UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO SISTEMAS DE INFORMAÇÃO

HIGOR PIRES OLIVEIRA

DESENVOLVIMENTO DE UMA FERRAMENTA *WEB* PARA SUPORTE AO DESIGN VISUAL DE INTERFACE DE USUÁRIO DE APLICATIVOS MÓVEIS

FLORIANÓPOLIS
2024

HIGOR PIRES OLIVEIRA

DESENVOLVIMENTO DE UMA FERRAMENTA *WEB* PARA SUPORTE AO DESIGN VISUAL DE INTERFACE DE USUÁRIO DE APLICATIVOS MÓVEIS

Trabalho de Conclusão do Curso de Graduação em Sistemas de Informação, do Departamento de Informática e Estatística, do Centro Tecnológico da Universidade Federal de Santa Catarina, requisito parcial à obtenção do título de Bacharel em Sistemas de Informação.

Orientadora: Prof.ª Dr.ª rer. nat. Christiane Gresse von Wangenheim, PMP

FLORIANÓPOLIS
2024

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.

Pires Oliveira, Higor
DESENVOLVIMENTO DE UMA FERRAMENTA WEB PARA SUPORTE AO DESIGN VISUAL DE INTERFACE DE USUÁRIO DE APLICATIVOS MÓVEIS / Higor Pires Oliveira ; orientadora, Christiane Gresse von Wangenheim, 2024.
79 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Sistemas de Informação, Florianópolis, 2024.

Inclui referências.

1. Sistemas de Informação. 2. Ensino de Computação. 3. Design visual. 4. Aplicativo móvel. 5. Educação Básica. I. Gresse von Wangenheim, Christiane. II. Universidade Federal de Santa Catarina. Graduação em Sistemas de Informação. III. Título.

HIGOR PIRES OLIVEIRA

DESENVOLVIMENTO DE UMA FERRAMENTA *WEB* PARA SUPORTE AO DESIGN VISUAL DE INTERFACE DE USUÁRIO DE APLICATIVOS MÓVEIS

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do título de Bacharel e aprovado em sua forma final pelo Curso Sistemas de Informação.

Florianópolis, 10 de junho de 2024.

_____

Coordenação do Curso

Banca examinadora

_____

Prof.ª Dr.ª rer. nat. Christiane Gresse von Wangenheim, PMP

Orientador(a)

_____

Prof. Dr. Jean C. R. Hauck

Co-orientador(a)

_____

Miriam Nathalie Fortuna Ferreiramm, MSc

Florianópolis, 2024.

RESUMO

O desenvolvimento tecnológico dos últimos anos tornou a computação indispensável no cotidiano da população, tanto na carreira profissional quanto na vida pessoal e, dessa forma, tornou-se essencial o ensino de computação na Educação Básica. Neste estágio escolar, a computação pode ser ensinada por meio do desenvolvimento de aplicativos móveis, utilizando ferramentas baseadas em blocos como o App Inventor, que permite não só programar as funcionalidades do aplicativo, mas também projetar o design de interface. Assim, pode ser ensinado também o *design* visual abordando conceitos como *layout*, cores, tipografia entre outros. Atualmente o *design* de interface de aplicativos criados com App Inventor é tipicamente feito diretamente no App Inventor, sem que seja fornecido um suporte para o *design* visual. Assim, observa-se que muitos aplicativos criados nesse contexto educacional apresentam uma interface pouco atrativa, violando vários princípios de *design* visual. Portanto, o objetivo do presente trabalho é a criação de uma ferramenta *web* que suporte o *design* visual de interface de usuário de *apps* com App Inventor no contexto de ensino de computação na Educação Básica. Com isso, espera-se facilitar significativamente o ensino de conceitos de design no contexto da computação nas escolas brasileiras, promovendo uma aprendizagem mais prática e intuitiva para os estudantes e contribuindo diretamente para a formação de habilidades em design visual durante o desenvolvimento de aplicativos móveis.

Palavras-chave: Ensino de Computação, Design visual, Aplicativo móvel, Educação Básica.

ABSTRACT

The technological advancements of recent years have made computing indispensable in the daily lives of the population, both in professional careers and personal life. As a result, it has become essential to teach computing in K-12. At this educational stage, computing can be taught through the development of mobile applications using block-based tools such as App Inventor, which allows not only the programming of application functionalities but also the design of the interface. This approach enables the teaching of visual design by addressing concepts such as layout, colors, typography, among others. Currently, the interface design of applications created with App Inventor is typically done directly within the App Inventor environment, without providing significant support for visual design. Consequently, many applications created in this educational context have unappealing interfaces that violate several principles of visual design. Therefore, the objective of this work is to create a web tool that supports the visual interface design of user apps with App Inventor in the context of computing education in K-12. This aims to facilitate the teaching of design concepts in the context of computing in Brazilian schools, thereby contributing to the students' education.

Keywords: Computing Education, Visual Design, Mobile Application, K-12.

SUMÁRIO

1. INTRODUÇÃO

Nos dias de hoje, a tecnologia vem desempenhando um papel de cada vez mais protagonismo em nossa sociedade (RAJA e NAGASUBRAMANI, 2018). Esse papel de protagonismo se estende por todas as áreas da sociedade que buscam se beneficiar por meio do uso de tecnologia. Por isso, cada vez mais profissionais nessa área serão requisitados pelo mercado. Além disso, de acordo com o World Economic Forum (2020), espera-se que cada vez mais as pessoas tenham familiaridade com o uso de recursos tecnológicos para desempenhar os mais variados tipos de tarefas, principalmente em relação à computação, já que a adoção de tecnologia em diferentes setores e indústrias é uma realidade que veio para ficar. No meio de tanta novidade e mudanças no mundo, é esperado que as escolas possam ser a base das transformações desde o nível de Educação Básica e sejam capazes de formar estudantes capacitados para este mundo com a crescente demanda por habilidades tecnológicas (CIEB, 2019). Por isso, o ensino de computação na Educação Básica é importante. Além disso, cabe ressaltar que em 2022, por meio de uma portaria, o Ministério da Educação incluiu a computação no currículo da Educação Básica, demonstrando mais uma vez a sua importância (MEC, 2022).

Uma das maneiras para ensinar computação na Educação Básica de maneira interativa, simples e eficiente é por meio do desenvolvimento de aplicativos móveis (MEDEIROS et al., 2020). Nesse nível educacional, de acordo com Medeiros et al. (2021), "tipicamente há o foco no ensino de algoritmos e programação em ambientes de programação visual baseados em bloco, como […] App Inventor", já que essas plataformas possuem uma curva de aprendizagem menor e necessidade de pouca ou nenhuma experiência em programação para começar, o que facilita a introdução de computação para esses alunos. O App Inventor é uma ferramenta visual de *drag-and-dro*p para construir aplicativos Android, na qual é possível construir o *design* de interface (aparência visual do aplicativo) do app usando uma interface *web* visual simples e intuitiva, além de definir o comportamento do aplicativo por meio do uso de blocos de programação (WOLBER, 2011).

Assim, junto com o ensino de conceitos de algoritmos e programação, pode-se ensinar também conceitos de design de interface de usuário (*User Interface* - UI), que estão se tornando cada vez mais importantes devido à sua relação com as habilidades do século XXI (AIGA, 2013). De acordo com Ferreira et al. (2020), o ensino desses conceitos na Educação Básica "proporciona diversos benefícios, incluindo a aprendizagem de métodos de pesquisa,

visualização e apresentação de informações, incentivo à análise crítica e colaboração e treinamento" e pode despertar um lado mais imaginativo nos alunos (AIGA, 2013).

Dentre os conceitos do design de UIs que podem ser abordados, está o design visual, que busca desenvolver a forma como as pessoas interagem com as interfaces dos sistemas, buscando criar produtos que atendam às necessidades dos usuários de maneira eficiente e satisfatória. O design visual tem efeitos na usabilidade e na atratividade estética dos sistemas para dar forma à experiência de usuário e melhorá-la por meio de meta-princípios, incluindo consistência, hierarquia e personalidade (SCHLATTER; LEVINSON, 2013). Na área de design de interfaces, isso também inclui conceitos como, por exemplo, *layout*, que se trata do posicionamento estruturado de componentes na interface; *tipografia,* que define o formato das letras e demais caracteres; *cores,* que são usadas para chamar a atenção para certos componentes, ajudando o usuário a identificá-los com maior facilidade; entre outros.

Porém, observa-se na prática que a estética visual de muitos aplicativos sendo criados com App Inventor apresentam pouca atratividade e violam diversos princípios de design visual. Resultados de uma análise sobre o design visual das interfaces de usuário de 88.861 aplicativos da galeria do App Inventor, quanto à conformidade com guias de estilo por meio da análise automatizada do código-fonte, bem como sua estética com base em um *survey* com 95 participantes avaliando 110 interfaces de usuário (Solecki et al., 2020), mostra que a maioria das IUs não está em conformidade com diretrizes de design e não possui boa estética visual. Assim, mostra-se importante abordar não apenas conceitos de programação, mas também o design de UI no ensino de computação por meio do desenvolvimento de aplicativos móveis.

Ao trabalhar especificamente com o App Inventor para construção de aplicativos Android, podem ser ensinados os conceitos de design de UIs por meio da prototipação de interfaces e fazer uso de algumas ferramentas e materiais que auxiliem nesse processo. O design de interface de aplicativos móveis, por exemplo, pode ser orientado por guias de estilo que padronizam o design visual de interfaces, como, por exemplo, o Material Design 3 (GOOGLE, 2022), que fornece recomendações para o design de interface de aplicativos Android, abordando elementos como cores, tipografia, layout e componentes. Além disso, diretrizes de acessibilidade como as *Web Content Accessibility Guidelines* 2.1 (WCAG 2.1; W3C, 2018) podem orientar o design visual para assegurar a acessibilidade do conteúdo para o maior número possível de usuários.

O design de interfaces de usuário (UIs) pode ser realizado com aplicativos de design gráfico como o Figma (FIGMA, 2023) ou com alternativas gratuitas e de código aberto como

o Penpot (PENPOT, 2023). Embora existam vários templates e plugins que fornecem suporte adicional no design visual, como na escolha de paletas de cores harmônicas com contraste adequado, atualmente não há uma ferramenta ou plugin que ofereça suporte completo ao processo de design visual. Outra desvantagem das ferramentas e plugins existentes é que eles não necessariamente focam no design de interfaces de aplicativos móveis, seguindo as diretrizes específicas dessas plataformas. Além disso, muitas dessas ferramentas possuem funcionalidades complexas projetadas para profissionais de design, tornando-se menos acessíveis para iniciantes no contexto da educação básica.

Portanto, este trabalho propõe a criação de uma ferramenta *web* que suporte o design visual de UIs de aplicativos no App Inventor, especificamente no contexto do ensino de computação na educação básica.

1.1. Objetivos

Objetivo geral

O objetivo deste trabalho é o desenvolvimento de uma ferramenta *web* para suportar o design visual de interfaces de usuário de apps a ser utilizada em cursos no ensino de computação na Educação Básica voltados ao desenvolvimento de aplicativos móveis com App Inventor.

Objetivos específicos

O1. Sintetizar a fundamentação teórica sobre design de interface de usuário de apps (App Inventor).

O2. Analisar o estado da arte sobre plugins que oferecem suporte ao design de UIs de apps em ferramentas de design gráfico.

O3. Desenvolver uma ferramenta *web* para o suporte ao design de UIs de apps.

1.2. Premissas e restrições

O trabalho é realizado de acordo com o regulamento vigente do Departamento de Informática e Estatística (INE – UFSC) em relação aos Trabalhos de Conclusão de Curso.

A ferramenta desenvolvida tem como foco o suporte somente para o design visual de apps criados com App Inventor, não abordando outras etapas do processo de design e/ou implementação. Como o contexto do presente trabalho é o ensino de computação na Educação Básica, o suporte é voltado a objetivos de aprendizagem relacionados a este nível educacional.

1.3. Metodologia de pesquisa

A metodologia de pesquisa utilizada neste trabalho é dividida em três etapas.

Etapa 1 – Fundamentação teórica: Estudando, analisando e sintetizando os conceitos principais e a teoria referente aos temas a serem abordados neste trabalho. Nessa etapa são realizadas as seguintes atividades:

A1.1 – Análise teórica sobre o design de interfaces de usuário de apps App Inventor.

A1.2 – Análise teórica sobre design visual.

Etapa 2 – Estado da arte: Nessa etapa é realizado um mapeamento sistemático de plugins que dão suporte ao design de UIs seguindo o processo proposto por Petersen et al. (2008) para identificar e analisar plugins. Essa etapa é dividida nas seguintes atividades:

A2.1 – Definição do protocolo da revisão.

A2.2 – Execução da busca por plugins.

A2.3 – Extração e análise de informações relevantes.

Etapa 3 – Desenvolvimento: Nessa etapa é desenvolvida uma ferramenta *web* para suporte ao design de UIs de apps App Inventor seguindo um processo de engenharia de software proposto por Pressman (2016). Essa etapa é dividida nas seguintes atividades:

A3.1 – Análise de requisitos.

A3.2 – Modelagem da arquitetura da ferramenta.

A3.3 – Modelagem detalhada e implementação.

A3.4 – Testes da ferramenta.

1.4. Estrutura do documento

O restante deste documento está estruturado em vários capítulos, visando uma abordagem organizada e abrangente do tópico. No Capítulo 2, a Fundamentação Teórica, são abordados os conceitos e teorias fundamentais que embasam este trabalho. Isso inclui informações sobre o design de interfaces de usuário no contexto do App Inventor, os princípios de design visual, elementos como cores, contraste, tipografia e imagens, além das diretrizes de design de interface. A seção também explora as características da ferramenta Penpot.

O Capítulo 3, Estado da Arte, começa com uma descrição do protocolo de revisão adotado, detalhando como a busca por plugins foi realizada. Os resultados da revisão são discutidos, incluindo a identificação dos plugins existentes e uma análise de suas funcionalidades em relação aos aspectos do design visual.

O Capítulo 4 concentra-se em abordar a solução desenvolvida, bem como seus requisitos funcionais.

Por fim, no Capítulo 5, é apresentada a conclusão do trabalho.

2. FUNDAMENTAÇÃO TEÓRICA

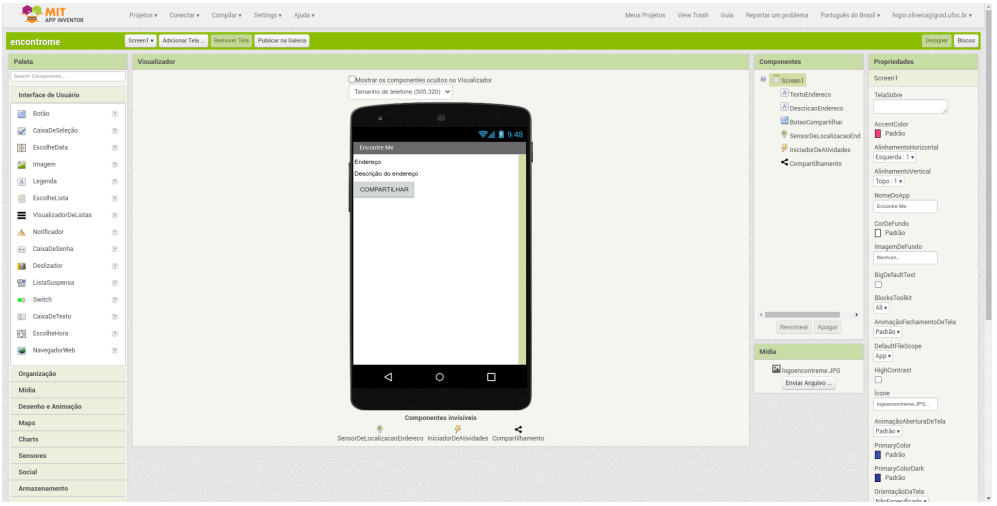
Nesta seção, são apresentados os principais conceitos e ferramentas que embasam o desenvolvimento da ferramenta proposta neste trabalho. Inicialmente, é discutido o design de interfaces de usuário de apps App Inventor, uma plataforma de código aberto que permite a criação de aplicativos móveis para Android. Em seguida, é abordado o design de interface de usuário e sua importância para a usabilidade, a estética e a comunicação dos aplicativos.

2.1. Design de interfaces de usuário de apps App Inventor

2.1.1. App Inventor

O App Inventor é uma ferramenta gratuita de código aberto desenvolvida pela Google em 2010 e atualmente mantida pelo Massachusetts Institute of Technology (MIT) (MIT, 2023). A ferramenta permite a criação de aplicativos móveis para Android sem a necessidade de conhecimento prévio em programação, de forma a auxiliar o ensino de computação por meio do desenvolvimento de aplicativos com uso de blocos, que permitem criar a programação das funcionalidades do app (MIT, 2023).

Figura 1 — Interface do App Inventor

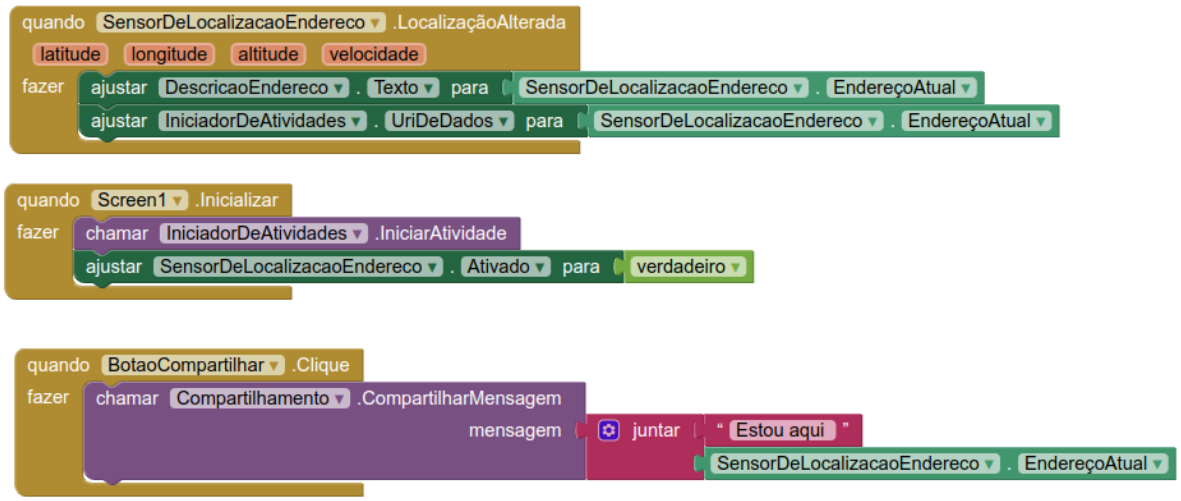


Fonte: Elaborada pelo autor

O App Inventor possui dois elementos principais para construção de aplicativos, o Designer e o Editor de Blocos, que juntos possibilitam o desenvolvimento de aplicativos móveis de maneira rápida e intuitiva (WOLBER, 2011). O Designer trata-se do primeiro

recurso a ser utilizado para construção dos apps e permite a construção da interface gráfica do aplicativo, ou seja, os elementos visuais que o usuário vê e interage na tela. Por outro lado, o Editor de Blocos possibilita programar o comportamento do app, ou seja, as funcionalidades do aplicativo. A programação em blocos é uma forma de representar algoritmos por meio do uso de blocos que se conectam de forma parecida a peças de um quebra-cabeça, sendo que cada bloco representa uma instrução, variável, condição, laço, função ou um evento (MUSTAFARAJ; TURBAK; SVANBERG, 2017). Dessa forma, a programação em blocos facilita o aprendizado de conceitos básicos de computação, como sequência, seleção, repetição e abstração, estimulando o pensamento computacional, a criatividade e a resolução de problemas (ALVES et al., 2021).

Figura 2 — Estrutura de blocos do App Inventor



Fonte: Elaborada pelo autor

Uma das facilidades do App Inventor é a visualização do funcionamento do aplicativo em construção de maneira simplificada e rápida, de forma a tornar possível testar um aplicativo desenvolvido à medida que o mesmo é evoluído. Para isso, um aplicativo chamado App Inventor Companion deve ser baixado da loja Google Play e configurado de maneira a possibilitar testes em tempo real do app construído com o App Inventor.

De maneira similar, a geração do arquivo necessário para instalação e disponibilização do aplicativo feito com o App inventor (.apk) também é simplificada, e pode ser feita diretamente pela interface web da ferramenta. O App Inventor fornece as opções de download nas extensões .apk e .aab, que são utilizadas para instalar aplicativos Android e para disponibilização dos mesmos na loja Google Play, tornando possível disponibilizar o aplicativo desenvolvido para outras pessoas (MIT, 2023).

2.1.2. Construindo interfaces de usuário no App Inventor

No App Inventor, o design de interface de usuário é feito no Designer (WOLBER, 2011), o qual oferece uma variedade de componentes que podem ser utilizados para desenvolver interfaces de usuário interativas. Embora esses recursos não sejam tão avançados como em ferramentas de design gráfico dedicadas para a utilização de ferramentas próprias para construção de aplicativos, eles permitem exercitar conceitos de design visual, como layout, cores, tipografia e elementos interativos. Esses componentes estão divididos em algumas categorias (MIT, 2023), sendo que os componentes relacionados à construção da interface do usuário podem ser encontrados nas categorias *User Interface* e *Layout*. Na Tabela 1 é apresentada uma lista de todos os componentes relacionados à construção da interface de usuário que são atualmente disponibilizados pelo *core* do App Inventor. Cada um desses componentes possui propriedades que podem ser alteradas para personalizar sua aparência e seu comportamento, por exemplo, em relação à cor de fundo ou texto, tipo e/ou tamanho da fonte etc.

Tabela 1 — Componentes visuais do App Inventor

Componentes visuais		
Categoria	**Componente**	**Descrição**
User Interface	Button	Botão com a habilidade de detectar cliques.
	CheckBox	Caixa de seleção que pode disparar um evento quando o usuário clica nela.
	DatePicker	Um botão que, quando clicado, inicia um diálogo que permite ao usuário selecionar uma data.
	Image	Componente para apresentação de imagens.
	Label	Uma Legenda mostra um trecho de texto, que é especificado através da propriedade Texto
	ListPicker	Um botão que, quando clicado, mostra uma lista de textos para o usuário escolher.
	ListView	Este é um componente visível que permite colocar uma lista de elementos de texto para apresentação na sua Tela
	Notifier	O componente Notificador mostra diálogos de alerta, mensagens, alertas temporários e cria entradas de log Android.
	TextBox	Uma caixa para o usuário inserir texto.
	PasswordTextBox	Uma caixa para digitar senhas. O mesmo que o componente CaixaDeTexto comum, exceto por não exibir os caracteres digitados pelo usuário.
	Slider	Um Deslizador é uma barra de progresso que adiciona um indicador arrastável. Você pode tocar no indicador e arrastar para a esquerda ou para a direita para ajustar sua posição.
	Spinner	Um componente para seleção que exibe um popup com uma lista de elementos.
	Switch	Botão de alternância que gera um evento quando o usuário clica nele.
	TimePicker	Um botão que, quando clicado, inicia um diálogo que permite ao usuário selecionar um horário.
	WebViewer	Componente para visualizar páginas da web.
Organização	HorizontalArrangement	Um elemento de formatação onde colocar componentes que devem ser mostrados da esquerda para a direita;

	VerticalArrangement	Um elemento de formatação onde colocar componentes que devem ser mostrados de cima para baixo.
	HorizontalScrollArrangement	Parecido com HorizontalArrangement, mas permite scroll.
	VerticalScrollArrangement	Parecido com VerticalArrangement, mas permite scroll.
	TableArrangement	Um elemento de formatação para se colocar componentes que devem ser exibidos em forma de tabela.

