UNIVERSIDADE ESTADUAL DO PIAUÍ
CENTRO DE TECNOLOGIA E URBANISMO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Denilson Mendes de Moura

DESAFIOS DA INSERÇÃO DE REQUISITOS DE ACESSIBILIDADE EM UMA APLICAÇÃO WEB: DESENVOLVENDO UM FÓRUM ACESSÍVEL PARA PESSOAS CEGAS E DE BAIXA VISÃO

TERESINA
2025

Denilson Mendes de Moura

DESAFIOS DA INSERÇÃO DE REQUISITOS DE ACESSIBILIDADE EM UMA APLICAÇÃO WEB: DESENVOLVENDO UM FÓRUM ACESSÍVEL PARA PESSOAS CEGAS E DE BAIXA VISÃO

Monografia submetido à Universidade Estadual do Piauí como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Lianna Mara Castro Duarte

TERESINA

2025

M929d Moura, Denilson Mendes de.
Desafios da inserção de requisitos de acessibilidade em uma aplicação web: desenvolvendo um fórum acessível para pessoas cegas e de baixa visão / Denilson Mendes de Moura. - 2025.
71f.: il.

Monografia (graduação) - Universidade Estadual do Piauí - UESPI, Curso de Bacharelado em Ciências da Computação, Campus Poeta Torquato Neto, Teresina - PI, 2025.
"Orientador: Prof.ª Dr.ª Lianna Mara Castro Duarte".

1. Acessibilidade Web. 2. WCAG. 3. Desenvolvimento de Software. I. Duarte, Lianna Mara Castro . II. Título.

CDD 004

Ficha elaborada pelo Serviço de Catalogação da Biblioteca da UESPI
ANA ANGELICA PEREIRA TEIXEIRA (Bibliotecário) CRB-3ª/1217

DESAFIOS DA INSERÇÃO DE REQUISITOS DE ACESSIBILIDADE EM UMA APLICAÇÃO WEB: DESENVOLVENDO UM FÓRUM ACESSÍVEL PARA PESSOAS CEGAS E DE BAIXA VISÃO

Denilson Mendes de Moura

Monografia submetido à Universidade Estadual do Piauí como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação.

gov.br
Documento assinado digitalmente
LIANNA MARA CASTRO DUARTE
Data: 12/07/2025 09:32:30-0300
Verifique em https://validar.iti.gov.br

Profa. Dra. Lianna Mara Castro Duarte
Orientador

Nota da Banca Examinadora: 8.8

Banca Examinadora:

gov.br
Documento assinado digitalmente
LIANNA MARA CASTRO DUARTE
Data: 12/07/2025 09:31:36-0300
Verifique em https://validar.iti.gov.br

Profa. Dra. Lianna Mara Castro Duarte
Presidente

gov.br
Documento assinado digitalmente
JOSE DE RIBAMAR MARTINS BRINGEL FILHO
Data: 11/07/2025 15:33:21-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. José de Ribamar Martins Bringel Filho
Membro

gov.br
Documento assinado digitalmente
Reginaldo Rodrigues das Gracas
Data: 11/07/2025 21:27:25-0300
Verifique em https://validar.iti.gov.br

Prof. Me. Reginaldo Rodrigues das Graças
Membro

Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.

AGRADECIMENTOS

Agradeço, em primeiro lugar, a Deus, pela Sua graça e bondade que me sustentaram ao longo de toda a jornada acadêmica. Sem o Seu amparo e as bênçãos concedidas, muitos desafios teriam sido mais difíceis de superar.

Aos meus pais, sou imensamente grato pelo amor incondicional, pela fé em meu potencial e pelos investimentos — financeiros, emocionais e de tempo — que fizeram para que eu pudesse chegar até aqui. Vocês sempre acreditaram em mim, mesmo nos momentos em que eu duvidava de minhas próprias capacidades.

À minha professora e orientadora, Lianna, devo um reconhecimento especial. Obrigado pela paciência, pelo compromisso e pela insistência construtiva ao me guiar em cada etapa do trabalho. Sua dedicação, críticas criteriosas e incentivo constante foram fundamentais para a concretização deste TCC.

Também agradeço a todos os amigos que, de diversas formas, me apoiaram e incentivaram ao longo do curso. As palavras de encorajamento, o companheirismo nas horas de estudo e as conversas que renovaram minha motivação contribuíram significativamente para que eu me mantivesse focado e confiante.

Por fim, a todos que, direta ou indiretamente, colaboraram para o meu crescimento pessoal e acadêmico, deixo meu sincero agradecimento. Cada gesto de apoio e cada ensinamento, mesmo nos detalhes mais simples, foram essenciais para a conclusão deste trabalho. Muito obrigado!

“A arma mais forte em dois milhões de anos
de história humana: tecnologia de comunicação.“
(Inagaki, Riichiro; Dr. Stone, 2019)

.

RESUMO

Este trabalho apresenta os desafios e resultados da inserção de requisitos de acessibilidade no desenvolvimento de uma aplicação web voltada a pessoas cegas e de baixa visão. Para isso, foi construída uma prova de conceito denominada *Inclusify*, um fórum acessível desenvolvido a partir das diretrizes WCAG 2.1 e com foco em boas práticas de usabilidade assistiva. A pesquisa envolveu uma revisão sistemática da literatura, a modelagem e codificação da aplicação utilizando metodologias ágeis, além de testes automatizados com o AXE-CORE e testes manuais com leitor de tela (NVDA). O processo evidenciou diversas dificuldades, como a ausência de suporte adequado em componentes de terceiros, a curva de aprendizado no uso de tecnologias assistivas e no uso correto das tags semânticas do HTML5. Como resultado, foram produzidos não apenas o fórum funcional e acessível, mas também um conjunto de recomendações práticas para orientar desenvolvedores quanto à inclusão digital desde as etapas iniciais de seus projetos. A experiência contribui para o avanço das discussões sobre acessibilidade digital aplicada e propõe caminhos para sua integração em ciclos ágeis de desenvolvimento.

Palavras-chave: Acessibilidade Web. WCAG. Desenvolvimento de Software. Pessoas cegas e de baixa visão.

ABSTRACT

This study presents the challenges and outcomes of incorporating accessibility requirements into the development of a web application aimed at blind and visually impaired users. To this end, a proof of concept named *Inclusify* was developed, an accessible forum built following the WCAG 2.1 guidelines and focused on assistive usability best practices. The research involved a systematic literature review, modeling and implementation of the application using agile methodologies, and both automated testing with AXE-CORE and manual testing with the NVDA screen reader. The development process revealed several challenges, including the lack of support in third-party components, the learning curve for assistive technologies, and ensuring the correct use of semantic HTML5 tags. As outcomes, the project delivered not only a functional and accessible forum but also a set of practical recommendations to guide developers in implementing inclusive design from the initial stages of their projects. This experience contributes to the advancement of applied digital accessibility and suggests ways to integrate it into agile development cycles.

Keywords: Web Accessibility. WCAG. Software Development. Blind and Visually Impaired People

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

ARIA	Accessible Rich Internet Applications
CI/CD	Continuous Integration/Continuous Deployment
NVDA	NonVisual Desktop Access
ORM	Object-Relational Mapping
TIC	Tecnologias da Informação e da Comunicação
W3C	World Wide Web Consortium
WAI	Web Accessibility Initiative
WCAG	Web Content Accessibility Guidelines
WebAIM	Web Accessibility In Mind
XP	eXtreme Programming

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

A Web tem crescido de maneira surpreendente desde sua criação, em 1992. Com sua expansão, surgiram várias formas de apresentar conteúdo (texto, áudio, vídeo) em diferentes idiomas e dispositivos (tablets, celulares, TVs). Todo esse avanço trouxe bastante comodidade e praticidade, de forma que se tornou uma ferramenta indispensável atualmente, uma vez que se transformou em uma das principais fontes de acesso à informação disponíveis, onde se pode obter notícias, informações governamentais, conteúdos educacionais e de lazer.

Uma vez que a internet tem ganhado tanta relevância, tem-se revelado notória a necessidade de esforços para tornar esta ferramenta comum e acessível a todos. Segundo o World Wide Web Consortium (W3C) e a Web Accessibility Initiative (WAI), acessibilidade significa que pessoas com deficiências, habilidades reduzidas ou deficiências induzidas pela situação devem ser capazes de acessar, navegar, interagir e contribuir com informações disponíveis em computadores, equipamentos eletrônicos e na Internet (W3C, 2021). Nesse sentido, a WAI elaborou o importante conjunto de diretrizes de acessibilidade, chamado Web Content Accessibility Guidelines (WCAG) (W3C, 2018).

Esse tema tem sido foco de diversas pesquisas na última década (Sanchez-Gordon et al., 2019; Bouraoui; Gharbi, 2019; Sánchez-Gordón; Moreno, 2014), contudo, apesar de sua evolução, a acessibilidade digital ainda encontra desafios em seu caminho. Isso se deve a fatores como:

- A falta de compreensão (por parte dos desenvolvedores) das técnicas que as pessoas com deficiência usam atualmente ao interagir com a tecnologia (Crabb et al., 2019);
- A falta de métodos fáceis de implementar recursos de acessibilidade de modo geral (Crabb et al., 2019);
- Cultura Ágil (Pellegrini et al., 2019);

De fato, diversos estudos apontam uma lacuna de conhecimento entre especialistas em software no que se refere às técnicas e regulamentações de acessibilidade digital. Segundo Bohman (2012) e Klironomos et al. (2006), essa deficiência de entendimento gera barreiras para pessoas com deficiência, físicas ou cognitivas, tornando-lhes difícil ou até impossível o acesso a produtos e conteúdos de grande relevância. Além disso, conforme argumentado por Baptista et al. (2016) e Inal, Rızvanoğlu e Yesilada (2019), os padrões e diretrizes existentes costumam ser complexos e de difícil compreensão, o que agrava ainda mais o desafio dos desenvolvedores.

Outro fator que atrasa o avanço da acessibilidade digital é a negligência na adoção de práticas ágeis pelas equipes de desenvolvimento. Embora os métodos ágeis promovam entregas rápidas e incrementais, várias pesquisas mostram que, nesses contextos, a acessibilidade costuma ficar em segundo plano. Miranda e Araujo (2022), por exemplo, investigaram 13 empresas que utilizam desenvolvimento ágil e concluíram que "a acessibilidade não é um item prioritário": a maior parte dos participantes possui conhecimento básico sobre diretrizes e tecnologias assistivas, e raramente emprega métodos ou ferramentas específicas para incorporar esses requisitos às histórias de usuário e aos critérios de aceitação.

Na prática, as equipes concentram-se em entregar o MVP (Minimum Viable Product) com as funcionalidades de maior valor nas primeiras iterações, deixando os requisitos de acessibilidade para versões posteriores. Conforme destacam Pellegrini et al. (2019), postergar o desenvolvimento de recursos acessíveis até depois que as funcionalidades principais estão prontas encarece sua implementação, pois demanda retrabalho e adaptações em partes já consolidadas do sistema. Esse ciclo de adiamentos não só eleva os custos como pode comprometer prazos e qualidade do produto final.

1.2 Justificativa

Como apontado por Pellegrini et al. (2019), a acessibilidade deve ser considerada logo no início do desenvolvimento do software, ainda em sua fase de requisitos e acompanhando por todas as fases até a entrega. De fato, no passado, estudos indicavam que a qualidade do produto depende da qualidade do processo (Humphrey, 1989; Fuggetta, 2000). Para isso, há vários esforços no sentido de integrar acessibilidade aos métodos ágeis, melhorando, assim, o processo de produção de software (Sanchez-Gordon et al., 2019; Bouraoui; Gharbi, 2019; Sánchez-Gordón; Moreno, 2014).

Considerando a necessidade de aplicações práticas de técnicas de acessibilidade no desenvolvimento Web, esse estudo justifica-se, entre outros fatores, pela necessidade de dirimir as lacunas entre as diretrizes e processos de acessibilidade web em âmbito teórico e a real aplicabilidade das mesmas. Contribuindo dessa forma para identificar as possíveis dificuldades enfrentadas pelos desenvolvedores e fornecer recomendações práticas para aprimorar a acessibilidade em projetos futuros por meio de uma prova de conceito.

Dessa forma, é possível que essa pesquisa impacte direta ou indiretamente o trabalho dos desenvolvedores. Além de contribuir para a ampliação do escopo científico de estudos dessa natureza.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo geral deste trabalho é aplicar as diretrizes de acessibilidade da WCAG em conjunto com metodologias ágeis no desenvolvimento de um fórum, que resultasse em uma aplicação web que seja acessível a pessoas cegas e de baixa visão. Tendo identificado e documentado as possíveis dificuldades enfrentadas no processo de desenvolvimento.

1.3.2 Objetivos Específicos

Para alcançar nosso objetivo geral, estabelecemos as seguintes metas específicas:

- Analisar as diretrizes de acessibilidade: compreender os princípios e critérios a serem considerados durante o desenvolvimento do fórum consoante as diretrizes de acessibilidade.
- Projetar e implementar um fórum acessível: desenvolver um fórum utilizando de métodos ágeis como *SCRUM* e eXtreme Programming (XP) seguindo as diretrizes de acessibilidade identificadas na etapa anterior. Neste trabalho o escopo foi limitado para atender às necessidades de pessoas cegas e de baixa visão.
- Documentar as possíveis dificuldades enfrentadas e fornecer recomendações: analisar e documentar as dificuldades encontradas durante o processo de desenvolvimento do fórum em relação à acessibilidade. Com base nisso, fornecer recomendações e estratégias para auxiliar os desenvolvedores na criação de aplicações web acessíveis.

1.4 Organização do Documento

Esta seção apresenta a estrutura do presente trabalho, destacando a organização dos capítulos que o compõem.

- Capítulo 1 - Introdução: O primeiro capítulo contextualiza o tema, fornecendo uma visão ampla sobre a acessibilidade digital, suas dificuldades de implementação e a relevância da engenharia de software no contexto. Além disso, expõe a justificativa para a escolha do tema e define os objetivos gerais e específicos da pesquisa.
- Capítulo 2 - Referencial Teórico: Este capítulo apresenta os fundamentos teóricos necessários para a compreensão do trabalho. Inicia-se com uma abordagem sobre acessibilidade digital, explorando suas nuances e desafios na implementação. Em seguida, destaca-se a importância da engenharia de software e introduzindo o contexto ágil.
- Capítulo 3 - Revisão Sistemática: O terceiro capítulo detalha o processo metodológico utilizado para a realização da revisão sistemática da literatura, com base na estratégia

PICOC, apresentando as perguntas de pesquisa, critérios de seleção, bases consultadas, strings de busca e principais resultados obtidos

- Capítulo 4 - Construção do Fórum: Neste capítulo é descrito o processo de desenvolvimento da aplicação acessível (*Inclusify*), abrangendo desde o levantamento de requisitos até a codificação e testes, com ênfase na aplicação das diretrizes WCAG e validações com ferramentas automatizadas e leitor de tela.
- Capítulo 5 – Conclusões: Aqui são apresentadas as conclusões obtidas a partir da implementação do trabalho, destacando as principais limitações, os resultados alcançados e recomendações para pesquisas futuras na área de acessibilidade digital em ambientes web.
- Referências: Ao final do documento, encontra-se a seção de referências bibliográficas, listando todas as fontes consultadas e citadas ao longo do trabalho.

2 REFERENCIAL TEÓRICO

2.1 Acessibilidade Digital

A acessibilidade digital é a possibilidade de que todos os indivíduos, independentemente de suas habilidades físicas, mentais ou sensoriais, possam acessar e utilizar Tecnologias da Informação e da Comunicação (TIC) (W3C, 2020). A W3C é uma organização internacional cujo objetivo é garantir a acessibilidade na web por meio de padrões e recomendações como o WCAG, que estabelece diretrizes para o desenvolvimento de conteúdo acessível na web. O WCAG é amplamente aceito como uma das principais referências para acessibilidade na web, sendo usado por desenvolvedores e *designers* para garantir que seus sites e aplicativos sejam acessíveis.

O WCAG 2.1, a versão mais recente, é baseado em quatro princípios: percepção, operabilidade, compreensão e robustez (W3C, 2018).

