UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

RAUL MISSFELDT FILHO

DESENVOLVIMENTO DE UMA UNIDADE INSTRUCIONAL PARA ENSINAR O DESENVOLVIMENTO DE APPS NO ENSINO FUNDAMENTAL COM O APP INVENTOR

FLORIANÓPOLIS
2019

RAUL MISSFELDT FILHO

DESENVOLVIMENTO DE UMA UNIDADE INSTRUCIONAL PARA ENSINAR O DESENVOLVIMENTO DE APPS NO ENSINO FUNDAMENTAL COM O APP INVENTOR

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

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

FLORIANÓPOLIS
2019

RAUL MISSFELDT FILHO

DESENVOLVIMENTO DE UMA UNIDADE INSTRUCIONAL PARA ENSINAR O DESENVOLVIMENTO DE APPS NO ENSINO FUNDAMENTAL COM O APP INVENTOR

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

Florianópolis, 25 de junho de 2019

Orientadora:

_____

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

Banca Examinadora:

_____

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

_____

Giselle Araújo e Silva de Medeiros
Avaliadora

RESUMO

A computação vem se tornando cada vez mais influente nos dias atuais, auxiliando e melhorando a vida das pessoas. Por esse motivo, a partir do Ensino Fundamental, a computação deve ser popularizada para que estudantes possam aprender ideias-chave de computação e programação resolvendo problemas da maneira mais fácil e eficiente possível em suas áreas de aplicação. Mesmo existindo atualmente várias iniciativas para ensinar computação no Ensino Fundamental, incluindo por exemplo, a programação de *apps* com *App Inventor*, basicamente não existem unidades instrucionais abrangendo também outros conceitos importantes como o design de interface. Neste contexto, o presente projeto visa o desenvolvimento sistemático de uma unidade instrucional para ensinar programação e design de interface de *apps* como competência importante na área de Computação alinhada aos currículos de referência do *ACM/CSTA K-12 Computer Science Framework* e da SBC no Ensino Fundamental. Esta unidade instrucional contempla o ensino de programação de *apps* seguindo um processo de software, *design thinking* e design de interface por meio do desenvolvimento de aplicativos (*apps*) para celulares *Android*, utilizando o ambiente de programação visual *App Inventor*. A unidade instrucional proposta é aplicada em duas versões em uma escola com turmas do Ensino Fundamental. Os resultados apresentam que os conteúdos foram ensinados de uma maneira adequada e bem estruturada, uma vez que os objetivos de aprendizagem são alcançados. Além disto, uma boa parte dos alunos se mostra interessado em aprender mais sobre os assuntos envolvidos e também indicam uma ótima experiência de aprendizado.

Palavras-chave: Computação, Programação, Design de Interface, Unidade Instrucional, Ensino Fundamental, *App Inventor*.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACM - *Association for Computing Machinery*
CASE - *Computer-Assisted Software Engineering*
CEPSH - Comitê de ética em Pesquisa com os Seres Humanos
CSTA - *Computer Science Teachers Association*
DI - *Design* de Interface
DT - *Design Thinking*
EF - Ensino Fundamental
EM - Ensino Médio
ES - Engenharia de *Software*
EU - Engenharia de Usabilidade
IEEE - *Institute of Electrical and Electronics Engineers*
SBC - Sociedade Brasileira de Computação
UI - Unidade Instrucional
UX - *User Experience*
XP - *Extreme Programming*

SUMÁRIO

1. INTRODUÇÃO

1.1. Contextualização

A influência da computação é sentida por todos individualmente, em sociedade e também de forma global (CSTA, 2016). A área de computação vem possibilitando grandes inovações em todos os campos de estudo e também na vida diária das pessoas. O poder dos computadores surge da habilidade de simular o mundo real/físico como um mundo virtual e ter a capacidade de seguir instruções que manipulam e modificam este mundo. Diversos tipos de informações podem ser transformados e processadas pelos computadores para criar *apps*, jogos, carros autônomos, robôs inteligentes e muito mais (CSTA, 2016). Portanto, é muito importante que as pessoas já aprendam desde pequenas os conceitos da computação.

Com o aprendizado da Computação, o estudante desenvolve seu raciocínio lógico, pensamento algorítmico, *design* e uma estruturada forma de resolução de problemas, conceitos e habilidades que podem ser usados muito além das salas de aula (CSTA, 2011). Essas competências proporcionam ao aluno a capacidade de implementar, testar e implantar uma solução, lidando com as restrições do mundo real, e essas habilidades são praticáveis em muitos contextos (CSTA, 2011). Esse cenário enfatiza a necessidade de oferecer uma educação que envolva diferentes conhecimentos de computação, de forma que computadores devem ser vistos muito além de ferramentas, mas, sim, um meio acessível de expressar ideias e expor a criatividade (CAMBRAIA; SCAICO, 2013).

O ensino de conceitos básicos de computação nas escolas é fundamental para desenvolver o raciocínio computacional e lógico das crianças, pelo seu caráter transversal às demais ciências (NUNES, 2011). É necessário o desenvolvimento de habilidades computacionais na educação básica, pois se caracteriza como algo importante intelectualmente. Pode promover múltiplos caminhos profissionais futuros, desenvolver a capacidade de resolução de problemas, apoiar e relacionar-se com outras ciências e motivar os estudantes (CSTA, 2011). O raciocínio lógico deveria ser ensinado desde cedo, pois aumenta a capacidade de dedução e

conclusão de problemas (SICA, 2011). Porém, o ensino dessas competências é reservado muitas vezes somente para o ensino superior que optam pelos cursos relacionados, mas as diversas outras áreas do saber ficam deficientes neste quesito (SICA, 2011). Atualmente o que é ensinado em escolas é o que se chama de "*computer literacy*" (CSTA, 2016), que se trata do ensino do uso de aplicativos tais como editores de texto, imagens, apresentações, etc. O objetivo é que seja ensinado nas escolas a proficiência digital, "*IT fluency*" (CSTA, 2011), que se refere a possibilitar os alunos a terem a capacidade de aprender e aplicar novas tecnologias de forma produtiva.

Já existem diversas iniciativas mundialmente focando no ensino de computação explorando diversas maneiras de ensino, como unidades escolares, *code clubs*, *summer camps*, etc. (MITc, 2019), focando principalmente no ensino de computação via programação de jogos, robôs ou *apps*. Tipicamente, neste estágio escolar, a forma como é ensinada a programação é via linguagens de programação baseadas em blocos, como Scratch[1] por exemplo.

O *App Inventor* (MITa, 2019) é um ambiente de programação visual baseado em blocos e permite qualquer pessoa, até mesmo uma criança, começar a programar e construir aplicativos completos para dispositivos *Android*[2]. Criando *apps* com o *App Inventor*, estudantes aprendem a pensar criativamente, a trabalhar de forma colaborativa e a pensar de forma sistemática na solução de problemas.

Já existem unidades instrucionais voltadas ao ensino de programação de *apps* com *App Inventor*. A maioria envolve unidades instrucionais ensinando a programação de um determinado tipo de *app* por meio de vídeos explicativos (MITb, 2019) e tutoriais *online*, como o "*Magic 8-ball*" que faz previsões do futuro (MITd, 2019) ou oficinas (DANIEL, 2016). A grande parte destes são em Inglês, com poucas exceções, como por exemplo o jogo do mosquito (DANIEL, 2016). Por outro lado, surgiram também iniciativas como a *Technovation*, que estimula meninas a aprenderem a programar por meio de um desafio, que dispõe um currículo e material didático guiando o desenvolvimento de um *app* incluindo conceitos de negócio e de *design* (TECHNOVATION, 2019). Contudo, grande parte das iniciativas ensina somente a programação dos *apps,* muitas vezes sem abordar outras áreas

1 https://scratch.mit.edu/

2 https://www.android.com/

importantes da computação como engenharia de *software*, *design thinking*, e design de interfaces. O *design thinking* fornece um processo dinâmico, participativo, e criativo para empatizar, definir, idealizar, prototipar e testar um sistema de *software* (RAZZOUK; SHUTE, 2012) focando na resolução de problemas que valoriza e explora várias perspectivas de um problema (CHEN; HUANG, 2017).

Dentro do contexto de *apps*, algo importante também é a sua usabilidade, sendo um dos principais critérios de sucesso. Assim, também é importante ensinar a competência relacionada à engenharia de usabilidade. Usabilidade/*User Experience* abrange todos os elementos que envolvem a experiência do usuário em um produto (NORMAN, 2014). A usabilidade é uma “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” (ABNT, 2002). Desta forma, este projeto se preocupa em seguir um processo sistemático de engenharia de usabilidade abrangendo a análise de contexto, a prototipação de interfaces e a avaliação.

De forma geral para obter produtos de *software* de qualidade espera-se seguir um processo sistemático de desenvolvimento adotando práticas de Engenharia de *Software* (ES). A ES é definida como a aplicação de uma abordagem sistemática, disciplinada, quantificável para o desenvolvimento, operação e manutenção de *software* (BOURQUE; FAIRLEY, 2014). Isto envolve o levantamento de requisitos, a modelagem de *software* (arquitetura, componentes e outras características), a construção do *software* (a programação) e o teste de *software* (validando se os requisitos são atendidos) (BOURQUE; FAIRLEY, 2014).

Assim, o ensino de desenvolvimento de *apps* úteis e com boa qualidade, principalmente usabilidade, deve abordar competências além da programação em si, incluindo competências de *design thinking*, UX e engenharia de *software*.

Desta forma, este projeto desenvolve uma unidade instrucional voltada ao ensino de computação integrando conceitos de ES, *design thinking* e UX para alunos do Ensino Fundamental no Brasil. A unidade é alinhada ao *framework* de currículo da CSTA (CSTA, 2016) e do guia de currículo da SBC (SBC, 2018). A unidade ensina a computação por meio de programação de *apps* com *App Inventor* no contexto de uma aprendizagem baseada em problemas, abrangendo desde a parte

do levantamento do problema, definição de solução, design de interface até o desenvolvimento de um aplicativo funcional. Espera-se que esta unidade instrucional criada possa ser aplicada amplamente em escolas do Brasil no ensino fundamental, popularizando não só competências de programação, mas também os conceitos de design e engenharia de *software* como elementos do aprendizado de computação.

1.2. Objetivos

Objetivo geral

O objetivo geral deste projeto é o desenvolvimento sistemático de uma unidade instrucional flexível de aprox. 30 horas/aula (10 horas/aula em uma versão curta) para ensinar competências de Engenharia de *Software*, de programação, de *design thinking* e design de interface em escolas brasileiras em nível de Ensino Fundamental. A unidade instrucional é alinhada ao currículo de referência do ensino de computação da ACM/CSTA (CSTA, 2016) e SBC (SBC, 2018). A UI é projetada de forma que permita a sua adoção de forma interdisciplinar inserida no conteúdo programático do Ensino Fundamental por meio da programação de um *app* para dispositivos móveis *Android* utilizando o *App Inventor*.

Este projeto visa todo o design instrucional dos elementos relativos à unidade instrucional, incluindo a análise do contexto (caracterizando o público alvo, ambiente, necessidades, etc.), definição dos objetivos de aprendizagem, definição da estratégia instrucional e a preparação de todos os recursos didáticos a serem utilizados durante a sua execução. Como resultado, são criados dois planos de ensino, todo o material didático para acompanhar a UI, apresentações para exposição de conceitos e exemplos, exercícios propostos, gabaritos e rubricas para avaliação dos alunos. Também é produzido tanto o material para o aluno quanto material auxiliar para o professor.

Visa-se também a aplicação e avaliação da unidade com o objetivo de analisar a qualidade tanto da percepção de ensino dos alunos quanto outros aspectos da unidade instrucional. Com isso, espera-se gerar uma forma eficaz de ensinar alunos do Ensino Fundamental sobre elementos importantes de computação (*design thinking*, programação, design de interface), podendo ser utilizada para

futuras pesquisas e trabalhos e também que seja amplamente aplicada em escolas brasileiras.

Objetivos específicos

Os objetivos específicos são:

O1. Elaborar a fundamentação teórica;
O2. Analisar o contexto referente aos alunos, ambiente e currículos de referência e definir os objetivos de aprendizagem;
O3. Levantar o estado da arte em relação a unidades instrucionais semelhantes;
O4. Definir o design instrucional da unidade instrucional (seleção e sequenciamento do conteúdo, estratégias de ensino, etc.);
O5. Desenvolver o material didático para a unidade instrucional;
O6. Aplicar e avaliar a unidade instrucional na prática.

1.3. Metodologia de Pesquisa e Trabalho

A fim de alcançar os resultados esperados com este trabalho, é adotada uma combinação de metodologias de pesquisa de acordo com o respectivo objetivo a ser buscado. Então, de acordo com os objetivos específicos do projeto são adotadas etapas da seguinte forma:

Etapa 1 – Fundamentação teórica: análise e síntese de conceitos básicos envolvidos no tema. São abordados conceitos de ensino de computação no Ensino Fundamental, conceitos básicos de aprendizagem e ensino, além de conceitos de engenharia de *software*/processo de *software*, da área de *design thinking* e design de interface/UX e sobre o ambiente de programação visual baseado em blocos *App Inventor*. Este estudo é feito por meio de uma análise e síntese da literatura.

Atividade 1.1: Sintetizar conceitos de aprendizagem e ensino;
Atividade 1.2: Sintetizar conceitos de ensino de computação no Ensino Fundamental;

Atividade 1.3: Sintetizar conceitos de Engenharia de *Software*;

Atividade 1.4: Sintetizar conceitos do cenário atual de *smartphones* no Brasil;

Atividade 1.5: Sintetizar conceitos de Design de Interface*;*

Atividade 1.6: Sintetizar conceitos sobre o *App Inventor*.

Etapa 2 - Levantamento do estado da arte: levantamento sobre trabalhos existentes relacionados à área do projeto. Essa questão é analisada por 2 pontos de vista: Engenharia de *Software* e Design de Interface. São realizados dois mapeamentos sistemáticos da literatura seguindo um processo proposto por Petersen, Vakkalanka e Kuzniarz (2015) para identificar e analisar unidades instrucionais/estratégias de ensino atualmente sendo utilizadas e voltadas ao ensino de computação em escolas.

Atividade 2.1: Mapeamento sistemático da literatura sobre o estado da arte e prática em relação à Engenharia de *Software;*

Atividade 2.1.1: Definir o protocolo de busca;

Atividade 2.1.2: Executar a busca;

Atividade 2.1.3: Extrair e analisar as informações.

Atividade 2.2: Mapeamento sistemático da literatura sobre o estado da arte e prática em relação ao Design de Interface e *design thinking.*

Atividade 2.2.1: Definir o protocolo de busca;

Atividade 2.2.2: Executar a busca;

Atividade 2.2.3: Extrair e analisar as informações.

Etapa 3 - Design da unidade instrucional: é analisado o contexto da unidade instrucional a ser desenvolvida, identificando características e restrições em relação ao público alvo, à infraestrutura e ao contexto educacional. Também engloba toda a parte de planejamento e design da unidade instrucional a ser realizada. A definição do design da unidade, segue a metodologia de design instrucional ADDIE (BRANCH, 2009).

Atividade 3.1: Analisar o contexto em termos de perfil dos aprendizes e instrutores;

Atividade 3.2: Analisar o contexto em termos de ambiente em escolas brasileiras;

Atividade 3.3: Analisar o contexto em termos de necessidades e objetivos de aprendizagem alinhado às diretrizes dos currículos;

Atividade 3.4: Definir e sequenciar o conteúdo da unidade instrucional e definir uma estratégia instrucional, criando o plano de ensino.

Etapa 4 - Desenvolvimento da unidade instrucional: nesta etapa, continuando seguindo o modelo ADDIE de design instrucional, é realizado o desenvolvimento de todo o material didático para a aplicação da unidade instrucional.

Atividade 4.1: Desenvolver o material de apresentação a ser utilizado durante as aulas;

Atividade 4.2: Desenvolver atividades e exercícios a serem aplicadas durante as aulas;

Atividade 4.3: Desenvolver avaliações do desempenho do aluno;

Etapa 5 - Aplicação e avaliação da unidade instrucional: nesta etapa a unidade instrucional desenvolvida é aplicada e avaliada em termos da sua qualidade por meio de estudos de casos (YIN, 2009; WOHLIN et al., 2012).

Atividade 5.1: Definição da avaliação;

Atividade 5.2: Submissão do projeto a CEPSH/UFSC;

Atividade 5.3: Aplicação da unidade instrucional e coleta de dados;

Atividade 5.3.1: Aplicação da versão longa

Atividade 5.3.2: Aplicação da versão curta

Atividade 5.4: Analisar e interpretar os dados.

1.4. Estrutura do Documento

No capítulo 2 é apresentada a fundamentação teórica dos conceitos relacionados com o desenvolvimento da unidade instrucional. O capítulo 3 apresenta o estado da arte em relação às unidades instrucionais existentes que ensinam os conhecimentos de Engenharia de *Software* e Design de Interface por meio da produção ou compreensão de produtos de *software*. O capítulo 4 apresenta o design da unidade instrucional deste trabalho, levantando as informações sobre o público-alvo, características das escolas brasileiras e o plano de ensino é apresentado e detalhado, bem como as rubricas de avaliação e os materiais didáticos. No capítulo

5 é apresentada a aplicação e avaliação da unidade instrucional em duas versões, uma completa e outra mais curta. Finalmente, o capítulo 6 apresenta as conclusões finais sobre o trabalho, apresentando os principais resultados e melhorias da unidade desenvolvida.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é apresentada a fundamentação teórica sobre os assuntos envolvidos neste trabalho. São apresentados conceitos básicos sobre ensino e aprendizagem, conceitos de ensino de computação no Ensino Fundamental, do cenário brasileiro de uso de *smartphones*, de Engenharia de *Software*, design de interface e do *App* Inventor.

2.1. Ensino e Aprendizagem

O conceito de aprendizagem é algo muito discutido por pesquisadores da área de psicologia e pedagogia. A princípio parece simples, mas há profundas diferenças entre as filosofias do conhecimento das abordagens destas duas áreas (SOUZA, 2012). A definição mais relevante a este trabalho trata aprendizagem como uma mudança no comportamento do indivíduo, alterando suas relações com o meio (BUSHELL, 1973), seu comportamento, sua orientação de atuação, atitudes e na sua personalidade, diferentemente da simples acumulação de fatos (ROGERS, 1969). Para a aprendizagem ocorrer de forma sistemática e efetiva é preciso um processo de assimilação onde o aluno orientado pelo professor passa a compreender, refletir e aplicar os conceitos obtidos (FREITAS, 2016), contemplando a concepção de ensino. O ensino é entendido como as ações, meios e condições para a realização da instrução, conhecida como a formação intelectual, formação e desenvolvimento das capacidades cognitivas mediante o domínio de certo nível de conhecimentos (LIBÂNEO, 2006). O ensino representa uma relação entre o que o professor faz e a efetiva aprendizagem do aluno (KUBO e BOTOMÉ, 2001). A utilização de um sistema de ensino facilita a complexidade de um contexto a múltiplas situações, interações dentro do contexto e interação entre contextos (BRANCH, 2009). A ideia geral de um sistema é prover um processo sistemático, com regras e procedimentos, responsivo, interdependente, redundante, dinâmico e criativo (Figura 1).

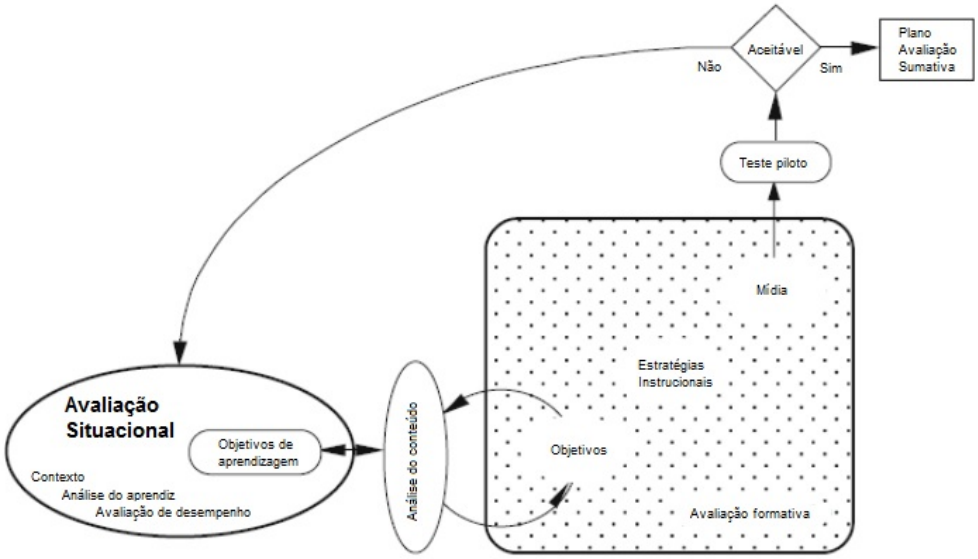


Figura 1: Aplicação de um modelo de sistema de ensino (BRANCH, 2009)

A instrução é o agrupamento intencional de condições de aprendizado para promover a realização de um objetivo específico (DRISCOLL, 2000). Envolve o direcionamento dos alunos para atividades apropriadas de ensino os guiando para um aprendizado apropriado. Também os ajuda a praticar e processar as informações, monitorando o processo dos alunos e provendo *feedback* em termos da adequação das atividades de aprendizado (MERRIL, 2002). Portanto, a proposta de instrução é de ser um método prescritivo, para prover os princípios os quais professores e designers instrucionais possam assegurar o aprendizado (DRISCOLL, 2000). Uma maneira de promover a instrução é por meio de uma unidade instrucional, podendo ser uma aula, uma oficina, uma disciplina, possuindo um tema integrado, provendo as informações necessárias aos estudantes para que possam adquirir competências (*KSA - Knowledge, Skills and Attitudes* [3]) especificadas (DICK; CAREY; CAREY, 2014).

Para o desenvolvimento de uma UI de maneira sistemática tipicamente se adota o design instrucional (BRANCH, 2009), usando uma abordagem de sistemas sobre conhecimento e aprendizagem humana. O design instrucional se caracteriza como um processo iterativo de planejar objetivos de desempenho, selecionar

3 Em Português: Conhecimento, Habilidades e Atitudes

estratégias instrucionais, selecionar ou criar materiais necessários e a aplicação e avaliação de UIs (BRANCH, 2009).

O modelo ADDIE é um modelo de design instrucional (BRANCH, 2009). Este modelo possui cinco fases que descrevem um processo aplicado ao design instrucional a fim de gerar a ocorrência intencional de aprendizado (Figura 2).

Figura 2: Fases do modelo ADDIE adaptado
(BRANCH, 2009)

As fases do modelo ADDIE são (BRANCH, 2009):

A etapa de **Análise** tem o propósito de determinar os objetivos e metas de instrução da unidade, identificar e analisar o público-alvo e avaliar os recursos disponíveis e quais são necessários para completar o processo. Os objetivos e metas são definidos a partir da direção onde todos os esforços são direcionados. Devem ser afirmações que descrevem o que os alunos conseguirão realizar após a participação da unidade. A identificação do público-alvo é uma das fases mais importantes da etapa de análise. Tem por objetivo identificar as habilidades, experiências, preferências e motivações dos alunos, identificando características gerais e do grupo, número de estudantes e seus interesses. Os recursos devem ser identificados em termos de recursos de conteúdo, tecnológicos, instrucionais e humanos, e analisados para que todos os recursos necessários sejam detectados. Em questão da determinação dos objetivos de aprendizado, existem muitos instrumentos atualmente para o apoio didático-pedagógico nesta escolha (FERRAZ; BELHOT, 2010). A Taxonomia de Bloom é um desses instrumentos, cujo principal

propósito é a definição dos objetivos ligados ao desenvolvimento cognitivo, facilitando o planejamento do processo de ensino e aprendizagem (FERRAZ; BELHOT, 2010). Bloom subdivide os objetivos em dois domínios, cognitivo e afetivo, e cada um destes é subdividido em níveis de aprendizagem que possibilitam a sua análise (DRISCOLL, 2000). Como aprimoramento dos trabalhos de Bloom, Simpson desenvolveu outro domínio, o psicomotor, com níveis de aprendizado relacionados a este meio (DRISCOLL, 2000). As Tabelas 1, 2 e 3 apresentam os domínios e seus níveis de aprendizado de acordo com Driscoll (2000) seguindo a Taxonomia de Bloom, tendo a cada nível (do menor ao maior) um aprimoramento sobre o domínio de conhecimento.

Tabela 1: Domínio cognitivo adaptado (DRISCOLL, 2000)

Nível	Percepção
1º - Conhecimento	Lembrança de material previamente aprendido, como fatos, vocabulário, conceitos e princípios
2º - Compreensão	Assimilação do conteúdo do material
3º - Aplicação	Utilização de abstrações, regras, princípios, ideias e outras informações aprendidas em situações reais
4º - Análise	Quebra do material em elementos ou partes tornando-as compreensíveis e identificação de suas relações
5º - Síntese	Combinação de elementos, pedaços ou partes formando algo concreto e completo, construindo um novo modelo ou estrutura
6º - Avaliação	Realização de julgamentos sobre se os métodos e/ou materiais utilizados para a resolução de tal problema satisfizeram o critério desejado

Tabela 2: Domínio afetivo adaptado (DRISCOLL, 2000)

Nível	Percepção
1º - Receptividade	Sensibilização ou disposição a receber certas informações
2º - Resposta	Torna-se envolvido ou realiza algo, reage a um estímulo
3º - Valorização	Apresentação de comprometimento sobre algo, por causa de seu valor característico
4º - Organização	Organização de um conjunto de valores e determinação da relação entre eles, incluindo hierarquização
5º - Caracterização	Integração de valores em uma filosofia e ação consistente a esta filosofia

Tabela 3: Domínio psicomotor adaptado (DRISCOLL, 2000)

Nível	Percepção
1º - Percepção	Se torna consciente da estimulação e necessidade de ação
2º - Prontidão	Preparação para uma ação
3º - Resposta guiada	Responde/reage com assistência de um professor ou instrutor
4º - Mecanismo	Responde/reage de forma habitual
5º - Resposta complexa	Resolução de incertezas e realização de tarefas difíceis automaticamente
6º - Adaptação	Alteração de respostas/ações para se encaixarem em novas situações
7º - Originalidade	Criação de novas atos ou expressões

O propósito da etapa de **Design** é estipular o desempenho desejada dos alunos e métodos apropriados de teste. Nesta fase, os principais conceitos são selecionados de acordo com os objetivos de aprendizagem para que os alunos possam adquirir as competências definidas. Testes, questões objetivas sobre os assuntos, auxiliam os professores e alunos a verificarem seu desempenho relacionado às metas e objetivos de ensino. Nesta fase também é definido como será avaliado o nível de aprendizado dos alunos.

Durante o **Desenvolvimento** são criados e/ou validados materiais que são utilizados durante a unidade instrucional de acordo com as estratégias instrucionais definidas. Esta etapa também pode envolver a seleção e/ou desenvolvimento de ferramentas de suporte a UI, como, por exemplo, analisadores de código. Entende-se como materiais tudo que pode ser utilizado para facilitar/auxiliar no ensino, como guias para os alunos e professores, slides, exemplos de resoluções de problemas. Além disso, é definido como a instrução ocorrerá, que direções são tomadas durante o ensino para que os alunos possam construir os conhecimentos e habilidades. Também é criado um plano de avaliação formativa, para que os recursos possam ser avaliados e, possivelmente, revisados.

A estratégia instrucional é utilizada para determinar a abordagem a qual um professor se baseia para atingir os objetivos de aprendizagem (SASKATCHEWAN, 1991). Um ensino eficiente não é apenas um conjunto de práticas genéricas, mas, sim, um conjunto de decisões orientadas ao contexto de ensino (GLICKMAN, 1991). Os métodos instrucionais são utilizados pelos professores para criar meios de ensino e especificar a natureza das atividades envolvidas no aprendizado. Usualmente os métodos estão associados a uma categoria de estratégia instrucional, mas também podem estar presentes em mais de uma (KEESEE, 2008). Existem várias categorias de estratégia instrucional (Tabela 4), também apresentando os métodos instrucionais associados.

Tabela 4: Estratégias instrucionais (SASKATCHEWAN, 1991)

Estratégia instrucional	Descrição
Instrução direta	É uma estratégia dirigida por professores e muito comumente utilizada. Inclui métodos como leituras, questionamentos, ensino explícito, práticas e demonstrações. Esta estratégia é eficiente para o conhecimento das informações e para o desenvolvimento de habilidades passo-a-passo. A instrução direta é geralmente dedutiva, ou seja, o conteúdo é apresentado e ilustrado através de exemplos para chegar a certa conclusão lógica
Instrução indireta	A instrução indireta é realizada através de investigação, indução, resolução de problemas, tomada de decisões, entre outros. Ao contrário da instrução direta, a indireta é centrada no aluno, mas as duas categorias podem coexistir para um bem maior. Exemplos de métodos dessa categoria de instrução são a discussão reflexiva, formação de conceitos, realização de conceitos, resolução de problemas e investigação guiada. A instrução indireta necessita de um grau maior de envolvimento dos alunos na observação, investigação e formação de hipóteses. Toma vantagem no interesse e curiosidade dos estudantes, geralmente os encorajando a gerar alternativas ou resolvendo problemas. Aqui, o papel do professor é ser um facilitador, um apoiador, provendo oportunidades para os alunos se envolverem e também gera feedback enquanto eles conduzem a investigação
Instrução interativa	A instrução interativa é fortemente baseada na discussão e compartilhamento entre os participantes. Os alunos conseguem aprender com seus colegas enquanto professores os instigam de forma a desenvolverem habilidades sociais, organização de pensamentos e formação de argumentos racionais. Os métodos que podem ser utilizados são discussões em classe, discussões em pequenos grupos ou um conjunto de alunos trabalhando em uma tarefa coletivamente. É importante que o professor informe o tópico a ser discutido, o tempo de discussão, a composição e o tamanho dos grupos e as técnicas de compartilhamento. O sucesso desta abordagem é fortemente dependente da expertise do professor na estruturação e desenvolvimento das dinâmicas em grupo
Aprendizado experimental	O aprendizado experimental é indutivo, centrado no aprendiz e orientado a atividades. O professor pode utilizar esta abordagem tanto dentro quanto fora da sala de aula. Em ambos os lugares os alunos constroem conhecimento sobre o conteúdo informado. O aprendizado experimental pode ser visto como a realização cíclica de cinco fase, são elas experimentação, compartilhamento, análise, conclusão e aplicação. Alguns métodos utilizados são experimentos conduzidos, jogos, simulações, entre outros
Estudo independente	O estudo independente refere-se ao conjunto de métodos instrucionais os quais são propositalmente utilizados para o desenvolvimento da iniciativa individual dos estudantes, autoconfiança e auto aperfeiçoamento. Se baseia no planejamento de estudo independente sem a orientação ou supervisão de um professor. O estudo independente encoraja os alunos a tomarem responsabilidade em planejar e colocar em prática seu próprio aprendizado

A Figura 3 ilustra as estratégias instrucionais e os métodos instrucionais associados.

Figura 3: Estratégias instrucionais e seus métodos (SASKATCHEWAN, 1991)

Outro aspecto importante a ser mencionado é o sequenciamento da unidade instrucional como todo. A ordenação e organização das atividades afetam a maneira que a informação é processada pelos alunos e como a aprendizagem é realizada (FLÔRES; TAROUCO; REATEGUI, 2009). Conforme Bruner (1966), as atividades devem ser organizadas em uma forma espiral de modo que os alunos produzam continuamente sobre o que eles já aprenderam. O sequenciamento de conteúdos em uma unidade instrucional pode ser guiado pelo modelo de estágios de aprendizado de Dreyfus e Dreyfus (1980). Este modelo consiste em analisar e sistematizar a descrição das mudanças na percepção dos participantes na execução e aquisição de habilidades complexas. A abordagem utilizada se baseia na ideia de que a familiaridade em resolver problemas do dia-a-dia é uma característica dominante no comportamento inteligente dos seres humanos. Um detalhamento dos estágios é apresentado na Tabela 5 conforme o autor. Em cada um dos estágios é importante identificar as habilidades que cada participante adquiriu e quais está apto a alcançar.

Tabela 5: Apresentação e descrição dos estágios de aprendizado (DREYFUS; DREYFUS, 1980)

Estágio	Percepção
1º - Principiante	O participante normalmente decompõe a tarefa em características as quais consegue reconhecer sem experiência no assunto. Ao principiante, então, são dadas regras para determinar as ações de acordo com as características identificadas. Para seu aprimoramento, o principiante deve ser monitorado, seja por auto-observação ou *feedback* instrucional. Desta forma, ele consegue tornar seu comportamento cada vez mais completo em conformidade com as regras
2º - Capaz	O participante é considerado capaz depois de ter vivenciado situações reais onde ele próprio ou o instrutor levantou aspectos relevantes e padrões. O reconhecimento destes aspectos, agora, deve ser alcançado pela atenção a exemplos escolhidos propositalmente. Neste estágio, o participante deve ser capaz de armazenar estas informações adquiridas nas vivências dos exemplos de forma a promover uma base para o reconhecimento de futuros aspectos similares
3º - Proficiente	O participante já foi exposto a diversos conjuntos de situações. Estes conjuntos, pela primeira vez, têm um sentido na conquista do objetivo final. Os aspectos se mostram muito ou pouco relevantes, de acordo com este objetivo. Com as situações vivenciadas e os aspectos levantados e suas importâncias, o proficiente utiliza as características memorizadas para determinar uma ação apropriada ao vivenciar novas situações
4º - Experiente	O participante, em uma determinada tarefa, já alcançou o nível final no seu avanço mental em relação ao objetivo. Agora, ele possui um repertório de situações vivenciadas que normalmente ditam apropriadas ações intuitivamente
5º - Dominante	O participante desde o estágio anterior já possui o nível mental atingido. Agora, ele deixa de conscientemente prestar atenção em seu desempenho e é capaz de produzir instantaneamente a perspectiva apropriada e sua ação associada

A Tabela 6 faz uma síntese dos cinco estágios de atividade mental envolvida na aquisição das competências.

Tabela 6: Síntese dos estágios do modelo de Dreyfus e Dreyfus (1980)

	Novato	Competente	Proficiente	Experiente	Dominante
Lembrança	não situacional	situacional	situacional	situacional	situacional
Reconhecimento	decomposto	decomposto	decomposto	completo	completo
Decisão	analítica	analítica	analítica	intuitiva	intuitiva
Consciência	monitorada	monitorada	monitorada	monitorada	absorvida

Em relação ao sequenciamento de eventos dentro de unidades instrucionais, Gagné indica que o sequenciamento das informações está fortemente ligado ao processo de instrução, estes processos são os responsáveis pelas transformações da memória dos estudantes nas etapas do aprendizado (DRISCOLL, 2000). Ele, então, propõe nove eventos de instrução para a realização do processo de instrução (Tabela 7).

Tabela 7: Nove eventos de instrução de Gagné adaptado (DRISCOLL, 2000)

Evento	**Ação**
1º - Ganho da atenção	Utilizar mudanças de estímulo
2º - Informação dos objetivos aos alunos	Informar aos alunos do que eles serão capazes após o aprendizado
3º - Estímulo a lembrança de aprendizagem prévia	Perguntar sobre assuntos já lecionados anteriormente
4º - Apresentação do conteúdo	Apresentar o conteúdo de diferentes formas
5º - Realização de aprendizagem guiada	Sugerir uma organização do conteúdo que faz sentido
6º - Instigação de desempenho	Exigir aos alunos a realização de atividades
7º - Apresentação de *feedback*	Retornar *feedback* aos alunos
8º - Avaliação de desempenho	Requisitar desempenho dos alunos por meio do *feedback*
9º - Aumento da retenção e transferência do conteúdo	Providenciar variedade de práticas e avaliações

A etapa de **Implementação** do modelo ADDIE tem por objetivo a preparação do meio de aprendizado e engajamento dos alunos. Tanto os alunos quanto os professores devem ser treinados de certa forma, utilizando os materiais desenvolvidos na etapa anterior. Os professores devem ser capazes de atuar como facilitadores do aprendizado dos alunos e estes devem assumir uma independência em seu próprio aprendizado. Nesta etapa é aplicada na prática a unidade instrucional com o público-alvo identificado, fazendo com que este público desenvolva as competências desejadas. E, então, analisar se os objetivos de aprendizagem foram alcançados, bem como se a unidade instrucional se mostrou eficaz como um todo.

A etapa de **Avaliação** tem por propósito avaliar a qualidade dos produtos e processos instrucionais, tanto durante quanto depois da Implementação. Os processos executados são de determinar um critério de avaliação, selecionar uma ferramenta adequada e conduzir a avaliação. Concluída esta etapa, a unidade instrucional pode ser analisada em termos de seu sucesso, podendo ser recomendadas melhorias para futuros trabalhos neste escopo. Existem diferentes enfoques na avaliação, um mais voltado a correção e auxílio durante a aprendizagem e outro ao final de um período de aprendizado.

Essa avaliação geralmente é realizada por meio de um estudo empírico (WOHLIN et al., 2012). Diferentes designs de pesquisa podem ser utilizados, dependendo do tipo de estudo variando não experimentais (estudos de caso, por exemplo) até experimentais (Tabela 8).

Tabela 8: Tipos comuns de design de pesquisa (SHADISH; COOK; CAMPBELL, 2001; WANGENHEIM; SHULL, 2009)

Nível de avaliação (KIRKPATRICK; KIRKPATRICK, 2006)	**Tipo de estudo**	**Design**	**Representação X=Tratamento O=Medição R=Tarefa aleatória**
1 - Reação: Avalia como os participantes se sentiram sobre o treinamento ou a experiência de aprendizado	Estudo de caso	Apenas um pós-teste	X O
2 - Aprendizagem: avalia o aumento de conhecimento ou habilidades	Estudo de caso	Apenas um pós-teste/pré-teste	O X O
	Quase-experimental	Grupo de comparação estático	X O O
		Grupo estático pré-teste/pós-teste	O X O O O
		Série de vezes	O O X O O
	Experimental	Randomizado somente pós-teste	R X O R O
		Randomizado pré-teste/pós-teste	R O X O R O O
		Randomizado com grupo de controle pré-teste/pós-teste	R O X1 O R O X2 O

Para alcançar os objetivos de avaliação, a medição deve ser explicitamente definida de uma forma que trace uma ligação entre os objetivos e os dados coletados e também que gerem uma maneira genérica para analisar e interpretar os dados com os objetivos associados (WOHLIN et al., 2012). Podem ser utilizados diversos tipos de instrumentos de coleta de dados como observações, questionários, entrevistas ou os próprios artefatos criados pelos alunos (BRANCH, 2009). De acordo com o objetivo da avaliação e a natureza dos dados coletados, diferentes métodos de análise qualitativa ou quantitativa podem ser utilizados (Figura 4) (WOHLIN et al., 2012; FREEDMAN; PISANI; PURVES, 2007).

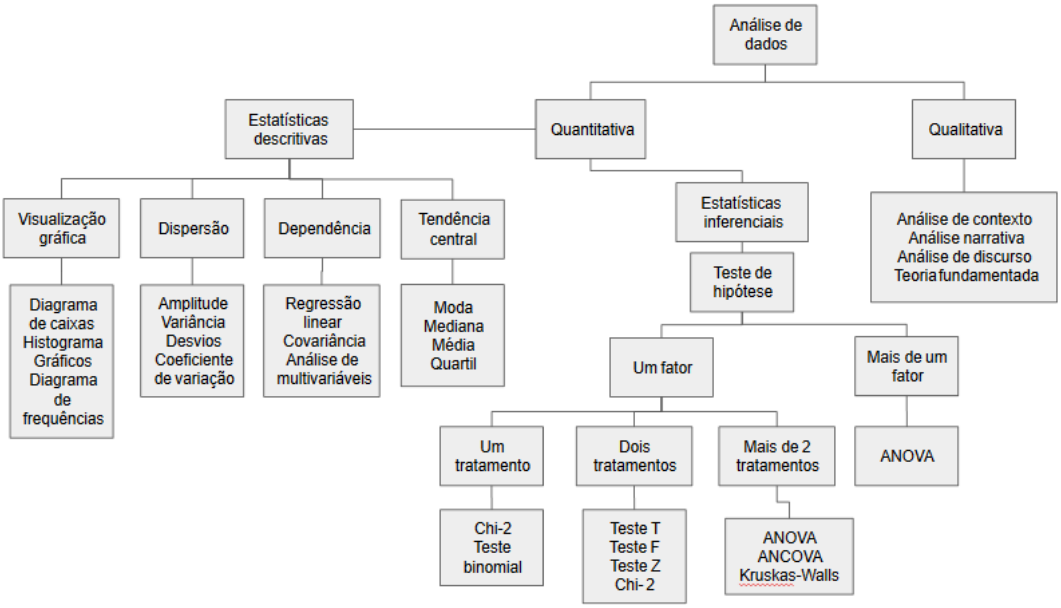


Figura 4: Métodos de análise de dados (WOHLIN et al., 2012; FREEDMAN; PISANI; PURVES, 2007)

Então, os dados analisados são interpretados de maneira a responder as perguntas de análise atingindo o objetivo de avaliação.

A avaliação do aprendizado dos alunos pode ser realizada de duas maneiras, formativa e somativa. A avaliação formativa ocorre durante um processo de aprendizagem, controlando o processo de instrução. Seu objetivo é a correção de falhas do processo educacional e promover *feedback* em relação a recuperação de falhas de aprendizagem (CALDEIRA, 2004; FENILI et al., 2002). Já a avaliação somativa ocorre no final de um processo de aprendizado, com objetivos de medição de resultados bem definidos (CALDEIRA, 2004). Tem a finalidade de classificar os alunos ao fim do processo, com atribuição de notas e certificados, e prestar a comparação dos resultados obtidos com os diferentes alunos, materiais e métodos de ensino envolvidos (FENILI et al., 2002).

2.2. Ensino de Computação no Ensino Fundamental

O Ministério da Educação infere às escolas brasileiras diretrizes curriculares do ensino básico (DNC) (MEC, 2017). O Ensino Fundamental brasileiro duração de nove anos, a etapa mais longa do ensino básico, abrangendo a faixa etária de seis aos quatorze anos de idade. Os alunos do Ensino Fundamental regular são crianças

e adolescentes de faixas etárias cujo desenvolvimento está marcado por interesses próprios, relacionado aos seus aspectos físico, emocional, social e cognitivo, em constante interação. Nos anos iniciais do Ensino Fundamental, a criança desenvolve a capacidade de representação, indispensável para a aprendizagem da leitura, dos conceitos matemáticos básicos e para a compreensão da realidade que a cerca, conhecimentos que se postulam para esse período da escolarização. O currículo do Ensino Fundamental tem uma base nacional comum (BNCC), complementada em cada sistema de ensino e em cada estabelecimento escolar por uma parte diversificada (MEC, 2017).

Os conteúdos curriculares que compõem a parte diversificada do currículo serão definidos pelos sistemas de ensino e pelas escolas, de modo a complementar e enriquecer o currículo, assegurando a contextualização dos conhecimentos escolares diante das diferentes realidades. Os conteúdos que compõem a base nacional comum e a parte diversificada têm origem nas disciplinas científicas, no desenvolvimento das linguagens, no mundo do trabalho e na tecnologia, na produção artística, nas atividades desportivas e corporais, na área da saúde, nos movimentos sociais, e ainda incorporam saberes como os que advêm das formas diversas de exercício da cidadania, da experiência docente, do cotidiano e dos alunos (MEC, 2017).

Os conteúdos sistematizados que fazem parte do currículo são denominados componentes curriculares, os quais, por sua vez, se articulam às áreas de conhecimento (MEC, 2017):

- Linguagens:
    - Língua Portuguesa
    - Língua materna, para populações indígenas
    - Língua Estrangeira moderna
    - Artes
    - Educação Física
- Matemática
- Ciências da Natureza
- Ciências Humanas:
    - História

    - Geografia
- Ensino Religioso

Portanto, a Computação não está atualmente inserida no contexto da base nacional comum, mesmo abrangendo aprendizado sobre tecnologia e ciências. Entretanto, a Sociedade Brasileira de Computação, criou um currículo de referência para o ensino de computação da educação básica (SBC, 2018). Este documento define os conteúdos a serem inseridos ao longo dos anos escolares (SBC, 2018). A Tabela 9 apresenta e descreve estes eixos principais.

Tabela 9: Principais eixos do ensino da Computação (SBC, 2018)

Eixo	**Descrição**	**Pilares**
Pensamento Computacional	Refere-se à capacidade de sistematizar, representar, analisar e resolver problemas	Abstração (modelos e representações que descrevem informações), automação (descrever soluções tais que uma máquina possa executá-la) e análise (analisar criticamente problemas e soluções)
Mundo Digital	Entendimento que é preciso codificar informações e organizá-las de forma que possam ser armazenadas e recuperadas quando necessário. Também é necessário compreender como as informações são transmitidas e a estrutura da Internet	Codificação (descrição e armazenamento de informações), processamento (processamento da informação por computadores e os diferentes níveis de relação entre *hardware* e *software*) e distribuição (entender a comunicação entre dispositivos, transmissão de dados e como é garantida sua integridade e segurança)
Cultura Digital	Efeitos e impactos da computação para a sociedade.	Computação e sociedade (impactos e decorrências da “revolução digital”), fluência tecnológica (uso eficiente e crítico de ferramentas computacionais) e ética digital

Nestes referenciais as competências adquiridas para cada um destes eixos varia com o grau de escolaridade dos estudantes em questão. Para cada etapa da educação básica, os alunos devem ter um nível de competência diferente para cada área principal (SBC, 2018). A Tabela 10 apresenta as competências relacionadas com o ensino de Computação para cada etapa da educação básica.

