Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Criação de uma Biblioteca de Componentes com foco em Acessibilidade para Pessoas Daltônicas

Yasmine de Melo Leite

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Fernando Bernardes de Oliveira

Fevereiro, 2024
João Monlevade–MG

Yasmine de Melo Leite

Criação de uma Biblioteca de Componentes com foco em Acessibilidade para Pessoas Daltônicas

Orientador: Fernando Bernardes de Oliveira

Monografia apresentada ao curso de Sistemas de Informação do Instituto de Ciências Exatas e Aplicadas da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação

Universidade Federal de Ouro Preto

João Monlevade

Fevereiro de 2024

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Yasmine de Melo Leite

Criação de uma Biblioteca de Componentes com foco em Acessibilidade para Pessoas Daltônicas

Monografia apresentada ao Curso de Sistemas de Informação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação

Aprovada em 22 de fevereiro de 2024

Membros da banca

Prof. Dr. Fernando Bernardes de Oliveira - Orientador (Universidade Federal de Ouro Preto)
Profª. Drª. Gilda Aparecida de Assis - Avaliadora (Universidade Federal de Ouro Preto)
Profª. Drª. Lucinéia Souza Maia - Avaliadora (Universidade Federal de Ouro Preto)

Fernando Bernardes de Oliveira, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 03/03/2024

Documento assinado eletronicamente por **Fernando Bernardes de Oliveira**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 03/03/2024, às 10:53, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0676746** e o código CRC **6799DC6B**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: (31)3808-0819 - www.ufop.br

Dedico este trabalho a todos os que me ajudaram ao longo desta caminhada. A minha mãe que sempre me apoia em todos os projetos, ao meu namorado que além de apoiar me ajudou na escolha do tema.

"Se você está procurando algo que o faça feliz,
provavelmente o encontrará em seu próprio coração."

— Hedy Lamarr

Resumo

Este trabalho apresenta o desenvolvimento e disponibilização de uma biblioteca de componentes acessíveis para aplicações web, com foco na inclusão de usuários com daltonismo. A metodologia envolveu revisão da literatura, definição de modelo, prototipação, implementação e testes de nove componentes. Utilizando tecnologias como *React* e *TypeScript*, e ferramentas como *Material UI* e *GitHub*, a biblioteca foi desenvolvida seguindo as diretrizes de acessibilidade da W3C, visando proporcionar uma experiência digital mais equitativa. O trabalho destaca a importância da consideração da acessibilidade desde o início do processo de desenvolvimento, contribuindo para um ecossistema web mais inclusivo.

Palavras-chaves: Acessibilidade. Biblioteca de Componentes. Desenvolvimento Web.

Abstract

This work presents the development and availability of a library of accessible components for web applications, with a focus on including users with color blindness. The methodology involved literature review, model definition, prototyping, implementation and testing of nine components. Using technologies such as *React* and *TypeScript*, and tools such as *Material UI* and *GitHub*, the library was developed following W3C accessibility guidelines, aiming to provide a more equitable digital experience. The work highlights the importance of considering accessibility from the beginning of the development process, contributing to a more inclusive web ecosystem.

Key-words: Accessibility. Component Library. Web development.

Lista de ilustrações

Lista de abreviaturas e siglas

AVA Ambiente Virtual de Aprendizagem

IBGE Instituto Brasileiro de Geografia e Estatística

OMS Organização Mundial da Saúde

LBI Lei Brasileira de Inclusão da Pessoa com Deficiência

MUI *Material UI*

TI Tecnologia da Informação

UX *User Experience*

UI *User Interface*

WEB *World Wide Web*

Lista de Algoritmos

Sumário

1 Introdução

A acessibilidade é uma condição que permite à todas as pessoas, seja ela possuindo ou não qualquer tipo de deficiência, ter acesso, usar produtos, serviços e ambientes de forma segura e eficaz (BRASIL, 2015). O artigo 3º da Lei Brasileira de Inclusão dispõe sobre a inclusão da pessoa com deficiência, sua reabilitação, reinserção social e o combate à discriminação e define acessibilidade como:

> Acessibilidade: possibilidade e condição de alcance, percepção e entendimento para a utilização, com segurança e autonomia, de espaços, mobiliários, equipamentos urbanos, edificações, transportes, informação e comunicação, inclusive seus sistemas e tecnologias, bem como de outros serviços e instalações abertos ao público, de uso público ou privados de uso coletivo, tanto na zona urbana como na rural, por pessoa com deficiência ou com mobilidade reduzida (BRASIL, 2015).

A Lei Brasileira de Inclusão da Pessoa com Deficiência (LBI) define os tipos de acessibilidade cinco tipos de acessibilidade, são elas: acessibilidade arquitetônica, acessibilidade comunicacional, acessibilidade urbanística, acessibilidade nos transportes e acessibilidade na informação e comunicação (BRASIL, 2015). Essa ampla definição não apenas assegura a inclusão em ambientes físicos, mas também proporciona a base necessária para a inclusão no mundo virtual, onde as tecnologias estão cada vez mais avançadas e precisam abranger os aspectos de acessibilidade.

No campo da *World Wide Web* (WEB), a acessibilidade vem ganhando espaço, principalmente entre *User Experience* (UX) e *User Interface* (UI) *designer* (World Wide Web Consortium (W3C), 2018). Como mecanismo de inclusão digital, essas áreas envolvem a criação de experiências e interfaces que atendam a uma ampla variedade de usuários. Além disso, leva-se em consideração diversas atividades, necessidades, habilidades e contextos, sejam necessidades cognitivas ou até mesmo física. Isso inclui o *design* de interações intuitivas, navegação simples e apresentação de informações compreensíveis.

Entretanto, algumas estratégias já foram criadas para buscar essa equidade entre os indivíduos e trazer a inclusão, mesmo assim alguns desafios são apresentados na atualidade (COSTA et al., 2017; TEIXEIRA, 2018; LOPES; TERCIC, 2020). A falta de adaptações adequadas em *softwares* e interfaces de usuário pode levar a uma experiência de usuário insatisfatória principalmente para pessoas com problemas de visão.

O daltonismo é um distúrbio da visão, gerada a partir de uma condição genética que afeta principalmente homens e é causada por uma anomalia nos cones da retina (MELO;

GALON; FONTANELLA, 2014; SWAMINATHAN; KULKARNI, 2017). De acordo com a Organização Mundial da Saúde (OMS), cerca de 5% da população mundial é afetada pelo daltonismo, o que equivale a 350 milhões de pessoas. Já no Brasil, o Conselho Federal de Medicina, estima que mais de 8 milhões de pessoas tenham daltonismo (CFM, 2022). São inúmeros os desafios enfrentados pelos usuários WEB que possuem essa condição. A capacidade de distinguir entre cores desempenha um papel fundamental na compreensão e interação com computadores. Alguns exemplos são: a interação com o sistema operacional e aplicativos, compreensão de informações visuais, dificuldades de compreender cores com diferentes tonalidades (NEIVA, 2018; SILVA; OLIVEIRA, 2018; LOPES; TERCIC, 2020).

A dificuldade em distinguir cores pode afetar a capacidade de ler gráficos, tabelas e mapas, além de dificultar a identificação de botões e ícones em interfaces de usuário, além disso, algumas cores podem ser usadas para indicar informações importantes, como alertas de erro ou avisos de segurança(LOPES; TERCIC, 2020; SWAMINATHAN; KULKARNI, 2017). Essas e outras limitações impostas pelo daltonismo geram implicações diretas na eficiência, produtividade e experiência do usuário (MELO; GALON; FONTANELLA, 2014; LOPES; TERCIC, 2020).

Conforme apresentado por Lopes e Tercic (2020), a falta de adaptações adequadas em *softwares* e interfaces de usuário leva a uma experiência de usuário insatisfatória para pessoas com daltonismo. Por exemplo, a falta de contraste adequado entre o texto e o fundo pode dificultar a leitura de pessoas com daltonismo (MELO; GALON; FONTANELLA, 2014; NEIVA, 2018; SILVA; OLIVEIRA, 2018). A inclusão de recursos de acessibilidade, como a opção de aumentar o tamanho do texto ou alterar a cor de fundo, pode ajudar a melhorar a experiência do usuário para pessoas com daltonismo. No entanto, conforme apresentado pelos autores Silva e Oliveira (2018) e Lopes e Tercic (2020), muitos *softwares* e interfaces de usuário não incluem esses recursos.

A construção de sistemas WEB acessíveis enfrentam diversos desafios como por exemplo, falta de documentação sobre acessibilidade, falta de testes de acessibilidade, falta de suporte para personalização (VACCANI, 2023). Isso ocorre porque muitas bibliotecas de componentes não são projetadas com a acessibilidade em mente. Abordar tais aspectos, é interessante para explorar estratégias de *design* inclusivo, assim por meio de bibliotecas de componentes funcionais, com aplicações voltadas para este público alvo, pode tornar o acesso as tecnologias da informação mais acessíveis.

A biblioteca de componentes é um conjunto de componentes, dos quais incluem: botões, menus, caixas de seleção, tabelas, gráficos, entre outros (BRAMBILLA; CERI; FRATERNALI, 2006). A biblioteca de componentes WEB permite que os desenvolvedores criem interfaces de usuário consistentes e de alta qualidade com menos esforço e tempo (BRAMBILLA; CERI; FRATERNALI, 2006; ZHOU; ZHANG; LIU, 2007). Eles são projetados para serem facilmente integrados em aplicativos e ajudam a acelerar o processo

de desenvolvimento de sistemas limpos e e coesos preservando uma padronização.

Portanto, conceber uma solução que consolide os princípios de WEB UX/UI e acessibilidade, ancorada no estudo voltado para o público daltonista, não apenas simplifica o desenvolvimento de aplicações acessíveis, mas também transmite ao usuário final com limitações a sensação de conforto. Reforçando a intuitividade por meio de escolhas cuidadosas de cores e formas, é possível promover uma experiência digital acessível e equitativa. A integração da acessibilidade na WEB com estratégias inclusivas de UX/UI não apenas impulsiona a equidade, mas também resulta em interfaces mais robustas e eficientes para todos os usuários, independentemente de suas habilidades ou limitações.

1.1 Objetivos

Como objetivo geral o trabalho propõe desenvolver e disponibilizar uma biblioteca de componentes acessíveis para aplicações WEB, com foco na inclusão de usuários com daltonismo, visando aprimorar a experiência digital e promover a equidade no acesso às tecnologias da informação. São listados abaixo os objetivos específicos:

- Definir as cores que serão usadas, seguindo a documentação da W3C para acessibilidade e levantar boas práticas para componentes acessíveis em geral.
- Implementar a biblioteca de componentes seguindo as regras estudadas.
- Disponibilizar a biblioteca no *Github* ou em plataforma equivalente para utilização sob licença livre para a comunidade de desenvolvedores.

1.2 Justificativa

A partir de uma análise detalhada das necessidades e desafios enfrentados por essa parcela da população, serão identificadas as melhores práticas de *design* e desenvolvimento para superar tais obstáculos. Utilizando uma abordagem centrada no usuário e respaldada pelos princípios de *UX/UI* e acessibilidade WEB, a metodologia adotada concentra-se na criação de soluções inovadoras.

Ademais, a escolha das tecnologias de ponta, como *React* e *TypeScript*, confere à biblioteca de componentes uma base sólida e flexível, capaz de proporcionar interfaces consistentes, intuitivas e inclusivas. Ao disponibilizar essa biblioteca de forma aberta para a comunidade, busca-se não apenas atender às necessidades específicas dos usuários com daltonismo, mas também contribuir para um ecossistema WEB mais acessível e inclusivo como um todo.

