UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Daniel Melo da Silva

SUPORTE A UNIDADE INSTRUCIONAL DE DESENVOLVIMENTO DE APLICATIVOS COM TÉCNICAS DE UX DESIGN PARA O ENSINO BÁSICO

Florianópolis

2017/1

SUPORTE A UNIDADE INSTRUCIONAL DE DESENVOLVIMENTO DE APLICATIVOS COM TÉCNICAS DE UX DESIGN PARA O ENSINO BÁSICO

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.

Orientador: Prof. Dr. rer. nat. Christiane A. Gresse von Wangenheim, PMP.

Florianópolis
2017/1

Daniel Melo da Silva

SUPORTE A UNIDADE INSTRUCIONAL DE DESENVOLVIMENTO DE APLICATIVOS COM TÉCNICAS DE UX DESIGN PARA O ENSINO BÁSICO

Trabalho de conclusão de curso submetido ao Departamento de Informática e Estatística da Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharelado em Sistemas de Informação.

Florianópolis, __ de _____ de 2017.

Orientadora:

_____

Prof.ª Dr.ª rer. nat. Christiane Gresse von Wangenheim, PMP
Orientador
Universidade Federal de Santa Catarina

Banca Examinadora:

_____

Prof. Dr. Jean Carlo R. Hauck
Avaliador
Universidade Federal de Santa Catarina

_____

Prof. ª Giselle Araújo e Silva de Medeiros

Avaliador
Universidade Federal de Santa Catarina

_____

Suleica Fernanda Biesdorf Kretzer

Avaliador
Universidade Federal de Santa Catarina

AGRADECIMENTOS

Agradeço antes de mais nada aos meus pais, pela paciência e amor incondicional. Vocês são e sempre serão meu porto seguro e quem me impulsiona ao impossível.

Agradeço a professora Christiane Gresse von Wangenheim pelas críticas quando necessárias e pelos elogios quando merecidos. Obrigado pela orientação e por todo suporte durante o desenvolvimento deste trabalho.

Agradeço também, a todos os professores da UFSC que de alguma forma me inspiraram a ser o profissional que sou hoje.

Agradeço a empresa Involves Tecnologia por todo o apoio e entendimento durante essa jornada.

Sem vocês eu não estaria aqui.

“Não sabendo que era impossível, ele foi lá e fez.”

(Jean Cocteau)

RESUMO

Com a crescente evolução tecnológica é evidente que para que o indivíduo se mantenha competitivo no mercado de trabalho novas competências são necessárias. Competências atreladas a tecnologia da informação, como pensamento computacional e *design thinking* são desejáveis e podem ser aplicadas nos mais variados campos de atuação. Essas competências podem ser introduzidas no contexto educacional no nível de educação básica (fundamental I e II) por meio de unidades instrucionais que ensinam programação de apps para dispositivos móveis utilizando a ferramenta App Inventor. Para tal é importante a utilização do guia de estilo Material Design para o *design* das interfaces gráficas dos apps criados, dado sua vasta utilização em apps modernos e seu alto grau de usabilidade além do seu alinhamento com a teoria das cores. Porém atualmente o App Inventor não oferece uma escolha fácil de cores compatíveis ao Material Design. Portanto o presente trabalho tem como objetivo desenvolver melhorias na ferramenta *App Inventor,* especificamente na paleta de cores, para que seja possível a criação de componentes visuais de aplicativos Android seguindo o Material Design. Foi feita uma análise do App Inventor e o *design* de cores levantando o estado da arte de ensino de *design* gráfico a nível de ensino básico, identificando requisitos funcionais referentes ao *design* de cores. Os requisitos então foram implementados e testados na ferramenta App Inventor, gerando uma versão customizada da ferramenta.

Espera-se que com esta evolução seja possível desenvolver apps com melhor usabilidade em termos de *design* de cores, criando apps com maior atratividade.

Palavras-chave: *Design* de cores, pensamento computacional, App Inventor, ensino básico, programação.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

AI – App Inventor 2

ACM – *Association for Computing Machinery*

API - *Application Program Interface*

CnE – Computação na Escola

GWT – *Google Web Toolkit*

KSA - K*nowledge Skills and Abilities*

JSON - *JavaScript Object Notation*

MIT - *Massachusetts Institute of Tecnology*

SO - Sistemas Operacional

UI – Unidade Instrucional

UX – *User Experience*

DT – *Design Thinking*

NPE - *Novice Programming Enviroment*

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

O cotidiano atual da sociedade encontra-se fortemente atrelado a tecnologia de informação e comunicações. A evolução oriunda da telefonia móvel, mudou a maneira como as pessoas se comunicam e interagem, transformando o estilo de vida dos indivíduos. O ritmo da crescente tecnológica impulsionou a competitividade industrial, e acelerou a popularização dos aparelhos móveis. Em 2015 foram estimados que existiam 7,1 bilhões de celulares ativos no mundo todo (Convergência Digital, 2015). No Brasil, estamos acima da média global em relação ao número de telefones per capita, atingindo 1,2 aparelhos por habitante (FGV, 2016). Tendo uma predominância de aparelhos do tipo *smartphone*. No final de 2015, esses aparelhos representavam 92% do mercado de celulares no país (IDC, 2016). A competividade dos sistemas operacionais de celulares atualmente é considerada bi polarizada, tendo de um lado o sistema operacional Android, no outro o IOS e outras minorias. Atualmente o sistema predominante é o Android. No primeiro bimestre de 2016, o *Market Share* do Android estava em 84% (GARTNER, 2016). O Android é *open-source* e seus muitos aplicativos estão disponíveis para *download* na loja oficial, *Google Play*[1].

Tendo em vista a expressividade dos celulares, principalmente com sistema operacional Android na sociedade atual, e dada as características dos

[1] https://play.google.com/store?hl=en

jovens na faixa de 10-14 anos, destaca-se esse grupo como o foco desse trabalho. Indivíduos dessa faixa são considerados a geração Y, ou *Millennial*. Estão abundantemente conectados socialmente por meio de redes virtuais (CROW, 2010). Estão fortemente ligados a seus aparelhos celulares, relacionam o celular como o item pessoal mais importante, antes mesmo do que chaves e carteira (HOLLEY, 2008). E a grande maioria desses jovens incluindo no Brasil, já são usuários de *smartphones* (IBGE, 2013). Os jovens do público alvo desse trabalho, estão passando grande parte de seu tempo em "ambientes digitais". Estes jovens, preferem aprender interagindo, se sentem mais confortáveis aprendendo em um ambiente onde a penalidade pela tentativa/erro seja baixa (CROW, 2010). Quase nunca leem manuais e preferem o *e-learning* sob demanda como meio de aprendizado (CROW, 2010). São considerados altamente alfabetizados digitalmente (IT *literacy*), ou seja, sabem usar TI. Apesar do termo remeter a alfabetização, o mesmo não se limita ao ensino formal de utilização de TI. Sendo assim, qualquer individuo é considerado alfabetizado digitalmente, quando não necessita de instruções ou ajuda para o manuseio de tecnologia, seja o conhecimento necessário para a execução da atividade, tácito ou não.

Porém, hoje é importante para qualquer cidadão, especialmente os jovens dessa faixa etária, possuir conhecimentos mais avançados que vão além da alfabetização digital. Possuir conhecimento de computação (IT *fluency*) atende além da necessidade pessoal, a necessidade do mercado atual para com futuros profissionais (CSTA, 2016). Entende-se computação como a área que abrange atividades que incluem a criação de *softwares* e *hardwares* com os mais variados propósitos computacionais (ACM, 2005). Também são

caracterizadas como parte da computação atividades relacionadas ao processamento, estruturação e manipulação dos mais variados tipos de informações, usando como ferramenta um computador (ACM, 2005).

Aprender conceitos de computação, e o fazer de forma colaborativa com outras pessoas, irá ajuda-los a entender melhor como o ecossistema da tecnologia da computação funciona (CSTA, 2016). Portanto deve-se incluir atividades visando o ensino de conceitos de computação o quanto antes na vida acadêmica dos jovens. O currículo da CSTA (CSTA, 2016), possui diretrizes curriculares que visam diminuir a lacuna presente nos sistemas de ensinos atuais e auxiliam no ensino da computação para os alunos do público alvo do presente trabalho. Essas diretrizes abrangem áreas como a resolução de problemas usando pensamento computacional, trabalho em equipe, conceitos de programação e o entendimento do impacto social que a computação exerce no mundo (CSTA, 2016), elevando o nível de conhecimento de computação adquirido pelos jovens do ensino básico.

Existem diferentes iniciativas (TECHNOVATION, 2016) (Code.org, 2017) (Computação na escola, 2017) ensinando programação como parte do ensino de computação para crianças do ensino básico. Dado que grande parte dos jovens são fortemente ligados a tecnologias móveis dos celulares, utilizar esse interesse para ensinar programação por meio do desenvolvimento de aplicativos é uma alternativa de ensino viável (SILVA, 2015). Um dos ambientes de programação mais utilizado para esse fim é o App Inventor, dado que sua utilização é fácil e sua interface amigável (SILVA, 2015). Por meio de seus componentes, é possível acessar inclusive os recursos de hardware dos celulares (câmera, sensores, GPS, teclado, etc.) e incluir funcionalidades que

os utilizem. Sendo assim, o App Inventor é considerado um bom ambiente de programação para a aplicação de unidades instrucionais (UI) voltadas ao ensino de programação por meio de aplicativos.

Os *Novice Programming Enviroments* (NPEs), são ambientes de programação concebidos para tornar acessível o ensino de conceitos introdutórios de programação (KELLEHER, 2005). Uma das formas utilizadas por NPEs para a construção de softwares é a utilização de programação visual. Na programação visual o usuário não precisa se preocupar com sintaxes e conceitos complexos inerentes a linguagens de programação (VALLARTA, 2007). Interagindo com objetos visuais na tela, é possível definir a aparência dos aplicativos, e por meio de movimentos de arraste-solte, juntando blocos lógicos, é possível programar o comportamento esperado do *software* em desenvolvimento (VALLARTA, 2007). Alguns exemplos de ferramentas que utilizam de programação visual são *Scratch*[2], *Snap*[3] e *App Inventor*[4].

Mesmo existindo diversos tutoriais ensinando programação com app inventor (App Inventor Tutorial, 2016) (Pura Vida Apps, 2016) (Imagnity, 2016), além de oficinas (TRILHA, 2016) que ensinam não só a fazer aplicativos como também jogos disponíveis na internet, são poucas as unidades instrucionais que ensinam também conceitos de usabilidade e experiência de usuário (UX), que são fatores importantes para que um aplicativo seja um sucesso (MOORE, 2016). A usabilidade segundo Nielsen (2007),

> "A usabilidade é um atributo de qualidade relacionado à facilidade do uso de algo. Mais especificamente, refere-se à rapidez com que os

[2] https://scratch.mit.edu/
[3] http://snap.berkeley.edu/
[4] http://appinventor.mit.edu/explore/

> usuários podem aprender a usar alguma coisa, a eficiência deles ao usá-la, o quanto lembram daquilo, seu grau de propensão a erros e o quanto gostam de utiliza-la. ”

Atualmente se usa abordagens de UX para melhorar a interação do usuário com o aplicativo, utilizando de técnicas como ciclos de entrevistas com público alvo, elaboração de protótipos, testes monitorados e obtenção de *feedbacks* contínuos dos usuários, espera-se melhorar a usabilidade da interação e por consequência aumentar a chance de se desenvolver um aplicativo fácil de ser usado (HASSENZAHL, 2008).

Dentro deste contexto, a iniciativa Computação na Escola (CnE) está desenvolvendo uma unidade instrucional que visa o ensino de ‘fazer seu próprio app’. Essa UI engloba o ensino de conceitos de programação e processos de UX. Ela utiliza a ferramenta App Inventor como a NPE para a criação dos aplicativos. Mesmo sendo uma ferramenta completa, observou-se oportunidade de melhoria em relação ao suporte para o processo de UX. Por exemplo as funcionalidades para o *design* de interface não possui recursos compatíveis com as definições atuais para aparências de interfaces de aplicativos Android, segundo o *Material Design* (Material Design, 2017).O *Material Design* é uma linguagem visual desenvolvida pelo Google para sintetizar os princípios clássicos de um bom *design* de interface e proporcionar uma experiência de uso uniforme independente de dispositivos móveis e seus diferentes tamanhos.

Dado este cenário o presente trabalho propõe o desenvolvimento de melhorias na ferramenta App Inventor, afim de adicionar funcionalidades compatíveis com conceitos do *Material Design* no desenvolvimento de

interfaces de aplicativos. Servindo desta forma como suporte à aplicação da unidade instrucional desenvolvida pela iniciativa Computação na Escola.

1.2 Objetivos

Objetivo geral. O objetivo do trabalho é desenvolver melhorias na ferramenta *App Inventor,* especificamente na paleta de cores, para que seja possível a criação de componentes visuais de aplicativos Android seguindo o *Material Design* em ordem a buscar maior atratividade na aprendizagem de estudantes do ensino básico.

Objetivos específicos. Os objetivos específicos deste trabalho são:

O1. Sintetizar a fundamentação teórica de aprendizagem e ensino no ensino básico, referente ao desenvolvimento de aplicativos móveis com a ferramenta App Inventor integrando *user experience* (UX) e design de cores.

O2. Analisar estado da arte e prática de unidades instrucionais existentes focadas no ensino de desenvolvimento de apps para alunos do ensino básico integrando *user experience*.

O3. Projetar e implementar melhorias em funcionalidades do App Inventor para servir de suporte a aplicação de uma unidade instrucional focada em ensino de programação e *user experience* utilizando a ferramenta App Inventor, tendo como público alvo alunos do ensino básico.

1.2.1 Limitações do Trabalho

As melhorias propostas e desenvolvidas pelo presente trabalho, são exclusivamente nas cores padrões para os componentes visuais do App Inventor. Sendo assim só serão modificados e/ou adicionados, componentes que estejam relacionados a *palette* de cores. A definição das cores é alinhado ao Material design não será considerado outro guia de estilo como guia para as cores..

1.3 Metodologia de pesquisa

A metodologia de pesquisa do trabalho é composta pelas seguintes etapas:

Etapa 1 – Fundamentação Teórica: sintetizar a fundamentação teórica de aprendizagem de conhecimento na área do ensino básico, assim como o desenvolvimento de aplicativos para dispositivos móveis com App Inventor, que apliquem processos de UX. São realizadas as seguintes atividades:

- A1 - Analisar a teoria de aprendizagem e ensino para estudantes do ensino básico.
- A2 - Analisar o suporte/funcionalidade do App Inventor
- A3 - Analisar conceitos relacionados a UX e usabilidade.

Etapa 2 – Levantar o estado da arte e prática de unidades instrucionais existentes que ensinam computação por meio do App Inventor. Realizar

revisão sistemática da literatura por meio de uma revisão sistemática (KITCHENHAM, 2004). Esta etapa inclui as atividades:

- A2.1 – Definir o protocolo da revisão
- A2.2 – Executar a busca
- A2.3 – Extrair informações e analisar os resultados

Etapa 3 – Levantar de Requisitos e Implementar Melhorias - etapa voltada para levantamentos de requisitos funcionais e casos de uso para implementação de melhorias relacionadas ao *palette* de cores do App Inventor, bem como a implementação das melhorias elencadas. Por meio de casos de uso identificar pontos de melhoria na ferramenta, e subsequente, desenvolver as melhorias. Ao final desta etapa deve-se possuir uma versão estável do App Inventor, com as customizações implementadas e prontas para uso. Atividades da etapa são:

- A3.1 - Analisar pontos de melhoria na ferramenta e casos de uso
- A3.2 - Projetar implementação e testes necessários
- A3.3 - Modificar código fonte e apresentar versão estável

1.4 Estrutura do documento

O capítulo 2 apresenta conceitos que suportam a fundamentação para o desenvolvimento do suporte a unidade instrucional. No capítulo 3 é apresentado o estado da arte referente a unidades instrucionais que ensinam conceitos de computação, juntamente com conceitos de UX por meio do desenvolvimento de aplicativos, usando a ferramenta App Inventor.

Subsequente o capítulo 4 contém o levantamento de casos de uso e funcionalidades que necessitam de alterações, bem como a proposta de implementação em alto nível. No capítulo 5 é apresentado a conclusão do trabalho, bem como os resultados e impactos obtidos, além de elencar trabalhos futuros relacionados.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Aprendizagem e ensino

A **aprendizagem** é uma transformação comportamental e/ou intelectual do indivíduo tida como resultado da experiência externa ou interna do mesmo diante de algum estimulo provocado em seu ambiente (LIBÂNEO, 2009). A exposição a novas experiências, seja elas formais ou informais, fomenta o aumento de conhecimento, habilidades e atitudes (KSA[5]) que culminam na competência do indivíduo em realizar alguma tarefa relacionada a algum domínio especifico (AL-KHALIFA, 2013). Uma das formas de se aprender tais competências é por intermédio do ensino.

Ensino é a forma de se institucionalizar o aprendizado, por intermédio de um sequenciamento intencional de atividades (SMOLKA;GÓES, 1995). O mediador destas atividades e o ensino é o instrutor, em contra partida, o aprendiz é o indivíduo afetado pelas atividades. Sendo assim, essa relação é caracterizada como uma relação sujeito-sujeito-objeto. Ou seja, um sujeito (Instrutor) estabelece relações com a competência e o sujeito (Aprendiz) (SMOLKA;GÓES, 1995). Para se alcançar o ensino, utiliza-se instruções, que

[5] http://www.shsu.edu/dept/human-resources/managerstoolkit/KSA.html

são atividades e/ou experiências desenvolvidas com o intuito de se adquirir novas competências. É importante que o plano de ensino seja desenvolvido de forma sistemática para que o aprendiz não perca o interesse no objetivo de aprendizagem (SMITH;RAGAN, 1999).

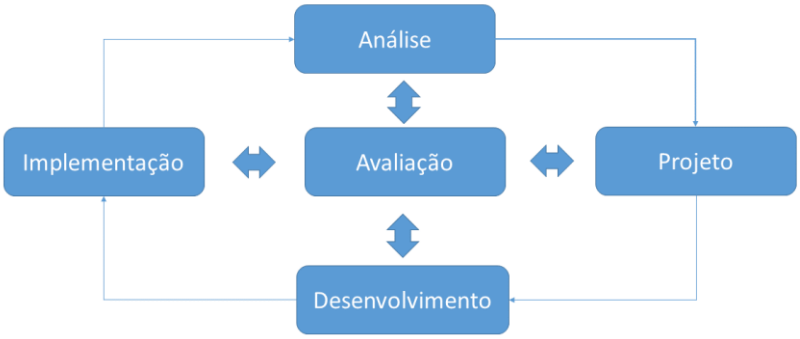
O conjunto de instruções é caracterizado como uma Unidade Instrucional. Qualquer tipo de interação estruturada e planejada, que possua objetivos específicos e bem definidos, pode ser caracterizada como uma unidade instrucional. Podendo assim ser uma aula, oficina, até mesmo uma disciplina, ou curso completo.

Para se assegurar que os objetivos de aprendizagem da unidade serão atingidos e o aprendiz alcance as competências ensinadas, a unidade deve ser criada utilizando *design* instrucional. *Design* instrucional caracteriza-se como:

> "A ação intencional e sistemática de ensino, que envolve o planejamento, o desenvolvimento e a aplicação de métodos, técnicas, atividades, materiais, eventos e produtos educacionais em situações didáticas específicas, a fim de promover, a partir dos princípios de aprendizagem e instrução conhecidos a aprendizagem humana" (FILATRO,2008).

É um processo focado em objetivos e performance do aprendiz. Existem diversos modelos de design instrucional, que indicam como o *design* instrucional deve ser executado. O modelo mais conhecido e adotado atualmente é o ADDIE, acrônimo para *Analyze, Design, Develop, Implement and Evaluate* (BRANCH, 2009).

Figura 1 - Visão geral do processo do Modelo Addie



Fonte: BRANCH, 2009

O modelo descreve o processo do *design* instrucional para o desenvolvimento de unidades instrucionais. O processo ADDIE é composto de cinco fases (BRANCH, 2009):

A **análise** constitui a primeira fase e foca em analisar o contexto como um todo, incluindo os aprendizes e o ambiente onde a unidade será aplicada. Observa-se quais recursos didáticos disponíveis (quadro negro, projetor, computadores, etc.) para a aplicação da UI, além das características físicas do ambiente propriamente dita, ou seja, se a unidade será aplicada em uma sala de aula, biblioteca, ou espaço aberto. Também verificam-se questões como posicionamento dos aprendizes, se sentarão em grupos, ou individualmente, etc. Nessa fase também são analisadas as diretrizes de currículo, alinhadas com as características pessoais e o nível de competência atual dos aprendizes. E a partir disso são identificadas as principais ameaças para que o aprendizado

não ocorra durante a execução da unidade, afim de que possam ser prevenidas. Como resultado dessa etapa, um sumário de análise é criado.

A segunda etapa de **Projeto**, constitui a definição do conteúdo (ementa) e estratégia instrucional correspondente a UI. Nessa etapa são definidos os objetivos de aprendizagem da UI com base na análise do contexto. Um objetivo de aprendizagem é uma descrição do que o aprendiz poderá realizar com as competências adquiridas na unidade. Uma forma de classificar um objetivo de aprendizagem é utilizando a taxonomia de Bloom (BLOOM,1956). Ela organiza o aprendizado em domínios distintos para identificar o nível de aprendizagem. Competências a serem tipicamente aprendidas podem ser classificadas nos domínios de: Cognitivo (conhecimento), Afetivo (atitudes) e Psicomotor (habilidades) conforme detalhado nas Tabelas 1 ,2 e 3 respectivamente.

Tabela 1 - Níveis do domínio cognitivo por BLOOM (1956)

	Nível	Descrição
Cognitivo	1. Conhecimento	Habilidade de lembrar informações e conteúdos previamente abordados. Objetivo principal dessa categoria é trazer a consciência esses conhecimentos
	2. Compreensão	Habilidade de compreender e dar significado ao conteúdo, por meio de tradução e interpretação de fatos e ideias.
	3. Aplicação	Aplicar as informações, métodos e conteúdos aprendidos em novas situações concretas
	4. Análise	Subdividir o conteúdo em partes menores para se facilitar o entendimento da estrutura final, e os relacionamentos entre as partes
	5. Síntese	Agregar e combinar partes para se criar um novo todo. Isso envolve compilar elementos separados e organiza-los de forma a gerar uma nova estrutura
	6. Avaliação	Julgar o valor do material para um propósito especifico. Critérios de julgamento bem definidos, podendo ser externos (relevância), e internos (organização)

Tabela 2 - Níveis do domínio afetivo por BLOOM (1956)

	Nível	Descrição
Afetivo	1. Receptividade	Trata-se da receptividade do aluno para receber novos estímulos, e sua atenção a eles
	2. Resposta	Momento em que o aprendiz age de forma ativa, reagindo primeiramente a seus próprios estímulos (motivação)
	3. Valorização	Quando o aprendiz atribui valor para objetivos, fenômenos ou informações
	4. Organização	O aprendiz organiza diferentes valores em sua própria hierarquia, criando um sistema único e pessoal
	5. Caracterização	Valores sedimentados no individuo, influenciando em seu comportamento

Tabela 3 - Categorias do domínio psicomotor de SIMPSON (1972)

	Nível	Descrição
Psicomotor	1. Percepção	Detectar sinais sensoriais, para iniciar o estimulo ao ato motor e orientar as atividades
	2. Prontidão	Estar familiarizado com a sequência de etapas de resposta em um processo de comunicação
	3. Resposta Guiada	Etapa inicial de aprendizagem de uma habilidade mais complexa. Visualizando atitudes de um indivíduo, o outro atinge o desempenho desejado utilizando de processos de imitação, e tentativa e erro
	4. Mecanismo	Etapa intermediária no aprendizado de uma habilidade. Respostas se tornam atitudes familiares, e os movimentos são executados com alguma confiança e proficiência
	5. Resposta Complexa	Desempenho automático e ação reacionária sem hesitação, sobre padrões complexos
	6. Adaptação	Individuo atingiu maturidade suficiente para modificar os padrões de movimentos para atender a suas necessidades individuais
	7. Originalidade	Cria-se novos padrões, devido a necessidade do individuo

Nesta fase também se define a estratégia instrucional para ter o sequenciamento correto das instruções, organizando-as de forma estruturada de forma a auxiliarem no atingimento dos objetivos de aprendizagem da UI. (MAZZIONI, 2009). Pode-se dizer que a definição da estratégia instrucional depende, inicialmente do conhecimento pedagógico do professor (PIMENTA;ANASTASIOU, 2002) além de considerar os objetivos que o instrutor estabelece e quais as competências serão abordadas (MAZZIONI, 2009).

Para que o processo de aprendizagem ocorra de forma eficaz faz-se necessário engajar o aprendiz do começo ao fim do processo. Mantendo seu

interesse para que ocorra a transmissão de conhecimento. Para tal, existem estratégias de ensino utilizadas pelo instrutor na articulação do processo de ensino, de acordo com cada atividade programada na UI (ANASTASIOU;ALVES, 2004). Existem diferentes categorias de estratégias instrucionais, alguns exemplos são apresentados na Tabela 4.