Tabela 10: Habilidades relacionadas ao ensino de Computação (SBC, 2018)

Escolaridade	Pensamento Computacional	Mundo Digital	Cultura Digital
Ensino Infantil	- Compreender uma situação-problema criando e identificando sequências de passos de uma tarefa para sua solução - Representar os passos de uma tarefa através de uma notação pictórica, de forma organizada e relacional - Criar passos para solução de problemas relacionados ao movimento do corpo e trajetórias espaciais	- Diferenciar objetos eletrônicos dos objetos não eletrônicos, descrevendo seus usos e finalidades	- Interagir com dispositivos computacionais por meio de diferentes interfaces como teclado, mouse, toque de tela e outros
Ensino Fundamental I	- Representar em experiências concretas as principais abstrações para descrever dados: registros, listas e grafos - Identificar as principais abstrações para construir processos: escolha, composição e repetição, simulando e definindo algoritmos simples que representem situações do cotidiano infantil - Utilizar linguagem lúdica visual para representar algoritmos - Compreender a técnica de decompor um problema para solucioná-lo	- Entender o conceito de informação, como armazená-la e codificá-la - Reconhecer a arquitetura básica de computadores digitais	- Identificar a presença da informática na vida das pessoas, bem como sua influência na sociedade atual, compreendendo seu impacto na evolução cultural da humanidade - Identificar critérios para avaliação de informações buscadas na internet que possibilitem entender a lógica de ordenamento de resultados e sua utilização para novas aprendizagens
Ensino Fundamental II	- Utilizar linguagens visuais e língua nativa para representar dados e processos - Formalizar os conceitos de dados estruturados (registros, listas, grafos) - Empregar o conceito de recursão - Construir soluções de problemas usando a técnica de generalização, permitindo o reuso de soluções de problemas em outros contextos, aperfeiçoando e articulando saberes escolares - Relacionar um algoritmo descrito em uma linguagem visual com sua representação em uma linguagem de programação	- Estabelecer relação entre *hardware* e *software* em um nível elementar - Reconhecer a estrutura e o funcionamento da *Internet*	- Identificar o uso de tecnologia nas diferentes dimensões da vida escolar, social, e profissional, analisando criticamente os riscos e impactos de seu uso - Utilizar ferramentas computacionais para agregar, manipular e gerar informações a partir dos dados - Estabelecer critérios para sistematizar a busca e seleção de dados e informações, de modo efetivo, ético e seguro

Ensino Médio	- Elaborar projetos integrados às áreas de conhecimento curriculares, em equipes, solucionando problemas, usando computadores, celulares, e outras máquinas processadoras de instruções - Compreender a técnica de solução de problemas através de transformações: comparar problemas para reusar soluções - Analisar algoritmos quanto ao seu custo (tempo, espaço, energia, …) para poder justificar a adequação das soluções a requisitos e escolhas entre diferentes soluções - Argumentar sobre a correção de algoritmos, permitindo justificar que uma solução de fato resolve o problema proposto - Reconhecer o conceito de meta-programação como uma forma de generalização - Entender os limites da Computação para diferenciar o que pode ou não ser mecanizado, buscando uma compreensão mais ampla dos processos mentais envolvidos na resolução de problemas	- Compreender princípios de segurança em computadores para evitar um uso inadequado, compreender as limitações e propiciar um uso mais seguro da internet em seus processos de pesquisa, uso de redes sociais e demais utilidades de seu cotidiano - Compreender em um nível detalhado a relação entre hardware e *software* (camadas/sistema operacional) para um uso competente e eficaz do computador - Entender como se dá a transmissão de dados entre computadores, a fim de compreender o funcionamento básico do mundo virtual, composto por diversos componentes distribuídos (processos, servidores, nuvem, etc.) que cooperam para realizar tarefas	- Discutir questões relacionadas à propriedade intelectual das informações, discutindo autoria e recursos livres, evitando copiar e colar, através de atividades de produção de material de divulgação de seus princípios - Analisar criticamente o impacto das tecnologias na sociedade, avaliando fatores éticos, sociais e de sustentabilidade, identificando como pode ter um impacto mais positivo na sociedade

Internacionalmente um dos principais guias de currículo desenvolvidos é o *K-12 Computer Science Framework* (CSTA, 2016). Esse framework divide os níveis de ensino em três partes conforme a Figura 5. Ele representa uma visão de ensino para que os alunos aprendam computação por meio dos conceitos e práticas associados. É dedicado a alunos desde os primeiros anos da escola até o último ano do ensino médio, levando estes alunos a desenvolverem uma fundamentação na área de Ciência da Computação e a aprenderem novas abordagens para a solução de problemas.

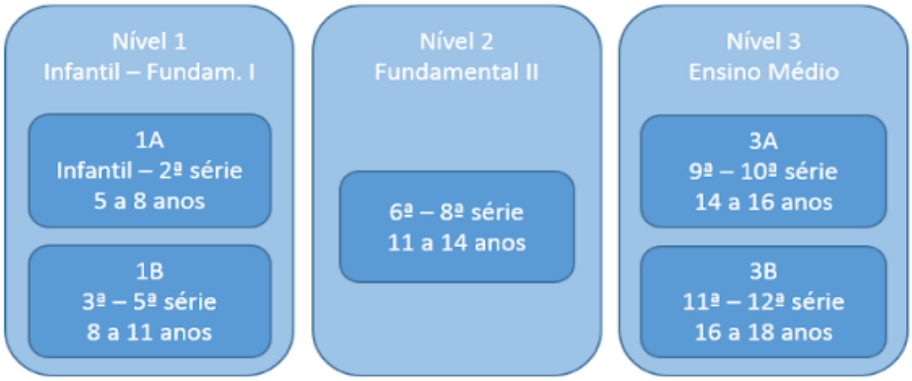


Figura 5: Níveis de ensino K-12 (WANGENHEIM; ALVES; WEBER, 2017)

Conforme o *K-12 Computer Science Framework,* o ensino de computação abrange conceitos e práticas básicos, e são integrados para prover experiências autênticas e significativas para o engajamento dos alunos com o aprendizado de computação (Figura 6).

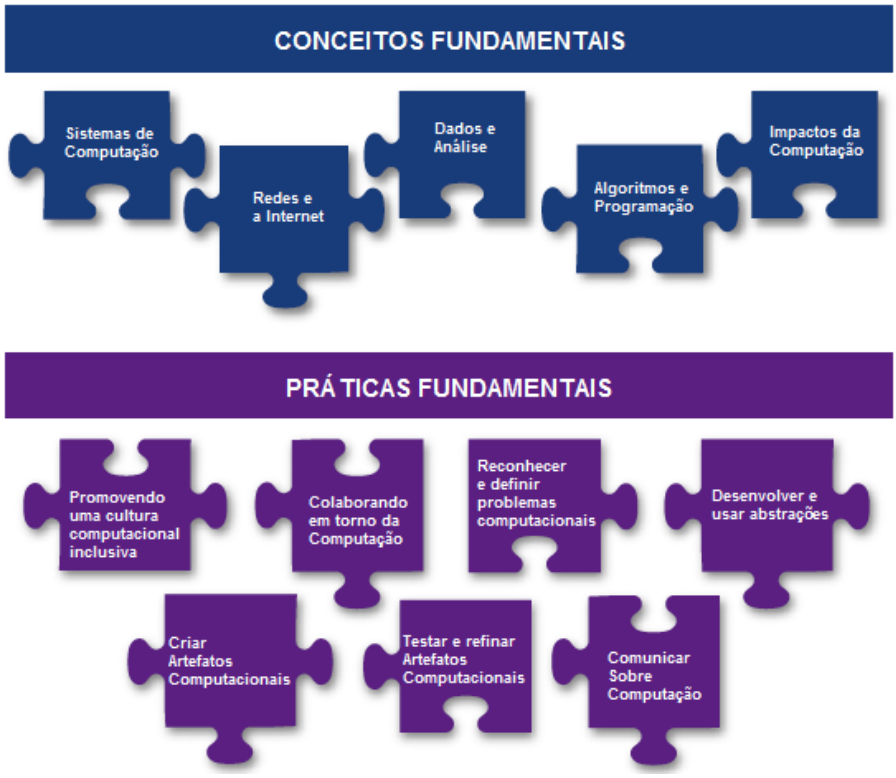


Figura 6: Conceitos e práticas do K-12 (CSTA, 2016)

A Tabela 11 apresenta os conceitos identificados na Figura 5.

Tabela 11: Conceitos presentes no K-12 (CSTA, 2016)

Conceito	Importância	Subconceitos
Sistemas de computação	As pessoas interagem com uma grande quantidade de dispositivos de computação que coletam, armazenam, analisam e atual de acordo com as informações. Se trata dos componentes físicos (*hardware*) e instruções (*software*) que fazem um sistema de computação se comunicar e processar informações.	Dispositivos, *hardware*, *software*, solução de problemas
Redes e a Internet	Dispositivos geralmente não atuam sozinhos. As redes conectam dispositivos a fim de compartilhar informações e recursos e são uma parte cada vez mais integrada da computação. Redes e sistemas de comunicação provêm grande conectividade entre dispositivos de maneira rápida e segura.	Comunicação e organização de rede, *cyber* segurança
Dados e análise	O objetivo do sistema de computação é o processamento de dados. Como a quantidade de dados gerados pelos vários sistemas digitais está rapidamente crescendo, é preciso que o processamento dos dados seja eficiente. Os dados são coletados e armazenados, podendo ser analisados para entender melhor o ambiente e tomar melhores decisões.	Coleta, armazenamento, visualização e transformação, inferência e modelos
Algoritmos e programação	Algoritmos e programação controlam todos os sistemas de computação, empoderando as pessoas a se comunicarem e a resolverem problemas. O processo de desenvolvimento para criar programas significantes e eficientes envolve a escolha de quais informações utilizar e como processá-las e armazená-las. Quebrando um problema em menores, combinando soluções existentes e as comparando.	Algoritmos, variáveis, modularidade, desenvolvimento de programas
Impactos da computação	A computação afeta em muitos aspectos do mundo tanto positiva quanto negativamente. Individualmente ou em comunidade, uma pessoa pode influenciar através de seus comportamentos e interações. Uma pessoa responsável e bem informada deve entender as implicações sociais do mundo digital, incluindo igualdade e acesso à computação.	Cultura, interações sociais, segurança, lei e ética

A Tabela 12 apresenta as práticas identificadas na Figura 5.

Tabela 12: Práticas presentes no K-12 (CSTA, 2016)

Prática	Visão geral
1 - Promover uma cultura de computação inclusiva	Criar uma cultura de computação que seja inclusiva e variada requer estratégias para incorporar pessoas de diferentes gêneros, etnias, habilidades, etc. Para isso, é preciso identificar características pessoais, étnicas, sociais, econômicas e contextos culturais.
2 - Colaborar em torno da computação	Computação colaborativa é o processo de executar uma tarefa computacional trabalhando em duplas ou grupos. O trabalho colaborativo pode ter melhores resultados, porque envolve interação entre os colaboradores e também uma forma de *feedback* instantâneo. A colaboração precisa que cada indivíduo explore as perspectivas, discuta ideias, apresente habilidades e consiga distinguir as personalidades de cada membro.
3 - Reconhecer e definir problemas computacionais	A habilidade de reconhecer oportunidades apropriadas para aplicar a computação é algo essencial. Resolver problemas com uma abordagem computacional requer uma definição do problema, sua quebra em menores partes e a avaliar em que partes uma solução computacional é apropriada.
4 - Desenvolver e usar abstrações	Abstrações são formadas pela identificação de padrões e características comuns de um exemplo específico, criando generalizações. O uso destas generalizações torna o desenvolvimento mais simplificado, diminuindo sua complexidade.
5 - Criar artefatos computacionais	O processo de desenvolvimento de artefatos computacionais engloba tanto expressões criativas quanto a exploração de ideias para criar protótipos e resolver problemas. Artefatos computacionais podem ser criados através da combinação e modificações de artefatos existentes ou desenvolvimento de novos artefatos. São exemplos de artefatos programas, simulações, animações, *apps*.
6 - Testar e refinar artefatos computacionais	Testar e refinar é o processo de melhorar um artefato computacional. Envolve a identificação e correção de erros e a comparação dos resultados esperados com os obtidos. Também é esperado a melhora de performance, usabilidade, confiabilidade e acessibilidade nos artefatos.

7 - Comunicar sobre computação	A comunicação envolve, principalmente, a troca de ideias entre os comunicantes. Na Ciência da Computação, há muita troca de informações sobre o uso e efeitos de certas escolhas computacionais. A troca de experiências é algo muito importante na área, incrementando na facilidade e eficiência na resolução de problemas.

O **pensamento computacional** é peça fundamental das práticas da área de Computação e está presente nas práticas 3 a 6 acima citadas (CSTA, 2016). Pensamento computacional refere-se ao processo envolvido na expressão das soluções como passos de computação ou algoritmos que podem ser executados por um computador (AHO, 2012). Com isso, deve-se levar em conta a definição de problemas e soluções de modo que possa ser realizada por um agente processador (CUNY et al., 2010) e a ideia de que soluções devem tomar forma de passos de computação e algoritmos para serem executadas por um computador (AHO, 2012). O pensamento computacional requer um entendimento das capacidades de um computador, da formulação de problemas que conseguem ser resolvidos por um computador e do desenvolvimento de algoritmos que um computador consegue executar. É essencialmente um processo de resolução de problemas que envolve o desenvolvimento de soluções que reúnem a capacidade dos computadores (CSTA, 2016)

Contudo, como o presente trabalho enfoca no nível 2 do K-12, são apresentados na Tabela 13 os conceitos referentes a esse nível escolar (CSTA, 2016). No Anexo I são apresentados os objetivos de aprendizagem para o nível 2 conforme estudo sobre este *framework*.

Tabela 13: Conceitos para o nível 2 (CSTA, 2016)

Subconceito	**Aprendizado**	***Crosscutting concepts***
Sistemas computacionais		
Dispositivos	A interação entre os humanos e os dispositivos apresenta vantagens e desvantagens. O estudo da relação humano-computador pode auxiliar no projeto de dispositivos e estender as habilidades dos humanos.	Interação Humano-Computador, Privacidade e Segurança
Hardware* e *software	Hardware e *software* determinam a capacidade de um sistema computacional de armazenar e processar informação. O projeto ou a escolha de um desses sistemas envolve algumas características importantes, como funcionalidade, custo, tamanho, velocidade, entre outras.	Relacionamentos do Sistema, Comunicação e Coordenação
Solução de problemas	A solução de problemas requer conhecimentos sobre como o sistema computacional funciona e interage. Um processo sistemático de resolução identifica a origem do problema, seja em um dispositivo ou em um sistema maior.	Relacionamentos do Sistema, Abstração
Redes e a *Internet*		
Comunicação e	Computadores enviam e recebem mensagens baseado em um	Comunicação e Coordenação,

organização de rede	conjunto de regras chamado de protocolo. Os protocolos definem como mensagens são trocadas entre computadores. Segurança, velocidade e confiabilidade são fatores importantes na escolha dos caminhos de envio e recebimento de dados.	Abstração, Privacidade e Segurança
***Cyber* segurança**	A informação enviada e recebida pelas redes pode ser protegida contra acesso não autorizado. As medidas de segurança para proteger as informações abordam proativamente a ameaça de violação de dados pessoais e privados.	Privacidade e Segurança
Dados e análise		
Coleta	As pessoas desenvolvem algoritmos e ferramentas para a automação da coleta de dados pelos computadores. Quando um dado é coletado, este é armazenado de forma que possa ser processado por um computador. A forma de coleta dos dados é influenciada pela viabilidade de ferramentas e pela intenção de uso destes dados.	Interação Humano-Computador
Armazenamento	As aplicações armazenam dados como uma representação. Representações ocorrem em múltiplos níveis, de formatos organizados até armazenamento físico em bits. As ferramentas de *software* utilizadas para acessar as informações traduzem estes dados na forma de bits para uma representação que possa ser compreendida pelo usuário humano.	Abstração
Visualização e transformação	Os dados podem ser transformado para a remoção de erros, expor relações e/ou facilitar o processo para os computadores.	Abstração
Inferência e modelos	Modelos de computação podem ser usados para a simulação de eventos, testar teorias e inferências ou fazer previsões com poucos até milhares de dados. São abstrações que representam fenômenos e usam os dados e algoritmos para enfatizar características e relações em um sistema. Enquanto mais dados forem automaticamente coletados, os modelos podem ser refinados.	Privacidade e Segurança, Abstração
Algoritmos e programação		
Algoritmos	Os algoritmos afetam como as pessoas interagem com os computadores e como eles respondem. Pessoas desenvolvem algoritmos que são generalizáveis a muitas situações. Algoritmos que são legíveis são mais facilmente testados e corrigidos.	Interação Humano-Computador, Abstração
Variáveis	Programadores criam variáveis para armazenar valores de dados de um tipo determinado. Um identificador significativo é associado a cada variável para acessar os dados e realizar operações através desse nome. Variáveis permitem uma flexibilidade para representar diferentes situações, processar diferentes conjuntos de dados e produzir diversos resultados.	Abstração
Controle	Programadores selecionam e combinam estruturas de controle, como laços, manipuladores de eventos e condicionais, a fim de criar um comportamento mais complexo ao programa.	Abstração
Modularidade	Programas usam procedimentos para organizar o código, esconder detalhes da implementação e tornar o código mais fácil de reusar. Procedimentos podem ser reutilizados com outros propósitos em novos programas. A definição de parâmetros para os procedimentos podem tornar o comportamento mais generalizado e aumentar a reusabilidade.	Abstração, Relacionamentos do Sistema
Desenvolvimento de programas	Pessoas projetam soluções pela definição do escopo e das restrições do problema, considerando suas necessidades e desejos, e testando onde este escopo e restrições se encontram.	Interação Humano-Computador, Abstração
Impactos da computação		
Cultura	Avanços em tecnologia relacionados a computação mudam as atividades do dia-a-dia das pessoas. A sociedade é colocada de frente a frente com o crescente aumento da globalização e automação que a computação traz.	Interação Humano-Computador, Relacionamentos do Sistema
Interações sociais	Pessoas podem organizar e se empenhar em volta de problemas e tópicos de interesse através de várias plataformas de comunicação proporcionadas pela computação, como redes sociais e meios de comunicação. Essas interações permitem que problemas possam ser	Interação Humano-Computador, Relacionamentos do Sistema

	analisados por vários pontos de vista de diferentes pessoas.	
Segurança, lei e ética	Existem compensações entre permitir que a informação seja pública e guardar a informação de forma privada e segura. Pessoas podem ser enganadas de maneira a revelarem suas informações pessoais quando mais informações públicas estão disponibilizadas *online* sobre elas.	Privacidade e Segurança, Comunicação e Coordenação

2.3. Engenharia de *Software*

A Engenharia de *Software* é definida como a aplicação de uma abordagem sistemática, disciplinada, quantificável para o desenvolvimento, operação e manutenção de *software*, é a aplicação da engenharia no desenvolvimento de um *software* (BOURQUE; FARLEY, 2014). O escopo da área da ES é composto por áreas do conhecimento e fundamentais (Tabela 14).

Tabela 14: Áreas de conhecimento da ES (BOURQUE; FARLEY, 2014)

Área de conhecimento	**Descrição**
Requisitos de *Software*	Área focada na elicitação, análise, especificação e validação de requisitos de *software*, bem como a gerência de requisitos durante todo o ciclo de vida do produto de *software*.
Modelagem de *Software*	O processo de definir a arquitetura, componentes, interfaces de usuário e outras características de um sistema ou componente e o resultado deste processo.
Construção de *Software*	Se refere a criação detalhada do *software* funcional por meio de uma combinação de codificação, verificação, testes de unidade, testes de integração e *debugging*.
Teste de *Software*	Verificação dinâmica do comportamento de um *software*, o comparando com o comportamento esperado para ele, em um conjunto finito de casos de teste, selecionados geralmente de um domínio de execução infinito.
Manutenção de *Software*	Totalidade das atividades necessárias para prover um suporte economicamente viável ao *software*, abrangendo várias técnicas (reengenharia, engenharia reversa, etc.).
Gerência de Configuração de *Software*	Processo de suporte ao ciclo de vida que beneficia atividades de gerência de projeto, desenvolvimento, manutenção e garantia de qualidade do *software*, como também clientes e usuários do produto final. O processo abrange a identificação, controle, documentação de status, auditoria de configuração de *software* como também a gerência e entrega de entregáveis de *software*.
Gerência de Engenharia de *Software*	Aplicação de atividades de gerência, como planejamento, coordenação, medição, monitoramento, controle e documentação, para assegurar que produtos de *software* e serviços de ES são entregues de forma eficaz e eficiente, com qualidade desejada e que beneficie as partes interessadas.
Processo de Engenharia de *Software*	Se concentra nas atividades realizadas pelos engenheiros de *software* para desenvolver, manter e operar *software*, como requisitos, modelagem, construção, testes, gerência de configuração e outros processos de ES. Essa área de conhecimento define ciclos de vida de *software*, a avaliação e melhoria de processos de *software* como também a medição de *software* e ferramentas de ES de forma geral (p.ex. ferramentas *CASE*).
Modelos e Métodos da Engenharia de *Software*	Impõem estrutura à ES com o objetivo de tornar essa atividade sistemática, repetível e, ultimamente, mais orientada ao sucesso. Essa área abrange tipos de modelos de ES, a análise de modelos de ES, como também vários tipos de métodos de ES (métodos ágeis, métodos formais, etc.).
Qualidade de *Software*	Aborda definições e fornece uma visão geral de práticas, ferramentas e técnicas para a garantia e controle da qualidade de *software* e avaliação do estado da qualidade durante o desenvolvimento, manutenção e implantação.

Prática Profissional de Engenharia de *Software*	Concentra os conhecimentos, habilidades e atitudes que os engenheiros de *software* devem possuir para praticar a ES de maneira profissional, responsável e ética.
Economia de Engenharia de *Software*	Se concentra na tomada de decisões relacionadas à ES em um contexto de negócios alinhando decisões técnicas de *software* com os objetivos de negócio da organização.
Áreas fundamentais da Engenharia de *Software*	
Fundamentos da Computação	Engloba o ambiente de desenvolvimento e operacional no qual o *software* evolui e é executado. O conhecimento sobre o computador e seus princípios subjacentes de hardware e *software* serve como uma estrutura na qual a engenharia de *software* é ancorada.
Fundamentos Matemáticos	Ajuda os engenheiros de *software* a compreender a lógica da resolução do problema, que por sua vez é traduzida em código de linguagem de programação.
Fundamentos da Engenharia	Define algumas habilidades e técnicas fundamentais da engenharia que são úteis para engenheiros de *software*.

As áreas de conhecimento de requisitos, modelagem, construção, testes e manutenção de *software* são consideradas como as fundamentais no processo de desenvolvimento de um *software*.

Requisitos de *Software*

Esta área de conhecimento se concentra na elicitação, análise, especificação, validação e gerenciamento de requisitos de *software* durante todo o ciclo de vida do produto de *software*. Estes requisitos expressam as necessidades e restrições que existem em um produto que contribuem para a solução de um problema real. Os requisitos podem ser subdivididos em funcionais e não funcionais. Os funcionais descrevem as funções que o *software* deve desempenhar, enquanto os não funcionais são os que agem para restringir a solução, também podem ser chamados de requisitos de qualidade. Existem diversas formas para a realização da elicitação de requisitos. As mais utilizadas e conhecidas são as técnicas de entrevista, cenários, prototipação, observações e *user stories* (utilizando-se um método ágil). Estes requisitos, após serem elicitados, são tipicamente priorizados. Ao final do processo de análise de requisitos de *software*, é criada uma especificação de requisitos, a qual pode ser avaliada e aprovada para a continuação do projeto.

Modelagem de *Software*

A modelagem de *software* é uma atividade do ciclo de vida de engenharia de *software* com o objetivo de produzir uma descrição da estrutura interna do *software*

que servirá de base para a sua construção. O resultado desta fase descreve a arquitetura do *software*, decompondo e organizando os componentes e interfaces que possibilitem se entendimento para a futura implementação. Nesta etapa do processo de desenvolvimento também é definida como será a interface do programa com o usuário, seguindo um processo definido para atingir um bom resultado. A documentação da modelagem de *software* pode ser realizada de diversas formas, são alguns exemplos: diagramas de classes e objetos, diagrama entidade-relacionamento, diagramas de atividade, diagramas de fluxos de dados, fluxogramas, diagramas de sequência, etc.

Construção de *Software*

A construção de *software* se refere de fato a criação detalhada do produto de *software* pela combinação da programação, verificação, testes de unidade, testes de integração e *debugging*. Para a construção do *software* ser uma etapa com um bom resultado, diversos padrões devem ser especificados, como a forma de documentação do código, nomeação de variáveis e/ou métodos do código, ferramentas utilizadas. Esta etapa também depende que os programadores levem em conta uma série de considerações sobre como escrever o código, sempre o testando e sempre pensando em criar módulos que sejam reusáveis. Existem algumas tecnologias de construção de *software* como a criação de módulos executáveis, métodos para construção de sistemas distribuídos, *test-first programming*, entre outras.

Teste de *Software*

A fase de teste de *software* consiste na verificação dinâmica de um programa, analisando se os comportamentos esperados são atingidos para um conjunto finitos de casos de testes, selecionados de um domínio de execução infinito.

Os testes podem ser definidos por uma série de níveis, como, por exemplo, os testes de unidade e de integração. Existem diversas técnicas para a realização de testes de *software*, algumas delas são: teste exploratório, heurísticas de

observação de usuário, máquina de estados finitos, especificações formais, modelos de fluxo de trabalho.

Manutenção de *Software*

Um produto de *software* pode sofrer mudanças mesmo após ser entregue, e é a fase de manutenção de *software* que trata deste assunto. Esta fase se concentra no suporte aos usuários com o produto já em uso, mas também já se preocupa com detalhes muito antes disso. Existem categorias de manutenção de *software*, são elas: manutenção corretiva, adaptativa, perfectiva e preventiva, cada uma com uma função bem definida nesta etapa. O processo de manutenção se baseia num ciclo contendo a identificação de problemas e/ou melhorias, aceitação ou não destas melhorias e modificação/implementação do que foi aceitado. O produto de *software* pode então ser aposentado ou migrado para outro ambiente.

Processo de Engenharia de *Software*

Um processo de Engenharia de *Software* consiste num conjunto de atividades inter-relacionadas que transformam uma ou mais entradas em saídas e para isto consome recursos para alcançar esta transformação. Nesta área de conhecimento, os processos de *software* se concentram nas atividades realizadas para produzir, manter e operar sistemas de *software*, como requerimentos, design, construção, testes, configuração, manutenção, entre outros processos. Um ciclo de vida de desenvolvimento de *software* inclui processos de *software* utilizados para especificar e transformar requisitos em um produto de *software*. Um ciclo de vida de um produto de *software* inclui um ciclo de vida de desenvolvimento mais processos adicionais que provêm a distribuição, manutenção, suporte, evolução, aposentadoria deste produto. Existem ferramentas que auxiliam os processos de *software* e suportam muitas das notações utilizadas para definir, implementar e gerenciar os processos e modelos de ciclo de vida de *software*. Estas ferramentas provêm editores para anotações, como diagramas de fluxo, diagrama de estados, redes de Petri e diagramas de atividade *UML*. As ferramentas *CASE* podem auxiliar na execução de

definições de processos e fornecer orientação a humanos na execução de processos bem definidos.

Modelos e Métodos da Engenharia de *Software*

Os modelos e métodos criam uma estrutura para a engenharia de *software* com o objetivo de fazer desta atividade algo sistemático e muito mais orientado ao sucesso. A utilização de modelos provê uma abordagem para a resolução de problemas, uma notação e procedimentos para a construção de modelos e análise. Já os métodos provêm uma abordagem para a especificação, design, construção, teste e verificação sistemática do produto final de *software*. Em mais detalhes, os modelos possibilitam a utilização da engenharia de *software* como uma abordagem organizada e sistemática para a representação de aspectos significantes do *software*, facilitando a tomada de decisões e a comunicação destas decisões aos responsáveis pelo projeto. Por sua vez, os métodos permitem uma abordagem organizada e sistemática para o desenvolvimento de *software*. Existem diversos métodos da engenharia de *software* e cabe ao engenheiro a escolha mais adequada dependendo do *software* a ser produzido. São exemplos de métodos: métodos heurísticos, métodos formais, de prototipação, métodos ágeis. Em especial, os métodos ágeis são considerados “métodos leves” e têm como características ciclos de desenvolvimento iterativos, times auto-organizados, designs mais simples, refatoração de código, desenvolvimento orientado a testes, envolvimento frequente com o cliente e ênfase em apresentar módulos funcionais do *software* em produção a cada ciclo. Um exemplo de método ágil é o *XP*, definido por utilizar *user stories* ou cenários para a análise de requisitos, desenvolve testes antes da implementação, tem envolvimento da equipe diretamente com o cliente, utiliza programação em pares e fornece refatoração e integração contínuas de código. Outro exemplo muito utilizado é o *Scrum*, que é mais voltado ao gerenciamento de projetos do que os outros. O desenvolvimento é baseado em *sprints* que não dura mais do que 30 dias, uma lista de tarefas (*product backlog*) é produzida, com a definição, priorização e estimação destas tarefas. O *Scrum* ainda fornece reuniões diárias, que garante sempre o acompanhamento do que foi planejado.

2.4. Design de Interface

Uma das qualidades mais importantes em aplicativos móveis é a utilidade. Os usuários, à primeira vista, podem até se interessar por um *app* e instalá-lo em seu celular. A tempo de vida deste aplicativo no dispositivo é que indica se ele era realmente útil ou não. Um aplicativo deve, além de ser agradável e funcional, facilitar a vida dos usuários e isso faz com que eles se engajem neste *app* (THINK WITH GOOGLE, 2016). Outra qualidade importante é a usabilidade. Um aplicativo ter uma boa usabilidade indica que o usuário terá facilidade ao interagir com ele, é ter uma fluidez entre as telas que garanta que o usuário se sentirá confortável ao utilizá-lo (VORTIGO, 2017).

Um método para alcançar a criação de produtos úteis e com usabilidade é o *design thinking* (REIS, 2016). O *design thinking* é um processo analítico e criativo que engaja uma pessoa a, durante o desenvolvimento, experimentar, criar, propor modelos, obter *feedback* e com esta análise incrementar o desenvolvimento (RAZZOUK; SHUTE, 2012). Ele se caracteriza por ser um processo de desenvolvimento que inclui a execução de fases conforme mostrado na Figura 10. O plano de desenvolvimento é um processo iterativo e cada fase deve atingir um número de resultados esperados através de entregáveis (CHEN; HUANG, 2017). A Tabela 15 apresenta as fases, mostrando as atividades que devem ser realizadas e os entregáveis do processo de *design thinking*.

Tabela 15: O processo do *design thinking* (CHEN; HUANG, 2017)

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

Testes	- Realização de testes	- Lista de feedback dos usuários - Observações - Avaliação

A usabilidade é definida como a medida a qual um produto pode ser usado por um usuário específico para alcançar seus objetivos em termos de efetividade, eficiência e satisfação em um contexto de uso específico (ABNT, 2017). A efetividade é definida pela precisão e completude em que os usuários atingem seus objetivos, a eficiência é relativa aos recursos gastos para a conclusão destes objetivos e a satisfação é analisada pelo estudo das atitudes positivas apresentadas no uso de um produto (ABNT, 2017).

A usabilidade e a experiência de usuário trabalham em conjunto. A usabilidade tem sua ênfase na conclusão de uma tarefa em particular em um contexto particular de uso. Entretanto, nas tecnologias atuais, os usuários não estão necessariamente preocupados em realizar apenas uma tarefa. Desta forma, o conceito de experiência de usuário surgiu para cobrir os componentes das interações e reações dos usuários que vão além da eficácia, eficiência e interpretações convencionais de satisfação (PETRIE; BEVAN, 2009).

Para resultar uma experiência esperada de um produto e com grau de usabilidade desejada, se adota um processo de Engenharia de Usabilidade. Este processo se caracteriza por ser uma abordagem centrada no usuário e é apresentado na Figura 7 (ISO, 1999).

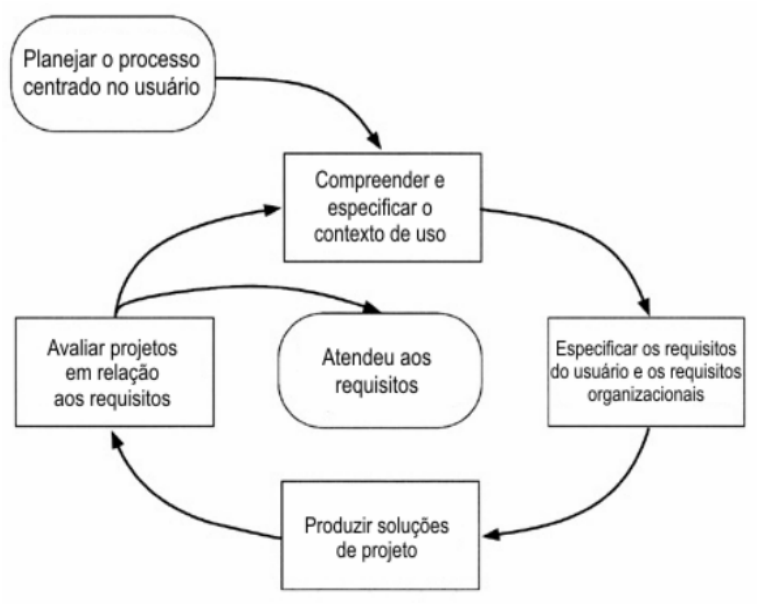


Figura 7: Processo centrado no usuário (ISO, 1999)

A atividade de “Compreender e especificar o contexto de uso” tem por objetivo obter informações sobre características dos usuários, ambiente de uso e as tarefas que serão executadas com o produto. “Especificar os requisitos do usuário e os requisitos organizacionais” se refere a especificação dos requisitos do usuário e da organização, determinando os critérios de sucesso para a usabilidade do produto, bem como limitações do projeto. “Produzir soluções do projeto” caracteriza a atividade de explorar as possíveis soluções, as descrevendo por meio de protótipos. A atividade de “Avaliar projetos em relação aos requisitos” diz respeito à avaliação da usabilidade do projeto em relação às tarefas do usuário, confirmando o nível em que os requisitos foram alcançados. Este processo é cíclico e as atividades são realizadas até que a usabilidade do projeto atinja todos os requisitos especificados (ISO, 1999).

O processo da Engenharia de Usabilidade é algo sistemático e passa por vários níveis, desde a análise do problema até o design visual final do produto de *software*. A Figura 8 apresenta estes níveis, indicando uma precedência sobre as atividades do processo (GARRET, 2010).

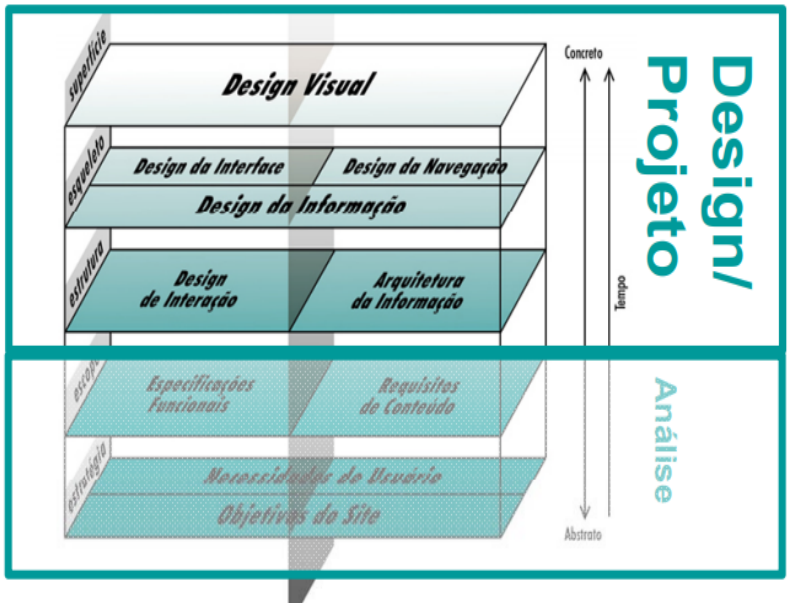


Figura 8: Níveis do processo de design centrado no usuário (GARRET, 2010)

O design visual ou gráfico garante que as informações que o produto possui sejam descritas de tal forma que expressem o processo de construção da experiência do usuário e a comunicação entre ele e a interface (FEIJÓ; BALDESSAR; VIEIRA, 2013). O design visual envolve a seleção de fontes, cores, imagens, espaçamento, alinhamento com o objetivo de criar uma hierarquia e um significado aos componentes na tela (GOOGLE, 2017).

Um dos guias de estilo existentes para o design de interface é o *Material Design*, utilizado nos dispositivos *Android* (e *iOS*), tanto no sistema operacional como em aplicativos sendo proposto pela *Google*. Um dos seus principais objetivos é criar um sistema que tenha uma experiência unificada para o usuário em diferentes plataformas e tamanhos de dispositivos (GOOGLE, 2017). As Figuras 9-11 apresentam alguns exemplos disso.

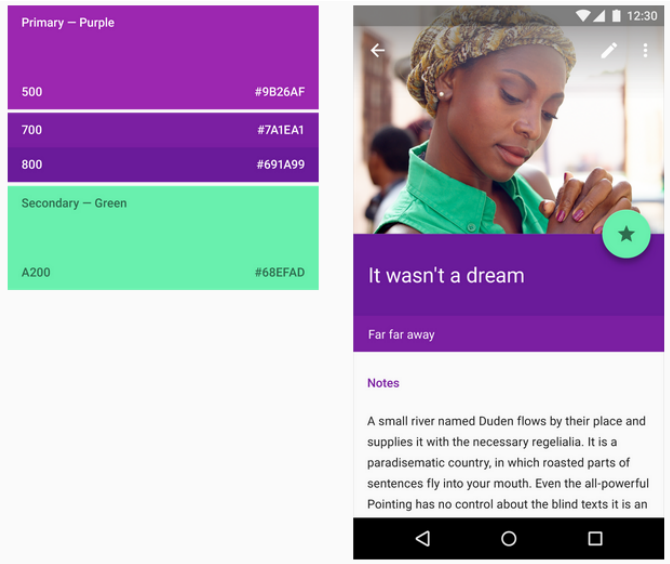


Figura 9: Exemplo de paleta de cores e uso de imagens (GOOGLE, 2017)

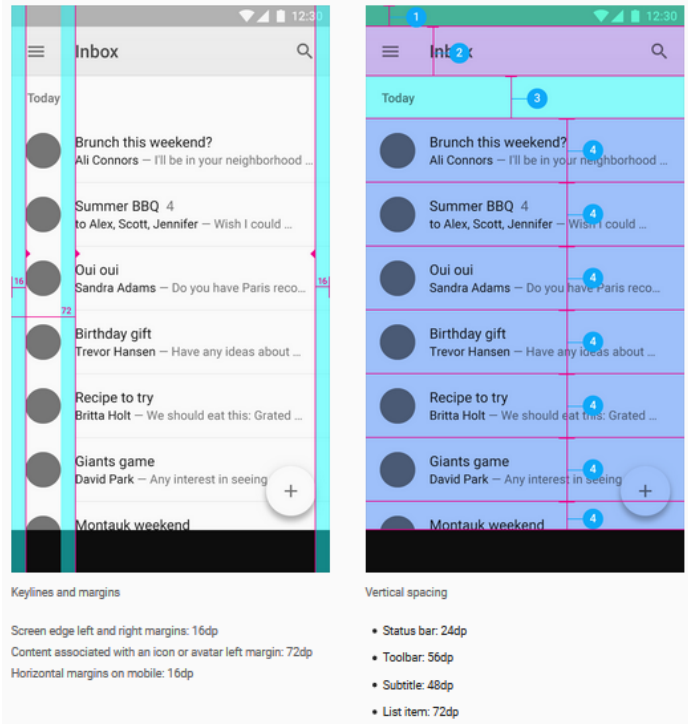


Figura 10: Diretrizes para o desenvolvimento da interface (GOOGLE, 2017)

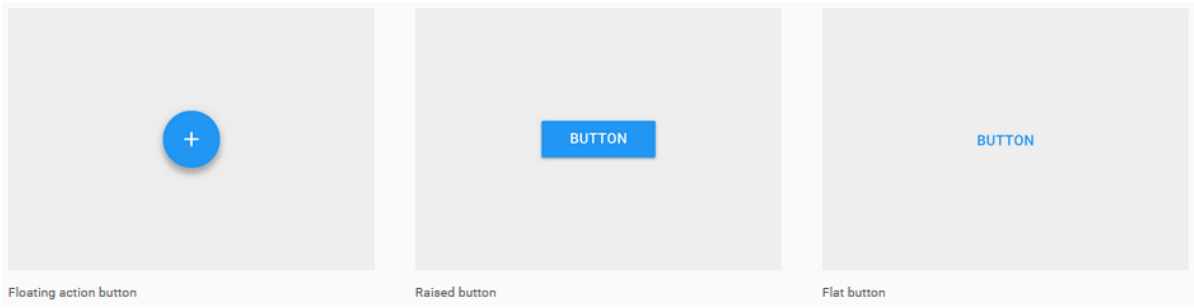


Figura 11: Estilos de botões que devem ser utilizados na interface (GOOGLE, 2017)

As competências de Design de Interface são muito importantes porque criam uma forma sistemática e eficiente de resolução de problemas. Desta forma, o ensino destes competências já a nível de Ensino Fundamental trará muitos benefícios aos estudantes, que já desde cedo dominariam esta abordagem.

2.5. *App Inventor*

Visando o forte uso de celulares (principalmente do o sistema operacional *Android),* identifica-se a oportunidade de ensinar computação por meio do desenvolvimento de *apps*. Um ambiente de programação voltado ao ensino de desenvolvimento de *apps* no ensino básico é o *App Inventor*. O *MIT App Inventor* (MITa, 2017) é uma ferramenta intuitiva de programação visual que permite qualquer pessoa criar *apps* funcionais para *Android*. Esta ferramenta baseada em programação com blocos facilita a criação de aplicativos complexos em um tempo muito menor do que se fosse utilizado a programação tradicional. A sua forma de programação e desenvolvimento incremental permite ao desenvolvedor o foco na lógica da programação em vez de pensar na sintaxe de uma linguagem (POKRESS; VEIGA, 2013). Programar por meio de blocos inspira aos programadores seu poder intelectual e criativo, possibilitando até mesmo jovens alcançarem um impacto social, trazendo um valor a sua comunidade (MITa, 2017). Uma das principais características do *App Inventor* é a possibilidade do desenvolvedor poder analisar sua criação ao mesmo tempo que a constrói. Isto permite que o programa seja feito de forma incremental, diminuindo consideravelmente o número de erros produzidos com os testes realizados no mesmo momento (POKRESS; VEIGA, 2013). Ao mesmo tempo que modificações são realizadas, as mudanças podem ser vistas

diretamente no *smartphone* e a corretude ser analisada. Além de o programador ter um *feedback* imediato do seu trabalho, esta é uma maneira de testar e encontrar erros no programa enquanto ele está em construção.

Uma vez que o *app* esteja terminado, ele pode ser enviado diretamente ao *smartphone* conectado ou exportado no formato *apk* para distribuição ou *upload* na *Google Play Store*. Caso o usuário deseje, ele pode postar seu aplicativo desenvolvido em uma galeria para que outros usuários possam visualizá-lo ou também utilizá-lo como base para outra aplicação. De fato, o *App Inventor* é uma ferramenta para o pensamento e engajamento computacional, tendo provado isto em diversas aplicações bem sucedidas (POKRESS; VEIGA, 2013).

A interface de usuário do *App Inventor* é baseada em duas partes: *Designer* (Figura 12) para a seleção dos componentes visuais e não visuais que farão parte da interface do *app* e *Blocos* (Figura 13) para fazer a programação dos comportamentos do aplicativo (MITe, 2017).

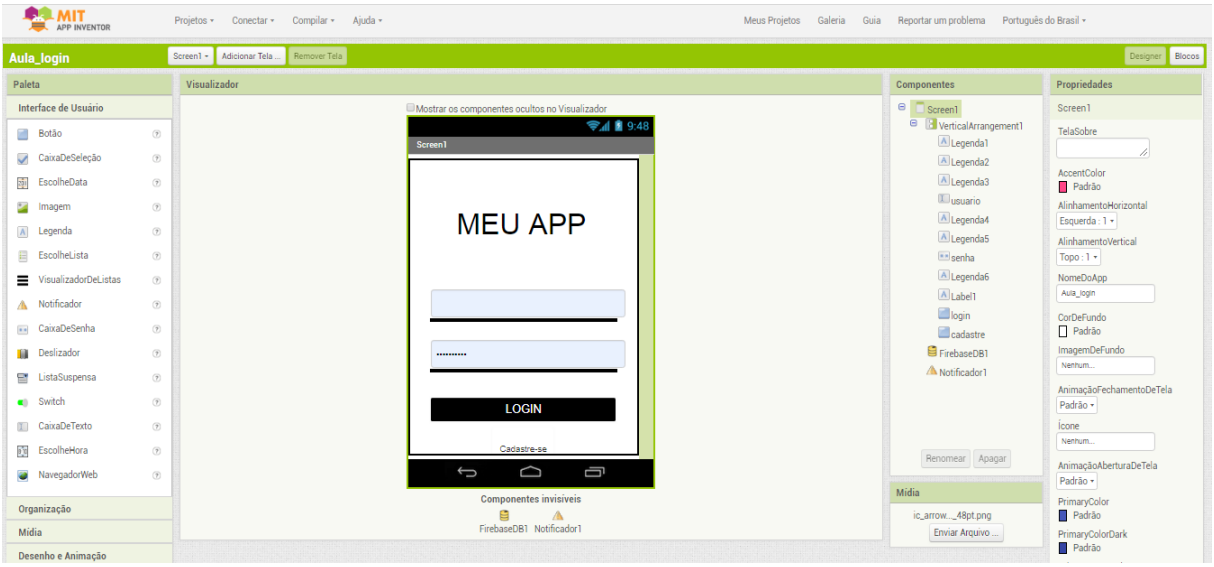


Figura 12: Designer no App Inventor (MITe, 2017)

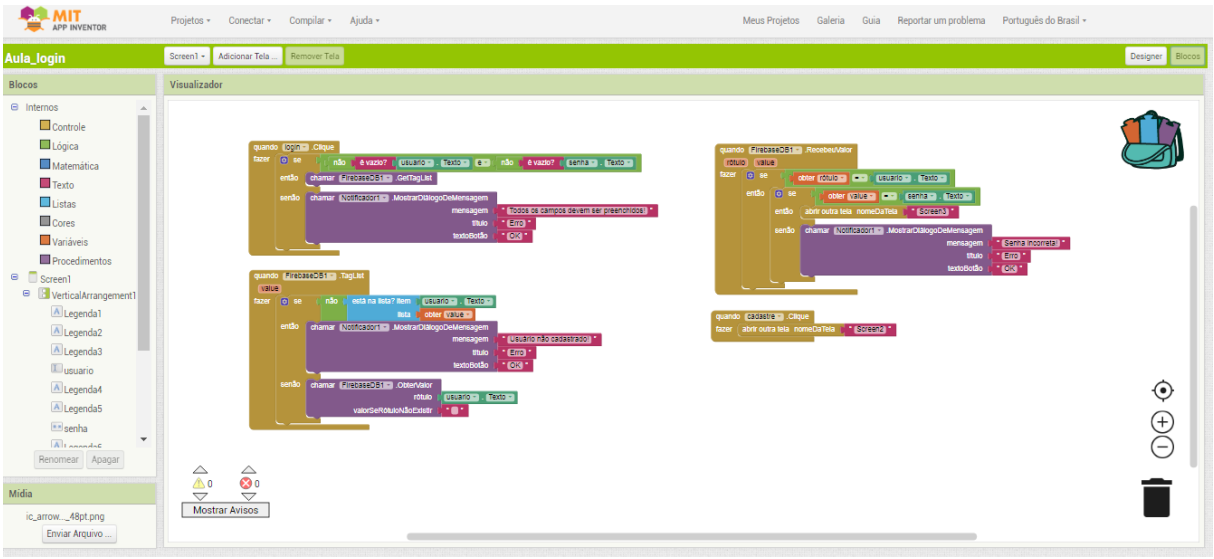


Figura 13: Blocos no App Inventor (MITe, 2017)

O *Designer* apresenta quatro áreas: Paleta, Visualizador, Componentes e Propriedades. Na Paleta estão presentes os componentes disponíveis para uso na interface com o usuário e no *app* em si, sendo eles visuais ou não. Todos os componentes têm categorias específicas, facilitando sua identificação. Para adicionar um item ao *app*, basta clicar e arrastá-lo para dentro da tela presente no Visualizador. O Visualizador apresenta os componentes que já foram adicionados ao *app* e o usuário consegue ter um *preview* de como o *app* se mostrará no smartphone. Também mostra os componentes não visuais que foram adicionados. Os itens arrastados para dentro da tela do *app* não se posicionam onde foram soltos, sendo preciso um esforço para organizá-los corretamente onde desejado. A área de Componentes apresenta todos os componentes adicionados ao *app* de maneira hierarquizada, podendo identificar que componentes estão “dentro” de outros. É possível selecionar um item e em seguida removê-lo ou renomeá-lo. A última área é a de Propriedades. Quando um item é selecionado, esta área mostra todas as propriedades que podem ser alteradas neste componente, como cor, imagem, altura, largura, tamanho da fonte, entre outras. Estas propriedades variam de acordo com o tipo de componente que foi selecionado (MITe, 2017).