1.3 Metodologia

A metodologia desse trabalho é apresentada a seguir, e detalha o conjunto de procedimentos e técnicas utilizadas para realizar o trabalho de forma sistemática. Isso inclui a descrição detalhada dos passos que foram seguidos para alcançar os objetivos.

1. Revisar a literatura: estudar as melhores práticas para criação de bibliotecas de componentes em *React* e definir os tipos de componentes que estarão presentes na biblioteca.

2. Definir um modelo: estudar e estabelecer as melhores práticas para a documentação de uma biblioteca de componentes.

3. Estudar representações: realizar a prototipação da biblioteca de acordo com os estudos realizados sobre os contrastes e *tags* a serem utilizados com o intuito de facilitar o entendimento do *software* e o desenvolvimento do código.

4. Implementar o *software*: desenvolvimento do código fonte com os componentes estabelecidos e de acordo com as melhores práticas que foram estudadas, como por exemplo, o contraste de cores, padronização na documentação do *software* e uso de padrão de cores universais.

5. Realizar testes com a biblioteca proposta.

6. Analisar e discutir os resultados obtidos, além de identificar possíveis melhorias e considerações gerais sobre a biblioteca.

1.4 Organização do trabalho

O restante deste trabalho está estruturado do seguinte modo. No Capítulo 2 são apresentados os principais conceitos abordados neste projeto e os trabalhos correlatados. O Capítulo 3 aborda os processos utilizados para desenvolvimento da biblioteca. Já no Capítulo 4, é apresentado a análise dos resultados e, ao findar, no Capítulo 5 são discutidas as considerações finais e os trabalhos futuros.

2 Revisão de literatura

A interseção entre o daltonismo e a WEB representa um campo fundamental na busca pela acessibilidade digital. As tecnologias modernas desempenham um papel importante nesse contexto, não apenas por facilitar o acesso a serviços *online*, mas também por proporcionar recursos especializados para atender às necessidades das pessoas com deficiência. Por exemplo, a transcrição de áudio é um recurso essencial para aqueles com deficiência visual ou auditiva, permitindo que consumam conteúdo digital de forma equivalente aos demais. A WEB oferece uma plataforma vasta e diversificada, onde pessoas com deficiência podem encontrar uma gama de informações, produtos e serviços que anteriormente estavam fora de seu alcance. Desde conteúdo educacional até oportunidades de emprego e entretenimento, a acessibilidade na WEB é fundamental para garantir a participação plena e igualitária de todos os usuários.

Este capítulo apresenta um estudo sobre o desenvolvimento de uma biblioteca de componentes acessíveis para pessoas com daltonismo. O capítulo começa com uma discussão sobre o daltonismo visual e os principais tipos. Em seguida, é discutido o daltonismo, sua conceitualização e dificuldades. Posteriormente, é apresentado como esses fatores impactam na distinção de cores. Para findar, é apresentado o conceito de acessibilidade e a participação das áreas de UX/UI para pessoas com daltonismo, no desenvolvimento de uma biblioteca de componentes, com o uso de tecnologias como *React* e *TypeScript*.

2.1 Fronteiras da visão

A visão é um dos cinco sentidos presentes nos seres humanos, os olhos são os órgãos responsáveis por esse sentido tão importante. É por meio da visão que temos a capacidade de enxergar e perceber o que está à nossa volta. É por meio da visão que somos capazes de andar e se locomover, ler e escrever, reconhecer pessoas e objetos. Os seres humanos são capazes de capturar em até 80% das informações do ambiente, através da nossa visão (TEIXEIRA, 2018). Em esfera global, segundo a Organização Mundial da Saúde (2019), pelo menos 2,2 bilhões de pessoas possuem problemas de visão. Já no Brasil, esse número aproxima de 35 milhões de pessoas, segundo dados do censo demográfico de 2010 realizado pelo Instituto Brasileiro de Geografia e Estatística (IBGE), 18,6% da população brasileira possui algum tipo de deficiência visual. O uso de tecnologias, como computadores, *smartphones* e *tablets*, pode apresentar desafios para pessoas com problemas de visão, como, por exemplo: o tamanho das telas, a resolução das imagens, a falta de acessibilidade de recursos e *softwares*.

2.1.1 Tipos de problemas visuais

A distinção de cores é a capacidade de perceber e diferenciar entre diferentes cores. Este é um processo complexo que envolve os olhos, o cérebro e o sistema nervoso central. Segundo o médico , alguns dos problemas de visão incluem a dificuldade em distinguir cores, são elas: daltonismo, achromatopsia, degeneração macular relacionada à idade, retinopatia diabética. Esses problemas podem dificultar a leitura de sinais de trânsito, a navegação por sites da WEB e a visualização de imagens, devido ao contraste entre os elementos de uma página, por exemplo. Dentre esses problemas, o Organização Mundial da Saúde (2019) destaca o daltonismo como o mais comum.

Segundo Smith e Jones (2014), o daltonismo é uma condição visual caracterizada pela dificuldade ou incapacidade de perceber algumas cores normalmente. A forma mais comum está relacionada à distinção entre vermelho e verde, embora também possa ocorrer dificuldade com outras cores, como azul e amarelo. Trata-se de um distúrbio genético da visão, afetando a capacidade de discernir cores devido a uma mutação no gene que codifica a opsina, uma proteína essencial para a detecção de luz nos olhos. Quando ocorre uma mutação nesse gene, a opsina não funciona corretamente, resultando em dificuldades na percepção de uma ou mais cores (SMITH; JONES, 2014).

Existem três tipos de daltonismo, são eles: deuteranopia, protanopia e tritanopia. A deuteranopia é caracterizada pela dificuldade ou incapacidade de distinguir entre verde e vermelho. É o tipo mais comum do daltonismo. O protanopia trata-se da dificuldade ou incapacidade de distinguir entre vermelho e verde, com uma maior afetação no espectro vermelho. Já o tritanopia, é a dificuldade ou incapacidade de distinguir entre azul e amarelo, sendo o tipo menos frequente, é menos comum em comparação com os dois anteriores (PEREIRA, 2022).

Em face dos desafios enfrentados por indivíduos com daltonismo, torna-se evidente a complexidade dessa condição visual que afeta a percepção das cores. A limitação na capacidade de discernir tonalidades específicas, como vermelho e verde, azul e amarelo, revela a importância de considerar a inclusão e acessibilidade para essas pessoas em diversos contextos. Embora o daltonismo seja, na maioria, um distúrbio genético, compreender as nuances e variações entre os tipos de daltonismo, como deuteranopia, protanopia e tritanopia, é fundamental para proporcionar suporte adequado. À medida que a sociedade avança em termos de tecnologia e *design*, a consideração desses desafios torna-se essencial para garantir um ambiente mais inclusivo e acessível a todos, independentemente de suas habilidades visuais.

2.2 Acessibilidade

A acessibilidade é a possibilidade e condição de alcance, percepção e entendimento para a utilização, com segurança e autonomia, do meio físico, do transporte, da informação e da comunicação, inclusive dos sistemas e tecnologias de informação e comunicação, bem como de outros serviços e instalações abertos ao público, de uso público ou privados de uso coletivo, tanto na zona urbana como na rural, por pessoa com deficiência ou com mobilidade reduzida (BRASIL, 2004; BERTAGLIA, 2022; Ministério do Planejamento, Desenvolvimento e Gestão, 2016).

Segundo o Brasil (2004), a acessibilidade, enquanto princípio fundamental para promover a inclusão social, manifesta-se em diversas dimensões, refletindo a necessidade de superar barreiras e garantir a participação plena de todas as pessoas. Neste contexto, é apresentado nove tipos distintos de acessibilidade, cada um direcionado a aspectos específicos da vida cotidiana e social. De acordo com (BERTAGLIA, 2022), existem 9 tipos de acessibilidade e são exemplificados a seguir:

- Acessibilidade Atitudinal: Refere-se às ações individuais que visam diminuir barreiras entre pessoas com e sem deficiência. Requer empatia, compreensão e ações que promovam um ambiente mais inclusivo e justo.
- Acessibilidade Arquitetônica: Envolvendo recursos como elevadores, rampas e corrimãos, a acessibilidade arquitetônica destaca-se na facilitação da locomoção de pessoas com deficiência física ou mobilidade reduzida, assegurando autonomia em diversos espaços.
- Acessibilidade Metodológica: Focado no âmbito educacional, esse tipo de acessibilidade busca diversificar metodologias e técnicas para garantir o acesso pleno de pessoas com deficiência à educação, superando desafios presentes no contexto acadêmico.
- Acessibilidade Programática: Relacionada à sensibilização e aplicação de normas, leis e políticas públicas, a acessibilidade programática visa garantir o cumprimento de direitos estabelecidos para pessoas com deficiência, promovendo uma sociedade mais inclusiva.
- Acessibilidade Instrumental: Direcionada ao uso de utensílios e ferramentas, a acessibilidade instrumental busca superar barreiras no uso de recursos necessários para atividades escolares, profissionais e de lazer, incluindo *softwares* específicos e dispositivos adaptativos.
- Acessibilidade nos Transportes: Ampliando a mobilidade de pessoas com deficiência ou mobilidade reduzida, essa categoria abrange não apenas assentos preferenciais,

mas todo o processo de viagem, incluindo catracas adaptadas, piso tátil e sinalização acessível.

- Acessibilidade Natural: Este tipo busca superar obstáculos impostos pela própria natureza, possibilitando, por exemplo, a locomoção de pessoas com deficiência física em ambientes naturais como praias, através do fornecimento de cadeiras de rodas anfíbias.

- Acessibilidade nas Comunicações: Priorizando a compreensão universal, a acessibilidade nas comunicações busca tornar informações acessíveis a todos. Intérpretes de Libras, assistentes virtuais e legendas em vídeos são exemplos práticos desse tipo de acessibilidade.

- Acessibilidade Digital: Diante do avanço tecnológico, a acessibilidade digital emerge como um tema emergente. Apesar da crescente importância, segundo a Forbes (2021), estudos indicam que menos de 1% dos *sites* atendem adequadamente a pessoas com deficiência. Estratégias como textos alternativos e alto contraste, dentre outras, são essenciais nesse contexto.

Em um cenário marcado pela diversidade, a compreensão e implementação desses diferentes tipos de acessibilidade são fundamentais para promover uma sociedade verdadeiramente inclusiva, onde cada indivíduo, independentemente de suas capacidades, tenha igualdade de oportunidades e participação plena.

A acessibilidade para pessoas com daltonismo enfrenta múltiplos desafios, muitos dos quais decorrem da falta de conscientização generalizada sobre esse distúrbio que afeta milhões em todo o mundo. A ausência de recursos e ferramentas acessíveis amplia as dificuldades, impactando atividades cotidianas como dirigir, trabalhar e aprender. Além disso, a falta de fiscalização sobre regulamentação de padrões de acessibilidade deixa a questão nas mãos da boa vontade de empresas e instituições (BERTAGLIA, 2022).

Esses obstáculos se manifestam na dificuldade em distinguir cores, prejudicando a interpretação de sinais de trânsito e a leitura de informações visuais, como mapas e gráficos. Para superar esses desafios, é essencial elevar a conscientização sobre o daltonismo, desenvolver recursos e ferramentas acessíveis e engajar o desenvolvimento de aplicações que garantam a inclusão equitativa dessas pessoas. Ações como educar o público, disponibilizar amplamente recursos específicos são passos cruciais para assegurar que as pessoas com daltonismo possam participar plenamente na sociedade, garantindo, assim, seu direito fundamental à acessibilidade.