- O princípio de percepção se refere à capacidade de usuários de perceber o conteúdo, independentemente de suas habilidades sensoriais.
- O princípio de operabilidade se refere à facilidade de uso do conteúdo, independentemente de suas habilidades físicas ou cognitivas.
- O princípio de compreensão se refere à capacidade dos usuários de entender o conteúdo.
- E o princípio de robustez se refere à compatibilidade do conteúdo com tecnologias assistivas, como leitor de tela.

Cada princípio citado possui uma ou mais diretrizes, compondo uma lista de 13 no total. Além disso, cada diretriz é composta de um ou mais critérios de conformidade divididos em três níveis: A, AA e AAA. O nível A é o nível mínimo de conformidade, enquanto o nível AAA é o nível mais alto e exige o máximo de acessibilidade (W3C, 2018).

É importante lembrar que, embora estas diretrizes cubram uma ampla diversidade de situações, elas não são capazes de abordar as necessidades das pessoas com todos os tipos, graus e combinações de deficiências. Portanto, deve-se considerar as necessidades específicas de cada usuário e utilizar testes com usuários reais para garantir a acessibilidade (W3C, 2018).

A Tabela 1 Exemplifica as camadas que compõem o WCAG 2.1.

Tabela 1 – Diretrizes WCAG 2.1

Princípio	**Descrição das Diretrizes**	**Nº de Critérios de Sucesso**
Perceptível	1.1. Fornecer alternativas de texto para qualquer conteúdo não textual, para que possa ser convertido em outros formatos de que as pessoas necessitem, como letra ampliada, braille, fala, símbolos ou linguagem mais simples.	29
	1.2. Fornecer alternativas para mídias baseadas em tempo.	
	1.3. Criar conteúdo que possa ser apresentado de diferentes formas (por exemplo, layout mais simples) sem perder informação ou estrutura.	
	1.4. Facilitar que os usuários vejam e ouçam o conteúdo, incluindo separar corretamente primeiro plano e fundo.	
Operável	2.1. Tornar toda a funcionalidade acessível via teclado.	29
	2.2. Conceder tempo suficiente aos usuários para ler e usar o conteúdo.	
	2.3. Não projetar conteúdo de modo conhecido por causar convulsões ou reações físicas.	
	2.4. Fornecer meios para ajudar os usuários a navegar, encontrar conteúdo e determinar onde estão.	
	2.5. Facilitar a operação de funcionalidade por diversos meios de entrada além do teclado.	
Compreensível	3.1. Tornar o conteúdo textual legível e compreensível.	17
	3.2. Fazer com que páginas web apareçam e operem de maneiras previsíveis.	
	3.3. Ajudar os usuários a evitar e corrigir erros.	
Robusto	4.1. Maximizar compatibilidade com agentes de usuário atuais e futuros, incluindo tecnologias assistivas.	3

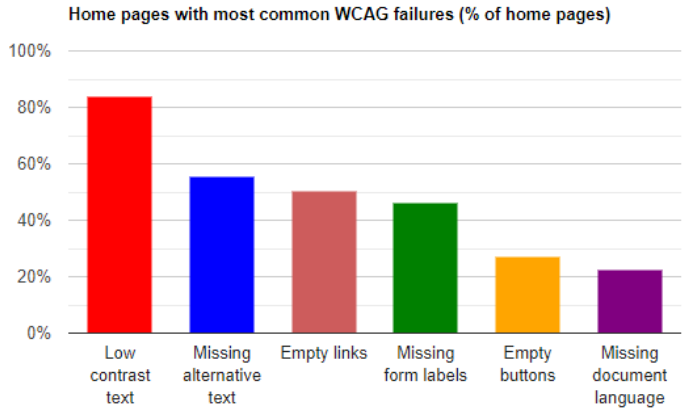
2.2 Dificuldades na implementação da Acessibilidade Digital

Apesar das recomendações apresentadas, a acessibilidade digital ainda se configura como um desafio para os desenvolvedores de software, por ser necessário garantir que as aplicações e sites sejam acessíveis para todos os usuários, incluindo aqueles com necessidades especiais. No entanto, a implementação da acessibilidade digital ainda apresenta muitos desafios.

Um dos principais desafios é a falta de conhecimento dos desenvolvedores sobre as necessidades das pessoas com deficiências e sobre as boas práticas de acessibilidade (Crabb et al., 2019). Muitos desenvolvedores e *designers* de sistemas digitais não têm conhecimento sobre as necessidades dos usuários com deficiências (Kawas; Vonessen; Ko, 2019) e, portanto, não incluem recursos de acessibilidade em seus projetos.

A pesquisa realizada pela ONG Web Accessibility In Mind (WebAIM), realizada entre 2019 e 2022, analisou a acessibilidade das principais páginas iniciais listadas na Majestic Milions, os 1.000.000 sites da Alexa e os 10 milhões de domínios do DomCop. Esses sites possuíam o maior número de acessos durante o tempo de pesquisa, sendo concluído que 96,8% desses sites apresentam falhas na acessibilidade. Os tipos mais comuns de falhas segundo a WCAG 2 podem ser vistos na Figura 1. Estas falhas foram referentes a textos de baixo contraste (*low contrast text*), falta de textos alternativos para imagens (*missing alternative text*), falta de *labels* nos *inputs* dos formulários (*missing form labels*), *links* vazios (*empty links*), idioma do documento ausente (*missing document language*) e botões vazios (*empty buttons*) (WebAIM, 2022).

Figura 1 – Páginas iniciais com falhas WCAG mais comuns (% das páginas iniciais)



Fonte: (WebAIM, 2022)

Outro desafio é a falta de ferramentas e recursos para auxiliar na implementação da acessibilidade. Segundo (Crabb et al., 2019) os desenvolvedores exigem métodos fáceis de implementar recursos de acessibilidade para aumentar a acessibilidade geral da tecnologia. Pois, como aponta Baptista et al. (2016) e Inal, Rızvanoğlu e Yesilada (2019), os padrões e diretrizes existentes sobre o tema tendem a ser muito complexos e difíceis de entender. Também, como apontam Paiva, Freire e Fortes (2021), em sua última revisão da literatura, não foi encontrada uma arquitetura de referência no domínio de acessibilidade de software que norteasse os desenvolvedores na fase de projeto do sistema. A falta de ferramentas automatizadas para verificação e correção de acessibilidade também é outro fator que pode dificultar a implementação de recursos de acessibilidade nos sistemas digitais. Além disso, poucas empresas possuem ferramentas e recursos específicos para acessibilidade.

Por fim, a falta de prioridade e investimento na acessibilidade por parte dos gestores de projetos e empresas é outro fator que dificulta a implementação da acessibilidade digital. Muitas empresas e organizações não têm orçamento para investir em recursos de acessibilidade, o que pode levar a sistemas inacessíveis. Segundo (Gonçalves et al., 2019), a mídia tem dado pouca atenção ao assunto, e fazer as alterações necessárias para tornar o software acessível a todos

ainda é muito visto mais como um custo do que como uma oportunidade de ampliar a base de clientes.

Em resumo, a implementação da acessibilidade digital ainda apresenta muitos desafios, como a falta de conhecimento dos desenvolvedores sobre as necessidades das pessoas com deficiências e boas práticas de acessibilidade, falta de ferramentas e recursos para auxiliar na implementação da acessibilidade e falta de prioridade e investimento na acessibilidade por parte dos gestores de projetos e empresas. Nesse sentido, a Engenharia de Software desempenha um papel fundamental no desenvolvimento de aplicações acessíveis, pois pode promover a integração entre metodologias e atividades e técnicas específicas de acessibilidade em um processo de desenvolvimento de software, bem como estabelecer processos que priorizem acessibilidade (Paiva; Freire; Fortes, 2021). Dessa forma, tornando sua implementação um pouco mais fácil para os desenvolvedores e barata para as empresas.

2.3 Engenharia de Software

A engenharia de software é um processo sistemático e disciplinado para o desenvolvimento, operação e manutenção de software. É uma área multidisciplinar que envolve metodologias, técnicas e ferramentas para garantir a qualidade, confiabilidade e eficiência do software.

Segundo PRESSMAN (2010), a engenharia de software é composta por cinco fases principais: análise de requisitos, projeto, construção, teste e manutenção. Cada fase é composta por atividades específicas que visam garantir a qualidade e a eficiência do software.

- A análise de requisitos é a primeira fase e é responsável por intender as necessidades dos usuários e traduzi-las em requisitos do software. É importante que essa fase seja cuidadosamente planejada e executada, pois é a base para o sucesso do projeto.
- A fase de projeto é responsável por transformar os requisitos em um modelo de software. Isso envolve a escolha de arquitetura, a definição de interfaces e a criação de diagramas de fluxo de dados.
- A fase de construção é responsável por implementar o projeto em código. Isso envolve a escrita de código, teste de unidade e integração.
- A fase de teste é responsável por validar se o software atende aos requisitos e se está livre de erros. Isso envolve testes de unidade, testes de sistema e testes de aceitação.
- Por fim, a fase de manutenção é responsável por garantir que o software continue funcionando corretamente e atendendo às necessidades dos usuários. Isso envolve correção de erros, atualizações e melhorias.

É importante destacar que a engenharia de software é uma área em constante evolução e que existem muitas metodologias e técnicas diferentes que podem ser utilizadas para percorrer suas fases. Uma das metodologias mais populares são os métodos ágeis.

2.4 Manifesto Ágil e Métodos Ágeis.

O Manifesto Ágil é um conjunto de valores e princípios, que foram estabelecidos em 2001 para guiar o desenvolvimento de software de forma ágil. Ele foi criado por um grupo de profissionais de desenvolvimento de software que se reuniram para discutir as melhores práticas para o desenvolvimento de software.

Os quatro valores do Manifesto Ágil são: indivíduos e interações, software funcionando, colaboração com o cliente e resposta a mudanças (Beck et al., 2001a). Esses valores são considerados os pilares fundamentais para o desenvolvimento de software de forma ágil. Conforme o Manifesto Ágil, os 12 (doze) princípios são (Beck et al., 2001b):

1. Satisfação do cliente por meio da entrega contínua e rápida de software funcionando;
2. Acolhimento de mudanças de requisitos, mesmo tardiamente no desenvolvimento;
3. Entregas frequentes de trabalho completado;
4. Colaboração diária entre desenvolvedores e clientes;
5. Projetos construídos em torno de indivíduos motivados, dando-lhes o ambiente e suporte necessários;
6. Sempre transmitir informações para a equipe de desenvolvimento em conversas face a face;
7. O software funcionando é a principal medida de progresso;
8. Processos ágeis promovem o desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente;
9. Atenção contínua à excelência técnica e boas práticas de engenharia;
10. Simplicidade;
11. *Self-organization* e autogerenciamento;
12. Revisões regulares e retrospectivas;

Os métodos ágeis são uma abordagem para o gerenciamento de projetos de software que se concentra na entrega rápida, na adaptabilidade e na colaboração contínua entre desenvolvedores e usuários. Essa abordagem foi desenvolvida como uma alternativa aos métodos tradicionais

de gerenciamento de projetos, como o desenvolvimento de ciclo de vida único e o modelo de cascata, que eram considerados burocráticos e pouco flexíveis. De acordo com Schwaber (2004), Highsmith (2013), os métodos ágeis, baseados no Manifesto Ágil, são considerados mais eficazes do que os métodos tradicionais de desenvolvimento de software para garantir a qualidade, a flexibilidade e a satisfação do cliente. Além disso, os métodos ágeis também são mais eficazes para gerenciar projetos com requisitos incertos ou em constante mudança. Entre os métodos ágeis mais conhecidos estão o *Scrum*, o *Kanban* e o *Extreme Programming* (XP).

2.4.1 *Scrum*

O *Scrum* é uma metodologia ágil para gerenciamento de projetos de desenvolvimento de software introduzida pela primeira vez por Jeff Sutherland em 1995. Ele foi formalizado e descrito detalhadamente em seu livro *"Scrum: The Art of Doing Twice the Work in Half the Time"* (Sutherland; Sutherland, 2014). *Scrum* é baseado em princípios ágeis e valoriza indivíduos e interações, trabalho em equipe, colaboração e capacidade de adaptação a mudanças.

Uma das principais características do *Scrum* é a utilização de ciclos curtos de trabalho, chamados de *Sprints*, que geralmente duram entre 1 e 4 semanas. Durante cada *Sprint*, uma equipe de desenvolvimento trabalha em um conjunto definido de tarefas, chamadas de itens do *Product Backlog*, com o objetivo de entregar um incremento funcional do produto final.

Outra característica importante do *Scrum* é a utilização de papéis específicos, como o *Scrum Master*, o *Product Owner* e os membros do time de desenvolvimento. O *Scrum Master* é responsável por garantir que a metodologia *Scrum* seja seguida, enquanto o *Product Owner* é responsável por representar os interesses do cliente e garantir que o trabalho seja realizado consoante as prioridades do negócio.

2.4.2 *Kanban*

A metodologia *Kanban* se baseia em três princípios básicos: visualizar o fluxo de trabalho, limitar o trabalho em progresso e estabelecer políticas de fluxo. A visualização do fluxo de trabalho é feita por meio da utilização de um quadro *Kanban*, onde as tarefas são representadas por cartões e movem-se através das diferentes etapas do processo. Limitar o trabalho em progresso permite que a equipe se concentre em tarefas específicas e evita o sobrecarregamento de trabalho. Estabelecer políticas de fluxo é importante para garantir que as tarefas sejam entregues consistentemente e no prazo estabelecido.

2.4.3 *Extreme Programming*

A *Extreme Programming* (XP) é uma metodologia ágil para gerenciamento de projetos de desenvolvimento de software introduzida pela primeira vez por Kent Beck em 1999. Ele descreveu detalhadamente a metodologia em seu livro *"Extreme Programming Explained: Em-*

brace Change" (Beck, 2000). A XP se baseia em 12 princípios que visam aumentar a qualidade do software, aumentar a comunicação e a colaboração entre os membros da equipe e tornar o processo de desenvolvimento mais ágil.

Uma das principais características da XP é o uso de ciclos curtos de desenvolvimento, chamados de *Sprints*, que geralmente duram entre 1 e 4 semanas. Durante cada *Sprint*, a equipe de desenvolvimento trabalha em um conjunto definido de tarefas, chamadas de histórias do usuário, visando entregar um incremento funcional do produto final.

Outra característica importante da XP é a utilização de práticas específicas, como programação em par, testes automatizados, refatoração e garantia da qualidade. A programação em par é um processo de desenvolvimento em que dois programadores trabalham juntos em um computador, enquanto os testes automatizados são utilizados para garantir que o software esteja livre de falhas. A refatoração é um processo de melhoria contínua do código, enquanto a garantia da qualidade é um processo para garantir que o software atenda aos requisitos e esteja livre de falhas.

2.5 Desvantagens da cultura ágil para a implementação de acessibilidade

A cultura ágil tem se mostrado eficaz em muitos aspectos do gerenciamento de projetos de desenvolvimento de software, mas também pode ter algumas desvantagens quando se trata de implementação de acessibilidade. Por conta da falta de conhecimento e conscientização de boa parte dos desenvolvedores de software eme relação à acessibilidade, vemos que o Manifesto Ágil não é lido de forma inclusiva (Pellegrini et al., 2019). Algumas práticas podem facilmente contribuir para que requisitos de acessibilidade sejam deixados de lado durante o projeto. Uma delas é o MVP (*Minimum Viable Product*).

O MVP é uma abordagem popular para o desenvolvimento ágil, que visa lançar uma versão básica do produto o mais rapidamente possível e, em seguida, melhorá-lo com base nas respostas do usuário. No entanto, a implementação da acessibilidade em um projeto MVP pode apresentar algumas desvantagens.

Uma dessas desvantagens é que a acessibilidade pode ser vista como um "requisito adicional"e não como uma parte integrante do desenvolvimento do produto. Isso pode levar a uma implementação tardia da acessibilidade, ou mesmo a sua ausência no MVP (Pellegrini et al., 2019). Pellegrini et al. (2019) acrescenta ainda que:

> A acessibilidade deve ser priorizada desde o início dos projetos de software, nos primeiros sprints, versões e releases. Se for adiada para o final do projeto, sua implantação ficará mais cara, sendo muito difícil mudar o que já foi pensado, implantado, testado e já está em funcionamento (Pellegrini et al., 2019).