Fonte: Elaborada pelo autor

Além dos componentes visuais citados acima, o App Inventor permite que seja feita uma série de configurações na aparência geral do design de interface de usuário do app (MIT, 2023), como, por exemplo, a possibilidade de selecionar cores primárias e secundárias, configurar o alinhamento horizontal e vertical da interface como um todo, a configuração do nome do aplicativo e ícone, a seleção de uma cor de fundo, a configuração de animações de navegação entre telas, a configuração de orientação padrão do aplicativo (retrato ou paisagem), além de permitir a seleção de temas.

A configuração de temas no App Inventor é um dos aspectos que facilitam a construção do design de interface de usuário, pois possibilita torná-lo mais próximo de aplicativos comuns, uma vez que o tema é o conjunto de atributos visuais que definem o estilo geral da aplicação (MIT, 2023). O App Inventor oferece quatro temas pré-definidos como parâmetro da tela principal do aplicativo (*Screen1*):

- *Classic* (clássico),
- *Device Default* (padrão do dispositivo dependendo da versão Android),
- *Black Title Text* (texto preto no título) e
- *Dark* (componentes em modo escuro),

Sendo que por padrão o tema definido é o *Classic*. Uma vez definido um tema, essa alteração irá afetar toda a aparência dos componentes de *User Interface*, como botões, caixas de texto e listas.

Apesar da facilidade e da versatilidade do App Inventor para criar aplicativos móveis, o design de interface de usuário nessa plataforma ainda apresenta algumas limitações, principalmente pela "falta de suporte de forma mais completa tanto para a documentação de artefatos como histórias de usuário e personas, quanto a um suporte ao design visual mais alinhado aos meta-princípios e guias de estilo visual como o Material Design e *Web Content Accessibility Guidelines* 2.0 (WCAG)" (PINHEIRO et al., 2019).

Além disso, atualmente não é possível, com o App Inventor, criar layouts responsivos que se adaptem a diferentes tamanhos e orientações de tela. Também não é possível usar gradientes, sombras ou transparências nos componentes visuais. Ademais, o App Inventor não

possui ferramentas específicas para auxiliar na escolha de cores harmônicas ou na criação de ícones personalizados, nem na seleção de fontes personalizadas.

2.2. Design visual voltado ao design de interface de aplicativos móveis

O design visual é a etapa final do design de interfaces de usuário que define em detalhes a aparência do aplicativo, de maneira a determinar como a informação na tela será apresentada visualmente e a tornar o design da interface algo concreto (GARRETT , 2011). Sendo assim, ao desenvolver o design visual de um aplicativo, é essencial considerar a estética visual, um fator de extrema importância para o sucesso de aplicativos móveis (XU; PEAK; PRYBUTOK, 2015). A estética tem um impacto significativo em atributos como usabilidade percebida, usabilidade efetiva, satisfação do usuário e outros aspectos relacionados à experiência do usuário, uma vez que a estética dos elementos da interface pode influenciar o interesse dos usuários em utilizar o aplicativo e aumentar suas chances de sucesso (XU; PEAK; PRYBUTOK, 2015).

Desta forma, o design visual engloba elementos como *layout*, cores, tipografia, imagens e ícones, os quais devem ser cuidadosamente selecionados e projetados para criar uma experiência agradável e atrativa para o usuário. Além disso o design é guiado por meta-princípios como consistência, hierarquia e personalidade, que desempenham um papel fundamental na estética visual do aplicativo, uma vez que o uso balanceado dos elementos de design aliado aos meta-princípios pode resultar em uma interface bonita e amigável (GARRET, 2011) (SCHLATTER; LEVINSON, 2013).

O design visual pode ser guiado por diretrizes do design de interfaces de usuário, que são regras bem definidas que norteiam o processo de design e que, quando cumpridas corretamente, auxiliam na consistência do design entre diferentes aplicativos da mesma plataforma, satisfazendo as expectativas dos usuários e facilitando o uso (WASSERMAN, 2010).

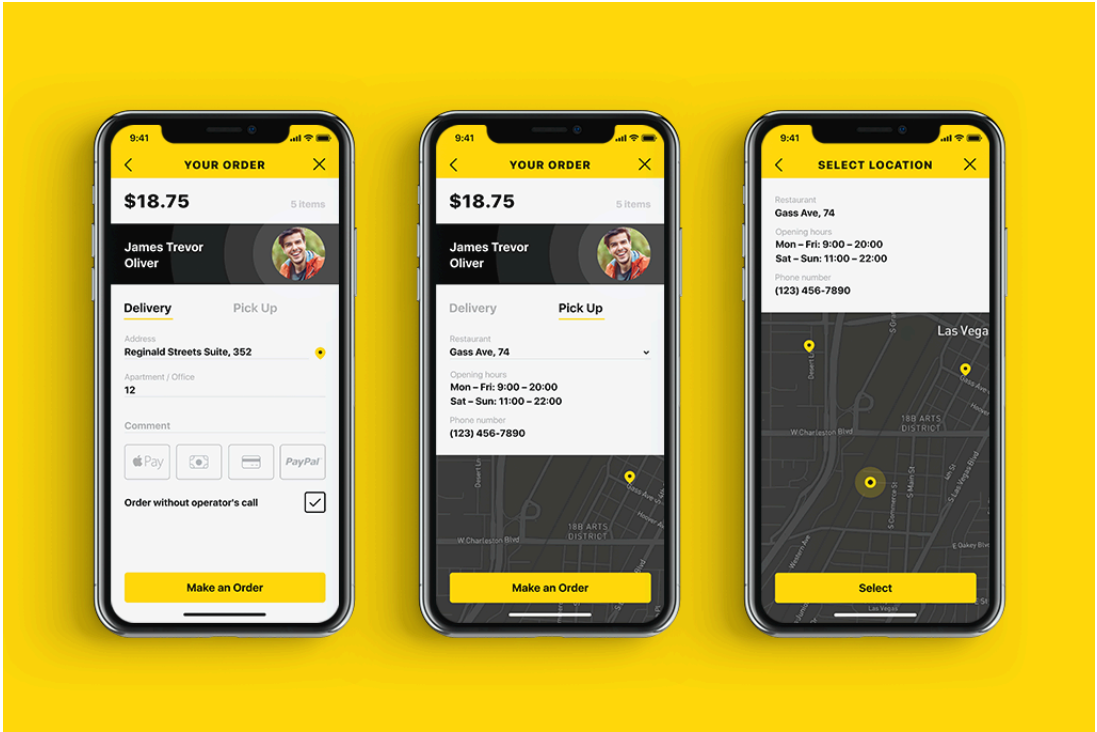
Nas próximas seções, são apresentados mais detalhes sobre os meta-princípios, os elementos do design visual e as diretrizes do design de interfaces.

2.2.1. Meta-princípios do Design Visual

Os meta-princípios do design visual são, dentre os muitos princípios do design, os três considerados fundamentais para o design de interfaces elencados por Schlatter e Levinson (2013), de maneira a auxiliar o designer a projetar interfaces de aplicativos por meio de uma linguagem visual clara e bem definida. Para Schlatter e Levinson (2013), os três meta-princípios fundamentais para o design visual são: consistência, hierarquia e personalidade. Dessa forma, a partir de uma estrutura baseada nesses meta-princípios, seria possível a tomada de decisões para proporcionar a associação de harmonia e utilidade (FERREIRA et al., 2019).

Consistência. A consistência é definida como um estabelecimento de padrões a fim de "definir e manter expectativas do usuário usando elementos com os quais as pessoas estão familiarizadas" (SCHLATTER; LEVINSON, 2013), de modo que essas expectativas se formam tanto do que o usuário visualiza na tela quanto do que ele já viu no passado. A consistência entra em diversos aspectos do design de interfaces, tais como *layouts*, cores, tipografias e recursos imagéticos. A Figura 3 ilustra exemplos de consistência de cores, *layout* e tipografia.

Figura 3 — Consistência no design visual



Fonte: Yalanska (2023)

Schlatter e Levinson (2013) estabelecem tipos que constituem esse meta-princípio:

- Consistência externa: refere-se ao design da aplicação, conteúdo ou comportamento ser similar a outras aplicações usada pelos mesmos usuários;
- Consistência interna: refere-se ao design da aplicação, conteúdo ou comportamento ser similar entre as telas e funcionalidades do próprio aplicativo.

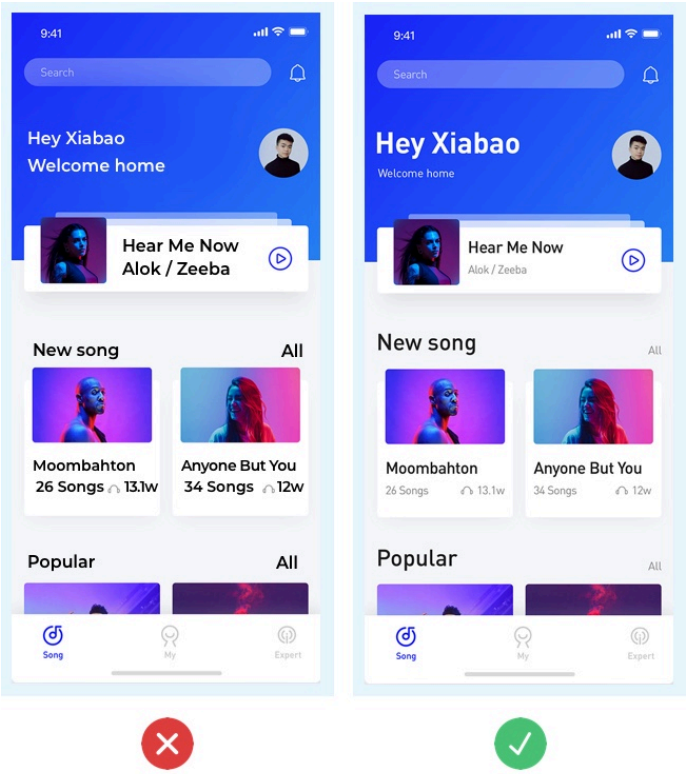
Hierarquia. A hierarquia é um meta-princípio que é definido como crucial no design pois desempenha papel na transmissão da importância de cada elemento que é exibido em tela, de maneira a formar uma percepção e interpretação a respeito dos elementos por meio do seu posicionamento, tamanho, cor, estilo de interface e tipo de controle utilizado, bem como pela maneira como eles se relacionam entre si. Dessa forma, ao construir um aplicativo, é essencial considerar cuidadosamente a hierarquia, definindo estrategicamente a localização dos elementos com base em sua relevância relativa (SCHLATTER; LEVINSON, 2013).

Schlatter e Levinson (2013) definem como as principais características da hierarquia visual: o contraste, o posicionamento e o tratamento:

- Contraste: envolve a criação de diferenças visíveis entre os elementos, ajudando a estabelecer uma distinção entre os elementos de diferentes importâncias, de modo a permitir que o usuário identifique rapidamente o que é mais relevante na interface.
- Posicionamento: refere-se à disposição dos elementos em tela e como isso afeta a percepção de hierarquia, bem como o contraste.
- Tratamento: está relacionado ao estilo visual aplicado aos elementos, o qual afeta a importância percebida por parte do usuário. Elementos como tamanho, cores, detalhes e outros atributos visuais distintos tendem a se destacar mais e serem considerados mais relevantes na percepção do usuário.

A Figura 4 ilustra as diferenças entre um design com uma boa hierarquia *versus* um design com uma hierarquia defeituosa. É possível notar que, na imagem da esquerda, é difícil de compreender a diferença entre os elementos em tela, o que torna a leitura das informações cansativa. Na imagem da direita, a disposição dos elementos torna possível escanear pela tela em busca dos dados mais relevantes que estamos procurando, facilitando a compreensão das informações.

Figura 4 — Diferenças entre as hierarquias de design



Fonte: Niebla (2023)

Sendo assim, a hierarquia visual ajuda os usuários a identificarem de forma rápida e intuitiva os elementos mais relevantes, proporcionando uma experiência de uso mais eficiente e satisfatória (SCHLATTER; LEVINSON, 2013). Na prática, a falta de hierarquia ocorre quando todos os elementos atraem igualmente a atenção do usuário, sem que algo se destaque de forma perceptível.

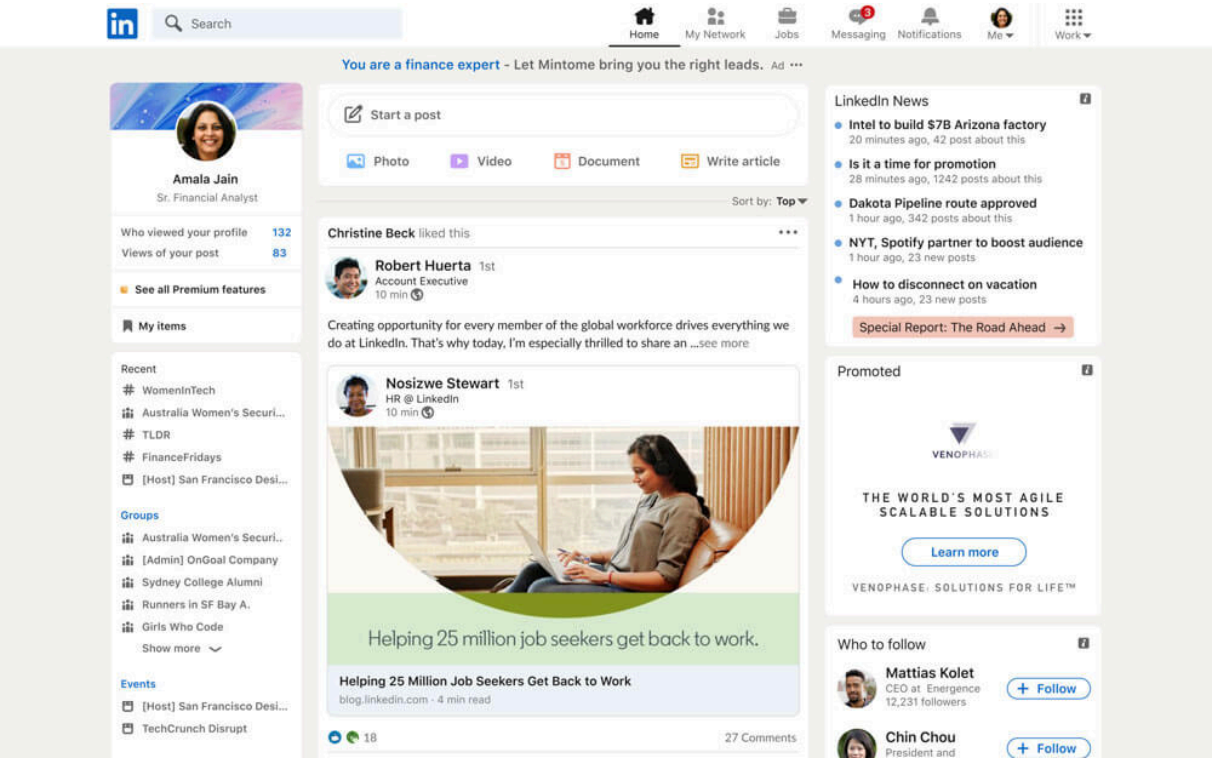
Personalidade. A personalidade é o meta-princípio que está relacionado com a forma como os usuários percebem e interagem com um aplicativo, de maneira a se atentar às impressões que são formadas pelos usuários, de forma consciente ou não, com base em alguns elementos como a aparência, comportamento e satisfação proporcionados pelo aplicativo (SCHLATTER; LEVINSON, 2013). Embora cada interação do usuário com o app possa influenciar sua percepção e avaliação, é nos aspectos visuais que se concentra o estudo da personalidade de um aplicativo.

Dessa forma, a personalidade de um aplicativo é construída por meio de elementos visuais, como cores, tipografia, *layout*, estilo de ícones, entre outros. Esses elementos, quando combinados, podem transmitir sensações aos usuários, tais como seriedade, diversão,

elegância, modernidade, confiança, entre outros atributos. Sendo assim, a aparência visual de um aplicativo pode evocar emoções e criar uma conexão emocional com os usuários, influenciando sua experiência e percepção geral do app.

Portanto, a personalidade de um aplicativo deve ser alinhada com seu propósito e público-alvo (SCHLATTER; LEVINSON, 2013), para que seja possível transmitir a mensagem correta aos usuários. Por exemplo, um aplicativo corporativo deve adotar uma personalidade mais formal, com cores sóbrias e um design mais minimalista, como é o caso, por exemplo, do LinkedIn, um aplicativo de rede social que é voltado mais ao aspecto corporativo.

Figura 5 — Interface do LinkedIn



Fonte: Chartier (2020)

Logo, levando em consideração este meta-princípio, designers podem criar interfaces que transmitem a identidade desejada para o aplicativo, gerando uma experiência envolvente e coerente com a proposta do app.

2.2.2. Elementos do design visual

No design visual, os elementos são cuidadosamente manipulados com o objetivo de transmitir uma mensagem específica aos usuários (SCHLATTER; LEVINSON, 2013). Nesta seção, são apresentados os principais elementos do design visual e como cada um deles afeta a interpretação individual de cada usuário.

Layout. O *layout* é um aspecto essencial do design visual de uma interface de usuário que se refere à organização e disposição dos elementos visuais dentro de uma tela, de forma a determinar como esses elementos são estruturados e interagem uns com os outros (SCHLATTER; LEVINSON, 2013). Entre os termos que envolvem o *layout*, Schlatter e Levinson (2013) definem:

- Tamanho de tela, que no contexto do desenvolvimento de aplicativos está ligado a entender que existem diferentes tipos de dispositivos com diferentes tamanhos de telas;
- Posicionamento, que é estruturação dos componentes em tela de forma a obedecer alguma ordem lógica e coerente;
- Espaços em branco, que são áreas em branco que servem para separar diferentes tipos de conteúdo por meio do uso de margins, por exemplo;
- Proximidade, a qual se refere a manter itens de um mesmo contexto próximos entre si;
- Escala, como a configuração de importância entre os diferentes elementos visuais em tela;
- Alinhamento, que é a organização dos elementos visuais dada uma determinada ordem.

A Figura 6 exemplifica como as alterações nas margens afetam o visual de um aplicativo.

Figura 6 — Uso de margens no design de interfaces

Fonte: Schlatter e Levinson (2013)

Controles. Controles tratam-se de elementos visuais que permitem ao usuário interagir com um aplicativo, como, por exemplo, botões, menus, caixas de seleção, barras de rolagem, entre outros (SCHLATTER; LEVINSON, 2013). Esses controles devem ser projetados de forma a indicar claramente aos usuários sua função e seu estado, além de também fornecer *feedback* adequado em relação a suas ações. Dessa forma, os controles devem ser consistentes, intuitivos e acessíveis aos usuários.

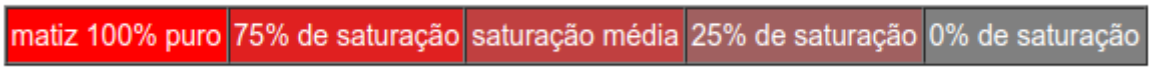
Cores. As cores no contexto de design visual são elementos visuais que se baseiam na percepção de luz para transmitir informações, emoções e significados aos usuários, de maneira a afetar o humor, a atenção e a motivação dos mesmos (SCHLATTER; LEVINSON, 2013). Elas são descritas tipicamente por meio de três atributos principais:

- A matiz é a qualidade que distingue uma cor de outra, como vermelho, verde ou azul.
- A saturação é a intensidade ou pureza de uma cor, que varia do cinza (sem saturação) ao colorido (alta saturação).
- A luminosidade é o brilho ou a claridade de uma cor, que varia do preto (baixa luminosidade) ao branco (alta luminosidade).

Figura 7 — Matiz - Matizes numericamente ordenados num espaço de cor HSL

Fonte: Wikipédia (2023)

Figura 8 — Saturação - Gradação de saturação no modelo HSL



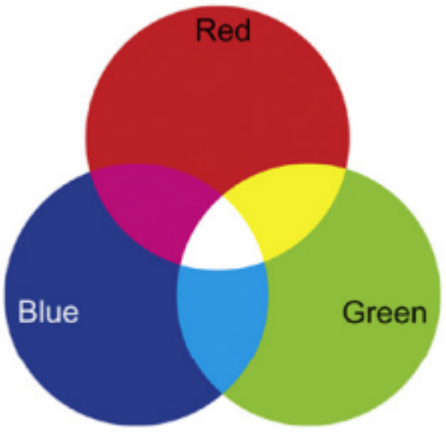
Fonte: Wikipédia (2023)

Figura 9 — Luminosidade - escala de claridade para o cinza neutro

Fonte: Wikipédia (2023)

Diferentemente da representação/classificação das cores em outros meios, em sistemas eletrônicos, as cores são representadas por meio de uma mistura aditiva de cores, utilizando a representação das luzes vermelho (R), verde (G) e azul (B), que, quando projetadas simultaneamente, formam o branco (GONÇALVES, 2014). Essa mistura das três cores é conhecida como padrão RGB e possibilita a representação de diferentes cores em meios eletrônicos.

Figura 10 — Representação de cores RGB



Fonte: Schlatter e Levinson (2013)

Além disso, no design visual, o uso de códigos de cores como o código hexadecimal é importante para assegurar a consistência na reprodução das cores em diferentes dispositivos e plataformas. No sistema de representação hexadecimal, as cores são representadas por meio

da combinação de números e letras, onde cada combinação representa um valor específico para os componentes vermelho (R), verde (G) e azul (B) da cor, permitindo que sejam representadas mais de 16 milhões de cores, misturando as possibilidades. Na Figura 11 há exemplos de algumas cores juntamente com sua representação em hexadecimal.

Figura 11 — Representação de cores usando hexadecimal

Fonte: Elaborado pelo autor

Outros sistemas de representação de cores também podem ser adotados para facilitar o design visual, como o LCH e o CIELAB (KONICA MINOLTA, 2023). O sistema CIELAB, por exemplo, leva em consideração não apenas a combinação dos componentes RGB, mas também a luminância, as coordenadas de cor nos eixos a* e b*, bem como as diferenças perceptuais entre as cores. Isso é essencial para criar paletas de cores que sejam consistentes e agradáveis ao olho humano. O espaço de cores LCH, por sua vez, é uma extensão do CIELAB que enfatiza matiz (*Hue*), saturação (*Chroma*) e luminância (*Luminance*). Ele proporciona uma representação mais intuitiva das cores, facilitando a seleção de paletas harmoniosas e garantindo que as diferenças de cor sejam percebidas de maneira uniforme.

Além disso, sistemas como OKLCH, OKLAB e HSLuv também são adotados. O OKLCH, por exemplo, otimiza as distâncias perceptuais entre cores, tornando-se útil para aplicações sensíveis à percepção humana, enquanto o OKLAB oferece uma alternativa ao CIELAB com melhorias perceptuais adicionais. Por outro lado, o HSLuv combina matiz, saturação e luminância de maneira uniforme para garantir uma representação visualmente coerente das cores.

As cores por si só desempenham um papel importante no design visual e podem ter diferentes funções e efeitos (SCHLATTER; LEVINSON, 2013). É possível transmitir certas informações por meio das cores, como, por exemplo, o estado ou a categoria de um elemento

visual, como um botão, link ou gráfico. Por exemplo, as cores vermelho, amarelo e verde podem ser utilizadas para indicar diferentes níveis de urgência, prioridade ou disponibilidade.

- As cores têm o poder de produzir emoções nos usuários, como alegria, tristeza, raiva ou calma. Por exemplo, cores quentes, como vermelho, laranja e amarelo, estão associadas a energia, paixão ou alerta. Por outro lado, cores frias, como azul, verde e roxo, podem transmitir sensações relacionadas à tranquilidade, confiança ou mistério.
- As cores podem ter significados culturais, simbólicos ou pessoais para os usuários. Por exemplo, as cores branco e preto podem representar pureza e luto em algumas culturas, mas o inverso em outras. As cores também podem ter associações com marcas, conceitos ou valores. Por exemplo, a cor verde pode remeter à natureza, à saúde ou ao dinheiro.
- As cores podem ser utilizadas para estabelecer a hierarquia visual de um aplicativo, podendo ser usadas para destacar os elementos mais importantes ou as ações desejadas em uma interface. As cores também devem ser usadas para criar agrupamentos lógicos e visuais entre os elementos relacionados.