A busca por soluções acessíveis para pessoas com daltonismo torna-se essencial no desenvolvimento de aplicações WEB inclusivas (COSTA et al., 2017). Abordagens

baseadas em cores e texto apresentado pelo (BRASIL, 2020) emergem como estratégias eficazes para superar os desafios enfrentados por esse público específico.

Algumas soluções conforme abordado por Mendes et al. (2022) utilizam o recurso de cores complementares, posicionadas opostamente no círculo cromático, emerge como uma alternativa eficaz para facilitar a distinção de cores por indivíduos com daltonismo. Ou até mesmo o emprego de cores de alto contraste, com uma notável diferença de luminosidade, destaca-se como uma abordagem visualmente impactante e acessível para esse público. Existe também a possibilidade de cores com saturação diferente, essa variação na saturação de cores proporciona uma diferenciação adicional, auxiliando na distinção de tons semelhantes.

Outras soluções também apresentadas pelos autores, podem ser empregadas ao texto, ao utilizar-se da inclusão de legendas descritivas para imagens coloridas, pois oferece uma descrição textual para compensar a limitação visual das pessoas com daltonismo. Aplicar cores às legendas, essa prática não apenas enriquece a experiência do usuário, mas também proporciona uma compreensão aprofundada do significado das cores.

De acordo com BRASIL (2020), ao almejar soluções acessíveis, é imperativo considerar algumas diretrizes essenciais como o uso consciente de cores, ao evitar o uso de cores semelhantes que possam dificultar a distinção para pessoas com daltonismo. Ter uma consistência ao utilizar as cores, utilizando delas para representar conceitos específicos para promover uma familiarização contínua para usuários com daltonismo. Oferecer opções personalizáveis, pois assim permite que indivíduos com daltonismo adaptem a interface conforme suas necessidades específicas.

Dentre as ferramentas existentes para ajudar com a definição de cores para adequar as questões de acessibilidade, estão: WIX [1], um verificador de contraste ajuda os desenvolvedores a garantir que as cores usadas em suas aplicações WEB tenham um contraste adequado e Color Oracle [2], que ajuda as pessoas com daltonismo a visualizar cores.

Assim, ao adotar essas estratégias, os desenvolvedores não apenas garantem a acessibilidade, mas também promovem a inclusão digital, reforçando a importância de uma abordagem consciente e centrada no usuário.

2.3 Tecnologias

A integração de tecnologias web, *JavaScript*, *Node.js* e bibliotecas de componentes, bem como estratégias de como UX/UI emerge como uma estratégia interessante para impulsionar a acessibilidade e abordar as necessidades específicas de usuários com daltonismo.

[1] <https://support.wix.com/pt/article/aplicativos-wix-verificador-de-contraste>
[2] <https://color-oracle.en.softonic.com/>

As estratégias de *design* UX/UI desempenha um papel central na acessibilidade, pois propõe dicas práticas, como o uso consciente de cores, consistência na representação de conceitos e a oferta de opções personalizáveis, como princípios-chave para garantir uma experiência inclusiva. Já o *JavaScript*, *TypeScript* e *Node.js* são ferramentas essenciais para adicionar funcionalidades de acessibilidade. Desenvolver uma biblioteca de componentes combinando essas tecnologias permite a exploração e adaptação dessas tecnologias para contribuir para a acessibilidade na WEB.

Ao utilizar as tecnologias disponíveis de forma estratégica, os desenvolvedores não apenas aprimoram a acessibilidade de suas aplicações web, mas também contribuem significativamente para a inclusão digital, reconhecendo a importância fundamental das soluções tecnológicas na criação de experiências acessíveis para todos. A seguir é detalhado cada campo dessas tecnologias explorado neste trabalho.

2.3.1 UX/UI

User Experience, ou experiência do usuário, refere-se à percepção geral de uma pessoa ao interagir com um produto, sistema ou serviço. Nesses aspectos, são envolvidos a qualidade da interação, a facilidade de uso, a eficiência percebida, a satisfação do usuário e outros aspectos relacionados à experiência global durante a utilização. Essa área, presente no *designer* concentra-se na criação de produtos e serviços que sejam fáceis de usar e agradáveis para o usuário (SWAMINATHAN; KULKARNI, 2017; NEIVA, 2018).

User Interface, ou interface do usuário, é a camada visual e interativa de um produto, envolvendo elementos como botões, menus, ícones e outros componentes visuais. Sendo assim, responsável pela apresentação visual e pela interação direta entre o usuário e o sistema. É de fato o que o usuário vê e com o que ele interage (SWAMINATHAN; KULKARNI, 2017; NEIVA, 2018).

Os dois conceitos UX/UI estão interconectadas para criar uma experiência coesa e satisfatória nas aplicações WEB. A sinergia dessas áreas, contribui não só para a experiencia do usuário em quesito funcionalidade, mas também consegue captar aspectos emocionais, cognitivos e contextuais que moldam a experiência do usuário em sua totalidade.

2.3.2 *React*

O *React* [3] é uma biblioteca *JavaScript* declarativa, eficiente e flexível para criar interfaces com o usuário, a UI. É através dessa ferramenta que é possível compor UI complexas, a partir de pequenos e isolados códigos chamados “componentes” (ABRAMOV; NABORS, 2023).

[3] <https://legacy.Reactjs.org/docs/getting-started.html>

O *React* se destaca por ser uma biblioteca declarativa, isto é, permite que os desenvolvedores descrevam a aparência desejada da interface, enquanto o código bruto de como fazer, é totalmente implementado pelo *React*. Essa abordagem torna o código mais legível e fácil de manter. Além disso, o *React* é eficiente, pois ele utiliza seus recursos de forma otimizada para dispor de aplicações mais rápidas e responsivas. Sua flexibilidade, permite a criação de uma ampla variedade de interfaces, tornando-o uma escolha versátil para diferentes projetos (SACRAMENTO, 2023).

2.3.3 *TypeScript*

O *TypeScript* é uma linguagem de programação de código aberto desenvolvida pela Microsoft [4]. Ele é formado por um superconjunto sintático estrito de *JavaScript*, que acrescenta a tipagem estática à linguagem. Essa combinação o torna compatível com *JavaScript*, mas com regras de sintaxe e semântica adicionais, proporcionando maior segurança e confiabilidade em comparação com *JavaScript* puro (MELO, 2021) .

A tipagem estática envolve a definição explícita dos tipos de valores e variáveis no código, reduzindo erros e aprimorando a legibilidade. Segundo definições na literatura, por isso ele é descrito como uma linguagem que une a flexibilidade do *JavaScript* com a segurança e produtividade da tipagem estática (MELO, 2021).

Devido a esses requisitos, se torna versátil, o que possibilita o emprego em diversos projetos, como aplicações web, aplicativos móveis, jogos e outros *frameworks*. Ele é utilizado por empresas além da própria fundadora, como, *Google* e *Amazon*, pois oferece algumas vantagens como segurança, produtividade e legibilidade. No geral, é uma escolha sólida para projetos que requerem alto nível de segurança, produtividade e clareza no código.

2.3.4 Biblioteca de componentes

Uma biblioteca de componentes é um conjunto organizado de elementos de UI pré-desenvolvidos e reutilizáveis, projetados para serem incorporados em diferentes partes de um aplicativo ou sistema. Esses componentes, como, por exemplo, botões, campos de entrada e barras de navegação, oferecem uma abordagem eficiente para o desenvolvimento de interfaces consistentes e padronizadas (REACT, 2021).

Os desenvolvedores podem integrar esses elementos em seus projetos, economizando tempo e esforço. A funcionalidade da biblioteca é baseada na reutilização de componentes em vários projetos, promovendo consistência visual e funcional em toda a aplicação. As vantagens de utilizar uma biblioteca de componentes incluem a eficiência no desenvolvimento, facilitando a manutenção e melhorando a testabilidade.

[4] <https://www.microsoft.com/pt-br>

Além disso, as bibliotecas proporcionam benefícios como a padronização do *design*, colaboração eficaz entre equipes e a possibilidade de adaptação a diferentes estilos. Assim, são ferramentas valiosas que contribuem para interfaces de usuário mais eficientes e consistentes, proporcionando benefícios como economia de tempo, facilidade de manutenção e melhorias na colaboração entre desenvolvedores.

2.4 Trabalhos relacionados

O trabalho de Costa et al. (2017) aborda a implementação de um protótipo para tornar um Ambiente Virtual de Aprendizagem (AVA), especificamente o "Aprender Unoeste", mais acessível para pessoas com daltonismo. A pesquisa destaca a importância de tornar os ambientes virtuais inclusivos e propõe duas abordagens para a adaptação do AVA: "Legenda em Escrita" e "Legenda ColorAdd". O primeiro modelo exibe o nome da cor ao passar o mouse sobre elementos coloridos, enquanto o segundo utiliza ícones do sistema ColorADD. Ambos os modelos podem ser ativados por meio de um ícone de acessibilidade disponível na interface do AVA. A pesquisa também destaca a importância de orientar os profissionais de *design* de interface sobre o uso adequado de cores e aponta para futuras implementações e testes com estudantes daltônicos.

Existem algumas soluções voltadas para jogos, onde se destaca o trabalho de Valadares et al. (2022) que desenvolveu o jogo "*Upwards!*". A proposta é um jogo de plataforma 2D com foco na competição entre jogadores, cujo objetivo é alcançar o topo de uma grande árvore. O diferencial do jogo é a ênfase na acessibilidade, com mecânicas desenvolvidas para pessoas com deficiências motoras, auditivas e visuais. O jogo busca promover inclusão, equidade e diversão para todos os públicos. As mecânicas do jogo são simplificadas, utilizando apenas um botão para controle, e o *design* é voltado para alta visibilidade e compreensão. Recursos de acessibilidade são integrados, incluindo paletas de cores adaptadas para daltonismo, modos de alto contraste, e ajustes para melhorar a qualidade de vida dos jogadores. O texto destaca a importância da consulta a pessoas com diferentes deficiências para ajustar o jogo conforme as necessidades reais.

2.5 Considerações finais

Este capítulo explorou a importância do uso de tecnologias para promover a acessibilidade e a inclusão social, com foco especial na criação de uma biblioteca de componentes acessíveis para pessoas com deficiência visual. Através da WEB, indivíduos com deficiência visual podem acessar informações, produtos e serviços que anteriormente eram inacessíveis a eles, destacando a relevância da tecnologia como uma ferramenta poderosa para a inclusão.

A discussão sobre os diferentes tipos de problemas visuais, com ênfase no daltonismo, ressaltou os desafios enfrentados por indivíduos com dificuldades na distinção de cores. Ao compreender as nuances e variações entre os tipos de daltonismo, como deuteranopia, protanopia e tritanopia, torna-se possível oferecer suporte adequado e desenvolver soluções inclusivas que atendam às necessidades específicas desses usuários.

A acessibilidade, como princípio fundamental para promover a inclusão social, foi abordada em suas diversas dimensões, destacando a importância de superar barreiras e garantir a participação plena de todas as pessoas. A apresentação dos nove tipos de acessibilidade ressaltou a necessidade de considerar diferentes aspectos da vida cotidiana e social para garantir a inclusão equitativa de todos os indivíduos.

Os principais desafios enfrentados por pessoas com daltonismo foram discutidos, incluindo a falta de conscientização e a ausência de recursos acessíveis. No entanto, foram apresentadas possíveis soluções, como o uso de cores complementares, legendas descritivas e ferramentas de verificação de contraste, destacando a importância de uma abordagem consciente e centrada no usuário para garantir a acessibilidade digital.

