



UNIVERSIDADE FEDERAL DE SANTA CATARINA
Centro Tecnológico
Programa de Pós-Graduação em Ciência da Computação

Fernando da Cruz Pinheiro

**MODELO INSTRUCIONAL PARA O ENSINO DE ENGENHARIA DE SOFTWARE E
USABILIDADE VOLTADO AO ENSINO FUNDAMENTAL**

Florianópolis
2019

UNIVERSIDADE FEDERAL DE SANTA CATARINA
Centro Tecnológico
Programa de Pós-Graduação em Ciência da Computação

Fernando da Cruz Pinheiro

**MODELO INSTRUCIONAL PARA O ENSINO DE ENGENHARIA DE SOFTWARE E
USABILIDADE VOLTADO AO ENSINO FUNDAMENTAL**

Dissertação submetida ao Programa de Pós-Graduação em
Ciência da Computação, da Universidade Federal de Santa
Catarina, para a obtenção do Grau de Mestre em Ciência da
Computação.

Orientadora: Prof.^a Dr. rer. nat. Christiane Gresse von
Wangenheim, PMP
Coorientador: Prof. Dr. Jean Hauck

Florianópolis
2019

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Pinheiro, Fernando da Cruz
MODELO INSTRUCIONAL PARA O ENSINO DE ENGENHARIA DE
SOFTWARE E USABILIDADE VOLTADO AO ENSINO FUNDAMENTAL /
Fernando da Cruz Pinheiro ; orientadora, Christiane Gresse
von Wangenheim, coorientador, Jean Hauck, 2019.
230 p.

Dissertação (mestrado) - Universidade Federal de Santa
Catarina, Centro Tecnológico, Programa de Pós-Graduação em
Ciência da Computação, Florianópolis, 2019.

Inclui referências.

1. Ciência da Computação. 2. Engenharia de software. 3.
Engenharia de usabilidade. 4. App Inventor. 5. Programação.
I. Wangenheim, Christiane Gresse von. II. Hauck, Jean.
III. Universidade Federal de Santa Catarina. Programa de
Pós-Graduação em Ciência da Computação. IV. Título.

Fernando da Cruz Pinheiro

Título: MODELO INSTRUCIONAL PARA O ENSINO DE ENGENHARIA DE SOFTWARE E USABILIDADE VOLTADO AO ENSINO FUNDAMENTAL

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof.(a) Alice Theresinha Cybis Pereira, Dr(a).

Universidade Federal de Santa Catarina

Prof.(a) Silvia Modesto Nassar, Dr(a).

Universidade Federal de Santa Catarina

Prof.(a) Giliane Bernardi, Dr(a).

Universidade Federal de Santa Maria

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de mestre em Ciência da Computação.

Prof. Dr.(a) José Luís Almada Güntzel

Coordenador(a) do Programa

Prof. Dr.(a) Christiane Gresse von Wangenheim

Orientador(a)

Florianópolis, 1 de agosto de 2019.

AGRADECIMENTOS

Agradeço a Deus por ter me concedido a dádiva da vida, a saúde e a perseverança necessárias para condução deste trabalho.

Agradeço à professora Christiane, pelas suas orientações, prestatividade, diligência, paciência e apoio durante todo o período de desenvolvimento deste trabalho.

Agradeço à toda minha família, que sempre contribuiu para meu desenvolvimento pessoal e profissional.

Agradeço ao professor Jean Hauck, por todo apoio que contribuiu de maneira única para a melhoria da qualidade da pesquisa desenvolvida.

Aos colegas do Grupo de Qualidade de Software – GQS/UFSC e a todos os colegas do Instituto Nacional para Convergência Digital – INCoD/UFSC, pelo apoio oferecido ao longo deste trabalho.

RESUMO

Nos últimos anos, a computação tornou-se cada vez mais relevante e proeminente em nossas vidas diárias. Além disso, a maioria das profissões usa soluções computacionais para cumprir suas funções. Portanto, a educação em informática se tornou popular no ensino fundamental e médio, no qual crianças e jovens aprendem habilidades básicas em computação. Tipicamente, conceitos básicos de computação são ensinados por aplicativos de programação com o *App Inventor*. No entanto, além da programação, a computação envolve outras habilidades fundamentais, como engenharia de software e engenharia de usabilidade. Apesar disso, atualmente não há unidades de ensino cobrindo esses conceitos de engenharia de software/usabilidade na educação básica. Portanto, este trabalho enfoca o desenvolvimento de um modelo instrucional voltado ao ensino de programação e engenharia de software, *design thinking* e experiência do usuário em escolas no Brasil. Essas competências são ensinadas pelo ensino de desenvolvimento de aplicativos com o *App Inventor*, seguindo um processo sistemático. O modelo instrucional foi sistematicamente desenvolvido com base em revisões sistemáticas da literatura seguindo a abordagem de design instrucional do ADDIE. O modelo instrucional foi instanciado, aplicado e avaliado em duas turmas de uma escola pública de ensino fundamental. Os resultados dos aplicativos apresentam resultados motivadores, a maioria dos alunos achou o curso divertido. Os alunos também alcançaram os objetivos de aprendizagem executando o processo de software que desenvolvem um aplicativo. As interfaces de usuário criadas pelos alunos também apresentaram um grau de estética maior quando comparadas com outros aplicativos disponíveis na galeria do *App Inventor*. Esses resultados indicam que é possível popularizar a computação ensinando os alunos do ensino médio a desenvolver aplicativos por meio de um processo sistemático que aborda os conceitos de *design thinking*, engenharia de software e engenharia de usabilidade.

Palavras-chave: Programação, Engenharia de software, Engenharia de usabilidade, Educação Básica, *App Inventor*, *Design Thinking*.

ABSTRACT

In recent years, computing has become increasingly relevant and prominent in our daily lives. Also, most professions use computational solutions to fulfill their functions. Therefore, computing education has become popular in K-12 in which children and young people learn basic computing skills. Typically, basic computing concepts are taught by e.g. programming apps with App Inventor. However, in addition to programming, computing involves other fundamental skills, such as software engineering and usability engineering. Despite this, there are currently no instructional units covering these software engineering/usability concepts in K-12. Therefore, this work focuses on the development of an instructional model focused on teaching programming and software engineering, design thinking and user experience in schools in Brazil. These competencies are taught by teaching app development with App Inventor following a systematic process. The instructional model has been systematically developed based on systematic literature reviews following the ADDIE instructional design approach. The instructional model was instantiated, applied and evaluated in two classes of a public elementary school. The results of the applications present motivating results most students found the course entertaining. The students also achieved the learning objectives executing the software process developing an app. Student-created user interfaces also presented a greater degree of aesthetics when compared with other applications available in the App Inventor gallery. These results indicate that it is possible to popularize computing by teaching students in K-12 to develop applications through a systematic process approaching design thinking, software engineering and usability engineering concepts.

Keywords: Programming, Software engineering, Usability engineering, K-12, App Inventor, Design thinking.

LISTA DE FIGURAS

Figura 1: Etapas da pesquisa.	26
Figura 2: Grupos de processo do ciclo de vida	32
Figura 3: Estrutura do conceito de usabilidade.....	40
Figura 4: Atividades de design centradas no ser humano.	41
Figura 5: Níveis do processo de design centrado no usuário.	43
Figura 6: Diretrizes de interface de usuário do <i>Material Design</i>	44
Figura 7: Processo de Design Thinking.	45
Figura 8: Diagrama com as cinco fases do modelo ADDIE	47
Figura 9: Atividades do modelo ADDIE.....	48
Figura 10: Decomposição dos fatores de qualidade do modelo dTECT.....	54
Figura 11: Conceitos e práticas do K-12.....	57
Figura 12: Níveis de ensino K-12.....	58
Figura 13: Visão geral das várias partes interessadas dos apps.	61
Figura 14: CliqueDenúncia	62
Figura 15: Minigolf	62
Figura 16: MATHfull.....	62
Figura 17: Perspectiva de <i>Designer</i>	63
Figura 18: Editor de Blocos.....	64
Figura 19: Quantidade de UIs enfocando no ensino de ES na Educação Básica publicado por ano.	70
Figura 20: Áreas de conhecimento da ES.	71
Figura 21: Frequência dos modelos/métodos/técnicas utilizados para ensinar ES na Educação Básica.	73
Figura 22: Ambientes de programação utilizadas no ensino de ES na Educação Básica.....	74
Figura 23: Métodos instrucionais.	75
Figura 24: Materiais instrucionais.	76
Figura 25: Instrumentos de avaliação.....	76
Figura 26: Nível escolar.	77
Figura 27: Tipos de estudo.	80
Figura 28: Fatores de qualidade.	81
Figura 29: Métodos de coleta.	81

Figura 30: Métodos de análise de dados.	82
Figura 31: Categoria de tamanho de amostra.	83
Figura 32: Quantidade de UIs publicado por ano.	92
Figura 33: Frequência das técnicas utilizadas para ensinar design de interface.	93
Figura 34: Ambientes de programação utilizados nas UIs.	94
Figura 35: Métodos instrucionais adotados.	95
Figura 36: Materiais instrucionais utilizadas.	96
Figura 37: Métodos de avaliação.	97
Figura 38: Nível de ensino das UIs encontradas.	97
Figura 39: Quantidade de estudos por tamanho da amostra.	99
Figura 40: Modelo educacional “Faça o seu app”.	107
Figura 41: Processo de desenvolvimento de apps.	115
Figura 42: Trecho da rubrica CodeMaster v2.0.	122
Figura 43: Menu de cores do <i>App Inventor</i>	129
Figura 44: Paleta de cores do <i>Material Design</i>	129
Figura 45: Comparação do menu cor de fundo.	130
Figura 46: Catálogo de ícones do <i>Material Design</i>	131
Figura 47: Painel semântico com fotos relacionado ao natal.	133
Figura 48: Definindo cor dominante.	134
Figura 49: Definindo a cor primária.	134
Figura 50: Cores complementares no círculo cromático.	135
Figura 51: Cores análogas no círculo cromático.	136
Figura 52: Tonalidades da cor cerceta.	136
Figura 53: Paleta de cores.	137
Figura 54: Contraste para textos.	137
Figura 55: Tela com interface de usuário.	138
Figura 56: Modelo de uma <i>persona</i>	139
Figura 57: Modelo de história de usuário.	140
Figura 58: Diagrama de uso.	141
Figura 59: Estrutura do <i>App Inventor</i>	143
Figura 60: Esquematização do código fonte.	146
Figura 61: Diagrama de classe pertinente a USC01.	148
Figura 62: Paleta de cores desenvolvida pertinente a USC02.	151

Figura 63: Menu de cores das propriedades PrimaryColorCNE e SecondaryColorCNE.....	152
Figura 64: Diagrama de classe pertinente a USC02.....	154
Figura 65: Diagrama de classes pertinente a USC03 e USC04.	159
Figura 66: Primeiro encontro do projeto na UFSC.....	171
Figura 67: Encontros de ensino do processo de desenvolvimento de apps 1.	172
Figura 68: Alunos programando seu próprio <i>app</i>	173
Figura 69: <i>Workbook</i> do app “InfoCarvalho”.....	174
Figura 70: Apps desenvolvidos na unidade instrucional versão longa.....	175
Figura 71: Oficinas realizadas na escola.	176
Figura 72: Alunos nas atividades de <i>design visual</i>	176
Figura 73: Exemplos de design visual do app criado pelos alunos.....	177
Figura 74: Análise dos dados sobre o nível de conhecimento antes e depois da unidade.	178
Figura 75: Distribuição da frequência de itens relacionados à percepção da aprendizagem (pós-aplicação).....	179
Figura 76: Quantidade de alunos que conseguem explicar tópicos da UI.	180
Figura 77: Quantidade de acertos em cada questão.	180
Figura 78: Avaliando a aprendizagem de conceitos de programação usando CodeMaster 2.0.	181
Figura 79: Distribuição da frequência de respostas relacionadas a diversão.	185
Figura 80: Frequência de itens relacionados ao tempo das aulas.....	186
Figura 81: Distribuição da frequência da avaliação da qualidade dos encontros. ..	186
Figura 82: Resultado dos itens relacionados a interação social.	187
Figura 83: Distribuição da frequência de itens relacionados a sua facilidade de aprendizagem.	188
Figura 84: Distribuição da frequência de itens relacionados às aulas e o curso em geral.....	189
Figura 85: Distribuição da frequência de itens relacionados ao interesse na aprendizagem.	190
Figura 86: Distribuição da frequência de conteúdo relacionado a sua importância.	190
Figura 87: Percepção de aprendizagem dos conteúdos.....	191
Figura 88: Quantidade de alunos que conseguem explicar tópicos da UI.	192
Figura 89: Frequência dos níveis de desempenho do design visual.	193

Figura 90: Distribuição da frequência de respostas relacionadas a diversão.....	194
Figura 91: Frequência de itens relacionados ao tempo das aulas.	194
Figura 92: Resultado dos itens relacionados a interação social.....	195
Figura 93: Distribuição da frequência da avaliação da qualidade dos encontros. ...	195
Figura 94: Distribuição da frequência de itens relacionados às aulas e o curso em geral.	196
Figura 95: Distribuição da frequência de itens relacionados a sua facilidade de aprendizagem.....	197
Figura 96: Distribuição da frequência de itens relacionados ao interesse na aprendizagem.....	197
Figura 97: Distribuição da frequência de conteúdo relacionado a sua importância.	198

LISTA DE QUADROS

Quadro 1: Áreas de conhecimento da Engenharia de Software	30
Quadro 2 Características de um dispositivo móvel.....	33
Quadro 3: Contexto de uso.....	33
Quadro 4: Projeto para desenvolvimento de <i>apps</i>	34
Quadro 5: Características da Metodologia Ágil.....	36
Quadro 6: O processo do <i>Design Thinking</i>	45
Quadro 7: Níveis de aprendizagem do domínio cognitivo da taxonomia de Bloom. .	49
Quadro 8: Níveis de aprendizagem do domínio afetivo da taxonomia de Bloom.....	50
Quadro 9: Níveis de aprendizagem do domínio psicomotor de Simpson	50
Quadro 10: Métodos Instrucionais.	51
Quadro 11: Instrumento de avaliação.....	60
Quadro 12: Objetivos de aprendizagem de computação para Ensino Fundamental II.	58
Quadro 13: Objetivos de aprendizagem referente a ES e EU voltado para Ensino Fundamental II.....	64
Quadro 14: Comandos do <i>App Inventor</i> agrupados em categorias.	64
Quadro 15: Palavras-chave.....	73
Quadro 16: <i>String</i> de busca.	68
Quadro 17: Artigos relevantes.	69
Quadro 18: Visão geral das características de contexto das UIs.....	77
Quadro 19: Palavras-chave.	88
Quadro 20: <i>Strings</i> de busca.	88
Quadro 21: Artigos relevantes.	91
Quadro 22: Objetivos de aprendizagem da UI.....	104
Quadro 23: Plano de ensino da unidade instrucional “Faça seu <i>app</i> ”.	109
Quadro 24: Plano de ensino da unidade instrucional “Faça um <i>app</i> ”.	112
Quadro 25: Materiais didáticos desenvolvidos.....	116
Quadro 26: Rubrica para avaliação da aprendizagem relacionada aos artefatos criados.	123
Quadro 27: Rubrica para avaliação da aprendizagem sobre design visual.	124
Quadro 28: Questões aplicadas aos alunos da UI.....	125
Quadro 29: Requisitos funcionais em relação a UI.....	127

Quadro 30: Passos para incluir um ícone no Botão do <i>App Inventor</i>	131
Quadro 31: Definição de cores para elementos e componentes do <i>App Inventor</i> ...	138
Quadro 32: Classes editadas.	147
Quadro 33: Fluxo principal para selecionar um ícone.	148
Quadro 34: Classes editadas.	152
Quadro 35: Classes criadas.	153
Quadro 36: Fluxo principal para definir uma paleta de cores e aplicar em um <i>app</i> .	156
Quadro 37: Classes modificadas.,.....	157
Quadro 38: Fluxo principal para definir uma <i>Persona</i>	159
Quadro 39: Fluxo principal para definir uma história de usuário.	161
Quadro 40: Casos de teste.....	164
Quadro 41: Sugestões de melhorias implementadas.	166
Quadro 42: Síntese da definição da avaliação da UI.	169
Quadro 43: Descrição dos apps desenvolvidos na unidade instrucional versão longa.	175
Quadro 44: Comentários qualitativos (números indicando a frequência de citações).	187

LISTA DE TABELAS

Tabela 1: Estrutura do sistema de Ensino Básico.	56
Tabela 2: Quantidade de artigos por etapa de seleção por repositório.	69
Tabela 3: Quantidade de artigos por etapa de seleção por repositório.	89
Tabela 4: Pontuação das interfaces dos apps desenvolvidos.	182
Tabela 5: Pontuação de estéticas dos apps.	183
Tabela 6: Resultado das avaliações de desempenho dos artefatos dos apps criados pelos alunos.....	184
Tabela 7: Comentários qualitativos (números entre parênteses indicam a frequência de citações por cada categoria).....	196

LISTA DE ABREVIATURAS E SIGLAS

ACM - *Association for Computing Machinery*

APP - Aplicativo

CASE - *Computer-Assisted Software Engineering*

CSTA - *Computer Science Teachers Association*

DT - *Design Thinking*

EF - Ensino Fundamental

EM - Ensino Médio

ES - Engenharia de Software

EU - Engenharia de Usabilidade

GQM - *Goal Question Metric*

IEC - *International Electrotechnical Commission*

IEEE - *Institute of Electrical and Electronics Engineers*

INE - Departamento de Informática e Estatística

ISO - *International Standard Organization*

MEC - Ministério da Educação

PPGCC - Programa de Pós-Graduação em Ciência da Computação

SBC - Sociedade Brasileira de Computação

UI - Unidade Instrucional

UIs - Unidades Instrucionais

UX - *User Experience*

VPL – *Visual Programming Language*

XP - *Extreme Programming*

SUMARIO

AGRADECIMENTOS.....	5
RESUMO	6
ABSTRACT.....	7
LISTA DE FIGURAS.....	8
LISTA DE QUADROS.....	12
LISTA DE TABELAS.....	14
LISTA DE ABREVIATURAS E SIGLAS.....	15
1. INTRODUÇÃO	20
1.1. CONTEXTUALIZAÇÃO DO PROBLEMA.....	20
1.2. PERGUNTA DE PESQUISA	22
1.3. OBJETIVOS	22
1.3.1. Objetivo geral	22
1.4. DELIMITAÇÕES.....	23
1.5. ADERÊNCIA À CIÊNCIA DA COMPUTAÇÃO	24
1.6. METODOLOGIA DE PESQUISA	25
1.6.1. Contexto de pesquisa e classificação	25
1.7. ETAPAS DA PESQUISA.....	26
1.8. CONTRIBUIÇÕES	28
2. FUNDAMENTAÇÃO TEÓRICA	30
2.1. ENGENHARIA DE SOFTWARE	30
2.2. PROCESSO DE DESENVOLVIMENTO DE APPS.....	32
2.3. METODOLOGIA ÁGIL PARA DISPOSITIVOS MÓVEIS.....	35
2.4. INTERAÇÃO HUMANO-COMPUTADOR.....	39
2.5. PROCESSOS VOLTADOS À ENGENHARIA DE USABILIDADE.....	40
2.6. DESIGN DE EXPERIÊNCIA DO USUÁRIO PARA DISPOSITIVOS MÓVEIS	42
2.7. PROCESSO DE <i>DESIGN THINKING</i> PARA DISPOSITIVOS MÓVEIS	44
2.8. ENSINO E APRENDIZAGEM POR MEIO de design instrucional	46
2.8.1 MODELO ADDIE	47
2.9. O ENSINO DA COMPUTAÇÃO NO ENSINO BÁSICO.....	55
2.9.1. Framework Computer Science Teacher Association K-12	56
2.10. DESENVOLVIMENTO DE APPS COM APP INVENTOR	60
3. ESTADO DA ARTE	67

3.1. ESTADO DA ARTE DO ENSINO DE ES NA EDUCAÇÃO BÁSICA.....	67
3.1.1. Execução da busca	68
3.1.2. Análise dos dados	69
3.1.3. Discussão	83
3.1.4. Ameaças à validade	86
3.2. ESTADO DA ARTE DO ENSINO DE EU NA EDUCAÇÃO BÁSICA	87
3.2.1. Execução da busca	89
3.2.2. Análise de dados	90
3.2.3. Discussão	99
3.2.4. Ameaças à validade	100
4. MODELO EDUCACIONAL - “CURSO FAÇA O SEU APP”	102
4.1. ANÁLISE DO CONTEXTO	102
4.2. PROJETO	103
4.2.1. Objetivo de aprendizagem	103
4.2.2. Contexto de aplicação	106
4.2.3. Plano de ensino	106
4.3. MODELO DE DESENVOLVIMENTO DE APPS	114
4.4. DESENVOLVIMENTO DO MATERIAL DIDÁTICO	115
4.5. AVALIAÇÃO DO ALUNO	122
5. EVOLUÇÃO DO APP INVENTOR	127
5.1. ANÁLISE DOS REQUISITOS	127
5.2. CASOS DE USO	140
5.3. CONTEXTO DO FUNCIONAMENTO DO APP INVENTOR	143
5.4. CONFIGURAÇÃO DO AMBIENTE DE DESENVOLVIMENTO	144
5.5. ESTRUTURA DO CÓDIGO DO APP INVENTOR	145
5.6. IMPLEMENTAÇÃO	146
5.7. TESTE	162
6. INSTANCIAÇÃO, APLICAÇÃO E AVALIAÇÃO DA UNIDADE INSTRUCIONAL	168
6.1. AVALIAÇÃO do modelo educacional	168
6.1.1. Definição de avaliação	168
6.2. INSTANCIAÇÃO E APLICAÇÃO DA UI	170
6.2.1. Aplicação da versão longa - Jovens Tutores 2018	170
6.2.3. Aplicação da versão curta – ECOPET 2019-1	176
6.3. ANÁLISE DOS DADOS	177

6.3.1. Análise dos dados da aplicação da unidade instrucional versão longa – Jovens Tutores.....	177
6.3.2. Análise dos dados da aplicação da unidade instrucional versão curta 190	
6.4. DISCUSSÃO	198
6.5. AMEAÇAS À VALIDADE.....	200
7.CONCLUSÃO.....	202
REFERÊNCIAS	204
Anexo 1 - Medição pós treinamento - Jovens tutores – Alunos	220
Anexo 2 - Resultado de avaliação do design visual dos <i>apps</i> criado no Jovens Tutores 2018/2.	230
Anexo 3 - Pontuação dos <i>apps</i> em cada tópico dos <i>apps</i> criados pelos Jovens Tutores 2018/2.	231

1. INTRODUÇÃO

1.1. CONTEXTUALIZAÇÃO DO PROBLEMA

Para um cidadão estar preparado para as carreiras do século XXI, independentemente da sua área de conhecimento, é importante que este tenha uma compreensão clara dos princípios e práticas da Ciência da Computação (CSTA, 2017). Esta compreensão estimula o interesse das pessoas, sendo que no Brasil há uma necessidade de formar profissionais para o setor de software, pois existe um déficit de profissionais na área (CARDOSO & DAVID, 2017).

As competências da Ciência da Computação não se tratam apenas de saber navegar na internet, enviar um e-mail e utilizar as ferramentas de um computador, como um editor de texto ou uma planilha eletrônica, mas sim compreender o pensamento computacional e a autonomia para resolução de problemas (CAMBRAIA & SCAICO, 2013). O pensamento computacional se refere à capacidade de representar, analisar e resolver problemas por meio de um computador (SBC, 2018). Este conhecimento aumenta o poder cognitivo do aluno para analisar, implementar, testar e implantar a resolução de um problema por meio de um computador em diversas áreas de conhecimento (CSTA, 2016). Neste cenário, é realçada a importância de se fornecer competências de computação no campo da educação.

Para popularizar a computação, ela deve ser ensinada desde cedo, a exemplos de outras áreas de conhecimento, como a Física, a Matemática, a Química e a Biologia (SBC, 2017). Visando a inclusão do ensino da computação na Educação Básica, existem diversas iniciativas, como o Code.org (CODE, 2013), o Code.club (CODE CLUB, 2012) e a Computação na Escola (CNE, 2019). Existem também diversos ambientes de programação visual baseados em blocos e voltados ao ensino de computação na Educação Básica, como por exemplo Scratch¹, SNAP!² e App Inventor³. Estes ambientes de programação são tipicamente utilizados para criar animações ou jogos (SCRATCH & MIT, 2013) ou aplicativos móveis, robôs etc. Existem também vários currículos ou frameworks de referência para o ensino da computação na Educação Básica, como CSTA K-12 (CSTA, 2017), SBC (2017a), SAMSUNG (2015), etc. Vários cursos completos alinhados a estes currículos de

¹ scratch.mit.edu

² byob.berkeley.edu

³ appinventor.mit.edu

referência foram criados, como por exemplo, Code.org (CODE, 2013), Google (GOOGLE & CS4HS, 2009), Scratched (SCRATCHED, 2009), tipicamente compostos de diversos exercícios de programação predefinidos.

Uma alternativa para o ensino da computação para o público jovem é ensinar a desenvolver aplicativos (apps) em celulares, utilizando o App Inventor, uma vez que o celular se torna comum entre os adolescentes e que 93% dos adolescentes usam celular para acessar a internet (CGI.BR, 2018). O App Inventor (MIT & GOOGLE, 2019) é uma ferramenta de desenvolvimento online construída pela Google, sendo mantida pelo MIT (*Massachusetts Institute of Technology*), e que permite a qualquer pessoa, até mesmo uma criança, criar aplicativos para dispositivos móveis (celulares, tablets) que utilizam o sistema operacional Android. Ao criar apps com o App Inventor, crianças aprendem a pensar criativamente, a trabalhar de forma colaborativa e a pensar de forma sistemática na solução de problemas. Neste contexto, já existem alguns tutoriais predefinidos que ensinam a criar apps (MIT, 2019a). Também foram desenvolvidas unidades instrucionais e aplicadas em escolas de ensino médio com o objetivo de ensinar conceitos de computação (CHATZINIKOLAKIS & PAPADAKIS, 2014). Estas unidades instrucionais (UIs) geralmente desenvolvem jogos e animações (MEERBAUM-SALANT, 2013), e têm como enfoque principal o ensino da programação. Porém, o corpo de conhecimento de ciência da computação também abrange outras áreas, como engenharia de software (ES) e/ou engenharia de usabilidade (EU), que são essenciais para o desenvolvimento de um app de sucesso.

A Engenharia de Software é uma disciplina que está relacionada com a aplicação de modelos, práticas e técnicas para construir, de forma efetiva e eficiente, sistema de softwares confiáveis e que atendam às necessidades dos usuários (ACM/IEEE 2013). É relevante o ensino de competências desta disciplina associado ao ensino da computação para a Educação Básica, mais especificamente das atividades pertinente ao processo de ciclo de vida de software, como por exemplo, testes de software (CSTA, 2017). Complementarmente, a EU é uma disciplina que define aspectos ergonômicos para a ciência da computação, e trata de como projetar um software com uma boa usabilidade (NIELSEN, 1992). A capacidade de desenvolver sistemas de software com boa usabilidade é uma competência importante no ensino de computação, para que os alunos possam projetar interfaces visando a boa usabilidade na interação entre o usuário e o aplicativo (CSTA, 2017).

O design de interface de usuário e o *design thinking* são competências fundamentais pertinentes à esta disciplina, e são importantes não apenas para os profissionais de Tecnologia da Informação (TI), mas para qualquer pessoa, pois estão relacionadas às habilidades do século XXI (AIGA, 2013). Estas competências visam maximizar a usabilidade, utilidade e a experiência do usuário (COOPER, 2014). Embora estas competências sejam ensinadas, geralmente, apenas no ensino superior, diretrizes curriculares, como a K-12 Computer Science Framework (CSTA, 2017), indicam sua importância também na Educação Básica.

Mesmo que algumas unidades instrucionais já incluam algumas práticas de ES e/ou EU, como testes de software, fluxograma, design de interação e análise de contexto (requeridos pelo CSTA (2017)), elas tipicamente não abordam essas questões de forma mais abrangente. Neste contexto, este trabalho pretende customizar um modelo para ensinar alunos a desenvolver aplicativos integrando práticas da ES e EU no contexto do Ensino Fundamental.

1.2. PERGUNTA DE PESQUISA

A pergunta de pesquisa que norteia o desenvolvimento do presente trabalho é: como é possível, de forma motivadora, ensinar conceitos/conteúdos de ES e EU por meio do desenvolvimento de aplicativos móveis na Ensino Fundamental?

1.3. OBJETIVOS

1.3.1. Objetivo geral

Este trabalho tem como objetivo o desenvolvimento de um modelo de ensino de ES e EU para o desenvolvimento de aplicativos móveis no Ensino Fundamental, utilizando o ambiente de programação visual *App Inventor*. O desenvolvimento deste modelo engloba a definição de um processo de desenvolvimento de *apps* integrando sistematicamente práticas de ES e EU.

O conteúdo do modelo está alinhado ao modelo de currículo de referência CSTA/ACM K-12 (CSTA, 2017), e é operacionalizado por meio de uma unidade instrucional que contém materiais didáticos, como slides, roteiros, folhas de tarefas e avaliações. Será também evoluído o suporte oferecido pelo *App Inventor* para atender o ensino do processo de desenvolvimento de software. O modelo/unidade instrucional desenvolvido é instanciado, aplicado e avaliado em escolas e avaliado

em relação à qualidade da UI, experiência e percepção da aprendizagem do ponto de vista dos alunos (GRESSE et. al., 2017).

Objetivos específicos. Os objetivos específicos são:

OE1. Sintetizar um corpo teórico sobre o ensino de computação na Educação Básica, ambiente de programação visual *App Inventor*, conceitos e princípios relacionado às disciplinas de ES e EU;

OE2. Levantar o estado da arte e prática por meio de mapeamento sistemático da literatura, para entender como atualmente os conceitos de ES e EU são ensinados ao nível da Educação Básica;

OE3. Propor um modelo educacional para o ensino de desenvolvimento de aplicativos integrando o ensino de ES e EU;

OE3.1. Definir plano de ensino incluindo os objetivos de aprendizagem;

OE3.2. Definir de um processo de ES e EU voltado ao desenvolvimento de aplicativos no contexto do Fundamental 2;

OE3.3. Desenvolver materiais didáticos, como por exemplo, slides, roteiros, folhas de tarefas e avaliações;

OE3.4. Adaptar/evoluir o *App Inventor* para apoiar o ensino do processo de desenvolvimento de software;

OE4. Aplicar a unidade instrucional desenvolvida em escola e avaliar em relação à aprendizagem dos alunos e experiência da aprendizagem.

1.4. DELIMITAÇÕES

Este trabalho tem como principal foco o ensino de práticas de EU e ES no Ensino Fundamental, pois, analisando o estado da arte observa-se que as UIs criadas para o ensino da computação são voltadas para os cursos técnicos e superiores, ou então focam apenas em ensinar programação. Não são incluídas competências de outras disciplinas além da ES e EU para o ensino da computação.

O escopo da unidade instrucional visa o ensino de desenvolvimento de *apps* para os alunos do Ensino Fundamental. Desta forma, o modelo/unidade instrucional

é desenvolvido para os alunos do 6º ao 9º ano. Assume-se que poderá ser também utilizado em outros níveis da Educação Básica, porém o público alvo no contexto do presente projeto e em que os resultados são avaliados é principalmente este. A unidade instrucional limita-se em ensinar o desenvolvimento de aplicativos para as plataformas móveis que utilizam *Android* como sistema operacional.

Este trabalho aborda o ensino da computação utilizando exclusivamente a ferramenta de desenvolvimento *App Inventor*. Esta ferramenta foi escolhida por ser indicada para as pessoas que nunca programaram e desejam desenvolver seus primeiros aplicativos (WOLBER; ABELSON & FRIEDMAN, 2015). Portanto, o trabalho não abrange a utilização de outras ferramentas, como por exemplo, Scratch SNAP! e Blockly.

1.5. ADERÊNCIA À CIÊNCIA DA COMPUTAÇÃO

O tema deste trabalho está em conformidade aos objetivos do Programa de Pós-graduação em Ciência da Computação da Universidade Federal de Santa Catarina (PPGCC/UFSC), na linha de pesquisa de ES. Especificamente, insere-se nos tópicos de Processo de Desenvolvimento de Software e Qualidade de Software, conforme a definição da área de engenharia de software da Sociedade Brasileira de Computação (SBC,2005).

O regimento interno nº 01/PPGCC/2013, publicado em 01/10/2013, em seu artigo 1º, define os objetivos do programa como “desenvolvimento de novos conhecimentos em Ciência da Computação”. Dentre os conhecimentos que integram a Ciência da computação está a disciplina de ES. Esta aborda diversos tópicos, dentre elas a Qualidade de Software, que tem como um dos principais fatores a usabilidade (IEEE CS, 2014). Para assegurar a usabilidade de produtos de software é necessário a implementação da EU, sendo uma subárea de ES.

Neste sentido, este trabalho está integrado à Ciência da Computação por desenvolver um modelo de ensino de conceitos de ES e EU, além da definição de um processo de software customizado, para possibilitar o ensino destes conceitos no Ensino Fundamental.

Analisando com mais detalhe os objetivos da linha de pesquisa em ES do PPGCC/UFSC, observa-se a aderência da presente dissertação a estes objetivos, que são:

“Engenharia de Software: tem como objetivo formar indivíduos capazes de conduzir o processo de desenvolvimento de software e de investigar novas metodologias, técnicas e ferramentas para a concepção de sistemas” (PPGCC/UFSC, 2015).”

Posto isto, esta dissertação aborda primordialmente o ensino ES e EU no desenvolvimento de *apps*. Considerado que a ES é parte fundamental do processo de desenvolvimento de software de qualidade, estando presente entre os conhecimentos definidos pelo Software *Engineering Body of Knowledge* (SWEBOK) (IEEE CS, 2014), entende-se que há correlação do tema da dissertação a este objetivo da linha de pesquisa.

1.6. METODOLOGIA DE PESQUISA

1.6.1. Contexto de pesquisa e classificação

Esta pesquisa é classificada, quanto à sua natureza, como aplicada, pois o modelo educacional desenvolvido neste trabalho poderá ser utilizado por instrutores para ensinar computação no Ensino Fundamental. Quanto ao objetivo, a pesquisa se classifica como exploratória, pois para atingir o objetivo deste trabalho foi preciso analisar o estado da arte e levantamentos bibliográficos. Também é classificada como descritiva, uma vez que para inferir as conclusões gerais foi preciso aplicar questionários e observações no ensino da UI.

Detalhando o contexto de pesquisa deste trabalho com base no modelo científico em camadas (*Research-Process Onion*) (SAUDERS, LEWIS & THORNHILL, 2009), a pesquisa é classificada como:

a) *Cross-sectional*: pois é feita a análise dos indivíduos, no caso, dos alunos, durante o período de aplicação da UI;

b) *Multimétodo*: pois utiliza técnicas qualitativas e quantitativas, como procedimentos de pesquisa bibliográfica e documental (GIL, 2010), mapeamento sistemático da literatura (PETERSEN *et al.*, 2008), modelo ADDIE (BRANCH, 2009), estudo de caso (YIN, 2009), Goal Question Metric (GQM) (BASILI, CALDIERA & ROMBACH, 1994), entre outros;

c) *Estratégica*: são utilizadas diversas estratégias, tais como avaliação de desempenho, questionários e observação, por exemplo;

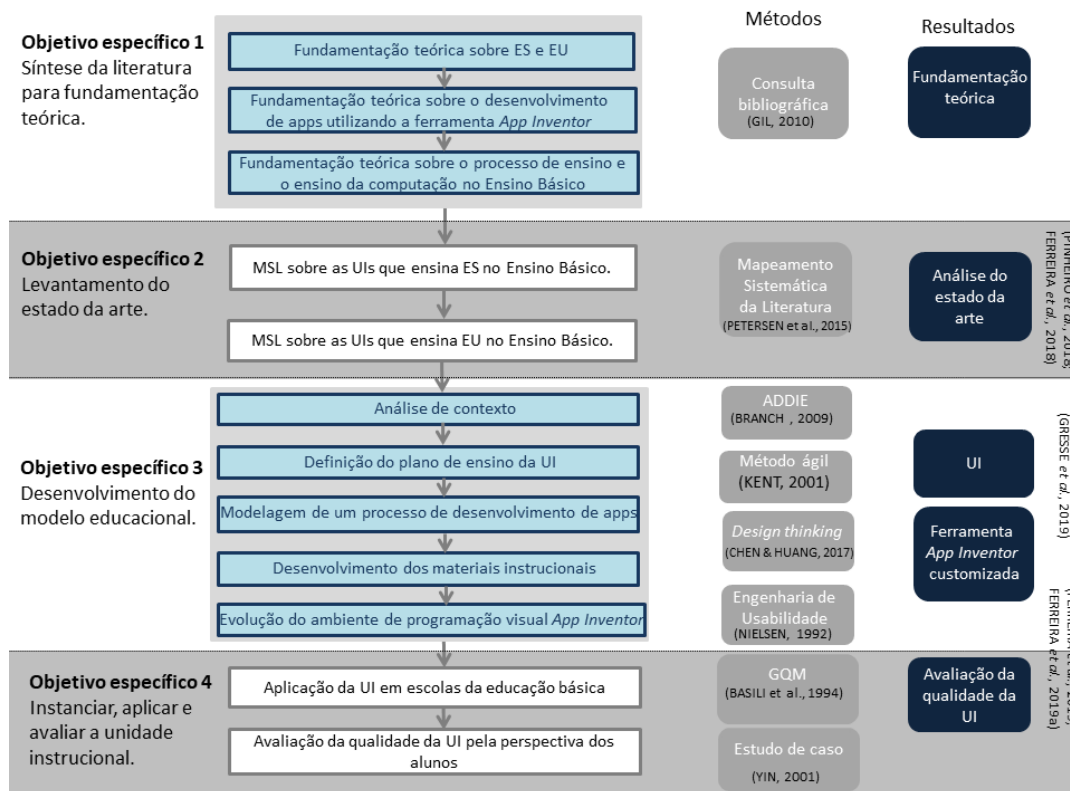
d) *Indutiva*: são analisados estudos de caso particulares de aplicação da unidade instrucional por meio de avaliações para inferir as conclusões gerais e para se basear na solução do problema;

e) *Interpretativista*: para atingir o objetivo desta pesquisa é preciso interpretar as informações coletadas durante a aplicação da UI.

1.7. ETAPAS DA PESQUISA

Para cada objetivos específicos são apresentados os métodos utilizados para sua realização e os resultados produzidos (Figura 1).

Figura 1 - Etapas da pesquisa.



Fonte: elaborada pelo autor.

O detalhamento do trabalho realizado em cada etapa é detalhado a seguir:

a) *Etapa 1 - Síntese da literatura para fundamentação teórica*

O objetivo desta etapa foi sintetizar os conceitos teóricos relevantes para esta dissertação. Isto inclui a síntese dos conceitos de ES e EU, voltados para o desenvolvimento de *apps*, e uma análise teórica sobre desenvolvimento de *apps* utilizando o ambiente de programação visual *App Inventor*. Além disso, são descritos

conceitos do processo de ensino e aprendizagem da Computação na Educação Básica. Esta etapa é feita por meio de consultas bibliográficas (GIL, 2010).

b) Etapa 2 - Levantamento do estado da arte

Nesta etapa foi feito o levantamento do estado da arte para entender como o ensino da computação é atualmente realizado e avaliado nas escolas. Para isso, foi feito um Mapeamento Sistemático da Literatura (MSL), no qual são identificadas e analisadas as principais unidades instrucionais/estratégias de ensino de computação aplicando práticas de ES e EU na Educação Básica.

O MSL segue o procedimento proposto por Petersen *et al.* (2008). Este procedimento inclui o planejamento do MSL, definindo o objetivo e a pergunta de pesquisa, a determinação das palavras-chave do estudo, as fontes de consulta, os critérios de inclusão/exclusão e de qualidade, para a definição do protocolo de busca. Após, foi executada a busca das referências, e a partir dos trabalhos encontrados foram extraídas as informações relevantes, que são presentemente analisadas em relação à pergunta de pesquisa.

c) Etapa 3 - Desenvolvimento do modelo educacional

Para o desenvolvimento do modelo educacional foi utilizado o método ADDIE (BRANCH, 2009), uma abordagem para o Design Instrucional. Na fase de Análise, foi realizada uma análise de contexto, que envolve a identificação e a caracterização do público alvo e as limitações de recursos para o ensino de desenvolvimento de aplicativos, relacionadas a fatores como infraestrutura, recursos humanos e tempo. Na fase de Projeto é definido o plano de ensino, que contém os objetivos do ensino, o conteúdo a ser abordado e o seu sequenciamento, bem como a estratégia e os métodos instrucionais. Também nesta etapa são definidas as estratégias instrucionais e outras atividades voltadas ao ensino. Na fase de Desenvolvimento são elaborados os materiais instrucionais, como slides, rubricas de avaliação e a customização de um modelo de processo de desenvolvimento de software. O conteúdo deste modelo customizado é baseado em análises feitas em consultas bibliográficas obtidas na fundamentação teórica, revisão do estado da arte e com base na análise do contexto instrucional (CSTA, o público alvo, infraestrutura, tempo disponível). As etapas deste processo de desenvolvimento são baseadas na abordagem *Design Thinking* (CHEN & HUANG, 2017), e conforme o modelo vai sendo desenvolvida o ambiente de programação visual *App Inventor* é

evoluído/adaptado. Os requisitos de modificações são levantados com base nas necessidades de adaptação ao modelo de processo de software customizado.

d) Etapa 4 – Instanciação, aplicação e avaliação da UI

A partir do modelo educacional são criadas instâncias de unidades instrucionais aplicadas e avaliadas por uma série de 2 estudos de caso (YIN, 2009). A avaliação tem como objetivo: i) avaliar o desempenho dos alunos por meio de uma série de avaliações no decorrer do ensino; e, ii) avaliar a qualidade da unidade instrucional sob a perspectiva dos alunos, analisando a percepção de aprendizagem, conteúdo, materiais, os objetivos de aprendizagem, estratégia instrucional. Para avaliar a qualidade da unidade instrucional é definido um plano de medição e os instrumentos de coleta de dados a serem analisados utilizando a abordagem *Goal Question Metric* (GQM) (BASILI, CALDIERA & ROMBACH, 1994). Nesse estudo é adotado o modelo de avaliação dTECT (GRESSE *et al.*, 2017) e MEEGA+KIDS (GRESSE VON WANGENHEIM *et al.*, 2018). Durante a aplicação são coletados os dados. Ao final, os dados coletados foram analisados e interpretados em relação à pergunta de pesquisa.

A aplicação da unidade instrucional foi aprovada pela da Comissão de Ética de Pesquisa com Seres Humanos da Universidade Federal de Santa Catarina com o número do parecer 2.677.698.

1.8. CONTRIBUIÇÕES

A realização desta dissertação produz contribuições nos âmbitos científico, tecnológico e social.

Contribuição científica. Este trabalho tem como principal contribuição científica a elaboração de um modelo de ensino de ES e EU para o Ensino Fundamental 2. Este modelo possibilitará aos alunos aprender a aplicar os conceitos dessas áreas de conhecimento no desenvolvimento de *apps* confiáveis e de qualidade. Como parte deste modelo este trabalho também define um processo de desenvolvimento de *apps* integrando práticas de ES e EU customizado ao contexto ensino de computação do Ensino Fundamental 2.

Outras contribuições incluem o levantamento do estado da arte atual em relação aos trabalhos existentes, sintetizando uma visão geral em relação a UIs para o ensino da computação no Ensino Fundamental 2. O trabalho também gera dados

e resultados do impacto da aplicação das instâncias do modelo em relação a aprendizagem dos alunos e experiência de aprendizagem.

Contribuição tecnológica. Em relação ao âmbito tecnológico, a contribuição deste trabalho é o aprimoramento do ambiente de programação visual *App Inventor* para suportar adequadamente o processo e o modelo de ensino desenvolvido. O aprimoramento desta ferramenta permitirá introduzir a ES/EU na Educação Básica, fornecendo o suporte necessário para poder executar todo o processo e usando exclusivamente o *App Inventor*. Sendo de código aberto, a evolução desta ferramenta é disponibilizada sob a mesma licença ao público em geral, e possibilita a sua ampla adoção por interessados (escolas, professores, instituições de ensino, etc.).

Contribuição social. As contribuições sociais do presente projeto são significantes, já que neste âmbito as escolas do Ensino Fundamental poderão utilizar a unidade instrucional desenvolvida, incluindo todo material didático e a versão aprimorada do *App Inventor*. Dessa forma, os professores poderão ensinar, além da programação e do pensamento computacional, a ES, EU, tornando o ensino da computação mais completa. Assim, este trabalho completa a formação dos alunos da Educação Básica no que se refere ao ensino de computação, visto que é a tendência do mercado o cidadão ter conhecimento desta área de conhecimento independente da área de estudo e carreira.

De forma geral, o presente projeto visa contribuir com a popularização da computação de forma prática à sua aplicação e para o crescimento desta área de conhecimento. Além disso, contribui com a formação da população em geral, como também estimula o interesse para a atuação nesta área.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são abordados os principais conceitos de ES e EU, explanando os seus processos e atividade no contexto do desenvolvimento de *app* móveis. É abordada também a conceituação do processo de aprendizagem e ensino, focando no ensino da computação na Educação Básica no Brasil. Voltado ao objetivo do presente trabalho, são apresentados também os conceitos referentes ao desenvolvimento de *app* móveis utilizando *App Inventor*.

2.1. ENGENHARIA DE SOFTWARE

A engenharia de software (ES) é uma área de conhecimento da computação que define abordagens sistemáticas, disciplinadas e quantificáveis para o desenvolvimento, operação e manutenção de software (IEEE, 2010). Esta área de conhecimento envolve diversas áreas de conhecimento (IEEE CS, 2013), conforme o quadro 1.

Quadro 1: Áreas de conhecimento da Engenharia de Software

Áreas de conhecimento	Descrição
Requisitos de Software	Esta área preocupa-se com a gestão dos requisitos, bem como a elicitaco, anlise, especificaco e validaco de requisitos e das restrioes de software. Geralmente é uma fase em que o desenvolvedor interage com o cliente para definir as características do novo software.
Modelagem de Software	Define arquitetura, componentes, interfaces e outras características de um sistema ou componente.
Construo de Software	Esta área envolve a implementaco do software, verificaco, testes de unidade, testes de integrao e <i>debugging</i> .
Teste de Software	Área que aborda a verificaco dinâmica que um programa fornece e comportamentos esperados em um conjunto finito de casos de teste. Assim, o software é validado para garantir que todas as funcionalidades especificadas foram implementadas.
Manuteno de Software	Envolve atividades para dar suporte a um sistema de software, que pode ocorrer antes ou depois da entrega.
Gerenciamento e Configurao de Software	Esta área estabelece e mantém a integridade dos produtos de software durante todo seu ciclo de vida por meio de controle de verso.
Gerenciamento de Engenharia de Software	Esta área define como a aplicao de atividades de planejamento, coordenao, medio, monitoramento e controle garantem que o software e servios serão entregues de forma eficiente, eficaz e em benefício das partes interessadas.
Processo de Engenharia de Software	Preocupa-se com atividades de trabalho realizado por engenheiros de software para desenvolver, manter e operar software, como requisitos, design, construo, teste, configurao, gestão e outras engenharias de software de processos.
Modelos e métodos de Engenharia	Foca em impor estrutura na engenharia de software, com o

de Software	objetivo de tornar essa atividade sistemática, repetível e, finalmente, mais orientada para o sucesso.
Qualidade de Software	Visa estabelecer métodos e tecnologias para construir produtos de software de qualidade dentro dos limites de tempo e recursos disponíveis.
Práticas Profissionais de Engenharia de Software	Preocupa-se com o conhecimento, habilidades e atitudes que os engenheiros de software devem possuir para praticar a engenharia de software de forma profissional, responsável e ética.
Economia de Engenharia de Software	Área focada sobre como fazer decisões relacionadas à engenharia de software em um contexto empresarial.
Áreas fundamentais da engenharia de software	
Fundamentos da Computação	Engloba o desenvolvimento e o ambiente operacional em que o software evolui e é executado.
Fundamentos Matemáticos	Está área de conhecimento auxilia engenheiros de software a compreender a lógica para que possa ser traduzida em código de linguagem de programação.
Fundamentos da Engenharia	Inclui habilidades e técnicas fundamentais de engenharia que são úteis para um engenheiro de software.

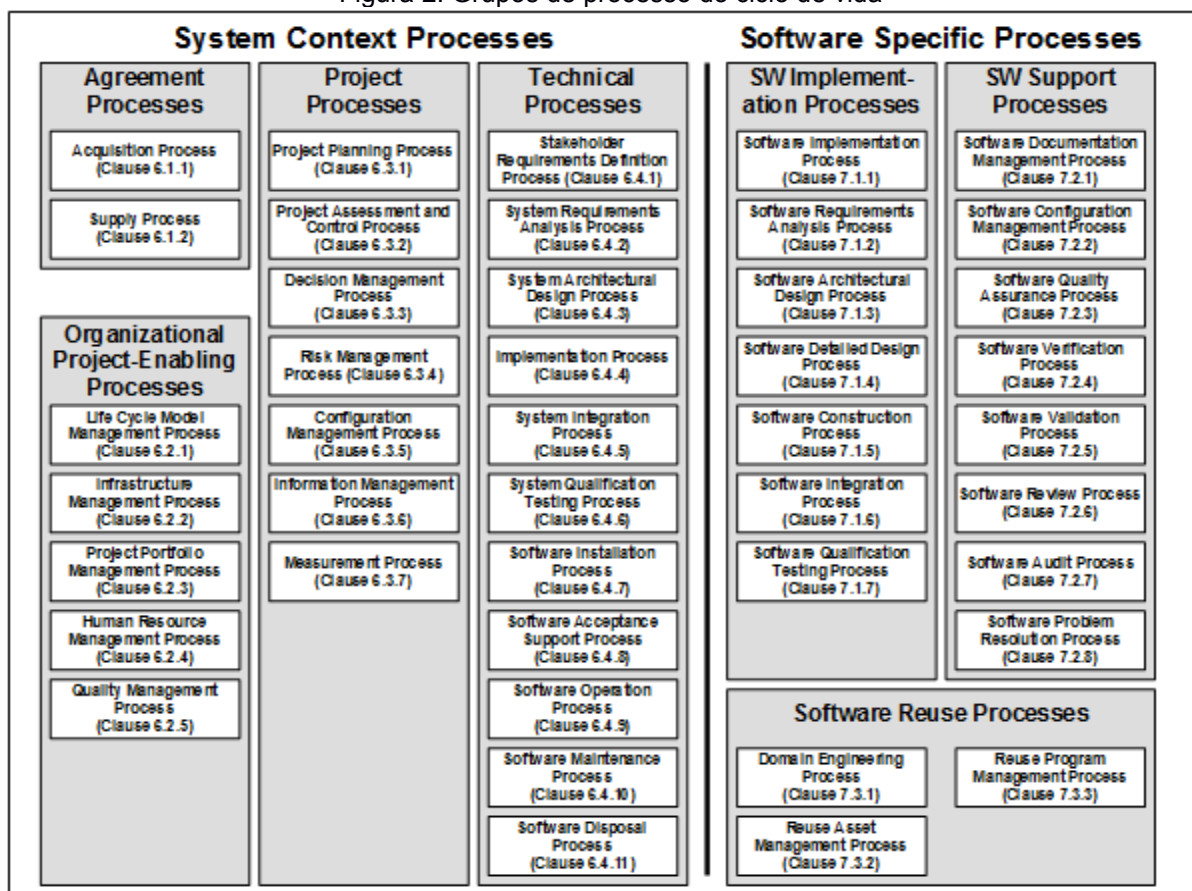
Fonte: IEEE CS (2013).

Uma área de conhecimento define um escopo e as informações de todos os tópicos que a envolvem; como exemplo, a área construção de software que aborda a implementação, verificação e testes de software (IEEE CS, 2013). Para construir um software com um determinado grau de qualidade, é preciso adotar um processo de software, geralmente baseado nestas áreas de conhecimento. Um processo de software é um conjunto de atividades realizadas para gerenciar, desenvolver e manter sistemas de software (ACUÑA & FERRÉ, 2001).

Segundo a ISO/IEC 12207 (ISO/IEC 12207, 2009), o processo de software envolve dois principais grupos de processos (Figura 2):

- *Processos de contexto de sistemas*: processos para lidar com um produto ou serviço de software de forma independente. Este grupo contém processos relativos a contratos, informações técnicas e projetos
- *Processos de sistemas específicos*: processos para o desenvolvimento de um produto ou serviço de software. Este grupo contém, por exemplo, processos para levantamentos dos requisitos, implementação, teste, suporte e o reuso de softwares.

Figura 2: Grupos de processo do ciclo de vida



Fonte: ISO/IEC 12207 (2009)

Embora existam vários processos para o desenvolvimento de software, seus principais processos abordam diferentes áreas de conhecimento (requisito de software, modelagem de software, construção de software, teste de software e manutenção de software). De acordo com as características do contexto, objetivos de negócios e do domínio da aplicação, o processo de software adotado para o desenvolvimento é customizado para atingir o seu objetivo (ISO/IEC 12207, 2009).

2.2. PROCESSO DE DESENVOLVIMENTO DE APPS

Assim como no desenvolvimento de softwares tradicionais, o desenvolvimento de aplicativos (*apps*) também requer a adoção de um processo sistemático de desenvolvimento (RAHIMIAN & RAMSIN, 2008). Porém, este processo é diferenciado devido a características, restrições e desafios impostos pelos dispositivos móveis que impactam nas necessidades referentes a um processo de desenvolvimento (ROSEN & SHIHAB, 2016). Algumas das peculiaridades referentes aos dispositivos móveis são apresentadas no quadro 2.

Quadro 2 Características de um dispositivo móvel.

Características dos dispositivos móveis	Descrição
1. Sensores	Um dispositivo móvel possui diversos sensores integrados a ele, como localização do dispositivo, câmera, leitor de código de barra, de proximidade (do celular com o corpo humano), acelerômetro (detecção de movimento do celular), etc. (ROSEN & SHIHAB, 2016).
2. Dimensão e Resolução de Tela	Há uma gama diversa de dimensões de tela entre as diversas plataformas de hardware de um dispositivo móvel. Geralmente o espaço disponível de celulares são menores que dos computadores <i>desktops</i> , dificultando ao organizar a interface para expor as informações e formulários na aplicação (MUCCINI, DI FRANCESCO & ESPOSITO, 2012). Além disso, existem diversas resoluções de telas que podem ser rotacionadas. No caso dos celulares, seu espaço pode ser reduzido por causa do teclado virtual (FLORA, CHANDE & WANG, 2014).
3. Conectividade a redes	Geralmente a conexão de rede dos dispositivos móveis é mais instável que em <i>desktops</i> . Assim, é comum a internet ficar desconectada devido a falhas nas redes. Caso o <i>app</i> precise utilizar algum recurso remoto, é preciso pensar em como tratar esses casos, por exemplo, verificar se a conexão está ativa e funcional. (RAHIMIAN & RAMSIN, 2008).
4. Recursos limitados	Recursos como espaço de armazenamento, processamento gráfico, poder de processamento, capacidade de memória, reserva limitada de energia são mais restritos em relação a dispositivos do que em computadores (YANG-JAE, JI-HYEON, GYU-SANG, 2008). Além disso, o dispositivo móvel contém apenas o teclado móvel, o que acaba sendo mais adequado à entrada de números em vez da entrada de texto, pois os botões pequenos limitam a eficiência do usuário na entrada de dados. (FLORA, CHANDE & WANG, 2014)
5. Multiplataformas	Atualmente os <i>apps</i> abrangem várias plataformas de sistemas operacionais (IOS, <i>Android</i> , Windows 7), diferentes fabricantes de hardware (<i>Apple</i> , Samsung, Google) e plataformas de computação (smartphone, tablet) (DEHLINGER & DIXON, 2011). Cada uma destas opções deve ser considerada durante o desenvolvimento de aplicativos móveis, pois eles têm uma influência direta sobre os requisitos de software (DEHLINGER & DIXON, 2011).

Além das restrições vinculadas ao dispositivo móvel, há outras questões que surgem em relação aos aplicativos e que impactam no seu processo de desenvolvimento, como por exemplo, as características inerentes ao contexto de uso (Quadro 3) e os projetos que desenvolvem *apps* (Quadro 4).

Quadro 3: Contexto de uso

	Características de <i>App</i>	Impacto no processo de desenvolvimento
--	-------------------------------	--

Intolerância a erros e falhas	Os usuários de <i>apps</i> geralmente demonstram intolerância a erros e falhas. Caso um aplicativo venha a falhar algumas vezes, o usuário mudará para outro <i>app</i> disponível (FLORA, CHANDE & WANG, 2014).	Foco na facilidade de uso e prevenção de erros de interação (DEHLINGER, DIXON, 2011). Isto aponta a importância do processo de usabilidade/UX.
Ambiente de uso	O ambiente de uso de um aplicativo móvel pode variar muito, e muitas vezes o usuário está andando ou dirigindo um carro, ou distraído com outra atividade (KUUSINEN & MIKKONEN, 2014).	Aumenta a importância de uma análise do contexto tipicamente feita no processo de engenharia de usabilidade/UX.
Recursos	Os <i>apps</i> contêm recursos de gestos, sensores e dados de localização, que são importantes e causam impactos na concepção de um <i>app</i> (HARLEEN, XIAOFENG & SWATI, 2014).	No desenvolvimento de <i>apps</i> se necessita desta consideração de software e <i>hardware</i> , indicando a importância de engenharia de processos de sistemas de software abordando processos de contexto de sistemas (ISO/IEC 12207, 2009).
Público alvo de uso de Apps	Além de existirem grupos de pessoas que têm maior afinidade com softwares desktops e/ou pouca familiaridade com <i>apps</i> (ARHIPAINEN & TÄHTI, 2003), abriu-se muito o público alvo de <i>apps</i> popularizando o uso de software de forma geral (KEMP, 2015).	Neste contexto, é essencial a análise das características e limitações do público alvo para assegurar o desenvolvimento de um <i>app</i> usável (ARHIPAINEN & TÄHTI, 2003). Esta atividade geralmente é parte da engenharia de usabilidade/UX.
Tipos de Apps	Existem aplicativos para diversos fins, como usos comerciais, domésticos, escritórios, entretenimento, etc. Cada um pode conter uma peculiaridade específica no seu contexto de uso, por exemplo, sistemas comerciais precisam de bom desempenho por conta da grande quantidade de transações processadas em pouco tempo (VARSHNEY & VETTER, 2001)	É necessário analisar os requisitos relacionados ao tipo específico do aplicativo (DEHLINGER, DIXON, 2011).
Utilidade	Um <i>app</i> de sucesso caracteriza-se por ser útil para resolver algum problema ou atender uma necessidade existente (FLORA, CHANDE & WANG, 2014).	Visando a assegurar a utilidade de uma <i>app</i> , métodos como o <i>Design Thinking</i> integrado ao processo de desenvolvimento de <i>apps</i> podem ser uma solução para desenvolvê-los com real valor.

Quadro 4: Projeto para desenvolvimento de *apps*

	Características de App móveis	Impacto no processo de desenvolvimento
--	--------------------------------------	---

Requisitos voláteis	Os requisitos pertinentes aos <i>apps</i> são altamente voláteis, causados por rápida mudança dos requisitos de negócios, além de mudanças de características técnicas relativas ao próprio dispositivo móvel (FLORA, CHANDE & WANG, 2014).	A natureza dinâmica do conteúdo de um <i>app</i> exige um processo flexível para se adaptar rapidamente às mudanças, tendências e tecnologias, como por exemplo, abordagens ágeis. (KALEEL & HARISHANKAR, 2013).
Equipe de desenvolvimento	Geralmente os <i>apps</i> são desenvolvidas por equipes de pequeno e médio porte (SPATARU, 2010).	Adequando-se a esta característica, surge a adoção de processos/métodos de desenvolvimento para pequenas equipes, como por exemplo, abordagens ágeis.
Funcionalidades curtas e rápidas	As funcionalidades do <i>app</i> tipicamente são rápidas, simples, focadas, para serem realizadas com curto período de tempo e com poucos toques digitais (KUUSINEN & MIKKONEN, 2014).	Como a funcionalidades de <i>apps</i> móveis são mais simples e pequenas do que sistemas de software para computadores, o processo de desenvolvimento pode ser também simplificado em termos de quantidade de artefatos, sendo criados com abordagens ágeis (ABRAHAMSSON, 2005).
Complexidade de Teste	Os testes de um <i>app</i> tornam-se complexo porque contém diversas plataformas de <i>hardware</i> , rede e sistemas operacionais para testar. Também fica embutida a dificuldade relacionada à necessidade de testes no próprio aparelho, com possibilidades de gravação de <i>log files</i> , gravação de telas e do rosto do usuário, no caso de testes de usuário no campo (MUCCINI, DI FRANCESCO & ESPOSITO, 2012).	Esta característica demonstra a importância dos testes e a sua complexidade no desenvolvimento de <i>apps</i> , com resultados que podem ser atingidos adotando-se abordagens ágeis, que enfatizam a parte dos testes aplicando ciclos repetitivos de testes e práticas de garantia da qualidade (FLORA, CHANDE & WANG, 2014).

Dessa forma, é necessário adaptar o processo de desenvolvimento de software tradicional para cobrir todas estas características (STAPIĆ, MIJAČ & STRAHONJA, 2016). Neste contexto, são propostas diversas metodologias de desenvolvimento de *apps*, com a finalidade de suprir essas características e restrições. Essas metodologias abordam questões de desenvolvimento de software com metodologias ágeis (YANG-JAE, JI-HYEON & GYU-SANG, 2008), experiência do usuário (KUUSINEN & MIKKONEN, 2014) e *Design Thinking* (BURNETT, 2009).

2.3. METODOLOGIA ÁGIL PARA DISPOSITIVOS MÓVEIS

A metodologia de desenvolvimento ágil segue uma abordagem interativa e incremental, na qual o software é desenvolvido com múltiplas entregas e funcionalidades crescentes (KENT, 2001). Os conceitos-chave da Metodologia Ágil são (HIGHSMITH & COCKBURN, 2001):

1) *Respostas rápidas a mudanças ao invés de seguir um planejamento*: valoriza mais satisfazer o cliente com entregas rápidas e contínuas, respondendo a mudanças mais que seguir um plano;

2) *Indivíduos e interações ao invés de processos e ferramentas*: o foco maior é no indivíduo, que tem características únicas como conhecimento e habilidades, do que em processos de softwares ou ferramentas;

3) *Software em funcionamento ao invés de documentação abrangente*: entregar valor de negócio aos usuários com as entregas de softwares em funcionamento, com frequência e períodos curtos, é mais importante do que documentação;

4) *Colaboração com o cliente ao invés de negociação de contratos*: pessoas relacionadas a negócios e desenvolvedores devem trabalhar juntos para agilizar a comunicação, e poder tomar decisões em conjunto.

Assim, o desenvolvimento do software na Metodologia Ágil possui diversas características (Quadro 5).

Quadro 5: Características da Metodologia Ágil.

Características da Metodologia Ágil	Tipo de característica	Descrição
1. Iterativa	Prática	O desenvolvimento das funcionalidades do software é feito por progressos sucessivos e refinado em ciclos de interação. Assim, a cada ciclo, o software é melhorado e adicionam-se mais detalhes (ABRAHAMSSON <i>et al.</i> , 2017).
2. Incremental	Prática	O software é desenvolvido em pedaços, isto é, em um subconjunto de funcionalidades definidas e que foram priorizadas conforme a necessidade do usuário (GREER & RUHE, 2004). A cada ciclo de desenvolvimento as funcionalidades dos softwares a serem desenvolvidos podem ser repriorizadas conforme as necessidades do cliente/usuário (HIGHSMITH & COCKBURN, 2001). A adoção em conjunto do desenvolvimento iterativo e do desenvolvimento incremental reduzem o risco pertinente ao projeto (tempo, custo), pois o escopo do desenvolvimento é menor e claro (HIGHSMITH & COCKBURN, 2001).
3. Ciclos curtos de	Prática	A abordagem ágil recomenda curtas interações de

desenvolvimento		desenvolvimento, com duração de 2 a 6 semanas (HIGHSMITH & COCKBURN, 2001). Este cenário é propício para o desenvolvimento de <i>apps</i> , pois os seus requisitos estão em constante mudança, e contém funcionalidades simples e rápidas (FLORA, CHANDE & WANG, 2014).
4. Programação em pares	Prática	O código produzido no projeto é desenvolvido por duas pessoas juntas no mesmo computador, revezando o teclado. Um dos desenvolvedores digita o código enquanto o outro revisa o que está sendo codificado, relatando eventual problema e pensando em uma solução (KENT, 2000). Esta prática visa compartilhar o conhecimento do código, experiência de programação e melhorar a qualidade do software com as correções de falhas na codificação que o observador aponta (KENT, 2000).
5. Histórias de Usuários	Artefato	Tem como objetivo definir e priorizar os requisitos do usuário. As histórias de usuários descrevem de forma simples, curta e clara uma funcionalidade do sistema, segundo o seu ponto de vista (KENT, 2000).
6. Desenvolvimento orientado a teste	Paradigma de desenvolvimento	Os testes são primeiramente codificados para depois implementar as funcionalidades do sistema (KENT, 2000). Esta prática torna o código simples e claro, pois a funcionalidade é desenvolvida apenas o suficiente para passar no teste. Além disso, torna o código seguro, pois o desenvolvimento é orientado a teste (KENT, 2000).

Existem atualmente diversos métodos que seguem as práticas ágeis, como: *Extreme Programming (XP)*, *Crystal*, *Kanban*, *Feature Driven Development (FDD)*, entre outros (ABRAHAMSSON *et al.*, 2017).

