir que o design final atenda às necessidades dos usuários. Os protótipos são particularmente valiosos em projetos focados na acessibilidade, pois permitem a validação de elementos de design e interatividade que sejam facilmente navegáveis e compreensíveis por todos os usuários, incluindo aqueles com deficiências visuais [18].

F. Páginas web Estáticas

Páginas *web* estáticas [19] são caracterizadas por seu conteúdo fixo e imutável, servindo como a forma final de apresentação de informações na *web*. Ao contrário das páginas dinâmicas, que são geradas em tempo real, as páginas estáticas são armazenadas no servidor exatamente como serão apresentadas ao usuário. Essa natureza estática as torna mais rápidas para carregar e menos complexas em termos de desenvolvimento e manutenção. Páginas estáticas são ideais para conteúdos que não necessitam de atualizações frequentes, sendo uma escolha eficiente para sites focados na entrega direta de informações e na manutenção da acessibilidade como prioridade.

IV. Metodologia

Nesta seção, será detalhada a abordagem utilizada para o desenvolvimento do WAE. Serão exploradas as escolhas técnicas, desde a adoção do paradigma *Grid-Coding* [7] e da linguagem YAML [8], até a definição do escopo dos componentes *web* implementados e do desenvolvimento do transpilador, ilustrando como essas decisões se alinham ao objetivo de atender às necessidades de programadores cegos ou com baixa visão. Além disso, será discutido o processo de integração entre o ambiente de desenvolvimento e o transpilador, bem como as estratégias para avaliação da acessibilidade das páginas HTML geradas, enfatizando a importância de proporcionar uma experiência de navegação inclusiva e acessível.

A. Pesquisa Bibliográfica

A pesquisa bibliográfica constitui uma parte fundamental deste estudo, fornecendo a base teórica e conceitual necessária para o desenvolvimento do projeto. Esta fase envolveu um extenso levantamento e análise de literatura relacionada, incluindo artigos acadêmicos, publicações sobre tecnologias assistivas, estudos sobre acessibilidade *web* e documentação técnica sobre linguagens de programação, como YAML, além de ferramentas de desenvolvimento *web*. O objetivo foi compreender as práticas atuais, identificar lacunas na literatura existente e determinar as melhores práticas e abordagens para a criação de ferramentas de desenvolvimento *web* acessíveis.

A pesquisa bibliográfica também desempenhou um papel importante na definição do escopo e da direção do projeto. Ao revisar o estado da arte em acessibilidade *web* e em ferramentas de desenvolvimento, foi possível moldar a proposta do WAE. Esta etapa assegurou que o projeto estivesse alinhado com as necessidades atuais de programadores com deficiência visual e que contribuísse de forma significativa para o campo do desenvolvimento *web* acessível.

B. Pesquisa Exploratória

A pesquisa exploratória realizada neste estudo teve como objetivo identificar as necessidades específicas e os desafios enfrentados por programadores cegos ou com baixa visão no desenvolvimento *web*. Este aspecto da metodologia envolveu a análise de estudos de caso e a exploração de cenários de uso prático. Por meio desta pesquisa, foi possível obter *insights* valiosos sobre as limitações das ferramentas de desenvolvimento existentes e as demandas específicas para a criação de um ambiente de desenvolvimento mais acessível e intuitivo.

Além disso, a pesquisa exploratória contribuiu para moldar o design e as funcionalidades do WAE, garantindo que a ferramenta atendesse às necessidades reais dos usuários. A avaliação de diferentes abordagens de codificação acessível, como o *Grid-Coding*, e a investigação de como as linguagens de programação podem ser otimizadas para a acessibilidade foram de grande valia para o desenvolvimento de um transpilador eficaz e prático para programadores com deficiência visual.

Nesta etapa, destacou-se a realização de um teste de usabilidade [20] com um usuário cego, utilizando a metodologia *Think Aloud* [21]. Este teste envolveu a plataforma de submissão e avaliação de códigos "*sharif*" [22], utilizada pela Universidade XXX. Os resultados obtidos foram fundamentais para orientar o desenvolvimento da pesquisa, especialmente ao evidenciar as principais dificuldades e necessidades de um usuário cego ao acessar uma plataforma essencial para sua vida acadêmica. Consequentemente, os componentes transpilados pelo WAE foram ajustados para atender a essas necessidades, garantindo maior acessibilidade e melhorando a experiência de uso das páginas geradas.

C. Análise de Acessibilidade das Páginas HTML Geradas Através da Transpilação

Nesta seção, focaremos na análise da acessibilidade das páginas HTML geradas pelo transpilador deste projeto. Detalharemos a importância dessa avaliação, a metodologia adotada e o processo técnico envolvido, incluindo a adesão às diretrizes WCAG [12] e a utilização do ASES [9], para garantir altos padrões de acessibilidade nas páginas criadas.