Tabela 4 - Categorias de estratégias instrucionais

Estratégia	**Descrição**	**Exemplo**
Instrução Direta	Normalmente usada, onde o instrutor é o centro da unidade instrucional, executando a unidade de forma expositiva, com o instrutor explicando o conteúdo.	Palestras.
Instrução Indireta	O aprendiz é o centro da UI nessa estratégia, tendo o Instrutor o papel apenas de facilitador. Aprendiz infere o novo conhecimento por meio de investigações, formação de hipóteses.	Resolução de problemas.
Instrução Interativa	Nessa metodologia a formação de conhecimento se dá de forma colaborativa, diante de discussões de grupo e compartilhamento de ideias. A efetividade dessa estratégia encontra-se diretamente proporcional a competência do instrutor de conduzir dinâmicas em grupo.	Projetos em grupo.
Aprendizagem Experiencial	Metodologia focada em atividades pré-definidas. Nessa modalidade a ênfase se encontra na aprendizagem experiencial, e não no seu produto. Possui um ciclo de 5 fases: experienciar, compartilhar, analisar , inferir e aplicar.	conduzir uma pesquisa de opinião pública.
Estudo Independente	Focada no aprendizado de forma autodidata, onde o aprendiz desenvolve o interesse e tem atitudes que o orientam gradativamente ao aprendizado. Apesar do foco individual, pode se ter a supervisão de um instrutor.	*e-learning* de alguma habilidade

Fonte: SASKATCHEWAN EDUCATION, 1991)

Ao final da etapa de projeto, documenta-se o plano de ensino, identificando quais serão os objetivos de aprendizados da UI, bem como o seu

sequenciamento lógico de atividades e seus conteúdos, além das estratégias instrucionais que serão utilizadas para que ocorra o ensino, e os métodos de avaliação que serão utilizados para a verificação se o objetivo de aprendizagem foi atingido.

O **desenvolvimento** da unidade instrucional compreende a fase que produzirá todos os recursos visando operacionalizar a unidade conforme o plano de ensino definido na etapa anterior. Para tal, são selecionados ou criados os materiais que darão suporte as atividades propostas pela unidade instrucional. Podendo estes ser de diversos formatos físicos (*slideshow*, transparências, cartazes, etc), ou formatos digitais, como é o caso de *softwares* de desenvolvimento. Caso seja observado que existe a possibilidade de um dos materiais de suporte limitar ou ameaçar o alcance dos objetivos da UI, os mesmo podem ser modificados e melhorados afim de que ofereçam um melhor suporte na aplicação da UI. Após a criação do material instrucional, o mesmo passa por um tipo de validação, seja sob forma de revisão, ou por meio da aplicação de testes pilotos. É criado como resultado desta etapa todo o material instrucional da unidade instrucional.

Na fase de **implementação** é executada a Unidade Instrucional na prática. Isso envolve a preparação do ambiente e dos instrutores e a aplicação da unidade conforme previsto no plano de ensino utilizando os materiais desenvolvidos. Realiza-se a avaliação definida na etapa de projeto validando assim se os objetivos de aprendizagem da UI foram atingidos. Também são coletados dados para sistematicamente avaliar a própria unidade instrucional.

A **avaliação** é voltada para a análise da unidade instrucional com o objetivo de identificar pontos de melhoria e evoluções na UI, visando a melhoria

continua da UI. Faz-se necessária uma definição sistemática dos objetivos de avaliação, decompondo em perguntas de análise e métricas, e projetar instrumentos de coleta/análise de dados em paralelo a aplicação da unidade. A avaliação tipicamente é executada por meio de estudos empíricos durante a implementação da unidade instrucional. Analisando os dados coletados durante a implementação apresenta-se uma especificação de melhorias para próximas interações, caso o escopo seja o mesmo.

Por meio do desenvolvimento de unidades instrucionais é possível sistematizar o ensino de competências especificas. Focando na melhoria continua do aprendiz, as atividades são estruturadas de forma a aumentar a complexidade, conforme a evolução das mesmas. A utilização do modelo ADDIE para o desenvolvimento da UI auxilia na obtenção de uma unidade com objetivos claros e concisos de forma a buscar o melhor desempenho do aprendiz, sendo este modelo cíclico, a UI pode sofrer mudanças caso seja necessário.

2.1.1 Ensino de computação no ensino básico

No Brasil a educação escolar é formada pela **educação básica** e pela educação superior**.** A educação básica inclui 3 fases: educação infantil, ensino fundamental e ensino médio (MEC, 2011)**.** O ensino fundamental é subdividido em 2, sendo eles o fundamental I e fundamental II.

Tabela 5 - Estrutura sistema de ensino básico

	Infantil	Fundamental I	Fundamental II	Ensino médio
Idade (anos)	0 a 6	6 a 11	11 a 15	A partir dos 15

Fonte: MEC, 2011

É de responsabilidade do Estado proporcionar acesso universal ao nível de educação básica para qualquer cidadão brasileiro, sobre a prerrogativa de que com a conclusão do ensino básico o educando estará desenvolvido, assegurar-lhe a formação comum indispensável para o exercício da cidadania e fornecer-lhe meios para progredir no trabalho e em estudos posteriores (MEC, 2011).

Dentro do âmbito do presente trabalho, com o foco no ensino básico, especificadamente o fundamental II, descrições e subdivisões de outras fases constituintes da educação básica serão abstraídas.

Anos Finais Fundamental II : 6º ao 9º ano

Para se garantir que todas as escolas a nível nacional compreendam os objetivos de aprendizagem propostos para cada uma das fases do ensino básico, o Ministério da Educação (MEC, 2011) disponibiliza os Parâmetros Curriculares Nacionais, para que todas as escolas sigam garantindo assim uma uniformidade no ensino. Porém dado a diversificação cultural e regional no país, faz-se necessário diferentes complementos desse currículo único, por parte de cada escola (PCN, 2015).

No ensino básico devem ser abordadas diversas áreas de conhecimento como: Língua Portuguesa, Matemática, História, Geografia, Ciências Naturais,

Educação Física, Arte e Língua Estrangeira (PCN, 2015). Espera-se do indivíduo que completa o ensino fundamental competências suficientes para se exercer atividades básicas na sociedade. O ensino fundamental desenvolve as habilidades dos domínios da linguagem escrita e comunicação verbal, bem como da área da matemática. No domínio da comunicação verbal, ao término do ensino fundamental o indivíduo possui a competência necessária para leitura de textos em Português. No âmbito da linguagem escrita, tem-se conhecimento para que seja possível escrever textos formais. Já na área matemática são desenvolvidas atividades para que o indivíduo seja capaz de executar funções matemáticas básicas (adição, subtração, multiplicação e divisão). Porém, observa-se que atualmente não se prevê a aprendizagem de competências de computação no ensino básico (PCN, 2015).

A Computação estuda a fundamentação teórica das construções computacionais, bem como suas aplicações em dispositivos tecnológicos e sistemas de computação (MEC, 2011). Dado que para a grande maioria das profissões atuais exige uma compreensão da Computação (CSTA, 2016), diversas diretrizes de currículos foram desenvolvidas no âmbito internacional, para contemplar o ensino da Computação no ensino básico. O mais comumente utilizado é o CSTA/ACM K12 (CSTA, 2016). O currículo CSTA/ACM -K12 enfoca todos os níveis do sistema de ensino básico. Ele possui 3 divisões com focos diferentes e complementares, baseado na etapa em que o indivíduo se encontra (fundamental I, fundamental II fases iniciais, ou fundamental III). Na fase 1(K6) correspondente ao fundamental I, são criadas experiências de aprendizado para que o aprendiz identifique a importância da computação no cotidiano dele mesmo. Na fase 2 (K6-K9) análogo ao ensino

fundamental II, o indivíduo deve começar a usar o pensamento computacional como uma ferramenta para solução de problemas. Nessa fase deve-se desenvolver experiências para que se adquira a percepção própria de pró atividade e capacidade de solucionar problemas. Atividades devem ter carácter exploratórios e podem ser ensinadas em unidades especificas de computação, ou agregadas a outras áreas de conhecimento. Na 3 fase do currículo (K9-K12) que representa o ensino médio, deve-se criar momentos para a aplicação de conhecimentos mais avançados de computação para criar soluções aplicáveis no “mundo real”. O ensino de computação segundo o CSTA deve abordar 5 áreas, conforme mostrado na figura 2.

Figura 2 - Áreas de conhecimento abordadas no CSTA K12

Fonte: CSTA, 2016

O pensamento computacional fomenta competências que favorecem a dedução e soluções de problemas (SICA, 2011), utilizando-se de conceitos como abstração, recursão, iteração e processamento, torna-se possível a solução de problemas dos mais variados domínios de conhecimento. O pensamento computacional auxilia os aprendizes a melhor conceituar, analisar e resolver problemas complexos, por meio da seleção e aplicação de estratégias apropriadas para solução de problemas tanto virtuais, quanto do mundo real. O pensamento computacional é um dos elementos principais no ensino de computação (CSTA, 2016).

No âmbito da computação é importante adquirir competências que permitem a programação, e todas as tomadas de decisões envolvendo a solução de um problema computacional/criação de software. Sendo um dos focos aprender conceitos que sirvam de apoio para a criação de programas de diversos tipos, como p.ex. aplicativos para dispositivos móveis. Tais conceitos, englobam laços de repetições, condicionais, e diversos outros que facilitem o entendimento de algoritmos, e por consequência, sirvam na resolução de problemas computacionais (CSTA, 2016). É importante que os elementos da computação moderna sejam entendidos, e como se usar de forma apropriada e correta terminologias tecnológicas (CSTA, 2016).

A ética no uso de computadores e redes é algo intrínseco aos fundamentos do ensino de computação, portanto deve ser ensinado conceitos sobre a ética de uso, princípios de privacidade, licenças de software e direitos autorais bem como a interpretação da confiabilidade e a acurácia das informações que são encontradas online. Aspectos sociais e econômicos influenciam no desenvolvimento de inovações computacionais, portanto os

aprendizes devem estar preparados para avaliar aspectos positivos e negativos referentes a disponibilidade de acesso a recursos computacionais, na sociedade (CSTA, 2016).

A seguir nas Tabelas 6 e 7 são detalhados os objetivos de aprendizagem para o nível 2 (6-9 ano. Ensino Fundamental II) público alvo enfocado no presente trabalho.

Tabela 6 - Objetivos de aprendizagem CSTA-K12 (anos inicias fundamental II)

ID	Objetivo	Área de conhecimento	Prática do Framework
2-A-2-1	Solicitar e integrar o *feedback* pareado para refinar programas	Algoritmos e programação	Colaboração
2-A-7-2	Comparar diferentes algoritmos para solução de um mesmo problema em termos de velocidade, clareza e tamanho	Algoritmos e programação	Falar sobre computação
2-A-7-3	Prover autoria para código emprestado ou construído a partir de algo pronto	Algoritmos e programação	Falar sobre computação
2-A-7-4	Interpretar o fluxo de execução de um algoritmo e prever seu resultado (algoritmos podem ser expressos em linguagem natural, fluxos, controles de diagramas e pseudocódigo)	Algoritmos e programação	Falar sobre computação
2-A-7-5	Desenhar, desenvolver e apresentar artefatos computacionais como aplicativos mobile que tenham impacto social, independentemente ou colaborativamente	Algoritmo e programação	Criar artefatos computacionais
2-A-7-6	Desenvolver programas que incluam sequências com loops aninhados (nesse nível pode-se utilizar linguagens de programação baseado em blocos ou linguagens textuais)	Algoritmo e programação	Criar artefatos computacionais
2-A-5-7	Criar variáveis que representem diferentes tipos de dados e manipular seus valores	Algoritmo e programação	Criar artefatos computacionais
2-A-4-8	Definir e usar processamentos que escondam a complexidade de uma tarefa e possam ser reusados para resolução de outros problemas	Algoritmo e programação	Desenvolvendo e usando abstrações
2-A-3-9	Decompor um problema em partes e criar soluções para cada parte	Algoritmo e programação	Reconhecendo e definindo problemas computacionais
2-A-6-10	Usar um processo de design iterativo (definição de um problema, ideação, construção, teste) para resolver problemas, de forma colaborativa ou independente	Algoritmo e programação	Testando e refinando
2-C-7-11	Justificar a escolha de hardware e software para alcançar um objetivo	Sistemas computacionais	Falar sobre computação
2-C-4-12	Analisar o relacionamento entre os	Sistemas	Desenvolvendo e

	componentes computacionais de um dispositivo e suas capacidades	computacionais	usando abstrações
2-C-6-13	Usar um processo sistemático para identificar a causa de um problema em um dispositivo (Ex: verificar o manual, mudar o software para verificar se o hardware funciona, reinciar o equipamento)	Sistemas computacionais	Testando e refinando
2-D-7-14	Descrever como diferentes formatos de arquivos representam balanceamento entre qualidade vs tamanho	Análise de dados	Falar sobre computação
2-D-7-15	Explicar o processo para coletar, transformar e analisar dado para resolver um problema usando *softwares*	Análise de dados	Falar sobre computação
2-D-5-16	Revisar modelos computacionais que mais refletem sistemas do "mundo real"	Análise de dados	Criando artefatos computacionais
2-D-4-17	Representar dados usando diferentes enconding schemas (Ex: binário, morse, Unicode)	Análise de dados	Desenvolvendo e usando abstrações
2-I-7-18	Sumarizar impactos positivos e negativos no uso de dados e informações para categorizar pessoas, predizer comportamentos e fazer recomentações	Impactos da computação	Falar sobre computação
2-I-7-19	Explicar como a ciência da computação fomenta a inovação e melhora qualquer carreira e disciplina	Impactos da computação	Falar sobre computação
2-I-1-20	Prover exemplos de como artefatos computacionais e dispositivos impactam na saúde e bem estar, positivamente ou negativamente	Impactos da computação	Fomentar a cultura de computação inclusiva
2-I-1-21	Descrever maneiras que a internet impacta na comunicação global e colaboração	Impactos da computação	Fomentar a cultura de computação inclusiva
2-I-1-22	Descrever problemas éticos relacionados a dispositivos computacionais e redes (Ex: hacking, igualdade de acesso, segurança, direitos autorais)	Impactos da computação	Fomentar a cultura da computação inclusiva
2-I-6-23	Redesenhar um artefato computacional para remover barreiras para acesso universal	Impactos da computação	Testando e refinando
2-N-7-24	Sumarizar riscos de securança associados a senhas fracas, falta de criptografia, transações inseguras e persistência de dados	Redes e a internet	Falar sobre computação
2-N-4-25	Simular como a informação é transmitida em pacotes por múltiplos dispositivos pela internet e redes	Redes e a internet	Desenvolvendo e usando abstrações

Fonte: CSTA, 2016

Tabela 7 - Objetivos de aprendizagem CSTA-K12 (anos finais fundamental II)

ID	Objetivo	Área de conhecimento	Prática do framework
3A-A-2-1	Desenhar e desenvolver um *software* funcional em equipe	Algoritmos e programação	Colaboração
3A-A-2-2	Demonstrar como a colaboração impacta no *design* e desenvolvimento de *software*	Algoritmos e programação	Colaboração
3-A-A-7-3	Comparar contraste entre várias licenças de *software* (Ex: *open-source*, *freeware*, comercial)	Algortimos e programação	Falar sobre computação
3A-A-5-4	Desenvolver e implementar um artefato computacional que responda a um evento (Ex: apps que respondem a uma mensagem, robô que responde a um sensor)	Algoritmos e programação	Criar artefatos computacionais
3A-A-5-5	Usar técnicas de *design* centrado em humanos para desenvolver soluções de *software* (Ex: Pesquisas, entrevistas)	Algoritmos e programação	Criar artefatos computacionais
3A-A-5-6	Integrar técnicas matemáticas adequadas para o nível no processo de criação de artefatos computacionais	Algoritmos e programação	Criar artefatos computacionais
3A-A-4-7	Entender a noção de hierarquia e abstração em linguagens de alto nível, traduções, instruções e circuitos lógicos	Algoritmos e programação	Desenvolvendo e usando abstrações
3A-A-4-8	Desconstruir um problema complexo em partes mais simples usando construções pré-definidas (Ex: funções, parâmetros, e/ou classes)	Algoritmos e programação	Desenvolvendo e usando abstrações
3A-A-4-9	Demonstrar o valor de abstrações para gerenciamento de complexidade de problemas (Ex: usando listas ao invés de variáveis discretas)	Algoritmos e programação	Desenvolvendo e usando abstrações
3A-A-3-10	*Design* de algoritmos usando sequencia, seleção e interações	Algortimos e programação	Reconhecendo e definindo problemas computacionais
3A-A-3-11	Explicar e demonstrar como modelagem e simulação podem ser usadas para explorar fenômenos naturais (Ex: filas, ciclos de vida)	Algoritmos e programação	Reconhecendo e definindo problemas computacionais
3A-A-6-12	Usar uma abordagem sistemática e ferramenta de debug para debuggar um programa (Ex: *breakpoints*, inspecionar variáveis)	Algortimos e programação	Testando e refinando
3A-C-7-13	Desenvolver e aplicar critérios para avaliação de um sistema computacional	Sistemas computacionais	Falar sobre computação
3A-C-5-14	Criar, entender e modificar programas existentes adicionando novas funcionalidades e comportamentos usando diferentes formas de entrada/saída	Sistemas computacionais	Criar artefatos computacionais

3A-C-4-15	Demonstrar o papel e interação de um computador embarcado com um sistema físico, como um sistema biológico, ou veiculo, ou criando um diagrama, modelo, simulação,,etc	Sistemas computacionais	Desenvolvendo e usando abstrações
3A-C-4-16	Descrever os passo necessários para um computador executar código de alto nível	Sistemas computacionais	Desenvolvendo e usando abstrações
3A-D-5-17	Criar modelos computacionais que simulem sistemas do mundo “real”	Análise de dados	Criar artefatos computacionais
3A-D-4-18	Conversão entre representações binária, decimal e hexadecimal	Análise de dados	Desenvolvendo e usando abstrações
3A-D-4-19	Analisar os prós e contras entre as formas digitais de uma informação	Análise de dados	Desenvolvendo e usando abstrações
3A-D-3-20	Discursar sobre técnicas de armazenamento, processamento e recuperação de informações	Análise de dados	Reconhecendo e definindo problemas computacionais
3A-D-3-21	Aplicar técnicas básicas para localizar e coletar conjuntos de dados(pequenos e grandes)	Análise de dados	Reconhecendo e definindo problemas computacionais
3A-I-2-22	Debater as implicações sociais e econômicas associadas a práticas computacionais éticas e não éticas	Impactos da computação	Colaboração
3A-I-7-23	Comparar e constratar o acesso e distribuição de direitos	Impactos da computação	Falar sobre computação
3A-I-7-24	Discutir implicações na coleta e análise de *big data* de um individuo	Impactos da computação	Falar sobre computação
3A-I-7-25	Descrever como a computação se aproxima da arte e música, traduzindo a intenção de um humano em um artefato	Impactos da computação	Falar sobre computação
3A-I-1-26	Comparar e debater impactos positivos e negativos da computação no comportamento e cultura	Impactos da computação	Fomentar a cultura da computação inclusiva
3A-I-1-27	Demonstrar como computação permite novas formas de experiencia, expressão, comunicação e colaboração	Impactos da computação	Fomentar a cultura da computação inclusiva
3A-I-1-28	Explicar o impacto da divisão digital de acesso a informações criticas (Ex: Acesso desproporcional a educação computacional)	Impactos da computação	Fomentar a cultura da computação inclusiva
3A-I-6-29	Redesenhar uma interface de usuário para ser mais inclusiva e acessível.	Impactos da computação	Testando e refinando
3A-N-7-30	Descrever os principais protocolos usados em serviços da internet (HTTP/HTTPS. SMTP/IMAP)	Redes e internet	Falar sobre computação
3A-N-4-31	Ilustrar os componentes básico de computação de redes(Topologias, diagramas de redes)	Redes e internet	Desenvolvendo e usando abstrações
3A-N-1-32	Comparar múltiplos pontos de vistas sobre *cyber* segurança	Redes e internet	Fomentar a cultura da computação inclusiva
3A-N-3-33	Explicar os princípios de segurança da	Redes e internet	Reconhecendo e

	informação (confiabilidade, integridade e disponibilidade) e técnicas de autenticação		definindo problemas computacionais
3A-N-3-34	Usar criptografia para enviar e receber mensagens	Redes e internet	Reconhecendo e definindo problemas computacionais
3A-N-6-35	Identificar estratégias digitas e físicas para segurança de redes e discutir sobre os prós/contras entre acesso a informação e segurança	Redes e internet	Testando e refinando

Fonte: CSTA, 2016

Observa-se que os objetivos de aprendizagem do CSTA-K12 estão alinhados de forma a conforme o grau de escolaridade aumenta, os objetivos ficam mais complexos. Observa-se também que o *framework* possui focos em diferentes domínios como colaboração, programação, análise social do impacto da computação, além de auxiliar na obtenção de conhecimento para discorrer sobre temas relacionados a computação, sendo assim não somente jovens que pretendem seguir a carreira de programadores podem tirar proveito do conhecimento adquiridos pelos objetivos de aprendizagem apresentados no currículo.

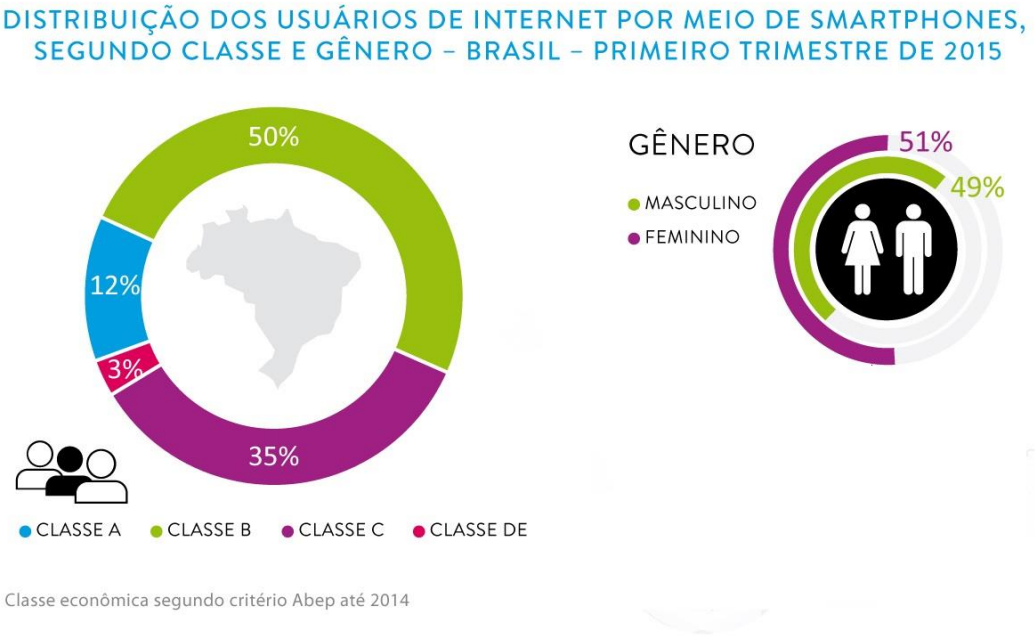
2.2 Cenário atual *Smartphones* e apps no Brasil

O pesquisador T. G. Paraskevakos em 1970 já conceituava dispositivos que integravam funcionalidades de telefonia, com computação (ISLAM;WANT, 2014). Tais conceitos serviram como base para o desenvolvimento dos celulares modernos que utilizamos hoje, e que são conhecidos como *smartphones*. Diferentemente dos celulares convencionais, os *smartphones* possuem capacidade e funcionalidades que vão além de efetuar ligações,

possuindo diversos serviços para os mais variados fins (ISLAM;WANT, 2014). Um dos fatores determinantes para que os *smartphones* e toda a indústria de comunicação se tornassem parte central na economia e sociedade, foi a evolução da tecnologia *wireless* e dos *hardwares* que permitiu a proliferação dos aparelhos para quase metade da população global (RAHA, 2009).

As projeções para o fechamento de 2016 em relação ao número de celulares *smartphones* e convencionais no Brasil eram de 94% para *smartphones* para 6% de celulares convencionais (IDC, 2016). Já as estimativas em relação a números de aparelhos *smartphones* no Brasil até o ano de 2018, são de 72 milhões (JANA, 2016). O Brasil é caracterizado como um país emergente para o mercado tanto de fabricantes de *smartphones*, quanto para a indústria de aplicativos. O número de pessoas que utilizam o aparelho celular como dispositivo para acesso à internet, chegou a 68 milhões em 2015 (IBOPE, 2015). Na Figura 3 é possível observar um panorama de usuários segundo a classe e gênero.