Voltado ao desenvolvimento de *apps*, diversos estudos concluem que a metodologia ágil é adequada para o desenvolvimento de *app* (ABRAHAMSSON, 2005; STAPIĆ, MIJAČ & STRAHONJA, 2016) conforme identificado no quadro 4. Dessa forma, foram customizadas para esse contexto diversas especializações de processo de desenvolvimento de software baseadas nas características, práticas e valores das metodologias ágeis, como o MASSAM (JEONG, LEE & SHIN, 2008), o modelo SLeSS (*Scrum and Lean Six Sigma*) (DA CUNHA, DANTAS & ANDRADE, 2011), a metodologia HME (*Hybrid Method Engineering*) (RAHIMIAN & RAMSIN, 2008). Dentre os mais abrangentes está o Mobile-D que é o pioneiro e também aceito ao ser avaliado em relação à certificação CMMI nível 2 (ABRAHAMSSON *et al.*, 2004). Este processo é baseado no Extreme Programming XP em relação às práticas e metodologias Crystal e é composto por cinco fases (VTT ELECTRONICS, 2006):

1. Explorar: a equipe de desenvolvimento deve gerar um plano e estabelecer. O objetivo desta fase é ter uma noção do produto e planejar o processo de seu desenvolvimento
2. Inicializar: é feita uma preparação e verificação de todos os problemas críticos de desenvolvimento, de modo que todos estejam prontos no final da fase para implementar os requisitos selecionados pelo cliente.
3. Produção: nesta fase as funcionalidades são implementadas adotando o processo de desenvolvimento iterativo e incremental.
4. Estabilizar: esta fase visa assegurar a qualidade do projeto com o cliente participando ativamente das atividades do projeto para garantir que os requisitos forneçam valor de negócios e estão sendo entendidos corretamente. Além disso é feita a documentação dos requisitos desenvolvidos.
5. Teste de sistema: o objetivo desta fase é verificar se a funcionalidade definida pelo cliente foi implementada corretamente. O cliente fornece feedback para equipe do projeto para cada defeito encontrado.

Este processo envolve práticas ágeis, como por exemplo, integração contínua, programação em pares, desenvolvimento orientado a testes e processos ágeis. Recomenda-se a utilização do processo Mobile-D para equipes pequenas, no máximo 10 pessoas, que se concentram no desenvolvimento de *apps*. O Mobile-D já foi aplicado em projetos de desenvolvimento, e algumas vantagens foram observadas, como aumento da visibilidade do progresso, identificação precoce e resolução de problemas técnicos, responsabilidade compartilhada, baixa número de de defeitos no produto final e um ritmo constante no processo de desenvolvimento (ABRAHAMSSON *et al.*, 2004).

Como visto, abordagens ágeis são amplamente utilizadas no que se refere ao processo de desenvolvimento de *app* pelo fato de poder adaptar as suas características. Estas são supridas principalmente pelo fato da abordagem ter a estratégia de desenvolvimento iterativo e incremental, tornando-se um processo flexível para se adaptar rapidamente às mudanças, tendências e tecnologias (FLORA, CHANDE & WANG, 2014). Além disso, a abordagem é voltada para sistemas com funcionalidades simples e menores, conforme identificado no quadro 4.

2.4. INTERAÇÃO HUMANO-COMPUTADOR

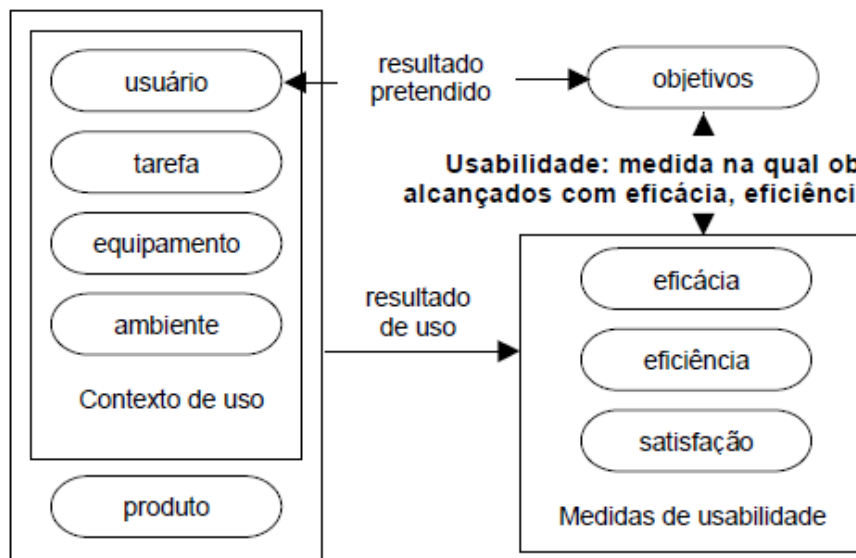
A interação entre o usuário e um dispositivo móvel pode ocorrer por meio de elementos visuais de tela, teclado digital, comando de voz, gestos e sensores que capturam informações sobre o contexto e as atividades que estiverem em volta do usuário (COOPER et. Al, 2014). Os usuários buscam em um *app* uma interface de tela que seja simples, intuitiva, atrativa e de fácil adaptação, conforme o seu contexto de uso (KUUSINEN & MIKKONEN, 2014). Assim, o desenvolvimento de *apps* deve abordar assuntos relacionadas a interações entre o usuário interfaces computacionais. Estas questões estão relacionadas à disciplina Interação Humano-Computador (IHC), uma área de conhecimento relacionada ao projeto, avaliação e à implementação de sistemas computacionais interativos para o uso humano, e aos estudos dos fenômenos que os cercam (HEWETT et al., 1992).

No contexto de desenvolvimento de aplicativos, grande parte da pesquisa neste campo busca melhorar diversos critérios de sucesso de um *app* aliando a utilização dos recursos que conduz à interação usuário e dispositivo móvel. Dentre esses critérios está a questão da utilidade do aplicativo que busca a melhor solução, focando nas necessidades do usuário (VALENTIM, SILVA & CONTE, 2017). Uma das abordagens para suprir esta necessidade é o *Design Thinking* que fornece uma metodologia para suscitar as necessidades dos usuários, produzindo protótipos rápidos e simples, que geralmente resultam em soluções criativas e inovadoras (BURNETT, 2009).

Outro aspecto a ser considerado para que um *app* tenha mais chance de sucesso é ser agradável na sua utilização, fornecendo ao usuário a satisfação de uso (YIN et al., 2014). Assim, a experiência de usuário torna-se um fator importante no desenvolvimento de um *app*. O termo Experiência de Usuário (UX) trata das respostas e percepções de uma pessoa resultantes do uso e/ou antecipação de uso de um produto, sistema ou serviço (ISO/IEC 9241-210). A sua área de atuação é denominada *UX Design*, que define a criação e sincronização de elementos que afetam a experiência do usuário, de modo a influenciar suas percepções e comportamento ao realizar a tarefa de um produto (UNGER & CHANDLER, 2010). Essas tarefas podem envolver emoções, crenças, preferências, percepções, respostas físicas e psicológicas, comportamentos e realizações do usuário, que ocorrem antes, durante e após o uso (ISO/IEC 9241-210:2010).

Além disso, a disciplina IHC tem como objetivo assegurar que a interação entre usuário e o software apresentem boa usabilidade (KIM, 2015). A usabilidade é a medida na qual um produto pode ser usado por usuários específicos, para alcançar objetivos específicos, com eficácia, eficiência e satisfação, em um contexto específico de uso (ISO/IEC 9241:210, 2010). A Figura 3 ilustra a estrutura do conceito de usabilidade apresentando seus componentes e relacionamentos.

Figura 3: Estrutura do conceito de usabilidade.



Fonte: adaptada de ISO/IEC 9241-210:2010 (2010) – versão traduzida.

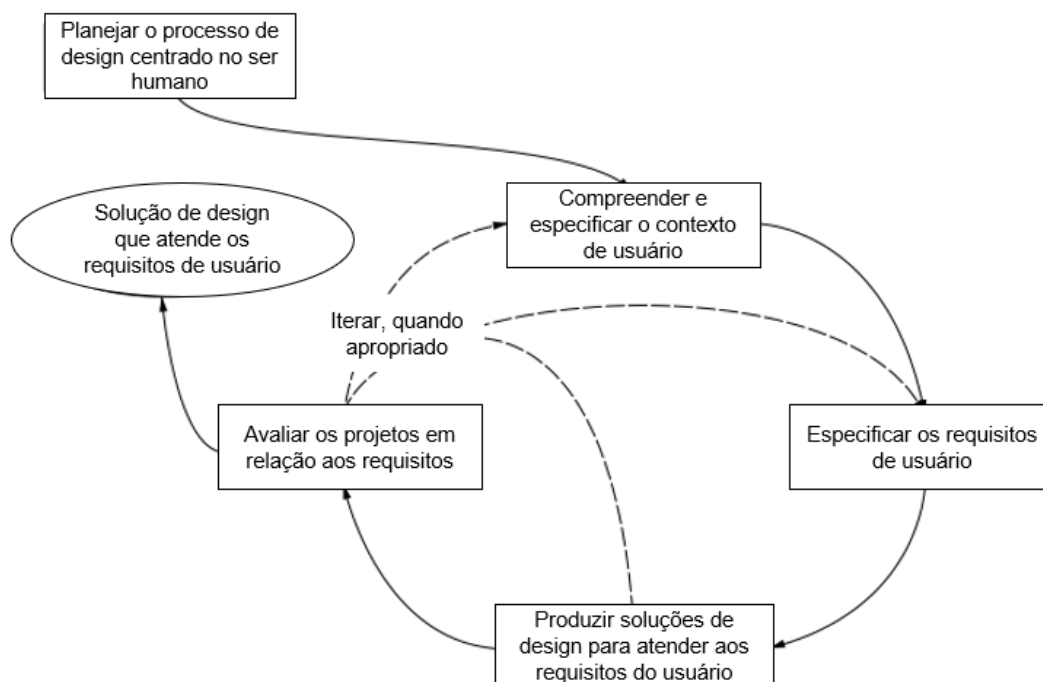
Os objetivos da interação são os resultados pretendidos e que devem ser alcançados com: i) eficácia: acurácia e completude com as quais usuários alcançam objetivos específicos; ii), eficiência: recursos gastos em relação à acurácia e abrangência, com as quais usuários atingem objetivos; e, iii) satisfação: ausência do desconforto e presença de atitudes positivas para com o uso de um produto (ISO/IEC 9241-210, 2010). A usabilidade é dependente do contexto de uso, que consiste em usuários, tarefas, equipamentos (tais como hardware), software e materiais, e o ambiente físico e social no qual um produto é usado. Estes critérios de sucesso na interação são assegurados pela adoção de processos sistemáticos. Voltando-se para assegurar a usabilidade, são adotados processos de EU.

2.5. PROCESSOS VOLTADOS À ENGENHARIA DE USABILIDADE

A usabilidade de um software é assegurada seguindo-se um processo sistemático de engenharia de usabilidade, que por sua vez, é definida como um processo que fornece métodos estruturados para que se possa atingir um alto grau de usabilidade da interação com o usuário durante o desenvolvimento do produto (MAYHEW, 1999). Existem diversos modelos de EU, entre os quais o definido pela norma ISO/IEC 9241:210 (2010), que propõe um processo de design centrado no usuário para sistemas interativos (Figura 4).

Este processo tem como princípio focar na interação do usuário e nas tarefas realizadas em um determinado ambiente utilizando um dispositivo. Além disso, o processo mede a utilização do produto observando a interação do usuário com ele. Este é um processo iterativo, no qual o design pode ser modificado a cada ciclo do processo (ISO/IEC 9241:210, 2010).

Figura 4: Atividades de design centradas no ser humano.



Fonte: adaptada da ISO/IEC 9241:210 (2010) – versão traduzida.

O processo define quatro principais atividades no processo de desenvolvimento de software centrado no usuário (ISO/IEC 9241:210, 2010):

1) *Analisar e especificar o contexto de uso*: O objetivo é coletar as informações sobre as características dos usuários, o contexto de uso e as tarefas que serão executadas com o produto. A saída desta fase deve ser um documento descrevendo

as características relevantes que o sistema deve ter. A descrição das características do usuário pode ser definida por meio *personas*. *Persona* consiste na criação de perfis e personificação de grupo de usuários, ou seja, representa uma caracterização de um personagem que, embora seja fictício, expõe as características importantes da população de usuários para a qual se destina o produto o projeto (ADLIN, 2006);

2) *Especificar os requisitos de usuário*: Os requisitos do usuário são coletados e a partir deles são definidos os critérios de sucesso em relação à usabilidade. Os requisitos são especificação contendo as necessidades do usuário em relação ao contexto de uso. Além disso, deve conter objetivos de usabilidade de forma mensurável e critérios de satisfação no contexto de uso;

3) *Produzir soluções de projeto de interface*: O design centrado no usuário visa alcançar uma boa experiência de usuário ao longo do processo de design. Assim, nesta fase as soluções de design são produzidas baseado na descrição do contexto do usuário, no domínio da aplicação, nos padrões de design e usabilidade. Para produzir estas soluções, deve-se entender as tarefas do usuário e a interação do usuário com a interface do software. As concretizações destas soluções são geralmente feitas por meio de protótipos. O foco desta prática é desenvolver um protótipo na fase inicial do desenvolvimento, podendo assim avaliar a usabilidade por meio de inspeções ou testes de usuários.

4) *Avaliar o projeto em relação aos requisitos do usuário*: A usabilidade do projeto deve ser avaliada em relação às tarefas dos usuários, tendo como objetivo coletar novas informações sobre suas necessidades e fornecer *feedback* sobre os pontos fortes e fracos da solução de design do ponto de vista do usuário, para avaliar se os requisitos foram alcançados.

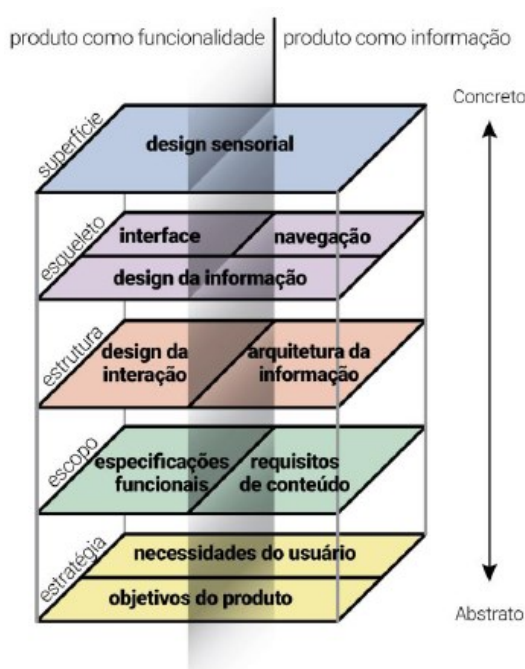
Este processo é uma abordagem de projeto interativa, onde o design é evoluído/melhorado a cada interação, pois é difícil compreender todas as necessidades do usuário. O ciclo dessas atividades termina quando a “avaliação do projeto em relação aos requisitos do usuário” é executada com um resultado satisfatório.

2.6. DESIGN DE EXPERIÊNCIA DO USUÁRIO PARA DISPOSITIVOS MÓVEIS

O desenvolvimento de *apps* de qualidade envolvem aspectos relacionados à experiência de usuários (UX), como a sensação, a emoção e a satisfação nas

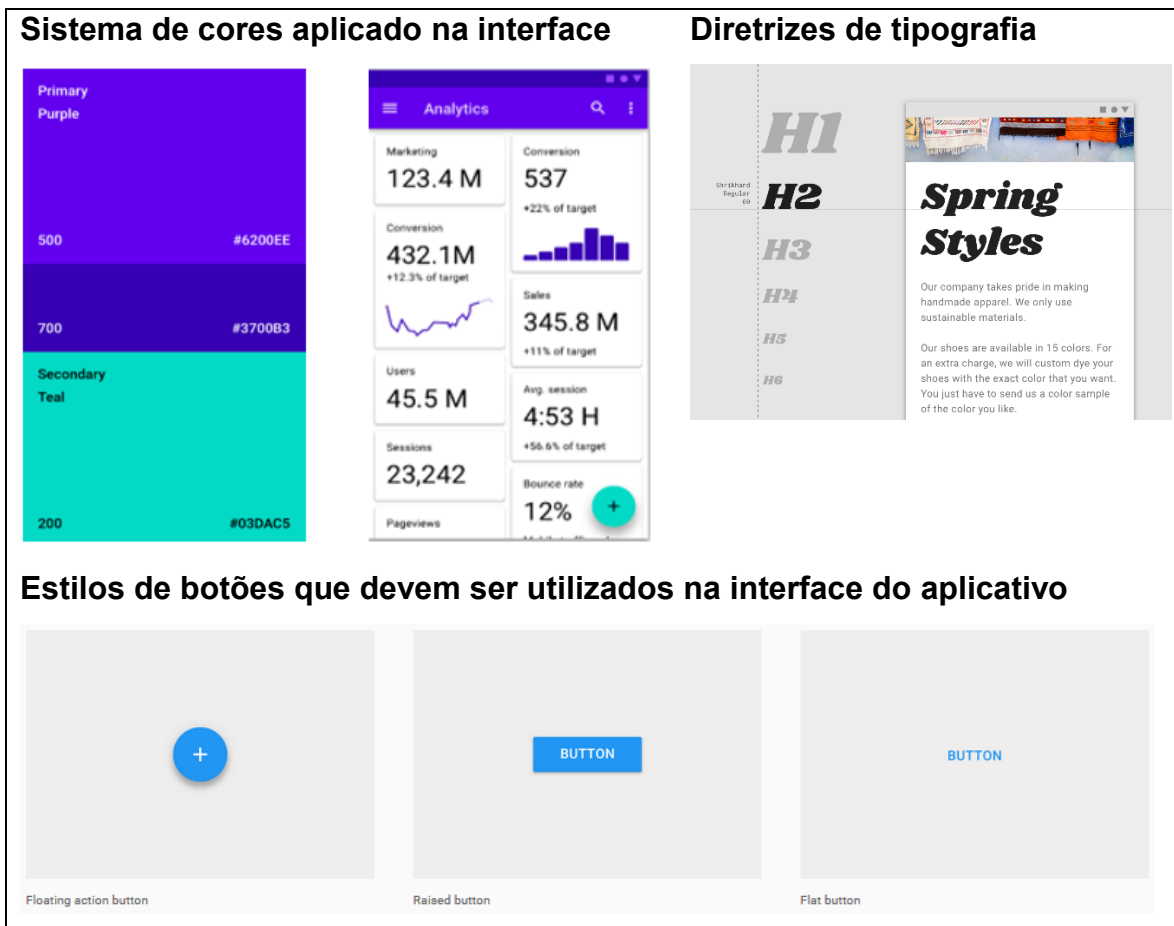
diversas etapas de uso (antes, durante e depois) (ISMAIL *et al.*, 2016). O objetivo do UX é melhorar a satisfação e a fidelização do cliente por meio da utilidade, facilidade de uso e prazer, proporcionados na interação com um produto (ISMAIL *et al.*, 2016). Para alcançar uma boa experiência é preciso se preocupar com diversos elementos de design de interface de usuário. Neste sentido, Garret (2011) desenvolveu um *framework* relativo ao design dos elementos da experiência do usuário. A Figura 5 apresenta estes elementos, indicando uma precedência sobre as atividades do processo (GARRET, 2011).

Figura 5: Níveis do processo de design centrado no usuário.



Fonte: GARRET (2011) - versão traduzida.

O design sensorial são os primeiros aspectos percebidos pelos usuários, como tipografia, imagens, elementos esquemáticos, cor e multimídia (FEIJÓ, BALDESSAR & VIEIRA, 2013). Existem diversos guias de estilos com o propósito de auxiliar na aplicação de elementos de design visual, dentre eles, o *Material Design* (GOOGLE, 2019), voltado para aplicativos *Android*. Este guia cria uma linguagem visual que sintetiza os princípios clássicos do bom design com a inovação e a possibilidade da tecnologia e da ciência (GOOGLE, 2019). Neste guia são definidos, por exemplo, um sistema de cores que permite a definição de uma paleta aplicável nos componentes, a definição da tipografia, dos estilos de botões, etc. (Figura 6).

Figura 6: Diretrizes de interface de usuário do *Material Design*.

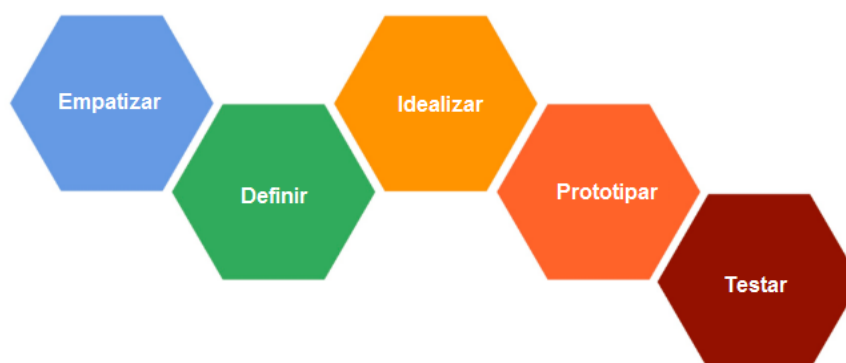
Fonte: elaborada pelo autor com base no *Material Design* (GOOGLE, 2019).

Utilizando este guia é possível melhorar a experiência de usuário, no entanto, não garante que o *app* será criativo, inovador e com utilidade para o usuário final. Para isso, geralmente adota-se abordagens de *Design Thinking* no processo de desenvolvimento (DE PAULA; MENEZES & ARAÚJO, 2014).

2.7. PROCESSO DE *DESIGN THINKING* PARA DISPOSITIVOS MÓVEIS

Um dos principais desafios ao criar *apps* de sucesso é ter ideias e soluções inovadoras, criativas e focadas no usuário final (DE PAULA; MENEZES & ARAÚJO, 2014). Um método para alcançar a criação de produtos úteis e com usabilidade é o *Design Thinking* (DT), que é um processo para identificar problemas e gerar soluções inovadoras, competitivas e úteis (CHEN & HUANG, 2017). Com a finalidade desenvolver um *app* de jogo utilizando *App Inventor*, Chen & Huang (2017) adotaram um processo de *Design Thinking* que inclui as etapas de empatizar, definir, idealizar, prototipar e testar (Figura 7).

Figura 7: Processo de Design Thinking.



Fonte: adaptada de D.SCHOOL (2019) - versão traduzida.

Empatizar é a etapa na qual tem um contato com as pessoas buscando se aprofundar no assunto e se colocar no lugar do usuário. Na etapa *Definir* são agrupados os conhecimentos adquiridos com a empatia, e é definido o problema analisando aquilo que pode ser um problema para os usuários. Após entender as necessidades e os problemas dos usuários é realizada a etapa de *Idealizar*, que resulta no desenvolvimento do produto. Nesta fase algumas técnicas podem ser utilizadas, como *brainstorming*, em que o grupo de desenvolvedores reúne as melhores ideias, estimulando o processo criativo. *Prototipar* é a etapa na qual a ideia se concretiza por meio de um protótipo. Por fim, na etapa *Testar* o protótipo é avaliado, visando a utilidade oferecida para o usuário. No Quadro 6 são apresentadas as atividades e entregas em cada fase do processo.

Quadro 6: O processo do *Design Thinking*.

Etapa	Atividades	Entregáveis
Empatizar	- Entrevistas com usuários; - Observações; - Imersão.	- Mapa de empatia; - Lista de <i>feedback</i> dos usuários; - Problemas identificados.
Definir	- <i>Workshops</i> ; - Reuniões com responsáveis.	- Design inicial; - Mapa de contexto; - Mapa de oportunidades.
Idealizar	- Atividades de idealização; - <i>Brainstorming</i> ; - Mapas mentais; - <i>Sketches</i> /desenhos.	- Ideias/conceitos; - <i>Sketches</i> ; - Mapa de priorização; - Mapa de afinidade; - Avaliação das ideias.
Prototipar	- Prototipação espacial; - Prototipação física; - Construção no papel;	- Protótipos físicos; - <i>Wireframes</i> ; - <i>Storyboards</i> .

	- Construção em nível <i>wireframe</i> ; - <i>Storyboards</i> ; - Atribuição de papéis.	
Testar	- Realização de testes.	- Lista de <i>feedback</i> dos usuários; - Observações; - Avaliação.

Este processo é interativo, no qual o produto é melhorado a cada interação, identificando novos problemas até atender a todas as necessidades do usuário (CHEN & HUANG, 2017).

2.8. ENSINO E APRENDIZAGEM POR MEIO DE DESIGN INSTRUCIONAL

O processo de ensino e aprendizagem é um complexo sistema de interação comportamental entre o aluno e o professor (KUBO & BOTOMÉ, 2001). A maneira de trabalhar sobre um tema e a diferença entre ensinar e aprender é essencial para o educador transmitir um conhecimento (SPRINTHALL & SPRINTHALL, 1993).

O ensino é considerado como um ato ou experiência transmitido de uma pessoa para outra, adquirindo novos conhecimentos (SPRINTHALL & SPRINTHALL, 1993). Por meio do ensino ocorre a aprendizagem, que é compreendida como qualquer experiência que vivemos, que afete nossa mente e que mude nosso comportamento em resultado desta prática, sendo esta mudança muitas vezes permanente (GONÇALVES, 1993). De modo geral, o ensino está relacionado à transmissão de conhecimentos e a aprendizagem à aquisição de conhecimentos (SPRINTHALL & SPRINTHALL, 1993).

O ensino geralmente ocorre por meio de Unidades Instrucionais (UIs), que por sua vez, são definidas como um conjunto de aulas projetadas para atingir objetivos de desempenho para um público-alvo (BRANCH, 2009). Elas podem ser aplicadas em aula, oficina, curso, e são compostas por atividades, exercícios, materiais para alunos e instrutores, possibilitando assim a aprendizagem de um conteúdo em um determinado contexto de aprendizagem (HILL, ROWAN & BALL, 2005). Para assegurar que os objetivos de aprendizagem de uma unidade instrucional sejam atingidos e que o aluno alcance as competências ensinadas, a unidade pode ser criada utilizando design instrucional.

Design Instrucional corresponde à, segundo FILATRO (2004):

“Ação intencional e sistemática de ensino que envolve o planejamento, o desenvolvimento e a utilização de métodos, técnicas,

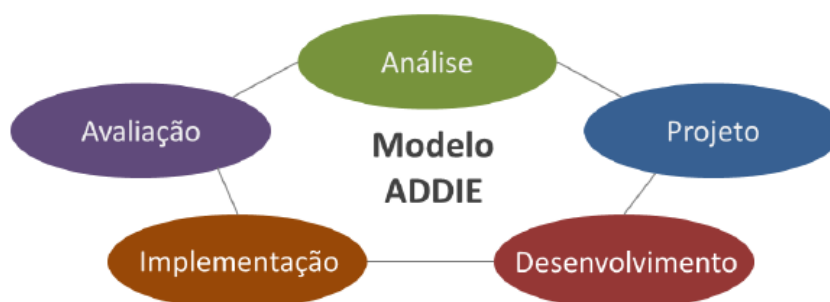
atividades, materiais, eventos e produtos educacionais em situações didáticas específicas, a fim de facilitar a aprendizagem humana a partir dos princípios de aprendizagem e instrução conhecidos”.

Além disso, estabelece a realização de uma sequência de passos para identificar necessidades de conhecimento, e assim, encontrar meios para supri-las (QUINN, 2005). Existem diversas abordagens para criar um design instrucional, dentre os principais está o modelo ADDIE.

2.8.1 MODELO ADDIE

O modelo ADDIE (*Analyze, Design, Develop, Implement e Evaluate*) (BRANCH, 2009) trata-se de um processo cíclico, composto por fases e várias atividades que são realizadas simultaneamente (Figura 8). Com este processo é possível construir sistematicamente uma UI, e sua escolha para aplicação no presente trabalho foi baseada por: i) ter uma abordagem consolidada com ampla utilização e aceitação; ii) definir um modelo genérico, aplicável a qualquer tipo de UI; e, iii) seguir um processo linear, cuja execução de suas fases é bem definida (BRANCH, 2009).

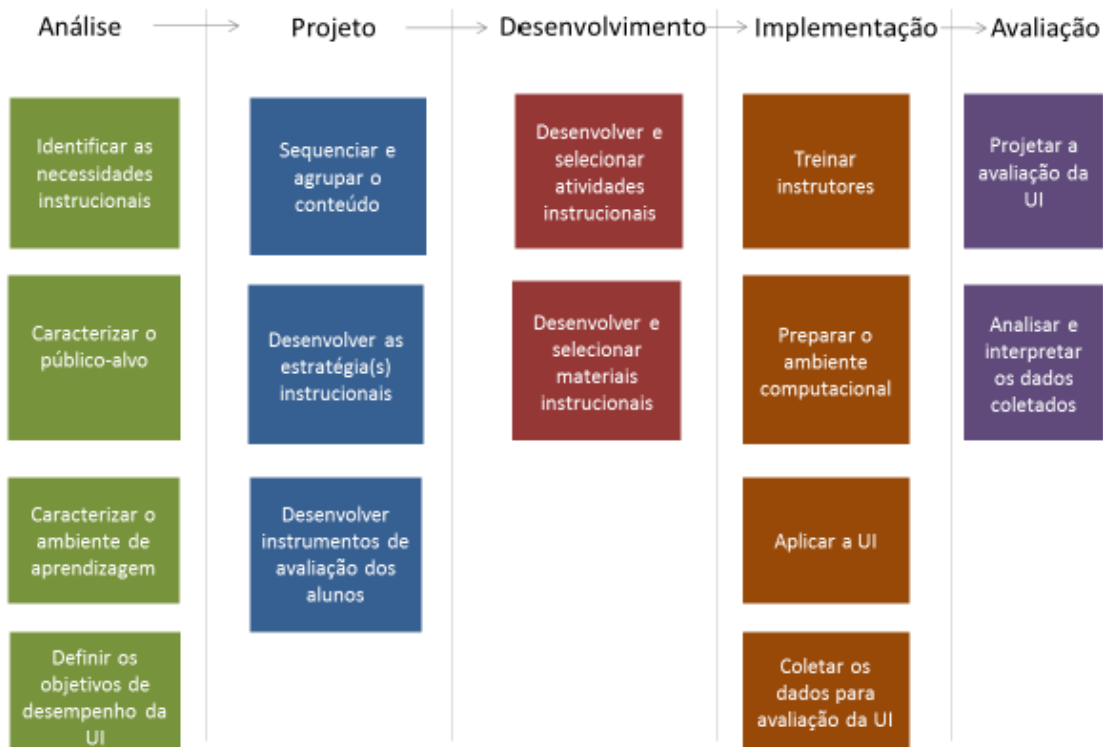
Figura 8: Diagrama com as cinco fases do modelo ADDIE



Fonte: BRANCH (2009).

Nos próximos itens são descritas cada uma das fases do modelo ADDIE e suas respectivas atividades (Figura 9).

Figura 9: Atividades do modelo ADDIE.



Fonte: BRANCH (2009).

Análise

A fase de análise envolve caracterizar o contexto em que será aplicada a UI, pois estas informações podem influenciar no seu desenvolvimento. A caracterização do contexto aborda as atividades de caracterizar o ambiente de aprendizagem e o público alvo.

A caracterização do ambiente de aprendizagem inclui analisar o tamanho das salas de aula, a capacidade de alunos, a disposição das mesas e cadeiras, recursos tecnológicos disponíveis (como projetor, computadores, internet cabeada e/ou wi-fi, ferramentas de software), horário da aula, a carga horária disponível para UI, e a periodicidade dos encontros. Além disso, é importante identificar a proporção de computadores por aluno e a configuração dos computadores, como tamanho das telas, a resolução utilizada, se possuem acesso à Internet, a largura de banda disponibilizada no laboratório, se os alunos podem levar seus próprios notebooks, etc. (AZIZ, RASLI & RAMLI, 2010).

Na caracterização do público-alvo é realizada uma análise das características demográficas, as preferências de aprendizagem e as pré-competências do público-

alvo. As características demográficas remetem-se em analisar o gênero dos alunos, a faixa etária, a instituição de ensino em que será aplicada a UI. Nas preferências de aprendizagem, busca-se entender se os alunos preferem realizar atividades em grupo ou individuais, estudar no laboratório ou em casa. Outras preferências vinculadas ao ensino e aprendizagem são as formas de avaliação do aluno, que podem variar entre testes discursivos e objetivos, trabalhos de pesquisa, trabalhos de implementação etc. No âmbito deste trabalho é preciso entender as preferências dos aspectos técnicos, já que as aulas exigem o uso de ferramenta de software. Tais preferências podem ser sobre o sistema operacional, o navegador de Internet, a resolução de tela. As pré-competências são a identificação se o público alvo já tem alguma experiência ou conhecimento prévio com o conteúdo da unidade instrucional (GERMANIA, PATRICIA, & EDMUNDO, 2011). Ainda nesta fase são analisadas as competências que se quer que os alunos desenvolvam ao terminarem a unidade instrucional (metas) e as competências atuais dos alunos.

O objetivo de desempenho é uma descrição específica do que os alunos serão capazes de realizar quando completarem a unidade instrucional (DICK & CAREY, 2006). Ele é definido a partir das necessidades de aprendizagem, da caracterização do público-alvo e do contexto de aprendizagem. Geralmente os objetivos de desempenho devem conter um resultado do aluno que demonstre se ele aprendeu a competência desejada, e especificar o que o aluno precisará realizar antes de atingir o objetivo e o critério, que é o nível de precisão atribuído ao desempenho.

Geralmente é utilizada a taxonomia de Bloom para desenvolver os objetivos de desempenho, um *framework* estruturado que define uma classificação de diferentes graus de aprendizagem definido para os alunos (BLOOM, 1956). É composto por 3 domínios: cognitivo (conhecimentos), psicomotor (habilidades) e afetivo (atitudes). Cada um dos domínios contém níveis de aprendizagem no qual o nível mais alto possui o conhecimento dos níveis mais baixos:

• **Cognitivo:** Está relacionado ao evento de conhecer, compreender e aplicar um novo assunto. É dividida em 6 níveis (Quadro 7).

Quadro 7: Níveis de aprendizagem do domínio cognitivo da taxonomia de Bloom.

Categorias	Descrição
------------	-----------

1. Conhecimento	É o uso da memória para recuperar, reconhecer ou reproduzir uma informação que foi ensinada.
2. Compreensão	Habilidade de compreender e modificar o conteúdo mantendo o seu significado. Pode ser realizado por meio de tradução e interpretação de fatos e ideias.
3. Aplicação	O aluno consegue aplicar as informações, métodos e conteúdos aprendidos em novas situações concretas.
4. Análise	Capacidade do aluno examinar e dividir informações em partes, identificando causas. Fazer inferência e encontrar provas
5. Síntese	Envolve em combinar partes de informações para criar uma nova informação.
6. Avaliação	É desenvolver a capacidade de fazer julgamentos do conteúdo baseado em critérios ou padrões.

Fonte: BLOOM (1956)

• **Afetivo:** Este domínio lida com a expressão de sentimentos e a aceitação de atitudes, opiniões ou valores. Ela envolve a maneira como as pessoas reagem emocionalmente, e é dividida em 5 níveis (Quadro 8).

Quadro 8: Níveis de aprendizagem do domínio afetivo da taxonomia de Bloom.

Níveis	Descrição
1. Receber	Refere-se a sensibilidade dos alunos para receber novos estímulos e a motivação da sua atenção a ele.
2. Responder	Momento em que o aluno age de forma ativa, reagindo aos estímulos recebidos expressando sua vontade de responder (motivação).
3. Valorizar	Quando o aluno atribui valor para objetivos, fenômenos ou informações.
4. Organizar	O aluno pode organizar diferentes valores, informações e ideias em sua própria hierarquia, criando um sistema único e pessoal.
5. Caracterizar	Os valores recebidos pelo aluno são influenciados em seu comportamento, tornando uma característica pessoal.

Fonte: BLOOM (1956).

• **Psicomotor:** domínio relacionado à forma como os alunos integram as atividades mentais e físicas para manipular ferramentas ou objetos (Quadro 9). Os níveis criados neste domínio são classificados conforme SIMPSON (1972).

Quadro 9: Níveis de aprendizagem do domínio psicomotor de Simpson

Níveis	Descrição
1. Percepção	Ter consciência de objetos ou qualidades, através dos estímulos sensoriais, para realizar as atividades.
2. Prontidão	O aluno está atento para agir nas atividades de ensino.
3. Resposta Guiada	É o desempenho de um ato sob orientação de um instrutor, envolvendo imitação de um dado ato demonstrado.
4. Mecanismo	Respostas aprendidas se tornam habituais, realizando movimentos com alguma confiança e proficiência.
Resposta Complexa	As habilidades motoras que requerem movimentos complexos são feitas de forma calma e precisa.
6. Adaptação	Habilidades no qual o indivíduo muda uma resposta motora quando ocorrem problemas inesperados, por exemplo, responder eficazmente às experiências inesperadas.

7. Originalidade	Utilização das habilidades psicomotoras para realizar um ato altamente complexo e que envolve a criação de novos padrões de movimento.
-------------------------	--

Fonte: SIMPSON (1972).

Projeto

Na etapa de projeto devem ser especificadas estratégias instrucionais e os instrumentos de avaliação. A estratégia instrucional define como as informações serão estruturadas e sequenciadas, e como serão apresentadas as atividades instrucionais (palestras, aulas, cursos, uso de ferramentas) e utilizados os materiais instrucionais (mídias, materiais de consulta, livros, apostilas) para alcançar os objetivos de desempenho da unidade instrucional (MAZZIONI, 2009). O desenvolvimento da estratégia instrucional deve abordar 4 elementos (DICK & CAREY, 2006):

- *Sequenciamento e agrupamento de conteúdo*: sequencia o conteúdo em uma estrutura lógica, e os agrupa conforme as atividades a serem ensinadas;

- *Componentes de aprendizagem*: define eventos de instrução que ajudarão os alunos a atingirem as competências conforme desejado pela instrução. Esses eventos podem ser atividades de ensino externo, para prender a atenção do aluno, estimular a memórias e reter os conteúdos, e também atividades de ensino interno, que foca na transmissão do conteúdo da UI;

- *Agrupamento dos alunos*: define a necessidade de agrupamento dos alunos para que possam se interagir durante o ensino. Este agrupamento será definido conforme preciso para atingir os objetivos de desempenho;

- *Seleção do método instrucional*: é feita a escolha de um método instrucional, que é uma abordagem utilizada para criar uma experiência de ensino que torna a aprendizagem eficiente, eficaz e atraente (MERRILL, 1996). Os cinco tipos de métodos instrucionais e exemplos de práticas que podem ser utilizadas estão descritos no Quadro 10.

Quadro 10: Métodos Instrucionais.

Método	Descrição
1. Instrução Direta	O professor é o centro da unidade instrucional e o responsável por transmitir o conhecimento para os alunos. É o método mais tradicional e consiste em aulas expositivas, por exemplo: palestras, prática e exercício, e demonstração.
2. Instrução Indireta	O aluno é o centro da unidade instrucional e tem participação ativa. Isto permite que eles se observem, investiguem, façam inferências a partir de dados e formulem hipóteses, incentivando-os a se tornarem alunos ativos. O professor tem função de facilitador, fornecendo

	suporte e dando <i>feedbacks</i> quando necessário. Atividades desenvolvidas podem ser: resolução de problemas, formação e obtenção de conceitos, discussão reflexiva.
3.Instrução Interativa	O ambiente de ensino é criado para estimular a interação entre o professor e o aluno, possibilitando a colaboração e debates construtivos. Este método é realizado com discussões e projetos em grupos.
4.Aprendizagem Experiencial	Esta estratégia tem foco no processo de aprendizagem, sendo centrado no aluno. A ênfase no aprendizado experiencial está no processo de aprendizado e não no seu produto, no qual testes e provas perdem importância. Exemplos podem ser: conduzir uma pesquisa de opinião pública, realizar simulações.
5.Estudo Independente	Este método foca em estimular o desenvolvimento de iniciativa individual do aluno. Os alunos, de maneira independente, interagem com o conteúdo sem o controle externo do professor. Podem ser desenvolvidas também atividades sobre a supervisão de um professor.

Fonte: elaborado pelo autor.

A escolha do método instrucional deve ser adotada focando em alcançar os objetivos de aprendizagem pretendidos. Além de especificar as estratégias instrucionais, também são definidos os instrumentos de avaliação. Exemplos de instrumentos de avaliação são: provas discursivas e/ou objetivas, trabalhos de pesquisa, trabalhos de implementação, apresentações.

Desenvolvimento

Após a fase de projeto estar definido, é realizada a fase de desenvolvimento, na qual são elaborados os conteúdos e os materiais didáticos a serem utilizados durante a unidade, colocando em ação o que foi planejado nas fases de planejamento e projeto. O material é preparado conforme foi projetado na fase anterior e na seleção do método instrucional selecionado. Tais materiais podem ser: slides para aulas expositivas, listas de exercícios, rubricas, guias, aplicativos exemplos, entre outros. Outras ferramentas também são utilizadas, como papel, caneta, processadores de texto, editor de gráfico, software de programação etc.

Nesta fase também devem ocorrer as instalações e configurações de ambientes virtuais e outras ferramentas de software para o ensino da UI. Além disso, é feito o preparo administrativo, como a criação de contas para os alunos acessarem estes ambientes e/ou ferramentas que devem ser realizados (GERMANIA, PATRICIA & EDMUNDO, 2011).

Por fim, são realizados testes e revisões para verificar se as atividades e os materiais instrucionais atendem às especificações definidas, e avaliar se existem

problemas e restrições que inviabilizem o ensino da unidade instrucional realizados. Esses testes são realizados nos materiais instrucionais, como slides, especificação do trabalho prático, os instrumentos de avaliação dos alunos, as ferramentas de software etc.

Implementação

Nesta etapa a unidade instrucional desenvolvida é aplicada para direcionar os alunos à aprendizagem. Os alunos interagem com os materiais instrucionais, instrutores e com outros alunos. Caso o instrutor não possua todas as qualificações para aplicar a UI, deve-se realizar um treinamento prévio à aplicação na unidade (GERMANIA, PATRICIA, & EDMUNDO, 2011).

Nas primeiras aplicações da unidade instrucional podem ser realizadas observações piloto para analisar a aplicação dos materiais instrucionais, das atividades, e verificar se os instrutores e alunos estão utilizando os materiais de forma adequada (GERMANIA, PATRICIA, & EDMUNDO, 2011). É recomendável também realizar controle do tempo das aulas, para verificar se o tempo está conforme o planejado. Durante a aplicação da unidade instrucional pode ser que sejam necessárias mudanças na sequência lógica; assim, essas e outras mudanças devem ser anotadas para serem utilizadas como melhoria nas próximas aplicações (YUSOF & HASSAN, 2011).

Realizar *feedback* dos instrutores pode ser importante para verificar se a unidade instrucional está sendo conduzida conforme planejado, se os alunos estão aprendendo, se os materiais estão atendendo as expectativas, e também se há alguma melhoria para aprimorar o ensino da UI. Também é importante realizar *feedback* dos alunos, buscando entender como está sendo a experiência de aprendizagem, podendo assim se obter uma melhor compreensão do público alvo realizado na fase de análise (AZIZ, RASLI, & RAMLI, 2010). Por fim, é feita a coleta de dados para avaliação da UI.

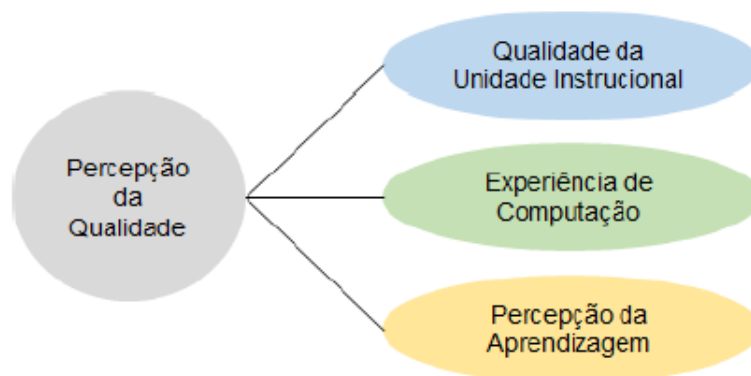
Avaliação

Nesta fase é verificada se a unidade instrucional atendeu aos objetivos de aprendizagem e identificar os pontos de melhoria. Com a finalidade de atingir o objetivo da avaliação, é abordado um estudo de caso sobre a aplicação da UI. Esta

abordagem é um estudo empírico, profundo e exaustivo de um ou poucos objetos, no qual se compreendem/investigam os fenômenos do objeto de estudo, no caso da aplicação da unidade instrucional desenvolvida (YIN, 2014).

Neste trabalho, um processo de avaliação focado em estudos empíricos é adotado para conduzir as etapas da avaliação da unidade instrucional (WOHLIN, RUNESON, & HÖST, 2012). Como parte das etapas deste processo este trabalho utiliza o modelo *Evaluating TEaching CompuTing* (dTECT) (GRESSE *et al.*, 2017). O modelo dTECT é focado no ensino de computação na Educação Básica e tem como objetivo avaliar a qualidade de uma unidade instrucional sob a perspectiva dos alunos, definindo o objetivo, perguntas de pesquisa e questionários da avaliação. A avaliação ocorre por meio dos seguintes aspectos de qualidade (dimensões): qualidade da UI, experiência de computação e percepção da aprendizagem do ponto de vista de alunos (Figura 10) (GRESSE *et al.*, 2017).

Figura 10: Decomposição dos fatores de qualidade do modelo dTECT.



Fonte: GRESSE *et al.* (2017).

O modelo dTECT define um instrumento de medição (questionário) ser respondido pelos alunos após o término da unidade instrucional, de modo a obter a sua percepção sobre qualidade do curso (Quadro 11).

Quadro 11: Instrumento de avaliação.

No.	Descrição do Item	Formato das respostas
Qualidade da Unidade Instrucional		
1	A oficina foi:	(1) Muito Divertida (2) Divertida (3) Chata (4) Muito chata
2	O tempo das aulas passou:	(1) Muito rápido (2) Rápido (3) Devagar (4) Muito devagar
3	A oficina foi:	(1) Excelente (2) Boa (3) Regular (4) Ruim
Experiência de Computação		
4	Vou mostrar meu programa de computador para outras pessoas:	(1) Sim (2) Não
5	Quero aprender mais sobre como fazer programas de computador:	(1) Sim (2) Não
6	Fazer um programa de computador é:	(1) Muito Divertido (2) Divertido (3) Chato (4) Muito chato
7	Eu gosto de fazer programas de computador:	(1) Sim (2) Não
8	A computação é útil no dia a dia:	(1) Sim (2) Não
9	Você já pensou em trabalhar com computação?	(1) Sim (2) Não
Percepção da Aprendizagem		
10	A oficina foi:	(1) Muito fácil (2) Fácil (3) Difícil (4) Muito Difícil
11	Eu consigo fazer programas de computador:	(1) Sim (2) Não
12	Conseguo explicar para um amigo/amiga como fazer um programa de computador:	(1) Sim (2) Não
13	Fazer um programa de computador é:	(1) Muito fácil (2) Fácil (3) Difícil (4) Muito Difícil

Fonte: GRESSE *et al.* (2017).

2.9. O ENSINO DA COMPUTAÇÃO NO ENSINO BÁSICO

A estrutura do ensino no Brasil é composta por Ensino Básico e Ensino Superior. O Ensino Básico é composto pela Educação Infantil, Ensino Fundamental e o Ensino Médio, sendo o Ensino Fundamental subdividido em Fundamental I e Fundamental II (MEC, 2011) (Tabela 1).

Tabela 1: Estrutura do 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).

Os objetivos e conteúdo dos ensinos são realizados conforme estabelecido nos Parâmetros Curriculares Nacionais (PCNs), que formam uma base comum para os currículos nacionais. No Ensino Fundamental, foco específico deste trabalho, espera-se do indivíduo que completa este nível de ensino competências suficientes para se exercer atividades básicas na sociedade. Para isso, deve-se abordar diversas áreas de conhecimento como: Língua Portuguesa, Matemática, História, Geografia, Ciências Naturais, Educação Física, Artes e Língua Estrangeira, porém não consta nenhuma referência à área de Computação (PCN, 2015).

A computação é a habilidade mais requerida do indivíduo no século XXI, sendo importante para o cidadão ser bem-educado em um mundo intensivo em computação e estar preparado para as carreiras do século XXI independente da área de atuação (BLIKSTEIN, 2008). Nesta linha, a Sociedade Brasileira da Computação salientou a necessidade da computação dentro da PNC ao citar no seu plano de gestão:

"... É importante salientar que devemos primar pela qualidade do ensino em todos os níveis da cadeia de formação de recursos humanos. Entendemos que a Computação deva ser ensinada desde o Ensino Fundamental, a exemplo de outras ciências como Física, Matemática, Química e Biologia. Esses são pontos muito importantes para que no futuro tenhamos recursos humanos qualificados para enfrentar os desafios que advirão" (SBC, 2015).

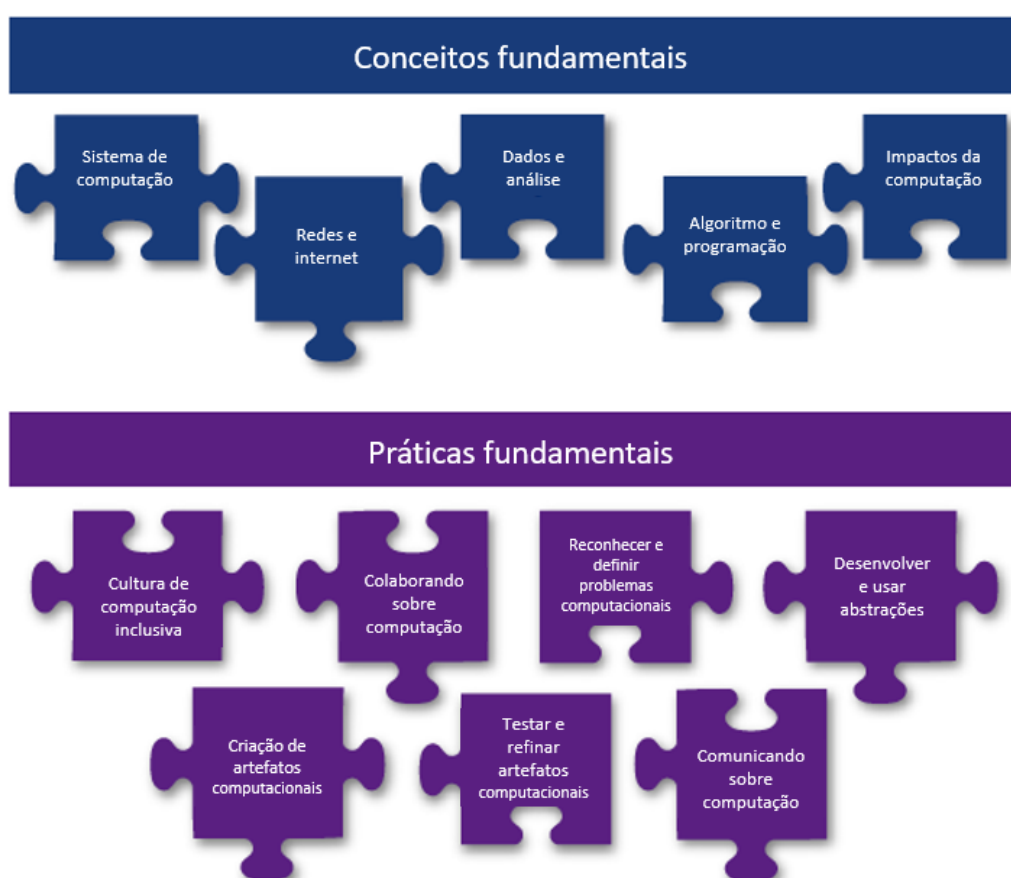
Ainda, a SBC estabeleceu uma proposta inicial que trata as competências e habilidades que compõem a computação (SBC, 2017a). No Brasil, atualmente, não existe um currículo de referência para o ensino de computação no Ensino Básico, mas internacionalmente existem vários, e entre os mais reconhecidos está o CSTA/ACM K-12 (CSTA, 2017) criado pela *Computer Science Teacher Association e Association for Computing Machinery* (ACM).

2.9.1. Framework Computer Science Teacher Association K-12

O *framework* CSTA/ACM K-12 é um guia padronizado, para que as escolas de ensino básico possam implantar o ensino da computação. O *framework* serve como um guia de alto nível, que pode ser usado para o desenvolvimento de currículos personalizados especificamente de computação (CSTA, 2017).

Neste guia o ensino da computação é baseado nos conceitos e práticas (Figura 11). Os conceitos são as principais áreas de conteúdo da computação, que são: sistemas da computação, internet e redes de dados, análise de dados, algoritmos e programação e impactos da computação. As *práticas* são comportamentos e formas de pensar que os alunos devem adotar enquanto aprendem e implementam os conceitos descritos na grade curricular. Por exemplo, os alunos devem saber abstrair, modelar e decompor um problema, saber avaliar e selecionar ferramentas tecnológicas que podem ser usadas para colaborar em um projeto, criar artefatos computacionais para demonstrar e aumentar seu conhecimento de algoritmos etc. (CSTA, 2017).

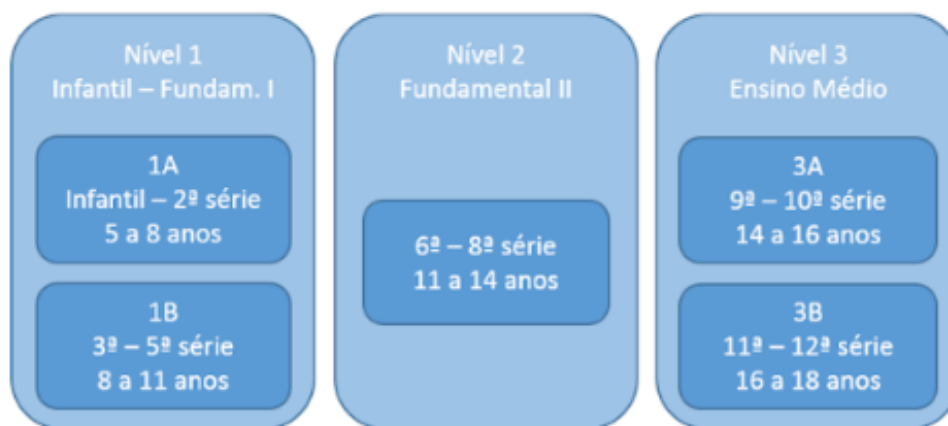
Figura 11: Conceitos e práticas do K-12.



Fonte: CSTA (2017) - versão traduzida.

O pensamento computacional é um dos elementos principais no ensino de computação. O aluno deve desenvolver a capacidade humana de formular problemas, para que suas soluções possam ser representadas como etapas ou algoritmos computacionais, a serem realizados por meio de um computador. Nesse contexto, o guia possui um currículo contendo práticas e conceitos de computação que deve ser ensinado para alunos em cada fase do Ensino Infantil, do Ensino Básico, até o Ensino Médio (Figura 12).

Figura 12: Níveis de ensino K-12.



Fonte: WANGENHEIM, ALVES & WEBER (2017) traduzindo o CSTA (2017).

O nível 1 tem o propósito de fornecer padrões de aprendizagem para alunos do ensino primário até o quinto ano. Neste nível são criadas experiências de aprendizado, para que o aluno identifique a importância da computação no cotidiano dele mesmo. No nível 2, o indivíduo deve começar a usar o pensamento computacional como uma ferramenta para solução de problemas. No nível 3 o currículo define que é preciso criar momentos para a aplicação de conhecimentos mais avançados de computação para criar soluções aplicáveis no “mundo real” (CSTA, 2017). No contexto deste projeto, o ensino da computação é limitado somente no nível 2 da Educação Básica. O Quadro 12 apresenta o que um aluno até o final do nível 2 deve ter aprendido utilizando a Educação Básica no que se refere apenas a aprendizagem de conceitos de algoritmo e programação.

Quadro 12: Objetivos de aprendizagem de computação para Ensino Fundamental II.

Objetivo de aprendizagem	Subconceito	Prática
Usar fluxogramas e/ou pseudocódigo para resolver problemas complexos como algoritmos.	Algoritmos	Desenvolver e usar abstrações
Criar variáveis com definição clara que representem diferentes tipos de dados e executem operações sobre seus valores.	Variáveis	Criar artefatos computacionais
Projetar e desenvolver interativamente programas que combinem estruturas de controle, incluindo loops aninhados e condicionais compostos.	Controle	Criar artefatos computacionais
Decompor problemas e subproblemas em partes para facilitar o projeto, implementação e revisão de programas.	Modularidade	Reconhecer e definir problemas computacionais
Criar procedimentos com parâmetros para organizar o código e torná-lo mais fácil de reutilizar.	Modularidade	Desenvolver e usar abstrações
Distribuir tarefas e manter uma linha de tempo para o projeto, quando se está desenvolvendo artefatos computacionais de forma colaborativa.	Desenvolvimento de Programas	Colaborar em torno da computação
Procurar e incorporar comentários (<i>feedback</i>) de membros da equipe e usuários, para refinar uma solução que atenda às necessidades dos usuários.	Desenvolvimento de Programas	Colaborar em torno da computação; promover uma cultura de computação inclusiva
Incorporar código, mídia e bibliotecas existentes em programas originais e fornecer atribuição (ao autor original).	Desenvolvimento de Programas	Desenvolver e usar abstrações; criar artefatos computacionais; comunicar sobre a computação
Testar sistematicamente e refinar programas usando uma variedade de casos de teste.	Desenvolvimento de Programas	Testar e refinar artefatos computacionais
Documentar programas para torná-los mais fáceis de entender, testar e depurar.	Desenvolvimento de Programas	Comunicar sobre a computação

Fonte: WANGENHEIM, ALVES & WEBER (2017).

O currículo fornece também alguns objetivos de aprendizagem relacionados referente a competências da disciplina de ES e EU (Quadro 13).

Quadro 13: Objetivos de aprendizagem referente a ES e EU voltado para Ensino Fundamental II.

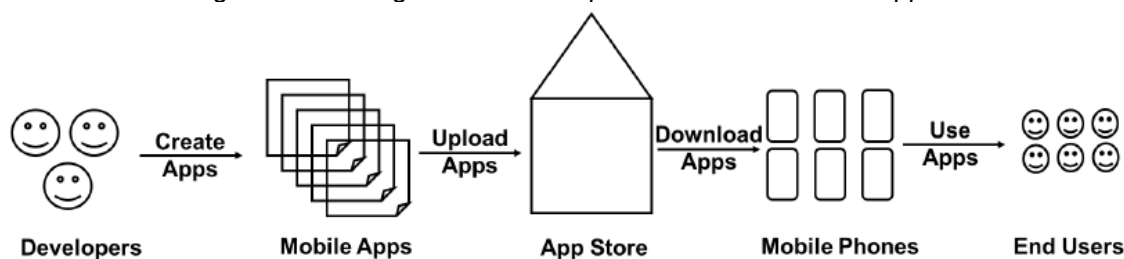
Objetivo de aprendizagem	Área de conhecimento	Fonte
Desenvolver artefatos computacionais interativamente de forma colaborativa, seguindo um cronograma.	Engenharia de Software	(CSTA, 2017: 2-AP-18)
Identificar e resolver problemas criando sistemas de software interativos.	Algoritmo e programação / Engenharia de Usabilidade	(CSTA, 2017: 1B-CS-03, 3A-AP-13)
Analisar o contexto de sistemas de software interativo em termos de usuários, tarefas,	Engenharia de Usabilidade	(CSTA, 2017: 1B-IC-19)

dispositivos e ambientes de uso.		
Especificar requisitos de sistemas de software interativos em termos de funcionalidade e usabilidade.	Engenharia de Software / Engenharia de Usabilidade	(CSTA, 2017: 2-AP-19),
Criar protótipos de sistemas de software interativos em diferentes níveis (esboços, baixa fidelidade, alta fidelidade, funcional).	Engenharia de Usabilidade	(CSTA, 2017: 2-AP-19, 1B-AP-13, 2-AP-13),
Projetar design que combine componentes de <i>hardware</i> e software para coletar e trocar dados (sensores, APIs, etc.).	Engenharia de Software / Engenharia de Usabilidade	(CSTA, 2017: 2-CS-02),
Projetar o design visual (cores, tipografia, ícones, imagens, etc.) do sistema de software interativo.	Engenharia de Usabilidade	(CSTA, 2017: 2-IC-21)
Procurar e incorporar o <i>feedback</i> dos membros da equipe e dos usuários para refinar uma solução que atenda às necessidades do usuário.	Algoritmo e programação/ Engenharia de Software/Engenharia de Usabilidade	(CSTA, 2017: 2-AP-15)
Testar e refinar um sistema de software interativo para funcionalidade e usabilidade.	Engenharia de Software	(CSTA, 2017: 2-AP-17, 1B-AP-15, 3A-AP-21)
Recomendar melhorias no design de dispositivos de computação, com base nos resultados de verificação e validação.	Engenharia de Software	(CSTA, 2017: 2-CS-01)

2.10. DESENVOLVIMENTO DE APPS COM APP INVENTOR

Aplicativo (*app*) é um programa de computador, desenvolvido para ser instalado em um dispositivo móvel, como *tablets*, *smartphones*, Assistente Pessoal Digital (PDA), etc. (FLING, 2009). A disponibilização de um *app* fica centralizada em um mercado de *apps*, como a *Google Play Store* do sistema operacional *Android*, no qual os desenvolvedores fazem *uploads* dos *apps* para que os usuários possam baixá-los e instalá-los (NAGAPPAN & SHIHAB, 2016) (Figura 13).

Figura 13: Visão geral das várias partes interessadas dos apps.



Fonte: NAGAPPAN & SHIHAB (2016).

Os *apps* podem ser: i) nativos, exclusivos para um determinado sistema operacional, e ii) multiplataformas, compatíveis com mais de um sistema operacional móvel existente, como *Android*, *IOS*, *WindowsPhone*, *BlackBerry*, dentre outros (RAJ & TOLETY, 2012).

Existem diversas ferramentas para criar *apps*, para as mais diversas plataformas de *smartphones*. No contexto educacional, para o ensino de computação para o público mais jovem, tipicamente se adotam ambientes de programação baseados em blocos, como o *App Inventor* (WOLBER, 2011). *App Inventor*⁴ (AI) é um ambiente de programação que permite às pessoas de todas as idades criarem aplicativos para dispositivos móveis (*tablets*, celulares) que utilizam o sistema operacional *Android* (MIT & GOOGLE, 2012). Com o AI é possível criar *apps* de diversas finalidades, como, por exemplo, utilidade pública (Figura 14), entretenimento (Figura 15) e fins educacionais (Figura 16).

⁴ <http://appinventor.mit.edu>

<p>Figura 14: CliqueDenúncia⁵</p> 	<p>Figura 15: Minigolf</p> 	<p>Figura 16: MATHfull⁶</p> 
<p>App que realiza as denúncias de problemas que são recorrentes na comunidade.</p>	<p>App que simula um jogo de Golf (MIT & GOOGLE, 2012b).</p>	<p>Quiz contendo as quatro operações matemáticas e que pode ser utilizado por estudantes do Ensino Fundamental.</p>

O AI é uma ferramenta acessada via *web* (navegador) e adota o paradigma de programação visual baseado em blocos. Dessa forma, para programar as instruções de código os blocos são arrastados na tela (WOLBER, 2011). Isso torna a programação intuitiva e diminui o tempo para a criação de aplicativos complexos e de alto impacto em comparação com ambientes de programação tradicionais (POKRESS & DOMÍNGUEZ, 2013). Além disso, permite ao programador se concentrar na lógica de programação ao invés da sintaxe da linguagem de codificação (POKRESS & DOMÍNGUEZ, 2013).

A interface de usuário do AI consiste em 2 perspectivas:

1. *Designer* (Figura 17): utilizado para o desenvolvimento da interface do aplicativo, permitindo que o usuário selecione componentes (imagem, botão, textos etc.) e os arraste para a tela, como também definir suas propriedades. Esta visão é dividida em 4 colunas:

⁵ ClicDenuncia: http://www.computacaonaescola.ufsc.br/?page_id=2412

⁶ Mathfull: <http://www.computacaonaescola.ufsc.br/?p=2387>

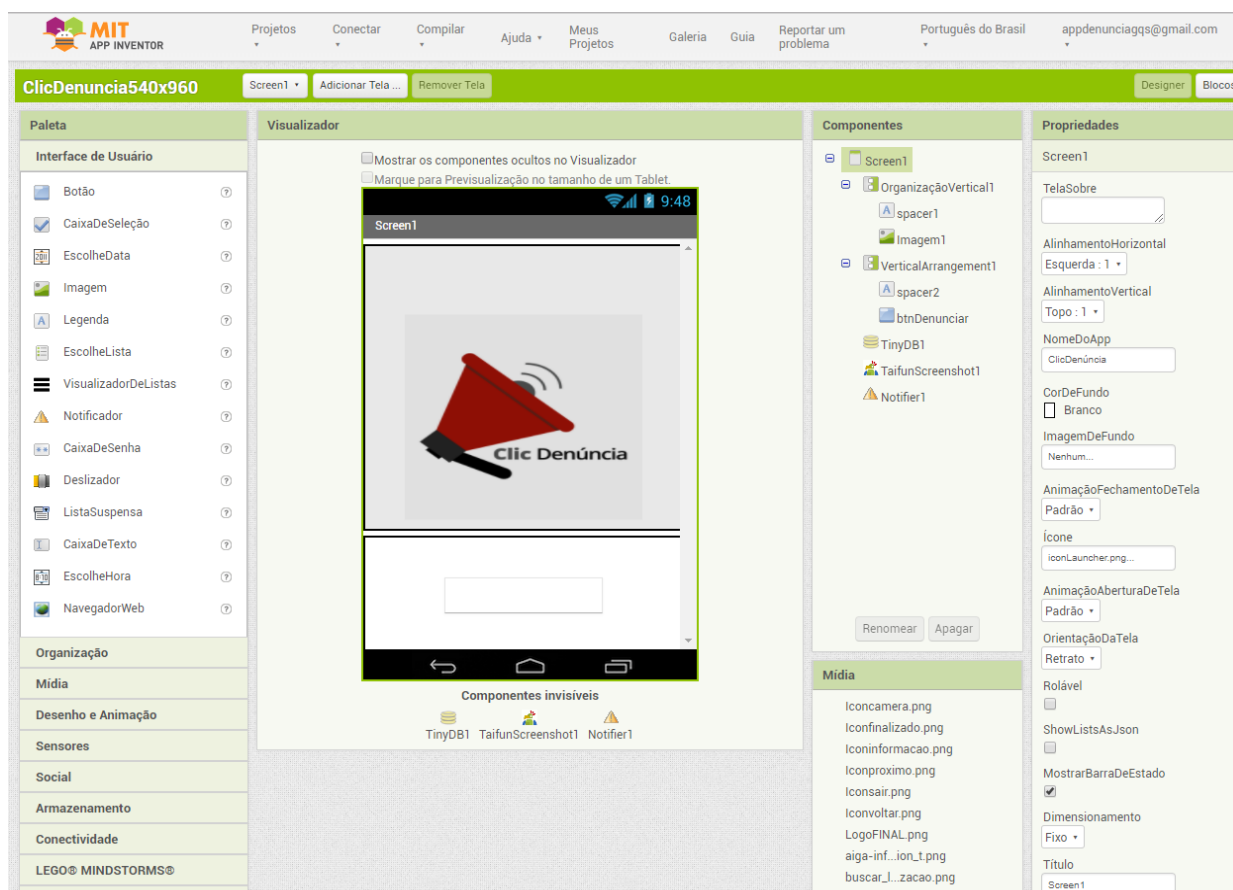
a) *Paleta*: coluna que contém os componentes utilizáveis em um *app*. Ela é dividida em seções para facilitar a localização dos componentes. Um componente pode ser um botão, imagem, notificadores etc.;

b) *Visualizador*: Neste lugar são incluídos os componentes para montar o aplicativo. Esta coluna simula a tela de um *smartphone* e apresenta uma versão visual da interface gráfica do *app*.;

c) *Componentes*: Nesta coluna ficam os componentes, seja eles visíveis ou não, organizados de forma hierárquica. Nesta coluna pode-se renomear os componentes e inserir arquivos de mídia (sons, fotos, vídeos);

d) *Propriedades*: Coluna que contém as propriedades de cada componente, como o seu tamanho, altura, conteúdo textuais dos botões, tamanho de imagens, cores de fundo, etc.