1) Importância da Avaliação de Acessibilidade: A avaliação da acessibilidade das páginas HTML geradas é um componente crítico da metodologia deste estudo. Sua importância reside na necessidade de garantir que as páginas *web* desenvolvidas sejam não apenas funcionalmente adequadas, mas também acessíveis a todos os usuários, incluindo aqueles com deficiência visual. A adoção desta metodologia é fundamental, pois visa assegurar que desenvolvedores cegos ou com baixa visão possam criar protótipos e páginas *web* que sejam igualmente acessíveis para usuários com necessidades semelhantes.

2) Avaliação de Acessibilidade: O processo de avaliação inicia-se com a etapa da transpilação, onde o código é convertido em HTML pelo transpilador. Este passo é fundamental, pois é nesse momento que as diretrizes da *Web Content Accessibility Guidelines* (WCAG) [23] são incorporadas ao código-fonte. A aderência a essas diretrizes durante a transpilação assegura que as páginas HTML geradas estejam alinhadas aos mais altos padrões de acessibilidade desde o início. Esse cuidado inicial é vital para garantir que os elementos fundamentais da acessibilidade *web* estejam intrinsecamente presentes no design e na estrutura das páginas.

Após a transpilação, as páginas HTML são submetidas a uma análise minuciosa utilizando o Avaliador e Simulador de Acessibilidade em Sítios (ASES) [9]. Essa avaliação detalhada é essencial para identificar e corrigir quaisquer lacunas remanescentes em termos de acessibilidade. O ASES examina diversos aspectos da página, desde a estrutura lógica até a compatibilidade com tecnologias assistivas, garantindo que as páginas não apenas atendam às normas técnicas, mas também garantam uma experiência de navegação universalmente acessível, atendendo às necessidades de cada usuário, incluindo aqueles com deficiência visual.

No processo de avaliação realizado pelo ASES, cada página *web* é verificada quanto à sua aderência às diretrizes de acessibilidade [24]. Elementos cruciais, como alternativas textuais para conteúdo não textual, estrutura lógica e facilidade de navegação, são examinados. Essa análise é essencial para assegurar que as páginas sejam compreensíveis e navegáveis por usuários com deficiência visual, utilizando tecnologias

assistivas como leitores de tela. A avaliação também se estende à compatibilidade com uma variedade de ferramentas de acessibilidade, visando proporcionar uma experiência de usuário completa e inclusiva [9].

Na metodologia estabelecida para avaliar a acessibilidade dos componentes do WAE neste estudo, cada elemento será testado individualmente utilizando a ferramenta ASES. Considerando que os componentes de segundo nível são incorporados dentro dos de primeiro nível, os testes serão organizados conforme os três principais componentes estruturais da ferramenta. O primeiro conjunto de testes avaliará os componentes do cabeçalho, seguido pela avaliação dos componentes do corpo principal e, por fim, dos componentes do rodapé. A conformidade de cada componente com as diretrizes de acessibilidade será mensurada, sendo considerado acessível aquele que atingir uma pontuação mínima de 95%. Esta pontuação foi escolhida com base na faixa de classificação do ASES, em que 95% a 100% reflete a nota mais alta de acessibilidade, assegurando que os componentes do WAE atendam ao mais elevado padrão de exigência em termos de acessibilidade. Essa metodologia visa garantir a plena aderência de todos os elementos do WAE aos padrões de acessibilidade estabelecidos. Os componentes de configuração e de agrupamento de conteúdo serão excluídos dos testes, uma vez que não geram código HTML, atuando apenas no suporte e na alteração da exibição dos demais componentes.

A Figura 1 apresenta um exemplo de arquivo *YAML* utilizado como entrada para o *Web Accessibility Engine*. O documento define parâmetros de configuração, estrutura de cabeçalho, corpo principal e rodapé, evidenciando a sintaxe hierárquica que facilita a leitura por pessoas cegas ou com baixa visão. Já a Figura 2 mostra a página *web* estática gerada pelo processo de transpilação, na qual é possível observar a correta renderização dos componentes declarados, a preservação da semântica e o alinhamento com as diretrizes de acessibilidade.

exemplo.yaml

```yaml
configurações:
  idioma: "pt"
  título: Minha primeira página web

cabeçalho:
  posição: superior
  imagem:
    caminho: Logo.png
    texto alternativo: Texto alternativo da imagem do header
  navegação:
    - Início
    - Quem Somos
    - Trabalhe Conosco
    - Fale conosco
  grupo:
    busca:
      texto: Digite o que procura
      borda: linha
    botão:
      texto: Log in
      contorno: linha
    botão:
      texto: Cadastro
      contorno: preencher

corpo principal:
  título: Título da página
  artigo:
    seção:
      título: Título da seção
      subtítulo: Subtítulo da seção
      paragrafo: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas facilisis convallis mi, vitae dictum nisl euismod in. Suspendisse consequat,
      imagem:
        caminho: https://images.unsplash.com/photo-1519002437326-bfd121eb53ae?q=80&w=2070&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1vYWdl
        texto alternativo: Texto alternativo da imagem do body

rodapé:
  paragrafo: Todos os direitos reservados
```

Fig. 1. Exemplo da estrutura de um arquivo YAML submetido à etapa de transpilação.

