Sistema de IA Multi-Agente com RAG para Análise de Acessibilidade Digital

Avaliação Automatizada de Heurísticas UX/UI em Aplicações Web e Mobile

Daniela Brazolin Flauto [1] Marcello Gonzatto Birkan [1]
Prof. Dr. Valéria Farinazzo Martins [1,2]

[1]Faculdade de Computação e Informática (FCI)
Universidade Presbiteriana Mackenzie São Paulo, SP — Brasil

<daniela.flauto,marcello.birkan@mackenzista.com.br>
<valeria.farinazzo@mackenzie.br>

2025

Resumo

A acessibilidade digital está relacionada a mais de 2,5 bilhões de pessoas que dependem de tecnologias assistivas (World Health Organization, 2022). Ferramentas convencionais de análise automatizada, baseadas em regras determinísticas, alcançam apenas 31,3% de cobertura em aplicações mobile e falham sistematicamente na detecção de violações semânticas. A conformidade multiplataforma exige conhecimento especializado de diretrizes distintas (WCAG 2.2, Apple HIG, WAI-ARIA), dificultando sua adoção em fluxos de desenvolvimento ágil. Este trabalho propõe o ***a11y-lint****, sistema multi-agente baseado em Large Language Models com Retrieval-Augmented Generation para análise contextualizada de acessibilidade cross-platform. A arquitetura vertical coordena subagentes especializados por plataforma (web, React, SwiftUI) via padrão ReAct, integrando análise determinística com avaliação semântica via LLM. O sistema RAG indexa diretrizes oficiais com metadados de rastreabilidade, fundamentando recomendações em documentação verificável. O protocolo de validação inclui comparação quantitativa com ferramentas estabelecidas, avaliação com especialistas em acessibilidade e testes de usabilidade com desenvolvedores. Contribuições esperadas: ferramenta open-source integrável a workflows de AI-assisted development, arquitetura multi-agente validada para análise de código e metodologia RAG com diretrizes citáveis.*

Palavras-chave: acessibilidade digital; large language models; sistemas multi-agente; retrieval-augmented generation; análise automatizada de código.

Abstract

Digital accessibility is related to more than 2.5 billion people who rely on assistive technologies (World Health Organization, 2022). Conventional automated analysis tools, based on deterministic rules, achieve only 31.3% coverage in mobile applications and systematically fail to detect semantic violations. Cross-platform compliance requires specialized knowledge of distinct guidelines (WCAG 2.2, Apple HIG, WAI-ARIA), hindering adoption in agile development workflows. This work proposes ***a11y-lint****, a multi-agent system based on Large Language Models with Retrieval-Augmented Generation for cross-platform contextualized accessibility analysis. The vertical architecture coordinates platform-specialized subagents (web, React, SwiftUI) via ReAct pattern, integrating deterministic analysis with semantic evaluation via LLM. The RAG system indexes official guidelines with traceability metadata, grounding recommendations in verifiable documentation. The validation protocol includes quantitative comparison with established tools, evaluation with accessibility experts, and usability testing with developers. Expected contributions: open-source tool integrable into AI-assisted development workflows, validated multi-agent architecture for code analysis, and RAG methodology with citable guidelines.*

Keywords: digital accessibility; large language models; multi-agent systems; retrieval-augmented generation; automated code analysis.

1 Introdução

A acessibilidade digital constitui requisito fundamental para a inclusão de mais de 2,5 bilhões de pessoas que necessitam de tecnologias assistivas globalmente, número que tende a crescer com o envelhecimento populacional (World Health Organization, 2022). Diretrizes como Web Content Accessibility Guidelines (WCAG) 2.2 (World Wide Web Consortium (W3C), 2024) e Apple Human Interface Guidelines (HIG) (Apple, 2024) estabelecem critérios técnicos para conformidade, porém sua aplicação prática enfrenta desafios significativos.

Ferramentas automatizadas convencionais de análise de acessibilidade como axe-core (Deque Labs, 2025) e Lighthouse (Google, 2025b) operam via regras determinísticas, detectando violações sintáticas (ausência de atributos obrigatórios, malformação estrutural) com alta precisão. Contudo, tais ferramentas alcançam apenas 31,3% de cobertura em aplicações mobile (ZHONG et al., 2025) e falham sistematicamente em avaliar adequação semântica: elementos sintaticamente corretos mas semanticamente inadequados, como `<img alt="imagem»` ou texto de link ambíguo “clique aqui”, passam despercebidos por verificações automáticas, porém constituem barreiras significativas para usuários de tecnologias assistivas.

Adicionalmente, a conformidade multiplataforma exige conhecimento especializado de diretrizes distintas e potencialmente conflitantes: WCAG 2.2 para web, Apple Human Interface Guidelines para iOS e WAI-ARIA para aplicações web acessíveis. Desenvolvedores enfrentam sobrecarga cognitiva ao navegar múltiplas fontes normativas, resultando em baixa adoção sistemática de práticas de acessibilidade em fluxos de trabalho de desenvolvimento ágil.

Paralelamente, Large Language Models (LLMs) demonstram capacidades emergentes promissoras para análise contextual de código. Zhong et al. (2025) reportam que abordagem baseada em GPT-4o alcança 69,2% de cobertura em detecção de problemas de

acessibilidade mobile, superando em 121% ferramentas convencionais (31,3%), com precisão validada por especialistas (71,3%). Fathallah, Hernández e Staab (2025) demonstram que análise híbrida, combinando verificação sintática determinística com avaliação semântica via LLM, reduz violation scores em 84%, alcançando similaridade de 0,77 com correções humanas. Contudo, trabalhos existentes apresentam limitações críticas: (i) escopo restrito à plataforma única (AccessGuru para *web*, ScreenAudit para mobile); (ii) ausência de arquiteturas multi-agente com especialização por domínio; (iii) falta de integração com Retrieval-Augmented Generation (RAG) para fundamentação verificável em diretrizes oficiais; (iv) *outputs* não estruturados para integração com ferramentas de AI-assisted development (Cursor, GitHub Copilot).

1.1 Problema de Pesquisa

Questão Principal: Como desenvolver um sistema multi-agente baseado em Large Language Models para análise automatizada de acessibilidade digital cross-platform que supere as limitações de ferramentas convencionais?

Questões Derivadas:

1. Como fundamentar análises em diretrizes oficiais verificáveis (WCAG 2.2, Apple HIG, WAI-ARIA) por meio de Retrieval-Augmented Generation com metadados de rastreabilidade?
2. Como integrar análise determinística (ferramentas existentes como axe-core (Deque Labs, 2025)) com análise contextual via LLM para maximizar cobertura sem redundância?
3. Como coordenar múltiplos agentes especializados por plataforma (web HTML/React, mobile SwiftUI) mantendo contextos isolados e especialização de domínio?
4. Como gerar outputs estruturados compatíveis com ferramentas de AI-assisted development (Cursor, GitHub Copilot) facilitando integração em workflows modernos?

1.2 Objetivo Geral

Desenvolver e validar um sistema multi-agent baseado em Large Language Models com Retrieval-Augmented Generation para análise automatizada de acessibilidade digital e heurísticas de UX/UI em aplicações web (HTML, React) e mobile (SwiftUI), gerando recomendações contextualizadas fundamentadas em diretrizes normativas oficiais.

1.3 Objetivos Específicos

1. Implementar arquitetura multi-agent com Claude Agent SDK, composta por agente orquestrador e subagentes especializados por plataforma (web HTML/React, mobile SwiftUI), com contextos isolados e permissões granulares de ferramentas.
2. Integrar sistema RAG com base de conhecimento estruturada contendo diretrizes de acessibilidade (WCAG 2.2, Apple HIG, WAI-ARIA, MDN Web Docs) com metadados de rastreabilidade (success criterion ID, URL canônico, excerpts textuais) para citação verificável.

3. Desenvolver pipeline híbrido combinando análise estática determinística (axe-core (Deque Labs, 2025), parsers AST) com análise contextual via LLM, avaliando ganhos em cobertura e precisão através de métricas quantitativas (precision, recall, F1-score).

4. Desenvolver e validar estratégias de prompt engineering incorporando heurísticas de Nielsen, Shneiderman e Norman, medindo impacto na qualidade das recomendações através de avaliação com especialistas (relevância, clareza, aplicabilidade).

5. Gerar outputs estruturados compatíveis com ferramentas de AI-assisted development (Cursor, GitHub Copilot), incluindo código corrigido, explicações contextualizadas e citações de diretrizes, validando integrabilidade através de testes com desenvolvedores.

6. Validar eficácia do sistema através de: (i) comparação quantitativa com ferramentas estabelecidas medindo precision, recall e F1-score; (ii) avaliação qualitativa com especialistas em acessibilidade analisando concordância inter-avaliadores; (iii) testes de usabilidade com desenvolvedores.

1.4 Justificativa e Relevância

Este trabalho justifica-se pela convergência de três fatores críticos: (i) crescente demanda social por inclusão digital de pessoas com deficiência e idosos; (ii) lacuna técnica em ferramentas de análise automatizada cross-platform com fundamentação verificável; (iii) emergência de paradigmas de desenvolvimento assistido por IA (vibe coding) que demandam integração com agentes autônomos.