Dessa forma, saber utilizar as cores de forma correta e coerente ajuda a atrair a atenção do usuário, fragmentar áreas da interface e auxiliar no processo de memorização, sendo que todos esses aspectos fazem parte de uma interface considerada boa (GONÇALVES, 2004). Assim, o processo de design visual no desenvolvimento de aplicativos deve considerar a escolha das cores de maneira a torná-las rapidamente identificáveis e categorizáveis pelo usuário.

2.2.2.1. A importância do contraste

De acordo com Schlatter e Levinson (2013), contraste é importante para criação de hierarquia dentro do design visual e pode ajudar a melhorar a usabilidade, legibilidade, a ênfase e o interesse dos elementos visuais dispostos em tela, estando diretamente relacionado à diferença de luminosidade ou cor entre dois elementos, como cor de texto e cor de fundo, tornando-se fundamental para, por exemplo, garantir a legibilidade e a facilidade de percepção dos diferentes elementos.

Devido a importância do contraste entre a cor de texto e a cor de fundo, especialmente para pessoas com dificuldades visuais ou deficiências de cores, existem uma série de

diretrizes de acessibilidade e padrões estabelecidos, como por exemplo o *Web Content Accessibility Guidelines* (WCAG) (2023), que ajudam os designers a projetarem suas interfaces ao estabelecerem uma série de critérios específicos voltados ao contraste da interface visando garantir a acessibilidade do conteúdo on-line.

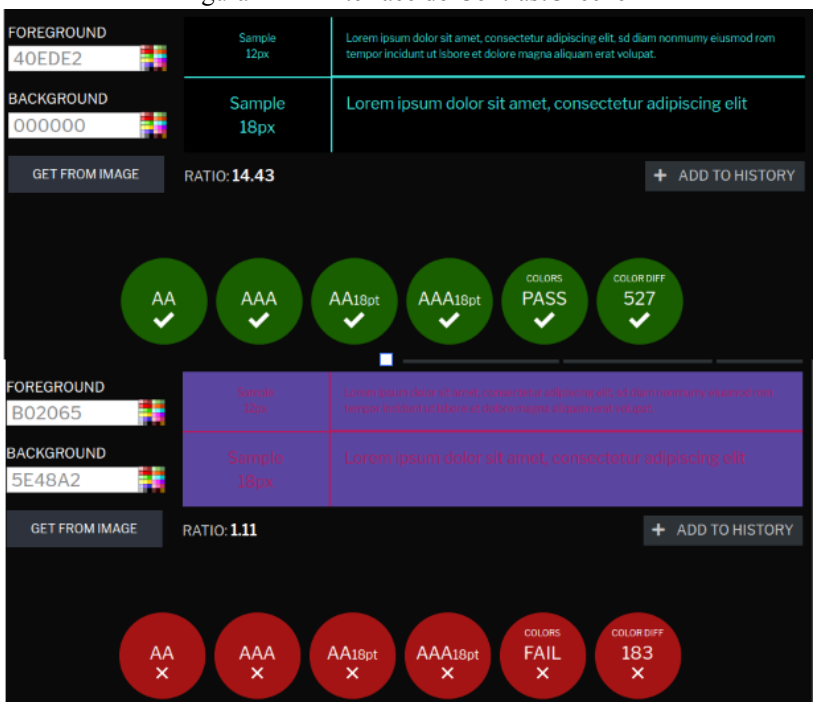
Essas diretrizes e padrões de acessibilidade também são abordados pelo Material Design 3[1], fornecendo orientações específicas em relação aos níveis mínimos de contraste a serem adotados ao projetar interfaces. Por exemplo, o Material Design 3 segue os critérios estabelecidos pela WCAG 2.1 (2023), abordando contraste mínimo para diferentes níveis de conformidade, como o nível AA e o nível AAA. Para garantir uma legibilidade adequada, especialmente para pessoas com deficiências visuais ou dificuldades de leitura, o Material Design 3 (GOOGLE, 2023b) estabelece que o texto de tamanho normal (corpo do texto) deve ter um contraste mínimo de 4,5:1 em relação ao seu fundo, para atender ao nível AA. Já para o texto de tamanho grande (cabeçalhos, títulos etc.), o contraste mínimo exigido é de 3:1 em relação ao fundo, também para atender ao nível AA. Esses critérios consideram fatores como tamanho do texto e tipo de conteúdo, assegurando que os valores de contraste recomendados sejam adequados para promover a legibilidade e a acessibilidade das interfaces desenvolvidas com base no Material Design 3.

É fundamental seguir esses critérios para garantir que a interface seja inclusiva e acessível a todos os usuários.

Para facilitar a escolha de boas cores que sigam os padrões destacados nas diretrizes, comumente são utilizadas ferramentas de avaliação de contraste, como, por exemplo, o site *contrastchecker.com*, que auxilia na seleção de cores contrastantes e as classifica com base nos critérios adotados pela WCAG 2.1. Na Figura 12 estão destacados dois exemplos de utilização da ferramenta, uma usando cores com um bom nível de contraste e outra usando cores com um baixo nível de contraste.

[1] https://m3.material.io/

Figura 12 — Interface do ContrastChecker



Fonte: Elaborado pelo autor

Dessa forma, ao projetar interfaces de usuário, deve-se buscar seguir as diretrizes de acessibilidade para garantia de um contraste adequado. Isso pode envolver o uso de combinações de cores de alto contraste, como texto preto em fundo branco ou vice-versa, ou a seleção de cores complementares que proporcionem uma diferença significativa em termos de luminosidade.

2.2.2.2. Harmonia entre as cores

A harmonia de cores desempenha um papel importante no design de interfaces de usuário, contribuindo para a estética, legibilidade e a comunicação eficaz (ADRIYANTO et al., 2017) e está muito relacionada com o uso de cores contrastantes. Para definir uma boa harmonia, é possível utilizar-se dos princípios da Teoria das Cores, usando o círculo cromático (SILVEIRA, 2015), que representa as cores organizadas em um círculo, permitindo visualizar as relações entre elas. O círculo cromático é útil para identificar combinações de cores complementares, análogas, triádicas e outras, auxiliando na escolha de cores harmônicas entre si que possuem um bom contraste. Na Figura 13 é representado o círculo cromático com diferentes técnicas de combinação de cores harmônicas.

Figura 13 — Círculos cromáticos

Fonte: Tecidos (2023)

Sendo assim, por meio do uso do contraste, é possível criar harmonia entre as cores, uma vez que o contraste não se limita apenas à diferença de brilho entre o texto e o fundo, mas também pode ser aplicado na seleção de cores complementares ou contrastantes em toda a interface. Há diversos tipos de contraste que podem ser feitos em relação à harmonia de cores, como, por exemplo, entre cores quentes e frias, usando a diferença que as cores transmitem em relação à sensação de calor ou frio (SCHLATTER; LEVINSON, 2013), conforme ilustrado na Figura 14.

Figura 14 — Harmonia de cores

Fonte: Schlatter e Levinson (2013)

Um outro tipo de harmonia por contraste é a utilização de cores complementares, que usa pares de cores que se opõem entre si no círculo cromático para criar vibração e luminosidade. Nessa disposição, Schlatter e Levinson (2013) definem que não significa que as cores nesse formato de contraste formarão um par atraente, mas sim que, quando colocadas uma ao lado da outra, cada uma fará com que a outra pareça mais vibrante, em comparação com quando vistas sozinhas ou combinadas com outras cores.

Outra técnica é o uso de cores análogas, que são cores que estão próximas umas às outras no círculo cromático, criando uma combinação de cores com uma aparência harmoniosa e suave, pois as cores compartilham características visuais semelhantes (SILVEIRA, 2015).

Cores monocromáticas envolvem o uso de variações de uma única cor, criando assim uma paleta de cores harmoniosa e equilibrada por meio do uso de diferentes tons, saturação e luminosidades (SILVEIRA, 2015). Essa alternativa é útil para criar interfaces elegantes e minimalistas, em que a ênfase é dada à sutileza e à variação tonal.

Existem ainda outras formas de combinações envolvendo mais cores em uma paleta, como, por exemplo, a técnica de cores semi-complementares, cores triádicas, cores tetraédricas, entre outras conforme ilustrado na Figura 13.

Cada uma dessas técnicas de combinação de cores oferece possibilidades diferentes para a criação de paletas harmoniosas e atraentes. Ao utilizar essas combinações de forma adequada, é possível alcançar resultados visuais impactantes e coerentes, garantindo a estética e a efetividade do design de interface.

2.2.2.3. Tipografia

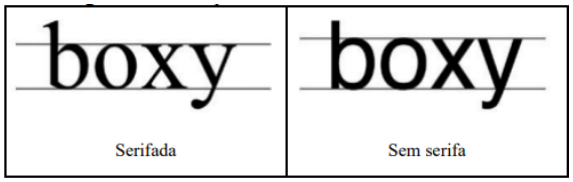
Tipografia é a arte e a técnica de criar e usar tipos para compor textos visuais, envolvendo o desenho, a seleção, a combinação e a disposição dos tipos em uma superfície ou em um meio digital (MOTA; AMENDOLA, 2016). Uma família tipográfica é composta por diversas fontes, que são variações de um mesmo formato básico, como itálico, negrito e diferentes espessuras de traço (Figura 15).

Figura 15 — Variações tipográficas

Fonte: Google (2023b)

As fontes tipográficas podem ser classificadas como sendo do tipo serif ou sans-serif. Fontes do tipo serif apresentam pequenos traços decorativos nas extremidades dos caracteres, enquanto que fontes do tipo sans-serif não possuem traços decorativos e são compostas de linhas retas (Figura 16).

Figura 16 — Exemplo de fonte serifada e sem serifa



Fonte: Schlatter e Levinson (2013)

Visando facilitar a seleção de fontes, em 2010 o Google lançou o Google Fonts[2], um repositório de fontes sem direitos autorais, com mais de 1.531 famílias tipográficas disponíveis para uso gratuito em projetos. Essa biblioteca abrangente oferece uma ampla variedade de fontes com diferentes estilos, o que permite aos designers ter uma maior flexibilidade e opções para expressar a identidade visual desejada em suas criações.

No contexto do sistema Android, a fonte padrão utilizada é a Roboto, também disponível no Google Fonts gratuitamente, sendo também a família tipográfica recomendada pelo time de desenvolvimento do Android para utilização em interfaces voltadas para o Sistema Operacional.

Dessa forma, a tipografia, juntamente com as cores, é um elemento visual fundamental para o design, permitindo transmitir informações, emoções e significados aos leitores e usuários de uma interface (MOTA; AMENDOLA, 2016). Assim, a seleção adequada de uma

[2] https://fonts.google.com/

família tipográfica desempenha um papel crucial na criação de uma estética visual atraente e na comunicação efetiva de mensagens por meio do design tipográfico.

Uma forma para selecionar tipografias é baseando-se no tema da aplicação, como, por exemplo, temas medievais ou infantis. Nesse contexto, o site FontLibrary[3] apoia a seleção de tipografias baseada nos temas utilizando uma classificação das *Google Fonts* em diferentes categorias relacionadas ao estilo que a fonte representa, associando tags a cada uma dessas fontes, como, por exemplo, fontes dos anos 50, fontes cartunescas ou fontes que apresentam gradientes.

2.2.2.4. Imagens

As imagens são elementos visuais que representam objetos, pessoas, lugares, conceitos ou ideias, podendo elas serem fotográficas, animadas, por meio de uso de logos, ícones ou símbolos, para visualização de dados etc. (SCHLATTER; LEVINSON, 2013). Dessa forma, as imagens conseguem complementar, reforçar ou substituir os textos visuais em uma interface, tornando-se, portanto, uma ferramenta importante no design visual de aplicativos. Entre os tipos de imagens definidos por Schlatter e Levinson (2013) que podem ser utilizados para compor design de interfaces, estão:

- Imagens Fotográficas: utilizadas para ilustrar conceitos, pessoas, lugares ou produtos específicos dentro de uma interface, fornecendo um meio de ajudar os usuários a entender e se conectar com o conteúdo apresentado na interface.
- Imagens Animadas: adicionam interatividade e dinamismo à interface. São usadas para fornecer feedback visual durante a interação do usuário, transmitir microinterações ou simplesmente adicionar interesse visual.
- Ícones: Os ícones são representações visuais compactas que são amplamente reconhecidas e compreendidas pelos usuários e desempenham um papel importante na comunicação rápida e eficaz de ações, funções e elementos de navegação em uma interface, sendo fundamental a escolha de ícones que sejam visualmente claros, únicos e apropriados para o contexto do aplicativo.
- Visualização de Dados: As imagens também podem ser usadas para visualizar dados complexos de uma maneira mais compreensível e envolvente. Gráficos, infográficos e

[3] https://fontlibrary.dev/

outros elementos visuais podem ser usados para apresentar informações e estatísticas de forma clara e concisa.

A consistência no uso das imagens, principalmente no caso dos ícones, é importante para criar uma interface coesa e intuitiva (SCHLATTER; LEVINSON, 2013). Os ícones devem seguir um estilo visual consistente em termos de tamanho, proporções, estilo gráfico e cor. Isso permite que os usuários identifiquem rapidamente os ícones e os associem aos seus significados e funções correspondentes. A falta de consistência no uso do ícone pode causar confusão e dificuldade na interação do usuário.

Além disso, a consistência de cores também é importante, tanto para ícones quanto para imagens em geral. Ao manter uma paleta de cores consistente, os elementos visuais se complementam de maneira harmônica, criando uma experiência de usuário mais agradável e intuitiva. A consistência de cores nos ícones ajuda os usuários a identificarem mais facilmente as funcionalidades representadas, enquanto a consistência nas imagens em geral contribui para uma estética visual unificada e uma comunicação visual efetiva.

2.2.3. Diretrizes do design de interface

As diretrizes do design de interface são um conjunto de regras que permitem os envolvidos no processo de design terem uma documentação clara e bem estruturada, de modo a previnir e corrigir erros de usabilidade que já são bem conhecidos, facilitando o trabalho em equipe e uniformizando as interfaces desenvolvidas (SHNEIDERMAN; PLAISANT, 2009)(PREECE, 1994).

Em aplicativos móveis para Android, tipicamente é utilizado o Material Design (GOOGLE, 2022), um guia de estilo de design de interface voltado para o desenvolvimento de interfaces para a plataforma Android, desenvolvida pela Google em 2014, que atualmente já está em sua terceira versão. O Material Design oferece uma série de orientações detalhadas sobre diversos aspectos do design de interface, como, por exemplo, estilo de cores, ícones e tipografia, além de definir diversos componentes específicos que podem ser utilizados na interface.

Em relação ao estilo de cores, o Material Design 3 apresenta uma paleta de cores ampla e vibrante, com ênfase em tons saturados e contrastes bem definidos. As cores são formadas a partir de um grupo de cinco cores-chave, dividas em dois grupos, chamados de *accent colors* e *neutral colors*, que juntos formam a paleta de cores de um aplicativo ao variar

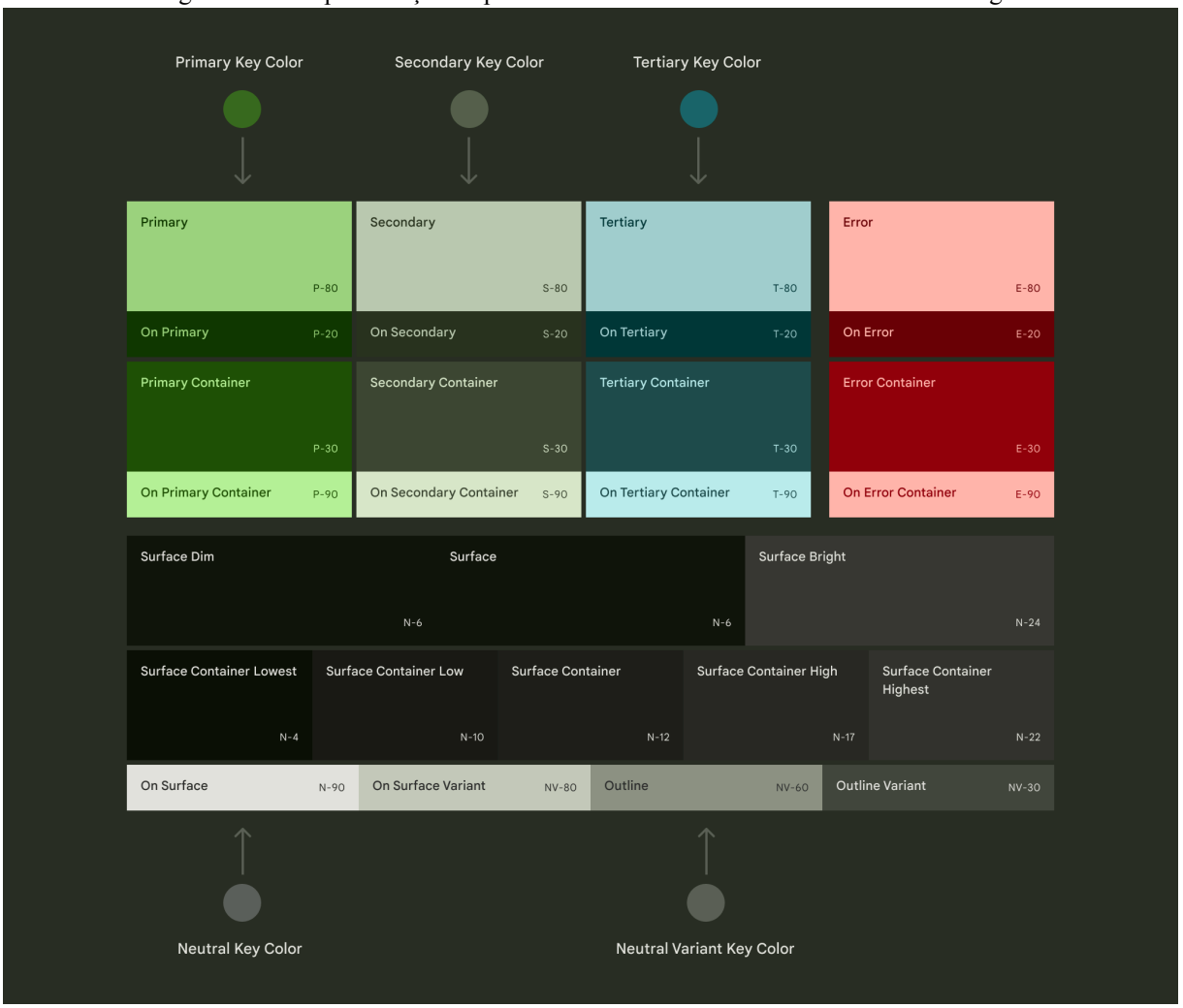
e misturar as cinco cores-chave. A escolha dessas cores deve ser feita cuidadosamente para transmitir significados e intenções específicas, contribuindo para a estética visual e a comunicação efetiva da interface. Na Figura 17 há a representação de uma paleta de cores para o conjunto das cinco cores-chave definidas.

Figura 17 — Representação de paleta de cores do Material Design

Fonte: Material Design (2023b)

Além da paleta de cores principal, o Material Design 3 também define a criação de uma paleta de cores alternativa, utilizada no modo escuro de um aplicativo, visando proporcionar uma experiência visual agradável e confortável aos usuários. No modo escuro, as cores são projetadas para reduzir a fadiga visual, e para isso elas são selecionadas para garantir um bom nível de contraste e legibilidade. Essa paleta alternativa contribui para uma interface adaptada às preferências dos usuários, oferecendo a opção de uma experiência visual mais suave e menos intrusiva, especialmente em ambientes com pouca iluminação. A Figura 18 apresenta a paleta de cores no modo escuro, seguindo os mesmos princípios definidos na paleta de cores anterior.

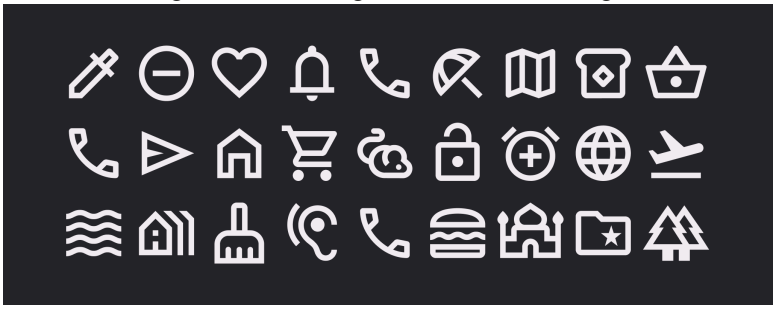
Figura 18 — Representação de paleta de cores no modo escuro do Material Design



Fonte: Material Design (2023b)

O Material Design também traz uma orientação quanto à utilização de ícones em interfaces de usuário, estabelecendo um conjunto de normas e padronizações, por meio de uma seleção de ícones (Figura 19) previamente definidos que são consistentes, reconhecíveis e significativos para representar diversas funcionalidades e ações. Esses ícones seguem um estilo visual específico, com formas nítidas e cantos arredondados, proporcionando uma aparência moderna e harmoniosa.

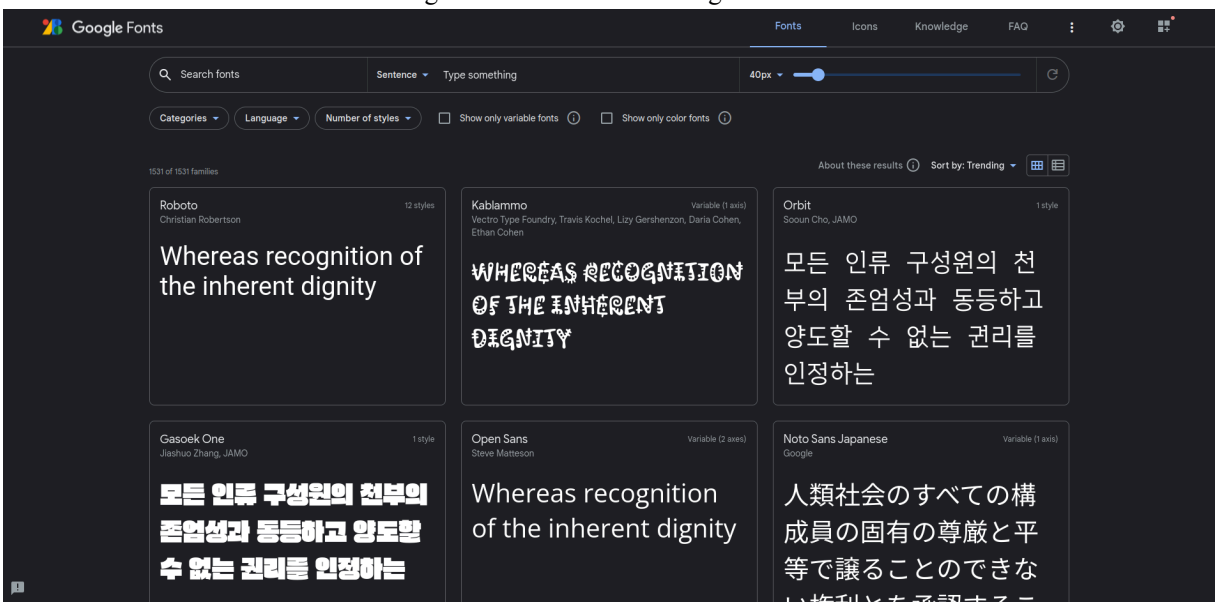
Figura 19 — Iconografia do Material Design



Fonte: Material Design (2023b)

A tipografia também é um aspecto fundamental do Material Design. O uso de famílias tipográficas adequadas, como a Roboto, que é a fonte-padrão do Android, contribui para a legibilidade e a clareza do texto na interface. O Material Design define diretrizes sobre o tamanho, espaçamento e estilo dos elementos de texto, para assegurar uma experiência de leitura agradável e coerente. Para encontrar fontes que seguem as normas definidas no Material Design, é possível utilizar o site Google Fonts, que traz mais de 1.000 fontes que podem ser utilizadas gratuitamente (Figura 20).