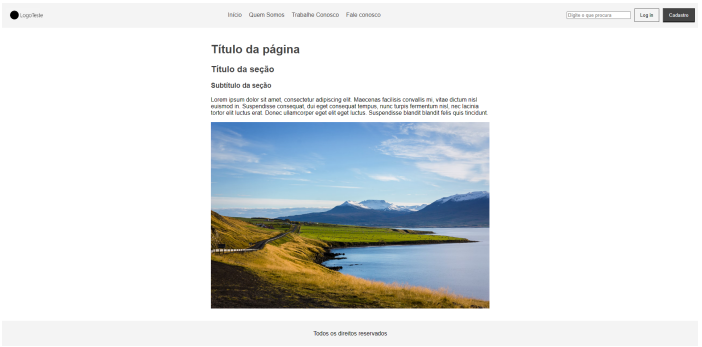

Fig. 2. Exemplo de página gerada por meio da ferramenta de transpilação WAE.

V. Desenvolvimento do Transpilador

Na condução desta pesquisa, a escolha da linguagem YAML como meio de representação em alto nível para a prototipação de páginas *web* foi guiada principalmente por sua acessibilidade e simplicidade. O YAML, notável por sua clareza e estrutura direta, oferece uma abordagem que se assemelha à linguagem natural, tornando-o uma opção intuitiva para a representação de componentes HTML. Essa característica é particularmente benéfica para programadores com deficiência visual, facilitando o entendimento e a manipulação do código devido à sua estrutura simples e facilidade de leitura.

A seleção de uma ferramenta apropriada para a codificação também foi importante nesta pesquisa. O *Grid-Coding* foi escolhido devido às suas características específicas. Projetado especialmente para programadores com deficiência visual, o *Grid-Coding* proporciona uma interface de programação acessível, estruturada e adaptável [7].

O *Grid-Coding* [7] distingue-se por sua metodologia única de organização do código em uma grade, otimizando a interação com tecnologias assistivas e facilitando a navegação e compreensão do código. A implementação do *Grid-Coding* neste projeto visa estabelecer um ambiente de desenvolvimento inclusivo, permitindo que programadores cegos participem de maneira significativa no desenvolvimento *web*.

Para a implementação do transpilador, optou-se pela linguagem C# em conjunto com o *framework* .NET 8, dada sua robustez e eficiência em aplicações distribuídas de alto desempenho. O C# é notável por seu manejo eficaz de grandes volumes de trabalho, assegurando um processamento confiável e rápido. O .NET 8, com seu suporte abrangente para a construção de APIs *web* modernas, proporciona a plataforma ideal para o desenvolvimento do serviço de transpilação, facilitando uma integração harmoniosa com a IDE baseada na *web*.

VI. Definição e Implantação da Arquitetura

Neste projeto, optou-se por uma arquitetura distribuída, caracterizada pela separação entre a Integrated Development Environment (IDE, Ambiente de Desenvolvimento Integrado) e o serviço de transpilação. A IDE (denominada *Grid Editor*) foi concebida como uma aplicação *web*, uma escolha que visa maximizar a acessibilidade e a facilidade de distribuição. Essa IDE *web*, implementada como um único arquivo HTML, destaca-se por sua simplicidade e capacidade de replicação. O uso exclusivo de componentes nativos no desenvolvimento da IDE foi uma decisão intencional, para assegurar a máxima compatibilidade com leitores de tela e outras ferramentas assistivas, fundamentais para usuários com deficiência visual.

Por outro lado, a lógica de transpilação foi encapsulada em uma API separada, na qual o transpilador propriamente dito foi implementado. Essa separação entre a interface e o processamento permite uma estrutura mais flexível e escalável, facilitando atualizações e manutenções.

A arquitetura detalhada e o fluxo de trabalho do sistema proposto são ilustrados na Figura 3. Essa figura delineia os dois módulos principais da aplicação e seus componentes interconectados.

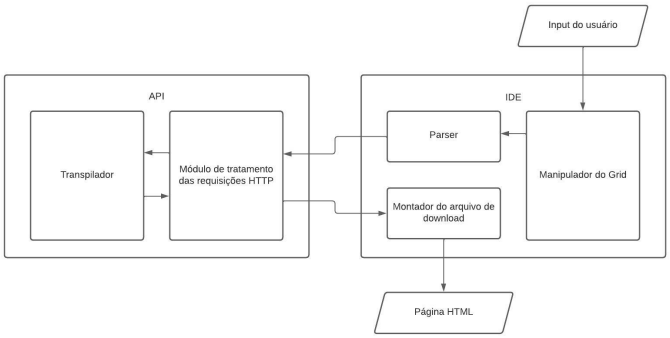

Fig. 3. Fluxo de trabalho e representação da arquitetura adotada.