3 REVISÃO SISTEMÁTICA

Neste capítulo, será descrito o processo de revisão sistemática de literatura, estruturado a partir da estratégia PICO (Population, Intervention, Comparison, Outcome). Adotamos as diretrizes gerais de protocolos consolidadas em Engenharia de Software, em especial Kitchenham e Charters (2007), de modo a garantir que cada etapa, da formulação das perguntas de pesquisa até a interpretação dos achados, esteja alinhada aos objetivos do estudo. Como apoio para a seleção de estudos, utilizou-se a ferramenta Parsifal [1].

No nosso caso, a população corresponde a desenvolvedores de software web; a intervenção, às práticas e ferramentas de acessibilidade (WCAG); a comparação, às diferentes abordagens e desafios reportados; o desfecho, às evidências de eficácia e obstáculos enfrentados; e o contexto, aplicações web para pessoas cegas ou com baixa visão.

3.1 Planejamento

O planejamento da revisão sistemática foi conduzido em quatro etapas interdependentes, garantindo que a estratégia de busca e seleção dos estudos fosse rigorosa e transparente.

3.1.1 Definição das Perguntas de Pesquisa (RQs)

Primeiramente, traduziu-se o escopo do PICOC em questões de pesquisa claras e mensuráveis. A RQ1 investiga “quais métodos, técnicas e processos são aplicados para tornar aplicações web conformes às WCAG?”, visando mapear o estado da arte em *frameworks* de teste, *linters* e recursos de automação. Já a RQ2 busca compreender “quais desafios e barreiras os desenvolvedores enfrentam na adoção dessas práticas?”, permitindo capturar tanto lacunas tecnológicas quanto aspectos humanos (conhecimento, cultura organizacional, suporte às ferramentas).

3.1.2 Elaboração dos critérios de inclusão e exclusão

Em seguida, estabeleceram-se critérios rígidos para garantir que apenas estudos relevantes e de qualidade integrassem a revisão, como mostra a Tabela 2. Os critérios de inclusão priorizaram trabalhos publicados entre 2018 e 2024, em inglês ou português, que descrevessem explicitamente a aplicação de diretrizes WCAG ou relatassem estudos empíricos sobre dificuldades de implementação. Os critérios de exclusão eliminaram duplicatas, resumos sem texto

[1] <https://parsif.al/>

completo disponível e artigos cujo foco principal não fosse acessibilidade web (por exemplo, acessibilidade em dispositivos móveis ou hardware específico).

Tabela 2 – Critérios de Inclusão e Exclusão

Critérios de Inclusão	**Critérios de Exclusão**
• Trabalhos publicados entre 2018 e 2024; • Idioma: inglês ou português; • Descrição explícita da aplicação de diretrizes WCAG; • Estudos empíricos sobre dificuldades de implementação.	• Duplicatas; • Resumos sem texto completo disponível; • Foco diferente de acessibilidade web (e.g., dispositivos móveis, hardware).

3.1.3 Seleção de bases de dados

Para maximizar a cobertura da literatura relevante, optou-se por cinco grandes repositórios acadêmicos:

- ACM Digital Library (https://dl.acm.org/)
- IEEE (http://ieeexplore.ieee.org.)
- ScienceDirect (https://www.sciencedirect.com/)
- Springer Link (https://link.springer.com/)
- DBLP (https://dblp.org)

3.1.4 Construção das *Strings* de Busca

A formulação da estratégia de busca seguiu o *framework* PICOC, conforme apresentado no Quadro 1. Para isso, agrupamos três blocos de termos:

- **População**: software developers, web developers;
- **Intervenção**: Web accessibility, WCAG compliance;
- **Desfecho**: barriers, challenges, difficulties, obstacles, issues, hurdles.

A *string* de busca resultante, que foi utilizada como entrada nos mecanismos de busca das bases de dados, foi:

```
("software developers"OR "web developers")
AND ("WCAG compliance"OR "Web accessibility methods")
AND ("barriers"OR "challenges"OR "difficulties"OR "hurdles"
OR "issues"OR "obstacles")
```

Os operadores booleanos OR (dentro de cada bloco) e AND (entre blocos) garantem tanto a abrangência quanto a precisão da busca. Em bases que limitam o uso de aspas ou parênteses, introduziram-se pequenas adaptações (por exemplo, remoção de aspas ou uso de colchetes) sem comprometer o sentido original. Esse cuidado equilibra o retorno de estudos relevantes e evita um volume excessivo de registros para triagem.

Quadro 1: Palavras-Chave e Sinônimos

Keyword	Synonyms	PICOC
software developers		Population
web developers		Population
Web accessibility		Intervention
WCAG compliance		Intervention
challenges		Outcome
difficulties	barriers, obstacles, issues, hurdles	Outcome

3.2 Condução

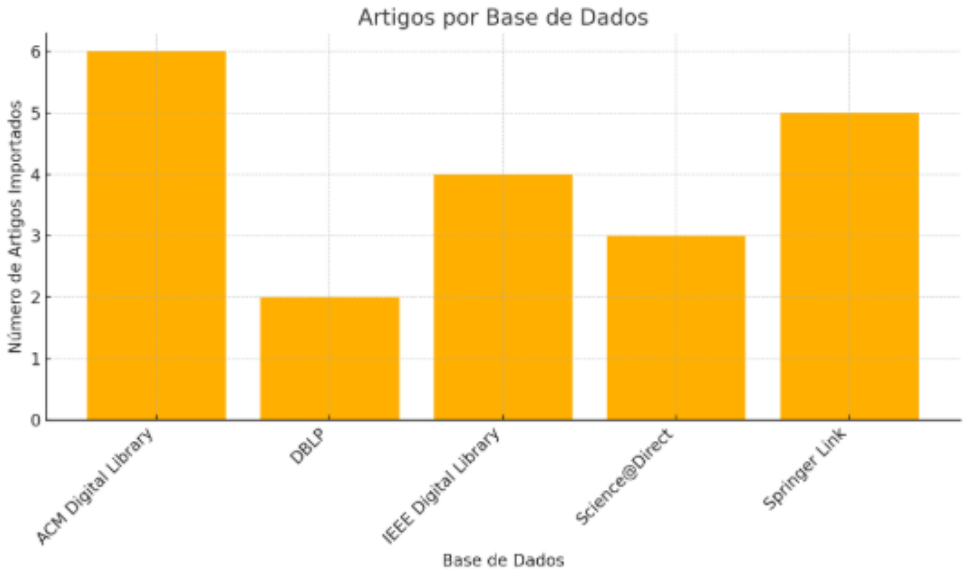
A fase de condução da revisão sistemática envolveu a operacionalização das etapas planejadas, garantindo rigor e rastreabilidade em cada passo. Inicialmente, executou-se a busca nas cinco bases selecionadas, aplicando de forma uniforme as strings construídas.

Em seguida, procedeu-se à importação dos estudos para um gerenciador de referências. Foram coletados 20 registros ao todo, distribuídos conforme segue: ACM Digital Library (6), DBLP (2), IEEE Xplore (4), ScienceDirect (3) e Springer Link (5). Cada registro foi exportado no formato BibTeX, garantindo compatibilidade com a ferramenta de triagem. Além do título e autores, armazenamos campos como resumo, ano de publicação e palavras-chave, o que facilitará a fase subsequente de seleção e extração de dados.

O passo de seleção dos estudos dividiu-se em duas subfases. Primeiramente, eliminaram-se duplicatas e artigos redundantes. Depois, cada título e resumo foi avaliado à luz dos critérios definidos: se havia menção clara a práticas WCAG e desafios de implementação, textos em inglês ou português, e publicação nos últimos seis anos.

Por fim, na extração de dados aplicou-se um formulário padronizado construído previamente em planilha compartilhada. Para cada estudo selecionado, registrou-se informações sobre objetivo, metodologia, principais artefatos (ferramentas, *frameworks* ou *checklists* testados), métricas de avaliação utilizadas, limitações apontadas pelos autores e conclusões principais.

Figura 2 – Artigos coletados por base de dados.



3.3 Resultados

A fase de resultados apresenta os principais achados extraídos dos artigos e, por fim, a resposta articulada às perguntas de pesquisa formuladas.

3.3.1 Descrição dos achados

A análise detalhada dos artigos importados revelou um conjunto de métodos e ferramentas recorrentes. Testes automatizados de conformidade WCAG, utilizando *frameworks* como axe-core e Pa11y, aparecem em mais de metade dos estudos selecionados; esses trabalhos demonstram que a automatização pode reduzir o esforço manual de verificação, mas esbarram em limitações na detecção de problemas de semântica visual. Linters e plugins para IDEs, capazes de apontar violações em tempo real durante a codificação, mostraram-se promissores para incorporar acessibilidade ao fluxo contínuo de desenvolvimento, embora ainda careçam de integração nativa em ferramentas populares. Já os estudos qualitativos, que envolveram entrevistas e grupos focais com desenvolvedores, destacam a frequência de obstáculos como entendimento superficial das WCAG, resistência a mudanças no pipeline de Continuous Integration/Continuous Deployment (CI/CD) e carência de documentação padronizada.

Além disso, identificou-se que poucos trabalhos abordam métricas quantitativas robustas para avaliar ganhos de acessibilidade após a aplicação de práticas recomendadas. Em geral, os autores recorrem à contagem de violações antes e depois da intervenção, mas raramente medem o impacto na experiência de usuários com deficiência visual. Isso sugere uma lacuna metodológica: a eficácia das ferramentas automatizadas é bem documentada, mas seu efeito real na usabilidade permanece subexplorado.

3.3.2 Resposta às questões de pesquisa

- Em relação à RQ1, “Quais métodos, ferramentas e processos suportam a conformidade com WCAG em aplicações web?”, a revisão evidencia que o uso de ferramentas automatizadas, a integração de *linters* e *checklists* no CI/CD, e o emprego correto de atributos Accessible Rich Internet Applications (ARIA) são as práticas mais relatadas como eficazes para garantir conformidade com WCAG. Estudos como o de Chiou, Alotaibi e Halfond (2023) e de Tafreshipour et al. (2024) corroboram a redução significativa de esforço manual e o aumento de cobertura de testes automatizados .
- Para a RQ2, “Quais desafios os desenvolvedores encontram na adoção dessas práticas?”, destaca que os principais obstáculos concentram-se na falta de habilidades práticas em testes assistivos, na ausência de requisitos bem definidos e no suporte desigual de ambientes de execução. Essa constatação é reforçada tanto pelos relatos de universidades no estudo de Oswal e Palmer (2024) quanto pelas avaliações de governo local sobre uso de Accessible Rich Internet Applications (ARIA) em Król e Zdonek (2020) .

Em suma, a síntese dos resultados aponta para a eficácia de abordagens automatizadas e integradas, mas destaca a necessidade de formação específica e de processos mais rigorosos de levantamento de requisitos para elevar a qualidade da acessibilidade web de forma sustentável.

4 CONSTRUÇÃO DO FÓRUM *INCLUSIFY*

Neste capítulo, apresenta-se o processo de construção do protótipo de fórum acessível, desde a definição dos artefatos de projeto até a implementação e validação das soluções. Partindo das diretrizes WCAG 2.1 e dos requisitos identificados, serão descritas as quatro fases do desenvolvimento de software: coleta de requisitos, projeto, codificação e testes. Para cada etapa, busca-se explicitar como a acessibilidade para pessoas cegas e de baixa visão foi priorizada, bem como documentar as principais dificuldades encontradas e as estratégias adotadas.

Como prova de conceito para o estudo realizado, escolhemos desenvolver um sistema web simples de fórum denominado *Inclusify*. O fórum reúne funcionalidades típicas de fóruns, como criação de tópicos, postagens, comentários e votação, mas prioriza a acessibilidade para pessoas cegas e com baixa visão por meio de HTML semântico, rotulagem adequada de elementos, navegação completa por teclado e compatibilidade com leitores de tela. Dessa forma, o *Inclusify* serve não apenas como demonstração técnica, mas também como um ambiente de experimentação para identificar e documentar barreiras de acessibilidade e recomendações durante todo o processo de codificação.

4.1 Levantamento de Requisitos

Para guiar a construção de um fórum acessível, começamos pelo levantamento de requisitos. Inspirados em casos de uso consolidados (Reddit, StackOverflow, DEV Community) e amparados pelas diretrizes WCAG 2.1 (W3C, 2018), mapeamos tanto funcionalidades essenciais de fórum (tópicos, postagens, comentários, votos) quanto exigências de usabilidade para pessoas cegas e de baixa visão (leitores de tela, navegação por teclado, contraste e *labels* semânticos). Esses requisitos formam a base para todos os artefatos seguintes.

Com base nessas referências e nas diretrizes WCAG 2.1, os principais requisitos foram divididos em:

- Requisitos Funcionais

    1. Navegar por postagens populares (mais votos ou visualizações).
    2. Autenticar-se e gerenciar conta (login, criação de conta, recuperação de senha).
    3. Filtrar e pesquisar por categorias, palavras-chave e ordem cronológica.
    4. Permitir criação, edição e exclusão de postagens e comentários (para usuários autenticados).

5. Votar em postagens e comentários.
6. Visualizar e editar perfil pessoal (nome de exibição, foto).

- Requisitos de Administração

1. Remover conteúdo que viole as regras da comunidade.
2. Banir usuários por comportamento inadequado.

- Requisitos Não Funcionais de Acessibilidade

1. Compatibilidade total com leitores de tela com o NonVisual Desktop Access (NVDA).
2. Navegação completa por teclado (foco visível em todos os controles).
3. Rotulagem semântica de elementos (uso apropriado de <header>, <nav>, <main>, <button>, etc.).
4. Contraste mínimo de 4,5:1 em textos e 3:1 em elementos de interface.
5. *Feedback* sonoro e visual para mensagens de erro e sucesso.
6. Textos alternativos descritivos em imagens e ícones.

Esse levantamento definiu o escopo do desenvolvimento, alinhando funcionalidades essenciais de fórum com práticas de acessibilidade que serão detalhadas no decorrer do capítulo.

4.2 Projeto

A partir dos requisitos elencados na etapa anterior, na fase de projeto buscamos transformar esses requisitos em modelos conceituais e protótipos capazes de orientar toda a implementação, garantindo que as necessidades de acessibilidade para pessoas cegas e de baixa visão sejam plenamente atendidas.Essa etapa englobou:

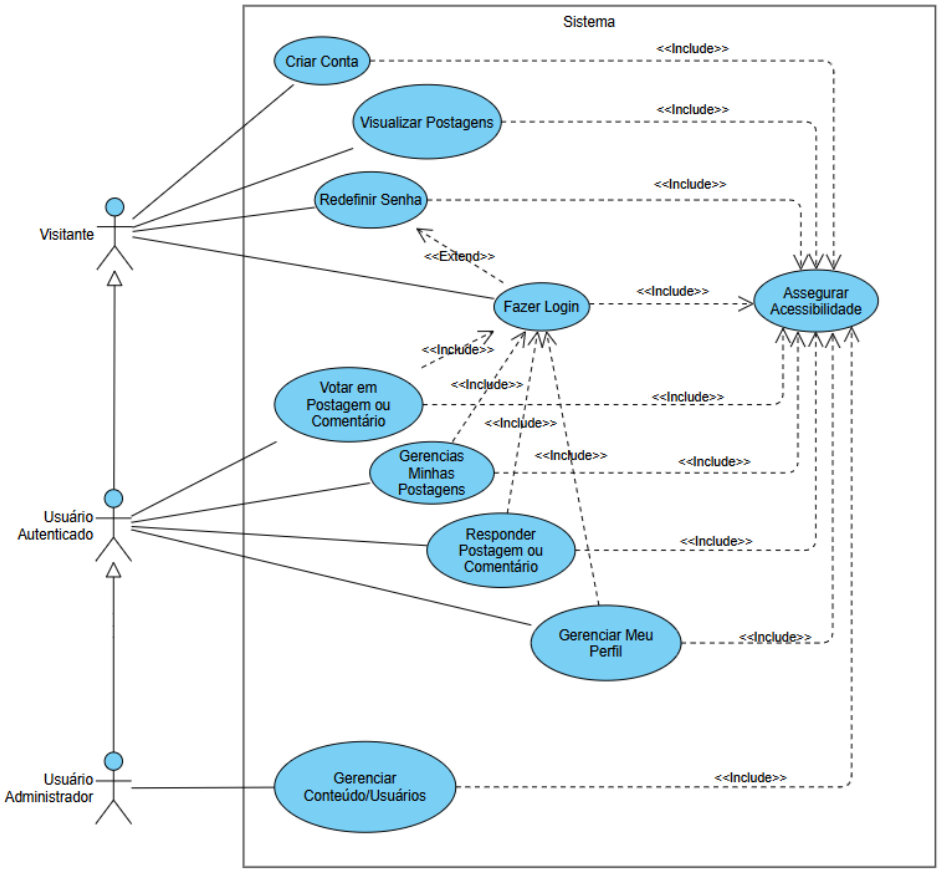
- Casos de Uso Cenários que descrevem as interações dos diferentes perfis de usuário (visitante, autenticado e administrador) com o fórum.
- Diagrama Entidade–Relacionamento Estruturação das entidades de domínio e seus relacionamentos, garantindo a coerência dos dados e suportando as operações de navegação assistiva.
- Dicionário de Dados Detalhamento dos atributos de cada entidade, tipos e restrições.
- Protótipo de Telas Modelagem das interfaces de usuário no Figma, considerando navegação por teclado, leitores de tela e contraste de cores.