O detalhamento das categorias do *App Inventor,* mostrando todos os componentes e funções, é apresentado no Anexo III.

O **Blocos** do *App Inventor* possibilita a criação da lógica nos eventos do aplicativo por meio da programação. Todos os componentes adicionados na aba Designer podem ser utilizados na criação de algoritmos para realizarem ações de acordo com o necessário. A programação é realizada com blocos lógicos que, como mencionado acima, simulam a criação de *software* por linhas de código. Esta parte é dividida em duas áreas, a área Blocos e a Visualizador (MITe, 2017).

A área de Blocos contém, para cada componente adicionado ao *app* e mais os componentes internos, os blocos de programação relacionados. Ao selecionar um dos componentes, é apresentado um número finito de blocos na área Visualizador relacionados ao componente, representando tudo o que é possível realizar com ele. Além disso, para cada componente interno é atribuída uma cor a fim de facilitar a visualização e entendimento quando os blocos estão ligados. A Figura 14 mostra como isso ocorre ao clicar no componente interno Controle, apresentando todos os blocos de programação deste componente (MITe, 2017).

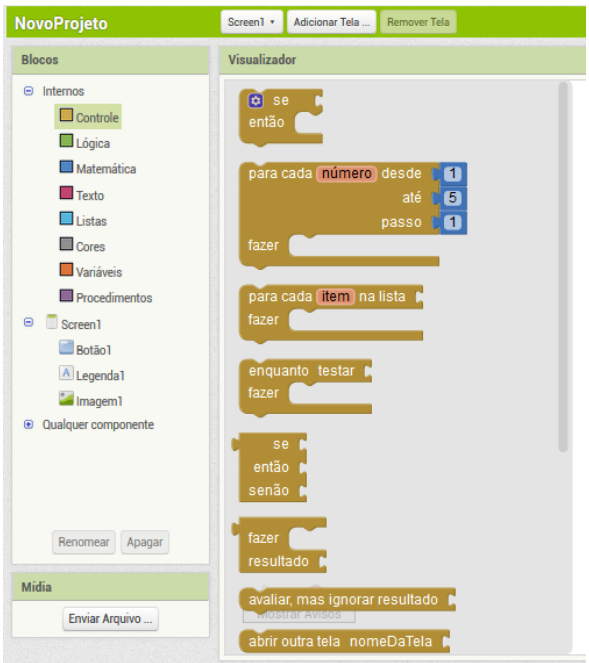


Figura 14: Visualização dos blocos de um componente interno (MITe, 2017)

A Figura 15 mostra como isso ocorre ao clicar no componente adicionado ao *app* Botão1, apresentando todos os blocos de programação deste componente.

Figura 15: Visualização dos blocos de um componente adicionado (MITe, 2017)

A Tabela 16 descreve todos os componentes internos da área Blocos os quais principalmente representam a lógica de programação do app.

Tabela 16: Detalhamento dos componentes internos da área Blocos (MIT, 2017e)

Componente	**Descrição**
Controle	Este componente possui os blocos responsáveis pelo controle das ações no *app*. Estão presentes os blocos de condicionais *if-else-elseif* e laços (para-faça e enquanto). Também estão presentes os blocos referentes a movimentação entre as telas.
Lógica	Os blocos incluídos neste componente são relacionados a lógica booleana, como o verdadeiro e falso, igualdade e comparações “e” e “ou”.
Matemática	O componente Matemática possui os blocos que envolvem a lógica matemática, incluindo igualdade numérica e todas as 4 operações. Também existem blocos para selecionar um número randômico e vários outros tipos.
Texto	Neste componente estão os blocos que se referem a criação, operação e modificação de textos. Como por exemplo concatenar dois textos e verificar se possui uma tal parte em um texto existente.
Listas	Os blocos incluídos neste componente se referem a criação, operação e modificação de listas, como verificar se tal item está na lista e inserir, modificar e remover algum item.
Cores	Este componente possui todos os blocos com as cores padrão do *App Inventor*. Também possui um bloco em que é possível a alteração da cor de um componente pela inserção do código *RGB* de uma cor específica.
Variáveis	O componente Variáveis inclui todos os blocos relacionados a variáveis do *app*, como a criação e modificação de uma variável e a obtenção do seu valor.
Procedimentos	Este componente possui os blocos relacionados a criação de um novo procedimento para o *app*.

A área Visualizador da aba Blocos é o local onde os blocos são alocados, como se fosse o fundo branco de uma pintura (Figura 16). Além disso, quando um componente é selecionado na área Blocos, é no Visualizador que são listados os blocos de programação. Nesta área é possível arrastar blocos para uma “mochila” onde eles ficam armazenados para a utilização em outras telas, simulando um “copiar e colar”. Também é possível a opção de aproximar ou afastar para melhor visualizar os blocos e, além disso, a opção de excluir um ou mais blocos os arrastando para a “lixeira”.

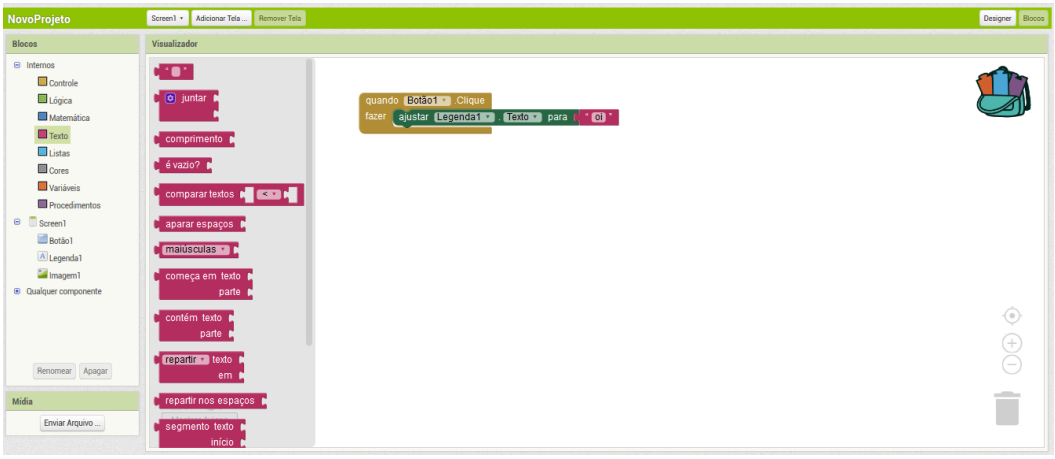


Figura 16: Visualização da área Blocos (MITe, 2017)

Voltado ao ensino de desenvolvimento de *apps*, foi criada uma ferramenta para automatizar a correção e aplicação de uma nota a aplicativos criados no *App Inventor* chamada *CodeMaster* (DEMETRIO, 2017). Esta ferramenta permite um professor avaliar os aplicativos criados por seus alunos avaliando-os automaticamente com uma nota de acordo com o número de componentes utilizados. Como a avaliação manual de todos os programas construídos seria algo muito complicado, a ferramenta se torna algo que facilita muito o trabalho do professor ao avaliar a aprendizagem de seus alunos (WANGENHEIM et al., 2018). Como uma evolução da ferramenta *CodeMaster* surgiu sua segunda versão (ALVES, 2019), a qual utiliza uma rubrica que avalia uma série de questões sobre o desempenho em relação aos conceitos do pensamento computacional baseada sobretudo no currículo da CSTA (2016). Esta ferramenta permite também que os alunos avaliem seus *app*s sem a necessidade um professor.

3. ESTADO DA ARTE E PRÁTICA

Para elicitar o estado da arte e a prática sobre como ensino de ES e design de interface/Engenharia de Usabilidade é abordado na educação básica, foram realizados dois estudos de mapeamento sistemático detalhados a seguir.

3.1. Estado da arte e prática do ensino de ES na educação básica

Para elicitar o estado da arte e a prática sobre se e como são ensinados conceitos de Engenharia de *Software* na educação básica, foi realizado um estudo de mapeamento sistemático seguindo o procedimento proposto por Petersen, Vakkalanka e Kuzniarz (2015). Este mapeamento foi realizado por meio da produção de um relatório técnico, executado em cooperação no laboratório de pesquisa (PINHEIRO et al., 2018). O detalhamento das informações apresentadas abaixo pode ser conferido no relatório.

3.1.1. Definição do Mapeamento do ensino de ES

Questão de Pesquisa. Existem (e quais as suas características) unidades instrucionais que ensinam competências de Engenharia de *Software* 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 ES são ensinados na UI?
PA3. Quais as características instrucionais da UI?
PA4. Quais são as características de contexto da UI?
PA5. Como a UI foi desenvolvida?
PA6. Como a qualidade da UI é avaliada?

Critérios de inclusão/exclusão. Considera-se apenas artigos revisados por pares cujo foco é ensinar computação adotando práticas e/ou atividades de ES na educação básica. São excluídos artigos que focam em ensinar computação para na

educação superior e/ou artigos que apresentam UIs para o ensino de computação sem abordar conceitos de Engenharia de *Software*. Incluímos também literatura secundária descoberta por meio das referências da literatura primária encontrada (Verhoeff, 2006).

Critério de Qualidade. Considera-se apenas artigos que apresentam informações substanciais em relação ao ensino de conceitos de Engenharia de *Software*, indicando, por exemplo, conteúdo das aulas, materiais didáticos, etc.

Fonte de dados. A busca foi realizada 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 *online* foram utilizados para pesquisar UIs ensinadas por meio de cursos *online*.

Definição da *string* de busca. A *string* de busca foi composta de conceitos relacionados à questão de pesquisa, considerando também sinônimos, conforme indicado na Tabela 17. O termo “Engenharia de *Software*” foi escolhido por ser o principal conceito a ser pesquisado. O termo “educação básica” é utilizado para restringir o nível de ensino enfocado pela UI. O termo “unidade instrucional” representa o objetivo de identificar e caracterizar diversos tipos de unidades instrucionais.

Tabela 17: Palavras chaves referente ao ensino de ES

Conceito principal (termo)	Sinônimos	Tradução
Engenharia de *Software*	*UML*, processo de desenvolvimento de *software*	*software engineering, UML, software development process*
Educação básica	ensino primário, ensino fundamental, ensino médio	*school, K-12*
Unidade instrucional	curso, ensino, aprendizagem	*learn, teaching, MOOC*

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 na Tabela 18.

Tabela 18: String de busca referente ao ensino de ES

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

3.1.2. Execução da busca referente a ensino de ES

A pesquisa foi executada em março de 2018 pelo autor em conjunto com outros pesquisadores do GQS/INCoD/INE/UFSC. A busca foi feita em duas etapas. Na primeira etapa a busca foi feita via SCOPUS com o objetivo de encontrar publicações sobre UIs existentes. Essa busca inicial retornou 466 artigos (Tabela 19). A partir dos resultados da busca foram selecionados artigos potencialmente relevantes de acordo com os critérios de inclusão e exclusão, analisando rapidamente o título, resumo e palavras-chave. Como resultado foram identificados 29 artigos potencialmente relevantes. Na segunda etapa de seleção, analisamos o texto completo dos artigos pré-selecionados para analisar sua conformidade com os critérios de inclusão/exclusão e o critério de qualidade. Como resultado foram identificados 15 artigos relevantes. Todo este processo foi feito pelos pesquisadores em conjunto, sempre discutindo a seleção até chegar a um consenso.

Tabela 19: Quantidade de artigos por etapa de seleção por repositório referente ao ensino de ES

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

Diversas UIs encontradas foram excluídas por não estarem no foco da pergunta de pesquisa, por exemplo, UIs que apenas relatam a importância do ensino de ES na educação básica (BELL et al., 2014; BOLLIN et al., 2016) ou por não apresentam detalhes sobre o ensino de ES (AZENKOT et al., 2011).

Numa segunda etapa da busca foram também procuradas informações sobre cursos voltados ao ensino de ES na educação básica nos fornecedores de ensino *on-line*. Foram analisados os cursos disponíveis adotando os mesmos critérios de inclusão/exclusão. Como resultado desta pesquisa adicional foi encontrada uma UI relevante (DE KEREKI; MANATAKI, 2015). Outros cursos encontrados voltados ao

ensino de computação na educação básica foram desconsiderados por não abrangerem conceitos de ES de forma explícita (technovationchallenge.org) e/ou por não disponibilizarem detalhes da(s) UI(s) (afsenyc.org).

Foi encontrada também uma UI por meio das referências da literatura primária (VERHOEFF, 2006).

3.1.3. Análise dos dados referente ao ensino de ES

Para responder à questão de pesquisa, extraímos informações relevantes às perguntas de análise conforme especificado na Tabela 20.

Tabela 20: Especificação das informações extraídas referente ao ensino de ES

Pergunta de análise	Dados a extrair	Descrição
PA1. Quais UIs existem?	Nome	O nome ou o autor da UI
	Referência	Referência bibliográfica
PA2. Qual(is) competências de ES são ensinados na UI?	Objetivo(s) de aprendizagem referente a ES	Identificação do(s) objetivo(s) descrevendo o que o aluno deve aprender em relação a ES
	Áreas de conhecimento de ES	Áreas de conhecimentos da ES abordados na UI
	Métodos/técnicas de ES	Métodos/técnicas de ES abordados na UI
	Ferramentas de ES	Ferramentas de *software* adotadas no ensino de ES
PA3. Quais as características instrucionais da UI?	Objetivo de aprendizagem da UI como um todo	Identificação dos objetivos de aprendizagem em geral da UI
	Descrição geral	Breve descrição geral da UI apresentando as suas principais características
	Modo de ensino	Identificação do modo do ensino (presencial ou *on-line*)
	Ambiente de programação	Ambiente/linguagem de programação utilizado na UI
	Método Instrucional	Métodos instrucionais utilizados na a UI
	Material Instrucional	Material(is) instrucional(is) utilizados na a UI
	Método/instrumento de avaliação	Método(s)/instrumento(s) utilizado(s) para a avaliação de aprendizagem do aluno utilizados na UI
	Língua	A(s) língua(s) em qual a UI está disponível
	Licença	Licença de uso da UI
	Capacitação dos instrutores	Indicação se a UI está sendo acompanhado por uma capacitação dos instrutores
PA4. Quais são as características de contexto da UI?	Nível escolar	Nível escolar para qual a UI é projetada
	Duração da UI	Duração da UI (número de hora/aula)
	Pré-requisitos	Pré-requisitos em relação a competências da computação prévias do aluno
PA5. Como a UI foi desenvolvida?	Método de desenvolvimento da UI	Indicação do método adotado para o desenvolvimento da UI
PA6. Como a qualidade da UI é avaliada?	Tipo de estudo de avaliação da UI	Indicação do tipo de estudo (research *design*) da avaliação da UI
	Fatores avaliados	Indicação dos fatores que foram avaliados
	Método de coleta de dados	Indicação do(s) método(s) de coleta de dados adotado(s) na avaliação da UI
	Tamanho de amostra	Quantidade de pontos de dados utilizadas na a avaliação da UI

	Avaliações replicadas	Indicação de possíveis replicações da avaliação em vários contextos
	Método de análise de dados	Indicação do(s) método(s) de análise de dados utilizado(s) na avaliação da UI
	Descobertas	Descrição dos principais resultados, pontos positivos e negativos identificadas na avaliação da UI

Os artigos foram lidos de forma completa e os dados foram extraídos pelos autores. A extração dos dados foi dificultada em vários casos pela 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 suficientes sobre a UI. Nestes casos, algumas informações foram inferidas a partir do artigo, incluindo, por exemplo, a descrição dos objetivos de aprendizagem, idioma da UI, necessidade de competências prévias de computação. Observamos também que a maioria dos estudos não explicita a maneira como as UIs foram desenvolvidas, além de falta de informações relevantes em relação a sua avaliação, por exemplo, não abordando ameaças à validade. Nos casos em que o artigo não apresenta nenhuma informação a ser extraída, indicamos a falta desta informação como não informado (NI).

São apresentados os dados analisados em seguida para cada uma das perguntas de pesquisa. Detalhes em relação aos dados extraídos são documentados em Pinheiro, Missfeldt Filho e Wangenheim (2018).

PA1. Quais UIs existem?

Como resultado da pesquisa foram identificadas no total 17 unidades instrucionais voltadas ao ensino de computação que de alguma forma abordam também o ensino de Engenharia de *Software* a nível da educação básica (Tabela 21).

Tabela 21: Artigos descobertos na pesquisa referente ao ensino de ES

Citação	Referência bibliográfica
(Bollin; Sabitzer, 2015)	Bollin, A; Sabitzer, B. (2015) Teaching *Software* Engineering in schools on the right time to introduce *Software* Engineering concepts. In: Proceedings of the Global Engineering Education Conference, Talín, Estônia, p. 518-525.
(Collofello, 2002)	Collofello, J. S. (2002) Creation, deployment and evaluation of an innovative secondary school *software* development curriculum module. In: Proceedings of the 32nd Annual Frontiers in Education, Boston, MA, EUA, p. 1-4.
(Corbett & Nesiba, 2015)	Corbett, K.; Nesiba, N. (2015) Programming *design* process: Providing K-12 students with a structure to attain programming goals. In: Proceeding of the Frontiers in Education Conference, El Paso, TX, EUA, p. 1-4.
(De Kereki & Manataki, 2016)	De Kereki, I. F.; Manataki, A. Code Yourself! An introduction a programming. Disponível em: <https://pt.coursera.org/learn/intro-programming>. Acesso em: 29 de março de 2018. De Kereki, I. F.; Manataki, A. (2016) "Code Yourself" and "A Programar": a bilingual MOOC for teaching Computer Science to teenagers. In: Proceeding of the Frontiers in Education Conference (FIE), Erie, Pensilvânia, EUA, p. 1-9.
(Fronza, El Ioini & Corral, 2017)	Fronza, I.; Ioini, N.; Corral L. (2017) Teaching Computational Thinking Using Agile *Software* Engineering Methods: A *Framework* for Middle Schools. ACM Transactions on Computing Education, v. 17, n. 4.
(Fronza, El Ioini & Corral, 2016)	Fronza, I.; Ioini, N.; Corral L. (2016) Blending Mobile Programming and Liberal Education in a Social-Economic High School. In: Porceedings of the IEEE/ACM International Conference on Mobile *Software* Engineering and Systems, Austin, Texas, EUA, p. 123-126
(Fronza, El Ioini & Corral, 2015)	Fronza, I.; El Ioini, N.; Corral, L. (2015) Students want to create *app*s: leveraging computational thinking to teach mobile *software* development. In: Proceedings of the 16th Annual Conference on Information Technology Education, Berlim, Alemanha, p. 21-26.
(Hermans & Aivaloglou, 2017)	Hermans, F; Aivaloglou, E. (2017) Teaching *software* engineering principles to K-12 students: a MOOC on *Scratch*. In: Proceedings of the 39th International Conference on *Software* Engineering: *Software* Engineering and Education Track, Buenos Aires, Argentina, p. 13-22.
(Köhler; Gluchow & Brugge, 2012)	Köhler, B; Gluchow, M; Brugge, B. (2012) Teaching Basic *Software* Engineering to Senior High School Students. In: Proceeding of the IADIS International Conference e-Society, Munique, Alemanha, p. 149.
(Missiroli, Russo & Ciancarini, 2017)	Missiroli, M.; Russo, D.; Ciancarini, P. (2017) Agile for Millennials: A Comparative Study. In: Proceedings of IEEE/ACM 1st International Workshop on *Software* Engineering Curricula for Millennials (SECM), Buenos Aires, Argentina, p. 47-53
(Missiroli, Russo & Ciancarini, 2016)	Missiroli, M.; Russo, D.; Ciancarini, P. (2016) Learning Agile *software* development in high school: an investigation. In: Proceedings of the 38th International Conference on *Software*, Austin, TX, EUA, p. 293-302,
(Rusu, A et al, 2011)	Rusu, A. et al. (2011) Employing *software* maintenance techniques via a tower-defense serious computer game. In: Proceedings of the International Conference on Technologies for E-Learning and Digital Entertainment, Berlim, Alemanha, p. 176-184.
(Rusu et al., 2010)	Rusu, A. et al. (2010) Learning *software* engineering basic concepts using a five-phase game. In: Proceedings of IEEE Frontiers in Education Conference, Washington, DC, EUA, p. 1-6.
(Sarkar & Bell, 2013)	Sarkar, A; Bell, T. (2013) Teaching black-box testing to high school students. In: Proceedings of the 8th Workshop in Primary and Secondary Computing Education, Aarhus, Dinamarca, p. 75-78.
(Serrano & Serrano, 2013)	Serrano, M.; Serrano, M. (2013) Requisitos ao Código: Uma Proposta para o Ensino da Engenharia de *Software* no Ensino Médio. In: Proceedings of the International Requirements Engineering Conference, Rio de Janeiro, Brasil, p. 37-42.
(Starrett, 2007)	Starrett, C. (2007) Teaching UML Modeling Before Programming at the High School Level. In: Proceedings of Seventh IEEE International Conference on Advanced Learning Technologies, Niigata, Japão, p. 713-714.

(Verhoeff, 2006)	Verhoeff, T. (2006) A master class *software* engineering for secondary education. In: Proceeding of the International Conference on Informatics in Secondary Schools-Evolution and Perspectives. Heidelberg, Alemanha, p. 150-158.

Pode-se observar que até 2014 foram divulgadas poucas UIs desde 2002 em que foi encontrado o primeiro relato. Somente nos últimos anos teve um leve aumento, provavelmente também relacionado a tendência de aumento de ensino de computação de forma geral na educação básica (Figura 17).

Figura 17: Quantidade de UIs enfocando no ensino de ES na educação básica publicado por ano

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 ensinadas na UI?

As UIs encontradas ensinam competências relacionadas a diversas áreas de conhecimento de ES de acordo com o *SWEBOK* (BOURQUE; FAIRLEY, 2014). Dentre as mais ensinadas estão as áreas relacionadas as principais fases do processo de *software*: requisitos de *software*, modelagem, construção e teste de *software* (Figura 18). A área de conhecimento mais ensinada é a de teste de *software* abordando teste de unidade, teste funcionais e teste de aceitação. O desenvolvimento de requisitos de *software* é também amplamente ensinado aplicando diversas técnicas de levantamento e análise de requisitos, como *User stories*, diagramas de caso de uso, *storyboards*, especificações de requisitos de *software*, entre outras. A modelagem de *software* também é abordada por diversas

UIs, utilizando diagramas de fluxo, pseudocódigo e diagramas *UML* de classe e de máquinas de estado para visualizar a arquitetura e algoritmos dos sistemas de *software*. UIs voltadas ao desenvolvimento de aplicativos móveis também abordam o 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.

Em relação às fases do processo de *software*, um terço das UIs encontradas se destacam 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 re-engenharia, 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 baseado em blocos, como *Scratch*, que fortemente estimulam e suportem o remix de programas (BRENNAN; RESNICK, 2013). Outra estratégia instrucional é proposta por Rusu et. al (2011), o qual propõe um jogo que ensina os quatro tipos de manutenção de *software*.

Várias UIs encontradas também explicitam o ensino de conceitos de processos de *software*, modelos de ciclo de vida e metodologias de desenvolvimento, que de forma implícita e simplificada já são tipicamente adotados para ensinar a programação. Observa-se que predominam aqui modelos simples de ciclo de vida como cascata e o modelo iterativo adotando principalmente metodologias ágeis, como *Scrum*, *Extreme programming* (XP), *Model Driven Development* (MDD) e *Test-Driven Development* (TDD). Pode ser também observado a adoção do *Programming Design Process* (PDP) relacionado a área de Engenharia de forma geral e não especificamente da ES (Figura 19).

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. Foram encontradas tanto UIs que visam a execução de todas as principais fases do processo de *software*, como também outras que focam somente em algumas fases do processo, levando em consideração restrições práticas em relação a 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.

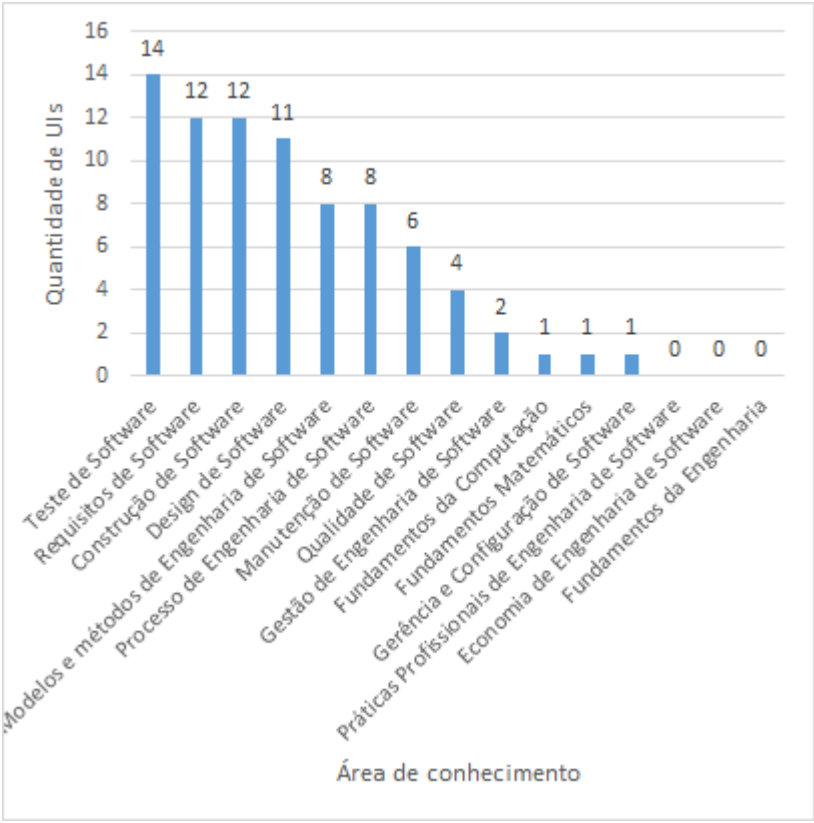


Figura 18: Áreas de conhecimento da ES ensinadas

Pouquíssimas UIs utilizam ferramentas de ES para auxiliar no ensino. Só duas UIs relatam o uso de ferramentas *CASE* (*Rational Rose* e *NetBeans*). Outras ferramentas utilizadas incluem ferramentas de testes (*JUnit*, ferramenta de validação de casos de teste próprio (testBedv9.html)) e uma ferramenta de requisitos *i**.

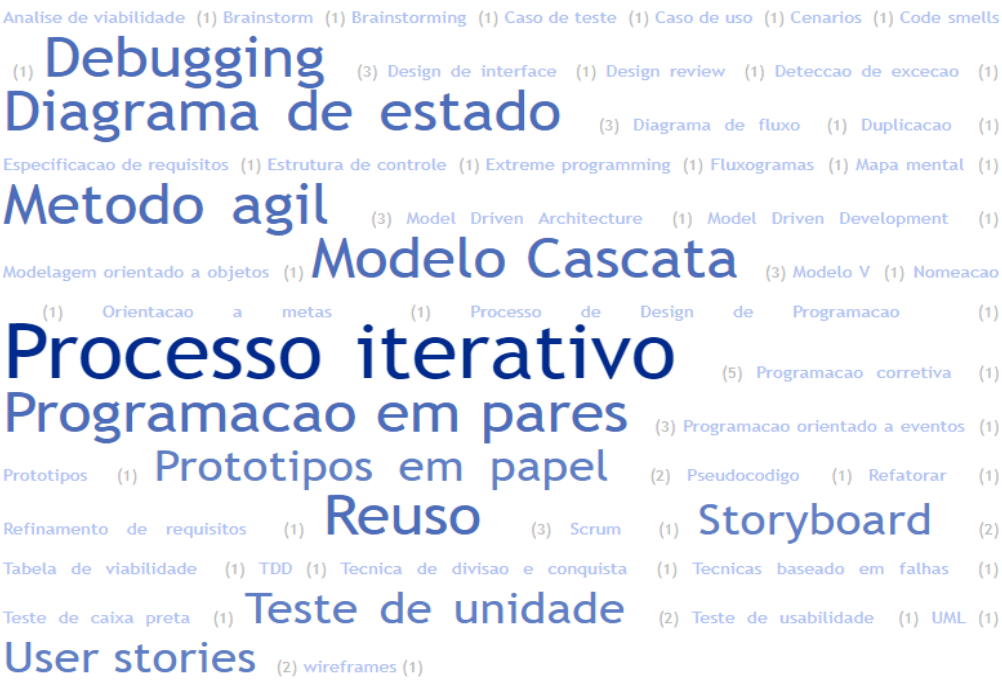


Figura 19: Frequência dos modelos/métodos/técnicas utilizados para ensinar ES na educação básica

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 algoritmos e programação, com enfoque significante na programação de *software*. Tipicamente essas UIs visam o desenvolvimento de animações, aplicações *web*, aplicativos móveis e robôs. Para isso, geralmente se adota ambientes de programação visual baseado em blocos, como *Scratch* e Alice para desenvolver *software desktop*, e *App inventor* para aplicativos móveis. Este tipo de linguagem torna a programação intuitiva e permite que o aprendiz 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*, *Delphi Pascal*, *ANSI C* e a linguagem *R* para computação estatística (Figura 20). Para ensinar o desenvolvimento de programas robóticos foram utilizados *PBASIC*, uma versão do *BASIC* baseado em microcontrolador e a linguagem do *Lego Robot*.

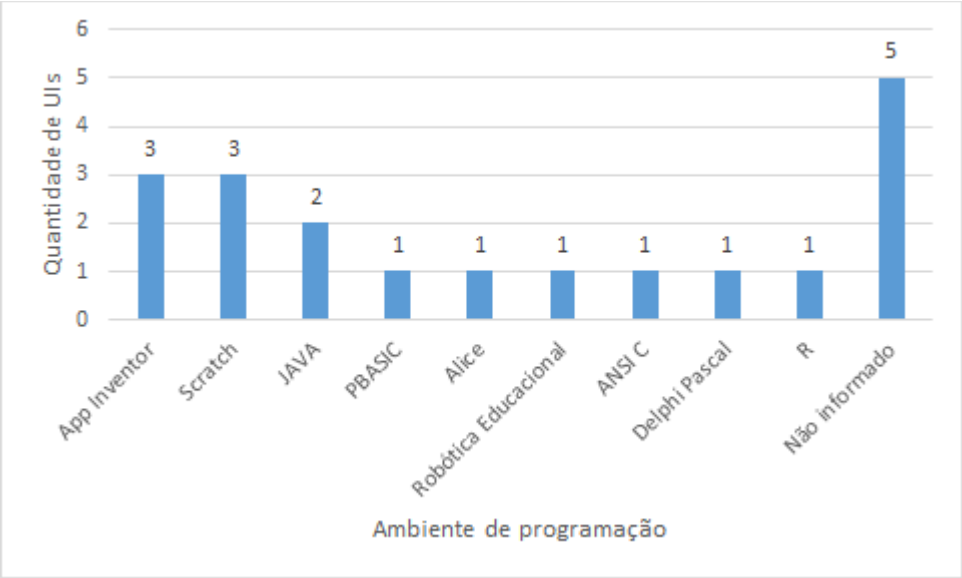


Figura 20: Ambientes de programação utilizados no ensino de ES na educação básica

Poucas UIs 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*. Outro exemplo é a UI apresentada por Sarkar e Bell (2013) visando a criação de casos de testes para um *software* pré-existente.

Em termos de métodos instrucionais observou-se uma forte predominância de abordagens voltadas a aprendizagem ativa (instrução experimental) relacionadas aos objetivos de aprendizagem visando a aprendizagem ao nível de aplicação. Muitas UIs envolvem a realização de um trabalho de desenvolvimento de *software*. Esses trabalhos variam de tarefas bem definidas, com uma especificação do problema a ser resolvida e uma solução esperada, de trabalhos “mal-definidos”, sem uma especificação do problema predefinida e solução previamente conhecida.

Mesmo focando mais na aprendizagem ativa, diversas UIs também incluem outros métodos de instrução direta como aulas expositivas, principalmente na parte inicial da UI (Figura 21). Outros métodos instrucionais indiretos mais utilizados foram a resolução de problemas e exercícios em sala de aula e também as leituras e tarefas de casa. Também foram utilizados métodos interativos, como grupos de estudos, atividades em pares, grupo de aprendizagem cooperativa (fórum das UIs ensinado de modo *on-line*) e discussões/debates. Foram também encontradas duas UIs que adotam aprendizagem baseado em jogos (RUSU et al., 2011; RUSU et al., 2010).

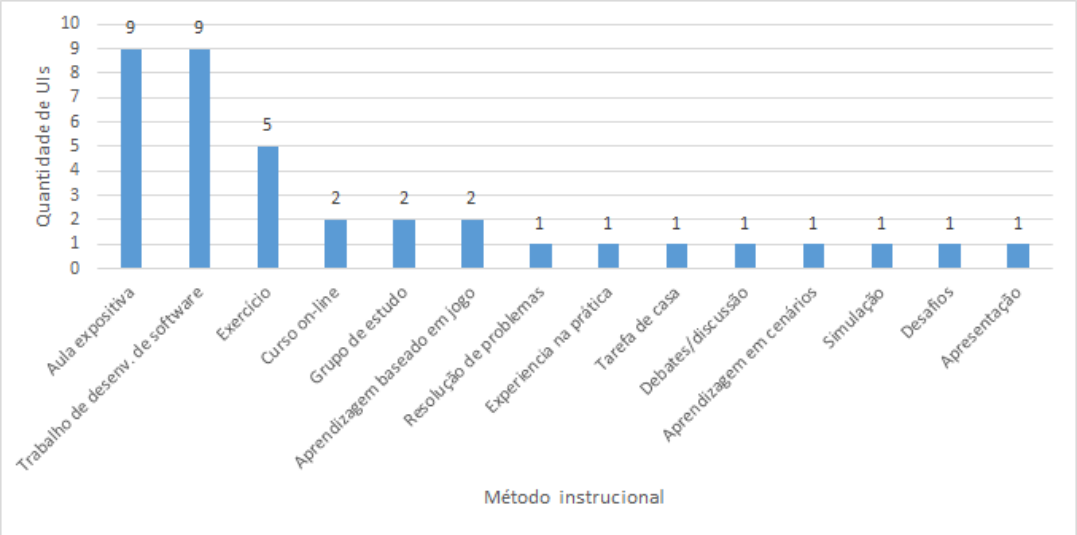


Figura 21: Métodos instrucionais ensino de ES

De acordo com esta variação de métodos instrucionais também são adotados diversos tipos de materiais instrucionais (Figura 22). O material comumente utilizado são artefatos de *softwares*. Estes artefatos 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. Também foram utilizados *softwares* prontos (i) para ensinar a criação de testes de aceitação (SARKAR; BELL, 2013), (ii) para 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 realizados por cursos *on-line*. Várias UIs também usam folhas de exercícios, cadernos de tarefa de casa ou diários para registrar as experiências adquiridas. Porém, de forma geral, observou-se a apresentação de poucas informações em relação aos materiais nas publicações, também em relação a 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), o que pode limitar também uma adoção mais ampla da UI em outros países que tipicamente necessitam de material instrucional na língua nativa nesse nível escolar.

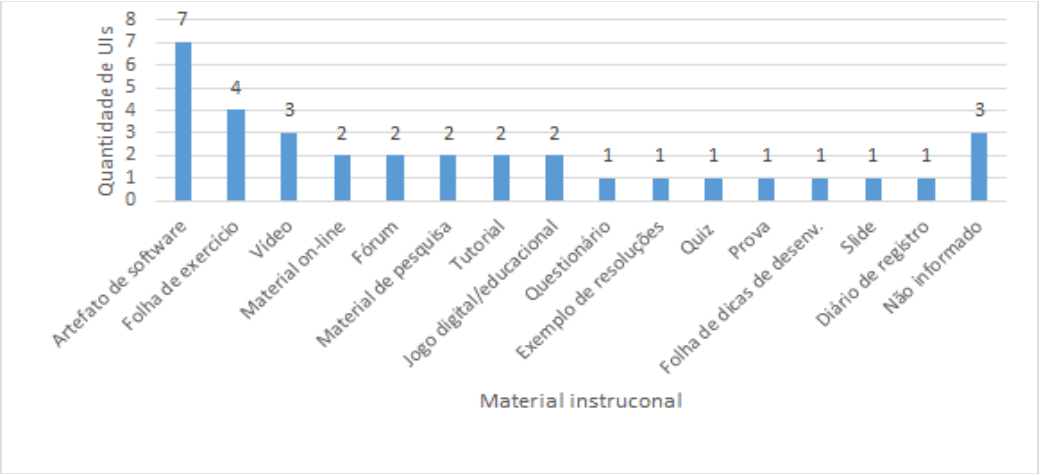


Figura 22: Materiais instrucionais ensino de ES

A aprendizagem dos alunos é avaliada 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*. Em alguns casos também se usou entrevistas em relação aos artefatos criados, provas e/ou projetos revisado por pares (Figura 23).

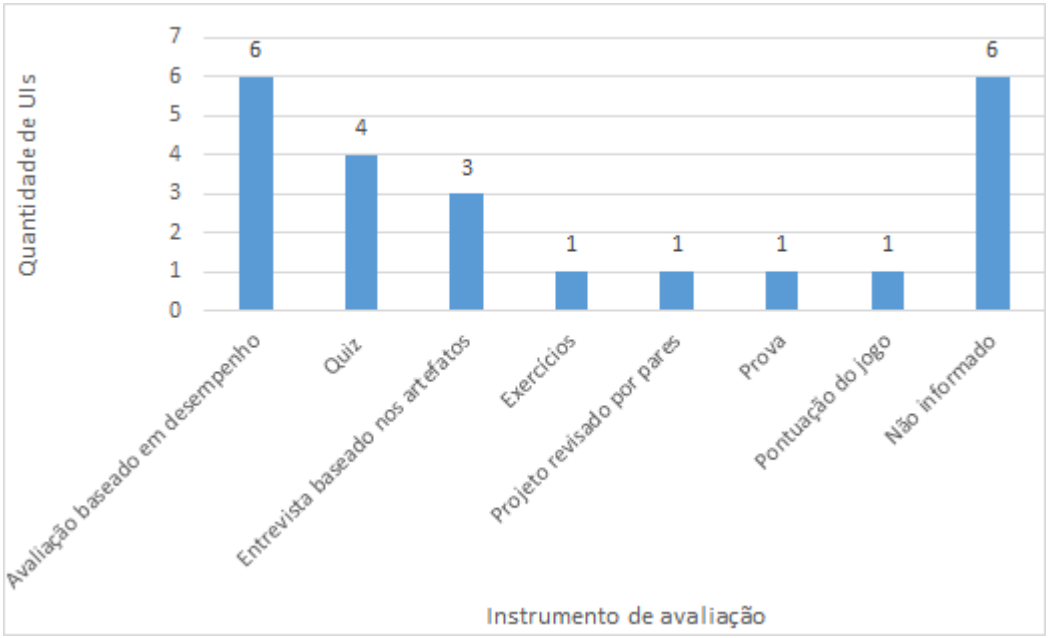


Figura 23: Instrumentos de avaliação ensino de ES

PA4. Quais são as características de contexto da UI?

A maioria das UIs encontradas abordam o ensino de ES (15 UIs) focado no Ensino Médio (Figura 24). Não foi encontrada nenhuma UI de ensino de ES específica para Ensino Fundamental I. Os resultados positivos relatados indicam então que pode ser beneficial ensinar ES não somente na educação superior, mas também já na educação básica. Por outro lado, observando a predominância de UIs encontradas para o Ensino Médio, se questiona qual a razão para a quase inexistência de UIs para o Ensino Fundamental. Mesmo assim, vários autores relatam que a inserção do ensino de ES pode ser benéfica também já no Ensino Fundamental, obviamente de forma limitada levando em consideração o conhecimento prévio dos alunos e as restrições curriculares (BOLLIN; SABITZER, 2015)

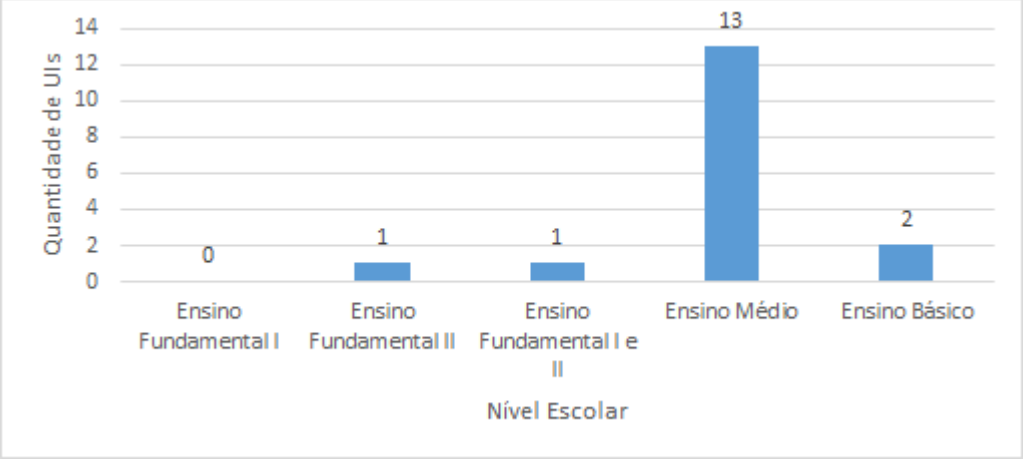


Figura 24: Nível escolar ensino de ES

Refletindo a grande diversidade do formato das UIs, a duração varia de atividades curtas e focadas de somente 30 minutos até cursos de longa duração de um ano. De acordo com o atual desconhecimento dos alunos referente a computação e/ou ES, a maioria das UIs é direcionada a iniciantes sem competências prévias de computação/ES. Somente quatro UIs indicam a necessidade de competências prévias principalmente em relação a programação.

PA5. Como a UI foi desenvolvida?

Para alcançar resultados de aprendizagem eficazes, as IUs precisam ser desenvolvidas sistematicamente seguindo modelos de design instrucional. 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 somente indicando as partes interessadas envolvidos no desenvolvimento da UI, por meio de cooperações entre escolas e/ou universidades envolvendo professores, instrutores e tutores. Somente Köhler et al. (2012) explicitamente relata os princípios instrucionais utilizados no desenvolvimento da UI e *scaffolding*.

PA6. Como a qualidade da UI é avaliada?

Essencial também como parte do processo sistemático de desenvolvimento e melhoria de uma UI, a avaliação é tipicamente realizada por meio de estudos empíricos em sala de aula. Como resultado observa-se que várias UIs foram avaliadas por meio de um estudo de caso (Figura 25). Nestes estudos, a avaliação foi sistematicamente definida e, durante e após o tratamento (ensino de ES), foram coletados dados em relação ao objetivo da avaliação. Apenas dois estudos adotaram um design de pesquisa mais rigoroso. Missiroli et. al (2017) realizaram um 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*. Fronza et. al, (2017) adotam uma abordagem quase-experimental para avaliar a eficácia de um *framework* baseado em métodos ágeis de ES para ensinar computação no ensino médio. Ambos os estudos experimentais seguem a metodologia de Wohlin et al. (2012) para a realização dos experimentos. Observa-se também que um número considerável (8 UIs) foram avaliadas de uma forma menos rigorosa por meio de avaliações *ad-hoc*, sem definição detalhada dos objetivos de avaliação, a medição e análise dos dados. Como resultado esses estudos tipicamente somente comentam o *feedback* informal dos alunos e/ou observações durante a aplicação.

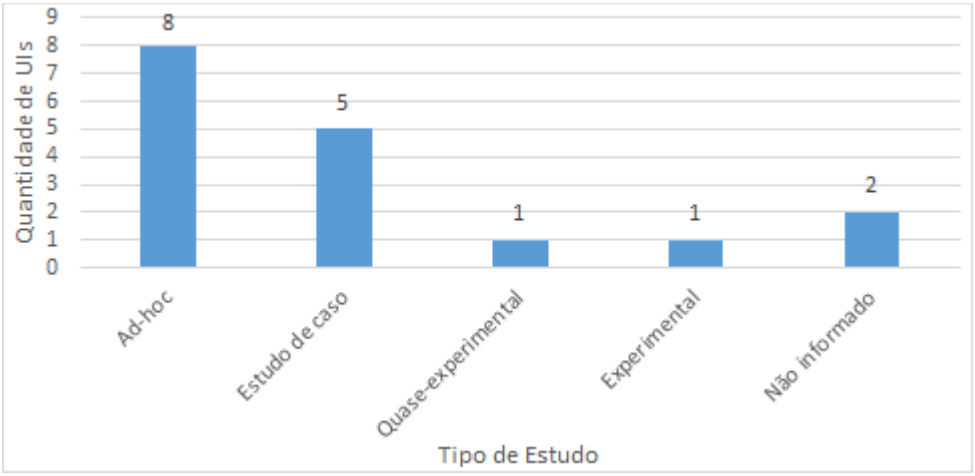


Figura 25: Tipos de estudo ensino de ES

A maioria dos estudos avalia mais de um fator de qualidade (Figura 26). A aprendizagem é o fator de qualidade mais avaliado. Isso demonstra que, de fato, a principal preocupação é o efeito de aprendizagem proporcionado pelo ensino. 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 a uma definição sistemática de níveis de aprendizagem, por exemplo, com base na taxonomia de Bloom. Vários estudos também avaliam o grau da satisfação para avaliar se os alunos sentem que o esforço dedicado resulta em aprendizado. Analisando os fatores avaliados identifique-se também a falta de uma conformidade entre os estudos. Com exceção da avaliação do grau da aprendizagem do aluno avaliado na maioria dos estudos, os fatores analisados variam muito indicando a falta de um modelo de avaliação para este tipo de UI.