Na IDE, o componente central é o Manipulador do *Grid*, que desempenha um papel vital na interação do usuário com a aplicação. Esse manipulador gerencia a tabela onde ocorre o desenvolvimento, ajustando dinamicamente sua estrutura e aparência com base nas entradas do usuário. Além disso, a IDE incorpora dois componentes adicionais: o *Parser* e o Montador do Arquivo de *Download*. O Parser é responsável por converter o conteúdo da tabela em formato YAML e enviá-lo para a API, enquanto o Montador do Arquivo de *Download* processa a resposta da API, compilando um arquivo para download que contém a página *web* transpilada.

No lado do servidor, a API divide-se em dois componentes principais. O Transpilador, que é o ponto focal da aplicação, encarrega-se de converter o YAML recebido em uma página HTML. Esse processo de transpilação é a chave para transformar a estrutura de alto nível do YAML em uma página *web* funcional. Paralelamente, o módulo de Tratamento de Requisições HTTP gerencia a comunicação entre a IDE e a API. Esse módulo recebe o arquivo YAML enviado pela IDE e retorna o HTML resultante da transpilação, estabelecendo, assim, um fluxo de trabalho integrado entre a interface do usuário e o processamento no servidor.

A. Especificações do Transpilador

Para a implementação do transpilador, adotou-se uma arquitetura baseada em componentes. Essa arquitetura é composta por quatro componentes principais, denominados componentes de primeiro nível, cada um desempenhando funções específicas na configuração e organização estrutural da página *web*. Cada componente principal possui subcomponentes, chamados de componentes de segundo nível, responsáveis por definir e configurar os elementos visuais dentro dos componentes de primeiro nível. Os componentes de primeiro nível são os seguintes:

1) Componente de Configuração: Este componente é fundamental para a personalização geral da página *web*. Ele contém subcomponentes destinados a ajustar configurações críticas, como o tema da página e o título. Esses subcomponentes oferecem uma interface flexível para modificar aspectos essenciais da página, garantindo que a personalização possa ser realizada de forma acessível e intuitiva.

2) Componentes Estruturais: Os demais componentes principais são dedicados à estrutura da página. Cada um desses componentes inclui subcomponentes internos que representam elementos comuns em páginas *web*, sendo:

1) O componente do cabeçalho pode incluir subcomponentes para logotipos, menus de navegação e *links*.
2) O corpo principal pode conter subcomponentes para textos, imagens, botões, formulários e outros elementos interativos.
3) O rodapé, por sua vez, pode abrigar subcomponentes para informações de contato, *links* de redes sociais e direitos autorais.

Na Tabela I, são apresentados os detalhes da relação entre os componentes principais e seus respectivos subcomponentes internos, proporcionando uma visão clara da arquitetura do transpilador e de como cada elemento contribui para a construção final da página *web*. Na Figura 1, é possível observar um exemplo de um arquivo YAML que utiliza os componentes descritos na tabela; esse arquivo servirá como entrada para gerar a página *web* apresentada na Figura 2.

TABELA I
TABELA DE COMPONENTES.

Componentes de Primeiro Nível	Componentes de Segundo Nível
"configurações"	"idioma" "título" "tema"
"cabeçalho"	"posição" "imagem" "navegação" "busca" "botão" "grupo"
"corpo principal"	"artigo" "seção" "título" "subtítulo" "parágrafo" "imagem" "grupo"
"rodapé"	"parágrafo"

Na arquitetura do transpilador desenvolvido, os componentes são concebidos como representações abstratas de elementos comuns de páginas *web*. Ao serem declarados no código, esses componentes criam automaticamente um elemento correspondente na página, seguindo uma estilização minimalista e neutra. Essa abordagem assegura que, independentemente do elemento específico, seja um cabeçalho, parágrafo de texto, imagem ou botão, o resultado final seja acessível e esteticamente coeso.

Cada componente é projetado para encapsular funcionalidades específicas e estilos pré-definidos, o que simplifica o processo de desenvolvimento. Por exemplo, ao declarar um componente de botão, o transpilador gera um botão na página *web* com uma estilização minimalista padrão, que abrange aspectos como tamanho, cor e fonte. Essa estilização neutra garante que os elementos sejam visualmente claros e facilmente adaptáveis.

3) Especificações dos Componentes do WAE: **Configurações.** Define as configurações gerais da página, incluindo:

1) **Idioma (idioma)**: Define o idioma da página *web*. Os valores possíveis seguem o padrão do HTML, incluindo "pt-BR" (Português do Brasil, padrão), "en" (Inglês) e "es" (Espanhol).
2) **Título (título)**: Especifica o título da página *web*, que é exibido na aba do navegador.
3) **Tema (tema)**: Permite definir o tema visual da página, podendo ser "padrão" ou "noturno".

Cabeçalho. Define a seção do cabeçalho da página, incluindo:

1) **Posição (posição)**: Determina a posição do cabeçalho na página, com "superior" como valor padrão, podendo assumir também "lateral esquerda" ou "lateral direita".
2) **Imagem (imagem)**: Permite inserir uma imagem no cabeçalho, especificando o caminho (URL) da imagem e um texto alternativo para acessibilidade.
3) **Navegação (navegação)**: Define uma lista de links de navegação, que podem ser nomeados diretamente ou associados a um identificador de seção no corpo da página.
4) **Grupo de Controles (grupo)**: Agrupa elementos de interação, como campos de busca e botões, permitindo especificações detalhadas como texto, borda e estilo do contorno.