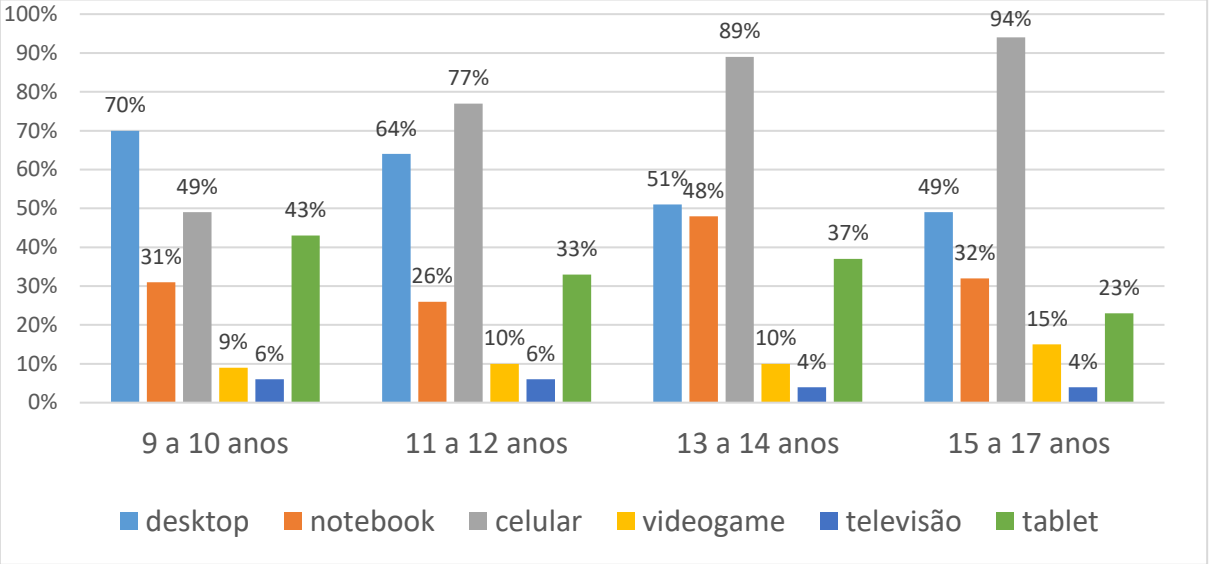
Figura 3 - Usuários de internet por meio de smartphones no Brasil



Fonte: IBOPE , 2015

A faixa etária que compreende indivíduos do ensino básico utiliza aparelhos celulares principalmente para acessar a internet (CETIC, 2014). A Figura 4 sub divide qual o tipo de dispositivo mais usado para acesso na internet por faixa etária. Pode-se perceber um ponto de “virada” em jovens a partir dos 11 anos, quando o dispositivo mais utilizado passa a ser os aparelhos celulares, em detrimento aos computadores.

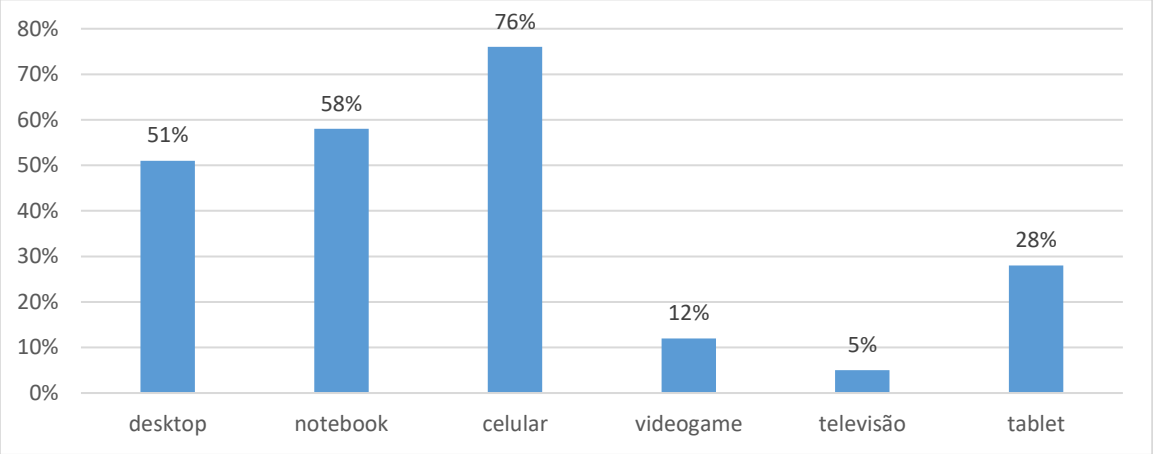
Figura 4 - Acesso à internet por dispositivo crianças/adolescentes



Fonte: Cetic, 2014

No âmbito da região sul do Brasil, 76% dos jovens da faixa etária de 9 a 17 anos utilizam o celular para acesso à internet como demonstra a Figura 5.

Figura 5 - Perfil de uso da internet de jovens na região Sul do Brasil



Fonte: Cetic, 2014

Os *smartphones* possuem *hardware* com alto poder de processamento, além de tela *touchscreen* com alta resolução. Possuem sensores dos mais

variados tipos como infravermelho, giroscópio, sensor de proximidade, entre outros. Eles podem se comunicar com outros dispositivos utilizando internet sem fio, *bluetooth* e outros protocolos. Possuem funcionalidades como leito de QR *code* e código de barras *(WORK;BAYEN, 2016).*

Os Sistemas Operacionais (SOs) para gerenciar todos os recursos de *hardware* dos celulares necessitam ser cada vez mais complexos. (LIN, 2009). Hoje existem dois SOs mais populares, e que dominam mais de 98% do mercado. São eles o Android, desenvolvido pelo Google e utilizado em aparelhos de fabricantes como Motorola, Samsung e LG (BUTLER, 2011). Outro SO dominante no mercado é o desenvolvido pela Apple, chamado de iOS e que roda nos aparelhos da própria marca. A Tabela 8 ilustra o *Market share* mundial por sistema operacional no 1 quarto de 2016.

Tabela 8 - Comparativo Market Share por SOs no 1Q 2016

Sistema Operacional	**1 quarto 2016(unidades)**	**1 quarto 2016 Market Share(%)**
Android	293.771,2	84,1
iOS	51.629,5	14,8
Windows	2.399.7	0,7
BlackBerry	659.9	0,2
Outros	791.1	0,2
Total	349.251,4	100

Fonte: GARDNER, 2016

No Brasil a tendência não é diferente do resto do mundo, sendo o Android o sistema dominante do mercado nacional. Com uma penetração de mercado de 90%, o Android é o sistema operacional mais usado (Jana, 2016).

Isso deve-se, principalmente pela estratégia de mercado da fabricante mais popular no país, que possui um portfólio de celulares acessíveis para todas as classes sociais. Popularizando assim, os aparelhos Android (Jana, 2016).

A partir do ambiente gerenciado pelos SOs, é possível o desenvolvimento de novos programas, chamados aplicativos. Esse tipo de *software* supre as necessidades da sociedade, utilizando-se das tecnologias disponíveis nos aparelhos celulares. Segundo CHENG ,

> "Um aplicativo mobile é um programa de computador feito para rodar em um smartphone, ou em outro aparelho móvel. Algumas características que são consideradas pontos chave para aplicativos são: Inicialização rápida, responsividade, foco no propósito e experiência consistente" (CHENG, 2016).

A quantidade de aplicativos disponíveis para utilização é vasta, basta o usuário efetuar o *download* na plataforma correspondente ao seu SOs. Na Google *Play Store*, loja oficial para aplicativos para Android, são mais de 2.2 milhões de apps, estimados em 2016 (STATISTA, 2016). Segundo indicadores, o Brasil é o segundo maior consumidor de aplicativos na *Google Play*. Em média, um brasileiro possuidor de celular Android, utiliza 18 aplicativos diariamente (Jana, 2016), e gasta 88% do tempo de utilização do celular com aplicativos (comScore, 2015).

2.3 App Inventor

O App Inventor[6] é uma ferramenta gratuita e *open-source* de criação de aplicativos para Android para dispositivos móveis. O ambiente de programação do app inventor pode ser acessado online ou offline via servidor dedicado (*localhost)*. Ele opera completamente na nuvem, não sendo necessário a instalação de nenhum *software* adicional no computador, ou seja, não é uma aplicação *desktop*. Foi criado em 2009 pela Google e vem sendo mantido pelo *Massachusetts Institute of Tecnology* (MIT*).* Ele é disponibilizado em diversos idiomas, inclusive em Português Brasileiro. Ele tem como missão democratizar o desenvolvimento de software dando poder para qualquer pessoa, especialmente jovens, desenvolverem aplicativos para celular (App Inventor, 2017). Possui uma extensa comunidade ativa em 2015 existiam mais de 3 milhões de usuários ativos em mais de 195 países. Já foram criados mais de 7 milhões de aplicativos para *Android* (App Inventor, 2017). O acervo de aplicativos desenvolvidos pode ser encontrado em uma plataforma web chamada *gallery*[7]. Após a criação do aplicativo, o usuário pode gerar o instalador, que possui extensão .apk, e serve para a instalação direta em celulares. Também há a possibilidade de disponibilizar o instalador na loja de aplicativos *Google play,* para que a comunidade de usuários de *smartphone* possa baixa-lo.

O App Inventor de forma nativa pode acessar quase todas as funcionalidades dos celulares, como: histórico de ligações, mensagens de texto, sensores de localização, orientação e acelerômetros, reconhecimento de

[6] http://appinventor.mit.edu/explore/
[7] http://classic-gallery.appinventor.mit.edu/#

voz, microfone, câmera do celular etc. Também existem possibilidades mais avançadas, que podem ser aplicadas dependendo do contexto com que o aplicativo a ser desenvolvido está inserido. Alguns exemplos são a invocação de outros aplicativos, o acesso a banco de dados para possíveis persistências de dados, acesso a serviços de informações de outros *softwares* via *Application Program Interfaces* (*APi´s)*. Acessando uma API é possível buscar dados presentes em outros domínios, como número de curtidas em um post no *facebook,* postar um novo twitter, entre outros.

A ferramenta App Inventor é composta de duas partes: uma voltada ao design de interfaces do aplicativo (Designer) e outra voltada a programação das funcionalidades (Blocos). No presente trabalho é usada a versão nb151a, compilada em setembro de 2016 (http://ai2.appinventor.mit.edu/).

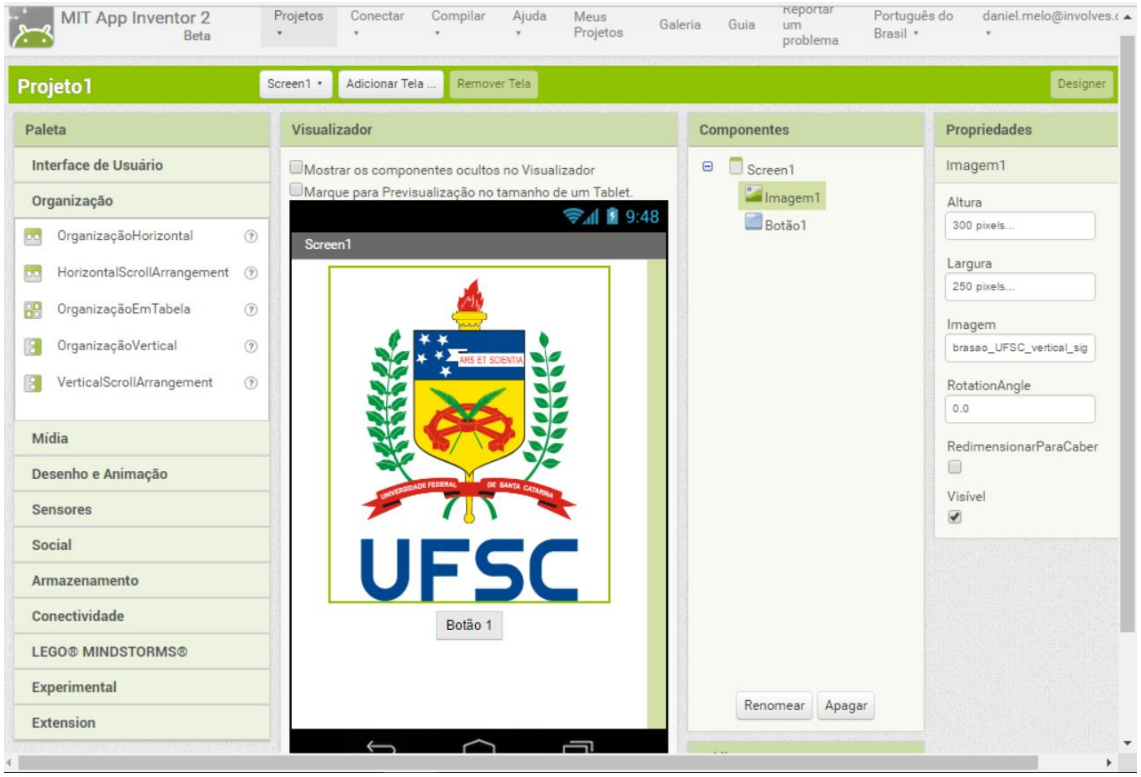
A linguagem utilizada pelo App Inventor é conhecida como programação baseada em blocos. Esse tipo de programação, visa deixar mais intuitivo e familiar o processo de desenvolvimento de *software* para pessoas que não possuem conhecimentos de programação. Dado que para se criar uma instrução usa-se blocos visuais.(App Inventor, 2017) Esses blocos são facilmente criados por meio de uma interface, arrastando blocos individuais, ou um conjunto, configurando comportamentos padrões ou customizar comportamentos de acordo com a necessidade do domínio do problema (App Inventor, 2017). Tais blocos representam execuções disparadas a partir de ações dos usuários do aplicativo. Este modelo de programação abstraí grande parte da complexidade, evitando assim o aspecto verboso característico de programações orientadas a texto (ROY, 2012). A ferramenta fornece a possibilidade de testes em tempo real, ou seja, é possível verificar a

visualização do resultado de mudanças programadas, em um celular compatível (com Android versão igual ou superior a 2.3), ou utilizando um ambiente de simulação no computador. Permitindo assim, um ambiente propicio para o ensino e a exploração das possibilidades por parte do aprendiz.

2.3.1 O Designer

O designer é a parte utilizada para se programar a interface gráfica da aplicação, e para adicionar comportamentos não visuais, porém que afetam a experiência do usuário ao utilizar o aplicativo, como por exemplo, adição de sons ao clicar em um botão, ou a vibração do celular ao digitar um texto em um formulário, etc. Sendo assim é possível posicionar caixas de texto, criar formulários, adicionar ícones e botões, estilizar a cor dos componentes, adicionar imagens, entre outros. Nessa parte podemos ter uma prévia de como os componentes visuais estarão dispostos na tela do celular ao utilizar o aplicativo. A Figura 6 é uma visão geral da área de trabalho do Designer.

Figura 6 - Área de trabalho do Designer



Fonte: App Inventor, 2017

A área de trabalho da parte do Designer é subdivida em 4 espaços:

- **Paleta**: Na paleta são organizados os componentes visuais e não-visuais (relacionados a experiência do usuário) que podem ser utilizados para o *design* da interface gráfica do aplicativo. Os componentes são categorizados para melhor organização. Para adicionar algum novo componente, é necessário selecionar o componente e utilizar o clique e arraste do mouse posicionar o componente na área desejada no visualizador.
- **Visualizador**: área onde é simulado um celular ou *tablet*, para visualizar a prévia da disposição dos componentes na tela.

- **Componentes**: mostra a organização dos componentes já adicionados, de forma hierárquica em uma árvore. É possível selecionar um item ou renomear o mesmo.
- **Propriedades**: Mostra todas as propriedades configuráveis para o componente selecionado na etapa anterior, como altura, largura, fontes, cor de fundo entre outros. As propriedades variam de acordo com o tipo de componente que está sendo editado.

As Tabelas de 9 a 15 mostram em detalhes a descrição dos tipos de componentes na parte da Paleta, e que são pertinentes ao escopo do presente trabalho.

Tabela 9 - Categoria Interface de Usuário da sessão Designer

Interface de Usuário	
Componente	Descrição
Botão	Componente com a habilidade para detectar cliques. Sua aparência e comportamento pode ser alterado via propriedades inerentes do componente.
Check Box	Componente representativo de estados booleanos (verdadeiro/falso). Quando clicado um evento é disparado. Assim como botão, aparência e comportamento alterado via propriedades.
Seleção de Data	Quando clicado, permite usuário selecionar uma data.
Imagem	Componente para a representação de uma imagem que aparecerá no app
Legenda	Trecho de texto utilizado como legenda, pode ter sua localização e outras propriedades alterada na etapa de *designer* ou blocos.
Seleção de Lista	Mostra ao usuário uma lista de textos para escolha. É possível atribuir a opção de filtragem por meio da propriedade especifica.
Visualizador de listas	Componente que mostra listagem na tela.
Notificador	Mostra caixas de diálogos de alertas, mensagens e alertas temporário, além de disparar eventos de log para o Android
Campo de Senha	Campo onde é possível adicionar uma senha. Comportamento similar ao de Campo de Texto, porém, não mostra os caracteres digitados.
Deslizador	Indicador onde é possível haver um movimento de arraste horizontal. Cada estado final pode executar diferentes ações.

Lista Suspensa	É componente apresenta janela com lista de elementos para seleção
Caixa de Texto	Componente para usuário inserir um texto.
Selecionador de Hora	Quando clicado, permite selecionar uma hora especifica.
Navegador Web	Componente para representar links da web, ao clicar direciona para algum site especifico
Pintura	Componente que cria um área para desenhos, utilizando a tela sensível ao toque
SpriteImagem	Componente que associa movimentos de toque e arrastes a objetos de imagens

Tabela 10 - Categoria Organização da sessão Designer

Organização	
Componente	Descrição
Organização Horizontal	Componente utilizado para formatação, organizando elementos da esquerda para direita, horizontalmente.
Organização Em Tabela	Organiza elementos em formato de tabela
Organização Vertical	Componente utilizado para formatação, organizando elementos de cima para baixo, verticalmente.

Tabela 11 - Categoria Mídia da sessão Designer

Mídia	
Componente	Descrição
Câmera de Vídeo	Componente que permite a gravação de vídeo utilizando a câmera do dispositivo.
Câmera	Permite a captura de uma foto, que é salva no dispositivo e pode ser usada com o componente de imagem.
Seletor de imagem	Acessa a galeria do dispositivo para seleção de uma foto.
Tocador	Componente que reproduz áudio e controla a vibração do dispositivo. Recomendado para tocar sons mais longos.
Som	Componente que reproduz áudio e controla a vibração do dispositivo. Recomendado para tocar sons mais curtos.
Gravador	Componente para gravar áudios.
Reconhecedor de Voz	Componente que reconhece voz e converte em texto.
Texto para Fala (*Text To*	Ao ser utilizado permite conversão de textos em voz humana

speech)	sintetizada.
Reprodutor de Vídeo	Reproduz vídeos
Tradutor	Usa a internet para fazer a tradução de palavras ou sentenças distintas.

Tabela 12 - Categoria Desenho e Animação da Sessão Designer

Desenho e Animação	
Componente	Descrição
Bola	Componente que simula uma bola, configurando suas propriedades é possível que ela se mova automaticamente.
Pintura	Painel sensível ao toque para se desenhar, ou editar imagens.
SpriteImagem	Imagens que sensíveis a toque, que podem interagir com outros elementos do aplicativo.

Tabela 13 - Categoria Sensores da sessão Designer

Sensores	
Componente	Descrição
Acelerômetro	Acessa sensor do dispositivo de medição de aceleração.
Código de Barras	Componente para reconhecimento de códigos de barra.
Temporizador	Componente invisível que mostra data/hora atual usando relógio interno do dispositivo. Também pode disparar eventos em um intervalo de tempo.
Giroscópio	Acessa sensor do dispositivo para medir velocidade angular do aparelho.
Sensor de Localização	Fornece a localização geo-espacial do aparelho, como latitude, longitude entre outros. Incluindo conversão para endereço.
Campo de aproximação	Acessa sensor do dispositivo para executar capacidade NFC.
Sensor de Orientação	Acessa sensor do dispositivo que informa orientação do aparelho, em graus, como é feito com bussolas.
Pedômetro	Componente para contar passos.
Sensor de Proximidade	Componente que identifica a distância de objetos em relação a tela do aparelho, usado por exemplo, para verificar se aparelho encontra-se em uso, pela distância da orelha do usuário a tela do aparelho.

Tabela 14 - Categoria Social da sessão Designer

Social	
Componente	Descrição
Escolher Contato	Botão para seleção de contatos em uma lista.
Escolher Email	Caixa de texto, com autocomplete em e-mails conhecidos.
Ligação	Componente invisível utilizado para executar ligações.
Escolher Telefone	Botão que exibe listagem de telefones para seleção.
Compartilhamento	Componente invisível para compartilhar dados entre o aplicativo, e outros aplicativos que sejam compatíveis.
SMS	Componente para envio de SMS.
Twitter	Componente para se realizar interações com a plataforma do Twitter. Como fazer um tweet, pesquisar tweets entre outros.

Tabela 15 - Categoria Armazenamento da sessão Designer

Armazenamento	
Componente	Descrição
Arquivo	Componente invisível para escrever/ler em arquivos no dispositivo.
Fusion Tables	Componente que permite conectar com *Google Fusion Tables*.
TinyDB	Componente invisível para executar persistência de dados interna no aplicativo.
TinyWebDB	Componente para se conectar ao serviço web para persistência de dados.

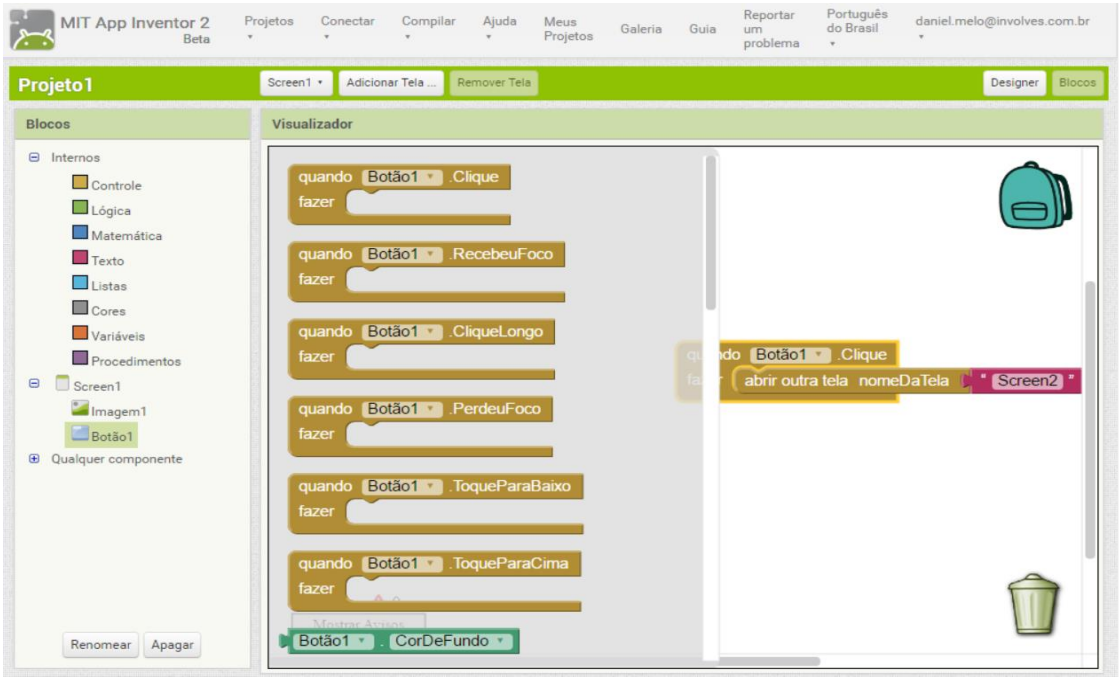
2.3.2 Blocos

O Blocos possibilita a criação da lógica das funcionalidades do aplicativo, de acordo com o comportamento esperado para o app. Adicionando blocos que são análogos a algoritmos, é possível a utilização de conceitos

básicos de lógica de programação. Os componentes definidos no *design* de interface são “carregados” para nessa etapa, serem utilizados na programação.

A Figura 7 mostra a visão geral da área de trabalho da sessão de Blocos. Nela é possível identificar que o botão1 foi selecionado na parte de Blocos. O visualizador então, mostra as opções de blocos viáveis para esse tipo de elemento, então foi criado o algoritmo para que ao clicar no botão1 a tela Screen2 seja aberta.

Figura 7 - Área de trabalho da Sessão de Blocos



Fonte: App Inventor , 2017

- **Blocos**: Nessa parte ficam organizados no modelo de árvore, os blocos considerados *default*, com comportamento padrão para cada uma das

suas categorias. Cada categoria possui uma cor característica, para facilitar o entendimento. Em seguida são listados os componentes adicionados na etapa de *Designer*, para que seja possível associar funcionalidades a esses componentes. Ao clicar sobre um desses itens, são listados na tela do visualizador todos os possíveis blocos para aquele tipo de componente.

- **Visualizador**: Área dedicada a visualização dos blocos de programação, divide-se em região para edição, onde é possível arrastar livremente os blocos, reordenar, ou até mesmo deletar blocos, e região onde é visível as possibilidades de blocos baseado na escolha realizada na parte anterior.

Existem tipos de categorias, e suas permutações possibilitam a criação de um alto índice de comportamentos diferentes, para a aplicação. As categorias possíveis para os blocos são detalhadas na Tabela 16.

Tabela 16 - Categoria de comandos da sessão de Blocos

Categorias de comandos	Descrição
Controle	Comandos que executam funções de controle, como verificações booleanas, comandos de decisões do tipo if-else, interações em listas.
Lógica	Componentes de blocos relacionados a operações lógicas, como verdadeiro/falso, e/ou.
Matemática	Blocos responsáveis por representar operações matemáticas como divisão, multiplicação, soma e cenários mais complexos como seno, cos e tangente.
Texto	Contém opções de operações sobre textos, como juntar dois textos, verificações se contém um certo padrão, retirar espaços em brancos de textos.
Listas	Executa operações sobre listas, criando/editando/deletando as mesmas, bem como verificações como tamanho da lista, se já existe o elemento na lista.
Cores	Blocos para definição de cores do aplicativo, disponibilizando cores

	padrões, ou customizadas pelo desenvolvedor.
Procedimentos	Blocos para definição e chamada de funções.
Variáveis	Blocos para inicialização de variáveis e edição das mesmas.