Cada um desses artefatos visa assegurar que, antes de iniciar a codificação, exista um entendimento claro das funcionalidades e dos aspectos de usabilidade e acessibilidade que serão implementados.

4.2.1 Casos de Uso

Nesta subseção, apresentam-se os principais casos de uso do fórum acessível, organizados por perfil de ator, como podemos ver na Figura 3. Cada caso de uso descreve a funcionalidade esperada, focando na experiência de navegação por teclado e leitura por leitores de tela.

Figura 3 – Casos de Uso



Fonte: Autoria própria

1. **Criar Conta**
    - **Atores Primários:** Visitante
    - **Objetivo:** Permitir que um visitante se registre no sistema, criando um novo perfil de usuário.

- **Pré-condições:**
  - O visitante não deve estar autenticado.
- **Fluxo Básico:**
  a) Visitante seleciona “Cadastrar-se”.
  b) Sistema exibe formulário de registro.
  c) Visitante preenche e submete o formulário.
  d) Sistema valida dados, gera nova conta e confirma criação.
  e) Sistema executa o caso de uso *Assegurar Acessibilidade*.
- **Fluxos Alternativos:**
  - Se os dados forem inválidos, exibe mensagem de erro (feedback sonoro e visual) e retorna ao passo 2.
- **Relações:** `<<include>>` Assegurar Acessibilidade

2. **Fazer Login**

- **Atores Primários:** Visitante
- **Objetivo:** Autenticar um usuário pré-cadastrado no sistema.
- **Pré-condições:**
  - Visitante deve ter uma conta existente.
- **Fluxo Básico:**
  a) Visitante clica em “Login”.
  b) Sistema exibe tela de entrada.
  c) Visitante informa credenciais e submete.
  d) Sistema valida credenciais:
    - Se válidas, concede acesso e muda ator para Usuário Autenticado.
    - Se inválidas, oferece opção “Esqueci minha senha”.
  e) Sistema executa *Assegurar Acessibilidade*.
- **Fluxos Alternativos:**
  - Visitante seleciona “Esqueci minha senha”: desvia para *Redefinir Senha*.
- **Relações:**
  - `<<include>>` Assegurar Acessibilidade
  - `<<extend>>` Redefinir Senha

3. **Redefinir Senha**

- **Atores Primários:** Visitante

- **Objetivo:** Permitir que o usuário recupere o acesso alterando sua senha.
- **Pré-condições:**
  - Deve ter conta cadastrada com e-mail válido.
- **Fluxo Básico:**
  a) Visitante clica em "Esqueci minha senha" na tela de login.
  b) Sistema solicita o e-mail cadastrado.
  c) Visitante informa e-mail; sistema envia link de redefinição.
  d) Visitante segue link, escolhe nova senha e confirma.
  e) Sistema atualiza senha e exibe confirmação (feedback acessível).
- **Fluxos Alternativos:**
  - E-mail não cadastrado: exibir mensagem de erro.
- **Relações:**
  - `<<extend>>` Fazer Login
  - `<<include>>` Assegurar Acessibilidade

4. **Visualizar Postagens**

- **Atores Primários:** Visitante, Usuário Autenticado
- **Objetivo:** Navegar e ler posts disponíveis no fórum.
- **Pré-condições:**
  - O sistema deve conter pelo menos uma postagem.
- **Fluxo Básico:**
  a) Ator acessa a área de "Postagens Recentes" ou "Postagens Relevantes".
  b) Sistema exibe lista paginada de posts.
  c) Ator aplica filtros ou pesquisa (categoria, palavra-chave).
  d) Ator seleciona um post para ler detalhes.
  e) Sistema apresenta conteúdo completo.
  f) Invoca *Assegurar Acessibilidade*.
- **Relações:** `<<include>>` Assegurar Acessibilidade

5. **Votar em Conteúdo**

- **Atores Primários:** Usuário Autenticado
- **Objetivo:** Dar upvote/downvote em posts ou comentários.
- **Pré-condições:**
  - Ator deve estar autenticado.

- **Fluxo Básico:**
  a) Usuário seleciona post ou comentário.
  b) Clica em “positivo” ou “negativo”.
  c) Sistema registra voto e atualiza visualmente e por leitor de tela.
  d) Invoca Assegurar Acessibilidade.
- **Fluxos Alternativos:**
  – Usuário já votou: informa que só é permitido um voto por conteúdo.
- **Relações:**
  – `<<include>>` Fazer Login
  – `<<include>>` Assegurar Acessibilidade

6. **Gerenciar Minhas Postagens**

- **Atores Primários:** Usuário Autenticado
- **Objetivo:** Criar, editar e excluir as próprias postagens.
- **Pré-condições:**
  – Usuário deve estar autenticado.
- **Fluxo Básico:**
  a) Usuário abre “Meu Perfil”.
  b) Sistema exibe pagina de perfil do usuário com lista de postagens.
  c) Usuário escolhe “Editar” ou “Excluir”.
  d) Editar: formulário pré-preenchido, submete alterações.
  e) Excluir: pede confirmação e remove.
  f) Invoca Assegurar Acessibilidade.
- **Fluxos Alternativos:**
  – Dados inválidos na edição: exibe erro e retorna ao formulário.
- **Relações:**
  – `<<include>>` Fazer Login
  – `<<include>>` Assegurar Acessibilidade

7. **Responder Conteúdo**

- **Atores Primários:** Usuário Autenticado
- **Objetivo:** Inserir novos comentários em posts ou respostas a comentários.
- **Pré-condições:**
  – Ator deve estar logado.

- **Fluxo Básico:**
  a) Usuário clica em “Comentar”.
  b) Sistema exibe área de texto; usuário digita e envia.
  c) Sistema adiciona comentário e atualiza a lista.
  d) Invoca Assegurar Acessibilidade.
- **Fluxos Alternativos:**
  – Comentário vazio: não permite envio e exibe aviso.
- **Relações:**
  – `<<include>>` Fazer Login
  – `<<include>>` Assegurar Acessibilidade

8. **Gerenciar Meu Perfil**

- **Atores Primários:** Usuário Autenticado
- **Objetivo:** Visualizar e editar dados da própria conta.
- **Pré-condições:**
  – Ator deve estar autenticado.
- **Fluxo Básico:**
  a) Usuário abre “Meu Perfil”.
  b) Sistema exibe informações.
  c) Usuário clica em “Editar”, modifica e salva.
  d) Sistema valida e confirma.
  e) Invoca Assegurar Acessibilidade.
- **Fluxos Alternativos:**
  – E-mail já em uso: exibe erro e permite correção.
- **Relações:**
  – `<<include>>` Fazer Login
  – `<<include>>` Assegurar Acessibilidade

9. **Gerenciar Conteúdo/Usuários**

- **Atores Primários:** Usuário Administrador
- **Objetivo:** Executar moderação (remover postagens/comentários, banir usuários).
- **Pré-condições:**
  – Ator deve ser administrador.
- **Fluxo Básico:**

a) Administrador acessa painel de moderação.
b) Escolhe “Remover Postagem/Comentário” ou “Banir Usuário”.
c) Sistema pede confirmação e executa ação.
d) Exibe mensagem de sucesso (*feedback* acessível).
e) Invoca *Assegurar Acessibilidade*.

- **Fluxos Alternativos:**
  - Tentativa de banir outro administrador: nega operação.
- **Relações:**
  - `<<include>>` Fazer Login (por generalização)
  - `<<include>>` Assegurar Acessibilidade

Esses casos de uso não só orientam o que o sistema deve fazer, mas também “como” cada ação deve ser percebida e executada por tecnologias assistivas, servindo de *checklist* direto durante a implementação.

4.2.2 Diagrama Entidade–Relacionamento

Com base nos requisitos, criamos um diagrama Entidade–Relacionamento (ER) apresentando de forma visual as principais entidades do domínio do fórum acessível e seus relacionamentos, servindo de base para a modelagem do banco de dados. Conforme ilustrado na Figura 4 as entidades centrais são:

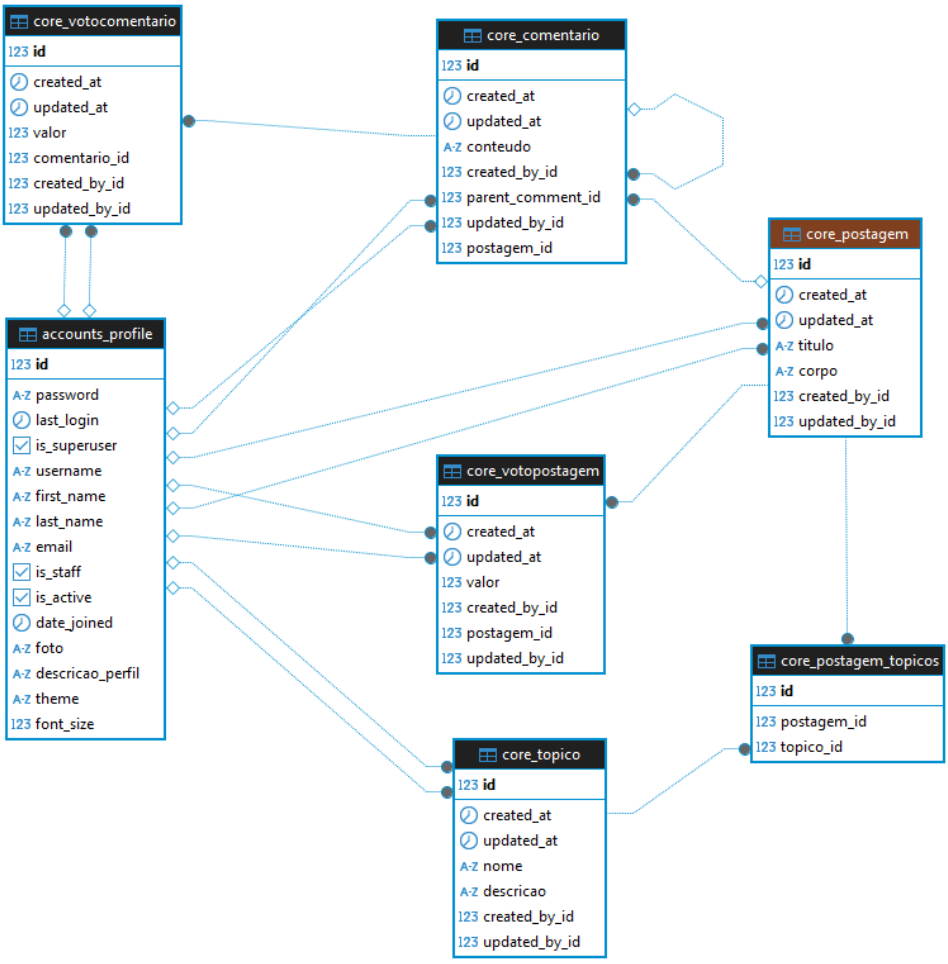
- Tópico: agrupa postagens por assunto.
- Postagem: contém o título, corpo do texto e metadados da discussão.
- Comentário: registra as respostas de usuários a uma determinada postagem.
- VotoPostagem: armazena os votos (positivo ou negativo) dados a cada postagem.
- VotoComentário: armazena os votos dados a cada comentário.
- Profile: representa o usuário do fórum, com dados de autenticação e personalização.

As relações são:

- Um Tópico pode possui muitas Postagens e uma Postagem pode possui muitos Tópicos.
- Uma Postagem é criada por um Profile, porém o Profile pode criar várias Postagens.
- Uma Postagem pode possui muitos Comentários.

- Um Comentário pertence a uma única Postagem e a um Profile.
- Um Profile pode votar em muitas Postagens e em muitos Comentários.
- Um voto aponta para apenas uma Postagem ou Comentário.

Figura 4 – Diagrama ER das principais tabelas do *Inclusify*.



Fonte: Autoria própria

4.2.3 Dicionário de Dados

A seguir, o dicionário de dados descreve de forma detalhada cada entidade com suas tabelas, atributos, tipos e restrições conforme Quadro 2, Quadro 3, Quadro 4, Quadro 5, Quadro 6 e Quadro 7. Criando um roteiro claro para a implementação.

Essa definição de dicionário de dados alinha-se ao escopo do fórum acessível, garantindo que cada atributo necessário para funcionalidade e para suportar requisitos de acessibilidade esteja devidamente especificado.

Quadro 2: Dicionário de Dados – Topico

Campo	**Tipo**	**Null**	**Blank**
id	AutoField (PK)	—	—
nome	CharField(100)	Não	Não
descricao	TextField(250)	Não	Não
created_at	DateTimeField	Não	Não
updated_at	DateTimeField	Não	Não
created_by	ForeignKey	Sim	Sim
updated_by	ForeignKey	Sim	Sim

Fonte: Autoria própria

Quadro 3: Dicionário de Dados – Postagem

Campo	**Tipo**	**Null**	**Blank**
id	AutoField (PK)	—	—
titulo	CharField(200)	Não	Não
corpo	TextField	Não	Não
topicos	ManyToManyField	—	Sim
created_at	DateTimeField	Não	Não
updated_at	DateTimeField	Não	Não
created_by	ForeignKey	Sim	Sim
updated_by	ForeignKey	Sim	Sim

Fonte: Autoria própria

Quadro 4: Dicionário de Dados – Comentario

Campo	**Tipo**	**Null**	**Blank**
id	AutoField (PK)	—	—
conteudo	TextField	Não	Não
postagem	ForeignKey	Sim	Sim
parent_comment	ForeignKey	Sim	Sim
created_at	DateTimeField	Não	Não
updated_at	DateTimeField	Não	Não
created_by	ForeignKey	Sim	Sim
updated_by	ForeignKey	Sim	Sim

Fonte: Autoria própria

4.2.4 Protótipo de Telas

Considerando os casos de uso, diagrama ER e dicionários de dados produzidos até aqui, elaboramos o protótipo do fórum usando a ferramenta Figma. Cada mockup foi construído pensando na semântica HTML e nos atributos ARIA necessários para navegação por teclado e compatibilidade com leitores de tela. Abaixo são apresentadas as principais telas e componentes do protótipo:

- Iniciando pela barra de navegação para visitantes, onde é possível acessar as postagens

Quadro 5: Dicionário de Dados – VotoPostagem

Campo	**Tipo**	**Null**	**Blank**
id	AutoField (PK)	—	—
valor	SmallIntegerField	Não	Não
postagem	ForeignKey	Não	Não
created_at	DateTimeField	Não	Não
updated_at	DateTimeField	Não	Não
created_by	ForeignKey	Sim	Sim
updated_by	ForeignKey	Sim	Sim

Fonte: Autoria própria

Quadro 6: Dicionário de Dados – VotoComentario

Campo	**Tipo**	**Null**	**Blank**
id	AutoField (PK)	—	—
valor	SmallIntegerField	Não	Não
comentario	ForeignKey	Não	Não
created_at	DateTimeField	Não	Não
updated_at	DateTimeField	Não	Não
created_by	ForeignKey	Sim	Sim
updated_by	ForeignKey	Sim	Sim

Fonte: autoria própria

Quadro 7: Dicionário de Dados – Profile

Campo	**Tipo**	**Null**	**Blank**
id	AutoField (PK)	—	—
username	CharField(150)	Não	Não
first_name	CharField(150)	Não	Sim
last_name	CharField(150)	Não	Sim
email	EmailField	Não	Sim
is_staff	BooleanField	Não	Não
is_active	BooleanField	Não	Não
date_joined	DateTimeField	Não	Não
foto	ImageField	Sim	Sim
descricao_perfil	TextField	Sim	Sim
theme	CharField(50)	Não	Não
font_size	IntegerField	Não	Não

Fonte: Autoria própria

mais relevantes com base em data de postagem e quantidades, votos e comentários. Há links dedicados para “Postagens Recentes” e “Tópicos”, além de um campo de busca, facilitando a localização de conteúdos específicos. O visitante encontra também os links “Login” e “Cadastrar-se”, Caso não possua uma conta. Veja a estrutura na Figura 5

- Na barra para usuários autenticados, além dos links de navegação encontrado na barra

Figura 5 – Barra de navegação para usuário visitante.