1.4.0.1 Relevância Científica:

O trabalho contribui para o avanço do estado da arte em sistemas multi-agente baseados em LLMs ao: (i) validar arquitetura vertical orchestrator-workers para análise de código com especialização por domínio técnico; (ii) propor metodologia RAG transformando análise de código em conformidade auditável com rastreabilidade completa; (iii) documentar padrões de coordenação entre análise determinística e contextual via LLM.

1.4.0.2 Relevância Tecnológica:

A ferramenta a11y-lint oferece à comunidade de desenvolvimento: (i) análise unificada cross-platform (web e mobile) em sistema único, reduzindo fragmentação de ferramentas; (ii) outputs estruturados compatíveis com ferramentas de AI-assisted development (Cursor, GitHub Copilot).

1.4.0.3 Relevância Social:

A democratização da acessibilidade digital através de automação reduz barreiras técnicas e cognitivas para adoção de práticas inclusivas. Ao fornecer recomendações contextualizadas com citações verificáveis das principais fontes normativas, o sistema atua como ferramenta educacional, elevando conhecimento de desenvolvedores sobre padrões de acessibilidade. Espera-se impacto indireto na qualidade de vida de milhões de usuários de tecnologias assistivas através de interfaces digitais mais acessíveis.

1.5 Estrutura do Documento

Este artigo de TCC I está organizado em quatro seções principais. A Seção 2 apresenta o referencial teórico, dividido em dois eixos: (i) fundamentos de acessibilidade digital, diretrizes normativas e heurísticas de usabilidade; (ii) Large Language Models, capacidades emergentes, sistemas multi-agente e Claude Agent SDK. A Seção 3 descreve a metodologia de pesquisa proposta, detalhando tipo de estudo, arquitetura multi-agente, sistema RAG de diretrizes, pipeline de análise híbrida, estratégias de prompt engineering, protocolo de validação e ferramentas. A Seção 4 apresenta o cronograma detalhado de execução das atividades para o TCC II, organizado em cinco fases abrangendo o período de janeiro a maio de 2026. A Seção 5 apresenta as considerações finais, sintetizando a proposta, contribuições esperadas e próximos passos. Por fim, são apresentadas as referências bibliográficas que fundamentam esta proposta.

2 Referencial Teórico

Conforme estabelecido na introdução, o desenvolvimento do sistema *a11y-lint* demanda fundamentação em dois eixos complementares: (i) diretrizes normativas de acessibilidade digital e heurísticas de UX/UI que orientarão os critérios de análise; (ii) fundamentos de Large Language Models e sistemas multi-agente que embasam a arquitetura técnica proposta. Esta seção apresenta ambos os pilares teóricos e sintetiza sua integração conceitual.

2.1 Diretrizes de Acessibilidade

Desenvolvidas pelo World Wide Web Consortium (W3C), em colaboração com organizações de diversos países, as Diretrizes de Acessibilidade para Conteúdo Web (WCAG) 2.2 constituem um guia voltado à promoção da acessibilidade na web para uma ampla variedade de usuários. Além de contemplar pessoas com deficiência, as diretrizes também visam tornar o conteúdo mais acessível para pessoas idosas e usuários em diferentes contextos de uso. Suas recomendações não se restringem a tecnologias específicas, podendo ser aplicadas em diferentes contextos e plataformas (World Wide Web Consortium (W3C), 2024).

As WCAG 2.2 baseiam-se em quatro princípios fundamentais: perceptível, que assegura que as informações possam ser percebidas; operável, que garante a utilização dos elementos por todos os usuários; compreensível, que busca tornar o conteúdo claro e de fácil entendimento; e robusto, que mantém a compatibilidade com diferentes tecnologias e dispositivos (World Wide Web Consortium (W3C), 2024; World Wide Web Consortium (W3C), 2025).

Além disso, para atender às necessidades de diferentes grupos e situações, as diretrizes estabelecem três níveis de conformidade: A, AA e AAA, do mais básico ao mais avançado. O grupo recomenda que os sites adotem a versão WCAG 2.2 como meta de conformidade, com o objetivo de aprimorar a acessibilidade e antecipar futuras mudanças nas políticas da área (World Wide Web Consortium (W3C), 2024).

Complementarmente às diretrizes W3C voltadas para conteúdo web, a Apple estabelece suas diretrizes de design por meio do documento *Human Interface Guidelines* (HIG), que orienta o desenvolvimento de interfaces para todas as suas plataformas, visando proporcionar uma experiência consistente e acessível aos usuários. Os princípios de design

da Apple enfatizam três pilares fundamentais: a hierarquia, que garante uma organização visual clara destacando o conteúdo principal; a harmonia, que busca integrar visualmente hardware, software e interface; e a consistência, que assegura que o design siga padrões da plataforma e se adapte a diferentes contextos de uso e tamanhos de tela (Apple, 2024).

A escolha do termo "Human Interface" em vez de "User Interface" reflete uma abordagem centrada no ser humano, considerando não apenas a interação funcional com sistemas, mas também as necessidades emocionais, cognitivas e práticas das pessoas (Apple, 2017). No vídeo *Essential Design Principles*, apresentado durante a WWDC 2017, a Apple enfatiza que o design de interfaces deve priorizar o bem-estar e a experiência completa do indivíduo, afirmando que "projetar uma interface é fundamentalmente sobre servir outros seres humanos. A única coisa que realmente importa é quão bem seu aplicativo satisfaz as necessidades emocionais e práticas das pessoas para as quais você está projetando" (Apple, 2017).

Além das diretrizes gerais de acessibilidade e design, existe a necessidade de especificações técnicas para implementação prática. Nesse contexto, a especificação *Web Accessibility Initiative – Accessible Rich Internet Applications* (WAI-ARIA), desenvolvida também pelo W3C, tem como objetivo definir um conjunto de atributos HTML para melhorar a acessibilidade e complementar a semântica de elementos web. Esses atributos são organizados em três categorias principais: Funções (*Roles*), que definem o papel ou a função de um elemento na interface; Propriedades, que adicionam informações complementares sobre suas características; e Estados, que indicam sua condição atual, podendo mudar dinamicamente conforme a interação do usuário (Mozilla Developer Network, 2024a).

O WAI-ARIA não altera a apresentação visual da página, atuando exclusivamente no nível da comunicação entre o navegador e as tecnologias assistivas, por meio das APIs de acessibilidade. Assim, permite que navegadores e tecnologias assistivas interpretem de maneira mais precisa a função e o estado dos componentes. O atributo de função define o tipo de elemento na interface, como um artigo, alerta ou controle deslizante, enquanto outros atributos ARIA adicionam informações complementares, como descrições ou valores exibidos em componentes interativos (Mozilla Developer Network, 2024a; Mozilla Developer Network, 2024b).

Além das diretrizes formais elaboradas pelo W3C e das recomendações de design da Apple, frameworks modernos de desenvolvimento front-end também fornecem orientações específicas para implementação de acessibilidade. A documentação oficial do React, por exemplo, enfatiza a importância do HTML semântico, o suporte nativo a atributos WAI-ARIA em JSX, o gerenciamento programático de foco do teclado e o uso de ferramentas de teste de acessibilidade (React, 2024). Essas práticas contribuem para que componentes dinâmicos e interativos desenvolvidos com React mantenham conformidade com os princípios estabelecidos pela WCAG e WAI-ARIA.

2.2 Tecnologias Assistivas

As diretrizes de acessibilidade apresentadas nas subseções anteriores fundamentam-se na existência e no funcionamento de tecnologias assistivas, que são os meios pelos quais pessoas com deficiência acessam conteúdo digital.

De acordo com a Organização Mundial da Saúde (World Health Organization, 2016), as tecnologias assistivas compreendem um conjunto de produtos, equipamentos e recursos desenvolvidos para promover a inclusão e a participação social de pessoas com deficiência,

idosos e indivíduos com mobilidade reduzida. Essas tecnologias têm como objetivo auxiliar no cuidado ou no aprimoramento das habilidades funcionais e da autonomia, contribuindo também para a prevenção de deficiências e de outras condições relacionadas à saúde. Estima-se que mais de 2,5 bilhões de pessoas em todo o mundo necessitam de algum tipo de tecnologia assistiva, número que tende a crescer com o envelhecimento populacional (World Health Organization, 2022).

As tecnologias assistivas podem ser categorizadas de acordo com o tipo de necessidade que atendem. Para pessoas com deficiência visual, destacam-se os leitores de tela, como o NVDA, JAWS e VoiceOver (Apple), que fornecem descrições auditivas do conteúdo exibido na tela, possibilitando a navegação sem a necessidade de visualização. Os ampliadores de tela aumentam textos e elementos visuais, podendo incluir recursos como leitura em voz alta e inversão de cores para facilitar a leitura (Level Access, 2024).

No contexto de deficiências motoras, ferramentas como o Voice Control (Apple) permitem que o usuário navegue e execute ações por meio de comandos de voz, sem precisar utilizar as mãos. O Switch Control (Apple) possibilita a interação com dispositivos por meio de interruptores físicos ou controladores adaptativos, sendo especialmente útil para pessoas com mobilidade reduzida. Sistemas operacionais como iOS, Android e Windows oferecem recursos similares, adaptados às suas respectivas plataformas (Apple Inc., 2025; Google, 2025a).