Figura 20 - Interface do Google Fonts



Fonte: Elaborado pelo autor

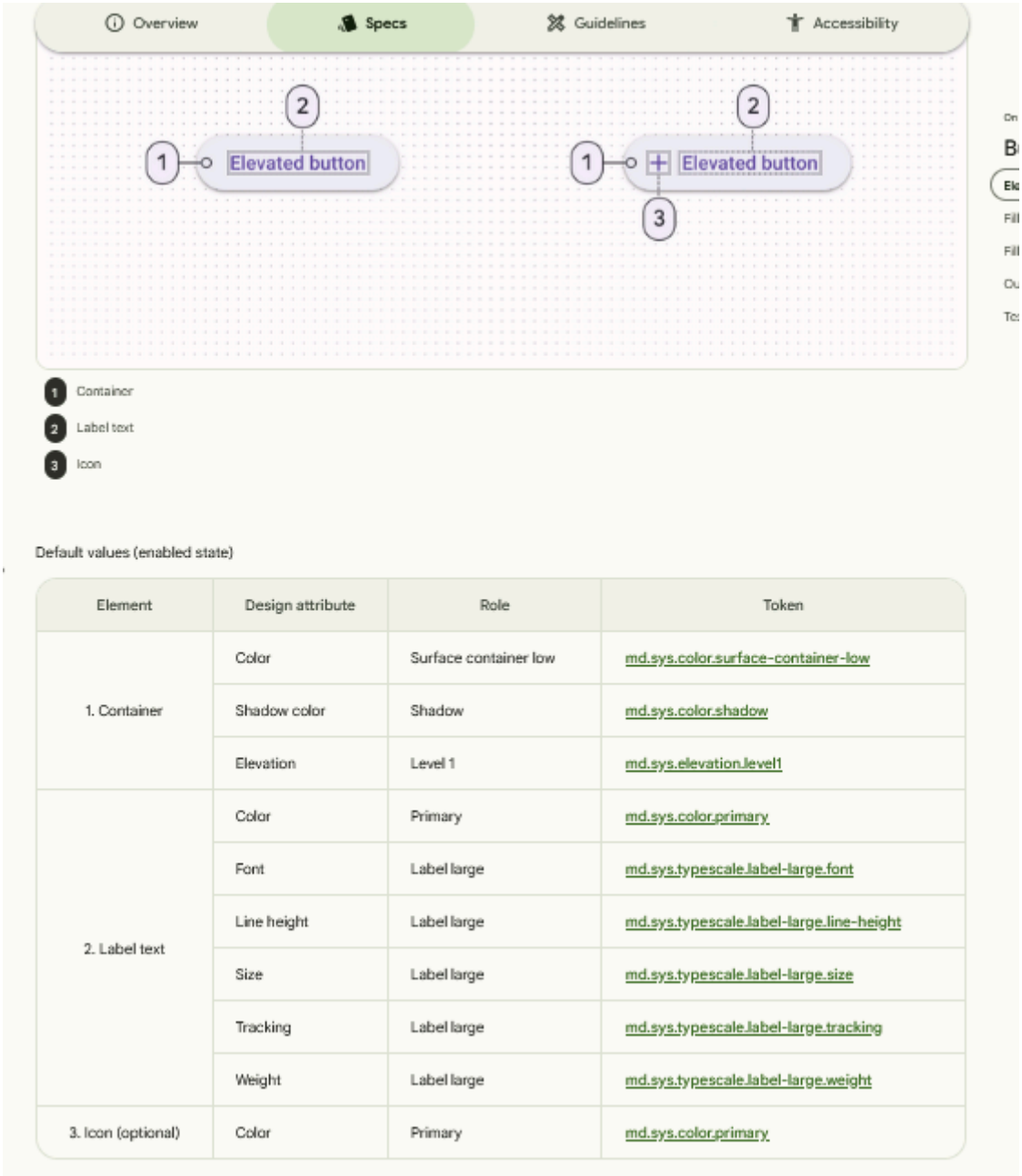
Além disso, o Material Design 3 fornece diretrizes sobre o design de uma variedade de componentes que podem ser utilizados no design de interfaces, proporcionando uma base sólida para o desenvolvimento consistente e intuitivo de aplicativos, por meio de uso de componentes já validados e amplamente reconhecidos pelos usuários. Entre os principais componentes definidos pelo Material Design 3, podemos destacar:

- *Badges*: pequenos elementos visuais utilizados para indicar status, notificações ou informações adicionais relacionadas a um item específico.
- *Buttons*: elementos interativos que representam ações e permitem que os usuários realizem ações específicas, como enviar formulários ou navegar para outras telas.
- *Cards*: elementos de design que agrupam informações relacionadas, como texto, imagens e botões, em um único bloco visualmente atraente.
- *Checkbox*: elementos de seleção que permitem aos usuários escolher uma ou mais opções de um conjunto disponível.
- *Chips*: componentes compactos que exibem informações ou atributos, como tags ou filtros, de forma visualmente destacada e interativa.
- *Dialogs*: janelas modais que exibem informações adicionais, solicitações de confirmação ou interações específicas, geralmente sobrepostas à tela principal.
- *Radio buttons*: elementos de seleção que permitem aos usuários escolher uma única opção de um conjunto disponível.
- *Switches*: elementos interativos que permitem aos usuários alternar entre estados ligado/desligado ou entre duas opções.

Esses são apenas alguns exemplos dos componentes definidos no Material Design 3.

Como parte das diretrizes de cada um dos componentes definidos no Material Design, há também orientações sobre a aparência desses elementos visuais e como eles devem responder às interações do usuário (Figura 21). Por exemplo, a aparência de um botão pode variar de acordo com a ênfase desejada para a ação, conforme ilustrado na Figura 22. É possível utilizar um botão preenchido para representar a ação principal de uma tela, ao lado de um botão *outlined* para uma ação secundária. A escolha do tipo de botão mais adequado depende do contexto e da importância atribuída a cada ação.

Figura 21 — Especificações de botões no Material Design

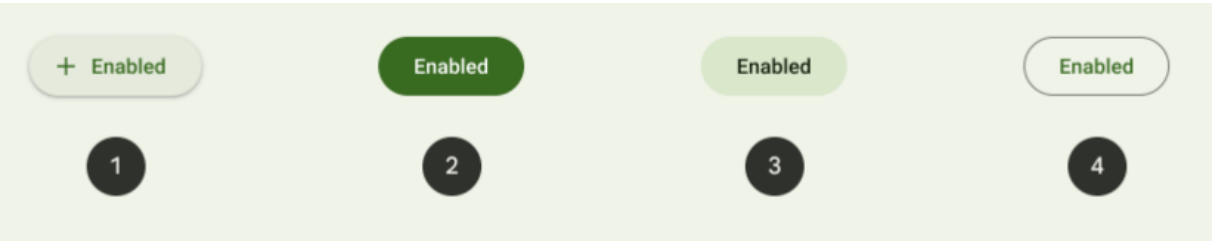


Default values (enabled state)

Element	Design attribute	Role	Token
1. Container	Color	Surface container low	md.sys.color.surface-container-low
	Shadow color	Shadow	md.sys.color.shadow
	Elevation	Level 1	md.sys.elevation.level1
2. Label text	Color	Primary	md.sys.color.primary
	Font	Label large	md.sys.typescale.label-large.font
	Line height	Label large	md.sys.typescale.label-large.line-height
	Size	Label large	md.sys.typescale.label-large.size
	Tracking	Label large	md.sys.typescale.label-large.tracking
	Weight	Label large	md.sys.typescale.label-large.weight
3. Icon (optional)	Color	Primary	md.sys.color.primary

Fonte: Material Design (2023b)

Figura 22 — Variações dos botões do Material Design



Fonte: Material Design (2023b)

É importante ressaltar que nem todos os componentes do Material Design 3 estão disponíveis nativamente no App Inventor. Embora o App Inventor ofereça uma variedade de componentes para criar interfaces de aplicativos, alguns elementos específicos do Material Design podem exigir a instalação de *plugins* adicionais ou até mesmo a necessidade de adaptar a interface de acordo com as opções disponibilizadas pela plataforma. Dessa forma, ao projetar interfaces voltadas ao App Inventor, é importante considerar essas limitações, assegurando a coerência visual e comportamental com o Material Design 3, mesmo que seja necessário utilizar alternativas ou fazer ajustes personalizados para alcançar o resultado desejado.

3. ESTADO DA ARTE

Para levantar o estado da arte sobre quais ferramentas (*plugins*) existem para os principais aplicativos de design gráfico como Figma ou Penpot, como uma alternativa gratuita de código aberto, para suportar o processo de design visual de apps (a serem criados com App Inventor), foi conduzido um mapeamento sistemático seguindo os procedimentos propostos por Petersen et al. (2008).

3.1. Definição do protocolo de revisão

O objetivo desta revisão é responder à seguinte questão: Quais *plugins* existem para o Figma ou Penpot que podem auxiliar no desenvolvimento de design de interfaces de usuários de aplicativos voltados ao App Inventor? Com base no objetivo desta revisão, a pergunta de pesquisa é refinada nas seguintes questões de análise:

QA1. Quais *plugins* existem e para qual plataforma?

QA2. Para quais aspectos do design visual eles fornecem suporte e quais suas funcionalidades?

Critérios de inclusão/exclusão. Foram considerados somente *plugins* para suporte do design visual de aplicativos de design gráfico. Mesmo com o foco principal do presente trabalho no aplicativo Penpot, como uma alternativa gratuita e de código aberto, observando-se ainda a falta de *plugins* para este fim nesta ferramenta, abriu-se o escopo, considerando um dos principais aplicativos de design gráfico, o Figma. O estudo limita-se a *plugins* voltados ao design de aplicativos móveis e/ou de design gráfico geral aplicável a aplicativos móveis. Pela ausência de pesquisas com foco em aplicativos desenvolvidos com App Inventor, foi aberto o escopo da pesquisa para aplicativos móveis em geral.

Critérios de qualidade. Foram considerados apenas *plugins* com informações suficientes ou disponíveis publicamente de forma gratuita.

Fontes dos dados. Como raramente são publicados artigos científicos sobre *plugins* deste tipo, foram examinados todos os materiais publicados em inglês, a partir de buscas via Google, por indexar um grande conjunto de dados de diferentes fontes (HADDAWAY et al.

2015). Dado o foco de pesquisa dos aplicativos Penpot e Figma, os *plugins* são buscados também nos grupos de comunidades destas ferramentas e nas plataformas de buscas disponibilizadas.

Definição da *string* de busca. A *string* de busca foi composta de conceitos relacionados à questão de pesquisa, incluindo sinônimos.

Tabela 2 — Definição da string de busca

Termo chave	**Sinônimos/termos relacionados**	**Termos em inglês**
Plugin	Extensão, complemento, add-on, webhook	Plugin, extension, add-on
“Design visual”	“Design de interface”, contraste, cor, tipografia,	“Visual design”, contrast, cor, typography
“Interface de usuário”	UI, design de UI, interação do usuário	“User interface”, “UI design”, “user interaction”
Penpot	Figma	–

Fonte: Elaborada pelo autor

A partir destes termos foi definido um search *string* de busca genérico:

(Plugin OR Extensão OR Complemento OR Add-On OR Extension) AND ("Design visual" OR "Design de interface" OR contraste OR cor OR tipografia OR "visual design" OR "contrast" OR "color" OR "typography") AND (“Interface de usuário” OR UI OR “design de UI” OR “interação do usuário” OR “User interface” OR “UI design” OR “User interaction”) AND (Penpot OR Figma).

A *string* de busca genérica foi adaptada para cada fonte de dados apresentada na Tabela 1. Levando em consideração também o funcionamento das buscas nas fontes, foram realizadas várias buscas para abordar de forma completa o escopo da revisão.

Tabela 3 — *String* de busca para cada fonte.

Fonte	**String de busca**
Google 1 - Penpot	(plugin OR webhook) AND (“visual design” OR contrast OR color OR typography) AND penpot
Google 2 - Penpot	(plugin OR webhook) AND (“user interface” OR “ui design” OR ui OR “user interaction”) AND penpot
Google 3 - Figma	(plugin OR extension) AND (“visual design” OR contrast OR color OR typography) AND figma

Google 4 - Figma	(plugin OR extension) AND (“user interface” OR “ui design” OR ui OR “user interaction”) AND figma
Figma Community 1 (https://www.figma.com/community/plugins)	contrast (restrito a plugins gratuitos)
Figma Community 2 (https://www.figma.com/community/plugins)	color (restrito a plugins gratuitos)
Figma Community 3 (https://www.figma.com/community/plugins)	typography (restrito a plugins gratuitos)

Fonte: Elaborada pelo autor

3.2. Execução da busca

A busca foi realizada entre julho e agosto de 2023, pelo autor, e revisada pela orientadora (Tabela 4). Foram considerados, como resultados relevantes, apenas os plugins gratuitos, não duplicados e que estivessem dentro dos critérios de busca apontados na Tabela 3.

Tabela 4 — Número de artigos identificados por repositório e por fase de seleção.

Fonte	No. de resultados da busca	No. de resultados analisados	No. de resultados potencialmente relevantes	No. de resultados relevantes
Google 1 - Penpot	7.060	100	0	0
Google 2 - Penpot	6.940	100	0	0
Google 3 - Figma	5.120.000	100	13	13
Google 4 - Figma	3.640.000	100	2	0
Figma Community 1	43	43	15	11
Figma Community 2	100	100	9	5
Figma Community 3	100	100	6	5
Total				**34**

Fonte: Elaborada pelo autor

Na primeira fase de análise, títulos, descrições e comentários foram analisados, resultando em 45 artefatos potencialmente relevantes. No segundo estágio, os materiais foram analisados um a um para conferência de suas funcionalidades e casos de uso, visando assegurar sua relevância com respeito aos critérios de inclusão/exclusão. Também foram excluídos *plugins* duplicados entre as buscas. Como resultado, 34 *plugins* foram considerados relevantes (Tabela 3).

3.3. Resultados da revisão

De acordo com as perguntas de análises, as informações relevantes foram extraídas dos materiais encontrados.

3.3.1. Quais plugins existem?

Observa-se que aparentemente existe a motivação e necessidade para este tipo de *plugin*, já que foram encontradas diversas soluções com funcionalidades voltadas ao suporte de design visual de interfaces de usuário. Porém observa-se que todos os *plugins* encontrados são para o aplicativo Figma. Não foi encontrado nenhum *plugin* especificamente para o Penpot, uma vez que a funcionalidade de integração de *plugins* desta ferramenta ainda está sendo desenvolvida e não foi lançada publicamente. Apesar do Penpot já disponibilizar uma forma de integração por meio de *webhooks*, também não foram encontradas soluções usando essa opção.

Tabela 5 — *Plugins* encontrados.

Nome do plugin/ extensão	**Breve descrição**	**Plataforma**	**Link**
CORES			
UI Color Palette	Cria paletas de cores acessíveis e consistentes para interfaces de usuário. Utiliza espaços de cores alternativas como LCH, OKLCH, CIELAB, OKLAB e HSLuv, garantindo conformidade com padrões WCAG e contraste adequado. Possibilita personalização das paletas em tempo real, verificação de contraste, edição de tonalidades e integração de equipe. Além disso, é possível sincronizar estilos locais, gerar ativos de cores e criar tokens para desenvolvimento, seguindo padrões da Apple, Android e Web.	Figma	https://www.figma.com/community/plugin/1063959496693642315/UI-Color-Palette
Color Designer	Gera nuances, tons e harmonias de cores com base em camadas selecionadas ou estilos locais. Também cria gradientes escalonados entre duas cores. Principais recursos: gera tons, tons e harmonias de cores, integrado à seleção do usuário e estilos locais, gerador de gradientes, fácil de usar. O *plugin* suporta apenas preenchimentos sólidos. É open source!	Figma	https://www.figma.com/community/plugin/739475857305927370/Color-Designer
Color Shades	Permite gerar várias tonalidades a partir da mesma cor base, bem como do ajuste do contraste desejado.	Figma	https://www.figma.com/community/plugin/929607085343688745/Color-Shades
Color Luminance	Exibe a luminância relativa (quão brilhantes as cores parecem para nossos olhos) das cores selecionadas. As cores podem ser ajustadas para corresponder a níveis de luminância desejados. Quando apenas duas cores são selecionadas, sua taxa de contraste também é exibida.	Figma	https://www.figma.com/community/plugin/872423614371258087/Color-Luminance

Foundation: Color Generator	Conjunto de ferramentas que ajuda a criar paletas de cores rapidamente. Permite verificar a taxa de contraste diretamente do gerador de tons, além de permitir renderizar esquemas de cores e adicioná-los ao guia de estilo. Principais recursos: Perfis, Verificador de Contraste, Tokens de Cores, Gerador de Estilos, Renderização de Paleta, Seletor de Cores e Gerador de Nomes de Cores.	Figma	https://www.figma.com/community/plugin/1024452006068794933/Foundation%3A-Color-Generator
Palettes	Permite gerar e gerenciar estilos de cores com base em paletas perceptualmente uniformes.	Figma	https://www.figma.com/community/plugin/847120744711293046/Palettes
HCL Color	Permite equalizar a luminosidade aparente dos matizes e um melhor controle sobre o contraste. Isso ajuda a transmitir significado com cores, especialmente em visualização de dados, evitando padrões inexistentes devido à variação de luminosidade nos matizes. Também ajuda na consistência ao visualizar o trabalho em P&B.	Figma	https://www.figma.com/community/plugin/889859275438287504/HCL-Color
Color Palettes	Permite encontrar paletas de cores de uma enorme lista das melhores paletas de cores.	Figma	https://www.figma.com/community/plugin/740832935938649295/Color-Palettes
Color Blind	Permite visualizar os designs das 8 diferentes deficiências de visão de cores. Basta fazer uma seleção e o *plugin* criará versões clonadas com as cores alteradas para simular como seriam vistas por alguém com cada tipo de daltonismo. Cada cópia está em um grupo rotulado com o tipo de daltonismo.	Figma	https://www.figma.com/community/plugin/733343906244951586/Color-Blind
Color Kit	Gera tonalidades claras e escuras para uma cor particular selecionada com base nas configurações de ajuste.	Figma	https://www.figma.com/community/plugin/797696673804519719/Color-Kit
CoolHue - Gradient Color Palette	Permite cores gradientes de uma enorme lista dos melhores gradientes previamente selecionados.	Figma	https://www.figma.com/community/plugin/807561639084281386/CoolHue---Gradient-Color-Palette
Color scale	Permite criar escala de cores para construção de um Design System a partir da cor de um elemento selecionado.	Figma	https://www.figma.com/community/plugin/1103971792306373442/Color-scale
Color Shades & Tints Generator	Simplifica a geração de tons e sombras de cores desejadas. Suas principais características incluem seleção fácil de formas preenchidas com cores e geração de vários tons ou sombras ao mesmo tempo.	Figma	https://www.figma.com/community/plugin/1221190866182222979/Color-Shades-%26-Tints-Generator
CONTRASTE			
Contrast	Facilita a verificação de contrastes de cores. Ao selecionar uma camada, o *plugin* identifica a cor diretamente atrás dela, exibindo a relação de contraste e níveis de conformidade com as diretrizes WCAG. Ele escaneia páginas inteiras para relatar problemas de contraste textual, permitindo correções passo a passo. Funciona em qualquer elemento, oferece atualizações em tempo real e suporta cores sólidas, gradientes e imagens de fundo.	Figma	https://www.figma.com/community/plugin/748533339900865323/Contrast
A11y - Color	Garanta a legibilidade do seu texto seguindo as normas	Figma	https://www.figma.com/

Contrast Checker	WCAG. Este *plugin* verifica o contraste de cor de todo o texto visível em um quadro, fornecendo feedback sobre se ele atende aos níveis de conformidade AA e/ou AAA da WCAG. Também possui controles deslizantes para ajustar cores e entender como o contraste muda em tempo real. Atualmente, suporta apenas preenchimentos únicos e sólidos de 100%, trabalhando em resolver problemas de detecção de camada de fundo.		community/plugin/733159460536249875/A11y---Color-Contrast-Checker
Color contrast	Permite verificar o contraste dos elementos de acordo com as normas WCAG por meio da seleção de camadas. Além disso, permite realizar as mudanças necessárias para garantir um bom contraste diretamente pelo *plugin*.	Figma	https://www.figma.com/community/plugin/937465522075454889/Color-contrast
Contrast Checker	O *plugin* permite verificar o contraste do seu design de UI com base nos Critérios de Sucesso de Contraste 1.4.3 e 1.4.6 da WCAG. Ao selecionar e confirmar as cores, o aplicativo avalia se o contraste é adequado para textos normais e grandes (acima de 18 pontos). As categorias de conformidade incluem Nível A (ruim), Nível AA (bom) e Nível AAA (excelente). Além disso, você pode redefinir a interface para reavaliar, sem pressionar um botão específico. Isso garante uma análise eficaz e melhora a acessibilidade do design.	Figma	https://www.figma.com/community/plugin/1195158716202754858/Contrast-Checker
Color Contrast Checker	Fornece informações também baseadas na WCAG para contraste, a partir da seleção de duas camadas. Também informa a proporção de contraste.	Figma	https://www.figma.com/community/plugin/1218666737106149812/Color-Contrast-Checker
Use Contrast	Oferece acesso rápido a taxas adequadas de contraste de cores da WCAG por meio da seleção de qualquer camada de texto ou objeto no canvas para obter a pontuação. Ajuste suas cores com os controles do *plugin*.	Figma	https://www.figma.com/community/plugin/1149686177449921115/Use-Contrast
Contrast Description	Adiciona informações de taxa de contraste WCAG aos estilos de cor na paleta de cores, não interrompendo o fluxo de design para verificar acessibilidade! Isso ajuda a equipe a saber quais pares de cores são seguros, fornecendo informações contextuais na interface Figma. As informações de contraste são visíveis ao passar o mouse sobre os estilos de cor. Limitação: funciona apenas com cores sólidas, sem gradientes ou imagens.	Figma	https://www.figma.com/community/plugin/1010556985353208247/Contrast-Description
Contrast Checker	Permite testar se as cores utilizadas seguem os padrões WCAG para contraste. A cor de fundo é inferida através da seleção, permitindo avaliar o contraste das demais cores em relação à cor de fundo.	Figma	https://www.figma.com/community/widget/1123669584321839030
Stark Accessibility Tools	Recursos incluem taxas de contraste, simulações visuais, ordem de foco, anotações Alt-Text, tipografia e tamanhos de alvo. Para contraste também usa as normas WCAG, informando se as cores possuem contraste suficiente bem como o nível WCAG alcançado (AA, AAA etc).	Figma	https://www.figma.com/community/plugin/732603254453395948/Stark-Accessibility-Tools
Visual Contrast: Everything clearly with APCA	Permite garantir a visibilidade clara de suas cores com facilidade, oferecendo pré-visualização de contraste e texto, listas de contraste por nível para diferentes tipos de conteúdo e sugestões de cores para corrigir problemas de visibilidade. Também usa como base as normas WCAG.	Figma	https://www.figma.com/community/plugin/1090308131937420683/Visual-Contrast%3A-Everything-clearly-with-APCA
Simple WCAG 2.1 color contrast checker	Ao informar as cores de fundo e cor de texto manualmente, permite verificar o contraste entre as cores em relação às normas WCAG. Informa se o nível de contraste obtido passa nas avaliações necessárias para ser considerada com um bom nível de contraste.	Figma	https://www.figma.com/community/plugin/1220101069607769508/Simple-WCAG-2.1-color-contrast-checker