Fonte: Autoria própria

de navegação do visitante, são exibidos, em um menu dropdown, atalhos para “Nova Postagem” e “Meu Perfil” bem como a opção de ”Sair”conforme ilustrado na Figura 6.

Figura 6 – Barra de navegação para usuário autenticado.

Fonte: Autoria própria

- Na tela de listagens podemos encontrar postagens ranqueadas por relevância ou data de postagem bem como botões de navegação entre as páginas conforme ilustrado na Figura 7

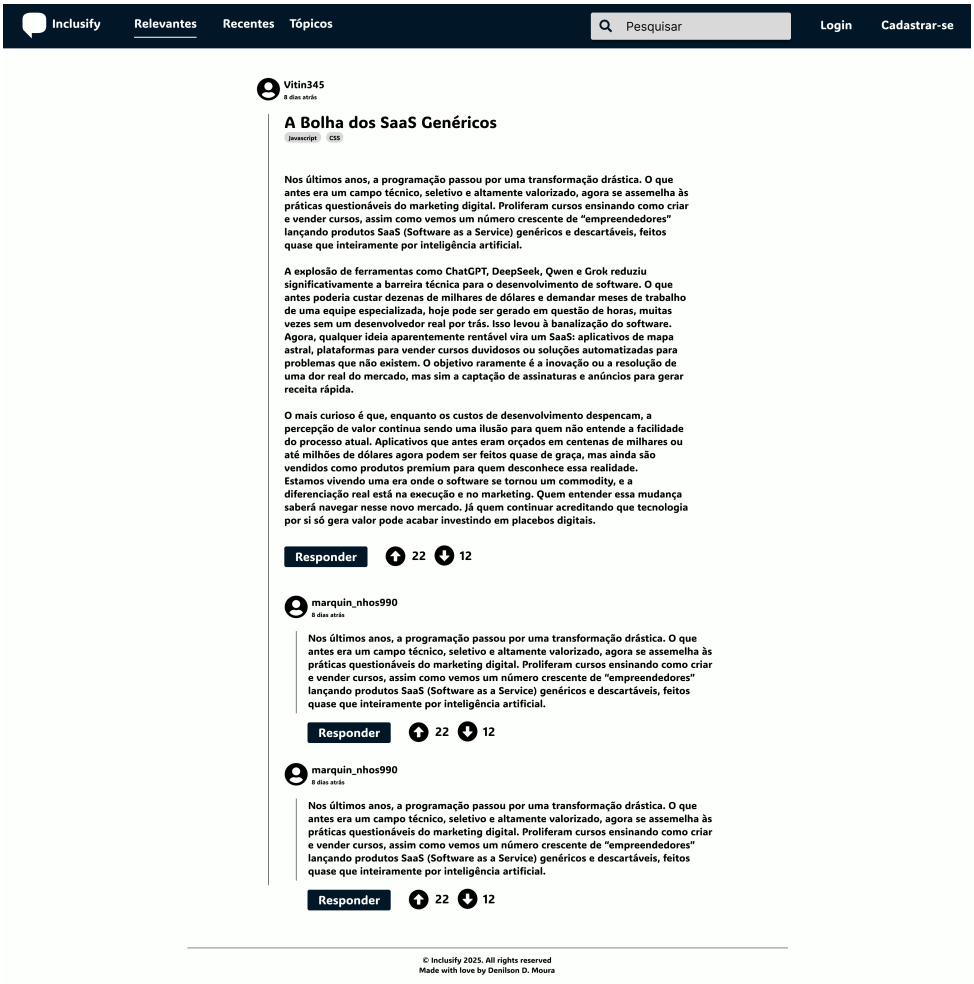
Figura 7 – Listagem de postagens.

Fonte: Autoria própria

- Na tela de detalhes da postagem exibimos o conteúdo completo, o nome do autor e a data de publicação. Além disso, são apresentados o número de votos (positivos e negativos) e a árvore de comentários de outros usuários em resposta, conforme ilustrado na Figura 8.

- Por fim, a tela de tópicos lista os assuntos disponíveis para facilitar a localização de postagens relevantes. Em cada item, são exibidos o título do tópico, uma breve descrição de seu conteúdo e o número de postagens relacionadas, conforme ilustrado na Figura 9.

Figura 8 – Tela de detalhes da postagem.



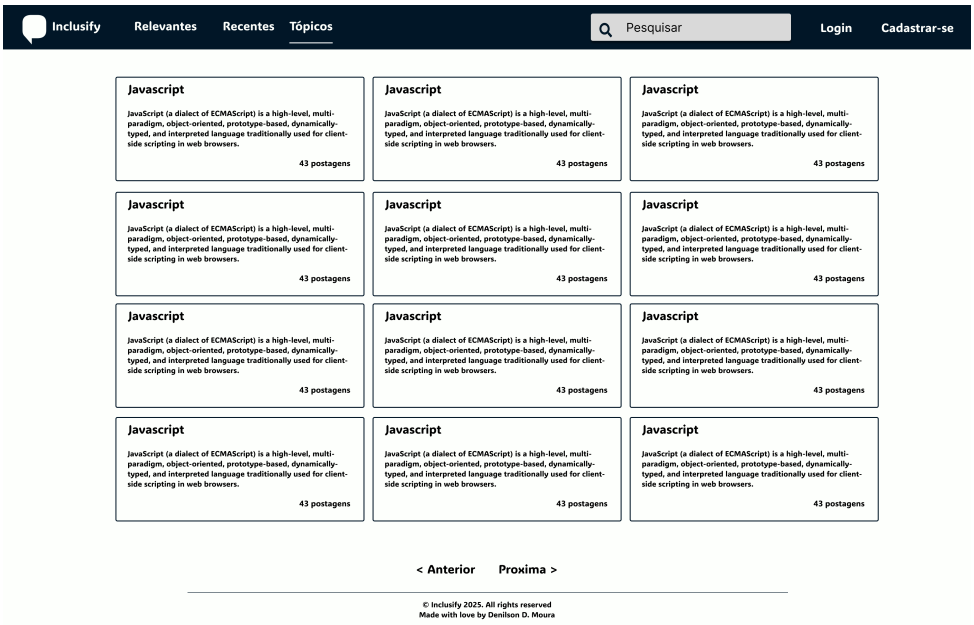
Fonte: Autoria própria

Ao final do projeto, dispomos de um conjunto coerente de diagramas e wireframes que amarram cada requisito funcional e de acessibilidade a um recurso técnico concreto. Isso evitou ambiguidades na codificação e já antecipou muitos problemas de usabilidade para pessoas cegas ou com baixa visão.

4.3 Codificação

Na codificação, aplicamos uma stack familiar (Django, Bootstrap, PostgreSQL, *Docker*) mas configurada desde o início pensando na acessibilidade. Cada componente foi criado utilizando o *Bootstrap 5*, adaptado para atender às diretrizes WCAG no que diz respeito ao contraste de cores, foco visível e semântica HTML apropriada. Em paralelo, adotamos um "*Scrum* solo"

Figura 9 – Listagem de tópicos.



Fonte: Autoria própria

com sprints de três semanas, apoiado por um quadro *Kanban* personalizado, no qual cada história transitava pelas colunas Funcionalidade pronta, Testes automatizados, Testes manuais e Documentação antes de ser considerada concluída.

4.3.1 Descrição da *Stack* utilizada

Para construir o fórum acessível, foram escolhidas tecnologias com as quais já existia familiaridade, garantindo agilidade na implementação e facilidade de manter padrões de qualidade, especialmente em relação à acessibilidade:

- Django 5.1.6:

  A escolha do Django se fundamenta na afinidade prévia com o framework e em sua capacidade de acelerar significativamente a construção de páginas. Com o server-side rendering, o Django gera HTML no servidor antes de enviá-lo ao navegador, o que facilita a indexação correta de marcos (landmarks) e o uso de leitores de tela como o NVDA. Essa renderização no back-end também reduz a dependência de scripts no cliente para estruturar o conteúdo, aumentando a confiabilidade da experiência para usuários cegos e de baixa visão.

- PostgreSQL 15:

  O PostgreSQL foi adotado devido à afinidade existente e à robustez para suportar operações de leitura e escrita de maneira eficiente. Além disso, a integração nativa com o

Object-Relational Mapping (ORM) do Django simplifica a modelagem de dados sem comprometer a performance.

- *Bootstrap 5*:

  O uso do *Bootstrap 5* se justifica pela familiaridade prévia e pela ampla adoção de suas classes utilitárias para garantir responsividade e padrões de acesso. Com recursos como estilos de foco visível e variáveis CSS para controle de cores, o Bootstrap 5 permite adaptar componentes visuais (botões, formulários, menus) de modo a cumprir as diretrizes WCAG, sem demandar customizações complexas. Essa abordagem acelera o desenvolvimento mantendo coerência no *design* acessível.

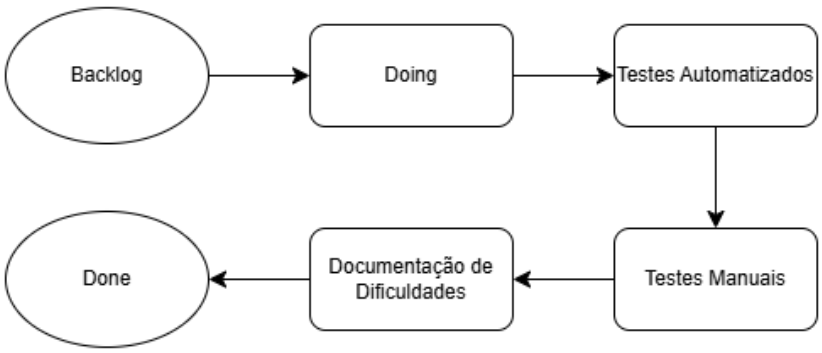
- *Docker*:

  O *Docker* foi incluído para simplificar a instalação do ambiente e o deploy da aplicação em diferentes estágios (desenvolvimento, homologação e produção). Ao encapsular cada serviço (Django, PostgreSQL, etc.) em contêineres isolados, é possível garantir que toda a equipe trabalhe com as mesmas versões de dependências e configurações, evitando divergências que poderiam prejudicar testes automatizados de acessibilidade. Além disso, o Docker facilita a replicação do ambiente em servidores de produção, tornando o processo de entrega contínua mais confiável.

4.3.2 *Definition of Done*

Para cada história de usuário, definimos que "pronto" seria mais do que código funcionando: precisava percorrer todas as colunas do nosso *Kanban* customizado como ilustrado na Figura 4.3.2. Isso garantiu que nenhuma entrega ficasse acessível apenas "no papel". A seguir, descreve-se cada etapa até o estado "Done":

Figura 10 – Descrição do fluxo de *Definition of Done*



Fonte: Autoria própria

1. Backlog

Aqui o desenvolvedor encontra uma lista priorizada de histórias de usuário aguardando início, organizada conforme valor de negócio e impacto em acessibilidade.

2. Doing

   Ao iniciar uma histórias de usuário, O desenvolvedor deve mover o card para esta caoluna do kanban. O card ficará aqui ate que seja completamente implementado a nivel de funcionalidade. Ou seja, toda a lógica de front-end e back-end deve estar implementada e funcionando.

3. Teste Automatizado

   Uma vez que a funcionalidade esta implementada, o desenvolvedor deve executar testes automatizados de acessibilidade com a biblioteca AxeCore na página ou componente correspondente. Não deve haver violações críticas (níveis AA ou AAA) — caso ocorram, elas devem ser corrigidas imediatamente antes de prosseguir. O relatório gerado pelo AxeCore ficará armazenado para consultas futuras.

4. Teste Manual de Leitor de Tela

   Após a aprovação nos testes automatizados, realiza-se a validação manual da usabilidade com o leitor de tela NVDA, seguindo um *checklist* específico encontrado na seção 4.4.2 deste documento. Em seguida, registra-se de forma concisa quais testes foram executados e quais ajustes se fizeram necessários.

5. Documentação

   Após a conclusão dos testes e a correção das falhas, documentam-se os principais erros de acessibilidade identificados, as dificuldades encontradas e as soluções implementadas.

6. Done

   Nesta etapa a história de usuário está totalmente concluída: a funcionalidade foi implementada conforme os requisitos, passou pelos testes automatizados e manuais de acessibilidade, e teve todas as falhas corrigidas. Além disso, a documentação está atualizada, incluindo o registro das abordagens adotadas, resultados dos testes e quaisquer observações relevantes.

Com esse fluxo, evitamos que implementações de acessibilidade ocorram somente no final do projeto e tornamos cada iteração verdadeiramente inclusiva, pois nada avançava sem aprovação automática e manual.

4.4 Testes

Para assegurar a conformidade contínua com os requisitos funcionais e de acessibilidade, adotamos uma estratégia de testes que combina verificações automatizadas e validações manuais em cada etapa de desenvolvimento, conforme previsto no *Definition of Done*.

4.4.1 Testes automatizados com AXE-CORE

Para execução dos testes automatizados, adotou-se o axe-core como ferramenta de testes automatizados. O axe-core permite a detecção precoce de violações de regras WCAG (nível AA/AAA) ainda na fase de desenvolvimento, evitando regressões e reforçando a qualidade contínua do código. Dessa forma, nenhuma funcionalidade é considerada "pronta" antes de passar pelos testes axe-core, garantindo que potenciais problemas de contraste, marcação semântica ou atributos ARIA sejam identificados e corrigidos imediatamente.

4.4.1.1 Funcionamento dos testes

Para automatizar a verificação de problemas de acessibilidade, criamos um script de testes em Python que abre cada página do nosso fórum e executa checagens usando o axe-core como ilustrado na Figura 11. A ideia é simular um usuário navegando no site, mas em modo "virtual", para identificar violações das regras WCAG sem depender de revisão manual a cada mudança.

Para isso, usamos as seguintes tecnologias:

1. *Chrome Headless*

   Pense no navegador Google Chrome como um programa que abre uma janela, carrega páginas da web e exibe o visual para o usuário. Quando falamos de "*Chrome Headless*", estamos usando exatamente o mesmo mecanismo de navegação do Chrome, mas sem abrir janelas na tela. Ou seja, ele carrega o site "em segundo plano", sem mostrar nada para quem está usando o computador. Isso é útil porque, num servidor ou contêiner, não há interface gráfica, e precisamos "ver" a página apenas no código para executar os testes.

2. *ChromeDriver*

   O *ChromeDriver* é um "pequeno tradutor" que permite que outro programa (no nosso caso, o *Selenium*) controle o Chrome. O *ChromeDriver* faz exatamente isso: recebe comandos do *Selenium* ("abra a página /login", "clique no botão Entrar"), repassa ao Chrome e devolve os resultados, como o HTML carregado e o que acontece quando um botão é clicado.