Corpo Principal. Descreve o conteúdo principal da página, incluindo:

1) **Título (título)**: O título principal da página.
2) **Artigos (artigo)**: Permite a definição de artigos contendo seções com títulos, subtítulos, parágrafos e imagens, promovendo a organização semântica do conteúdo.
3) **Subtítulo e Parágrafos**: Textos adicionais que fornecem contexto ou informações complementares.
4) **Imagem**: Uma imagem no corpo da página, com caminho e texto alternativo definidos para acessibilidade.

Rodapé. Descreve o conteúdo exibido no rodapé da página, incluindo:

1) **Parágrafo (parágrafo)**: Conteúdo textual, geralmente utilizado para direitos autorais ou informações de contato, identificado por um título de seção específico.

4) Diretrizes WCAG Utilizadas no WAE: Para garantir que os componentes gerados pelo transpilador sejam acessíveis e estejam em conformidade com as WCAG [23], diversas práticas recomendadas foram aplicadas no processo de transpilação do YAML para HTML.

Primeiramente, a inclusão de textos alternativos para imagens, conforme especificado nos campos "texto alternativo" nas definições de imagem tanto no cabeçalho quanto no corpo principal, é uma prática fundamental para usuários que dependem de leitores de tela. Essa prática cumpre o Critério de Sucesso WCAG 1.1.1 (Conteúdo Não Textual) [23], que exige que todo conteúdo não textual ofereça uma alternativa textual que possa ser transformada em outras formas, como grandes impressões, braille, fala, símbolos ou linguagem simplificada.

Além disso, a estrutura semântica do HTML gerado é reforçada pela utilização de cabeçalhos hierárquicos ($h1, h2, h3$), garantindo que o conteúdo seja organizado e navegável de forma lógica. Isso atende ao Critério de Sucesso WCAG 1.3.1 (Informação e Relações) [23], que visa garantir que informações, estruturas e relações visuais estejam disponíveis para todos os usuários, incluindo aqueles que utilizam tecnologias assistivas.

A definição de áreas específicas, como cabeçalho, navegação, corpo principal e rodapé, além de seções claramente delimitadas dentro dos artigos, facilita a navegação e a compreensão do conteúdo da página. A implementação de controles de formulário com descrições claras e botões com textos explicativos também contribui para a acessibilidade, atendendo ao Critério de Sucesso WCAG 2.1.1 (Teclado) [23], que assegura que todos os componentes da interface possam ser operados por meio do teclado, facilitando o uso por pessoas com limitações motoras.

B. Desenvolvimento do Grid Editor

O *Grid Editor*, atuando como um ambiente de desenvolvimento integrado (IDE), representa um avanço significativo na área da programação acessível, por meio da implementação do paradigma de codificação *Grid-Coding*. Originalmente projetado para a compilação da linguagem *Python*, caracteriza-se por sua abordagem baseada em indentação, uma característica também presente na linguagem YAML. No entanto, a versão inicial do *Grid Editor* não foi desenvolvida para operar diretamente com a sintaxe do YAML, o que levou à necessidade de adaptar a IDE para essa linguagem específica.

1) Adaptação para YAML e Integração com o WAE: A necessidade de uma IDE que suportasse YAML e estivesse alinhada com as funcionalidades do transpilador WAE motivou a criação de uma versão personalizada do *Grid Editor*. Essa versão adaptada foca em incorporar o paradigma do *Grid-Coding* de maneira compatível com a estrutura e sintaxe do YAML, facilitando o processo de desenvolvimento para programadores com deficiência visual. Além disso, foram adicionados os comandos e componentes específicos do WAE, conforme apresentados na Tabela I. A estrutura final da IDE pode ser visualizada na Figura 4.

Na interface da IDE, assim como na implementação original, atalhos de teclado foram implementados para agilizar o fluxo de trabalho: “Alt + 1” ativa a tabela de edição, enquanto “Alt + 2” inicia a compilação e o processo de *download* do HTML transpilado. Esses atalhos têm como objetivo oferecer um método eficiente para que usuários com deficiência visual possam navegar pela interface.

Quando um componente de primeiro nível é declarado pelo usuário, as células abaixo dessa declaração são atualizadas para exibir uma mensagem indicativa de que o componente está sendo editado. Essa funcionalidade, herdada da implementação original do *Grid Editor*, visa eliminar a necessidade de inserção manual de espaços para indentação, uma prática crítica no contexto do YAML, onde a hierarquia e a estrutura dos elementos dependem da correta organização de espaços. Ao automatizar esse processo, a edição se torna mais intuitiva e a probabilidade de erros de sintaxe é reduzida.