Para pessoas com deficiências cognitivas, foram desenvolvidas interfaces simplificadas, como o Assistive Access da Apple, disponível para iOS e iPadOS, que adapta a interface do sistema para torná-la mais intuitiva e acessível. Navegadores como o Google Chrome também oferecem modos de leitura que reduzem distrações visuais, apresentando o conteúdo de forma mais clara e objetiva (Apple Inc., 2025; Google, 2025a).

No que diz respeito à deficiência auditiva, os tradutores automáticos de conteúdo, como o Hand Talk e o VLibras, tornam a comunicação digital mais acessível por meio da tradução de textos e áudios para a Língua Brasileira de Sinais (Libras) em tempo real (Hand Talk, 2025; Brasil. Governo Federal, 2025). Essas ferramentas são especialmente relevantes para a inclusão digital da comunidade surda no Brasil.

2.3 Heurísticas de UX

As heurísticas de usabilidade são princípios gerais que orientam o design de interfaces. Segundo Nielsen e Molich (NIELSEN; MOLICH, 1990), essas diretrizes funcionam como regras de "bom senso" que ajudam a criar sistemas mais intuitivos e fáceis de usar, caracterizando-se por serem orientações amplas e não regras específicas e rígidas.

A avaliação de interfaces de usuário pode ser realizada de diferentes formas: a avaliação formal, que utiliza técnicas específicas de análise; a avaliação automática, realizada com auxílio de computadores; a avaliação empírica, conduzida por meio de testes com usuários reais; e a avaliação heurística, baseada na observação da interface e no julgamento fundamentado do avaliador (NIELSEN; MOLICH, 1990).

Embora os testes com usuários reais sejam considerados ideais, eles acabam sendo pouco utilizados na prática devido a fatores como falta de tempo, recursos ou até mesmo do hábito, fazendo com que a maioria das avaliações seja heurística. Os autores argumentam que, apesar desse método ter sido historicamente menos valorizado pela academia, faz mais sentido estudar e aprimorar justamente os métodos que realmente serão usados no cotidiano das empresas (NIELSEN; MOLICH, 1990).

Nesse contexto, Nielsen estabeleceu dez heurísticas de usabilidade que abordam aspectos fundamentais do design de interfaces, incluindo comunicação clara com o usuário, prevenção e recuperação de erros, consistência de padrões, flexibilidade de uso e simplicidade visual (NIELSEN, 1994).

Além das heurísticas de Nielsen, Ben Shneiderman e Don Norman são referências importantes no campo da usabilidade, oferecendo abordagens complementares para o design de interfaces.

Primeiramente, Shneiderman, através de suas Regras de Ouro, propõe princípios que orientam a criação de sistemas consistentes, acessíveis a diferentes perfis de usuários e capazes de fornecer feedback claro. Entre os conceitos centrais estão a prevenção de erros, a possibilidade de reverter ações, o controle pelo usuário e a redução da carga de memória de curto prazo (SHNEIDERMAN et al., 2016).

Já Don Norman é referência central no estudo de design centrado no usuário e apresenta princípios para a criação de produtos e interfaces mais intuitivas. Seus conceitos abordam como os usuários percebem e interagem com objetos e sistemas, focando em aspectos que facilitam a compreensão do uso, a comunicação visual de ações possíveis e o retorno informativo das interações. Essas ideias orientam o design de forma a reduzir erros e proporcionar uma experiência mais fluida (NORMAN, 2013).

Estabelecidos os fundamentos normativos de acessibilidade digital e as heurísticas de usabilidade que orientam os critérios de análise, esta subseção aborda as tecnologias de inteligência artificial que viabilizam a automação contextualizada: Large Language Models, capacidades emergentes, arquiteturas multi-agente e infraestrutura do Claude Agent SDK.

2.4 Fundamentos de Large Language Models e Capacidades Emergentes

Large Language Models (LLMs) são modelos de aprendizado profundo baseados na arquitetura Transformer (VASWANI et al., 2017), que permite processamento paralelo de sequências e modelagem de dependências de longo alcance, essencial para análise de código. A evolução recente caracteriza-se por expansão dramática de janelas de contexto: de 2.048 tokens (GPT-3, 2020) para 200.000 tokens (Claude Sonnet 4.5, 2025) (MINAEE et al., 2024), viabilizando processamento de arquivos extensos sem fragmentação. O Claude Agent SDK implementa **compaction** nativo (sumarização automática de histórico) e **agentic search** (recuperação just-in-time via ferramentas) para gestão eficiente de contexto (Anthropic, 2025).

Três capacidades emergentes fundamentam aplicações agênticas modernas: (i) *function calling*, permitindo invocação autônoma de APIs externas – Schick et al. (2023) demonstram ganhos expressivos ao integrar calculadoras (*Toolformer* dobra acurácia matemática); (ii) *Retrieval-Augmented Generation* (RAG), ancorando respostas em documentos recuperados – Lewis et al. (2020) reportam 54% de melhoria em acurácia ao reduzir alucinações via *grounding*; (iii) raciocínio multi-etapas via *chain-of-thought* (CoT), decompondo problemas em subtarefas verificáveis.

Para análise de acessibilidade, RAG é particularmente crítico: indexando diretrizes oficiais (WCAG 2.2, Apple HIG, ARIA APG, MDN) com metadados estruturados, permite citação precisa de critérios normativos, transformando recomendações de "sugestões do modelo" para "conformidade verificável com padrões" e garantindo rastreabilidade a versões específicas (e.g., WCAG 2.2 vs 2.1).

2.5 Sistemas Multi-Agente: Arquiteturas, Coordenação e Infraestrutura

2.5.1 Framework Conceitual de Agentes Autônomos

Wang et al. (2025) consolidam arquitetura de agentes autônomos baseados em LLMs através de quatro módulos essenciais: (i) *Profile* (papel, objetivos e restrições do agente); (ii) *Memory* (memória de curto prazo, contexto corrente, e longo prazo, conhecimento persistente via RAG ou bancos vetoriais); (iii) *Planning* (decomposição de tarefas via *chain-of-thought* ou planejamento hierárquico); (iv) *Action* (execução via *function calling*, gerando consultas de *retrieval*, invocando APIs ou produzindo código). Esta modularização orienta implementações: cada subagente especializado (HTML Scanner, React Linter, SwiftUI Checker, RAG Retriever) instancia esses quatro componentes com configurações específicas. O *profile* define domínio de competência (e.g., "especialista em contraste WCAG 1.4.3"), *memory* mantém contexto corrente e conhecimento persistente via RAG, *planning* determina estratégia de varredura (AST *traversal*, análise semântica contextual), *action* executa ferramentas determinísticas (calculadora de contraste, validadores ARIA) e consulta RAG para citações.

2.5.2 O Paradigma ReAct: Integração Raciocínio-Ação-Observação

O paradigma *ReAct* (*Reasoning and Acting*) integra cadeias de raciocínio com ações intercaladas e observações de ferramentas externas em um loop iterativo: *Thought* → *Action* → *Observation* (YAO et al., 2023). Em verificação factual (benchmark *Fever*), ReAct alcançou 60,9% de acurácia versus 56,3% de *chain-of-thought* isolado, reduzindo alucinações de 14% para 6%, uma diminuição relativa de 57%, ao fundamentar raciocínio em observações verificáveis de ferramentas externas (YAO et al., 2023). Este ciclo permite que agentes: (i) decomponham tarefas em etapas verificáveis (*thought*: "necessário verificar contraste entre cor de texto #333 e fundo #fff"); (ii) invoquem ferramentas para obter dados precisos (*action*: executar calculadora de contraste WCAG); (iii) autocorrijam quando observações indicam erros (*observation*: resultado da verificação indica conformidade ou violação).

Para análise de acessibilidade, o padrão ReAct viabiliza verificações iterativas contextuais: o agente raciocina sobre qual critério WCAG aplicar baseado em contexto semântico do componente UI (*thought*), invoca ferramenta de análise estrutural ou cálculo quantitativo (*action*), observa resultado e decide se há violação, solicitando *retrieval* de diretriz oficial caso necessário (*observation* → nova *action* de consulta RAG) antes de emitir diagnóstico final com citação verificável.

2.5.3 Coordenação Multi-Agente: Padrões Arquiteturais e Ganhos Empíricos

He, Treude e Lo (2025) documentam padrão *orchestrator-workers* em engenharia de software com papéis especializados (*Orchestrator*, *Programmer*, *Reviewer*, *Tester*, *Info Retriever*), alcançando 87,3% de taxa de sucesso em execução de software gerado no *ChatDev*. A especialização por papel permite otimização independente de *prompts* e ferramentas por domínio.

Masterman et al. (2024) distinguem arquiteturas com liderança designada (*vertical*: orquestrador coordena workers) de colaboração horizontal (agentes pares negociam), evidenciando que arquiteturas verticais completam tarefas 10% mais rápido ao reduzir overhead de coordenação. Para o a11y-lint, arquitetura vertical com orquestrador coordenando especialistas por plataforma (HTML/React/SwiftUI) permite análise paralela, isolamento

de contexto e integração via RAG compartilhado de diretrizes normativas, facilitando agregação de resultados.