3. *Selenium*

O *Selenium* é a biblioteca que funciona como "maestro" do processo: ele se comunica com o *ChromeDriver* e, indiretamente, com o *Chrome Headless*, para automatizar a navegação em um navegador real. Na prática, o *Selenium* executa passos como "carrega esta URL", "espere até que a página esteja totalmente carregada", "encontre um elemento com este identificador" e "mande o axe-core inspecionar a página que acabou de abrir". Por trás disso, o *Selenium* abre o Chrome em modo Headless, carrega cada rota do fórum e nos dá acesso ao conteúdo da página para que o axe-core possa analisar.

4. Sandbox e –no-sandbox

   Ao instalar e rodar o Chrome em ambientes restritos como contêineres *Docker*, o navegador costuma bloquear certas funcionalidades por questões de segurança. Esse isolamento se chama "sandbox": o Chrome roda dentro de um espaço controlado para não interferir no restante do sistema. Para que o *Chrome Headless* funcione corretamente dentro do contêiner (sem interface gráfica), precisamos desativar algumas dessas proteções, usando o parâmetro –no-sandbox. Isso permite que o navegador seja executado mesmo dentro do ambiente isolado, sem travar por falta de permissões.

5. *StaticLiveServerTestCase*

   No Django, temos um recurso que permite subir um servidor web temporário para testar nossas páginas como se fossem "ao vivo" (em "localhost"). A classe *StaticLiveServerTestCase* herda dessa funcionalidade e, além disso, já serve arquivos estáticos (CSS, JavaScript, imagens) automaticamente. Usando essa classe, conseguimos executar testes que abrem o site num endereço como http://127.0.0.1:8081/ e simulam acesso real, garantindo que o CSS e o JavaScript sejam carregados corretamente durante os testes de acessibilidade.

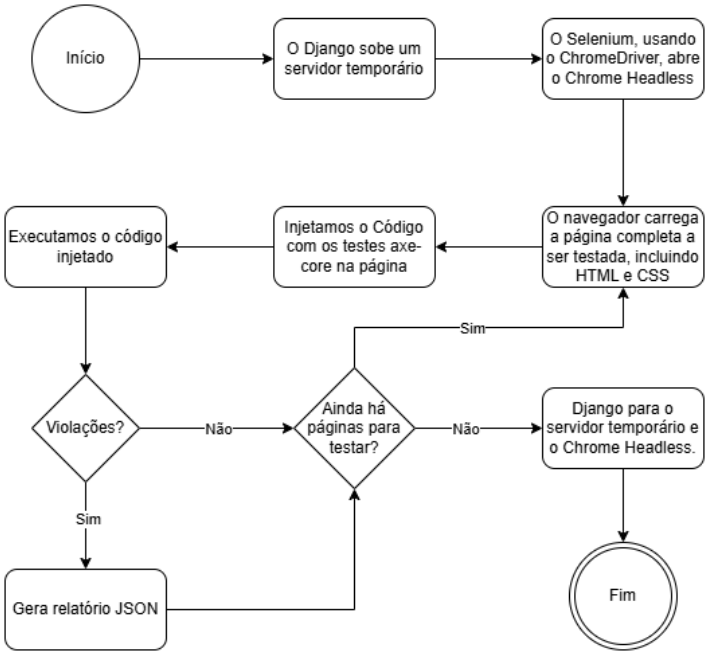
6. setUpClass

   Em testes automatizados, existe uma distinção entre "configurar algo antes de cada teste" e "configurar algo uma única vez antes que todos os testes daquela classe rodem". O método setUpClass cuida da segunda situação: ele é executado uma única vez quando a classe de testes começa a rodar. Nele, criamos e configuramos o navegador (*Chrome Headless* + *ChromeDriver*) e inicializamos o servidor Django temporário. Assim, não precisamos abrir e fechar o navegador para cada rota testada—o processo fica mais rápido e economizamos recursos.

7. Injeção do axe-core

   Quando o navegador carrega uma página (por exemplo, a rota de login), queremos que o axe-core analise o código HTML e detecte possíveis problemas de acessibilidade (como imagens sem texto alternativo ou botões sem aria-label). Mas o axe-core é uma biblioteca JavaScript que precisa "entrar" na página para fazer a varredura. O método axe.inject()

Figura 11 – Fluxo de Testes de acessibilidade com AXE-CORE em uma aplicação Django



Fonte: Autoria própria

faz exatamente isso: injeta (ou seja, "coloca") o código do axe-core dentro da página que o *Chrome Headless* acabou de carregar. Em seguida, a chamada axe.run() executa a checagem e retorna uma lista de violações encontradas.

4.4.1.2 Fluxo de testes

1. Início (setUpClass)

   - O Django sobe um servidor temporário em uma porta qualquer no localhost.
   - O *Selenium*, usando o *ChromeDriver*, abre o *Chrome Headless*, configurando-o com –headless –no-sandbox –disable-dev-shm-usage.
   - O navegador "aponta" para o endereço onde o Django está servindo o site.

2. Carregar página e executar o axe-core

   - Para cada rota do fórum, o teste chama algo como driver.get(self.live_server_url + path). O navegador carrega a página completa, incluindo HTML e CSS.
   - Logo em seguida, criamos um objeto Axe(self.driver), que representa o axe-core rodando dentro do Chrome.
   - Chamamos axe.inject(), que injeta o script do axe-core na página atual. Pense nisso como "colar" as ferramentas de inspeção dentro da página, para que elas consigam ver todos os elementos.

- Chamamos axe.run(), que é quando o axe-core faz a análise completa. Ele verifica dezenas de regras, como contraste de cores, presença de texto alternativo em imagens, uso correto de cabeçalhos e *labels*, marcadores ARIA e semântica HTML.

3. Coleta e tratamento dos resultados

   - Se o axe-core retornar uma lista de "violações" (por exemplo, "Elemento X não possui alt" ou "Cabeçalho H2 está sendo usado antes de H1"), o teste registra esses problemas.
   - Os detalhes de cada violação são salvos em arquivos JSON dentro de uma pasta tests/reports/, para que possamos revisá-los depois.
   - Caso existam violações com nível de gravidade crítico, o teste falha explicitamente, interrompendo o pipeline de integração contínua ou sinalizando para o desenvolvedor que algo precisa ser corrigido.

4. Finalização (tearDownClass)

   - Depois de correr todos os métodos de teste, o Django para o servidor temporário e nós fechamos o *Chrome Headless*.
   - Se não houver violações críticas, todos os testes passam, marcando a etapa de "Teste Automatizado: AxeCore" como concluída no nosso Definition of Done.

4.4.1.3 Principais falhas encontradas

Nesta seção, apresentamos as principais violações de acessibilidade encontradas pelo `axe-core` durante os testes automatizados. Cada falha é descrita de forma sucinta, seguida dos trechos de código que apresentam o erro e sua respectiva correção. O objetivo é tornar mais clara a natureza de cada problema e demonstrar como a solução foi aplicada no código do fórum acessível.

1. ARIA `role` não permitido no elemento `<section>`

   - Descrição: O `axe-core` sinalizou que o atributo `role="list"` não é permitido em um elemento `<section>`, uma vez que o elemento `<section>` já possui semântica própria e não deve receber diretamente esse tipo de `role` conforme exemplificado no Código 4.1. Para representar listas, recomenda-se utilizar elementos nativos de lista (`<ul>` / `<ol>`) que já atendem aos requisitos semânticos e de acessibilidade.
   - Código com erro:

Código 4.1 – Exemplo: ARIA role não permitido no elemento

```
<section class="list-group" role="list" aria-labelledby="
   postagens-titulo">
  <h1 id="postagens-titulo" class="mb-4">Relevantes</h1>
  <p class="text-muted">Nenhuma postagem disponível no
     momento.</p>
</section>
```

- Correção aplicada: Para corrigir, removemos o `role="list"` do `<section>` e criamos uma lista não ordenada (`<ul>`) contendo um único `<li>` que representa a mensagem de ausência de postagens como se pode ver no Código 4.2. Dessa forma, mantemos a semântica correta de lista, sem atribuir papéis ARIA indevidos ao `<section>`.

Código 4.2 – Correção: ARIA role não permitido no elemento

```
<section aria-labelledby="postagens-titulo">
  <h1 id="postagens-titulo" class="mb-4">Relevantes</h1>
  <ul class="list-group">
    <li class="list-group-item text-muted">
      Nenhuma postagem disponível no momento.
    </li>
  </ul>
</section>
```

2. ARIA `role="list"` exige filhos com `role="listitem"` (listitem ausente)

   - Descrição: O `axe-core` aponta que, ao definir `role="list"` em um contêiner, cada filho direto desse contêiner deve ter `role="listitem"` conforme Código 4.3. No exemplo abaixo, o elemento tinha apenas um parágrafo (`<p>`) como filho, o que não satisfaz o requisito de filho do tipo `listitem`.
   - Código com erro:

Código 4.3 – Exemplo: ARIA role list exige que os filhos tenham role listitem

```
<section class="list-group" role="list" aria-labelledby="
   postagens-titulo">
  <h1 id="postagens-titulo" class="mb-4">Relevantes</h1>
  <p class="text-muted">Nenhuma postagem disponível no
     momento.</p>
</section>
```

   - Correção aplicada: Removemos o `role="list"` do `<section>` e substituímos o parágrafo por uma lista (`<ul>`) cujo filho direto é um item de lista (`<li>`) conforme Código 4.4. Dessa forma, a estrutura ARIA torna-se válida e semanticamente coerente.

Código 4.4 – Correção: ARIA role list exige que os filhos tenham role listitem

```
<section aria-labelledby="postagens-titulo">
  <h1 id="postagens-titulo" class="mb-4">Relevantes</h1>
  <ul class="list-group">
    <li class="list-group-item text-muted">
      Nenhuma postagem disponível no momento.
    </li>
  </ul>
</section>
```

3. Estrutura incorreta de listas (`<ul>` com filho `<p>`)

   - Descrição: Outra violação frequente ocorre quando um elemento `<ul>` contém diretamente um parágrafo (`<p>`) em vez de itens de lista (`<li>`) conforme Código 4.5. Segundo as boas práticas de acessibilidade, o único filho permitido de `<ul>` ou `<ol>` é `<li>`.
   - Código com erro:

     Código 4.5 – Exemplo: Estrutura incorreta de lista: um <ul> contendo diretamente <p> como filho.

     ```
     <ul class="list-group">
       <p class="text-muted">Nenhuma postagem disponível no
          momento.</p>
     </ul>
     ```

   - Correção aplicada: Neste caso, transformamos o parágrafo em um item de lista, encapsulando-o em `<li>` conforme Código 4.6. Assim, mantemos a semântica correta de elemento de lista, e o leitor de tela consegue informar ao usuário que se trata de uma lista vazia com um item de aviso.

     Código 4.6 – Correção: Estrutura incorreta de lista: um <ul> contendo diretamente <p> como filho.

     ```
     <ul class="list-group">
       <li class="list-group-item text-muted">
         Nenhuma postagem disponível no momento.
       </li>
     </ul>
     ```

4. ARIA `role` não permitido no elemento `<form>`

   - Descrição: O `axe-core` indicou que não é necessário e nem permitido atribuir `role="form"` a um `<form>`, pois este elemento já possui semântica própria de formulário

como vemos no Código 4.7. Quando a página marca o contêiner como `<form>`, não se deve duplicar essa semântica via ARIA.

- Código com erro:

Código 4.7 – Exemplo: ARIA role não permitido no elemento <form>

```
<form method="post" role="form" aria-label="Formulário de
   login">
  <!-- campos de login -->
</form>
```

- Correção aplicada: Removemos o `role="form"`, mantendo apenas o atributo aria-label para descrever o propósito do formulário ao leitor de tela conforme Código 4.8.

Código 4.8 – Correção: ARIA role não permitido no elemento <form>

```
<form method="post" aria-label="Formulário de login">
  <!-- campos de login -->
</form>
```

5. ARIA `role` não permitido nos elementos `<article>` e `<input>`

- Descrição: Foram detectados casos em que atributos ARIA eram aplicados de forma indevida em elementos que já possuem semântica própria como mostra o Código 4.9. Exemplos incluem:
  - `role="article"` em `<article>`, redundante pois o próprio elemento já comunica essa função.
  - `role="checkbox"` em `<input type="checkbox»`, que já possui semântica de caixa de seleção nativa.
- Códigos com erro:

Código 4.9 – Exemplo: ARIA role não permitido nos elementos <article> e <input>

```
<article role="article">
  <!-- conteúdo do artigo -->
</article>

<input role="checkbox" type="checkbox" checked aria-checked=
   "true">
```

- Correção aplicada: Removemos o atributo ARIA redundante, mantendo apenas as classes de estilo e os atributos ARIA necessários para descrever estados dinâmicos (quando aplicável) conforme Código 4.10.

Código 4.10 – Correção: ARIA role não permitido nos elementos <article> e <input>

```
<article class="mb-4">
  <!-- conteúdo do artigo -->
</article>

<li class="note-dropdown-item" aria-label="p">
  <p>Normal</p>
</li>

<input type="checkbox" checked aria-checked="true">
```

4.4.2 Testes manuais com NVDA

Embora os testes automatizados com o axe-core sejam essenciais para detectar grande parte das violações de acessibilidade, eles não conseguem capturar todas as sutilezas da experiência real de navegação com leitores de tela. Por isso, para cumprir integralmente a nossa Definition of Done, é imprescindível complementar a automação com sessões de testes manuais utilizando o NVDA (NonVisual Desktop Access). Esses testes garantem que o usuário cego ou com baixa visão efetivamente consiga navegar, compreender e interagir com o fórum de forma fluida, sem ambiguidades ou barreiras.

No fluxo de Definition of Done, os testes manuais com NVDA ocorrem logo após a etapa de testes automatizados. Se o axe-core detectar alguma violação, a correção deve ser aplicada imediatamente antes de seguir. Após a correção e aprovação no axe-core, iniciam-se os testes manuais, que verificam aspectos como ordem de leitura, anúncios de elementos dinâmicos e usabilidade geral, aspectos que só podem ser totalmente validados quando um leitor de tela "fala" o conteúdo. Somente quando todas as etapas — automatizado e manual — estiverem concluídas com sucesso, a história de usuário é considerada "Done".

4.4.2.1 Testes realizados

A seguir, apresenta-se o *checklist* de testes realizados com NVDA para cada página ou componente do fórum:

1. Estrutura de cabeçalhos

    - Verificar se há apenas um <h1> e se os demais <h2>–<h6> seguem ordem lógica, sem pular níveis.
    - Confirmar que o NVDA anuncia cada nível corretamente, de modo que a hierarquia de tópicos faça sentido ao usuário.

2. *Landmarks* semânticos

    - Garantir que <header>, <nav>, <main>, <aside> e <footer> (ou role= equivalentes) estejam presentes e identificáveis pelo NVDA ao pressionar a tecla “D”.
    - Validar se as regiões marcadas como elementos de navegação (por exemplo, barras laterais ou seções de links) são detectadas corretamente como regiões distintas.

3. *Link* “Pular para o conteúdo”

    - Certificar que o link aparece visível quando em foco (geralmente após o primeiro Tab).
    - Confirmar que, ao ativar esse link, o foco salta diretamente para a região principal (<main>), evitando a leitura repetida de menus.

4. Imagens e texto alternativo

    - Verificar que todas as imagens significativas possuem atributo alt descritivo (por exemplo, <img src=”logo.png”alt=”Logo do Fórum Inclusivo»).
    - Garantir que imagens decorativas sejam definidas com alt= para serem ignoradas pelo NVDA.

5. Navegação por teclado (Tab)

    - Testar a sequência de pressionamentos de Tab para percorrer todos os controles interativos (botões, links, campos de formulário) na ordem esperada, sem “pular” elementos relevantes ou entrar em loops infinitos.
    - Confirmar que cada elemento em foco exibe um contorno visível (outline) para auxiliar também usuários com baixa visão.

6. Associação de *labels* em formulários

    - Verificar que cada <input>, <select> ou <textarea> possui um <label> associado (por exemplo, <label for=”email»E-mail</label>).
    - Caso não seja possível usar <label>, assegure que exista aria-label ou aria-labelledby.
    - Checar se mensagens de erro ou validações são associadas via aria-describedby, para que o NVDA leia o texto de erro junto ao campo correspondente.

7. Componentes dinâmicos

    - Em elementos como acordeons, modais e menus suspensos, certificar que atributos como aria-expanded, aria-hidden e aria-selected estejam presentes e atualizados conforme o estado do componente.