Figura 17: Perspectiva de *Designer*.



Fonte: elaborada pelo autor com base em appinventor.mit.edu.

Dentre os componentes presentes no Designer do AI, tem-se não só componentes que fazem parte da interface gráfica do usuário, como também

sensores (GPS, acelerômetro, pedômetro, leitor de código de barra), Mídia (Câmera, Som), armazenamento de dados, entre outros (MIT & GOOGLE, 2012c).

2. *Editor de blocos* (Figura 18): é a perspectiva que permite associar ações aos componentes e funcionalidades dos aplicativos. As instruções são representadas pela união dos blocos de programação, no estilo de um quebra cabeça.

Figura 18: Editor de Blocos.



Fonte: elaborada pelo autor com base em appinventor.mit.edu.

Os comandos do editor de blocos são agrupados em categorias, conforme Quadro 14:

Quadro 14: Comandos do *App Inventor* agrupados em categorias.

Categorias de blocos de código	Descrição
Controle	Blocos de controle para a programação. Exemplos: desvio condicional, loops.
Lógica	Blocos de operações lógicas como <i>AND</i> ou <i>OR</i> , para verificar se dois valores são iguais ou diferentes.
Matemática	Blocos com operações matemáticas (aritmética e geometria) e funções

	para selecionar um número randômico, entre outros.
Texto	Blocos para tratamentos de texto, como cortar textos, comparar textos, unir dois textos etc.
Lista	Blocos para criação e manipulação de listas de elementos.
Cores	Blocos para criar cores utilizando o sistema RGB.
Variáveis	Blocos para criação e manipulação de variáveis, como: declaração de variável local, atribuir valor a uma variável, obtenção de valor.
Procedimentos	Blocos para criação e manipulação de funções.

A ferramenta AI é um software *open source* que é executado via um navegador *web* (*browser*), necessitando que um computador *desktop* esteja conectado a uma rede internet (MIT & GOOGLE, 2012d). A AI *online* é acessado via conta Google⁷. Assim, a autenticação de *login* e o armazenamento do projeto são vinculados a uma conta de forma exclusiva. Ela salva automaticamente os projetos sendo desenvolvidos na base de projetos da conta do desenvolvedor *online*.

O *App Inventor* oferece a possibilidade de *live testing* para que o *app* em desenvolvimento seja testado em tempo real.

O *live testing* pode ser feito de 3 formas (MIT & GOOGLE, 2019e):

- *Via Celular (recomendado)*: Para isso, é preciso baixar e instalar o aplicativo denominado “MIT AI2 Companion” no celular e estar conectado na internet sem fio do tipo *Wi-Fi*. O dispositivo e o computador que estiverem executando a AI devem estar executando-os na mesma rede;

- *Emulador*: caso não tiver um *smartphone* ou acesso ao *Wi-Fi*, a AI fornece um celular virtual que pode ser executado por meio do computador;

- *Cabo USB*: nesta opção também não é necessária uma rede *Wi-Fi*, apenas um dispositivo móvel executando o *Android* e um cabo USB.

O *app* desenvolvido pode ser compartilhado das seguintes formas (MIT & GOOGLE, 2019e):

- *Exportar Projeto (arquivo aia)*: utilizada para exportar o projeto do AI para o computador pessoal, importá-lo em outro computador ou compartilhar o próprio código com outras pessoas;

- *Baixar Executável*: opção para gerar um executável do *app* para ser *uploaded* e executado em qualquer dispositivo móvel que tenha *Android*;

⁷ FAQ do *App Inventor*: <http://appinventor.mit.edu/explore/teach/faq.html>

- *Google Play Store*: Para isso é necessária uma conta do tipo desenvolvedor na *Google Play Store*, envolvendo uma taxa;

- *Gallery*: A galeria é um ambiente colaborativo de *apps* desenvolvido com a AI, permitindo o compartilhamento do projeto (aia) para outras pessoas poderem ter acesso e modificar/evoluir *apps*. Porém, *Gallery* não permite o upload de *apps* que usam extensões.

3. ESTADO DA ARTE

O presente capítulo apresenta o estado atual de pesquisas relacionadas a UIs que ensinam computação por meio do desenvolvimento de *apps*, abordando práticas das disciplinas de ES e EU no Ensino Básico. Foi levantado um estado da arte em relação a unidades instrucionais que abordam ES e outro em relação às que ensinam EU.

Esta análise é realizada seguindo o processo de Mapeamento Sistemático da Literatura (MSL), proposto por Petersen *et al.* (2008).

3.1. ESTADO DA ARTE DO ENSINO DE ES NA EDUCAÇÃO BÁSICA

Este mapeamento foi realizado por meio da produção de um relatório técnico, executado em cooperação no laboratório de pesquisa (PINHEIRO *et al.*, 2018).

O objetivo deste mapeamento sistemático da literatura é levantar quais UIs existentes (e quais as suas características) que ensinam competências de ES no contexto de ensino de computação na Educação Básica. Essa questão de pesquisa é refinada nas seguintes perguntas de análise:

PA1. Quais UIs existem?

PA2. Qual(is) competências de ES são ensinadas na UI?

PA3. Quais as características instrucionais da UI?

PA4. Quais são as características de contexto da UI?

PA5. Como a unidade instrucional foi desenvolvida?

PA6. Como a qualidade da unidade instrucional é avaliada?

- *Critérios de inclusão/exclusão:* São incluídos apenas artigos que passaram por revisão por pares e que visa ensinar computação adotando práticas e/ou atividades de ES na Educação Básica. Literaturas secundárias descoberta por meio das referências da literatura primária encontrada (VERHOEFF, 2006) também foram incluídas. São excluídos artigos que focam em ensinar computação para a educação superior e/ou artigos que apresentam UIs para o ensino de computação sem abordar conceitos de ES.

- *Critério de Qualidade:* Considera-se apenas artigos que apresentam informações relevantes sobre o ensino de conceitos de ES, indicando, por exemplo, conteúdo das aulas, materiais didáticos, etc.;

- *Fonte de dados:* A busca foi feita no Scopus (www.scopus.com) e sites para ensino online (MOOCs), incluindo Udemy (www.udemy.com), Edx (www.edx.org),

Khanacademy (www.khanacademy.org), Coursera (www.coursera.org), entre outros. O Scopus foi utilizado por realizar a busca nas bases das principais editoras científicas. Os sites de ensino *on-line* foram utilizados para pesquisar UIs ensinadas por meio de cursos on-line;

- *Definição da string de busca:* A *string* de busca foi composta de conceitos relacionados à questão de pesquisa, como por exemplo “Engenharia de Software” e também também sinônimos (Quadro 15). O termo “Educação Básica” é utilizado para restringir o nível de ensino focado pela UI. O termo “unidade instrucional” representa o objetivo de identificar e caracterizar diversos tipos de UIs.

Quadro 15: Palavras-chave.

Conceito principal (termo)	Sinônimos	Tradução
Engenharia de Software	UML, processo de desenvolvimento de Software	Software <i>engineering</i> , UML, Software <i>development process</i>
Educação Básica	Ensino Primário, Ensino Fundamental, Ensino Médio	School, K-12
Unidade instrucional	Curso, ensino, aprendizagem	Learn, teaching, MOOC

Usando essas palavras-chave, a *string* de pesquisa foi calibrada e personalizado em conformidade com a sintaxe específica para cada um dos dados fontes apresentadas (Quadro 16). A busca dos cursos foi feita nos MOOCs seguindo as categoria e subcategoria relacionada a Tecnologia da Informação e desenvolvimento de software, respectivamente. A partir disso foi feita uma busca manual de cursos que ensinam programação para o público mais jovem, contendo algum ensino de ES.

Quadro 16: *String* de busca.

Fonte de Dados	String de busca
Scopus	("software <i>engineering</i> " OR <i>uml</i> OR "software <i>development process</i> ") AND (<i>school</i> OR "K-12") AND (<i>teaching</i> OR <i>learn</i> OR MOOC)

3.1.1. Execução da busca

A pesquisa foi executada em março de 2018 pelo autor. A partir dos resultados da busca, foram selecionados artigos potencialmente relevantes de acordo com os critérios de inclusão e exclusão, analisando rapidamente o título, o resumo e as palavras-chave (Tabela 2). Como resultado, foram identificados 29 artigos potencialmente relevantes do SCOPUS sendo que apenas 15 são artigos relevantes.

Tabela 2: Quantidade de artigos por etapa de seleção por repositório.

Fonte de dados	Resultado da pesquisa inicial	Seleção depois do 1º estágio	Seleção depois do 2º estágio
SCOPUS	466	29	15

3.1.2. Análise dos dados

PA1. Quais UIs existem?

O resultado da pesquisa retornou 17 UIs voltadas ao ensino de computação, que de alguma forma abordam também o ensino de ES ao nível da Educação Básica (Quadro 17).

Quadro 17: Artigos relevantes.

Citação	Título do artigo
BOLLIN & SABITZER, (2015)	Teaching Software Engineering in schools on the right time to introduce Software Engineering concepts.
COLLOFELLO (2002)	Creation, deployment and evaluation of an innovative secondary school software development curriculum module.
CORBETT & NESIBA (2015)	Programming design process: Providing K-12 students with a structure to attain programming goals.
DE KERKEKI & MANATAKI (2018)	Code Yourself! An introduction a programming. "Code Yourself" and "A Programar": a bilingual MOOC for teaching Computer Science to teenagers.
FRONZA, IOINI & CORRAL (2017)	Teaching Computational Thinking Using Agile Software Engineering Methods: A Framework for Middle Schools.
FRONZA, IOINI & CORRAL (2016)	Blending Mobile Programming and Liberal Education in a Social-Economic High School.
FRONZA, IOINI & CORRAL (2015)	Students want to create <i>apps</i> : leveraging computational thinking to teach mobile software development.
HERMANS & AIVALOGLU (2017)	Teaching software engineering principles to K-12 students: a MOOC on <i>Scratch</i> .
KÖHLER <i>et al.</i> (2012)	Teaching Basic Software Engineering to Senior High School Students.
MISSIROLI <i>et al.</i> (2017)	Agile for Millennials: A Comparative Study.
MISSIROLI <i>et al.</i> (2016)	Learning Agile Software development in high school: an investigation.
RUSU (2011)	Employing Software maintenance techniques via a tower-defense serious computer game.
RUSU (2010)	Learning Software engineering basic concepts using a five-phase game.

SARKAR & BELL (2010)	Teaching black-box testing to high school students.
SERRANO & SERRANO (2013)	Requisitos ao Código: Uma Proposta para o Ensino da Engenharia de Software no Ensino Médio.
STARRETT (2007)	Teaching UML Modeling Before Programming at the High School Level.
VERHOEFF (2006)	A master class software engineering for secondary education.

Observa-se que até 2012 foram divulgadas poucas UIs, desde que foi encontrado o primeiro relato em 2002. Somente nos últimos anos teve-se um leve aumento, provavelmente também relacionado à tendência de aumento de ensino de computação de forma geral na Educação Básica (Figura 19).

Figura 19: Quantidade de UIs enfocando no ensino de ES na Educação Básica publicado por ano.



Fonte: Elaborada pelo autor.

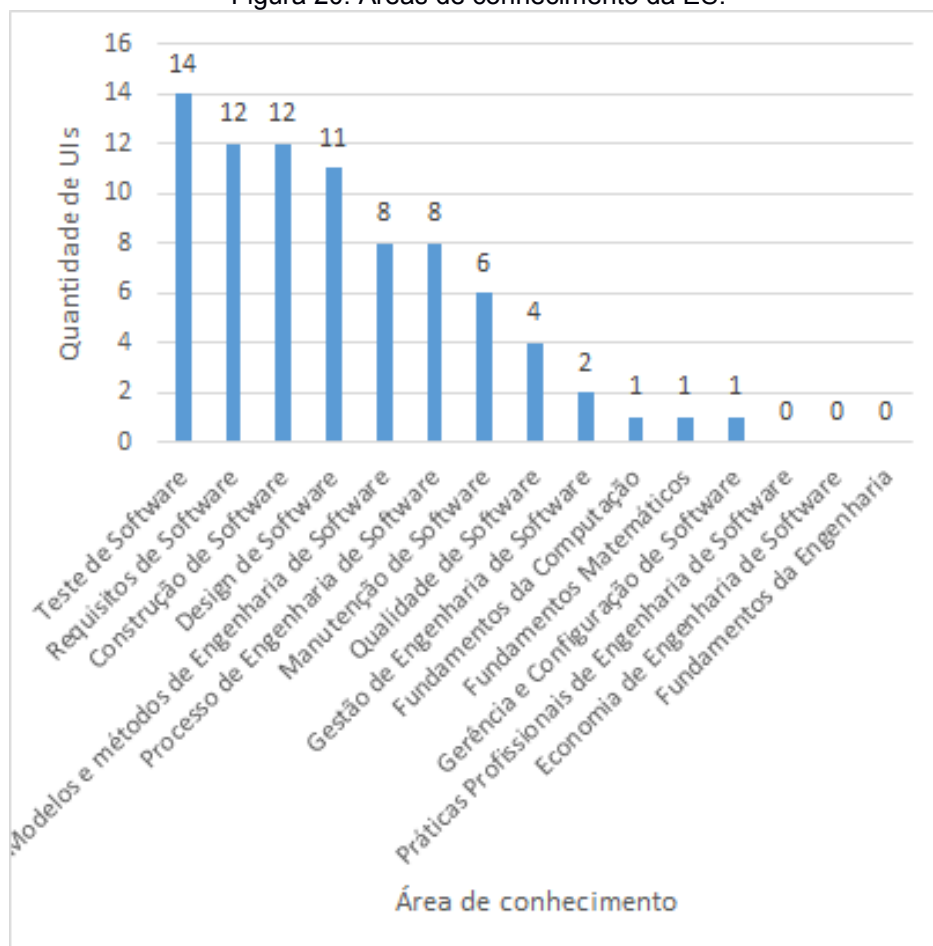
Porém, mesmo com o crescimento nos últimos anos, observa-se que foi encontrado no total um número muito pequeno de UIs que abordam o ensino de ES no nível da Educação Básica.

PA2. Qual(is) competências de ES são ensinados na UI?

As UIs encontradas geralmente focam em ensinar competências de programação incluindo práticas relacionadas a diversas áreas de conhecimento de ES de acordo com o SWEBOK, como: requisitos de software, modelagem, construção e teste de software (Figura 20). A área de conhecimento mais ensinada é a de teste de software, abordando teste de unidade, teste funcionais e teste de aceitação. Algumas técnicas de levantamento e análise de requisitos também amplamente, como *user stories*, diagramas de caso de uso, *storyboards*, especificações de requisitos de software, entre outras (MISSIROLI *et al.*, 2017; FRONZA *et al.*, 2016). Diagramas de classe e de máquina de estado da UML foram utilizados para ensinar modelagem de software por diversas UIs. Também foram

utiliza diagramas de fluxo e pseudocódigo para visualizar a arquitetura e algoritmos dos sistemas de software.

Figura 20: Áreas de conhecimento da ES.



Fonte: Elaborada pelo autor.

Algumas UIs ensinam desenvolver aplicativos móveis abordando design de interface criando protótipos de papel das telas. Referente à construção de software, são abordadas técnicas como a criação de código fonte compreensível (nomeação), reuso de código, programação em pares e estrutura de controle.

Um terço das UIs encontradas relacionadas às fases do processo de software se destaca abordando assuntos da área de manutenção de software. Isso se destaca mais ainda por ser uma área essencial de ES na prática, mas muitas vezes não sendo abordada no ensino de ES na Educação Superior. Assim, várias UIs encontradas focam na reengenharia, adaptação e/ou evolução de software pré-existente. O que aparentemente facilita o ensino de conceitos de manutenção no contexto da Educação Básica são os ambientes de programação baseados em

blocos, como Scratch, que fortemente estimulam e suportam o *remix* de programas (BRENNAN & RESNICK, 2013). Outra estratégia instrucional é propor um jogo que ensina os quatro tipos de manutenção de software (RUSU *et al.*, 2011).

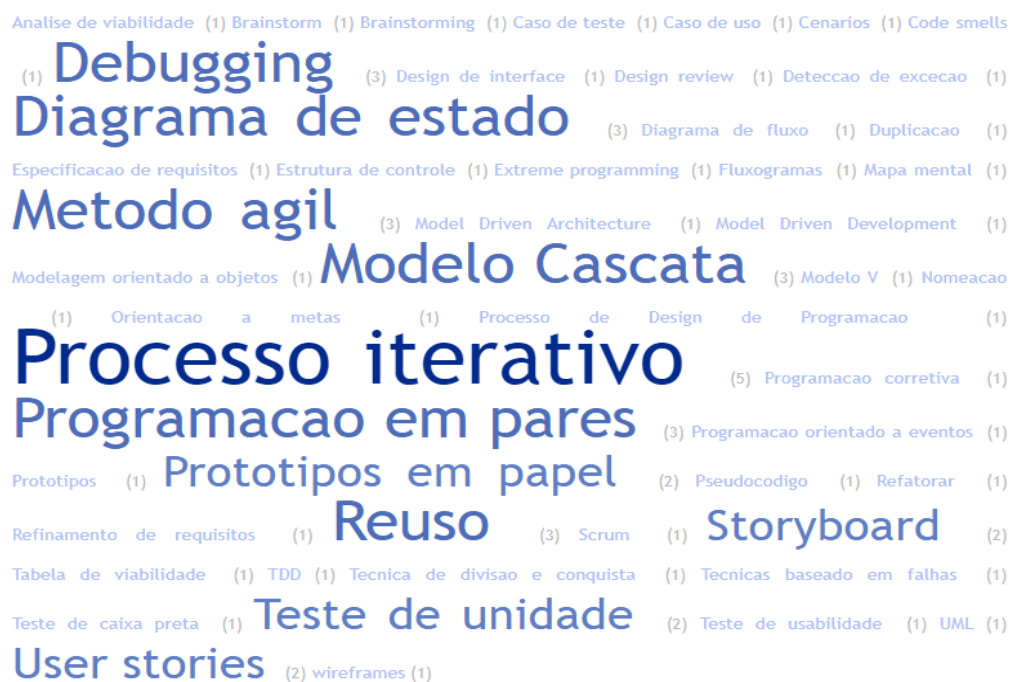
Várias UIs encontradas também ensinam o ensino de conceitos de processos de software, modelos de ciclo de vida e metodologias de desenvolvimento predominando modelos simples de ciclo de vida, como cascata, e o modelo interativo, adotando principalmente metodologias ágeis, como *Scrum* (MISSIROLI *et al.*, 2017), *Extreme programming* (XP) (MISSIROLI *et al.*, 2016), *Model Driven Development* (MDD) (STARRETT, 2007) e *Test-Driven Development* (TDD) (MISSIROLI *et al.*, 2016) (Figura 21). Pode ser também observada a adoção do *Programming Design Process* (PDP), relacionado à área de Engenharia de forma geral, e não especificamente da ES (CORBETT & NESIBA, 2015).

Basicamente todas as UIs encontradas visam levar a aprendizagem do aluno ao nível de aplicação, sendo projetadas de forma a dar a oportunidade aos alunos de executarem processos de ES (BOLLIN & SABITZER, 2015; CORBETT & NESIBA, 2015). As UIs visam a execução de todas as principais fases do processo de software e também outras que focam somente em algumas fases do processo, levando em consideração restrições práticas em relação à duração da UI. Observou-se também que várias UIs foram aplicadas de forma multidisciplinar, integradas em outras disciplinas do currículo da Educação Básica, como Física e Artes.

Pouquíssimas UIs utilizam ferramentas de ES para auxiliar no ensino. Só duas UIs relatam o uso de ferramentas CASE (*Rational Rose* e *NetBeans*) (MISSIROLI *et al.*, 2016; COLLOFELLO, 2002). Outras ferramentas utilizadas incluem ferramentas de testes (JUnit, ferramenta de validação de casos de teste próprio⁸) (MISSIROLI *et al.*, 2016) e uma ferramenta de requisitos i*.

⁸ testBedv9.html

Figura 21: Frequência dos modelos/métodos/técnicas utilizados para ensinar ES na Educação Básica.

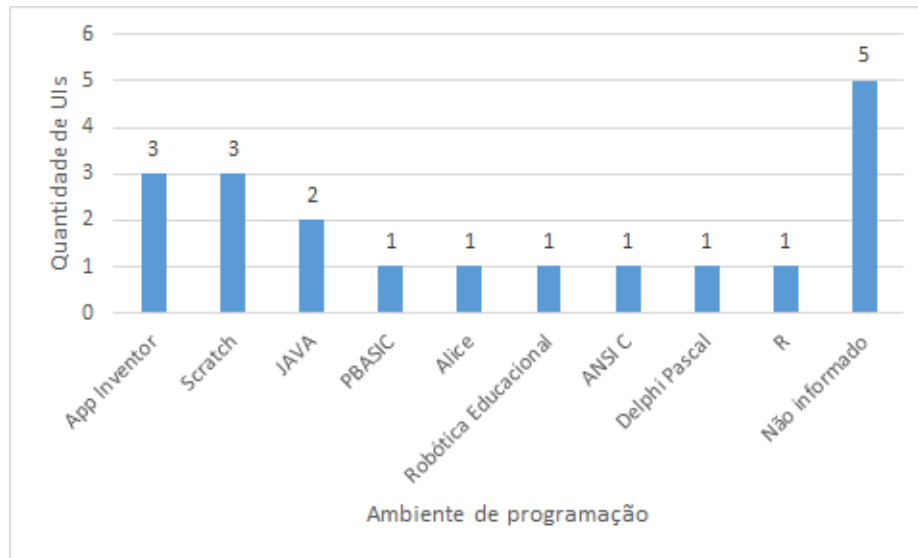


Fonte: elaborada pelo autor.

PA3. Quais as características instrucionais da UI?

O ensino de competências de ES tipicamente está inserido em UIs voltadas, de forma geral, ao ensino de programação de software, com enfoque significativo em algoritmos e programação. Tipicamente essas UIs visam o desenvolvimento de animações, aplicações *web*, aplicativos móveis e robôs (SERRANO & SERRANO, 2013; STARRETT, 2007). Para isso, geralmente se adotam ambientes de programação visual baseados em blocos, como *Scratch* e *Alice*, para desenvolver softwares desktop, e *App Inventor*, para aplicativos móveis. Este tipo de linguagem torna a programação intuitiva e permite que o aluno se concentre na lógica de programação ao invés da sintaxe da linguagem de programação (POKRESS & VEIGA, 2013). Porém, observou-se também a adoção de várias linguagens de programação baseado em textos, como Java (MISSIROLI *et al.*, 2016), Delphi Pascal (VERHOEFF, 2006), ANSI C (STARRETT, 2007) e a linguagem R para computação estatística (Figura 22). Para ensinar o desenvolvimento de programas robóticos, foram utilizados PBASIC, uma versão do BASIC, baseado em microcontrolador, e a linguagem do Lego Robot (STARRETT, 2007).

Figura 22: Ambientes de programação utilizadas no ensino de ES na Educação Básica.



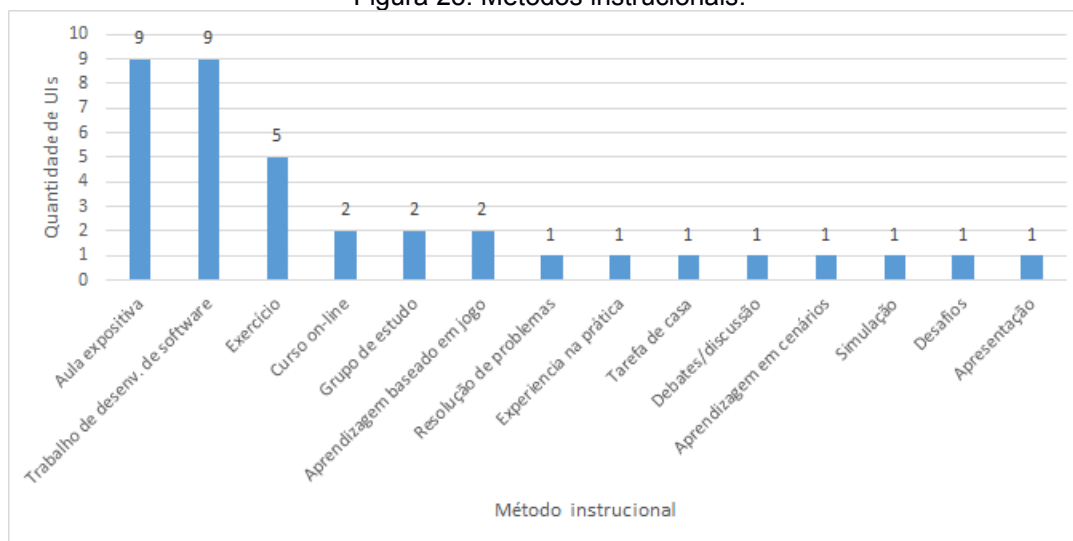
Fonte: elaborada pelo autor.

Poucas Us encontradas focam especificamente no ensino de competências de ES, como por exemplo o *Tower-Defense Serious Computer Game*, para ensinar conceitos de manutenção de software e também a unidade instrucional apresentada por Sarkar e Bell (2013) visando a criação de casos de testes para um software pré-existente.

A aprendizagem ativa (instrução experimental) é o métodos instrucionais relacionadas aos objetivos de aprendizagem e visando a aprendizagem ao nível de aplicação. Muitas Us envolvem a realização de um trabalho de desenvolvimento de software que variam de tarefas bem definidas, com uma especificação do problema a ser resolvido e uma solução esperada como também trabalhos “mal definidos”, sem uma especificação do problema predefinida e sem solução previamente conhecida.

Mesmo focando mais na aprendizagem ativa, diversas Us também incluem outros métodos de instrução direta como aulas expositivas, principalmente na parte inicial da unidade instrucional (BOLLIN & SABITZER, 2015; COLLOFELLO, 2002). Outros métodos instrucionais indiretos mais utilizados foram a resolução de problemas e exercícios em sala de aula, e as leituras e tarefas de casa (Figura 23). Também foram utilizados métodos interativos, como grupos de estudos, atividades em pares, grupos de aprendizagem cooperativa (fórum das Us e ensinado em modo *on-line*) e discussões/debates. Foram também encontradas duas Us que adotam aprendizagem baseada em jogos (RUSU *et al.*, 2011; RUSU *et al.*, 2010).

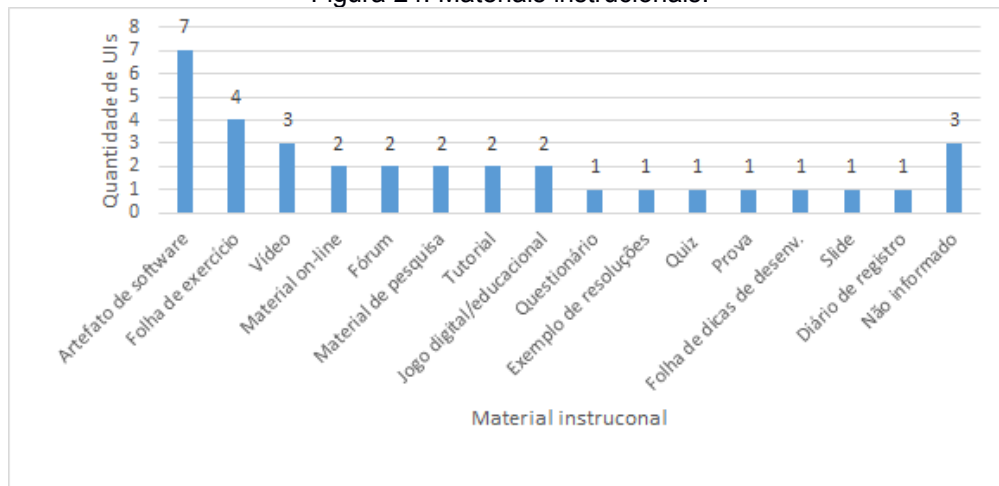
Figura 23: Métodos instrucionais.



Fonte: elaborada pelo autor.

De acordo com esta variação de métodos instrucionais, também são adotados diversos tipos de materiais instrucionais, como artefatos de software que são tipicamente utilizados para auxiliar na aplicação do ensino de ES, como a especificação de requisitos de um software pré-definido, descrição de caso de usos, user stories, amostras de código, entre outros (Figura 24). Também foram utilizados softwares prontos para: i) ensinar a criação de testes de aceitação (SARKAR & BELL, 2013); ii) exemplificar uma solução pronta; e, iii) como estratégia de ensino (RUSU et al., 2011; RUSU et al., 2010). Vídeos instrucionais, tutoriais, fóruns etc., são específicos das UIs realizadas por cursos on-line (COLLOFELLO, 2002). Folhas de exercícios, cadernos de tarefa de casa ou diários, para registrar as experiências adquiridas também foram utilizadas por algumas UIs. Porém, de forma geral, observou-se a apresentação de poucas informações em relação aos materiais nas publicações, e também em relação à sua disponibilidade e licença de uso, o que dificulta o seu uso por outros interessados. A maioria dos materiais também está disponível em somente uma única língua (predominantemente em inglês) (CORBETT & NESIBA, 2015; DE KEREDI & MANATAKI, 2015). A disponibilização de apenas uma versão da UI pode limitar também uma adoção mais ampla da unidade instrucional em outros países.

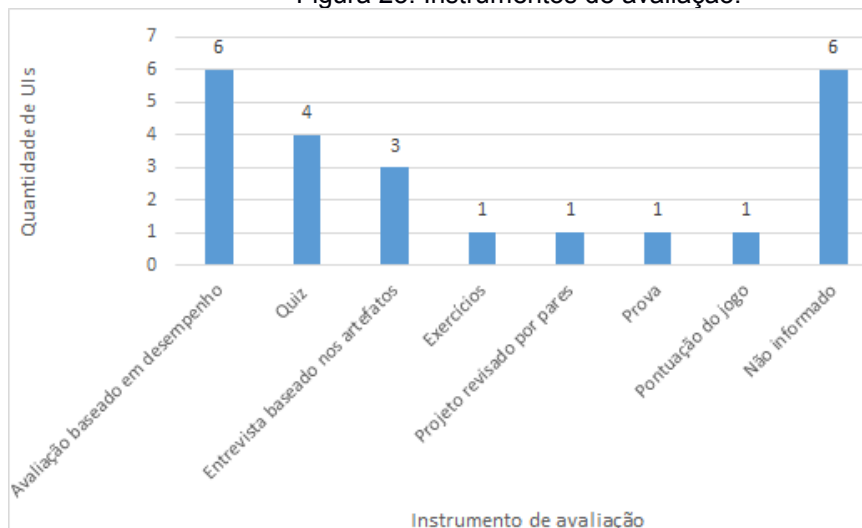
Figura 24: Materiais instrucionais.



Fonte: elaborada pelo autor.

A aprendizagem dos alunos é medida principalmente por meio de avaliações baseadas no desempenho, analisando artefatos criados no contexto do processo de software e/ou programas de software, além de *quizzes* (HERMANS & AIVALOGLOU, 2017; KÖHLER, GLUCHOW & BRUGGE, 2012). Em alguns casos, também se usou entrevistas em relação aos artefatos criados, provas e/ou projetos revisados por pares (Figura 25).

Figura 25: Instrumentos de avaliação.



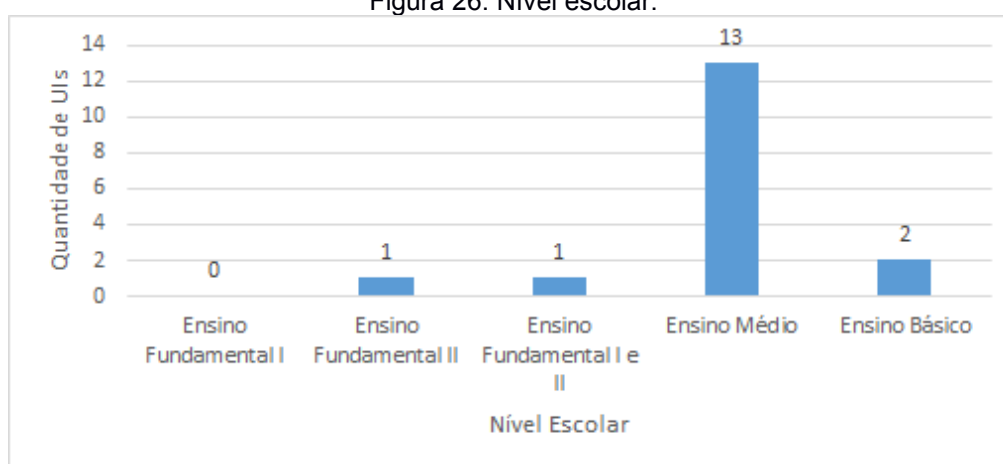
Fonte: elaborada pelo autor.

PA4. Quais são as características de contexto da UI?

A maioria das UIs encontradas abordam o ensino de ES (15 UIs), focadas no Ensino Médio (FRONZA *et al.*, 2015; CORBETT & NESIBA, 2015). Não foi

encontrada nenhuma unidade instrucional de ensino de ES específica para Ensino Fundamental I (FRONZA *et al.*, 2017; HERMANS & AIVALOGLU, 2017). Os resultados positivos relatados indicam então que pode ser benéfico ensinar ES não somente na educação superior, mas também já na Educação Básica (Figura 26). Por outro lado, observando a predominância de UIs encontradas para o Ensino Médio, questiona-se qual a razão para a quase inexistência de UIs para o Ensino Fundamental. Apesar disso, vários autores relatam a importância dos benefícios do ensino de ES já no Ensino Fundamental, levando em consideração o conhecimento prévio dos alunos e as restrições curriculares (BOLLIN & SABITZER, 2015).

Figura 26: Nível escolar.



Fonte: elaborada pelo autor.

Refletindo a grande diversidade do formato das UIs, a duração varia de atividades curtas e focadas, de somente 30 minutos (SARKAR & BELL, 2013), até cursos de longa duração, de um ano (STARRETT, 2007). De acordo com o atual desconhecimento dos alunos no que se refere à computação e/ou ES, a maioria das UIs é direcionada a iniciantes, sem competências prévias de computação/ES. Somente 4 unidades instrucionais indicam a necessidade de competências prévias, principalmente em relação à programação (Quadro 18).

Quadro 18: Visão geral das características de contexto das UIs.

Referência	Nível de ensino	Duração da UI	Competências da computação/ES prévias do aluno
BOLLIN & SABITZER (2015)	Ensino médio	NI	Sem
COLLOFELLO	Ensino Médio	NI	NI

(2002)			
CORBETT & NESIBA (2015)	Ensino Médio	NI	NI
DE KEREKI & MANATAKI (2015)	Educação Básica	15-20 horas	Sem
FRONZA <i>et al.</i> (2017)	Fundamental II	60h (4h por semana) - "Framework"	NI
FRONZA <i>et al.</i> (2016)	Ensino Médio	20-30 horas	Sem
FRONZA <i>et al.</i> (2015)	Ensino médio	5 dias (total de 40 horas)	Sem
HERMANS & AIVALOGLOU (2017)	Fundamental I e II	6 semanas, com total de 12 a 36 horas (estimado um esforço semanal de 2-6 horas)	Sem
KÖHLER; GLUCHOW & BRUGGE (2012)	Ensino médio	3 dias	Sem
MISSIROLI <i>et al.</i> (2017)	Ensino Médio	25 horas	Mínimo de 2 anos de experiência com programação
MISSIROLI <i>et al.</i> (2016)	Ensino médio	Caso 1: 2 horas; Caso 2: 5,5 horas	1 ano de experiência em programação em Java OO/html, <i>web service</i> , bancos de dados
RUSU <i>et al.</i> (2011)	Educação Básica	NI	Com competência prévia de programação.
RUSU <i>et al.</i> (2010)	Ensino médio	NI	Sem
SARKAR & BELL (2013)	Ensino médio	30 minutos	NI
SERRANO & SERRANO (2013)	Ensino médio	10 horas (5 aulas)	NI
STARRETT (2007)	Ensino médio	1 ano	NI
VERHOEFF (2006)	Ensino médio	3 dias	Competência prévia referente à programação (variáveis globais, locais e procedimentos)

Fonte: elaborada pelo autor.

PA5. Como a unidade instrucional foi desenvolvida?

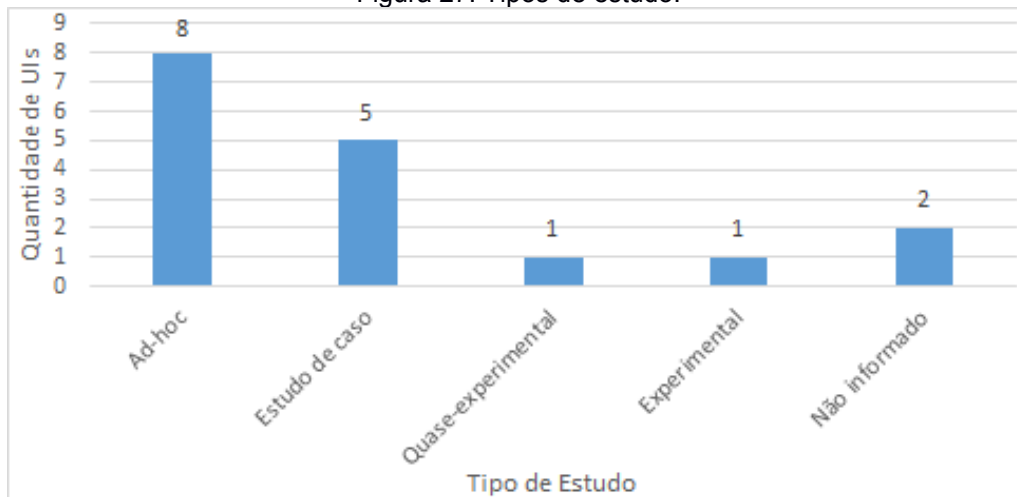
As UIs precisam ser desenvolvidas sistematicamente seguindo modelos de design instrucional para alcançar resultados de aprendizagem eficazes. No entanto,

observamos uma falta generalizada de informações nos artigos encontrados em relação à forma como as UIs foram desenvolvidas. Poucas publicações apresentaram informações em relação a esta questão, tipicamente indicando somente as partes interessadas e envolvidas no desenvolvimento da UI, por meio de cooperações entre escolas e/ou universidades, envolvendo professores, instrutores e tutores (DE KERKEKI & MANATAKI, 2015; KÖHLER, GLUCHOW & BRUGGE, 2012). Apenas uma unidade instrucional (KÖHLER *et al.*, 2012) explicitamente relata os princípios instrucionais utilizados no desenvolvimento da unidade instrucional (cenários baseados em metas) (SCHANK, 1996; SCHANK, 1992; SCHANK *et al.* 1994) e *scaffolding* (VYGOTSKY, 1978)).

PA6. Como a qualidade da unidade instrucional é avaliada?

A avaliação é geralmente feita por meio de estudos empíricos em sala de aula, sendo que várias UIs foram avaliadas por meio de um estudo de caso. Nestes estudos, a avaliação foi sistematicamente definida e, durante e após o tratamento (ensino de ES), foram coletados os dados em relação ao objetivo da avaliação (Figura 27) (MISSIROLI *et al.*, 2016; HERMANS & AIVALOGLU, 2017). Apenas dois estudos adotaram um design de pesquisa mais rigoroso. Uma UI realizou experimento comparando o desempenho e a satisfação dos alunos e professores no uso de duas estratégias de desenvolvimento de software no ensino da computação, o modelo Cascata e o método Scrum (MISSIROLI *et al.*, 2017). A unidade instrucional de Fronza *et al.* (2017) abordaram um quase-experimental para avaliar a eficácia de um *framework* baseado em métodos ágeis de ES para ensinar computação no ensino médio. Estes dois estudos experimentais seguem a metodologia de Wohlin *et al.* (2012). Observa-se também que um número considerável de unidades instrucionais foram avaliadas de uma forma menos rigorosa, por meio de avaliações *ad-hoc*, sem definição detalhada dos objetivos de avaliação, de medição e de análise dos dados (KÖHLER; GLUCHOW & BRUGGE, 2012; FRONZA *et al.*, 2016). Como resultado, esses estudos tipicamente somente comentam o *feedback* informal dos alunos e/ou observações durante a aplicação.

Figura 27: Tipos de estudo.



Fonte: elaborada pelo autor.

A aprendizagem é o fator de qualidade mais avaliado, demonstrando que a principal preocupação é o efeito de aprendizagem proporcionado pelo ensino (Figura 28). A avaliação deste fator geralmente se refere à melhoria da competência, comparando o nível de competência dos alunos após o ensino com seu nível de competência antes do ensino, geralmente baseado em uma pontuação pós-teste. Nenhum estudo avalia o efeito de aprendizagem em relação à uma definição sistemática de níveis de aprendizagem, por exemplo, com base na taxonomia de Bloom. Vários estudos também avaliam o grau de satisfação (MISSIROLI *et al.*, 2016), para determinar se os alunos sentem que o esforço dedicado resulta em aprendizado. Analisando os fatores avaliados, identificou-se também a falta de uma conformidade entre os estudos. Com exceção da avaliação do grau da aprendizagem do aluno, parâmetro avaliado na maioria dos estudos, os fatores analisados variam muito entre estudos, indicando a falta de um modelo de avaliação para este tipo de UI.

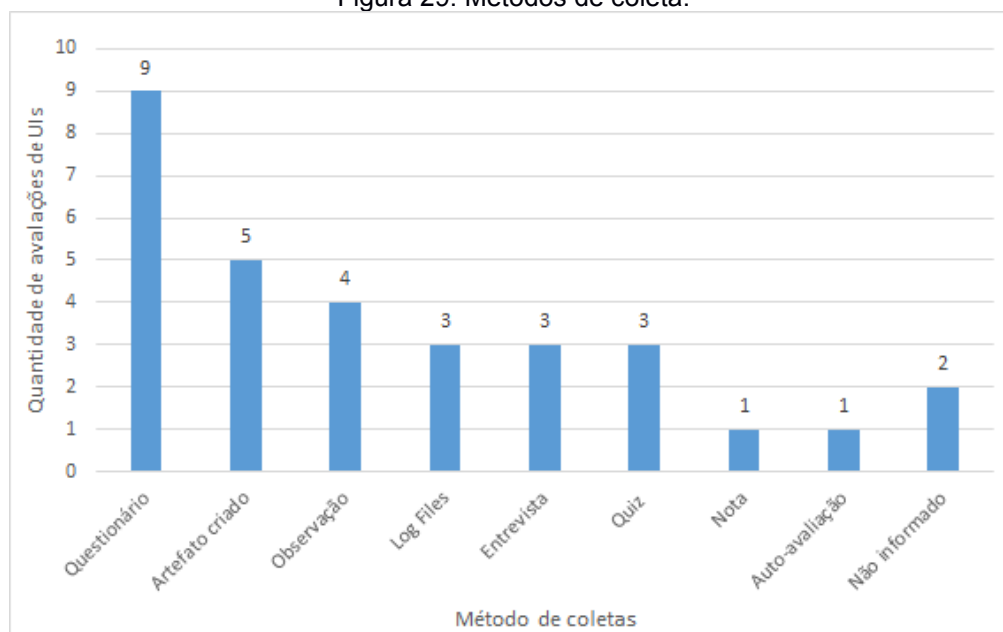
Figura 28: Fatores de qualidade.



Fonte: elaborada pelo autor.

A maioria dos dados é coletada via questionários (Figura 29) (FRONZA *et al.*, 2017). Vários estudos também extraem dados de avaliação baseados no desempenho dos artefatos criados pelos alunos durante a unidade instrucional e/ou via observações durante as UI (KÖHLER; GLUCHOW & BRUGGE, 2012; MISSIROLI *et al.*, 2016).

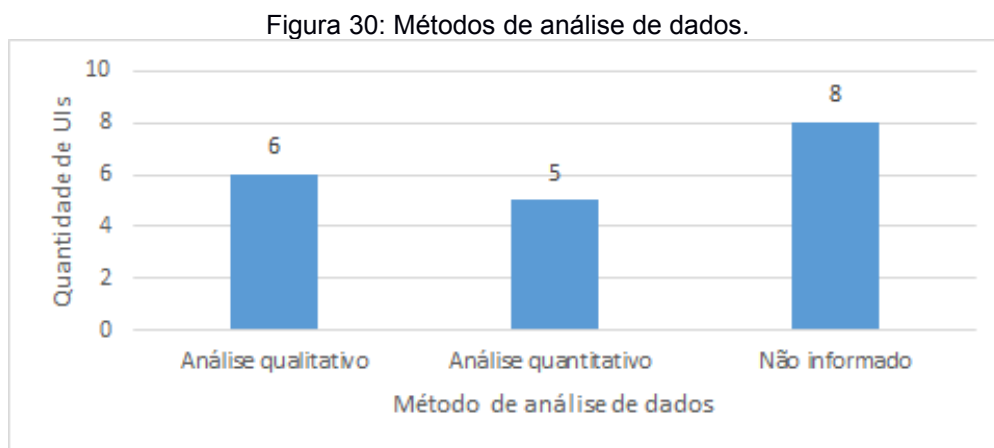
Figura 29: Métodos de coleta.



Fonte: elaborada pelo autor.

Para analisar fatores de desempenho, diversão e satisfação do aluno foi feita a observação. Os *log files* de fóruns, vídeos, *wikis* e registros de experiências adquiridas foram na maioria utilizados por cursos realizados de modo *on-line* para analisar o engajamento do aluno no curso.

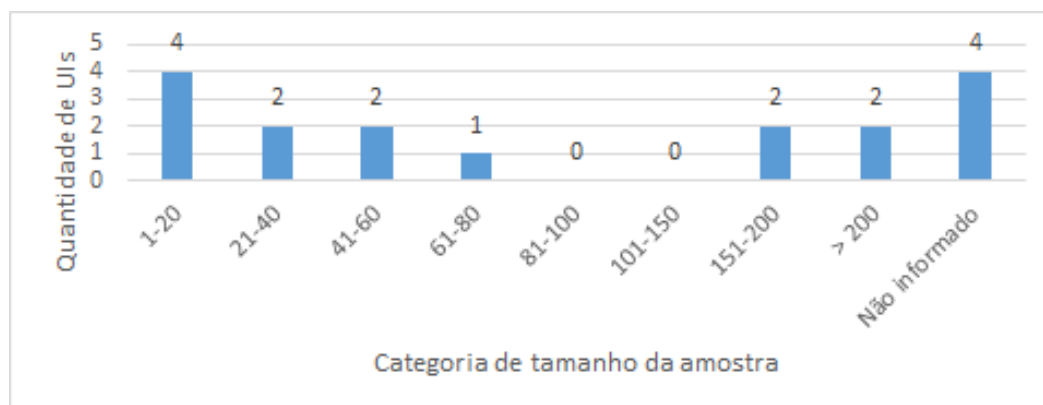
De acordo com os designs de pesquisa menos rigorosos adotados, a maioria dos estudos realiza somente análises qualitativas dos dados e/ou análises quantitativas de forma descritiva (Figura 30) (MISSIROLI *et al.*, 2016; RUSU *et al.*, 2010). Apenas dois estudos realizaram testes de estatísticos (HERMANS & AIVALOGLOU, 2017; MISSIROLI *et al.*, 2017).



Fonte: elaborada pelo autor.

Como mostra a figura 31, a maioria das avaliações foi realizada com amostras pequenas, variando de 1 a 60 participantes (SARKAR & BELL, 2013; SERRANO & SERRANO, 2013). Esse baixo número de participantes normalmente corresponde ao tamanho de uma classe, na qual a unidade instrucional é aplicada e avaliada. No entanto, vale ressaltar que também encontramos quatro avaliações realizadas com mais de 150 participantes. O tamanho da amostra não foi informada por uma significativa quantidade de estudos (4 UIs).

Figura 31: Categoria de tamanho de amostra.



Fonte: elaborada pelo autor.

Observa-se que quase a metade dos estudos foi replicada em mais do que um contexto, contribuindo para a validade externa dos resultados da avaliação. Contudo, uma grande parte das replicações ocorreu apenas por meio de um único estudo e em um contexto específico pelos próprios criadores da UI (STARRETT, 2007; SARKAR & BELL, 2013).

3.1.3. Discussão

Observou-se que, levando em consideração a importância da ES no desenvolvimento de softwares, foi encontrado um número muito pequeno de UIs (somente 17 no total) visando o ensino destas competências importantes no nível da Educação Básica.

Em termos de áreas de conhecimento de ES, as UIs focam nas fases principais do processo de software, incluindo a análise de requisitos, modelagem, construção e teste de software. Algumas UIs também abordam explicitamente a manutenção de software. Uma parte das UIs encontradas segue um modelo de ciclo de vida tradicional, como cascata ou modelo V. Por outro lado, uma grande parte adota metodologias ágeis, seguindo um processo iterativo e criando artefatos como *User stories*, *storyboards* etc. Assim, infere-se que tanto a adoção de modelos de ciclo de vida simples e/ou interativos pode ser benéfica para introduzir o processo de software nesse nível escolar.

Observou-se ainda que a maioria dos UIs se concentraram no Ensino Médio, mesmo que alguns autores relatem também benefícios observados na introdução do ensino de ES no Ensino Fundamental (BOLLIN & SABITZER, 2015). Apesar disso,

nenhum dos autores relata dificuldades ou baixo desempenho dos alunos em aprender conceitos de ES. Além disso, Bollin & Sabitzer (2015), após aplicarem uma unidade instrucional no Ensino Médio, concluem que o ensino de ES pode ser iniciado no Ensino Fundamental sem dificuldades. Porém, é preciso identificar quais áreas de conhecimentos, assuntos e o grau que deve ser ensinado, conforme a idade do aluno (BOLLIN & SABITZER, 2015). Outra descoberta conclui que não há diferença de desempenho em relação à aprendizagem de ES e programação para alunos da Educação Básica (HERMANS & AIVALOGLU, 2017).

Notamos também que as UIs geralmente ensinam competências de programação juntamente com a ES. Bollin & Sabitzer (2015) concluíram que não existe a necessidade de os alunos terem experiência prévia em computação, e que a ES pode ser ensinada com fundamentos básicos de lógica de programação e modelagem (diagramas de fluxo), com exceção da unidade instrucional apresentada por Starrett (2007), que ensina alunos a modelar software utilizando a linguagem UML antes de ensinar programação. Conforme o autor, a modelagem e as abstrações são fundamentais para o pensamento analítico. Ele destaca também que a modelagem fornece um método para ajudar os alunos a abordar problemas e soluções passo-a-passo. Por fim, o resultado demonstra que os alunos aprenderam os conceitos centrais da abstração de maneira rápida e natural.

Em geral, são abordados conceitos mais básicos de ES nesse nível escolar, mais relacionados ao domínio cognitivo, variando também em relação à duração da UI. Deficiências tipicamente observadas em relação ao ensino de ES na educação superior são restritos a projetos em pequena escala, não apresentando características de projetos reais, e estão ainda mais presentes nas UIs encontradas no nível escolar (MALIK *et al.*, 2012). De forma geral, as UIs na Educação Básica também têm uma ênfase no conhecimento básico de ES, não abordando de forma mais abrangente experiências e habilidades práticas.

Além disto, os resultados demonstram também uma preferência na adoção de métodos e técnicas ágeis, que parecem mais adequadas para iniciar o ensino de ES. De forma interessante, apenas sete UIs relatam o uso de ferramentas CASE no ensino, já que o uso desse tipo de ferramenta ajuda a executar as atividades dos processos com maior qualidade (IEEE CS, 2014). Desta forma, surge a questão se o ensino de ferramentas CASE está inapropriado para esta faixa etária e/ou se é

causado pela falta deste tipo de funcionalidade nos ambientes de programação tipicamente usados na Educação Básica.

As UIs abordam o ensino de várias fases do processo ou focam somente em uma determinada área, predefinindo os artefatos de entrada para esta fase. Isso pode representar uma alternativa do ensino, principalmente quando há restrições de tempo à UI. A grande maioria está inserida no contexto de UIs centradas no ensino de programação, e poucos focam explicitamente no ensino de conceitos de ES. A integração do ensino de ES em UIs, ensinando programação, pode ser benéfica tanto em relação às restrições de tempo quanto na aprendizagem de uma compreensão de escopo mais amplo e variada da área de computação. Para viabilizar a adoção do ensino de computação/ES, várias UIs são realizadas de forma multidisciplinar, integradas em outras disciplinas, como Física ou Artes.

A maioria das UIs visa a aprendizagem dos conceitos de ES no nível de aplicação, adotando abordagens de aprendizagem ativa. Tipicamente, os alunos, depois de uma parte introdutória, desenvolvem um software (animação, aplicativo móvel, aplicação *web* ou robôs). A avaliação do grau de aprendizagem dos alunos é comumente baseada no desempenho a partir dos artefatos criados pelos alunos e/ou *quizzes*. Entretanto, os artigos encontrados não fornecem maiores informações em relação à avaliação da aprendizagem das competências de ES, seja, por exemplo, por meio de rubricas, análises automatizadas etc.

Visando uma disseminação das UIs apresentadas nos artigos, têm-se uma indisponibilidade de informações detalhadas das UIs e/ou dos materiais instrucionais. A grande maioria das UIs foi criada em somente uma única linguagem e não está acessível, seja gratuitamente ou paga. Essa indisponibilidade das UIs impede a ampliação da sua aplicação.

Relacionado a essa questão também há uma falta em relação à capacitação de instrutores para treiná-los para a aplicação das UIs em sala de aula. Levando em consideração que hoje há uma grande falta de professores na Educação Básica com formação na computação, há como solução somente a adoção de uma abordagem multidisciplinar, em que a computação é ensinada por professores formados em outras disciplinas. Portanto, a motivação e a formação de professores em serviço tornam-se cruciais, uma vez que precisam estes ter conhecimentos de computação, ES e tecnologia, bem como conhecimento de conteúdo pedagógico (GAL-EZER & STEPHENSON, 2010; BOLLIN & SABITZER, 2015).

O que também chama atenção é o fato de que muitos artigos não apresentam informações essenciais em relação ao(s) objetivo(s) de aprendizagem e/ou estratégia instrucional, nem indicam os métodos utilizados para sistematicamente desenvolver as UIs. Este ponto fraco pode ser também observado em relação à avaliação da maioria das UIs. Vários artigos não relatam avaliações ou os realizaram somente de forma *ad-hoc*, o que torna os resultados relatados questionáveis. A grande variação dos fatores avaliados, em diversas formas, também indica a falta de modelos de avaliação nesta área para facilitar de forma mais uniforme uma avaliação dessas UIs. Esses fatores visam avaliar o quanto os alunos aprenderam sobre a computação em geral, e não sobre competências de ES.

3.1.4. Ameaças à validade

Como em qualquer revisão sistemática, existem algumas ameaças à validade dos resultados. Portanto, foram identificadas ameaças potenciais e aplicadas estratégias de mitigação para minimizar seu impacto:

-Viés de publicação: Mapeamentos sistemáticos podem sofrer do viés comum de que os resultados positivos têm maior probabilidade de serem publicados do que os negativos. No entanto, foi considerado que os resultados dos artigos, sejam positivos ou negativos, têm apenas uma pequena influência sobre esse mapeamento sistemático, uma vez que UIs foram caracterizadas, em vez de analisar seus impactos sobre a aprendizagem;

-Identificação de estudos: Outro risco é a omissão de estudos relevantes. A fim de mitigar esse risco, foi construído cuidadosamente as palavras de busca para serem o mais abrangentes possível, considerando não apenas os principais conceitos, mas também sinônimos. Devido ao risco de excluir UIs existentes, as quais ainda não foram relatadas por meio de artigos científicos, foi realizada uma busca de forma mais abrangente no Google. Além disso, foram analisadas as próprias referências da literatura primária encontrada e também artigos relevantes citados como literatura secundária;

-Seleção e extração de dados de estudos: Ameaças para estudar seleção e extração de dados foram mitigadas por meio do fornecimento de uma definição detalhada dos critérios de inclusão/exclusão e de qualidade. Foi definido documentado um protocolo rígido para a seleção dos estudos, e todos os autores realizaram a seleção juntos, discutindo a seleção até que o consenso fosse

alcançado. A extração de dados foi prejudicada em alguns casos, uma vez que as informações relevantes nem sempre foram apresentadas explicitamente e/ou usando uma terminologia comumente aceita e, portanto, em alguns casos tiveram que ser inferidas. No entanto, essa inferência foi feita pelo primeiro autor e cuidadosamente revisada pelos coautores.

3.2. ESTADO DA ARTE DO ENSINO DE EU NA EDUCAÇÃO BÁSICA

Este mapeamento foi realizado por meio da produção de um relatório técnico, executado em cooperação no laboratório de pesquisa (FERREIRA, M. N. F. *et al.*, 2018). O objetivo deste MSL é levantar quais UIs existentes (e quais as suas características) que ensinam competências de EU de usuário no contexto de ensino de computação na Educação Básica?

Esta questão de pesquisa é refinada nas seguintes perguntas de análise:

PA1. Quais UIs existem?

PA2. Qual(is) competências de EU são ensinadas na UI?

PA3. Quais as características instrucionais da UI?

PA4. Quais são as características de contexto da UI?

PA5. Como a unidade instrucional foi desenvolvida?

PA6. Como a qualidade da unidade instrucional é avaliada?

-Critérios de inclusão/exclusão: São incluídos apenas artigos revisados por pares, cujo foco é ensinar computação adotando práticas e/ou atividades de design de interface de usuário na Educação Básica, incluindo também o UX e/ou *Design Thinking*, se voltado ao desenvolvimento de softwares/aplicativos. Foram excluídos artigos que focam em robótica, no ensino superior e/ou artigos que apresentam UIs para o ensino de computação sem abordar conceitos de design de interface de usuário. Incluímos também literatura secundária, descoberta por meio das referências da literatura primária encontrada (VERHOEFF, 2006);

-Critério de Qualidade: Foi considerado apenas artigos que apresentam informações substanciais em relação ao ensino de conceitos de design de interface do usuário, indicando, por exemplo, conteúdo das aulas, materiais didáticos etc.;

-*Fonte de dados:* A busca foi feita no Scopus⁹, Science Direct¹⁰, Google Scholar¹¹ e ERIC¹², representando as principais editoras científicas, como também em sites para o ensino online (MOOCs), incluindo Udemy¹³, Edx¹⁴, Khanacademy¹⁵ e Coursera¹⁶. Com o intuito de abranger uma maior gama de publicações, foram realizadas buscas no Google Scholar, que indexa um grande conjunto de dados de diversas fontes de produção científica (HADDAWAY *et al.*, 2015). A busca no Google foi realizada a fim de ampliar e atingir cursos, estudos e currículos não encontrados em outra base de dados;

-*Definição da string de busca:* A *string* de busca foi composta de conceitos relacionados à questão de pesquisa, considerando também sinônimos, conforme indicado no Quadro 19.

Quadro 19: Palavras-chave.

Conceito principal (termo)	Sinônimos
<i>user interface design</i>	<i>graphic design, visual design, user experience, UX, Design Thinking</i>
<i>computing</i>	<i>coding, programming</i>
<i>K-12</i>	<i>school, kids, children</i>

Fonte: próprio autor.

O termo *user interface design* foi escolhido por ser o principal conceito a ser pesquisado. Com o objetivo de identificar estudos que foquem em UIs de programação, foi utilizado o termo *computing*. O termo *K-12* é utilizado para restringir o nível de ensino focado pela UI. A partir dessas palavras-chave foi calibrada a *string* de busca e adaptada de acordo com a sintaxe específica da fonte de dados conforme apresentado no Quadro 20.

Quadro 20: *Strings* de busca.

Fonte de Dados	<i>String</i> de busca
Scopus	<i>("user interface design" OR "graphic design" OR "visual design" OR "user experience" OR UX OR "design thinking") AND (computing OR coding OR programming) AND ("K-12" OR school OR kids OR children)</i>
Science Direct	<i>("user interface design" OR "graphic design" OR "visual design" OR "user experience" OR UX OR "design thinking") AND (computing OR coding OR programming) AND ("K-</i>

⁹ www.scopus.com

¹⁰ <https://www.sciencedirect.com>

¹¹ <https://scholar.google.com.br>

¹² <https://eric.ed.gov/>

¹³ www.udemy.com

¹⁴ www.edx.org

¹⁵ www.khanacademy.org

¹⁶ www.coursera.org

	<i>12" OR school OR kids OR children)</i>
ERIC	<i>("user interface design" OR "graphic design" OR "visual design" OR "user experience" OR "UX" OR "design thinking") AND ("coding" OR "programming") AND ("K-12" OR "school" OR "kids" OR "children")</i>
Google scholar	<i>("design thinking" OR "user interface design") AND ("programming" OR "coding" OR "apps") AND ("K-12" OR "school" OR "kids" OR "children")</i>
Google	<i>"design thinking" (computing OR coding OR programming) (kids OR K-12 OR school)</i>
	<i>"user experience" design (computing OR coding OR programming) (kids OR K-12 OR school)</i>
	<i>"user interface design" (computing OR coding OR programming) (kids OR K-12 OR school)</i>

Fonte: dados do autor.

3.2.1. Execução da busca

A busca foi executada em maio de 2018 pelos autores em duas etapas. Na primeira etapa, a busca foi feita via Scopus, Science Direct, ERIC, Google Scholar e Google (Tabela 3). Diversos resultados encontrados foram excluídos por não estarem no foco da pergunta de pesquisa, como, por exemplo, artigos que apenas relatam a importância do ensino de design de interface do usuário na Educação Básica (MALIK *et al.*, 2012; BOLLIN *et al.*, 2016), ou por não apresentarem detalhes sobre o ensino de design de interface do usuário (AZENKOT *et al.*, 2011). As buscas foram realizadas nas plataformas Google e Google Scholar e 89 artigos potencialmente relevantes foram selecionados, de acordo com os critérios de inclusão e exclusão, analisando rapidamente o título, o resumo e as palavras-chave. Na segunda etapa de seleção, analisamos o texto completo dos artigos pré-selecionados, para verificar sua conformidade com os critérios de inclusão/exclusão e o critério de qualidade. No final, foram identificados 16 artigos relevantes. Todo este processo foi feito pelos pesquisadores em conjunto, sempre discutindo a seleção, até chegar a um consenso.

Tabela 3: Quantidade de artigos por etapa de seleção por repositório.

Fonte de dados	Resultado da pesquisa inicial	Resultados analisados	Resultado da seleção depois do 1º estágio	Resultado da seleção depois do 2º estágio
Scopus	160	160	13	4
Science Direct	83	83	1	0
ERIC	57	57	3	0
Google Scholar	17.800	300	22	7
Google	<i>Design Thinking: 1.160.000 UX: 8.770.000 Interface design:</i>	<i>Design Thinking: 200 UX: 150 Interface</i>	<i>Design Thinking: 25 UX: 6 Interface design: 19</i>	5

	7.150.000	<i>design:200</i>		
Total				16

Fonte: dados do autor;

Numa segunda etapa da busca, foram também procuradas explicitamente informações sobre cursos de ensino *on-line*. Foram analisados os cursos disponíveis, adotando os mesmos critérios de inclusão/exclusão. Como resultado desta pesquisa adicional, foi encontrada uma unidade instrucional relevante (COLLOFELLO, 2002). Outros cursos encontrados foram desconsiderados, por não abrangerem conceitos de design de interface de forma explícita (MECCAWEY, 2017; SHEEHAN, 2000) e/ou por não disponibilizarem detalhes da(s) UI(s) (GUPTA, TEJOVANTH & MURTHY, 2012). Outros foram desconsiderados por serem voltados à formação de professores da Educação Básica, e não diretamente dos alunos. Excluimos também várias UIs que abordam o ensino de design no contexto de robótica e/ou computação física (FRANCIS *et al.*, 2017; BEKKER *et al.*, 2015). Foram encontradas diversas UIs voltadas ao ensino de *Design Thinking* para crianças, como por exemplo, CityX project¹⁷, ou o K-12 lab da Stanford dschool¹⁸. Porém, por abordar o ensino de design fora do contexto do ensino de computação, foram excluídos desse mapeamento.