.

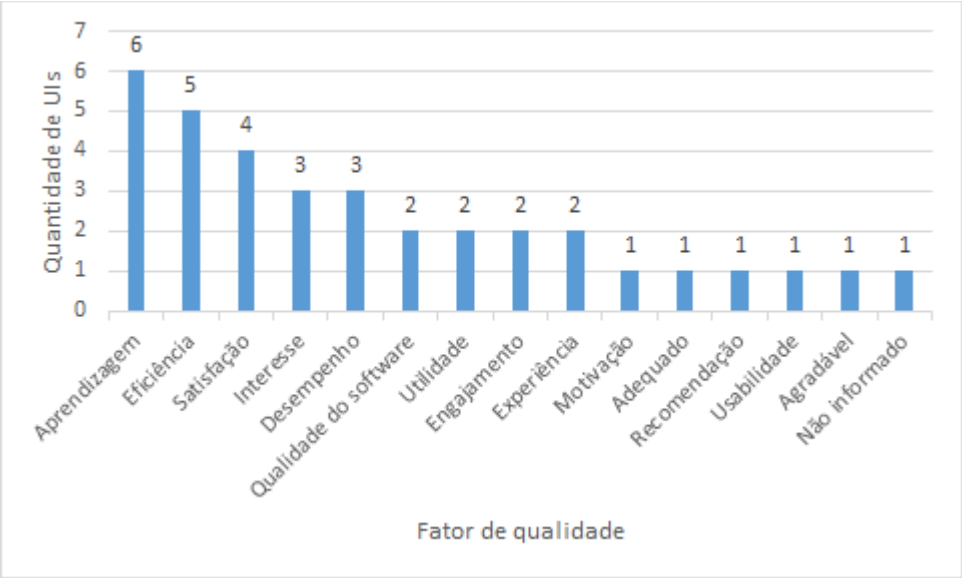


Figura 26: Fatores de qualidade ensino de ES

Dados referentes a avaliação são coletados de diversas formas (Figura 27). A maioria dos dados é coletado via questionários. Vários estudos também extraem dados da avaliação baseados no desempenho dos artefatos criados pelos alunos durante a UI e/ou via observações durante as UI.

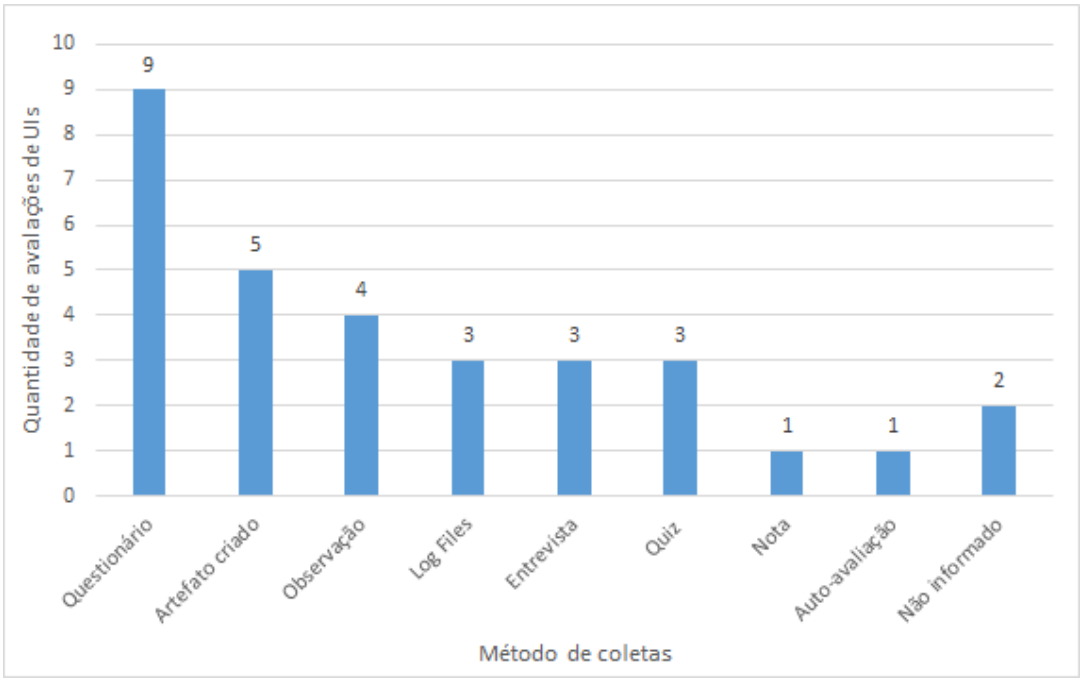


Figura 27: Métodos de coleta ensino de ES

A observação foi utilizada tipicamente para analisar o desempenho do aluno, mas também para avaliar fatores de diversão e satisfação. 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 28). Foram encontrados somente dois estudos realizando testes de estatísticos (HERMANS; AIVALOGLOU, 2017; MISSIROLI et al., 2017).

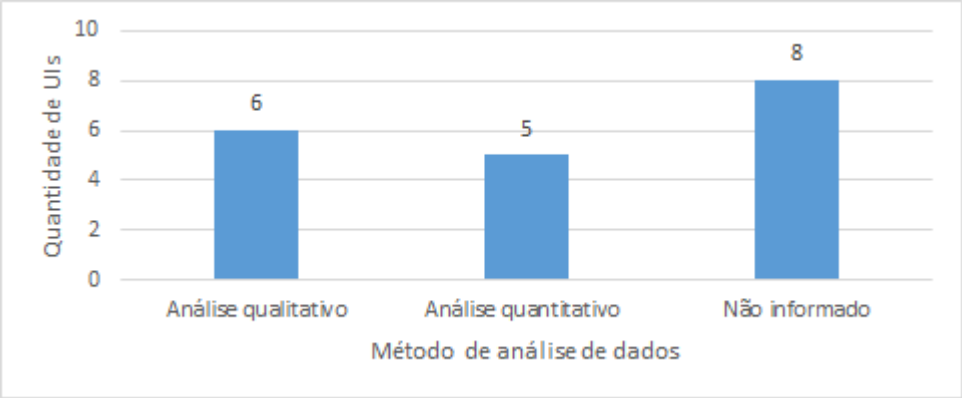


Figura 28: Método de análise de dados ensino de ES

Como mostra a Figura 29, a maioria das avaliações foi realizada com amostras pequenas, variando de 1 a 60 participantes. Esse baixo número de participantes normalmente corresponde ao tamanho de uma classe na qual a UI é aplicada e avaliada. No entanto, vale ressaltar que também encontramos quatro avaliações realizadas com mais de 150 participantes. Uma significativa quantidade de estudos (4 UIs) não informou o tamanho da amostra.

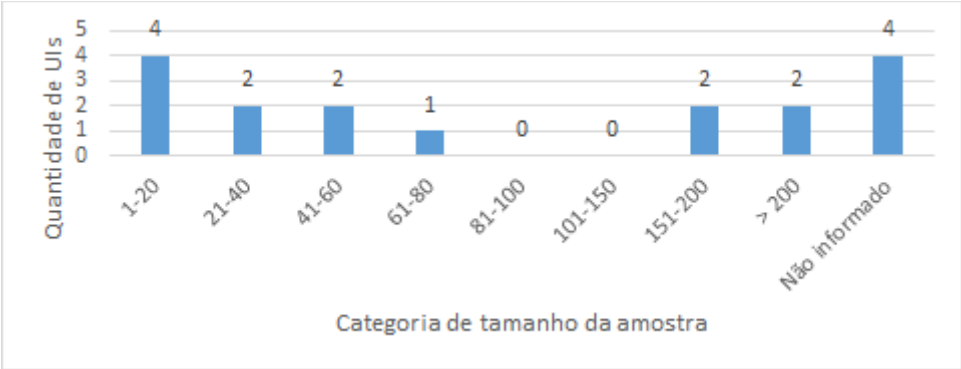


Figura 29: Categoria de tamanho de amostra ensino de ES

Como ponto positivo pode-se observar que quase metade dos estudos foi replicado em mais do que um contexto, contribuindo para a validade externa dos resultados da avaliação (Figura 30). Porém, ainda uma grande parte das replicações ocorreu apenas por meio de um único estudo em um contexto específico, geralmente pelos próprios criadores da UI.

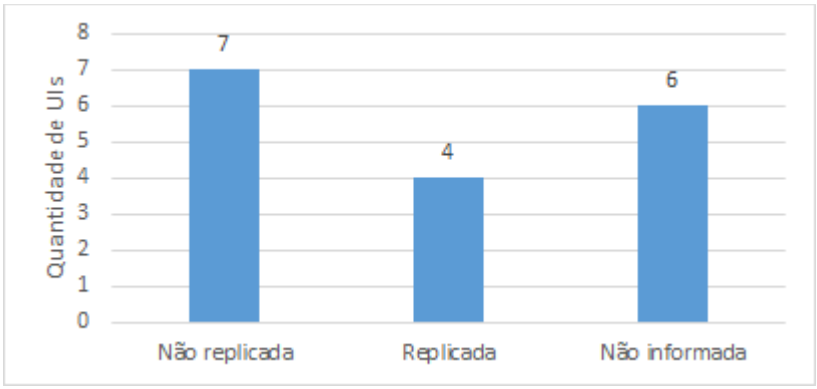


Figura 30: Números de estudos replicados ensino de ES

3.2. Estado da Arte e Prática de Design de Interface

Para elicitar o estado da arte e a prática sobre se e como é ensinado o design de interface na educação básica, realiza-se um estudo de mapeamento sistemático seguindo o procedimento proposto por Petersen, Vakkalanka e Kuzniarz (2015). Este mapeamento foi realizado por meio da produção de um relatório técnico não publicado, realizado pelos mesmos autores da análise anterior. As tabelas contendo uma visão mais detalhada da análise de cada pergunta podem ser visualizadas no Anexo II.

3.2.1. Definição do Mapeamento Sistemático da Literatura

Questão de Pesquisa. Existem (e quais as suas características) unidades instrucionais que ensinam competências de design de interface 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 design de interface são ensinadas na UI?
PA3. Quais as características instrucionais da UI?
PA4. Quais são as características de contexto da UI?
PA5. Como a UI foi desenvolvida?
PA6. Como a qualidade da UI é avaliada?

Critérios de inclusão/exclusão. Consideramos apenas artigos revisados por pares cujo foco é ensinar computação adotando práticas e/ou atividades de design de interface na educação básica, incluindo, por exemplo, também o *UX* design e/ou *design thinking* se voltado ao desenvolvimento de *software*/aplicativos. São excluídos artigos que focam em ensinar computação para graduação ou pós-graduação e/ou artigos que apresentam UIs para o ensino de computação sem abordar conceitos de design de interface do usuário. Incluímos também literatura secundária descoberta por meio das referências da literatura primária encontrada.

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

Fonte de dados. A busca foi feita no *Scopus* (www.scopus.com), *Google Scholar* (https://scholar.google.com.br/), *Science Direct* (https://www.sciencedirect.com) e no Google para encontrar sites de ensino *online* (MOOCs), incluindo *Udemy* (www.udemy.com), *Edx* (www.edx.org), *Khanacademy* (www.khanacademy.org), *Coursera* (www.coursera.org), entre outros. O *Scopus* e o *Scholar* foram utilizados para realizar a busca nas bases das principais editoras científicas. O *Science Direct* também foi incluído como fonte de pesquisa. Alguns sites de ensino *online* foram utilizados para pesquisar UIs ensinadas por meio de cursos a distância.

Definição da string de busca. A *string* de busca é composta de conceitos relacionados à questão de pesquisa, considerando também sinônimos, conforme indicado na Tabela 22. O termo "*user interface design*" foi escolhido por ser o principal conceito a ser pesquisado. O termo "educação básica" é utilizado para restringir o nível de ensino enfocado pela UI. O termo "unidade instrucional" representa o objetivo de identificar e caracterizar diversos tipos de unidades instrucionais.

Tabela 22: Palavras chave referente ao ensino de design de interface

Conceito principal (termo)	Sinônimos
user interface design	UI design, graphic design, visual design, UX , *design thinking*,
computing	coding, programming, *app* inventor, computer science
K-12	*school, kids, children*

A partir destas palavras-chave, foi calibrada a *string* de busca e adaptada de acordo com a sintaxe específica da fonte de dados conforme apresentado na Tabela 23.

Tabela 23: String de busca referente ao ensino de design de interface

Fonte de Dados	String de busca
Scopus	*("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)*
Science Direct	*("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)*
ERIC	*("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")*
Google Scholar	*("design thinking" OR "user interface design") AND ("programming" OR "coding" OR "apps") AND ("K-12" OR "school" OR "kids" OR "children")*
Google	*"design thinking" (computing OR coding OR programming) (kids OR K-12 OR school) teach*
	"user experience" design (computing OR coding OR programming) (kids OR K-12 OR school) teach
	user interface design (computing OR coding OR programming) (kids OR K-12 OR school) teach

Os sites de cursos online foram buscados no Google, utilizando a *search string* acima identificada. As informações sobre o que foi ensinado foram analisadas pelos currículos disponíveis nos sites. Nas buscas no *Google* foram utilizadas mais *strings* pelo fato de que nesta base os resultados são mais abrangentes e diversificados em relação ao tema buscado.

3.2.2. Execução da Busca Referente ao Ensino de Design de Interface

A pesquisa foi executada em maio de 2018 pelo autor em conjunto com outros pesquisadores do GQS/INCoD/INE/UFSC. A busca foi feita em duas etapas. Na primeira etapa a busca foi realizada nas bases especificadas e as informações sobre quantidade de resultados obtidos e possíveis cursos/artigos relevantes foram anotadas (Tabela 32). Na segunda etapa de seleção, foi analisado o texto completo dos artigos e currículos dos cursos pré-selecionados para analisar sua conformidade com os critérios de inclusão/exclusão e o critério de qualidade. Como resultado foram identificados 16 artigos relevantes. Todo este processo foi feito pelos pesquisadores em conjunto, sempre discutindo a seleção até chegar a um consenso. Como houve diversas bases na pesquisa, os dados na Tabela 24 podem estar duplicados, ou seja, a soma das linhas não necessariamente é o número final encontrado.

Tabela 24: Quantidade de artigos por etapa de seleção por repositório referente ao ensino de design de interface

Fonte de dados	**Resultado da pesquisa inicial**	**Resultados analisados**	**Seleção depois do 1° estágio**	**Seleção depois do 2° 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	DT = 1.160.000 UX = 8.770.000 Interface design = 7.150.000	DT = 200 UX = 150 ID = 200	DT = 25 UX = 6 ID = 19	9

3.2.3. Análise dos dados referente a ensino de design de interface

Para responder à questão de pesquisa, extraímos informações relevantes às perguntas de análise conforme especificado na Tabela 25.

Tabela 25: Especificação das informações extraídas referente ao ensino de design de interface

Pergunta de análise	**Dados a extrair**	**Descrição**	**Taxonomia**
PA1. Quais UIs existem?	Nome	O nome ou o autor da UI	--
	Referência	Referência bibliográfica	--
PA2. Qual(is) competências de UI design são ensinados na UI?	Objetivo(s) de aprendizagem referente a **UI *design***	Identificação do(s) objetivo(s) descrevendo o que o aluno deve aprender em relação a **UI *design***	--

	Áreas de conhecimento de **UI *design***	Áreas de conhecimentos da **UI *design*** abordados na UI	*Design thinking*, *UX*, *design* interface de usuário/gráfico/visual
	Métodos/técnicas de **UI *design***	Métodos/técnicas de **UI *design*** abordados na UI	*Design* iterativo, *brainstorming*, prototipação, testes com usuários, persona
	Ferramentas de **UI *design***	Ferramentas de *software* adotadas no ensino de **UI *design***	--
PA3. Quais as características instrucionais da UI?	Objetivo de aprendizagem da UI como um todo	Identificação dos objetivos de aprendizagem em geral da UI	--
	Descrição geral	Breve descrição geral da UI apresentando as suas principais características	--
	Modo de ensino	Identificação do modo do ensino (presencial ou *online*)	Online ou Presencial
	Ambiente de programação	Ambiente/linguagem de programação utilizado na UI	*Scratch*, *App Inventor*, *Snap!*....
	Método Instrucional	Métodos instrucionais utilizados na a UI	...
	Material Instrucional	Material(is) instrucional(is) utilizados na UI	*Slides*, folhas de exercícios, livros, rubricas, guias, *worksheet*, jogos entre outros
	Recursos instrucionais	Recurso(s) instrucional(is) utilizados na UI	Computador, dispositivo móveis e outros
	Método/instrumento de avaliação	Método(s)/instrumento(s) utilizado(s) para a avaliação de aprendizagem do aluno utilizados na UI	Questionários, *performance-based*, testes, *survey*, exercícios, entrevista, observações.
	Língua	A(s) língua(s) em qual a UI está disponível	Inglês, Português, Espanhol, etc.
	Licença	Licença de uso da UI	Gratuita (Creative Com mons CC-BY-NC-SA 4.0 InternationalLicense), etc.
	Capacitação dos instrutores	Indicação se a UI está sendo acompanhado por uma capacitação dos instrutores	Sim ou não
PA4. Quais são as características de contexto da UI?	Nível escolar	Nível escolar para qual a UI é projetada	...
	Duração da UI	Duração da UI (número de hora/aula)	--
	Pré-requisitos	Pré-requisitos em relação a competências da computação prévias do aluno	Sim ou não.
PA5. Como a UI foi desenvolvida?	Método de desenvolvimento da UI	Indicação do método adotado para o desenvolvimento da UI	ADDIE, ISD, etc.
PA6. Como a qualidade da UI é avaliada?	Tipo de estudo de avaliação da UI	Indicação do tipo de estudo (research design) da avaliação da UI	Experimental, Semi-experimental, Não-experimental ou ad-hoc
	Fatores avaliados	Indicação dos fatores que foram avaliados	Aprendizado, engajamento, motivação, qualidade da UI: divertida, usabilidade, facilidade de aprendizagem
	Método de coleta de dados	Indicação do(s) método(s) de coleta de dados adotado(s) na avaliação da UI	Questionários, *performance-based*, testes, *survey*, exercícios, entrevista, observações
	Tamanho de amostra	Quantidade de pontos de dados utilizadas na a avaliação da UI	--

	Avaliações replicadas	Indicação de possíveis replicações da avaliação em vários contextos	--
	Método de análise de dados	Indicação do(s) método(s) de análise de dados utilizado(s) na avaliação da UI	- Qualitativo - Quantitativo
	Descobertas	Descrição dos principais resultados, pontos positivos e negativos identificadas na avaliação da UI	--

Os artigos foram lidos de forma completa e os dados foram extraídos pelos autores. A extração dos dados foi dificultada em vários casos pela 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 suficientes sobre a UI. Nestes casos, algumas informações foram inferidas a partir do artigo, incluindo, por exemplo, a descrição dos objetivos de aprendizagem, idioma da UI, necessidade de competências prévias de computação. Observamos também que a maioria dos estudos não explicita a maneira como as UIs foram desenvolvidas, além de falta de informações relevantes em relação a sua avaliação, por exemplo não abordando ameaças a validade. Nos casos em que o artigo não apresenta nenhuma informação a ser extraída, indicamos a falta desta informação como não informado (NI).

PA1. Quais UIs existem?

Como resultado da pesquisa foram identificadas no total 16 unidades instrucionais voltadas ao ensino de computação que de alguma forma abordam também o ensino de design de interface e *design thinking* a nível da educação básica (Tabela 26).

Tabela 26: Artigos descobertos na pesquisa no ensino de design de interface

Citação	Referência bibliográfica
(CHEN, P.; HUANG, R., 2017)	CHEN, Peng; HUANG, Ronghuai. Design thinking in *App inventor* game design and development: A case study. In: **Advanced Learning Technologies (ICALT), 2017 IEEE 17th International Conference on**. IEEE, 2017. p. 139-141.
(SULLIVAN, J. F. et. al., 2003)	SULLIVAN, J. F., REAMON D., LOUIE B (2003) Girls embrace technology: a summer internship for high school girls. In: **FIE 2003 33rd Annual Frontiers in Education**, Westminster, CO, USA
(KE, F.; IM, T., 2014)	KE, F.; IM, T. (2014) 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
(ROBINSON, A.; PÉREZ-QUIÑONES M. A., 2014)	ROBINSON, Ashley; PÉREZ-QUIÑONES, Manuel A. Underrepresented middle school girls: on the path to computer science through paper prototyping. In: **Proceedings of the 45th ACM technical symposium on Computer science education**. ACM, 2014. p. 97-102.
(VAN WART, S. et al., 2014)	VAN WART, S.; VAKIL, S.; PARIKH T. S. (2014) *Apps* for Social Justice: Motivating Computer Science Learning with Design and Real-World Problem Solving. In: **ITiCSE '14 Proceedings of the 2014 conference on Innovation & technology in computer science education**, p. 123-128.
(DENNER, J. et al., 2005)	DENNER, Jill et al. 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, 2005.
(EDUTOPIA, 2015)	https://www.edutopia.org/blog/coding-by-design-first-approach-douglas-kiang
(CODELIKEAGIRL, 2017)	https://code.likeagirl.io/learning-design-by-making-games-in-scratch-6b90cd9e8a83
(TEKKIE UNI, 2018)	Build Your First *App*: https://tekkieuni.com/courses/build-your-first-app/
(CREATELAB, 2017)	http://www.mycreatelab.com/ais/
(CODE, 2018)	https://curriculum.code.org/csd-1718/unit4/
(ROBINSON, A.; PÉREZ-QUINONES, M. A.; SCALES, G., 2015)	ROBINSON, Ashley; PÉREZ-QUINONES, Manuel A.; SCALES, Glenda. Understanding the attitudes of African American middle school girls toward computer science. In: **Research in Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT), 2015**. IEEE, 2015. p. 1-8.
(TECHNOVATION, 2018)	Technovation Challenge: http://www.technovationbrasil.org/curriculo
(CODEHS,2018)	https://codehs.com/info/curriculum
(CODE.ORG/APP LAB, 2018)	https://code.org/educate/applab
(GET STARTED WITH CODE 2, 2017)	https://itunes.apple.com/us/book/get-started-with-code-2/id1226776857?mt=11

Pode-se observar que até antes de 2017 foram divulgadas poucas UIs desde 2004, ano em que foi encontrado o primeiro relato. Somente nos últimos anos teve um leve aumento, provavelmente também relacionado a tendência de aumento de ensino de computação de forma geral na educação básica (Figura 31).

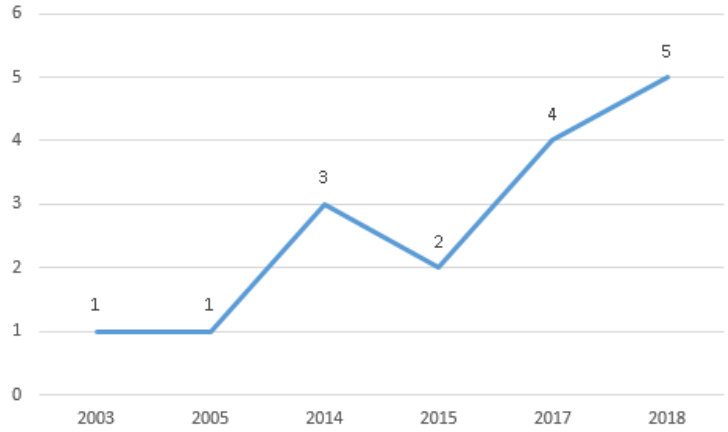


Figura 31: Quantidade de UIs enfocando no ensino de design de interface na educação básica publicado por ano

No entanto, 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 design de interface no nível da educação básica.

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

Os objetivos de aprendizagem em todas as UIs se basearam na aplicação dos conceitos de ensino especificados pelos autores. Com isso, além de apenas o conhecimento dos conceitos, as unidades também fizeram os alunos os aplicarem na prática, tendo assim um nível maior de aprendizagem sobre os assuntos.

As UIs encontradas mencionaram o ensino de diversas competências relacionadas ao design de interface. Dentre as mais ensinadas estão *design thinking* e o próprio design de interface do usuário. Outras competências também mencionadas foram design visual, usabilidade, interação humano-computador, processo de design, *participatory design*, design centrado no usuário e *UX design* (Figura 32). Para o ensino de design de interface do usuário foi amplamente utilizada a técnica de prototipação. A prototipação foi realizada principalmente em papel, mas também em meio digital, em alguns casos até os dois. Outra técnica muito utilizada foi a de *brainstorming*. Com ela os alunos puderam trocar ideias sobre a melhor forma de resolver um problema encontrado. Chamaram atenção também as técnicas de testes com usuários e design iterativo, sendo ensinadas em algumas UIs (Figura 33).

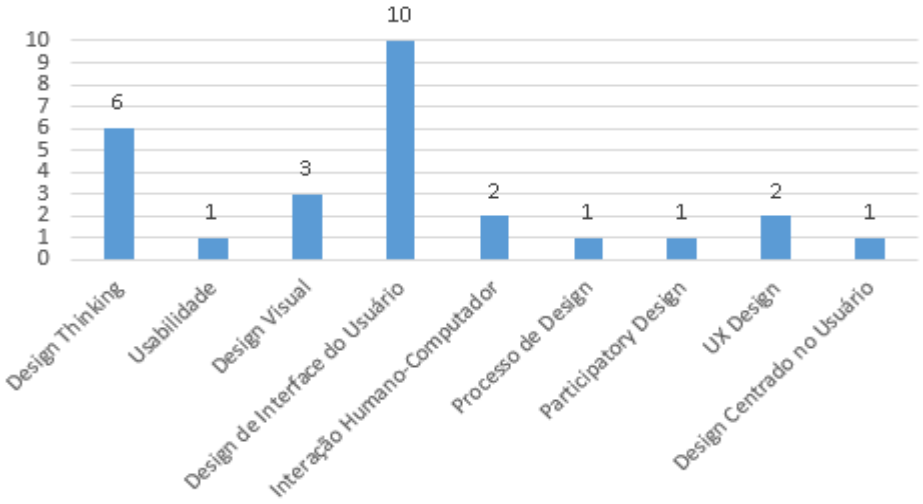


Figura 32: Conceitos ensinados referentes a design de interface

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

Outra competência muito ensinada pelas UIs analisadas foi o *design thinking*. Para isto, a principal técnica utilizada foi a prototipação, da mesma forma que foi mencionado acima. Isso mostra que realmente esta técnica é amplamente utilizada e que crianças se sentem confortáveis em utilizá-la. Em alguns casos, as técnicas de entrevista com o usuário final, *brainstorming* e diagramas de fluxo também foram utilizadas.

Diversas ferramentas de design de interface foram utilizadas, mas a maioria das UIs não utilizaram ferramentas específicas para o design da interface. Todas as ferramentas utilizadas foram diferentes, são elas: *Lingo*, *Adobe Photoshop*, *Macromedia Director*, *POP app*, Editor *online* do *CodeHS*, *Balsamiq*, *Invision app*,

Pages e *Mini Monet*. A maioria destas ferramentas citadas são especificamente para design de interface e a produção de mapas de navegação/*workflows*. Apenas o *Photoshop*, o *Director*, *Pages* e *Mini Monet* são ferramentas para produção de imagens em geral.

PA3. Quais as características instrucionais da UI?

O ensino de competências de design de interface está voltado, de forma geral, ao ensino do processo de *design thinking* e do design de interface gráfica, com enfoque significante no desenvolvimento de jogos e *apps*. No ensino de *Design Thinking* as UIs geralmente estimulam os alunos a desenvolver um *software* de forma útil e criativa visando solucionar problemas da comunidade no contexto cotidiano. Já o ensino de design da interface do usuário é abordado principalmente o desenvolvimento de jogos e animações gráficas.

Para o desenvolvimento desses *softwares* geralmente são adotados ambientes com paradigma de programação visual baseado em blocos, como *Scratch*, *Alice* e G*oogle Blockly*, para *softwares desktops*. Já para *apps* são utilizados *Tynker*, *App inventor* e *App Lab*. Este tipo de paradigma torna a programação intuitiva e permite que o aprendiz se concentre na lógica de programação ao invés da sintaxe da linguagem de programação (Pokress & Veiga, 2013). 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* (Figura 34). Outras ferramentas também foram utilizadas para o ensino de design da interface do usuário com criação de animações e/ou jogos como *Adobe Flash* (Figura 34).

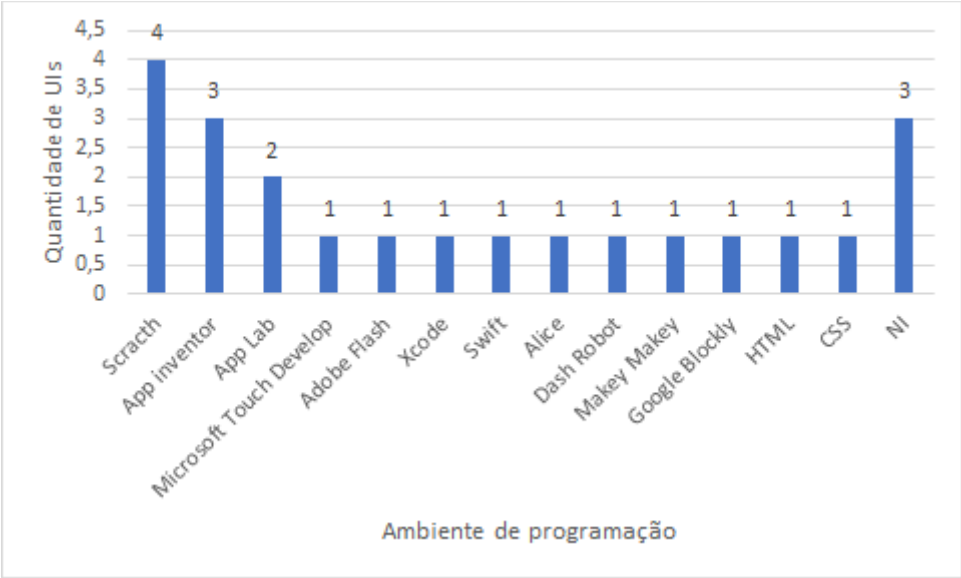


Figura 34: Ambientes de programação ensino de design de interface

Diversas UIs que ensinam principalmente design da interface do usuário tem o seu programa restrito apenas para meninas pelo fato de ter poucas mulheres trabalhando na área de computação (TECHNOVATION, 2018). As mulheres muitas vezes são motivadas a buscar a computação por causa da percepção de que a computação oferece oportunidades de ser criativa, comunicativa e aborda questões da sociedade (ROBINSON; PÉREZ-QUIÑONES, 2014). A disciplina pertinente a interação humano-computador contém várias características nas quais as mulheres podem se relacionar, por exemplo, problemas voltado às pessoas, ser interdisciplinar, criativa e colaborativa (ROBINSON; PÉREZ-QUIÑONES, 2014).

Essas UIs ensinam habilidades técnicas em design gráfico, desenvolvimento de interface com o usuário, programação visual, manipulação de imagens digitais, criação de multimídia e testes com usuários. Com isso, essas UIs obtiveram resultados positivo no que se refere a fazer com que as alunas desenvolvam uma visão positiva em relação à criação e uso de tecnologia, desenvolvimento de habilidades técnicas e maior conscientização sobre oportunidades da carreira de TI.

Em termos de métodos instrucionais observou-se uma forte predominância de abordagens voltadas a aprendizagem ativa por meio de métodos de ensino com grupos de aprendizagem cooperativa e experiência na prática (Figura 35). 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 utilizada 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 DT, a maioria desses trabalhos não teve um problema e uma solução previamente conhecida no desenvolvimento do *software* estimulando os alunos a pensarem em algum problema de um determinado contexto e também gerar sua solução.

Mesmo focando mais na aprendizagem ativa, diversas UIs também incluem outros métodos de instrução direta como aulas expositivas, vídeo aulas, demonstrações, ensino explícito, principalmente na parte inicial da UI. Outros métodos instrucionais indiretos mais utilizados são investigação e exercícios em sala de aula. Outros métodos interativos são utilizados, como *brainstorming*, desafios, entrevista, trabalho em grupos e discussões. O ensino por meio de métodos de aprendizagem experimental também foram aplicadas por meio de simulações, interpretação de papéis e condução de experimentos.

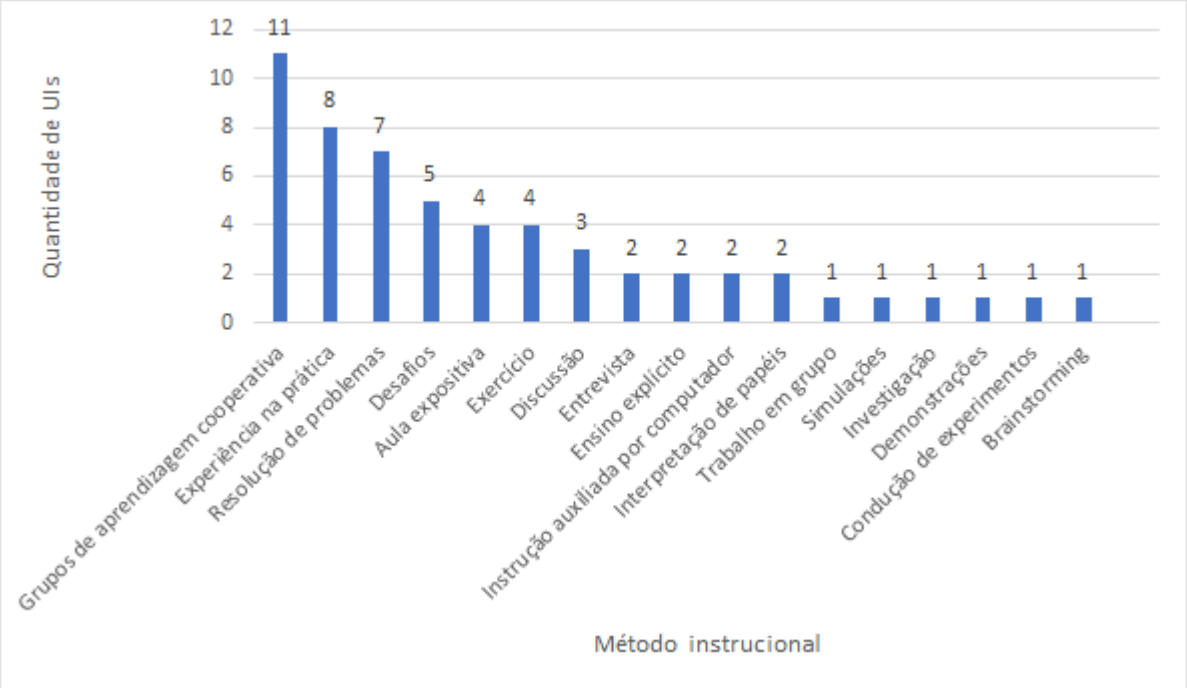


Figura 35: Métodos instrucionais ensino de design de interface

De acordo com esta variação de métodos instrucionais também são adotados diversos tipos de materiais instrucionais. Os materiais comumente utilizado são

papéis, lápis, marcadores, etc. utilizados para construir protótipos de papel ou *wireframes* (Figura 36). Também são utilizados *post-its* para diversos fins, como definir *fluxogramas*, registrar as idéias e requisitos de *software*. Os *post-its* tornam as informações mais visuais, fica fácil criar categorias e fazer novas associações. Artefatos de *softwares* são utilizados para auxiliar no desenvolvimento do *software* por meio de amostras de código. Guia de diretriz de interface humana também foi utilizada no desenvolvimento de interface para *apps*. O guia fornece detalhes sobre o que torna os aplicativos fáceis e intuitivos de usar (EDUTOPIA, 2015). Outros materiais de ensino também foram utilizados como *slides*, vídeos, materiais eletrônicos, folhas modelo, entre outros. Porém, de forma geral, observou-se a apresentação de poucas informações em relação aos materiais nas publicações, também em relação a 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), o que pode limitar também uma adoção mais ampla da UI em outros países que tipicamente necessitam de material instrucional na língua nativa nesse nível escolar.

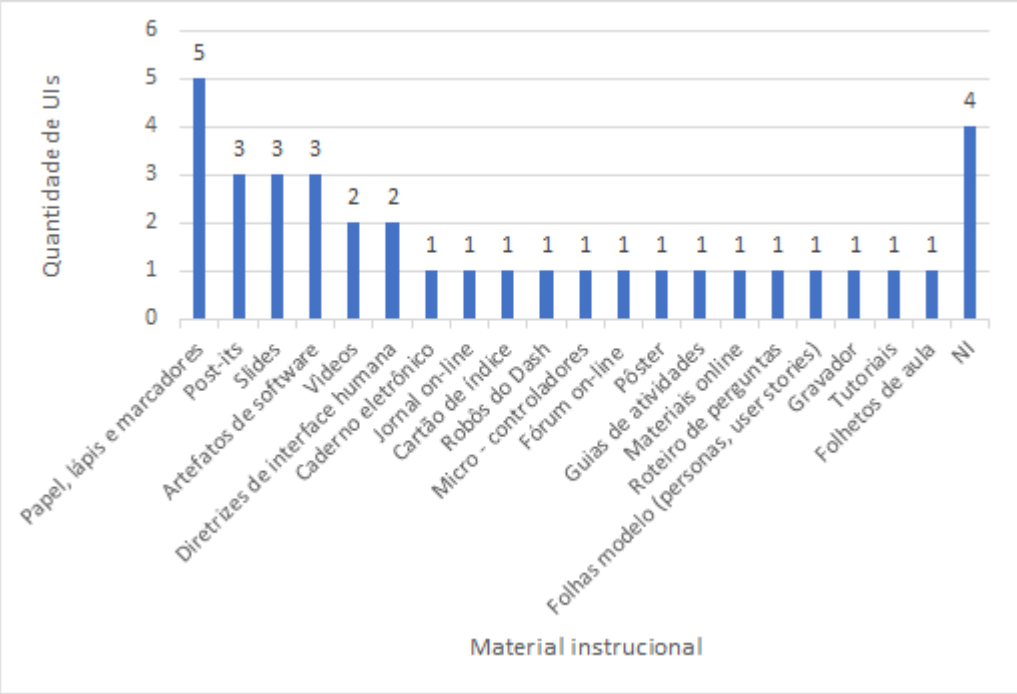


Figura 36: Material instrucional ensino de design de interface

Foi possível identificar alguns recursos instrucionais na execução operacional das atividades (Figura 37). Os dispositivos móveis foram amplamente utilizados, como *iPads*, *notebooks*, *laptops*, *tablets* e *smartphones*. Os *iPads* tipicamente são utilizados para criar protótipo digital da interface de usuário.

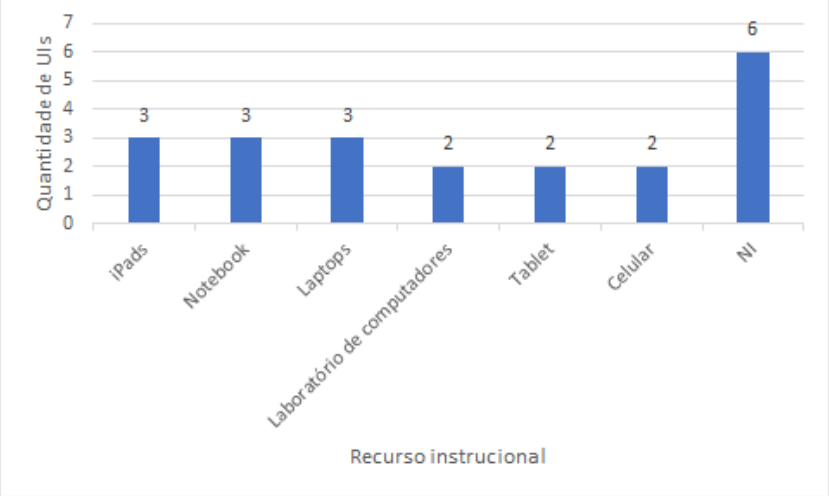


Figura 37: Recursos Instrucionais ensino de design de interface

A aprendizagem dos alunos é avaliada principalmente por meio de avaliações baseadas no desempenho, analisando artefatos criados no contexto do ensino de design de interface. A avaliação sobre os artefatos é geralmente realizada pela análise da interface de um *software* funcional como jogo, *website*, *app*. Em alguns casos também se usaram gravações, entrevistas, observações e apresentações em relação aos artefatos criados. Outras formas de avaliação também foram utilizadas, como questionários, exercícios e questionários realizados durante o curso (Figura 38).

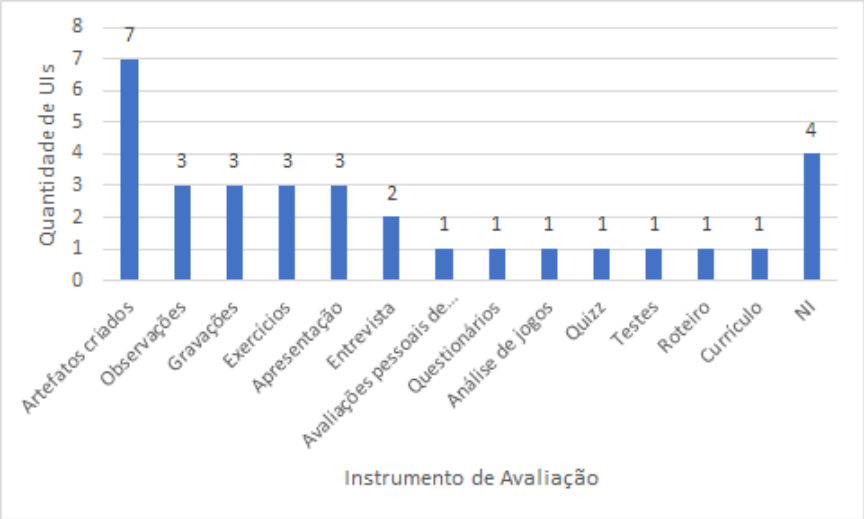


Figura 38: Instrumentos de avaliação ensino de design de inteface

PA4. Quais são as características de contexto da UI?

A maioria das UIs encontradas abordam o ensino de design de interface para crianças do nível de Ensino Fundamental, tanto para EF I quanto EF II (Figura 39). Destas unidades para o Ensino Fundamental, duas delas ensinaram para ambos os níveis escolares. Para o Ensino Médio foram aplicadas no total 8 UIs, o que demonstra um certo equilíbrio entre os níveis escolares alvo. Também houve unidades que ensinaram tanto para EF quanto para EM, somando um total de 4 UIs. Estes dados demonstram que realmente o Ensino Fundamental é o nível de ensino mais procurado para o ensino de conceitos de design de interface e que realmente é possível já inserir esses conteúdos não só em nível superior.

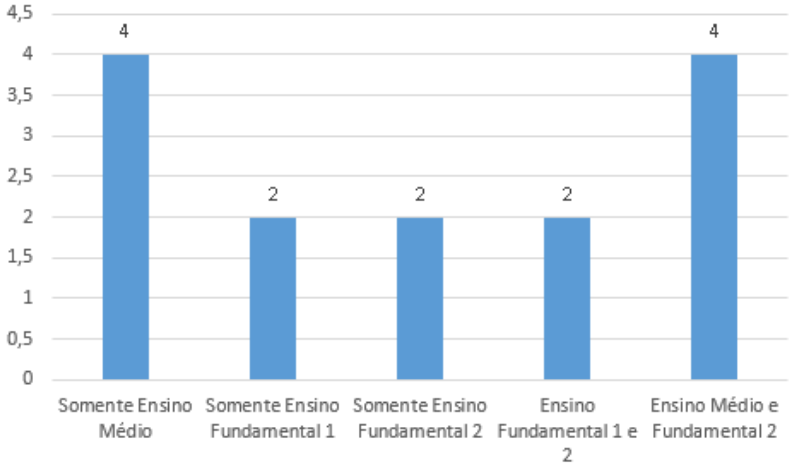


Figura 39: Níveis de ensino das UIs encontradas ensino de design de interface

Houve uma grande diversidade de formatos de aplicação das UIs encontradas, variando de *workshops* de poucas horas até cursos mais longos de 1 ano. Em geral, as UIs que se baseiam no ensino não presencial possuem uma duração mais prolongada, assim os alunos podem aprender com mais calma por si mesmos os conteúdos. Já os presenciais são tipicamente mais curtos, já que com um professor presente o ensino se dá de maneira mais direta e as dúvidas são solucionadas instantaneamente.

Competências prévias não foram requisitadas pela grande parte das UIs, apenas uma delas cobrou que seus alunos já possuíssem algum conhecimento prévio (CODELIKEAGIRL, 2017). As outras unidades encontradas afirmaram que os alunos não necessitavam conhecimento prévio (ou foi implicitamente observado) ou somente não comentaram sobre, o que indica não afetar no decorrer da sua aplicação. Estes dados mostram que crianças com até mesmo nenhum conhecimento na área podem aprender sobre programação e design de interface sem nenhuma complicação já a nível de Ensino Fundamental.

PA5. Como a UI foi desenvolvida?

Para que haja eficácia de aprendizagem as UIs precisam ser sistemáticas, orientadas por modelos de design instrucional. Entretanto, foi possível observar uma ausência generalizada de informações nos artigos pesquisados em relação à forma como as UIs foram desenvolvidas. Poucas unidades apresentaram informações em

relação a esta questão. Os registros neste contexto, geralmente baseiam-se na indicação das partes interessadas envolvidos no desenvolvimento da UI, por meio de cooperações entre escolas e/ou universidades envolvendo professores, instrutores e tutores.

PA6. Como a qualidade da UI é avaliada?

A avaliação é tipicamente realizada por meio de estudos empíricos em sala de aula como parte do processo sistemático de desenvolvimento e melhoria de uma UI. Nesse contexto, observa-se que várias UIs foram avaliadas por meio de um estudo de caso (Figura 40). Apenas um estudo adotou um design de pesquisa mais rigoroso (SULLIVAN et. al. 2003). Este estudo dedicou esforços em desenvolver atitudes positivas em relação à criação e uso de tecnologia, habilidades técnicas e maior conscientização sobre oportunidades da carreira de TI.

Observa-se também que alguns estudos (2 UIs) foram avaliadas de uma forma menos rigorosa por meio de avaliações *ad-hoc*, sem definição detalhada sobre os objetivos de avaliação, a medição e análise dos dados. No entanto, um número considerável de estudos (7 UIs) não registraram informações ou observações nesse sentido. Como resultado esses estudos tipicamente somente comentam o *feedback* informal dos alunos e/ou observações durante a aplicação.