2.5.4 Infraestrutura: Claude Agent SDK e Model Context Protocol

O Claude Agent SDK (Anthropic, 2025) fornece primitivas para construção de agentes autônomos implementando o loop *Gather Context* → *Take Action* → *Verify Work*. Suas capacidades essenciais incluem: (i) subagentes com contextos isolados, cada um com *prompt* otimizado e ferramentas específicas, implementando o padrão *orchestrator-workers*; (ii) integração com Model Context Protocol (MCP) para ferramentas determinísticas (*RAG search*, parsers AST, validadores WCAG/HIG, calculadoras de contraste); (iii) *compaction* automático que sumariza histórico ao aproximar-se do limite de contexto, permitindo análises extensas sem degradação.

Para o a11y-lint, o SDK viabiliza arquitetura onde orquestrador coordena especialistas em paralelo via loop *Gather-Act-Verify*: coleta contexto via análise estrutural e RAG, delega a subagentes com ferramentas determinísticas, e valida consistência antes de emitir diagnóstico com citações verificáveis, combinando análise contextual LLM com verificação determinística.

2.6 LLMs para Análise Automatizada de Código e Acessibilidade

Ferramentas tradicionais (*axe-core* (Deque Labs, 2025), *Lighthouse* (Google, 2025b)) operam via regras determinísticas, detectando violações sintáticas com alta precisão mas falhando em avaliar adequação semântica (FATHALLAH; HERNáNDEZ; STAAB, 2025). Para aplicações móveis, ferramentas convencionais alcançam apenas 31,3% de cobertura versus 69,2% de abordagens baseadas em LLM (ZHONG et al., 2025). A limitação fundamental reside na incapacidade de avaliar adequação semântica: `<img alt="imagem">` passa em verificações sintáticas mas é inútil para leitores de tela.

LLMs superam esta barreira através de compreensão contextual. *AccessGuru* (FATHALLAH; HERNáNDEZ; STAAB, 2025) combina análise determinística com GPT-4o multimodal, alcançando 84% de redução em *violation scores* e 96% de resolução de violações semânticas. *ScreenAudit* (ZHONG et al., 2025) demonstra ganhos em aplicações móveis, detectando categorias negligenciadas como qualidade de rótulos e ordem de navegação.

Fathallah, Hernández e Staab (2025) propõem taxonomia tri-dimensional: (i) violações sintáticas (elementos ausentes/malformados); (ii) violações semânticas (conteúdo inadequado, *alt* genérico, links ambíguos); (iii) violações de layout (contraste insuficiente, *touch targets* inadequados). Esta classificação orienta arquitetura híbrida: verificações determinísticas para sintaxe, análise LLM para semântica, RAG para citação verificável de critérios normativos.

A eficácia de abordagens baseadas em LLM atribui-se a estratégias avançadas de *prompting*: *metacognitive prompting* (estruturação do raciocínio em estágios explícitos) e *corrective re-prompting* (refinamento iterativo baseado em *feedback* determinístico), alcançando 84% de redução em *violation scores* no *AccessGuru* (FATHALLAH; HERNáNDEZ; STAAB, 2025).

2.7 Síntese e Integração Conceitual

Este referencial estabeleceu três pilares convergentes: (i) capacidades emergentes de LLMs (*function calling*, RAG, *chain-of-thought*) materializadas pelo Claude Agent SDK; (ii) sistemas multi-agente com padrões validados como ReAct (reduz alucinações) e *orchestrator-workers* (especialização por domínio); (iii) aplicações demonstrando superioridade de abordagens híbridas combinando verificação determinística com análise contextual LLM.

2.7.1 Análise Comparativa do Estado da Arte

A Tabela 1 sintetiza trabalhos relacionados, evidenciando lacunas que o sistema proposto visa endereçar. Ferramentas tradicionais (*axe-core* (Deque Labs, 2025), Lighthouse (Google, 2025b)) apresentam cobertura limitada em problemas semânticos. Trabalhos recentes com LLMs demonstram ganhos substanciais: Zhong et al. (2025) reportam 69,2% versus 31,3% de cobertura em mobile; Fathallah, Hernández e Staab (2025) alcançam 84% de redução em *violation scores*.

Tabela 1 – Análise comparativa de sistemas de análise de acessibilidade

Sistema	**Plataformas**	**Arquitetura**	**RAG Citável**	**Análise Híbrida**
axe-core	Web	Regras determ.	Não	Não
Lighthouse	Web	Regras determ.	Não	Não
AccessGuru (FATHALLAH; HERNáNDEZ; STAAB, 2025)	Web (HTML)	LLM único	Não	Sim
ScreenAudit (ZHONG et al., 2025)	Mobile (Android)	LLM único	Não	Sim
a11y-lint (proposto)	**Web + React + SwiftUI**	**Multi-agent orq.-workers**	**Sim**	**Sim**

Trabalhos existentes apresentam três limitações: (i) escopo monoplatforma; (ii) ausência de arquiteturas multi-agente com especialização por domínio; (iii) falta de RAG para fundamentação verificável. Nenhum sistema analisado aborda especificamente iOS/SwiftUI. Esta análise posiciona o a11y-lint como primeira abordagem multi-agente *cross-platform* com RAG citável, preenchendo lacuna metodológica na interseção de sistemas multi-agente, análise de acessibilidade e recomendações verificáveis.

3 Metodologia

Fundamentado no referencial teórico que estabeleceu os conceitos de LLMs, sistemas multi-agente, RAG e diretrizes de acessibilidade, esta seção descreve a metodologia de pesquisa proposta para desenvolvimento e validação do sistema *a11y-lint*. A abordagem integra os pilares conceituais apresentados em uma arquitetura prática de análise automatizada cross-platform.

Esta seção descreve o tipo de pesquisa, arquitetura do sistema a ser desenvolvido, sistema RAG, pipeline de análise, estratégias de validação e ferramentas utilizadas.

3.1 Tipo de Pesquisa e Abordagem Metodológica

Pesquisa experimental-comparativa de natureza mista (quanti-qualitativa), orientada ao desenvolvimento e validação de um sistema de software inovador. Fundamenta-se em dois componentes: (1) desenvolvimento de artefato tecnológico multi-agent com Claude Agent SDK; (2) validação experimental via comparação com ferramentas estabelecidas e avaliação com especialistas em acessibilidade.

A abordagem mista justifica-se pela necessidade de combinar métricas quantitativas com avaliações qualitativas de especialistas sobre adequação contextual das recomendações. Esta triangulação metodológica fortalece a validade dos resultados.

3.2 Arquitetura Multi-Agent do Sistema a11y-lint

O sistema *a11y-lint* implementa arquitetura multi-agent vertical hierárquica com agente orquestrador e subagentes especializados: Web Scanner (HTML), React Linter, SwiftUI Checker, e agente RAG.

A arquitetura vertical é fundamentada em Masterman et al. (2024), que demonstra que equipes com liderança designada completam tarefas 10% mais rapidamente com menor overhead de comunicação. O ciclo de análise de cada agente segue o padrão ReAct (YAO et al., 2023): Gather (recuperar contexto via RAG) → Act (executar ferramentas) → Verify (validar com base de conhecimento), reduzindo alucinações de 14% para 6%.

3.2.1 Detalhamento Arquitetural e Fluxo de Dados

O sistema *a11y-lint* implementa uma arquitetura multi-agent hierárquica com orquestrador central e subagentes especializados por plataforma (Web, React, SwiftUI), inspirada em padrões documentados por Masterman et al. (2024) e He, Treude e Lo (2025).

O fluxo geral segue o padrão ReAct (YAO et al., 2023):

1. **Entrada:** Developer submete path do arquivo via CLI.
2. **Classificação:** Orquestrador detecta plataforma (extensão de arquivo, syntax hints) e encaminha para subagente apropriado.
3. **Análise:** Subagente raciocina sobre o código, consulta guidelines relevantes via RAG, executa ferramentas determinísticas quando disponíveis, gera recomendações.
4. **Consolidação:** Orquestrador agrega outputs dos subagentes ativos, resolve discordâncias e retorna relatório em formato HTML ou Markdown.

O sistema combina análise determinística (axe-core (Deque Labs, 2025), parsers de linguagem) com análise semântica via LLM, permitindo detecção tanto de violações estruturais quanto de problemas contextuais. Subagentes especializados consultam base de conhecimento (WCAG 2.2, Apple HIG, ARIA APG, MDN) via RAG e geram recomendações estruturadas com código corrigido, explicação contextualizada e citação verificável da diretriz oficial.

3.3 Sistema RAG de Guidelines

O sistema utiliza Retrieval-Augmented Generation (RAG) para recuperar guidelines relevantes durante a análise. Base de conhecimento contém WCAG 2.2, Apple HIG, ARIA APG e MDN Web Docs, indexados semanticamente.

Quando um subagente detecta uma potencial violação, consulta o RAG para recuperar critérios e diretrizes aplicáveis. Cada recomendação inclui rastreabilidade (ID do critério, URL canônica, excerto textual), garantindo fundamentação em diretrizes oficiais verificáveis. Esta abordagem reduz alucinações via grounding em documentos recuperados (LEWIS et al., 2020).