3.2.2. Análise de dados

Para responder a questão de pesquisa, extraímos informações relevantes às perguntas de análise. A extração dos dados foi dificultada em virtude da forma como os estudos foram relatados. Como as publicações nesta área não seguem nenhum protocolo estruturado, os artigos não descrevem necessariamente as informações a serem extraídas de maneira explícita. A maioria dos trabalhos carece de detalhes sobre a UI. Nestes casos, algumas informações foram inferidas a partir do artigo, como, por exemplo, a descrição dos objetivos de aprendizagem, idioma e pré-requisitos. Nos casos em que o artigo não apresenta nenhuma informação a ser extraída, indicamos a falta desta informação como não informado (NI). As informações que foram extraídas dos artigos são detalhadas em Ferreira *et al.* (2018).

¹⁷ <http://www.cityxproject.com>

¹⁸ <https://dschool.stanford.edu/programs/k12-lab-network>

PA1. Quais UIs existem?

Como resultado da pesquisa, foram identificadas 16 UIs voltadas ao ensino de computação, que de alguma forma abordam também o ensino de design de interface do usuário no nível da Educação Básica (Quadro 21).

Quadro 21: Artigos relevantes.

Referência	Título
CHEN & HUANG (2017)	<i>Design Thinking in App Inventor game design and development: A case study</i>
SULLIVAN, REAMON & LOUIE (2003)	<i>Girls embrace technology: A summer internship for high school girls</i>
KE & IM (2014)	<i>A case study on collective cognition and operation in team-based computer game design by middle-school children</i>
ROBINSON & PÉREZ-QUIÑONES (2014)	<i>Underrepresented middle school girls: on the path to computer science through paper prototyping</i>
VAN WART <i>et al.</i> (2014)	<i>Apps for social justice: Motivating computer science learning with design and real-world problem solving</i>
DENNER <i>et al.</i> (2005)	<i>The girls creating games program: Strategies for engaging middle-school girls in information technology</i>
EDUTOPIA (2015)	<i>Coding by design first approach</i>
CODELIKEAGIRL (2017)	<i>Learning design by making games in scratch</i>
TEKKIE UNI (2018)	<i>Build Your First App</i>
CREATELAB (2017)	<i>My createlab</i>
CODE (2018)	<i>Curriculum code</i>
ROBINSON, PÉREZ-QUINONES & SCALES (2015)	<i>Understanding the attitudes of African American middle school girls toward computer science.</i>
TECHNOVATION (2018)	<i>Technovation Challenge</i>
CODEHS (2018)	<i>Codehs</i>
CODE.ORG/APP LAB (2018)	<i>Applab</i>
GET STARTED WITH CODE 2 (2017)	<i>Get started with code 2</i>

O primeiro registro de uma unidade instrucional que abrange o ensino de *design* de interface foi encontrado em 2004. Embora ainda nos dias atuais não haja uma variedade significativa de UIs nesse âmbito, é possível perceber que houve um crescimento significativo nos últimos anos, provavelmente relacionado à tendência

do aumento de ensino de computação de forma geral na Educação Básica (Figura 32).

Figura 32: Quantidade de UIs publicado por ano.

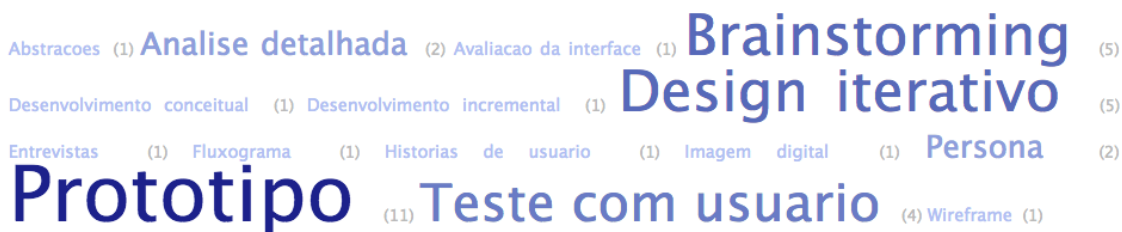


Fonte: elaborada pelo autor.

PA2. Qual(is) competências de design de interface são ensinadas nas UIs?

Nas UIs encontradas, nota-se que são abordadas competências de design de interface incluindo *design thinking*, design de interface do usuário, design visual, design centrado no usuário, usabilidade, interação humano-computador e UX. A técnica de prototipação foi a mais utilizada para o ensino de design de interface de usuário. Os alunos aprenderam a criar protótipos do software utilizando papel e/ou mídias digitais. A técnica de brainstorming também foi amplamente ensinada com o objetivo de fazer os alunos se interagir em grupo e estimular pensamentos e experiências para gerar soluções inovadoras. As UIs também focaram e realizar testes centrados no usuário para avaliar os requisitos de usabilidade juntamente com usuários. Algumas UIs, além de visar a aprendizagem da lembrança e compreensão desses conceitos, também visam a aprendizagem de diversas técnicas, permitindo ao aluno entender, explorar e materializar, buscando ideias sobre a melhor forma de resolver um problema, viabilizando assim um nível maior de aprendizagem (Figura 33).

Figura 33: Frequência das técnicas utilizadas para ensinar design de interface.



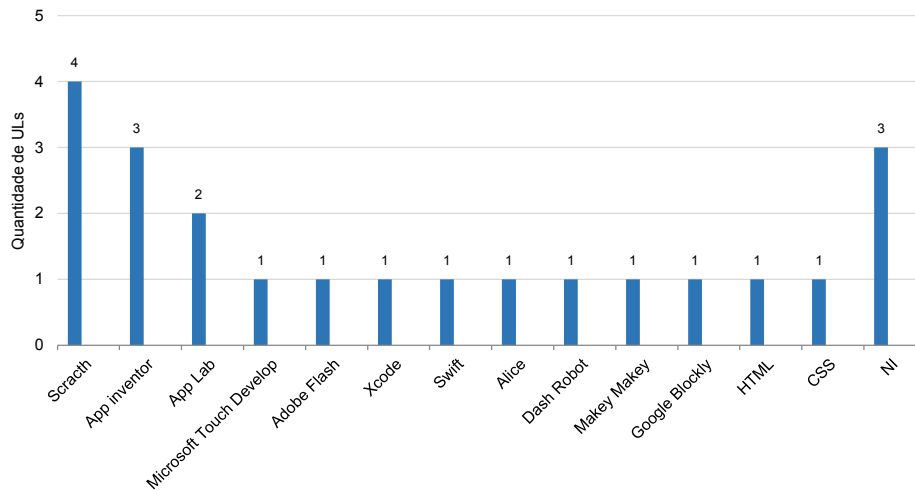
Fonte: elaborada pelo autor.

As UIs fazem uso de ferramentas específicas para o design da interface. No que se refere à produção de imagens, foram utilizados Adobe Photoshop Macromedia Director (SULLIVAN, REAMON & LOUIE, 2003), Pages e Mini Monet. Para o design de interface e a produção de mapas de navegação/workflows, algumas UIs utilizaram Lingo, POP *app*, Invision *app*, editor on-line do CodeHS e Balsamiq.

PA3. Quais as características instrucionais das UIs?

De forma geral, no contexto do ensino de desenvolvimento de jogos, animações ou aplicativos, as UIs ensinam competências do processo de *Design Thinking* e práticas de design de interface visual. Para o desenvolvimento desses softwares, geralmente são adotados ambientes de programação visuais baseados em blocos, como, por exemplo, o Scratch para jogos e animações (KE & IM, 2014; CODELIKEAGIRL, 2017), Alice (TEKKIE UNI, 2018) ou Google Blockly (CREATELAB, 2017) e para o desenvolvimento de *apps* App Inventor (CHEN E HUANG, 2017; VAN WART *et al.*, 2014; TECHNOVATION, 2018) ou App Lab (CODE, 2018; CODE.ORG/APP LAB, 2018). Observou-se também a adoção de linguagens de programação baseado em textos, como as linguagens de marcação HTML e CSS, que definem a parte visual de um website. Outras ferramentas foram utilizadas para o ensino de design de interface do usuário, como a criação de animações e/ou jogos com o Adobe Flash (Figura 34).

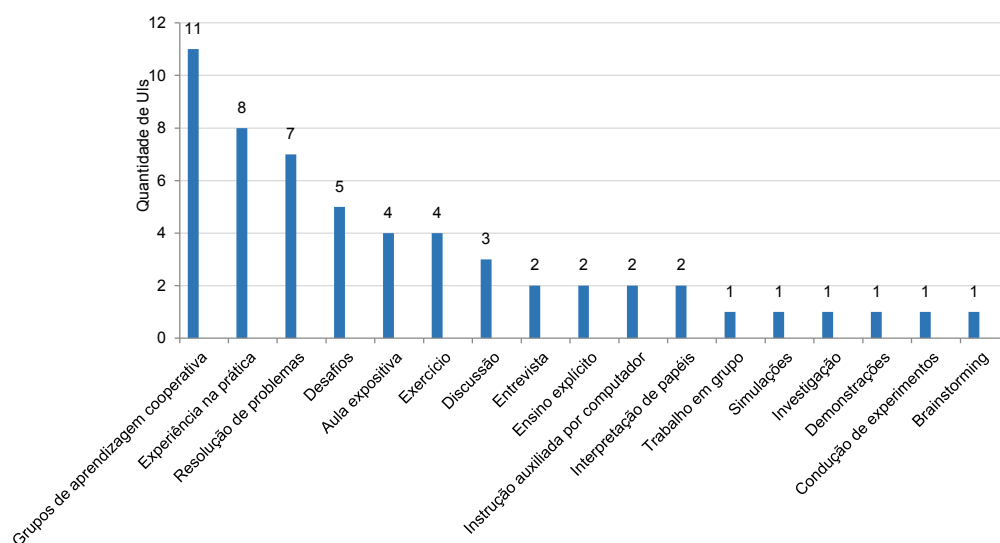
Figura 34: Ambientes de programação utilizados nas UIs.



Fonte: elaborada pelo autor.

Em termos de métodos instrucionais, observou-se uma predominância de abordagens voltadas à aprendizagem ativa por meio de métodos de ensino em grupos e cooperativos (Figura 35). De acordo com a taxonomia do Bloom (1973), estas abordagens estão relacionadas aos objetivos de aprendizagem, que visam a aprendizagem ao nível de aplicação. O ensino por meio de resolução de problemas também foi utilizado como parte do ensino de *design thinking*, tendo o objetivo de incentivar o pensamento criativo dos alunos. Como a idealização e resolução criativa de um problema faz parte do processo de *design thinking*, a maioria desses trabalhos não teve um problema e uma solução previamente definida, estimulando os alunos a identificar o problema e a imaginar a solução. Mesmo focando mais na aprendizagem ativa, diversas UIs também incluem outros métodos de instrução direta, como aulas expositivas, vídeo aula, demonstrações, ensino explícito, principalmente na parte inicial da UI. Alguns métodos instrucionais indiretos são investigação e exercícios em sala de aula. Entre os métodos interativos, os mais utilizados foram: *brainstorming*, desafios, entrevistas, trabalhos em grupo e discussões. O ensino por meio de métodos de aprendizagem experimental também foi aplicado através de simulações, interpretação de papéis e condução de experimentos.

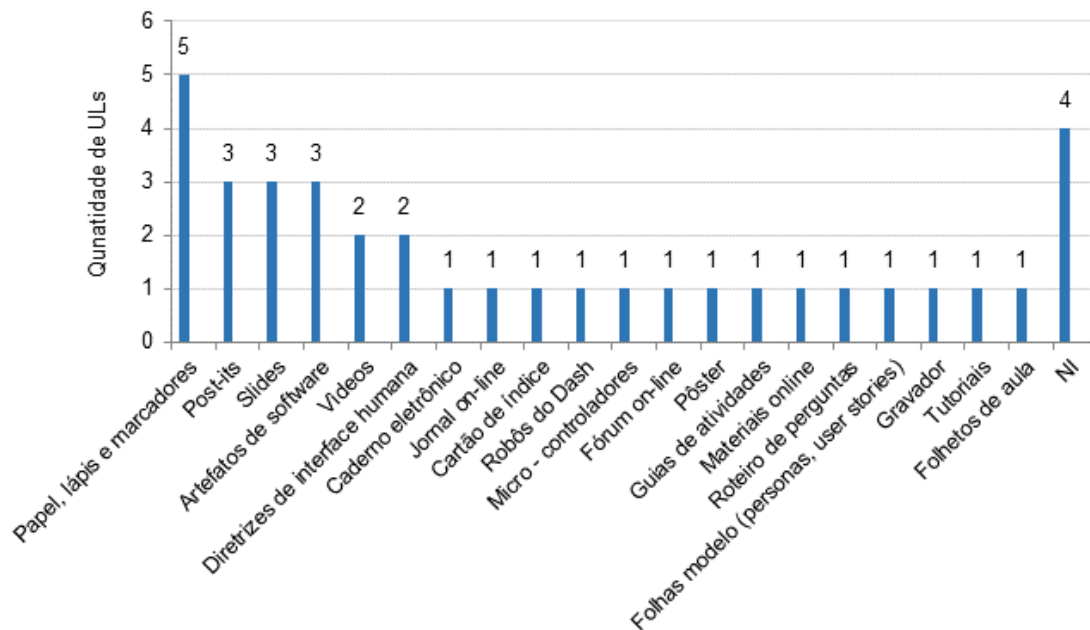
Figura 35: Métodos instrucionais adotados.



Fonte: elaborada pelo autor.

Em virtude da variação dos métodos instrucionais, são também utilizados diferentes tipos de materiais instrucionais. Os materiais mais utilizados para construir protótipos de papel, ou *wireframes* são papéis, lápis, marcadores etc. (Figura 36). Também são utilizados post-its para diversos fins, como definir fluxogramas, registrar as ideias e requisitos de software. Artefatos do processo de software são utilizados para auxiliar no desenvolvimento do software, como, por exemplo, amostras de código e user stories. Foram também utilizados outros materiais instrucionais, como slides, vídeos, entre outros. No entanto, de forma geral, observou-se a apresentação de poucas informações em relação aos materiais nas publicações, suas disponibilidades e licenças de uso, o que dificultam o uso por outros interessados. A maioria dos materiais também está disponível em somente uma única língua (predominantemente em Inglês), o que pode limitar também uma adoção mais ampla das UIs em outros países que tipicamente necessitam do material instrucional na língua nativa nesse nível escolar.

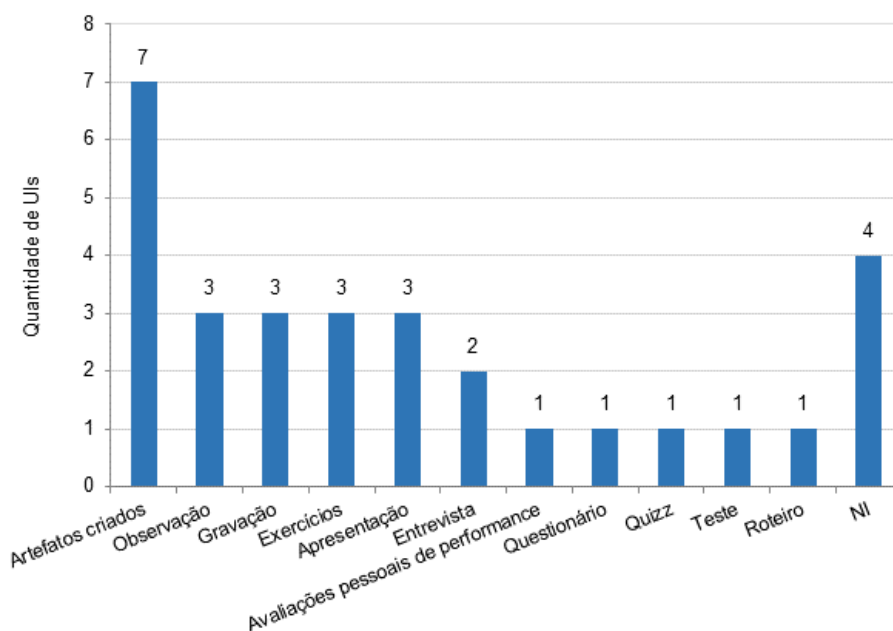
Figura 36: Materiais instrucionais utilizadas.



Fonte: elaborada pelo autor.

A aprendizagem dos alunos é medida principalmente por meio de avaliações baseadas no desempenho, analisando artefatos criados no contexto do ensino de design de interface (Figura 37). Essas avaliações contemplam a interação da interface de um software funcional, como jogos, websites ou *apps*. Observou-se também que foram aplicados exercícios durante as Uls com o intuito de fazer o aluno empregar os conceitos ensinados, como por exemplo, avaliar o design de objetos, projetar um produto por meio de protótipos em papel e simular testes de usabilidade.

Figura 37: Métodos de avaliação.

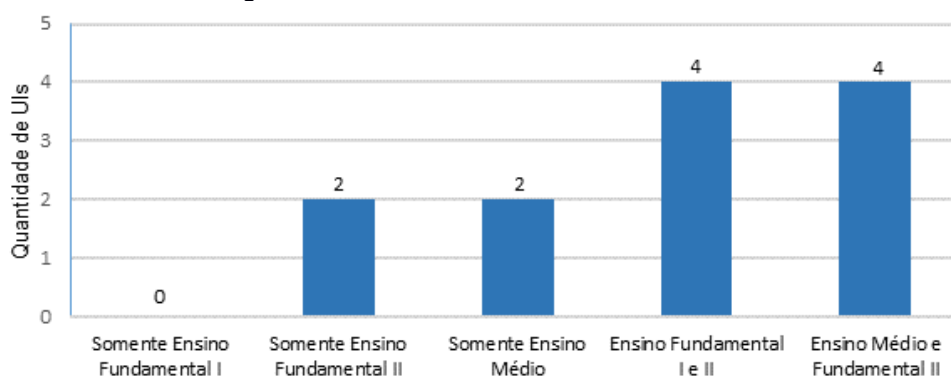


Fonte: elaborada pelo autor.

PA4. Quais são as características de contexto da UI?

De modo geral, a maioria das UIs encontradas não requisita conhecimento prévio, podendo ser ensinadas tanto para crianças do nível de Ensino Fundamental I e II quanto para jovens do Ensino Médio (Figura 38). Essa característica demonstra que é possível inserir conceitos de design de interface já no Ensino Fundamental.

Figura 38: Nível de ensino das UIs encontradas.



Fonte: elaborada pelo autor.

No que diz respeito ao período das UIs, não existe um padrão. Foram encontradas tanto oficinas de poucas horas, quanto cursos longos de 1 ano. A maioria das UIs analisadas são de curta duração, variando de 4 a 18 horas/aula.

PA5. Como as Uis foram desenvolvidas?

A maioria dos artigos não apresenta informações em relação à forma como as Uis foram desenvolvidas. Os registros, neste contexto, geralmente restringem-se à indicação das partes envolvidas, por meio de cooperações entre escolas e/ou universidades, envolvendo professores, instrutores e tutores.

PA6. Como a qualidade da unidade instrucional é avaliada?

A avaliação das UIs é tipicamente realizada por meio de estudos empíricos em sala de aula, como parte do processo sistemático de desenvolvimento e melhoria de uma UI. Várias UIs foram avaliadas por meio de um estudo de caso. Apenas um estudo adotou um design de pesquisa mais rigoroso (SULLIVAN *et al.*, 2003). Este estudo enfocou em desenvolver atitudes positivas em relação à criação e ao uso de tecnologia, habilidades técnicas e maior conscientização sobre oportunidades da carreira de TI. Duas UIs foram avaliadas de forma *ad-hoc*, sem definição detalhada dos objetivos de avaliação, medição e análise dos dados. Como resultado, esses estudos comentam somente o *feedback* informal dos alunos e/ou observações feitas durante a aplicação.

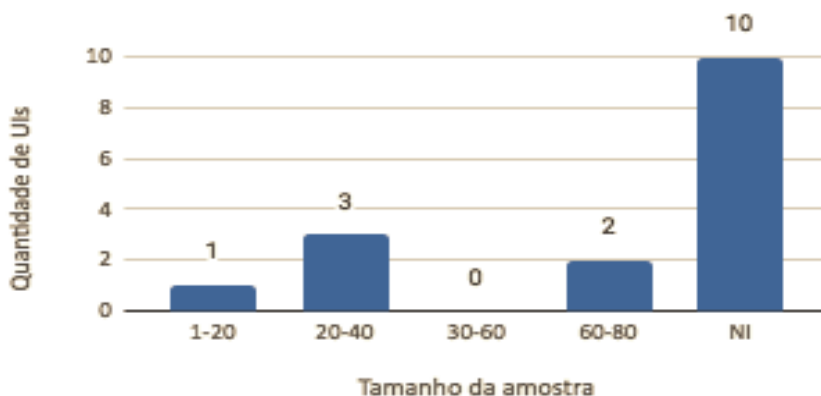
As avaliações relatadas focam em demonstrar que os alunos conseguem aprender conceitos relacionados ao design de interface além de vários outros fatores, como utilidade, diversão, motivação, satisfação, entre outros. A diversidade de indicadores dos fatores de qualidade reflete a falta de um modelo de avaliação para este tipo de UI.

Quanto às coletas de dados, os métodos mais utilizados foram entrevistas e questionários. O método da observação, além de permitir analisar o desempenho do aluno, também propicia a avaliação de fatores como a diversão e a satisfação. Os recursos de vídeos, *log files* de fóruns e registros de experiências adquiridas foram utilizados em cursos realizados em modo *on-line* para analisar o engajamento do aluno no curso.

Salienta-se que a maioria das avaliações foi realizada com amostras pequenas, variando de 1 a 60 participantes (Figura 39). Esse baixo número de participantes normalmente corresponde ao tamanho de uma turma, na qual a unidade instrucional é aplicada e avaliada. No entanto, vale ressaltar que uma significativa quantidade de estudos (10) não informou o tamanho da amostra.

Somente dois estudos utilizaram testes estatísticos para a análise dos dados coletados (ROBINSON, PÉREZ-QUINONES & SCALES, 2015; KE & IM, 2014). Outros estudos, devido à adoção de designs de pesquisa menos rigorosos, não detalharam a análise dos dados.

Figura 39: Quantidade de estudos por tamanho da amostra.



Fonte: elaborada pelo autor

3.2.3. Discussão

Considerando a relevância do design de interface de usuário no desenvolvimento de software, observa-se que há uma carência de ensino destas competências no nível da Educação Básica. Encontramos somente dezesseis UIs que contemplam o design de interface de usuário. Inferiu-se que as UIs geralmente ensinam competências de programação juntamente com o design de interface de usuário, incluindo *Design Thinking*, design de interface do usuário, *design* visual, *design* centrado no usuário, usabilidade, interação humano-computador e UX. O *Design Thinking* é abordado para estimular os alunos a desenvolverem um software útil e criativo, visando solucionar problemas cotidianos de uma comunidade ou problema de algum contexto pré-definido. O design de interfaces de usuário é abordado no contexto da criação de uma interface no desenvolvimento de software, tipicamente jogos/animações e/ou *apps*. O ensino dessas competências está bem integrado no ensino da computação, tipicamente adotando abordagens ágeis nesse nível educacional. Porém, mesmo abordando conceitos de design de interface também se observou a falta de inclusão de conceitos mais concretos em alguns casos. Por exemplo, o curso da Technovation (2018) aborda em uma unidade o design de interface de forma geral, principalmente motivando conceitos gerais,

porém não oferece suporte mais concreto em termos de definição de cores, tipografia, ícones etc., no design de interfaces de *apps* a serem desenvolvidos. Assim, identificamos a necessidade de detalhamento de UIs existentes, para abordar de forma mais concreta e detalhada conceitos de design de interface.

Outro fator a considerar trata-se da omissão de informações sobre e/ou acesso aos materiais instrucionais. A predominância de material somente na língua inglesa também pode vir a dificultar uma aplicação em outros países.

Identificam-se também características no que se refere à avaliação do atingimento dos objetivos de aprendizagem de competências do design de interface. Vários artigos relatam a avaliação com base nos artefatos criados, porém, analisando-se os critérios de avaliação em detalhe, observou-se que muitas vezes a avaliação de competências do design de interface é resumida em um único critério, aquém de vários outros critérios voltados à avaliação da aprendizagem de programação. Provas ou testes abordando também questões sobre o design de interface também foram relatados, mas somente de forma pontual. Isso indica claramente também a necessidade de pesquisas voltadas ao desenvolvimento de modelos de avaliação dessas competências, como parte essencial do processo de aprendizagem.

UIs devem ser desenvolvidas de forma sistemática, orientadas por modelos de design instrucional, para que haja eficácia de aprendizagem (Branch, 2009). Porém, percebe-se que existem muitos artigos que não apresentam informações essenciais em relação ao(s) objetivo(s) de aprendizagem e/ou estratégia instrucional, nem os métodos utilizados para sistematicamente desenvolver as UIs. Este ponto fraco pode ser também observado em relação à avaliação da maioria das UIs. Vários não relatam avaliações e/ou somente de forma *ad-hoc*, com pouco rigor científico. A grande variação dos fatores avaliados de diversas formas também indica a falta de modelos de avaliação nesta área, para facilitar de forma mais uniforme a avaliação. Porém, para guiar uma adoção dessas UIs na prática e o seu desenvolvimento e melhoria de forma sistemática, precisa-se de evidências empíricas, sistematicamente levantadas.

3.2.4. Ameaças à validade

Mapeamentos sistemáticos podem ser influenciados pelo viés comum, em que resultados positivos têm maior probabilidade de serem publicados do que os

negativos. No entanto, é considerado que os resultados dos artigos, sejam positivos ou negativos, têm apenas uma pequena influência sobre esse mapeamento sistemático, uma vez que as UIs foram caracterizadas, em vez de analisar seus impactos sobre a aprendizagem. Outro risco é a omissão de estudos relevantes, e para mitigar esse risco foi construído cuidadosamente a *string* de busca, para ser a mais abrangente possível, considerando não apenas os principais conceitos, mas também os sinônimos. Também para minimizar o risco de excluir UIs existentes e que ainda não foram relatadas por meio de artigos científicos, foi realizada uma busca de forma mais abrangente nos sites de MOOCs, Google Scholar e Google, e incluímos literatura secundária a partir das referências da literatura primária. Ameaças para a seleção e a extração de dados foram mitigadas por meio do fornecimento de uma definição detalhada dos critérios de inclusão/exclusão, e de qualidade. Os estudos foram selecionados por todos os autores, discutindo a seleção até que o consenso fosse alcançado. A extração de dados foi prejudicada em alguns casos, uma vez que as informações relevantes nem sempre foram apresentadas explicitamente e/ou usando uma terminologia comumente aceita e, portanto, em alguns casos tiveram que ser inferidas. No entanto, essa inferência foi feita pelos coautores em conjunto.

4. MODELO EDUCACIONAL - “CURSO FAÇA O SEU APP”

Na análise do estado da arte, foi identificado que o ensino de computação no ensino básico não aborda tópicos de ES e EU, de forma sistemática, no processo de desenvolvimento de software. Observando esta oportunidade de pesquisa, este trabalho desenvolve um modelo educacional que ensina os alunos a desenvolver um *app* seguindo um processo sistemático de desenvolvimento de software, incluindo conceitos de ES e EU. O desenvolvimento deste modelo segue a abordagem ADDIE (BRANCH, 2009). São neste capítulo apresentado os resultados da análise, projeto e desenvolvimento dos materiais.

4.1. ANÁLISE DO CONTEXTO

De acordo com o modelo de design instrucional (BRANCH, 2009), para se definir os objetivos de aprendizagem é necessário realizar uma análise do público alvo e do ambiente. Assim, são analisados i) o público-alvo e ii) as características das escolas.

Público-alvo. O público-alvo são alunos do Ensino Fundamental II das escolas brasileiras, com idade entre 10 a 15 anos (a partir do quinto e até a nona série). Os alunos neste nível escolar são tipicamente alfabetizados e possuem conhecimento básico da língua inglesa. Além disso, este público tem a capacidade de controlar dispositivos eletrônicos, como smartphones, tablets e computadores (D'ANGELO, 2017). A grande maioria já possui um smartphone próprio e dentre as atividades é realizado o acesso à internet (CGI.BR, 2018). Dos aparelhos celulares mais vendidos no Brasil, cerca de 95,1% utilizam o sistema operacional Android (IDC BRASIL, 2018). Como o ensino de computação no Brasil só é realizado em nível superior, esses estudantes tipicamente não possuem nenhum conhecimento de conceitos de programação e computação em geral. As atividades que mais lhe chamam a atenção incluem a prática de esportes, o uso de redes sociais e encontros com amigos (PISA, 2015).

Características das escolas. O modelo educacional é projetado para o ensino de computação em escolas, logo, ela só pode ser aplicada em instituições de ensino que possuam em sua infraestrutura um laboratório de informática ou alguma sala que disponha de computadores para a utilização em aula.

A grande maioria das escolas públicas municipais têm computadores e acesso à internet por meio de rede cabeada ou Wi-Fi. Geralmente esses computadores possuem instalado o sistema operacional Linux Educacional, entretanto, podem existir escolas que utilizam outro sistema operacional livre ou proprietário (PROINFODATA, 2019). Sendo assim, o *App Inventor* é compatível com os sistemas operacionais Linux, Windows e Macintosh (MIT & GOOGLE, 2019).

Estes computadores geralmente são alocados em uma sala de informática que comporta entre 10 a 25 alunos, e que possui um professor responsável. A maioria das escolas não possui professores de computação. O professor responsável pela sala de informática tem tipicamente formação em Pedagogia, com conhecimentos básicos em informática e pouco conhecimento em computação. Esse fato se dá muito pelo motivo de que poucos profissionais se formam como professores de computação, apenas 0,6% (INEP, 2016).

As atividades de informática geralmente envolvem outra disciplina e de maneira interdisciplinar, utilizando os conteúdos vistos em sala para aprender os conceitos relacionados à informática. Um exemplo típico disto é o uso do Microsoft Word em uma aula de Português, na qual os alunos aprendem a controlar a ferramenta aprendendo os conteúdos de uma aula comum.

As aulas no Ensino Fundamental II têm duração aproximada de 45 minutos, e são distribuídas no período matutino ou vespertino. As turmas possuem uma média de 21 a 27 alunos. Geralmente a carga horária para o ensino da computação é escasso, pois a carga de conhecimento requerido é cada vez maior.

4.2. PROJETO

De acordo com a análise de contexto, foi definido o design do modelo educacional “Faça seu *app*”. Nesta etapa foi definido o objetivo de aprendizagem e o plano de ensino, que inclui o sequenciamento, conteúdo, estratégias e métodos instrucionais e as avaliações das aulas.

4.2.1. Objetivo de aprendizagem

Os objetivos de aprendizagem estão alinhados com os conhecimentos de programação, ES e EU, requeridos principalmente pelo currículo voltado ao ensino de computação no Ensino Básico CSTA/ACM K-12 (2017). Como este currículo não contempla um processo completo de ES/EU, conhecimentos de outros currículos

também foram utilizados para dar suporte ao seu processo de desenvolvimento. Para possibilitar ampla aplicação do modelo educacional visando a sua adaptação a contextos variados nas escolas são propostas duas versões com escopo e duração diferentes:

i. “Faça seu *app*” (versão longa): voltada para ensinar a desenvolver um *app* dedicado à solução de uma necessidade/problema na comunidade. Esta unidade instrucional tem 12 encontros cada um de 3 horas/aula, totalizando 36 horas/aula;

ii. “Faça um *app*” (versão curta): voltada para ensinar o desenvolvimento de um *app* pré-definido para visualizar pontos de coletas de tampas PET. Esta unidade instrucional tem 5 encontros cada um de 3 horas/aula, totalizando 15 horas/aula.

O Quadro 22 demonstra os objetivos de aprendizagem e em qual versão da unidade instrucional foi aplicada.

Quadro 22: Objetivos de aprendizagem da UI.

Id	Objetivo de aprendizagem	Área de conhecimento	Fonte	UI (i)	UI (ii)
OA1	Compreender algoritmos como um conjunto de instruções passo-a-passo para realizar tarefas.	Algoritmo e Programação	(CSTA, 2017: 1A-AP-08)	x	x
OA2	Explicar o conceito de um ciclo de vida de software e fornecer um exemplo, ilustrando suas fases, incluindo as entregas que são produzidas.	Engenharia de Software/ Engenharia de Usabilidade	ACM/IEEE (2013), UXQB (2018)	x	
OA3	Desenvolver artefatos computacionais interativamente de forma colaborativa, seguindo um cronograma.	Engenharia de Software	(CSTA, 2017: 2-AP-18)	x	
OA4	Identificar e resolver problemas criando sistemas de software interativos.	Algoritmo e programação / Engenharia de Usabilidade	(CSTA, 2017: 1B-CS-03, 3A-AP-13), (AIGA, 2013)	x	
OA5	Analisar o contexto de sistemas de software interativo em termos de usuários, tarefas, dispositivos e ambientes de uso.	Engenharia de Usabilidade	(AIGA, 2013), (ISO/IEC 9241-220:2019, 2019) (CSTA, 2017: 1B-IC-19), ACM/IEEE (2013)	x	

OA6	Especificar requisitos de sistemas de software interativos em termos de funcionalidade e usabilidade.	Engenharia de Software / Engenharia de Usabilidade	(CSTA, 2017: 2-AP-19), ACM/IEEE (2013)	x	
OA7	Criar protótipos de sistemas de software interativos em diferentes níveis (esboços, baixa fidelidade, alta fidelidade, funcional).	Engenharia de Usabilidade	(CSTA, 2017: 2-AP-19, 1B-AP-13, 2-AP-13), ACM/IEEE (2013)	x	
OA8	Projetar design que combine componentes de <i>hardware</i> e software para coletar e trocar dados (sensores, APIs, etc.).	Engenharia de Software / Engenharia de Usabilidade	(CSTA, 2017: 2-CS-02), ACM/IEEE (2013), (GARRET, 2011)	x	
OA9	Modelar processos criando e seguindo algoritmos/mapas de navegação para concluir tarefas.	Algoritmo e programação	(CSTA, 2017: 1A-AP-08)	x	
OA10	Usar fluxogramas, pseudocódigo e/ou mapas de navegação para resolver problemas complexos.	Algoritmo e programação	(CSTA, 2017: 2-AP-10)	x	
OA11	Projetar o design visual (cores, tipografia, ícones, imagens, etc.) do sistema de software interativo.	Engenharia de Usabilidade	ACM/IEEE (2013), (CSTA, 2017: 2-IC-21), (GARRET, 2011)	x	
OA12	Construir sistemas de software interativos que incluam sequenciamento, eventos, condicionais, variáveis, listas e <i>strings</i> , usando uma linguagem de programação visual baseada em blocos.	Algoritmo e programação	(CSTA, 2017: 1B-AP-09, 1B-AP-10, 2-AP-11, 2-AP-12, 3A-AP-14, 3A-AP-16)	x	x
OA13	Procurar e incorporar o <i>feedback</i> dos membros da equipe e dos usuários para refinar uma solução que atenda às necessidades do usuário.	Algoritmo e programação/ Engenharia de Software/Engenharia de Usabilidade	(CSTA, 2017: 2-AP-15)	x	
OA14	Testar e refinar um sistema de software interativo para funcionalidade e usabilidade.	Engenharia de Software/ Engenharia de Usabilidade	(CSTA, 2017: 2-AP-17, 1B-AP-15, 3A-AP-21)	x	x

OA15	Recomendar melhorias no design de dispositivos de computação, com base nos resultados de verificação e validação.	Engenharia de Software/ Engenharia de Usabilidade	(CSTA, 2017: 2-CS-01)	x	
OA16	Compartilhar o sistema de software interativo desenvolvido.	Algoritmo e programação	(CSTA, 2017: 1B-AP-12, 1B-AP-17, 2-AP-16), (LEE <i>et al.</i> , 2007)	x	x

4.2.2. Contexto de aplicação

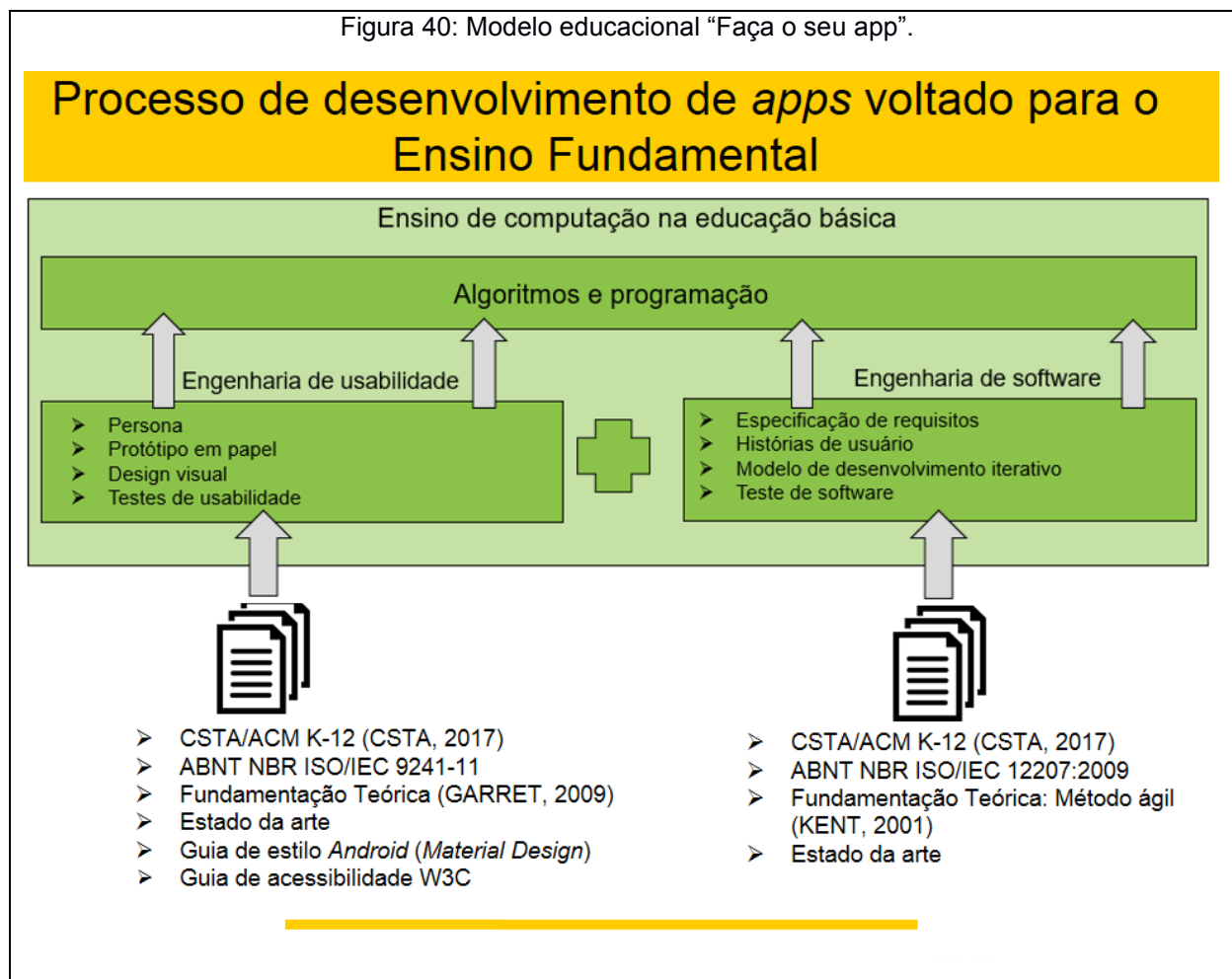
O modelo educacional é projetado para ser aplicado de forma interdisciplinar, com disciplinas do Ensino Fundamental e/ou no contraturno, ou como atividade extracurricular. A disciplina de Ciências é um exemplo no qual a unidade pode explorar alguns conceitos práticos de computação (programação) e o conteúdo da disciplina. A unidade instrucional visa atender aos objetivos da unidade temática: Ambientes, Recursos e Responsabilidades da Base Nacional Comum Curricular (MEC, 2018), da disciplina de Ciências para o 5º ano. Em relação ao modo de ensino, a unidade instrucional é projetada com a possibilidade de ser ministrada de forma presencial, sendo coordenada por um professor da Educação Básica, com auxílio, ou não, de um mentor capacitado no assunto e/ou via ensino a distância.

O modelo educacional segue a abordagem de ação computacional, uma nova concepção para a educação em computação. Esta abordagem propõe que ao mesmo tempo em que aprendem computação, os alunos também devem ter oportunidades de criar soluções de problemas com uso de computação, e que impactem diretamente em suas vidas e em suas comunidades (TISSENBAUM, SHELDON & ABELSON, 2019).

4.2.3. Plano de ensino

O plano de ensino é desenvolvido com base nos objetivos de aprendizagem, análise do público-alvo e do ambiente escolar. Os conteúdos são abordados conforme os objetivos de aprendizagem e foram baseados em método e modelo de desenvolvimento de *apps*, guias de desenvolvedores, fundamentação teórica e estado da arte (Figura 40).

Figura 40: Modelo educacional “Faça o seu app”.



Fonte: elaborada pelo autor.

O plano de ensino da unidade “Faça seu *app*” é dividido em 2 partes (Quadro 23).

As duas primeiras aulas têm o propósito de motivar os alunos sobre o ensino de computação e o desenvolvimento de aplicativos usando a ferramenta *App Inventor*. Na primeira aula, é apresentado como pode ser divertido e útil aprender computação, bem como explicar como é a carreira de um profissional nesta área. Em seguida, os alunos são convidados a responder um questionário sobre seus conhecimentos e gostos relacionados a Computação. Após, o conceito de algoritmo é ensinado e posteriormente praticado por meio de um jogo de tabuleiro.

A segunda aula tem o propósito de ensinar conceitos de programação. Os alunos aprendem como utilizar a ferramenta *App Inventor*, desenvolvendo um *app* seguindo um tutorial pré-definido. Os alunos aprendem a programar o *app* “Encontre-me”, que tem como objetivo permitir os usuários mandarem sua localização atual via *Whatsapp* (WHATSAPP, 2019). No decorrer do processo de

desenvolvimento, são ensinados diversos conceitos de programação: variáveis, estrutura condicional, laços, lógica booleana, evento, sensor, API e *string*.

A partir da terceira aula, a unidade instrucional ensina o desenvolvimento completo de *apps* úteis, e todos os passos necessários da criação são ensinados. Na aplicação presencial, os alunos são encorajados a formarem duplas, e com o apoio dos instrutores, seguem no processo de desenvolvimento de *apps*, conforme o modelo apresentado na seção 4.3. Neste processo os alunos são estimulados a descobrir um problema e a idealizar uma solução em qualquer área do cotidiano por meio de *brainstorming*. Em seguida, os alunos aprendem a fazer análise de contexto e definir os requisitos funcionais e de usabilidade necessários. Com os requisitos definidos, é adotado o método ágil como base para o desenvolvimento do *app*, no qual primeiramente os alunos desenvolvem e testam um design de baixa fidelidade das telas por meio de protótipos em papel. O *app* então é programado e testado com a ferramenta *App Inventor*. Os alunos são ensinados a desenvolver e a testar as funcionalidades definidas nos requisitos funcionais, de forma interativa e incremental.

Quadro 23: Plano de ensino da unidade instrucional “Faça seu *app*”.

Parte 1 - Conceitos fundamentais	Aula	Conteúdo em geral	Algoritmo e programação	ES	EU	ID objetivo de aprendizagem	Estratégia instrucional	Método instrucional	Avaliação
	1	Motivação sobre ensino de computação e o desenvolvimento de aplicativos	Algoritmo, pseudocódigo	-	-	OA1	Aprendizagem experimental, instrução direta, estudo independente	Jogo, ensino explícito, exercitar e praticar, tarefas de casa	Observação, questionário
	2	Conceitos básicos de computação: algoritmo/programação	Condicional, evento, sensor, API, string, variável	-	Design visual, Guia de estilo (<i>Material Design</i>) (apenas nos extras)	OA8, OA10, OA12	Instrução direta, aprendizado experimental	Ensino explícito, demonstrações, condução de experimentos	Observação, <i>app</i> “Encontre-me” desenvolvido
Parte 2 - Faça seu <i>app</i> !	Aula	Conteúdo em geral	Algoritmo e programação	ES	EU	ID objetivo de aprendizagem	Estratégia instrucional	Método instrucional	Avaliação
	1	Processo de desenvolvimento de <i>apps</i> e identificação do problema e solução	-	Processo de desenvolvimento de <i>apps</i>	<i>Design Thinking</i>	OA2, OA4	Instrução direta, instrução interativa, estudo independente	Ensino explícito, demonstrações, resolução de problemas, prática em pares, tarefas de casa	Artefatos desenvolvidos
	2	Análise de contexto e especificação de requisitos	-	História de usuário, especificação de requisitos de software, <i>Task case</i>	Análise de contexto (usuário, tarefa(s), equipamento e ambiente(s)), persona, especificação de requisitos de usabilidade	OA5, OA6	Instrução direta, instrução interativa	Ensino explícito, demonstrações, prática em pares	Artefatos desenvolvidos
	3	Design de baixa fidelidade do <i>app</i> e testes	-	-	<i>Sketch</i> /protótipo de baixa fidelidade (em papel), teste de <i>sketch</i>	OA7	Instrução direta, instrução interativa	Ensino explícito, demonstrações, prática em pares	Artefatos desenvolvidos, <i>app</i> desenvolvido

4, 5, 6	Programação e teste do protótipo no <i>App Inventor</i>	Fluxograma, (opcional: banco de dados, listas, notificação),	Desenvolvimento iterativo: codificação e teste de unidade	Criar e testar sketch (protótipo em papel)	OA3, OA9, OA10	Instrução direta, instrução interativa	Ensino explícito, demonstrações, prática em pares	Observação, <i>app</i> desenvolvido
7	Criação e teste do design visual no <i>App Inventor</i>	-	-	Design visual: cores, tipografia, ícones, imagens, marca, <i>launcher icon</i> , Guia de estilo (<i>Material Design</i>),	OA11	Instrução direta, instrução interativa	Ensino explícito, demonstrações, prática em pares	Observação, <i>app</i> desenvolvido
8	Teste de sistema	-	Teste funcional, caso de teste	Teste de usabilidade	OA13, OA14, OA15	Instrução direta, instrução interativa	Ensino explícito, demonstrações, prática em pares	Observação, <i>app</i> desenvolvido, questionário
9	Compartilhando o <i>app</i>	Compartilhamento de <i>apps</i> , publicar (galeria e <i>playstore</i>), versionar software, distribuir código fonte	-	-	OA16	Instrução direta, instrução interativa	Ensino explícito, demonstrações, prática em pares	Observação

Fonte: dados do autor.

A próxima etapa é ensinar aspectos mais detalhados sobre o design visual das telas do aplicativo, como tipografia, cores, menus etc. A escolha/produção de imagens e ícones também é ensinada. Os alunos então utilizam as telas dos *apps* e aplicam os conhecimentos de acordo com o guia de estilo *Material Design*. Após a produção do aplicativo com design visual, os alunos aplicam um teste de caso de uso e teste de usabilidade, com base nos requisitos funcionais e de usabilidade definidos anteriormente. Os testes de usabilidade são realizados entre os próprios alunos.

Na última aula os alunos aprendem a compartilhar seu *app* na galeria do *App Inventor*. Desta forma, os alunos podem visualizar o aplicativo criado por seus colegas, podendo modificá-lo e/ou utilizá-lo. O instrutor pode realizar uma avaliação teórica sobre todos os conhecimentos ensinados em todo o decorrer da unidade. A prova possui perguntas de múltipla escolha, que devem ser assinaladas pelos estudantes, sendo mais uma forma de avaliação dos alunos após aplicação da UI. Ao final, os alunos são convidados a responder um questionário para avaliar o que mudou em suas percepções em relação ao questionário respondido antes da oficina.

A unidade instrucional “Faça um *app*” aborda apenas conceitos de EU (Quadro 24).

As duas primeiras aulas têm o mesmo propósito que a v.1.0, a diferença é que os alunos aprendem a programar o aplicativo EcoPET seguindo um tutorial. Este *app* tem como objetivo apresentar informações sobre os locais e ações do projeto EcoPET¹⁹. Este projeto é organização não governamental que coleta tampas de garrafas PET para ajudar animais que vivem abandonados nas ruas das cidades de Florianópolis. Esta unidade instrucional tem uma duração menor que a unidade instrucional “Faça seu *app*”, assim, abrange menos conceitos de programação: variáveis, evento e *string*. Além disso, o escopo de ensino aborda apenas *design visual*, não incluindo conhecimentos de ES. Na terceira e quarta aulas, os alunos apenas aprendem a aplicar design visual no *app* EcoPET. Na última aula os alunos aprendem a compartilharem seu *app* na galeria do *App Inventor*.

¹⁹ Projeto EcoPET: <https://www.facebook.com/ecopettampastampinhas/>

Quadro 24: Plano de ensino da unidade instrucional “Faça um app”.

Aula	Conteúdo em geral	Algoritmo e programação	EU	ID objetivo de aprendizagem	Estratégia instrucional	Método instrucional	Material Instrucional	Avaliação
1	Motivação sobre ensino de computação e o desenvolvimento de aplicativos; Conceitos básicos de computação: algoritmo/programação. Visão geral do processo de desenvolvimento de <i>apps</i>	Algoritmo, pseudocódigo e Ciências	-	OA1	-Instrução direta (aula expositiva); -Atividade coletiva; -Jogo educacional.		-Slides; -Vídeo; -Jogo SplashCode.	-Avaliação processual; -Observação; -Diário de campo; -Avaliação do jogo de tabuleiro.
2	Introdução a programação com <i>App Inventor</i> – ecopet	Evento, string, variável	-	OA12	-Instrução direta (aula expositiva); -Atividade prática de programação e teste.		-Slides; -Ambiente <i>App Inventor</i> - <i>App Inventor</i> companion para uso no celular;	-Avaliação processual; -Observação; -Diário de Campo.
3	Criação, programação e teste do design visual no <i>App Inventor</i> Cores Tipografia Desenvolvimento de uma prática que gere impacto social	Engenharia de Usabilidade Impactos da computação ciências	<i>Design</i> visual: cores e tipografia.	OA14	-Instrução direta (aula expositiva); -Atividade de fixação (exercício); -Atividade prática de programação e teste.		-Slides	-Avaliação processual; -Observação; -Diário de Campo.
4	Criação, programação e teste do design visual no <i>App Inventor</i> : Imagens e hierarquia	- <i>Design</i> visual; - Algoritmo e programação; - Verificação e validação de software.	<i>Design</i> visual: imagem, composição.	OA14	-Teste do protótipo no <i>App Inventor</i>		Slides	-Avaliação processual; -Observação; -Diário de Campo; - <i>App</i> desenvolvido

								pelos alunos.
5	Compartilhando os <i>apps</i> Divulgando o <i>app</i>	-Algoritmo e programação; -Habilidade de pensamento crítico, resolução problema, comunicação e colaboração; -Ciências.	-	OA16			-Slides; -Vídeos; Uso do Google drive; -Câmera -Criação de slides por alunos; - <i>App Inventor</i> .	-Avaliação processual; -Diário de Campo.

4.3. MODELO DE DESENVOLVIMENTO DE APPS

Para guiar os alunos do Ensino Fundamental a desenvolverem seus próprios apps é definido um modelo de processo de desenvolvimento de software. O modelo é desenvolvido seguindo o método de modelagem de processo proposto por Acuña & Ferré (2001). O processo de desenvolvimento é baseado na abordagem *Design Thinking* (CHEN & HUANG, 2017), adotando uma forma ágil de desenvolvimento de apps e integrando elementos de ES e EU. O modelo desenvolvido especifica atores, papéis, atividades e produtos:

-*Atores*: os atores são subdivididos em a) atores humanos, que são os alunos que vão desenvolver o app e os instrutores que os guiam na execução do processo, e b) atores de sistemas, que neste processo é a ferramenta *App Inventor*;

-*Papéis*: o aluno é responsável pelo desenvolvimento do app e por criar seus devidos artefatos. O instrutor guia os alunos em relação ao processo de desenvolvimento, aplica os questionários e avaliações, além de ajudá-los a superar as dificuldades que possam comprometer o andamento do processo ou desenvolvimento;

-*Atividades*: a figura 41 apresenta as atividades e artefatos gerados no processo. Essas atividades têm como base modelos de processo ágil de desenvolvimento de apps (ABRAHAMSSON, 2004; JEONG, LEE & SHIN, 2008) e processo de *Design Thinking* (CHEN & HUANG, 2017):

a) *Imaginar*: os alunos realizam um *brainstorming* para identificar problemas relacionados a um conjunto de temas especificado, e uma solução por meio de um app. Os alunos devem documentar a descrição do problema, os usuários envolvidos e a idealização da solução, identificando, por exemplo, os componentes do app para atender à solução;

b) *Analisar*: Com base no problema definido, os alunos definem e documentam características, como perfil dos usuários, objetivos/tarefas que devem ser cumpridas por meio de história de usuários, dispositivos tipicamente utilizados e ambiente de uso. Os alunos devem definir e documentar também os requisitos funcionais e de usabilidade;

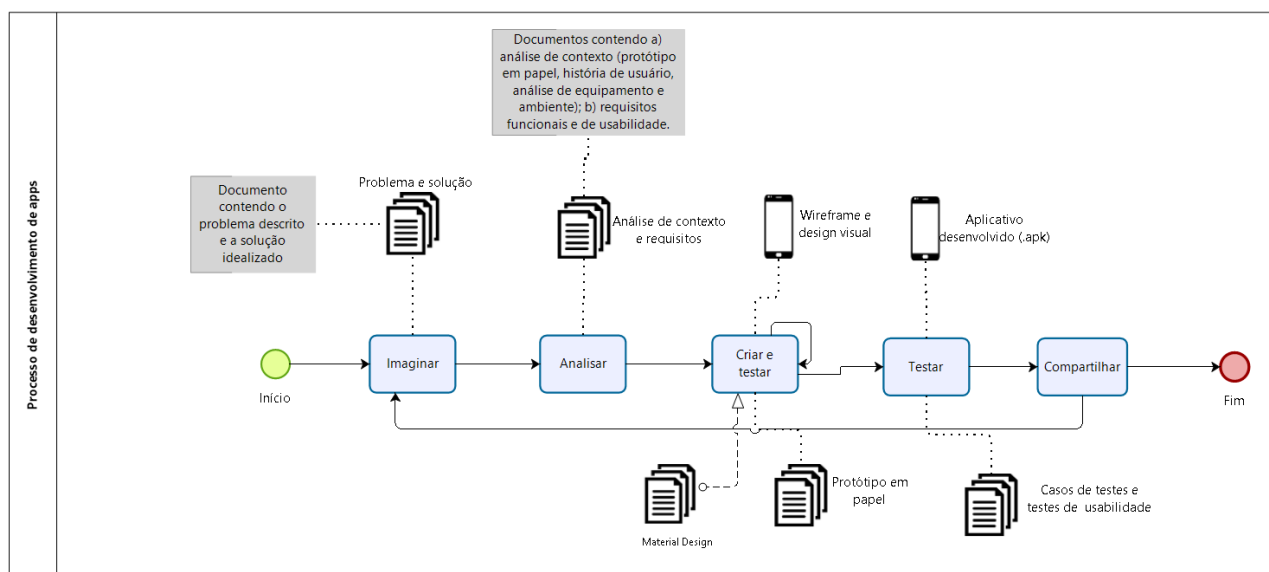
c) *Criar e testar*: os alunos desenvolvem e documentam o app de forma interativa e incremental. Na primeira interação é desenvolvido o protótipo em papel, e na segunda já utilizam o *App Inventor* para desenvolver o *wireframe*, e posteriormente desenvolver e testar as funcionalidades definido nos requisitos

funcionais. Uma última interação é realizada com o desenvolvimento dos aspectos mais detalhados sobre o design visual das telas do aplicativo, como tipografia, cores, menus, produção de imagens e ícones;

d) *Testar*: os alunos realizam os testes funcionais e de usabilidade com base nos requisitos definidos na análise. Esses testes são realizados por meio de testes de caso de uso e de métricas de usabilidade. Os testes de usabilidade são aplicados com os próprios alunos, e conforme forem encontrando melhorias os alunos realizam a correção. No final desta etapa os alunos já devem apresentar o *app* desenvolvido por completo;

e) *Compartilhar*: os estudantes compartilham seu *app* na galeria do *App Inventor* para os outros alunos visualizarem o *app* desenvolvido pelos colegas. Após o compartilhamento, o aluno pode incluir novas funcionalidades/melhorias necessárias, percorrendo as atividades novamente, caso necessário.

Figura 41: Processo de desenvolvimento de apps.


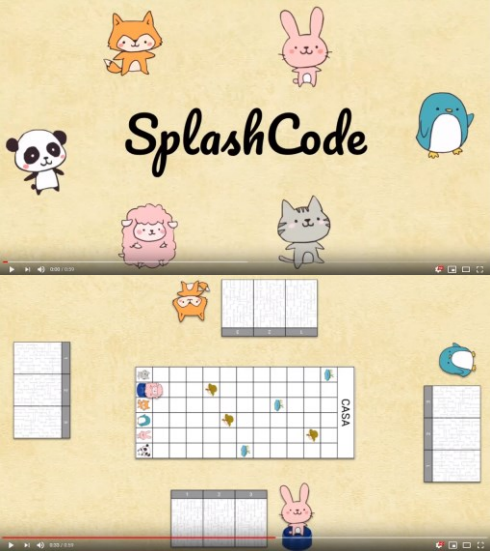



Fonte: elaborada pelo autor.


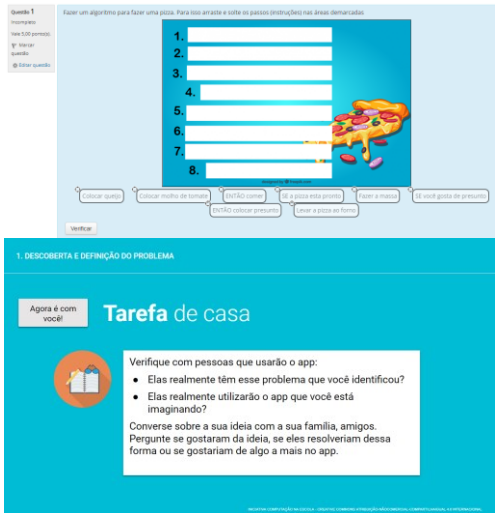
4.4. DESENVOLVIMENTO DO MATERIAL DIDÁTICO

De acordo com o plano de ensino definido, são desenvolvidos os materiais instrucionais, que servem como recurso didático de apoio para a unidade instrucional “Faça seu *app*”. Alguns materiais foram adaptados e desenvolvidos para o uso na unidade instrucional “Faça um *app*”. Esses materiais foram utilizados tanto nas aulas presenciais quanto como material para casa (Quadro 25).

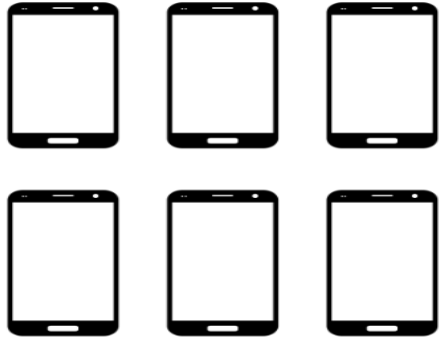
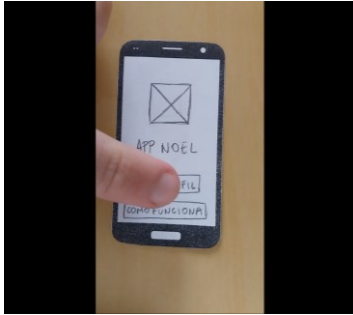

Quadro 25: Materiais didáticos desenvolvidos.

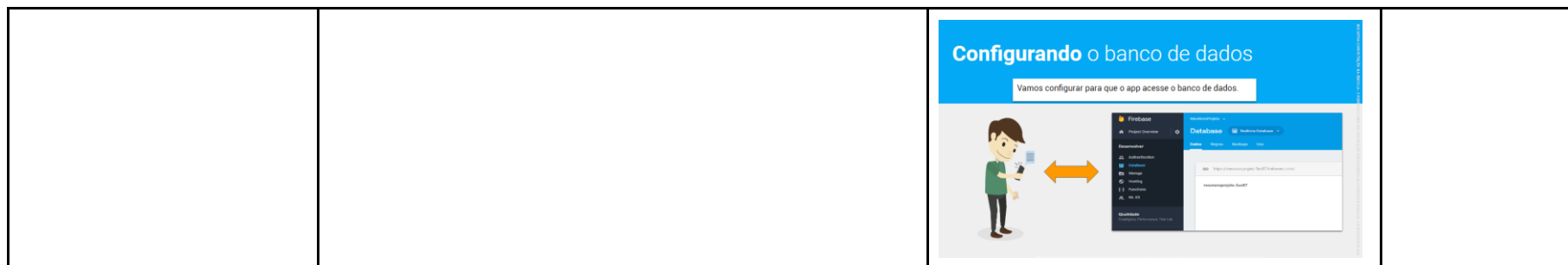
Material	Descrição	Imagem	UI utilizada
<p>Jogo SplashCode (GRESSE <i>et al.</i>, 2019)</p>	<p>Jogo de tabuleiro para aplicar conceitos de algoritmos. Os jogadores utilizam as cartas de movimento para fazer os personagens chegarem até o campo <i>casa</i>, desviando dos obstáculos.</p>		<p>Ambas</p>
<p>Vídeos jogo <i>SplashCode</i></p>	<p>Vídeos de auxílio para aprender a como jogar o jogo SplashCode. Existe um conjunto de três vídeos curtos: “preparação”, “cartas” e “como jogar”.</p>		<p>Ambas</p>

<p>Tutorial <i>app</i> “EcoPET”</p>	<p>Desenvolvimento e tutorial para construção do <i>app</i> “EcoPET”. Este <i>app</i> tem por objetivo apresentar informações sobre os locais e ações do projeto EcoPET. Durante o desenvolvimento do <i>app</i> pelo tutorial, vários outros conceitos de computação são abordados, como variável, eventos e outros.</p>		<p>Faça um <i>app</i></p>
<p>Tutorial <i>app</i> “Encontre-me”</p>	<p>Desenvolvimento e tutorial para construção do <i>app</i> “Encontre-me”. Este <i>app</i> tem por objetivo permitir os usuários mandarem sua localização atual via <i>WhatsApp</i>. Durante o desenvolvimento do <i>app</i> pelo tutorial, vários outros conceitos de computação são abordados, como <i>APIs</i>, eventos e outros.</p>		<p>Faça seu <i>app</i></p>

<p>Mini-tutoriais para evoluir o <i>app</i> “Encontre-me”</p>	<p>Conjunto de três mini-tutoriais para a evolução do <i>app</i> “Encontre-me”. Os tutoriais ensinam a adicionar um campo para o usuário digitar alguma mensagem além da localização, compartilhar via outros <i>apps</i> e melhorar o design visual.</p>	 <p>Mini Tutorial Design visual</p> <p>COMPUTAÇÃO NA ESCOLA</p> <p>PROGRAMANDO</p> <p>Explicando o funcionamento da funcionalidade de compartilhamento.</p> <p>Diagrama de fluxo: Seu app (círculo laranja) envia uma mensagem (seta verde) para o App alvo (círculo cinza) via API. O App alvo responde com uma mensagem enviada (seta verde) de volta para o Seu app.</p>	<p>Faça seu <i>app</i></p>
<p>Tarefas de casa</p>	<p>Materiais para tarefas de casa para os alunos colocarem em prática o que foi visto no conteúdo. Após o primeiro encontro, os alunos devem aplicar o conceito de algoritmos para montar uma pizza e também para colocar o lixo na lixeira correta. Outra tarefa de casa é a verificação do problema com a comunidade acerca dos alunos, confirmando sua proposta de problema.</p>	 <p>Questão 1</p> <p>Escreva um algoritmo para fazer uma pizza. Para isso arraste e solte os passos (instruções) nas áreas demarcadas.</p> <p>1. _____ 2. _____ 3. _____ 4. _____ 5. _____ 6. _____ 7. _____ 8. _____</p> <p>Colocar queijo Colocar molho de tomate Cortar tomates Colocar o resto da pizza Cortar a massa Colocar a pizza no forno</p> <p>1. DESCOBERTA E DEFINIÇÃO DO PROBLEMA</p> <p>Agora é com você! Tarefa de casa</p> <p>Verifique com pessoas que usarão o app:</p> <ul style="list-style-type: none"> Elas realmente têm esse problema que você identificou? Elas realmente utilizarão o app que você está imaginando? <p>Converse sobre a sua ideia com a sua família, amigos. Pergunte se gostaram da ideia, se eles resolveriam dessa forma ou se gostariam de algo a mais no app.</p>	<p>Ambas</p>

<p>Slides a serem apresentados nas aulas presenciais</p>	<p>Slides com o conteúdo envolvendo todos os conceitos abordados no projeto. Os slides podem ser utilizados para ministrar aulas presenciais ou seguir de maneira autônoma.</p>		<p>Faça seu <i>app</i></p>
<p><i>Workbooks</i></p>	<p>Guias de referência para a produção dos artefatos envolvidos no processo de desenvolvimento de <i>apps</i> proposto no projeto.</p>	<p>Descrição da solução</p> <p>Nome do seu app</p> <p>Quem vai fazer o que com o app?</p> <p>Por que seu app é impressionante?</p> <p>Circule os componentes que seu app conterá</p>	<p>Faça seu <i>app</i></p>

		<p>Workbook aula 2.3 Template para o sketch - Design inicial</p> 	
Vídeo teste de usabilidade	Vídeo para ensinar a como realizar o teste de usabilidade do protótipo de interface em papel.		Faça seu <i>app</i>
Tutoriais de funcionalidades específicas do <i>App Inventor</i>	Tutoriais para desenvolvimento de funcionalidades características de diversos <i>apps</i> no <i>App Inventor</i> , como banco de dados, listas, <i>login</i> , entre outros.	<p>PARA QUE SERVE LOGIN?</p> <p>A identificação de acesso ao um aplicativo é feito por Login. Neste tutorial vamos aprender a criar um login onde o usuário vai se cadastrar no app e depois acessá-lo.</p> 	Faça seu <i>app</i>



Fonte: dados do autor.