Fonte: App Inventor, 2017

Apesar de seu uso intuitivo e forte utilização pela comunidade, o App Inventor possui algumas limitações, tal como em relação a mudança de orientação da tela, ajustando a posição do celular de horizontal para vertical, em alguns casos o comportamento esperado não é executado (App Inventor, 2017). Outras limitações são ligadas a usos mais específicos e técnicos, voltados para aplicativos mais complexos e conceitos de programação avançados, porém essas limitações são conhecidas e já estão em processo de resolução. (App Inventor, 2017). Para o presente trabalho o foco encontra-se nas limitações relacionadas a paleta de cores disponíveis para estilização de componentes. Sendo que não existe uma forma intuitiva, de se utilizar as cores padronizadas de aplicativos Android (Material Design), comprometendo assim o *design* das interfaces gráficas, e sua compatibilidade com as tendências relacionadas a *user experience* (UX) de aplicativos modernos.

2.4 User Experience

No contexto atual as organizações estão cada vez mais dependendo de inovação e *design* para atingir as estratégias de negócio, alcançar a diferenciação no mercado, criando vantagem competitiva (VINNAKOTA, 2014). Na indústria de criação de aplicativos não é diferente. Para que um aplicativo tenha mais chances de sucesso, ele deve ser inovador e criativo, além de ser

agradável na sua utilização, combinando eficiência com satisfação de uso (YIN *et al.* 2014). Dado esse cenário, a UX é um importante fator de medida da sua qualidade (NASCIMENTO *et al.*, 2016). Algo que todos os aplicativos de sucesso têm em comum, é o fato de a experiência de usuário que eles proporcionam, é eficaz e marcante (EVEN, 2016).

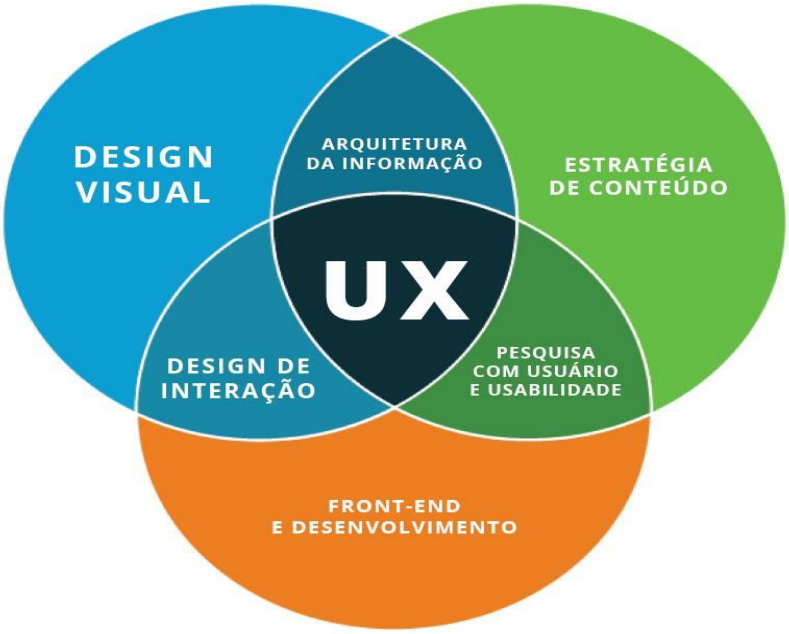
Toda e qualquer interação que temos com objetos, a fim de realizar alguma tarefa, nos proporciona uma experiência. Seja essa tarefa funcional (pagar contas no *internet banking*) ou emocional (visualizar perfil social de amigos) (TEIXEIRA, 2014). A qualidade dessa experiência está diretamente ligada a percepção humana, portanto é subjetiva. Cada pessoa tem uma experiência diferente, executando a mesma tarefa. Isso é influenciado por fatores humanos (experiência no uso, capacidades motoras, humor no momento do uso) e fatores externos (ambiente de uso, horário do dia, fatores climáticos) (TEIXEIRA, 2014). Segundo HESS,

> "A maioria das pessoas acredita que User Experience é somente encontrar a melhor solução para os seus usuários – mas não é. UX se trata sobre definir o problema que precisar ser resolvido (porquê), definir para quem esse problema precisa ser resolvido (quem), e definir o caminho que deve ser percorrido para resolve-lo. (como)." (HESS,2010)

Toda e qualquer experiência é temporal, ou seja, possui um começo meio e fim. Portanto UX pode ser definido como sendo um sentimento momentâneo de avaliação (boa/ruim) enquanto se interagindo com um produto ou serviço (HASSENZAHL, 2008). Uma boa experiência em aplicativos móvel resulta de sistematicamente se buscar decisões de projeto que apoiem a entrega final de um aplicativo que exceda as expectativas dos usuários (SHAH, 2016). Para tal, UX estuda todos os aspectos que envolvem a interação entre

um usuário com uma interface ou sistema (ZAHIDI, 2014). Utilizando conceitos derivados do campo de pesquisa que cobre termos mais abrangentes de *design* de interações, conhecido como interação “homem-computador” (ISLEIFSDOTTIR, 2014). A UX compreende vários domínios como a usabilidade, arquitetura da informação, estratégia de conteúdo, design de conteúdo (HARTSON, 2012) (TRENDER, 2014).

Figura 8 - Domínios envolvidos no UX



Fonte: TRENDER, 2014

A maioria dos aplicativos possuem interfaces de usuário pouco intuitivas e que não acessam informações de forma satisfatória. Nese contexto, o domínio de *design* visual pode determinar o sucesso ou fracasso de um aplicativo (UMGEHER, 2008). Usuários de aplicativos móvel interagem com as

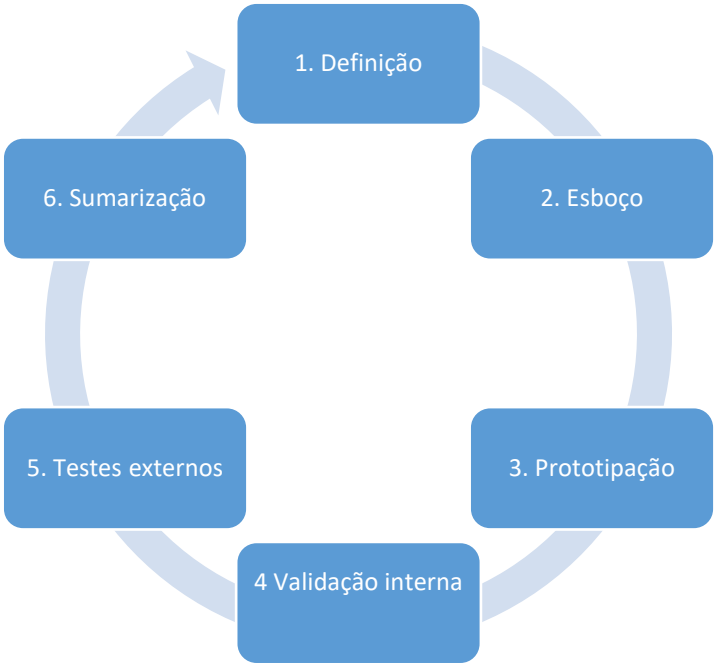
interfaces dos aplicativos por meio de menus e ícones. Portanto sendo esses elementos o mais intuitivo possível., há uma melhoria na usabilidade e consequentemente melhoria na UX (NAZRUL, 2015). Buscar projetar aplicativos que apresentem boa UX significa permitir que os usuários do aplicativo possam atingir seus objetivos de forma satisfatória em um contexto particular de uso (ISO/IEC 9241). Para tal busca-se desenvolver um aplicativo que possua uma boa usabilidade. Usabilidade é,

> "uma medida na qual um produto é usado por um usuário especifico, para alcançar objetivos específicos com eficácia, eficiência e satisfação de uso em um contexto de uso" (ISO/IEC 9241).

Tratando-se de aplicativos móvel, existem alguns fatores adicionais que implicam na UX. Por exemplo, a diferença de tamanho das telas de dispositivos móveis influência na natureza de interação do usuário, já que os dispositivos móveis oferecem novas formas de interações (via gestos, sensores, câmeras) (WASSERMAN, 2010). O contexto no qual o usuário se encontra inserido também influência. Aplicativos de dispositivos móveis, diferentemente de outros *softwares* podem ser utilizados em qualquer lugar. Utilizar um aplicativo correndo, pode ser mais difícil do que parado, ou dirigindo (KUUSINEN;MIKKONEN, 2014). Ou seja, o *design* de *interfaces* para aplicativos móveis, é mais desafiador ainda, tendo em vista a vasta variedade de pessoas com objetivos diferentes, interagindo em qualquer momento e lugar (HUANG, 2009). Sendo assim, para o *design* de UX de aplicativos mobile é importante entender a necessidade do usuário, e também entender as limitações de *hardware* dos dispositivos que rodaram o app, afim de promover uma consistência na experiência de uso (KUUSINEN;MIKKONEN, 2014).

Para desenvolver aplicativos com boa experiência de usuário, é necessário seguir um processo sistemático do *design* de interface/interação. Existem vários modelos de desenvolvimento de aplicativos voltado a UX. Um dos modelos de processo de *design* de interface é Nielsen (1992) ilustrado na Figura 9.

Figura 9 - Ciclo de Design Interface



Fonte: Nielsen, 1992

Estas etapas são caracterizadas como:

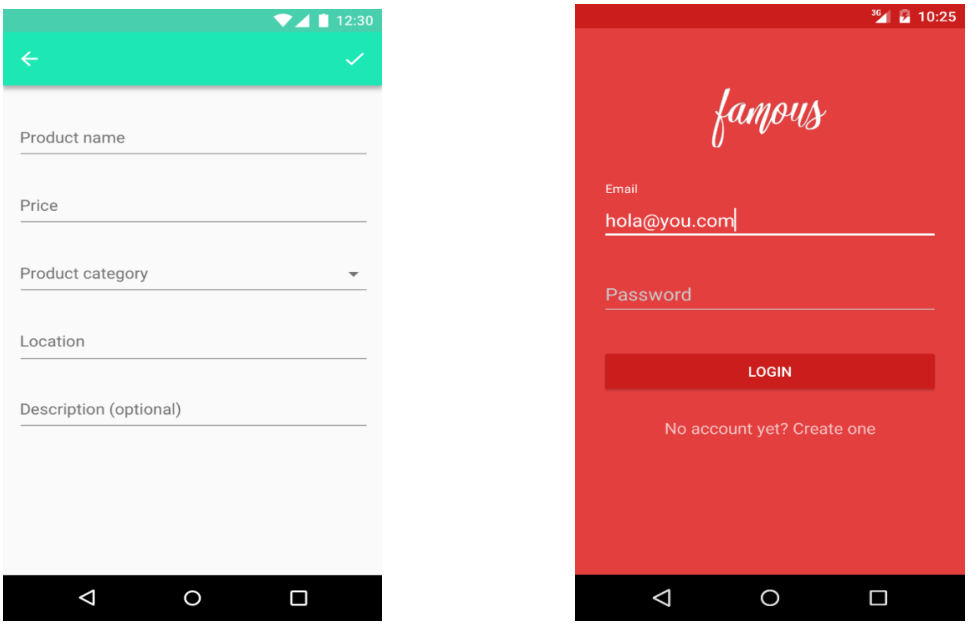
- Definição: Nesta etapa se elenca as informações necessárias para o desenvolvimento da interface. Só sabendo essas informações é possível se tomar decisões válidas relacionadas ao *design* da UI. Para cumprir com esse objetivo é possível elucidar as seguintes atividades:

  - Conhecer o usuário: Levantar todos os tipos de usuários, conhecer suas diferenças, e suas semelhanças para traçar personas.
  - Análise competitiva: Analisar como são interfaces de aplicativos similares do mesmo segmento. Identificar pontos fortes, e pontos fracos, afim de desenvolver uma UI que apresente boa competitividade em relação a concorrentes.
  - Definir requisitos de usabilidade: definir os requisitos de usabilidade de acordo com as necessidades dos usuários e das características do contexto.
- Esboço: Esboçar primeiros protótipos de baixa fidelidade, para serem refinados na etapa de prototipação.
- Prototipação: Fase em que os esboços são refinados e protótipos de alta fidelidade são desenvolvidos.
- Validação Interna: São realizados testes internos na equipe e/ou empresa, analisando se são atendidos os requisitos de usabilidade ou identificar pontos fortes e fracos.
- Teste externos: Avaliação com usuários “reais”, aplicar testes empíricos com o intuito de gerar feedback, que serão insumo para melhora do protótipo.
- Sumarização: Etapa de documentação do aprendizado referente ao ciclo. Afim de que esse sirva como base para novas interações de design de interfaces no futuro. Informações que podem ser documentadas incluem os *feedbacks* de testes e “lições aprendidas” em cada uma das etapas.

Para suportar o *design* de interfaces de aplicativos moveis também já existem guias de estilo que ajudam em fazer protótipos de *designs* de interface uniforme e alinhado a princípios básicos de usabilidade. Estes guias de estilo são diferentes para cada SO. Para Android, existe o Material Design[8] criado pela Google. O Material Design é um framework de padrões de *design*, que permite criar uma identidade visual, capaz de sintetizar bons princípios de design gráfico, com as possibilidades de inovação associadas a tecnologia (Google Material, 2016). Possibilitando uma experiência unificada entre diferentes dispositivos. O Material Design se tornou o padrão do Android desde a versão 5.0 *lollipop* (ComputerWorld, 2015). Existem também padrões de designs de telas que são convenções adotadas para os elementos gráficos em telas, de acordo com o objetivo da tela, para proporcionem UX semelhantes, independente do aplicativo. Como por exemplo tela de login, formulário, busca etc. Esses padrões são alinhados ao guia de estilo Material Design.

[8] https://material.google.com/

Figura 10 - Exemplos formulários Material Design



Fonte: Material Design, 2016

2.4.1 Design de Cores

Tratando-se do *design* de interfaces de *softwares* um dos elementos mais importante para uma melhor comunicação de ideias e conceitos, é *o design* das cores dos componentes gráficos (GONÇALVES, 2014). Saber utilizar as cores de forma coerente, ajuda a alcançar objetivos como atrair a atenção do observador, fragmentar áreas da interface, além de auxiliar no processo de memorização. Sendo estes aspectos importantes para a obtenção de uma interface com uma boa UX (GONÇALVES, 2014).

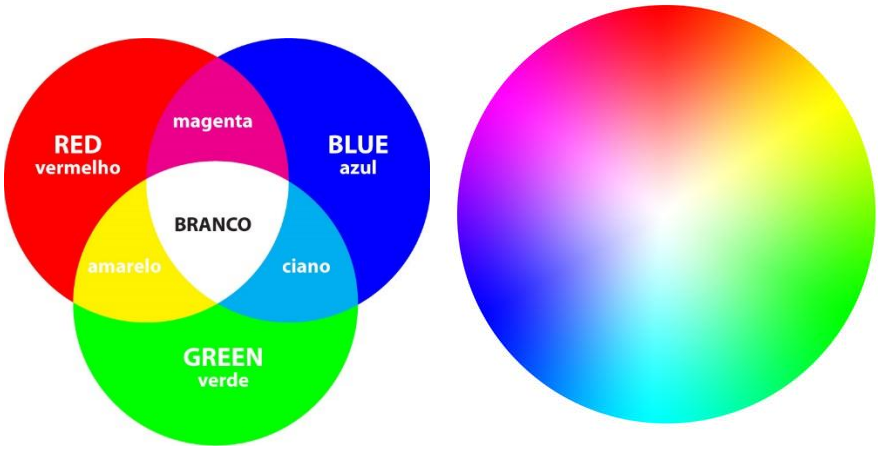
O processo de percepção e reconhecimento das cores caracteriza-se como o resultado da interação entre uma fonte de luz, um

objeto e um observador (BERNS, 2000). Vários aspectos estão envolvidos no estudo das cores, como a fonte de luz (quantidade de luz), da forma do componente gráfico (tamanho) e do observador (sensibilidade espectral, adaptação luminosa e cromática, etc), o que torna esse tema complexo (GONÇALVES, 2014).

O que faz com que o olho humano reconheça a cor de um objeto, é a quantidade de absorção/reflexão da luz sobre o objeto (RAUTENBERG, 1998). Assim sendo, um objeto que absorve toda a luz, é reconhecido como negro, quando reflete por completo a luz é reconhecido como branco, e quando absorve parcialmente a luz, é considerado colorido.

A representação de cores em dispositivos eletrônicos é diferente da representação/classificação em outros meios. O sistema de representação de cores em monitores e telas de celulares, segue o padrão de mistura aditiva de cor, utilizando a representação das luzes vermelho **(R)** verde **(G)** e azul **(B),** que quando projetadas simultaneamente formam o branco (GONÇALVES, 2004). São consideradas “cores primárias”: vermelho, verde e azul-violeta, já as cores compostas – ou secundárias – são, amarelo, ciano e magenta. A mistura das tonalidades **RGB**, formam todas as cores possíveis dentro do espectro de tonalidades.

Figura 11- Roda de cores RGB



É de conhecimento da ciência que o ser humano tem dificuldades em classificar algumas tonalidades, quando as mesmas encontram-se tecnicamente, no limiar da categoria da cor (alguns tons de azul e verde, por exemplo) (FOLEY; MATLIN, 1986). Além da limitação do olho humano, fatores culturais e psicológicos também influem na seleção de cores (GONÇALVES, 2004). A teoria das cores inclusive estuda fatores emocionais e de sentimentos, que são associados a cores especificas. Empresas utilizam dessa técnica para transmitir por meio da seleção de cores de suas Logomarcas, sentimentos associados a empresa, como demonstra a Figura 12.

Figura 12 - Guia emocional das cores

Fonte: Mundo da Psicologia, 2015

Dada a influência que a cor tem na percepção (FRASER, 2007), e por se tratar de um fenômeno que nos atinge de modo intenso e direto (ARNHEIM, 2002), o processo de *design* das cores de um *software* deve considerar a escolha de cores que sejam rapidamente identificadas e categorizadas pelo usuário, e sigam padrões e recomendações referente ao *design* de interfaces, de modo a contribuir positivamente para a usabilidade e atratividade do *software*.

O *design* de cores também faz parte de guias de estilo, como por exemplo para aplicativos da plataforma Android definidos pelo Material Design (Google, 2017). As cores sugeridas no Material Design são inspiradas por contrastes fortes justapostos por sombras aprofundadas, e destaques claros (Material Design Color, 2017). A seleção de cores do Material Design compreende cores primárias e suas ascendências. Elas foram selecionadas de

forma a serem harmônicas entre si, iniciando com as cores primárias e englobando o espectro criando uma gama completa de cores para utilização em aplicativos Android. Como sugestão deve-se utilizar as cores 500 que são as cores definidas como cores bases do Material Design, como as cores primárias do aplicativo. As cores representadas com o valor A (por exemplo, A109) são cores ascendentes e mais saturadas, fatores esses que encorajam a interação.

Figura 13 - Amostras vermelho, rosa e purpurina

Fonte: Material Design Color, 2017

Figura 14 - Amostras purpurina profundo, índigo e azul

Fonte: Material Design Color, 2017

Figura 15 - Amostras azul claro, ciano e teal

Fonte: Material Design Color, 2017

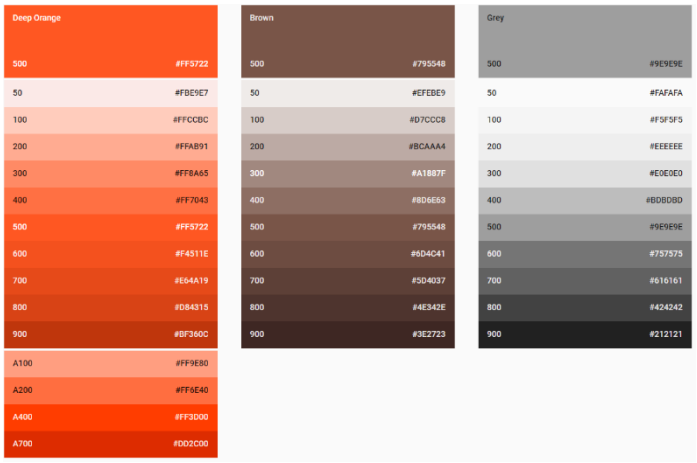
Figura 16 - Amostras verde, verde claro e limão

Fonte: Material Design Color, 2017

Figura 17 - Amostras amarelo, âmbar e laranja

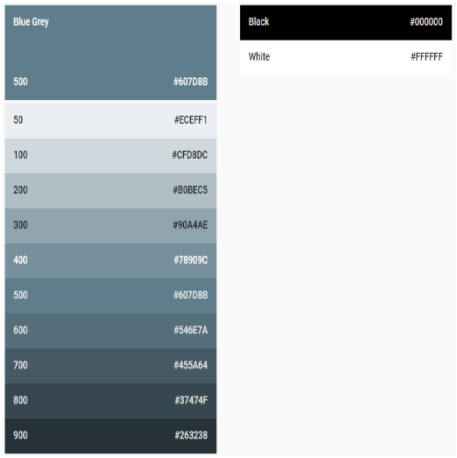
Fonte: Material Design Color, 2017

Figura 18 - Amostras laranja profundo, marrom e cinza



Fonte: Material Design Color, 2017

Figura 19 - Amostras cinza azulado, branco e preto

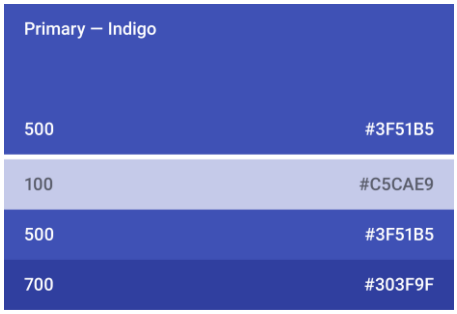


Fonte: Material Design Color, 2017

O sistema de cores do Material Design faz referência a uma cor primária, como sendo a cor que mais deve aparecer na interface do seu aplicativo. Caso cores secundárias não forem utilizadas, utiliza-se variações do tom da amostra de cor primária escolhida para obtenção de contraste entre os elementos. O

contraste ajuda na divisão da superfície dos elementos, como por exemplo entre uma barra de status e um menu *toolbar.*

Figura 20 - Exemplo cor primária

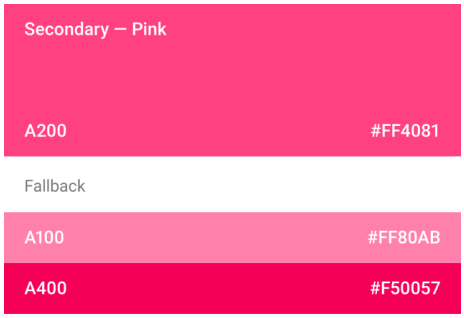


Fonte: Material Design Color, 2017

Em caso de utilização de cores secundárias, tendo em vista que seu uso é opcional, essas assumem o papel de destacar os elementos da sua interface gráfica. Elas podem ser complementares, ou análogas a cor primária utilizada e não necessariamente são cores chamativas, elas simplesmente devem contrastar com seus elementos próximos. Seu uso mais indicado é nos elementos como:

- Botões (flutuantes ou textuais)
- Campos de texto, cursores e seleção de textos
- Barras de progresso
- *Sliders*
- *Links*
- Cabeçalhos

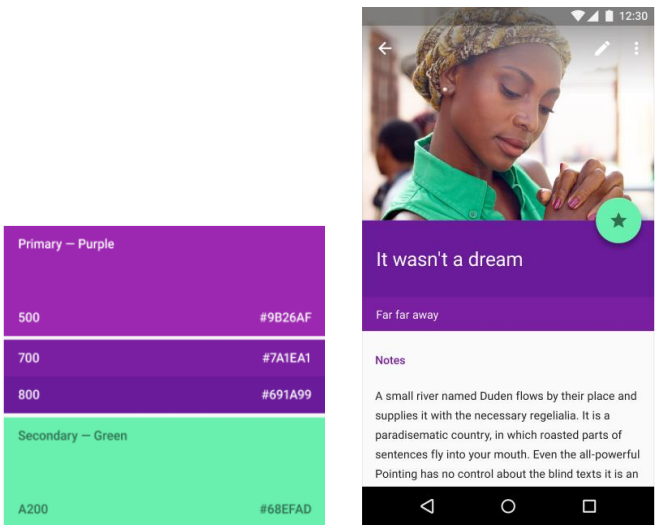
Figura 21 - Exemplo cor secundária



Fonte: Material Design Color, 2017

Na Figura 22, observa-se um esquema de cores primárias (purpura com três tons), e cores secundárias (verde com dois tons), e sua utilização em uma tela modelo. Observa-se utilização da cor primária principal em contraste com tons mais claros para definir o espaçamento entre os elementos. Já a cor secundária, é utilizada para dar um destaque no botão flutuante, presente na interface desenvolvida.

Figura 22 - Exemplo esquema de cor



Fonte: Material Design Color, 2017

A utilização de cores primárias e secundárias pode variar conforme o tipo de elemento (botão, menu, texto, *switches,* imagens) além do seu contexto (*links*, títulos, seleção textual). Áreas maiores da interface gráfica devem possuir cores primárias enquanto destaques em áreas pequenas devem utilizar cores secundárias. Em casos de interfaces com características mais monocromáticas, o destaque se dá pela variação de tons primários.