Outro benefício importante dessa prática é a melhoria da leitura por tecnologias assistivas: em vez de relatar espaços em branco, o leitor de tela anuncia diretamente onde o usuário está editando, o que contribui significativamente para uma experiência mais fluida e acessível.

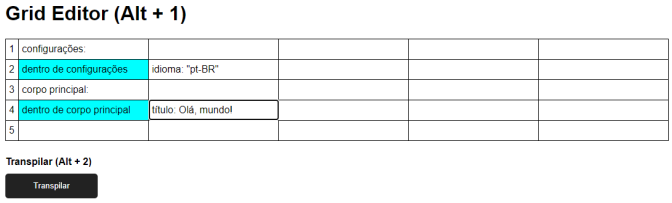

Fig. 4. IDE - *Grid Editor.*

VII. Resultados

Após a execução dos testes de acessibilidade com a ferramenta ASES, o foco concentrou-se nos componentes de *design*, correspondentes aos elementos efetivamente transpilados para HTML. Elementos dedicados à configuração e à marcação de conteúdo, que não se traduzem diretamente em elementos HTML, não foram incluídos nos testes. Para uma distinção clara, a Tabela II lista os componentes que não foram submetidos à avaliação, enquanto a Tabela III detalha os componentes analisados, juntamente com os respectivos resultados obtidos nos testes de acessibilidade.

Na Tabela III, que apresenta os resultados da avaliação de acessibilidade, destaca-se o componente *subtítulo*, pertencente à seção *corpo principal*, que alcançou 90,35% na avaliação do ASES, ficando abaixo do limiar estabelecido de 95%. Embora essa pontuação represente um bom nível de acessibilidade, ela impede que o componente *subtítulo* seja considerado totalmente acessível. O desempenho inferior deve-se ao não cumprimento do critério “Utilizar corretamente os níveis de cabeçalho”.

Na metodologia adotada, cada componente foi testado individualmente. O componente *subtítulo* do WAE, ao gerar um elemento de cabeçalho “<h2>” no HTML, não observou a diretriz 2.4.10 do WCAG [12], que recomenda uma estrutura hierárquica adequada para títulos, exigindo que um componente com a *tag* “<h1>” anteceda qualquer “<h2>”.

Observa-se que, embora o WAE produza componentes nativos e acessíveis em HTML, parte da responsabilidade pela acessibilidade continua a ser do desenvolvedor. É necessário, por exemplo, seguir corretamente a hierarquia de títulos na página, fazer uso de componentes como *artigo* e *seção* para organizar o conteúdo eficientemente, entre outras práticas recomendadas de codificação HTML. Dessa forma, ainda que o WAE simplifique a criação de estruturas acessíveis, a obtenção

de acessibilidade plena depende da atenção e ação consciente dos desenvolvedores ao construírem suas páginas.

TABELA II
COMPONENTES EXCLUÍDOS DOS TESTES.

Componentes Excluídos da Avaliação de Acessibilidade	
"configurações"	"idioma", "título", "tema"
"cabeçalho"	"posição", "grupo"
"corpo principal"	"artigo", "seção", "grupo"

TABELA III
TABELA COM O RESULTADO DA AVALIAÇÃO DOS COMPONENTES DO WAE.

Componentes Avaliados na Ferramenta ASES		
Componente Pai	**Componente Avaliado**	**Resultado ASES**
"cabeçalho"	"imagem"	96.58%
	"navegação"	98.03%
	"busca"	96.97%
	"botão"	96.07%
"corpo principal"	"título"	97.5%
	"subtítulo"	90.35%
	"parágrafo"	96.07%
	"imagem"	96.58%
"rodapé"	"parágrafo"	96.07%

VIII. CONCEITOS ÉTICOS E CONSIDERAÇÕES ÉTICAS

O desenvolvimento e a adoção do *Web Accessibility Engine* (WAE) estão ancorados em princípios éticos que asseguram não apenas sua robustez técnica, mas também a promoção da inclusão, da autonomia e da responsabilidade social. A seguir, discutem-se os principais conceitos e implicações éticas que nortearam esta pesquisa.

A. Inclusão e Justiça Social

A acessibilidade digital é um direito fundamental, reforçado pelas *Web Content Accessibility Guidelines* (WCAG) [23]. Ao incorporar essas diretrizes durante a transpilação, o WAE contribui para reduzir barreiras históricas que marginalizam programadores cegos ou com baixa visão. Estudos demonstram que a falta de ferramentas acessíveis afeta negativamente a motivação e a permanência de estudantes e profissionais cegos na área de Computação [1], [5].

B. Autonomia e Agência do Usuário

Apesar de automatizar grande parte das verificações de acessibilidade, o WAE mantém o programador no centro das decisões de design. A combinação do paradigma *Grid-Coding* com YAML oferece atalhos de teclado consistentes e *feedback* auditivo, favorecendo a autonomia do desenvolvedor [4], [7].

C. Privacidade e Segurança de Dados

A arquitetura distribuída *Grid Editor* no navegador e serviço de transpilação dedicado, minimiza o compartilhamento de dados sensíveis e evita o armazenamento de informações pessoais em servidores externos, alinhando-se às recomendações brasileiras de acessibilidade e segurança [24].