4.5. AVALIAÇÃO DO ALUNO

Para avaliar a aprendizagem dos alunos são adotados diversos tipos de avaliações:

- *Avaliação de desempenho*: tem o propósito de avaliar o desempenho do aluno, com base nos artefatos de software criados durante o desenvolvimento. O *app* desenvolvido pelo aluno é avaliado utilizando a ferramenta de avaliação automatizada CodeMaster (ALVES, 2019). Esta ferramenta avalia conceitos de programação baseado na rubrica definida por Alves (2019), avaliando conceitos básicos de operadores, variáveis, *strings*, nomeação, listas, persistência, eventos, laços, condicionais, sincronização, abstração, extensões, sensores, animação, mapas e telas (Figura 42).

Figura 42: Trecho da rubrica CodeMaster v2.0.

PA2. Qual o nível de desempenho em representação de dados com relação às práticas do pensamento computacional? (CSTA, 2016; 2017)				
Item	0 pontos	1 ponto	2 pontos	3 pontos
Variáveis: verificar se criou ou modificou valores de variáveis.	Sem uso de variáveis.	Modificação ou uso de variáveis predefinidas.	Criação e operação com variáveis	
Strings: verificar se criou ou modificou valores de strings.	Sem uso de strings.	Uso do comando de criação de string para alterar textos de elementos.	Criação e operação com strings.	
Nomeação: verificar se os nomes de variáveis são alterados do padrão.	Nenhum ou poucos nomes são alterado do padrão. (menos do que 10%)	De 10 a 25% dos nomes são alterados do padrão.	De 26 a 75% dos nomes são alterados do padrão.	Mais de 76% dos nomes são alterados do padrão.
Listas: verificar se são usadas listas.	Não usa listas.	Usa uma lista unidimensional.	Usa mais de uma lista unidimensional.	Usa uma lista de tuplas (map).
Persistência de dados: verificar se são usados componentes de persistência de dados	Dados são armazenados em variáveis ou propriedades de componentes e não tem persistência quando app é	Usa persistência em arquivo (File ou Fusion Tables).	Usa algum dos bancos de dados locais do App Inventor (TinyDB).	Usa uma base de dados web tinywebdb ou Firebase do App Inventor (firebase ou TinyWebDB).

Fonte: ALVES (2019).

Os artefatos criados durante a unidade, referentes a cada aula, são avaliados por meio de uma rubrica (Quadro 26), possibilitando avaliar os alunos em termos do desempenho na criação destes artefatos com uma nota de 0 (zero, mínima) a 30 (trinta, máxima). Esta rubrica foi desenvolvida visando os objetivos de aprendizagem da unidade instrucional.

Quadro 26: Rubrica para avaliação da aprendizagem relacionada aos artefatos criados.

Workbook	Item	0 pontos	1 ponto	2 pontos	3 pontos
Aula 2.1	Descrição do problema	Não soube descrever	Descreveu incompleto	Descreveu com pouca clareza	Descreveu com clareza
	Descrição da solução	Não soube descrever	Descreveu incompleto	Descreveu com pouca clareza	Descreveu com clareza
Aula 2.2	<i>Persona</i>	Não soube caracterizar a <i>persona</i>	Caracterizou incompleto	Caracterizou com pouca clareza	Caracterizou com clareza
	Histórias de usuário	Não soube contar	Descreveu incompleto	Descreveu com pouca clareza	Descreveu com clareza
	Passos de interação	Não soube descrever	Descreveu incompleto	Descreveu com pouca clareza	Descreveu com clareza
	Resultados de usabilidade	Não soube descrever	Descreveu incompleto	Descreveu com pouca clareza	Descreveu com clareza
Aula 2.3	Design de interface inicial	Não soube realizar	Realizou incompleto	Realizou com pouca clareza	Realizou com clareza
	Resultados do teste do design da interface	Não soube descrever	Descreveu incompleto	Descreveu com pouca clareza	Descreveu com clareza
Aula 2.8	Resultados do teste funcional	Não soube descrever	Descreveu incompleto	Descreveu com pouca clareza	Descreveu com clareza
	Resultados do teste de usabilidade	Não soube descrever	Descreveu incompleto	Descreveu com pouca clareza	Descreveu com clareza
Total de pontos atingível: 30					

O design visual dos *apps* desenvolvidos também são avaliados tanto por meio da análise individual dos elementos da interface (como cor, ícones e tipografia), quanto pela composição da interface como um todo, conforme definido na rubrica de unidade instrucional *Design visual* (Quadro 27). O objetivo desta é avaliar os *apps*

em termos de componentes visuais ensinados em sala de aula estão em concordância com o guia de estilo *Material Design*. Esta rubrica foi baseada no estudo realizado por Ferreira, Gonçalves e Wangenheim (2019) que analisa diversos princípios, modelos e frameworks de design visual de interface de aplicativos. Neste estudo são levantados os principais elementos que compõem o design visual de um aplicativo, como paleta de cores, tipografia, contraste, etc.

Quadro 27: Rubrica para avaliação da aprendizagem sobre design visual (FERREIRA, GONÇALVES & WANGENHEIM, 2019).

Item	0 pontos	1 ponto	2 pontos	3 pontos	4 pontos
A paleta de cores está coerente com o tema escolhido	Não	Vagamente	Parcialmente, em alguns casos	Largamente	Completamente
A organização das cores facilita a interação do usuário	Não	Vagamente	Parcialmente, em alguns casos	Largamente	Completamente
As cores auxiliam na hierarquia de informações	Não	Vagamente	Parcialmente, em alguns casos	Largamente	Completamente
O contraste entre a cor do texto e do fundo da tela asseguram a legibilidade	Não	Vagamente	Parcialmente, em alguns casos	Largamente	Completamente
Os textos apresentam tamanho de fonte agradáveis para leitura	Não	Vagamente	Parcialmente, em alguns casos	Largamente	Completamente
O alinhamento do texto contribui para leitura e harmonia do design visual	Não	Vagamente	Parcialmente, em alguns casos	Largamente	Completamente
O tamanho das fontes auxilia na hierarquia de informações	Não	Vagamente	Parcialmente, em alguns casos	Largamente	Completamente
As imagens estão de acordo com o tema	Não	Vagamente	Parcialmente, em alguns casos	Largamente	Completamente
As imagens estão nítidas (não estão pixeladas e	Não	Vagamente	Parcialmente, em alguns casos	Largamente	Completamente

distorcidas)					
Os ícones são fáceis de interpretar	Não	Vagamente	Parcialmente, em alguns casos	Largamente	Completamente
As telas respeitam o mesmo padrão em relação aos elementos de cor, imagem e tipografia (tamanho, família, estilo, peso)	Não	Vagamente	Parcialmente, em alguns casos	Largamente	Completamente
De modo geral o design visual está agradável e organizado	Não	Vagamente	Parcialmente	Largamente	Completamente
Total de pontos atingível: 48					

• *Teste*: o aprendizado do aluno também é testado no final do curso, com perguntas objetivas sobre conhecimentos nos assuntos abordados na unidade (Quadro 28).

Quadro 28: Questões aplicadas aos alunos da UI.

<p>1. A descrição “como Papai Noel eu preciso receber as cartas das crianças para que eu possa levar o presente de natal” é?</p> <p>(a) um programa de <i>app</i></p> <p>(b) uma história de usuário</p> <p>(c) uma caracterização das pessoas que vão utilizar o meu <i>app</i></p> <p>(d) o design visual do <i>app</i></p>
<p>2. Quais elementos compõem o design visual de um <i>app</i>?</p> <p>(a) Wi-Fi, som, jogo</p> <p>(b) cores, tipografia, ícones e imagens</p> <p>(c) sensor de localização, mapa</p> <p>(d) som, áudio, vídeo</p>
<p>3. O teste funcional serve para:</p> <p>(a) desenvolver um <i>app</i> mais rápido</p> <p>(b) programar um jogo</p> <p>(c) verificar se o <i>app</i> está apto a realizar todas as tarefas para as quais foi desenvolvido</p> <p>(d) deixar o <i>app</i> bonito</p>

4. Antes de programar o *app*, devemos fazer uma análise do usuário para:

- (a) identificar um problema na comunidade
- (b) entender as características de pessoas que vão utilizar o *app*
- (c) verificar se o *app* está correto
- (d) definir o design das telas

5. Descreva a história de alguma tarefa do seu *app* seguindo o modelo abaixo.

História: <Título da história/nome da tarefa>	
Como um(a)	<Papel/tipo de usuário>
eu preciso	<tarefa que o usuário irá realizar>
para que eu possa	<objetivo a ser atingido>

5. EVOLUÇÃO DO APP INVENTOR

Com o objetivo de suportar o ensino de computação no Ensino Fundamental no contexto de UIs, voltadas ao desenvolvimento de *apps* com *App Inventor*, foram identificadas várias oportunidades de melhoria no seu suporte. Assim, para suportar adequadamente o processo desenvolvido, foi preciso adaptado/evoluído a ferramenta *App Inventor*.

5.1. ANÁLISE DOS REQUISITOS

Os requisitos funcionais foram identificados visando atender aos objetivos de aprendizagem da unidade instrucional “Faça seu *app*” (seção 4.2.1). Esses requisitos levam em consideração tópicos relacionados à ES e EU, mais especificamente, ao design visual (Quadro 29), conforme o processo de desenvolvimento de *apps* definido na seção 4.3.

Quadro 29: Requisitos funcionais em relação a UI.

Requisito	Nome	Descrição	Objetivo de aprendizagem	Conteúdo educacional
R1	Menu de cores	Exibir o menu de cores em conformidade com as cores do <i>Material Design</i> na versão atual do <i>App Inventor</i> seguindo a proposta de SILVA (2017).	OA11	Design visual: cores
R2	Catálogo de ícones	Permitir selecionar um ícone a partir de um catálogo de ícones do <i>Material Design</i> no <i>App Inventor</i> .	OA11	Design visual: ícones
R3	Paleta de cores	Suportar passo-a-passo a escolha de uma paleta de cores, incluindo a criação de um painel semântico com uma foto, seleção automática da cor primária, opções de seleção de cor secundária com base no círculo cromático, e consequentemente a atualização automática das cores de todos os componentes	OA11	Design visual: cores

		(botão, <i>checkbox</i>) como opção padrão, possibilitando que o usuário ainda pode mudar manualmente se quiser.		
R4	Documentar persona	Suportar a documentação de persona.	OA3, OA5	Análise de contexto
R5	Documentar história de usuário	Suportar a documentação de história de usuário.	OA3, OA6	Especificação de requisitos

Alguns requisitos não funcionais (RNF) também foram levantados:

-RNF-1: Código fonte modificado/adicionado, deve ser integrado na versão nb171 do *App Inventor*;

-RNF-2: Apenas códigos na linguagem java devem ser adicionados ao projeto;

-RNF-3: A ferramenta deve ser compatível com os navegadores Google Chrome e Firefox (não incluso Internet Explorer);

-RNF-4: A paleta de cores deve ser compatível com o sistema de cores do *Material Design*²⁰.

Um detalhamento da implantação desses requisitos no *App Inventor* é apresentado para compreender melhor os requisitos identificados:

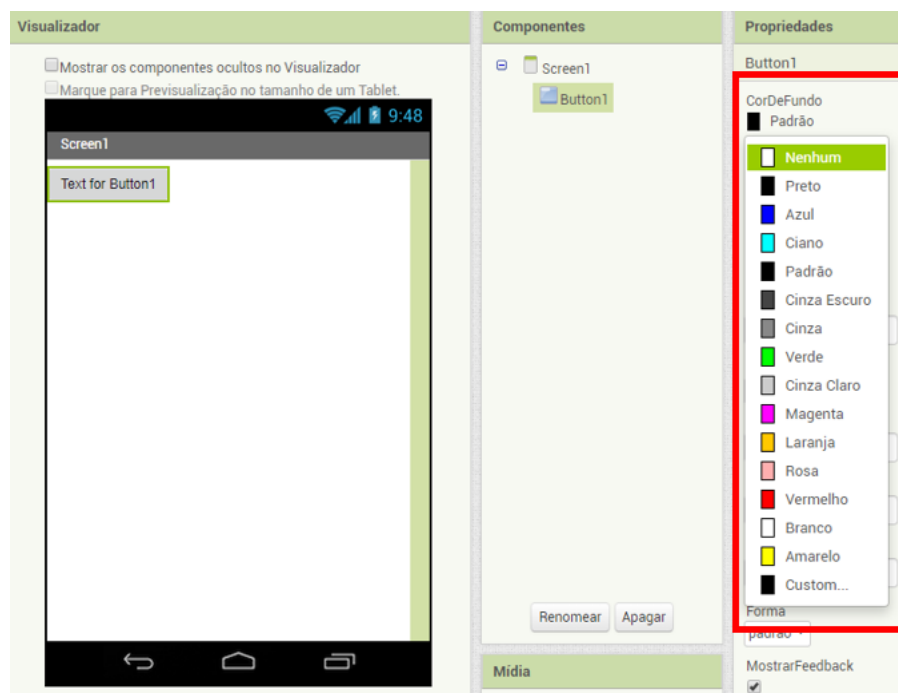
- **R1 – Menu de cores**

Atualmente, uma forma de escolher uma cor de um componente no *App Inventor* é via menu de cores (Figura 43). Neste menu existe a opção de escolher diretamente uma opção de cor ou por meio de um editor de cores customizado, onde o usuário apresenta o código RGB associado à cor desejada. Esta abordagem é inviável para o público alvo da unidade instrucional “Faça o seu *app*”, já que é preciso um conhecimento mais avançado em tópicos de computação para entender o conceito de formação de cores RGB, além de saber o código necessário para criar as cores customizadas. Sendo assim, o público alvo da unidade instrucional é limitado à utilização das cores padrão, disponíveis no seletor de cores, sendo que as

²⁰ <https://www.materialpalette.com/>

mesmas não estão de acordo com as orientações presentes no *Material Design*, *framework* de estilo visual mais utilizado para *apps Android* (SILVA, 2017).

Figura 43: Menu de cores do *App Inventor*.



Fonte: elaborada pelo autor com base em *appinventor.mit.edu*.

Dessa forma, faz-se necessário alterar as opções de cores disponíveis nos seletores de cores. Essas opções devem se basear nas orientações do *Material Design*, no qual é definida uma paleta de cores com 19 categorias e subcategorias que são tipicamente utilizadas em *apps Android* (Figura 44).

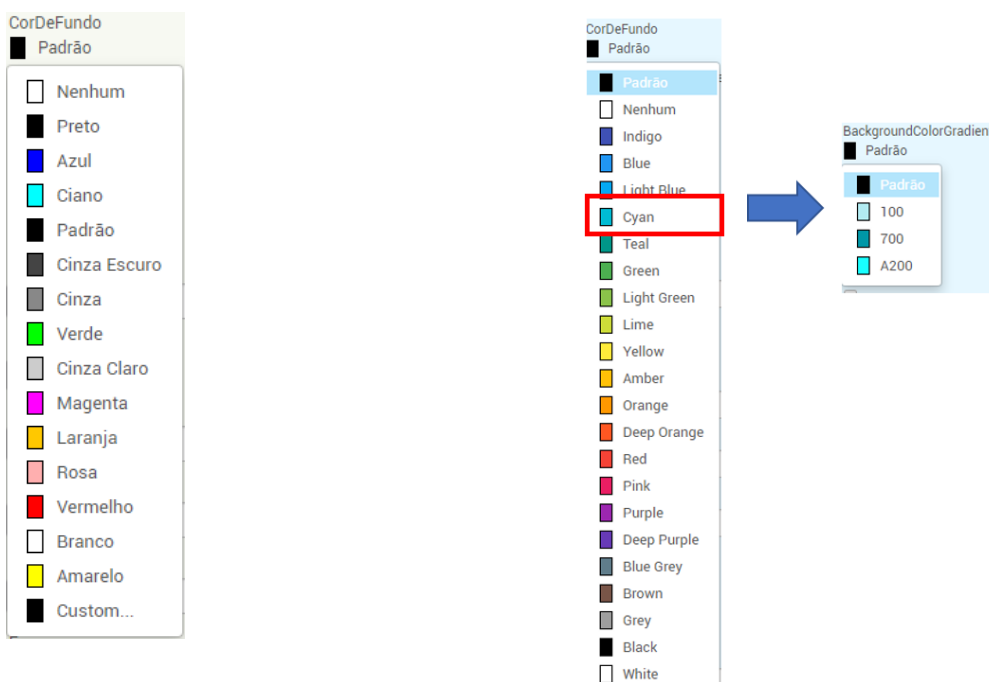
Figura 44: Paleta de cores do *Material Design*.



Fonte: GOOGLE (2019).

A paleta possui uma ampla quantidade de cores, portanto, colocar todas em uma única listagem de seleção implicaria na perda de usabilidade da ferramenta. Assim, um fluxo ideal 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 uma nova propriedade, na qual as variações de tons da cor principal selecionada serão mostradas (Figura 45).

Figura 45: Comparação do menu cor de fundo.



46.1 - Menu antigo

46.2 - Menu novo

Fonte: elaborada pelo autor.

Essa modificação já foi proposta e implementada previamente em um trabalho acadêmico (SILVA, 2017). Entretanto, por motivo de atualização de versão do *App Inventor* pelo MIT, necessitou-se a re-implementação dessa mudança.

- **R2 – Catálogo de ícone**

Ícones são imagens que nos dizem, de forma direta, o que alguma coisa significa, para que serve um objeto ou que tipo de conteúdo vamos encontrar pela frente (*Material Design*, 2019). Geralmente, os *apps* tem ícones para indicar sua função, tornando-se um fator importante para o design visual e experiência de usuário. Com o objetivo de padronizar ícones para *apps* que usam celulares

Android, o *Material Design* disponibiliza na *web* um catálogo de ícones, de diversos temas, categorias e tamanhos (Figura 46).



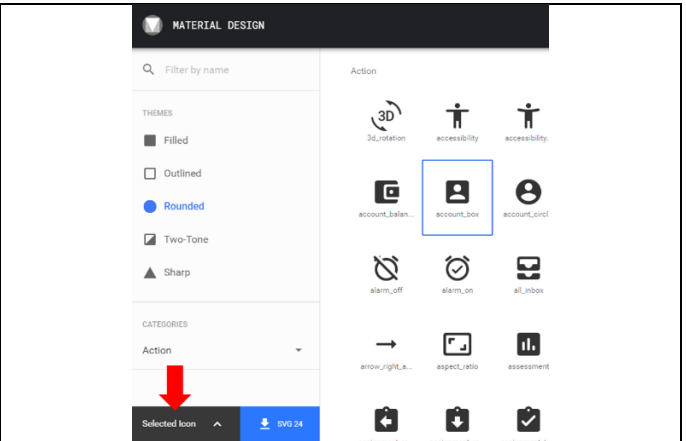
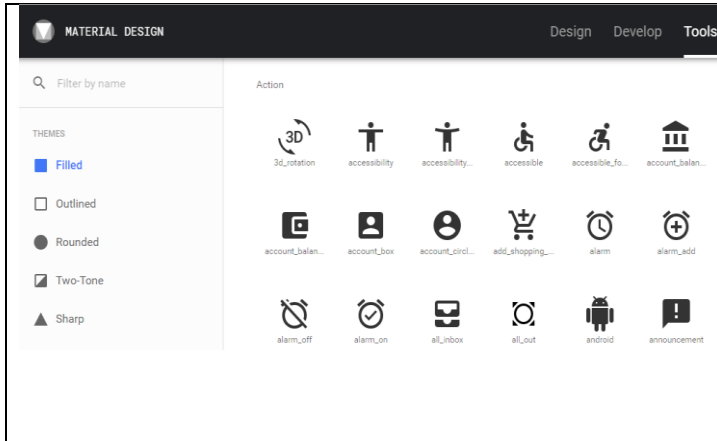
Fonte: GOOGLE (2019).

Na unidade instrucional “Faça seu *app*” os alunos aprendem a inserir esses ícones nos *apps* em desenvolvimento. No *App Inventor*, o ícone é uma propriedade que pode ser utilizada nos componentes “Botão” e “Imagem”. No entanto, alterar o ícone de um componente visual no *App Inventor* por um ícone do *Material Design* pode ser uma tarefa complexa. O Quadro 30 apresenta os passos necessários para incluir um ícone no componente “Botão”. Ensinar este procedimento para um público jovem gera distrações/desfoco dos alunos, e atrapalha o andamento do curso, além de ser demorado.

Quadro 30: Passos para incluir um ícone no Botão do *App Inventor*.

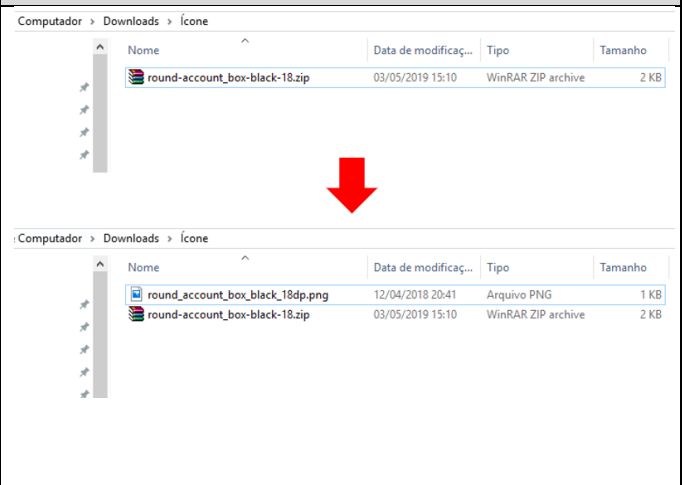
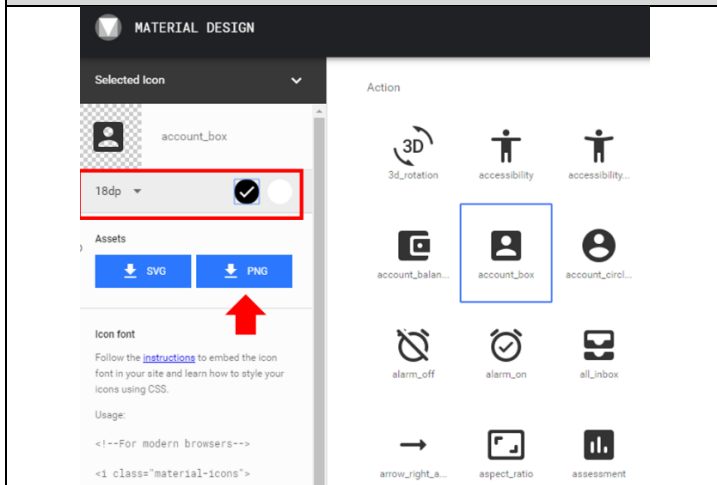
Passo 1: Realizar o download do ícone acessando, o link <https://material.io/>, e navegar até o menu *Tools* -> *Icons*.

Passo 2: Filtrar, por categoria e tema, selecionar o ícone desejado e clicar em *Select Icon*, para abrir a opção do ícone no formato *.png* (*App Inventor* não aceita *.svg*)



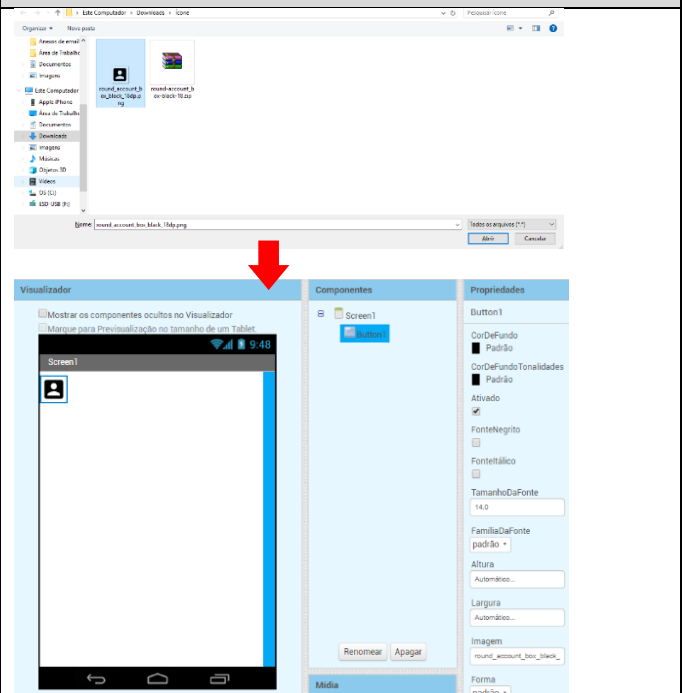
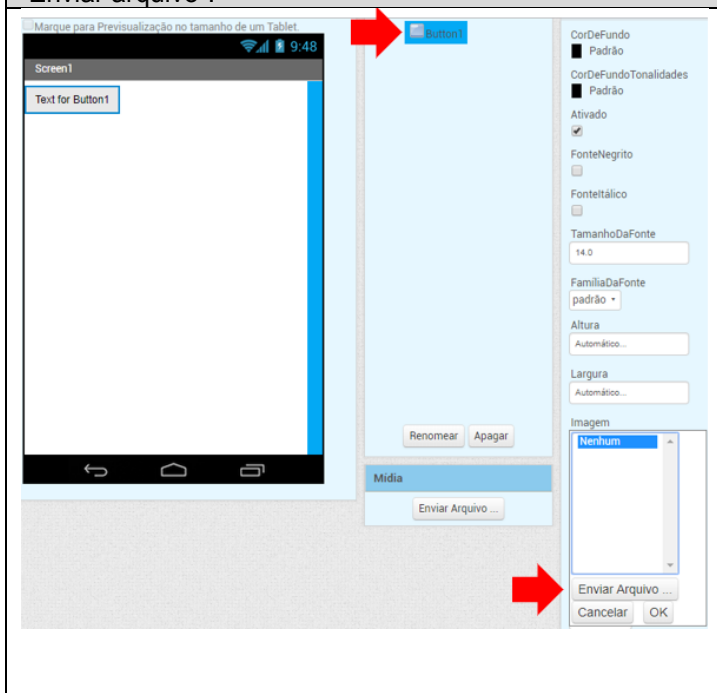
Passo 3: Filtrar o ícone por tamanho e cor, e clicar no botão PNG.

Passo 4: Realizar o *download* e descompactar o ícone.



Passo 5: No *App Inventor*, selecionar o componente "Botão" e, na propriedade "Imagem", clicar no botão "Enviar arquivo".

Passo 6: Selecionar o ícone armazenado no computador e clicar em "OK".



Dessa forma, existe a necessidade de implementar o catálogo de ícones do *Material Design* dentro da ferramenta, com o intuito de facilitar, simplificar e agilizar a inclusão de um ícone. O local mais viável para implementar a inserção é na propriedade “Imagem” dos componentes “Botão” e “Imagem”. Nesta propriedade, deve ser criada a possibilidade de acessar o catálogo dos ícones e incluí-lo no componente “Botão” ou “Imagem”.

- **R3 – Paleta de Cores**

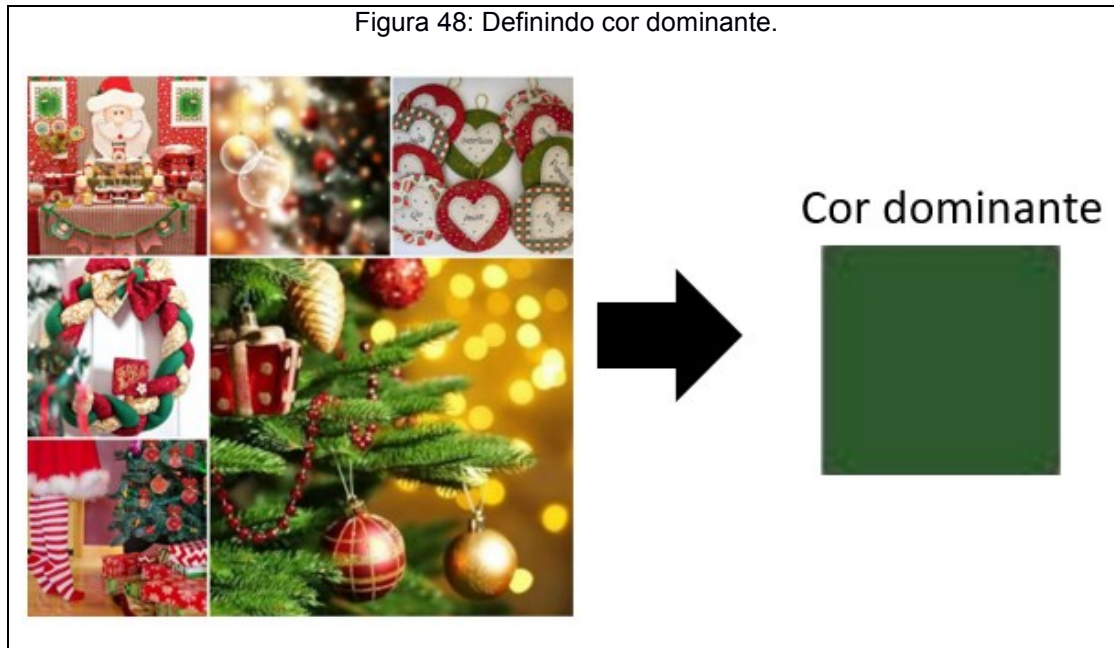
A definição da paleta de cores é essencial para o sucesso visual de um *app*. O *layout* e outras opções de design, incluindo fonte, devem ser desenvolvidos em conjunto com seu esquema de cores, o que pode garantir legibilidade, coesão e beleza no produto final. Porém, definir uma paleta de cores a partir do zero pode ser um grande desafio. Uma forma de gerar a paleta de cores é realizando o seguinte procedimento:

1) *Definir um painel semântico*: um painel semântico é tipicamente composto de várias imagens/fotos, para expressar sentimentos que remetem ao *app* (Figura 47).



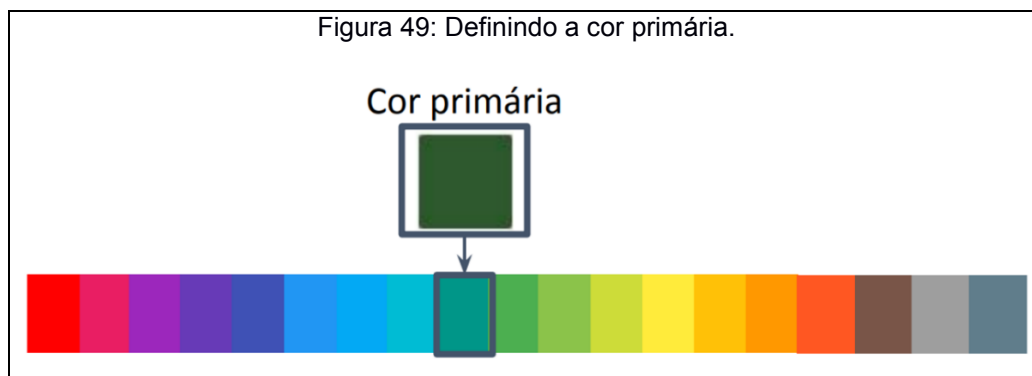
Fonte: elaborada pelo autor.

2) *Definir cor dominante*: a partir do painel semântico é selecionada a cor dominante, que é a cor do pixel com a maior quantidade na imagem do painel semântico (Figura 48).



Fonte: elaborada pelo autor.

3) *Definir cor primária*: o sistema de cores do *Material Design* define que uma paleta de cores, para um *app Android*, deve conter uma cor primária e uma cor secundária. A cor primária é a cor exibida com mais frequência nas telas e nos componentes do aplicativo. Neste procedimento, usa-se como proposta de cor primária a cor dominante aproximada a uma das 19 cores do *Material Design* (Figura 49).



Fonte: elaborada pelo autor.

4) *Definir opções de cores secundárias*: as cores secundárias são de uso opcional, e assumem o papel de destacar os elementos da sua interface gráfica. Seu uso mais indicado é nos elementos como botões (flutuantes ou textuais), campos de texto, cursores, seleção de texto, barra de progresso, *sliders*, *links* e cabeçalhos. Seguindo a teoria das cores (GOETHE & EASTLAKE, 2006), existem várias alternativas para selecionar uma cor secundária harmônica em relação à cor primária. Para isso, é preciso utilizar o círculo cromático do tipo RYB, que organiza matizes de cores em torno de um círculo e que ilustra a relação das cores a partir das cores vermelho, amarelo e azul, consideradas como as cores principais (WEGMAN & SAID, 2011). O RYB foi o sistema de cores utilizado no círculo cromático pois é o utilizado pelo sistema de cores do *Material Design*. Dentre estas alternativas de cores secundárias estão:

- **Cor complementar**: são as cores que estão posicionadas nas extremidades opostas do círculo cromático. Elas apresentam maior contraste comparadas à cor primária, por exemplo, as cores cerceta e a vermelho (Figura 50).

Figura 50: Cores complementares no círculo cromático.



Fonte: elaborada pelo autor.

- **Cores análogas**: cores análogas são as cores vizinhas à cor primária no círculo cromático, por exemplo, a cor verde e ciano são análogas à cor cerceta (Figura 51).

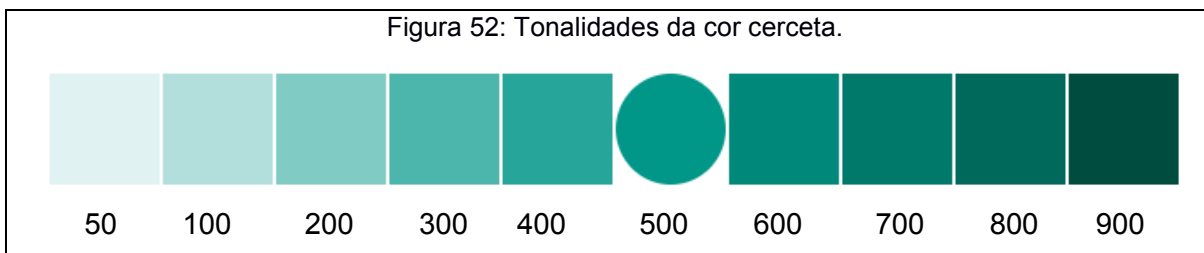
Figura 51: Cores análogas no círculo cromático.



Fonte: elaborada pelo autor.

- **Cor variante:** além das opções de cor secundárias, usando o círculo cromático, também pode-se utilizar uma tonalidade mais clara ou escura da cor primária como opção de cor secundária, para criar contrastes entre os elementos da interface de usuário (Figura 52). As tonalidades de cores estão disponíveis no sistema de cores do *Material Design*.

Figura 52: Tonalidades da cor cerceta.



Fonte: GOOGLE (2019).

5) *Inclusão das cores cinza, preto e branco:* além das cores primária e secundária, faz parte da paleta de cores o branco, o preto e o cinza, que podem definir a cor de alguns elementos, como textos, plano de fundo, ícone, etc.

Dessa forma, a partir de uma imagem, é possível implementar uma paleta de cores composta por uma cor primária, uma cor secundária, com 4 opções de cores, preto, branco e cinza (Figura 53).


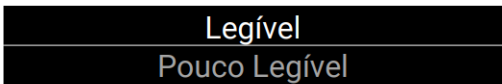


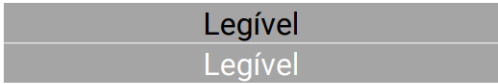
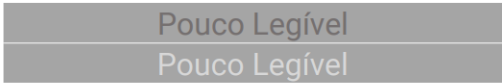




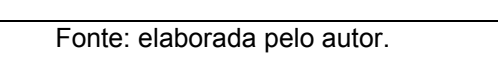
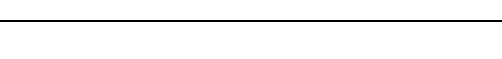
Figura 53: Paleta de cores.

Cor primária	Cores secundárias			Preto	Cinza	Branco	
	Variante	Complementar	Análoga 1 Análoga 2				
							

Fonte: elaborada pelo autor.

O sistema de cores do *Material Design* também define diretrizes para aplicar as cores da paleta em componentes de interface de usuário de um *app Android*, de maneira consistente e significativa. Seguindo essas diretrizes, a cor primária e a secundária podem ser aplicadas para cores dos componentes do *App Inventor*. Além das cores dos componentes, o *Material Design* também define diretrizes relacionadas ao contraste dos textos, visando melhorar a legibilidade e para ajudar os usuários a verem e interpretarem o conteúdo do seu aplicativo, interagir com os elementos certos e entender as ações (Figura 54).

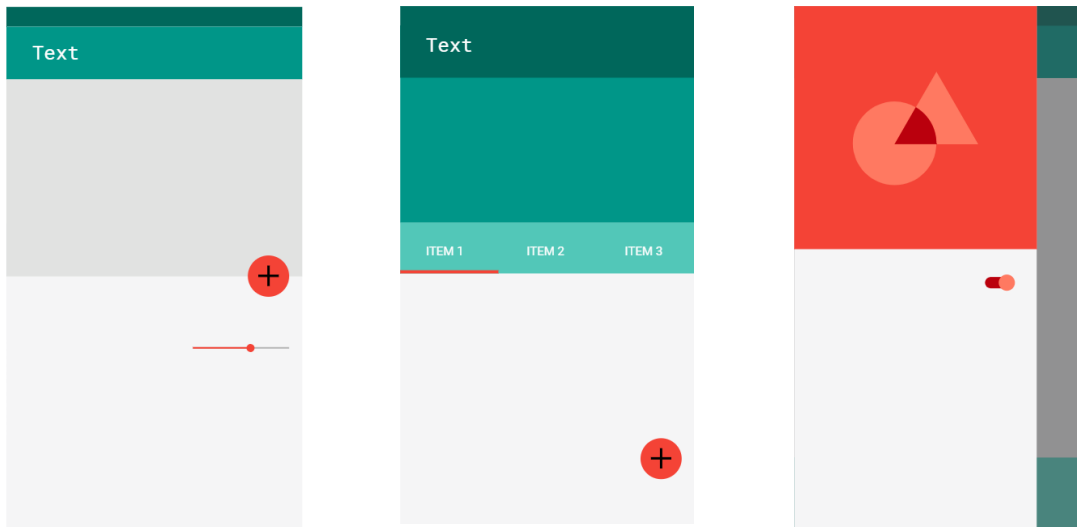
Figura 54: Contraste para textos.

Fonte: elaborada pelo autor.

Definir o texto do botão em caixa alta e textos longos alinhados à esquerda também é uma das diretrizes do *Material Design*. A figura 55 é um exemplo de uma interface de usuário, de uma tela inicial de um *app* com a cor primária cerceta e a cor secundária vermelha.

Figura 55: Tela com interface de usuário.



Fonte: elaborada pelo autor com base em GOOGLE (2019).

Abordar estas recomendações na unidade instrucional deste trabalho aumenta a usabilidade dos *apps* desenvolvidos pelos alunos, no que se refere ao design visual de cores. Assim, faz-se necessário aplicar essas diretrizes nos componentes do *App Inventor*. Para isso, foram levantados valores (Quadro 31) de algumas propriedades dos componentes mais utilizados do *App Inventor* (SOLECKI *et al.*, 2019; ALVES *et al.*, 2019) visando aplicar as seguintes diretrizes:

- Aplicar a paleta de cores nos componentes visuais.
- Definir automaticamente a cor do texto (preto ou branco) visando um melhor contraste com a cor de fundo.
- Colocar os textos dos botões em caixa alta.
- Deixar os textos longos alinhados à esquerda.

Assim, com o intuito de dar suporte ao ensino de cores do app, faz-se necessário o desenvolvimento de uma funcionalidade que automatize o processo de definição de uma paleta de cores.

Quadro 31: Definição de cores para elementos e componentes do *App Inventor*.

Componente	Cor de Fundo	Cor de texto	Cor do texto do item	Cor de seleção	Texto	Alinhamento do texto
Legenda	Sem cor	Branco/preto	-----	-----	-----	Se a quantidade de caractere ≥ 40 , alinhar a esquerda, senão centralizado
Botão	Cor	Branco/preto	-----	-----	Caixa	Centro

	secundária				alta	
Caixa de texto	Branco	Preto	-----	-----	-----	-----
Selecionador de tempo	Cor secundária	Branco/preto	-----	-----	-----	-----
Lista suspensa	Cor secundária	Branco/preto	Cor primária	Branco	-----	-----
Caixa de seleção	Sem cor	Branco/preto	-----	-----	-----	-----
Tela	Branco	-----	-----	-----	-----	-----
Alinhamento horizontal (função de Toolbar)	Cor primária	-----	-----	-----	-----	-----
Alinhamento horizontal	Sem cor	-----	-----	-----	-----	-----
Alinhamento vertical	Sem cor	-----	-----	-----	-----	-----

Fonte: elaborado pelo autor.

- **R4 – Documentar persona**

Como resultado da análise de contexto (público alvo) no desenvolvimento de *app*, os alunos devem caracterizar categorias dos usuários finais por meio de *personas* (Figura 56).

Figura 56: Modelo de uma *persona*.

The image shows a hand-drawn user persona card on a dark background. At the top, the name 'Dawson, Guilherme' is written in blue ink. Below the name is a simple line drawing of a person's head and shoulders wearing a hat and sunglasses. To the right of the drawing, the text 'CARACTERIZE SEUS USUÁRIOS' is printed in white. The card is divided into several sections with handwritten text in blue ink:

- Biografia:** 'Ele mora na capital de Rio grande do Sul tem 2 filhos e uma mulher'.
- Qual seu objetivo?:** 'Curtir as férias com sua família'.
- O que gosta?:** 'gosta de sair com a família nos finais'.
- O que não gosta?:** 'Não gosto de lotação'.
- Conhecimento de TI:** 'Ele usa Smartphone Android e ele já tem um cartão com'.
- TI e Internet:** 00000
- Software:** 00000
- Apps móveis:** 00000
- Redes sociais:** 00000

At the bottom of the card, there is a small line of text: 'INICIATIVA COMPUTAÇÃO NA ESCOLA - CREATIVE COMMONS ATRIBUIÇÃO NÃO COMERCIAL - COMPARTILHAMENTO E HABILIDADES 4.0 INTERNACIONAL'.

Fonte: elaborada pelo autor.

Na ferramenta *App Inventor* ainda não existe uma funcionalidade para o aluno caracterizar as *personas* do seu *app*. Desta forma, para acompanhar melhor o processo de desenvolvimento de *apps*, faz-se necessária a criação de uma

funcionalidade na ferramenta *App Inventor*, que possibilite aos alunos documentar uma ou mais *personas*. Esta funcionalidade deve possibilitar inserir informações como: biografia, objetivo, o que gosta, o que não gosta, experiência no uso de software ou *app*, etc.

- **R5 – História de usuário**

Um dos artefatos gerados pelos alunos, como resultado de uma análise das tarefas do *app* em desenvolvimento, é a história de usuário. História de usuário é um método de levantamento de requisitos, no qual são descritos, na linguagem cotidiana do usuário final, os requisitos de software (COHN, 2004). Na ferramenta *App Inventor* ainda não existe uma funcionalidade para o aluno descrever histórias de usuário do seu *app*. Assim, faz-se necessária a criação de um formulário que possibilite aos alunos documentar histórias de usuário no *App Inventor*, conforme o modelo ilustrado na Figura 57.

Figura 57: Modelo de história de usuário.

História:	Ele veio passar as férias em Florianópolis	
Como um(a)	Turista	
eu preciso	Saber como chegar nas Praia.	
para que eu possa	Curtir minhas férias.	

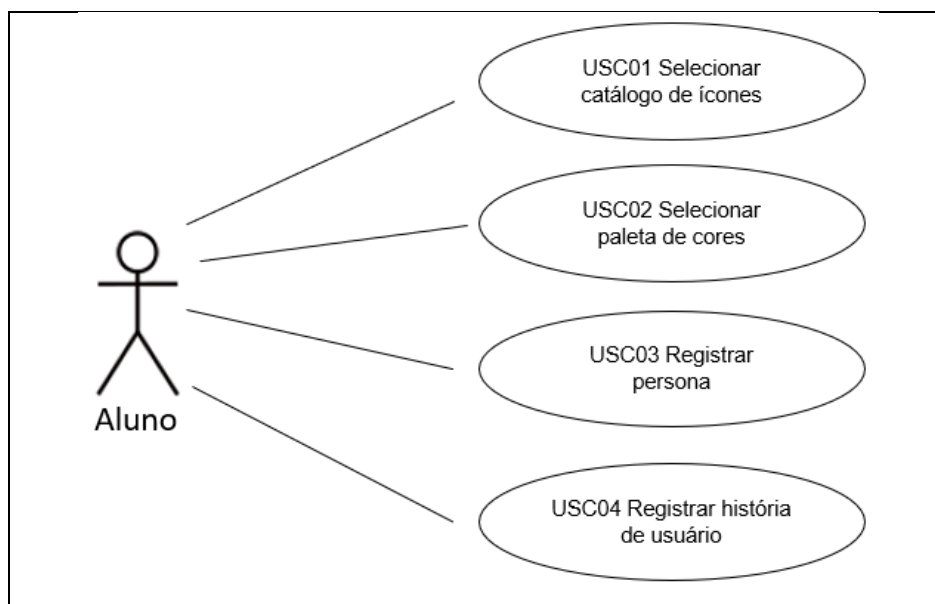
Fonte: elaborada pelo autor.

5.2. CASOS DE USO

Analisando os requisitos funcionais, foram elaborados os casos de uso do sistema, bem como seus principais fluxos de execução. O aluno é o único ator que interage com todas as funcionalidades.

- **Definição do ator:** [A01] Aluno – Seleciona um ícone para o botão no USC01. Faz upload de uma imagem e seleciona uma cor secundária da paleta de cores no USC02. Também descreve uma história de usuário e uma persona no USC03 e USC04 respectivamente (Figura 58).

Figura 58: Diagrama de uso.



Fonte: elaborada pelo autor.

O requisito “R1 – Menu de cores” e sua implementação são detalhados no trabalho de SILVA (2017), pois o presente trabalho limitou-se à re-implementação seguindo o seu modelo.

- **Caso de Uso 1 – USC01 Selecionar catálogo de ícone**

-Ator: Aluno

-Pré-condição: O aluno deve ter feito o login na ferramenta *App Inventor*.

-Fluxo principal:

- 1) Aluno insere um botão na interface de tela do celular;
- 2) Aluno clica na propriedade Imagem/Ícone do botão;
- 3) Aluno clica no botão “Ícone”;
- 4) O sistema mostra o catálogo de ícone;
- 5) Aluno filtra o catálogo de ícone;
- 6) Aluno seleciona um ícone;
- 7) Aluno clica em adicionar;
- 8) Sistema altera plano de fundo do botão para a imagem do ícone.

-Fluxo alternativo (FA):

FA01 - Aluno clica no botão “Ok” sem selecionar um ícone. Sistema apenas fecha a tela.

- **Caso de Uso 2 – USC02 Selecionar paleta de cores**

-Ator: Aluno

-Pré-condição: O aluno deve ter feito o login na ferramenta *App Inventor* e criado/importado um *app*.

-Fluxo principal:

- 1) Aluno acessa o menu Ferramentas -> Paleta de cores;
- 2) O sistema apresenta a tela de paleta de cores;
- 3) Aluno faz upload de uma imagem que lembre o *app* desenvolvido;
- 4) Aluno clica no botão “Enviar”;
- 5) O sistema apresenta a paleta de cores e as opções de cores secundárias;
- 6) Aluno seleciona uma cor secundária da paleta de cores ou do círculo cromático;
- 7) Aluno clica no botão “Concluir”;
- 8) O sistema fecha a paleta e aplica as cores da paleta nos componentes do *app*.

-Fluxo de exceção (FE):

FE01 – Aluno seleciona uma imagem com tamanho maior que 5 megabytes. Sistema informa a mensagem: “A imagem selecionada deve ser menor que 5 megabytes”;

FE02 – Aluno clica em concluir sem selecionar uma opção de cor secundária e o sistema informa a mensagem: “Complete a paleta de cores selecionando uma opção de cor secundária”.

• Caso de Uso 3 – USC03 Registrar persona

-Ator: Aluno

-Pré-condição: O aluno deve ter feito o login na ferramenta *App Inventor*.

-Fluxo principal:

- 1) Aluno acessa o menu Ferramentas -> Persona;
- 2) O sistema apresenta a tela de persona;
- 3) Aluno faz *upload* de uma imagem que representa a persona clicando no botão “Escolher arquivo”;
- 4) Aluno preenche o formulário com as características da persona;
- 5) Aluno clica no botão “Incluir/Editar”;
- 6) Aluno clica no botão “Salvar persona(s)”.

-Fluxo de exceção (FE):

FE01 – Aluno seleciona uma imagem com tamanho maior que 5 megabytes. Sistema informa a mensagem: “A imagem selecionada deve ser menor que 5 megabytes”.

-Fluxo alternativo (FA):

FA01 – Aluno clica no botão “Criar persona”, caso deseje documentar mais uma persona;

FA02 – Aluno clica no botão “Cancelar”, caso deseje cancelar a operação

• Caso de Uso 4 – USC04 Registrar história de usuário

-Ator: Aluno

-Pré-condição: O aluno deve ter feito o login na ferramenta *App Inventor*.

-Fluxo principal:

- 1) Aluno acessa o menu Ferramentas -> História de usuário;
- 2) O sistema apresenta a tela de história de usuário;
- 3) Aluno preenche o formulário com a história de usuário;

- 4) Aluno clica no botão “Incluir/Editar”;
- 5) Aluno clica no botão “Salvar *história(s)*”.

-Fluxo alternativo (FA):

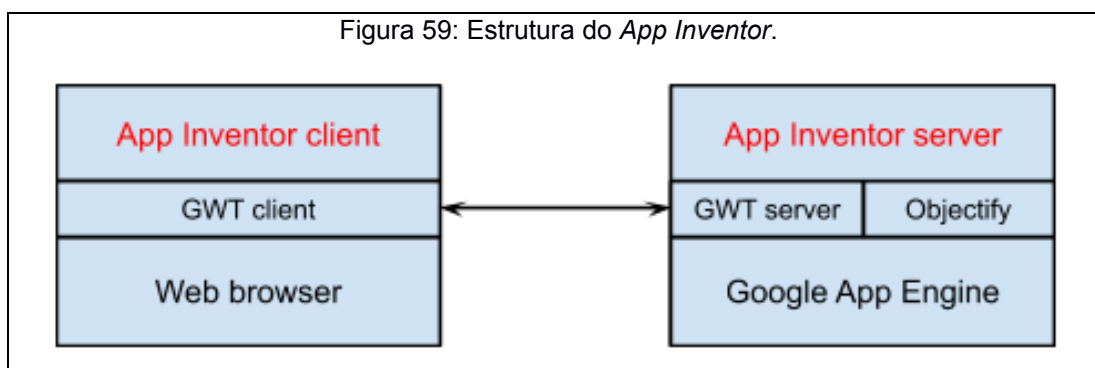
FA01 – Aluno clica no botão “Criar história”, caso deseje documentar mais uma história de usuário;

FA02 – Aluno clica no botão “Cancelar”, caso deseje cancelar a operação.

5.3. CONTEXTO DO FUNCIONAMENTO DO *APP INVENTOR*

A partir da versão nb171 do *App Inventor* do MIT (código aberto) foi criada uma versão, hospedada em um servidor da Computação na Escola (CNE, 2019), na UFSC.

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* Google Web Toolkit (GWT²¹) para a tradução do código do *frontend*. Outra tecnologia utilizada é o *Google App Engine*²², que é uma plataforma de *cloud computing* que permite que programas Java rodem e mantenham dados nos servidores do Google. Essas duas tecnologias desenvolvidas pelo Google formam a estrutura do *App Inventor* (Figura 59). O cliente e o servidor são criados utilizando GWT, que converte o código de *frontend* para Javascript, e este 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*²³ para persistir dados no sistema de arquivos distribuídos.



Fonte: MIT & GOOGLE (2019f)

²¹ www.gwtproject.org

²² <https://cloud.google.com/appengine/>

²³ <https://github.com/objectify/objectify/wiki>

O *App Inventor* é um sistema de código aberto, o que possibilita realizar alterações e customizações. Para isso, primeiramente é preciso realizar a configuração do ambiente de desenvolvimento.

5.4. CONFIGURAÇÃO DO AMBIENTE DE DESENVOLVIMENTO

A visualização e edição dos projetos do *App Inventor* foram realizadas utilizando o sistema operacional *Windows 10* e a *Integrated Development Enviroment* (IDE) Eclipse²⁴, que é voltado para desenvolvimento de softwares em Java. Essas escolhas foram realizadas por questões de familiaridade e gosto pessoal. O código fonte pode ser encontrado no repositório público do *App Inventor* no *Github*²⁵. Para rodar o *App Inventor* localmente, também foi preciso a instalação de outros softwares na seguinte ordem:

- 1) Instalar o Git Client²⁶;
- 2) Instalar a versão mais atual do Ant²⁷;
- 3) Instalar JDK 1.7²⁸;
- 4) Instalar o Python²⁹;
- 5) Instalar a versão mais atual do *Android* SDK³⁰;
- 6) Instalar o plugin eclipse *AppEngine*³¹.

Para carregar o código fonte do *App Inventor* no Eclipse e para poder compilar, é necessário primeiramente abrir um terminal (indicado o uso do Git Bash para usuários Windows) e realizar, em ordem, a execução dos seguintes comandos:

- 1) `git clone https://github.com/mit-cml/appinventor-sources.git`
- 2) `git submodule update --init`
- 3) `ant clean` (no diretório *appinventor*)
- 4) `ant MakeAuthKey`
- 5) `ant`
- 6) `mkdir bin` (no diretório *appinventor-sources*)
- 7) Abrir o Eclipse

²⁴ www.eclipse.org

²⁵ <https://github.com/>

²⁶ <https://git-scm.com/downloads>

²⁷ <https://ant.apache.org/bindownload.cgi>

²⁸ <http://www.oracle.com/technetwork/pt/java/javase/downloads/jdk7-downloads-1880260.html>

²⁹ <https://www.python.org/downloads>

³⁰ <https://developer.Android.com/studio/index.html>

³¹ <https://dl.google.com/eclipse/plugin/4.5> <https://developers.google.com/eclipse/docs/install-eclipse-4.5>

8) File>New>Java Project

9) Em "Use default Location" selecionar a pasta *appinventor-sources* e NEXT

10) Em "Default Output Folder": colocar a pasta "*appinventor-source\bin*"

11) Em "Order and Export" mover as duas ocorrências de "*Android.jar*" para o final da lista

-Observação: 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").

Por fim, para que seja rodado o sistema localmente, é necessário rodar o servidor de aplicação executando o seguinte comando:

```
>C:\caminhodapastadeinstalaçãodoappengine\bin\dev_appserver --
port=8888 --adress=0.0.0.0
>C:\caminhodapastaclonedoprojeto\appinventor\appengine\build\war
```

Para acessar o *App Inventor* abrir de preferência o navegador Google Chrome, e digitar a url: <http://localhost:8888>.

5.5. ESTRUTURA DO CÓDIGO DO 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*:

1) **Aiphoneapp**: interpretador que roda no dispositivo móvel ou no emulador.

2) **Aiplayapp**: outra parte do interpretador que roda no dispositivo móvel, esse diretório é responsável pelo chamado MIT *App InventorApp Inventor Companion*;

3) **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

4) **Blockyeditor**: código que representa a parte de edição de blocos (montagem da lógica do *app*) do *App Inventor*;

5) **Buildserver**: servidor/*servlet* http que transforma o .zip em um .apk;

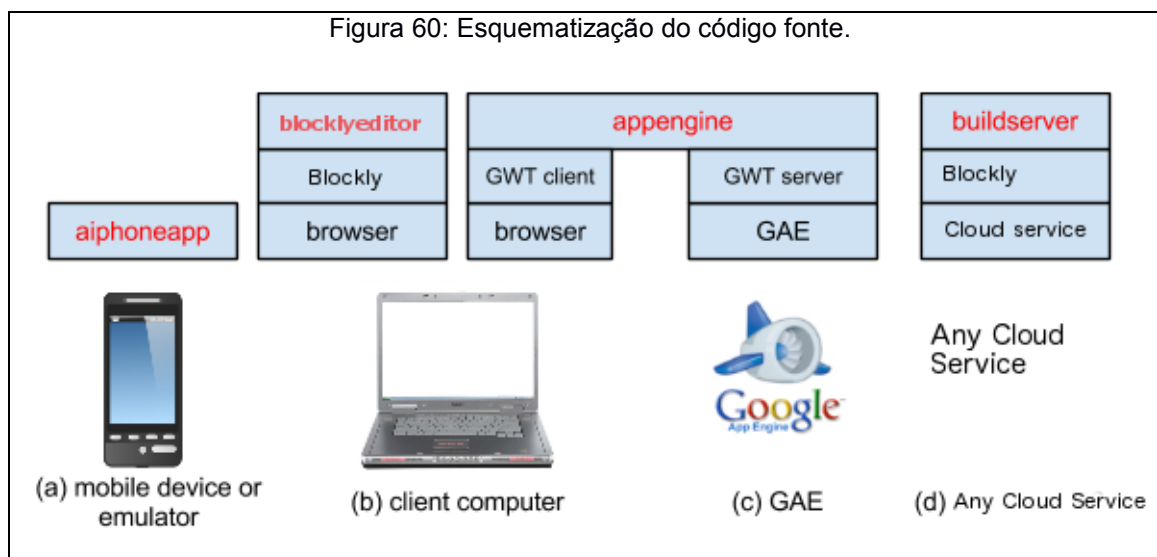
6) **Common**: constantes e classes utilitárias usadas pelos outros subprojetos;

7) **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;

8) **Docs**: documentação;

9) **Lib**: bibliotecas externas ao projeto e que servem de apoio como JUnit;

10) Cada subprojeto roda dentro de um contexto que pode ser representado em alto nível na Figura 60. As partes em vermelho representam a camada interna do *App Inventor* enquanto partes em preto representam infraestruturas externas auxiliares ao *App Inventor*.



Fonte: MIT & GOOGLE (2019f)

5.6. IMPLEMENTAÇÃO

As funcionalidades foram implementadas utilizando a linguagem de programação Java e o *framework Google Web Toolkit (GWT)*. Essas tecnologias foram escolhidas por já serem utilizadas e integradas no sistema atual, e o aluno ter conhecimento prévio da linguagem Java. A seguir são apresentadas as implementações por caso de uso.

- **USC01 – SELECIONAR CATÁLOGO DE ÍCONES**

Para desenvolver o catálogo de ícones primeiro foi preciso realizar o *download* dos ícones, disponibilizado pelo *Material Design* por meio do repositório

GitHub³². Esses ícones foram inseridos dentro do projeto do *App Inventor* na pasta `C:\caminhodapastaclonedoprojeto\appinventor\appengine\war\ícones`. Para acessar esses ícones e apresentá-los em forma de catálogo, foi preciso editar as classes do quadro 32.

Quadro 32: Classes editadas.

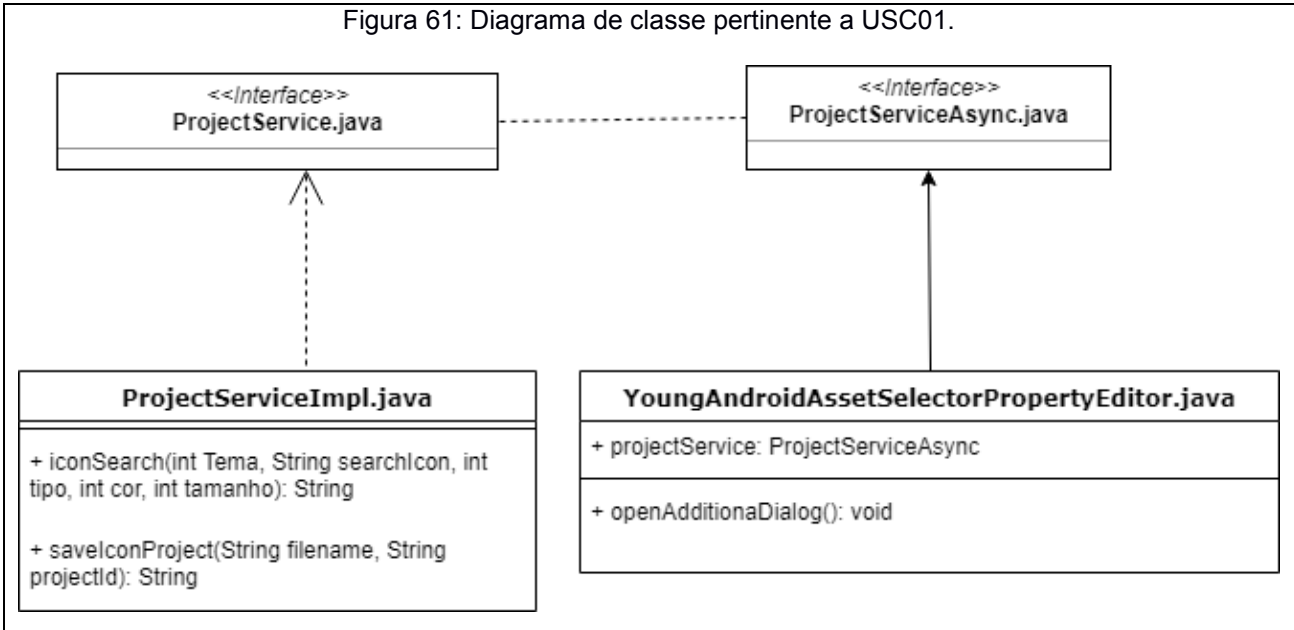
YoungAndroidAssetSelectorPropertyEditor.java
Essa classe foi editada para incluir o botão “Ícones” na propriedade “Imagem” dos componentes “Botão” e “Imagem”. Nesta classe também foi implementada a tela de catálogo de ícones. Como esta classe fica no lado cliente do sistema, foi preciso realizar uma chamada de procedimento remota (RPC) para acessar os ícones no servidor.
ProjectServiceImpl.java
O código do lado servidor que é chamado a partir do cliente é conhecido como um serviço. Assim, nesta classe foi implementado o serviço RPC executado no servidor para retorna os ícones.
ProjectService.java
Classe responsável por definir as interfaces dos serviços de um projeto. Assim, foi implementado a interface para acessar o ícone.
ProjectServiceAsync.java
Classe responsável por definir uma interface assíncrona, para que os serviços sejam chamados a partir do cliente.

Fonte: elaborado pelo autor.

A classe `YoungAndroidAssetSelectorPropertyEditor.java` realiza uma chamada remota por meio da interface assíncrona. A Figura 61 apresenta um diagrama de classe para apresentar as conexões entre as classes envolvidas.

³² <https://github.com/google/material-design-icons>

Figura 61: Diagrama de classe pertinente a USC01.

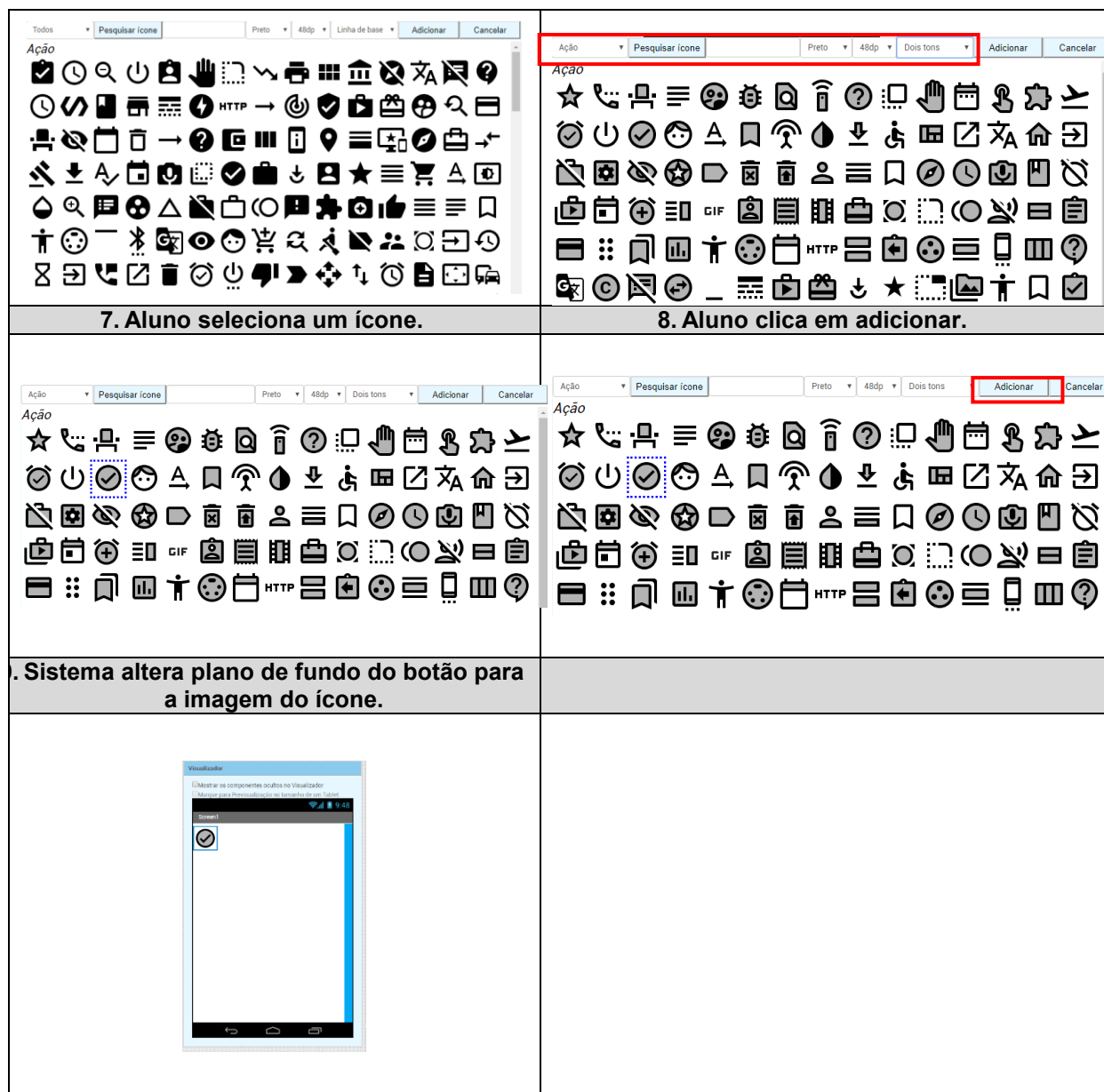


Fonte: elaborada pelo autor.