- Confirmar que, quando o estado muda (por exemplo, abre um dropdown), o NVDA anuncia a mudança (por exemplo, “Menu X expandido”, “Botão Salvar desativado” etc.).

8. Regiões dinâmicas e alertas

   - Testar se *feedbacks* dinâmicos, como mensagens de sucesso/erro ou notificações, utilizam role=”alert”ou aria-live, de modo que o NVDA leia automaticamente o novo texto sem que o usuário tenha que navegar novamente para aquele ponto.

9. Fluxo de leitura contínua

   - Utilizar o NVDA + seta para baixo para percorrer o texto de forma linear. Observar se há anúncios confusos, repetições desnecessárias ou lacunas na leitura. Garantir que o fluxo textual faça sentido quando ouvido.

10. Tabelas de dados

    - Para cada <table>, confirmar que exista <th> definido para cada linha/coluna e que o atributo scope=”col”ou scope=”row”esteja corretamente aplicado.
    - Testar com NVDA + Ctrl + T (relatório de cabeçalhos de tabela) para verificar se o leitor entende a relação entre dados e cabeçalhos.

11. Links independentes de contexto

    - Pressionar “K” no NVDA para listar todos os links da página e verificar se cada texto de link soa coerente mesmo fora de contexto. Por exemplo, evitar “clique aqui” sem indicar o destino.

12. Foco e navegação rápida

    - Além de confirmar a aparência visual do :focus, usar NVDA para verificar que o leitor posiciona o foco no elemento correto ao pressionar Tab.
    - Testar também atalhos de navegação rápida do NVDA, como “H” para headings, “F” para formulários, “B” para botões, garantindo que o NVDA leva o usuário ao elemento correto.

13. Elementos automáticos e animações

    - Em componentes que mudam automaticamente (carrosséis, cronômetros, contadores), verificar se existe opção para pausar ou desativar a animação.
    - Garantir que tais mudanças automáticas não interrompam abruptamente a leitura do NVDA, permitindo que o usuário inicie ou pare o fluxo manualmente.

14. Idioma do documento

    - Certificar que o <html lang=”pt-BR» esteja presente para toda a página.
    - Caso haja trechos em outro idioma (por exemplo, citações em inglês), verificar que utilizam lang=”en”ou aria-language=”en”para garantir pronúncia correta.

Seguindo esse *checklist*, cada página ou componente do fórum é validado de forma abrangente, assegurando não apenas o cumprimento técnico das regras WCAG, mas também a experiência efetiva do usuário com baixa visão ou cego. Somente após a conclusão bem-sucedida desses testes, a etapa “Teste Manual de Leitor de Tela” é considerada pronta, permitindo que a história de usuário avance para a última fase de documentação e publicação.

4.4.2.2 Principais falhas encontradas

Nesta seção, relatamos as principais dificuldades observadas durante a validação manual com o leitor de tela NVDA (NonVisual Desktop Access). Embora os testes automatizados cubram diversos tipos de violações, existem aspectos de usabilidade e semântica que somente a interação real com o NVDA consegue revelar. Para cada falha, apresentamos sua descrição, o trecho de código original e a correção adotada.

1. Elementos de navegação sem descrição ou com descrição insuficiente

    - Descrição: O NVDA depende de textos visíveis ou atributos ARIA (`aria-label`, `aria-labelledby`) para anunciar de forma clara links, botões e outros controles de navegação. Quando um elemento interativo aparece sem texto visível ou sem um atributo de acessibilidade como vemos no Código 4.11, o leitor de tela não consegue informar ao usuário qual é a sua função.
    - Código sem descrição:

      Código 4.11 – Exemplo: Elementos de navegação sem descrição ou com descrição insuficiente.

```
<a class="nav-link text-white" href="{% url '
   postagens_relevantes_list' %}">
  Relevantes
</a>
```

    - Correção aplicada: Adicionamos o atributo aria-label=”Ver postagens relevantes”para reforçar, ao NVDA, o propósito exato do link conforme Código 4.12. Assim, mesmo que o texto visível seja suficiente para leitores que enxergam, o usuário de leitor de tela terá confirmação precisa do destino da navegação.

Código 4.12 – Correção: Elementos de navegação sem descrição ou com descrição insuficiente.

```
<a class="nav-link text-white" href="{% url '
   postagens_relevantes_list' %}"
   aria-label="Ver postagens relevantes">
  Relevantes
</a>
```

O mesmo princípio foi aplicado em botões com ícones, garantindo que o NVDA anuncie tanto a ação do botão quanto, quando necessário, a quantidade de votos ou outra informação dinâmica conforme Código 4.13:

Código 4.13 – Exemplo 2: Elementos de navegação sem descrição ou com descrição insuficiente.

```
<a href="#" role="button" aria-label="Up Vote"
   class="btn px-1 d-flex align-items-center">
  <i class="fas fa-arrow-alt-circle-up fs-3"></i>
  <span class="mx-2" aria-label="Up Votes">
    {{ comentario.upvotes_count }}
  </span>
</a>
```

2. Elementos de navegação redundantes ao leitor de tela

   - Descrição: Quando o mesmo *link* ou item de navegação aparece em mais de um ponto da interface — por exemplo, um logo que redireciona à página principal e também um *link* textual “Home” como exemplificado no Código 4.14 — o NVDA pode anunciá-los de forma duplicada, causando confusão. Para evitar essa redundância, deve-se ocultar determinados elementos visuais do leitor de tela usando `aria-hidden="true"`.
   - Código antes (sem ocultação):

   Código 4.14 – Exemplo: Elementos de navegação redundantes ao leitor de tela podem confundir o usuário.

```
<a class="navbar-brand text-white" href="{% url '
   postagens_relevantes_list' %}">
  <img src="{% static 'img/logo.svg' %}" width="30" alt="
     Logo do Inclusify"/>
  Inclusify
</a>
```

```
<a class="nav-link text-white" href="{% url '
   postagens_relevantes_list' %}"
   aria-label="Ver postagens relevantes">
  Relevantes
</a>
```

- Correção aplicada: Ocultamos o link do logo do leitor de tela, pois o texto ”Relevantes”já aponta para a mesma página conforme Código 4.15. Assim, o NVDA anuncia apenas o link textual principal, otimizando a navegação.

  Código 4.15 – Correção: Elementos de navegação redundantes ao leitor de tela podem confundir o usuário.

```
<a class="navbar-brand text-white" href="{% url '
   postagens_relevantes_list' %}"
   aria-hidden="true">
  <img src="{% static 'img/logo.svg' %}" width="30" alt="
     Logo do Inclusify"/>
  Inclusify
</a>

<a class="nav-link text-white" href="{% url '
   postagens_relevantes_list' %}"
   aria-label="Ver postagens relevantes">
  Relevantes
</a>
```

3. Áreas de exibição de elementos dinâmicos (*toasts* e alertas) sem a devida *role*

   - Descrição: Mensagens dinâmicas, como *toasts* (notificações temporárias) e alertas exemplificadas no Código 4.16, devem ser anunciadas automaticamente pelo leitor de tela. Para isso, precisam estar marcadas com `role="alert"` e possuir uma região viva de atualização, por meio de `aria-live="assertive"` e `aria-atomic="true"`. Sem esses atributos, o NVDA não detecta a chegada de novos conteúdos em tempo real.
   - Código antes:

     Código 4.16 – Exemplo: Exibição de toasts/alertas sem role.

```
<div class="toast align-items-center text-white bg-info">
  <!-- Conteúdo da notificação -->
</div>
```

- Correção aplicada: Adicionamos `role="alert"` e atributos `aria-live` para que o NVDA anuncie imediatamente o conteúdo da notificação assim que ela aparecer conforme Código 4.17.

Código 4.17 – Correção: Exibição de toasts/alertas sem role.

```
<div class="toast align-items-center text-white bg-info"
     role="alert" aria-live="assertive" aria-atomic="true">
  <!-- Conteúdo da notificação -->
</div>
```

Vale destacar que o projeto já implementava corretamente esses atributos em outros contextos, mas este exemplo evidencia a importância de verificar cada componente dinâmico.

4. Elementos dinâmicos (acordeons, modais, menus) sem sinalização de estado via ARIA

- Descrição: Componentes interativos que mudam de estado (por exemplo, acordeons que se expandem ou colapsam, modais que abrem e fecham, menus suspensos) como exemplificado no Código 4.18, devem informar ao leitor de tela quando sua condição é alterada. Para isso, usa-se `aria-expanded` e `aria-controls`, garantindo que o NVDA anuncie corretamente as mudanças.
- Código sem atributo ARIA:

Código 4.18 – Exemplo: Elementos dinâmicos sem sinalização de estado ARIA.

```
<button class="btn" data-bs-toggle="collapse"
        data-bs-target="#respostaForm-123">
  Responder
</button>
```

- Correção aplicada: Incluímos `aria-expanded = "false"` e `aria-controls = "respostaForm-123"` no botão. Também adicionamos `aria-label` descritivo ao próprio botão como mostra o Código 4.19.

Código 4.19 – Correção: Elementos dinâmicos sem sinalização de estado ARIA.

```
<button class="btn" data-bs-toggle="collapse"
        data-bs-target="#respostaForm-123"
        aria-expanded="false"
        aria-controls="respostaForm-123"
        aria-label="Responder postagem">
  Responder
</button>
```

4.5 Principais Dificuldades Encontradas

Após todo o ciclo de desenvolvimento do fórum, com base nos registros colhidos ao longo do processo durante a etapa de "documentar dificuldades" do Definition of Done, levantaram-se as principais barreiras enfrentadas na aplicação prática dos critérios de acessibilidade. Vale relembrar que um dos objetivos deste trabalho era, justamente, identificar as maiores dificuldades de incorporar a acessibilidade desde o início até a entrega do sistema. A seguir, detalham-se cada uma dessas dificuldades de forma didática:

4.5.1 Decidir como integrar acessibilidade desde o início

Logo no planejamento e durante a construção do Backlog, surgiu a dúvida de como "embutir" critérios WCAG em cada card de história de usuário, pois os critérios abrangem desde exigências específicas de componentes até diretrizes globais para toda a aplicação. Para desenvolvedores sem familiaridade prévia, catalogar manualmente cada critério em cards de user story mostrou-se trabalhoso e pouco produtivo. Sem um mecanismo automatizado, correções levavam tempo e geravam débito técnico, pois era fácil esquecer de incluir uma checagem de acessibilidade ao criar novos componentes. A solução encontrada foi o uso combinado de testes automatizados baseados em axe-core e *checklists* de auditoria com leitor de tela (NVDA) que provou-se uma saída eficaz para mapear requisitos WCAG a elementos HTML conhecidos, reduzindo a sobrecarga de catalogação manual.

4.5.2 Revisão e uso correto das tags semânticas do HTML5

Foi necessário revisar todo o vocabulário de tags do HTML5, aprendendo a distinguir as tags com significado semântico, como <main>, <nav> e <article>, daquelas usadas apenas para estruturação visual, como e . No início, erros de semântica — como o uso de <section role="list" > no lugar de uma lista propriamente dita, como <ul> — prejudicavam a leitura contínua por leitores de tela, dificultando a navegação para usuários com deficiência visual. Esses problemas exigiram revisões e reestruturação do código para corrigir as falhas de acessibilidade. No entanto, com o tempo e a prática, esses erros tornaram-se mais fáceis de identificar, e logo deixaram de aparecer nas novas histórias de usuário, demonstrando a evolução na aplicação correta da semântica HTML.

4.5.3 Curva de aprendizado no uso do leitor de tela NVDA

A curva de aprendizado no uso do NVDA também se mostrou um desafio.Foi preciso tempo e prática para assimilar comandos básicos, como a tecla "H" para navegar entre cabeçalhos ou "K" para avançar por links. No início, a ausência de um roteiro estruturado tornava os testes manuais irregulares e, muitas vezes, superficiais, dificultando a identificação precisa dos problemas de acessibilidade. Com a prática contínua e, principalmente, após a criação de

um roteiro passo a passo, os testes tornaram-se mais confiáveis e eficientes, permitindo detectar falhas com mais agilidade e segurança.

4.5.4 Erros de acessibilidade em componentes de terceiros

Por fim, a integração de componentes de terceiros, como o editor Summernote, apresentou diversas dificuldades relacionadas à personalização de propriedades de acessibilidade. Ajustes como a modificação de rótulos (*labels*), o controle do foco automático e a correção de elementos com descrições em idioma diferente do restante do sistema exigiram soluções específicas. Como esses componentes não ofereciam suporte nativo a muitos dos requisitos de acessibilidade, foi necessário recorrer ao uso de seletores com jQuery para modificar diretamente o HTML após a renderização, inserindo manualmente os atributos acessíveis adequados. Esse processo aumentou a complexidade da implementação e exigiu atenção constante para garantir a conformidade com as diretrizes.

Em síntese, essas dificuldades ressaltam como aplicar acessibilidade na prática exige não apenas conhecimento teórico das WCAG, mas também disciplina na organização dos critérios, domínio de semântica HTML, atenção aos rótulos ARIA, familiaridade com leitores de tela e capacidade de adaptar componentes externos. Superar esses desafios foi fundamental para garantir que o fórum oferecesse uma experiência inclusiva desde a concepção até a entrega, servindo de alicerce para as recomendações que se seguem.

4.6 Principais Recomendações

As recomendações a seguir foram levantadas com base nos desafios encontrados durante o desenvolvimento do projeto, especialmente no que diz respeito à acessibilidade. Um dos objetivos específicos deste trabalho era justamente reunir e sistematizar boas práticas que contribuíssem para a construção de interfaces mais acessíveis e inclusivas. Ao longo da implementação e dos testes, identificaram-se diversos obstáculos recorrentes, e, a partir deles, foram formuladas as seguintes recomendações de forma didática e aplicada:

- **Incluir acessibilidade na definição de pronto das tarefas.** Incorpore testes automatizados e testes com leitor de tela na *definition of done* de cada história de usuário, evitando o acúmulo de débito técnico e assegurando tratamento contínuo da acessibilidade.
- **Implementar skip-links em todas as páginas.** Adicione links “Pular para o conteúdo principal” e, em páginas mais complexas, “Pular para navegação”, “Pular para busca” ou “Pular para o rodapé”, cada um apontando para um `id` distinto.
- **Usar HTML semântico com atributos ARIA.** Utilize tags HTML5 como `<header>`, `<main>`, `<nav>` e `<footer>`, acompanhadas de `role` e `aria-*` adequados, para facilitar a interpretação por tecnologias assistivas.

- **Manter hierarquia coerente de títulos.** Estruture seções com `<h1>` a `<h6>` de forma lógica, permitindo que usuários de leitores de tela naveguem rapidamente entre níveis por meio de atalhos.
- **Gerenciar corretamente o foco em elementos dinâmicos.** Em modais, mova o foco ao abrir para o primeiro elemento interativo e devolva ao gatilho ao fechar; em menus expansíveis, controle `aria-expanded` e restrinja o foco enquanto estiverem abertos.
- **Evitar links genéricos.** Cada link deve ser autoexplicativo fora de contexto (por exemplo, "Saiba mais sobre o Produto X" em vez de "Clique aqui"), garantindo sentido mesmo quando listado isoladamente.
- **Utilizar corretamente tabelas e listas.** Em tabelas, aplique `<th scope="col»` e `<th scope="row»` para cabeçalhos semânticos; para listas longas, separe conteúdos com títulos intermediários e use adequadamente `<ul>` e `<ol>`.
- **Reforçar visualmente o foco.** Assegure que o estilo de `:focus` seja altamente contrastado e perceptível, não dependendo apenas do contorno nativo, para suportar navegação por teclado. Utilize bibliotecas de estilo, como o **Bootstrap**, que já oferecem utilitários de foco aprimorado, garantindo contraste e feedback visual consistente.

Implementar essas recomendações contribui para que a navegação em sua aplicação seja mais rápida, compreensível e amigável, especialmente para quem depende exclusivamente do teclado ou de tecnologias assistivas. Além de cumprir critérios técnicos, essas práticas refletem um compromisso com a inclusão digital, promovendo o acesso equitativo à informação e aos serviços oferecidos pelas aplicações web.