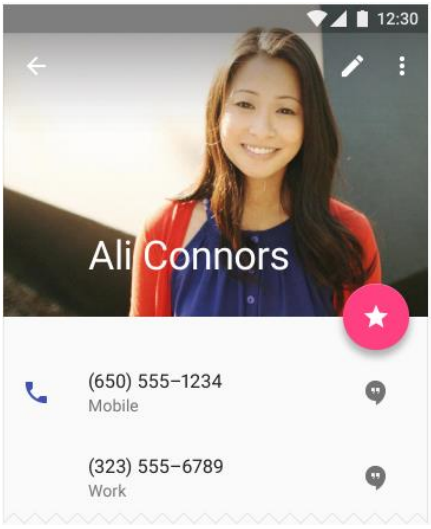
O guia de estilo Material Design também propõe diretrizes referente ao *design* de cores para elementos específicos, que são apresentados nas próximas seções.

2.4.1.1 Botões

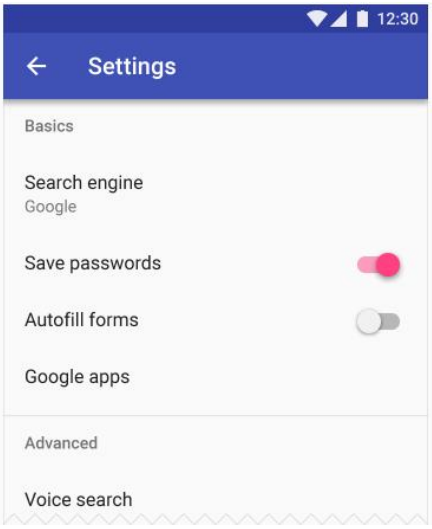
Interfaces possuem diversos botões, e escolher a cor destes, implica em chamar a atenção do olho do usuário para determinada região/ação. Na figura 23.1, podemos observar que a cor secundária é utilizada para direcionar o olhar para a ação de favoritos, enquanto a cor primária é utilizada nos outros botões, de chat e telefone. Já na figura 23.2 é possível visualizar como a cor secundária é utilizada, para enfocar no *switch* que representa o status de ativo, enquanto a cor primária é utilizada para inativo. O exemplo 23.3 demonstra a utilização correta da cor secundária em um botão de ação principal em contraste com seus elementos próximos que utilizam cores primárias. Proporcionando assim a sensação de destaque. Em contrapartida na figura

23.4 utiliza de forma **errônea** a cor secundária no botão de ação principal, na barra superior, e em grande parte da área da UI.

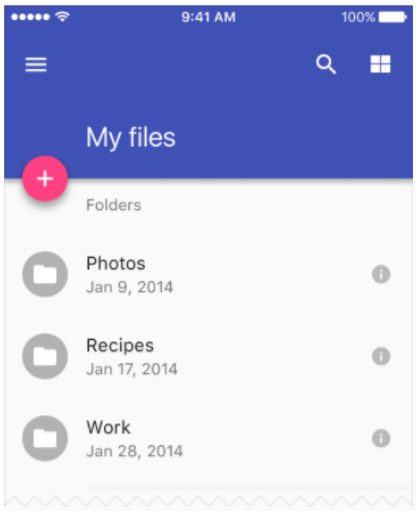
Figura 23 - Cores primárias/secundárias em botões



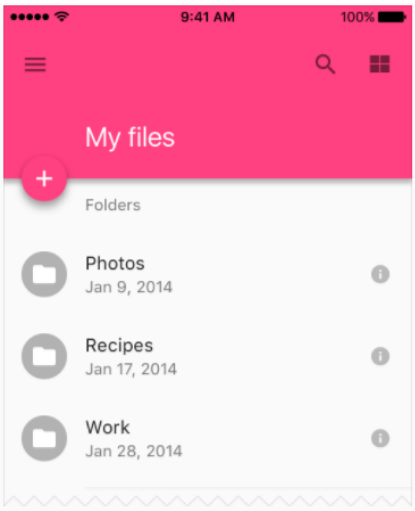
23.1 – Cor Direciona em botão ação



23.2 – Cor enfoca no *switch*



23.3 – Cor secundária botão de ação



23.4 – Erro uso cor primária

Fonte: Material Design Color, 2017

2.4.1.2 Textos

Qualquer texto mostrado na interface gráfica, deve ser legível comparado com o fundo no qual ele está inserido. Textos pretos por exemplo, não podem ficar sobre um fundo preto, pois assim não existiria contraste suficiente para a sua visualização. Recomenda-se que textos com cores escuras, devem estar sobre fundo com tons mais claros, enquanto textos em tons claros, devem estar sobre fundo escuros. O grau de opacidade do texto, depende no tom da cor do texto (escuro/claro). As Tabelas 17 e 18, definem a opacidade para o tipo do texto, relativo ao seu tom.

Tabela 17 - Opacidade texto tons escuros

Contexto	Opacidade (%)
Texto principal	87
Texto secundário	54
Texto desabilitado ou *hints*	38
Divisores	12

Fonte: Material Design Color, 2017

Tabela 18 - Opacidade texto tons claros

Contexto	Opacidade (%)
Texto principal	100
Texto secundário	70
Texto desabilitado ou *hints*	50
Divisores	12

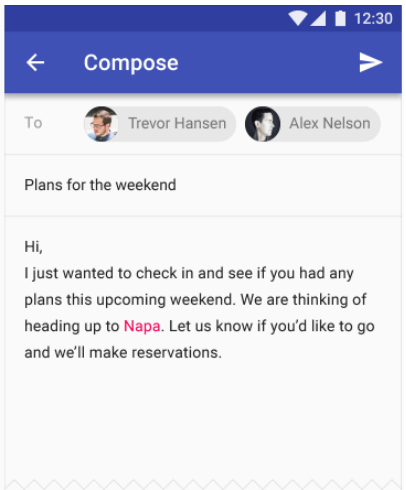
Fonte: Material Design Color, 2017

Utilizando a opacidade correta para os textos implica em aplicar o nível hierárquico de importância para os textos, aumentando a proeminência visual nos elementos mais ao topo da hierarquia. Como por exemplo, um texto que

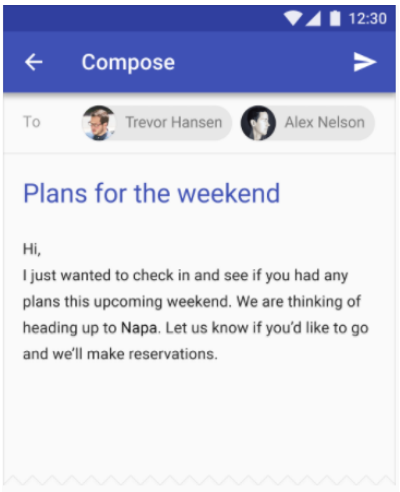
representa um *link*, ou um pedaço de texto que foi selecionado por ação do usuário.

Em textos não se deve utilizar cores com alta taxa de brilho para textos longos. Já a recomendação para a utilização de textos coloridos em textos com fundos coloridos, é restrita a apenas pequenas porções, que devem destacar a importância do texto. A Figura 24 demonstra exemplos de como a escolha correta da cor, pode contextualizar e destacar um pedaço do texto. No exemplo 24.1 é observada a utilização da cor secundária como destaque para um *link* presente no texto. No exemplo 24.2 a cor primária foi utilizada para destacar o assunto da conversa. Já no exemplo 24.3, a cor secundária foi utilizada para destacar o título, bem como a caixa de texto selecionada. No exemplo 24.4 a cor secundária é utilizada para promover o destaque do texto sendo selecionado pelo usuário.

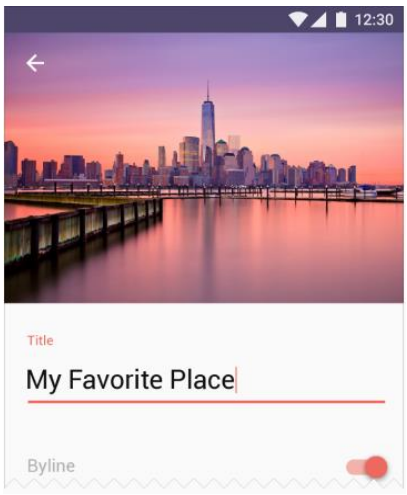
Figura 24 - Cores primárias/secundárias textos



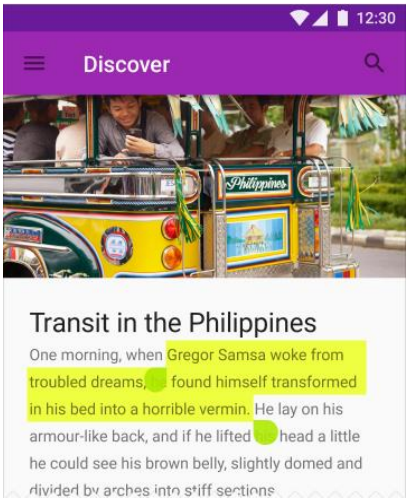
24.1 – Destaque *Link*



24.2 – Destaque assunto conversa



24.3 – Destaque titulo



24.4 – Destaque seleção

Fonte; Material Design, 2017

Desenvolver interfaces para dispositivos móveis é um processo desafiador, portanto utilizar um modelo de desenvolvimento pautado em etapas bem definidas, focado em entregar uma experiência de utilização adequada, aumenta a competitividade do mesmo dentro do seu mercado de atuação. A seleção de cores da interface do aplicativo tem grande importância na entrega de uma boa UX, dado que as cores dos elementos pode auxiliar no entendimento do contexto e na utilização do aplicativo. Harmonizar as cores dos elementos trás uma atratividade para a utilização do aplicativo que provoca uma experiência positiva por parte do usuário. A utilização dos padrões de cores do Material Design auxilia na obtenção de *interfaces* gráficas em conformidade com as *interfaces* mais modernas e utilizadas na plataforma Android, aumentando assim a possibilidade de se promover uma boa experiência e uma boa usabilidade para o usuário final.

3 ESTADO DA ARTE

Neste capitulo é levantado o estado arte atual de pesquisas relacionadas a unidades instrucionais focadas no ensino de computação por meio da programação de aplicativos, utilizando a ferramenta App Inventor, e que incluam conceitos de UX *design* no ensino fundamental II. A análise do estado da arte foi realizada seguindo o método de revisão sistemática definido por Kitchenham (2004).

3.1 Definição do protocolo de revisão

A revisão sistemática da literatura realizada tem como objetivo analisar e sintetizar a literatura existente sobre as unidades instrucionais que ensinam computação por meio da programação de um app de autoria própria e se possível, que inclua conceitos de UX *design* no ensino fundamental II com o App Inventor.

A busca a literatura existente é feita via Google Scholar, por ser uma ferramenta de busca aberta e que abrange uma gama ampla de fontes de bibliotecas digitais e bases de dados neste domínio (IEEEXplore, ACM Digital Library). São considerados somente artigos acessíveis via o Portal CAPES [9].

São consideradas publicações com data superior a 2010, data da qual o ambiente de desenvolvimento App Inventor foi lançado. Só são selecionadas publicações na língua Português e Inglês.

[9] http://www.periodicos.capes.gov.br/index.php?option=com_pcontent&view=pcontent&alias=quem-participa&Itemid=101/

De acordo com a pergunta da pesquisa são definidos os termos de busca, sinônimos e traduções conforme apresentado na Tabela 19.

Tabela 19 – Termos de Busca

Termo	Sinônimos	Tradução Inglês
App Inventor	Desenvolvimento de app	*App inventor, app development*
Unidade Instrucional	Curso, aula, disciplina, oficina, ensinar	*Instructional unit, lesson, workshop, class, course, teaching*
Smartphone	Dispositivo móvel	*Smartphone, mobile device*
Ensino Básico	Ensino fundamental II	*Middle school*
User Experience	UX, experiência de usuário, ui design,	*UX, user experience, ui design,*
Computação	Ciência da computação, Programação	*Computing, computer Science, CS, Programming, coding*

Critérios de inclusão/exclusão. Os critérios para inclusão/exclusão são:

- São considerados somente Unidades Instrucionais voltadas ao desenvolvimento de apps

Critérios de qualidade. Os critérios de qualidade são:

- Priorizar tutoriais que ensinam a fazer um app de criação do aprendiz, e não um app pré-definido
- Priorizar unidades voltadas para o ensino básico (fundamental II), porém pela falta de material encontrado serão levados em consideração unidades instrucionais voltadas a outras faixas etárias incluindo o ensino médio.

- São considerados apenas artigos que apresentam um grau mínimo de detalhamento de uma unidade instrucional. Sendo assim, o artigo deve indicar o objetivo de aprendizagem, plano de ensino ou o sequenciamento de conteúdo.

3.2 Execução da Busca

A busca foi realizada em novembro de 2016, usando combinações dos termos presentes na Tabela 20. Foram analisadas diversas publicações das mais variadas fontes indexadas via buscas no Google Scholar .

Tabela 20 - Termos de Busca no google scholar

Termo de Busca	Quantidade de resultados da busca	Quantidade possíveis resultados relevante	Quantidade relevante
("Unidade Instrucional" OR "Curso" OR "aula", OR "disciplina" OR "oficina" OR "ensinar") AND ("App Inventor" OR "Desenvolvimento de aplicativos")2010...2016	285	4	3
("Unidade Instrucional" OR "Curso" OR "aula", OR "disciplina" OR "oficina" OR "ensinar") AND ("App Inventor" OR "Desenvolvimento de app") ("Computação" OR "Programação" OR "Ciência da computação") 2010...2016	91	1	0
("App Inventor" OR "Desenvolvimento de aplicativos") AND ("Smartphone" OR "celular" OR	190	2	1

"dispositivo móvel") AND ("User Experience" OR "Experiência de Usuário") 2010...2016			
("Instructional Unit" OR "lesson" OR "workshop" OR "class" OR "course" OR "teaching") AND ("App Inventor" OR "App Development") AND ("K-12" OR "middle school")2010...2016	734	8	4

Dentre todas as publicações encontradas foram selecionadas as 50 primeiras ocorrências para análise. Os demais resultados não foram considerados por não atenderam alguns dos critérios de inclusão, como por exemplo, artigos que não focavam o ensino de computação utilizando o app inventor, ou artigos que focavam mais em um relato de experiência na aplicação de uma UI, e não focava na UI em si. Alguns artigos apresentavam boa qualidade na descrição do plano de ensino, usavam o App Inventor, porém a faixa etária não era compatível com a faixa etária foco do presente trabalho.

Aplicando os critérios de inclusão e exclusão foram encontrados 11 artigos relevantes relacionados a pergunta desta pesquisa. Existem diversos artigos relacionados a alguns dos critérios, como descrição das ferramentas do App Inventor, ou sobre ensino de computação para o ensino básico, ou artigos relacionados a UX *design*. Porém ressalta-se que não foram encontradas nenhuma publicação, que possua todos os critérios propostos pela UI da qual este presente trabalho será suporte. Ou seja, apresentar uma UI para ensinar conceitos de computação, por meio do desenvolvimento de apps, usando App Inventor, e que incluam conceitos de UX. Portanto foram levados em consideração também artigos em que é apresentada uma UI que ensine computação por intermédio da criação de apps usando App Inventor.

Também, foi considerado o material da competição *technovation challenge* que utiliza o App Inventor, na criação de apps sociais focando em meninas entre 10 e 18 anos.

Sendo assim, foram no total considerados 9 publicações como relevantes no levantamento do estado da arte.

3.3 Extração das informações e análise dos resultados

Na Tabela 21 são listados os artigos considerados relevantes.

Tabela 21 - Artigos relevantes

ID	Título	Referência
1	O ensino de programação para dispositivos móveis utilizando o MIT-App Inventor com alunos do ensino médio.	(FINIZOLA;HENNING, 2014)
2	Desenvolvimento de Aplicativos para android com uso do app inventor: uso de novas tecnologias no processo de ensino-aprendizagem em matemática	(DUDA;SILVA, 2015)
3	App Inventor for Android: Uma nova possibilidade para o ensino de lógica de programação	(GOMES;MELO, 2013)
4	Motivating K-12 Students learning fundamental computer Science conecepts with App Inventor	(CHATZINIKOLAKIS; PAPADAKIS, 2014)
5	Can Android app Invetor bring computional thinking to K-12	(MORELLI;LANEROLLE, 2011)
6	Using a Discourse-Intensive Pedagogy and Android's App Inventor for Introducing Computational Concepts to Middle School Students	(GROVER;PEA, 2013)
7	Using App Inventor in a k-12 Summer Camp.	(WAGNER *et al.* , 2013)
8	Using App Inventor 2 in A Summer Programming Worskhop: Improvements Over Previous Years	(A-GHAMDI ;AL- RAJHI, 2016)
9	Curriculum technovation	(Technovation, 2016)

Todos os artigos relevantes atendem os requisitos da seção 3.1 do presente trabalho. Foram extraídas informações sobre 3 tópicos: **contexto de aplicação, objetivos de aprendizagem da UI, e grau de suporte da UI**. Para cada tópico, foram extraídas as seguintes informações:

- Contexto de Aplicação das UIs:
    - Faixa etária: Representa a faixa etária do público alvo da UI.
    - Tipo de Evento: Se a UI foi aplicada em formato de curso, workshop, aulas.
    - Tamanho turma: Número total de aprendizes que participaram da UI.

- Objetivos de Aprendizagens da UI:
    - Objetivo de aprendizagem: Informa se a UI possuía conceitos de programação e de computação, e/ou conceitos de UX *Design*
    - Tipo Aplicativo: Informação sobre o tipo de aplicativo que foi desenvolvido durante a UI.

- Grau de suporte da UI:
    - Metodologia de ensino: Compreende informações resumidas sobre como a UI foi conduzida, e como os objetivos de ensino foram alcançados.
    - Material Instrucional: Caracteriza e localiza materiais de apoio que foram usados durante a aplicaçãoda UI.

- Língua: Indica a linguagem em que os materiais de apoio estão escritos.

Os dados extraídos são representados nas Tabelas 22 a 24

Tabela 22 - Contexto de Aplicação das UIs

ID	Faixa etária	Tipo de evento	Tamanho da turma
1	Ensino médio	Curso de curta duração(4 encontros de 3 horas cada)	20
2	Ensino médio	Curso de longa duração(2 semestres)	5
3	Ensino médio	Curso de longa duração (1 semestre)	Não informado
4	Ensino médio (13 anos a 16 anos)	Curso de curta duração (2 encontros de 4 horas cada)	35
5	Ensino médio e professores	Curso de média duração (6 semanas)	Não informado
6	Ensino fundamental II	Oficina de duração de 1 dia	7
7	Ensino médio	Oficina de 1 semana	Não informado
8	Ensino fundamental e médio	Oficina de 1 semana	16
9	Meninas entre 10 a 18 anos	Programas de 12 semanas, ou 20 semanas	Não informado

Tabela 23 - Objetivos de aprendizagem das UIs

ID	Objetivo de Aprendizagem		Tipo aplicativo	
	Programação e Conceitos básicos de computação	DT/UX Design	Pré -definido	Criação própria
1	x	-	x	
2	x	-		x
3	x	-	x	

4	x	-	x	
5	x	-	x	
6	x	-	x	x
7	x	-	x	x
8	x	-		
9	x	x	x	x

Tabela 24 - Grau de suporte das UIs

ID	Metodologia de Ensino	Material Instrucional	Língua
1	Primeiro módulo de apresentação da ferramenta, e a cada novo módulo novos conceitos eram apresentados crescendo em complexidade	Inspirado pelo utilizado no portal do app inventor, mas não disponibilizado	Português
2	Apps matemáticos desenvolvidos seguem etapas conforme: Etapa 1. Pesquisa = Tema é proposto, e alunos devem pesquisar sobre ele. Etapa 2. Arquitetura = Criação do app para execução dos cálculos propostos Etapa 3. Revisão = Professor revisa possíveis erros na parte matemática do app Etapa 4. Consolidação = apps validados em questão de estrutura e funcionalidades	Disponível os tutoriais completos em: www.ifdroid-irati.blogspot.com Licenças sobre direitos autorais não informado, necessário consultar caso necessário.	Português
3	Módulo 1. Introdução = Introdução do curso Módulo 2. App Inventor = Explicação e desenvolvimento Módulo 3. Testes = Etapa de validação dos apps Módulo 4. Publicando = Etapa de como publicar o app	Não informado	-
4	Alunos trabalharam em pares, e foram encorajados a trabalharem por conta própria, escolhendo quais exercícios do material fazer. O erro era encorajado, e também a solução do mesmo pela colaboração.	Desenvolvido pelos professores. Material completo em forma de livro (Uso sobre licença do tipo - CC-BY-AS). isponível em: http://www.sepchiou.gr/inde	Grego

		x.php/yliko/86-appinvcodeclub	
5	Primeiras 4 semanas alunos estudaram por conta própria usando o material disponível do AI. Após esse período os estudantes ensinaram os "professores" conceitos do App Inventor. No final, professores e alunos trabalharam juntos para desenvolver o último aplicativo do curso.	Tutoriais do AI e google	Inglês
6	Na parte da manhã o workshop foca em aspectos introdutórios e conceitos básicos de programação, exemplos de apps prontos foram feitos de forma coletiva. Na parte da tarde, os conceitos são colocados em práticas em apps de autoria própria	Usado tutoriais de exemplos de WOLBER, 2011.	Inglês
7	Utilizou nas primeiras aulas exemplos que foram se tornando mais complexos com o passar dos dias, usando tutoriais prontos e executados de forma colaborativa. Como um projeto final, foram feitos aplicativos próprios desenvolvidos em times	Usado tutoriais de exemplos de WOLBER, 2011.	Inglês
8	Em grupos separados de 4 alunos por grupo, foram feitos apps pré definidos, a cada novo módulo do workshop, sendo associados a conceitos cada vez mais complexos, de computação, como variáveis, loops, e ifs. Ao final, um aplicativo próprio era desenvolvido, e submetido em uma "competição" interna do workshop	Não divulgado	-
9	Tutoriais dos mais diferentes níveis, contextualizando o uso do app inventor para criação de aplicativos pré definidos. Conceitos que servirão de suporte no desenvolvimento dos apps proprietários para a competição	http://www.technovationchallenge.org/curriculum/	Inglês

3.4 Discussão

A revisão do estado da arte, demonstra que existem várias pessoas aplicando unidades instrucionais com o intuito de ensinar programação

utilizando o *software* do App Inventor como NPE. No mesmo ano de lançamento do ambiente de programação em 2010, já foram criadas unidades instrucionais em outros idiomas. Nos últimos anos houve uma crescente no número de UIs, tanto internacionalmente quanto dentro do Brasil. Com a popularização do App Inventor foi observada inclusive criação de tutoriais e *workshops* em lugares menos comuns, como Grécia e Arábia Saudita.

Dentre as UIs encontradas existe uma variação em duração de curso, podendo variar entre 1 dia a até dois semestres. O contexto de aplicação das UIs são demonstrações em sala de aula utilizando materiais instrucionais padrões como *slideshows,* demonstrações em quadro ou computador. Muitas focam no desenvolvimento de apps pré-definidos, e que estão disponíveis *online.* Ao verificar a evolução da turma, porém, algumas unidades fomentam a criação de aplicativos próprios como método final de avaliação. Os objetivos de aprendizagem englobam conceitos de programação básicos, como *loops* e condicionais. A estratégia instrucional utilizada é uma mistura de estratégia direta, onde o instrutor demonstra de forma retórica e os aprendizes reproduzem, com estratégia interativa, onde de forma colaborativa, os problemas são discutidos e solucionados em grupos formados pelos aprendizes. Existe porém uma exceção, em uma das UI a estratégia instrucional foge do padrão apresentado pelas outras e foi considerada interessante. Nessa estratégia os alunos tiveram um mês para trabalhar de forma autônoma, estudando de forma autodidata, e relatando a experiência respondendo questionários. Em uma segunda etapa do curso, os instrutores eram associados aos aprendizes, que tinham como missão ensinar os instrutores. Essa inversão de papeis se mostrou eficaz, e motivou os

aprendizes e instrutores, que ao final, produziram aplicativos próprios, trabalhando de forma conjunta.

Apesar de haver uma comunidade ativa desenvolvendo e aplicando UIs com o App Inventor, não existe nenhuma que apresente conceitos de UX de forma satisfatória, e que esteja alinhada com os últimos padrões de interfaces para Android, seguindo a linguagem visual Material Design.

3.5 Ameaças à validade da Revisão da Literatura

Alguns fatores podem ameaçar a validade dos resultados desta revisão sistemática da literatura, como por exemplo, a possibilidade de não ter sido encontrado algum artigo que apresentasse uma unidade instrucional relevante para o contexto do presente trabalho. Para mitigar o risco dessa ameaça, foi utilizada uma ferramenta de busca bastante abrangente, o Google Scholar. Também foram utilizados sinônimos das palavras chaves, tanto na língua inglesa, quanto no Português. Buscando variações de combinações possíveis em uma mesma *search string,* subentende-se que foi reduzido o risco de algum artigo importante não ter sido detectado e analisado como estado da arte.

O processo de extração de informações foi realizado pelo autor deste trabalho, de forma cuidadosa e sistemática, e acompanhado por pesquisador sênior, tentando aprovar a acurácia das informações fornecidas.

4 SUPORTE AO ENSINO DE DESIGN COM APP INVENTOR

Com o objetivo de fomentar o ensino de computação para estudantes do ensino básico a iniciativa Computação na Escola (http://www.computacaonaescola.ufsc.br/) desenvolve diferentes unidades instrucionais. Atuando atualmente em três domínios diferentes (criação de aplicativos, robótica e criação de jogos) são desenvolvidas unidades instrucionais voltadas a facilitar o ensino de computação de acordo com diretrizes curriculares adequadas para as faixas etárias do ensino básico.