Como trabalho futuro, planeja-se implementar funcionalidade de customização da base de conhecimento, permitindo que desenvolvedores selecionem especificamente quais guidelines desejam aplicar na análise (e.g., apenas WCAG 2.2 nível AA, ou combinação de WCAG + ARIA), ou optem por um conjunto padrão predefinido. Esta flexibilidade atenderia diferentes contextos de projeto e requisitos de conformidade, adaptando-se a necessidades específicas de equipes de desenvolvimento.

3.4 Pipeline de Análise de Código

Implementa abordagem híbrida combinando análise determinística e semântica:

Para aplicações *web* (HTML, *React*):

- Análise determinística via *axe-core* (Deque Labs, 2025) para detecção automática de violações estruturais de acessibilidade.
- Análise contextual via LLM com RAG para avaliação semântica de acessibilidade, adequação de *labels*, fluxo de navegação, UX/UI e heurísticas de usabilidade.
- União dos resultados de ambas as análises gerando recomendações consolidadas.

Para aplicações *mobile* (*SwiftUI*):

- Análise contextual via LLM com RAG para avaliação de acessibilidade conforme *Apple HIG*, UX/UI e heurísticas de usabilidade, sem análise determinística (ferramentas equivalentes ao *axe-core* (Deque Labs, 2025) não estão disponíveis para *SwiftUI*).

Esta abordagem é inspirada em Zhong et al. (2025), que demonstra que sistemas LLM puros alcançam 69.2% de cobertura vs 31.3% de ferramentas automatizadas, mas a combinação de ambas maximiza *precision* e *recall*. Por exemplo, análise determinística detecta ausência de atributos obrigatórios, enquanto análise semântica identifica conteúdo inadequado (e.g., `alt` genérico em gráfico informativo).

3.5 Prompt Engineering

O sistema utiliza prompts com heurísticas gerais de acessibilidade e usabilidade (Nielsen, Shneiderman, Norman) para análise contextualizada de código. Outputs seguem schema compatível com ferramentas de AI-assisted development.

3.6 Dataset de Validação

O dataset de validação será composto por 10 aplicações: 5 desenvolvidas em React (JSX/TSX) para plataforma web e 5 em SwiftUI para iOS. Esta distribuição permite avaliar a eficácia do sistema em diferentes paradigmas de desenvolvimento (componentes reativos web e UI declarativa mobile).

As aplicações serão selecionadas ou desenvolvidas para cobrir sistematicamente os três níveis de conformidade WCAG (A, AA, AAA) e distribuição balanceada de tipos de violação conforme taxonomia de Fathallah, Hernández e Staab (2025): sintáticas (ausência ou malformação de elementos obrigatórios), semânticas (conteúdo inadequado ou ambíguo), e layout (contraste insuficiente, dimensões inadequadas).

O ground truth será estabelecido através de análise colaborativa com especialistas em acessibilidade, documentando problemas reais com suas respectivas classificações WCAG, severidades e localizações precisas no código. Cada aplicação conterá diversidade de componentes UI incluindo formulários, navegação, conteúdo multimídia e widgets interativos, permitindo avaliação abrangente da capacidade de detecção do sistema.

3.7 Validação do Sistema

A validação será conduzida através de múltiplas perspectivas complementares, combinando avaliação técnica quantitativa, comparação com ferramentas estabelecidas, avaliação qualitativa com especialistas e testes de usabilidade com desenvolvedores.

3.7.0.1 Avaliação Técnica Quantitativa.

O sistema será executado sobre o dataset descrito anteriormente, com detecções comparadas ao ground truth estabelecido. Métricas padrão de classificação serão calculadas:

- **Precision (Precisão):** Proporção de violations corretamente identificadas dentre todas as detecções
- **Recall (Cobertura):** Proporção de violations reais efetivamente detectadas pelo sistema
- **F1-Score:** Média harmônica de precision e recall
- **Citation Fidelity:** Percentual de recomendações com citações corretas e verificáveis de diretrizes

Métricas serão calculadas globalmente e estratificadas por categoria de violação e por plataforma, permitindo identificar pontos fortes e limitações do sistema.

3.7.0.2 Comparação com Ferramentas Estabelecidas.

Comparação head-to-head com ferramentas baseline de análise de acessibilidade será conduzida sobre o mesmo dataset: para aplicações React, serão utilizadas ferramentas estabelecidas como axe-core (Deque Labs, 2025), Lighthouse (Google, 2025b) e WAVE; para SwiftUI, a comparação será limitada devido à ausência de ferramentas automatizadas maduras no ecossistema iOS.

As métricas de precision, recall e F1-score do a11y-lint serão comparadas com os baselines estabelecidos. Testes estatísticos não-paramétricos serão aplicados considerando o tamanho amostral limitado, com intervalos de confiança calculados via bootstrap. Tamanho de efeito será reportado para contextualizar a magnitude prática das diferenças observadas além de significância estatística.

3.7.0.3 Avaliação Qualitativa com Especialistas.

Especialistas em acessibilidade digital avaliarão amostra representativa de issues reportadas pelo sistema através de protocolo estruturado. Dimensões avaliadas incluirão: correção técnica da detecção, adequação contextual da recomendação, clareza e aplicabilidade da mensagem, e pertinência da prioridade atribuída. Cada dimensão será avaliada através de escalas apropriadas.

A concordância entre avaliadores independentes será medida através de coeficientes estatísticos adequados para verificar consistência das avaliações. Análise descritiva incluirá distribuições de scores por dimensão e categoria de violação. Feedback qualitativo aberto será coletado para identificar padrões de forças e fragilidades do sistema não capturados por escalas quantitativas.

3.7.0.4 Testes de Usabilidade com Desenvolvedores.

Desenvolvedores com experiência em React ou SwiftUI participarão de sessões práticas de correção de issues detectados pelo a11y-lint. As sessões seguirão protocolo controlado onde participantes utilizarão os reports gerados para aplicar correções em aplicações do dataset, enquanto pesquisadores observam e coletam métricas.

Métricas quantitativas incluirão: tempo médio de correção por issue, taxa de aceitação das recomendações automáticas, taxa de sucesso nas correções aplicadas, e satisfação geral com a ferramenta medida através de questionário padronizado. Dados qualitativos serão obtidos através de observações estruturadas durante as sessões e entrevistas semi-estruturadas ao final, identificando pontos de atrito, aspectos positivos percebidos e sugestões de melhoria. Análise temática será aplicada aos dados qualitativos para identificar padrões emergentes.

3.7.0.5 Considerações Éticas.

Todos os testes envolvendo participantes humanos seguirão protocolo aprovado pelo Comitê de Ética em Pesquisa (CEP) da instituição, incluindo Termo de Consentimento Livre e Esclarecido, garantia de anonimato e confidencialidade dos dados, direito de retirada sem penalização, e armazenamento seguro com acesso restrito aos pesquisadores.

3.8 Ferramentas e Tecnologias

A implementação do sistema a11y-lint requer stack tecnológica integrando infraestrutura de agentes, análise determinística e RAG. A Tabela 2 sintetiza as tecnologias principais e suas justificativas.

A escolha da stack RAG (bibliotecas de embeddings, banco vetorial) será definida durante implementação após avaliação comparativa considerando qualidade de retrieval, latência e custos. Ferramentas determinísticas serão integradas via MCP tools customizadas, permitindo análise híbrida (sintática + semântica) com resultados agregados.

Tabela 2 – Stack tecnológica do sistema a11y-lint

Categoria	**Tecnologia**	**Justificativa/Uso**
Infraestrutura LLM	Claude Agent SDK (Python)	Coordenação multi-agente, compaction, integração MCP
Modelos LLM	Claude Sonnet 4.5 / Haiku 4.5	Sonnet: análise contextual complexa; Haiku: tarefas estruturais (roteamento, agregação)
Linguagem	Python 3.11+	Ecossistema maduro para NLP, integração APIs
Interface	Textual (TUI)	Interação rica via terminal, navegação de issues
Sistema RAG	Indexação vetorial (a definir)	Diretrizes estruturadas com metadados rastreáveis (WCAG, HIG, ARIA, MDN)
Análise Determinística	axe-core (Deque Labs, 2025), AST parsers	axe-core: violations sintáticas web; Parsers JS/TS e Swift: análise estrutural
Controle de Versão	Git/GitHub	Versionamento, documentação Markdown, rastreamento de issues

3.9 Implementação da Arquitetura Multi-Agent

A arquitetura será implementada utilizando o Claude Agent SDK com os seguintes componentes principais:

3.9.0.1 Agente Orquestrador.

Agente central responsável por receber input do usuário via TUI Textual, classificar arquivos por plataforma através de análise de extensão e sintaxe, delegar análises aos subagentes especializados apropriados, e agregar resultados em relatório consolidado. Utilizará Claude Haiku 4.5 para decisões de roteamento e agregação por serem tarefas estruturais de menor complexidade.

3.9.0.2 Subagentes Especializados.

Três subagentes principais com contextos isolados e ferramentas específicas: (i) Web Scanner para HTML puro, com acesso a ferramentas de parsing DOM e validadores WCAG sintáticos; (ii) React Linter para componentes JSX/TSX, integrando análise AST e axe-core via MCP tools customizadas; (iii) SwiftUI Checker para interfaces iOS, com ferramentas de parsing de código Swift e análise exclusivamente contextual via LLM. Todos utilizarão Claude Sonnet 4.5 para análise semântica profunda, consultando o sistema RAG através de ferramentas MCP dedicadas que retornam trechos relevantes de diretrizes com metadados completos.