Com a edição dessas classes, foi implementado o requisito R2 – Catálogo de ícones (Quadro 33). Essas alterações simplificou o acesso e inclusão de ícones.

Quadro 33: Fluxo principal para selecionar um ícone.

<p>1. Aluno acessa a ferramenta App Inventor.</p> 	<p>2. Aluno insere um botão na interface de tela do celular.</p> 
<p>3. Aluno clica na propriedade Imagem/Ícone do componente botão.</p> 	<p>4. Aluno clica no botão “Ícone”.</p> 
<p>5. Sistema mostra o catálogo de ícone.</p>	<p>6. Aluno filtra o catálogo de ícone por tipo, cor, tamanho e tema.</p>



- **USC02 – SELECIONAR PALETA DE CORES**

O desenvolvimento da paleta de cores seguiu o procedimento proposto para o requisito R3 – Paleta de cores, da seção 6.2.1.

1. *Definir um painel semântico:* foi desenvolvido um painel semântico, no qual é possível inserir uma imagem relacionada ao *app*. Para isso, foi utilizada a classe `FileUpload.java` do framework GWT. Esta classe permite ao usuário selecionar uma imagem do seu computador e enviá-la para o servidor do sistema. Por restrições da plataforma *Google App Engine* o sistema aceita apenas imagens com extensões `.png`;

2. *Definir cor dominante*: foi desenvolvido um método que itera sobre cada pixel e conta quantas vezes cada cor está contida. A cor dominante é a cor que tiver maior quantidade de pixels. Esta implementação foi desenvolvida no método `retornaCorPredominante` da classe `UploadServlet.java`;

3. *Definir cor primária*: foi desenvolvido um método para aproximar a cor dominante com uma das 19 cores do *Material Design*. Esta aproximação baseou-se na fórmula de distância euclidiana, para calcular a distância entre dois pontos (neste caso, duas cores). Sabendo que a cor pode ser representada pela tupla RGB (*Red, Green, Blue*), pode-se definir que cada elemento da tupla é uma dimensão linear que define o espaço das cores. Dessa forma, a distância entre dois pontos pode ser representada pela fórmula (1):

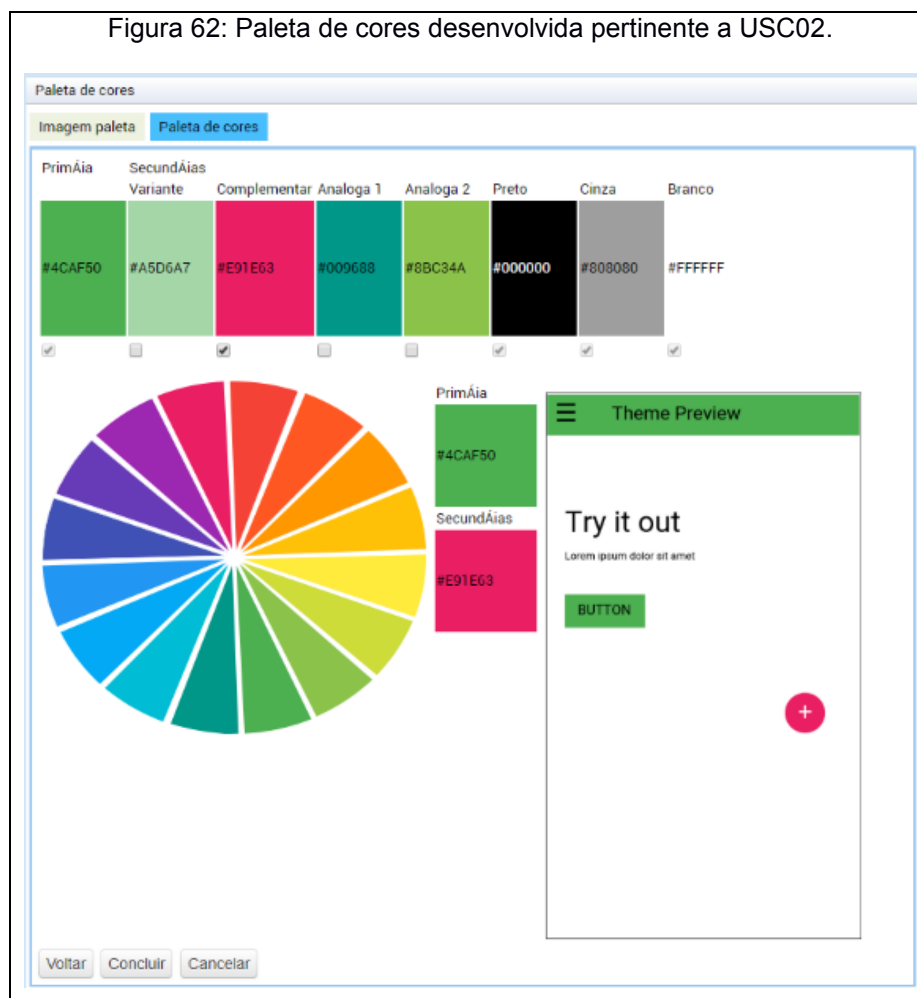
$$\text{Distância}^2 = (R_2 - R_1)^2 + (G_2 - G_1)^2 + (B_2 - B_1)^2 \quad (1)$$

Assim, a cor do *Material Design* que retornar à menor distância em relação à cor dominante é considerada a cor primária. Esta fórmula foi implementada no método `retornaCorSimilar` da classe `UploadServlet.java`.

4. *Definir opções de cores secundárias*: para cada cor primária são definidas 4 opções de cores secundárias (variante, complementar e duas análogas), por meio de um mapeamento. A cor complementar e as duas análogas foram definidas utilizando o círculo cromático. A cor variante foi utilizada na tonalidade 100 (mais clara) da cor primária. Essa tonalidade foi retirada do sistema de cores do *Material Design*.

5. *Cores cinza, preto e branco*: essas cores são fixas para qualquer paleta.

O resultado deste procedimento é uma paleta de cores com uma cor primária e 4 opções de cores secundárias, as quais o usuário deve escolher, e as cores preto, branco e cinza. Além dessas 4 opções de cores secundárias, o usuário pode escolher qualquer outra cor do *Material Design* por meio de um círculo cromático. Conforme o usuário seleciona a cor secundária desejada, é apresentada de forma dinâmica um exemplo de uma tela inicial de um *app*, com alguns componentes de tela coloridos, conforme as cores definidas na paleta (Figura 62). O círculo cromático e essa tela fazem parte de um conjunto de imagens previamente criadas e inseridas dentro do projeto do *App Inventor*, na pasta `C:\caminhodapastaclonedoprojeto\appinventor\appengine\war\ColorWheel`.



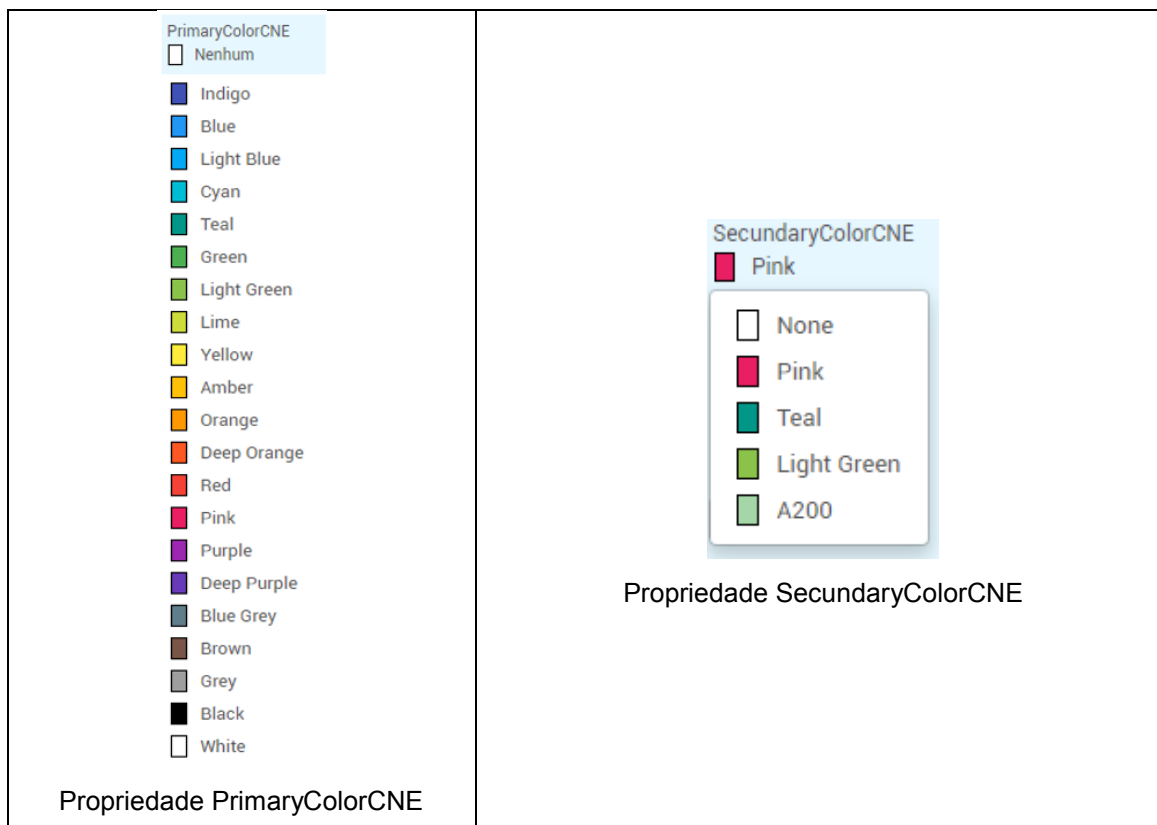
Fonte: elaborada pelo autor.

Definida a paleta de cores, o usuário poderá aplicar as cores definidas na paleta nos componentes, conforme definido no quadro 31 da seção 5.1. Para implementar esta aplicação, foi preciso alterar os componentes visuais (botão, toolbar, legenda, etc.) do ambiente *web* do *App Inventor*. Cada um desses componentes visuais possui uma classe que define seus atributos. O nome desta classe por padrão segue o formato “MocknomeComponente.java”. Por exemplo, o *MockButtonBase.java* é a classe que define os atributos para criar um botão na tela do celular da interface *web*. Assim, para alterar a cor de fundo do componente botão, é necessário chamar o método *setBackgroundModelProperty* da classe *MockButtonBase.java*. Além da cor de fundo desses componentes, foram alterados os métodos responsáveis por alterar a cor de texto, visando melhorar o contraste.

As informações de cores primária e secundária da paleta de cores foram armazenadas por meio de duas propriedades criadas no componente *Screen*:

PrimaryColorCNE e SecondaryColorCNE. A propriedade PrimaryColorCNE contém um menu com as 19 cores do *Material Design*. A propriedade SecondaryColorCNE contém um menu de cores com as 4 opções de cores secundária. Com essas propriedades, os alunos podem alterá-las caso queiram e mudar o valor das cores primária ou secundária (Figura 63).

Figura 63: Menu de cores das propriedades PrimaryColorCNE e SecondaryColorCNE.



Fonte: elaborada pelo autor.

A criação dessas propriedades foi realizada modificando as classes Form.java, para criar as propriedades, e a classe MockForm.java, para alterar as propriedades visuais do componente. O quadro 34 mostra mais detalhes sobre classes específicas que foram modificadas.

Quadro 34: Classes editadas.

TopToolbar.java
Classe responsável por criar os menus, submenus e suas respectivas telas do <i>App Inventor</i> . Neste requisito, foi implementado o menu “Ferramentas->Paleta de cores”. Além disso, nesta classe também foi implementada a interface de usuário da tela e todos as funcionalidades pertinentes ao

<p>processo de definição da paleta, como: definir a paleta, setar os valores das propriedades PrimaryColorCNE e SecondaryColorCNE no componente Screen, aplicar a cor nos componentes das telas do <i>app</i>, enviar a imagem que representa o <i>app</i> para o servidor.</p>
UploadServlet.java
<p>Esta classe é um servlet que recebe requisições HTTP e é responsável por receber requisições do cliente para realizar upload de arquivos de um projeto do <i>App Inventor</i>. Neste requisito, a classe foi utilizada para receber e armazenar a imagem que representa o <i>app</i>. Além disso, nesta classe foram implementados os métodos para definir a paleta de cores, como retornarCorSimilar e retornaCorPredominante.</p>
MockVisibleComponent.java
<p>Classe abstrata, que possui métodos que rastreiam mudanças em propriedades e as processam. Os métodos <code>onPropertyChange</code>, <code>fireColorChangeEvent</code> e <code>fireSecondaryColorChange</code> foram customizadas, para que quando cor primária for alterada na paleta de cores essa classe seja invocada para realizar a substituição da lista das 4 cores disponíveis na propriedade <code>secondaryColorCNE</code>.</p>
PropertiesUtil.java
<p>Classe responsável por iniciar as propriedades dos componentes e instanciá-las. Portanto foi necessário modificar o método <code>populatePorperties</code>, para quando um componente for criado e a paleta de cor estiver definida, atribuir a cor primária e secundária para as respectivas propriedades e componentes conforme definida no quadro 31.</p>
SettingsConstants.java
<p>Classe auxiliar para retornar string constantes que são utilizadas em diferentes lugares da aplicação. Foram adicionadas duas constantes com o nome das propriedades <code>primaryColorCNE</code> e <code>secondaryColorCNE</code>.</p>

Fonte: elaborada pelo autor.

Também foram criadas classes auxiliares, disponíveis no pacote `com.google.appinventor.client.editor.youngAndroid.properties.gradient.color.utils` (Quadro 35).

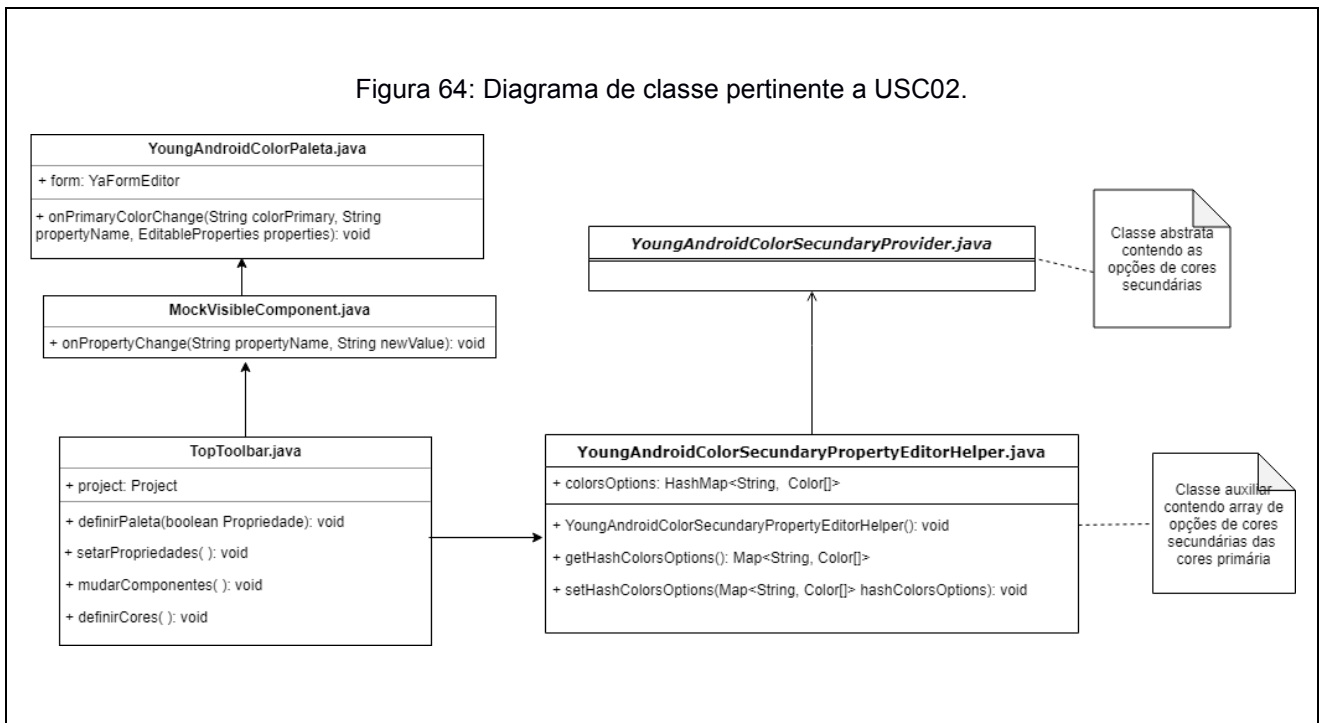
Quadro 35: Classes criadas.

YoungColorPaleta.java
<p>Classe responsável por instanciar e configurar as propriedades <code>primaryColorCNE</code> e <code>secondaryColorCNE</code>.</p>
YoungAndroidColorSecondaryProvider.java
<p>Classe abstrata, que define constantes utilizadas na configuração dos grupos de cores que a propriedade <code>secondaryColorCNE</code> pode ter. Criada apenas com o propósito de melhor organização de código.</p>
YoungAndroidColorSecondaryPropertyEditorHelper.java

Classe criada para auxiliar na obtenção do array de opções de cores secundárias que uma determinada cor primária pode ter. A classe possui uma variável do tipo HashMap onde a chave do mapeamento é o hexadecimal RGB de cada uma das cores primárias e o valor é um array de cores que forma as cores secundárias. Assim, é possível facilmente localizar o conjunto de cores necessários para a atualização da propriedade.

Fonte: elaborado pelo autor.

A figura 64 apresenta um diagrama de classe, com as conexões das principais classes e elementos envolvidos no processo de definição da paleta de cores.



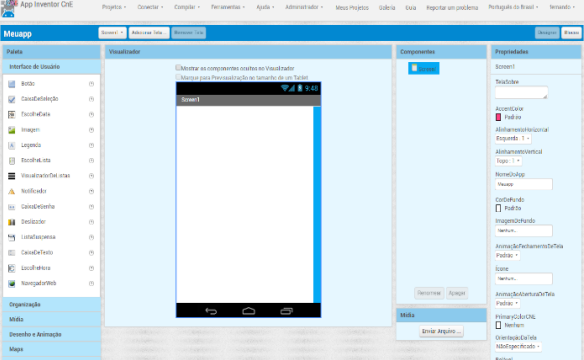
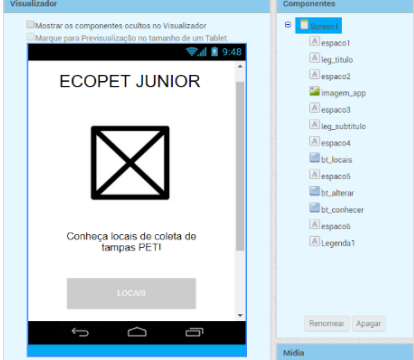
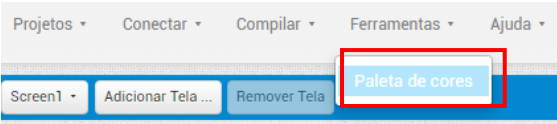
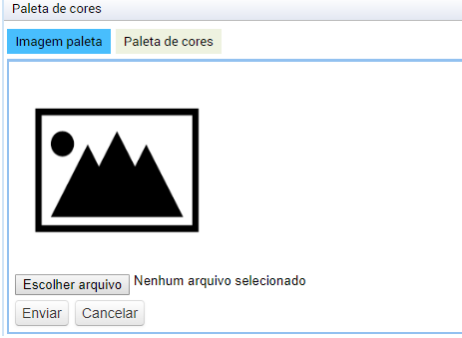
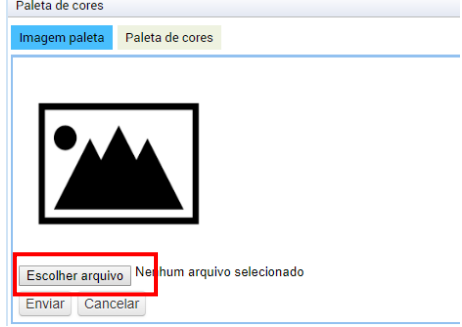
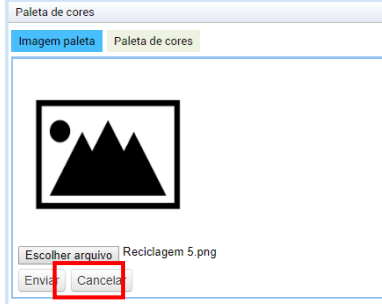
Fonte: elaborada pelo autor.

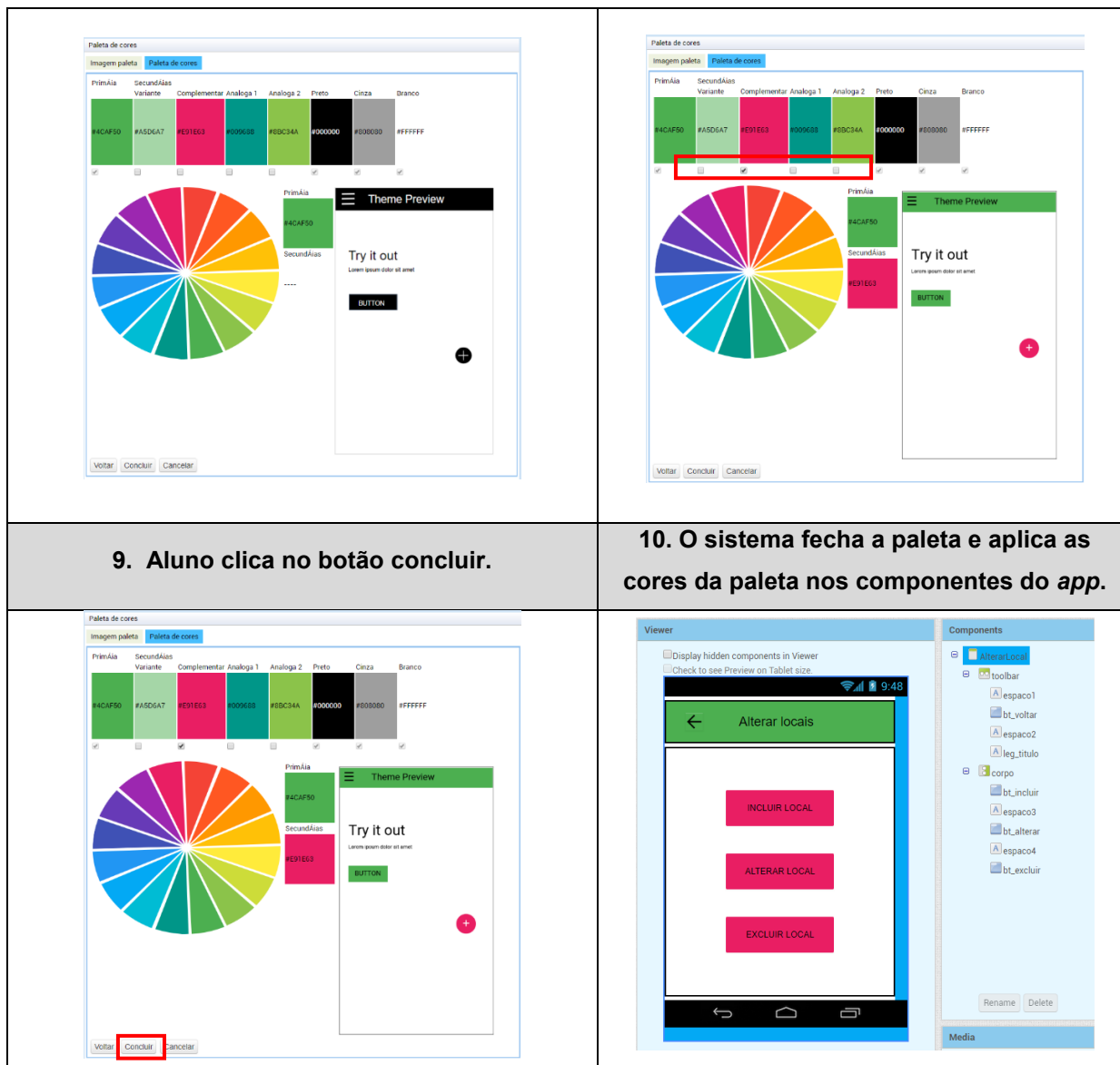
Com a edição/criação dessas classes, foi implementado o requisito R3 – Paleta de cores (Quadro 36) possibilitando ao aluno:

- Definir uma paleta de cores, por meio de uma imagem relacionado ao *app*;
- Aplicar as cores das paletas nos componentes de um *app* já desenvolvido no *App Inventor*;
- Adicionar componentes no *app* com a cor a cor primária/secundária por padrão;
- Alterar as cores primárias e secundárias;

- Definir automaticamente a cor dos textos para preto ou branco, conforme o melhor contraste com a cor do plano de fundo.

Quadro 36: Fluxo principal para definir uma paleta de cores e aplicar em um *app*.

<p>1. Aluno acessa a ferramenta <i>App Inventor</i>.</p>	<p>2. Aluno cria/importa um <i>app</i>.</p>
	
<p>3. Aluno acessa o menu “Ferramentas-> Paleta de cores”.</p>	<p>4. Sistema apresenta a tela de paleta de cores.</p>
	
<p>5. Aluno faz upload de uma imagem que lembre o <i>app</i> desenvolvido.</p>	<p>6. Aluno clica no botão “Enviar”.</p>
	
<p>7. O sistema apresenta a paleta de cores e as opções de cores secundárias.</p>	<p>8. Aluno seleciona uma cor secundária da paleta de cores ou do círculo cromático.</p>



9. Aluno clica no botão concluir.

10. O sistema fecha a paleta e aplica as cores da paleta nos componentes do app.

Fonte: elaborado pelo autor.

• USC03 REGISTRAR PERSONA E USC04 REGISTRAR HISTÓRIA DE USUÁRIO

O desenvolvimento dos requisitos R4 – Documentar persona e R5 – Documentar história de usuário foram baseados nos modelos ilustrados nas figuras 56 e 57, respectivamente, da seção 5.1. Neste desenvolvimento, foram modificadas 5 classes (Quadro 37).

Quadro 37: Classes modificadas.,

TopToolbar.java

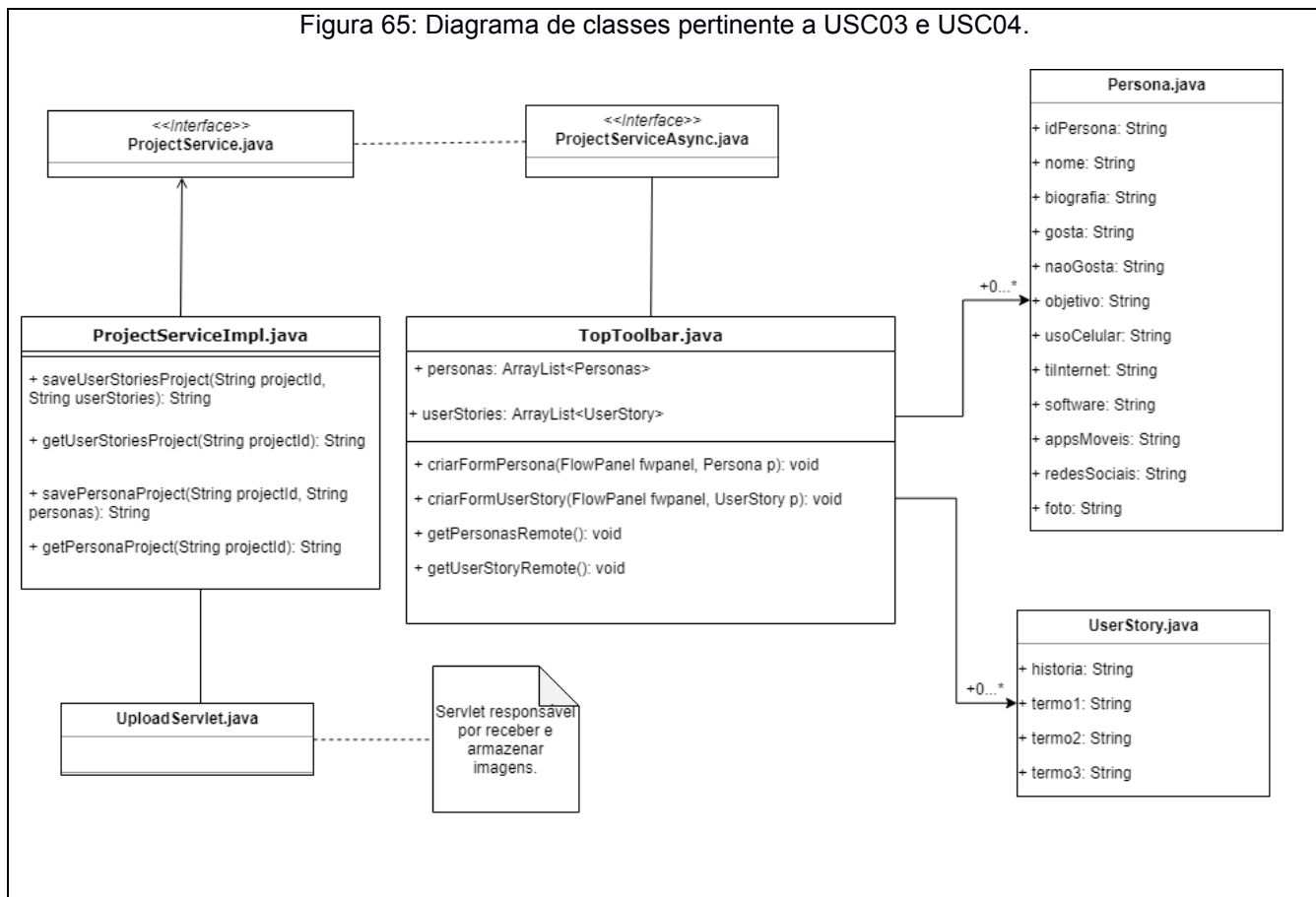
Neste requisito foram implementados os submenus “Ferramentas -> Persona” e “Ferramentas -> História de usuário”. Além disso, foi implementada a interface de telas e também o envio

das informações por meio de RPC para o servidor salvar.
ProjectServiceImpl.java
Nesta classe foi implementado o serviço RPC, para salvar e acessar informações da persona e história de usuário.
ProjectService.java
Foi implementado a interface para salvar e acessar informações da persona e a história de usuário.
ProjectServiceAsync.java
Foi implementado a interface assíncrona para salvar e acessar informações da persona e história de usuário.
UploadServlet.java
Esta classe foi utilizada apenas para atender ao requisito R4 - Documentar persona. Para cadastrar uma persona o usuário deve enviar uma imagem que represente a persona do <i>app</i> . A classe UploadServlet.java foi utilizada para armazenar e acessar a imagem utilizando a classe FileUpload.java.

Fonte: elaborado pelo autor.



A Figura 655 apresenta um diagrama de classe, com as conexões das principais classes e elementos envolvidos no processo de registro de personas e histórias de usuário.

Figura 65: Diagrama de classes pertinente a USC03 e USC04.



Com a edição dessas classes foram implementados os requisitos R4 – Documentar persona e R5 – Documentar história de usuário. Após esta implementação é possível criar, editar ou excluir uma ou mais *personas* (Quadro 38) e histórias de usuário (Quadro 39).

Quadro 38: Fluxo principal para definir uma *Persona*.

<p>Persona</p> <p>Persona 1</p>  <p>Escolher arquivo Nenhum arquivo selecionado</p> <p>Nome: <input type="text"/></p> <p>Biografia <input type="text"/> O que gosta? <input type="text"/></p> <p>O que não gosta? <input type="text"/> Qual o seu objetivo? <input type="text"/></p> <p>Uso de celular <input type="text"/> TI e Internet: <input type="text" value="5"/></p> <p>Software: <input type="text" value="5"/></p> <p>App: <input type="text" value="5"/></p> <p>Rede social: <input type="text" value="5"/></p> <p>Incluir/editar Remover</p> <p>Crear persona Salvar persona(s) Cancelar</p>	<p>Persona</p> <p>Persona 1</p>  <p>Escolher arquivo persona.PNG</p>
--	--

5. Aluno preenche o formulário com as características da persona.

6. Aluno clica em Incluir/editar.

<p>Nome: <input type="text" value="Andrézinho"/></p> <p>Biografia <input type="text" value="Andrézinho estuda na escola do bairro durante as manhãs. As tardes ele faz a tarefa de casa e brinca com os seus irmãos e amigos."/></p> <p>O que gosta? <input type="text" value="Andar de bicicleta e jogar bola os seus amigos. Jogar jogos no celular."/></p> <p>O que não gosta? <input type="text" value="Ficar sem presente de natal. Ficar de castigo. Dormir cedo."/></p> <p>Qual o seu objetivo? <input type="text" value="Ganhar um videogame de natal do papai noel."/></p> <p>Uso de celular <input type="text" value="Smartphone Android: usa diariamente para se comunicar com os amigos e pais (redes sociais)."/></p> <p>TI e Internet: <input type="text" value="4"/></p> <p>Software: <input type="text" value="4"/></p> <p>App: <input type="text" value="5"/></p> <p>Rede social: <input type="text" value="3"/></p> <p>Incluir/editar Remover</p>	<p>Nome: <input type="text" value="Andrézinho"/></p> <p>Biografia <input type="text" value="Andrézinho estuda na escola do bairro durante as manhãs. As tardes ele faz a tarefa de casa e brinca com os seus irmãos e amigos."/></p> <p>O que gosta? <input type="text" value="Andar de bicicleta e jogar bola com os seus amigos. Jogar jogos no celular."/></p> <p>O que não gosta? <input type="text" value="Ficar sem presente de natal. Ficar de castigo. Dormir cedo."/></p> <p>Qual o seu objetivo? <input type="text" value="Ganhar um videogame de natal do papai noel."/></p> <p>Uso de celular <input type="text" value="Smartphone Android: usa diariamente para se comunicar com os amigos e pais (redes sociais)."/></p> <p>TI e Internet: <input type="text" value="4"/></p> <p>Software: <input type="text" value="4"/></p> <p>App: <input type="text" value="5"/></p> <p>Rede social: <input type="text" value="3"/></p> <p>Incluir/editar Remover</p>
---	---

7. Aluno clica em "Salvar persona(s)"

Escolher arquivo Nenhum arquivo selecionado	
Nome: <input type="text" value="Andrézinho"/>	
Biografia	O que gosta?
Andrézinho estuda na escola do bairro durante as manhãs. As tardes ele faz a tarefa de casa e brinca com os seus irmãos e amigos.	Andar de bicicleta e jogar bola com os seus amigos. Jogar jogos no celular.
O que não gosta?	Qual o seu objetivo?
Ficar sem presente de natal. Ficar de castigo. Dormir cedo.	Ganhar um videogame de natal do papai noel.
Uso de celular	TI e Internet: 5 ▾
Smartphone Android: usa diariamente para se comunicar com os amigos e pais (redes sociais).	Software: 3 ▾
	App: 4 ▾
	Rede social: 4 ▾
<input type="button" value="Incluir/editar"/> <input type="button" value="Remover"/>	
<input type="button" value="Criar persona"/> <input type="button" value="Salvar persona(s)"/> <input type="button" value="Cancelar"/>	

Fonte: elaborado pelo autor.

Quadro 39: Fluxo principal para definir uma história de usuário.



<p>História de usuário</p> <p>História de usuário 1</p> <p>"Nome da história": <input type="text"/></p> <p>Como um: <input type="text"/></p> <p>eu preciso: <input type="text"/></p> <p>para que eu possa: <input type="text"/></p> <p><input type="button" value="Incluir/editar"/> <input type="button" value="Remover"/></p> <p><input type="button" value="Criar história"/> <input type="button" value="Salvar história(s)"/> <input type="button" value="Cancelar"/></p>	<p>História de usuário</p> <p>História de usuário 1</p> <p>"Nome da história": <input type="text" value="Enviar carta para o papai noel"/></p> <p>Como um: <input type="text" value="criança"/></p> <p>eu preciso: <input type="text" value="mandar uma carta para o papai noel"/></p> <p>para que eu possa: <input type="text" value="receber o presente que desejo no natal"/></p> <p><input type="button" value="Incluir/editar"/> <input type="button" value="Remover"/></p> <p><input type="button" value="Criar história"/> <input type="button" value="Salvar história(s)"/> <input type="button" value="Cancelar"/></p>
<p>5. Aluno clica em Incluir/editar.</p>	<p>6. Aluno clica em "Salvar história(s)"</p>
<p>História de usuário</p> <p>História de usuário 1</p> <p>"Nome da história": <input type="text" value="Enviar carta para o papai noel"/></p> <p>Como um: <input type="text" value="criança"/></p> <p>eu preciso: <input type="text" value="mandar uma carta para o papai noel"/></p> <p>para que eu possa: <input type="text" value="receber o presente que desejo no natal"/></p> <p><input type="button" value="Incluir/editar"/> <input type="button" value="Remover"/></p> <p><input type="button" value="Criar história"/> <input type="button" value="Salvar história(s)"/> <input type="button" value="Cancelar"/></p>	<p>História de usuário</p> <p>História de usuário 1</p> <p>"Nome da história": <input type="text" value="Enviar carta para o papai noel"/></p> <p>Como um: <input type="text" value="criança"/></p> <p>eu preciso: <input type="text" value="mandar uma carta para o papai noel"/></p> <p>para que eu possa: <input type="text" value="receber o presente que desejo no natal"/></p> <p><input type="button" value="Incluir/editar"/> <input type="button" value="Remover"/></p> <p><input type="button" value="Criar história"/> <input type="button" value="Salvar história(s)"/> <input type="button" value="Cancelar"/></p>

A 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. A classe responsável pela internacionalização do sistema é a `OdeMessages.java`, que define os métodos que retornam o valor a ser traduzido. O arquivo que contém as traduções dos termos é o `OdeMessages_pt_BR.properties`. Dado o escopo do presente trabalho, foram traduzidos somente termos para o inglês (default) e português (brasileiro). Após a conclusão da implementação, testes foram feitos para validar se essas funcionalidades atendem os requisitos funcionais modelados na sessão 4.2.

5.7. TESTE

Os cinco requisitos implementados no sistema foram testados por meio de testes exploratórios. Para cada um dos casos de uso foram definidos casos de teste,

para garantir que os requisitos sejam atendidos e identificar possíveis defeitos. Membros do grupo de pesquisa GQS³³ (3 bolsistas, 1 mestranda e 1 aluno de TCC), orientadora e coorientador do trabalho também realizaram os testes exploratórios. Os testes da re-implementação do menu de cores foram realizados utilizando os casos de testes definidos no trabalho de SILVA (2017), e nenhum *bug* foi encontrado. O quadro 40 apresenta os casos de testes e os resultados realizados para cada caso de uso implementado neste trabalho.

³³ <https://www.gqs.ufsc.br>

Quadro 40: Casos de teste.

Caso de uso	Caso de teste	Procedimento	Resultado esperado	Status
USC01 – Selecionar ícone	Incluir ícone no botão	Acessar o <i>App Inventor</i> , inserir um botão na tela do <i>app</i> e incluir um ícone acessando o catálogo de ícones desenvolvido.	Ícone como plano de fundo do botão.	Ok
USC01 – Selecionar ícone	Incluir ícone na imagem	Acessar o <i>App Inventor</i> , inserir o componente “Imagem” na tela do <i>app</i> e incluir um ícone acessando o catálogo de ícones desenvolvido.	Ícone como plano de fundo da imagem.	Ok
USC02 – Paleta de cores	Atualizar valores das propriedades <i>primaryColorCNE</i> e <i>secondaryColorCNE</i>	Acessar o menu “Ferramentas->Paleta de cores”, definir uma paleta de cores por meio de uma imagem .png e clicar em “Ok”	As propriedades <i>primaryColorCNE</i> e <i>secondaryColorCNE</i> do componente Screen deve receber como valor a cor primária e secundária definida na paleta de cores.	Ok
USC02 – Paleta de cores	Aplicar cores da paleta aos componentes a serem incluídos.	Acessar o menu “Ferramentas->Paleta de cores”, definir uma paleta de cores por meio de uma imagem .png e clicar em “Ok”	A cor de fundo dos componentes deve iniciar com a cor previamente definido na paleta de cores.	Ok
USC02 – Paleta de cores	Colorir um <i>app</i> conforme as cores definidas na paleta de cores.	Criar ou importar um projeto, acessar o menu “Ferramentas->Paleta de cores”, definir uma paleta de cores por meio de uma imagem .png e clicar em “Ok”	A cor dos componentes dos aplicativos desenvolvido devem seguir o que está definido no Quadro 31.	A funcionalidade não pintou os componentes AlinhamentoHorizontal e alguns botões para <i>apps</i> com diversas telas.
USC02 – Paleta de cores	Contraste dos textos	Arrastar o componente legenda para o <i>app</i> .	A cor de texto do label deve ser preto ou branco, conforme for o melhor contraste.	Ok
USC02 – Paleta de cores	Aplicar caixa alta nos textos dos botões	Arrastar um botão na tela do <i>app</i> , acessar o menu “Ferramentas->Paleta de cores”, definir uma paleta de cores por meio de uma imagem .png e clicar em “Ok”	O texto do botão deve estar em caixa alta.	Ok
USC02 – Paleta de cores	Textos alinhados à esquerda	Incluir um label na tela do <i>app</i> com mais de 40 caracteres, acessar o menu “Ferramentas->Paleta de cores”, definir uma paleta de cores por meio de uma imagem .png e	O alinhamento do texto deve estar à esquerda.	Ok

		clicar em "Ok"		
USC03 – Definir persona	Incluir persona	No <i>App Inventor</i> , clicar no menu "Ferramentas->Persona" e preencher todo o formulário. Clicar no botão "Incluir/Editar" e depois no botão "Salvar persona".	Salva as informações e fecha a tela. Ao abrir a tela da persona novamente deve abrir as informações salvas.	Ok
USC03 – Definir persona	Editar persona	No <i>App Inventor</i> , clicar no menu "Ferramentas->Persona" e editar alguma informação da persona. Clicar no botão "Incluir/Editar" e depois no botão "Salvar persona".	Salva as informações e fecha a tela. Ao abrir a tela da persona novamente deve abrir as informações salvas.	Ok
USC03 – Definir persona	Excluir persona	No <i>App Inventor</i> , clicar no menu "Ferramentas->Persona". Com a persona previamente preenchida, clicar no botão "Excluir" e depois no botão "Salvar persona".	Exclui a persona e fecha a tela. Ao abrir a tela da persona novamente não deve abrir as informações.	ok
USC04 – Definir história de usuário	Incluir história de usuário	No <i>App Inventor</i> , clicar no menu "Ferramentas->História de usuário" e preencher todo o formulário. Clicar no botão "Incluir/Editar" e depois no botão "Salvar história de usuário".	Salva as informações e fecha a tela. Ao abrir a tela da história de usuário novamente deve abrir as informações salvas.	Ok
USC04 – Definir história de usuário	Editar história de usuário	No <i>App Inventor</i> , clicar no menu "Ferramentas->História de usuário" e editar alguma informação da história de usuário. Clicar no botão "Incluir/Editar" e depois no botão "Salvar história de usuário".	Salva as informações e fecha a tela. Ao abrir a tela da história de usuário novamente deve abrir as informações salvas.	Ok
USC04 – Definir história de usuário	Excluir história de usuário	No <i>App Inventor</i> , clicar no menu "Ferramentas->História de usuário". Com a história de usuário previamente preenchida, clicar no botão "Excluir" e depois no botão "Salvar história de usuário".	Exclui a história de usuário e fecha a tela. Ao abrir a tela da persona novamente não deve abrir as informações.	Ok

Fonte: elaborado pelo autor.

Dentre os 14 testes executados, apenas um apresentou resultado diferente do esperado. O erro ocorreu por falha lógica de processamento e foi corrigido. Os testes também resultaram em algumas sugestões de melhorias, que então foram implementadas (quadro 41).

Quadro 41: Sugestões de melhorias implementadas.

Caso de uso	Status
USC01 – Selecionar ícone	<ul style="list-style-type: none"> -Inserir que o ícone fosse inserido no botão quando o aluno dar um duplo clique em cima; -Trocar o nome do botão “Adicionar” para “Ok”; -Fechar a tela de catálogo de ícone ao clicar fora dela; -Aumentar espaçamento entre os botões de filtragem; -Trocar as ordens dos filtros; -Retirar o texto do botão ao incluir o ícone.
USC02 – Paleta de cores	<ul style="list-style-type: none"> -Colorir o componente Alinhamento horizontal quando estiver fazendo função de caixa de ferramenta do <i>app</i>. -Maior espaçamento entre os botões. -Posicionar a palavra "Secundárias" de forma a permitir maior compreensão de que se trata de 4 opções da cor secundária a ser selecionada.
USC03 – Definir persona e USC04 – Definir história de usuário	Melhorar o espaçamento e alinhamento dos componentes de tela.

Fonte: elaborado pelo autor.

O controle de versionamento dos arquivos editados e criados, das listas de tarefas realizadas e os bugs resolvidos foi realizado por meio do ambiente de desenvolvimento da UFSC – GitLab³⁴. O código desenvolvido é livre e encontra-se disponível para melhorias e para ser utilizado em trabalhos futuros. Este é distribuído de forma gratuita, em forma de código aberto, sob a licença *Apache License 2.0* podendo ser clonado via git no ssh: `git@codigos.ufsc.br:100000000394729/InventorCnE.git` mediante a liberação de acesso por parte do administrador do repositório no ambiente UFSC – GitLab.

³⁴ <https://codigos.ufsc.br>

6. INSTANCIAMENTO, APLICAÇÃO E AVALIAÇÃO DA UNIDADE INSTRUCIONAL

Neste capítulo é apresentado a instanciamento, aplicação e avaliação do modelo educacional por meio de 2 estudos de casos.

6.1. AVALIAÇÃO DO MODELO EDUCACIONAL

6.1.1. Definição de avaliação

O objetivo geral da avaliação consiste em explorar e compreender aspectos relacionados ao modelo educacional, para o ensino de conceitos de computação no Ensino Fundamental por meio de desenvolvimento de aplicativos. Esta avaliação visa responder a seguinte pergunta de pesquisa: práticas pedagógicas de ensino de ES e EU, de forma extracurricular, inseridas no ensino de desenvolvimento de aplicativos móveis com *App Inventor*, contribuem para aumentar as competências e motivação/interesse dos alunos referente à computação e apresentam qualidade suficiente?

Para definir as medições em relação a esta pergunta de pesquisa, foi adotada a abordagem definida no método GQM (BASILI *et al.*, 1994), com a intenção de elaborar um plano de medição para a realização da avaliação. Esta abordagem consiste em sistematicamente decompor a pergunta de pesquisa em perguntas de análise e medidas, e o desenvolvimento de instrumentos de medição. Com base na pergunta de pesquisa, foram definidas as seguintes perguntas de análise:

PA1. Práticas pedagógicas para o ensino de ES e EU inseridas no ensino de desenvolvimento de aplicativos móveis com *App Inventor* aumentam competências dos alunos referentes a aprendizagem de conteúdos de computação?

PA2. Práticas pedagógicas para o ensino de ES e EU inseridas no ensino de desenvolvimento de aplicativos móveis com *App Inventor* promovem uma experiência de aprendizagem agradável e divertida?

PA3. As práticas pedagógicas projetadas nessas UIs para o ensino de ES e EU inseridas no ensino de desenvolvimento de aplicativos móveis com *App Inventor* facilitam a aprendizagem (usabilidade da unidade)?

PA4. As práticas pedagógicas projetadas nessas UIs para o ensino de ES e EU inseridas no ensino de desenvolvimento de aplicativos móveis com *App Inventor*

aumentam o interesse dos alunos em relação à aprendizagem de conteúdos de computação?

Por meio das perguntas de análise, foram desmembradas as medidas e seus respectivos instrumentos de medição (Quadro 42).

Quadro 42: Síntese da definição da avaliação da UI.

Pergunta de análise	Medida	Instrumento de medição
PA1. A unidade instrucional aumenta competências dos alunos referentes a aprendizagem de conteúdos de computação?	M1.1 Grau de aprendizagem referente à capacidade de fazer aplicativos para smartphones M1.2 Grau de aprendizagem referente à capacidade de descrever um algoritmo em uma sequência de instruções M1.3 Grau de aprendizagem referente à capacidade do uso do <i>App Inventor</i> para desenvolver um <i>app</i> em duplas, instalar no smartphone e compartilhar M1.4 Grau de habilidade para ensinar o aprendido a outras pessoas M1.5 Grau de aprendizagem referente à capacidade de fazer aplicativos para smartphones, sendo um processo de desenvolvimento de <i>apps</i>	- Rubricas - Questionário pré/pós-unidade alunos - Questionário pós-jogo SplashCode (oficinas) - Observações - CodeMaster 2.0 - Questionário pré/pós-aula Design Visual - Questionário pós-unidade pais - Questionário pré/pós-oficina alunos
PA2. A unidade instrucional promove uma experiência de aprendizagem agradável e divertida?	M2.1 Grau de diversão das aulas M2.2 Grau de percepção do tempo durante as aulas M2.3 Grau da interação social (compartilhar a experiência com outros colegas) M2.4 Opinião subjetiva sobre a aula M2.5 Pontos fortes em relação à experiência das aulas M2.6 Pontos fracos em relação à experiência das aulas M2.7 Vontade de participar novamente da unidade instrucional em outro momento	- Questionário pós-unidade alunos - Questionário pós-unidade pais - Questionário pós-oficina alunos
PA3. A unidade instrucional facilita a aprendizagem?	M3.1 Grau de facilidade das aulas M3.2 Grau de facilidade do processo de desenvolvimento de aplicativos para smartphones M3.3 Grau de qualidade geral das aulas	- Questionário pós-unidade alunos - Questionário pós-unidade pais - Questionário pós-oficina alunos
PA4. A unidade instrucional aumenta o interesse dos alunos referente a aprendizagem de conteúdo da computação?	M4.1 Vontade de aprender computação na escola M4.2 Grau de satisfação em fazer <i>apps</i> para smartphones M4.3 Percepção da importância da computação no dia-a-dia M4.4 Vontade de trabalhar com computação no futuro	- Questionário pós-unidade alunos - Questionário pós-unidade pais - Questionário pós-oficina alunos

Fonte: elaborado pelo autor.

Os questionários de pós-unidade (Anexo 1), que avaliam a UI, foram baseados nos modelos dTECT (Gresse von Wangenheim et al., 2017) e MEEGA+KIDS (Gresse von Wangenheim et al., 2018), derivados do modelo MEEGA (Petri et al., 2018).

6.2. INSTANCIACÃO E APLICAÇÃO DA UI

O modelo educacional foi aplicado em duas versões e em dois momentos diferentes. Nas seções abaixo são apresentadas e ilustradas as aplicações da unidade.

6.2.1. Aplicação da versão longa - Jovens Tutores 2018

A versão longa/completa da unidade instrucional foi aplicada com as turmas do 8º e 9º anos da escola básica municipal Almirante Carvalhal em Florianópolis/SC, no segundo semestre de 2018. Esta aplicação foi realizada no contexto do projeto Jovens Tutores de Programação³⁵, coordenado pela iniciativa Computação na Escola/INCoD/INE/UFSC. Este projeto visa formar Jovens Tutores (alunos do ensino Fundamental) para programar aplicativos móveis, e para que futuramente disseminem os conhecimentos de computação para outros alunos.

As aulas foram ministradas pelo autor deste projeto, juntamente com um aluno de graduação em ciência da computação do Departamento de Informática e Estatística (INE) da UFSC. Estas aulas tiveram auxílio de mentores voluntários que são profissionais da área de TI de uma empresa patrocinadora do projeto, além da professora de informática e de um professor responsável pelo laboratório de informática. O projeto envolveu dez alunos (jovens tutores), sendo cinco meninas e cinco meninos entre 13 e 15 anos. A aplicação teve duração de aproximadamente 4 meses, em encontros semanais de três horas, na própria escola. Os alunos foram selecionados e convidados pela professora de informática da escola, os quais participaram de maneira voluntária no contraturno de suas aulas.

O primeiro encontro foi realizado na UFSC, o qual reuniu todos os envolvidos do projeto (Figura 66). Neste encontro foi feita uma apresentação dos envolvidos e a definição do cronograma. A seguir os jovens tutores aprenderam conceitos básicos de computação, começando com o conhecimento de algoritmos. Para aplicar este

³⁵ <http://www.computacaonaescola.ufsc.br/?p=2739>

conhecimento na prática os jovens tutores jogaram o jogo SplashCode (GRESSE *et al.*, 2019) junto com os mentores voluntários. Ainda neste encontro, os alunos conheceram a ferramenta de programação *App Inventor* desenvolvendo o *app* “Encontre-me” com auxílio dos mentores voluntários e seguindo um tutorial.

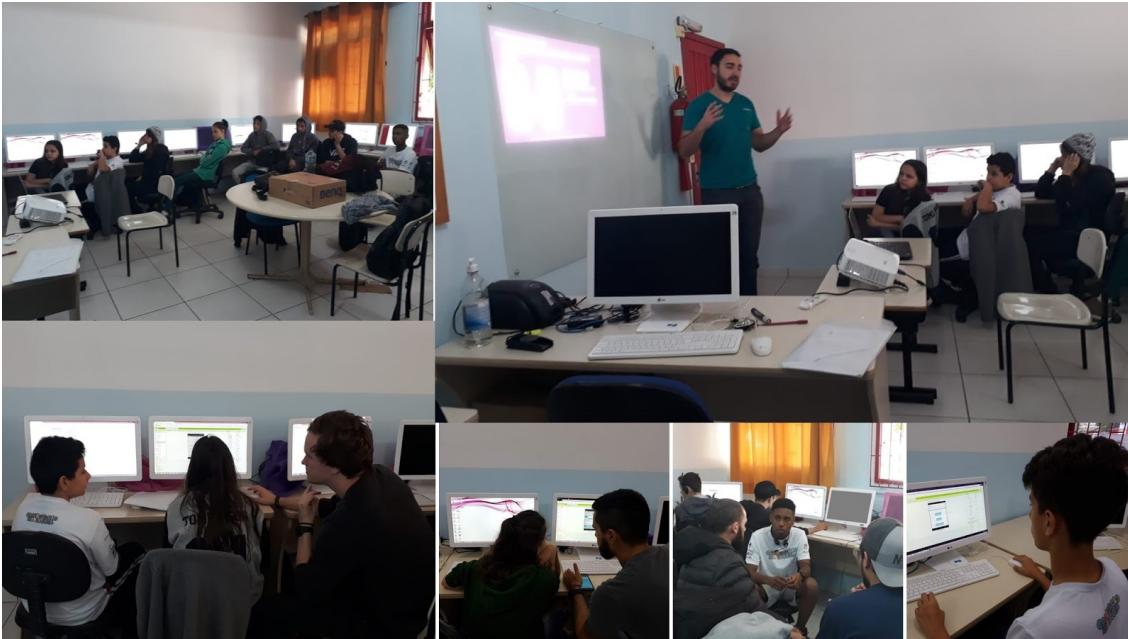
Figura 66: Primeiro encontro do projeto na UFSC.



Fonte: elaborada pelo autor baseada nas fotos do encontro.

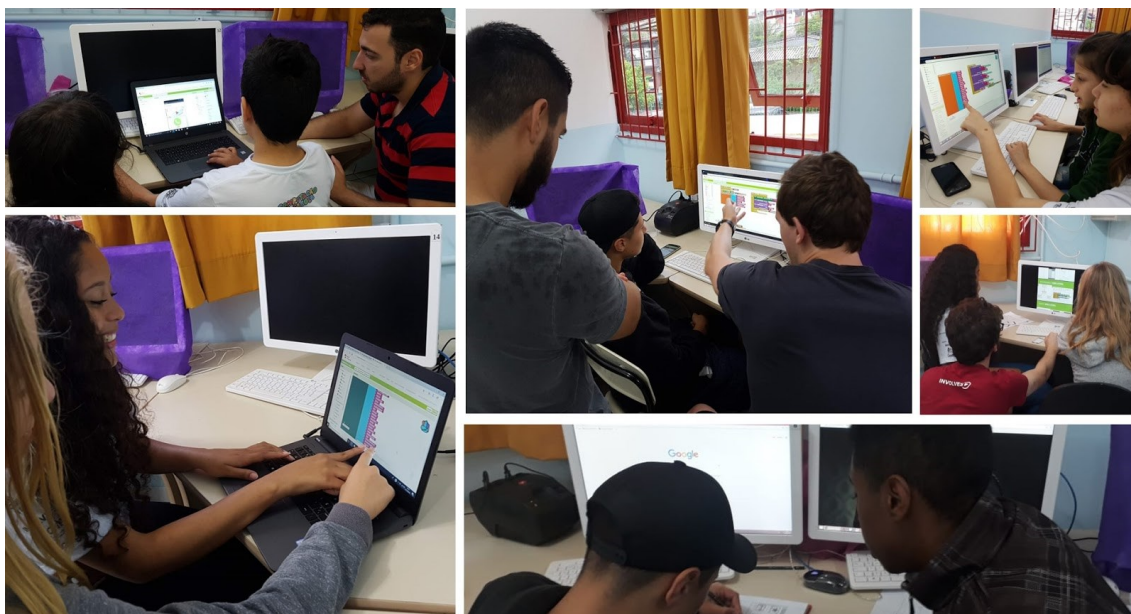
Os próximos encontros do curso ocorreram semanalmente na escola e tiveram como propósito ensinar aos alunos a desenvolver seu próprio *app*, seguindo um processo de desenvolvimento de *apps*, adotando uma abordagem de *Design Thinking* (Figura 67). Na primeira aula, os dez jovens tutores foram divididos em pares, formando cinco grupos nos quais cada par deveria desenvolver um *app*. Para isso, foi apresentado o processo de desenvolvimento de *apps* que eles devem seguir. Após, os alunos aprenderam a identificar um problema e a idealizar uma solução por meio de um *smartphone*. Cada par de alunos levantou um problema e a solução idealizada e, por meio de um debate entre eles, a solução foi refinada. Nesta aula, também ocorreu o ensino da análise do contexto e a especificação de requisitos, definindo quais as tarefas cada *app* deve realizar, os requisitos funcionais e de usabilidade. No terceiro encontro, os alunos criaram um design de baixa fidelidade construindo protótipo em papel e testando, até chegarem no *design* final do protótipo de baixa fidelidade.

Figura 67: Encontros de ensino do processo de desenvolvimento de apps 1.



Fonte: elaborada pelo autor baseada nas fotos do encontro.

Nas aulas 4, 5 e 6 os alunos programaram o *app* e realizaram testes a cada funcionalidade, definidos nos requisitos funcionais (Figura 68). Nestes encontros, os alunos foram auxiliados pelos mentores, professores e material didático extra produzido no âmbito deste projeto. Ao final do sexto encontro, os jovens tutores finalizaram uma primeira versão funcional de seus *apps*. O encontro 7 envolveu o ensino da criação e testes do design visual dos aplicativos. Os alunos padronizaram os ícones, definiram cores, tipografia e imagens para o seu *app*. No oitavo encontro, os jovens tutores realizaram um teste de sistema para validar se os requisitos funcionais e de usabilidade foram alcançados. Para a realização dos testes funcionais, os alunos aprenderam a definir casos de testes para cada funcionalidade. Os testes de usabilidade foram realizados entre os pares de alunos, seguindo os requisitos de usabilidade pré-definido na aula 2. No último encontro eles aprenderam como gerar uma versão *.apk* do aplicativo para publicar no Google Play, e a compartilhar informações de seus aplicativos por meio de cartazes e de uma página da *web*. Os alunos também tiveram uma revisão geral dos conceitos ensinados em todo curso, para depois responderem os questionários finais de avaliação.

Figura 68: Alunos programando seu próprio *app*.

Fonte: elaborada pelo autor baseada nas fotos do encontro.

No decorrer do curso os alunos geraram *workbooks*, relatando as tarefas realizadas no processo de desenvolvimento de *apps*. A Figura 69 ilustra os *workbooks* completos para um dos *apps* desenvolvidos. No final do curso, um aluno saiu por motivos pessoais, sendo assim, um dos alunos teve que terminar o desenvolvimento do *app* sozinho. Como o curso já estava nas últimas aulas, o *app* já estava em sua fase final de desenvolvimento, não prejudicando a aprendizagem do aluno.

Figura 69: Workbook do app "InfoCarvalho".

Identificação do problema

Qual é o problema identificado?	Falta de informação em escola!
Quem é afetado pelo problema e como é afetado?	Alunos e Pais que recebem as informações da escola
Um aplicativo pode resolver este problema?	Sim

User story

História: *Conhecer colegas de sala*
 Como um(a) *Novo aluno*
 eu preciso *Saber todos alunos*
 para que eu possa *Fazer novas amizades*

Identificação da solução

Nome do seu app: *Almirante Carvalho - Carvalhos Escolar*

Quem vai fazer o que com o app? *Se utilizar sobre a Escola*

Por que seu app é impressionante? *Filou nos embolamos os alunos.*

Personas

Requerimentos de usabilidade

Tarefa: *Saber os professores de cada matéria*

- Todos os usuários conseguem completar a tarefa.
- Todos os usuários conseguem completar a tarefa em no máximo 5 segundos (minutos).
- Todos os usuários respondem a pergunta "essa tarefa em geral foi difícil, difícil, nem fácil-nem difícil, fácil, muito fácil" no mínimo com Pazil (muito difícil, difícil, nem fácil-nem difícil, fácil...).

Sketches

Resultados teste de interface

O que funcionou...	<i>Judo</i>
Dúvidas...	<i>Nenhuma</i>
O que pode ser melhorado...	<i>Design</i> <small>do conteúdo da lista</small>
Ideias...	<i>Nenhuma</i>

Resultados teste de usabilidade

Tarefa: *Ver a lista de alunos da sua turma*

- Todos os usuários conseguem completar a tarefa.
- Todos os usuários conseguem completar a tarefa em no máximo 15 seg minutos/segundos.
- Todos os usuários acham que foi Muito fácil realizar essa tarefa.

Wireframe

Resultados teste funcional

Teste cada funcionalidade do seu app para cada ação e entrada.

Funcionalidade:	Ação	Entrada	Resultado esperado	Resultado retornado
<i>Uma vez se a decisão está no meu tempo</i>	Selecionar a turma e escolher a lista de alunos	<i>botão turma</i>	Aparecer a lista de alunos	OK
	Selecionar a turma e escolher a lista de horários		Aparecer a lista de horários	OK
	Selecionar a turma e escolher a lista de professores		Aparecer a lista de professores	OK
	Clicar em voltar		Voltar uma tela	OK

Design final

Fonte: elaborada pelo autor.

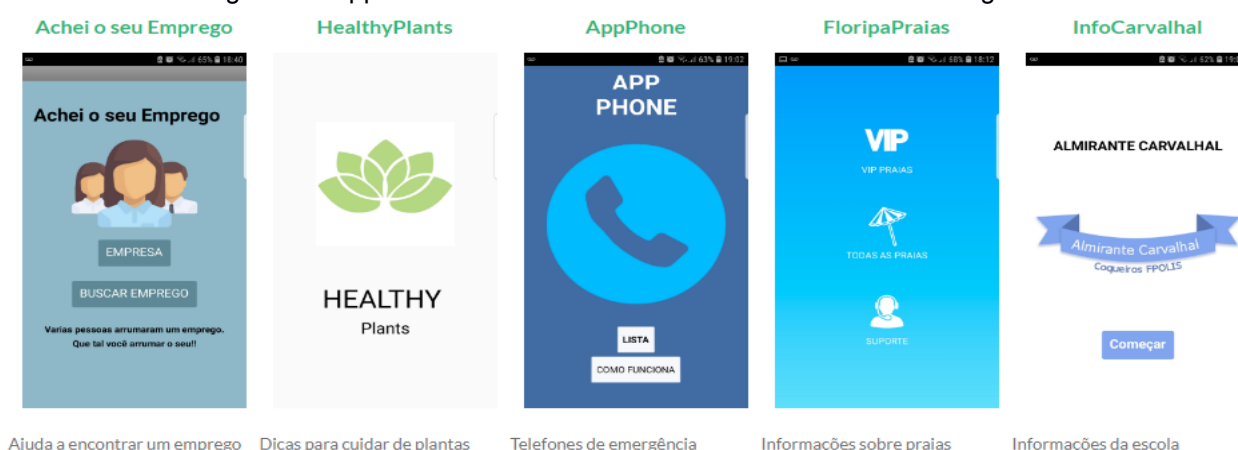
Os cinco *apps* criados (Quadro 43 e Figura 70) foram apresentados na própria escola e em uma empresa de TI local.

Quadro 43: Descrição dos apps desenvolvidos na unidade instrucional versão longa.

Nome	Descrição
Achei o seu emprego	Auxilia a encontrar um emprego. Permite o cadastro de vagas pelas empresas e também a busca por empregos
HealthyPlants	Fornecer informações importantes para ajudar a cuidar das plantas em casa
AppPhone	Ajuda a encontrar números de telefone importantes para casos de emergência
FloripaPraias	Exibe informações sobre praias de Florianópolis como balneabilidade, acesso, trilhas, etc.
InfoCarvalho	Apresenta informações úteis sobre a Escola B

Fonte: elaborado pelo autor.

Figura 70: Apps desenvolvidos na unidade instrucional versão longa.



Fonte: elaborada pelo autor baseado no *apps* desenvolvidos.

Além da aplicação desta UI, o projeto Jovens Tutores de Programação também realizou 3 oficinas para crianças de 10 a 14 anos (Figura 71). Estas oficinas também fazem parte do projeto, e têm como propósito introduzir o conhecimento da computação com conceitos básicos de algoritmo e programação. Isto foi feito utilizando o jogo SplashCode e ensinando a construção do *app* Encontre-me, em 2 horas/aula. Os jovens tutores foram os responsáveis por auxiliar os outros alunos e por passar conhecimentos que já adquiriram durante a aplicação do projeto. Ao todo, foram realizadas três oficinas, atingindo 51 crianças.

Figura 71: Oficinas realizadas na escola.



Fonte: elaborada pelo autor baseada nas fotos do encontro.