A integração de tecnologias como UX/UI, *React*, *TypeScript* e bibliotecas de componentes foi explorada como uma estratégia eficaz para impulsionar a acessibilidade e abordar as necessidades específicas de usuários com daltonismo. O *design* UX/UI desempenha um papel fundamental na criação de interfaces acessíveis, enquanto o uso de tecnologias como *React* e *TypeScript* permite o desenvolvimento de aplicações WEB inclusivas e adaptáveis.

3 Desenvolvimento

No âmbito do desenvolvimento WEB moderno, a construção de uma biblioteca de componentes eficiente é um componente essencial para a criação de interfaces robustas e consistentes. Nesse contexto, o uso de ferramentas adequadas desempenha um papel crucial, capacitando os desenvolvedores a otimizarem o processo de desenvolvimento e aprimorarem a qualidade do produto final.

Ferramentas e técnicas como: *Material UI*, *GitHub*, *React TypeScript*, Testes Unitários e *Storybook*, desempenham papéis complementares e essenciais na construção de uma biblioteca de componentes eficiente. Ao adotar uma abordagem centrada nas melhores práticas e na utilização dessas tecnologias, os desenvolvedores podem maximizar a eficiência do desenvolvimento WEB e oferecer produtos de alta qualidade que atendam às necessidades e expectativas dos usuários. Abaixo é apresentado um *overview* sobre algumas tecnologias, métodos e ferramentas utilizadas.

3.1 Linguagem de programação

O *React* e *TypeScript* desempenham papéis fundamentais na construção de uma biblioteca de componentes, oferecendo uma poderosa combinação de eficiência e robustez. O *React* fornece uma arquitetura baseada em componentes reutilizáveis e uma abordagem declarativa para o desenvolvimento de interfaces de usuário. Essa abordagem permite que os desenvolvedores criem componentes independentes que podem ser reutilizados em todo o aplicativo, simplificando assim o desenvolvimento e a manutenção do código.

Por outro lado, o *TypeScript* adiciona tipagem estática ao *JavaScript*, o que eleva a qualidade e a manutenibilidade do código. Ao permitir a definição de tipos para variáveis, parâmetros de função e retornos de função, o *TypeScript* ajuda a capturar erros de forma mais rápida e eficaz durante o desenvolvimento, além de facilitar a compreensão do código por outros desenvolvedores.

Juntas, essas duas tecnologias proporcionam aos desenvolvedores a capacidade de criar componentes altamente personalizáveis, escaláveis e fáceis de manter. Ao utilizar o *React* em conjunto com o *TypeScript*, o processo de desenvolvimento é acelerado, pois a combinação dessas ferramentas oferece uma base sólida para a criação de aplicativos consistentes e confiáveis.

Neste trabalho, foram utilizadas as versões específicas dessas tecnologias: *React* na versão 18.1.0 e *TypeScript* na versão 4.6.4. Essas versões representam a última atualização estável no momento da redação deste texto, garantindo assim compatibilidade e acesso

aos recursos mais recentes oferecidos por essas ferramentas.

3.2 *Storybook*

Na área de Tecnologia da Informação (TI) os testes desempenham um papel fundamental na garantia da qualidade e no funcionamento adequado de sistemas, aplicativos e componentes de *software*. Os testes são procedimentos realizados para verificar se um determinado aspecto do *software* atende aos requisitos especificados e se funciona conforme o esperado. Eles podem abranger diferentes áreas, como funcionalidade, desempenho, segurança e usabilidade, e são conduzidos em diferentes estágios do ciclo de vida do desenvolvimento de *software*, desde a fase de desenvolvimento até a implantação e manutenção (MESZAROS, 2007).

Devido à natureza dinâmica e interativa das aplicações WEB, é importante a realização de testes unitários nesse contexto. Eles são responsáveis por testar unidades individuais de código, como funções ou métodos, de forma isolada do restante do sistema. Para os componentes de uma biblioteca, os testes unitários garantem que cada componente funcione conforme o esperado em diferentes cenários e que eventuais alterações no código não introduzam regressões ou quebras de funcionalidade (SILVA, 2021).

Os testes unitários oferecem uma série de benefícios na construção de bibliotecas de componentes, pois ajudam a identificar e corrigir problemas de forma mais rápida e eficiente, facilitam a refatoração do código com confiança e promovem a reutilização e modularidade do código. Além disso, os testes fornecem uma representação atualizada do comportamento esperado dos componentes, facilitando a compreensão do código por parte de outros desenvolvedores e contribuindo para a manutenção de um código limpo e de alta qualidade. (SILVA, 2021; COUTINHO, 2021).

O *Storybook*[1] é uma ferramenta valiosa para documentar os testes de uma aplicação, pois ele fornece um ambiente de desenvolvimento visual e interativo para os componentes de interface de usuário, o que simplifica a revisão, execução e comunicação dos testes entre os membros da equipe. É uma ferramenta de desenvolvimento de código aberto que permite aos desenvolvedores criar e visualizar componentes de interface de usuário de forma isolada e interativa. Ele é frequentemente utilizado durante o processo de desenvolvimento de aplicações WEB para construir e testar componentes individualmente, sem a necessidade de integração com o restante do aplicativo.

Conforme apresentado por Coutinho (2021), as vantagens de utilizar o *Storybook* para documentar os testes de uma aplicação são várias, uma delas é por ele fornecer um ambiente centralizado para visualizar e interagir com os componentes, o que facilita a revisão e a validação dos testes em diferentes cenários. Além disso, ao documentar os com-

[1] <https://storybook.js.org/>

ponentes de forma isolada, os desenvolvedores podem entender melhor seu comportamento e funcionalidades específicas, o que ajuda na elaboração e execução dos testes unitários. Outra vantagem é que o *Storybook* permite a geração automática de documentação dos componentes, incluindo exemplos de uso, propriedades disponíveis e estados possíveis, o que simplifica a comunicação entre os membros da equipe e os *stakeholders*.

A seguir é apresentado um exemplo do código utilizado para construir um componente no *Storybook*.

```
const meta: Meta = {
  title: 'Components/ButtonAppBar',
  component: ButtonAppBar,
  argTypes: {
    color: {
      description: 'The color the AppBar will have.',
      control: { type: 'radio' },
      options: ['primary', 'secondary','success', 'error', 'info
         ', 'warning'],
    },
    buttonText: {
      description: 'Text that will be displayed inside the
         button.',
      control: 'text',
    },
    typographyText: {
      description: 'Text that will be displayed inside the bar.'
         ,
      control: 'text',
    },
  },
};

export default meta;
const Template: ComponentStory<typeof ButtonAppBar> = (args) =>
   <ButtonAppBar {...args} />;
export const Default = Template.bind({});

Default.args = {
    color: 'primary',
    buttonText: 'button',
    typographyText: 'Title',
};
```

Algoritmo 3.1 – Componente: ButtonAppBar.storie.tsx

Inicialmente, são descritas as opções fornecidas pelo componente, exemplificadas pela barra de menu, no código onde o componente *ButtonAppBar* é definido. Este componente requer três parâmetros: *color*, que determina a cor da barra e é obrigatório; *buttonText*, opcional, que define o texto exibido dentro do botão da barra, sendo utilizado o texto padrão *"button"* caso nenhum valor seja especificado; e o último parâmetro, *typographyText*, que representa o texto do título exibido na barra.

Após a enumeração dos atributos recebidos pelo elemento, o componente original pode ser empregado no código, permitindo sua renderização na tela. Isso possibilita que interessados acessem o elemento, testem cores e textos, sem a necessidade de integrá-lo em uma tela específica.

A seguir, é apresentada uma imagem do componente no *Storybook*, juntamente com as opções de personalização mencionadas.

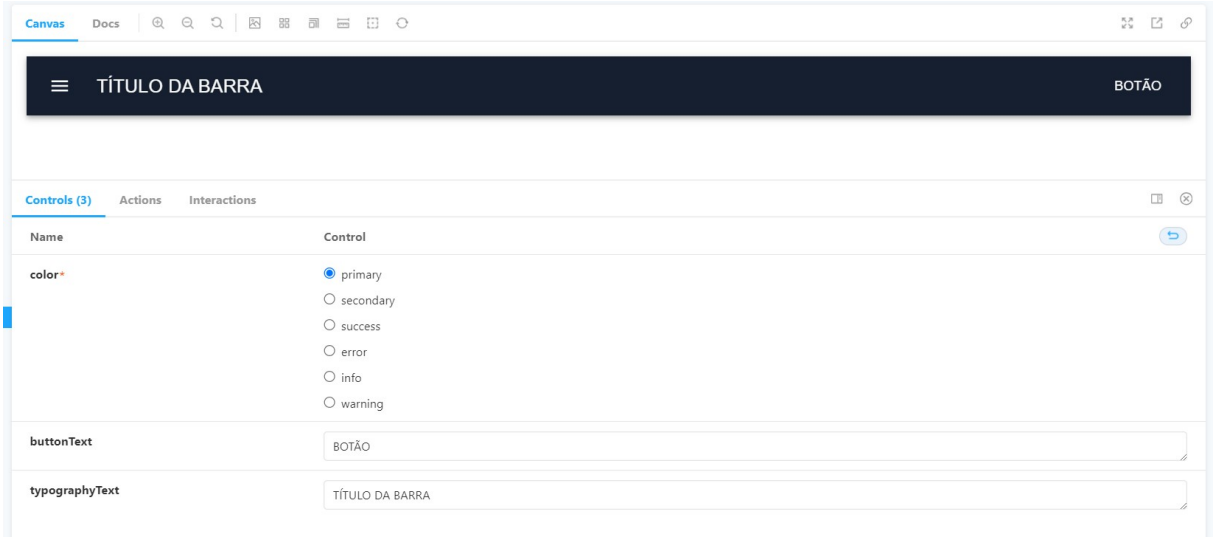

Figura 1 – Barra de menu *Storybook*

AWEB *Testing Library* é uma biblioteca de testes para aplicações *React* desenvolvida com foco na experiência do usuário. Ela permite que os desenvolvedores escrevam testes que simulam o comportamento do usuário ao interagir com os componentes da aplicação. Em contraste com a abordagem tradicional de testar a implementação interna dos componentes, a *React Testing Library* prioriza o teste do comportamento visível e acessível da interface do usuário.

Essa metodologia de teste enfatiza a interação do usuário com a aplicação, assegurando que os componentes funcionem corretamente do ponto de vista do usuário final. Consequentemente, os testes elaborados com a *React Testing Library* concentram-se na

renderização dos componentes na tela, em como eles respondem às interações do usuário e em seu comportamento em diversos contextos.

3.3 Testes unitários

Utilizando o componente *Button* são exemplificados dois testes unitários. O primeiro garante que, durante a renderização do botão, o subcomponente *children* seja exibido corretamente, com o valor especificado. Já o segundo teste assegura que, ao realizar a ação de clicar no botão, a função *onClick*, responsável por executar a ação definida pelo programador, seja chamada adequadamente.