Color Contrast Generator	Permite selecionar uma cor de texto e uma cor de fundo e gerar um frame dentro do Figma que mostra as cores selecionadas, bem como a classificação se as cores selecionadas possuem um bom contraste ou não.	Figma	https://www.figma.com/community/plugin/1119749356529142486/Color-Contrast-Generator
Color Contrast Grid	Fornece as taxas de contraste para todas as combinações possíveis de estilos de cor. Principais funções: gera tabela de taxas de contraste para todos os estilos de cor locais, flexível com componentização, permite limitar exibição conforme o contraste.	Figma	https://www.figma.com/community/plugin/1039910246084959068/Color-Contrast-Grid
Caravage	Permite verificar a acessibilidade de contraste do design com base nas normas WCAG por meio da seleção de camadas de texto individualmente ou de varreduras que o *plugin* fornece automaticamente para cada texto para visualização das cores do texto, as cores do fundo diretamente atrás do texto, tamanho e peso da fonte, valor de contraste e o método de contraste de cores chamado *Accessible Perceptual Contrast Algorithm* (APCA), além do valor mínimo de contraste fornecido pela tabela de busca de contraste APCA.		https://www.figma.com/community/plugin/1059851135811973410/Caravage
TIPOGRAFIA			
Typestyles	Com este *plugin*, você pode gerar em lote estilos de texto de forma simples. Basta selecionar as tipografias desejadas, escolher modificadores para ajustar os estilos, como por exemplo tamanho de fonte, estilo da fonte e etc e gerar.	Figma	https://www.figma.com/community/plugin/803311677045533625/Typestyles
StyleList – text and color styles	Permite criar estilos de texto e cores de forma mais rápida e fácil. Para estilos de texto, é possível inserir o tamanho base, selecionar tamanhos desejados, escolher a escala e configurações de fonte para obter todos os estilos de texto. Em estilos de cores existem uma série de atributos configuráveis para gerar a paleta de cores desejada, como cor base, saturação, brilho e outros. O *plugin* é predominantemente voltado a tipografia, mas também oferece funcionalidades voltadas a cores.	Figma	https://www.figma.com/community/plugin/927255248672920500/StyleList-%E2%80%93-text-and-color-styles
Font Scale	Gera uma hierarquia tipográfica harmoniosa e consistente. Especialmente útil para manter proporções entre texto e títulos iguais. Permite selecionar um tamanho base, um fator de escala e conferir a pré-visualização.	Figma	https://www.figma.com/community/plugin/741231992144144738/Font-Scale
Typescale	Permite gerar rapidamente uma escala modular para tipografia. Ao selecionar um texto dentro do canva do Figma, o *plugin* permite configurar quantos tamanhos acima e abaixo do tamanho de fonte selecionado ele deve gerar com base no fator de escala.	Figma	https://www.figma.com/community/plugin/967802396210455992/Typescale
Textyles	Permite gerar rapidamente estilos de texto. Principais características incluem gerar estilos de texto com base em uma escala tipográfica modular usando valores arredondados ou exatos, começar com uma camada de texto selecionada para basear estilos em suas propriedades, apelidar a fonte e nomear cada tamanho para consistência, além de limpar todos os estilos de texto com um clique.	Figma	https://www.figma.com/community/plugin/804843548882105498/Textyles
Font Suggestion	O *plugin* utiliza a AI da API ChatGPT para sugerir a fonte ideal com base em requisitos de design, agilizando a seleção de fontes e substituindo a busca manual.	Figma	https://www.figma.com/community/plugin/1215375908423658346/Font-Suggestion
Fontpair	Permite testar combinações de fontes do Google Fonts de maneira rápida e intuitiva. Especialmente útil para seleção de fontes distintas para títulos e texto de corpo.	Figma	https://www.figma.com/community/plugin/1105220730154050357/Font

			pair
OUTROS			
UI Faces	O UI Faces reúne milhares de avatares que você pode filtrar cuidadosamente para criar personas perfeitas ou gerar avatares aleatórios. Os avatares são provenientes de várias fontes e podem ser filtrados por idade, gênero, emoção, etc.	Figma	https://www.figma.com/community/plugin/769664006254845172/UI-Faces

Fonte: Elaborada pelo autor

3.3.2. Para quais aspectos do design visual eles fornecem suporte e quais as suas funcionalidades?

Realizou-se uma análise de cada um dos *plugins* para determinar quais aspectos do design visual são abrangidos (Tabela 4). Para fins desta análise, os *plugins* foram categorizados em sete aspectos do design visual, previamente descritos na seção 2.2:

- **Acessibilidade Visual**: auxilia na garantia de que a interface seja projetada de maneira apropriada para pessoas com deficiências visuais, considerando contraste adequado, tamanhos de fonte escaláveis, entre outros.
- **Contraste**: auxilia na diferenciação entre elementos visuais, como texto e fundo, para garantir legibilidade e destacar informações importantes.
- **Cores**: auxilia na seleção e combinação de cores para transmitir informações, criar identidade visual e evocar emoções nos usuários.
- **Consistência**: auxilia na adoção de padrões de design consistentes ao longo da interface, para garantir que os elementos visuais se comportem de maneira previsível para os usuários.
- **Elementos Gráficos**: auxilia no uso de ícones, ilustrações e outros elementos visuais para representar ações, conceitos e informações de maneira visualmente clara e intuitiva.
- **Harmonia**: auxilia na avaliação de combinação equilibrada de elementos visuais, como cores, tipografia e elementos gráficos, para criar uma experiência esteticamente agradável e coesa.
- **Tipografia**: auxilia na escolha das fontes, tamanhos e estilos de texto para garantir a legibilidade, coerência e adequação ao contexto.

Tabela 6 — Funcionalidades dos *plugins* encontrados.

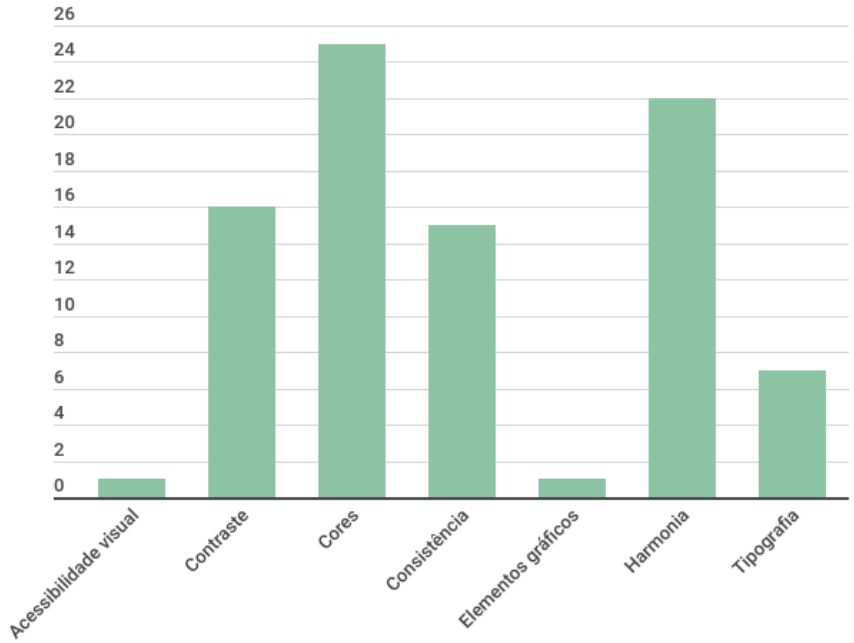
Nome do *plugin*/extensão	**Aspectos do design visual suportados**
UI Color Palette	cores, contraste, consistência
Color Designer	cores, harmonia
Color Shades	cores, harmonia, consistência
Color Luminance	cores, contraste
Foundation: Color Generator	cores, harmonia, consistência
Palettes	cores, harmonia, consistência
HCL Color	cores, harmonia
Color Palettes	cores, harmonia
Color Blind	cores, acessibilidade visual
Color Kit	cores, harmonia
CoolHue - Gradient Color Palette	cores, harmonia
Color scale	cores, harmonia, consistência
Color Shades & Tints Generator	cores, harmonia, consistência
Contrast	contraste, cores, harmonia
A11y - Color Contrast Checker	contraste, cores, harmonia
Color contrast	contraste, cores, harmonia
Contrast Checker	contraste, cores, harmonia
Color Contrast Checker	contraste, cores, harmonia
Use Contrast	contraste, cores, harmonia
Contrast Description	contraste, cores, harmonia, consistência
Contrast Checker	contraste, cores, harmonia, consistência
Stark Accessibility Tools	contraste, cores, harmonia, consistência
Visual Contrast: Everything clearly with APCA	contraste, cores, harmonia
Simple WCAG 2.1 color contrast checker	contraste
Color Contrast Generator	contraste
Color Contrast Grid	contraste
Caravage	contraste, cores, harmonia, consistência
Typestyles	tipografia, consistência, hierarquia
StyleList – text and color styles	tipografia, cores, harmonia, consistência, hierarquia
Font Scale	tipografia, consistência, hierarquia

Typescale	tipografia, consistência, hierarquia
Textyles	tipografia, consistência, hierarquia
Font Suggestion	tipografia
Fontpair	tipografia
UI Faces	elementos gráficos

Fonte: Elaborada pelo autor

Analisando o enfoque dos *plugins* encontrados, torna-se evidente que a maioria deles direciona sua atenção para os aspectos de cores, totalizando 25 *plugins*, e harmonia, com 22 *plugins* específicos voltados para essas áreas de design visual. Essa concentração expressiva em cores e harmonia reflete a importância fundamental desses elementos no processo de design e na criação de interfaces de usuário eficazes (Figura 25).

Figura 25 — Aspectos do design visual abordados pelos plugins



Fonte: Elaborado pelo autor

Observa-se também que a maioria desses *plugins* aborda mais do que um dos sete aspectos mapeados, o que sugere uma tendência de fornecer um suporte mais abrangente ao design visual. Esse enfoque em múltiplos aspectos do design visual tem o potencial de simplificar consideravelmente o processo de design, reduzindo a necessidade de utilizar vários *plugins* em paralelo para abordar diferentes aspectos do design.

Essa abordagem abrangente é particularmente benéfica porque reconhece a interconexão intrínseca entre os vários elementos de design visual. Por exemplo, ao

considerar cores, a harmonia desempenha um papel crucial, e o contraste é fundamental para a legibilidade. Portanto, *plugins* que abrangem cores, harmonia e contraste simultaneamente podem fornecer uma solução mais integrada e eficiente.

Vale destacar que, embora a maioria dos *plugins* ofereça suporte a uma variedade de aspectos do design visual, ainda existem algumas exceções. Alguns *plugins* se concentram exclusivamente em funcionalidades específicas, como contraste, tipografia e elementos gráficos. Isso pode ser atribuído à complexidade desses aspectos individuais e à necessidade de ferramentas especializadas para atender a requisitos específicos.

3.4. Discussão

A análise dos 34 *plugins* identificados nesta revisão revela um panorama diversificado e abrangente das ferramentas disponíveis para aprimorar o design visual em aplicações móveis. Entre esses *plugins*, pode-se destacar pontos de similaridade e diferença, proporcionando *insights* para determinar a utilidade de cada uma conforme as diferentes necessidades de design e desenvolvimento.

Dentre os *plugins* examinados, observa-se que um conjunto substancial deles foca no aspecto das cores, contraste, harmonia e tipografia. Dentre eles, destacam-se aqueles que auxiliam na geração de paletas de cores acessíveis e consistentes, verificação de contraste e harmonia, bem como aqueles voltados para a seleção e combinação de tipografias. Além disso, vários *plugins* se concentram especificamente em garantir a acessibilidade visual, incluindo a verificação do contraste de cores de acordo com as diretrizes WCAG.

Por exemplo, o *plugin* "*UI Color Palette*" se sobressai ao permitir a criação de paletas de cores coerentes e acessíveis, garantindo que as escolhas cromáticas atendam às diretrizes de acessibilidade. Além disso, o *plugin* oferece alternativas de espaços de cores como LCH, OKLCH, CIELAB, OKLAB e HSLuv, promovendo versatilidade na escolha de cores.

Em relação ao suporte ao contraste e acessibilidade visual, um grupo significativo de *plugins* se destaca por oferecer ferramentas para verificar e corrigir o contraste das cores utilizadas nas interfaces de usuário. A presença de diversos *plugins* com foco nesse aspecto é um reflexo da crescente importância atribuída à acessibilidade digital, visando garantir que os designs sejam legíveis e utilizáveis por todos os usuários, independentemente de suas limitações visuais. Um exemplo é o *plugin* "*Color Blind*" como uma ferramenta focada na

acessibilidade visual. Ao possibilitar a simulação de como diferentes deficiências de visão de cores perceberiam o design, ela pode contribuir para a criação de interfaces mais inclusivas. O "*Contrast Description*", por sua vez, difere-se ao fornecer informações detalhadas sobre as taxas de contraste WCAG diretamente na paleta de cores, permitindo uma análise mais abrangente do design.

No que diz respeito à harmonia e consistência das cores, vários *plugins* se sobressaem ao oferecer funcionalidades para criar paletas de cores coesas e uniformes, além de permitir a geração de gradientes e tons a partir de cores-base. Esses *plugins* são especialmente valiosos para manter uma identidade visual consistente ao longo do design de um aplicativo.

No entanto, é importante notar que, embora muitos *plugins* ofereçam recursos robustos para tratamento de cores, harmonia e consistência, nem todos abordam igualmente outros aspectos do design visual. Por exemplo, o *plugin* "*Font Suggestion*" se destaca ao usar a API ChatGPT para sugerir fontes ideais com base nos requisitos de design, destacando a importância da tipografia na experiência do usuário, além de também ser um exemplo de aplicação de técnicas de Inteligência Artificial (IA) no campo de design visual. Mesmo que nesta revisão a API ChatGPT seja a única utilizando técnicas de IA, espera-se pelo caráter emergente o surgimento de mais soluções utilizando a IA no futuro.

Em relação ao escopo de design suportado, os *plugins* variam consideravelmente. Enquanto muitos são especializados em cores, contraste e tipografia, alguns lidam com aspectos únicos. A "*UI Faces*" é uma ferramenta especializada na geração de avatares para personas, destacando como o design de personas também é fundamental no processo de desenvolvimento de aplicativos.

Ao considerar o contexto dos *plugins*, é crucial reconhecer que cada uma atende a diferentes necessidades e propósitos. Enquanto algumas oferecem soluções abrangentes para design visual, outras abordam especificidades, como a acessibilidade.

Ao final observa-se que não foi encontrado nenhum *plugin* para Penpot, uma vez que essa funcionalidade ainda não existe na ferramenta, como também não foram encontrados *plugins* específicos para o desenvolvimento de apps com App Inventor.

Ameaças à validade. No esforço de assegurar a validade dos resultados deste mapeamento, foram identificadas potenciais ameaças e implementadas estratégias correspondentes. O maior risco é a omissão de um *plugin* relevante. Para prevenir esse risco, foram realizadas buscas em várias fontes com strings de busca, incluindo sinônimos. A seleção dos *plugins* relevantes foi feita sistematicamente, a partir de critérios de inclusão e

exclusão. A extração das informações relevantes também foi feita por meio de uma análise minuciosa de cada um dos *plugins*, além da descrição geral de cada um deles.

4. SOLUÇÃO

Nesta seção, é apresentado o desenvolvimento de uma ferramenta *web* voltada para auxiliar no design visual de apps a serem implementados com App inventor no contexto do ensino de computação na Educação Básica.

Essa ferramenta visa auxiliar o processo de desenvolvimento de interfaces de aplicativos móveis, fornecendo assistência em áreas críticas do design visual, incluindo a definição da paleta de cores, seleção de tipografia e outros aspectos essenciais do design de UI de aplicativos, visando contribuir com a melhoria da usabilidade e estética visual dos apps a serem desenvolvidos.

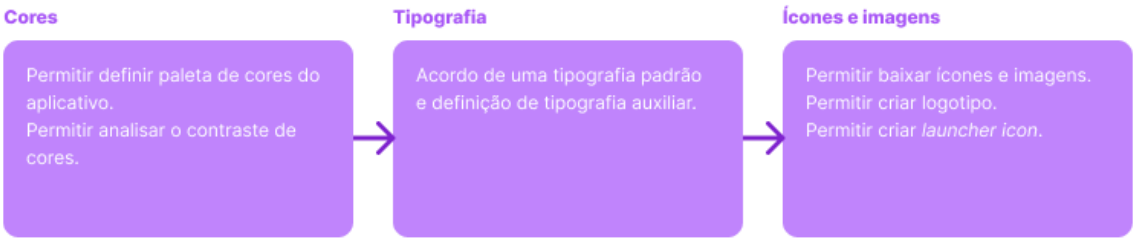
No âmbito da aplicação no ensino de computação na Educação Básica, por meio do desenvolvimento de aplicativos com o App Inventor, esta ferramenta visa fornecer um suporte educacional direcionado para atingir objetivos de aprendizagem relacionados ao design visual de interfaces. Especificamente, será alinhada com as diretrizes estabelecidas pelo *American Institute of Graphic Arts* (AIGA), que se concentram em conceitos de design visual adequados ao contexto da Educação Básica. Isso é particularmente significativo, considerando a necessidade de levar em consideração as limitações inerentes à ferramenta App Inventor em relação ao design de interfaces, com o objetivo final de possibilitar a criação de aplicativos funcionais nesse ambiente baseado em blocos, adequado a esse nível escolar. Além disso, todo o suporte oferecido pela solução está alinhado com a literatura de referência e os principais guias de estilo, incluindo o Material Design 3 e as diretrizes do WGCA. Isso assegura que os princípios do design de interface estejam alinhados com as melhores práticas adotadas.

Com o objetivo de facilitar a adoção prática, objetiva-se criar uma solução única e abrangente que englobe o suporte a diversos aspectos do design visual por meio de uma plataforma on-line, acessível de forma gratuita. Essa abordagem objetiva a simplicidade e acessibilidade, eliminando possíveis barreiras de entrada.

4.1. Análise de requisitos

Considerando o objetivo de desenvolver uma ferramenta *web* que auxilie no design de interfaces de aplicativos móveis a serem implementados posteriormente usando o App Inventor, conforme o processo apresentado na Seção 4.2, foram identificados os requisitos funcionais e não funcionais. Na Figura 16, é apresentado o fluxo esperado para o uso da ferramenta, abrangendo todo o processo de design visual e adaptando-o para o contexto educacional na educação básica.

Figura 26 — Fluxo de trabalho esperado para ferramenta web



Fonte: Elaborado pelo autor

Requisitos funcionais

Os requisitos funcionais da ferramenta são apresentados na Tabela 7.

Tabela 7 — Requisitos funcionais da ferramenta.

ID	Requisito	Descrição
RF01	Criação de painel semântico para seleção de paleta de cores	A ferramenta deve permitir que o usuário crie um painel semântico personalizado, que servirá como base para a seleção de cores da paleta a ser utilizada no design da interface.
RF02	Seleção de paleta de cores	A ferramenta deve permitir que o usuário selecione e personalize uma paleta de cores com base em a) painel semântico b) uma imagem escolhida extraindo cores representativas da imagem para uso no design da interface ou c) escolhendo uma cor primária aleatoriamente ou c) uma cor escolhida.
RF03	Seleção de cores harmônicas	A ferramenta deve permitir que o usuário escolha cores que funcionem bem juntas, garantindo uma estética equilibrada e agradável considerando as combinações complementar, análoga e monocromática conforme o círculo cromático
RF04	Verificação de contraste de cores	A ferramenta deve ser capaz de analisar o contraste entre as cores selecionadas na interface do usuário e fornecer feedback sobre a

		conformidade com os padrões de acessibilidade, como as diretrizes WCAG.
RF05	Definição de tipografia padrão	A ferramenta deve propor a tipografia padrão conforme as diretrizes do Material Design 3.
RF06	Definição de tipografia auxiliar por tema do app	A ferramenta deve permitir a definição da tipografia de um aplicativo com base no tema escolhido, que deve ter a ver com o propósito do aplicativo.
RF07	Seleção de ícones de sistema	A ferramenta deve integrar a biblioteca de ícones acessíveis e sem direitos autorais (Google Material Icons) que possam ser usadas nas interfaces de apps. Além disso, a ferramenta deve permitir a pesquisa e a seleção de ícones apropriados para diferentes finalidades. Deve também permitir parametrizar os ícones em relação ao estilo, preenchimento, e cor (propondo também a cor primária da paleta).
RF08	Seleção de imagens	A ferramenta deve permitir usuários buscarem por imagens com base em palavras-chave e fornecerá opções para recolorir imagens de acordo com a paleta de cores definida para o aplicativo.
RF09	Criação de logotipo para o aplicativo	A ferramenta deve permitir criar um logo para o app usando a tipografia auxiliar e um ícone que pode ser escolhido pelo usuário. Deve usar a paleta de cores para propor automaticamente alternativas de cores para o logo.
RF10	Criação de ícone de produto	A ferramenta deve automaticamente criar um *launcher icon* usando a paleta de cores e o ícone do logotipo.
RF11	Exportação do design visual	A ferramenta deve exportar todas as informações do design visual em formato json ou pdf, quando conveniente, e todas as imagens em formato png, com exceção de ícones que serão exportados em svg.

Fonte: Elaborada pelo autor

Requisitos não funcionais

Os requisitos não funcionais da ferramenta identificados conforme a análise de contexto são apresentados na Tabela 8.

Tabela 8 — Requisitos não funcionais da ferramenta.

ID	Requisito	Descrição
RFN01	Interface intuitiva	A interface da ferramenta deve ser intuitiva e fácil de usar, para atender às necessidades de usuários de diferentes níveis de experiência. Sendo assim, 75% dos usuários do público-alvo devem conseguir completar as tarefas com uma satisfação de no mínimo 75 pontos no System Usability

		Scale[4].
RFN02	Linguagem de programação JavaScript	A ferramenta deve ser desenvolvida na linguagem de programação JavaScript, juntamente com a linguagem de marcação HTML e a linguagem de estilização CSS.
RFN03	Tempo de resposta	A ferramenta deve ser otimizada para oferecer tempos de resposta dentro do tempo aceitável. Sendo assim, o tempo de resposta de qualquer funcionalidade deve ser inferior a 10 segundos.

Fonte: Elaborada pelo autor

4.2. Design conceitual da ferramenta

Para resolver os problemas relacionados ao desenvolvimento do design visual no contexto educacional do App Inventor, foi decidido criar uma ferramenta *web* acessível e intuitiva. Os principais motivos para essa decisão incluíram a necessidade de facilitar o acesso e uso da ferramenta sem a necessidade de instalar ou baixar qualquer software adicional, garantindo assim que tanto alunos quanto professores possam utilizar a ferramenta diretamente a partir de qualquer navegador. A simplicidade e objetividade foram critérios fundamentais no desenvolvimento da ferramenta, visando minimizar a curva de aprendizado e proporcionar uma experiência de usuário fluida e eficiente.