D. Transparência, Responsabilidade e Sustentabilidade

O código-fonte será disponibilizado sob licença aberta, permitindo auditoria e extensões pela comunidade. Essa transparência facilita a identificação de vieses e assegura a atualização do sistema conforme novas versões das WCAG ou avanços em transpiladores [14]. A manutenção contínua é essencial para garantir a relevância e a confiabilidade da ferramenta a longo prazo.

E. Ética na Pesquisa com Populações Vulneráveis

Os estudos de usabilidade planejados seguirão protocolos éticos rigorosos: consentimento informado, anonimato dos participantes e direito de recusa sem prejuízo. As técnicas de *Think-Aloud* empregadas em testes de acessibilidade são reconhecidas pela literatura como adequadas na investigação de experiências de usuários com deficiência visual [20], [21].

F. Mitigação de Limitações e Viéses

Ainda que nove dos dez componentes avaliados tenham superado o limiar de 95% de conformidade, práticas inadequadas de desenvolvimento, como a hierarquia incorreta de títulos, podem deteriorar a acessibilidade final [12]. Para mitigar esse risco, futuras versões do WAE devem incluir verificações em tempo real que alertem o programador sobre violações semânticas e recomendem correções imediatas.

IX. CONCLUSÃO

O *Web Accessibility Engine* (WAE) demonstrou ser um avanço significativo na promoção da acessibilidade para programadores cegos ou com baixa visão. A adoção da arquitetura distribuída, composta pelo *Grid Editor* em *front-end* acessível e por um serviço de transpilação em *back-end*, viabilizou um fluxo de trabalho que preserva a simplicidade de uso sem abdicar de robustez técnica. Os testes conduzidos com o Avaliador e Simulador de Acessibilidade em Sítios (ASES) confirmaram que nove dos dez componentes avaliados ultrapassaram o limiar de 95% de conformidade, atestando a eficácia do enfoque metodológico centrado na inserção de boas práticas das WCAG durante a própria etapa de transpilação.

Ainda assim, a análise evidenciou que a totalidade da acessibilidade não depende unicamente da ferramenta: práticas de desenvolvimento inadequadas, como a violação da hierarquia de títulos que comprometeu o desempenho do componente

subtítulo (90,35%), podem degradar a experiência do usuário final. Esses resultados reforçam que o WAE reduz, mas não elimina, a carga de responsabilidade dos desenvolvedores quanto ao cumprimento das diretrizes de acessibilidade.

Entre as principais contribuições desta pesquisa destacam-se a integração inédita do paradigma *Grid-Coding* com YAML para a geração automática de páginas HTML acessíveis; a disponibilização de uma IDE 100% baseada em tecnologias nativas do navegador, compatível com leitores de tela, que dispensa instalação local; e a validação empírica, por meio do ASES, que mostrou que o modelo proposto alcança padrões de excelência comparáveis aos obtidos manualmente por especialistas.

Como continuidade, propõe-se expandir o catálogo de componentes, contemplando elementos dinâmicos (por exemplo, carrosséis e *hyperlinks* avançados), incorporar verificações automáticas de hierarquia semântica e sugestões de correção em tempo real, e realizar estudos longitudinais com grupos maiores de programadores cegos para mensurar ganhos de produtividade e aprendizagem. A médio prazo, almeja-se disponibilizar o WAE como serviço independente da IDE, viabilizando sua integração em ambientes de desenvolvimento já consolidados e potencializando seu impacto em cenários educacionais, corporativos e governamentais.

Ao conjugar inclusão, autonomia, privacidade, transparência e responsabilidade continuada, o WAE demonstra que é possível aliar excelência técnica a valores éticos. A ferramenta amplia oportunidades de formação e atuação profissional para desenvolvedores com deficiência visual, contribuindo para um ecossistema de desenvolvimento *web* inclusivo.

Em síntese, o WAE contribui para tornar o desenvolvimento *web* mais inclusivo sem exigir conhecimentos profundos de acessibilidade por parte do programador. Ao conciliar acessibilidade intrínseca, facilidade de uso e validação formal, a ferramenta se posiciona como referência promissora para iniciativas que visem democratizar o acesso à programação e ao conteúdo digital para todos.

Referências

[1] J. M. Fernández-Batanero, M. Montenegro-Rueda, J. Fernández-Cerero, and I. García-Martínez, "Assistive technology for the inclusion of students with disabilities: a systematic review," *Educational technology research and development*, vol. 70, no. 5, pp. 1911–1930, 2022.

[2] M. Manzoor and V. Vimarlund, "Digital technologies for social inclusion of individuals with disabilities," *Health and technology*, vol. 8, pp. 377–390, 2018.

[3] V. Potluri, P. Vaithilingam, S. Iyengar, Y. Vidya, M. Swaminathan, and G. Srinivasa, "Codetalk: Improving programming environment accessibility for visually impaired developers," in *Proceedings of the 2018 chi conference on human factors in computing systems*, 2018, pp. 1–11.