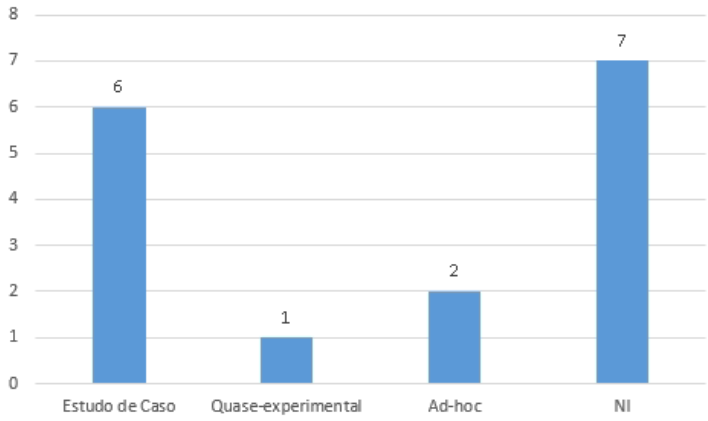


Figura 40: Tipos de estudo ensino de design de interface

A maioria dos estudos avalia mais de um fator de qualidade (Figura 41). Porém, alguns estudos não registraram as informações evidenciadas. Nota-se que,

a aprendizagem aliada a utilidade são os fatores mais citados. O principal objetivo das UIs analisadas é prover ensino de design de interface de forma que seus alunos consigam absorver a aprendizagem. Neste contexto, a avaliação deste fator em geral agrega à melhoria da competência. Pré-teste e pós-teste do ensino são largamente utilizados em estudos ao comparar o nível de competência dos alunos. No entanto, pré-teste e pós-teste foram registrados somente em dois estudos. Com exceção da avaliação do grau da aprendizagem do aluno avaliado na maioria dos estudos, os fatores analisados variam muito indicando a falta de um modelo de avaliação para este tipo de UI.

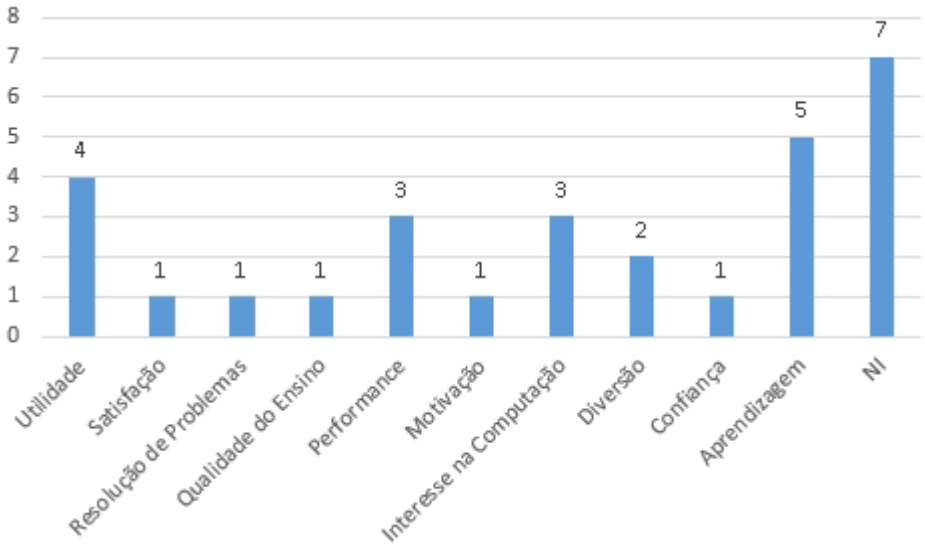


Figura 41: Fatores de qualidade ensino de design de interface

Quanto a coleta de dados, de acordo com a Figura 42, evidenciamos uma carência de registro em um pouco menos da metade (7 UIs). No restante dos estudos foi possível perceber que há variação de métodos referente a avaliação dos dados coletados. A maioria dos dados é coletado via entrevista e questionários. Porém, também foram utilizados como métodos de coleta: observação, testes de habilidades, gravação e alguns artefatos criados exclusivamente para avaliação das UIs.

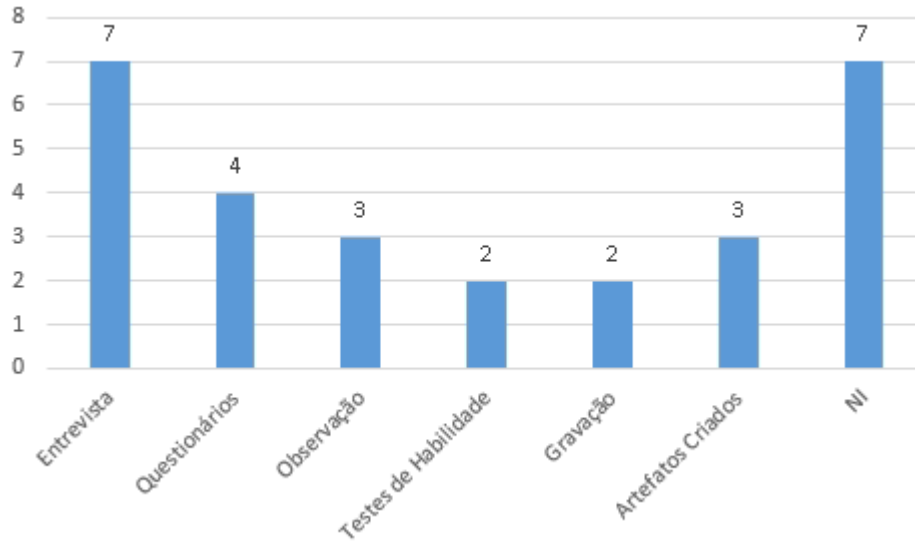


Figura 42: Métodos de coleta ensino de design de interface

A partir do método da observação, além de permitir analisar o desempenho do aluno, também propiciou fatores para avaliar a diversão e satisfação. Os *log files* de fóruns, vídeos e registros de experiências adquiridas foram na maioria utilizados por cursos realizados de modo *on-line* para analisar o engajamento do aluno no curso.

Dos 16 estudos analisados, a metade das UIs são classificadas por realizarem análise qualitativa dos dados e/ou análise qualitativa de forma descritiva, sendo dois deles aplicando ambos os tipos de análise. Destes, somente dois estudos realizaram testes estatísticos (ROBINSON; PÉREZ-QUINONES; SCALES, G., 2015; KE; IM, 2014). A outra metade, devido à adoção de designs de pesquisa menos rigorosos adotados, não há referência no método de análise de dados.

A maioria das avaliações foi realizada com amostras pequenas, variando de 1 a 60 participantes, com destaque para a faixa entre 20 a 40 alunos. Esse baixo número de participantes normalmente corresponde ao tamanho de uma classe na qual a UI é aplicada e avaliada. No entanto, vale ressaltar que uma significativa quantidade de estudos (10 UIs) não informou o tamanho da amostra.

Como característica relevante, podemos perceber que, na maior parte dos estudos analisados, não encontramos registros de réplicas em mais do que um contexto que puderam contribuir para a validade externa dos resultados da avaliação. Em nossa busca dentre os estudos relevantes, foi encontrado uma única

réplica (SULLIVAN et. al., 2003), mas que foi registrada em um pesquisa posterior, em um contexto específico pelos próprios criadores da UI.

3.3. Discussão

Observa-se que, levando em consideração a importância da ES no desenvolvimento de *software*, foi encontrado um número muito pequeno de unidades instrucionais (somente 17 no total) visando o ensino destas competências importantes no nível da educação básica. Da mesma forma ocorre no ensino de design de interface, somente 15 unidades ensinaram competências de design de interface juntamente ao desenvolvimento de *software*. Estes fatos mostram uma carência de ensino neste sentido e que mais iniciativas deveriam existir.

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 tradicionais, 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, se infere que tanto a adoção de modelos de ciclo de vida simples e/ou iterativos pode ser beneficial para introduzir o processo de *software* nesse nível escolar. De uma maneira generalizada, as áreas de conhecimento mais tratadas no ensino de design de interface foram o *design thinking* e o próprio design de interface do usuário. Com isso, pode-se concluir que estas áreas são muito bem aceitas neste nível de ensino e ensiná-las pode ser algo muito benéfico aos estudantes. As técnicas que largamente chamaram atenção foram a prototipação e o design iterativo. O grande uso da prototipação indica a importância de ensino desta técnica e que ela atrai os estudantes para o desenvolvimento passo-a-passo do software. Já a técnica de desenvolvimento iterativo aponta que esta maneira de desenvolvimento, sendo testando, avaliando e redesenhando os produtos e/ou protótipos, traz grandes benefícios para o aprendizado e para o produto final.

Observa-se que a maioria dos UIs se concentram no Ensino Médio, mesmo que alguns autores relatam 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 aplicar uma UI no Ensino Médio, conclui que o ensino de ES pode ser iniciado no Ensino Fundamental sem dificuldades. Porém, é preciso identificar quais áreas de conhecimentos, assunto 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; AIVALOGLOU, 2017). No ensino de design de interface, tanto o Ensino Fundamental quanto o Ensino Médio foram alvo da aplicação das unidades. Isso demonstra que para a aprendizagem dos conceitos relacionados a design de interface não importa muito o nível escolar apresentado pelos alunos, claro que o grau de dificuldade dos conceitos devem ser medido de acordo com o nível escolar, não ultrapassando os limites de educação. Nenhuma UI relatou dificuldades em relação ao ensino dos alunos, bem pelo contrário. Foi relatado que os estudantes conseguiram compreender e aplicar os conceitos ensinados, seja os conceitos de programação e design de interface ou seja os conceitos interdisciplinares envolvidos na aplicação.

Nota-se também que as UIs geralmente ensinam competências de programação juntamente à ES. Bollin & Sabitzer (2015) concluíram que não existe a necessidade dos 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). Exceto a UI apresentada por Starrett (2007) ensina alunos a modelar *software* utilizando a linguagem *UML* antes de ensinar programação. Conforme o autor, a modelagem e 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. As UIs que ensinam o design de interface não requerem que seus estudantes necessitem de conhecimentos prévios, com exceção de uma

(CODELIKEAGIRL, 2017). Por mais que o ensino principal seja sobre os conceitos do design, a programação pode ser um problema para os alunos na hora da construção do *software*. Apesar disto, nenhuma UI relatou algum problema relacionado a programação, mostrando que os alunos neste nível escolar já estão preparados a receber este tipo de conteúdo e os resultados podem ser positivos.

Em geral, são abordados conceitos mais básicos de ES nesse nível escolar mais relacionados ao domínio cognitivo variando também em relação a duração da UI. Deficiências tipicamente observadas em relação ao ensino de ES na educação superior restrito a projetos em pequena escala não apresentando características de projetos reais, 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. O ensino dos conceitos de design de interface foi inteiramente realizado na prática, produzindo e modificando os artefatos criados geralmente a cada iteração do processo de desenvolvimento (nas UIs que utilizaram esta técnica).

Observou-se também uma preferência da adoção de métodos e técnicas ágeis que parecem mais adequados para iniciar o ensino de ES. Interessantemente, 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 (BOURQUE; FAIRLEY, 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. Várias ferramentas de construção de interfaces foram utilizadas, até mesmo aplicativos em *smartphones* ou *tablets* para este objetivo. Na produção de imagens e/ou animações também foram utilizadas ferramentas para auxílio dos estudantes, como por exemplo o *Photoshop*. Desta maneira, é possível concluir que a utilização de ferramentas no ensino de design de interface pode ser muito beneficial, já que auxilia os estudantes e facilita a produção dos artefatos necessários no ensino.

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 a UI. A grande maioria está inserida no contexto de UIs centrado no ensino de programação, poucas focam explicitamente no ensino de conceitos de ES. A integração do ensino de ES em UIs ensinando programação pode ser benéfica tanto em relação à 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, algo que não ocorre no ensino de design de interface. Na maioria dos casos, as UIs optaram por levar os alunos a resolver um problema identificado por eles mesmos. Apenas 3 unidades envolveram conceitos interdisciplinares na produção de um sistema de *software* em um processo de design.

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, aplicativos móveis, aplicação *web* ou robôs). A avaliação do grau de aprendizagem dos alunos é comumente com base no desempenho a partir dos artefatos criados pelos alunos e/ou *quizzes*. Porém, os artigos encontrados não fornecem maiores informações em relação a avaliação da aprendizagem das competências de ES seja, por exemplo, por meio de rubricas, análises automatizadas, etc. De uma maneira geral, o aprendizado do design de interface do usuário foi realizado com a análise dos artefatos criados pelos alunos no desenvolvimento de um programa. Questionários também foram muito utilizados, até mesmo como uma forma de auto avaliação (SULLIVAN et. al., 2003). Por meio de observações sobre os alunos também foram retiradas informações sobre o seu aprendizado.

Visando uma disseminação das UIs apresentadas nos artigos no ensino da ES, se observa uma indisponibilidade de informações detalhadas das UIs e/ou dos materiais instrucionais. O que não ocorre no ensino de design de interface. Grande parte das UIs apresentaram os materiais utilizados, nem sempre em um nível detalhado. A grande maioria das UIs, nos dois casos de ensino, foi criada em somente uma única linguagem e não está acessível, seja gratuitamente ou pago. Essa indisponibilidade das UIs impede a ampliação da sua aplicação.

Relacionado a essa questão, também se observa uma falta em relação a 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 da educação básica com formação na computação, deixando como solução somente a adoção de uma abordagem multidisciplinar em que computação é ensinado por professores formados em outras disciplinas. Em algumas UIs de ensino de design de interface foram utilizados mentores capacitados que auxiliaram na produção e avaliação dos artefatos criados pelos alunos. Este fato faz com que os professores não possuam necessariamente capacitação para o ensino das unidades, sendo fortemente amparados por estes mentores. As UIs que adotaram estes mentores relataram que isto foi algo realmente muito benéfico e muito bem aceito (SULLIVAN et. al., 2003).

O que também chama atenção é o fato que muitos artigos e cursos não apresentam informações essenciais em relação ao(s) objetivo(s) de aprendizagem e/ou estratégia instrucional, nem indicam a metodologia utilizada para sistematicamente desenvolver as UIs. A grande parte das informações recolhidas requereram uma análise mais indireta para o entendimento claro do que foi ensinado. Este ponto fraco pode ser também observado em relação a avaliação da maioria das UIs. As avaliações foram raramente apresentadas ou possuíam pouquíssimos detalhes, o que deixa os resultados relatados questionáveis. 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 uma avaliação dessas Uis.

3.4. Ameaças à validade

Como em qualquer revisão sistemática, existem algumas ameaças à validade dos resultados. Foram identificadas ameaças potenciais e aplicou-se 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, considera-se que os resultados dos artigos, sejam positivos

ou negativos, têm apenas uma pequena influência sobre esse mapeamento sistemático, uma vez que se busca caracterizar as UIs, 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, construiu-se cuidadosamente a *search string* para ser o mais abrangente possível, considerando não apenas os principais conceitos, mas também sinônimos. O risco de excluir UIs existentes as quais ainda não foram relatadas por meio de artigos científicos, realizou-se uma busca de forma mais abrangente no *Google*. Além disso, analisou-se as próprias referências da literatura primária encontrada considerando também artigos relevantes citados como literatura secundária. Além da busca por artigos não publicados ainda, as buscas no *Google* auxiliar na identificação de cursos online que também ensinam os conceitos especificados. Também buscou-se analisar em algumas editoras, englobando ainda mais possibilidades de publicações.

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 e documentado um protocolo rígido para a seleção do estudo 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.

4. DESIGN DA UNIDADE INSTRUCIONAL

O design da unidade instrucional é definido com base na definição e análise do contexto em termos do público-alvo, características das escolas e objetivos de aprendizagem. Os planos de ensino das versões da UI proposta, ao final, também são definidos.

4.1. Análise do Contexto

Seguindo o modelo de design instrucional (BRANCH, 2009) é analisado o contexto em termos do público-alvo e das características das escolas.

4.1.1. Análise do Público-Alvo

O público-alvo são alunos do Ensino Fundamental das escolas brasileiras com idade entre dez a quinze anos, aproximadamente. Os alunos neste nível escolar são tipicamente alfabetizados e em alguns casos possuem conhecimento básico da língua inglesa. Além disso, atualmente existe fortemente neste público a capacidade de utilizar e controlar dispositivos eletrônicos, como *smartphones*, *tablets* e computadores, podendo também acessar à *Internet,* já que a grande maioria já possui um *smartphone* próprio (D'ANGELO, 2017). Como o ensino de computação no Brasil normalmente 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).

4.1.2. Análise das Características das Escolas

As escolas públicas municipais de Florianópolis têm computadores com acesso à *Internet,* alocados em uma sala informatizada a qual normalmente possui um professor responsável. O professor responsável pela sala de informática tem tipicamente formação em alguma licenciatura com alguma especialização em tecnologias na educação, porém não costumam ter conhecimentos em computação

(PMF, 2019). 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 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*[4] 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 têm duração geralmente de 45 minutos e são distribuídas no período matutino ou vespertino. A carga de conhecimentos requeridos atualmente está forçando as escolas a realizarem o ensino integral. Desta forma, a criança passa boa parte do dia na escola realizando diversas atividades. Com isto, o tempo para outras atividades além das já programadas se torna muito escasso, tendo que ser planejado com alguma antecedência. Professores formados em outras disciplinas tipicamente não possuem competências de computação, geralmente apresentam somente conhecimentos básicos em informática.

Como o ensino de computação ainda não faz parte do currículo base no Brasil, as aplicações são planejadas para o período contraturno e/ou de forma interdisciplinar em outras disciplinas tendo a disponibilidade de horas variadas. Por este motivo, são projetadas duas unidades instrucionais com duas durações diferentes. A primeira, de mais longa duração, é chamada de “Faça o seu *app*” que visa o aprendizado do processo de desenvolvimento de *apps* completo e como consequência disto a produção dos próprios *apps* pelos alunos. A segunda, de mais curta duração, tem por objetivo também o ensino de conceitos de computação, mas tem um foco maior no tema de design de interfaces, utilizando um *app* como base e aprimorando-o a cada aula e é pensando para ser aplicado de forma interdisciplinar.

4.1.3. Objetivos de Aprendizagem

Os objetivos de aprendizagem definidos para a UI são apresentados na Tabela 27, apresentando o nome, descrição, a área de conhecimento relacionada e a fonte da criação do objetivo. Cada objetivo foi derivado de outros objetivos

4 https://products.office.com/pt-br/word

relacionados presentes em currículos das áreas apresentadas, tendo a principal fonte o guia de currículo CSTA (2016). Os objetivos em *itálico* estão somente na unidade versão curta ou em ambas as unidades.

Tabela 27: Objetivos de aprendizagem da UI

ID	Objetivo de aprendizagem	Área de conhecimento	Fonte
OA1	*Compreender algoritmos como um conjunto de instruções passo-a-passo para realizar tarefas*	*Algoritmo e Programação*	*(CSTA, 2016: 1A-AP-08)*
OA2	Explicar o conceito de um ciclo de vida de *software* e forneça um exemplo, ilustrando suas fases, incluindo as entregas que são produzidas	Engenharia de *Software*/ Engenharia de Usabilidade	(ACM/IEEE, 2013), (UXQB, 2018)
OA3	Desenvolver artefatos computacionais iterativamente de forma colaborativa, seguindo um cronograma	Engenharia de *Software*	(CSTA, 2016: 2-AP-18)
OA4	Identificar e resolver problemas criando sistemas de *software* interativos	Algoritmo e programação / Engenharia de Usabilidade	(CSTA, 2016: 1B-CS-03, 3A-AP-13), (AIGA, 2008)
OA5	Analisar o contexto de sistemas de *software* interativo em termos de usuários, tarefas, dispositivos e ambientes de uso.	Engenharia de Usabilidade	(AIGA, 2008), (ISO 9241-220, 2019), (CSTA, 2016: 1B-IC-19), (ACM/IEEE, 2013)
OA6	Especificar requisitos de sistemas de *software* interativos em termos de funcionalidade e usabilidade.	Engenharia de *Software* / Engenharia de Usabilidade	(CSTA, 2016: 2-AP-19), (ACM/IEEE, 2013)
OA7	Criar protótipos de sistemas de *software* interativos em diferentes níveis (esboços, baixa fidelidade, alta fidelidade, funcional).	Engenharia de Usabilidade	(CSTA, 2016: 2-AP-19, 1B-AP-13, 2-AP-13), (ACM/IEEE, 2013)
OA8	Projetar design que combinam componentes de hardware e *software* para coletar e trocar dados (sensores, APIs, etc.).	Engenharia de *Software* / Engenharia de Usabilidade	(CSTA, 2016: 2-CS-02), (ACM/IEEE, 2013), (GARRET, 2010)
OA9	Modelar processos criando e seguindo algoritmos / mapas de navegação para concluir tarefas.	Algoritimo e programação	(CSTA, 2016: 1A-AP-08)
OA10	Usar fluxogramas, pseudocódigo e / ou mapas de navegação para resolver problemas complexos.	Algoritimo e programação	(CSTA, 2016: 2-AP-10)
OA11	*Projetar o design visual (cores, tipografia, ícones, imagens, etc.) do sistema de software interativo.*	*Engenharia de Usabilidade*	*(ACM/IEEE, 2013), (CSTA, 2016: 2-IC-21), (GARRET, 2010)*
OA12	*Construir sistemas de software interativos que incluam sequenciamento, eventos, condicionais, variáveis, listas e strings usando uma linguagem de programação visual baseada em blocos.*	*Algoritimo e programação*	*(CSTA, 2016: 1B-AP-09, 1B-AP-10, 2-AP-11, 2-AP-12, 3A-AP-14, 3A-AP-16)*
OA13	Procurar e incorporar o feedback dos membros da equipe e dos usuários para refinar uma solução que atenda às necessidades do usuário.	Algorithms and Programming/ Engenharia de Software/Engenharia de Usabilidade	(CSTA, 2016: 2-AP-15)
OA14	*Testar e refinar um sistema de software interativo para funcionalidade e usabilidade.*	*Engenharia de Software*	*(CSTA, 2016: 2-AP-17, 1B-AP-15, 3A-AP-21)*
OA15	*Recomendar melhorias no design de dispositivos de computação, com base nos resultados de verificação e validação.*	*Engenharia de Software*	*(CSTA, 2016: 2-CS-01)*
OA16	*Compartilhar o sistema de software interativo desenvolvido.*	*Algoritmo e programação*	*(CSTA, 2016: 1B-AP-12, 1B-AP-17, 2-AP-16),*

			(LEE et al., 2007)
OA17	*Modificar, remixar ou incorporar partes de um programa existente em seu próprio trabalho, para desenvolver algo novo ou adicionar recursos mais avançados*	*Algoritmo e programação*	*(CSTA, 2016: 1B-AP-12)*
OA18	*Descrever escolhas tomadas durante o desenvolvimento do programa usando comentários, apresentações e demonstrações.*	*Impactos da computação*	*(CSTA, 2016: 1B-AP-17)*

Integrando a aplicação do ensino de computação de forma interdisciplinar, acrescenta-se também objetivos de aprendizagem referentes ao conteúdo da disciplina. Como exemplo uma aplicação na disciplina de Ciências voltada ao consumo e desenvolvimento sustentável poderá incluir objetivos de aprendizagem conforme apresentado na Tabela 28.

Tabela 28: Objetivos de aprendizagem específicos para disciplina de Ciências

ID	Objetivo de aprendizagem	Área de conhecimento	Fonte
OA19	*Construir propostas coletivas para um consumo mais consciente e criar soluções tecnológicas para o descarte adequado e a reutilização ou reciclagem de materiais consumidos na vida cotidiana.*	*Ciências (sustentabilidade)*	*(MEC, 2017)*
OA20	*Tomar ações individuais e coletivas para enfrentar os desafios ambientais*	*Ciências (sustentabilidade)*	*(P21, 2015)*

4.2. Projeto da Unidade Instrucional

A partir da análise de contexto e levando em consideração as necessidades referentes ao ensino de computação no Ensino Fundamental com base no guia de currículo CSTA (2017) voltado ao ensino de computação no K-12 são definidos os objetivos de aprendizagem.

4.2.1. Contexto de Aplicação

A UI é projetada para ser aplicada de forma interdisciplinar em outras disciplinas da educação básica e/ou no contra turno ou como atividade extracurricular. A versão mais curta é projetada para a disciplina de Ciências com o tema desenvolvimento e consumo sustentável, o qual compreende ensino de conceitos de computação ao mesmo tempo que os assuntos de sustentabilidade são envolvidos. A disciplina de Artes também é um exemplo no qual a unidade pode ser

inserida, pois a criatividade e o próprio desenvolvimento da interface do usuário são conceitos derivados da área de design. A unidade em sua versão longa é projetada para ser aplicada durante um tempo mais prolongado, com diversas aulas, cada uma abordando conceitos específicos do processo de desenvolvimento de *apps*.

As versões da UI são projetadas com o objetivo de serem ministradas de forma presencial, sendo coordenadas por um professor da educação básica com auxílio ou não de um mentor capacitado nos assuntos envolvidos na unidade. Há também a possibilidade da unidade ser aplicada à distância, via *Moodle*[5] por exemplo.

4.2.2. Planos de Ensino

A construção do plano de ensino das versões da UI é baseada nos objetivos de aprendizagem, análise do público-alvo e do ambiente das escolas (Tabela 29 e 30).

Tabela 29: Plano de ensino da UI da versão longa

Parte	Aula(duração)	Conteúdo em geral	Algoritmo e programação	ES	EU	ID objetivo de aprendizagem	Estratégia instrucional	Método instrucional	Avaliação
Conceitos fundamentais									
1	1 (2h 30min)	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 (2h 30min)	Conceitos básicos de computação: algoritmo/programação	Condicional, evento, sensor, API, string	-	Design visual, Guia de estilo (Material Design) (apenas nos extras)	OA8, OA10, OA12	Instrução direta, aprendizado experimental	Ensino explícito, demonstrações, condução de experimentos	Observação, *app* "Encontre-me" desenvolvido
Faça seu *app*!									
2	1 (2h 30min)	Processo de desenvolvimento de *apps* e identificação do problema e solução	-	Processo de desenvolvimento de *apps*	*Design Thinking*	OA2, OA4	Instrução direta, instrução interativa, estudo independente	Ensino explícito, demonstrações, resolução de problemas, prática em pares, tarefas de casa	Artefatos desenvolvidos

5 https://moodle.org/?lang=pt_br

	2 (2h 30min)	Análise de contexto e especificação de requisitos	-	User stories, especificação de requisitos de *software*, Task case	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 (2h 30min)	Design de baixa fidelidade do *app* e testes	-	-	Sketch/protótipo de baixa fidelidade (em papel), teste de sketch	OA7	Instrução direta, instrução interativa	Ensino explícito, demonstrações, prática em pares	Artefatos desenvolvidos, *app* desenvolvido
	4, 5, 6 (2h 30min cada)	Programação e teste do protótipo no *App Inventor*	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, *app* desenvolvido
	7 (2h 30min)	Criação e teste do design visual no *App Inventor*	-	-	Design visual: cores, tipografia, ícones, imagens, marca, launcher icon, Guia de estilo (Material Design),	OA11	Instrução direta, instrução interativa	Ensino explícito, demonstrações, prática em pares	Observação, *app* desenvolvido
	8 (2h 30min)	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, *app* desenvolvido, questionário
	9 (2h 30min)	Compartilhando o *app*	Compartilhamento de *apps*, publicar (galeria e playstore), versionar *software*, distribuir código fonte	-	-	OA16	Instrução direta, instrução interativa	Ensino explícito, demonstrações, prática em pares	Observação
	10 (2h 30min)	Apresentação do *app*	-	-	-	OA18	Instrução direta, atividade de fixação (treino da apresentação)	Ensino explícito, demonstrações, prática em pares	Observação, qualidade da apresentação teste

Tabela 30: Plano de ensino da versão curta

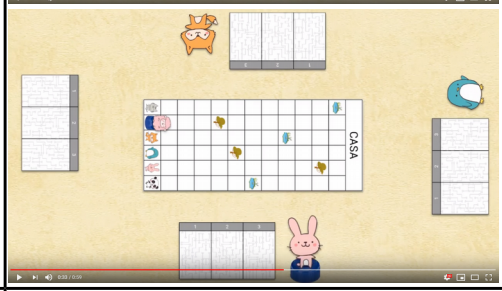
Encontros/Carga horária		**Conteúdo**	**Área de conhecimento**	**ID objetivo de aprendizagem**	**Estratégia Instrucional**	**Material Instrucional**	**Avaliação**
1	**1h30 (2 aulas)**	Motivação sobre ensino de computação e o desenvolvimento de aplicativos. Conceitos básicos de computação: algoritmo/programação. Visão geral do processo de desenvolvimento de *apps*	Algoritmo e Programação, Ciências	OA1, OA19	Instrução direta (aula expositiva), atividade coletiva e jogo educacional	Slides, vídeo, jogo SplashCode	Avaliação processual, observação, diário de campo, avaliação do jogo de tabuleiro

2	**1h30 (2 aulas)**	Introdução a programação com *App Inventor*	Algoritmo e programação	OA17, OA12	Instrução direta (aula expositiva), atividade prática de programação e teste	Slides, ambiente *App Inventor*, *App Inventor companion* para uso no celular	Avaliação processual, observação, diário de campo
3	**1h30 (2 aulas)**	Criação, programação e teste do design visual no *App Inventor*, cores, tipografia, desenvolvimento de uma prática que gere impacto social	Engenharia de Usabilidade, impactos da computação, Ciências	OA11, OA19, OA20	Instrução direta (aula expositiva), atividade de fixação (exercício), atividade prática de programação e teste	Slides	Avaliação processual, observação, diário de campo
4	**1h30 (2 aulas)**	Criação, programação e teste do design visual no *App Inventor*, imagens, hierarquia.	Design visual, algoritmo e programação, verificação e validação de *software*	OA14, OA15	Teste do protótipo no *App Inventor*	Slides	Avaliação processual, observação, diário de campo, *app* desenvolvido pelos alunos
5	**1h30 (2 aulas)**	Compartilhando os *apps*, divulgando o *app*	Algoritmo e programação, habilidade de pensamento crítico, a resolução problema, comunicação e colaboração, Ciências	OA16, OA19, OA20	Instrução direta (aula expositiva), atividade de fixação	- Slides; - Vídeos; - Uso do google drive; - Câmera;	- Avaliação processual; - Diário de Campo
6	**1h30 (2 aulas)**	Apresentação do *app*	Habilidade de pensamento crítico, resolução problema, comunicação e colaboração, Ciências, impactos da computação, linguagens	OA18, OA19, OA20	Instrução direta, atividade de fixação (treino da apresentação)	Slides	- Avaliação processual; - Diário de Campo; - Apresentação dos alunos

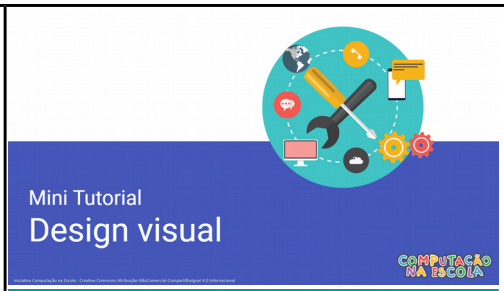
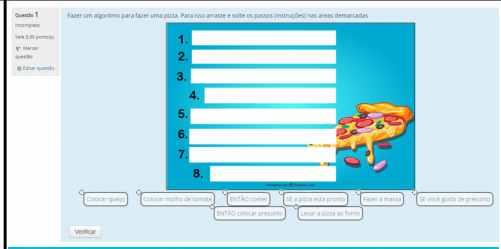
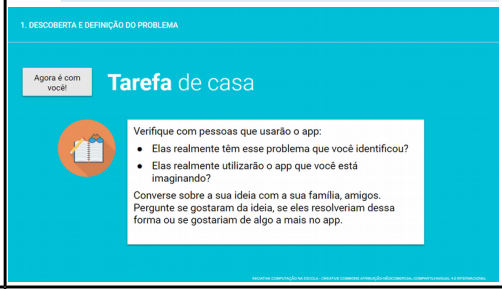
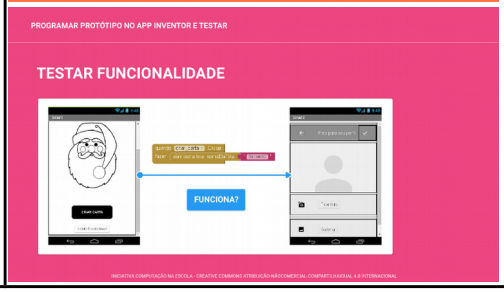
4.2.3. Desenvolvimento do Material Didático

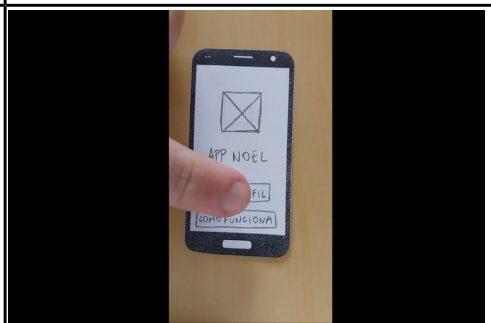
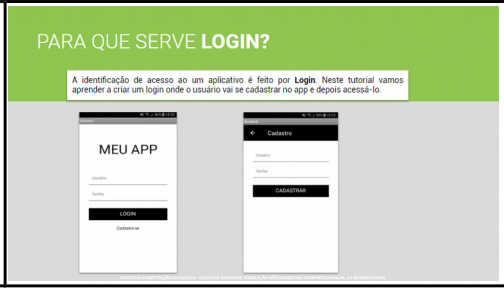
Com base nos planos de ensino desenvolvidos, os materiais didáticos foram então construídos. Diversos materiais foram utilizados nas aulas presenciais e também como material para casa. A Tabela 31 abaixo apresenta, descreve e ilustra estes materiais.

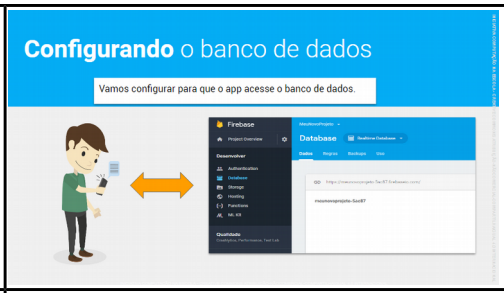
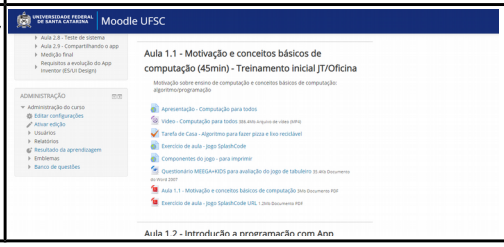
Tabela 31: Materiais didáticos desenvolvidos

Material	Descrição	Imagem
Jogo SplashCode	Jogo de tabuleiro para a aplicação do conceito de algoritmos. Os jogadores utilizam as cartas de movimento para fazer os personagens chegarem até o campo *casa*, desviando dos obstáculos.	
Vídeos - jogo *SplashCode*	Vídeos de auxílio para aprender a como jogar o jogo *SplashCode*. Existe um conjunto de três vídeos curtos: “preparação”, “cartas” e “como jogar”.	 
Tutorial *app* “Encontre-me”	Desenvolvimento e tutorial para construção do *app* “Encontre-me”. Este *app* tem por objetivo permitir os usuários mandarem sua localização atual via *WhatsApp*[6]. Durante o desenvolvimento do *app* pelo tutorial, vários outros conceitos de computação são abordados, como *APIs*, eventos e outros.	 

6 https://www.whatsapp.com/

Mini-tutoriais para evoluir o *app* “Encontre-me”	Conjunto de três mini-tutoriais para a evolução do *app* “Encontre-me”. Os tutoriais ensinam a adicionar um campo para o usuário digitar alguma mensagem além da localização, compartilhar via outros *apps* e melhorar o design visual.	 
Tarefas de casa	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.	 
Slides a serem apresentados nas aulas presenciais	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.	 

Workbooks	Guias de referência para a produção dos artefatos envolvidos no processo de desenvolvimento de *apps* proposto no projeto.	
Vídeo teste de usabilidade	Vídeo para ensinar a como realizar o teste de usabilidade do protótipo de interface em papel.	
Tutoriais de funcionalidades específicas do *App Inventor*	Tutoriais para desenvolvimento de funcionalidades características de diversos *apps* no *App Inventor*, como banco de dados, listas, *login*, entre outros.	

		
Moodle do curso	Criação de um *Moodle* para facilitar o acesso aos materiais de aula e tarefas aos alunos. Também é um meio de realizar o curso na modalidade à distância.	

4.2.3. Avaliação do Aluno

A avaliação do aluno é realizada de várias formas, incluindo por desempenho, testes (questões objetivas) e autoavaliação.

Uma das formas de avaliação é a utilização da rubrica criada no projeto *CodeMaster* v2.0 (ALVES, 2019). Esta rubrica avalia para cada *app* desenvolvido uma série de questões sobre o desempenho em relação aos conceitos do pensamento computacional baseada sobretudo no currículo da CSTA (2017). A Figura 43 ilustra um extrato dessa rubrica.

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 unidimension al.	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	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).

Figura 43: Extrato da rubrica CodeMaster v2.0 (ALVES, 2019)

Além da rubrica para avaliar os conceitos relacionados ao pensamento computacional, é utilizada uma outra rubrica para avaliar o desempenho dos alunos em relação ao design visual em seus *apps* desenvolvidos. O objetivo desta é avaliar os alunos com base no que seus *apps* apresentam em termos de componentes visuais analisando se estão em concordância com o guia de estilo *Material Design*.

Além dos *apps* (código) criados pelos alunos são avaliados na versão longa do curso também os artefatos criados durante a unidade referente a cada aula por meio da Rubrica 1 (Tabela 32), possibilitando avaliar os alunos em termos do desempenho na criação destes artefatos com uma pontuação de 0 (mínima) a 30 (máxima). A avaliação por meio desta rubrica é feita pelo professor.

Tabela 32: Rubrica 1 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	Não soube	Descreveu	Descreveu com	Descreveu com

	solução	descrever	incompleto	pouca clareza	clareza
Aula 2.2	Persona	Não soube caracterizar a persona	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					

Em relação ao atingimento do objetivo de aprendizagem do design visual do *app* também baseado no aplicativo criado, a avaliação é realizada por meio da Rubrica 2 (Tabela 33) com níveis de desempenho representando o grau de atingimento. A pontuação varia entre 0 (mínima) e 48 (máxima). Esta rubrica visa a avaliação para ambas as unidades criadas, possibilitando a atribuição de uma nota para os alunos em termos da interface do *app* desenvolvida. Essa avaliação é feita manualmente pelo professor.

Tabela 33: Rubrica 2 para avaliação da aprendizagem sobre design visual

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	Não	Vagamente	Parcialmente, em alguns casos	Largamente	Completamente

tela asseguram a legibilidade					
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 auxiliam 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 distorcidas)	Não	Vagamente	Parcialmente, em alguns casos	Largamente	Completamente
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

Outra forma utilizada de avaliação do aprendizado dos alunos é a avaliação por meio de testes, perguntas objetivas sobre conhecimentos nos assuntos abordados na unidade. Estas questões são aplicadas no último encontro da UI, no qual os alunos devem responder perguntas sobre design visual e o processo de desenvolvimento de *apps.* Dependendo da versão da UI aplicada, as questões podem não ser adequadas e devem ser retiradas do conjunto de questões. A Tabela 34 apresenta as questões incluídas no teste.

Tabela 34: Questões aplicadas aos alunos da UI

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. 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. APLICAÇÃO E AVALIAÇÃO DA UNIDADE INSTRUCIONAL

5.1. Avaliação da unidade instrucional

As seções a seguir descrevem como ocorreu a avaliação da unidade instrucional desenvolvida neste trabalho.

5.1.1. Definição da Avaliação

Pergunta de pesquisa

Práticas pedagógicas de ensino de engenharia de *software* e design de interfaces de forma extracurricular inseridas no ensino de desenvolvimento de aplicativos móveis com *App Inventor* contribuem para o desenvolvimento competências e motivação/interesse à computação do aluno e apresentam qualidade suficiente?

Hipóteses

A utilização de práticas pedagógicas para o ensino de engenharia de *software* e design de interfaces de forma extracurricular inseridas no ensino de desenvolvimento de aplicativos móveis com *App Inventor* (h1) aumenta as competências, (h2) aumenta o interesse pela experiência do aluno, (h3) possui boa usabilidade e (h4) facilita a aprendizagem referente ao ensino de conteúdos de computação.

Design de pesquisa

Estudo de caso: *one-shot pre-test - post-test* (O X O).

Medição

Para a medição da avaliação foi utilizada a abordagem GQM (BASILI et al., 1994), que consiste em sistematicamente decompor a pergunta de pesquisa em perguntas de análise e medidas e o desenvolvimento de instrumentos de medição.

Perguntas de análise

PA1. Práticas pedagógicas para o ensino de engenharia de *software* e design de interfaces 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 engenharia de *software* e design de interfaces 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 unidades instrucionais para o ensino de engenharia de *software* e design de interfaces inseridas no ensino de desenvolvimento de aplicativos móveis com *App Inventor* facilitam a aprendizagem (usabilidade da unidade)?

PA4. As práticas pedagógicas projetadas nessas unidades instrucionais para o ensino de engenharia de *software* e design de interfaces inseridas no ensino de desenvolvimento de aplicativos móveis com *App Inventor* aumentam o interesse dos alunos em relação a aprendizagem de conteúdos de computação?

A Tabela 35 sintetiza a definição da avaliação da unidade instrucional.

Tabela 35: Síntese da definição da avaliação da unidade

Pergunta de análise	**Medida**	**Instrumento de medição**
PA1. A UI aumenta competências dos alunos referente a aprendizagem de conteúdos de computação?	**M1.1** Grau de aprendizagem referente a 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 *App Inventor* para desenvolver um *app* 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 a capacidade de fazer aplicativos para *smartphones*, sendo um processo de desenvolvimento de *apps*	- 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 UI promove uma	**M2.1** Grau de diversão das aulas	- Questionário pós-unidade

experiência de aprendizagem agradável e divertida?	**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 UI em outro momento	alunos - Questionário pós-unidade pais - Questionário pós-oficina alunos
PA3. A UI 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 UI aumenta o interesse dos alunos referente a aprendizagem de conteúdos do computação?	**M4.1** Vontade de aprender computação na escola **M4.2** Grau de satisfação em fazer *apps* 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

Para a realização do plano da avaliação durante a aplicação da unidade instrucional em sua versão longa, foi criado um planejamento aula a aula para a aplicação dos instrumentos de medição. A Tabela 36 ilustra este planejamento, utilizado durante o período de aplicação. Cabe ressaltar que em todas as aulas o instrumento de medição de observação foi utilizado, levantando a análise tanto da qualidade da UI quanto da aprendizagem dos alunos. Os *workbooks* citados na tabela abaixo foram utilizados para medir os resultados a fim de responder a pergunta de análise PA1, com a utilização das rubricas apresentadas na seção 4 deste trabalho. O questionário *MEEGA+Kids*[7] também mencionado na Tabela 36 é um modelo de avaliação de jogos no ensino de computação para a educação básica desenvolvido no laboratório GQS/INCoD/INE/UFSC.

7 http://www.gqs.ufsc.br/meegakids-avaliar-jogos-para-ensinar-computacao-na-educacao-basica/

Tabela 36: Planejamento da avaliação da unidade versão longa

Aula	**Momento da coleta de dados**	**Quem responde**	**Instrumento de medição**
Oficinas	Começo da aula	Alunos	Questionário pré-oficina alunos
	Após o jogo	Alunos	Questionário *MEEGA+Kids*
	Término da aula	Alunos	Questionário pós-oficina alunos
	Após a aula	Pais	Questionário pós-oficina pais
Primeiro encontro na UFSC (também foi realizada uma oficina com os Jovens Tutores e Mentores)	Começo da aula	Alunos	Questionário pré-unidade
Aula 2.1 e 2.2 - Processo de desenvolvimento de *apps*, identificação problema/solução e análise do contexto e requisitos	Término da aula	Alunos	*Workbooks* aula 2.1 e 2.2
Aula 2.3 - Design de baixa fidelidade	Final da aula	Alunos	*Workbook* aula 2.3
Aula 2.7 - Design visual e testes	Início da aula	Alunos	Questionário pré-aula design visual
	Final da aula	Alunos	Questionário pós-aula design visual
Aula 2.8 - Testes funcional e de usabilidade	Final da aula	Alunos	*Workbook* aula 2.8
Último encontro	Final da aula	Alunos	Questionário pós-unidade alunos
	Após a aula	Pais	Questionário pós-unidade pais

A Tabela 37 apresenta o planejamento da avaliação para a versão curta.

Tabela 37: Planejamento da avaliação da unidade versão curta

Aula	**Momento da coleta de dados**	**Quem**	**Questionários**
Aula 1	Início da aula	Alunos	Questionário pré-unidade alunos
	Após o jogo	Todos que jogaram o jogo (professores e alunos)	Questionário MEEGA+KIDS
	Após a aula	Alunos	Tarefa de casa sobre algoritmos
Aula 3	Durante a aula	Alunos	Questionário sondagem design visual
	Durante a semana da aula	Alunos	Três perguntas a serem respondidas em sala de aula
Aula 4	Após a aula	Alunos e professor	Questionário de avaliação do design visual do *app* (alunos se autoavaliam e professor avalia os *apps*). Questionário aula design visual.

Aula 6	Após a aula	Alunos	Questionário pós-unidade alunos. Questionários habilidades do século XXI.

5.2. Aplicação da Unidade Instrucional

A unidade instrucional foi aplicada em duas versões e em dois momentos diferentes. Nas seções abaixo são apresentadas e ilustradas as aplicações da unidade.

5.2.1. Aplicação da Versão Longa - Jovens Tutores 2018-2

A unidade instrucional em sua versão longa/completa foi aplicada no contexto do projeto Jovens Tutores de Programação[8], coordenado pela iniciativa Computação na Escola/INCoD/INE/UFSC. O projeto envolveu dez (cinco meninas e cinco meninos) alunos (chamados de jovens tutores) da E.B.M. Almirante Carvalhal durante aproximadamente 4 meses, em encontros semanais de três horas na própria escola. O projeto foi aprovado pela prefeitura de Florianópolis (Anexo IV) e pela CEPSH - Comitê de Ética em Pesquisa com Seres Humanos da UFSC com o número do parecer 1.807.891. Os alunos foram selecionados e convidados pela professora de tecnologia educional da escola, os quais participaram de maneira voluntária no contraturno de suas aulas. As aulas da unidade foram ministradas pelo autor deste projeto, com auxílio de um mestrando e um professor do mesmo laboratório (chamados de professores), a professora da sala informatizada da escola e mentores voluntários da empresa patrocinadora do projeto. A sala informatizada conta com aproximadamente 20 computadores, todos com o sistema operacional Windows 10, 4GB de memória RAM e conectados à internet por meio de uma rede por cabos. A Tabela 38 faz um resumo da quantidade dos dados coletados durante a aplicação.