Nesta seção, é apresentada a solução desenvolvida para suportar o design visual de interfaces de usuário de aplicativos móveis, especificamente no contexto do ensino de computação na Educação Básica com o uso do App Inventor para a implementação do aplicativo projetado. A ferramenta web criada deve abranger várias funcionalidades essenciais que permitam aos usuários desenvolver interfaces de aplicativos de maneira eficiente e estética. A ferramenta, portanto, foi organizada em cinco módulos principais: cores, tipografia, ícones, imagens e criação de logo. Cada módulo atua separadamente, mas pode ser utilizado em qualquer ordem conforme a necessidade do usuário, no entanto, recomenda-se um fluxo padrão que começa com a especificação de cores, seguido pela tipografia, imagens, ícones e, por fim, a criação de logo, para garantir um processo de design coeso e bem-estruturado. A separação da ferramenta em diferentes módulos foi estrategicamente planejada para dividir o processo de design visual em etapas bem definidas, facilitando o entendimento e a usabilidade para os usuários, além de permitir uma maior flexibilidade e foco em cada aspecto específico do design. Além disso, essa divisão modular também

[4] https://www.interaction-design.org/literature/article/system-usability-scale

facilitaria a construção da ferramenta, uma vez que seu desenvolvimento pode ser realizado de modo incremental, permitindo a finalização e validação de cada módulo individualmente para garantir a qualidade esperada.

Módulo de cores

O módulo de cores na ferramenta possui um conjunto de 3 funcionalidades, que estão interligadas entre si: definição de painel semântico, definição de paleta de cores e análise de contraste. A criação de um painel semântico personalizado é uma funcionalidade onde o usuário pode buscar imagens sem direitos autorais usando a API pública do Pexels[5]. Este painel semântico pode servir como base para a seleção de uma paleta de cores, extraindo cores representativas das imagens selecionadas a fim de se escolher uma cor principal para montar a paleta de cores. Alternativamente, o usuário pode fazer *upload* de uma imagem própria ou escolher a cor primária aleatoriamente. A ferramenta também suporta também a seleção de cores harmônicas, visando uma combinação estética equilibrada com base em princípios de teoria da cor, como combinações complementares, análogas e monocromáticas. Por fim, a funcionalidade de verificação de contraste das cores é feita a partir das cores escolhidas na paleta de cores ou de cores informadas pelo usuário através dos códigos hexadecimais fornecidos, que verifica se o conjunto de cores fornecidos está em conformidade com as diretrizes de acessibilidade WCAG, fornecendo *feedback* em tempo real.

Em relação às exportações nesse módulo, o painel semântico pode ser baixado pelo usuário em formato PNG, enquanto a paleta de cores gera um arquivo PDF contendo a definição dos tipos de cores seguindo os princípios do Material Design.

Módulo de tipografia

O módulo de tipografia propõe como tipografia padrão a Roboto, conforme as diretrizes do Material Design 3, para assegurar a legibilidade e coerência visual. Além disso, os usuários podem definir a tipografia auxiliar com base no tema do aplicativo, ajustando fontes de acordo com o propósito do app. Para suportar a escolha de fonte auxiliar, a ferramenta sugere fontes Google de acordo com temas e permite personalizações adicionais para atender às necessidades específicas do design.

[5] https://www.pexels.com/api/

Nesse módulo as tipografias podem ser baixadas diretamente a partir do site Google Fonts, que hospeda todas as tipografias apresentadas na ferramenta como recomendações. Todas as fontes apresentadas possuem a licença SIL Open Fonts[6], que permite o uso livre das fontes recomendadas.

Módulo de ícones

Para a seleção de ícones, a ferramenta integra a biblioteca de ícones do Google Material Icons, permitindo que os usuários busquem e selecionem ícones acessíveis e sem direitos autorais. Os ícones podem ser parametrizados em relação ao estilo, preenchimento e cor, para assegurar consistência visual com a paleta de cores escolhida. Isso facilita a incorporação de ícones apropriados para diferentes finalidades na interface do aplicativo.

A ferramenta permite exportar cada um dos ícones em formato SVG, que podem ser posteriormente importados em ferramentas de design.

Módulo de imagens

No módulo de imagens, a ferramenta permite a busca de imagens a serem utilizadas nas telas do aplicativo com base em palavras-chave por meio da API do Pexels, que possuem licença permissiva para download e uso livre[7]. Além disso, a ferramenta oferece opções para recolorir as imagens de acordo com a paleta de cores definida, visando assim que todas as imagens usadas no aplicativo estejam em harmonia com o design geral.

A exportação dessas imagens é feita individualmente em formato PNG.

Módulo de criação de logo e ícone de produto

Para a criação de logotipos, a ferramenta permite que os usuários combinem a tipografia auxiliar com um ícone escolhido, utilizando a paleta de cores definida para propor automaticamente alternativas de cores para um logo do aplicativo. A criação do logo pode ser configurada em termos de cor de texto/ícone e fundo, layout, fontes e outras opções que permitem tornar cada logo gerado algo único e específico pro contexto do aplicativo.

---

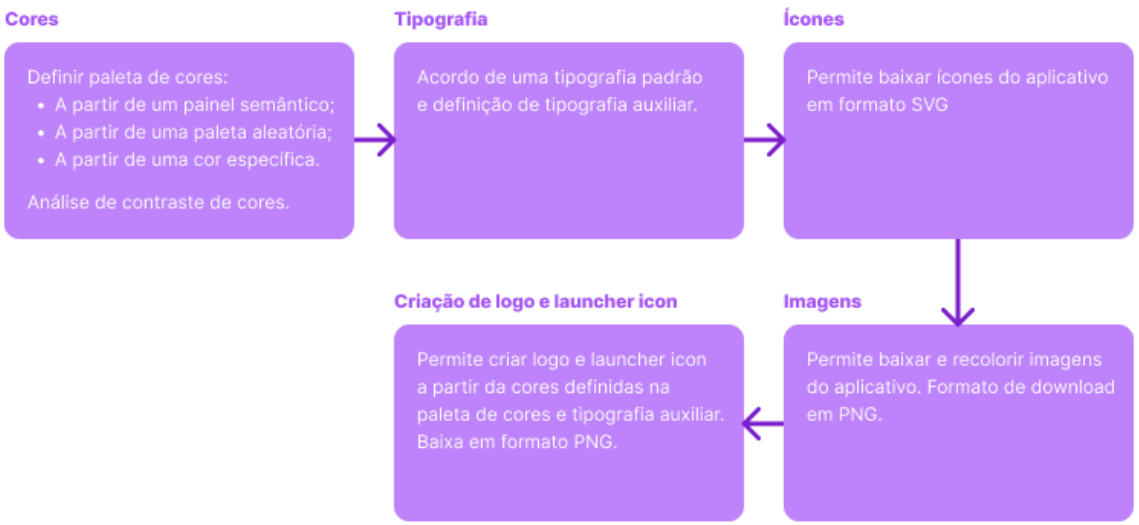
[6] https://openfontlicense.org/
[7] https://www.pexels.com/license/

Além disso, a ferramenta pode gerar automaticamente um ícone de produto usando a paleta de cores e o ícone do logotipo, assegurando que todos os elementos visuais estejam alinhados e coesos.

Após concluída a elaboração do logotipo e ícone de produto, os usuários podem baixar os artefatos gerados em formato PNG.

A divisão da plataforma em módulos foi uma escolha estratégica para separar os processos do design visual em etapas bem definidas, facilitando o entendimento e a usabilidade para os usuários. Essa abordagem modular permite uma flexibilidade maior no processo de design, permitindo que os usuários se concentrem em um aspecto específico do design visual de cada vez, mas ainda mantendo a capacidade de integrar todos os elementos de maneira coesa e harmoniosa. Apesar dos módulos serem individuais e funcionarem de maneira interdependente, alguns podem trocar informações entre si. Por exemplo, o usuário pode definir uma paleta de cores padrão no módulo de cores, que pode ser utilizada em outros módulos, como no de ícones e no de imagens. Da mesma forma, o módulo de tipografia permite ao usuário criar um logotipo a partir de uma tipografia auxiliar, garantindo que todos os elementos visuais estejam alinhados com a identidade visual do aplicativo. Na Figura 26, é apresentado todos os módulos da ferramenta e o fluxo esperado para o uso da ferramenta.

Figura 27 — Módulos da ferramenta e fluxo padrão.



Fonte: Elaborado pelo autor

4.3. Prototipação

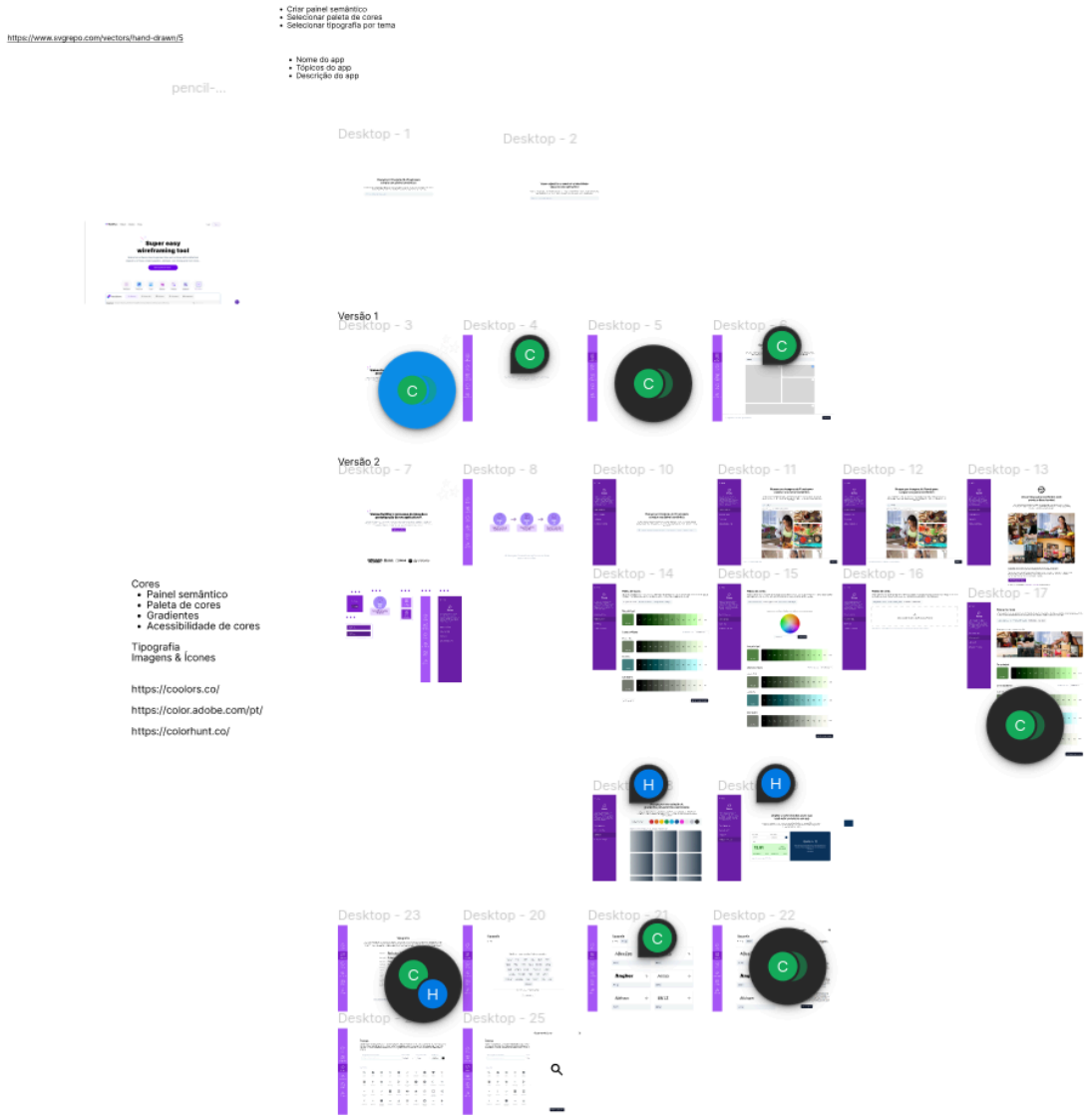
A prototipação é uma etapa crucial no desenvolvimento de qualquer projeto de design, especialmente no contexto do design de interfaces de usuário, pois permite a visualização e experimentação das ideias antes de investir tempo e recursos significativos na implementação final. Por meio de protótipos, é possível identificar e corrigir problemas de usabilidade, validar conceitos e obter *feedback*, visando que o produto final atenda às expectativas e necessidades dos usuários.

Para construção desta ferramenta *web*, a prototipação desempenhou um papel fundamental na sua construção. Para assegurar a qualidade e a eficácia de cada módulo da ferramenta, foi adotada a prática de criar protótipos de alta fidelidade no Figma antes de iniciar a implementação, o que permitiu desenvolver protótipos detalhados que refletiam com precisão o design e as funcionalidades pretendidas.

Cada módulo, incluindo cores, tipografia, ícones, imagens e criação de logo, foi primeiro prototipado no Figma, e durante essa fase, os esforços foram concentrados em aspectos críticos do design, como a usabilidade e a estética visual. Esses protótipos foram então validados em sessões de revisão com a orientadora do trabalho, onde após cada feedback foram realizados refinamentos no design proposto até chegar em sua versão final.

Essa abordagem economizou tempo e recursos, uma vez que implementar a solução de fato seria muito mais custosa do que realizar seu protótipo primeiro e validar todas as ideias a fim de se eliminar todas as possíveis lacunas. Figura 27 apresenta de maneira geral como foram conduzidos os protótipos dentro do Figma.

Figura 28 — Panorama geral dos protótipos no Figma.



Fonte: Elaborado pelo autor

4.4. Definição de identidade visual

Para assegurar que a ferramenta web desenvolvida possua uma identidade visual coerente e reconhecível, foi criada uma identidade visual abrangente. Esta identidade compreende a definição de uma cor principal, a criação de logotipos para serem utilizados na ferramenta, e a escolha de um nome que represente claramente o propósito e a funcionalidade da plataforma.

Cor principal

A cor principal escolhida para a identidade visual da ferramenta foi a cor roxa. Ela foi selecionada para transmitir modernidade. Esta cor é utilizada consistentemente em todos os elementos visuais da ferramenta, desde os botões e menus até os detalhes decorativos, garantindo uma aparência coesa e profissional. Figura 28 apresenta a paleta de cores definida.

Figura 29 — Paleta de cores da ferramenta.

Fonte: Elaborado pelo autor

Definição do nome da ferramenta

O nome da ferramenta foi cuidadosamente escolhido para refletir seu propósito e funcionalidades. "AppDesignFlow" é uma junção de três conceitos-chave:

- App: Abreviação de aplicativo, indicando o foco da ferramenta no design de interfaces de aplicativos móveis.
- Design: Enfatiza o aspecto criativo e visual da ferramenta, que auxilia no desenvolvimento de interfaces estéticamente agradáveis e funcionais.
- Flow: Representa o fluxo de design, que é a sequência de etapas e processos que a ferramenta facilita para a construção de aplicativos.

O nome "AppDesignFlow" comunica claramente a essência da ferramenta, que é proporcionar um fluxo contínuo e eficiente para o design de interfaces de aplicativos, desde a concepção até a implementação.

Logotipos

Além da definição da paleta de cores da ferramenta, diversos logotipos foram criados para representar a ferramenta de maneira visualmente atraente e memorável. Estes logotipos incorporam a cor principal definida e foram projetados para serem utilizados em diferentes contextos. A Figura 29 apresenta o logotipo e suas variações.

Figura 30 — Logotipo da ferramenta e variações.

Fonte: Elaborado pelo autor

4.5. Implementação

A implementação da ferramenta foi realizada de maneira incremental, abordando um módulo por vez após a conclusão e validação de cada protótipo. Este método assegurou que cada funcionalidade fosse desenvolvida de forma coerente e com a qualidade esperada, permitindo ajustes e melhorias contínuas com base no feedback recebido durante a fase de prototipação. Além disso, a ferramenta foi desenvolvida usando React[8] e Next.js[9].

Principais tecnologias utilizadas

React é uma biblioteca JavaScript amplamente utilizada para a construção de interfaces de usuário devido a sua capacidade de criar componentes reutilizáveis e seu eficiente sistema de gerenciamento de estados, que o tornam ideal para aplicações dinâmicas e interativas. Next.js, por sua vez, é um *framework* de desenvolvimento baseado em React que oferece funcionalidades como renderização no lado do servidor e geração de sites estáticos, proporcionando uma performance otimizada e uma excelente experiência de

[8] https://react.dev/
[9] https://nextjs.org/

desenvolvimento. JavaScript, é a linguagem de programação subjacente a ambas as tecnologias, é essencial para o desenvolvimento *web* moderno e é conhecida por sua versatilidade e ampla adoção na indústria.

Essas tecnologias foram escolhidas não apenas por estarem em ascensão no mercado e serem altamente robustas, mas também devido ao conhecimento prévio do autor nessas ferramentas. Além disso, as necessidades do projeto, que exigiam uma plataforma interativa e eficiente, justificaram a escolha dessas ferramentas, assegurando que a implementação atendesse aos requisitos funcionais e de desempenho.

Bibliotecas utilizadas

O projeto utilizou diversas bibliotecas para facilitar o desenvolvimento e aprimorar a funcionalidade da ferramenta:

Tabela 9 - Lista de bibliotecas das principais bibliotecas utilizadas no projeto e suas respectivas licenças.

Nome da Biblioteca	**Descrição**	**Licença**
@material/material-color-utilities	Utilizada para gerenciar e aplicar esquemas de cores seguindo os padrões do Material Design.	Licença Apache 2.0
@radix-ui/react-(label, radio-group, select, slider, slot, switch, tabs, tooltip)	Conjunto de componentes acessíveis e estilizáveis que facilitaram a criação de interfaces de usuário consistentes e acessíveis.	Licença MIT
chroma-js	Biblioteca para manipulação de cores, usada para calcular e ajustar esquemas de cores.	Licença BSD 3
class-variance-authority e clsx	Ferramentas para aplicar classes condicionalmente, ajudando a gerenciar estilos de maneira eficiente.	Licença Apache 2.0
framer-motion	Utilizada para adicionar animações fluidas e interações dinâmicas na interface.	Licença MIT
html-to-image	Ferramenta para converter elementos HTML em imagens, útil para a exportação de designs.	Licença MIT
lodash	Biblioteca utilitária que facilita a manipulação de arrays, objetos e outros tipos de dados.	Licença MIT
lucide-react	Conjunto de ícones utilizados na interface da ferramenta.	Licença ISC

material-symbols	Biblioteca de símbolos do Material Design.	Licença Apache 2.0
node-vibrant	Utilizada para extração de cores dominantes de imagens.	Licença MIT
react-color	Componentes de seleção de cores para React, essenciais para a funcionalidade de seleção e personalização de paletas de cores.	Licença MIT
tailwind-merge	Ferramenta para combinar classes utilitárias de Tailwind CSS.	Licença MIT
tailwindcss-animate	Biblioteca para adicionar animações aos componentes estilizados com Tailwind CSS.	Licença MIT

Fonte: Elaborada pelo autor

Essas bibliotecas foram integradas para oferecer uma experiência de usuário rica e funcional, atendendo aos requisitos definidos.

4.5.1. Arquitetura do sistema

A arquitetura da ferramenta é composta por vários componentes que interagem entre si para fornecer uma plataforma coesa e eficiente. Essa arquitetura baseada em componentes possui uma clara separação entre o *frontend* e o *backend*, respeitando também os princípios de desenvolvimento adotados no *framework* Next.js. Esta abordagem facilita a manutenção e a escalabilidade da ferramenta.

Componentes principais

Frontend:

- React: Utilizado para construir a interface de usuário com componentes reutilizáveis.
- Next.js: Utilizado para renderização no lado do servidor e geração de sites estáticos.
- Chroma-js e react-color: Utilizados para manipulação e seleção de cores.
- Framer-motion: Utilizado para animações.
- Bibliotecas @radix-ui: Utilizadas para componentes de interface acessíveis.

Backend:

- Next.js API Routes: Utilizadas para criar endpoints para manipulação de dados e lógica de negócio.
- Node.js: Ambiente de execução para construir o backend.

Além desses componentes, houve também a integração da ferramenta com plataformas externas, como é o caso das APIs do Pexels e também do Google Fonts. Para fazer a integração com essas plataformas, foram utilizadas APIs no padrão REST que são chamadas através do backend da aplicação, através do Next.js API Routes, por meio da qual trocam informações com as plataformas por meio de chamadas HTTP/HTTPS utilizando JSON como formato de troca de dados.

Compatibilidade com dispositivos

A ferramenta web foi projetada para funcionar em dispositivos *desktop*, com suporte otimizado para telas a partir de 800px de largura. A decisão de não oferecer suporte a dispositivos móveis foi baseada na observação de que os casos de uso principais não incluíam esses dispositivos. Portanto, a interface foi desenhada para maximizar a eficiência e a usabilidade em ambientes de desktop, onde a maioria dos usuários tende a realizar tarefas de design visual.

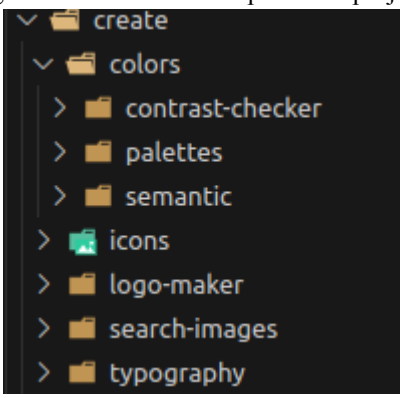
Escalabilidade e manutenibilidade

A arquitetura da ferramenta foi projetada para ser escalável, utilizando componentes desacoplados e serviços independentes. As práticas de codificação limpa e a utilização de padrões de design garantem que a ferramenta possa ser facilmente mantida e expandida no futuro. Cada módulo foi desenvolvido de forma independente e pode funcionar de forma separada dos demais módulos da ferramenta, de forma a tornar mais fácil a manutenção dos módulos no futuro e a adição de novos módulos.

Organização de pastas da ferramenta

Para se ter uma boa arquitetura, além dos fatores relacionados ao planejamento, definição dos componentes, pensamento voltado a escalabilidade e manutenibilidade, a organização das pastas da ferramenta também desempenhou um papel crucial para garantir a manutenibilidade, escalabilidade e a clareza do código da ferramenta. Para essa ferramenta, a estrutura de pastas foi planejada para refletir os diferentes módulos e funcionalidades do sistema, facilitando o desenvolvimento incremental e a navegação pelo código. A Figura 30 ilustra a hierarquia de pastas utilizada na ferramenta.

Figura 31 — Estrutura de pastas do projeto.



Fonte: Elaborado pelo autor

A pasta *create* é a principal, contendo todos os módulos relacionados à criação e design dentro da ferramenta. Esta pasta é a raiz dos diferentes componentes e funcionalidades de design que a ferramenta oferece.

Dentro da pasta *colors*, tem a representação do módulo de cores com várias subpastas especializadas, para cada uma das funcionalidades específicas:

- *contrast-checker* que contém os arquivos e componentes responsáveis pela verificação de contraste entre cores;
- *palettes* que abriga os componentes relacionados à criação e personalização de paletas de cores e
- *semantic* que contém os componentes para a criação de painéis semânticos.

A pasta *icons* contém os arquivos e componentes para a seleção e personalização de ícones. Na pasta *logo-maker*, tem os componentes necessários para a criação de logotipos. A pasta *search-images* abriga os componentes que permitem aos usuários buscar por imagens utilizando palavras-chave. Finalmente, a pasta *typography* contém os arquivos e componentes para a definição de tipografia.