Os testes apresentados abaixo demonstram uma implementação simples, porém eficaz, para garantir que as funcionalidades e funções oferecidas pelo componente estejam executando de maneira correta, reduzindo a possibilidade de ocorrência de *bugs*.

```
1  import '@testing-library/jest-dom';
2  import { render, screen } from '@testing-library/react';
3  import userEvent from '@testing-library/user-event';
4  import { Button } from './Button';
5
6  describe('<Button/>', () => {
7    it('should has an element children', () => {
8      render(
9          <Button color='primary'>
10           BUTTON
11         </Button>
12     );
13     const buttonElementChildren = screen.getByText('BUTTON');
14     expect(buttonElementChildren).toHaveTextContent('BUTTON');
15   });
16
17   it('should call onClick function when the button is clicked',
        () => {
18     const onClick = jest.fn();
19     render(
20         <Button onClick={onClick} color='primary'>
21           UPDATE
22         </Button>
23     );
24
25     const clickAtButton = screen.getByRole('button');
```

```
26      userEvent.click(clickAtButton);
27      expect(onClick).toBeCalledTimes(1);
28    });
29  });
```

Algoritmo 3.2 – Componente: Button.test.tsx

Para a realização dos testes unitários, adotou-se o padrão Triple A (*Arrange, Act, Assert*) com o objetivo de garantir uma melhor legibilidade e manutenibilidade do código (DARDE, 2020).

1. *Arrange*: Nesta etapa, são configurados todos os elementos necessários para que o teste possa ser executado. Isso inclui a inicialização de variáveis, a criação de mocks e outras preparações.

2. *Act*: Durante esta etapa, o teste é efetivamente executado. São realizadas chamadas a funções ou métodos que serão avaliados.

3. *Assert*: Na última etapa, é feita a verificação do resultado obtido na etapa anterior (*Act*). Aqui, é onde se realiza o *assert* para garantir se a operação executada produziu o resultado esperado. Isso determina se o teste foi bem-sucedido ou não.

Utilizando o segundo teste como exemplo, denominado *"should call onClick function when the button is clicked"*, é possível analisar as etapas da seguinte maneira. O *Arrange* é realizado das linhas 18 à 23, onde é primeiramente provisionado um *mock*, que são objetos que simulam o comportamento de objetos reais de forma controlada, logo, para a função de clique no botão utilizando `jest.fn()`, seguido pela renderização do botão com as propriedades desejadas.

O *Act* ocorre nas linhas 25 e 26, onde o elemento do botão renderizado é capturado na variável *clickAtButton*, seguido pela ação de clicar no botão.

Posteriormente à execução da ação, a etapa *Assert* é realizada na linha 27, onde o teste espera que o evento de clicar no botão seja chamado apenas uma vez. Caso esse seja o resultado obtido, o teste será finalizado com sucesso.

3.4 *GitHub*

O *GitHub* [2] é uma plataforma de hospedagem de código-fonte baseada em nuvem, que utiliza o sistema de controle de versão. Ele serve como um repositório centralizado para armazenar, gerenciar e colaborar em projetos de *software*. Ele funciona de tal modo

[2] Documentação oficial *GitHub*: <https://github.com/>

que permite os desenvolvedores carregar seus projetos para o *GitHub*, onde outros membros da equipe ou colaboradores externos podem acessar, visualizar, clonar, contribuir e sugerir alterações por meio de *pull requests* (FERNANDA; LOUZADA, 2023).

A importância do *GitHub* nas aplicações é vasta, primeiramente, ele facilita a colaboração entre desenvolvedores, permitindo que trabalhem em equipe de forma eficiente, mesmo que estejam geograficamente dispersos. Além disso, o *GitHub* oferece um histórico completo de alterações em cada arquivo, o que possibilita o rastreamento de mudanças e a reversão para versões anteriores, se necessário. Essas qualidades contribuem para a integridade e a segurança do código.

Além disso, conforme mencionado pelos autores Fernanda e Louzada (2023) o *GitHub* serve como uma vitrine para desenvolvedores individuais, permitindo que mostrem seus projetos e contribuições para potenciais empregadores ou colaboradores. Essa tecnologia também permite a automação de fluxos de trabalho, como testes de integração contínua, implantação automatizada e *Material UI* (MUI)to mais. Isso aumenta a eficiência do desenvolvimento, garantindo a qualidade do código e acelerando o processo de entrega.

Neste projeto, o *GitHub* desempenhou um papel fundamental no desenvolvimento para o gerenciamento de código-fonte, disponível em Melo (2024), e automação de fluxos de trabalho. Além disso, a partir dele é possível disponibilizar a aplicação para a comunidade de desenvolvedores.

3.5 *Material UI*

O *Material UI* (MUI) [3] é um biblioteca disponível para a comunidade de forma gratuita, assim, é uma escolha proeminente para a construção de interfaces, pois oferece uma vasta coleção de componentes já prontos para uso. Essa ferramenta não só acelera o desenvolvimento, mas fornece componentes pré-estilizados e consistentes, que promove uma experiência de usuário coesa e intuitiva, essencial para a aceitação do produto final (RIBEIRO, 2018).

O MUI oferece uma série de vantagens significativas que o tornam uma escolha popular para o desenvolvimento de interfaces de usuário em projetos WEB. Embora seja amplamente personalizável, sua adaptabilidade pode variar dependendo das necessidades específicas do projeto.

Segundo a documentação oficial do MUI, ele fornece uma extensa biblioteca de componentes prontos para uso, seguindo as diretrizes fixas de *design* já implementados. Isso acelera significativamente o processo de desenvolvimento, permitindo que os desenvolvedores construam interfaces de usuário de alta qualidade de forma mais rápida e eficiente. Ao

[3] Documentação oficial MUI: <https://mui.com/>

seguir as diretrizes internas de *design* garante uma experiência de usuário consistente e intuitiva. Os componentes têm um estilo coeso que se integra perfeitamente em diferentes partes do aplicativo, criando uma experiência visualmente unificada para os usuários. Além disso, seus componentes do são projetados para serem responsivos por padrão, adaptando-se automaticamente a diferentes tamanhos de tela e dispositivos. Isso simplifica o processo de criação de interfaces que funcionam bem em uma variedade de dispositivos, desde *desktops* até os *smartphones*.

O MUI é devidamente documentado, com uma ampla variedade de exemplos, guias e tutoriais disponíveis para ajudar os desenvolvedores a entenderem e utilizarem efetivamente os componentes. Isso facilita a aprendizagem e a adoção dessa tecnologia, até mesmo para os iniciantes. Embora os componentes do MUI tenham estilos pré-definidos que seguem as suas próprias diretrizes, eles também são personalizáveis. Os desenvolvedores podem facilmente substituir estilos, cores e outros atributos para se adequar às necessidades específicas de *design* do projeto.

Ademais, o MUI possui uma comunidade de desenvolvedores ativa e vibrante, o que significa que há uma ampla gama de recursos disponíveis, incluindo bibliotecas de terceiros, *plugins* e suporte em fóruns de discussão. Isso pode ser extremamente útil para resolver problemas, obter conselhos e compartilhar conhecimento com outros desenvolvedores que usam a ferramenta.

3.6 Desenvolvimento da aplicação

Após a seleção das tecnologias pertinentes, o objetivo é estabelecer estratégias eficazes para o desenvolvimento de uma biblioteca de componentes, que assegure a integração harmoniosa com os requisitos de acessibilidade. Neste contexto, é fundamental adotar medidas proativas para garantir que os componentes desenvolvidos sejam acessíveis e utilizáveis por usuários com daltonismo.

3.6.1 Dinâmica de cores

As principais dificuldades enfrentadas por pessoas com daltonismo estão relacionadas à dificuldade em distinguir certas cores e tons, especialmente entre tons de vermelho e verde, e entre tons de azul e amarelo. Isso pode afetar sua capacidade de ler informações em gráficos, mapas, sinais de trânsito, interfaces digitais e outras situações cotidianas.

As pessoas com daltonismo podem enxergar as cores de maneira alterada em comparação com pessoas com visão normal. Por exemplo, um indivíduo com daltonismo vermelho-verde pode ter dificuldade em diferenciar o vermelho do verde, vendo ambos como tons de cinza ou marrom. Já um indivíduo com daltonismo azul-amarelo pode confundir essas cores, vendo tons de cinza ou verde em vez de azul.

O uso de tecnologia pode ajudar a mitigar essas dificuldades de diversas maneiras, uma abordagem comum é a utilização de esquemas de cores acessíveis, que garantem que as informações sejam apresentadas de maneira clara e legível para todas as pessoas, independentemente de suas limitações de percepção de cor. Conforme apresentado por Dana (2019), Medeiros (2023), isso pode incluir o uso de cores de alto contraste, padrões texturizados e símbolos adicionais para transmitir informações importantes.

3.6.1.1 Cores de alto contraste

Cores de alto contraste são combinações de cores que têm uma diferença significativa em luminosidade ou tonalidade, facilitando a distinção entre elementos visuais. Essas combinações são especialmente importantes para pessoas com deficiências visuais, como baixa visão ou daltonismo, pois ajudam a tornar o conteúdo mais legível e acessível.

A W3C (2023) apresenta algumas indicações de exemplos de alto contraste, como a combinação de preto e branco. Essa combinação oferece uma diferença notável em luminosidade, tornando os textos e elementos visuais claramente visíveis. Isso pode ser observado:

Nesse primeiro exemplo é apresentado um botão com contraste baixo, a cor de fundo é a cor preta (`#000000`) e o texto cinza médio (`#4F4F4F`).

Figura 2 – Botão com texto em contraste baixo

No segundo exemplo, o botão apresenta alto contraste, mantendo a cor de fundo como preta (`#000000`) e o texto em branco (`#ffffff`).

Figura 3 – Botão com texto em contrate alto

A organização WCAG (2023) também disponibiliza alguns padrões de cores para evidenciar o alto contrate, são alguns pares apresentados, como por exemplo, amarelo (`#FFD700`) com preto, azul (`#4169E1`) com branco, e verde (`#00FF7F`) com preto. Essas combinações proporcionam um alto contraste visual. Exemplos dessas combinações podem ser visualizados nas Figuras 4, 5 e 6, respectivamente.

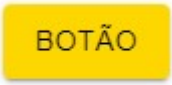

Figura 4 – Botão amarelo - Contraste alto

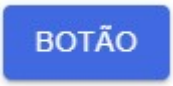

Figura 5 – Botão azul - Contraste alto

Figura 6 – Botão verde - Contraste alto

Um exemplo de texto (*label*) pode ser observado na Figura 7.

Label

Figura 7 – Texto

Ícones e elementos visuais – `color: black; background-color: white`. Um exemplo de utilização é mostrado na Figura 8.

Figura 8 – Letter Avatar

Um aplicação para alertas e mensagens de erro é apresentado na Figura 9 – `color: red (#97031F); background-color: white`.

Figura 9 – Alerta de erro

Botões pode ser utilizados como apresentado na Figura 10 – `color: white; background-color: black`.

Figura 10 – Exemplo para botão branco

A utilização de *links* pode ser realizada conforme apresentado na Figura 11 – `color: blue(#4169E1); background-color: black`.

Figura 11 – Exemplo para link

3.6.1.2 Padrões texturizados

Os padrões texturizados referem-se à aplicação de texturas ou padrões visuais em elementos de interface de usuário para melhorar a acessibilidade e usabilidade, especialmente para pessoas com deficiências visuais ou sensitivas. Esses padrões podem ser utilizados em botões, ícones, *backgrounds*, ou outros elementos visuais para fornecer uma representação tátil ou diferenciada, ajudando os usuários a distinguir entre diferentes elementos e interagir com a interface de forma mais eficaz (PILLEGI, 2021).

Ao criar uma biblioteca de componentes com foco em acessibilidade, é importante considerar a utilização de padrões texturizados como parte das diretrizes de *design*. Isso pode incluir a definição de padrões de textura específicos para diferentes tipos de elementos, como botões, *links*, campos de entrada, dentre outros. Além disso, é importante fornecer orientações claras sobre como aplicar esses padrões de forma consistente em toda a biblioteca de componentes (PILLEGI, 2021; SANTOS, 2020).