No âmbito de criação de aplicativos para celulares foi desenvolvida a unidade instrucional “Caça Mosquito” (TRILHA *et al.* 2017), na qual os participantes se familiarizam com conceitos básicos de programação, como condições, *loops*, variáveis. Utilizando esses conceitos um jogo de celular pré-definido é desenvolvido utilizando a ferramenta App Inventor. Dando continuidade a essa unidade instrucional foi criada pela iniciativa computação na escola também uma unidade instrucional com maior carga horária que abordar conceitos mais avançados em relação ao tópico de computação e criação de aplicativos dos mais variados escopos, ou seja, cada participante idealiza seu próprio app.

Para tal foi desenvolvida a unidade instrucional “Faça o seu app” que além de ensinar a programação de apps também aborda conceitos de *Design Thinking* (DT) e *UX design*. Essa unidade proporciona a criação de um app autoral que será desenvolvido utilizando a ferramenta App Inventor. Utilizando técnicas de *design thinking* os participantes são levados a identificação de uma problemática que deve ser resolvida com a utilização de um aplicativo para

celular. Os mesmos então deverão utilizar conceitos de UX *design* para prototipação de telas, utilizando um processo iterativo de engenharia de *software* similar ao utilizado em empresas de desenvolvimento de apps. O público alvo principal da unidade instrucional são estudantes do ensino básico (10-14 anos). O objetivo da unidade é ensinar conceitos básicos de computação relacionados à práticas da computação e programação, colaboração e pensamento computacional por meio da criação de apps para celulares, de forma interdisciplinar com disciplinas do ensino básico alinhados com o currículo referência (CSTA, 2016). A Tabela 25 detalha os objetivos de aprendizagens da unidade “Faça o seu app”.

Tabela 25 - Objetivos de aprendizagem UI “Faça o seu app”

Objetivos de aprendizagem	**Relação com currículos de referência (CSTA, 2016)**	
	Conceitos de framework	**Prática de framework**
Solicitar e integrar o feedback, conforme apropriado, para desenvolver ou aperfeiçoar um programa.	Algoritmos e programação	Colaboração. 2-A-2-1 (CSTA – Grade 6-8, 2016)
Design, desenvolvimento e apresentação de artefatos computacionais como aplicações moveis que abordam problemas sociais de forma independente e colaborativa.	Algoritmos e programação	Criando artefatos computacionais. 2-A-5-5 (CSTA – Grade 6-8, 2016)
Desenvolver programas, de forma independente e colaborativa, que incluem sequências com loops aninhados e ramos múltiplos.	Algoritmos e programação	Criando artefatos computacionais. 2-A-5-6 (CSTA – Grade 6-8, 2016)
Criar variáveis que representam diferentes tipos de dados e manipular seus valores.	Algoritmos e programação	Criando artefatos computacionais. 2-A-5-7 (CSTA – Grade 6-8, 2016)
Definir e usar procedimentos que escondam a complexidade de uma tarefa e que podem ser reutilizados para resolver tarefas semelhantes.	Algoritmos e programação	Desenvolvendo e usando abstrações. 2-A-4-8 (CSTA – Grade 6-8, 2016)
Decompor um problema em	Algoritmos e programação	Reconhecendo e definindo

partes e criar soluções para cada uma das partes.		problemas computacionais. 2-A-3-9 (CSTA – Grade 6-8, 2016)
Usar um processo de design iterativo (por exemplo, definir o problema, gerar ideias, construir, testar e melhorar soluções) para resolver problemas, de forma independente e colaborativa.	Algoritmos e programação	Testando e refinando. 2-A-6-10 (CSTA – Grade 6-8, 2016)
Criar o design, desenvolver e implementar um artefato de computação que responde a um evento (por exemplo, robô que responde a um sensor, aplicativo para celular que responde a uma mensagem de texto, um objeto gráfico que responde a uma transmissão).	Algoritmos e programação	Criando artefatos computacionais. 3A-A-5-4 (CSTA – Grade 9-10, 2016)
Usar técnicas de pesquisa e design centradas no usuário (por exemplo, pesquisas, entrevistas) para criar soluções de software	Algoritmos e programação	Criando artefatos computacionais. 3A-A-5-5 (CSTA – Grade 9-10, 2016)
Projetar algoritmos usando sequência, seleção e iteração.	Algoritmos e programação	Reconhecendo e definindo problemas computacionais. 3A-A-3-10 (CSTA – Grade 9-10, 2016)

Os objetivos de aprendizagem são abordados em um formato sugerido de 12-15 aulas de 45 minutos cada, divididas em 2 módulos. O primeiro módulo é voltado para conhecer a ferramenta App Inventor. Neste módulo os participantes fazem o app “Caça o Mosquito” (TRILHA, 2016) para se familiarizar com o funcionamento da ferramenta App Inventor. O segundo módulo possui o enfoque de desenvolver um app próprio utilizando técnicas de DT UX. A Tabela 26 demonstra a sequência da unidade instrucional.

Tabela 26 - Sequencia da UI “faça o seu app”

Aula	**Objetivo**
1	Oficina “caça mosquito”

2	Oficina “caça mosquito”
3	Oficina “caça mosquito”
4	Descoberta e definição de problema
5	Descoberta e definição de problema
6	Ideação de uma solução (app)
7	Ideação de uma solução (app)
8	*Design* de interface de baixa fidelidade
9	Programação e teste protótipo de app
10	Programação e teste protótipo de app
11	*Design* gráfico das interfaces (App Inventor)
12	Ajustes finais
13 (opcional)	Avaliação
14	Compartilhamento do app
15	Compartilhamento do app

Dado que a unidade instrucional “faça seu app” possui ênfase em UX *design* o presente trabalho visa identificar e desenvolver melhorias na ferramenta App Inventor de forma que o *software* App Inventor possua um melhor suporte ao atingimento dos objetivos de aprendizagem da unidade instrucional supracitada.

Um dos domínios que engloba uma boa experiência de usuário é o *design* das cores do aplicativo. Por meio da escolha correta de cores agrega-se atratividade visual e até despertar sentimentos específicos nos usuários de um app (GONÇALVES, 2014). No contexto da ferramenta App Inventor existem duas formas de se selecionar a cor para um elemento gráfico. Uma é por meio dos seletores de cores das propriedades de cada elemento, como por exemplo, a cor de fundo ou de texto de um botão.

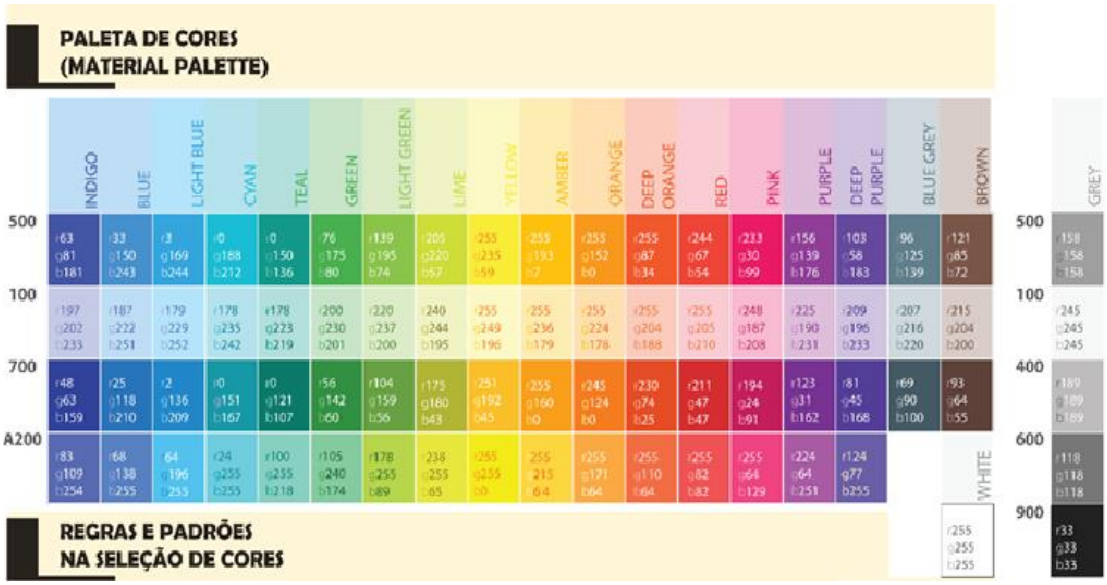
Ao clicar no seletor ele mostra uma listagem de cores *default* para uso. Nesse formato, existe uma cor para cada tom “principal”, ou seja, uma possibilidade de vermelho, uma de azul, uma de verde entre outras. Outra forma de se selecionar uma cor para um elemento, é por meio de um editor de

cores customizadas onde o usuário apresenta o código RGB associado a cor desejada. Utilizando essa forma porém, requer do usuário um conhecimento mais avançado em tópicos de computação, já que é preciso entender o conceito de formação de cores RGB, além de saber o código necessário para criar as cores customizadas. Essa abordagem é inviável para o público alvo da unidade instrucional “faça o seu app”, já que lhes faltaria o conhecimento necessário para realizar a operação. Sendo assim o público alvo da unidade instrucional são limitados a utilização das cores padrão do disponíveis no seletor de cores, sendo que as mesmas não estão de acordo com as orientações presentes no Material Design, *framework* de estilo visual mais utilizado de apps Android.

4.1 Modelo Conceitual

Portanto para que a ferramenta apresente um melhor suporte à unidade instrucional, faz-se necessária a modelagem de uma customização para a alteração das cores padrões disponíveis nos seletores de cores. Baseado nas orientações do Material Design é possível o desenvolvimento de uma paleta de cores com 19 categorias e subcategorias, dando uma ampla gama de possibilidades para o usuário. As cores presentes na paleta são cores tipicamente utilizadas em apps Android. A Figura 25 apresenta a paleta de cor criada.

Figura 25 - Paleta de cores Material Design



Observa-se que a paleta criada possui uma ampla quantidade de cores, portanto colocar todas em uma única listagem de seleção implicaria na perda de usabilidade da ferramenta. Assim, o fluxo correto para uma boa usabilidade seria editar os seletores de cores atuais da ferramenta para disponibilizar as cores principais (indicadas com o valor de saturação 500), além de adicionar um novo componente no qual as variações de tons para a cor principal selecionada são mostradas. Tal abordagem deve ser seguida para todo e qualquer elemento visual que possua propriedades de cor (cor de fundo, cor de texto, cor de item entre outras).

Na ferramenta App Inventor somente os menus “Interface de Usuário”, “Organização”, “Desenho e Animação”. “Mídia” e “Social” possuem componentes com algum tipo de seletor de cores. As Tabelas 27 a 31

decompõem a presença de cada tipo de seletor, para cada componente do menu.

Tabela 27 - Menu interface com usuário elementos com seletor de cores

Componente	Cor de fundo	Cor de texto	Cor de fundo do Item	Cor do texto do item	Cor de seleção	Cor a esquerda	Cor a direita
Botão	x	x					
Caixa de seleção	x	x					
Escolhe data	x	x					
Legenda	x	x					
Imagem							
Escolhe lista	x	x	x	x			
Visualizador de Listas	x	x			x		
Notificador	x	x					
Caixa de senha	x	x					
Deslizador						x	x
Lista suspensa							
Caixa de texto	x	x					
Escolhe hora	x	x					

Fonte: App Inventor, 2017

Tabela 28 - Seletor de cores menu organização

Componente	Cor de fundo
Organização horizontal	x
Horizontal scroll arrangement	x
Organização em tabela	
Organização vertical	x
Vertical scroll arrangement	x

Fonte: App Inventor, 2017

Tabela 29 - Seletor de cores menu mídia

Componente	Cor de fundo	Cor de texto
Camcorder		
Camera		
SeletorImagem	x	x
Sound		
SoundRecorder		
Speechecognizer		
TextoToSpeech		
VideoPlayer		

YandexTranslate		

Fonte: App Inventor, 2017

Tabela 30 - Seletor de cores menu desenho e animação

Componente	Cor de fundo	Cor de pintura
Bola		x
Pintura	x	x
Sprite imagem		

Fonte App Inventor, 2017

Tabela 31 - Seletor de cores menu social

Componente	Cor de fundo	Cor de texto
Escolhe contato	x	x
Escolhe email	x	x
Escolhe número de telefone	x	x
Ligação		
Compartilhamento		
Mensagem SMS		
Twitter		

Fonte: App Inventor, 2017

Realizar a alteração no seletor de cor e adicionar um novo seletor de tonalidades possibilita um melhor suporte à unidade instrucional, dado que conceitos de UX *design* seriam abordados com mais facilidade. As seções subsequentes detalham o levantamento de requisitos além da modelagem e implementação do modelo conceitual descrito na presente seção.

4.2 Requisitos

Conforme especificado na seção 4.1, são identificados os requisitos funcionais e não funcionais para a implementação da proposta do modelo conceitual no App Inventor.

Requisitos funcionais

R.F-1: O usuário deverá ser capaz de selecionar uma cor principal, de acordo com a paleta definida na seção 4.1.para todos os componentes gráficos presentes nas tabelas 26, 27, 28 e 29.

R.F-2: Para cada seletor de cores principais presente nas propriedades de um elemento, tais como cor de fundo, ou cor de texto, o sistema deve exibir uma propriedade análoga que apresente tonalidades secundárias para a cor selecionada, conforme a paleta definida na seção 4.1.

R.F-3: Ao trocar a seleção da cor principal o sistema deve atualizar a listagem de tonalidades secundárias da propriedade adequada. As tonalidades secundárias são definidas conforme paleta apresentada na seção 4.1.

R.F-4: As cores selecionadas dos componentes devem ser armazenadas, de forma que se usuário sair do App Inventor, ao abrir novamente o projeto as cores dos elementos devem permanecer as mesmas.

R.F-5: Todas as alterações de cores devem ser visualizadas no aplicativo do aparelho celular.

Requisitos não-funcionais

R.N.F-1: código fonte modificado/adicionado deve ser integrado na versão nb155 do App Inventor

R.N.F-2: Apenas códigos na linguagem java devem ser adicionados ao projeto.

R.N.F-3: Tratamentos de eventos oriundos da interface gráfica devem utilizar a biblioteca GWTEvent.

R.N.F-4: O *build* do aplicativo compilado para o celular (.APK) deve utilizar o buildServer do servidor *localhost.*

R.N.F-5: O padrão de usabilidade para a escolha de uma cor é de eficácia 90%, eficiência 3 segundos e com grau de satisfação bom (SUS > 75).

4.3 Contexto do funcionamento do App Inventor

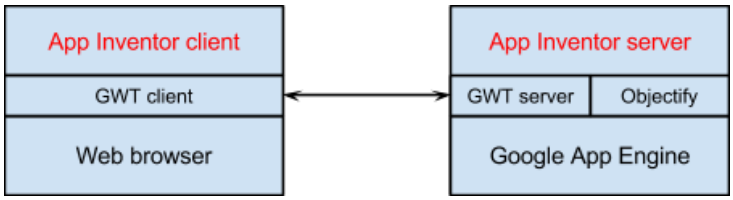
O App Inventor foi desenvolvido utilizando a linguagem de programação Java, e usa diferentes tecnologias nos seus múltiplos projetos. Uma das tecnologias utilizadas é o *framework* **G**oogle **W**eb **T**oolkit (GWT[10]) para a tradução do código do *frontend*. O GWT permite a criação e otimização de aplicações web, seu objetivo é abstrair a complexidade de desenvolvimento do *frontend*, fazendo com que o desenvolvedor não precise conhecer nuances do funcionamento dos diferentes navegadores, ou ainda conhecer a linguagem Javascript. Ele é *open source* e muitos produtos da Google utilizam GWT. Como por exemplo, o AdWords e o Google Wallet. O GWT provê uma gama alta de APIs Java que possibilitam a criação de aplicações AJAX em Java, que são então compiladas e otimizadas em

[10] www.gwtproject.org

código Javascript, que é compatível com todos os navegadores, inclusive navegadores de aparelhos móveis Android.

Outra tecnologia utilizada é o Google App Engine (https://cloud.google.com/appengine/) que é uma plataforma de *cloud computing* que permite que programas Java rodem e mantenham dados nos servidores do Google. Tanto o GWT quanto o GAE são desenvolvidos pelo Google, portanto trabalham bem juntos e formam a estrutura do App Inventor. A figura 30 mostra como a estrutura do App Inventor funciona. O cliente e o servidor são criados utilizando GWT, que converte o código de *frontend* para Javascript, esse código roda sobre a biblioteca GWT presente no navegador do computador do usuário. Já o *backend* roda na biblioteca do servidor como um serviço Google App Engine, e utiliza a API Objectify para persistir dados no sistema de arquivos distribuídos.

Figura 26 - Estrutura App Inventor



Fonte: App Inventor, 2017

Por ser *open source* o código fonte do App Inventor está disponível *online* e alterações podem ser feitas para customizações. Nas próximas sessões são discutidas as formas de configuração do ambiente de desenvolvimento, a estrutura básica do código fonte, além das alterações que foram feitas para contemplar o suporte a UI “faça o seu app”.

4.3.1 Configuração do ambiente de desenvolvimento

Para rodar o App Inventor localmente e executar mudanças no código fonte, é necessário a instalação de alguns *softwares* para executar a compilação da aplicação. Para a visualização e edição do código fonte, pode-se usar qualquer *Integrated development enviroment* (IDE) para Java. No presente trabalho foi utilizada a IDE Eclipse[11] e o sistema operacional Windows 10 por questões de familiaridade e gosto pessoal. O código fonte pode ser encontrado no repositório público do App Inventor no Github. A Tabela 32 representa os passos necessários para a configuração do ambiente de desenvolvimento.

Tabela 32 - Passos configuração ambiente de desenvolvimento

Ordem	**Instalar**	***Link***
1	Clonar o projeto do repositório	https://github.com/mit-cml/appinventor-sources
2	Git Client	https://git-scm.com/downloads
3	Ant versão mais atual	https://ant.apache.org/bindownload.cgi
4	JDK 1.7	http://www.oracle.com/technetwork/pt/java/javase/downloads/jdk7-downloads-1880260.html
5	Python	https://www.python.org/downloads
6	Android SDK versão mais atual	https://developer.android.com/studio/index.html
7	Plugin eclipse AppEngine	https://dl.google.com/eclipse/plugin/4.5 https://developers.google.com/eclipse/docs/install-eclipse-4.5

[11] https://eclipse.org/

Importante I: o App Inventor só consegue compilar localmente, se o JDK for a versao 1.7.

Importante II: a pasta que receberá o clone do projeto, não pode possuir espaços em seu nome. (Exemplo Incorreto: “Nome pasta projeto”. Exemplo Correto: “nome_pasta_projeto”).

Para compilar é necessário abrir um terminal (indicado o uso do Git Bash para usuários Windows) e realizar a execução dos seguintes comandos, em ordem:

1) ant clean (no diretório appinventor)
2) ant MakeAuthKey
3) ant
4) mkdir bin (no diretório appinventor-sources)
5) Abrir o Eclipse
6) File>New>Java Project
7) Em "Use default Location" selecionar a pasta appinventor-sources e NEXT
8) Em "Default Output Folder": colocar a pasta "appinventor-source\bin"
9) Em "Order and Export" mover as duas ocorrências de "android.jar" para o final da lista

Ao concluir todos os passos da configuração de ambiente, e da compilação da aplicação, é necessário rodar o servidor da aplicação localmente para que seja disponibilizado o sistema. Para rodar o servidor executar o comando:

C:**caminhodapastadeinstalaçãodoappengine**\bin\dev_appserver--port=8888--address=0.0.0.0C:**caminhodapastaclonedoprojeto** \appinventor\appengine\build\war

Exemplo:

C:\appengine\appengine-java-sdk-1.9.34\bin\dev_appserver--port=8888 --address=0.0.0.0 C:\workspace\AppInventorCnE\appinventor\appengine\build\war

Com o servidor rodando, para acessar o App Inventor abrir qualquer navegador, e digitar a *url*: http://localhost:8888

4.3.2 Estrutura código fonte App Inventor

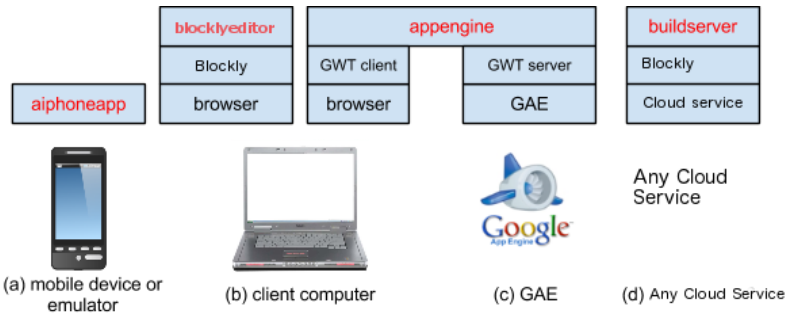
O código fonte do App Inventor é composto de 9 subprojetos, cada qual responsável por uma camada diferente para criação/documentação da aplicação web:

- **aiphoneapp**: interpretador que roda no dispositivo móvel ou no emulador.
- **Aiplayapp**: outra versão do interpretador que roda no dispositivo móvel, esse diretório é responsável pelo chamado MIT App Inventor Companion
- **Appengine**: a aplicação GWT que provê o código Javascript para o navegador mostrar o *frontend* da aplicação. Também é responsável por requisitar compilações para o buildserver, e buscar e salvar projetos do usuário.
- **Blockyeditor**: código que representa a parte de edição de blocos (montagem da lógica do app) do App Inventor,

- **Buildserver**: servidor/*servlet* http que transforma o .zip em um .apk correspondente.
- **Common**: constantes e classes utilitárias usadas pelos outros subprojetos.
- **Components**: projeto que possui toda a lógica que dá suporte aos componentes da aplicação, tal como os componentes de cada menu, e as propriedades dos elementos, como tamanho, cor, estilo.
- **Docs**: documentação.
- **Lib**: bibliotecas externas ao projeto e que servem de apoio como JUnit.

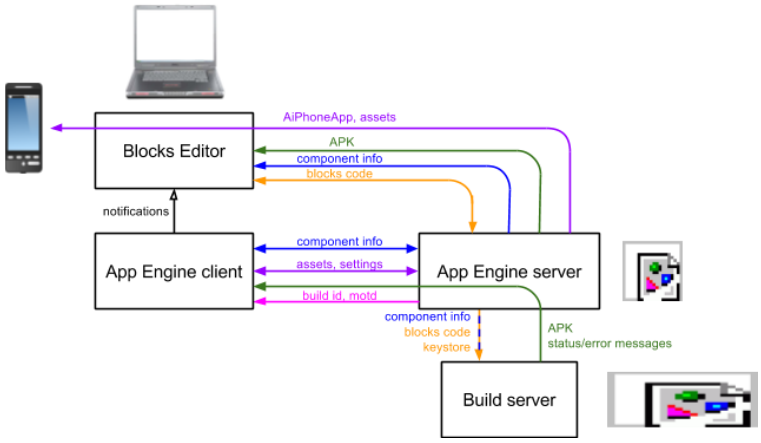
A Figura 27 mostra uma representação em alto nível do contexto em que cada subprojeto roda. As partes em vermelho representam a camada interna do App Inventor enquanto partes em preto representam infraestruturas externas auxiliares ao App Inventor. Já a Figura 28 mostra a esquematização da comunicação em tempo real entre os diferentes subprojetos.

Figura 27 - Esquematização código fonte



Fonte: App Inventor, 2017

Figura 28 - Esquema comunicação em tempo de execução App Inventor



Fonte: App Inventor, 2017

4.3.3 Implementação

No App Inventor cada opção presente em um dos 11 menus da Paleta é um componente. Cada componente possui uma lista de propriedades que podem ser configuradas. Como por exemplo, no menu Interface de usuário, existe a propriedade botão que possui a propriedade de cor do fundo.

Para o contexto da inicialização do App Inventor, precisa-se de uma forma de saber a configuração inicial para cada um dos componentes. Ou seja, quais as propriedades cada componente possui e quais os valores *default* de cada. Para tal, existe um arquivo em formato *JavaScript Object Notation* (JSON), chamado **simple_components.json** que possui o esquema de configuração que é compilado em tempo de execução, e por meio das anotações feitas no código fonte são gerados os componentes corretos para cada parte do sistema. Portanto, para a criação de uma nova propriedade ou

componente, esse arquivo precisa ser modificado. A Tabela 33 mostra a semântica necessária para a configuração de um componente no arquivo de inicialização.

Tabela 33 - Configuração de componente

Chave	Significado
Name	Nome do componente
version	Número que representa a versão do componente. Incrementada quando uma nova propriedade é adicionada
categoryString	Em qual sessão do menu o componente tem que aparecer
showOnPalette	Se o componente deve aparecer na Paleta ou não
nonVisible	Se deve aparecer na parte do Designer que representa a tela do celular. Verdadeiro para elementos visuais, e falso para elementos não visuais
iconName	O caminho para o ícone do componente
properties	Listagem contendo o nome, tipo e valor *default* de cada propriedade do componente

Fonte: App Iventor, 2017

Cada propriedade de um componente possui uma classe .java que lhe atribui as características e auxilia na montagem do elemento que será enviado ao *frontend.* No pacote **com.google.appinventor.client.editor.youngandroid.properties** estão todas as classes que configuram uma propriedade. Todas são nomeadas segundo o padrão *nomeExemplo*PropertyEditor. As próximas seções detalham cada um dos arquivos que foram modificados ou criados, para contemplar todos os requisitos definidos na seção 4.2.