5 CONCLUSÕES

Retomando o objetivo geral definido no Capítulo 1 na subseção 1.3.1, este estudo teve por propósito *"aplicar as diretrizes de acessibilidade da* Web Content Accessibility Guidelines *(WCAG) em conjunto com metodologias ágeis no desenvolvimento de um blog, a fim de criar uma aplicação web que seja acessível a pessoas cegas e de baixa visão, identificando e documentando as possíveis dificuldades enfrentadas no processo de desenvolvimento"*.

Para isso, foram estabelecidos três objetivos específicos: (1) analisar as diretrizes de acessibilidade, compreendendo seus princípios e critérios para o desenvolvimento do blog conforme as WCAG; (2) projetar e implementar um blog acessível utilizando métodos ágeis como SCRUM, com escopo limitado às necessidades de pessoas cegas e de baixa visão; e (3) documentar as dificuldades enfrentadas no processo de desenvolvimento e fornecer recomendações práticas para apoiar futuros desenvolvedores.

Para alcançar o primeiro objetivo específico, conduziu-se uma revisão da literatura, seguindo a estratégia PICOC e critérios de inclusão/exclusão, abrangendo publicações de 2018 a 2024 em repositórios reconhecidos. Identificaram-se métodos, ferramentas e processos recorrentes para verificação de conformidade WCAG (testes automatizados com axe-core, *linters*, *checklists*) e desafios relatados (entendimento superficial das diretrizes, ausência de requisitos definidos, barreiras culturais ágeis), evidenciando a lacuna entre verificação automatizada e experiência real de usuários com deficiência visual.

Em relação ao segundo objetivo, elaborou-se o protótipo de um fórum acessível (denominado *Inclusify*) como prova de conceito. Foi realizado o levantamento de requisitos gerais de fórum e requisitos não funcionais de acessibilidade centrados em compatibilidade com leitores de tela, navegação por teclado, semântica HTML e contraste adequado. A fase de projeto incluiu a modelagem conceitual, ER e protótipos de telas (em Figma), pensando desde o início em *landmarks*, hierarquia de títulos e *labels* semânticos. Na codificação, adotou-se Django, *Bootstrap 5* e *Docker*, com configuração e práticas desde o *backlog* até o *deploy* orientadas à acessibilidade. Implementou-se um fluxo de *Definition of Done* que incorporou testes automatizados com *axe-core* e testes manuais com NVDA em cada história de usuário, garantindo que nenhuma funcionalidade avançasse sem atender aos critérios de acessibilidade previamente definidos.

Para o terceiro objetivo, documentaram-se as principais dificuldades encontradas, como a integração de critérios WCAG em user stories ágeis sem sobrecarga manual, a revisão contínua de semântica HTML5, a curva de aprendizado no uso de leitores de tela (NVDA), e a adaptação de componentes de terceiros com suporte limitado a atributos ARIA. A partir desses registros, formularam-se recomendações práticas: incluir acessibilidade na definição de pronto; usar skip-

links; empregar HTML semântico com ARIA adequado; manter hierarquia coerente de títulos; automatizar testagem com axe-core e complementar com testes manuais; assegurar contraste e foco visível; documentar e compartilhar lições aprendidas.

5.1 Contribuiçoes

Este trabalho traz contribuições relevantes para a área de acessibilidade digital, destacando-se em três frentes:

- Integração prática de WCAG e metodologias ágeis: demonstra como incorporar critérios de acessibilidade desde o planejamento até a entrega em um fluxo ágil, definindo um Definition of Done que inclui testes automatizados (axe-core) e validação manual (NVDA). Essa abordagem fornece um modelo replicável para equipes que desejam considerar acessibilidade como parte integrante do processo de desenvolvimento, evitando que seja deixada para iterações tardias.
- Prova de conceito concreta (*Inclusify*): ao projetar e implementar um fórum acessível para pessoas cegas e de baixa visão, o estudo oferece artefatos reais (modelos de requisitos, protótipos de telas, dicionário de dados, exemplos de correções de código) que ilustram boas práticas e ajustes finos no HTML, ARIA e CSS para atender às WCAG. Tais exemplos contribuem para o repertório de desenvolvedores e pesquisadores interessados em aplicar acessibilidade em contextos reais de projeto.
- Recomendações e *checklist* ampliados: com base nos desafios documentados, o trabalho propõe um conjunto de recomendações práticas e um *checklist* de testes manuais com NVDA detalhado para cada componente/página, cobrindo cabeçalhos, *landmarks*, *skip-links*, *labels*, navegação por teclado, elementos dinâmicos, idioma e animações. Esse material auxilia profissionais a planejar e validar acessibilidade em projetos futuros, enriquecendo o conhecimento aplicado na área.

Essas contribuições ampliam o escopo científico de estudos sobre acessibilidade em contextos ágeis e oferecem subsídios para equipes de desenvolvimento que buscam integrar de modo efetivo as WCAG em suas práticas de engenharia de software.

5.2 Limitações

Apesar dos avanços obtidos, o trabalho apresenta algumas limitações importantes:

- Limitações dos testes automatizados: o uso de axe-core foi central para detectar muitas violações conforme WCAG, mas ferramentas automatizadas não capturam falhas como

clareza semântica em contexto, experiência do fluxo de navegação ou adequação de textos alternativos em cenários específicos (Vigo; Brown; Conway, 2013; Parvin et al., 2021). Logo, embora tenham reduzido débito técnico e permitido correções precoces, esses testes não identificam todas as falhas contempladas pela WCAG.

- Ausência de testes com usuários cegos e de baixa visão: embora tenha-se realizado testes manuais com NVDA para simular a experiência de navegação, não houve envolvimento direto de pessoas com deficiência visual para validar a usabilidade e detectar nuances de interação real. Conforme preconizado pela própria W3C, testes com usuários reais são imprescindíveis para garantir a adequação às necessidades específicas de cada público. Sem esse passo, não se pode afirmar com precisão o quão acessível o fórum é na prática para o público-alvo.
- Ausência de catalogação dos critérios WCAG específicos para usuários cegos e de baixa visão: não foi realizado um levantamento sistemático e categorizado dos critérios de acessibilidade definidos na WCAG com foco direto em como cada um se aplica a pessoas cegas e de baixa visão. Sem essa catalogação, tornou-se impraticável mapear quais critérios seriam mais relevantes para esse público e, consequentemente, quantificar exatamente quantos dos requisitos identificados foram efetivamente implementados no projeto. Essa falta de mapeamento detalhado impede avaliar de forma precisa o nível de cobertura das diretrizes WCAG voltadas para necessidades visuais, bem como priorizar ajustes futuros com base em métricas claras de conformidade para o público-alvo. A ausência desse inventário também limita a possibilidade de comparação entre a aplicação realizada e *benchmarks* de acessibilidade, dificultando demonstrar objetivamente o grau de aderência às recomendações para usuários cegos e de baixa visão.

Essas limitações mostram a necessidade de aprofundamentos que envolvam avaliação com usuários reais, estudos quantitativos de uso e adaptação da abordagem a diferentes *stacks* tecnológicas e contextos de produção.

5.3 Trabalhos Futuros

A partir das limitações identificadas e dos resultados obtidos, sugerem-se os seguintes trabalhos futuros:

- Catalogação e priorização em trabalhos futuros: recomenda-se que estudos subsequentes incluam uma fase dedicada à identificação e categorização dos critérios WCAG relevantes para pessoas cegas e de baixa visão, mapeando-os em cenários de uso reais. Essa abordagem permitirá mensurar quantitativamente a implementação de cada critério, orientar correções e evoluções com base em dados concretos e comparar o nível de conformidade com *benchmarks* e boas práticas na área de acessibilidade.

- Testes de usabilidade com usuários reais integrados ao ciclo ágil: planejar iterações de avaliação com pessoas cegas e de baixa visão em estágios distintos (protótipos e versão em produção), coletando *feedback* qualitativo e quantitativo que retroalimente o backlog. Essas sessões identificarão barreiras não detectáveis por testes automatizados ou simulações, orientando ajustes finos de navegação, semântica e interações dinâmicas.

Em suma, ao priorizar a catalogação sistemática dos critérios WCAG voltados a pessoas cegas e de baixa visão e integrar testes de usabilidade com usuários reais ao ciclo ágil de desenvolvimento, estabelece-se uma base robusta para aferir objetivamente o grau de conformidade e identificar barreiras que só emergem em contextos de uso real. Essa combinação de mapeamento criterioso e validação contínua com o público-alvo não apenas fornece métricas claras para orientar melhorias, mas também reforça uma cultura de desenvolvimento centrado no usuário, garantindo que a evolução do fórum seja pautada por evidências concretas sobre necessidades e desafios reais. Dessa forma, tais iniciativas formam alicerces sólidos para que o trabalho avance de maneira sustentável e com impacto efetivo na acessibilidade digital.

REFERÊNCIAS

BAPTISTA, A. et al. Web accessibility challenges and perspectives: A systematic literature review. In: *2016 11th Iberian Conference on Information Systems and Technologies (CISTI)*. [S.l.: s.n.], 2016. p. 1–6. Citado 2 vezes nas páginas 16 e 22.

BECK, K. *Extreme programming explained: embrace change*. [S.l.]: addison-wesley professional, 2000. Citado na página 26.

BECK, K. et al. *Manifesto for Agile Software Development*. 2001. Disponível em: <https://agilemanifesto.org/>. Citado na página 24.

BECK, K. et al. *Principles behind the Agile Manifesto*. 2001. Disponível em: <http://agilemanifesto.org/principles.html>. Citado na página 24.

BOHMAN, P. R. *Teaching accessibility and design-for-all in the information and communication technology curriculum: Three case studies of universities in the United States, England, and Austria*. [S.l.]: Utah State University, 2012. Citado na página 16.

BOURAOUI, A.; GHARBI, I. Model driven engineering of accessible and multi-platform graphical user interfaces by parameterized model transformations. *Science of computer programming*, Elsevier, v. 172, p. 63–101, 2019. Citado 2 vezes nas páginas 16 e 17.

CHIOU, P. T.; ALOTAIBI, A. S.; HALFOND, W. G. Bagel: An approach to automatically detect navigation-based web accessibility barriers for keyboard users. In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2023. (CHI '23). ISBN 9781450394215. Disponível em: <https://doi.org/10.1145/3544548.3580749>. Citado na página 31.

CRABB, M. et al. Developing accessible services: Understanding current knowledge and areas for future support. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. [S.l.: s.n.], 2019. p. 1–12. Citado 3 vezes nas páginas 16, 21 e 22.

FUGGETTA, A. Software process: a roadmap. In: *Proceedings of the Conference on the Future of Software Engineering*. [S.l.: s.n.], 2000. p. 25–34. Citado na página 17.

GONÇALVES, R. et al. Accessible software development: a conceptual model proposal. *Universal Access in the Information Society*, Springer, v. 18, n. 3, p. 703–716, 2019. Citado na página 22.

HIGHSMITH, J. *Adaptive software development: a collaborative approach to managing complex systems*. [S.l.]: Addison-Wesley, 2013. Citado na página 25.

HUMPHREY, W. S. *Managing the software process*. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 1989. Citado na página 17.

INAL, Y.; RIZVANOĞLU, K.; YESILADA, Y. Web accessibility in turkey: awareness, understanding and practices of user experience professionals. *Universal Access in the Information Society*, Springer, v. 18, n. 2, p. 387–398, 2019. Citado 2 vezes nas páginas 16 e 22.

KAWAS, S.; VONESSEN, L.; KO, A. J. Teaching accessibility: A design exploration of faculty professional development at scale. In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. New York, NY, USA: Association for Computing Machinery, 2019. (SIGCSE '19), p. 983–989. ISBN 9781450358903. Disponível em: <https://doi.org/10.1145/3287324.3287399>. Citado na página 21.

KITCHENHAM, B. A.; CHARTERS, S. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. [S.l.], 2007. Disponível em: <https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf>. Citado na página 27.

KLIRONOMOS, I. et al. White paper: Promoting design for all and e-accessibility in europe. *Universal Access in the Information Society*, Springer, v. 5, n. 1, p. 105–119, 2006. Citado na página 16.

KRóL, K.; ZDONEK, D. Local government website accessibility—evidence from poland. *Administrative Sciences*, v. 10, n. 2, 2020. ISSN 2076-3387. Disponível em: <https://www.mdpi.com/2076-3387/10/2/22>. Citado na página 31.

MIRANDA, D.; ARAUJO, J. a. Studying industry practices of accessibility requirements in agile development. In: *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. New York, NY, USA: Association for Computing Machinery, 2022. (SAC '22), p. 1309–1317. ISBN 9781450387132. Disponível em: <https://doi.org/10.1145/3477314.3507041>. Citado na página 17.

OSWAL, S. K.; PALMER, Z. B. Compliance, commitment, and web accessibility: Findings of an organizational study engaging accessibility experts at twenty u.s. universities. In: *Proceedings of the 42nd ACM International Conference on Design of Communication*. New York, NY, USA: Association for Computing Machinery, 2024. (SIGDOC '24), p. 17–25. ISBN 9798400705199. Disponível em: <https://doi.org/10.1145/3641237.3691647>. Citado na página 31.

PAIVA, D. M. B.; FREIRE, A. P.; FORTES, R. P. de M. Accessibility and software engineering processes: A systematic literature review. *Journal of Systems and Software*, Elsevier, v. 171, p. 110819, 2021. Citado 2 vezes nas páginas 22 e 23.

PARVIN, P. et al. The transparency of automatic accessibility evaluation tools. In: *Proceedings of the 18th International Web for All Conference (W4A '21)*. Ljubljana, Slovenia: ACM, 2021. Disponível em: <https://doi.org/10.1145/3430263.3452436>. Citado na página 67.

PELLEGRINI, F. et al. How to prioritize accessibility in agile projects. In: SPRINGER. *International Conference on Applied Human Factors and Ergonomics*. [S.l.], 2019. p. 271–280. Citado 3 vezes nas páginas 16, 17 e 26.

PRESSMAN, R. S. *Engenharia de Software: Uma abordagem profissional*. 7ª. ed. [S.l.]: McGraw-Hill, 2010. Citado na página 23.

SÁNCHEZ-GORDÓN, M.-L.; MORENO, L. Toward an integration of web accessibility into testing processes. *Procedia Computer Science*, Elsevier, v. 27, p. 281–291, 2014. Citado 2 vezes nas páginas 16 e 17.

SANCHEZ-GORDON, S. et al. Integration of accessibility design patterns with the software implementation process of iso/iec 29110. *Journal of Software: Evolution and Process*, Wiley Online Library, v. 31, n. 1, p. e1987, 2019. Citado 2 vezes nas páginas 16 e 17.

SCHWABER, K. *Agile project management with Scrum*. [S.l.]: Microsoft press, 2004. Citado na página 25.

SUTHERLAND, J.; SUTHERLAND, J. *Scrum: the art of doing twice the work in half the time*. [S.l.]: Currency, 2014. Citado na página 25.

TAFRESHIPOUR, M. et al. Ma11y: A mutation framework for web accessibility testing. In: *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*. New York, NY, USA: Association for Computing Machinery, 2024. (ISSTA 2024), p. 100–111. ISBN 9798400706127. Disponível em: <https://doi.org/10.1145/3650212.3652113>. Citado na página 31.

VIGO, M.; BROWN, J.; CONWAY, V. Benchmarking web accessibility evaluation tools: Measuring the harm of sole reliance on automated tests. In: *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility (W4A '13)*. ACM, 2013. Disponível em: <https://doi.org/10.1145/2461121.2461124>. Citado na página 67.

W3C. *Web Content Accessibility Guidelines (WCAG) 2.1*. 2018. [Acessado em 2. Feb. 2021]. Disponível em: <https://www.w3.org/TR/WCAG21/>. Citado 3 vezes nas páginas 16, 20 e 32.

W3C. *Acessibilidade na Web*. 2020. <https://www.w3.org/standards/webdesign/accessibility>. Citado na página 20.

W3C. *Introduction to Web Accessibility*. 2021. <https://www.w3.org/WAI/fundamentals/accessibility-intro/>. Citado na página 16.

WEBAIM. *WebAIM Million*. 2022. <https://webaim.org/projects/million/>. Citado na página 22.