A seguir são apresentados as aplicações indicadas para a aplicação de texturização de acordo com cada grupo de componentes:

- Botões:

  Os botões podem usar padrões texturizados para destacar sua importância ou função. Por exemplo, um botão de ação importante pode usar um padrão grande, redondo e contrastante.

- Links:

Os *links* podem usar padrões texturizados para ajudar a distingui-los de outros elementos. Por exemplo, um *link* pode usar um padrão pontilhado ou ondulado.

- Icones:

  Os ícones podem usar padrões texturizados para fornecer *feedback* tátil para pessoas com deficiência visual. Por exemplo, um ícone de botão pode usar um padrão de relevo.

A seguir, são apresentados exemplos de botões em tamanhos pequeno, médio e grande, conforme ilustrado nas Figuras 12, 13 e 14, respectivamente. Destaca-se que o tamanho grande é reservado para ações de maior importância. Os exemplos ilustram a capacidade do componente botão em adaptar-se a diferentes tamanhos, proporcionando flexibilidade e adequação às necessidades do usuário.

Figura 12 – Botão pequeno

Figura 13 – Botão médio

Figura 14 – Botão grande

A seguir, o componente *LetterAvatar* é apresentado nas Figuras 15, 16, 17 em diferentes formatos. Ele oferece diversas opções de texturas para a renderização de avatares. Entre as opções disponíveis, destacam-se as moldura arredondada, quadrada e circular. Essa variedade de texturas proporciona ao usuário uma experiência personalizada, permitindo a escolha da opção que melhor se adapta ao contexto de uso.

Figura 15 – LetterAvatar arredondado

Figura 16 – LetterAvatar quadrado

Figura 17 – LetterAvatar redondo

3.7 Considerações finais

O capítulo abordou o desenvolvimento de uma biblioteca de componentes eficientes para interfaces WEB modernas, com destaque para *React*, *TypeScript*, *Storybook*, *GitHub* e *Material UI*. O *React* e o *TypeScript* são destacados por sua capacidade de criar componentes reutilizáveis e de fácil manutenção, enquanto o *Storybook* é ressaltado como uma ferramenta valiosa para documentação e teste interativo. O *GitHub* é mencionado como essencial para gerenciamento de código e colaboração, e o *Material UI* é elogiado por sua vasta coleção de componentes prontos para uso. Além disso, a aplicação de estratégias para garantir a acessibilidade dos componentes são discutidas, como o uso de cores de alto contraste e padrões texturizados.

4 Análise dos resultados

A garantia da acessibilidade em interfaces WEB é uma preocupação fundamental para promover a inclusão digital e atender a diversos públicos, incluindo pessoas com deficiências visuais. Nesse contexto, a definição adequada das cores desempenha um papel crucial. Este capítulo explora as estratégias adotadas para assegurar o cumprimento das diretrizes de acessibilidade. Os detalhes da aplicação, são discutidos no presente capítulo.

4.1 Definição das cores

Os *sites* utilizados para avaliar o contraste das cores foram o *Verificador de Contraste*[1] e o *Color Contrast Analyzer*[2]. Durante o desenvolvimento da aplicação, esses recursos desempenharam um papel importante na garantia da conformidade com as diretrizes de acessibilidade, incluindo o requisito de contraste AAA estabelecido pela W3C [3], esses requisitos incluem a razão de contraste de texto e a razão de contraste de texto grande e negrito.

Para texto normal, de acordo com o padrão de contraste AAA da W3C, o texto normal em um site deve ter um contraste de pelo menos 7:1 em relação ao plano de fundo. Isso significa que a diferença entre a cor do texto e a cor de fundo deve ser significativa o suficiente para garantir uma boa legibilidade (RAFAEL, 2013).7 Um contraste maior significa uma maior diferença entre as cores, tornando o texto mais fácil de ler para pessoas com deficiências visuais ou em condições de iluminação desfavoráveis. Já para textos grande ou em negrito, o padrão de contraste exigido é um pouco menor, mas ainda significativo. De acordo com a mesma diretriz AAA, o texto grande ou em negrito deve ter um contraste de pelo menos 4.5:1 em relação ao plano de fundo. Isso reconhece que textos maiores ou em negrito podem ser mais fáceis de ler mesmo com um contraste ligeiramente menor, mas ainda é necessário garantir uma legibilidade adequada (RAFAEL, 2013). A definição das cores foi realizada no arquivo *theme.ts*.

No contexto do desenvolvimento de interfaces utilizando o MUI, o arquivo *theme.ts* assume uma importância significativa ao estabelecer as características visuais essenciais da aplicação. Ele encapsula os parâmetros estilísticos fundamentais, como cores, tipografia e espaçamento, em um objeto temático centralizado. A função *createTheme*, fornecida pelo MUI, possibilita aos desenvolvedores configurar um tema coeso e uniforme para toda a aplicação. A definição consistente do tema promove a manutenção eficiente do design

[1] https://corhexa.com/verificador-contraste
[2] https://color.adobe.com/pt/create/color-contrast-analyzer
[3] Documentação oficial:<https://www.w3.org>

visual, assegurando a coerência estética em todas as partes da aplicação.

No código abaixo é apresentada a implementação do tema neste trabalho. Os estilos mais utilizados, *primary* e *secondary*, foram cuidadosamente ajustados para garantir as diretrizes do contraste AAA para os respectivos estilos de texto *primary* e *secondary*. Os estilos *success*, *error*, *warning* e *info* seguem uma dinâmica similar, apresentando em seu escopo *main* a cor sólida esperada do componente utilizado e *contrastText* a cor do texto dentro do elemento.

```
1  import { createTheme } from "@mui/material/styles";
2
3  const theme = createTheme({
4    palette: {
5      primary: {
6        main: process.env.COLOR_PRIMARY !== undefined ? process.
             env.COLOR_PRIMARY : '#151F30',
7      },
8      secondary: {
9        main: process.env.COLOR_SECONDARY !== undefined ? process.
             env.COLOR_SECONDARY : '#103778',
10     },
11     text: {
12       primary: process.env.TEXT_PRIMARY !== undefined ? process.
             env.TEXT_PRIMARY : '#fafafa',
13       secondary: process.env.TEXT_SECONDARY !== undefined ?
             process.env.SECONDARY : '#fafafa',
14     },
15     success: {
16       main: process.env.SUCCESS_MAIN !== undefined ? process.env
             .SUCCESS_MAIN : '#0593A2',
17       contrastText: process.env.SUCCESS_CONTRAST_TEXT!==
             undefined ? process.env.SUCCESS_CONTRAST_TEXT : '#
             fafafa',
18     },
19     error: {
20       main: process.env.ERROR_MAIN !== undefined ? process.env.
             ERROR_MAIN : '#7F0000',
21       contrastText: process.env.ERROR_CONTRAST_TEXT !==
             undefined ? process.env.ERROR_CONTRAST_TEXT : '#fafafa'
             ,
22     },
23     warning: {
```

```
24        main: process.env.WARNING_MAIN !== undefined ? process.env
              .WARNING_MAIN : '#8F3B07',
25        contrastText: process.env.WARNING_CONTRAST_TEXT !==
              undefined ? process.env.WARNING_CONTRAST_TEXT : '#
              fafafa',
26      },
27      info:{
28        main: process.env.INFO_MAIN !== undefined ? process.env.
              INFO_MAIN : '#4971AD',
29        contrastText: process.env.INFO_CONTRAST_TEXT !== undefined
               ? process.env.INFO_CONTRAST_TEXT : '#fafafa',
30      },
31
32    },
33  });
34
35  export default theme;
```

Algoritmo 4.1 – Componente: Theme.tsx

O arquivo "themeProvider.tsx" desempenha um papel importante na disseminação do tema definido pelo arquivo "theme.tsx" por toda a aplicação. Este componente, conhecido como *ThemeProvider*, encapsula a aplicação e fornece acesso ao tema para todos os componentes descendentes. Ao envolver a estrutura da aplicação com o *ThemeProvider* e disponibilizar o tema definido, os desenvolvedores garantem que todos os elementos da interface tenham acesso consistente às configurações de estilo definidas. Isso simplifica significativamente o processo de estilização e promove a coesão visual em toda a aplicação.

```
1  interface Props {
2    children: JSX.Element[] | JSX.Element}
3
4  const ThemeProvider: React.FC<Props> = ({ children }) => (
5    <StyledEngineProvider injectFirst>
6      <ThemeProviderMaterial theme={theme}>
7        <ThemeProviderEmotion theme={theme}>
8          {children}
9        </ThemeProviderEmotion>
10     </ThemeProviderMaterial>
11   </StyledEngineProvider>
12 );
13
14 export { ThemeProvider };
```

Algoritmo 4.2 – Componente: ThemeProvider.tsx

Dessa maneira, a biblioteca de componentes não precisa listar repetidamente as cores desejadas para cada componente em todos os locais. Isso mantém um dos princípios fundamentais de padronização de estilo em uma biblioteca de componentes. Por exemplo, supondo desejar que um componente tenha a cor *primary*, basta utilizar a chamada no código de estilização, conforme o trecho abaixo:

```
1 background: \${theme.palette.primary.main};
```

Algoritmo 4.3 – Exemplo: Utilizando cor do tema

Assim, não é necessário utilizar o código em hexadecimal para especificar a cor, evitando assim erros de digitação e cores incorretamente aplicadas pelo desenvolvedor.Os arquivos desempenham funções distintas, porém complementares, no processo de estilização e design de aplicações utilizando o MUI. Eles contribuem para a coesão visual, a manutenção eficiente do código e a integridade do tipo em projetos *TypeScript*.Além das considerações de contraste feitas neste trabalho, a escolha das cores também se baseou no guia de paletas do Adobe, garantindo harmonia e coesão entre as cores selecionadas.

Todos os textos foram definidos com uma cor estratégica que se aproxima do branco, apresentando alto contraste com quase todas as cores escuras escolhidas como cor sólida de fundo. A cor sólida de fundo escolhida foi verificada para apresentar contraste AAA com a cor do texto. Por exemplo, o componente *ButtonAppBar* utiliza a cor *primary.main* e o texto *text.primary* para assegurar a conformidade com os padrões de acessibilidade e contraste, conforme a Figura 18.

Figura 18 – ButtonAppBar - Menu inicial

Como citado acima, utilizando o *Color Contrast Analyzer*[4], podemos observar na Figura 19 a seguir, como as cores de fundo e cor de texto apresentam o contraste AAA e para textos e ícones.