4.3.3.1 Classes editadas

Para todas as propriedades que possuíam algum tipo de seletor de cores, foi necessária a adição de instrução que invoca a mudança do comportamento esperado. Por exemplo, ao selecionar uma nova cor de fundo de um botão é necessário chamar o método que realiza a alteração da mudança de cor do fundo. Portanto foi adicionada a instrução setBackgroundProperty na classe ButtonBase.java para realizar a troca da cor de fundo. A Tabela 34 mostra detalhes sobre classes específicas que foram modificadas:

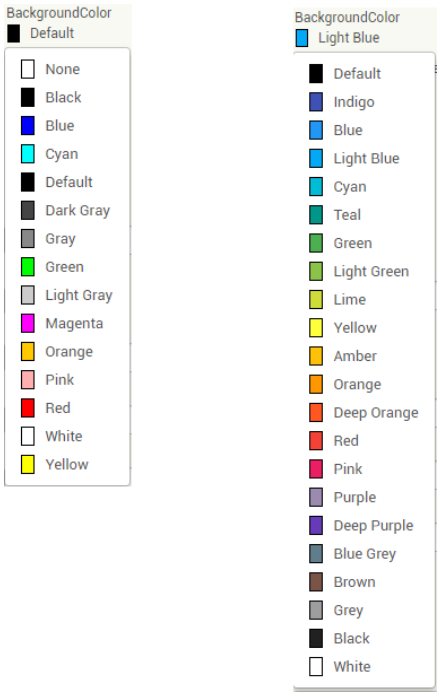
Tabela 34 - Lista classes editadas

YoungAndroidColorChoicePropertEditor.java
Dado que um dos requisitos definidos na seção 4.2 é a possibilidade de seleção das cores primárias do Material Design, foram substituídas as cores da propriedade que é classe responsável pela listagem de seleção de cores. Portanto essa classe foi editada para construir um *array* de cores compatíveis com as cores primárias do Material Design.
OdeMessages.java
Classe auxiliar que cria strings para construção dos elementos visuais, como o titulo de uma propriedade, ou o nome de uma cor. Por meio de anotações foram adicionadas nomenclaturas para todas as cores do Material Design.
PropertiesUtil.java
Classe responsável por iniciar as propriedades dos componentes lidos do simple_properties.json, e instancia-las. Portanto foi necessário modificar o método createPropertyEditor para quando receber por parâmetro uma propriedade do tipo de seleção de tonalidades de cor, chamar o construtor da classe que criará a propriedade YoungAndroidColorGradientChoicePropertyEditor (nova classe criada que será detalhada na sessão 4.3.3.2).
PropertyTypeConstants.java
Classe auxiliar para retorno de string constantes, utilizadas em diferentes lugares da aplicação. Adicionado constante com o nome da propriedade nova sendo criada.
MockVisibleComponents.java

Classe abstrata que possui métodos que rastreiam mudanças em propriedades e as processam. Por exemplo, quando uma das cores é selecionada, essa classe é chamada para invocar o método correto para realizar a substituição da cor do elemento. Método onPropertyChange foi modificado para quando uma nova cor de fundo for selecionada, disparar um evento customizado para a mudança da listagem de tonalidades presentes na propriedade BackgroundColorGradient ou TextcolorGradient (propriedades sendo criadas e que representam a variação de tonalidades modeladas na seção 4.2).

Com a edição das classes foi cumprido o requisito R.F-1, mudança das cores *default* para as primárias do Material Design. Conforme demonstrado na Figura 29 onde a Figura 29.1 mostra o menu de cor do fundo antes da alteração do código, e a Figura 29.2 o menu de cor do fundo após edição. Todos os outros seletores de cores (cor do texto, cor do item, cor do *slider*) também foram alterados. Sendo assim, ao final dessa etapa as cores padrões foram alteradas, e os elementos gráficos do ambiente web já utilizam essas cores.

Figura 29 -Comparação menu cor do fundo



29.1- Menu Antigo 29.2 – Menu Novo

A edição das classes também prepara o código para a criação de um evento customizado, disparado na seleção de uma nova cor primária de algum seletor de cor.

4.3.3.2 Classes criadas

Com a mudança das cores primárias da listagem do editor da propriedade seletor de cor, foi possível criar novas classes que configurassem uma nova propriedade que mostrasse variações de tonalidades da cor primária

selecionada. Também foram criadas classes auxiliares disponíveis no pacote **com.google.appinventor.client.editor.youngandroid.properties.gradient.color.utils**

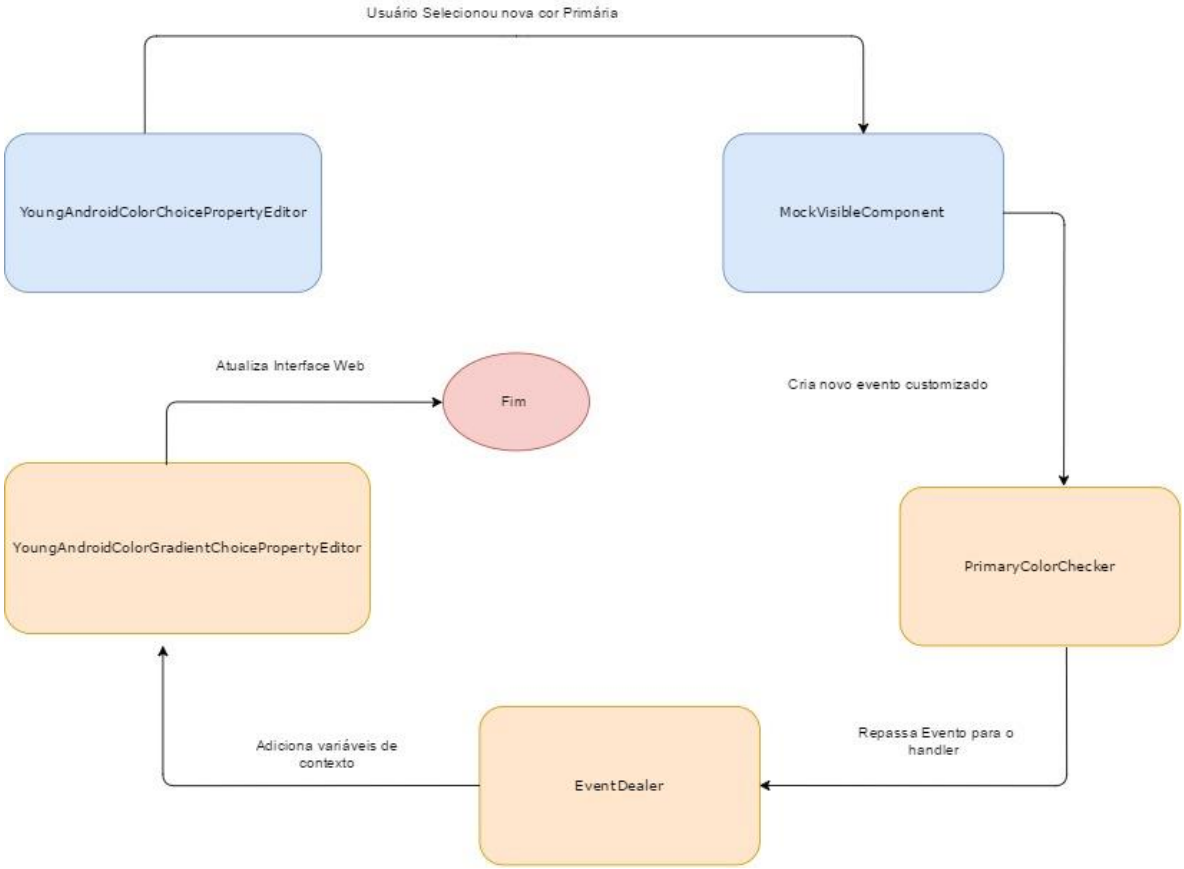
A seguir são detalhadas as classes criadas para cumprimento do R.F-2 na Tabela 35:

Tabela 35 – Lista classes criadas

PrimaryColorChoiceChangeHandler.java
Interface que estende um EventHandler, define a assinatura do método que irá tratar o evento customizado PrimaryColorChoiceChangeEvent
PrimaryColorChoiceChangeEvent.java
Classe estende um GWTEvent, e sobrescreve seu método dispatch. Responsável por configurar os atributos do evento customizado a ser despachado. Atributos são a cor primária selecionada e o nome da propriedade que foi alterada (Por exemplo, a cor do texto de um botão).
PrimaryColorChecker.java
Classe que implementa a interface abstrata HasHandlers e é responsável por disparar o evento usando o gerenciador de handlers.
EventDealer.java
Classe que implementa o PrimaryColorChoiceChangeHandler. Responsável por repassar no evento as variáveis de contexto necessárias para a atualização das cores dos tons.
YoungAndroidColorGradientChoicePropertyEditorHelper.java
Classe criada para auxiliar na obtenção do array de opções de tons da classe YoungAndroidColorGradientChoicePropertyEditor. Classe possui um HashMap onde a chave é o hexadecimal RGB de cada uma das cores primárias, e o valor é o array de tons. Assim, é possível facilmente localizar o conjunto de tons necessários para a atualização da propriedade
YoungAndroidColorGradientColorProvider.java
Classe abstrata que define constantes utilizadas nas classes criadas. Criada apenas com o propósito de melhor organização de código.
YoungAndroidColorGradientChoicePropertyEditor.java
Classe responsável por instanciar e configurar uma nova propriedade para algum elemento gráfico que tenha um seletor de tonalidades de cores.

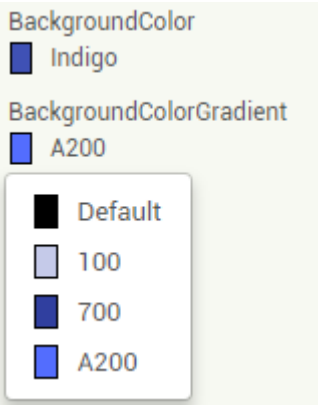
Em uma abstração de alto nível as classes criadas e suas responsabilidades podem ser representadas pela Figura 30. Representações em azul indicam classes que já existiam no sistema, enquanto as classes em amarelo foram criadas dentro do escopo do presente trabalho.

Figura 30 - Fluxograma evento PrimaryColorChangeEvent



Após a implementação das novas classes e suas responsabilidades foi possível visualizar e utilizar para cada propriedade de cor primária, uma propriedade “gradiente” com opções de tonalidades, conforme demonstrado na Figura 31.

Figura 31 -Menu de tonalidades



Nessa etapa de desenvolvimento o ambiente Web possuía o comportamento esperado de todas as funcionalidades. Faltava a implementação de códigos para que as novas cores aparecessem nos elementos gráficos no dispositivo móvel. Sem essa implementação, os aplicativos não abrirão no dispositivo móvel.

Todo elemento visual do ambiente Web, ou seja, criado pelo *backend* possui uma classe que define seus atributos. Essa classe possui o padrão de nome *MocknomeElemento*.java, como por exemplo, o MockButtonBase.java cria um botão na interface web. O dispositivo móvel funciona da mesma forma, porém suas classes são compiladas em momento de execução e ficam no pacote **com.google.appinventor.components.runtime** portanto fez-se necessária a alteração das classes dos elementos correspondentes para que o celular também mostrasse as alterações realizadas pela nova propriedade criada. Há próxima etapa no desenvolvimento, é internacionalizar os termos para que os mesmos apareçam traduzidos no ambiente web de acordo com a linguagem escolhida pelo usuário.

4.3.3.3 Internacionalização

As classes responsáveis pela internacionalização do sistema são as classes *TranslationDesignerPallete e TranslationComponentParams.java* nessas são traduzidos os termos do menu esquerdo e seus componentes, e das propriedades de cada componente, respectivamente. A classe *TranslationComponentParams.java* traduz em tempo de execução os termos das propriedades, utilizando a notação realizada no arquivo **simple_components.json** o sistema busca baseado na linguagem escolhida pelo usuário, no arquivo correspondente qual é a tradução do termo. Caso não exista uma tradução para o termo, o *default* da língua inglesa é utilizado. Os arquivos de tradução existentes são:

Tabela 36 - Idiomas internacionalização

Arquivo	Idioma
OdeMessages_es_ES.properties	Espanhol
OdeMessages_fr_FR.properties	Francês
OdeMessages_it_IT.properties	Italiano
OdeMessages_ko_KR.properties	Coreano
OdeMessages_pt_BR	Português do Brasil
OdeMessages_ru.properties	Russo
OdeMessages_sv.properties	Suiço
OdeMessages_zh_CN.properties	Chinês
OdeMessages_zh_TW.properties	Chinês tradicional

Dado o escopo do presente trabalho somente foram traduzidos termos para o inglês e português brasileiro. Com a possibilidade de internacionalização o sistema fica preparado para gerar o instalador(.apk) e disponibilizar o aplicativo para uso no celular.

A seção 4.3.3.4 discorre sobre como gerar em ambiente *localhost* o arquivo para instalação no dispositivo móvel.

4.3.3.4 Criando o APK

Em ambiente *localhost* o servidor de compilação do aplicativo roda separadamente do servidor da aplicação web. Portanto para criar o instalador que será utilizado no dispositivo móvel deve-se subir um servidor local, utilizando as *libs* corretas para o uso com a ferramenta modificada. Para tal deve-se criar uma pasta chamada dxcache na pasta onde o projeto foi colocado. Conforme exemplo: `C:\<caminho_para_projeto>\dxcache` no ambiente de desenvolvimento utilizado a pasta ficou localizada em `C:\workspace\dxcache`. Com a pasta criada para subir o servidor gerador do .apk deve abrir um terminal do próprio Windows e executar o comando a seguir:

```
cd C:\<caminho_para_projeto>\appinventor-
sources\appinventor\buildserver\build\run\lib

java -Xms512m -Xmx512m -cp
BuildServer.jar;CommonUtils.jar;CommonVersion.jar;FastInfoset-
1.2.2.jar;activation-1.1.jar;args4j-2.0.18.jar;asm-3.1.jar;bcpkix-
jdk15on-149.jar;bcprov-jdk15on-149.jar;commons-io-2.0.1.jar;grizzly-
servlet-webserver-1.9.18-i.jar;guava-14.0.1.jar;http-
20070405.jar;jackson-core-asl-1.9.4.jar;jaxb-api-2.1.jar;jaxb-impl-
2.1.10.jar;jaxb-xjc.jar;jdom-1.0.jar;jersey-bundle-1.3.jar;jersey-
multipart-1.3.jar;jettison-1.1.jar;json.jar;jsr311-api-
1.1.1.jar;localizer;jar;mail-1.4.jar;rome-0.9.jar;sdklib.jar;stax-api-
1.0-2.jar;wadl-cmdline.jar;wadl-core.jar;wadl2java.jar -
Dfile.encoding=UTF-8 com.google.appinventor.buildserver.BuildServer --
dexCacheDir C:\<caminho_para_projeto>\dxcache
```

No contexto do presente trabalho o comando para subir o servidor é

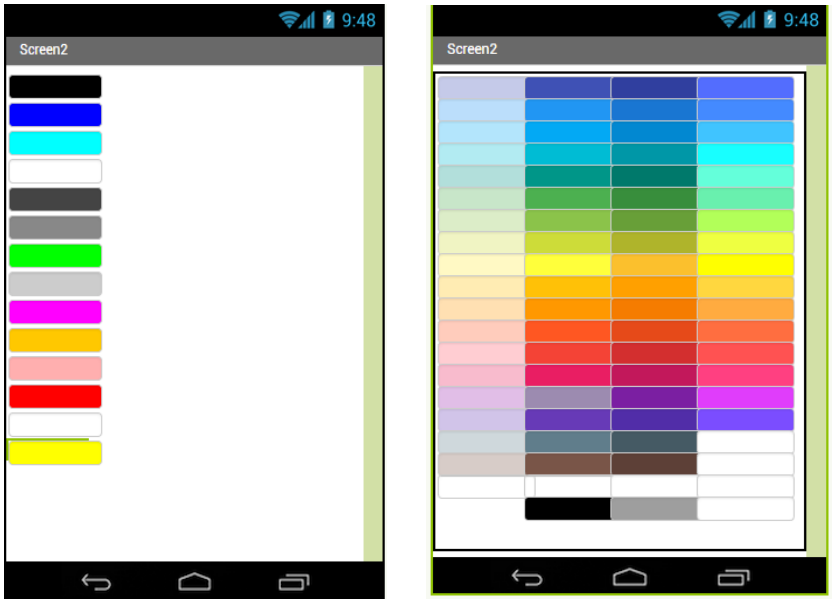
```
Cd C:\workspace\AppInventorCnE\appinventor\buildserver\build\run\lib

java -Xms512m -Xmx512m -cp
BuildServer.jar;CommonUtils.jar;CommonVersion.jar;FastInfoset-
1.2.2.jar;activation-1.1.jar;args4j-2.0.18.jar;asm-3.1.jar;bcpkix-
jdk15on-149.jar;bcprov-jdk15on-149.jar;commons-io-2.0.1.jar;grizzly-
servlet-webserver-1.9.18-i.jar;guava-14.0.1.jar;http-
20070405.jar;jackson-core-asl-1.9.4.jar;jaxb-api-2.1.jar;jaxb-impl-
2.1.10.jar;jaxb-xjc.jar;jdom-1.0.jar;jersey-bundle-1.3.jar;jersey-
multipart-1.3.jar;jettison-1.1.jar;json.jar;jsr311-api-
1.1.1.jar;localizer;jar;mail-1.4.jar;rome-0.9.jar;sdklib.jar;stax-api-
1.0-2.jar;wadl-cmdline.jar;wadl-core.jar;wadl2java.jar -
Dfile.encoding=UTF-8 com.google.appinventor.buildserver.BuildServer --
dexCacheDir C:\workspace\dxcache
```

Com o servidor rodando ao utilizar a opção de criar .apk da ferramenta Web, o servidor construirá o *build* e converterá os arquivos para o formato de .apk sendo este passível de instalação em qualquer dispositivo móvel que utiliza o sistema operacional Android. Ao observar o terminal onde o servidor de construção do .apk foi instanciado, é possível visualizar o log de progresso da criação do aplicativo, e seu status.

Para a visualização do aplicativo criado existem duas alternativas viáveis. A primeira é visualizar no dispositivo móvel. Para tal, basta transferir o .apk criado para o dispositivo móvel e executar o mesmo. Se gerado com sucesso, o app criado via ferramenta App Inventor deve aparecer. Outra alternativa possível é a utilização de emuladores virtuais de celulares Android. No desenvolvimento do presente trabalho foram testados ambos os casos. Recomenda-se a utilização de emuladores pois existe a possibilidade de se instanciar uma grande variedade de celulares diferentes, com tamanhos de telas e especificações de *hardware* diferentes e distribuições diferentes do Android. Além da possibilidade de verificar os logs no momento de instalação do app. Foi utilizado o AVD Manager (Android Virtual Device Manager), *software* embutido no Android Studio.

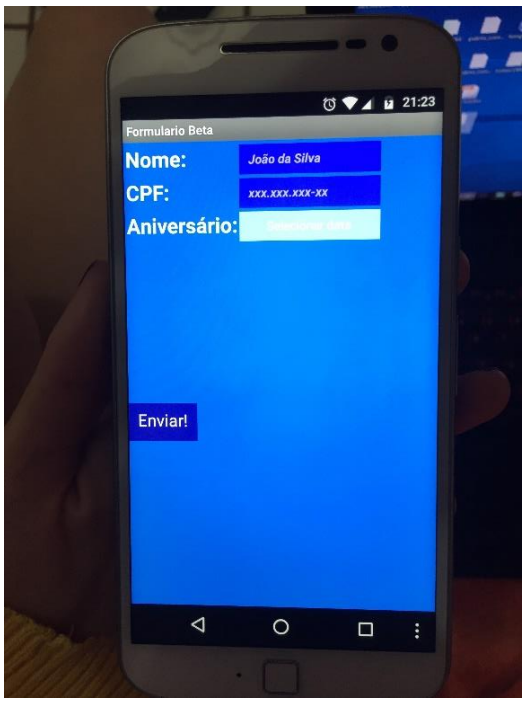
Como resultado da implementação observou-se um aumento considerável nas possibilidades de cores que o usuário poderá utilizar para seus elementos visuais. Além das cores estarem mais condicentes com as utilizadas em apps modernos, o acréscimo dos tons proporciona mais possibilidades para combinação de cores. Considerando a seleção de cores primárias, a versão antiga possuía 14 possibilidades enquanto que a ferramenta customizada apresenta 22 opções de cores primárias. Cada cor primária apresenta opções de cores secundárias, totalizando 75 cores disponibilizadas. Se levado em conta o contexto de *design* de cores do Material Design que indica a escolha de duas cores principais para o app, sendo uma primária e uma secundária podemos verificar via análise combinatória que são possíveis 5.550 combinações tomadas de dois a dois, em detrimento a 182 da versão antiga. A Figura 32 ilustra as possibilidades para cores antes e após a implementação.

Figura 32 – Comparação possibilidades de cores



Após a conclusão da implementação testes foram feitos para a garantia da entrega dos requisitos funcionais modelados na sessão 4.2 do presente trabalho. A Figura 33 demonstra o resultado de um teste utilizando um aparelho móvel visualizando um formulário simples utilizando azul e tons derivados. Nota-se que com sem a customização seria impossível construir tal formulário, dado que só existia um tom de azul na paleta antiga.

Figura 33 – Beta teste cores Material Design



Para o controle de versionamento dos arquivos editados e criados, foi utilizado o ambiente de desenvolvimento da UFSC – GitLab (https://codigos.ufsc.br), no qual foram reportados os bugs resolvidos e listadas as tarefas realizadas.

O código desenvolvido é livre e encontra-se disponível para melhorias e para ser utilizado em trabalhos futuros. O mesmo é distribuído de forma GRATUITA em forma de código aberto sob a licença Creative Commons Attribution 4.0 International License (CC BY 4.0) podendo ser clonado via git no ssh: git@codigos.ufsc.br:100000000394729/AppInventorCnE.git mediante a liberação de acesso por parte do administrador do ambiente UFSC – GitLab.

4.4 Testes

Foram utilizados testes exploratórios para a identificação de possíveis defeitos nas novas funcionalidades desenvolvidas. Foram utilizadas todas as opções de cores primárias/secundárias para cada um dos componentes que possuíam algum tipo de seletor de cores. A Tabela 37 apresenta o resumo dos casos de testes realizados.

Tabela 37 – Testes exploratórios

P = Cores Principais T= Tonalides

Menu	**Componente**	**Cor de fundo**		**Cor do texto**		**Cor fundo do Item**		**Cor texto do item**		**seleção**		**Cor a esquerda**		**Cor a direita**		**Cor da Pintura**	
		P	T	P	T	P	T	P	T	P	T	P	T	P	T	P	T
Interface do Usuário	Botão	x	x	x	x												
	Caixa de seleção	x	x	x	x												
	Escolhe data	x	x	x	x												
	Legenda	x	x	x	x												

	Escolhe lista	x	x	x	x	x	x	x	x							
	Visualizador de lista	x	x	x	x					x	x					
	Notificador	x	x	x	x											
	Caixa de senha	x	x	x	x											
	Deslizador											x	x	x	x	
	Caixa de texto	x	x	x	x											
	Escolhe hora	x	x	x	x											
Organização	Organização horizontal	x	x													
	Horizontal scroll arrangement	x	x													
	Organização vertical	x	x													
	Vertical scroll arrangement	x	x													
Animação	Bola														x	x
	Pintura	x	x												x	x
Social	Escolhe contato	x	x	x	x											
	Escolhe email	x	x	x	x											
	Escolhe número telefone	x	x	x	x											

Durante os testes exploratórios foram encontrados diferentes bugs críticos no sistema. Abaixo são listados alguns exemplos e como esses foram resolvidos:

- Sistema não mostrava cor inicial dos componentes: Ao abrir a ferramenta, nenhuma das propriedades de cor tinha a cor *default* selecionada, vindo sempre um html quebrado. Para solucionar foi

necessário trocar as cores *default* no json de configuração dos componentes.

- Novas labels não estavam traduzidas: Sempre era mostrado o nome da nova propriedade em inglês.
- Bus principal de disparo de eventos não disparava evento personalizado: O app inventor possui um bus para tratar eventos, porém este não estava disparando de forma correta o evento customizado criado. Portanto foi implementada um *dispatcher* exclusivo para a aplicação.
- Sistema buildava o app porém ao abrir no celular ele quebrava o aplicativo: Faltava implementar as alterações de cores nas classes representativas dos componentes que interagiam com a parte mobile.

5 CONCLUSÃO

O objetivo do presente trabalho é a identificação e implementação de melhorias na ferramenta App Inventor em relação ao UX *design*. No desenvolvimento de apps oferecendo um suporte melhor alinhado ao guia de estilo Material de Design. Nesse contexto, foi sintetizada a fundamentação teórica de aprendizagem e ensino no ensino básico, referente ao desenvolvimento de aplicativos móveis com a ferramenta App Inventor integrando *user experience* (UX) e do design de cores (O1).Foi levantado o estado da arte de unidades instrucionais para o ensino de conceitos de programação, UX *design* e *design thinking* para jovens do ensino básico (O2).

Foi identificada a carência de unidades instrucionais que abordem além do ensino de programação também o ensino de *design*. Com base na unidade instrucional “Faca o seu app” sendo desenvolvida pela iniciativa Computação na Escola foram levantados requisitos funcionais focados em melhorar a ferramenta App Inventor voltado ao suporte de *design* de cores no *design* de interfaces dos apps. De acordo com os requisitos foram implementadas e testadas alterações no código fonte da ferramenta.

A nova versão oferece a possibilidade de escolha de 75 cores alinhadas ao material design para qualquer propriedade de cor dos componentes gráficos. Espera-se que com este suporte melhorado, seja possível desenvolver apps com uma maior diversidade de cores em maior conformidade com as cores padrões resultando em apps com maior usabilidade e atratividade.