3.9.0.3 Integração de Ferramentas via MCP.

O Model Context Protocol será utilizado para expor ferramentas customizadas aos agentes: RAG search tools para consulta de diretrizes oficiais, axe-core (Deque Labs, 2025) wrapper para execução de análise determinística, AST parsers para navegação estrutural de código, e calculadoras especializadas para verificações quantitativas (contraste, dimensões). Cada subagente terá whitelist restrita às ferramentas necessárias para seu domínio, evitando confusão e reduzindo surface de possíveis erros.

3.9.0.4 Gestão de Contexto e Outputs.

O SDK gerenciará automaticamente compaction de contexto durante análises extensas, preservando decisões arquiteturais relevantes enquanto descarta raciocínio intermediário redundante. Subagentes retornarão estruturas de dados padronizadas (violations com localização, severidade, critério WCAG, recomendação, código corrigido, citação verificável) que serão agregadas pelo orquestrador e renderizadas pela TUI Textual em formatos navegáveis e exportáveis.

4 Resultados Parciais do Desenvolvimento

Durante a fase de planejamento do TCC I, desenvolveu-se protótipo funcional (MVP) do sistema *a11y-lint* que materializou parcialmente os conceitos teóricos (Seção 2) e a arquitetura proposta (Seção 3), validando viabilidade técnica da abordagem multi-agente para análise contextualizada de acessibilidade. A implementação atual consiste em sistema completo incluindo arquitetura multi-agente hierárquica, codificação de diretrizes em prompts, pipeline híbrido de análise e interface terminal interativa. A Tabela 3 apresenta métricas gerais evidenciando sistema testado e funcional.

Tabela 3 – Métricas gerais de implementação do sistema a11y-lint

Métrica	**Valor**
Módulos funcionais implementados	6
Agentes especializados implementados	4
Regras de acessibilidade codificadas	30+
Testes automatizados (taxa de sucesso)	25 (100%)
Custo por scan (30 arquivos)	$0,30
Tempo de scan (30 arquivos)	2-3min

4.1 Arquitetura Multi-Agente Implementada

A arquitetura materializa o padrão orchestrator-workers (MASTERMAN et al., 2024; HE; TREUDE; LO, 2025) com orquestrador central coordenando quatro subagentes especializados por plataforma conforme Tabela 4. O orquestrador executa workflow em três fases: (1) detecção de framework; (2) delegação para subagentes; (3) agregação de resultados, seguindo padrão ReAct (YAO et al., 2023) (Gather-Act-Verify) que reduz alucinações de 14% para 6%.

Tabela 4 – Subagentes especializados implementados no sistema a11y-lint

Subagente	**Modelo**	**Responsabilidade**	**Ferramentas**
framework-detector	Haiku 4.5	Identificação de plataforma	Glob, Read, Grep
unified-web-scanner	Haiku 4.5	Análise web: HTML, ARIA, contraste, React	Read, Grep, MCP tools
swiftui-a11y-checker	Haiku 4.5	Análise iOS: modificadores, VoiceOver, contraste	Read, Grep, MCP tools
visual-a11y-analyzer	Haiku 4.5	Análise de screenshots	MCP tools

A especialização é garantida por prompts de sistema distintos codificando conhecimento de domínio (WCAG 2.2 para web, Apple HIG para SwiftUI), contextos isolados

e whitelists de ferramentas. Decisão de utilizar Haiku 4.5 fundamentou-se em trade-off custo/acurácia: 90-95% da capacidade de Sonnet com custo 5-7x menor, viabilizando integração em CI/CD. O Código 1 ilustra coordenação de fases.

Listing 1 – Coordenação de fases pelo orquestrador

```
async def scan(self, project_path: str):
    # Fase 1: Deteccao de framework
    framework = await self._extract_framework_from_response()

    # Fase 2: Analise especializada por plataforma
    if framework in ["react", "html"]:
        await self._invoke_subagent("unified-web-scanner")
    elif framework == "swiftui":
        await self._invoke_subagent("swiftui-a11y-checker")

    # Fase 3: Coleta e relatorio
    issues = get_collected_issues()
    return self._generate_markdown_report(issues, framework)
```

4.2 Codificação de Diretrizes em Prompts

Para o MVP do TCC I, as diretrizes de acessibilidade e heurísticas de UX/UI foram codificadas diretamente nos prompts de sistema dos subagentes especializados. Esta abordagem simplificada permite validação inicial da arquitetura multi-agente e pipeline de análise, mas não implementa RAG propriamente dito. O sistema RAG completo, com retrieval dinâmico de diretrizes via bancos vetoriais (LEWIS et al., 2020), será desenvolvido no TCC II conforme metodologia proposta na Seção 3.3. A Tabela 5 apresenta cobertura de diretrizes atualmente codificadas em prompts.

Tabela 5 – Cobertura de análise de acessibilidade por plataforma e categoria

Categoria	**Web/React**	**SwiftUI**	**Critérios**
Semântica HTML/UI	6	5	WCAG 1.3.1, 4.1.2
ARIA / Accessibility Modifiers	5	3	WCAG 4.1.2, HIG
Contraste visual	3	2	WCAG 1.4.3, 1.4.11
Navegação e foco	4	2	WCAG 2.4.3, 2.4.7
Total	**18**	**12**	**30 checks**

O Código 2 apresenta excerto de prompt do `unified-web-scanner`, ilustrando codificação estática de WCAG 2.2 com rastreabilidade (ID, nível).

Listing 2 – Codificação de diretrizes WCAG em prompt

```
system_prompt = """
Analise codigo e identifique violations:

HTML & Structure:
- Missing alt text on images (WCAG 1.1.1 Level A)
- Incorrect heading hierarchy (WCAG 1.3.1 Level A)

ARIA:
- Invalid ARIA roles/properties (WCAG 4.1.2 Level A)
- Input fields without labels (WCAG 1.3.1 Level A)

CSS & Colors:
- Contrast violations: 4.5:1 normal, 3:1 large (WCAG 1.4.3 AA)

React Patterns:
- onClick on div/span vs button (WCAG 2.1.1 A)
- Missing aria-live for dynamic updates (WCAG 4.1.3 AA)

Reporte: criterio WCAG (ID+Level), file:line, codigo,
```

```
correcao sugerida.
"""
```

Esta abordagem de prompts estáticos é suficiente para MVP e validação inicial, mas apresenta limitações de escalabilidade e manutenção. A implementação de RAG propriamente dito no TCC II permitirá retrieval dinâmico e contextual de diretrizes, conforme princípios de grounding documentados por Lewis et al. (2020).

4.3 Análise Contextual e Otimizações

O MVP implementa análise exclusivamente contextual via LLM para todas as plataformas (web, React, SwiftUI). A análise determinística (axe-core (Deque Labs, 2025), parsers AST) e pipeline híbrido serão desenvolvidos no TCC II conforme metodologia da Seção 3.4. A análise contextual via LLM demonstra capacidade de detectar violações semânticas que requerem compreensão de contexto: e.g., `<img alt="imagem"/>` contém atributo sintaticamente correto mas semanticamente inadequado (alt genérico não descritivo).

Durante desenvolvimento, implementou-se otimizações de performance. A Tabela 6 apresenta estratégias aplicadas e impactos observados informalmente.

Tabela 6 – Otimizações de performance implementadas

Otimização	**Antes**	**Depois**
Consolidação de agentes	6 agentes	3 agentes
Batch reporting de issues	1 issue/call	5-15 issues/call
Deduplicação de arquivos	Análise redundante	Cache inteligente
API calls (30 files)	240	15
Custo por scan	$0,75	$0,30
Tempo de scan (30 files)	5-6min	2-3min

Batch reporting foi estratégia mais impactante observada: subagentes acumulam múltiplas issues e reportam em lote via `ReportIssuesBatch`, reduzindo substancialmente número de chamadas de API. Custo atual de $0,30/scan ainda requer otimizações adicionais para viabilizar uso em larga escala em CI/CD, incluindo implementação de prompt caching quando SDK suportar.

4.4 Abordagem de Validação e Testes

O sistema implementa estratégia de validação híbrida considerando componentes determinísticos e não-determinísticos da arquitetura. Para componentes determinísticos — ferramentas MCP customizadas, gerenciamento de configuração, parsing de arquivos, deduplicação de issues — desenvolveu-se 25 testes unitários automatizados (pytest) com 100% de taxa de sucesso em 0,24s. Type checking via mypy em modo strict foi aplicado ao codebase completo, garantindo type safety.

Para componentes não-determinísticos — análises via LLM que dependem de raciocínio contextual e podem apresentar variabilidade entre execuções — adotou-se abordagem de validação manual devido ao custo atual de $0,30/scan. Mantém-se conjunto de arquivos de referência em HTML, React (JSX/TSX) e SwiftUI representando casos típicos de análise. A cada modificação em prompts de agentes, configurações de modelo ou lógica de coordenação que possa influenciar análises não-determinísticas, executa-se protocolo de validação manual: (1) executar análise nos arquivos de referência; (2) comparar outputs com versão anterior do sistema; (3) executar múltiplas vezes (tipicamente 2-3

execuções) para observar variabilidade entre runs; (4) validar coerência e qualidade das detecções.