6.2.3. Aplicação da versão curta – ECOPET 2019-1

A aplicação da versão curta da unidade instrucional também ocorreu na escola básica municipal Almirante Carvalho, com 26 alunos (11 meninas e 15 meninos), do 5º ano do Ensino Fundamental (9 a 11 anos). Esta aplicação ocorreu durante o primeiro semestre de 2019, sob a iniciativa Computação na Escola/INCoD/INE/ UFSC (Figura 72).

Figura 72: Alunos nas atividades de *design visual*.



Fonte: elaborada pelo autor baseada nas fotos do encontro.

Na primeira parte do curso os alunos aprenderam sobre computação e algoritmo. Para complementar este conhecimento, os alunos sentaram-se em grupos para jogar o jogo SplasCode. No segundo encontro, os alunos tiveram o primeiro

contato com a ferramenta *App Inventor*. Seguindo um tutorial e com ajuda dos monitores, os alunos programaram uma versão simples do *app* EcoPET. No terceiro e quarto encontro os alunos aprenderam a aplicar conceitos básicos de *design visual*. Nestas aulas os alunos aplicaram os tópicos design visual abordados em aula em uma versão completa do *app* EcoPET disponibilizado para os alunos. Os alunos aplicaram no *app* os seguintes tópicos: cores, tipografia e imagem (Figura 73).

Figura 73: Exemplos de design visual do app criado pelos alunos.



Fonte: elaborada pelo autor baseado no *apps* desenvolvidos.

Na última aula os alunos apresentaram os *apps* na escola durante um evento com os pais e outros colegas. Durante o curso 2 alunos desistiram por motivos não definidos.

6.3. ANÁLISE DOS DADOS

No final das aplicações os alunos foram avaliados conforme definido previamente na seção 4.5. Com isso, foi possível realizar uma análise dos dados com o objetivo de avaliar a UI. A seguir, é apresentada a análise dos dados separadamente para as duas aplicações.

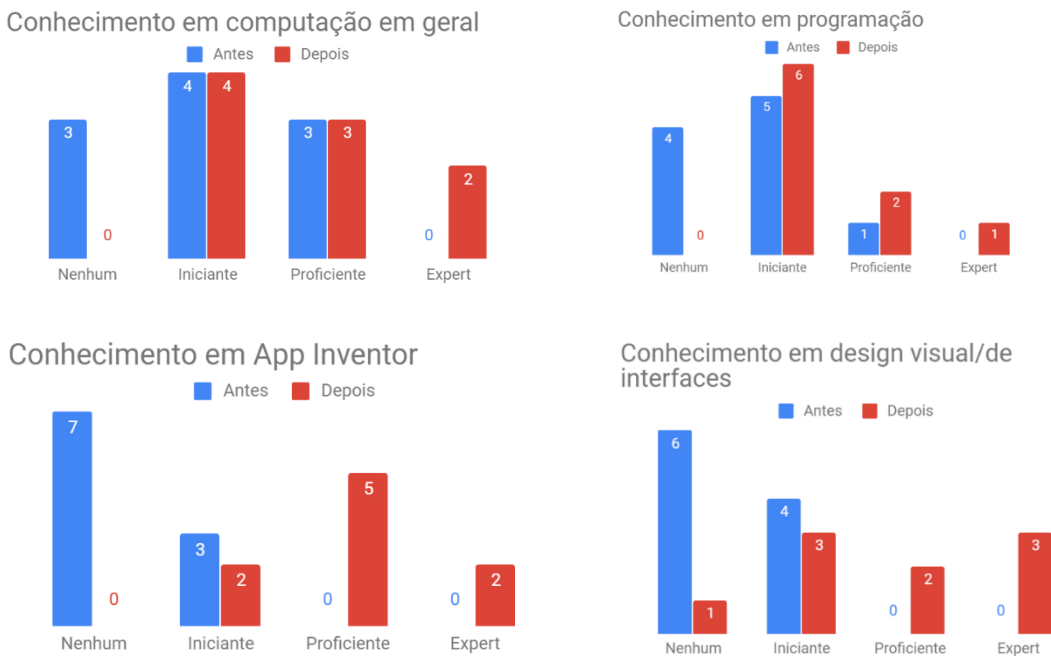
6.3.1. Análise dos dados da aplicação da unidade instrucional versão longa – Jovens Tutores

Os dados analisados foram coletados dos questionários pré- e pós-aplicação e dos *workbooks*. Para detalhar melhor os resultados obtidos, os resultados são apresentados para cada pergunta de análise definidas no quadro 41.

PA1. A unidade instrucional aumenta competências dos alunos referentes a aprendizagem de conteúdos de computação?

Para analisar as competências dos alunos referentes à aprendizagem, foram utilizados questionários pré e pós-aplicação de autoavaliação, avaliação por desempenho e testes. No questionário pós-aplicação de autoavaliação todos os alunos consideraram que tem pelo menos algum conhecimento, e dois deles consideram ser “*Expert*”. Percebe-se que a unidade instrucional transmitiu conhecimentos de computação, pois antes a maioria dos alunos consideraram ter nenhum conhecimento de computação em geral. Competências que envolvem a computação, como programação, juntamente com a ferramenta *App Inventor* e design visual, também apresentaram resultados positivos (Figura 74).

Figura 74: Análise dos dados sobre o nível de conhecimento antes e depois da unidade.

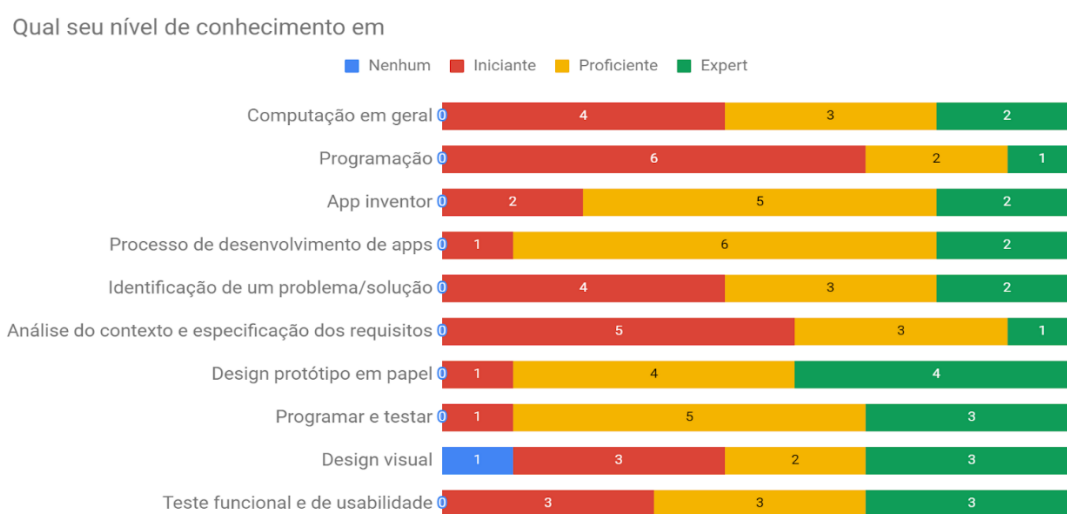


Fonte: elaborada pelo autor.

Os alunos em geral consideram que tem algum conhecimento nos tópicos abordado na UI, exceto para o design visual, em que um aluno respondeu não ter aprendido nenhum conhecimento, provavelmente por alguma dificuldade encontrada

ou insatisfação. Destacam-se os tópicos processos de desenvolvimento de *apps*, design de protótipo em papel, programar e testar, teste funcional e de usabilidade, sendo os que tiveram maior quantidade de alunos que consideram ter obtido o nível de conhecimento “proficiente” e “expert”. Os tópicos análise de contexto e especificação dos requisitos foram os que tiveram maior quantidade de alunos que sentem ter pouco conhecimento (Figura 75).

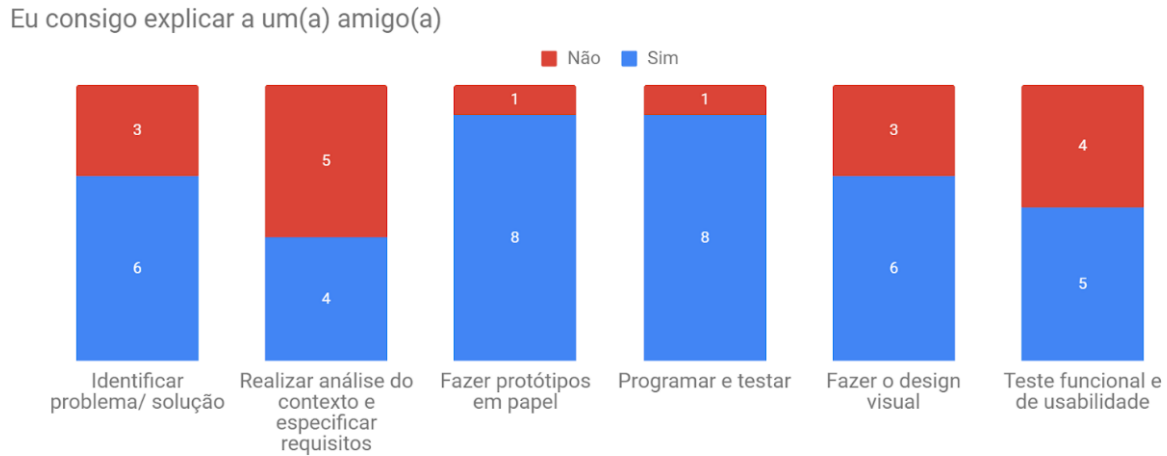
Figura 75: Distribuição da frequência de itens relacionados à percepção da aprendizagem (pós-aplicação).



Fonte: elaborada pelo autor.

Além do nível de conhecimento, foi avaliado o nível de compreensão observando se os alunos conseguem explicar para um colega os tópicos da UI. Apesar de este nível de aprendizagem ter maior grau de dificuldade em relação ao de conhecimento, a maioria dos alunos julga saber explicar, principalmente as atividades de criação de protótipos em papel e de programar e testar. O tópico realizar análise de contexto e especificar os requisitos são os tópicos que os alunos sentiram mais dificuldade em compreender (Figura 76).

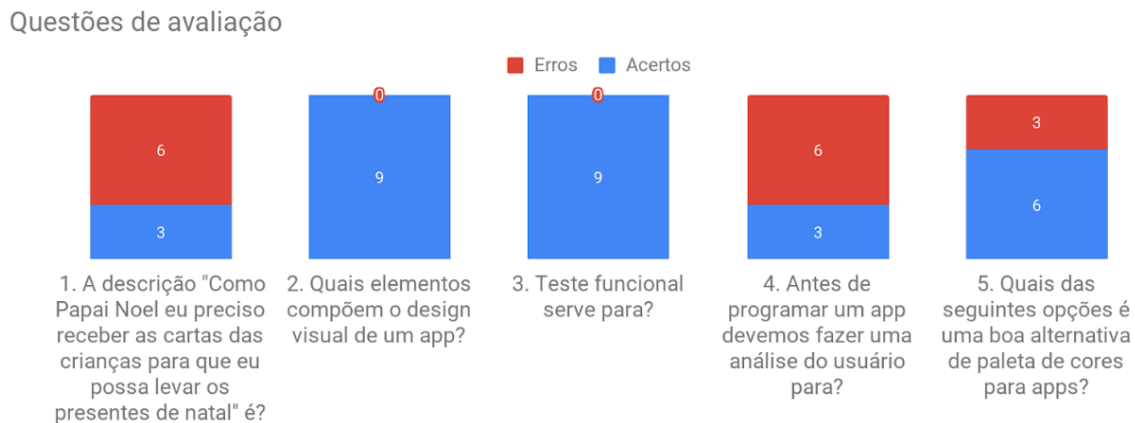
Figura 76: Quantidade de alunos que conseguem explicar tópicos da UI.



Fonte: elaborada pelo autor.

O resultado do teste aplicado também demonstra que a maioria dos alunos teve dificuldade em aprender assuntos relacionados à análise de contexto e especificação de requisitos, visto que a maioria dos alunos erraram as questões 1 e 4 (Figura 77). Infere-se que este resultado pode estar relacionado ao desinteresse dos alunos em aprender conceitos de natureza teórica relacionado a ES, como documentação de histórias de usuários e requisitos funcionais. Nesta avaliação, destaca-se que todos os alunos obtiveram conhecimento sobre os principais elementos de design visual e compreenderam qual o propósito do teste funcional, ao acertarem as questões 2 e 3, respectivamente. O fato de todos os alunos terem acertado estas questões também pode estar relacionado ao nível de dificuldade, que pode ser menor que os demais.

Figura 77: Quantidade de acertos em cada questão.



Fonte: elaborada pelo autor.

A aprendizagem de competências de computação também foi avaliada por meio da avaliação de desempenho. Esta análise é realizada por meio dos *apps* que os alunos criaram em pares e pelos artefatos gerados durante o seu desenvolvimento. Por meio dos *apps* criados foram avaliados:

- *Aspectos sobre programação*: por meio da ferramenta CodeMaster 2.0 é possível avaliar os *apps* criados pelos alunos. Esta ferramenta utiliza uma rubrica para avaliar conceitos relacionados à programação, incluindo laços, operadores, *strings*, nomeação de componentes etc. O resultado desta ferramenta gera pontuações para cada conceito e uma nota final, com base na soma das pontuações (Figura 78). Analisando o resultado, diversos conceitos apresentados nas aulas foram devidamente aplicados nos *apps* desenvolvidos pelos alunos. Percebe-se que os alunos não pontuaram em alguns conceitos, pois estes não foram ensinados durante a aplicação, como abstração de procedimento, desenho e animação, extensões. Em um caso excepcional, um *app* aplicou um dos itens avaliados, por ter alguma particularidade. As notas apresentadas demonstram que os alunos conseguiram compreender e aplicar em seus *apps* o que lhes foi ensinado.

Figura 78: Avaliando a aprendizagem de conceitos de programação usando CodeMaster 2.0.

Projeto	Telas	Nomeação de Componentes	Eventos	Abstração de Procedimentos	Laços	Condicionais	Listas	Persistência de Dados	Sensores	Desenho e Animação	Operadores	Variáveis	Strings	Sincronização	Mapas	Extensões	Pontuação total	Nota	Nível
InfoCarvalho.aia	3	2	3	0	0	1	2	2	0	0	2	2	2	0	0	0	19	4,63	faixa roxa
FloripaPraias.aia	3	2	3	0	0	3	1	3	0	0	2	2	2	0	0	0	21	5,12	faixa azul
HealthyPlants.aia	3	1	3	0	0	1	1	3	1	0	2	2	2	1	0	0	20	4,88	faixa roxa
AcheiOseuEmprego.aia	3	1	3	0	3	3	2	3	1	0	0	2	2	1	0	0	24	5,85	faixa azul
AppPhone.aia	3	2	3	0	0	3	1	0	0	0	3	2	1	0	0	0	18	4,39	faixa roxa
Média	3.00	1.60	3.00	0.00	0.60	2.20	1.40	2.20	0.40	0.00	1.80	2.00	1.80	0.40	0.00	0.00	20.40	4.98	

Fonte: retirada da ferramenta CodeMaster 2.0.

- *O design da interface de usuário*: a avaliação de desempenho, referente ao design visual, foi baseada na rubrica da quadro 28. A pontuação de cada item de design de interface avaliado foi definida por uma mestrandia do curso de Design da UFSC. Com base nesta rubrica, os cinco aplicativos desenvolvidos pelos jovens tutores, de forma geral, atingiram boas pontuações, com destaque para os *apps*

“HealthyPlants” e o “Floripa Praias”. Conforme a tabela 4, os resultados da avaliação demonstram que todos os conceitos de design visual abordados no curso (tipografia, imagem, ícones e cores) foram aplicados nos *apps* dos alunos. Entretanto, constata-se que os seguintes tópicos de design não foram bem avaliados por alguns dos aplicativos: definição de ícone (que seja fácil de interpretar) e padronização das telas (em relação a elementos de cor, imagem e tipografia). Destacam-se positivamente os tópicos que avaliam o tamanho dos textos e qualidade das imagens nos *apps* desenvolvidos, os quais apresentam nota máxima em todos eles. Percebe-se também que os *apps* apresentam cores coerentes com seus temas e contraste (entre a cor do texto e fundo da tela) que contribuem para a interação do usuário.

Tabela 4: Pontuação das interfaces dos apps desenvolvidos.

Crítérios	Achei o seu Emprego	Healthy plants	App Phone	Floripa praias	Info Carvalhal
A escolha de cores (paleta de cores) está coerente com o tema escolhido	4	4	3	3	4
A organização das cores facilita a interação do usuário	3	4	3	4	4
As cores auxiliam na hierarquia de informações	4	4	3	3	4
O contraste entre cor e texto e fundo da tela facilita a leitura das informações (asseguram a legibilidade)	3	4	3	3	4
Os textos apresentam tamanho das letras (fonte) agradáveis para leitura	4	4	4	4	2
O alinhamento do texto contribui para leitura e harmonia do design visual	2	3	3	4	4
O tamanho das letras (fontes) auxilia no destaque de informações (hierarquia)	4	3	4	4	3
As imagens estão de acordo com o tema	4	4	4	4	4
As imagens estão nítidas (não estão pixeladas e/ou distorcidas)	4	4	4	4	4
Os ícones são fáceis de interpretar	4	2	4	4	2
As telas respeitam o mesmo padrão em	2	4	1	2	3

relação aos elementos de cor, imagem e tipografia (tamanho, família, estilo, peso)					
De modo geral o design visual está agradável e organizado	3	4	3	4	4
Pontuação	42	44	38	43	41
Nota (0-10)	8,8	9,2	7,9	9,0	8,5

Fonte: elaborada pelo autor.

• *Estética do app*: este critério foi analisado por meio de um *survey* com 95 participantes, avaliando 110 *screenshots* de telas de aplicativos do *App Inventor*, incluindo 105 *apps* aleatoriamente escolhidos da Galeria do *App Inventor* (SOLECKI *et al.*, 2019). Este estudo foi realizado com base em 95 respostas de pessoas voluntárias, classificando as telas como “feias”, “mais ou menos” ou “bonitas”. Com base nas respostas dos participantes, calculou-se uma pontuação de grau de estética entre 0 e 95 para cada UI, somando as respostas (“feia” = 0 ponto; “mais ou menos” = 0,5 ponto; “bonita” = 1 ponto) (Tabela 5).

Tabela 5: Pontuação de estéticas dos apps.

Aplicativo	Pontuação (média)
Achei o seu emprego	56,25
HealthyPlants	84,5
<i>AppPhone</i>	59,25
FloripaPraias	75,5
InfoCarvalho	60

Fonte: elaborada pelo autor.

Comparada à média/mediana das 110 interfaces de 26,5 pontos, em geral, os *apps* desenvolvidos pelos alunos se destacam por um grau de estética bem maior. De forma geral, as interfaces dos *apps* dos alunos destacam-se por um sistema de cores bem-definido, layout equilibrado e imagens adequadas.

A avaliação dos artefatos criados no processo de desenvolvimento documentado nos *workbooks*, utilizando a rubrica da quadro 28, em geral apresentou bons resultados (Anexo 2). Todos os pares de alunos aprenderam a

buscar soluções de problemas de forma coletiva e colaborativa. Esta aprendizagem foi relatada no *workbook* ao descrever um problema corretamente e quem é afetado por ele. As avaliações constataam que todos os alunos aprenderam como representar os tipos de usuários do app ao caracterizar uma persona. Percebe-se também que todos os alunos compreenderam como especificar uma funcionalidade focada no usuário, ao descrever corretamente as histórias de usuários. Os alunos tiveram dificuldades de relatar os passos de interação dos seus apps, sendo que 3 grupos não souberam responder e dos outros 2, apenas 1 soube responder completamente. Entende-se que os alunos tiveram dificuldade em visualizar o seu app funcionando em cada passo a ser construído ainda no processo, e por este fato não conseguiram descrever os passos de interação de seus apps. Pelo mesmo motivo, os alunos tiveram dificuldade em definir alguns resultados de usabilidade. Os tópicos relacionados ao design de interface foram descritos de forma correta, já que os alunos analisaram o funcionamento dos elementos de interface de usuário do app e quais as possíveis melhorias. A maioria dos alunos conseguiu aplicar os testes funcionais e de usabilidade ao descrever os casos de testes e seus resultados, assim como validaram se os requisitos de usabilidade foram atendidos. Dentre os apps, o que mais se destacou foi o AcheiOSEuEmprego, o qual apresentou melhor detalhamento dos tópicos dos *workbooks* e atingiu a maior nota (Tabela 6).

Tabela 6: Resultado das avaliações de desempenho dos artefatos dos apps criados pelos alunos

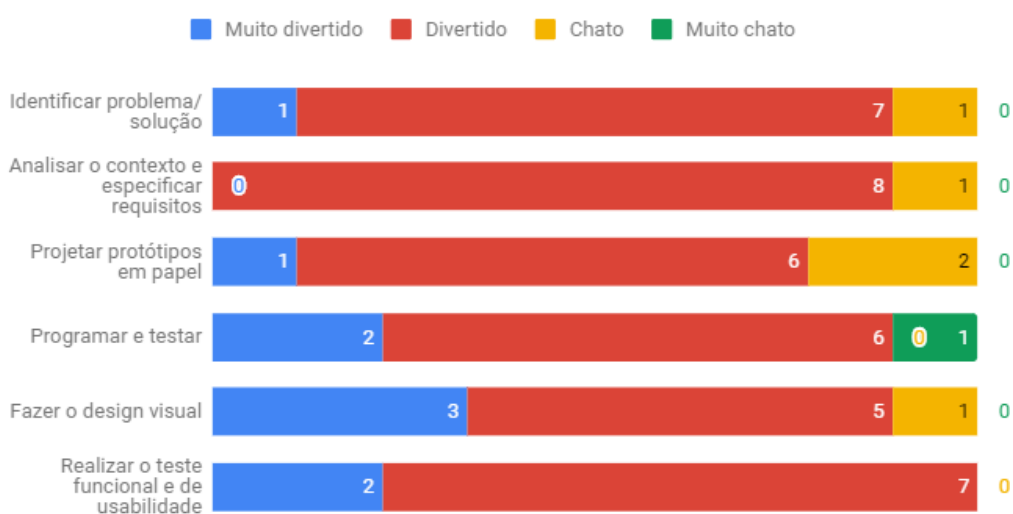
<i>App</i>	Pontuação	Nota (0-10)
InfoCarvalho	26	8,7
HealthyPlants	21	7,0
AppPhone	24	8,0
Floripa Praias	25	8,3
AcheiOSEuEmprego	27	9,0

PA2. A unidade instrucional promove uma experiência de aprendizagem agradável e divertida?

A experiência de aprendizagem dos tópicos da unidade instrucional foi positivamente avaliada, conforme as respostas obtidas nos pós questionários de autoavaliação. De modo geral, os alunos ficaram satisfeitos ao acharem que aprender estes tópicos é muito divertido ou divertido. Poucos alunos responderam negativamente ao achar chato algum tópico, e apenas um aluno achou o tópico programar e testar muito chato (Figura 79).

Figura 79: Distribuição da frequência de respostas relacionadas a diversão.

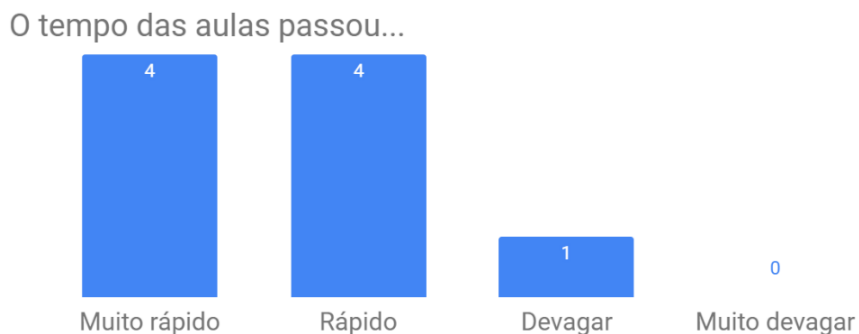
Eu achei...



Fonte: elaborada pelo autor.

A percepção positiva sobre a unidade instrucional também é identificada quando questionado aos alunos sobre o tempo das aulas. A maioria dos alunos respondeu que o tempo de aula passou muito rápido ou rápido (Figura 80). Essas respostas positivas podem estar ligadas à facilidade de acompanhar as instruções do instrutor no processo de desenvolvimento do aplicativo, e conseqüentemente, motivaram-nos a estarem mais entusiasmados e focados.

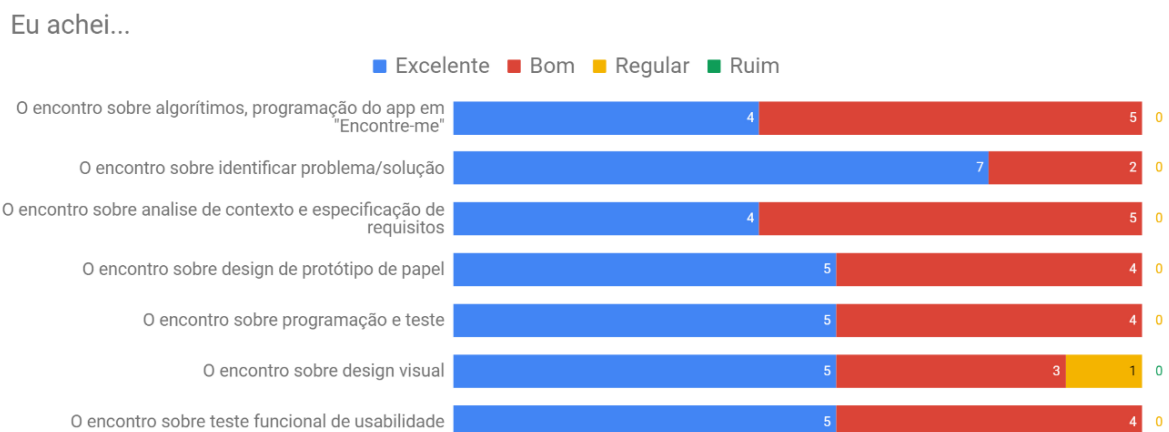
Figura 80: Frequência de itens relacionados ao tempo das aulas.



Fonte: elaborada pelo autor.

Além de demonstrarem que as aulas passaram rápidas, os alunos gostaram dos encontros presenciais, ao avaliarem que estes foram excelentes ou bons (Figura 81). Apenas um aluno achou o encontro sobre design visual regular.

Figura 81: Distribuição da frequência da avaliação da qualidade dos encontros.

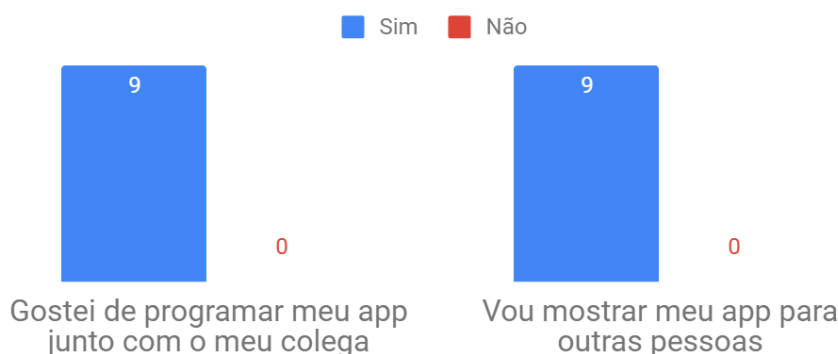


Fonte: elaborada pelo autor.

O aspecto de interação social também foi positivamente avaliado pelos alunos (Figura 82). Todos gostaram de aprender a programar o *app* junto com o colega, e notou-se nas aulas que todos os alunos gostaram da ideia de programar em par. Frequentemente, no par de alunos, estes estavam interagindo com o amigo para ajudar na aprendizagem ou compartilhar ideias referente ao *app*. Além disso, todos os alunos pretendem compartilhar o seu *app*, mostrando-o para outras pessoas, demonstrando assim que a unidade instrucional obteve resultados positivos.

Figura 82: Resultado dos itens relacionados a interação social.

Interação social



Fonte: elaborada pelo autor.

Em geral, os alunos tiveram uma boa experiência durante a unidade instrucional (Quadro 44). A aula voltada para o design foi considerada por muitos alunos a melhor parte. Alguns alunos gostaram tudo relacionado ao curso, um aluno indicou gosto por programar e um pela criação de um *app*. O que os alunos menos gostaram foi da parte de buscar informações para apresentar no *app*, construir o protótipo em papel e responder os questionários de avaliação. Vários alunos indicaram que não houve nada de que não gostassem no curso.

Quadro 44: Comentários qualitativos (números indicando a frequência de citações).

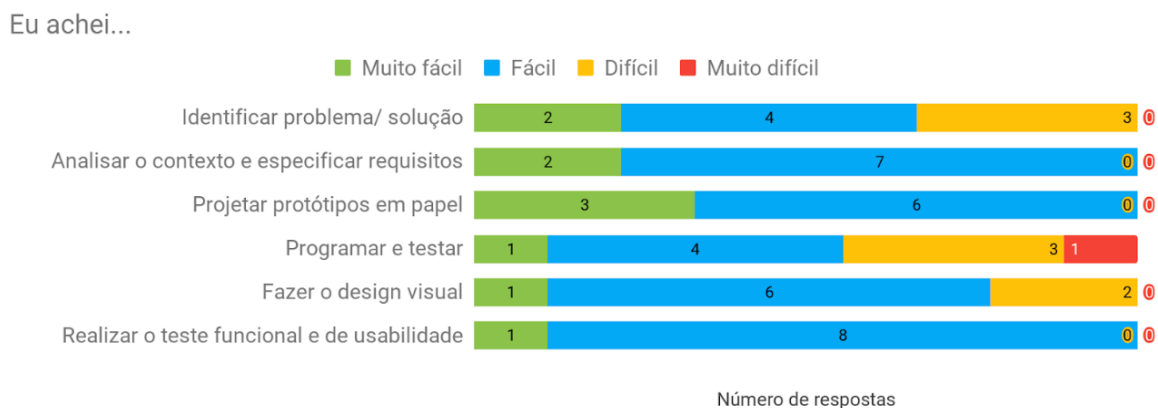
O que mais gostei no curso de fazer um <i>app</i> foi?	Design Visual, paleta de cores etc. Programar Aprender a fazer um <i>app</i> Design (2) Tudo (3) Design visual
O que menos gostei no curso de fazer um <i>app</i> foi?	Escolher as plantas e pesquisar descrição delas Nada (5) Protótipo de papel Fazer questionário Pesquisar plantas, e descrições

Fonte: elaborado pelo autor.

PA3. A unidade instrucional facilita a aprendizagem?

Por meio dos questionários após as unidades, os alunos avaliaram o nível de facilidade de cada um dos tópicos abordados na UI. De modo geral, a maioria dos alunos considerou que os tópicos foram muito fáceis ou de fácil aprendizagem, exceto o tópico programar e testar, para o qual alguns alunos acharam difícil ou muito difícil. Observou-se durante as aulas que os alunos demonstraram dificuldade em utilizar alguns componentes do *App Inventor* que não foram abordados nas primeiras aulas de programação, como mapa, lista e banco de dados. Com ajuda dos mentores e instrutores, e utilizando mini-tutoriais que abordam especificamente cada um desses componentes, as dúvidas foram sanadas. Alguns alunos também reconheceram dificuldade em identificar um problema e solução, e algum aspecto sobre design visual (Figura 83).

Figura 83: Distribuição da frequência de itens relacionados a sua facilidade de aprendizagem.

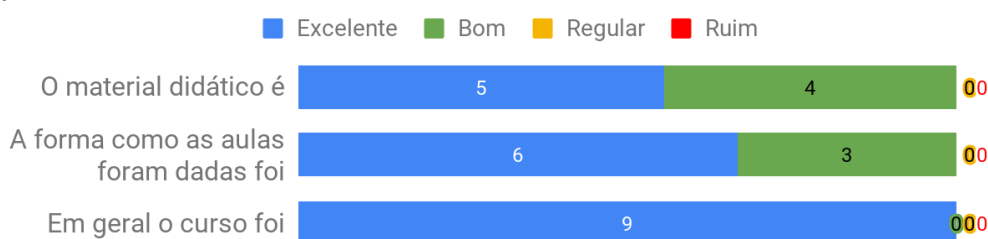


Fonte: elaborada pelo autor.

A qualidade das aulas e materiais utilizados foram avaliados de maneira positiva (Figura 84). Um dos motivos pelo qual a maioria dos alunos considerou excelente ou bom o material instrucional pode ter relação com a facilidade em acompanhar e compreender as instruções dos materiais. A forma como as aulas foram dadas também agradou aos alunos. Em relação ao curso em geral, todos os alunos consideraram que foi excelente. Esses resultados positivos podem estar relacionados com a quantidade satisfatória de mentores e instrutores em relação à quantidade de alunos. Assim, os alunos receberam auxílio em todos os momentos de dificuldade.

Figura 84: Distribuição da frequência de itens relacionados às aulas e o curso em geral.

O que você achou?

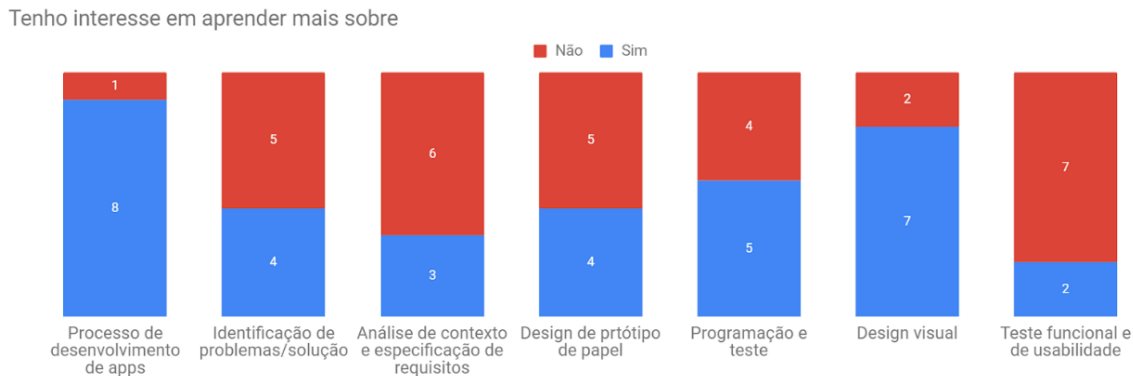


Fonte: elaborada pelo autor.

PA4. A unidade instrucional aumenta o interesse dos alunos referente à aprendizagem de conteúdo da computação?

Ao final da unidade instrucional os alunos demonstraram um interesse parcial em aprender mais sobre os conteúdos ensinados (Figura 85). Os alunos demonstraram um maior interesse em aprender mais sobre o processo desenvolvimento de *apps* como um todo. São poucos os que gostariam de aprender sobre tópicos específicos do processo. O único tópico no qual a maioria dos alunos tem interesse em aprender mais é sobre o design visual. Este é um assunto que atrai os alunos, pois notou-se desde o início da aula de programação que os alunos se preocupavam com aspectos de interface de tela, como definir cores, layout e imagens. Além disso, este tópico contém atividades recreativas e que tornam a aprendizagem prazerosa, como colorir um círculo cromático para definir a paleta de cores e definir as imagens para o aplicativo. Observa-se também que as aulas que tinham menos atividades lúdicas foram as que despertaram menos interesse em aprender mais, incluindo análise de contexto e requisitos e testes funcionais e de usabilidade.

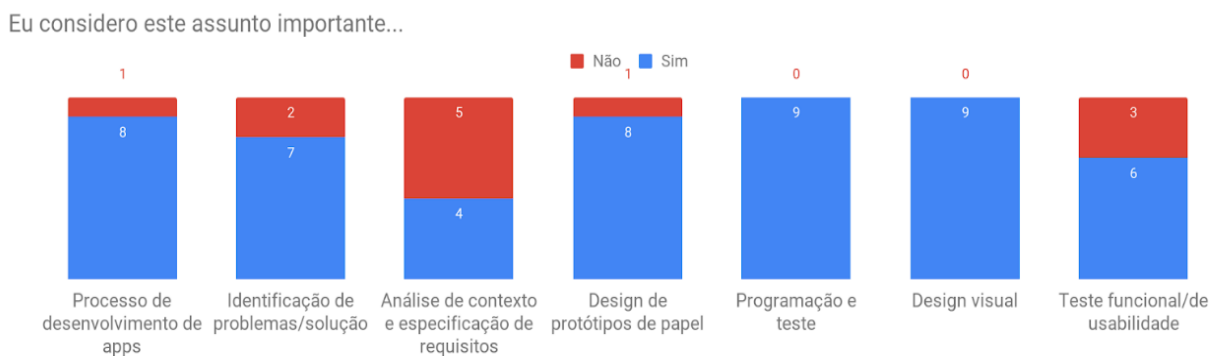
Figura 85: Distribuição da frequência de itens relacionados ao interesse na aprendizagem.



Fonte: elaborada pelo autor.

Em relação à consideração de importância dos tópicos ensinados, os alunos responderam em geral que reconhecem o seu valor (Figura 86). Apenas a análise do contexto e especificação dos requisitos foram tópicos considerados não tão importantes pela maioria dos alunos. Este resultado pode estar relacionado ao fato de o aluno não entender o propósito de se realizar análise de contexto e especificar os requisitos.

Figura 86: Distribuição da frequência de conteúdo relacionado a sua importância.



Fonte: elaborada pelo autor.

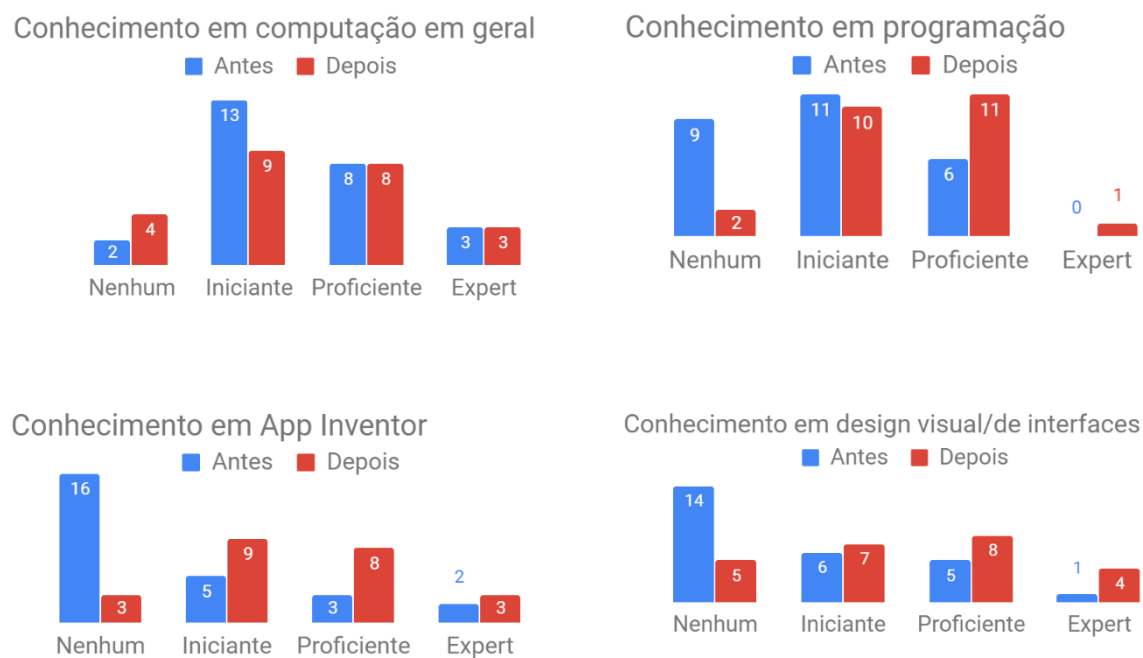
6.3.2. Análise dos dados da aplicação da unidade instrucional versão curta

Os dados analisados foram retirados de questionários pré- e pós-aplicação, e tem o propósito de responder as perguntas de análise.

PA1. A unidade instrucional aumenta competências dos alunos referentes a aprendizagem de conteúdos de computação?

O nível de aprendizagem referente à computação em geral não demonstrou evolução. Entretanto, para conhecimentos específicos, como programação, *App Inventor* e design *interface* houve um resultado significativo, no qual os alunos consideraram que obtiveram algum conhecimento (Figura 87).

Figura 87: Percepção de aprendizagem dos conteúdos.

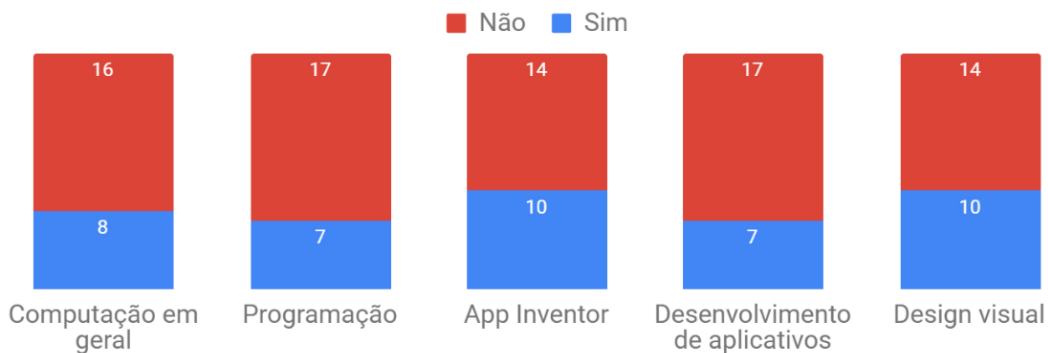


Fonte: elaborada pelo autor.

Além do nível de conhecimento, foi observado o nível de compreensão ao avaliar se os alunos conseguem explicar para um colega os tópicos da UI. Em todos os assuntos, a maioria dos alunos não conseguiu explicar, deixando dúvidas quanto ao nível de aprendizagem obtido durante a unidade (Figura 88).

Figura 88: Quantidade de alunos que conseguem explicar tópicos da UI.

Eu consigo explicar para um(a) amigo(a)



Fonte: elaborada pelo autor.

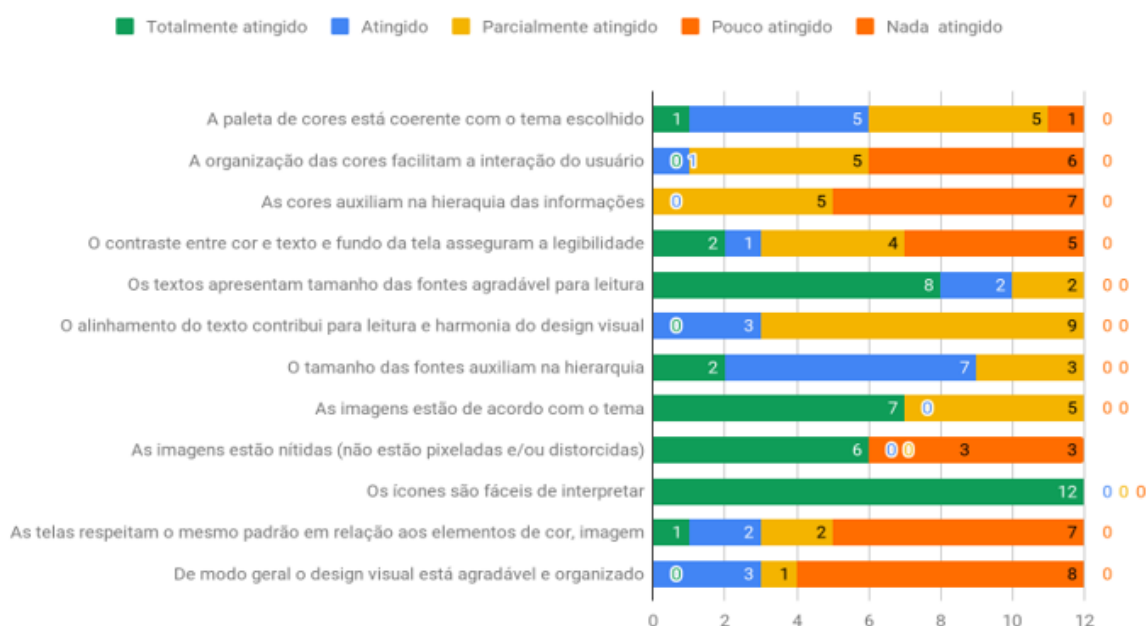
Como resultado da UI, os alunos criaram, em pares, o design visual de um *app* com *App Inventor*. Para avaliar o desempenho dos artefatos criados, foi utilizada a rubrica da quadro 26. O resultado desta análise está apresentado na Figura 89. De acordo com os critérios de análise, os alunos tiveram dificuldade em aplicar os meta-princípios do design em seus projetos. Percebe-se que a maioria dos aplicativos apresentaram cores saturadas e excesso de elementos, prejudicando a identificação da personalidade, hierarquia e consistência.

Constatou-se que a cor é o elemento que os alunos tiveram maior dificuldade de escolher e aplicar de forma harmônica. Foi possível observar que, em alguns casos, os alunos escolheram as cores de acordo com suas preferências pessoais, e não levando em consideração o contexto do aplicativo.

Quanto às imagens, identificou-se que algumas duplas encontraram dificuldades, tanto no que diz respeito ao dimensionamento quanto à harmonia com a paleta de cores de seus *apps*. Os projetos que utilizaram somente a imagem da marca não tiveram esse problema.

Em geral, os aplicativos apresentaram hierarquia nos textos, utilizando pesos e estilos diferentes. Somente em uma única tela, em todos aplicativos, observou-se o uso de formatação pouco recomendada para blocos de texto: alinhamento centralizado e itálico (Figura 89).

Figura 89: Frequência dos níveis de desempenho do design visual.



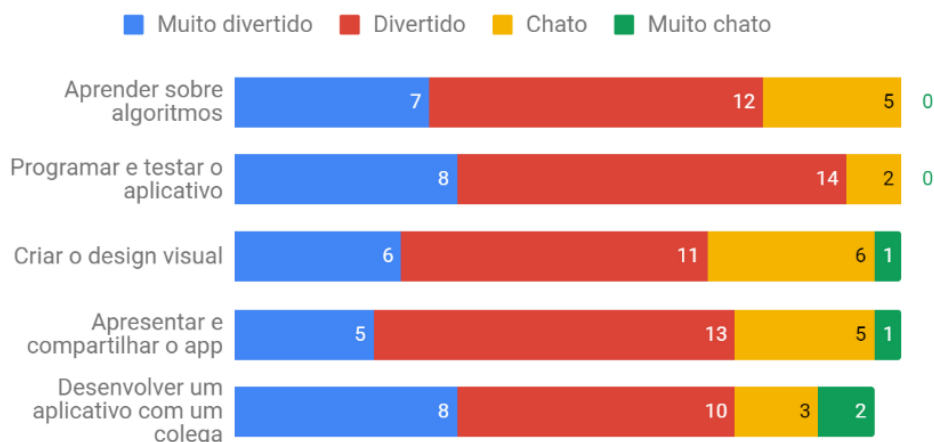
Fonte: elaborada pelo autor.

PA2. A unidade instrucional promove uma experiência de aprendizagem agradável e divertida?

Em geral a experiência de aprendizado foi positiva (Figura 90). Os alunos demonstraram satisfação significativa, e em sua grande maioria acharam divertido aprender os conteúdos abordados. Poucos alunos apresentaram respostas negativas em relação aos temas, o que representa algo natural pela diversidade dos alunos. A criação do design visual e o compartilhamento do *app* foram os tópicos mais rejeitados pelos alunos, enquanto programar e testar e desenvolver o *app* com um colega foram as melhores avaliações.

Figura 90: Distribuição da frequência de respostas relacionadas a diversão.

Eu achei...

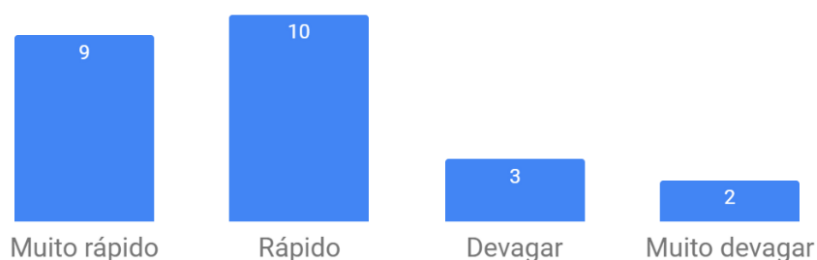


Fonte: elaborada pelo autor.

As respostas dos alunos indicam que eles não sentiram o tempo passar durante as aulas. A maioria dos alunos achou que as aulas passaram “rápido” ou “muito rápido”. Esse resultado indica que a maioria dos alunos ficou concentrada no conteúdo das aulas, não notando a passagem do tempo (Figura 91).

Figura 91: Frequência de itens relacionados ao tempo das aulas.

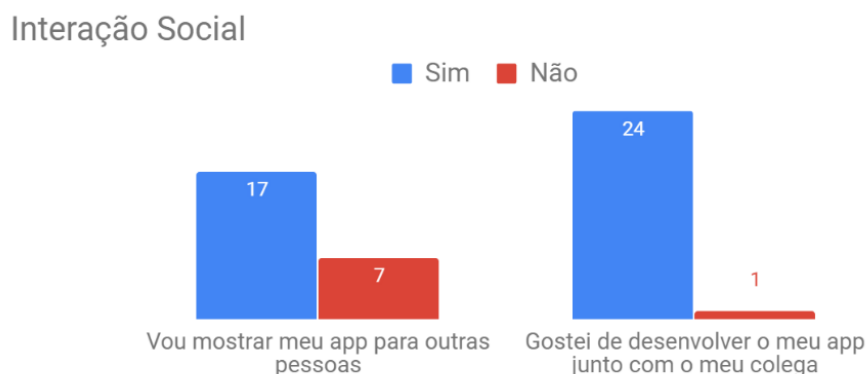
O tempo das aulas passou...



Fonte: elaborada pelo autor.

A maior parte dos alunos demonstrou satisfação e conforto em desenvolver os seus *apps* junto com um colega, existindo poucas exceções em que preferiram fazer a atividade sozinhos. Os alunos também avaliaram muito positivamente a apresentação e compartilhamento dos resultados (Figura 92).

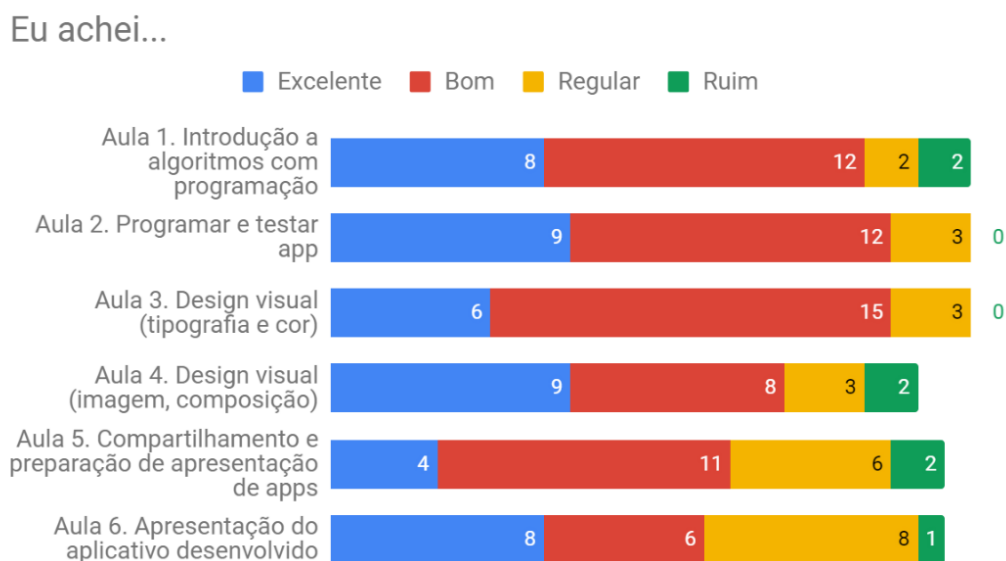
Figura 92: Resultado dos itens relacionados a interação social.



Fonte: elaborada pelo autor.

Comparando as aulas da UI, observa-se que a maioria dos alunos considerou as aulas sobre o design visual como excelente ou boa (Figura 93).

Figura 93: Distribuição da frequência da avaliação da qualidade dos encontros.



Fonte: elaborada pelo autor.

Os comentários dos alunos indicaram também uma grande apreciação por todas as etapas envolvidas na criação de um *app* (Tabela 7). Destacou-se a parte do design do *app* por ser mencionado por vários alunos como o que mais gostaram na UI. Também indicaram gosto por programar e pela criação de um *app*. Quanto à parte que menos gostaram, segundo o questionário, foi buscar informações para apresentar no *app*, construir o protótipo em papel (e responder os questionários de

avaliação). Ainda, vários alunos indicaram que não houve nada de que não gostassem no curso.

Tabela 7: Comentários qualitativos (números entre parênteses indicam a frequência de citações por cada categoria).

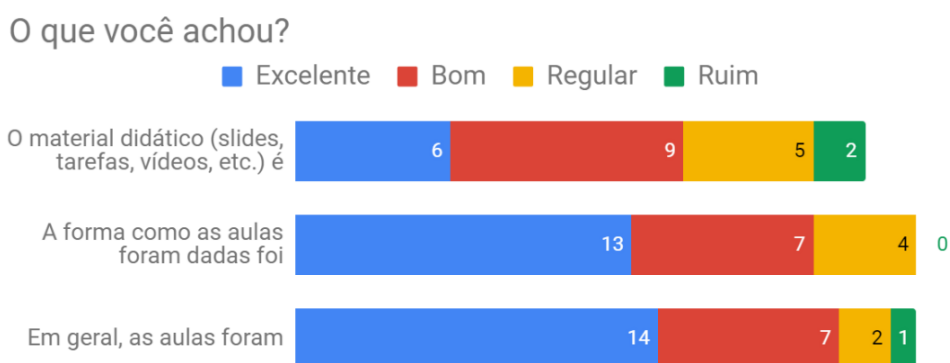
O que menos gostei de fazer um <i>app</i> foi	O que mais gostei de fazer um <i>app</i> foi
Nada (6)	Colocar e tirar as fotos (7)
Foto (5)	Tudo (5)
Escrever (4)	Personalizar o <i>app</i> (cores, legendas e ícones) (4)
Tudo (2)	Programação (1)
Tirar a foto, fazer os textos e das explicações demoradas (1)	Bom (1)
Fazer com um colega (1)	Nada (1)
Tirar foto (1)	Explicação dos professores (1)
Editar a foto com meus amigos (1)	Escrita (1)
Os slides (1)	Inventar o <i>app</i> (1)
	Conversar (1)

Fonte: elaborada pelo autor.

PA3. A unidade instrucional facilita a aprendizagem?

Os alunos avaliaram a qualidade das aulas e dos materiais utilizados de maneira muito positiva (Figura 94). A maioria dos alunos considerou “excelente” ou “bom” o material instrucional e também a forma como as aulas foram realizadas.

Figura 94: Distribuição da frequência de itens relacionados às aulas e o curso em geral.

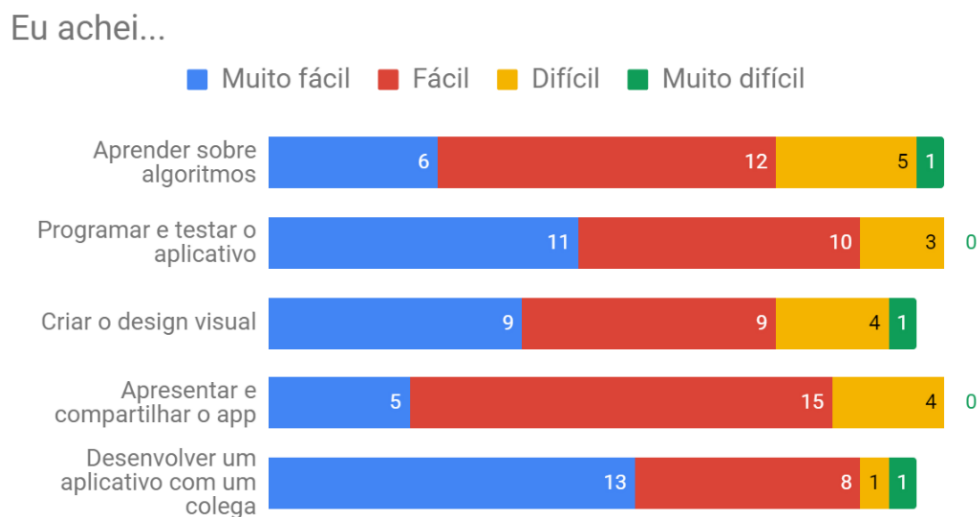


Fonte: elaborada pelo autor.

A grande maioria dos alunos considerou os tópicos ensinados “fáceis” de aprender, indicando confiança no que estavam desenvolvendo, e também que conseguiram compreender realmente o que foi ensinado (Figura 95). Os tópicos

“aprender sobre algoritmos” e “criar o design visual” foram os que os alunos sentiram maiores dificuldades.

Figura 95: Distribuição da frequência de itens relacionados a sua facilidade de aprendizagem.

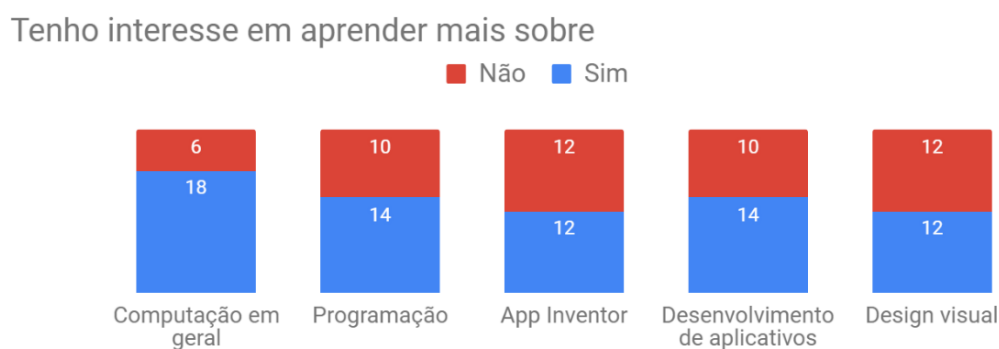


Fonte: elaborada pelo autor.

PA4. A unidade instrucional aumenta o interesse dos alunos referente à aprendizagem de conteúdo da computação?

Ao final da unidade instrucional os alunos demonstraram um interesse parcial em aprender mais sobre os conteúdos ensinados (Figura 96). A computação em geral foi o tópico em que os alunos mais demonstraram interesse. Em relação aos outros conteúdos do curso, de acordo com os dados, houve divergência entre opiniões.

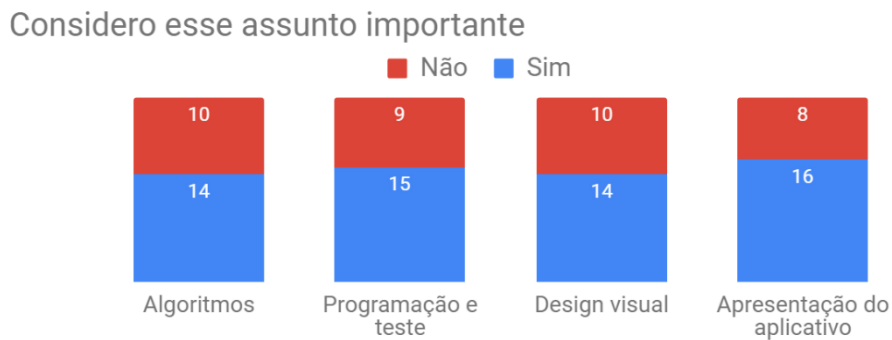
Figura 96: Distribuição da frequência de itens relacionados ao interesse na aprendizagem.



Fonte: elaborada pelo autor.

A maioria dos alunos conseguiu compreender a importância dos assuntos envolvidos no curso (Figura 97). O assunto que os alunos mais consideraram importante foi a apresentação do *app*, enxergando que uma boa apresentação faz um *app* destacar-se entre os demais.

Figura 97: Distribuição da frequência de conteúdo relacionado a sua importância.



Fonte: elaborada pelo autor.

6.4. DISCUSSÃO

Os resultados da análise dos dados demonstraram que a maioria dos alunos avaliou positivamente a UI. Em ambas as aplicações, os alunos consideraram a participação no curso uma experiência agradável e divertida. Os alunos se demonstraram entusiasmados e curiosos com cada etapa do desenvolvimento do aplicativo. Isto demonstra que as atividades do curso foram percebidas mais como uma atividade lúdica e menos como uma atividade instrucional. No geral, as aplicações demonstraram resultados expressivos, sendo que a maioria dos alunos achou as aulas “muito divertida” ou “divertida”, e se sentiu envolvida no curso ao perceber que as aulas passaram rápidas demais. A estratégia das aulas foi predominantemente realizada por meio de instrução passo-a-passo e atividades práticas imediatas. Esta estratégia pode ter ajudado a manter os alunos imersos no processo de aprendizagem.

Todos os alunos conseguiram desenvolver um aplicativo e os resultados demonstram que houve um aumento expressivo nas competências dos alunos referente a diversos conteúdos de computação. A utilização de jogo de tabuleiro para ensinar algoritmo no início do curso teve um impacto positivo na motivação do aluno, além de contribuir positivamente para a interação social e diversão. A

utilização da ferramenta de programação baseada em blocos, o *App Inventor*, também contribuiu na experiência de aprendizagem e no aprendizado de programação. Esta linguagem permitiu aos alunos o desenvolvimento de programas de forma mais fácil. Os resultados evidenciaram que houve um aumento de aprendizado expressivo em programação em ambas as aplicações.

Além da programação, a unidade instrucional contribuiu para ensinar outras competências relacionadas à computação. Nas duas aplicações os alunos seguiram um processo de desenvolvimento e obtiveram um acréscimo em seu aprendizado referente à disciplina de EU. Na versão longa da UI, os alunos criaram seu próprio aplicativo seguindo etapas de um processo baseado no *design thinking*. Esta abordagem incentivou a criatividade e a imaginação dos alunos, para desenvolver soluções inteligentes e empáticas para um problema. Seguindo este processo todos os alunos conseguiram criar seu próprio aplicativo, para que ele possa ser útil e ajudar em suas vidas e em suas comunidades. Os resultados demonstram também que os alunos têm um maior interesse em aprender sobre design de interface de usuário, juntamente com a programação. Este resultado pode estar relacionado à empolgação que alunos têm neste nível de ensino, e em deixar o aplicativo com uma interface de usuário esteticamente atraente. Nota-se que os *apps* criados pelos jovens tutores se destacaram por ter um grau de estética maior que os *apps* da galeria do *App Inventor*. Porém, na aplicação curta, o desempenho dos alunos em relação ao design visual dos *apps* obteve um resultado abaixo do esperado. Percebe-se que os alunos tiveram dificuldades em aplicar alguns conceitos de design nos *apps*. Infere-se que esta dificuldade pode estar relacionada à dificuldade de compreender e assimilar os conteúdos, devido à ausência de conhecimento prévio. Contata-se também que a faixa etária dos alunos nessa aplicação foi menor, fazendo-se necessário adaptar o conteúdo no que se refere à linguagem visual para simplificar o material. Além disso, nesta faixa etária os alunos se dispersam frequentemente, e assim, os instrutores tinham dificuldade em manter o foco dos alunos por muito tempo. Apesar disso, em ambas aplicações, observamos que durante as apresentações a maioria dos alunos demonstrou estar orgulhosa e satisfeita com o *app* desenvolvido. Poucos alunos mostraram alguma insatisfação e/ou relataram a percepção de emoções negativas, sem maiores explicações.

Na aplicação longa os alunos não demonstraram dificuldades referentes às competências que envolvem a ES. Desenvolver o *app* seguindo abordagens ágeis,

como o desenvolvimento interativo e incremental, e a programação em pares foi absorvido e realizado de forma natural pelos alunos. Vale ressaltar que os alunos gostaram de trabalhar em pares e se divertiram enquanto aprendiam. Com base nas avaliações de desempenho, principalmente nos artefatos criados, a maioria dos alunos soube definir os requisitos funcionais dos *apps* ao descrever história de usuários. Porém, durante a aula de especificação de requisitos, notou-se que os alunos demonstraram uma insatisfação por ser um assunto com muitos conceitos e atividades escritas, como a documentação dos requisitos. Por ser um assunto predominantemente teórico, os alunos ficaram frequentemente dispersos e desconcentrados, demonstrando uma possível insatisfação.

A adoção de um processo sistemático abordando tópicos de ES e EU estimulou os alunos a desenvolverem diversas habilidades do século XXI, como por exemplo a criatividade, o pensamento crítico, a resolução de problemas, a comunicação e a colaboração. Assim, a unidade instrucional demonstrou resultados promissores em termos de aprendizagem de computação e impactando positivamente na motivação, na experiência do usuário e na aprendizagem, como também em empoderar os alunos, que assim tornam-se criadores ativos de produtos de TI.

6.5. AMEAÇAS À VALIDADE

O estudo de caso aplicado nesta pesquisa pode apresentar vários problemas em relação à validade dos resultados, e algumas das ameaças estão relacionadas ao seu projeto. A fim de mitigar estas ameaças, foi definida e documentada uma metodologia sistemática para o estudo, usando a abordagem GQM (BASILI; CALDIERA; ROMBACH, 1994). Outra ameaça é a falta de um *benchmark*, pré-testando o conhecimento dos alunos antes do curso, para fazer uma comparação e análise das diferenças. Isso é ainda mais exacerbado pelo fato de que não há grupo controle para comparar os efeitos identificados. No entanto, devido às características deste projeto específico, só foi possível realizar dois estudos de caso, em vez de um experimento. Além disso, o fato de a unidade instrucional ter sido aplicada apenas com um pequeno conjunto de participantes de cada vez, e dentro da mesma escola, reduz a possibilidade de generalização dos resultados. Entretanto, considerando a natureza exploratória de nossa pesquisa neste momento, consideramos aceitável o rigor científico de um estudo de caso cuidadosamente definido. Outra possível

ameaça é que aspectos como diversão e satisfação são difíceis de medir, e são capturados por medidas subjetivas. Para contornar essa ameaça à validade, os itens dos questionários foram sistematicamente derivados em base no instrumento de medição padronizado MEEGA+, que foi avaliado em termos de validade e confiabilidade em larga escala (PETRI *ET AL.*, 2017).