Dando continuidade do presente trabalho, trabalhos futuros podem abranger novos requisitos funcionais relacionados ao suporte de UX *design (*por exemplo, ícones, padrões de tela, tipografias, botão suspenso, etc.) pelo App Inventor no contexto do ensino de computação deixando o suporte oferecido também cada vez mais alinhado com conceitos referentes a *design* de interfaces de apps.

REFERÊNCIAS

A-GHAMDI, S. A ;AL-RAJHI , N. (2016**) Using App Inventor 2 in A Summer Programming Worskhop: Improvements Over Previous Years**. *IEEE Global Engineering Education Conference* (EDUCON). pp.383 -388

ANASTASIOU, L. ; ALVES, . (2004). **Processos de ensinagem na universidade. Pressupostos para as estratégias de trabalho em aula**. 3. ed. Joinville: Univille, p. 67-100.

App Inventor (2017). Disponível e m https://sites.google.com/site/appinventor/capabilities-limitations Acesso em: 25 de setembro de 2016

App Inventor (2017). Disponível em: http://ai2.appinventor.mit.edu Acesso em: 13 de maio de 2017.

AL-KHALIFA H. S (2013) **Applying Knowledge, Skills and Abilities in undergraduate research seminar course** *12th International Conference on Information Technology Based Higher Education and Training (ITHET),* Antalya, pp. 1-3.

BERNS, R. (2000**) Principles of color technology** 3. Ed.. New York: John Wiley & Sons

BLOOM, B.S (1956) **Taxonomy of Educational Objectives: The Classification of Educational Goals**; pp. 201-207; (Ed.) David McKay Company, Inc

BRANCH R. (2009) **Instructional Design: The ADDIE Approach**. Vol. 722 New York: Springer & Media. .

BUTLER, M. (2011) **Android: Changing the Mobile Landscape** in IEEE Pervasive Computing, vol. 10, no.1, pp. 4-7

CAPES, **Quem Participa**. Disponível em: http://www.periodicos.capes.gov.br/index.php?option=com_pcontent&view=pcontent&alias=quem-participa&Itemid=101/ Acesso em: 30 de outubro de 2016.

CETIC (2014). **TIC Kids Online**. Disponível em http://data.cetic.br/cetic/dados?idPesquisa=TIC_KIDS Acessado em: 25 de setembro de 2016

CHENG; M. J.; HUNG S. W; H. H. TSAI AND P. W. CHEN (2016) **The Adoption Intentions Of Mobile Applications**. *IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS),* Okayama, pp-1-3.

Code.org (2017). Disponível em: https://code.org/ . Acesso em: 19 de fevereiro de 2017.

comScore (2015). **O Cenário Mobile e multi plataforma no Brasil**. Disponível em: http://www.comscore.com/por/Imprensa-e-eventos/Apresentacoes-e-documentos/2015/O-Cenario-Mobile-Multi-Plataforma-e-as-Tendencias-para-2016 Acesso em: 09 de abril de 2017.

Computação na escola ,2017. Disponível em: http://www.computacaonaescola.ufsc.br/ Acessado em: 19 de fevereiro de 2017.

ComputerWorld , 2015 **Material Design Apps Android**. Disponivel em: http://www.computerworld.com/article/2909897/material-design-apps-android.html/ Acesso em: 20 de setembro de 2016.

Convergência Digital (2015) **Mundo tem 7,1 bilhões de celulares ativos**. Disponível em: http://convergenciadigital.uol.com.br/cgi/cgilua.exe/sys/start.htm?UserActiveTemplate=site&infoid=40220&sid=8 Acesso em: 25 de junho de 2016.

CSTA (2016). **INTERIM K-12 Computer Science Standards**. Disponível em https://c.ymcdn.com/sites/www.csteachers.org/resource/resmgr/Docs/Standards/2016StandardsRevision/INTERIM_StandardsFINAL_07222.pdf Acesso em: 08 de abril de 2016.

CROW, M. ; LYNN, S. (2010) **The New Centurions**. *IEE Power And Energy Magazine*., Vol. 8, n 4, pp. 20-26

DUDA, R. ; SILVA , S.C (2015**) Desenvolvimento de Aplicativos para android com uso do app inventor: uso de novas tecnologias no processo de ensino-aprendizagem em matemática**. Revista Conexão UEPG, v.11, n.3

FGV - Fundação Getúlio Vargas (2016). **27º Pesquisa Anual do uso de T.I**. Disponivel em: http://eaesp.fgvsp.br/sites/eaesp.fgvsp.br/files/pesti2016gvciappt.pdf acesso em: 30 de junho de 2016.

FINIZOLA, A. B; HENNING, E. (2014) **O ensino de programação para dispositivos móveis utilizando o MIT-App Inventor com alunos do ensino médio**. 20º *Workshop* de Educação em Informática, p. 337–341

FILATRO, A. (2008) **Design instrucional na prática**. São Paulo: Pearson p3?. 2008

FOLEY, H.;MATLIN, M. (1996) **Sensación y percepción**. México: Prentice Hall

GARTNER (2016). **Gartner Says Worldwide smartphone sales grew 3.9 percent in first quarter of 2016**. Disponivel em: http://www.gartner.com/newsroom/id/3323017 Acesso em: 30 de junho de 2016

GONÇALVES B.S (2004) **Cor Aplicada ao Design Gráfico: Um modelo de núcleo virtual para aprendizagem baseado na resolução de problemas** UFSC, Santa Catarina.

GOMES, T.; MELO, J. (2013) **App Inventor for Android: Uma nova possibilidade para o ensino de lógica de programação**. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, Vol. 2, No. 1.

GROVER, S. ; PEA , R. (2013) **Using a Discourse-Intensive Pedagogy and Android's App Inventor for Introducing Computational Concepts to Middle School Students**. *Proceeding of the 44th ACM technical symposium on Computer science education* (SIGCSE '13). pp 723-728

HARTSON, R.;PYLA, P. S. (2012). **The UX Book: Process and Guidelines for Ensuring a Quality User Experience**, Morgan Kaufmann, San Francisco, USA

HASSENZAHL M.(2008) **User Experience (UX): Towards an experiential perspective on product quality**. *Proceedings of the 20th Conference on Interaction Homme-Machine, pp. 11-15*, Metz, France.

HESS, M. (2010) **A discourse of the process of designing for real people**. Disponivel em: http://52weeksofux.com/post/890289075/startuxs Acesso em: 07 de novembro de 2016.

HOLLEY, J.(2008) **Generation Y: Understanding the Trend and Planning for the Impact**. *In proceddings of the 32nd Annual IEEE International Computer Software and Applications Conference,* Turku, pp.2-2.

HUANG, K. (2009) **Challenges in Human-Computer Interaction Design for Mobile Devices**. *Proceedings of the World Congress on Engineering and Computer Science 2009* Vol I

IBGE (2013) **Acesso à internet e à televisão e posse de telefone móvel celular para uso pessoal** Disponivel em: http://ibge.gov.br/home/estatistica/populacao/acessoainternet2013/default.shtm Acesso em: 15 de agosto de 2016.

IBOPE (2015). **Mobile Report**. Disponível em http://www.nielsen.com/br/pt/press-room/2015/68-milhoes-usam-a-internet-pelo-smartphone-no-Brasil.html Acesso em: 28 de setembro de 2016.

Imagnity (2016). Disponível em: http://www.imagnity.com/tutorial-index/ Acesso em: 15 de agosto de 2016.

ISO 9241 (2006). Disponivel em: www.labiutil.inf.ufsc.br/cpqd-capacitacao/iso9241-11F2.doc/ Acesso em: 26 de outubro de 2016.

ISLAM, N.; WANT, R. (2014) **Smartphones: Past, Present, and Future**. In *IEEE Pervasive Computing* , Vol. 13, no. 4, , pp. 89-92

ISLEIFSDOTTIR, J.; LARUSDOTTIR M**.** (2014) **Measuring the User Experience of a Task Oriented Software**. *Proceedings of the International Workshop Science and Information Conference,* Vol. 8

Jana (2016) **The Affortable Smartphone Oportunity in Brazil**. Disponível em: http://blog.jana.com/blog/2015/06/05/global-markets-foreshadow-low-cost-smartphone-opportunity-in-brazil Acesso em: 29 de setembro de 2016

Jana (2016) **Brazil´s Mobile Landscape. A snapshot** Disponível em: http://www.tudocelular.com/android/noticias/n61396/Android-Brasil-Mercado.html Acesso em: 22 de setembro de 2016

KITCHENHAM, B.(2004) **Procedures for Performing Systematic Reviews**. Uk: NICTA Technical Report 0400011T.1, Keele University, KEELE, GB,

KUUSINEN, K.; MIKKONEN, T. (2014) **On Designing UX for Mobile Enterprise Apps** *40th EUROMICRO Conference on Sotfware Engineering and Avanced Applications*, Verona, pp. 221-228.

KELLEHER C; PAUSCH., R. (2005) **Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers**. ACM *Computer Surveys*, Vol. 37, no. 2, pp. 83-137

LIBÂNEO, J. C. (2009) **Didática**. 29 ed. Cortez

LIN, F.; YE, W. (2009) **Operation System Battle in the Ecosystem of Smartphone Industry** *International Siymposium on Information Engineering and Eletric Commerce,* Ternopil, pp. 617-621

Material Design(2017). Disponível em: https://material.io/guidelines/ Acessado em: 28 de março de 2017.

Material Design Color, (2017). Disponível em: https://material.io/guidelines/style/color.html# Acessado em: 05 de maio de 2017.

MARCHETI F. A. P; BELHOT R. V. (2010) **Taxonomia de Bloom: revisão teórica e apresentação das adequações do instrumento para definição de objetivos instrucionais** Gest. Prod. Por extenso, vol. 17, n. 2, pp. 421-431

MAZZIONI, S. (2009) **As Estratégias Utilizadas no Processo de Ensino-Aprendizagem: Concepções de Alunos e Professores de Ciências Contábeis**. In: Anais do Congresso USP Controladoria e Contabilidade, São Paulo/Brasil

MEC (2011) **Diretrizes Curriculares Nacionais - Do Histórico da Computação, do Computador e dos Cursos**. Disponível em: http://homepages.dcc.ufmg.br/~bigonha/Bigonha/dcn-versao%20final-f.pdf Acesso em: 07 de abril de 2017.

MINISTÉRIO DA EDUCAÇÃO (2016). **Lei de Diretrizes e Bases da Educação** Disponível em: http://www.planalto.gov.br/ccivil_03/leis/L9394.html Acessado em: 31 de agosto de 2016

MOORE, S. (2016) **Three Elements Of a Successful Mobile apps**. Disponível em: http://www.gartner.com/smarterwithgartner/three-elements-of-successful-mobile-apps/ Acesso em: 14 de abril de 2017.

Mundo da Psicologia (2017) **Psicologia das cores e suas representações pelo mundo**. Dísponivel em: http://mundodapsi.com/cores-e-emocoes/ Acesso em: 11 de maio de 2017.

NASCIMENTO, I. et al.. (2016) **An Empirical Study to Evaluate the Feasibility of a UX and Usability Inspection Technique for Mobile Applications.** *In 28th International Conference on Software Engineering & Knowledge Engineering*, California, USA.

NAZRUL, M. (2015) **Exploring the Intuitiveness of Iconic, Textual and Icon with Texts Signs for Designing User-Intuitive Web Interfaces** *18th International Conference on Computer and Information Technology*, Dhaka, pp. 450-455.

NIELSEN, J. (1992) **The Usability Engineering Life Cycle** *IEEE Computer Society Press* Vol. 25, No. 3, pp. 12-22.

NIELSEN, J. (2007) **Usabilidade na Web – Projetando Websites com Qualidade** Editora Campus. Prefácio.

PCN (2015) **Parâmetros Curriculares Nacionais, Terceiro e Quarto ciclos do Ensino Fundamental**. Disponível em: http://portal.mec.gov.br/seb/arquivos/pdf/introducao.pdf Acesso em: 25 de agosto de 2016

PIMENTA, S ; ANASTASIOU, L. (2002) **Docência no ensino superior**. São Paulo: Cortez, pp. 195

Pura Vida Apps, (2016) Disponível em: http://puravidaapps.com/tutorials.php Acesso em: 14 de agosto de 2016.

RAHA, D. (2009) **The Mobile Revolution and the Cellphone as an Antipoverty Vaccine** *Global Information Infrastructure Symposium*, Hammemet, pp. 1-10

SASKATCHEWAN EDUCATION (1991) . **Instructional Approaches: A Framework for Professional Practice**. Canada: Saskatchewan Education

SHAH A. (2016). **Overcoming UX Designing Callenges**. Disponivel em: https://uxmag.com/articles/overcoming-ux-designing-challenges Acesso em: 24 de outubro de 2016.

SIMPSON, E. J. (1972) **The classification of educational objectives, psychomotor domain**. Washington: Gryphon House.

SICA, C. (2008) **Ciência da Computação no Ensino Básico e Médio**.

SILVA , S.(2015) **Desenvolvimento de aplicativos para android com o uso de app inventor: Uso de novas tecnologias no processo de ensino-aprendizagem em matemática** . Revista Conexão UEPG,Vol.11 no. 2

SMITH, P. L; RAGAN, T.J (1999) **Instrucional design.** John Wiley & Sons, Inc.

SMOLKA, A. L. B;GÓES, M. C. (1995) **A linguagem e o outro no espaço escolar: Vygotsky e a construção do conhecimento**. São Paulo: Editora Papirus

STATISTA (2016) **Number of apps available in leading app stores as of June 2016**. Disponível em: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/ Acesso em: 29 de setembro de 2016

TECHNOVATION , 2016. Disponível em: http://www.technovationchallenge.org/ Acesso em: 10 de novembro e 2016.

TEIXEIRA, F. (2014) **Introdução e boas práticas em UX Design**. Editora casa do Código pp .12-13

TRENDER M. (2014) **The History of user Experience Design**. Disponível em: https://medium.com/@marcintreder/the-history-of-user-experience-design-5d87d1f81f5a#.jt8g0wqdp Acesso em: 08 de outubro de 2016.

TRILHA, G. (2016) **Design de unidade instrucional de desenvolvimento de aplicativos para o ensino fundamental**. Trabalho de conclusão de curso UFSC.

TRILHA, D. G...et. al (2017**) Ensinando a Computação por meio de Programação com App Inventor.** Computer on the Beach, Florianópolis/Brazil

VALLARTA R. (2007) **OpenBlocks: An Extendable Framework for Graphical Block Programming Systems**. *Master´s Thesis, Massachusetts Institue of Technology, Cambridge, MA, USA.*

VINNAKOTA, T. R. (2014) .**Integration of Design Thinking with Strategy and Innovation in an Enterprise Context** *IEEE International Conference on Management of Innovation and Technology*, Singapore, pp. 131-136

WAGNER, A. *et al.* (2013) **Using App Inventor in a k-12 Summer Camp**. *Proceeding of the 44th ACM technical symposium on Computer science education* pp. 621-626

WASSERMAN A. I. (2010) **Software engineering issues for mobile application development**. *In: Proceedings of FoSER´10 FSE/SDP Workshop on Future of software engineering research*, Santa Fe, New México/USA

WOLBER, D. *et al.* (2011) **App inventor: create your own android apps.** Sebastopol, CA: O'Reilly Media.

YIN P.L *et al.* (2014) **Entrepreneurial Innovation: Killer Apps in the iPhone Ecosystem** *American Economic Review, vol. 104, no. 5, pp. 255-259*

ZAHIDI Z. (2014) **Understanding the User Experience (UX) Factors that Influence User Satisfaction in Digital Culture Heritage Online Collections for Non-Expert Users** *Science and Information Conference*, London, pp. 57-63

ANEXO A – ARTIGO

Suporte a unidade instrucional de desenvolvimento de aplicativos com técnicas de UX design para o Ensino básico

Daniel Melo da Silva

[1]Instituto Nacional para Convergência Digital (INCoD)/ Departamento de informática e Estatística (INE)/ Universidade Federal de Santa Catarina (UFSC)

daniel.melo@involves.com.br

Abstract. UX design of mobile apps can determinate if the app will be a success or not. Among of another factors that determinate if a user interface will deliver a good user experience, the colors of the app is one of the most important. Using a wrong color will implicate on not sending the right message to the user, or make more diifficult to accomplish a task. The teaching tool App Inventor is a really complete tool to teach kids how to programa apps, but its support on the color palette of android applications is low. The objetive of this paper is to study and develop new features that will be more align with the Android style guide Material Design. The new features were implemented and tested, creating an customized App Inventor tool. This features change the color behavior of the App Inventor application to provide better support on creating apps with better UX.

Resumo. *UX design de aplicativos móveis pode determinar se um aplicativo vai ser um sucesso ou não. Entre todos fatores que determinam se uma interface de usuário vai entregar uma boa experiencia de uso, as cores dos elementos gráficos são uma das mais importantes. Usar cores erradas pode implicar em não passar a mensagem correta para o usuário, ou dificultar a realização de alguma tarefa. A ferramenta de ensino de programação para iniciantes App Inventor é uma ferramenta bem completa, porém o seu suporte a paleta de cores de aplicativos android é baixa. O objetivo deste trabalho é o estudo e desenvolvimento de novas funcionalidades que estarão mais alinhadas ao guia de estilo Material Design, auxiliando no suporte a unidades instrucionais que ensinem conceitos de UX design. As novas funcionalidades foram implementadas e testadas, criando uma versão customizada da ferramenta App Inventor. Essas funcionalidades mudam o comportamento das*

cores do App Inventor para prover um melhor suporte na criação de apps com uma boa UX.

1. Introdução teórica

O cotidiano atual da sociedade encontra-se fortemente atrelado a tecnologia de informação e comunicações. Em 2015 foram estimados que existiam 7,1 bilhões de celulares ativos no mundo todo (Convergência Digital, 2015). No Brasil, estamos acima da média global em relação ao número de telefones per capita, atingindo 1,2 aparelhos por habitante (FGV, 2016). Utilizar dessa onipresença e do interesse em utiliza-lo, principalmente pelos Millenials pode ser uma forma viável de ensinar conceitos de programação e UX Design. Estes jovens, preferem aprender interagindo, se sentem mais confortáveis aprendendo em um ambiente onde a penalidade pela tentativa/erro seja baixa (CROW, 2010). Quase nunca leem manuais e preferem o *e-learning* sob demanda como meio de aprendizado (CROW, 2010). São considerados altamente alfabetizados digitalmente (IT *literacy*), ou seja, sabem usar TI.

Uma forma de ensinar conceitos avançados de programação para os Millenials é utilizando a ferramenta App Inventor. O App Inventor é um ambiente de desenvolvimento que utiliza programação visual em blocos e permite qualquer pessoa, até mesmo crianças, começar a programar e construir aplicativos completos para dispositivos Android. Porém as funcionalidades presentes na ferramenta para o *design* de interface não possui recursos compatíveis com as definições atuais para aparências de interfaces de aplicativos Android, segundo o *Material Design* (Material Design, 2017).O *Material Design* é uma linguagem visual desenvolvida pelo Google para sintetizar os princípios clássicos de um bom *design* de interface e proporcionar uma experiência de uso uniforme independente de dispositivos móveis e seus diferentes tamanhos.

Dado este cenário o presente artigo tem como objetivo apresentar o desenvolvimento de melhorias na ferramenta App Inventor, afim de adicionar funcionalidades compatíveis com conceitos do *Material Design* no desenvolvimento de interfaces de aplicativos. Servindo desta forma como suporte à aplicação da unidade instrucional desenvolvida pela iniciativa Computação na Escola.

2. Levantamento funcionalidades e desenvolvimento

Para entender melhor como a cor impacta em uma experiência de usuário e na transmissão de sentimentos foi feito um estudo sobre a teoria das cores. A teoria das cores estuda fatores emocionais e de sentimentos, que são associados a cores especificas. Empresas utilizam dessa técnica para transmitir por meio da seleção de

cores de suas Logomarcas, sentimentos associados a empresa, como demonstra a Figura 1.

Figura 1 - Guia emocional das cores (Mundo da Psicologia, 2015)

Identificando então como as cores podem impactar e transmitir sentimentos para o usuário de um app, foram levantados os fundamentos do Material Design relacionado a cores, e selecionada cores que fizessem sentido para a aplicação no App Inventor. Chegando na *palette* de cores ilustrada na Figura 2.

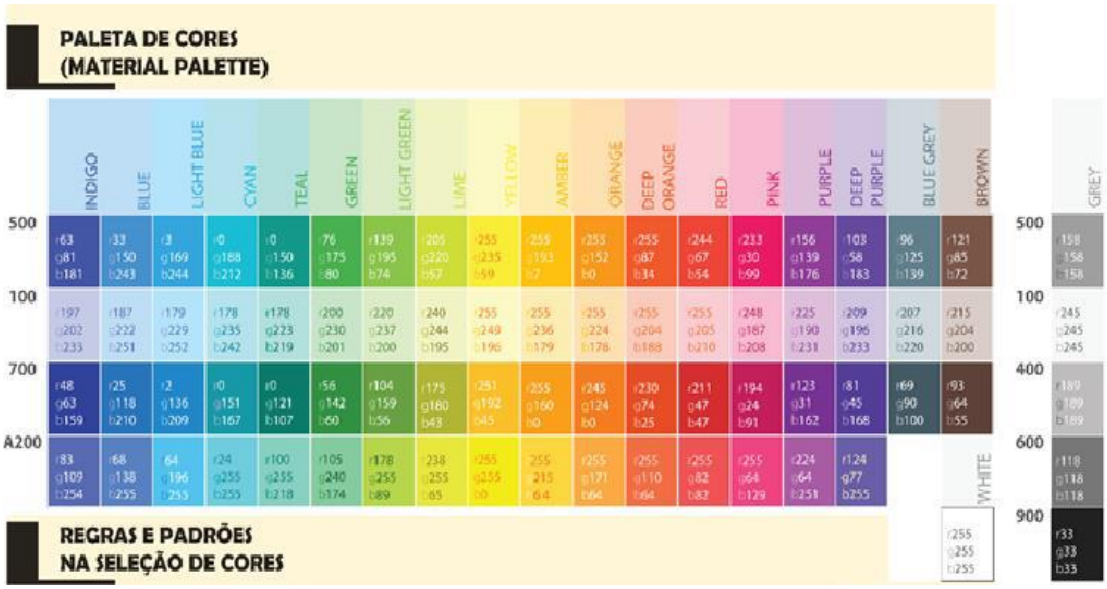


Figura 2 – Palette de cores sugeridas para implementação

Com a *palette* definida foram levantados requisites funcionais de como disponibilizar a seleção dessas cores, de forma a não comprometer a usabilidade da ferramenta. Portanto foram levantados os sequintes requisitos:

1- O usuário deve ser capaz de selecionar uma cor principal.
2- Para cara seletor de cores principais presente nas propriedades de um elemento, tais como cores de fundo, cor de texto, deve-se ter uma propriedade análoga que apresente tonalidades secundárias para a cor selecionada.
3- Ao trocar a cor principal o sistema deve atualizar a propriedade que lista as cores secundárias.

3. Testes e resultados

Após a implementação que contemplou os requisitos funcionais foram realizados testes exploratórios em mais de 20 propriedades de cores que tiveram seu comportamento modificado, tanto na interface Web quanto no aplicativo compilado no celular. Considerando a seleção de cores primárias, a versão antiga possuía 14 possibilidades enquanto que a ferramenta customizada apresenta 22 opções de cores primárias. Cada cor primária apresenta opções de cores secundárias, totalizando 75 cores disponibilizadas. Se levado em conta o contexto de *design* de cores do Material Design que indica a escolha de duas cores principais para o app, sendo uma primária e uma secundária podemos verificar via análise combinatória que são possíveis 5.550 combinações tomadas de dois a dois, em detrimento a 182 da versão antiga.

5. Conclusão

Com base na unidade instrucional “Faca o seu app” sendo desenvolvida pela iniciativa Computação na Escola foram levantados requisitos funcionais focados em melhorar a ferramenta App Inventor voltado ao suporte de *design* de cores no *design* de interfaces dos apps. De acordo com os requisitos foram implementadas e testadas alterações no código fonte da ferramenta. A nova versão oferece a possibilidade de escolha de 75 cores alinhadas ao material design para qualquer propriedade de cor dos

componentes gráficos. Espera-se que com este suporte melhorado, seja possível desenvolver apps com uma maior diversidade de cores em maior conformidade com as cores padrões resultando em apps com maior usabilidade e atratividade.

Referências

Convergência Digital (2015) **Mundo tem 7,1 bilhões de celulares ativos**. Disponível em: http://convergenciadigital.uol.com.br/cgi/cgilua.exe/sys/start.htm?UserActiveTemplate=site&infoid=40220&sid=8 Acesso em: 25 de junho de 2016.

FGV - Fundação Getúlio Vargas (2016). **27º Pesquisa Anual do uso de T.I**. Disponivel em: http://eaesp.fgvsp.br/sites/eaesp.fgvsp.br/files/pesti2016gvciappt.pdf Acesso em: 30 de junho de 2016.

CROW, M. ; LYNN, S. (2010) **The New Centurions**. IEE Power And Energy Magazine., Vol. 8, n 4, pp. 20-26

Material Design(2017). Disponível em: https://material.io/guidelines/ Acesso em: 28 de março de 2017

Mundo da Psicologia (2017) **Psicologia das cores e suas representações pelo mundo**. Dísponivel em: http://mundodapsi.com/cores-e-emocoes/ Acesso em: 11 de maio de 2017.