[4] https://color.adobe.com/pt/create/color-contrast-analyzer

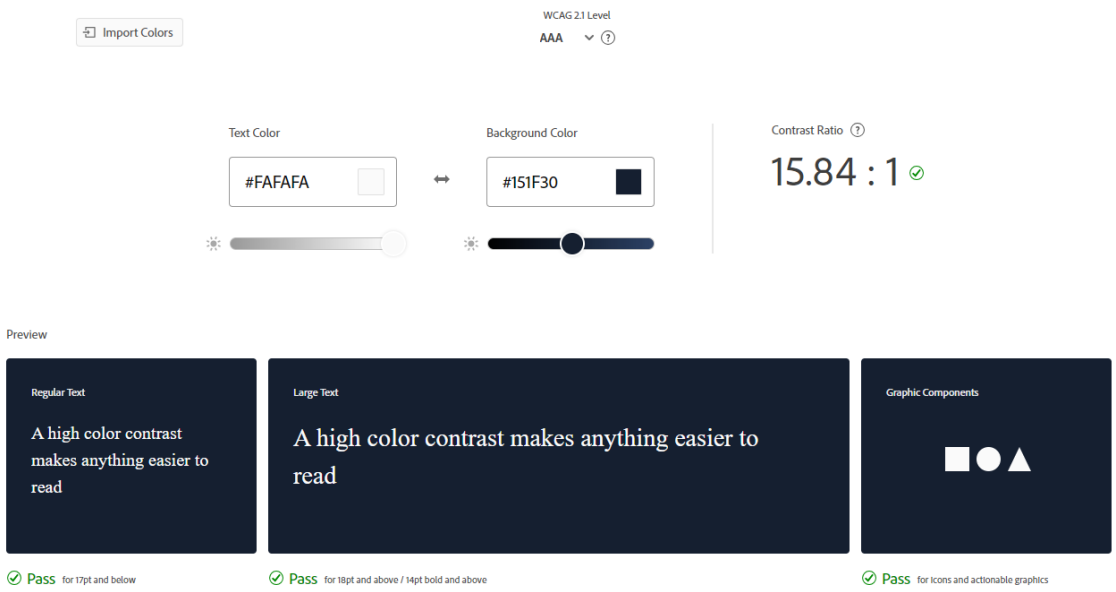

Figura 19 – Avaliação de contraste

4.2 Componentes implementados

A implementação dos componentes e sua organização no projeto foram planejadas para manter uma estrutura arquitetural coesa, que facilitasse o entendimento e mantivesse cada elemento na sua respectiva pasta. Essa abordagem de organização foi orientada pelas práticas recomendadas e pela estruturação de arquivos sugerida na documentação oficial do *React*[5].

Seguindo as diretrizes da documentação, foram adotadas boas práticas de estruturação de arquivos, o que contribuiu para uma melhor organização e manutenção do projeto. Cada componente foi agrupado em sua própria pasta, o que facilitou a localização e a compreensão de sua funcionalidade.

Essa abordagem não apenas promoveu uma organização eficiente do código-fonte, mas também proporcionou uma base sólida para o desenvolvimento escalável e aprimoramentos futuros do projeto.

[5] Arquitetura de pastas: <https://pt-br.legacy.reactjs.org/docs/faq-structure.html>

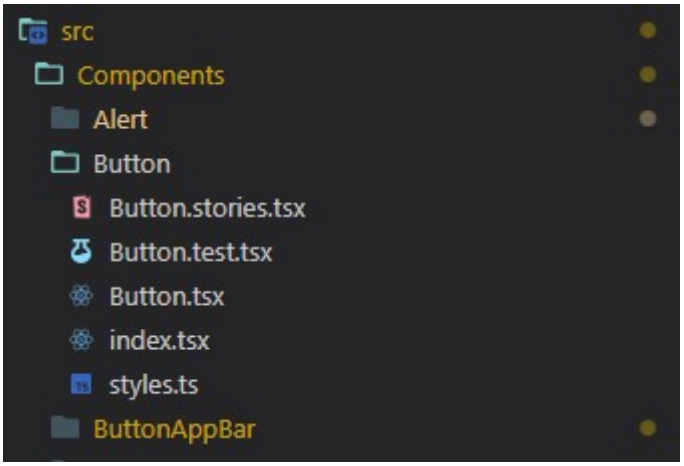

Figura 20 – Arquitetura de pastas

A Figura 20 ilustra parte da organização adotada no projeto. Cada componente é acompanhado pela sua estilização, testes unitários, caminho de exportação e documentação no *Storybook*, todos agrupados na pasta correspondente ao componente. Essa arquitetura foi aplicada de forma consistente em todos os componentes do projeto, proporcionando uma melhor visualização tanto dos componentes quanto das cores utilizadas.

Com o intuito de apresentar mais componentes e cores utilizadas neste projeto, serão apresentados mais exemplos a seguir. Abaixo, é utilizado o mesmo componente apresentado na sessão anterior, *ButtonAppBar*, agora utilizando a cor *secondary.main* e o texto *text.secondary*, conforme mostrado na Figura 21.

Figura 21 – ButtonAppBar - Menu inicial utilizando *secondary color*

Para o componente de alerta, foi necessário analisar e definir três cores distintas: a cor sólida do próprio elemento, a cor do texto contido no elemento e a cor do ícone associado ao mesmo. Essas três cores foram cuidadosamente escolhidas para garantir um contraste AAA e assegurar a destacada visibilidade tanto do texto quanto do ícone dentro do componente *Alert*. Essas cores são exemplificadas a seguir para os estados de sucesso na Figura 22, de erro na Figura 23, de aviso na Figura 24, e informativo na Figura 25 do componente.

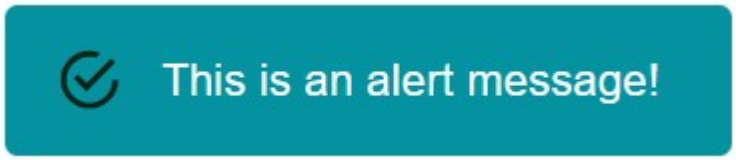

Figura 22 – Alerta de sucesso

Figura 23 – Alerta de erro

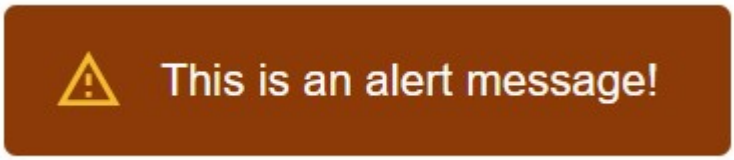

Figura 24 – Alerta de aviso

Figura 25 – Alerta informativo

Um componente da biblioteca não oferece opções de escolha de cores durante o seu desenvolvimento, pois se trata de um *ImageAvatar*. Este componente permite ao usuário, por exemplo, selecionar uma foto de perfil, que será exibida dentro do próprio componente *ImageAvatar*.

Como o conteúdo principal desse componente é a imagem escolhida pelo usuário final, não houve a implementação de configurações de cores para ele. No entanto, é possível apresentar as diferentes texturas que podem ser aplicadas a ele. A estilização do *ImageAvatar* será determinada pelo formato em que ele é exibido.

As opções de textura são representadas pelos formatos com canto arredondado na Figura 26, circular na Figura 27 e quadrado na Figura 28.

Figura 26 – *ImageAvatar* arredondado

Figura 27 – *ImageAvatar* circular

Figura 28 – *ImageAvatar* quadrado

Incluindo os componente apresentados, ao total foram implementados os seguintes: *Alert*, *Button*, *ButtonAppBar*, *ImageAvatar*, *Label*, *LetterAvatar*, *Link* e *Select*. Todos seguem o mesmo padrão de cores, utilizando as cores definidas no tema apresentado.

4.3 Considerações finais

O desenvolvimento de uma biblioteca de componentes para interfaces WEB, não se resume apenas à implementação de funcionalidades, mas também à preocupação com a acessibilidade e a usabilidade. A definição das cores, conforme discutido neste capítulo, desempenha um papel essencial nesse processo, assegurando que todos os usuários possam interagir com a aplicação de maneira eficaz. Ao adotar práticas que garantam o contraste adequado, como evidenciado na implementação dos componentes apresentados, é possível criar interfaces mais inclusivas e acessíveis.

5 Conclusão

A construção de uma biblioteca de componentes eficiente para o desenvolvimento web moderno é fundamental para garantir interfaces robustas e consistentes. Nesse contexto, a utilização de ferramentas adequadas desempenha um papel crucial, capacitando os desenvolvedores a aperfeiçoarem o processo de desenvolvimento e aprimorarem a qualidade do produto final.

Ao longo deste trabalho, foram exploradas diversas tecnologias e metodologias essenciais para o desenvolvimento de uma biblioteca de componentes, incluindo o uso do *React* e *TypeScript* como linguagens de programação e a *React Testing Library* para testes de comportamento do usuário. O *Material UI* foi utilizado para construção de interfaces com componentes pré-estilizados. O *Storybook* foi aplicado para documentação e teste visual dos componentes. A plataforma *GitHub* foi escolhida para gerenciamento de código-fonte e colaboração.

Destaca-se a importância de considerar a acessibilidade ao desenvolver componentes, especialmente para pessoas com daltonismo. Estratégias foram exploradas como o uso de cores de alto contraste e padrões texturizados para garantir que os componentes sejam acessíveis e utilizáveis por todos os usuários.

A partir das diretrizes apresentadas, foi possível desenvolver uma biblioteca de componentes com ênfase na acessibilidade, proporcionando uma experiência de usuário consistente e intuitiva. Através da adoção de práticas recomendadas e tecnologias modernas, os desenvolvedores podem maximizar a eficiência do desenvolvimento web e oferecer produtos de alta qualidade que atendam às necessidades e expectativas dos usuários.

A construção de uma biblioteca de componentes eficiente requer uma abordagem centrada nas melhores práticas, utilizando tecnologias adequadas e considerando sempre a acessibilidade como um aspecto fundamental. Com o uso correto de ferramentas e metodologias, os desenvolvedores podem criar interfaces de usuário de alta qualidade que proporcionem uma experiência excepcional para todos os usuários.

5.1 Trabalhos futuros

A construção de uma biblioteca de componentes acessíveis para aplicações WEB é um processo contínuo e dinâmico. Esta seção apresenta considerações sobre possíveis direções para futuros trabalhos, no qual é possível destacar áreas de melhoria e expansão que podem ser exploradas para aprimorar ainda mais a acessibilidade e usabilidade da biblioteca desenvolvida.

- Embora a biblioteca desenvolvida neste trabalho tenha sido projetada com foco na inclusão de usuários com daltonismo, há espaço para expandir as funcionalidades de acessibilidade para atender a uma gama mais ampla de necessidades. Futuros trabalhos podem explorar a integração de recursos adicionais, como suporte a leitores de tela, controle por voz e navegação por teclado, garantindo que a biblioteca seja ainda mais acessível para todos os usuários, incluindo aqueles com deficiências visuais, motoras ou cognitivas.

- Ampliar a variedade de componentes uma vez que, embora a biblioteca atual ofereça uma variedade de componentes básicos essenciais para o desenvolvimento de interfaces WEB, há oportunidades para expandir e diversificar ainda mais o conjunto de componentes disponíveis. Futuros trabalhos podem incluir o desenvolvimento de novos componentes especializados, como gráficos interativos, controles de formulário avançados e elementos de navegação personalizados, atendendo a uma ampla gama de casos de uso e necessidades de design.

- Aprimorar a documentação e exemplos de uso, pois a documentação clara e abrangente é essencial para facilitar o uso e a adoção da biblioteca por parte dos desenvolvedores. Futuros trabalhos podem focar no aprimoramento da documentação existente, fornecendo exemplos de código mais detalhados, guias de estilo e práticas recomendadas, além de tutoriais passo a passo para auxiliar os usuários na implementação e personalização dos componentes.

- Explorar a integração com outras tecnologias e *Frameworks*. Embora a biblioteca atual seja desenvolvida com base nas tecnologias *React* e *TypeScript*, há oportunidades para explorar integrações com outros *frameworks* e bibliotecas populares. Futuros trabalhos podem investigar a possibilidade de adaptar os componentes para funcionar em outros ecossistemas, como *Angular*, *Vue.js* ou *Svelte*, ampliando assim o alcance e a utilidade da biblioteca para uma variedade de projetos e plataformas.