7.CONCLUSÃO

O objetivo do presente trabalho foi o desenvolvimento de um modelo educacional para ensinar, de forma motivadora, os conhecimentos de ES/EU por meio do desenvolvimento de aplicativos móveis para alunos do Ensino Básico e utilizando o ambiente de programação visual *App Inventor*. Nesse contexto, foi feita uma síntese da fundamentação teórica sobre o ensino de computação no Ensino Fundamental, o ambiente de programação visual *App Inventor*, e ES e EU (Objetivo 1). Além disso, foi analisado o estado da arte por meio de um mapeamento sistemático referente à ES (PINHEIRO *et al.*, 2018) e ao design de interface (FERREIRA *et al.*, 2018) (Objetivo 2). Como resultado desses mapeamentos, foi evidenciado a ausência de UIs abordando de forma sistemática competências de ES e EU no ensino da computação neste nível de ensino.

Como primeiro passo para o desenvolvimento do modelo educacional foi feita uma análise de contexto dos alunos e das escolas. Com base nesta análise, foi definido o plano de ensino, incluindo os objetivos de aprendizado a serem abordados (Objetivo 3.1). Seguindo um processo de design instrucional, foi projetada e desenvolvidos todos os materiais didáticos necessários. Para guiar os alunos a desenvolver seu próprio *app*, também foi desenvolvido um modelo de processo de desenvolvimento de software baseado no *Design Thinking* (Objetivo 3.2). Neste processo são abordadas práticas de ES e EU, incluindo a história de usuário, os testes de sistemas, a análise de contexto, o design visual, etc. Diversos materiais instrucionais foram criados para a realização das aulas, como por exemplo, slides, jogos lúdicos (GRESSE VON WANGENHEIM *et al.*, 2019), tarefa de casa e *workbooks* (Objetivo 3.3). Com o propósito de fornecer suporte ao processo desenvolvimento, a ferramenta *App Inventor* foi customizada, incluindo funcionalidades para que o aluno possa definir cores, ícones, histórias de usuários e *personas* para o seu aplicativo (Objetivo 3.4).

A unidade foi aplicada em escola pública e em duas turmas, a primeira em uma turma do 8º e 9º ano, e a segunda em uma turma do 5º ano do Ensino Fundamental (FERREIRA *et al.*, 2019; FERREIRA *et al.*, 2019a). A qualidade da unidade instrucional foi avaliada pelos alunos e apresentou resultados positivos nas duas aplicações. Na primeira aplicação, os alunos compreenderam todas as competências ensinadas, e todas as expectativas foram alcançadas. Os aplicativos criados pelos alunos foram muito além do esperado, especialmente ao comparar a

qualidade do design da interface com outros aplicativos criados com o *App Inventor*. Porém, na segunda aplicação, analisando o desempenho dos alunos em relação ao design visual criado por eles, observou-se um resultado abaixo do esperado. Percebe-se que a unidade instrucional precisa ser adaptada para a idade, além de aumentar o tempo de aula, para que o aluno possa assimilar o conteúdo.

Como resultado do presente trabalho, é disponibilizado uma unidade instrucional para ensinar computação por meio de um processo sistemático de desenvolvimento de aplicativos utilizando o *App Inventor*, e integrando práticas de ES/EU. Essa unidade pode ser aplicada no ensino básico, a partir de 12 anos, com ou sem experiência em computação. Este trabalho também disponibiliza as customizações realizadas na ferramenta *App Inventor*, como a funcionalidade paleta de cores, catálogo de ícones e a documentação de personas e histórias de usuário. Estas customizações permitem introduzir práticas de ES/EU no desenvolvimento de um *app*. Dessa forma, escolas de Educação Básica poderão ensinar computação de forma mais abrangente utilizando a unidade instrucional desenvolvida, incluindo todo material didático e a versão aprimorada do *App Inventor*.

Como trabalhos futuros, sugere-se a inclusão de novas funcionalidades na ferramenta *App Inventor* para um melhor suporte ao ensino de ES e EU. Por exemplo, funcionalidades para documentar requisitos funcionais e de usabilidade, testes de sistemas, e também funcionalidades voltadas a seguir o guia de estilo do *Material Design* no que se refere a *layout*, tipografia, navegação, etc. A inclusão de rubricas de avaliação na ferramenta CodeMaster, para avaliar essas funcionalidades, ajudaria no processo de avaliação de desempenho dos alunos. Em relação ao conteúdo da UI, um estudo sobre linguagens visuais pode ser aplicado em algumas aulas de natureza conceitual, como por exemplo na aula de análise de contexto e especificação de requisitos. A inclusão de desenho, vídeos e outros artefatos visuais pode simplificar o conteúdo e ajudar os alunos na sua compreensão.

REFERÊNCIAS

ABNT - ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR ISO/IEC 12207:2009. (2009). **Engenharia de Sistemas e Software - Processos de Ciclo de Vida de Software**. Rio de Janeiro, RJ, Brasil.

ABNT - ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR ISO/IEC 9241-210:2011. (2011). **Ergonomia da Interação Humano-sistema**: Parte 210: Projeto centrado no ser humano para sistemas interativos. Rio de Janeiro, RJ, Brasil.

ABRAHAMSSON, Pekka *et al.* (2004). Mobile-D: An Agile Approach for Mobile Application Development. In: ANNUAL ACM SIGPLAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING SYSTEMS, LANGUAGES, AND APPLICATIONS, 19 ed., ACM. p. 174 - 175.

ABRAHAMSSON, Pekka. (2005). Keynote: Mobile software development - the business opportunity of today. In: INTERNATIONAL CONFERENCE ON SOFTWARE DEVELOPMENT, Reykjavik, Iceland. **Proceedings....** Reykjavik: 2005. p. 20 - 23.

ABRAHAMSSON, P *et al.* (2017). **Agile software development methods**: Review and analysis. VTT Publications.

ACM/IEEE. (2013). **Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science**. New York, NY, USA: ACM.

ACUÑA, Silvia Teresita; XAVIER, Ferré. Software Process Modelling. (2001). In: WORLD MULTICONFERENCE ON SYSTEMICS, CYBERNETICS AND INFORMATICS., 2001, Orlando. Orlando, Flórida, EUA, v. 1, p. 237 - 242.

AIGA. (2019). **Graphics design training curriculum for high school teachers**. Disponível em: <<https://www.aiga.org/aiga/content/events-and-competitions/competitions/graphic-design-training-curriculum-for-high-school-teachers/>>. Acesso em: Setembro de 2019.

AIGA. (2013). *The Professional Association for Design*. Disponível em: <<https://www.aiga.org/>>. Acesso em: 16 de abril de 2018.

ALVES, Nathalia da Cruz; GRESSE VON WANGENHEIM, Christiane; HAUCK, Jean Rauck. (2019). **A Large-scale Analysis of App Inventor Projects**. Submetido para publicação.

ARHIPAINEN, Leena; TÄHTI, Marika. (2003). Empirical Evaluation of User Experience in Two Adaptive Mobile Application Prototypes. In: INTERNATIONAL CONFERENCE ON MOBILE AND UBIQUITOUS MULTIMEDIA, 2 ed, Linköping, Suécia. **Proceedings....** Linköping University: Electronic Press. p. 27 - 34.

AZENKOT, Shiri *et al.* (2011). Overcoming barriers among Israeli and Palestinian students via computer science. In: TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION, 42 ed, Dallas. **Proceedings....** Dallas, TX, EUA: ACM. p. 667 - 672.

AZIZ, Nor Azah Abdul; RASLI, Roznim Mohamad; RAMLI, Khairunnisa. (2010). Preschool multimedia interactive courseware : Classifying object (mengelaskan objek). In: WRI WORLD CONGRESS ON SOFTWARE ENGINEERING, 2 ed, Wuhan. **Proceedings....** Wuhan, China: WRI, 2010. p. 318 - 322.

BASILI, Victor R.; CALDIERA, Gianluigi; ROMBACH, H. Dieter. (1994). The Goal Question Metric Approach. **Encyclopedia Of Software Engineering**, New Jersey: John Wiley & Sons, Inc. p. 528-532.

BEKKER, T. *et al.* (2015) Teaching children digital literacy through design-based learning with digital toolkits in schools. **International Journal of Child-Computer Interaction**, v. 5, p. 29-38.

BLIKSTEIN, Paulo. (2008). “**O Pensamento Computacional e a Reinvenção do Computador na Educação**”. Disponível em: <<http://bit.ly/1IXIbNn>>. Acesso em: Junho de 2017.

BLOOM, Benjamin S. (1956). **Taxonomy of Educational Objectives: The Classification of Educational Goals**. United States: Addison-Wesley Longman Ltd. (Handbook I: Cognitive Domain Edition).

BOLLIN, Andreas; SABITZER, Barbara. (2015). Teaching Software Engineering in schools on the right time to introduce Software Engineering concepts. In: GLOBAL ENGINEERING EDUCATION CONFERENCE, 2015, Talín, Estônia. **Proceedings....** Talín, Estônia. p. 518 - 525.

BRANCH, Robert Maribe (2009). **Instructional Design: The ADDIE Approach**. 2. ed. New York: Springer Science & Business Media. 203 p

BRENNAN, Karen, & RESNICK Mitchel. (2013). Imagining, creating, playing, sharing, reflecting: How online community supports young people as designers of interactive media. *Emerging Technologies for the Classroom: A Learning Sciences Perspective*, 253-268. Springer.

BURNETT, Charles. (2009). A theory of design thinking. In: TORQUAY CONFERENCE ON DESIGN THINKING, 2009, Swinburne University Of Technology. **Proceedings....** Melbourne, Australia: Swinburne University Of Technology.

CGI.BR (2018). **Cresce número de crianças e adolescentes que buscam notícias na Internet, aponta Cetic.br**. Disponível em: <<https://cgi.br/noticia/releases/cresce-numero-de-criancas-e-adolescentes-que-buscam-noticias-na-internet-aponta-cetic-br/>>. Acesso em: 17 de abril de 2019.

CAMBRAIA, Adão Caron; SCAICO, Pasqueline Dantas. (2013). Os desafios da Educação em Computação no Brasil: um relato de experiências com Projetos PIBID no Sul e Nordeste do país. **Revista Espaço Acadêmico**, v. 13, n. 148, p.01-09.

CARDOSO, Érico Edú Corrêa; DAVID, Tobias de. (2017). A falta de profissionais de tecnologia de informação no mercado de trabalho. In: CONGRESSO INTERNACIONAL: UMA NOVA PEDAGOGIA PARA A SOCIEDADE FUTURA, 2017, Rio Grande do Sul, RS. **Proceedings....** Rio Grande do Sul. p. 697 – 700.

CHATZINIKOLAKIS, Giorgos; PAPADAKIS, Spyros. (2014) Motivating K-12 students learning fundamental Computer Science concepts with App Inventor. In: INTERNATIONAL CONFERENCE ON INTERACTIVE MOBILE COMMUNICATION TECHNOLOGIES AND LEARNING, 2014. **Proceedings... .** IEEE. p. 152 - 159.

CHEN, Peng; HUANG, Ronghuai. (2017). Design thinking in App inventor game design and development: A case study. In: INTERNATIONAL CONFERENCE ON ADVANCED LEARNING TECHNOLOGIES (ICALT), 17 ed, 2017, Timisoara, Romania. **Anais....** Timisoara: IEEE. p. 139 - 141.

CGI.BR. (2018). **Cresce número de crianças e adolescentes que buscam notícias na Internet**. Disponível em: <<https://cgi.br/noticia/releases/cresce-numero-de-criancas-e-adolescentes-que-buscam-noticias-na-internet-aponta-cetic-br/>>. Acesso em: Julho de 2019.

CODE. (2013). **Code.org**. Disponível em: <<https://code.org/>>. Acesso em: Julho de 2017.

CODE CLUB. (2012). **Codeclubworld.org**. Disponível em: <<https://www.codeclubworld.org/>>. Acesso em: Julho de 2017.

CODEHS. (2018). Disponível em: <<https://codehs.com/info/curriculum> >. Acesso em: 16 de abril de 2018.

CODELIKEAGIRL. (2017). Disponível em: <<https://code.likeagirl.io/learning-design-by-makinggames-in-scratch-6b90cd9e8a83>>. Acesso em: 16 de abril de 2018.

CODE.ORG/APP LAB. (2018). Disponível em: <<https://code.org/educate/applab>>. Acesso em: 16 de abril de 2018.

COHN, Mike. (2004). **User stories applied: For agile software development**. Boston: Addison-wesley Professional. (Coleção Addison-Wesley Signature Series).

COLLOFELLO, James S. (2002). Creation, deployment and evaluation of an innovative secondary school Software development curriculum module. In: ANNUAL FRONTIERS IN EDUCATION, 32 ed, 2002, Boston. **Proceedings... .** Boston, MA: IEEE. p. 1 - 4.

COOPER, Alan. *et al.* (2014). **About face: the essentials of interaction design**. Indianápolis: John Wiley e Sons.

CORBETT, Krystal; NESIBA, Natasha. (2015). Programming design process: Providing K-12 students with a structure to attain programming goals. In: FRONTIERS IN EDUCATION CONFERENCE, 2015, El Paso. **Proceedings...** . El Paso, TX: IEEE. p. 1 - 4.

CNE. (2019). **Iniciativa Computação na Escola**. Disponível em:<<http://www.computacaonaescola.ufsc.br>>. Acesso em: Julho de 2017.

CREATELAB (2017). Disponível em: < <http://www.mycreatelab.com/ais/>>. Acesso em: 16 de abril de 2018.

CSTA. (2017). **CSTA K–12 Computer Science Standards, Revised**. The CSTA Standards Task Force - Revised, ACM, New York/USA: ACM.

CSTA. (2016). **CSTA K–12 Computer Science Framework**. The CSTA Standards, ACM, New York/USA: ACM.

CUNHA, Thiago Ferraz V. da; DANTAS, Valeria Li; ANDRADE., Rossana Mc. (2011). SLeSS: A Scrum and Lean Six Sigma integration approach for the development of software customization for mobile phones. In: BRAZILIAN SYMPOSIUM ON SOFTWARE ENGINEERING, 25 ed, São Paulo, Brasil. **Proceedings...** . São Paulo: IEEE, 2011. p. 283 - 292.

D'ANGELO, Pedro. (2018). **Pesquisa sobre smartphones: a relação dos pais e das crianças com smartphones no Brasil**. Disponível em: <<https://blog.opinionbox.com/pesquisa-smartphones-criancas/>>. Acesso em: junho de 2018.

DE KEREKI, Inés. F. & MANATAKI, Areti. (2018). Code Yourself! An introduction a programming. Disponível em: <<https://pt.coursera.org/learn/intro-programming>>. Acesso em: 29 de março de 2018.

DE KEREKI, Inés. F.; MANATAKI, Areti. (2016). “Code Yourself” and “A Programar”: a bilingual MOOC for teaching Computer Science to teenagers. In: Proc. of the Frontiers in Education Conference, Erie, Pensilvânia, EUA, p. 1-9.

DEHLINGER, Josh; DIXON, Jeremy. (2011). Mobile application software engineering: Challenges and research directions. In: WORKSHOP ON MOBILE SOFTWARE ENGINEERING. p. 29 - 32.

DENNER, Jill et al. (2005). The Girls Creating Games Program: Strategies for Engaging Middle-School Girls in Information Technology. **Frontiers: A Journal of Women Studies**, v. 26, n. 1, p.90-98. JSTOR. <http://dx.doi.org/10.1353/fro.2005.0008>.

DICK, Walter. & CAREY, Lou. (2006). **The systematic design of instruction**. New York: Haper Collins College Publishers.

D.SCHOOL. (2019). **Design Thinking Bootleg — Stanford d.school**. Disponível em: <<https://dschool.stanford.edu/resources/design-thinking-bootleg>>. Acesso em: Abril de 2019.

EDUTOPIA (2015). *Edutopia*. Disponível em: <<https://www.edutopia.org/blog/coding-by-designfirst-approach-douglas-kiang>>. Acesso em: 16 de abril de 2018

FEIJÓ, Valéria Casaroto; BALDESSA, M.; VIEIRA, M. L. (2013). Elementos De Design Para Interface De Apps Em Smartphones: O Iphone 4s. In: SIMPÓSIO NACIONAL DE GEOMETRIA DESCRITIVA, XXI ed, Florianópolis, Sc. **Anais...**. Florianópolis: 2013.

FERREIRA, M. N.F., MISSFELDT FILHO, R., PINHEIRO, F. D. C., GRESSE VON WANGENHEIM, C. & HAUCK, J. C. R. (2018). **Ensinando Design de Interface de Usuário na Educação Básica: Um Mapeamento Sistemático do Estado da Arte e Prática**. submetido para publicação.

FERREIRA, M. N. F., GONÇALVES, B. S. & WANGENHEIM, C. G. (2019). Design Visual para Interfaces de Aplicativos: Análise de Modelos de Referência. **Revista Educação Gráfica**, v. 23 p. 79-95.

FERREIRA, M. N.F., MISSFELDT FILHO, R., PINHEIRO, F. D. C., GRESSE VON WANGENHEIM, C. & HAUCK, J. C. R. (2019). **Ensinando Design de Interface de Usuário de Aplicativos Móveis no Ensino Fundamental**. Submetido para publicação.

FERREIRA, M. N.F., MISSFELDT FILHO, R., PINHEIRO, F. D. C., GRESSE VON WANGENHEIM, C. & HAUCK, J. C. R. (2019a). **Ensinando Design Visual de Aplicativos Móveis no Ensino Fundamental**. Submetido para publicação.

FILATRO, Andrea. (2004). **Design instrucional contextualizado: educação e tecnologia**. São Paulo: SENAC.

FRANCIS, Krista *et al.* 2017. Multidisciplinary Perspectives on a Video Case of Children Designing and Coding for Robotics. **Canadian Journal Of Science, Mathematics And Technology Education**, v. 17, n. 3, p.165-178, 23 mar. Springer Nature. <http://dx.doi.org/10.1080/14926156.2017.1297510>.

FLING, Brian. (2009). **Mobile Design and Development: Practical Concepts and Techniques for Creating Mobile Sites and Web Apps**. 1ª ed. Califórnia, EUA: O'Reilly Media.

FLORA, Harleen; CHANDE, Swati V.; WANG, Xiaofeng. (2014). Adopting an Agile Approach for the Development of Mobile Applications. **International Journal Of Computer Applications**, v. 94, n. 17, p.43-50. Foundation of Computer Science. <http://dx.doi.org/10.5120/16454-6199>.

FRONZA, Ilenia; IOINI, Nabil El; CORRAL, Luis. (2017). Teaching Computational Thinking Using Agile Software Engineering Methods: A Framework for Middle Schools. In: TRANSACTIONS ON COMPUTING EDUCATION, ACM, v. 17, n. 4.

FRONZA, Ilenia; IOINI, Nabil El; CORRAL, Luis. (2016). Blending Mobile Programming and Liberal Education in a Social-Economic High School. In: INT. CONFERENCE ON MOBILE SOFTWARE ENGINEERING AND SYSTEMS, Austin, Tx. **Proceedings...** . Austin: IEEE/ACM. p. 123 - 126.

FRONZA, Ilenia; IOINI, Nabil El; CORRAL, Luis. (2015). Students want to create apps: leveraging computational thinking to teach mobile software development. In: ANNUAL CONFERENCE ON INFORMATION TECHNOLOGY EDUCATION, 16 ed, 2015, Berlim, Alemanha. **Proceedings...** . Berlim: Acm. p. 21 - 26.

GAL-EZER, Judith; STEPHENSON, Chris. (2010). Computer science teacher preparation is critical. In: ACM INROADS. v. 1, p. 61 – 66.

GARRET, Jesse James. (2011). **The Elements of User Experience: User-Centered Design for the Web and Beyond**, (2nd ed.). Berkeley: New Riders.

GET STARTED WITH CODE 2. (2017). Disponível em: <<https://itunes.apple.com/us/book/get-started-with-code-2/id1226776857?mt=11>>. Acesso em: 16 de abril de 2018.

GIL, Antônio Carlos. (2010). **Como elaborar projetos de pesquisa**. São Paulo: Atlas, 5ª edição.

GOETHE Johann Wolfgang Von & EASTLAKE, Charles Lock. (2006). **Theory of colours**. Dover Publications USA.

GOOGLE & CS4HS. (2009). **Google – Computer Science for High School**. Disponível em: <<https://www.cs4hs.com/index.html>>. Acesso em: Julho de 2017.

GOOGLE. (2019). **Material Design**. Disponível em: <<https://material.io/guidelines/material-design/introduction.html>>. Acesso em: 20 setembro de 2018.

GONÇALVES, S. (1993). **Teorias da aprendizagem**. Coimbra: Mcgraw Hill.

GONÇALVES, Rafael Queiroz. (2017). **Ensino de gerenciamento de projetos de software mediado por ferramentas**. 2017. 283 f. Tese (Doutorado) - Programa de Pós-graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis,

GREER, D; RUHE, G. (2004) Software release planning: an evolutionary and iterative approach. **Information And Software Technology**, v. 46, n. 4, p.243-253. Elsevier BV. <http://dx.doi.org/10.1016/j.infsof.2003.07.002>.

GRESSE VON WANGENHEIM, Christiane et al. (2017). **dTECT: Um Modelo para a Avaliação de Unidades Instrucionais para o Ensino de Computação na**

Educação BásicaEducação Básica. Florianópolis: Universidade Federal de Santa Catarina. (INCoD /GQS.02.2017.P, INCOD, INE).

GRESSE VON WANGENHEIM, Christiane ., PETRI, G. & BORGATTO, A. F. (2018). **MEEGA+KIDS: A Model for the Evaluation of Educational Games for Computing Education in Secondary School.** Florianópolis: Universidade Federal de Santa Catarina. (INCoD/GQS.06.2018.E, INCoD/INE/UFSC).

GRESSE VON WANGENHEIM, Christiane . *et al.* (2019). **SplashCode– A Board Game for Learning an Understanding of Algorithms in Middle School.** Informatics in Education, accepted for publication.

GUPTA, Niloy; N., Tejovanth; MURTHY, Preeti. (2012). Learning by creating: Interactive programming for Indian high schools. In: INTERNATIONAL CONFERENCE ON TECHNOLOGY ENHANCED EDUCATION, Kerala, India. **Proceedings...** . Kerala: IEEE, 2012. p. 1 - 3.

HADDAWAY, Neal Robert *et al.* (2015). The Role of Google Scholar in Evidence Reviews and Its Applicability to Grey Literature Searching. **Plos One**, v. 10, n. 9, p.1-17, 17 set. Public Library of Science (PLoS). <http://dx.doi.org/10.1371/journal.pone.0138237>.

FLORA, Harleen K.; WANG, Xiaofeng; CHANDE, Swati V. (2014). An Investigation on the Characteristics of Mobile Applications: A Survey Study. **International Journal Of Information Technology And Computer Science**, v. 6, n. 11, p.21-27. MECS Publisher. <http://dx.doi.org/10.5815/ijitcs.2014.11.03>.

HIGHSMITH, J.; COCKBURN, A. (2001). Agile software development: the business of innovation. **Computer**, v. 34, n. 9, p.120-127. Institute of Electrical and Electronics Engineers (IEEE). <http://dx.doi.org/10.1109/2.947100>.

HERMANS, Felienne; AIVALOGLOU, Efthimia. (2017). Teaching software engineering principles to K-12 students: a MOOC on Scratch. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING: SOFTWARE ENGINEERING EDUCATION AND TRAINING TRACK, 39 ed, Buenos Aires, Argentina. **Proceedings...** . Buenos Aires: IEEE/ACM, 2017. p. 13 - 22.

HEWETT, Thomas T.. *et al.* (1992). **ACM SIGCHI Curricula for human-computer interaction.** ACM: Nova Yorque, Estados Unidos.

HILL, Heather C.; ROWAN, Brian; BALL, Deborah Loewenberg. (2007). Effects of Teachers' Mathematical Knowledge for Teaching on Student Achievement. **American Educational Research Journal**, v. 42, n. 2, p.371-406.

IDC Brasil (2018). **Após dois anos, mercado de smartphones cresce em 2017 e atinge o segundo melhor desempenho de vendas.** Disponível em: <<http://br.idclatin.com/releases/news.aspx?id=2312>>. Acesso em: 17 de abril de 2019.

IEEE CS. (2013). **SWEBOK - Guide to the Software Engineering body of knowledge** (3th ed.). Silver Spring/USA: IEEE.

IEEE. (2010). **ISO/IEC/IEEE 24765:2010 Systems and Software Engineering — Vocabulary**.

INEP. (2016). **Censo da Educação Superior 2016**. Disponível em: <https://abmes.org.br/arquivos/documentos/censo_superior_tabelas.pdf>. Acesso em: junho de 2018.

ISMAIL, Nur Amie *et al.* (2016). A Review on Usability Issues in Mobile Applications. **IOSR Journal Of Mobile Computing & Application** v. 3, n. 3, p.47-52.

JEONG, Yang-jae; LEE, Ji-hyeon; SHIN, Gyu-sang. (2008). Development process of mobile application SW based on agile methodology. In: INTERNATIONAL CONFERENCE ON ADVANCED COMMUNICATION TECHNOLOGY., 10 ed, Gangwon-do, South Korea. **Proceedings...** . Gangwon-do: IEEE, 2008. p. 362 - 366.

KALEEL, Shakira Banu; HARISHANKAR, Ssowjanya. (2013). **Applying agile methodology in mobile software engineering: Android application development and its challenges**. Computer Science Technical Reports. 1-13 p.

KE, Fengfeng; IM, Tami. (2013). A case study on collective cognition and operation in team-based computer game design by middle-school children. **International Journal Of Technology And Design Education**, v. 24, n. 2, p.187-201, 24 ago. Springer Science and Business Media LLC. <http://dx.doi.org/10.1007/s10798-013-9248-6>

KEMP, S. (2015). **Digital, Social & Mobile Worldwide**. Disponível em *We are Social*: <<https://wearesocial.com/uk/special-reports/digital-social-mobile-worldwide-2015>>. Acesso em: Julho de 2017.

KENT, Beck. (2001). **Manifesto for Agile Software Development**. Disponível em: <<http://agilemanifesto.org>>. Acesso em: Outubro de 2017.

KENT, Beck. (2000). **Extreme programming explained: embrace change**. Addison-Wesley. Segunda edição, Boston, EUA.

KIM, Gerard Jounghyun. (2015). **Human-Computer Interaction: Fundamentals and Practice**. Boca Raton, Fl: Crc Press Taylor & Francis Group.

KÖHLER, Barbara; GLUCHOW, Michaela; BRÜGGE, Bernd. (2012). Teaching Basic Software Engineering to Senior High School Students. In: INFORMATION SYSTEMS AND TECHNOLOGY FOR ORGANIZATIONS IN A NETWORKED SOCIETY, 2012, Munique, Alemanha. IGI Global, p. 149 - 165.

KUBO, Olga Mitsue; BOTOMÉ, Sílvio Paulo. (2001). Ensino-aprendizagem: uma interação entre dois processos comportamentais. **Interação em Psicologia**, v. 5, n. 1, p.1-19. Universidade Federal do Parana. <http://dx.doi.org/10.5380/psi.v5i1.3321>.

KUUSINEN, Kati; MIKKONEN, Tommi. (2014). On designing UX for mobile enterprise apps. In: EUROMICRO CONFERENCE ON SOFTWARE ENGINEERING AND ADVANCED APPLICATIONS, 40., Verona, Itália: IEEE, 2014. p. 221 - 228.

LEE, Irene *et al.* (2011). Computational thinking for youth in practice. **Acm Inroads**, v. 2, n. 1, p.32-37. Association for Computing Machinery (ACM). <http://dx.doi.org/10.1145/1929887.1929902>.

MAZZIONI, Sady. (2009). As Estratégias Utilizadas no Processo de Ensino-Aprendizagem: Concepções de Alunos e Professores de Ciências Contábeis. In: CONGRESSO USP CONTROLADORIA E CONTABILIDADE, 2009, São Paulo, Brasil. **Anais... . USP**.

MAYHEW, Deborah J. (1999). **The Usability Engineering Lifecycle**. San Diego: Academic Press.

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: janeiro de 2017.

MEC (2018). **Base Nacional Comum Curricular**. Disponível em: <http://basenacionalcomum.mec.gov.br/images/BNCC_20dez_site.pdf>. Acesso em: abril de 2019.

MECCAWEY, Maram. (2017). Raising a programmer: Teaching Saudi children how to code. **International Journal Of Educational Technology**, v. 4, n. 2, p.56-65.

MEERBAUM-SALANT, Orni; ARMONI, Michal; BEN-ARI, Mordechai (moti). (2013). Learning computer science concepts with Scratch. **Computer Science Education**, v. 23, n. 3, p.239-264. Informa UK Limited. <http://dx.doi.org/10.1080/08993408.2013.832022>.

MERRILL, M. David *et al.* (1996). Reclaiming Instructional Design. **Educational Technology**, v. 36, n. 5, p.5-7,

MISSIROLI, Marcello; RUSSO, Daniel; CIANCARINI, Paolo. (2017). Agile for Millennials: A Comparative Study. In: INTERNATIONAL WORKSHOP ON SOFTWARE ENGINEERING CURRICULA FOR MILLENNIALS, 1 ed, 2017, Buenos Aires, Argentina. **Proceedings... . Buenos Aires: IEEE**. p. 47 - 53.

MISSIROLI, Marcello; RUSSO, Daniel; CIANCARINI, Paolo. (2016). Learning Agile Software development in high school: an investigation. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING COMPANION, 38 ed, 2016, Austin, TX. **Proceedings... . Austin: ACM**, p. 293 - 302.

MIT & GOOGLE. (2019). **App Inventor**. Disponível em: <<http://appinventor.mit.edu/explore/about-us.html>>. Acesso em: Abril de 2017.

MIT & GOOGLE. (2019a). **Tutorials for AppInventor**. Disponível em: <<http://appinventor.mit.edu/explore/ai2/tutorials.html>>. Acesso em: Maio de 2017

MIT & GOOGLE. (2019b). **App Inventor Beginner Tutorials**. Disponível em: <<http://appinventor.mit.edu/explore/sites/all/files/hourofcode/AppInventorTutorials.pdf>> Acesso em: Agosto de 2017.

MIT & GOOGLE. (2019c). **Storage – App Inventor for Android**. Disponível em: <<http://ai2.appinventor.mit.edu/reference/components/storage.html>> Acesso em: Outubro de 2017.

MIT & GOOGLE. (2019d). **MIT App Inventor Public Open Source**. Disponível em: <<http://appinventor.mit.edu/appinventor-sources/>> Acesso em: Novembro de 2017.

MIT & GOOGLE. (2019e). **Connect your Phone or Tablet over WiFi**. Disponível em: <<http://appinventor.mit.edu/appinventor-sources/>> Acesso em: novembro de 2017.

MIT & GOOGLE. (2019f). **App-inventor: MIT App Inventor Public Open Source**. Disponível em: <<http://appinventor.mit.edu/appinventor-sources/>>. Acesso em: junho de 2019.

MUCCINI, Henry; FRANCESCO, Antonio di; ESPOSITO, Patrizio. (2012). Software testing of mobile applications: Challenges and future research directions. In: INTERNATIONAL WORKSHOP ON AUTOMATION OF SOFTWARE TEST, 7 ed, Zurique, Suíça. **Proceedings...** . Zurique: IEEE, 2012. p. 29 - 35.

NAGAPPAN, Meiyappan; SHIHAB, Emad. (2016). Future trends in software engineering research for mobile apps. In: INTERNATIONAL CONFERENCE ON SOFTWARE ANALYSIS, 23 ed, 2016, Suita, Japan. **Proceedings...** . Suita: IEEE. p. 21 - 32.

NIELSEN, J. (1992). The usability engineering life cycle. **Computer**, v. 25, n. 3, p.12-22, mar. Institute of Electrical and Electronics Engineers (IEEE). <http://dx.doi.org/10.1109/2.121503>.

PAULA, Danielly Fo de; MENEZES, Bianca Hxm; ARAÚJO, Cristiano C. (2014). Building a quality mobile application: A user-centered study focusing on design thinking, user experience and usability. In: INTERNATIONAL CONFERENCE OF DESIGN, USER EXPERIENCE, AND USABILITY., 2014. **Proceedings...** . Springer. v. 8158, p. 313 - 322.

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: agosto de 2016.

PETERSEN, Kai. (2008). Systematic mapping studies on software engineering. In: INTERNATIONAL CONFERENCE ON EVALUATION AND ASSESSMENT IN SOFTWARE ENGINEERING, 12 ed, 2008, Itália. **Proceedings...** . Itália. p. 26 - 27.

PETRI, Giani; VON WANGENHEIM, Christiane Gresse; BORGATTO, ADRIANO Ferreti. (2017). MEEGA+, Systematic Model to Evaluate Educational Games. **Encyclopedia Of Computer Graphics And Games**, p.1-7. Springer International Publishing. http://dx.doi.org/10.1007/978-3-319-08234-9_214-1.

PINHEIRO, Fernando da Cruz; VON WANGENHEIM, Christiane Gresse; MISSFELDT FILHO, Raul. (2018). Teaching Software Engineering in K-12 Education: A Systematic Mapping Study. **Informatics In Education**, v. 17, n. 2, p.167-206. Vilnius University Press. <http://dx.doi.org/10.15388/infedu.2018.10>.

PISA. (2018). **PISA 2015 Results**. Disponível em: <<http://www.oecd.org/education/pisa-2015-results-volume-iii-9789264273856-en.htm>>. Acesso em: maio de 2018.

POKRESS, Shaileen Crawford; VEIGA, Jose Juan Dominguez. (2013). MIT App Inventor: Enabling personal mobile computing. In: PROGRAMMING FOR MOBILE AND TOUCH, 1., 2013, Indianópolis, IN. **Proceedings...** . Indianópolis: ACM, p. 1 - 3.

PPGCC/UFSC. (2015). **PPGCC: Linhas de pesquisa**. Disponível em Pós-Graduação em Ciência da Computação: <<http://ppgcc.posgrad.ufsc.br/linhas-de-pesquisa-2/>>. Acesso em: outubro de 2017.

PROINFODATA (2019). **Coleta de dados do projeto PROINFO/MEC de inclusão digital nas escolas públicas brasileiras**. Disponível em: <<http://proinfodata.c3sl.ufpr.br/attendance/availability/proinfo/>>. Acesso em: 17 de abril de 2019.

PRUITT, John; ADLIN, Tamara. (2006). **The Persona Lifecycle: Keeping People in Mind Throughout Product Design**. San Francisco, CA, USA: Morgan Kaufmann. 744 p. (Interactive Technologies).

QUINN, Clark. N. (2005). **Engaging Learning: Designing e-Learning Simulation Games**. EUA: Pfeiffer.

RAHIMIAN, Vahid; RAMSIN, Raman. (2008). Designing an agile methodology for mobile software development: A hybrid method engineering approach. In: SECOND INTERNATIONAL CONFERENCE ON RESEARCH CHALLENGES IN INFORMATION SCIENCE, Marrakech, Marocco. **Proceedings...** IEEE, p.337-342 <http://dx.doi.org/10.1109/rcis.2008.4632123>.

RAJ, CP Rahul; TOLETY, Seshu Babu. (2012). A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach. In: BALKAN CONFERENCE IN INFORMATICS, 6 ed, 2012, New York, NY. **Proceedings...** . New York: p. 625 - 629.

ROBINSON, Ashley; PÉREZ-QUIÑONES, Manuel A. (2014). Underrepresented middle school girls: on the path to computer science through paper prototyping. In: TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION, 45 ed, 2014, Atlanta, GA. **Proceedings...** . Atlanta: ACM. p. 97 - 102.

ROBINSON, Ashley; PÉREZ-QUIÑONES, Manuel A.; SCALES, Glenda. (2015). Understanding the attitudes of African American middle school girls toward computer science. In: RESEARCH IN EQUITY AND SUSTAINED PARTICIPATION IN ENGINEERING, COMPUTING, AND TECHNOLOGY, Charlotte, Nc. **Proceedings...** . Charlotte: IEEE. p. 1 - 8.

RODRIGUEZ, Germania M.; CARRIÓN, Samanta Patricia Cueva; CARO, Edmundo Tovar. (2011). Reusable and interoperative specifications for OERs based on standards. In: IEEE GLOBAL CONF. ON ENGINEERING EDUCATION, Amman, Jordan. **Proceedings...** . Amman: IEEE. p. 763 - 770.

ROSEN, Christoffer; SHIHAB, Emad. (2016). What are mobile developers asking about? A large scale study using stack overflow. **Empirical Software Engineering**, v. 21, n. 3, p.1192-1223. Springer Science and Business Media LLC. <http://dx.doi.org/10.1007/s10664-015-9379-3>.

RUSU, Adrian *et al.* (2011). Employing Software maintenance techniques via a tower-defense serious computer game. In: INT. CONFERENCE ON TECHNOLOGIES FOR E-LEARNING AND DIGITAL ENTERTAINMENT, 2011, Berlim, Alemanha. **Proceedings...** . Berlim: Springer. p. 176 - 184.

RUSU, Adrian *et al.* (2011). Learning Software engineering basic concepts using a five-phase game. In: FRONTIERS IN EDUCATION CONFERENCE., Washington, Dc. **Proceedings...** . Washington: IEEE. p. 1 - 6.

SAMSUNG. (2015). **Guide to teaching mobile App Development**. Disponível em: <http://www.scholastic.com/samsungacademy/downloads/SS4_C&B_TeacherGuide.pdf>. Acesso em: Julho de 2017.

ARKAR, Amitrajit; BELL, Tim. (2013). Teaching black-box testing to high school students. In: WORKSHOP IN PRIMARY AND SECONDARY COMPUTING EDUCATION, 8 ed, Aarhus, Dinamarca. **Proceedings...** . Aarhus: ACM, 2013. p. 75 - 78.

SAUDERS, Mark N. K., LEWIS, Phillip., & THORNHILL, Adrian. (2009). **Methods for business students**. (5ª edição), Prentice Hall.

SERRANO, Maurício; SERRANO, Milene. (2013). Requisitos ao Código: Uma Proposta para o Ensino da Engenharia de Software no Ensino Médio. In: INT. REQUIREMENTS ENGINEERING CONFERENCE, Rio de Janeiro, RJ. **Proceedings...** . Rio de Janeiro, p. 37 - 42.

SBC. (2005). **Currículo de Referência da SBC para Cursos de Graduação em Bacharelado em Ciência da Computação e Engenharia de Computação**. Sociedade Brasileira de Computação.

SBC. (2017). **Plano de Gestão para a SBC Biênio Agosto 2015 – Julho 2017**. Disponível em: <<http://www.sbc.org.br/documentos-da-sbc/send/135-eleicoes/999-plano-de-gestao-para-a-sbc-bienio-agosto-2015-julho-2017> > Acesso em: Maio de 2018

SBC (2017a). **Currículo de Referência**. Disponível em: <<http://www.sbc.org.br/documentos-da-sbc/summary/131-curriculos-de-referencia/760-curriculo-de-referencia-cc-ec-versao2005>>. Acesso em: Outubro de 2018.

SCHANK, R.C. (1992). Goal-Based Scenarios. In: David Leake (ed) Case-Based Reasoning. AAAI Press/The MIT Press.

SCHANK, Roger C. *et al.* (1994). The Design of Goal-Based Scenarios. **Journal Of The Learning Sciences**, v. 3, n. 4, p.305-345. Informa UK Limited. http://dx.doi.org/10.1207/s15327809jls0304_2.

SCHANK, R C. (1996). Goal-based scenarios: CASE-based reasoning meets learning by doing. **D. Leake (ed.), CASE-based reasoning: Experiences, lessons & future directions**. p. 295-347.

SCRATCH & MIT. (2013). **Scratch – Starter Projects**. Disponível em: <https://scratch.mit.edu/starter_projects/ > Acesso em: Julho de 2017.

SCRATCHED. (2009). **SCRATCHED**. Disponível em: <<http://scratched.gse.harvard.edu/stories/> > Acesso em: Julho de 2017.

SHEEHAN, Robert. (2010). Lower floor, lower ceiling: easily programming turtle-graphics. In: INT. SYMPOSIUM ON VISUAL LANGUAGES, **Proceedings...** IEEE, 2010. p. 87 - 88.

SPATARU, Andrei Cristian. (2010). **Agile development methods for mobile applications**. 68 f. Dissertação (Mestrado) - Curso de Computer Scien, University Of Edinburgh, Edinburgh, 2010.

SILVA, Daniel Melo da. (2017). **Suporte a unidade instrucional de desenvolvimento de aplicativos com técnicas de UX para o ensino básico**. 2017. 133 f. TCC (Graduação) - Curso de Sistemas de Informação, Universidade Federal de Santa Catarina, Florianópolis.

SILVA, Tiago Silva da, *et al.* (2011). User-centered design and agile methods: a systematic review. In: AGILE CONFERENCE, 2011, Salt Lake City, USA. **Proceedings...** . Salt Lake City: IEEE. p. 77 - 86.

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

SOLECKI, I. S., JUSTEN, K. A., PORTO, J. V. A., GRESSE VON WANGENHEIM, C., HAUCK, J. C. R. & BORGATTO, A. F. (2019). **Estado da Arte do Design Visual**

de Aplicativos Móveis desenvolvidos com App Inventor. Submetido para publicação.

STAPIĆ, Zlatko; MIJAČ, Marko; STRAHONJA, Vjeran. (2016). Methodologies for development of mobile applications. In: INTERNATIONAL CONVENTION ON INFORMATION AND COMMUNICATION TECHNOLOGY, ELECTRONICS AND MICROELECTRONICS, 39 ed, 2016, Opatija, Croatia. **Proceedings...** . Opatija: IEEE. p. 688 - 692.

STARRETT, Cortland. (2007). Teaching UML Modeling Before Programming at the High School Level. In: INTERNATIONAL CONFERENCE ON ADVANCED LEARNING TECHNOLOGIES, 7 ed, Niigata, Japão. **Proceedings...** . Niigata: IEEE, 2007. p. 713 - 714.

SPRINTHALL, Norman. A. & SPRINTHALL, Richard. (1993). **Psicologia Educacional: uma abordagem desenvolvimentista.** Lisboa: Mc Graw-Hill.

SULLIVAN, Jacquelyn F.; REAMON, Derek; LOUIE, Beverly. (2003). Girls embrace technology: A summer internship for high school girls. In: ANNUAL FRONTIERS IN EDUCATION, 33 ed, Westminster, CO. IEEE, 2003. v. 1.

TECHNOVATION. (2018). *Technovation Challenge*. Disponível em: <<http://www.technovationbrasil.org/curriculo>>. Acesso em: 16 de abril de 2018.

TEKKIE UNI (2018). *Build Your First App*. Disponível em: <<https://tekkieuni.com/courses/buildyour-first-app/>>. Acesso em: 16 de abril de 2018.

TISSENBAUM, Mike; SHELDON, Josh; ABELSON, Hal. (2019). From computational thinking to computational action. **Communications Of The ACM**, v. 62, n. 3, p.34-36. Association for Computing Machinery (ACM). <http://dx.doi.org/10.1145/3265747>.

UNGER, Russ; CHANDLER, Carolyn. (2010). **Guia Para Projetar UX.** Rio de Janeiro: Alta Books.

UXQB (2018). **CPUX-F Curriculum and Glossary**. Disponível em: <https://uxqb.org/wp-content/uploads/documents/CPUX-F_EN_Curriculum-and-Glossary.pdf>. Acesso em: Setembro de 2018.

VALENTIM, Natasha M. Costa; SILVA, Williamson; CONTE, Tayana. (2017). The Students' Perspectives on Applying Design Thinking for the Design of Mobile Applications. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING: SOFTWARE ENGINEERING AND EDUCATION TRACK, 39 ed, Buenos Aires, Argentina. **Proceedings...** . Buenos Aires: IEEE Press, 2017. v. 62, p. 77 - 86.

VAN WART, Sarah J.; VAKIL, Sepehr; PARIKH, Tapan S. (2014). Apps for social justice: Motivating computer science learning with design and real-world problem solving. In: CONFERENCE ON INNOVATION E TECHNOLOGY IN COMPUTER SCIENCE EDUCATION, 2014, Uppsala, Sweden. **Proceedings...** . Uppsala: ACM. p. 123 - 128.

VARSHNEY, Upkar; VETTER, Ron. (2001). A Framework for the Emerging Mobile Commerce Applications. In: ANNUAL HAWAII INTERNATIONAL CONFERENCE IN SYSTEM SCIENCES, 34 ed, Maui, Hawaii. **Proceedings...** . Maui: IEEE, 2001. p. 9014 - 9023.

VERHOEFF, Tom. (2006). A master class Software engineering for secondary education. In: INTERNATIONAL CONFERENCE ON INFORMATICS IN SECONDARY SCHOOLS-EVOLUTION AND PERSPECTIVES., 2006, Heidelberg, Alemanha. **Proceedings...** . Heidelberg: Springer, p. 150 - 158.

VTT ELECTRONICS. (2006). **Portal of Agile Software Development Methodologies. Retrieved from Mobile-D Method.** Disponível em: <<http://virtual.vtt.fi/virtual/agile/mobiled.html>>. Acesso em: Julho de 2019.

VYGOTSKY, Lev. S. (1978). **Mind in society:** The development of higher mental processes. Cambridge: Harvard University Press, p. 159.

WANGENHEIM, C. G., ALVES, N. C. & WEBER, A. R. (2018). **Resumo do K-12 Computer Science Standards (Versão 2017).** Disponível em: <http://www.incod.ufsc.br/wp-content/uploads/2017/04/Resumo_CSTA2017_INCOD_GQS04_2017_vf.pdf>. Acesso em: abril de 2018.

WEGMAN, Edward; SAID, Yasmin. (2011). Color theory and design. **Wiley Interdisciplinary Reviews: Computational Statistics**, v. 3, n. 2, p.104-117. Wiley. <http://dx.doi.org/10.1002/wics.146>.

WHATSAPP (2019). **Whatsapp.** Disponível em: <<https://www.whatsapp.com/>> Acesso em: Abril de 2019.

WING, J. M. (2006). Computational thinking. **Communications of the ACM**, v. 49, n. 3, p. 33–35.

WOHLIN, Claes, RUNESON, Per. & HÖST, Martin. (2012). **Experimentation in software engineering: An introduction** (2012 ed.). Springer.

WOLBER, David; ABELSON, Harold; FRIEDMAN, Mark. (2014). Democratizing Computing with App Inventor. **Getmobile: Mobile Computing And Communications.** New York, NY, USA, p. 53-58.

WOLBER, David. (2011). App Inventor and real-world motivation. In: ACM TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION, 42 ed, 2011, Dallas, Tx. **Proceedings...** . Dallas: ACM. p. 601 - 606.

YIN, R. K. (2009). **Case Study Research: Design and Methods.** SAGE, 4 ed., p. 219.

YIN, Pai-ling; DAVIS, Jason P.; MUZYRYA, Yulia. (2014). Entrepreneurial Innovation: Killer Apps in the Iphone Ecosystem. **American Economic Review**, p.255-259.

YUSOF, Norul Huda; HASSAN, Rosilah. (2011). Flash notes and easy electronic software (EES): New technique to improve digital logic design learning. In: INTERNATIONAL CONFERENCE ON ELECTRICAL ENGINEERING AND INFORMATICS, Bandung, Indonesia. **Proceedings...** . Bandung: IEEE, 2011. p. 1 - 6.

ANEXO 1 - MEDIÇÃO PÓS TREINAMENTO - JOVENS TUTORES – ALUNOS

Caro(a) Jovem Tutor(a),

Por favor, dedique alguns minutos para preencher este questionário. Estas informações são importantes para nos ajudar a entender sua experiência com a Computação na Escola e como podemos melhorar o nosso suporte aos Jovens Tutores. Suas respostas individuais são confidenciais.

Nome completo		Data	__/__/__
		a	__

1. Sobre os encontros para fazer o seu *app*

Eu achei ...				
identificar problema/solução	<input type="checkbox"/> Muito fácil	<input type="checkbox"/> Fácil	<input type="checkbox"/> Difícil	<input type="checkbox"/> Muito difícil
analisar o contexto (usuário, tarefa, equipamento e ambiente) e especificar requisitos	<input type="checkbox"/> Muito fácil	<input type="checkbox"/> Fácil	<input type="checkbox"/> Difícil	<input type="checkbox"/> Muito difícil
projetar protótipos em papel	<input type="checkbox"/> Muito fácil	<input type="checkbox"/> Fácil	<input type="checkbox"/> Difícil	<input type="checkbox"/> Muito difícil
programar e testar	<input type="checkbox"/> Muito fácil	<input type="checkbox"/> Fácil	<input type="checkbox"/> Difícil	<input type="checkbox"/> Muito difícil
fazer o <i>design</i> visual (cores, imagens, ícones, tipografia)	<input type="checkbox"/> Muito fácil	<input type="checkbox"/> Fácil	<input type="checkbox"/> Difícil	<input type="checkbox"/> Muito difícil
realizar o teste funcional e de usabilidade	<input type="checkbox"/> Muito fácil	<input type="checkbox"/> Fácil	<input type="checkbox"/> Difícil	<input type="checkbox"/> Muito difícil

Eu achei ...				
Identificar problema/solução	<input type="checkbox"/> Muito divertido	<input type="checkbox"/> Divertido	<input type="checkbox"/> Chato	<input type="checkbox"/> Muito chato
Analisar o contexto (usuário, tarefa, equipamento e ambiente) e especificar requisitos	<input type="checkbox"/> Muito divertido	<input type="checkbox"/> Divertido	<input type="checkbox"/> Chato	<input type="checkbox"/> Muito chato
Projetar protótipos em papel	<input type="checkbox"/> Muito divertido	<input type="checkbox"/> Divertido	<input type="checkbox"/> Chato	<input type="checkbox"/> Muito chato
Programar e testar	<input type="checkbox"/> Muito	<input type="checkbox"/> Divertido	<input type="checkbox"/> Chato	<input type="checkbox"/> Muito chato

	divertido			
Fazer o <i>design</i> visual (cores, imagens, ícones, tipografia)	<input type="checkbox"/> Muito divertido	<input type="checkbox"/> Divertido	<input type="checkbox"/> Chato	<input type="checkbox"/> Muito chato
Realizar o teste funcional e de usabilidade	<input type="checkbox"/> Muito divertido	<input type="checkbox"/> Divertido	<input type="checkbox"/> Chato	<input type="checkbox"/> Muito chato

Marque todas as opções com qual você concorda:

Considero este conteúdo importante:

- () processo de desenvolvimento de *apps*
- () identificação de problema/solução
- () análise de contexto (usuário, tarefa, equipamento e ambiente) e especificação de requisitos
- () *design* de protótipos de papel
- () programação e teste
- () *Design* visual (imagem, tipografia, cores, ícones)
- () teste funcional e de usabilidade

Eu achei ...				
Encontro 1 sobre algoritmos, programação do <i>app</i> encontre-me	<input type="checkbox"/> Excelente	<input type="checkbox"/> Bom	<input type="checkbox"/> Regular	<input type="checkbox"/> Ruim
Encontro 2 sobre processo de desenvolvimento de <i>apps</i> e identificação de problema/solução)	<input type="checkbox"/> Excelente	<input type="checkbox"/> Bom	<input type="checkbox"/> Regular	<input type="checkbox"/> Ruim
Encontro 3 sobre análise de contexto e especificação de requisitos	<input type="checkbox"/> Excelente	<input type="checkbox"/> Bom	<input type="checkbox"/> Regular	<input type="checkbox"/> Ruim
Encontro 4 sobre <i>design</i> de protótipos de papel	<input type="checkbox"/> Excelente	<input type="checkbox"/> Bom	<input type="checkbox"/> Regular	<input type="checkbox"/> Ruim
Encontros 5, 6, 7 sobre programação e teste	<input type="checkbox"/> Excelente	<input type="checkbox"/> Bom	<input type="checkbox"/> Regular	<input type="checkbox"/> Ruim
Encontro 8 sobre <i>design</i> visual (imagem, tipografia, cores, ícones)	<input type="checkbox"/> Excelente	<input type="checkbox"/> Bom	<input type="checkbox"/> Regular	<input type="checkbox"/> Ruim
Encontro 9 sobre teste funcional e de usabilidade	<input type="checkbox"/> Excelente	<input type="checkbox"/> Bom	<input type="checkbox"/> Regular	<input type="checkbox"/> Ruim

A forma como as aulas foram	<input type="checkbox"/> Excelente	<input type="checkbox"/> Bom	<input type="checkbox"/> Regular	<input type="checkbox"/> Ruim
-----------------------------	------------------------------------	------------------------------	----------------------------------	-------------------------------

dadas foi				
Gostei de programar o meu <i>app</i> junto com o meu colega	<input type="checkbox"/> Sim		<input type="checkbox"/> Não	
Vou mostrar meu <i>app</i> para outras pessoas	<input type="checkbox"/> Sim		<input type="checkbox"/> Não	
O material didático (slides, tarefas, vídeos, etc.) é	<input type="checkbox"/> Excelente	<input type="checkbox"/> Bom	<input type="checkbox"/> Regular	<input type="checkbox"/> Ruim
O tempo das aulas passou	<input type="checkbox"/> Muito rápido	<input type="checkbox"/> Rápido	<input type="checkbox"/> Devagar	<input type="checkbox"/> Muito devagar
Em geral, o curso foi	<input type="checkbox"/> Excelente	<input type="checkbox"/> Bom	<input type="checkbox"/> Regular	<input type="checkbox"/> Ruim
O que mais gostei no curso de fazer um <i>app</i> foi				
O que menos gostei no curso de fazer um <i>app</i> foi				

Como você considera seu nível de conhecimento atual	Nenhum	Iniciante	Proficiente	Expert
Computação em geral	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Programação	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<i>App Inventor</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Processo de desenvolvimento de <i>apps</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Identificação de problema/solução	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Análise de contexto (usuário, tarefa, equipamento e ambiente) e especificação de requisitos	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<i>Design</i> de protótipos de papel	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Programação e teste	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<i>Design</i> visual (cores, imagens, ícones, tipografia)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Teste funcional e de usabilidade	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Marque todas as opções com as quais você concorda:				
Consigno explicar para um amigo/amiga como:				

- identificar um problema/solução
- realizar análise de contexto (usuário, tarefa, equipamento e ambiente) e especificação de requisitos
- fazer protótipos em papel
- programar e testar
- fazer o *design* visual (cores, imagens, ícones, tipografia)
- teste funcional e de usabilidade (caso de teste, teste de usabilidade com o usuário)

Tenho interesse em aprender mais sobre:

- processo de desenvolvimento de *apps*
- identificação de problema/solução
- análise de contexto (usuário, tarefa, equipamento e ambiente) e especificação de requisitos
- design* de protótipos de papel
- programação e teste
- design* visual (cores, imagens, ícones, tipografia)
- teste funcional e de usabilidade (caso de teste, teste de usabilidade com o usuário)

Marque a resposta correta. Cada questão tem uma ÚNICA resposta correta.

1. A descrição “como Papai Noel eu preciso receber as cartas das crianças para que eu possa levar o presente de natal” é?

- (a) um programa de *app*
- (b) uma história de usuário
- (c) uma caracterização das pessoas que vão utilizar o meu *app*
- (d) o *design* visual do *app*

2. Quais elementos compõem o *design* visual de um *app*?

- (a) wifi, som, jogo
- (b) cores, tipografia, ícones e imagens
- (c) sensor de localização, mapa
- (d) som, áudio, vídeo

3. Teste funcional serve para:

- (a) desenvolver um *app* mais rápido
- (b) programar um jogo
- (c) verificar se o *app* está apto a realizar todas as tarefas para as quais foi desenvolvido.
- (d) deixar o *app* bonito

4. Antes de programar o *app*, devemos fazer uma análise do usuário para:

- (a) identificar um problema na comunidade
- (b) entender as características de pessoas que vão utilizar o *app*
- (c) verificar se o *app* está correto
- (d) definir o *design* das telas

5. Quais das seguintes opções é uma boa alternativa de paleta de cores para *apps*?

()	
()	
()	
()	

6. Descreva a história de alguma tarefa do seu *app* seguindo o modelo abaixo.

História: <Título da história/nome da tarefa>	
Como um(a)	<Papel/tipo de usuário>
eu preciso	<tarefa que o usuário irá realizar>
para que eu possa	<objetivo a ser atingido>

	Discordo totalmente	Discordo	Concordo	Concordo totalmente
Eu invento/imagino muitas coisas que ainda não existem	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Minhas ideias são úteis	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu consigo resolver um problema de maneiras diferentes	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu sou uma pessoa curiosa	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

	Discordo totalmente	Discordo	Concordo	Concordo totalmente
Não tenho vergonha de falar sobre as minhas ideias	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu aprendo com os meus erros ou quando minhas ideias dão errado	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu tento melhorar minhas ideias	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu comparo opiniões/ideias diferentes para ver qual é a melhor	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu tomo decisões de acordo com as informações que eu tenho	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu gosto de fazer e responder perguntas para aprender algo novo	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu ouço as ideias dos meus colegas e as considero quando formo minha opinião	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu tento entender um problema antes de tentar resolvê-lo	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu escolho e organizo o material que preciso quando vou fazer algo (tarefas de casa, trabalhos, estudar, etc.)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu me pergunto se estou fazendo bem as minhas tarefas da escola	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu me esforço quando faço as minhas tarefas da escola	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu consigo explicar as minhas opiniões e decisões	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu planejo como vou estudar (quais exercícios vou fazer em que dias/tempo, etc.)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Se estou tendo dificuldade em um assunto da matéria eu dedico mais tempo de estudo para esse assunto	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Acredito que consigo aprender tudo que quiser	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu gosto de aprender coisas novas	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu consigo me manter concentrado(a) por muito	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

	Discordo totalmente	Discordo	Concordo	Concordo totalmente
tempo				
Eu ouço com atenção para entender o que os outros falam	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Outras pessoas entendem o que eu falo	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Quando eu leio um texto, eu entendo sobre o que estou lendo	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Não tenho vergonha de falar em público	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Gosto de conversar e ouvir opiniões diferentes	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu consigo fazer bons argumentos em um debate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu gosto de trabalhar junto com os meus colegas para fazer trabalhos ou resolver problemas	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu consigo arranjar um tempo para ajudar outras pessoas	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu gosto de ser o(a) líder do grupo	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu sempre faço a minha parte quando trabalho em grupo	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu consigo criar uma sequência para as tarefas de um trabalho em grupo	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu gosto de ser um bom exemplo para os outros	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu respeito as diferenças das pessoas de outras regiões, países e religiões.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu me comprometo a fazer as tarefas necessárias para atingir o objetivo de um trabalho em grupo	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Em um trabalho em grupo, geralmente meus colegas concordam com as minhas decisões	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu não desisto facilmente	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu geralmente termino as coisas que começo	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu consigo encontrar as informações necessárias para fazer um trabalho/resolver um problema	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

	Discordo totalmente	Discordo	Concordo	Concordo totalmente
Eu analiso se uma informação é confiável ou não	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu posso mudar de opinião dependendo de quanto eu sei sobre o assunto	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu consigo explicar porque mudei de opinião	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu consigo interpretar gráficos e tabelas	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Considero errado copiar, compartilhar ou alterar coisas (informação, texto, fotos, etc.) de outras pessoas sem a permissão delas	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Quando eu estudo, eu busco mais informações além das anotações do meu caderno ou apostila/livro	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Quando eu estudo eu uso a internet para achar informações úteis	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu uso aplicativos de mensagem instantânea (<i>Whatsapp</i> , <i>Messenger</i> , etc.)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu sei como criar documentos (doc, pdf ou planilha etc.) ou apresentações de slides no computador	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu consigo usar aparelhos eletrônicos (Computador, internet, celular, etc.) para fazer trabalhos	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu entendo a importância de ter cuidado com minhas informações pessoais na internet	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu consigo criar programas de computador (jogos, <i>apps</i> , etc.)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu consigo identificar as partes mais importantes de um computador	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu sei o perigo de usar uma senha simples	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu sei como computadores se comunicam pela internet	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu sei como identificar, testar e corrigir erros de um programa de computador	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu tenho o direito de dar minha opinião	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

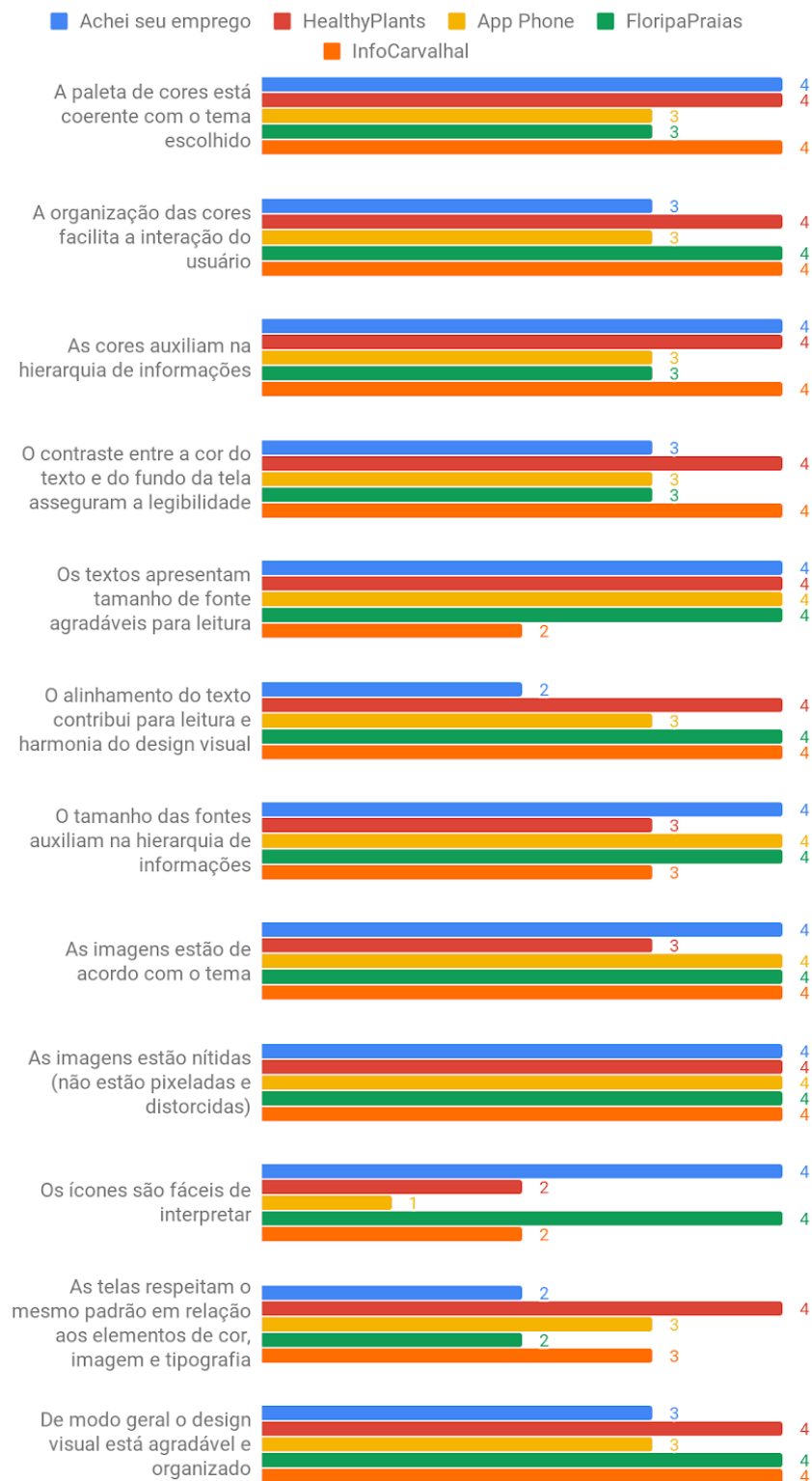
	Discordo totalmente	Discordo	Concordo	Concordo totalmente
Eu presto atenção nas notícias que aparecem nas mídias (TV, redes sociais, sites, etc.)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu respeito que pessoas podem ter diferentes culturas, religiões, estilos de vida e opiniões	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu falo/entendo bem outro idioma (inglês, espanhol, etc.) além do português	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu consigo ter um bom relacionamento com pessoas com personalidades ou interesses bem diferentes dos meus	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu sou amigável e gentil com novos colegas de classe	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu posso aprender muitas coisas com outras pessoas	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu posso ensinar algo a outras pessoas	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu me esforço o máximo possível para cumprir as promessas que eu faço	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu trato as pessoas como gostaria de ser tratado(a)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu admito meus erros e peço desculpas	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu sei que as decisões do governo podem me afetar de diferentes maneiras	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu imagino onde/no que quero trabalhar quando crescer	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu aceito críticas mesmo quando acredito que fiz um bom trabalho	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu sempre faço minhas tarefas da escola	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Se recebo uma nota baixa na escola, tento entender o porquê	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu faço listas de coisas que tenho que fazer	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu consigo fazer minha tarefa de casa sozinho(a)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu evito ao máximo conversar ou mexer no celular	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

	Discordo totalmente	Discordo	Concordo	Concordo totalmente
durante a aula				
Eu consigo me adaptar a mudanças na minha rotina	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu consigo alcançar os objetivos que eu crio para mim mesmo(a)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu entendo o que é necessário para ter uma vida saudável	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu sei como prevenir a dengue	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu sei como me cuidar para não ficar resfriado(a)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu sei as causas do aquecimento global	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu separo o lixo orgânico do reciclável	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eu tento não demorar no banho para economizar água	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Muito obrigado!

ANEXO 2 - RESULTADO DE AVALIAÇÃO DO *DESIGN* VISUAL DOS APPS CRIADO NO JOVENS TUTORES 2018/2.

Pontos nos tópicos da rubrica do design visual



ANEXO 3 - PONTUAÇÃO DOS APPS EM CADA TÓPICO DOS APPS CRIADOS PELOS JOVENS TUTORES 2018/2.

Pontos nos tópicos dos workbooks