Essa estrutura de pastas fornece diversos benefícios para construção da ferramenta, como, por exemplo:

1) modularidade: cada funcionalidade ou grupo de funcionalidades está claramente separado em sua própria pasta, facilitando a manutenção e a adição de novas características sem interferir em outras partes do projeto;

2) facilidade de navegação: a organização lógica e intuitiva das pastas torna mais fácil para encontrar e modificar componentes específicos, aumentando a eficiência durante o desenvolvimento;

3) manutenibilidade: a estrutura modular facilita a manutenibilidade do projeto. Novos módulos ou funcionalidades podem ser adicionados sem grandes reestruturações, garantindo que o projeto possa crescer de maneira organizada;

4) colaboração: apesar de a ferramenta ter sido construída por apenas um desenvolvedor, a clareza na organização das pastas facilita a colaboração entre diferentes desenvolvedores, permitindo que cada um trabalhe em módulos específicos sem causar conflitos no código. Isso significa que a estrutura de pastas adotada está preparada para múltiplos desenvolvedores trabalharem no mesmo código para adição de novas funcionalidades e módulos.

Em resumo, a organização das pastas da ferramenta foi planejada para suportar um desenvolvimento eficiente e escalável, refletindo a divisão modular dos diferentes aspectos do design visual que a ferramenta oferece e facilitar a construção do projeto de TCC.

Geração de esquemas de cores

A geração de esquemas de cores é feita utilizando a biblioteca *@material/material-color-utilities*, que segue os padrões do Material Design. Esta biblioteca define internamente o formato correto esperado pelo Material Design para paletas de cores, garantindo que as cores selecionadas estejam em conformidade com as melhores práticas de design visual.

Exportação de imagens

O *download* de imagens em formato PNG é realizado utilizando bibliotecas externas que fazem a conversão de HTML para esses outros formatos. A biblioteca *html-to-image* é utilizada para converter elementos HTML em imagens PNG, garantindo uma exportação

precisa e de alta qualidade dos designs visuais. Também são utilizadas outras bibliotecas para exportar arquivos em outros formatos, como é o caso do SVG para ícones.

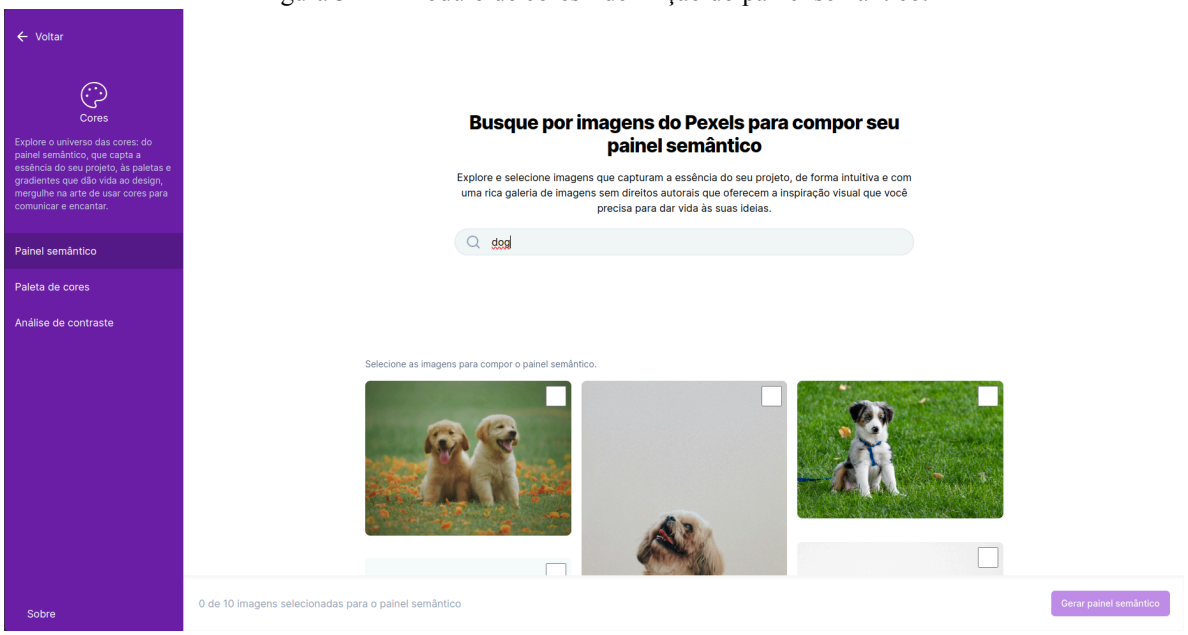
4.5.2. Demonstração do sistema

Para ilustrar as funcionalidades e a interface da ferramenta desenvolvida, esta seção apresenta capturas de tela (prints) de cada um dos módulos do sistema. Essas imagens demonstram as principais características e o fluxo de uso, proporcionando uma visão de como a ferramenta opera na prática.

Módulo de cores

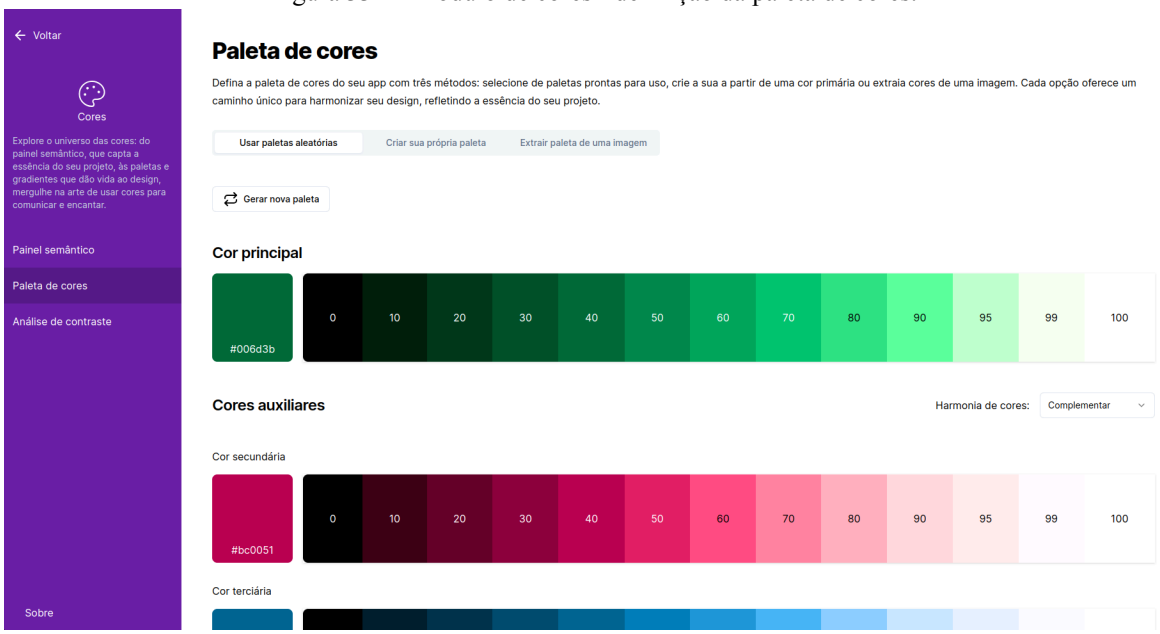
Abrange os requisitos funcionais 1, 2, 3, 4 e 11.

Figura 32 — Módulo de cores - definição do painel semântico.



Fonte: Elaborado pelo autor

Figura 33 — Módulo de cores - definição da paleta de cores.



Fonte: Elaborado pelo autor

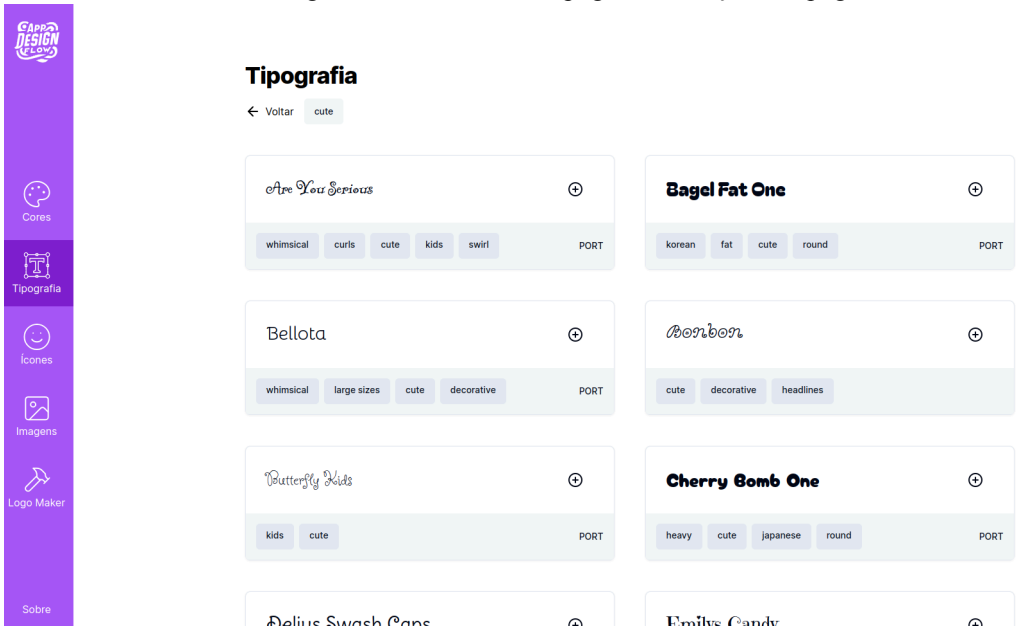
Figura 34 — Módulo de cores - análise de contraste.

Fonte: Elaborado pelo autor

Módulo de tipografia

Abrange os requisitos funcionais 5 e 6.

Figura 35 — Módulo de tipografia - seleção de tipografia.

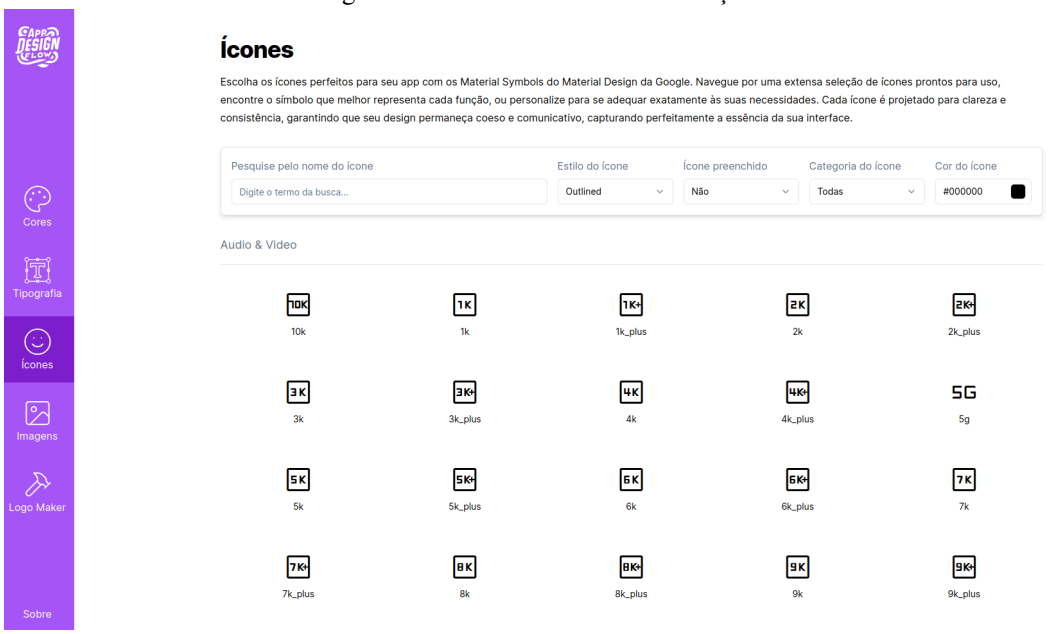


Fonte: Elaborado pelo autor

Módulo de ícones

Abrange os requisitos funcionais 7 e 11.

Figura 36 — Módulo de ícones - seleção de ícones.

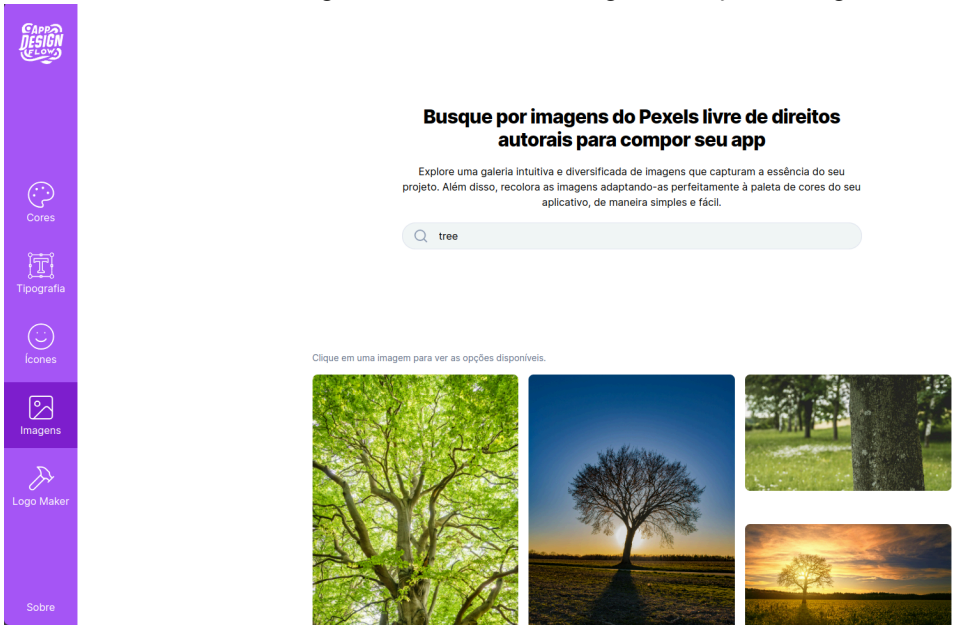


Fonte: Elaborado pelo autor

Módulo de imagens

Abrange os requisitos funcionais 8 e 11.

Figura 37 — Módulo de imagens - seleção de imagens.

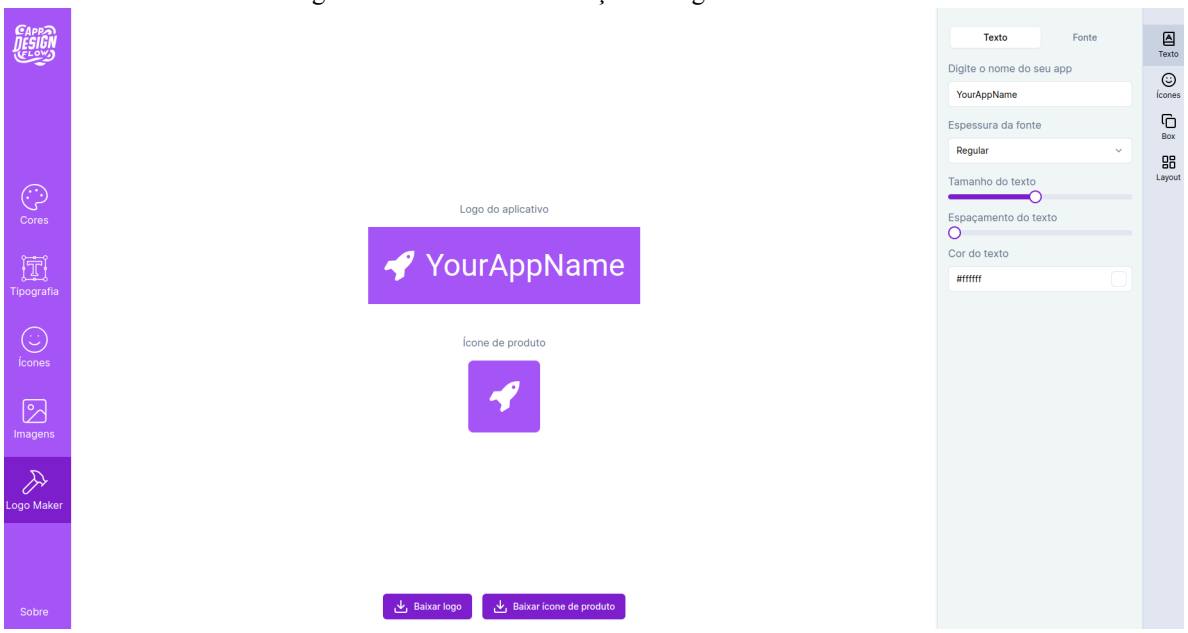


Fonte: Elaborado pelo autor

Módulo de criação de logo e *launcher icon*

Abrange os requisitos funcionais 9, 10 e 11.

Figura 38 — Módulo de criação de logo e *launcher icon*.



Fonte: Elaborado pelo autor

4.6. Testes do sistema

Os testes do sistema foram realizados de maneira sistemática e por etapas, assegurando que cada módulo fosse testado ao final da sua implementação. A cada módulo concluído, o código era implantado em um servidor gratuito da Vercel[10], permitindo que pudesse ser acessado e testado por um grupo seleto de usuários. Este grupo incluía o próprio autor da ferramenta, a orientadora do trabalho e alguns outros colaboradores e alunos que estavam desenvolvendo TCCs correlatos.

Os testes visavam analisar que todas as funcionalidades descritas nos requisitos funcionais fossem implementadas corretamente e operassem conforme esperado. A seguir, é apresentado um resumo dos testes de funcionalidade realizados, organizados por requisito funcional.

Tabela 11 — Testes dos requisitos funcionais da ferramenta.

ID	Requisito	Descrição	Funcionou
RF01	Criação de painel semântico para seleção de paleta de cores	A ferramenta deve permitir que o usuário crie um painel semântico personalizado, que servirá como base para a seleção de cores da paleta a ser utilizada no design da interface.	✔
RF02	Seleção de paleta de cores	A ferramenta deve permitir que o usuário selecione e personalize uma paleta de cores com base em a) painel semântico b) uma imagem escolhida extraindo cores representativas da imagem para uso no design da interface ou c) escolhendo uma cor primária aleatoriamente ou c) uma cor escolhida.	✔
RF03	Seleção de cores harmônicas	A ferramenta deve permitir que o usuário escolha cores que funcionem bem juntas, garantindo uma estética equilibrada e agradável considerando as combinações complementar, análoga e monocromática conforme o círculo cromático	✔
RF04	Verificação de contraste de cores	A ferramenta deve ser capaz de analisar o contraste entre as cores selecionadas na interface do usuário e fornecer feedback sobre a conformidade com os padrões de acessibilidade, como as diretrizes WCAG.	✔
RF05	Definição de tipografia padrão	A ferramenta deve propor a tipografia padrão conforme as diretrizes do Material Design 3.	✔
RF06	Definição de tipografia auxiliar por tema do app	A ferramenta deve permitir a definição da tipografia de um aplicativo com base no tema escolhido, que deve ter a ver com o propósito do aplicativo.	✔

[10] https://vercel.com/

RF07	Seleção de ícones de sistema	A ferramenta deve integrar a biblioteca de ícones acessíveis e sem direitos autorais (Google Material Icons) que possam ser usadas nas interfaces de apps. Além disso, a ferramenta deve permitir a pesquisa e a seleção de ícones apropriados para diferentes finalidades. Deve também permitir parametrizar os ícones em relação ao estilo, preenchimento, e cor (propondo também a cor primária da paleta).	✔
RF08	Seleção de imagens	A ferramenta deve permitir usuários buscarem por imagens com base em palavras-chave e fornecerá opções para recolorir imagens de acordo com a paleta de cores definida para o aplicativo.	✔
RF09	Criação de logotipo para o aplicativo	A ferramenta deve permitir criar um logo para o app usando a tipografia auxiliar e um ícone que pode ser escolhido pelo usuário. Deve usar a paleta de cores para propor automaticamente alternativas de cores para o logo.	✔
RF10	Criação de ícone de produto	A ferramenta deve automaticamente criar um *launcher icon* usando a paleta de cores e o ícone do logotipo.	✔
RF11	Exportação do design visual	A ferramenta deve exportar todas as informações do design visual em formato json ou pdf, quando conveniente, e todas as imagens em formato png, com exceção de ícones que serão exportados em svg.	✔

Fonte: Elaborada pelo autor

4.7. Código-fonte

O código-fonte da ferramenta desenvolvida pode ser encontrado no repositório de códigos da Universidade Federal de Santa Catarina (UFSC) por meio do seguinte endereço: https://codigos.ufsc.br/gqs/designhelper e está disponível sob a licença BSD 3. A licença BSD 3 (Berkeley Software Distribution) é uma licença de software de código aberto que permite a redistribuição e uso em formato de código-fonte e binário, com ou sem modificações, desde que atendidas certas condições, que incluem a necessidade de reconhecer a origem do software, não utilizar os nomes dos contribuidores para promoção de produtos derivados sem permissão, e incluir uma declaração de isenção de garantia.

5. CONCLUSÃO

Este Trabalho de Conclusão de Curso teve como objetivo geral desenvolver uma ferramenta *web* para suportar o design visual de interfaces de usuário de aplicativos móveis, destinada a ser utilizada em cursos de computação na Educação Básica, especialmente no desenvolvimento de aplicativos com o App Inventor. No decorrer do trabalho, foram alcançados diversos objetivos específicos que contribuíram significativamente para a construção desta ferramenta.

Primeiramente, foi sintetizada a fundamentação teórica sobre o design de interfaces de usuário, focando especificamente no App Inventor, o que permitiu compreender as capacidades e limitações da ferramenta, bem como os conceitos fundamentais do design visual aplicáveis no contexto educacional. Essa base teórica foi importante para direcionar o desenvolvimento de uma solução adequada às necessidades dos alunos e professores. Em seguida, foi realizada uma análise detalhada do estado da arte, investigando plugins e ferramentas de design gráfico que oferecem suporte ao design de interfaces de usuário. Esta análise forneceu *insights* valiosos sobre as funcionalidades existentes, as lacunas no mercado e as melhores práticas adotadas por outras ferramentas. Com essas informações, foi possível delinear os requisitos funcionais e não funcionais necessários para a ferramenta, possibilitando que ela fosse inovadora e eficaz. O desenvolvimento da ferramenta *web*, conforme proposto, envolveu a implementação de funcionalidades críticas para o design visual, incluindo a definição de paletas de cores, seleção de tipografia, verificação de contraste, seleção de ícones e imagens, e criação de logotipos e ícones de aplicativos. Esta etapa do projeto foi fundamental para materializar as ideias concebidas durante a fase teórica e de análise, resultando em uma solução prática e utilizável.

Como impacto esperado é não apenas facilitar o processo de design de interfaces de aplicativos móveis para alunos e professores na Educação Básica, mas também promover a aprendizagem de conceitos de design visual de forma prática e intuitiva. Ao disponibilizar a ferramenta de forma gratuita e acessível, espera-se democratizar o acesso a recursos avançados de design, contribuindo para a melhoria da qualidade do ensino de computação e incentivando a criatividade e inovação entre os alunos, tornando a ferramenta valiosa tanto para o ensino de computação quanto para a comunidade de design ao promover um ambiente de aprendizagem mais rico e acessível para todos.

Para futuros trabalhos, sugere-se a expansão das funcionalidades da ferramenta, incluindo uma maior integração entre cada um dos módulos da ferramenta, além disso, incorporar inteligência artificial para oferecer sugestões mais avançadas de design e a adaptação da ferramenta para suportar outros idioma, são caminhos promissores para tornar a solução ainda mais robusta e inclusiva.

REFERÊNCIAS

ADRIYANTO, A. et al. Does Color Matter on Web User Interface Design. CommIT Journal. 11(1). 2017.

AIGA. The Professional Association for Design, 2013. Disponível em: https://www.aiga.org. Acesso em: abril 2023.

ALVES, N. et al. An Item Response Theory Analysis of Algorithms and Programming Concepts in App Inventor Projects. In: Simpósio Brasileiro de Educação em Computação (EDUCOMP), 1. , 2021, On-line.

CIEB. 2019. Currículo de referência em tecnologia e computação. Disponível em: <https://curriculo.cieb.net.br/assets/docs/Curriculo_de_Referencia_em_Tecnologia_e_Computacao.pdf >. Acesso em: abril 2023.