Tabela 38: Quantidade de respostas para cada questionário aplicado versão longa

Questionário/instrumento	**Quantidade**
Quantidade de questionários pré unidade alunos respondidos	10
Quantidade de questionários pós-unidade alunos respondidos	9
Quantidade de questionários pré aula design visual respondidos	9

8 http://www.computacaonaescola.ufsc.br/?p=2645

Quantidade de questionários pós aula design visual respondidos	9
Quantidade de *workbooks* respondidos (para cada um)	5

A Figura 44 apresenta os dados demográficos desta aplicação.

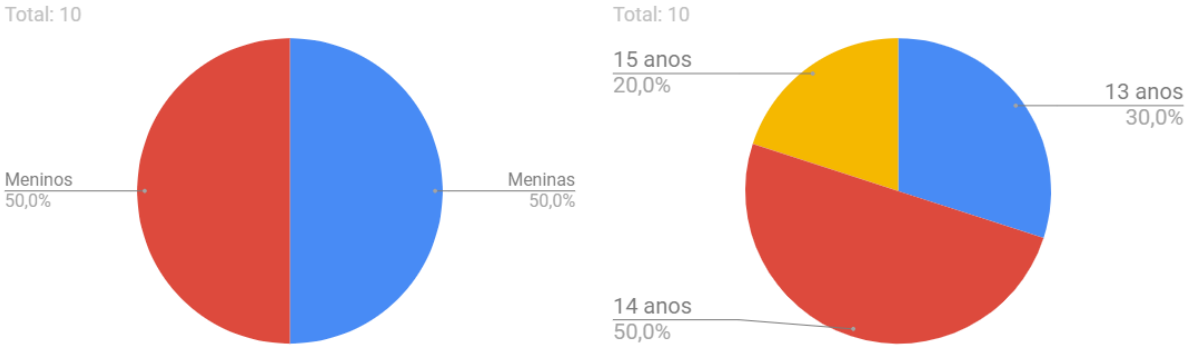


Figura 44: Dados demográficos da aplicação versão longa

O primeiro encontro foi realizado na UFSC, o qual proporcionou o conhecimento de todos os envolvidos no projeto, apresentação e definição de detalhes do cronograma, entre outros assuntos (Figura 45). Neste momento os alunos entregaram o TCLE assinado e os questionários de pré unidade foram preenchidos. Como uma parte introdutória, os jovens tutores aprenderam conceitos básicos de computação, começando com o conhecimento de algoritmos. Para isto, um conjunto de slides foram apresentados e após o jogo *SplashCode* foi jogado entre os jovens tutores e mentores voluntários. Foi neste encontro também que os alunos tiveram a oportunidade de conhecer o *App Inventor*, construindo o *app* "Encontre-me" com auxílio dos mentores voluntários e do tutorial desenvolvido como parte deste trabalho.

Figura 45: Primeiro encontro do projeto na UFSC versão longa

O restante dos encontros ocorreram semanalmente na escola (Figuras 46 e 47). A cada encontro diferente, os alunos foram ensinados sobre novos conceitos e então aplicaram na construção de seus próprios *apps*. Os dez alunos foram divididos em pares, formando cinco grupos (e cinco *apps* diferentes). No primeiro, os jovens tutores aprenderam sobre o processo de desenvolvimento de *apps* e identificação de um problema e solução, o qual tiveram que selecionar um problema e bolar uma solução para ele por meio de um aplicativo para *smartphone*. Além disso, também consistiu no ensino da análise do contexto e especificação de requisitos, definindo quais as tarefas cada *app* realizará, definições de requisitos de usabilidade e funcionalidade. No segundo, foi abordado o ensino do design de baixa fidelidade e testes. Os alunos construíram o protótipo em papel, testaram com os colegas, anotaram o que funcionou e o que não funcionou e, então, bolaram o design final do protótipo de baixa fidelidade.

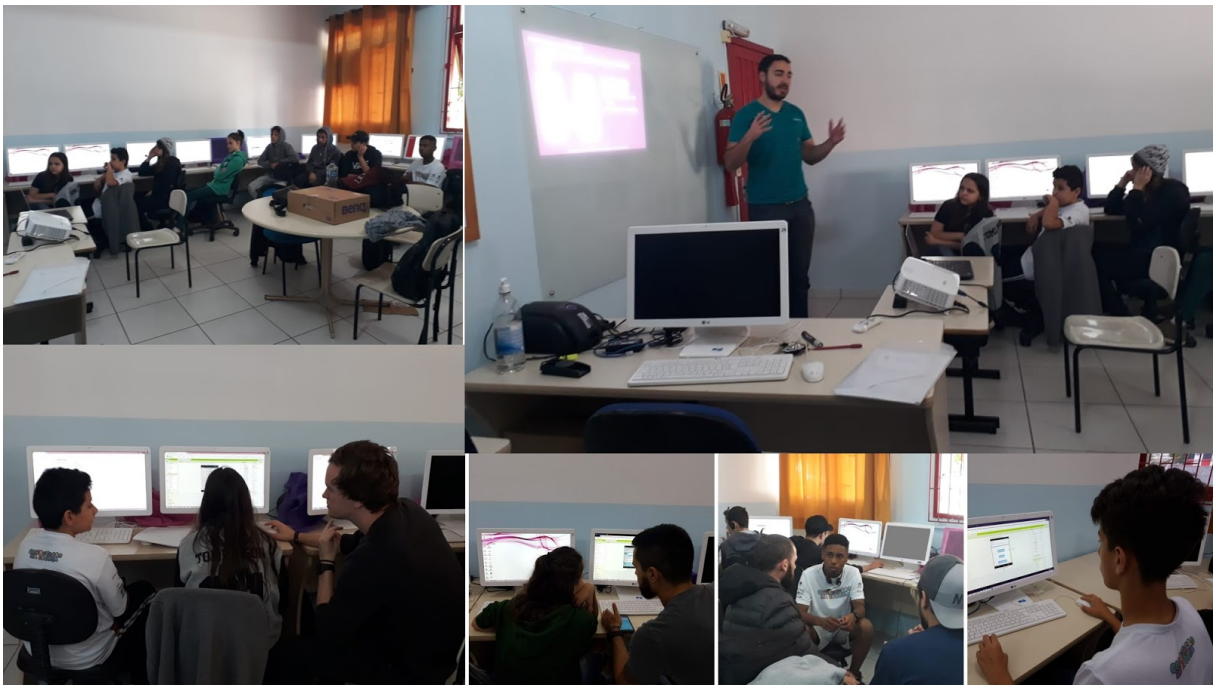


Figura 46: Encontros de ensino do processo de desenvolvimento de apps 1 versão longa

Os encontros 4, 5 e 6 consistiram na programação e testes dos *apps*. Nestes encontros, os alunos foram auxiliados na construção de seus aplicativos, com ajuda dos mentores, professores e material didático 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, o qual os alunos evoluíram seus *apps* em termos do design de interfaces e testaram com os demais colegas. No oitavo encontro, os jovens tutores aprenderam sobre os testes funcional e de usabilidade, testando seu *app* em termos das funcionalidades e também do design das interfaces criadas por eles. O último encontro consistiu no ensino de como compartilhar seu *app*, via arquivo *.aia*, arquivo *.apk*, galeria do *App Inventor* e na *App Store*. Os alunos também tiveram uma revisão geral dos conceitos ensinados em todo curso para depois responderem os questionários finais de avaliação.

Para todos os 5 *apps* construídos pelos alunos foram criados os *workbooks*, relatando todo o trabalho realizado no processo de desenvolvimento dos *apps*. Em cada aula que demandava o uso de *workbooks* os alunos desenvolveram uma evolução a uma nova parte, com base no que produziram anteriormente. A Figura 48 ilustra os *workbooks* completos para um dos *apps* desenvolvidos.

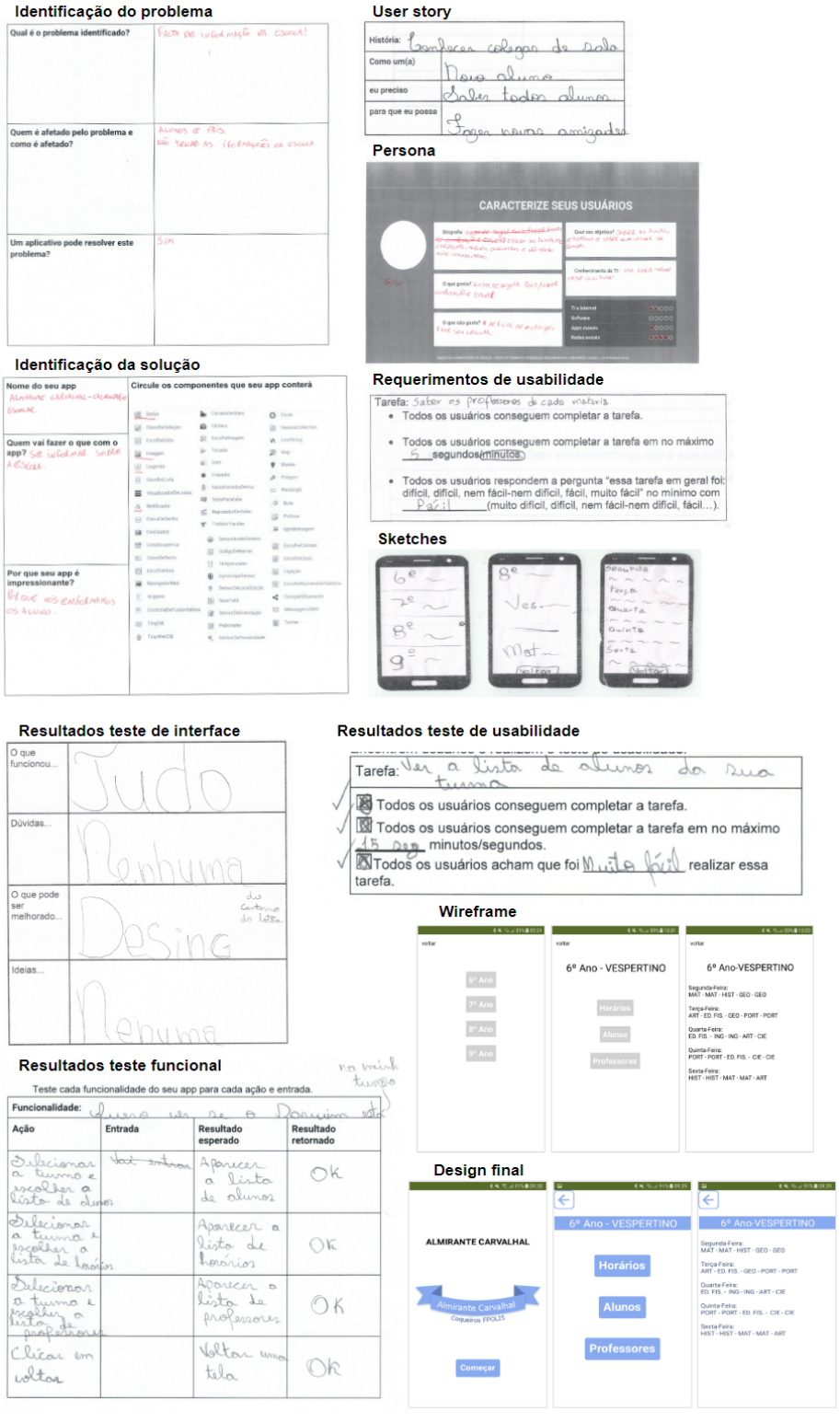


Figura 48: *Workbooks* do *app* "InfoCarvalhal" versão longa

No total, foram desenvolvidos 5 *apps*[9] pelos alunos na unidade versão longa (Figura 49). Os *apps* são os seguintes:

- **Achei o seu emprego**: auxilia a encontrar um emprego. Permite o cadastro de vagas pelas empresas e também a busca por empregos
- **HealthyPlants**: fornece 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.
- **InfoCarvalhal**: apresenta informações úteis sobre a Escola Básica Municipal Almirante Carvalhal: turmas, alunos, horários e professores.

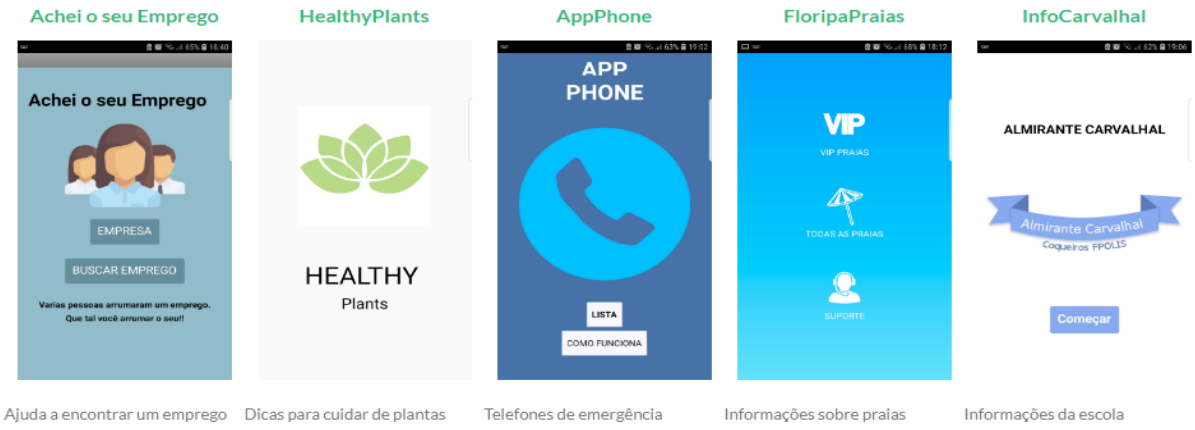


Figura 49: Apps desenvolvidos na UI versão longa

De acordo com o objetivo do projeto, em algumas semanas foram realizadas oficinas de programação de *apps* para outras crianças da escola (Figura 50). Estas oficinas englobam o ensino inicial de computação com o jogo *SplashCode* e a construção do *app* "Encontre-me". O objetivo dos jovens tutores nas oficinas é auxiliar os outros alunos, passando um pouco do conhecimento que já adquiriram durante a aplicação do projeto. Ao todo foram realizadas três oficinas, atingindo 51 crianças de faixa etária entre 10 e 14 anos.

9 http://www.computacaonaescola.ufsc.br/?p=2739

Figura 50: Oficinas realizadas na escola versão longa

5.2.2. Aplicação da Versão Curta – ECOPET 2019-1

A aplicação da versão curta da unidade instrucional teve sua realização nos meses abril e maio do ano de 2019, envolvendo alunos do quinto ano da Escola Básica Municipal Almirante Carvalhal da turma da professora Suleica Kretzer. O projeto foi aprovado pela prefeitura de Florianópolis (Anexo IV) e pela CEPSH com o número do parecer 2.677.698. O objetivo principal de toda a aplicação foi o desenvolvimento de um *app* para a visualização de pontos de coleta de tampas de plástico (em geral, de garrafas PET) reunidos pelo projeto ECOPET[10]. Ao total, 28 alunos participaram desta versão da unidade. As aulas ocorrem de maneira semanal com duração de 1 hora e 30 minutos cada, ministradas pela equipe da Computação na Escola/INCoD do Departamento de Informática e Estatística/UFSC. A sala informatizada onde ocorreram as aulas durante a aplicação conta com aproximadamente 20 computadores, todos com o sistema operacional Windows 10, 4GB de memória RAM e conectados à internet por meio de uma rede por cabos. A Tabela 39 faz um levantamento da quantidade dos dados coletados durante esta aplicação.

10 https://www.facebook.com/ecopettampastampinhas/

Tabela 39: Quantidade de respostas para cada questionário aplicado versão curta

Questionário/instrumento	Quantidade
Quantidade de questionários pré unidade alunos respondidos	26
Quantidade de questionários pós unidade alunos respondidos	24

Houve diferença na quantidade de respostas pela falta de alunos nos dias em que os questionários foram aplicados. Adicionalmente, observa-se que em alguns casos os alunos não necessariamente preencheram todos os itens do questionário, justificando a diferença da quantidade de respostas aos itens.

A Figura 51 apresenta os dados demográficos dos participantes da aplicação.

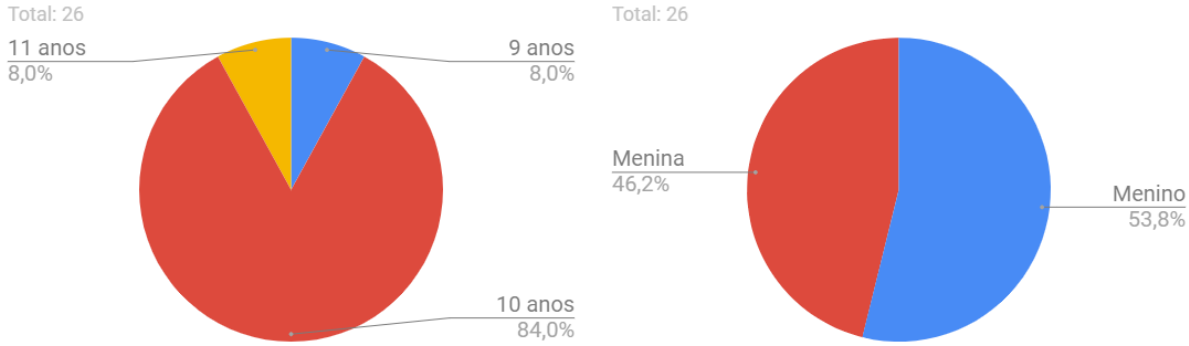


Figura 51: Dados demográficos aplicação versão curta

A aplicação desta versão da unidade ocorreu de forma interdisciplinar, envolvendo a programação de *apps* na disciplina de Ciências. A ideia é a construção de um *app* que possibilita a visualização de pontos de coleta de tampas de plástico que são vendidas para reciclagem e com a arrecadação do dinheiro possibilita a castração de animais abandonados pelo projeto ECOPET. Uma versão inicial do *app* foi previamente entregue aos alunos com a parte da programação mais avançada, possibilitando um ensino mais focado no desenvolvimento do design da interface do aplicativo. Como o número de aulas nesta aplicação é reduzido em comparação a aplicação da versão longa, alguns conteúdos de programação não foram ensinados.

Na primeira aula, os alunos tiveram uma breve motivação sobre o porquê de aprender conteúdos de computação e o processo de desenvolvimento de *apps*. Aprenderam de forma teórica e prática conceitos básicos de computação, como

algoritmos, podendo aplicar o conteúdo visto com o jogo SplashCode e por meio de tarefas de casa.

Na segunda aula, os alunos foram ensinados a desenvolver uma versão simplificada do *app* com a ferramenta *App Inventor*, permitindo entender melhor as particularidades da ferramenta e um primeiro contato com o desenvolvimento de *apps* para *Android* (Figura 52). Nesta primeira versão, os alunos adicionaram locais manualmente (“hard-coded”) no *app* que podiam ser vistos na tela, mas os locais eram apenas vistos localmente em cada *app* desenvolvido.

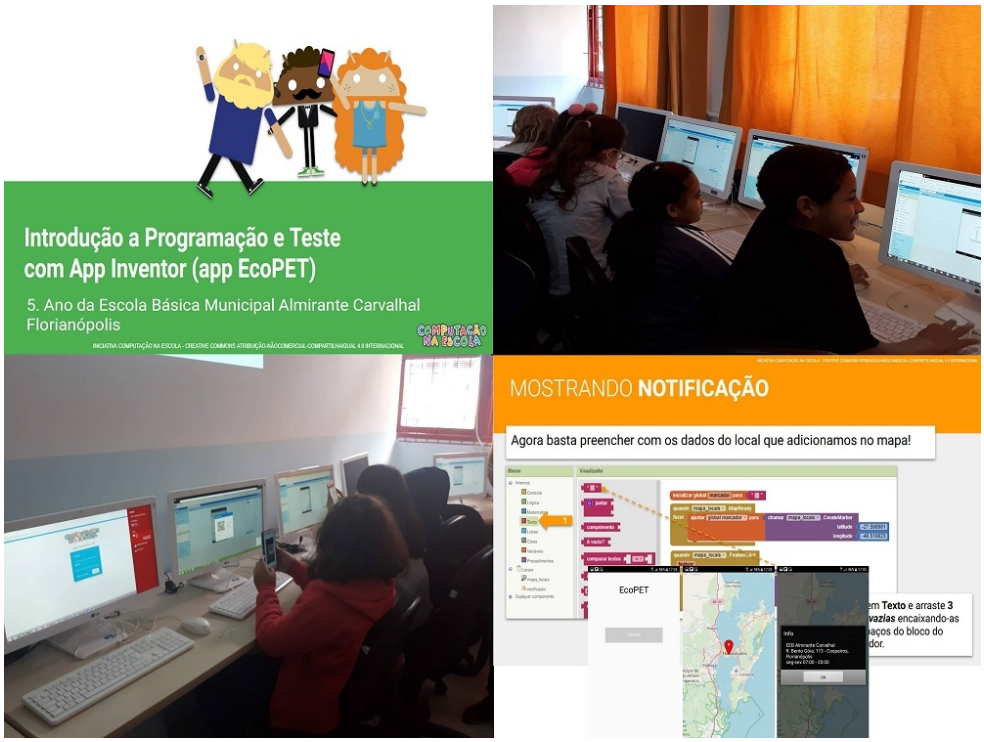


Figura 52: Segunda aula da aplicação curta da unidade instrucional

Nas aulas 3 e 4, o foco se voltou ao design de interface do *app*. Os alunos foram ensinados sobre os conceitos de cor, imagem, tipografia e composição (Figura 53). O objetivo era evoluir o *app* em uma versão mais completa entregue aos alunos em termos do design visual. A versão que os alunos receberam do *app* possuía um banco de dados com os locais adicionados, assim mostrando que a versão anteriormente desenvolvida era limitada por conta dos locais serem adicionados diferentemente em cada *app*. Foi possível perceber que quando os alunos

adicionaram um novo local pelo *app*, os outros colegas puderam ver este local adicionado por conta do armazenamento no banco de dados.

Figura 53: Encontros de design visual da versão curta da unidade

O quinto encontro consistiu no ensino do compartilhamento do *app* desenvolvido pelos alunos, apresentando as diversas formas possíveis para que outras pessoas possam ver/utilizar/aprimorar o *app* criado. Também foi ensinado aos alunos como apresentarem seus *apps* para outras pessoas, como uma habilidade necessária para seus futuros.

No último encontro os alunos apresentaram seus *apps* para a turma e o projeto como todo também para outras pessoas da comunidade, inclusive os pais. O último encontro foi no IFSC Continente Florianópolis. Lá eles apresentaram para os familiares, comunidade e representantes do projeto ECOPET.

5.3. Análise dos Dados

A análise dos dados é apresentada separadamente para as duas aplicações da unidade instrucional. As seções a seguir contém a análise dos dados referente a cada uma das aplicações.

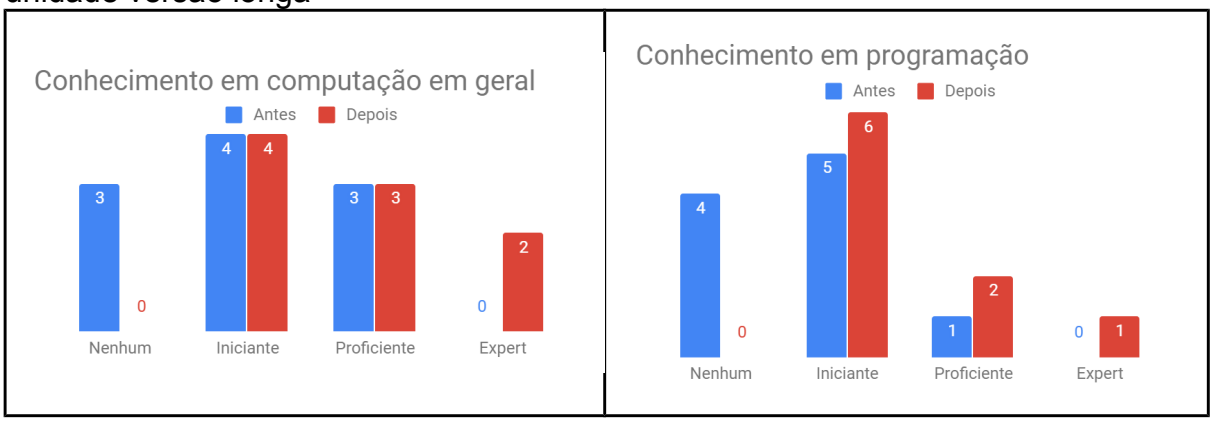
5.3.1. Análise dos Dados Aplicação da Versão Longa

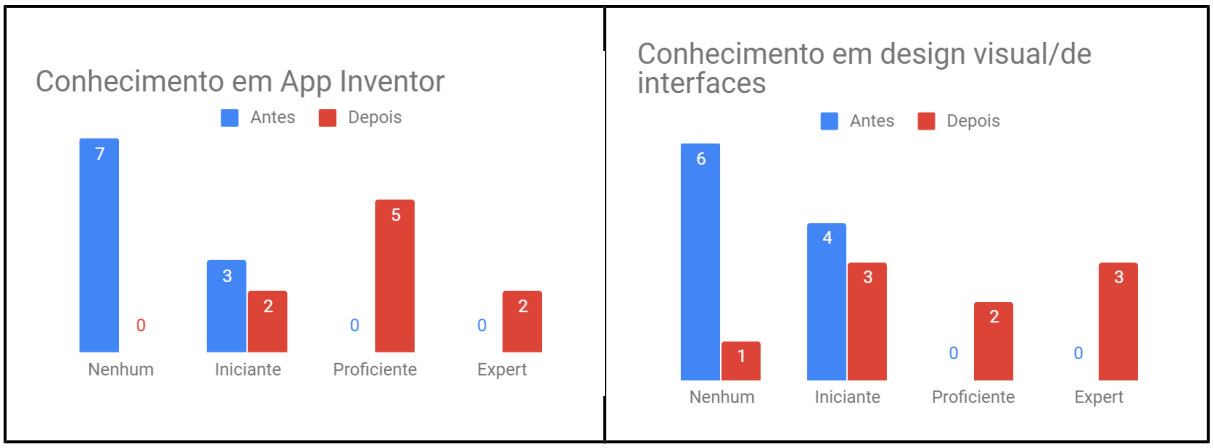
De acordo com as perguntas de pesquisas descritas anteriormente, os dados são analisados a fim de respondê-las.

Avaliação da aprendizagem

Os alunos demonstram de forma geral que realmente percebem que aprenderam os conceitos apresentados para cada gráfico da Tabela 40. Em comparação aos resultados de antes e depois da unidade, percebe-se claramente que houve um aumento do nível de conhecimento percebido em praticamente todos os alunos e assuntos abordados. Com grande destaque está o nível de conhecimento em *App Inventor*, antes da oficina sete alunos indicam não ter conhecimento algum com a ferramenta. Já depois da unidade, cinco alunos indicam um nível “proficiente” e dois “expert”, mostrando um grande aumento no grau de conhecimento constatado pelos alunos no desenvolvimento de *apps* com o *App Inventor*. A única situação em que um aluno acredita não ter adquirido conhecimento algum foi com as aulas de design visual, apresentando alguma dificuldade com o assunto ou até mesmo insatisfação.

Tabela 40: Gráficos dos dados sobre o nível de conhecimento antes e depois da unidade versão longa





Os resultados em relação ao nível de aprendizagem dos conteúdos foi bastante variado (Figura 54). Design do protótipo em papel, programar e testar, design visual e teste funcional e de usabilidade foram os conceitos em que mais os alunos percebem ter alto nível de conhecimento. Por outro lado, programação e análise do contexto e especificação dos requisitos são os conteúdos em que alunos mais se sentem iniciantes. Houve apenas uma resposta indicando nenhum conhecimento aprendido em relação ao design visual.

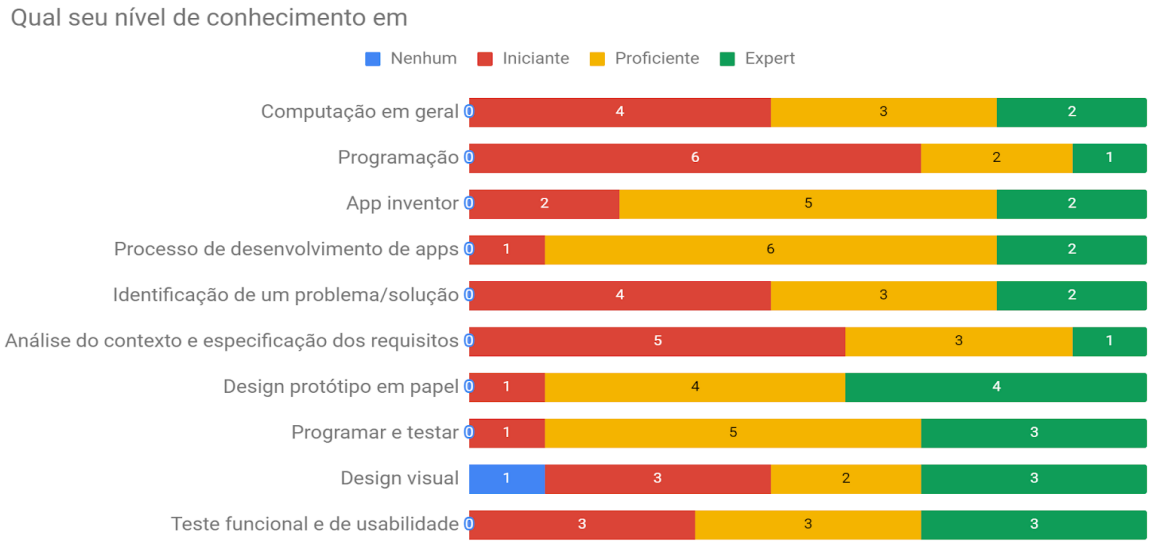


Figura 54: Distribuição da frequência de itens relacionados a percepção da aprendizagem (pós-aplicação versão longa)

Os alunos também se consideram capazes de explicar a maioria dos conceitos para outros, indicando inclusive níveis de aprendizagem mais altos (Figura 55). Especialmente em “Fazer protótipos em papel” e “Programar e testar”, que obtiveram apenas uma resposta negativa, indicando serem os mais compreendidos. Já “Realizar análise do contexto e especificar requisitos” não demonstra ser algo tão simples para os alunos em sua maioria. Acredita-se que as respostas entre “Programação” e “Programar e testar” possuem diferença pelo fato de que o tema programação é algo muito mais amplo e complexo, já programar e testar foi um dos temas das aulas aplicadas em que os alunos tiveram bons resultados.

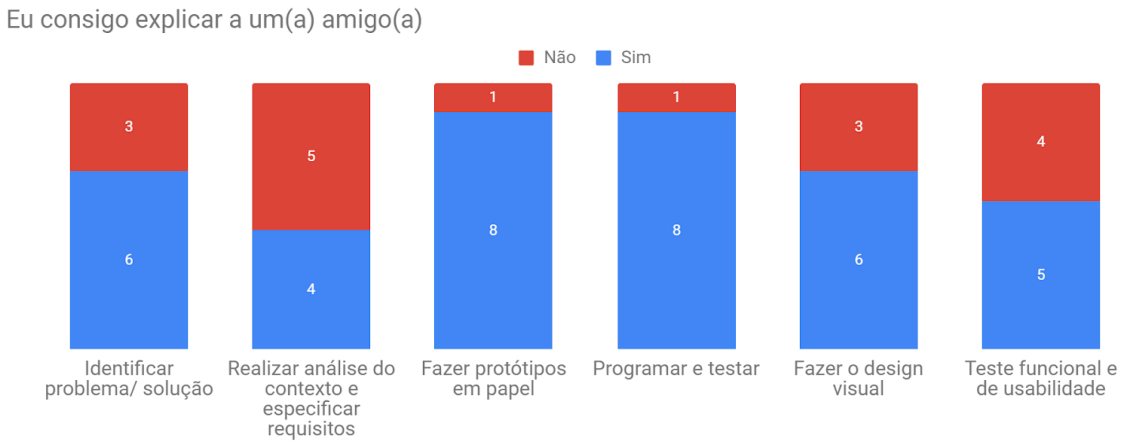


Figura 55: Distribuição da frequência de itens relacionados a confiança na aprendizagem versão longa

Em relação às perguntas objetivas aplicadas ao final do projeto, os alunos demonstraram um grau satisfatório de conhecimento (Figura 56). As questões sobre elementos do design visual e teste funcional tiveram um resultado perfeito, as quais obtiveram todas as respostas corretas. Apesar disto, as perguntas sobre a descrição de uma história de usuário e pra que serve a descrição do usuário obtiveram mais respostas erradas do que corretas, apresentando uma grande dificuldade, até falta de compreensão, por parte dos alunos. A questão para a descrição de uma história de usuário em relação aos *apps* criados (questão 5 das questões aplicadas) obteve uma boa pontuação, com 77,7% de acerto, apresentando que a grande maioria dos alunos sabe aplicar os conceitos relacionados a análise de contexto do *app*.

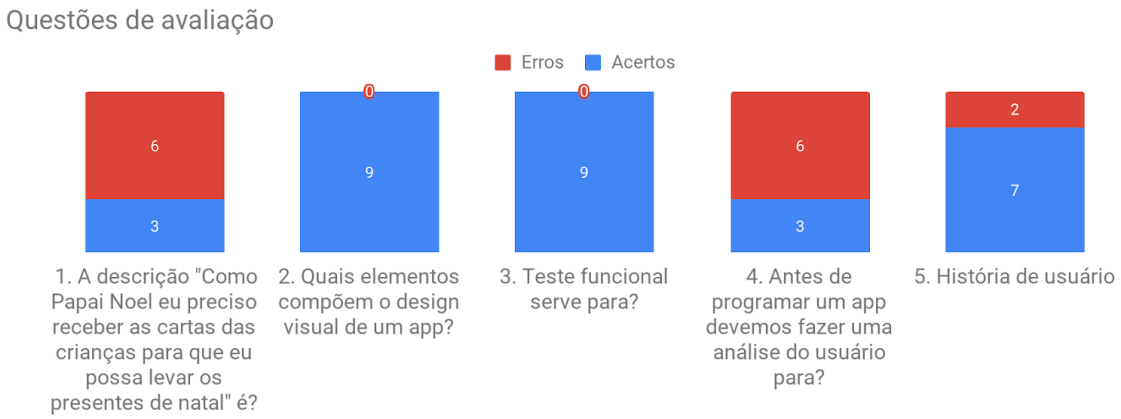


Figura 56: Distribuição da frequência de acertos e erros nas questões objetivas versão longa

A avaliação dos artefatos criados como resultado do processo de desenvolvimentos de *apps* documentados nos *workbooks* por meio da Rubrica 1, apresentou ótimos resultados em relação aos apps produzidos durante a unidade (Tabela 41). A média das notas da análise dos *workbooks* foi 8,4, com destaque para o *app* AcheiOSeuEmprego o qual apresentou melhor detalhamento dos tópicos dos *workbooks* e atingiu a maior nota (9). É possível perceber que o que os alunos mais tiveram dificuldades de relatar sobre seus *app* são os passos de interação, os quais 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*. Na maioria dos casos, os resultados do teste do design de interface não foram completamente descritos e detalhados, faltando uma melhor análise sobre os resultados adquiridos. É possível perceber também que os alunos tiveram alguma dificuldade em descrever os resultados do teste funcional, não conseguindo identificar muito bem quais eram as entradas do *app* em cada situação. Destaque para os tópicos de “persona”, “histórias de usuário”, “design de interface inicial” e “resultados do teste de usabilidade” os quais obtiveram descrições completas em todos os *apps*.

Tabela 41: Avaliação do desempenho na produção dos *workbooks* baseado na Rubrica 1 versão longa

App	**Descrição do problema**	**Descrição da solução**	**Persona**	**Histórias de usuário**	**Passos de interação - casos de uso**	**Resultados de usabilidade**	**Design de interface inicial**	**Resultados do teste do design da interface**
InfoCarvalhal	3	3	3	3	3	2	3	2
HealthyPlants	2	3	3	3	0	0	3	3
AppPhone	3	2	3	3	0	2	3	2
Floripa Praias	3	2	3	3	0	3	3	2
AcheiOSeu Emprego	3	3	3	3	1	3	3	2

A avaliação do design visual/de interface dos *apps* desenvolvidos pelos alunos apresenta um ótimo resultado geral. Como é apresentado na Tabela 42, os aplicativos atingiram pontuações excelentes nos critérios da Rubrica 2 de avaliação do design visual. Os *apps* com maior destaque são o “HealthyPlants” e o “Floripa Praias” os quais obtiveram nota igual ou superior a 9. Estes *apps* apresentaram um ótimo grau de organização e seleção dos elementos do design da interface. Por mais que os outros *apps* também atingiram uma nota alta, acabaram não alcançando um nível tão alto em certos itens da rubrica, justificando sua nota levemente inferior. Um dos tópicos em que os alunos mais tiveram dificuldade para implementar em seus *apps* é a compreensibilidade dos ícones, tendo apenas 2 *apps* atingindo a nota máxima. Outro tópico que apresenta uma certa nota nos *apps* inferior é o de padronização do design em todas as telas do *app*, tendo apenas 1 *app* atingido a nota máxima. Grande destaque para o tamanho dos textos e qualidade das imagens nos *apps* desenvolvidos, os quais apresentam nota máxima em todos eles.

Tabela 42: Avaliação do desempenho no desenvolvimento do design visual do *app* baseado na Rubrica 2 versão longa

	Achei seu emprego	HealthyPlants	App Phone	FloripaPraias	InfoCarvalhal
A 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 a cor do texto e do fundo da tela asseguram a legibilidade	3	4	3	3	4
Os textos apresentam tamanho de fonte agradáveis para leitura	4	4	4	4	2
O alinhamento do texto contribui para leitura e harmonia do design visual	2	4	3	4	4
O tamanho das fontes auxiliam na hierarquia de informações	4	3	4	4	3
As imagens estão de acordo com o tema	4	3	4	4	4
As imagens estão nítidas (não estão pixeladas e distorcidas)	4	4	4	4	4
Os ícones são fáceis de interpretar	4	2	1	4	2
As telas respeitam o mesmo padrão em relação aos elementos de cor, imagem e tipografia	2	4	3	2	3
De modo geral o design visual está agradável e organizado	3	4	3	4	4
Total	41	44	38	43	42
Nota	8,5	9,2	7,9	9,0	8,8

Com a avaliação por meio da rubrica CodeMaster 2.0 é possível perceber que os conceitos apresentados nas aulas foram devidamente aplicados nos *apps* desenvolvidos pelos alunos (Figura 57). Analisando o resultado, compreende-se que os itens os quais obtiveram alguma nota são os itens ensinados durante a aplicação e por este motivo foram incorporados aos *apps*. Em raros casos um *app* foi o único a aplicar um dos itens avaliados, por ter alguma particularidade. As notas apresentadas para cada item demonstram que os alunos conseguiram compreender o que lhes foi ensinado e por este motivo foram capazes de aplicar em seus *apps*. Com realce sobre os demais, o tópico de “eventos” obteve nota máxima para todos os *apps* desenvolvidos.

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
InfoCarvalhal.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**	

Figura 57: Resultados com base na rubrica CodeMaster 2.0 versão longa

A avaliação da estética dos *apps* foi analisado por meio de um survey com 95 participantes avaliando 110 screenshots de IUs de aplicativos do *App Inventor*, incluindo 105 *apps* aleatoriamente escolhidos da Galeria do *App Inventor* (SOLECKI et al., 2019) (Tabela 43). 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 interface, somando as respostas (“feia” = 0 ponto; “mais ou menos” = 0,5 ponto; “bonita” = 1 ponto).

Tabela 43: Pontuação das interfaces dos *apps* desenvolvidos versão longa

Aplicativo	**Pontuação (média)**
Achei o seu emprego	56,25
HealthyPlants	84,5
AppPhone	59,25
FloripaPraias	75,5
InfoCarvalhal	60

Comparada a 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.

Experiência de aprendizagem

Em geral a experiência de aprendizado dos tópicos envolvidos foi caracterizada positivamente (Figura 58). Os alunos demonstraram satisfação significativa , os quais, em sua grande maioria, acharam divertido aprender sobre os conteúdos ministrados. Com destaque positivo está o tópico de “Fazer o design visual”, que foi indicado pelo maior número de alunos como muito divertido. Por outro lado, “Programar e testar” foi o único tópico que foi considerado muito chato por um aluno.

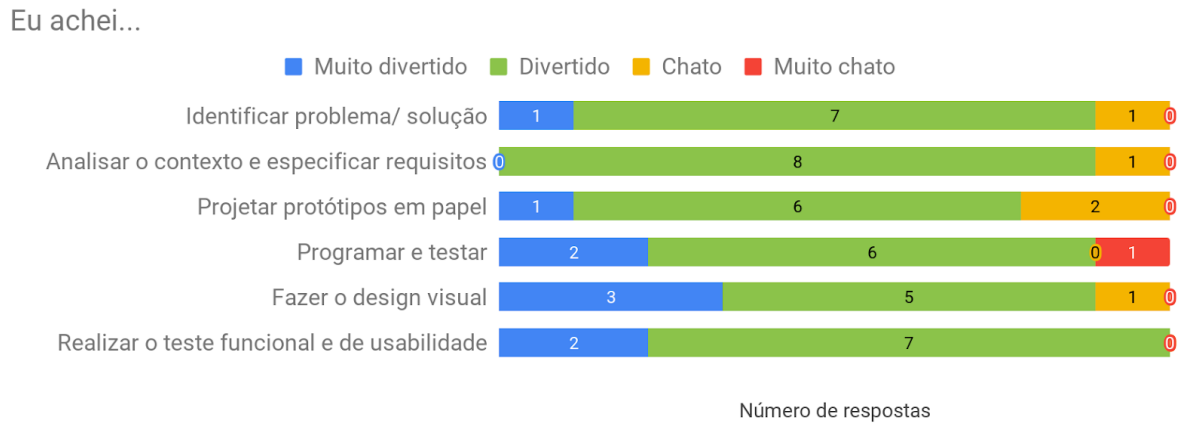


Figura 58: Distribuição da frequência de itens relacionados a diversão versão longa

As respostas dos alunos indicam que não sentiram o tempo passar durante as aulas, com 89% dessas indicando avaliações para rápido ou muito rápido (Figura 59). Apenas um aluno achou que as aulas passaram devagar. Esse resultado indica também que os alunos ficaram focados no conteúdo das aulas não notando a passagem do tempo.

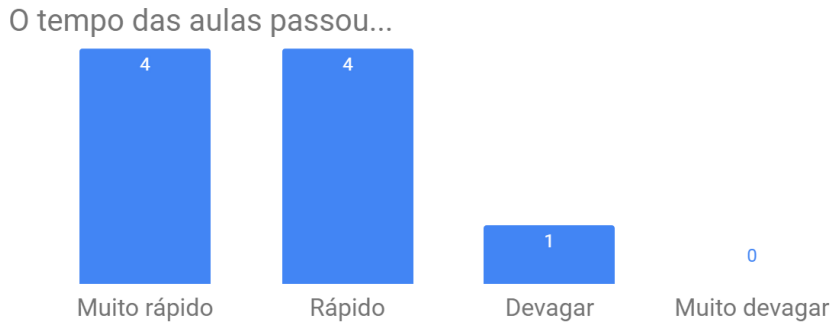


Figura 59: Frequência de itens relacionados ao tempo das aulas versão longa

Todos os alunos demonstraram satisfação em desenvolver os seus *apps* junto com seus colegas atuando como uma dupla (Figura 60). Os alunos demonstraram um maior conforto por seus colegas participarem junto do aprendizado. Da mesma forma ocorreu para o interesse em compartilhar os *apps* desenvolvidos com outras pessoas, obtendo todos os resultados positivos.

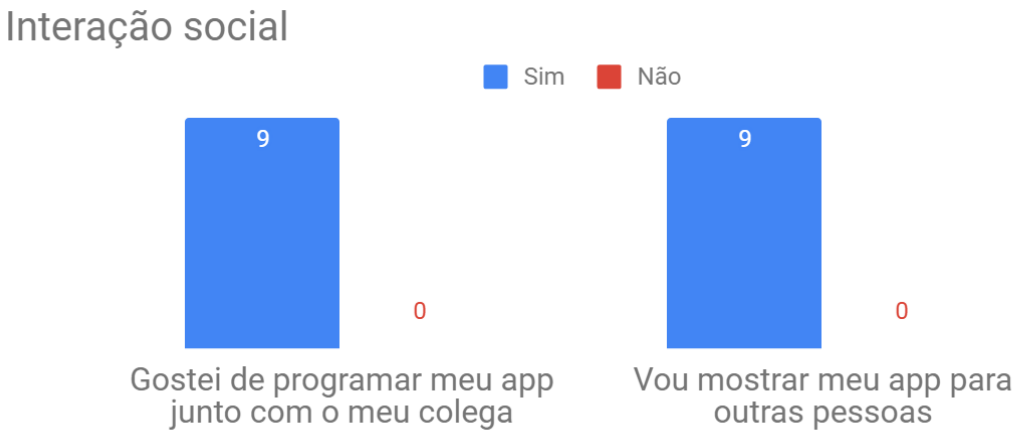


Figura 60: Frequência de itens relacionados a interação social versão longa

Os encontros presenciais foram avaliados de maneira muito positiva pelos alunos (Figura 61). Basicamente todos os alunos consideraram todos os encontros como excelentes ou bons, com exceção de um aluno referente ao encontro voltado ao design visual.

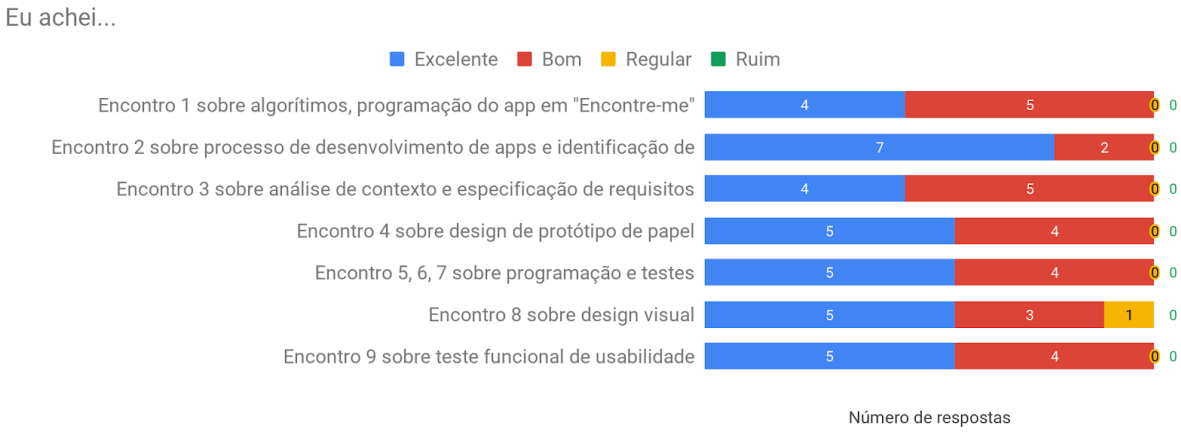


Figura 61: Distribuição da frequência da avaliação da qualidade dos encontros versão longa

Os comentários gerais dos alunos indicam uma grande apreciação por todas as etapas envolvidas na criação de um *app* (Tabela 44). A parte do design do *app* obtém destaque por ser mencionada por quatro alunos quando questionados sobre o que mais gostaram no curso. Também indicam gosto por programar e pela criação de um *app*. As respostas que envolvem o que os alunos desgostaram do curso incluem a parte de buscar informações para apresentar no *app*, construir o protótipo em papel e responder os questionários. Uma boa parte dos alunos indica não houve nada de que não gostassem no curso.