[4] M. Pandey, V. Kameswaran, H. V. Rao, S. O'Modhrain, and S. Oney, "Understanding accessibility and collaboration in programming for people with visual impairments," *Proceedings of the ACM on Human-Computer Interaction*, vol. 5, no. CSCW1, pp. 1–30, 2021.

[5] C. M. Baker, C. L. Bennett, and R. E. Ladner, "Educational experiences of blind programmers," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 2019, pp. 759–765.

[6] M. Research, "Web development market 2025-2034: Size, share, growth," *MarkWide Research*, 2025, relatório de mercado sobre desenvolvimento web. [Online]. Available: https://markwideresearch.com/web-development-market/

[7] M. Ehtesham-Ul-Haque, S. M. Monsur, and S. M. Billah, "Grid-coding: An accessible, efficient, and structured coding paradigm for blind and low-vision programmers," in *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, 2022, pp. 1–21.

[8] C. Evans, I. döt Net, and O. Ben-Kiki, "YAML Ain't Markup Language (YAML™) Version 1.2," 2024, acessado em: 02 de Janeiro. [Online]. Available: https://yaml.org/spec/1.2/spec.html

[9] A.-G. Federal, "Avaliador e simulador de acessibilidade em sıtios," 2014, acessado em: 23 de Janeiro. [Online]. Available: https://asesweb.governoeletronico.gov.br/criteriosSucesso

[10] E. Schanzer, S. Bahram, and S. Krishnamurthi, "Accessible ast-based programming for visually-impaired programmers," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 2019, pp. 773–779.

[11] M. Ribera, M. Porras, M. Boldu, M. Termens, A. Sule, and P. Paris, "Web content accessibility guidelines 2.0: A further step towards accessible digital information," *Program*, vol. 43, no. 4, pp. 392–406, 2009.

[12] F. Boland and E. Fong, "Challenges and benefits of a methodology for scoring web content accessibility guidelines (wcag) 2.0 conformance," National Institute of Standards and Technology (NIST), Tech. Rep. NISTIR 8010, September 2014. [Online]. Available: http://dx.doi.org/10.6028/NIST.IR.8010

[13] A. H. Al-Badi, A. O. Al Majeeni, P. J. Mayhew, and A. S. Al-Rashdi, "Improving website ranking through search engine optimization," *Journal of Internet and e-business Studies*, vol. 2011, pp. 1–11, 2011.

[14] A. Bastidas Fuertes, M. Pérez, and J. Meza Hormaza, "Transpilers: A systematic mapping review of their usage in research and industry," *Applied Sciences*, vol. 13, no. 6, p. 3667, 2023.

[15] A. Nuñez, A. Moquillaza, and F. Paz, "Web accessibility evaluation methods: a systematic review," in *Design, User Experience, and Usability. Practice and Case Studies: 8th International Conference, DUXU 2019, Held as Part of the 21st HCI International Conference, HCII 2019, Orlando, FL, USA, July 26–31, 2019, Proceedings, Part IV 21*. Springer, 2019, pp. 226–237.

[16] J. Abascal, M. Arrue, and X. Valencia, "Tools for web accessibility evaluation," *Web accessibility: a foundation for research*, pp. 479–503, 2019.

[17] M. Beaudouin-Lafon and W. E. Mackay, "Prototyping tools and techniques," in *The human-computer interaction handbook*. CRC Press, 2007, pp. 1043–1066.

[18] C. Kussmaul and R. Jack, "Prototyping in web development," in *Software Engineering for Modern Web Applications: Methodologies and Technologies*. IGI Global, 2008, pp. 191–206.

[19] C. Fry, M. Plusch, and H. Lieberman, "Static and dynamic semantics of the web." in *Spinning the semantic web*, 2003, pp. 377–401.

[20] J. R. Lewis, "Usability testing," *Handbook of human factors and ergonomics*, pp. 1267–1312, 2012.

[21] M. Nørgaard and K. Hornbæk, "What do usability evaluators do in practice? an explorative study of think-aloud testing," in *Proceedings of the 6th conference on Designing Interactive systems*, 2006, pp. 209–218.

[22] S. Vallian, "Kustomisasi sharif judge untuk kebutuhan program studi teknik informatika," 2018, tese sobre a customização do sistema Sharif Judge para atender às necessidades do curso de Engenharia de Informática. [Online]. Available: http://hdl.handle.net/123456789/7524

[23] B. Caldwell, M. Cooper, L. G. Reid, G. Vanderheiden, W. Chisholm, J. Slatin, and J. White, "Web content accessibility guidelines (wcag) 2.0," *WWW Consortium (W3C)*, vol. 290, pp. 1–34, 2008.

[24] J. A. P. Rocha and A. B. S. Duarte, "Diretrizes de acessibilidade web: um estudo comparativo entre as wcag 2.0 e o e-mag 3.0," *Inclusão Social*, vol. 5, no. 2, 2012.