- Realizar testes de usabilidade e *feedback* dos usuários incluindo a avaliação contínua da acessibilidade e eficácia da biblioteca, pois é fundamental para garantir sua relevância e utilidade a longo prazo. Futuros trabalhos podem incluir testes de usabilidade abrangentes, entrevistas com usuários e coleta de *feedback* para identificar áreas de

melhoria e refinamento. Com base nesse *feedback*, ajustes e atualizações podem ser implementados para aprimorar ainda mais a experiência do usuário e atender às necessidades em constante evolução da comunidade de desenvolvedores.

- Criar mecanismos de expansão da comunidade e colaboração aberta, já que o crescimento e a sustentabilidade da biblioteca dependem do envolvimento ativo da comunidade de desenvolvedores. Futuros trabalhos podem focar na promoção da biblioteca, estabelecendo canais de comunicação abertos, incentivando a colaboração e contribuição de código, e organizando eventos e *workshops* para compartilhar conhecimento e boas práticas. Ao cultivar uma comunidade vibrante e engajada em torno da biblioteca, é possível garantir sua longevidade e sucesso a longo prazo.

Referências

ABRAMOV, D.; NABORS, R. *Introducing react.dev*. 2023. Https://pt-br.react.dev/blog/2023/03/16/introducing-react-dev. Acessado em 10 de Janeiro de 2024. Citado na página 21.

BERTAGLIA, R. *Acessibilidade: exemplos, tipos e como se enquadrar às normas?* 2022. Acessado em 1º de fevereiro de 2024. Disponível em: <https://www.handtalk.me/br/blog/acessibilidade-exemplos/>. Citado 2 vezes nas páginas 18 e 19.

BRAMBILLA, M.; CERI, S.; FRATERNALI, P. Process support for developing web applications with reusable components. *Journal of Web Engineering*, v. 5, n. 2, p. 97–126, 2006. Disponível em: <https://www.rintonpress.com/xjwe.htm>. Citado na página 13.

BRASIL. *Decreto nº 5.296*. 2004. Regulamenta as Leis nos 10.048, de 8 de novembro de 2000, e 10.098, de 19 de dezembro de 2000. Disponível em: <https://www.planalto.gov.br/ccivil_03/_ato2004-2006/2004/decreto/d5296.htm>. Citado na página 18.

BRASIL. *Lei Brasileira de Inclusão da Pessoa com Deficiência (LBI)*. 2015. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2015/lei/l13146.htm>. Citado na página 12.

BRASIL, W. Cartilha de acessibilidade na web. 2020. Disponível em: <https://ceweb.br/publicacao/cartilha-de-acessibilidade-na-web-fasciculo-iii/>. Citado na página 20.

CFM. *Daltonismo: Distúrbio Atinge 5% da População Mundial*. 2022. Disponível em: <https://portal.cfm.org.br/noticias/daltonismo-disturbio-atinge-5-da-populacao-mundial/>. Citado na página 13.

COSTA, R. et al. A acessibilidade de pessoas com daltonismo: A construção de um protótipo de ava inclusivo. *Informática na educação: teoria & prática*, v. 20, 09 2017. Citado 3 vezes nas páginas 12, 19 e 23.

COUTINHO, T. *Descubra o que são Testes Unitários, suas vantagens e confira ferramentas para fazê-lo]*. 2021. Https://www.voitto.com.br/blog/artigo/testes-unitarios. Acessado em 08 de janeiro de 2023. Citado na página 26.

DANA, H. *UX e Acessibilidade: A importância do contraste para a interface*. 2019. Disponível em: <https://medium.com/ui-lab-school/ux-e-acessibilidade-a-import%C3%A2ncia-do-contraste-para-a-interface-fea5ff84ea2c>. Citado na página 33.

DARDE, P. R. *O Padrão Triple A (Arrange, Act, Assert)*. 2020. Acessado em 17 de zembro de 2023. Disponível em: <https://medium.com/@pablodarde/o-padr%C3%A3o-triple-a-arrange-act-assert-741e2a94cf88>. Citado na página 30.

FERNANDA, C.; LOUZADA, V. *O que é Git e Github: os primeiros passos nessas ferramentas*. 2023. Acessado em 9 de janeiro de 2024. Disponível em: <https://www.alura.com.br/artigos/o-que-e-git-github>. Citado na página 31.

FORBES. *Menos de 1% dos sites brasileiros são considerados acessíveis, diz pesquisa*. 2021. Disponível em: <https://forbes.com.br/forbesesg/2021/07/menos-de-1-dos-sites-brasileiros-sao-considerados-acessiveis-diz-pesquisa/>. Citado na página 19.

LOPES, M. B.; TERCIC, L. S. Dificuldades e avanços nos recursos de inclusão para daltônicos. *ComCiência*, n. 215, 2020. Citado 2 vezes nas páginas 12 e 13.

MEDEIROS, N. *Acessibilidade das Cores: Estratégias para um Design Inclusivo*. 2023. Disponível em: <https://rockcontent.com/br/blog/acessibilidade-das-cores-design-inclusivo/>. Citado na página 33.

MELO, D. *O que é TypeScript? [Guia para iniciantes]*. 2021. Https://tecnoblog.net/responde/o-que-e-typescript-guia-para-iniciantes/. Acessado em 20 de dezembro de 2023. Citado na página 22.

MELO, D. G.; GALON, J. E. V.; FONTANELLA, B. J. B. Os "daltônicos"' e suas dificuldades: Condição negligenciada no brasil? *Physis: Revista de Saúde Coletiva*, v. 24, n. 4, p. 1229–1253, 2014. Disponível em: <https://www.scielo.br/scielo.php?pid=S0103-73312014000401229&script=sci_arttext>. Citado na página 13.

MELO, Y. *Biblioteca de Componentes*. 2024. <https://github.com/MeloYasmine/component-library>. Citado na página 31.

MENDES, G. de A. et al. *O Poder das Cores: Moda Inclusiva para Daltônicos*. 2022. Disponível em: <https://ifpr.edu.br/goioere/wp-content/uploads/sites/13/2022/12/Artigo-Final-Daltonismo.pdf>. Citado na página 20.

MESZAROS, G. *xUnit Test Patterns: Refactoring Test Code*. [S.l.]: Addison-Wesley Professional, 2007. Citado na página 26.

Ministério do Planejamento, Desenvolvimento e Gestão. *Cartilha de Contratação de Soluções de Tecnologia da Informação*. 2016. Disponível em: <https://emag.governoeletronico.gov.br/cartilha-contratacao/>. Citado na página 18.

NEIVA, M. Coloradd: Um sistema de identificação de cores para daltônicos. *Revista de Design*, v. 25, n. 1, p. 1–14, 2018. Citado 2 vezes nas páginas 13 e 21.

Organização Mundial da Saúde. *Organização Mundial da Saúde lança primeiro relatório mundial sobre visão*. 2019. Publicado em 8 de outubro de 2019. Disponível em: <https://www.paho.org/pt/noticias/8-10-2019-organizacao-mundial-da-saude-lanca-primeiro-relatorio-mundial-sobre-visao>. Citado 2 vezes nas páginas 16 e 17.

PEREIRA, M. M. *Saúde e bem estar*. 2022. <https://www.saudebemestar.pt/pt/clinica/oftalmologia/daltonismo/>. Acessado em 5 de Janeiro de 2024. Citado na página 17.

PILLEGI, M. Dicas de ux/ui para interfaces mais acessíveis. *Medium*, 2021. Disponível em: <https://brasil.uxdesign.cc/dicas-de-ux-ui-para-interfaces-mais-acess%C3%ADveis-91f5f4c23bd3>. Citado na página 35.

RAFAEL, C. *Cores e acessibilidade*. 2013. Acessado em 25 de janeiro de 2024. Disponível em: <https://carlosrafaelgn.com.br/Aula/Cores.html>. Citado na página 38.

REACT. React resources - components and props. 2021. Acessado em 09 de dezembro de 2023. Disponível em: <https://reactresources.com/topics/components-and-props>. Citado na página 22.

RIBEIRO, H. *Integração do Material UI com ReactJS*. 2018. Acessado em 15 de janeiro de 2024. Disponível em: <https://blog.rocketseat.com.br/react-material-ui/>. Citado na página 31.

SACRAMENTO, G. *O que é React e como usar essa biblioteca JavaScript na programação?* 2023. Https://blog.somostera.com/desenvolvimento-web/o-que-e-react. Acessado em 23 de dezembro de 2023. Citado na página 22.

SANTOS, M. Design digital e acessibilidade — 70 dicas práticas. *Medium*, 2020. Disponível em: <https://brasil.uxdesign.cc/design-digital-e-acessibilidade-70-dicas-pr%C3%A1ticas-99e0324a5b2e>. Citado na página 35.

SILVA, G. L. *Tipos de testes: quais os principais e por que utilizá-los?* 2021. Https://www.alura.com.br/artigos/tipos-de-testes-principais-por-que-utiliza-los. Acessado em 08 de janeiro de 2023. Citado na página 26.

SILVA, L. C. D.; OLIVEIRA, J. V. de. Acessibilidade em interfaces de usuário: Um estudo sobre a percepção de usuários com daltonismo. *Revista de Sistemas e Computação*, v. 8, n. 1, p. 1–10, 2018. Citado na página 13.

SMITH, A. R.; JONES, R. L. Daltonismo. In: EAGLE, R. K.; ROSENBLOOM, R. A. (Ed.). *Oftalmologia Clínica*. 12. ed. Local de publicação, se disponível: Saunders Elsevier, 2014. cap. Número do capítulo, se disponível, p. Páginas do capítulo. Outras informações, se necessário. Citado na página 17.

SWAMINATHAN, R.; KULKARNI, A. A study on the impact of color vision deficiency in user interface design. *Procedia Computer Science*, v. 115, p. 218–225, 2017. Citado 2 vezes nas páginas 13 e 21.

TEIXEIRA, J. *Gestão visual de projetos: utilizando a informação para inovar*. Alta Books, 2018. ISBN 9788550801711. Disponível em: <https://books.google.com.br/books?id=eQ_3zwEACAAJ>. Citado 2 vezes nas páginas 12 e 16.

VACCANI, J. Ilustrações acessíveis: Melhorando experiências digitais para usuários com daltonismo. *UX Collective*, 2023. Citado na página 13.

VALADARES, L. et al. Upwards! um jogo acessível que incentiva inclusão através da competição. In: *Anais Estendidos do XXI Simpósio Brasileiro de Jogos e Entretenimento Digital*. Porto Alegre, RS, Brasil: SBC, 2022. p. 1450–1454. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/sbgames_estendido/article/view/23799>. Citado na página 23.

W3C. *Web Content Accessibility Guidelines (WCAG)*. 2023. <https://www.w3.org/WAI/standards-guidelines/wcag/>. Acessado em 10 dess dezembro de 2023. Citado na página 33.

WCAG. *Color-Carnegie Museums Web Accessibility*. 2023. <http://web-accessibility.carnegiemuseums.org/design/color/>. Acessado em 20 de dezembro de 2023. Citado na página 33.

World Wide Web Consortium (W3C). *Web Content Accessibility Guidelines (WCAG) 2.1*. [S.l.], 2018. Versão atualizada em dezembro de 2018. Disponível em: <https://www.w3.org/WAI/WCAG21/quickref/>. Disponível em: <https://www.w3.org/WAI/WCAG21/quickref/>. Citado na página 12.

ZHOU, M.; ZHANG, R.; LIU, Y. A methodology for component-based software development. *IEEE Transactions on Software Engineering*, v. 33, n. 10, p. 702–722, 2007. Citado na página 13.