Tabela 44: Comentários qualitativos (números indicando a frequência de citações) versão longa

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

Usabilidade da unidade

Em relação a qualidade das aulas e materiais utilizados pode-se concluir que os alunos avaliam de maneira muito positiva (Figura 62). Os alunos, em sua maioria, acham excelente ou bom o material didático e também a forma como as aulas foram realizadas. Todos consideraram o curso excelente.

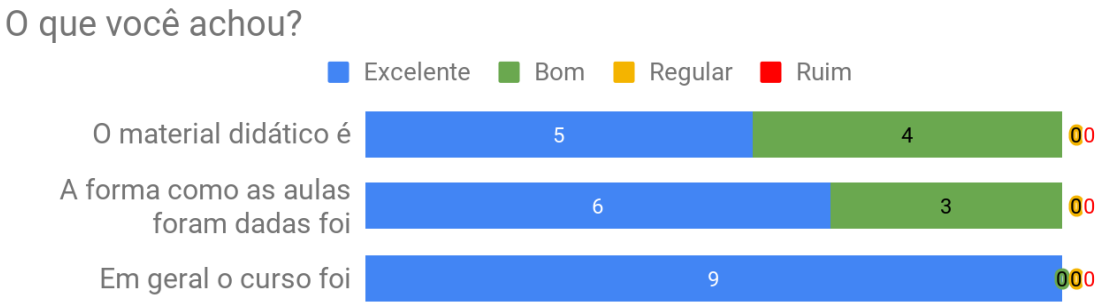


Figura 62: Distribuição da frequência de itens relacionados às aulas e o curso em geral versão longa

Na grande maioria das respostas, os tópicos ensinados foram percebidos como fáceis de aprender (Figura 63). Somente alguns alunos reconheceram uma dificuldade em identificar um problema/solução, programar e testar e fazer o design visual.

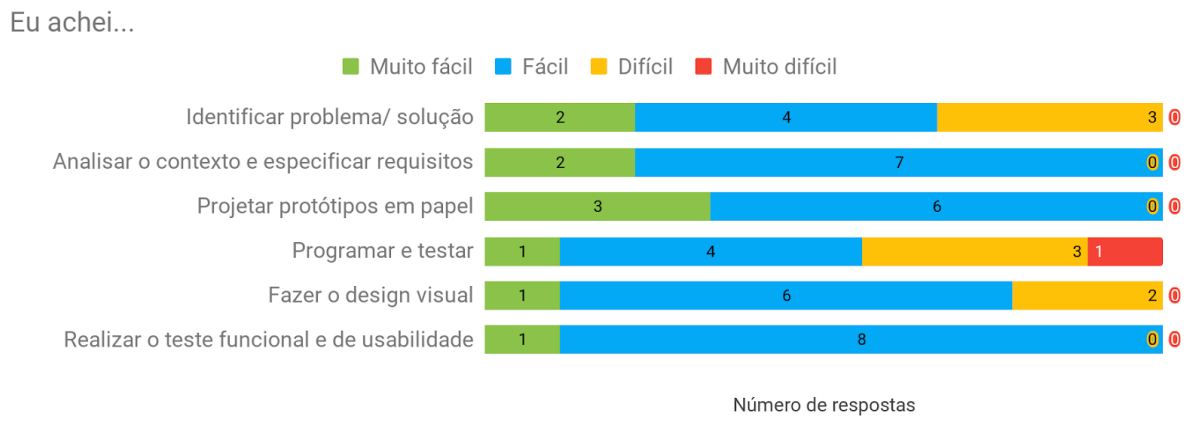


Figura 63: Distribuição da frequência de itens relacionados a sua facilidade de aprendizagem versão longa

Interesse dos alunos referente a aprendizagem de conteúdos da computação

Ao final da unidade instrucional os alunos demonstram um interesse parcial em aprender mais sobre os conteúdos ensinados (Figura 64). Processo de desenvolvimento de *apps* e design visual são os tópicos os quais os alunos mais demonstram interesse. Por outro lado, os outros conteúdos indicam uma divisão entre o interesse nas respostas.

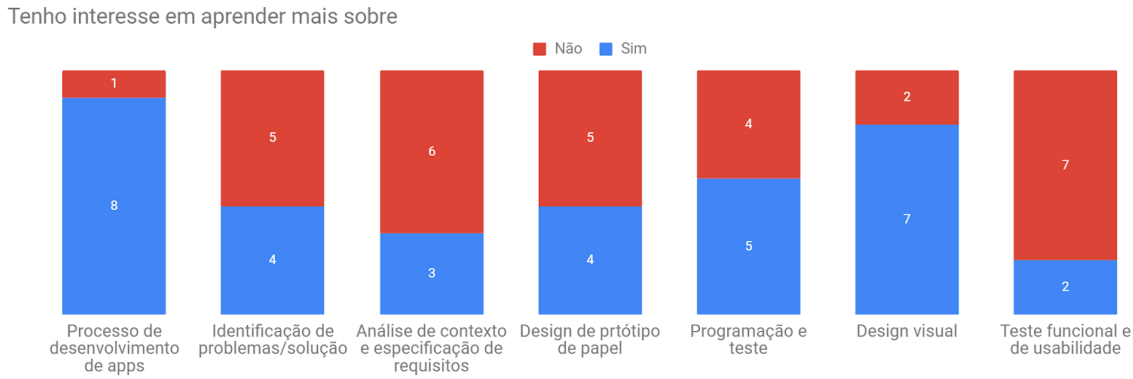


Figura 64: Distribuição da frequência de itens relacionados ao interesse na aprendizagem versão longa

Em relação a consideração de importância dos tópicos ensinados, os alunos responderam em geral que reconhecem o seu valor (Figura 65). Todos reconhecem para a programação e testes e também para o design visual para o desenvolvimento de *apps*. Os resultados demonstram que os alunos também compreenderam a importância do processo de desenvolvimento e da prototipação. Apenas a análise do contexto e especificação dos requisitos foi um tópico considerado não tão importante pela maioria dos alunos.

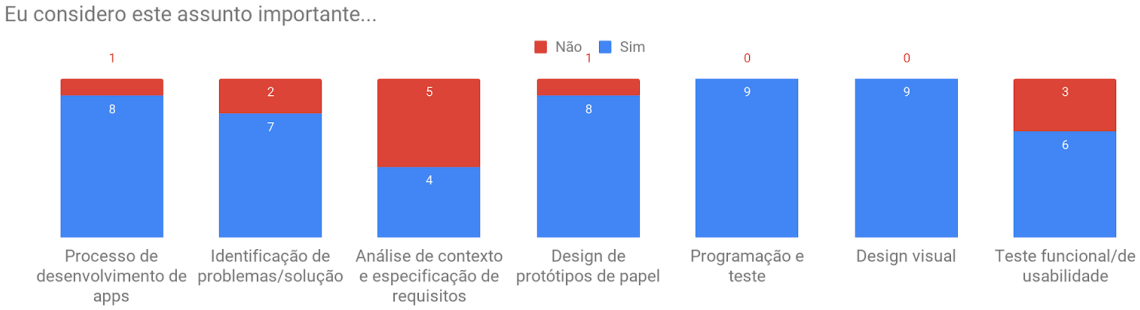


Figura 65: Distribuição da frequência de conteúdo relacionado a sua importância versão longa

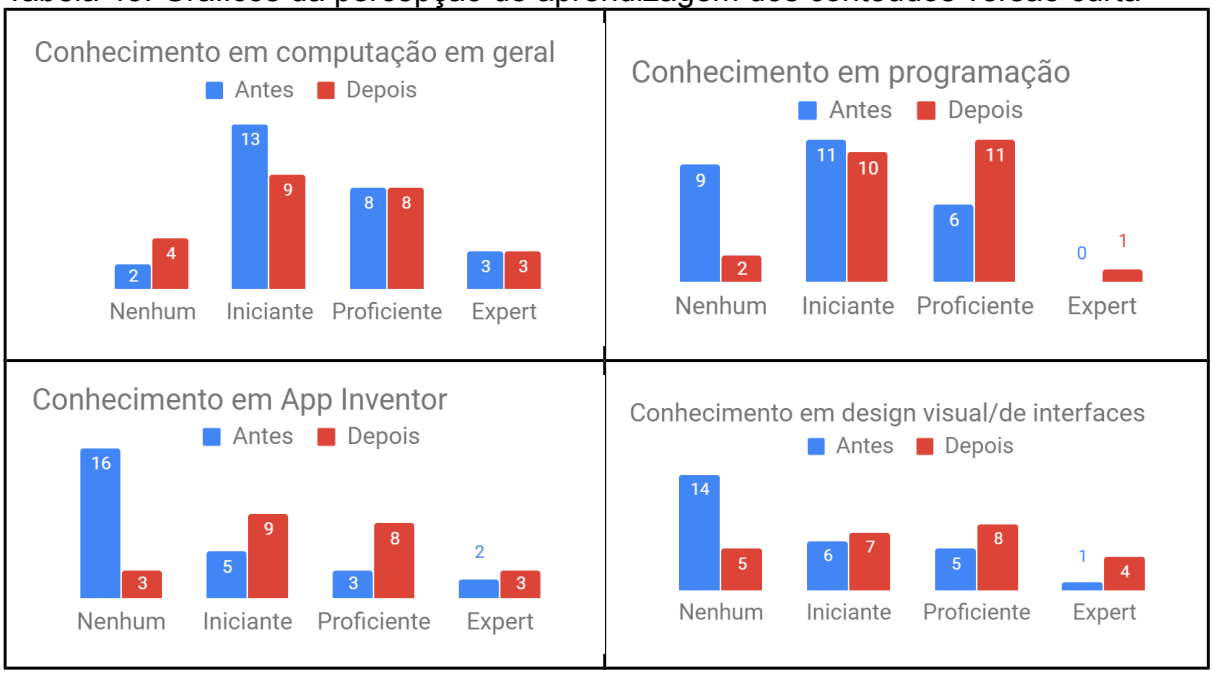
5.3.2. Análise dos Dados Aplicação da Versão Curta - ECOPET 2019-1

De acordo com as perguntas de pesquisas descritas anteriormente, os dados são analisados a fim de respondê-las.

Avaliação da aprendizagem

Em geral, é possível perceber um aumento na percepção de aprendizagem por parte dos alunos (Tabela 45). Na maioria dos casos, que é o esperado, o nível de conhecimento antes da aplicação da unidade é “nenhum”, e por consequência, após a aplicação, este nível se eleva. É o que ocorre com os conteúdos de programação, *App Inventor* e design visual, os quais os alunos percebem de uma maneira visível que o nível de aprendizado aumentou. Apenas com o conteúdo de computação em geral que isto não ocorreu, a percepção geral dos alunos é que o nível de aprendizado se manteve o mesmo antes e depois da unidade. Como um destaque negativo, em todos os casos, pelo menos 2 alunos sempre definiram seus níveis como “nenhum”, mesmo depois da aplicação.

Tabela 45: Gráficos da percepção de aprendizagem dos conteúdos versão curta



Os alunos, em sua maioria, acreditam que não têm a capacidade de explicar para um amigo os conceitos abordados na unidade (Figura 66). Isto apresenta que não possuem confiança no seu nível de aprendizagem ou que os conteúdos foram pouco aplicados na prática. Esta insegurança por parte dos alunos também pode partir por conta da idade deles, em que a média é de 10 anos. Geralmente nesta idade não é esperado de um aluno que ele consiga ensinar outros, ainda mais com conteúdos novos e significativamente complexos.

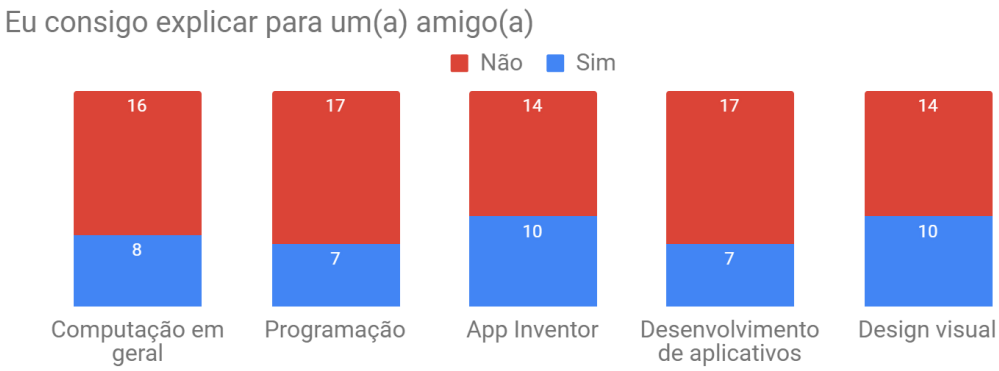


Figura 66: Capacidade de explicar os conteúdos a um(a) amigo(a) versão curta

Avaliação do desempenho

Os alunos apresentam um resultado razoável em relação aos *apps* desenvolvidos, tendo uma boa quantidade de conceitos de design visual que atingiram um certo nível de aceitação de corretude (Figura 67). É possível perceber que os conceitos relacionados a busca e utilização de imagens/ícones e textos são os que obtiveram as melhores pontuações, indicando que são os temas mais compreendidos pelos alunos. Como um destaque, a interpretação dos ícones adicionados é algo que foi totalmente atingido em todos os *apps*, somando a pontuação máxima. Por outro lado, os conceitos relacionados a organização e hierarquia de cores e padronização do design das telas são os que apresentam o pior resultado. Adicionalmente, a maioria dos *apps* não atinge um nível satisfatório do design de interface, sinalizando um resultado pobre em relação aos *apps* desenvolvidos nesta aplicação.

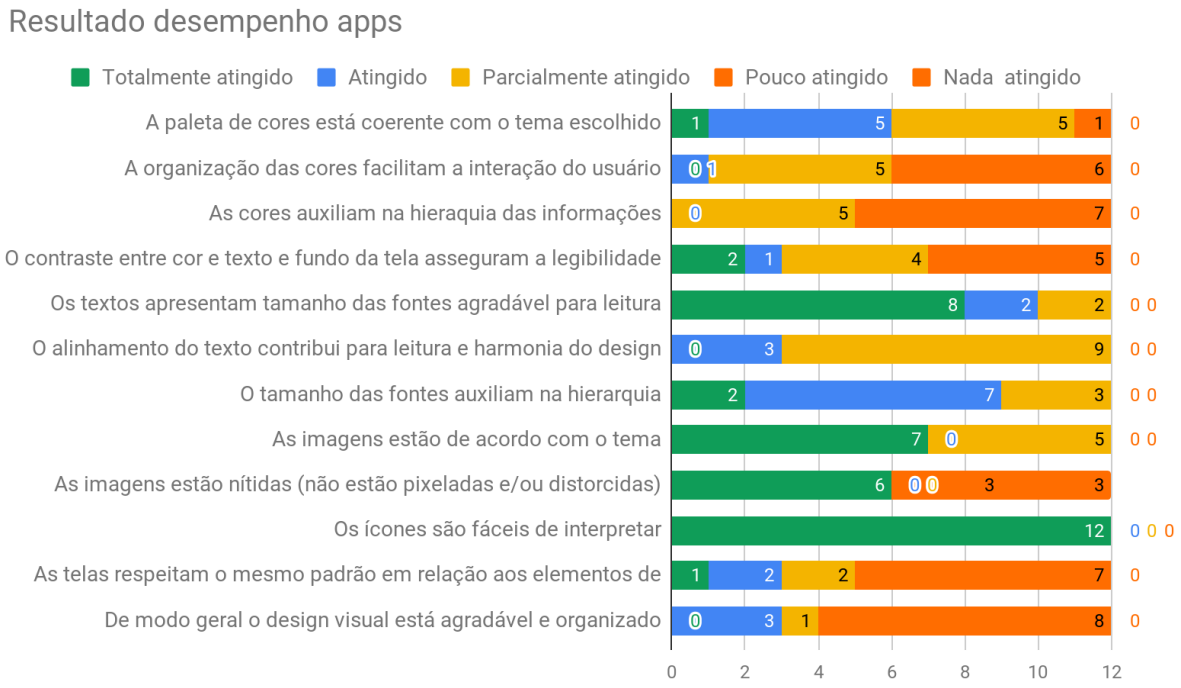


Figura 67: Resultado do desempenho dos *apps* criados versão curta

Os alunos acreditam que ter aprendido os conteúdos da unidade foi algo divertido (Figura 68). De forma geral, os alunos avaliaram a experiência de aprendizado como muito divertida e divertida, apresentando um ótimo resultado em relação a este tópico da avaliação. Poucos alunos apresentam 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* são os tópicos mais rejeitados pelos alunos, enquanto programar e testar e desenvolver o *app* com um colega foram os mais bem avaliados.

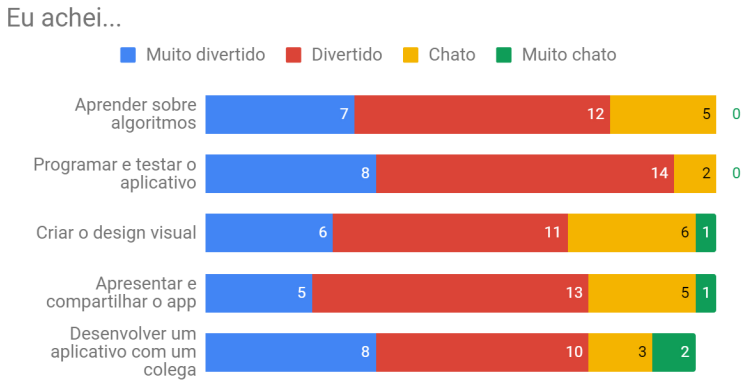


Figura 68: Experiência na unidade versão curta

Pela análise da percepção da passagem de tempo, é possível perceber que os alunos mantiveram a atenção voltada aos conteúdos durante as aulas, de modo que a grande maioria indica que as aulas passaram rápido (Figura 69). Além da atenção, estes dados apresentam que os alunos estavam gostando das aulas e se interessando pelos conteúdos ministrados.

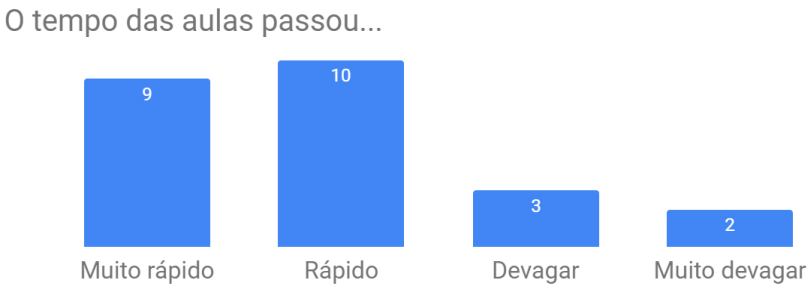


Figura 69: Percepção do tempo na unidade versão curta

Em geral, os alunos apresentam um interesse em compartilhar o *app* desenvolvido com outras pessoas, indicando uma apreciação pelo que desenvolveram (Figura 70). Apenas um dos alunos não gostou de construir o *app* junto com um colega, apresentando algum desconforto em relação ao desenvolvimento em equipe.

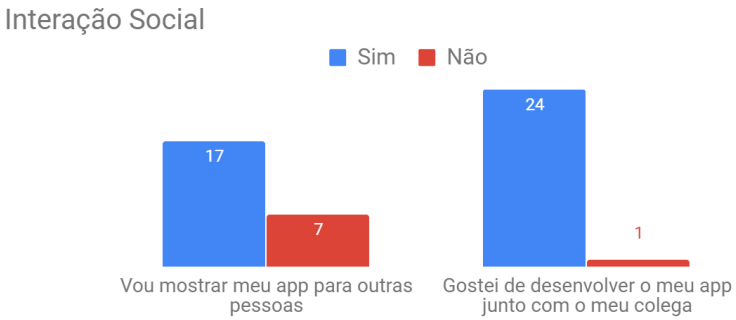


Figura 70: Interação social na unidade versão curta

Os resultados em relação a qualidade das aulas foi muito satisfatório, apresentando a maioria das respostas para bom ou excelente (Figura 71). As únicas aulas que obtiveram algum resultado mais negativo foram as duas últimas que tratam do compartilhamento e apresentação do *app*. Acredita-se que este resultado se dá pela característica dos alunos em terem vergonha de apresentar e compartilhar o que desenvolveram, muito por conta da idade. Muitos demonstram interesse em compartilhar seus *apps*, mas a realização de fato disto não é algo natural para muitos.

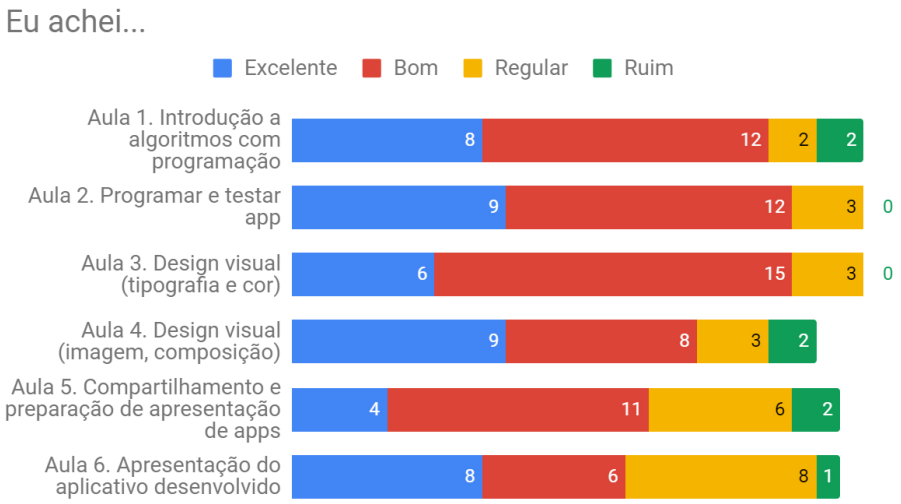


Figura 71: Qualidade das aulas na unidade versão curta

O que os alunos mais apreciaram no curso de desenvolvimento do *app* foi buscar e trabalhar com imagens a fim de colocá-las no *app* (Tabela 46). Também houve destaque para a personalização do *app* em termos de cores, ícones, etc. Alguns alunos também indicam que gostaram de tudo no curso. Por outro lado, os alunos também indicam que trabalhar com fotos foi uma das coisas que menos gostaram no curso. Este resultado demonstra que este tópico é algo bem dividido entre os alunos nesta turma. Outros assuntos que apresentaram desgosto pelos alunos foi escrever os textos do *app*, explicações demoradas e os slides. Uma boa quantidade de alunos demonstra que não há algo o qual não gostou, porém 2 alunos indicam que não gostaram de tudo do curso.

Tabela 46: Respostas abertas sobre o curso em geral versão curta

O que eu **mais** gostei no curso de fazer um *app* foi?	Colocar e tirar fotos (7) Personalizar o *app* (cores, legendas e ícones) (4) Programação (1) Bom (1) Tudo (5) Nada (1) Explicação dos professores (1) Escrita (1) Inventar o *app* (1) Conversar (1)
O que eu **menos** gostei no curso de fazer um *app* foi?	Escrever (4) Tirar a foto, fazer os textos e das explicações demoradas (1)

	Fazer com um colega (1) Nada (6) Tudo (2) Foto (5) Tirar a foto (1) Edita a foto com eu e meus amigos (1) Os slides (1)

Usabilidade da unidade

A qualidade do material didático teve uma avaliação satisfatória pelos alunos, que indicam, de maneira geral, como bom (Figura 72). Entretanto, poucos alunos avaliaram como excelente, sinalizando que melhorias podem ser feitas. Em relação às aulas, o resultado foi muito bom. A maioria dos alunos avalia as aulas como excelentes ou boas, apenas poucos alunos não concordam com os demais. Destaque para a maneira em que as aulas foram dadas, que não apresenta nenhuma avaliação como ruim.

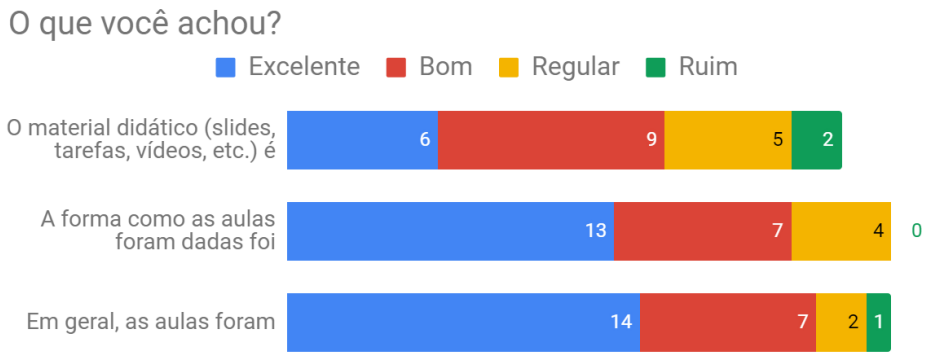


Figura 72: Qualidade em geral do curso na unidade versão curta

Os alunos indicam um alto grau de facilidade em relação aos conteúdos apresentados em sala, indicando confiança no que estavam desenvolvendo e que conseguiram compreender realmente o que foi ensinado (Figura 73). Os dois assuntos considerados mais difíceis são algoritmos e como compartilhar o *app*, os quais apresentam menos avaliações como muito fácil e mais para difícil. Desenvolver um *app* com um colega e programar e testar o *app* foram consideradas as partes menos complicadas do curso.

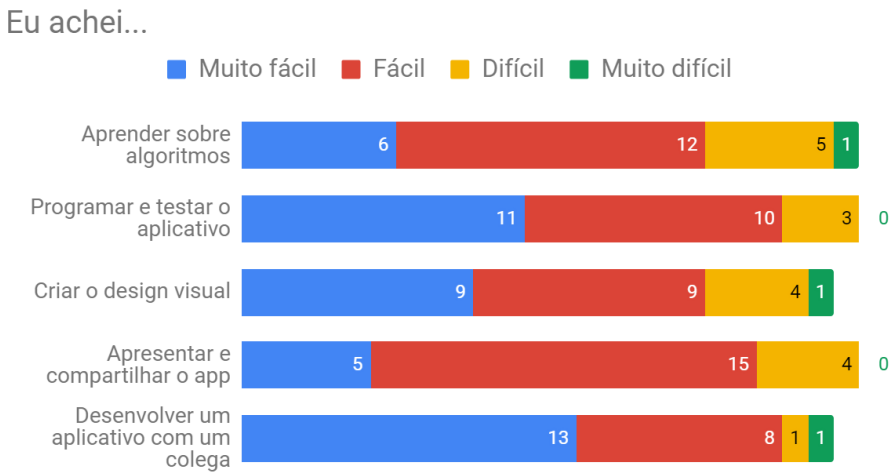


Figura 73: Facilidade dos conteúdos na unidade versão curta

Interesse dos alunos referente a aprendizagem de conteúdos da computação

De forma geral, os alunos ficaram bem divididos em relação ao interesse em aprender mais sobre os conteúdos ministrados em aula (Figura 74). Apenas computação em geral levanta um interesse maior nos alunos após a participação no curso. Os outros assuntos estão praticamente empatados em respostas para sim e não.

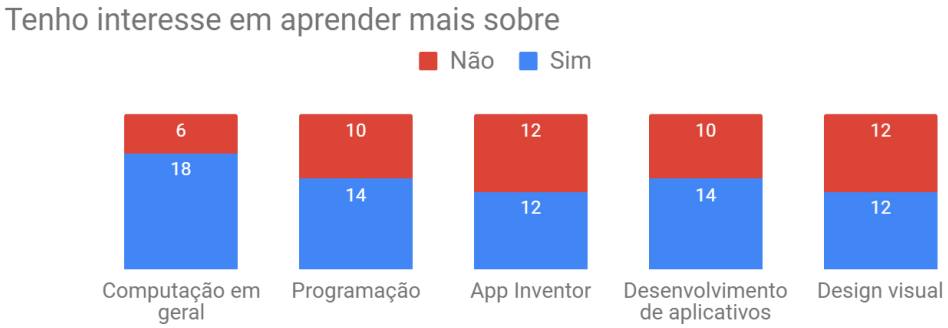


Figura 74: Interesse em aprender mais na unidade versão curta

A maioria dos alunos consegue compreender a importância dos assuntos envolvidos no curso (Figura 75). O assunto que os alunos mais consideram importante é a apresentação do *app*, enxergando que uma boa apresentação faz um *app* se destacar entre os demais.

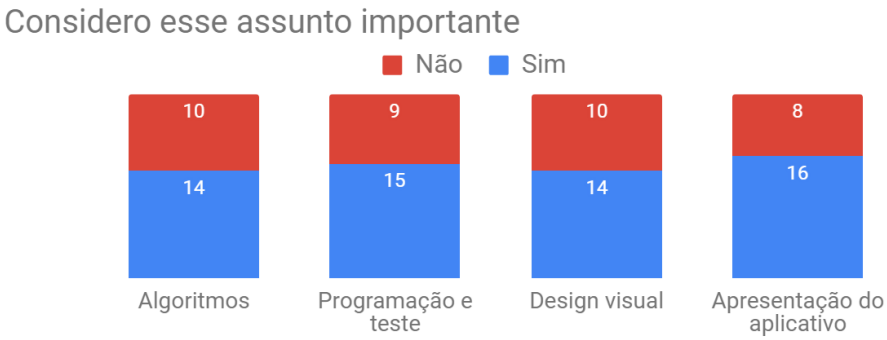


Figura 75: Consideração de importância na unidade versão curta

5.3.3. Discussão

Os resultados da análise dos dados coletados durante as duas aplicações indicam uma avaliação positiva da unidade instrucional. Em ambas, os alunos seguiram todo o processo de desenvolvimento proposto e obtiveram um acréscimo em seu aprendizado e um ótimo resultado nos *apps* desenvolvidos. Foi notável que a maioria dos alunos estava motivada e interessada em aprender mais sobre os conceitos de computação, o que acarretou em uma grande experiência para eles (h1).

A percepção do aumento do aprendizado foi algo notório pela grande maioria dos alunos. Os resultados indicam, para as duas aplicações, que antes e depois dos cursos há uma significativa diferença na aprendizagem percebida pelos alunos. Em praticamente todos os conceitos ensinados, muitos alunos demonstram ter passado do nível “nenhum” para “iniciante” e “proficiente”, apresentando que um aumento em seu nível de conhecimentos nos assuntos. Ter havido alunos que disseram ter nível “expert” em certos conteúdos, faz crer que os resultados foram ainda melhor do que o esperado e que estes alunos têm confiança de que o que aprenderam pode ser fielmente reproduzido por eles. A capacidade de explicar os conteúdos para outras pessoas foi um pouco controverso entre as duas aplicações.

Enquanto na primeira aplicação os alunos, em geral, se sentem capacitados a explicar a maioria dos conteúdos, na segunda, os participantes não demonstram habilidade e segurança para ensinar os conteúdos ensinados. Uma das respostas

para esta diferença é que os alunos na primeira aplicação tiveram mais tempo para aprender os conteúdos e também um contato mais direto com os instrutores, por serem em menor quantidade. Também, na segunda aplicação os alunos eram, aproximadamente, 4 anos mais novos e, tipicamente, não é esperado que alunos nessa idade tenham a habilidade de transferir conhecimentos, ainda mais algo tão novo e complexo para eles.

O desempenho dos alunos na primeira aplicação foi algo que chamou muito a atenção, superando expectativas pela qualidade dos *apps* criados. Este resultado demonstra que a utilização de um processo de desenvolvimento de *apps* bem definido, juntamente aos conceitos de *design thinking*, permite construir aplicativos úteis e com uma excelente interface com o usuário. Em comparação com outros aplicativos na galeria do *App Inventor*, os *apps* desenvolvidos pelos alunos se destacaram em termos do design visual apresentado. Pela avaliação dos artefatos desenvolvidos com base nos *workbooks*, é possível perceber que os alunos conseguiram compreender e produzir as etapas necessárias no processo de desenvolvimento de uma maneira adequada, comprovando a qualidade do resultado final. Em relação aos conceitos de engenharia de *software* como histórias de usuário e testes funcionais os alunos demonstram um significativo grau de conhecimento, demonstrando que foram conteúdos bem aceitos por eles. Já os conteúdos envolvendo casos de uso e funcionamento das tarefas dos *apps*, os alunos tiveram grande dificuldade para aplicar e mesmo com certa ajuda os resultados não foram satisfatórios.

Na segunda aplicação, o desempenho dos alunos em relação aos *apps* desenvolvidos não foi tão satisfatória quanto na primeira. Os alunos demonstraram mais dificuldade em aplicar os conteúdos estudados em sala e o resultado foi percebido nos *apps* construídos. Um dos fatores que influenciou este resultado negativo é a quantidade de alunos em relação ao tamanho da sala. Pelo fato da sala estar mais cheia, para prender a atenção dos alunos é muito mais difícil e distrações afetam o aprendizado dos conteúdos ensinados. Outro fato que fez a diferença é a idade dos alunos, diversos fatores podem ser afetados por causa disso. O interesse dos alunos se torna mais limitado, a capacidade de prestar atenção é afetada, a experiência em resolver problemas e a preferência pela diversão em vez de um bom

resultado. Além da idade, cerca de 3 alunos ainda não eram alfabetizados, o que dificulta as aulas em relação a produção de textos para colocar no *app*, apresentação, entre outras. Como o tempo geral do curso também foi reduzido em relação à primeira aplicação, as aulas tiveram que ser comprimidas ou não ensinadas, justificando os resultados inferiores. Outro fator relevante é o *layout* da sala informatizada, que para uma maior quantidade de alunos dificultou acompanhar a apresentação dos materiais didáticos durante as aulas.

A participação nas aplicações da unidade instrucional se mostrou algo muito prazeroso para os alunos, que acreditam ter sido uma experiência divertida (h3). Este fato demonstra que as aulas foram aplicadas de uma forma agradável e o nível de ensino dos conteúdos foi adequado a maioria dos alunos. A percepção do tempo durante as aulas foi outro parâmetro que indica a boa experiência do curso. Boa parte dos estudantes aponta que não sentiu a passagem do tempo durante o ensino dos conteúdos em sala, o que demonstra que eles voltaram seu foco na instrução e estavam interessados no aprendizado dos conceitos apresentados. Outro ponto que chama atenção é a apreciação pelo desenvolvimento dos *apps* com um colega, apresentada pela grande fração dos alunos. O desenvolvimento em equipe estimula a discussão entre os alunos, sempre buscando a melhor maneira de resolver os problemas ou na hora de decidir decisões de projeto. Além disto, a construção do *app* fica sujeita a análise por duas pessoas, o que leva a um resultado com mais qualidade e menos defeitos. Os comentários qualitativos escritos pelos alunos apontam uma boa avaliação de todas as partes do curso. Houve destaque para os conceitos do design visual, que estiveram presentes em diversos comentários dos alunos em ambas as aplicações. Estes dados demonstram que o ensino do desenvolvimento de *apps* utilizando o *design thinking* se apresenta com uma boa solução para esta faixa etária dos alunos, prendendo sua atenção e sendo algo agradável e gratificante.

A utilização da ferramenta *App Inventor* e a forma como as aulas foram planejadas auxiliou para que os alunos avaliassem o aprendizado dos conteúdos como fáceis (h4). Por mais que a maioria dos conceitos ensinados tenham um grau de complexidade elevado, os participantes das aplicações sentiram um nível adequado de aprendizagem, caracterizando com um certo grau de facilidade.

A adoção de um processo de desenvolvimento de *apps* com *design thinking* envolvendo conceitos design visual serviu como incentivo para que os alunos se interessassem mais pelos conteúdos, tornando-os mais criativos e participativos (h2). Também permitiu a construção de diversos *apps* úteis que resolvem problemas da sociedade, permitindo que os alunos se envolvam na mudança do mundo em que vivem de maneira prática. A percepção da importância dos conteúdos ensinados também comprova que os alunos compreendem que no futuro essa aprendizagem se tornará necessária a diferença no mundo em que vivem.

Alguns problemas em relação ao espaço físico das salas de aulas foram percebidos, concluindo-se que um preparo antecipado é essencial para um bom andamento do curso. A limitação da ferramenta *App Inventor* em relação à adequação ao *Material Design* dificulta de certa forma a aprendizagem, sendo necessárias algumas adaptações para que o resultado se torne adequado às suas diretrizes. Outro fato em que o *App Inventor* não auxilia no processo é prover suporte para a produção dos artefatos de engenharia de *software*. A quantidade de alunos é algo que interfere muito no resultado e controle da aplicação. Na segunda aplicação foi possível perceber que houve mais dificuldade em prender a atenção dos alunos e mantê-los motivados. Pelo fato de a sala estar mais cheia acaba atrapalhando nestes critérios e o progresso da aprendizagem se torna mais lento e conturbado. O tempo também foi um fator que demonstrou ser muito importante. Quanto mais tempo houver para ensinar com cuidado os conteúdos, possivelmente melhor será o resultado. Como os conceitos ensinados são novos para a grande maioria dos alunos, ter mais tempo para realmente compreenderem é fundamental, fora que possibilita maior interação entre alunos e instrutores.

A potencialidade positiva de ensinar o processo de desenvolvimento de *apps* e o design de interfaces como parte da educação em computação vai muito além do aprendizado de competências relacionadas somente a estas áreas. Enquanto os alunos desenvolvem aplicativos para resolver um problema do mundo real eles também desenvolvem habilidades essenciais para o século XXI e para seus próprios futuros em suas carreiras. Dentre as capacidades aprimoradas encontram-se a criatividade, trabalho em equipe, resolução de problemas, pensamento crítico, entre diversas outras. Desta forma, o aprendizado dos conteúdos envolvidos na unidade

instrucional proposta fornece soluções e formas para um melhor engajamento com o mundo que permitem que os alunos possam fazer a diferença no dia-a-dia, preparando-os para as habilidades necessárias no século XXI.

6. CONCLUSÃO

A finalidade deste projeto de conclusão de curso foi o desenvolvimento de uma unidade instrucional para ensinar o desenvolvimento de aplicativos para alunos do ensino fundamental de escolas brasileiras com a ferramenta *App Inventor*, envolvendo conceitos de engenharia de *software* e design de interfaces. Neste cenário, foi levantada a fundamentação teórica sobre os assuntos abordados, de modo a contextualizar o trabalho (O1). A análise do estado da arte foi realizada buscando unidades instrucionais que visam ensinar engenharia de *software* e design de interfaces separadamente, permitindo perceber a carência de unidades com este objetivo e a falta de informações sobre as existentes (O3). Foi desenvolvida a unidade instrucional (O4) primeiramente realizando uma análise do contexto sobre o perfil dos aprendizes e das escolas (O2). Com o resultado desta análise foi possível estabelecer os objetivos de aprendizagem e como consequência, a criação de um plano de ensino para a unidade proposta. Seguindo o processo de design instrucional, a unidade instrucional foi então desenvolvida e produzidos os materiais instrucionais (O5).

A unidade foi aplicada (O6) em dois momentos diferentes, em uma versão longa/completa e outra mais curta. A versão longa envolveu 10 alunos do ensino fundamental, com idade entre 13 e 15 anos, na produção de um aplicativo funcional. A avaliação desta aplicação permitiu notar que as expectativas foram cumpridas e os resultados foram muito além do esperado. A maioria dos conteúdos ensinados foram bem aceitos pelos alunos e as competências e o interesse dos alunos tiveram um acréscimo notável. Na aplicação mais curta, foram envolvidos 28 alunos do ensino fundamental, com idade entre 9 e 11 anos, no aprimoramento de um aplicativo já desenvolvido focando mais na construção do design de interfaces de maneira interdisciplinar com a disciplina de Ciências com o tema de sustentabilidade. A avaliação desta aplicação mostrou que o tempo mais curto, a idade e a quantidade

de alunos tiveram forte influência para que os resultados não fossem tão bons quanto na primeira aplicação. Entretanto, em geral, os alunos se mostraram interessados em aprender mais sobre os conteúdos e acreditam que foi uma experiência divertida ter participado do curso.

Como resultado deste trabalho, é disponibilizada uma unidade instrucional para ensinar o processo de desenvolvimento de *apps* por meio do desenvolvimento de aplicativos abrangendo conceitos de engenharia de *software* e design de interfaces. Indica-se que esta unidade seja aplicada para alunos a partir do ensino fundamental, com idade acima de 14 anos de idade com ou sem experiência prévia em computação/programação. A ideia geral desta unidade é capacitar os alunos à produção de *apps* utilizando um processo de desenvolvimento adequado, desenvolvendo habilidades de programação e design. Sua aplicação pode ser inserida tanto de forma interdisciplinar quanto extracurricular e, até mesmo, a distância (via *Moodle*, por exemplo).

Para trabalhos futuros, encontram-se alguns conteúdos pertinentes em relação a este trabalho. Um deles é o aprimorando da ferramenta *App Inventor* em termos do suporte ao *Material Design* (cores, layouts, etc.) e também em relação a criação dos artefatos do processo de desenvolvimento de *apps*, como uma maneira de documentação integrada ao *app*. O conteúdo de casos de uso e descrição da execução das tarefas poderia ser melhorado e encontrado uma forma mais natural para que os alunos compreendam e possam aplicar da maneira correta.

REFERÊNCIAS

ABNT. **NBR 9241-11 Requisitos Ergonômicos para Trabalho de Escritórios com Computadores Parte 11 – Orientações sobre Usabilidade**. Disponível em: <http://www.abntcatalogo.com.br/norma.aspx?ID=86090>. Acesso em: out. 2017.

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

AIGA. **Design thinking - An introduction to design thinking**. Disponível em: <https://www.aiga.org/aiga/content/events-and-competitions/competitions/graphic-design-training-curriculum-for-high-school-teachers/>. Acesso em: abr. 2019.

AHO, A. V. Computation and Computational Thinking. **The Computer Journal,** v. 55, n. 7, p. 832-835, jul. 2012.

ALVES, N. C. CodeMaster: **Um Modelo de Avaliação do Pensamento Computacional na Educação Básica Através da Análise de Código de Linguagem de Programação Visual**. Disponível em: <http://www.gqs.ufsc.br/wp-content/uploads/2011/11/dissertacao-vfpub.pdf>. Acesso em: 12 de mai. 2019.

BASILI, V; CALDIERA, G; ROMBACH, H. D.; **The Goal Question Metric Approach**. Disponível em: <http://www.cs.umd.edu/~mvz/handouts/gqm.pdf>. Acesso em: maio de 2019.

BOURQUE, P.; FAIRLEY, R. **Guide to the Software Engineering Body of Knowledge, Version 3.0**. Disponível em: <https://www.computer.org/web/swebok/v3>. Acesso em: 08 out. 2017.

BRANCH, R. M. **Instructional Design: The ADDIE Approach**. New York, New York, USA, Springer Science & Business Media, 2009.

BRUNER, J. S. **Toward a theory of instruction.** Harvard University Press, 1966, p. 176.

BUSHELL, D. **Classroom behavior: A little book for teachers**. New Jersey: Prentice-Hall, 1973, p.131.

CALDEIRA, A. C. M. **Avaliação da aprendizagem em meios digitais: novos contextos.** Disponível em: <http://www.abed.org.br/congresso2004/por/htm/033-TC-A4.htm>. Acesso em: nov. 2017.

CAMBRAIA, A. C.; SCAICO, P. D. Os desafios da Educação em Computação no Brasil: um relato de experiências com Projetos PIBID no Sul e Nordeste do país. **Revista Espaço Acadêmico,** n. 194, 2017.

CHEN, P.; HUANG, R. Design thinking in App inventor game design and development: A case study. In: **Anais do IEEE 17th International Conference on Advanced Learning Technologies (ICALT)**, Timisoara, Romania, 2017.

CSTA. **K-12 Computer Science Framework**. Disponível em: <http://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf>. Acesso em: 10 set. 2017.

CSTA. **K-12 Computer Science Standards** . Disponível em: <https://www.csteachers.org/page/standards>. Acesso em: dez. 2018.

DANIEL, G. T. **Design de unidade instrucional de desenvolvimento de aplicativos para o ensino fundamental**, 2016, p. 84, Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis.

D'ANGELO, P. **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: jun. 2018.

DEMETRIO, M. F. **Desenvolvimento de um analisador e avaliador de código de App Inventor para ensino de computação**, 2017, p. 126, Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis.

DICK, W.; CAREY L.; CAREY J. O. **The Systematic Design of Instruction**. Pearson Education, 2014, p. 448.

DREYFUS S. E.; DREYFUS, H. L. **A Five-Stage Model of the Mental Activities Involved in DIrected Skill Aquisition**. Disponível em: <http://www.dtic.mil/dtic/tr/fulltext/u2/a084551.pdf>. Acesso em: out. 2018.

DRISCOLL M. P. **Psychology of Learning for Instruction.** Allyn & Bacon, 2000.

FEIJÓ, V. C.; BALDESSAR, M. J.; VIEIRA M. L. H. Elementos De Design Para Interface De Apps Em Smartphones: O Iphone 4s. In: **Anais do XXI Simpósio Nacional de Geometria Descritiva**, Florianópolis, Santa Catarina, 2013.

FENILI, R. M. et al. Repensando a Avaliação da Aprendizagem. **Revista Eletrônica de Enfermagem,** v. 4, n. 2, 2002.

FERRAZ A. P. C. M..; BELHOT R. V. Taxonomia de Bloom: revisão teórica e apresentação das adequações do instrumento para definição de objetivos instrucionais. **Gest. Prod.**, São Carlos, v. 17, n. 2, p. 421-431, 2010.

FGV. **27º Pesquisa Nacional do Uso de TI**. Disponível em: <http://eaesp.fgvsp.br/sites/eaesp.fgvsp.br/files/pesti2016gvciappt.pdf>. Acesso em: 06 out. 2017.

FLÔRES, M. L. P.; TAROUCO, L. M. R.; REATEGUI, E. B. **Orientações para o sequenciamento das instruções em um objeto de aprendizagem.** Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/22976/000732392.pdf?sequence=1>. Acesso em: nov. 2017.

FRACTUS. **Bloom's Taxonomy Verbs – Free Classroom Chart.** Disponível em: <https://www.fractuslearning.com/2016/01/25/blooms-taxonomy-verbs-free-chart/>. Acesso em: fev. 2018.

FREEDMAN, D.; PISANI, R.; PURVES, R. **Statistics**. New York: W. W. Norton & Company, 2007.