Esta abordagem permite validação contínua durante desenvolvimento sem custos proibitivos, priorizando otimização de performance e qualidade de análise sobre minimização de custos nesta fase. Para TCC II, após implementação de otimizações de custo (prompt caching, batching adicional), planeja-se desenvolver suite de testes unitários baseados em chamadas de API para validação automatizada de componentes não-determinísticos, permitindo regression testing sistemático das capacidades de análise via LLM.

4.5 Interface de Usuário baseada em Texto (TUI)

O sistema implementa Interface de Usuário baseada em Texto (TUI) interativa desenvolvida com Textual (Textualize, 2024), framework Python para construção de aplicações terminais modernas. A interface suporta temas claro e escuro (Figura 3), detecção automática do tema do terminal, e elementos visuais enriquecidos incluindo ícones emoji, painéis estilizados, progress bars animadas e syntax highlighting para código e caminhos de arquivo.

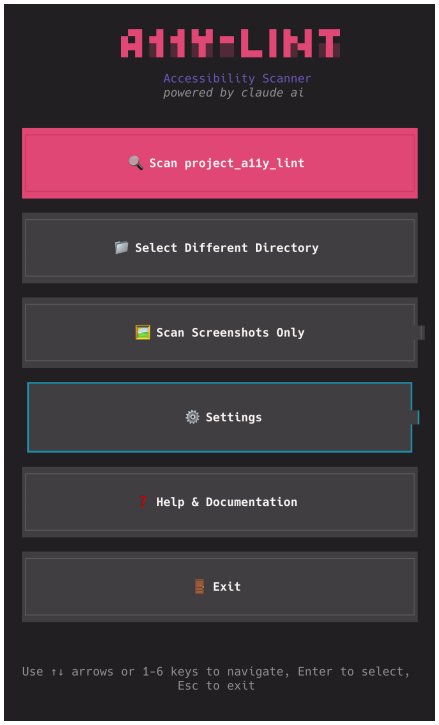

Figura 1 – *

(a) Tema escuro

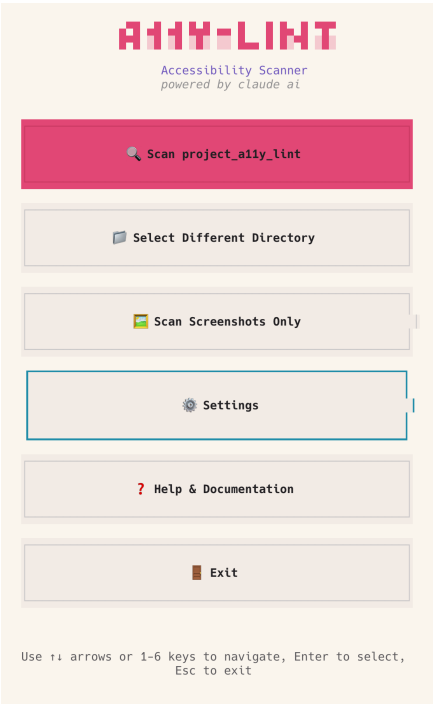

Figura 2 – *

(b) Tema claro

Figura 3 – Tela de boas-vindas do *a11y-lint* com suporte a temas claro e escuro

Durante a execução da análise, a interface apresenta feedback visual detalhado do progresso (Figura 4), incluindo indicação da fase atual (detecção de framework, análise de código), arquivos sendo processados, e quantidade de issues detectadas em tempo real. Ao

término, o sistema exibe sumário consolidado com estatísticas de análise e caminho do relatório gerado.

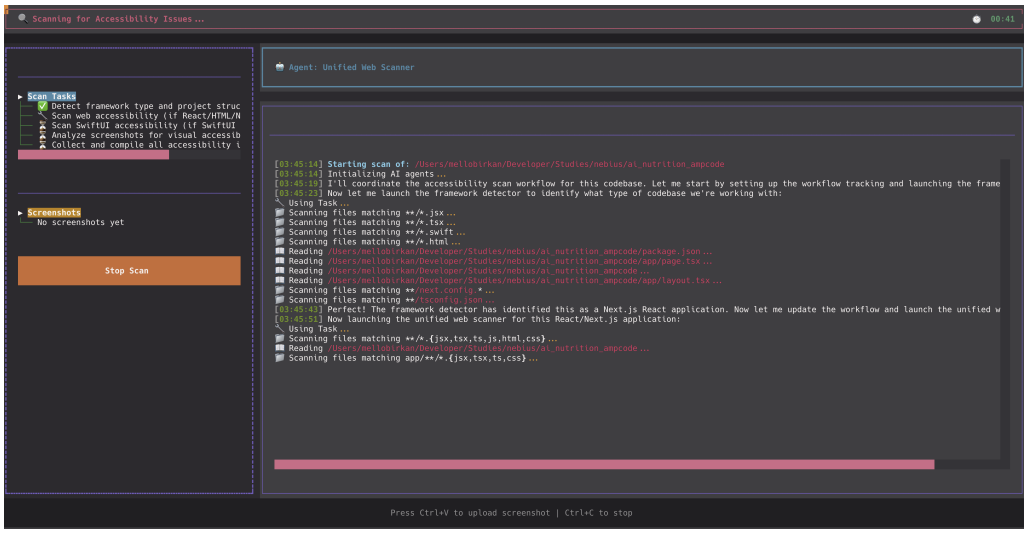

Figura 4 – Interface durante execução de análise mostrando progresso em tempo real

A ferramenta permite seleção flexível de diretórios através de navegador interativo (Figura 5), facilitando análise de diferentes projetos sem necessidade de especificar caminhos manualmente. Esta funcionalidade integra-se ao workflow típico de desenvolvimento, permitindo análise rápida de múltiplos projetos durante iterações de código.

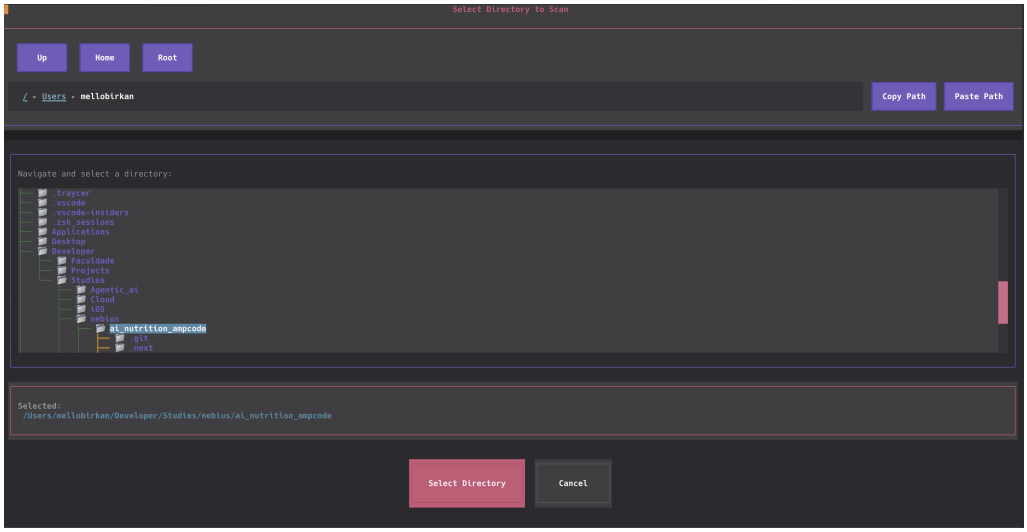

Figura 5 – Navegador interativo para seleção de diretório do projeto

5 Cronograma

O desenvolvimento e validação do sistema a11y-lint serão realizados durante o TCC II, no período de dezembro de 2025 a maio de 2026. Um protótipo funcional (MVP) do sistema multi-agente já foi implementado durante a fase de planejamento. O TCC II focará

na evolução da arquitetura, implementação do sistema RAG, integração de análise híbrida, validação experimental e redação do artigo científico. A Tabela 7 apresenta a distribuição das atividades planejadas.

Tabela 7 – Cronograma de Atividades - TCC 2

Atividade	**Dez**	**Jan**	**Fev**	**Mar**	**Abr**	**Mai**
Refatoração e otimização do código do MVP	X					
Implementação de subagentes especializados	X	X				
Coleta e estruturação de diretrizes oficiais		X				
Implementação de chunking e indexação vetorial		X	X			
Desenvolvimento de ferramentas MCP customizadas			X			
Integração de análise determinística			X			
Agregação inteligente e deduplicação			X			
Sistema de exportação multi-formato e melhorias TUI				X		
Envio e aprovação do CEP		X	X	X		
Testes de usabilidade	X	X	X	X	X	
Análise estatística dos dados			X	X	X	
Revisão final e preparação para banca						X
Escrita do artigo	X	X	X	X	X	X

6 Considerações Finais

Este trabalho apresentou a proposta do sistema a11y-lint, arquitetura multi-agente baseada em Large Language Models com Retrieval-Augmented Generation para análise automatizada de acessibilidade digital cross-platform. A motivação central reside na lacuna entre ferramentas convencionais que alcançam apenas 31,3% de cobertura em aplicações mobile e falham em detectar violações semânticas, e a necessidade crescente de conformidade multiplataforma com diretrizes distintas (WCAG 2.2, Apple HIG, WAI-ARIA).