CHARTIER, M. 2020. Linkedin refond son interface et la rend plus complète et intuitive. Disponível em: <https://blog.internet-formation.fr/2020/09/linkedin-refond-son-interface-et-la-rend-plus-complete-et-intuitive>. Acesso em: outubro 2023.

FERREIRA, N. et al. Ensinando design de interface de usuário de aplicativos móveis no ensino fundamental. RBIE - Revista Brasileira de Informática na Educação, vol(no), 2020.

FERREIRA. M. et al. Design visual para interfaces de aplicativos: análise de modelos de referência. Educação Gráfica. 23(1), 2019.

FIGMA. 2023. Disponível em: <https://figma.com>. Acesso em: abril 2023.

FONTLIBRARY. 2023. Disponível em: <https://fontlibrary.dev>. Acesso em: outubro 2023.

GARRETT, J. J. The elements of user experience: user-centered design for the web and beyond. 2nd ed. New Riders, 2011.

GONÇALVES B. Cor Aplicada ao Design Gráfico: Um modelo de núcleo virtual para aprendizagem baseado na resolução de problemas UFSC, Santa Catarina. 2014.

GOOGLE. 2022. Material Design. Disponível em: <https://m2.material.io/>. Acesso em: abril 2023.

GOOGLE. 2023. Google Fonts. Disponível em: <https://fonts.google.com/>. Acesso em: abril 2023.

GOOGLE. 2023b. Material Design. Disponível em: <https://m3.material.io/>. Acesso em: abril 2023.

HADDAWAY, N. R. et al. The role of Google Scholar in evidence reviews and its applicability to grey literature searching. PloS one, 10(9), 2015.

KONICA MINOLTA. 2023. Compreendendo o Espaço de Cor CIE L*C*h. Disponível em: <https://sensing.konicaminolta.us/br/blog/compreendendo-o-espaco-de-cor-cie-lch>. Acesso em: outubro 2023.

NIEBLA, H. 2023. 10 Common UI Design Mistakes (and How to Avoid Them). Disponível em: <https://careerfoundry.com/en/blog/ui-design/common-ui-design-mistakes>. Acesso em: outubro 2023.

MEC. 2022. Resolução n° 1 de 4 de outubro de 2022. 2022. Disponível em: <http://portal.mec.gov.br/docman/outubro-2022-pdf/241671-rceb001-22/file>. Acesso em: abril 2023.

MIT. 2023. Disponível em: <https://appinventor.mit.edu>. Acesso em: abril 2023.

MEDEIROS, G. et al. Práticas pedagógicas com o desenvolvimento de aplicativos para dispositivos móveis por estudantes da Educação Básica. TEXTURA-Revista de Educação e Letras, 22(49), 2020.

MEDEIROS, G. et al. O protagonismo de estudantes da educação básica a partir do desenvolvimento de aplicativos para smartphone. Perspectiva, 39(1), 1-18, 2021.

MOTA, M. J. da; AMENDOLA, M. F. Tipografia e design na construção da linguagem visual da letra. 12º Congresso Brasileiro de Pesquisa e Desenvolvimento em Design, Blucher Design Proceedings, v. 2, p. 777-789. 2016.

MUSTAFARAJ, E.; TURBAK, F.; SVANBERG, M. Identifying original projects in App Inventor. In: Proc. of the Thirtieth International Flairs Conference. Marco Island, FL, Estados Unidos, 2017.

RAJA, R., & NAGASUBRAMANI, P. C. Impact of Modern Technology in Education. Journal of Applied and Advanced Research, 3, 33-35, 2018.

SCHLATTER, T., LEVINSON, D. Visual usability: Principles and practices for designing digital applications. Morgan Kaufmann, 2013.

SHNEIDERMAN, B; PLAISANT, C. 2009. Designing the User Interface: Strategies for Effective Human-Computer Interaction. Addison-Wesley Longman, Boston, MA, USA, 5th edition.

SILVEIRA, L. Introdução à teoria da cor. Curitiba: UTFPR, 2015. 171 p.

SOLECKI, I. et al. 2020. Estado da Prática do Design Visual de Aplicativos Móveis desenvolvidos com App Inventor. Disponível em: <http://milanesa.ime.usp.br/rbie/index.php/rbie/article/view/v28p30>. Acesso em: maio 2024.

PENPOT. 2023. Disponivel em: <https://penpot.app/>. Acesso em: abril 2023.

PETERSEN, K. et al. Systematic mapping studies in software engineering. Proc. of the 12th International Conference on Evaluation and Assessment in Software Engineering, Bari, Italy, p. 68–77, 2008.

PINHEIRO, F. et al. Evolução da ferramenta App Inventor para suportar o ensino de design interface na Educação Básica. In: Workshop de Informática na escola (WIE), 25. , 2019, Brasília.

PREECE, J. 1994. Human-Computer Interaction. Concepts and Design. Ics Series. Addison-Wesley Pub. Co. Disponível em: <https://books.google.com.br/books?id=p99QAAAAMAAJ>.

TECIDOS, M. 2023. Como usar um círculo cromático. Disponível em: <https://blog.maximustecidos.com.br/como-usar-um-circulo-cromatico>. Acesso em: outubro 2023.

XU, C.; PEAK, D.; PRYBUTOK, V. A customer value, satisfaction, and loyalty perspective of mobile application recommendations. Decision Support Systems, v. 79, p. 171-183, 2015.

WASSERMAN, A. Software engineering issues for mobile application development. In: Proc. of the FSE/SDP workshop on Future of software engineering research. Nova York, NY, Estados Unidos, 2010. p. 397-400.

WCAG. 2023. Web Content Accessibility Guidelines (WCAG) 2.1. Disponível em: <https://www.w3.org/TR/WCAG21/>. Acesso em: outubro 2023.

WOLBER, D. et al. App inventor. O'Reilly Media, Inc., 2011.

WORLD ECONOMIC FORUM. The Future of Jobs Report 2020. Disponível em: <https://www3.weforum.org/docs/WEF_Future_of_Jobs_2020.pdf>. Acesso em: abril 2023.

WIKIPÉDIA. 2023. Matiz. Disponível em: <https://pt.wikipedia.org/wiki/Matiz>. Acesso em: outubro 2023.

WIKIPÉDIA. 2023. Saturação. Disponível em: <https://pt.wikipedia.org/wiki/Saturação>. Acesso em: outubro 2023.

WIKIPÉDIA. 2023. Luminosidade (cor). Disponível em: <https://pt.wikipedia.org/wiki/Luminosidade_(cor)>. Acesso em: outubro 2023.

W3C. Web Content Accessibility Guidelines 2.1. 2018. Disponível em: <https://www.w3.org/TR/2018/REC-WCAG21-20180605/>. Acesso em: abril 2023.

APÊNDICE A

Desenvolvimento de uma ferramenta web para suporte ao design visual de interface de usuário de aplicativos móveis

Higor Pires Oliveira, Christiane Gresse von Wangenheim, Jean C. R. Hauck

Dep. de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

higorpiresoliveira@gmail.com, c.wangenheim@ufsc.br, jean.hauck@ufsc.br

Abstract. *The technological advancements of recent years have made computing indispensable in daily life. Teaching computing in K-12 can be enhanced through mobile application development using block-based tools like App Inventor, which supports both functionality programming and interface design. This approach helps teach visual design concepts such as layout, colors, and typography. Currently, App Inventor lacks substantial support for visual design, leading to unappealing interfaces in educational apps. This work aims to create a web tool to support visual interface design for App Inventor in computing education, improving the learning experience in Brazilian schools.*

Resumo. *O desenvolvimento tecnológico tornou a computação essencial na vida cotidiana. O ensino de computação na Educação Básica pode ser aprimorado com o desenvolvimento de aplicativos móveis usando ferramentas baseadas em blocos como o App Inventor, que suporta tanto a programação de funcionalidades quanto o design da interface. Esta abordagem ajuda a ensinar conceitos de design visual como layout, cores e tipografia. Atualmente, o App Inventor carece de suporte significativo para design visual, resultando em interfaces pouco atraentes. Este trabalho visa criar uma ferramenta web para apoiar o design visual de interfaces no App Inventor, melhorando a experiência de aprendizado nas escolas brasileiras.*

1. Introdução

Nos últimos anos, a tecnologia tem desempenhado um papel central na sociedade, influenciando tanto a vida profissional quanto pessoal (RAJA e NAGASUBRAMANI, 2018). Este crescente protagonismo está gerando uma demanda significativa por profissionais capacitados na área de tecnologia, e portanto é crucial capacitá-los. O ensino de computação na Educação Básica tornou-se essencial, especialmente após a inclusão da computação no currículo escolar pelo Ministério da Educação em 2022 (MEC, 2022).

Uma maneira eficaz de ensinar computação nesse nível educacional é através do desenvolvimento de aplicativos móveis utilizando ferramentas de programação visual baseadas em blocos, como o App Inventor (MEDEIROS et al., 2020). O App Inventor facilita

o aprendizado de conceitos de programação e design de interface de usuário, permitindo a criação de aplicativos móveis sem necessidade de conhecimentos avançados de programação (MIT, 2023). Apesar das suas vantagens, muitos aplicativos desenvolvidos na plataforma apresentam interfaces pouco atraentes devido à falta de suporte adequado para o design visual (Solecki et al., 2020), que é crucial para a usabilidade e atratividade dos aplicativos, impactando diretamente a experiência do usuário (SCHLATTER; LEVINSON, 2013).

Este Trabalho de Conclusão de Curso, portanto, visa desenvolver uma ferramenta web que suporte o design visual de interfaces de usuário para aplicativos móveis criados com o App Inventor, voltada para o contexto da Educação Básica. A ferramenta busca auxiliar professores e estudantes na criação de aplicativos visualmente atraentes e coerentes com os princípios de design visual, promovendo uma aprendizagem prática e intuitiva.

O trabalho segue as diretrizes do Departamento de Informática e Estatística (INE – UFSC) para Trabalhos de Conclusão de Curso. A ferramenta focará exclusivamente no suporte ao design visual de aplicativos criados com App Inventor, não abordando outras etapas do processo de design ou implementação, e será voltada para os objetivos de aprendizagem da Educação Básica.

Objetivo Geral: Desenvolver uma ferramenta web para suportar o design visual de interfaces de usuário de aplicativos móveis, a ser utilizada em cursos de computação na Educação Básica com App Inventor.

Objetivos Específicos:

- Sintetizar a fundamentação teórica sobre design de interface de usuário de aplicativos (App Inventor).
- Analisar o estado da arte sobre plugins que oferecem suporte ao design de User Interfaces (UIs) de aplicativos em ferramentas de design gráfico.
- Desenvolver uma ferramenta web para suporte ao design de UIs de aplicativos.

2. Metodologia de pesquisa

A metodologia de pesquisa adotada para este trabalho é dividida em três etapas principais. A primeira etapa, denominada Fundamentação Teórica, envolve o estudo e análise dos principais conceitos e teorias relacionadas aos temas abordados. Nesta fase, são realizadas atividades como a análise teórica sobre o design de interfaces de usuário de aplicativos utilizando o App Inventor e a análise teórica sobre design visual. O objetivo é construir uma base sólida de conhecimento que suporte o desenvolvimento da ferramenta proposta.

Na segunda etapa, é realizado um mapeamento sistemático de plugins que dão suporte ao design de interfaces de usuário. Esse mapeamento segue o processo de revisão proposto por Petersen et al. (2008). As atividades incluem a definição do protocolo de revisão, a execução da busca por plugins relevantes e a extração e análise das informações coletadas. Essa etapa visa identificar as soluções existentes e avaliar suas funcionalidades, servindo como referência para o desenvolvimento da nova ferramenta.

A terceira etapa, Desenvolvimento, foca na criação da ferramenta web seguindo um processo de engenharia de software proposto por Pressman (2016). As atividades nesta fase

incluem a análise de requisitos, a modelagem da arquitetura da ferramenta, a modelagem detalhada e a implementação da solução, e a realização de testes para garantir a funcionalidade e usabilidade do sistema. Esta etapa é crucial para transformar o conhecimento teórico e os insights obtidos na análise do estado da arte em uma solução prática e eficaz para o suporte ao design visual de interfaces de usuário de aplicativos móveis no contexto educacional.

3. Fundamentação teórica e estado da arte

Esta seção aborda os principais conceitos e ferramentas relacionados ao desenvolvimento de uma ferramenta web para suporte ao design visual de interfaces de usuário de aplicativos móveis, utilizando o App Inventor no contexto da Educação Básica. Além disso, apresenta uma revisão sistemática do estado da arte para identificar soluções existentes e avaliar suas funcionalidades.

App Inventor

O App Inventor é uma ferramenta gratuita e de código aberto desenvolvida pelo MIT, que permite a criação de aplicativos móveis para Android através de uma interface de programação visual baseada em blocos (MIT, 2023). Ele possui dois principais elementos para construção de aplicativos: o Designer, que permite a criação da interface gráfica, e o Editor de Blocos, que possibilita a programação do comportamento do aplicativo (WOLBER, 2011). A programação em blocos facilita o aprendizado de conceitos básicos de computação, como sequência, seleção, repetição e abstração, estimulando o pensamento computacional e a criatividade (ALVES et al., 2021).

Design Visual

O design visual é um componente crítico para a usabilidade e atratividade dos aplicativos, afetando diretamente a experiência do usuário (SCHLATTER; LEVINSON, 2013). Entre os principais elementos do design visual, destacam-se o layout, cores, tipografia e imagens. O layout refere-se à organização e disposição dos elementos visuais dentro de uma tela, determinando como esses elementos interagem entre si (SCHLATTER; LEVINSON, 2013). As cores são usadas para transmitir informações, emoções e significados, afetando o humor e a atenção dos usuários. A tipografia envolve a criação e uso de tipos para compor textos visuais, influenciando a legibilidade e a estética do design (MOTA; AMENDOLA, 2016). As imagens complementam e reforçam o conteúdo textual, tornando a interface mais atraente e informativa.

Meta-princípios do Design Visual

Os meta-princípios do design visual incluem consistência, hierarquia e personalidade (SCHLATTER; LEVINSON, 2013). A consistência garante que os elementos visuais sigam padrões estabelecidos, facilitando a navegação e uso do aplicativo. A hierarquia organiza os

elementos em ordem de importância, melhorando a clareza e usabilidade. A personalidade transmite a identidade e propósito do aplicativo, criando uma conexão emocional com os usuários.

Diretrizes de Design de Interface

Diretrizes de design de interface, como o Material Design desenvolvido pelo Google, fornecem orientações específicas para a criação de interfaces de usuário consistentes e acessíveis (GOOGLE, 2022). Essas diretrizes abordam aspectos como cores, tipografia, layout e componentes, ajudando a garantir a usabilidade e a atratividade dos aplicativos.

Revisão Sistemática do Estado da Arte

Para identificar as soluções existentes e avaliar suas funcionalidades, foi realizada uma revisão sistemática do estado da arte seguindo o processo de revisão proposto por Petersen et al. (2008). O mapeamento sistemático incluiu a definição do protocolo de revisão, a execução da busca por plugins relevantes e a extração e análise das informações coletadas.

Os resultados da revisão sistemática revelaram que, apesar das vantagens do App Inventor, muitos aplicativos desenvolvidos na plataforma apresentam interfaces visualmente pouco atraentes. Isso se deve à falta de suporte adequado para o design visual dentro da ferramenta (Solecki et al., 2020). Foram identificados vários plugins que oferecem suporte ao design visual em ferramentas de design gráfico, porém, nenhum deles oferece suporte completo e específico para o contexto educacional do App Inventor.

Conclui-se que há uma necessidade de desenvolver uma ferramenta adicional que ofereça suporte completo ao design visual de interfaces de usuário no App Inventor. Essa ferramenta deve ser alinhada aos princípios e diretrizes de design reconhecidos, como o Material Design, para promover uma aprendizagem mais prática e intuitiva de conceitos de design e computação na Educação Básica.

Em resumo, a fundamentação teórica e a revisão sistemática do estado da arte destacam a importância do design visual no desenvolvimento de aplicativos móveis educacionais, a relevância do App Inventor como ferramenta de ensino de computação na Educação Básica, e a necessidade de uma solução que suporte o design visual de interfaces de usuário, contribuindo para a formação de habilidades em design visual durante o desenvolvimento de aplicativos móveis.

4. Solução desenvolvida: AppDesignFlow

A solução desenvolvida como resultado do Trabalho de Conclusão de Curso é denominada AppDesignFlow, uma ferramenta web projetada para suportar o design visual de interfaces de usuário de aplicativos móveis criados com o App Inventor, especificamente no contexto educacional da Educação Básica. O objetivo do AppDesignFlow é proporcionar um ambiente intuitivo e prático que auxilie professores e estudantes na criação de interfaces mais atraentes e coerentes com os princípios de design visual.

4.1. Análise de Requisitos

A análise de requisitos foi conduzida para identificar as necessidades dos usuários e as funcionalidades essenciais da ferramenta. Foram definidos os seguintes requisitos funcionais:

- Criação de Paletas de Cores: Permitir a criação e personalização de paletas de cores harmônicas, facilitando a escolha de combinações de cores adequadas para a interface.
- Seleção de Tipografia: Oferecer uma biblioteca de fontes tipográficas e orientações sobre a melhor utilização de tipografias serif e sans-serif, adequadas para diferentes contextos e finalidades.
- Contraste: Fornecer ferramentas para análise de contraste, garantindo que as interfaces sejam legíveis a todos os usuários, seguindo diretrizes como as WCAG 2.1.
- Criação de logotipos e ícones de produto: Facilitar a criação de logotipos e íconmes de produto do aplicativo, permitindo a crição de materiais de identificação do app desenvolvido de maneira intuitiva e rápida.
- Incorporação de Imagens e Ícones: Suporte à inclusão de imagens e ícones, com orientações sobre a escolha e uso consistente desses elementos visuais.

4.2. Design Conceitual da Ferramenta

O design conceitual do AppDesignFlow foi baseado nas necessidades identificadas durante a análise de requisitos e nas melhores práticas de design visual. A ferramenta foi desenvolvida para ser acessível via navegador web, garantindo compatibilidade com diferentes dispositivos e sistemas operacionais.

O layout da ferramenta foi projetado para ser simples e intuitivo, com uma interface de usuário que guia o usuário através do processo de design. A tela inicial apresenta opções para criar novas paletas de cores, selecionar tipografias e analisar o contraste de cores.

4.3. Prototipação

Durante a fase de prototipação, foram criados modelos de interface que ilustram como os usuários interagem com a ferramenta. Os protótipos foram validados através de testes com usuários reais, garantindo que a interface fosse intuitiva e atendesse às necessidades dos estudantes e professores.

4.4. Implementação

A implementação do AppDesignFlow seguiu um processo iterativo, onde cada funcionalidade foi desenvolvida e testada individualmente antes de ser integrada à ferramenta completa. A arquitetura do sistema foi projetada para ser modular, permitindo a adição de novas funcionalidades no futuro.

A ferramenta foi desenvolvida utilizando tecnologias web modernas, como React e Next.js, garantindo uma experiência de usuário fluida e responsiva.

4.5. Testes do Sistema

Os testes do sistema foram realizados de maneira incremental em cada um dos módulos desenvolvidos e envolveram a participação de outros alunos que desenvolveram trabalhos correlatos e da professora orientadora do trabalho. O feedback dos usuários foi coletado e utilizado para realizar ajustes e melhorias na ferramenta.

Em resumo, o AppDesignFlow é uma ferramenta web desenvolvida para suportar o design visual de interfaces de usuário de aplicativos móveis criados com o App Inventor, proporcionando uma aprendizagem prática e intuitiva de conceitos de design e computação na Educação Básica. A ferramenta atende às necessidades identificadas, oferecendo suporte completo ao processo de design visual e contribuindo para a formação de habilidades em design visual entre estudantes.

5. Conclusão

O desenvolvimento da ferramenta AppDesignFlow representa um avanço significativo no suporte ao design visual de interfaces de usuário de aplicativos móveis, especialmente no contexto da Educação Básica. Este Trabalho de Conclusão de Curso teve como objetivo abordar a lacuna existente no App Inventor, que, apesar de sua facilidade de uso e ampla adoção educacional, apresenta limitações no suporte ao design visual.

Por meio da análise de requisitos, design conceitual, prototipação, implementação e testes rigorosos, o AppDesignFlow foi desenvolvido para oferecer funcionalidades essenciais que auxiliam professores e estudantes na criação de interfaces de usuário mais atraentes e coerentes. Entre essas funcionalidades estão a criação de paletas de cores harmônicas, a seleção de tipografias adequadas, ferramentas para análise de contraste e acessibilidade, e suporte à prototipação de layouts e incorporação de imagens e ícones.

A revisão sistemática do estado da arte realizada como parte deste trabalho destacou a necessidade de soluções específicas para o contexto educacional do App Inventor. O AppDesignFlow, ao preencher essa lacuna, oferece uma plataforma prática e intuitiva que facilita o ensino e aprendizado de design visual e computação.

Em termos de impacto educacional, a ferramenta espera-se que a ferramenta promove uma aprendizagem mais prática e intuitiva de conceitos de design e computação, contribuindo para a formação de habilidades críticas no século XXI. Ao fornecer suporte ao processo de design visual, o AppDesignFlow ajuda a preparar estudantes para um futuro onde a competência tecnológica e o design são cada vez mais valorizados.

Futuras melhorias para o AppDesignFlow podem incluir a expansão de suas funcionalidades, integração entre módulos, e o suporte para outros idiomas, como o inglês.

Em conclusão, o AppDesignFlow não apenas atende às necessidades identificadas durante este trabalho, mas também estabelece uma base sólida para futuras inovações no suporte ao design visual no contexto educacional, promovendo uma experiência de aprendizagem enriquecida e eficiente para estudantes e professores na área de desenvolvimento de aplicativos móveis.

Referências

ALVES, N. et al. An Item Response Theory Analysis of Algorithms and Programming Concepts in App Inventor Projects. In: Simpósio Brasileiro de Educação em Computação (EDUCOMP), 1. , 2021, On-line.

Google. (2022). Material Design. Acessado em 2023, de https://material.io/design.

MEC. 2022. Resolução n° 1 de 4 de outubro de 2022. 2022. Disponível em: <http://portal.mec.gov.br/docman/outubro-2022-pdf/241671-rceb001-22/file>. Acesso em: abril 2023.

MEDEIROS, G. et al. Práticas pedagógicas com o desenvolvimento de aplicativos para dispositivos móveis por estudantes da Educação Básica. TEXTURA-Revista de Educação e Letras, 22(49), 2020.

MIT. (2023). MIT App Inventor. Massachusetts Institute of Technology. Disponível em http://appinventor.mit.edu.

MOTA, M. J. da; AMENDOLA, M. F. Tipografia e design na construção da linguagem visual da letra. 12º Congresso Brasileiro de Pesquisa e Desenvolvimento em Design, Blucher Design Proceedings, v. 2, p. 777-789. 2016.

PETERSEN, K. et al. Systematic mapping studies in software engineering. Proc. of the 12th International Conference on Evaluation and Assessment in Software Engineering, Bari, Italy, p. 68–77, 2008.

RAJA, R., & NAGASUBRAMANI, P. C. Impact of Modern Technology in Education. Journal of Applied and Advanced Research, 3, 33-35, 2018.

SCHLATTER, T., LEVINSON, D. Visual usability: Principles and practices for designing digital applications. Morgan Kaufmann, 2013.

SOLECKI, I. et al. 2020. Estado da Prática do Design Visual de Aplicativos Móveis desenvolvidos com App Inventor. Disponível em: <http://milanesa.ime.usp.br/rbie/index.php/rbie/article/view/v28p30>. Acesso em: maio 2024.

WOLBER, D. et al. App inventor. O'Reilly Media, Inc., 2011.