O referencial teórico estabeleceu três pilares convergentes: capacidades emergentes de LLMs (function calling, RAG, chain-of-thought) materializadas pelo Claude Agent SDK, padrões validados de coordenação multi-agente (ReAct reduz alucinações de 14% para 6%, arquitetura orchestrator-workers reduz overhead em 10%), e aplicações demonstrando superioridade de abordagens híbridas (AccessGuru com 84% de redução em violation scores, ScreenAudit com 69,2% de cobertura versus 31,3% de ferramentas determinísticas).

A metodologia propõe pesquisa experimental-comparativa com arquitetura vertical hierárquica coordenando subagentes especializados por plataforma via padrão ReAct, integrando análise determinística com contextual via LLM fundamentada em sistema RAG com metadados de rastreabilidade. A validação abrange quatro dimensões complementares: avaliação quantitativa (precision, recall, F1-score, citation fidelity), comparação com ferramentas estabelecidas, avaliação com especialistas em acessibilidade e testes de usabilidade com desenvolvedores.

As contribuições esperadas abrangem três dimensões: (i) **científica** – primeira arquitetura multi-agente cross-platform para análise de acessibilidade com RAG citável, validando eficácia de coordenação vertical para análise de código; (ii) **tecnológica** – ferra-

menta open-source integrável a workflows de AI-assisted development, base de conhecimento curada de diretrizes oficiais indexadas semanticamente, e biblioteca de prompts validados; (iii) **social** – democratização de expertise em acessibilidade através de automação contextualizada com fundamentação verificável, elevando conhecimento de desenvolvedores e facilitando conformidade com diretrizes.

A análise comparativa (Tabela 2.1) evidencia o posicionamento único do a11y-lint na interseção de análise cross-platform (web + React + SwiftUI), arquitetura multi-agente com especialização por domínio, e RAG com citações verificáveis. Nenhum trabalho correlato analisado combina essas três dimensões, representando contribuição metodológica ao transformar análise de acessibilidade de "detecção de violações" para "auditoria de conformidade verificável com rastreabilidade completa".

Reconhecem-se limitações metodológicas: dataset limitado a 10 aplicações justificado pela natureza quali-quantitativa, possíveis desafios de recrutamento de especialistas e desenvolvedores, dependência de APIs comerciais (Claude/Anthropic) com custos operacionais associados, e cobertura restrita a web/iOS excluindo Android e Flutter. Estratégias de mitigação incluem recrutamento antecipado via múltiplos canais, otimizações de custo implementadas no MVP, e arquitetura extensível facilitando adição de novos subagentes.

A execução do plano metodológico durante o TCC II seguirá cronograma em cinco fases (20 semanas) com marcos de validação: sistema evoluído com seis subagentes (Semana 4), RAG operacional (Semana 8), pipeline híbrido completo (Semana 12), validação experimental (Semana 18) e artigo científico finalizado (Semana 20). Os resultados empíricos demonstrarão viabilidade técnica e relevância prática desta contribuição na interseção de LLMs, sistemas multi-agente e análise de acessibilidade digital, preenchendo lacuna identificada no estado da arte e oferecendo ferramenta concreta para democratização de expertise em acessibilidade.

Referências

Anthropic. *Claude Agent SDK Documentation: Building Effective Agents, Model Context Protocol, Agent Skills, and Context Engineering*. Anthropic Documentation, 2025. Disponível em: <https://docs.anthropic.com/en/api/agent-sdk>. Acesso em: 29 out. 2025.

Apple. *Essential Design Principles*. 2017. WWDC 2017. Disponível em: <https://developer.apple.com/videos/play/wwdc2017/802/>. Acesso em: 26 out. 2025.

Apple. *Human Interface Guidelines*. 2024. Disponível em: <https://developer.apple.com/design/human-interface-guidelines/>. Acesso em: 26 out. 2025.

Apple Inc. *Accessibility – Assistive Technologies*. Cupertino, 2025. Acesso em: 1 nov. 2024. Disponível em: <https://developer.apple.com/documentation/accessibility/assistive-technologies>.

Brasil. Governo Federal. *VLibras*. Brasília: [s.n.], 2025. Acesso em: 3 nov. 2024. Disponível em: <https://www.gov.br/governodigital/pt-br/acessibilidade-e-usuario/vlibras>.

Deque Labs. *axe-core: Accessibility Engine for Automated Web UI Testing*. 2025. GitHub repository. Disponível em: <https://github.com/dequelabs/axe-core>. Acesso em: 09 nov. 2025.

FATHALLAH, N.; HERNáNDEZ, D.; STAAB, S. AccessGuru: Leveraging LLMs to detect and correct web accessibility violations in HTML code. In: *Proceedings of the 27th International ACM SIGACCESS Conference on Computers and Accessibility*. [S.l.: s.n.], 2025.

Google. *Disability Innovation*. Mountain View: [s.n.], 2025. Acesso em: 3 nov. 2024. Disponível em: <https://belonging.google/disability-innovation/>.

Google. *Lighthouse Overview*. 2025. Disponível em: <https://developer.chrome.com/docs/lighthouse/overview?hl=pt-br>. Acesso em: 09 nov. 2025.

Hand Talk. *Recursos de acessibilidade: entenda como o Hand Talk funciona*. Maceió: [s.n.], 2025. Acesso em: 3 nov. 2024. Disponível em: <https://www.handtalk.me/br/ajudaacessibilidade/>.

HE, J.; TREUDE, C.; LO, D. LLM-based multi-agent systems for software engineering: Literature review, vision and the road ahead. *arXiv preprint arXiv:2404.04834*, 2025.

Level Access. *The Role of Assistive Technology in Digital Inclusion*. 2024. Acesso em: 3 nov. 2024. Disponível em: <https://www.levelaccess.com/blog/assistive-technology/>.

LEWIS, P. et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *arXiv preprint arXiv:2005.11401*, 2020.

MASTERMAN, T. et al. The landscape of emerging AI agent architectures for reasoning, planning, and tool calling: A survey. *arXiv preprint arXiv:2404.11584*, 2024.

MINAEE, S. et al. Large language models: A survey. *arXiv preprint arXiv:2402.06196*, 2024.

Mozilla Developer Network. *ARIA*. 2024. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/Accessibility/ARIA>. Acesso em: 27 out. 2025.

Mozilla Developer Network. *WAI-ARIA basics*. 2024. Disponível em: <https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Accessibility/WAI-ARIA_basics>. Acesso em: 27 out. 2025.

NIELSEN, J. *10 Usability Heuristics for User Interface Design*. 1994. Nielsen Norman Group. Atualizado em: 30 jan. 2024. Disponível em: <https://www.nngroup.com/articles/ten-usability-heuristics/>.

NIELSEN, J.; MOLICH, R. Heuristic evaluation of user interfaces. In: *CHI '90: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Seattle, Washington, USA: ACM Press, 1990. p. 249–256. Disponível em: <https://doi.org/10.1145/97243.97281>.

NORMAN, D. A. *The Design of Everyday Things*. Revised and expanded edition. New York: Basic Books, 2013.

React. *Accessibility*. 2024. Disponível em: <https://legacy.reactjs.org/docs/accessibility.html>. Acesso em: 28 out. 2025.

SCHICK, T. et al. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

SHNEIDERMAN, B. et al. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. 6. ed. Upper Saddle River: Pearson, 2016.

Textualize. *Textual: TUI (Text User Interface) framework for Python*. 2024. Disponível em: <https://textual.textualize.io/>. Acesso em: 09 nov. 2025.

VASWANI, A. et al. Attention is all you need. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Long Beach, CA, USA: [s.n.], 2017. p. 6000–6010.

WANG, L. et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 2025.

World Health Organization. *Priority Assistive Products List*. Geneva, 2016. Acesso em: 1 nov. 2024. Disponível em: <https://www.who.int/publications/i/item/priority-assistive-products-list>.

World Health Organization. *Global Report on Assistive Technology*. Geneva, 2022. Acesso em: 1 nov. 2024. Disponível em: <https://www.who.int/publications/i/item/9789240049451>.

World Wide Web Consortium (W3C). *Diretrizes de Acessibilidade para Conteúdo Web (WCAG) 2.2*. São Paulo: Ceweb.br, 2024. Tradução autorizada para o português do Brasil. Disponível em: <https://www.w3c.br/traducoes/wcag/wcag22-pt-BR/>. Acesso em: 25 out. 2025.

World Wide Web Consortium (W3C). *Introduction to Understanding WCAG 2.2*. 2025. Disponível em: <https://www.w3.org/WAI/WCAG22/Understanding/intro>. Acesso em: 25 out. 2025.

YAO, S. et al. ReAct: Synergizing reasoning and acting in language models. In: *Proceedings of the International Conference on Learning Representations*. [S.l.: s.n.], 2023.

ZHONG, M. et al. ScreenAudit: Detecting screen reader accessibility errors in mobile apps using large language models. In: *Proceedings of the CHI Conference on Human Factors in Computing Systems*. [S.l.: s.n.], 2025.