FREITAS, S. R. P. C. O Processo de Ensino e Aprendizagem: A Importância da DIdática. In: **Anais do VIII Fórum Internacional de Pedagogia**, Imperatriz, Maranhão, 2016.

GARRETT, J. J. **The Elements of User Experience: User-Centered Design for the Web and Beyond**. New Riders, 2010.

GLICKMAN, C. Pretending Not to Know What We Know. **Educational Leadership**, v. 48, n. 8, p. 4-10, maio 1991.

GOMES, P. **Conheça as competências para o século 21.** Disponível em: <http://porvir.org/conheca-competencias-para-seculo-21/>. Acesso em: fev. 2019.

GOOGLE. **Material Design**. Disponível em: <https://material.io/guidelines/material-design/introduction.html>. Acesso em: 20 set. 2017.

HUMPHREY, W. **Managing the Software Process**. Addison-Wesley, 1989. 494 p.

IMASTERS. **Uso do Android cresce no Brasil**. Disponível em: <https://imasters.com.br/noticia/uso-do-android-cresce-no-brasil/>. Acesso em: fev. 2018.

INEP. **Censo da Educação Superior 2016**. Disponível em: <https://abmes.org.br/arquivos/documentos/censo_superior_tabelas.pdf>. Acesso em: jun. 2018.

ISO. **ISO 13407 Human-Centred Design process for interactive systems.** Disponível em: <https://www.iso.org/standard/21197.html>. Acesso em: jun. 2018.

ISO. **ISO 9241-220 Ergonomics of human-system interaction – Part 220: Processes for enabling, executing and assessing human-centered design within organizations.** Disponível em: <https://www.iso.org/obp/ui/#iso:std:iso:9241:-220:ed-1:v1:en>. Acesso em: mai. 2019.

KEESEE, G. S. **Instructional Approaches.** Disponível em: <http://teachinglearningresources.pbworks.com/w/page/19919560/Instructional%20Approaches>. Acesso em: out. 2017.

KIRKPATRICK, D. L.; KIRKPATRICK, J. D. **Evaluating training programs: the four levels**. Berrett-Koehler Publishers, 2006, p. 379.

KUBO, O. M.; BOTOMÉ, S. P. **Ensino-Aprendizagem: Uma Interação Entre Dois Processos Comportamentais.** Disponível em: <https://revistas.ufpr.br/psicologia/article/view/3321>. Acesso em: set. 2017.

LEE, I. et al. Computational thinking for Youth in Practice. **ACM Inroads**, v. 2, n. 1, p. 32-37, 2011.

LIBÂNEO, J. C. **Didática**. São Paulo: Cortez, 2006, p. 262

MEC. **Base Nacional Comum Curricular**. Disponível em: <http://basenacionalcomum.mec.gov.br/images/BNCC_EI_EF_110518_versaofinal_site.pdf>. Acesso em: jul. 2019.

MELLO, D. **Pesquisa: 80% da população brasileira entre 9 e 17 anos usam a internet**. Disponível em: <http://agenciabrasil.ebc.com.br/pesquisa-e-

inovacao/noticia/2016-10/pesquisa-80-da-populacao-brasileira-entre-9-e-17-anos-usam>. Acesso em: fev. 2018.

MERRIL, M. D. First principles of instruction. **Educational Technology Research and Development**, v. 50, n. 3, 2002, p. 43–59.

MIT (a). **About us**. Disponível em: <http://appinventor.mit.edu/explore/about-us.html>. Acesso em: 08 out. 2017.

MIT (b). **Beginner Video Tutorials**. Disponível em: <http://appinventor.mit.edu/explore/ai2/beginner-videos.html>. Acesso em: 09 out. 2017.

MIT (c). **News & Events**. Disponível em: <http://appinventor.mit.edu/explore/news-events.html>. Acesso em: 18 out. 2017.

MIT (d). **Magic 8-Ball Tutorial.** Disponível em: <http://appinventor.mit.edu/explore/teach/magic-8-ball.html>. Acesso em: 19 out. 2017.

MIT (e). ***Create apps*!**. Disponível em: <http://ai2.appinventor.mit.edu/>. Acesso em: 20 out. 2017.

NAPOL, I. **Brasil tem o mercado mais competitivo para aplicativos móveis**. Disponível em: <https://www.tecmundo.com.br/apps/105145-brasil-tem-mercado-competitivo-aplicativos-moveis.htm>. Acesso em: fev. 2018.

NORMAN, D**. Human Error? No, Bad Design**. Disponível em: <http://www.jnd.org/dn.mss/human_error_no_bad.html> Acesso em: 19 out. 2017.

NUNES, D. J. Ciência da Computação na Educação Básica. **Anais do XX Workshop sobre Educação em Computação (WEI)**, Curitiba, Paraná, 2012.

P21. Framework for 21st Century Learning. Disponível em: <http://www.battelleforkids.org/networks/p21/frameworks-resources>. Acesso em: 12 de mai. 2019.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: An update. **Information and Software Technology**, v. 64, p. 1-18.

PINHEIRO, F. C.; MISSFELDT FILHO, R.; WANGENHEIM, C. G. **Teaching Software Engineering in Basic Education: Supplementary Material.** Disponível em: <http://www.incod.ufsc.br/wp-content/uploads/2018/05/INCoD-GQS.01.2018.E-Relatorio_Tecnico_vf.pdf>. Acesso em: jun. 2018.

PISA. **PISA 2015 Results**. Disponível em: <http://www.oecd.org/education/pisa-2015-results-volume-iii-9789264273856-en.htm>. Acesso em: mai. 2018.

PMF. **Processo Seletivo De Substitutos - Edital nº 004/2018**. Disponível em: <http://substituto2019.fepese.org.br/?go=download&arquivo=2018_PMF_Substitutos_Ed_04.pdf>. Acesso em jul. 2019.

RALLO, R. **Material Design: aprenda tudo sobre o design do Google!**. Disponível em: <https://marketingdeconteudo.com/material-design/>. Acesso em: fev. 2018

RAZZOUK, R.; SHUTE, V. What Is Design Thinking and Why Is It Important? **Review of Educational Research**, v. 82, n. 3, p. 330-348.

REIS. **Usabilidade: como o design thinking contribui para a UX?** Disponível em: <http://blog.cedrotech.com/usabilidade-como-o-design-thinking-contribui-para-ux/>. Acesso em: jun. 2018.

ROGERS, C. R. **Freedom to Learn.** Merrill, 1969, p. 358.

SANTANA, B. **Brasil é o país cujos usuários de smartphones abrem o maior número de apps por dia, em média.** Disponível em: <https://macmagazine.com.br/2017/05/05/brasil-e-o-pais-cujos-usuarios-de-smartphones-abrem-o-maior-numero-de-apps-por-dia-em-media/>. Acesso em: fev. 2018.

SASKATCHEWAN. **Instructional Approaches: A Framework for Professional Practice.** Disponível em: <https://wikieducator.org/images/e/e2/Instructional-Approaches_Handbook.pdf>. Acesso em: nov. 2017.

SBC. **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: 14 out. 2017.

SBC - **Referenciais de Formação em Computação: Educação Básica.** Disponível em: <http://www.sbc.org.br/files/ComputacaoEducacaoBasica-versaofinal-julho2017.pdf>. Acesso em: mar. 2018.

SCRATCH. **Acerca do Scratch**. Disponível em: <https://scratch.mit.edu/about>. Acesso em: 08 out. 2017.

SICA, C. **Ciência da Computação no Ensino Básico e Médio**. Disponível em <http://www.odiario.com/blogs/carlossica/2011/10/07/ciencia-da-computacao-no-ensino-medio/>. Acesso em: 22 out. 2017.

SHADISH, W. R.; COOK, T. D.; CAMPBELL, D. T. (2002). **Experimental and quasi-experimental designs for generalized causal inference.** New York: Houghton Mifflin Company, 2001, p. 623.

SOARES DE FRANÇA, R. et al. Ensino de Ciência da Computação na Educação Básica: Experiências, Desafios e Possibilidades. In: **Anais do XX Workshop sobre Educação em Computação (WEI)**, Curitiba, Paraná, 2012.

SOLECKI, I. S. et al. Análise da Correlação entre Conformidade com Diretrizes de Design Visual e Estética de Interfaces de Usuário de Aplicativos Android. **XVIII Simpósio Brasileiro sobre Fatores Humanos em Sistemas Computacionais**. Vitória, Espírito Santo, 2019.

SOUZA, M. **O processo de Ensino e Aprendizagem**. Disponível em: <https://www.comportese.com/2012/03/o-processo-de-ensino-e-aprendizagem>. Acesso em: out. 2017.

STATISTA. **Number of available applications in the Google Play Store from December 2009 to December 2017**. Disponível em: <https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/>. Acesso em: fev. 2018.

TECHNOVATION. **We Are Technovation**. Disponível em: <http://technovationchallenge.org/about/>. Acessado em: out. de 2017

TECHNOVATION BRASIL. **Currículo**. Disponível em: <http://www.technovationbrasil.org/curriculo>. Acessado em: out. 2017.

THINK WITH GOOGLE. **Insights para Apps - Como os Consumidores Encontram e Usam os Aplicativos**. Disponível em: <https://www.thinkwithgoogle.com/intl/pt-br/tendencias-de-consumo/apps-marketing-insights/>. Acesso em: jun. 2018.

UNGER, R.; CHANDLER, C. **Guia Para Projetar UX**. Rio de Janeiro: Alta Books, 2010. 288 p.

UOL. **Uso de smartphones cresce 3,5 vezes no Brasil em quatro anos.** Disponível em: <https://tecnologia.uol.com.br/noticias/redacao/2017/02/28/uso-de-smartphones-cresce-35-vezes-no-brasil.htm>. Acesso em: fev. 2018.

UXQB (2018). **CPUX-F Curriculum and Glossary**. Disponível em: <https://uxqb.org/wp-content/uploads/documents/CPUX-F_EN_Curriculum-and-Glossary.pdf>. Acessado em: set. 2018.

VORTIGO. **A Importância Da Usabilidade Em Aplicativos Mobile E Como Melhorá-la**. DIsponível em: <http://blog.vortigo.com.br/a-importancia-da-usabilidade-em-aplicativos-mobile-e-como-melhora-la/>. Acesso em: jun. 2018.

WANGENHEIM, C. G.; ALVES, N. C.; WEBER, A. R. **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: abr. 2018.

WANGENHEIM, C. G.; SHULL, F. To game or not to game? **IEEE Software**, v. 26, n. 2, p. 92-94.

WANGENHEIM, C. G. et al. CodeMaster – Automatic Assessment and Grading of App Inventor and Snap! Programs. **Informatics in Education.** v. 17, n. 1, p. 117–150.

WOHLIN, C. et al. **Experimentation in Software Engineering**. Berlin: Springer-Verlag Berlin Heidelberg, 2012, p. 235.

YIN, R. K. **Case Study Research: Design and Methods.** SAGE, 4 ed., 2009, p. 219.

Anexo I

Tabela I.1. Detalhamento de objetivos de aprendizagem de computação para Ensino Fundamental II (WANGENHEIM; ALVES; WEBER, 2017)

ID	Objetivo de aprendizagem	Conceito	Subconceito	Prática
2-CS-01	Recomendar melhorias para o *design* de dispositivos de computação, com base em uma análise de como os usuários interagem com estes dispositivos	Sistemas de Computação	Dispositivos	Reconhecer e Definir Problemas Computacionais
2-CS-02	Planejar projeto que combinem componentes de *hardware* e *software* para coletar e trocar dados	Sistemas de Computação	*Hardware* e *Software*	Criar Artefatos Computacionais
2-CS-03	Identificar e solucionar de forma sistemática problemas com dispositivos de computação e seus componentes.	Sistemas de Computação	Solução de problemas	Testar e Refinar Artefatos Computacionais
2-NI-04	Modelar o papel dos protocolos na transmissão de dados em redes e na *Internet*	Redes e Internet	Comunicação e Organização da Rede	Desenvolver e Usar Abstrações
2-NI-05	Aplicar múltiplos métodos de criptografia para modelar a transmissão segura de informações	Redes e Internet	Cibersegurança	Desenvolver e Usar Abstrações
2-NI-06	Explicar como medidas de segurança físicas e digitais protegem a informação eletrônica	Redes e Internet	Cibersegurança	Comunicar Sobre a Computação
2-NI-07	Representar dados usando vários esquemas de codificação	Dados e Análise	Armazenamento	Desenvolver e Usar Abstrações
2-DA-08	Coletar dados usando ferramentas computacionais e transformar estes dados para torná-los mais úteis e confiáveis	Dados e Análise	Coleta; Visualização e Transformação	Testar e Refinar Artefatos Computacionais
2-DA-09	Refinar modelos computacionais com base nos dados que foram gerados por eles	Dados e Análise	Inferência e Modelos.	Criar Artefatos Computacionais & Desenvolver e Usar Abstrações
2-AP-10	Usar fluxogramas e/ou pseudocódigo para resolver problemas complexos como algoritmos	Algoritmos e Programação	Algoritmos	Desenvolver e Usar Abstrações
2-AP-11	Criar variáveis com definição clara que representem diferentes tipos de dados e executem operações sobre seus valores	Algoritmos e Programação	Variáveis	Criar Artefatos Computacionais
2-AP-12	Projetar e desenvolver iterativamente programas que combinem estruturas de controle, incluindo loops aninhados e condicionais compostos.	Algoritmos e Programação	Controle	Criar Artefatos Computacionais
2-AP-13	Decompor problemas e	Algoritmos e	Modularidade	Reconhecer e

	subproblemas em partes para facilitar o projeto, implementação e revisão de programas	Programação		Definir Problemas Computacionais
2-AP-14	Criar procedimentos com parâmetros para organizar o código e torná-lo mais fácil de reutilizar	Algoritmos e Programação	Modularidade	Desenvolver e Usar Abstrações
2-AP-15	Distribuir tarefas e manter uma linha de tempo para o projeto quando se está desenvolvendo artefatos computacionais de forma colaborativa	Algoritmos e Programação	Desenvolvimento de Programas	Colaborar em torno da computação
2-AP-16	Procurar e incorporar comentários (*feedback*) de membros da equipe e usuários para refinar uma solução que atenda às necessidades dos usuários	Algoritmos e Programação	Desenvolvimento de Programas	Colaborar em torno da com-putação & Promover uma cultura de computação inclusiva
2-AP-17	Incorporar código, mídia e bibliotecas existentes em programas originais e fornecer atribuição (ao autor original)	Algoritmos e Programação	Desenvolvimento de Programas	Desenvolver e Usar Abstrações, Criar Artefatos Computacionais & Comunicar Sobre a Computação
2-AP-18	Testar sistematicamente e refinar programas usando uma variedade de casos de teste	Algoritmos e Programação	Desenvolvimento de Programas	Testar e Refinar Artefatos Computacionais
2-AP-19	Documentar programas para torná-los mais fáceis de entender, testar e depurar	Algoritmos e Programação	Desenvolvimento de Programas	Comunicar Sobre a Computação
2-IC-20	Discutir questões de parcialidade/imparcialidade e acessibilidade na concepção de tecnologias existentes	Impactos da computação	Cultura	Promover uma cultura de computação inclusiva
2-IC-21	Comparar os prós e contras associados às tecnologias de computação que afetam as atividades cotidianas das pessoas e as opções de carreira	Impactos da computação	Cultura	Comunicar Sobre a Computação
2-IC-22	Colaborar com vários contribuidores por meio de estratégias como contribuição colaborativa (*crowdsourcing*) ou pesquisas (*surveys*) ao criar um artefato computacional	Impactos da computação	Interações sociais	Colaborar em torno da computação & Criar Artefatos Computacionais
2-IC-23	Descrever os prós e contras entre permitir que uma informação seja pública ou manter informações privadas e seguras	Impactos da computação	Leis de segurança e ética	Comunicar Sobre a Computação

Anexo II

Tabela II.1. Visão geral das competências de *design* de interface ensinadas

Referência	**Objetivos de aprendizagem referente a UI *design***	**Áreas de conhecimento de UI *design***	**Modelos/métodos/técnicas de UI *design***	**Ferramentas de UI *design***
	Após a UI espera-se que os alunos são capazes de:			
(CHEN, P.; HUANG, R., 2017)	**aplicar** competências do *design thinking*, envolvendo alunos na aprendizagem para criação de um projeto de jogo digital a partir do *App Inventor*	- *Design thinking* - Programação de *App* - Usabilidade;	- Protótipo em papel	NI
(SULLIVAN, J. F. et. al., 2003)	**aplicar** competências relacionadas ao *design* visual e ao *design* da interface do usuário	- *Design* visual - *Design* de interface do usuário	- Criação de imagens digitais - *Design* de interface do usuário por meio de análise, *design* conceitual e teste de usuário	- *Macromedia Director!* - *Lingo* - *Adobe Photoshop*
(KE, F.; IM, T., 2014)	**aplicar** competências de *design thinking* na criação de um jogo com *Scratch*	- *Design thinking*	- Análise mais detalhada do problema - *Brainstorming* - Prototipação	NI
(ROBINSON, A.; PÉREZ-QUIÑONES M. A., 2014)	**aplicar** competências de interação humano computador na criação de um *app* por meio de protótipo de papel	- Interação Humano computador - *Design* interface de usuário	- Protótipos de papel - Design centrado no usuário - Abstração de interface de usuário	Utilizado Aplicativo *POP* no *iPad* para conectar as telas de seu protótipo de interface de usuário
(VAN WART, S. et al., 2014)	**aplicar** competências de programação de *apps* num processo de *design*	- Design de interface do usuário - Processo de design - Interação humano-computador	- Protótipos em papel - Protótipos em meio digital - Abordagem baseada na interação humano-computador	*Balsamiq*
(DENNER, J. et al., 2005)	**aplicar** competências de *design* no desenvolvimento de jogos digitais interativos	- *Design* de interface de usuário	- Protótipo em papel - Fluxograma	NI
(EDUTOPIA, 2015)	**aplicar** competências de programação e *design thinking* no desenvolvimento de *apps*	- *Design thinking* - *Design* de interface de usuário	- Protótipo de papel - Desenvolvimento iterativo	- *Invision App* para criar protótipo de telas
(CODELIKEAGIRL, 2017)	**aplicar** competências de *design thinking* criando jogos	- *Design thinking*	- Entrevista - Design iterativo	NI
(TEKKIE UNI, 2018)	**aplicar** competências de programação de design na criação de *apps*	- *Design* de interface de usuário	NI	NI
(CREATELAB, 2017)	**aplicar** competências de computação e *design thinking* na criação *softwares* em computadores ou robóticos	- *Design thinking*	- Histórias de usuário	NI

(CODE, 2018)	**aplicar** competências de computação introduzindo um processo de *design*	- *Design* de interface de usuário	- Protótipo em papel - *Brainstorming* - Teste de usuário - *Design* iterativo	NI
(ROBINSON, A.; PÉREZ-QUINONES, M. A.; SCALES, G., 2015)	**aplicar** competências de *design* e avaliação de interface de usuário	- *Design* de interface de usuário	- Avaliação de interface de usuário - Protótipo em papel - Protótipo digital	NI
(TECHNOVATION, 2018)	**aplicar** competências do *design* na produção de um aplicativo para resolver um problema da sociedade	- *UX design*	- Protótipo em papel - Criação de persona - História de usuário	NI
(CODEHS,2018)	**aplicar** competências de computação introduzindo projeto de interface de usuário para sistemas *web*	- *Design* de interface de usuário	- *Design* iterativo - Prototipagem rápida - Testes de usuário - *Brainstorming* - Desenvolvimento incremental	- Editor *online CodeHS*
(CODE.ORG/APP LAB, 2018)	**aplicar** competências de *design* de interface de usuário por meio de desenvolvimento de *apps*	- *Design* de interface de usuário - *Design* centrado no usuário	- *Design* iterativo - Protótipo em papel - Teste de usuário - *Brainstorming* - *Wireframe* - Protótipo no app	NI
(GET STARTED WITH CODE 2, 2017)	**aplicar** competências de computação na sala de aula (independentemente da familiaridade do aluno com a programação) de forma a aplicá-los em contextos cotidianos	- *Design* visual - *Design* de interface do usuário - *UX design*	- *Design* de interface de usuário - *Brainstorming* - Persona	- *Pages* - *Mini Monet*

Tabela II.2. Visão geral das características das UIs

ID	Objetivo de aprendizagem da UI como um todo	Descrição geral	Ambiente(s) de programação	Métodos Instrucionais	Materiais	Recursos
(CHEN, P.; HUANG, R., 2017)	Desenvolver habilidades sobre o processo do *design* de criação de um jogos (baseado no *design thinking*), aprendendo pensamento criativo e o pensamento computacional	O curso consistiu em quatro módulos, com duração de uma semana cada. Participantes eram esperados para gastar 4 horas de esforço em cada módulo	*App Inventor*	- Experiência na prática - Resolução de problemas	NI	- Laboratório de computadores - Dispositivos móveis (incluindo *laptop*, computador, tablet e *smartphones*)
(SULLIVAN, J. F. et. al., 2003)	Desenvolver atitudes positivas em relação à criação e uso de tecnologia. Desenvolvimento de habilidades técnicas. Maior	Experiência voltada a meninas para desenvolverem habilidades técnicas em *design* gráfico, desenvolvimento de interface com	*NI*	- Simulações - Condução de experimentos - Grupos de aprendizagem colaborativa	NI	NI

	conscientização sobre oportunidades da carreira de TI	o usuário, programação visual, manipulação de imagens digitais, criação de multimídia e testes com usuários				
(KE, F.; IM, T., 2014)	Desenvolver habilidades sobre o processo do *design* de criação de um jogo (baseado no *design thinking*), aprendendo a agir coletivamente e proativamente	Os alunos levantaram um conceito matemático o qual gostaram e criaram um jogo usando *Scratch* sobre ele, seguindo os passos do *design*	*Scratch*	- Atribuição de papéis - Grupos de aprendizagem colaborativa - Ensino explícito	- Exemplares de jogos de matemática	- Laboratório de computadores
(ROBINSON, A.; PÉREZ-QUIÑONES M. A., 2014)	Desenvolver competências de interação humano-computador para meninas na criação de *apps* por meio de protótipos de papel	A UI ocorre em um *workshop* no qual ensina meninas a desenvolveram um *app* de chat de texto ou vídeo aplicando técnicas de interface de usuário, como protótipo de papel e várias diretrizes de *design* da interface do usuário (capacidade de aprendizado, eficiência, memorização, erro e satisfação)	NI	- Vídeos sobre computação - Discussão - Grupo de estudo - Experiência na prática - Diretrizes de *design* de interface do usuário	- Papel, lápis e marcadores para criar um protótipo de papel de baixa fidelidade	- *iPad* para avaliar a interface do usuário
(VAN WART, S. et al., 2014)	Desenvolver a habilidade de criação de aplicativos com *App Inventor* utilizando um processo de *design*	Os alunos leram literatura de justiça social, identificaram as necessidades da comunidade local e passaram por um processo de *design* para criar aplicativos móveis totalmente funcionais para atender a essas necessidades	*App Inventor*	- Resolução de problemas - Grupos de aprendizagem colaborativa - Exercício - Experiência na prática - Interpretação de papéis	NI	NI
(DENNER, J. et al., 2005)	Com o objetivo de aumentar a quantidade de mulheres na área de tecnologia da	Esta UI é um programa de verão onde colocaram as alunas no papel	NI	- Experiência na prática - Resolução de problemas - Trabalho em	- Caderno eletrônico - Jornal on-line (para as meninas	- Computador

	informação, esta UI envolve garotas do ensino médio em tecnologia da informação ensinando competências de *design* e programação no desenvolvimento de um jogo de computador interativo	de designer. As alunas tem que criar um jogo escolhendo um dos tópicos de contexto pré-definido. O desenvolvimento é guiado por 5 etapas: especificação de requisitos, *design* de interface de usuário, prototipagem, teste de protótipo e implementação		grupo	gravarem suas experiências) - Papel para projetar os jogos	
(EDUTOPIA, 2015)	Desenvolver habilidades de programação juntamente com *design thinking* resolvendo problema reais por meio de desenvolvimento de *apps*	A UI ensina a programação começando pelo *design*, pois pelo código não é interessante. Os alunos desenvolveram um app para resolver um problema da vida real utilizando técnicas de *design thinking*	- *Xcode* - *Swift*	- Grupo de aprendizagem cooperativa	- Cartão de índice contendo os protótipo das telas do *app* - Papel - Diretrizes de interface humana que fornecem detalhes sobre o que torna os aplicativos fáceis e intuitivos de usar	- *iPhone*
(CODELIKEAGIRL, 2017)	Projetar e desenvolver jogos utilizando conceitos de programação básica e um processo de *design thinking*	Os alunos trabalharam semanalmente para projetar e desenvolver um jogo de frações seguindo o *design thinking*. Alunos do 5º ano criaram o jogo tendo como público alvo alunos do 2º ano. Ao invés de definir um roteiro predefinido de DT, os alunos que construíram seus próprios conceitos do processo de design durante todo o projeto.	- *Scratch*	- Entrevista - Investigação - Discussão - Grupos de aprendizagem cooperativa - Experiência na prática	- *Post-its*	- *Laptops* - Computador *desktop*
(TEKKIE UNI, 2018)	Nas aulas *online*, é ensinado aos jovens como transformar suas ideias em	O curso é baseado na criação de *apps* para cada	- *Alice*	- Instrução auxiliada por computador - Aulas a	NI	NI

	realidade usando o poder do código. O curso se move da lógica básica e conceitos de programação para codificação avançada, tudo através de diversão, experiência e prática	conceito ou conjunto de conceitos a ser ensinado		distância - Exercitar e praticar - Ensino explícito - Demonstrações		
(CREATELAB, 2017)	- Desenvolver o pensamento computacional - Desenvolver habilidades do século XXI em pensamento crítico, criatividade, comunicação e colaboração - Entender e apreciar os conceitos básicos de computação, incluindo algoritmos, seqüências, *loops*, condicionais, variáveis etc. - Desenvolver programas de computador e *hardware* para jogos simples, histórias digitais e ferramentas de aprendizagem usando conceitos de *design* e inovação	Os alunos codificam os seus próprios jogos de computador e arte digital, programam robôs e desenvolvem *hardware* que irá melhorar progressivamente as suas capacidades de pensamento e ajudá-los a sobressair na escola e em casa. Os alunos são encorajados a explorar, em um ambiente facilitador de "aprender fazendo" no próprio ritmo, onde eles aprendem a relacionar *design* e inovação com os principais conceitos computacionais, incluindo abstração, variáveis, *loops* e condicionais	- *Scratch* - *Dash Robot* - *Makey Makey* - *Google Blockly Programming Language*	- Desafios - Resolução de problemas - Experiência na prática - Exercícios	- Robôs do *Dash* - Micro - controladores	- Laboratório com *tablets*, *laptops* e *notebooks*
(CODE, 2018)	Pensar sobre a ciência da computação como uma ferramenta para resolver seus próprios problemas no sentido de considerar os impactos sociais mais amplos da computação	Por meio de uma série de desafios de *design*, os alunos são convidados a considerar e compreender as necessidades dos usuários enquanto desenvolvem uma solução para um problema. A segunda metade	- *App Lab*	- Desafios - Resolução de problemas - Debate - Grupos de aprendizagem cooperativa - Aula expositiva - Exercícios	- Fórum *online* - Notas - *Post-its* - Pôster - *Slides* - Guias de atividades - Materiais *online*	- Papel

		da unidade consiste em um projeto iterativo de equipe, durante o qual os alunos têm a oportunidade de identificar uma necessidade com a qual se importam, prototipar soluções em papel e no *App Lab* (desenvolvimento de *apps*) e testar suas soluções com usuários reais para obter *feedback* e direcionar mais iteração				
(ROBINSON, A.; PÉREZ-QUINONES, M. A.; SCALES, G., 2015)	Aplicar as competências da computação para as meninas afro-americanas com o objetivo de compreender os fatores que levam o desinteresse dessas mulheres na área da computação	A UI foi realizada em duas oficinas, a primeira ensinou *design* e avaliação de interface de usuário e a segunda algoritmos. As estudantes desenvolveram uma rede social por meio *apps* e também controlaram robôs	NI	- Grupos de aprendizagem cooperativa	- Vídeos - *Slides* - Papel para projetar o protótipo	- *iPads* para protótipos fazer digitais
(TECHNOVATION, 2018)	Resolver um problema da sociedade aprendendo na prática a produzir um aplicativo com passos do *design*	As meninas tem a oportunidade de passar por todo o processo, desde a identifcação do problema e a geração de ideias para solucioná-lo, até a elaboração do plano de negócios e desenvolvimento para lançamento do produto digital no mercado e a apresentação para investidores	*App Inventor*	- Aula expositiva - Resolução de problemas - Grupos de aprendizagem cooperativa	- Roteiro de perguntas - Folhas modelo para as personas - Folhas modelo para histórias de usuário - Gravador - Papel	NI
(CODEHS,2018)	No final deste curso, os alunos poderão explicar como as páginas da *Web* são	O currículo é composto por diversos módulos, e	- *Html* - *CSS*	- Desafios - Aula expositiva em vídeo - Experiência na	- Tutoriais - Vídeos - Artefatos de *software*	NI

	desenvolvidas e visualizadas na *Internet*, analisar e corrigir erros em *sites* existentes e criar seus próprios *sites* de várias páginas	dentre eles está o módulo *Designer web*, que dentre o seu conteúdo ensina teoria e prática de Projeto de interface de usuário por meio de desenvolvimento *web*.		prática - Grupo de estudo - Exercício	- Folhetos de aula	
(CODE.ORG/APP LAB, 2018)	Os alunos vão aprender desenvolver *apps* baseado nas necessidades dos outros aplicando técnicas de *design* de interface de usuário e usabilidade	Nesta unidade, os alunos são convidados a considerar e compreender as necessidades dos outros enquanto desenvolvem uma solução para um problema por meio de uma série de desafios de projeto. A segunda metade da unidade consiste em um projeto iterativo de equipe, durante o qual os alunos têm a oportunidade de identificar uma necessidade com a qual se importam, prototipar soluções em papel e no *App Lab* e testar suas soluções com usuários reais para obter *feedback* e direcionar mais iteração	- *App lab*	- Desafios - Grupo de estudo - Entrevista em pares para definir necessidades de usuário	- Artefato de *software* - *Post-its*	NI
(GET STARTED WITH CODE 2, 2017)	Este guia é destinado a professores a ensinarem alunos (de oito a onze anos) a programarem *apps*, aplicando técnicas de *design* de interface de usuário e usabilidade.	O conteúdo deste documento, pode ser usado isoladamente ou como parte de uma introdução ao programa de codificação. Há cerca de 20 aulas de programação com 16 horas adicionais de	*Tynker*	- Aula expositiva - Desafios - Resolução de problemas - Experiência na prática	- *Slides*	- *iPads* para protótipos fazer digitais

		atividades de *design* de aplicativos. As lições podem ser ensinadas em um único bloco ou em seções				

ID	Métodos/instrumentos de avaliação	Língua	Licença	Modo de Ensino	Capacitação dos instrutores
(CHEN, P.; HUANG, R., 2017)	Observações, gravações, artefatos criados	Inglês	NI	Presencial	NI
(SULLIVAN, J. F. et. al., 2003)	Avaliações pessoais de *performance*, questionários pré e pós, testes	Inglês	NI	Presencial	Sim
(KE, F.; IM, T., 2014)	Observações, gravações, análise dos artefatos criados	Inglês	NI	Presencial	Sim
(ROBINSON, A.; PÉREZ-QUIÑONES M. A., 2014)	NI	Inglês	NI	Presencial	NI
(VAN WART, S. et al., 2014)	Observações, gravações, artefatos criados	Inglês	NI	Presencial	Sim
(DENNER, J. et al., 2005)	Entrevistas, análises de jogos, desafios	Inglês	NI	Presencial	Sim
(EDUTOPIA, 2015)	NI	Inglês	NI	Presencial	NI
(CODELIKEAGIRL, 2017)	Entrevistas, artefatos criados	Inglês	NI	Presencial	NI
(TEKKIE UNI, 2018)	NI	Inglês	NI	Online	Sim
(CREATELAB, 2017)	Exercícios	Inglês	Pago	Presencial	Sim
(CODE, 2018)	Exercícios, artefatos criados	Inglês	Creative Commons License (CC BY-NC-SA 4.0)	Presencial	Sim
(ROBINSON, A.; PÉREZ-QUINONES, M. A.; SCALES, G., 2015)	NI	Inglês	NI	Presencial	NI
(TECHNOVATION, 2018)	Artefatos criados, apresentação	Português (submissão em Inglês)	NI	Presencial	Sim
(CODEHS, 2018)	Quiz, exercícios, testes	Inglês	NI	*Online*	NI
(CODE.ORG/APP LAB, 2018)	Artefatos criados, apresentação	Inglês	Gratuito	*Online*	NI
(GET STARTED WITH CODE 2, 2017)	Artefatos criados, apresentação	Inglês	Gratuito, porém restrito a usuários de dispositivos de iPad com iBooks	*Online*	Não

			2 ou posterior e iOS 5 ou posterior, ou um iPhone com iOS 8.4 ou posterior, ou um Mac com OS X 10.9 ou posterior.		

Tabela II.3. Visão geral das características de contexto das UIs

ID	Nível de ensino	Duração da UI	Pré-requisitos
(CHEN, P.; HUANG, R., 2017)	Ensino Médio	4 semanas	NI
(SULLIVAN, J. F. et. al., 2003)	Ensino Médio	6 semanas	Não
(KE, F.; IM, T., 2014)	Ensino Fundamental	6 semanas	NI
(ROBINSON, A.; PÉREZ-QUIÑONES M. A., 2014)	Ensino Médio	1 hora por dia, 5 dias = 5 horas	NI
(VAN WART, S. et al., 2014)	Ensino Fundamental e Médio	12 semanas	NI
(DENNER, J. et al., 2005)	Ensino Fundamental II	NI	NI
(EDUTOPIA, 2015)	Ensino Fundamental I e II	NI	Não
(CODELIKEAGIRL, 2017)	Ensino Fundamental I	~52 horas (9 meses, com encontros semanais de 80 minutos)	Sim
(TEKKIE UNI, 2018)	NI	2h por semana, 9 meses	NI
(CREATELAB, 2017)	Ensino Fundamental I e II	40 horas (1 vez por semana e 20 aulas)	Não
(CODE, 2018)	Ensino Fundamental II	~18 horas	Não
(ROBINSON, A.; PÉREZ-QUINONES, M. A.; SCALES, G., 2015)	Ensino Médio	5 horas (1 hora por dia)	Não
(TECHNOVATION, 2018)	Ensino Fundamental II e ensino Médio	12 semanas	NI
(CODEHS,2018)	Ensino Fundamental II e ensino Médio	1 ano	Não
(CODE.ORG/APP LAB, 2018)	Ensino Fundamental II e Ensino Médio	~36 horas	Não
(GET STARTED WITH CODE 2, 2017)	Ensino Fundamental I	~20 aulas de programação com 16 horas adicionais de atividades de design de aplicativos	Não

Tabela II.4. Métodos de desenvolvimento das UIs

Citação	Método de desenvolvimento
(CHEN, P.; HUANG, R., 2017)	NI
(SULLIVAN, J. F. et. al., 2003)	Mentores de nível superior (não graduados) foram utilizados para auxiliar e controlar as equipes. Os materiais e o plano de ensino foram desenvolvidos de acordo com os objetivos

	de aprendizagem que também foram apresentados aos alunos e pais
(KE, F.; IM, T., 2014)	Um grupo de estudantes graduados facilitou todas as sessões de *design*. Eles responderam perguntas, deram *feedback* entre as etapas do *design* e, ocasionalmente, levaram as crianças a participarem de explicações sobre o *design* ou o conteúdo matemático. Eles também forneceram ajuda na programação do *Scratch* durante o desenvolvimento do jogo
(ROBINSON, A.; PÉREZ-QUIÑONES M. A., 2014)	NI
(VAN WART, S. et al., 2014)	NI
(DENNER, J. et al., 2005)	O desenvolvimento da UI se baseou em métodos que incorporam uma abordagem de pesquisa baseada em *design*, no qual foi criada e testada atividades programáticas e estratégias instrucionais, depois foram refinadas para aumentar a eficácia. Essa abordagem fornece dados valiosos sobre contextos e mecanismos para descrever relações causais plausíveis entre estratégias e objetivos do programa. Em relação aos instrutores, foram treinadas seis mulheres que trabalham como líderes de programa ou assistentes de professores na escola para serem professores, instrutores de software e/ou de programação
(EDUTOPIA, 2015)	NI
(CODELIKEAGIRL, 2017)	NI
(TEKKIE UNI, 2018)	NI
(CREATELAB, 2017)	A UI foi desenvolvida pelas principias universidade de Oxford, *MIT*, Universidade Nacional de Cingapura (NUS) e Universidade de Tecnologia de Cingapura (SUTD) e utiliza conceitos de codificação, robótica e design, fazendo com que os alunos sejam incentivados a se tornarem solucionadores de problemas e criadores de conteúdo. O curso utiliza a teoria construcionista de aprendizagem e da teoria de aprendizado aprimorada pela tecnologia, juntamente com as Etapas Chave 1-3 dos Programas Britânicos de Estudo de Computação
(CODE, 2018)	NI
(ROBINSON, A.; PÉREZ-QUINONES, M. A.; SCALES, G., 2015)	A Teoria Cognitiva Social de Bandura [1] é o arcabouço teórico abrangente usado no desenvolvimento da UI. O componente auto-eficácia, que é a crença nas capacidades de uma pessoa para atingir os resultados desejados, foi usado para conduzir o ensino. A auto-eficácia, influencia a persistência, bem como outros processos psicológicos. Assim, uma perda de confiança e auto-eficácia conduz frequentemente a uma perda de interesse afetando o desempenho, objetivos pessoais, resiliência a falhas, etc.
(TECHNOVATION, 2018)	Foram utilizados mentores para auxiliar os grupos em todas as partes do desenvolvimento
(CODEHS, 2018)	NI
(CODE.ORG/APP LAB, 2018)	NI
(GET STARTED WITH CODE 2, 2017)	NI

Tabela II.5. Visão geral da avaliação da qualidade das UIs.

ID	Que tipo de estudo	Fatores	Método(s) de coleta de dados	Tamanho de amostra	Avaliaçõe s replicadas	Método(s) de análise de dados	Descobertas
(CHEN, P.; HUANG, R., 2017)	Estudo de caso	Resolução de problemas, utilidade, aprendizagem	Entrevistas, questionário s, observação, testes de habilidades	25 alunos	Não foi replicado	Análise qualitativa	Com o uso do *App Inventor* e conceitos do *Design Thinking* foi possível perceber a melhora do

							desempenho da aprendizagem e engajamento dos alunos
(SULLIVAN, J. F. et. al., 2003)	Quase-experimental	*Performance*, qualidade do ensino	Entrevistas, questionários, testes de habilidades	36 alunos	Não	Análise qualitativa	Este modo de ensino mostrou ter muitos ganhos em relação às habilidades nos alunos. A utilização de mentores foi muito bem aceita.
(KE, F.; IM, T., 2014)	Estudo de caso	*Performance*	Entrevistas, observações, gravações, artefatos criados	64 alunos	NI	Análise qualitativa e quantitativa descritiva	O estudo indicou que as crianças perceberam o aprendizado de matemática e o desenvolvimento de habilidades de computação e *design* durante o *design* de jogos. A presença de desempenho e liderança de função na equipe contribui para um esforço coordenado na decomposição de uma solução de *design* complexa e, portanto, em um plano de *design* mais rico
(ROBINSON, A.; PÉREZ-QUIÑONES M. A., 2014)	Estudo de caso	Satisfação, diversão, interesse na computação	Entrevista com grupos focais, *survey* (pré e pós)	19 alunos	Não foi replicado	Análise qualitativo	A prototipagem de papel pode ser usada motivação para seguir carreira na ciência da computação. O processo centrado no usuário da ciência da computação deu às meninas uma percepção positiva da ciência da computação. O aplicativo POP foi essencial para ajudar os participantes a entender o

							processo de *design* da interface do usuário e para visualizar seus projetos como aplicativos realistas
(VAN WART, S. et al., 2014)	Estudo de caso	*Performance*, utilidade	Entrevista, artefatos criados, gravações	NI	NI	Análise qualitativa	Indica que o *design*, como se aplica à solução de problemas no mundo, pode ser uma maneira de atrair uma gama maior de alunos para a disciplina de computação. Em segundo lugar, sugere que um currículo centrado em aplicativos, em oposição a um currículo centrado em sistemas, pode ser uma maneira atraente de apresentar ideias de ciência da computação a novatos e jovens
(DENNER, J. et al., 2005)	Estudo de caso	Aprendizagem, diversão, interesse	Observações de adultos, registros do líder do programa, pesquisas e entrevistas com os participantes, registros das cadernetas eletrônicas das meninas e os jogos criados pelas meninas.	62 meninas	NI	Análise qualitativa	Descobriram que ensinar as meninas as habilidades de projetar e programar jogos baseia-se no seu prazer de contar histórias, gráficos e auto-expressão. Além disso, fornece uma oportunidade para as meninas explorarem ou afirmarem sua identificação na área de TI e possivelmente podendo despertar interesse na área
(EDUTOPIA, 2015)	*Ad-hoc*	Motivação, utilidade, aprendizagem	NI	NI	NI	NI	Ensinar programação junto com o

							design thinking promove o valor da empatia com outras pessoas nas comunidades. Esta conexão ajuda a identificar uma necessidade e projetar uma solução por meio de *feedbacks* e iterações
(CODELIKEAGIRL, 2017)	*Ad-hoc*	Aprendizagem, utilidade	Questionário	NI	NI	Análise qualitativa	Alunos tiveram a percepção de que podem utilizar o processo de *Design Thinking* para projetar ou reorganizar outras coisas, além de jogos
(TEKKIE UNI, 2018)	NI	NI	NI	NI	NI	NI	NI
(CREATELAB, 2017)	NI	NI	NI	NI	NI	NI	NI
(CODE, 2018)	NI	NI	NI	NI	NI	NI	NI
(ROBINSON, A.; PÉREZ-QUINONES, M. A.; SCALES, G., 2015)	Estudo de caso	Confiança, aprendizagem, interesse	Questionário (pré e pós), entrevistas com grupos focais	23 (HCI) + 14 (algoritmos)	NI	Análise quantitativa inferencial e análise qualitativa	Os resultados mostram que as meninas afro-americanas do ensino médio geralmente têm atitudes negativas em relação à ciência da computação. Após a intervenção de ciência da computação, os resultados também revelam que quatro fatores influenciam as atitudes das meninas afro-americanas de ensino médio em relação à ciência da computação, como 1) a participação em uma intervenção, 2) o domínio de conteúdo de intervenção, 3) a facilidade de

							realizar as atividades e 4) características dos participantes como status socioeconômico, educação da mãe, notas escolares e o uso de *smartphones* e consoles de *videogame* em casa
(TECHNOVATION, 2018)	NI	NI	NI	NI	NI	NI	NI
(CODEHS, 2018)	NI	NI	NI	NI	NI	NI	NI
(CODE.ORG/APP LAB, 2018)	NI	NI	NI	NI	NI	NI	NI
(GET STARTED WITH CODE 2, 2017)	NI	NI	NI	NI	NI	NI	NI

Anexo III

Detalhamento de todas as categorias presentes no App Inventor.

Tabela III.1 - Detalhamento da categoria de Interface do Usuário (MIT, 2017e)

Interface de usuário	
Componente	**Descrição**
Botão	Representa um botão com a capacidade de detectar cliques sobre ele. Vários de seus aspectos podem ser modificados tanto no Designer (Propriedades) quanto no Blocos.
Caixa de Seleção	A caixa de seleção é um componente capaz de detectar toques do usuário e modificar seu estado *booleano* em resposta. A caixa de seleção inicia um evento quando o usuário a toca. Vários de seus aspectos podem ser modificados tanto no Designer quanto no Blocos.
Escolher Data	Quando clicado, este botão inicia um diálogo em forma de *popup* que permite o usuário selecionar um data.
Imagem	Componente para exibir imagens. A imagem a ser exposta e outros aspectos do componente podem ser especificados tanto no Designer quanto no Blocos.
Legenda	É um componente utilizado para a exibição de texto. Uma legenda apresenta um texto que é especificado nas suas propriedades. Vários de seus aspectos podem ser modificados tanto no Designer quanto no Blocos.
Escolhe Lista	É um botão que, quando clicado, exibe uma lista de textos os quais o usuário pode selecionar. Os textos exibidos na lista podem ser modificados tanto no Designer quanto no Blocos.
Visualizador de Listas	Este é um componente que permite a exibição de uma lista de textos na tela. A lista pode ser modificada tanto no Designer quanto no Blocos.
Notificador	O notificador exibe diálogos de alerta, mensagens e alertas temporários.
Caixa de Senha	Na caixa de senha o usuário pode inserir uma senha e os caracteres inseridos são substituídos por asteriscos na tela. Esta senha pode ser armazenada ou utilizada para alguma outra função.
Deslizador	O deslizador é uma barra progressiva com um espaço arrastável. O usuário pode arrastar este espaço para a direita e esquerda a fim de inserir um valor a este componente. A posição deste espaço pode ser usada dinamicamente para qualquer outra função desejada.
Lista Suspensa	A lista suspensa exibe um *popup* com uma lista de elementos a serem selecionados pelo usuário. Estes elementos podem ser modificados tanto no Designer quanto no Blocos.
Caixa de Texto	Com a caixa de texto o usuário pode inserir um texto e este texto ser armazenado ou utilizado para alguma outra função.
Escolhe Hora	É um botão que, ao ser clicado, inicia um diálogo em forma de *popup* que permite o usuário escolher uma hora.
Navegador Web	O navegador Web é um componente que permite a visualização de páginas da Web. A URL de entrada pode ser especificada no Designer ou no Blocos.

Tabela III.2 - Detalhamento da categoria de Organização (MIT, 2017e)

Organização	
Componente	**Descrição**
Organização Horizontal	Uma organização horizontal é utilizada para exibir um grupo de componentes de maneira horizontal, da esquerda para a direita ou vice-versa.
Organização Scroll Horizontal	É igual ao componente de organização horizontal, mas esta é a versão rolável.
Organização em Tabela	A organização em tabela é utilizada para exibir componentes de maneira tabular. O número de linhas e colunas pode ser modificado no Designer, em suas propriedades. Para cada elemento desta tabela, apenas um componente será visível dentro dele.
Organização Vertical	Uma organização vertical é utilizada para exibir um grupo de componentes de maneira vertical, de cima para baixo ou vice-versa.
Orientação Scroll Vertical	É igual ao componente de organização vertical, mas esta é a versão rolável.

Tabela III.3 - Detalhamento da categoria de Mídia (MIT, 2017e)

Mídia	
Componente	**Descrição**
Câmera de Vídeo	Utilizado para a gravação de um vídeo usando a câmera do próprio smartphone. O vídeo gravado pode depois ser visualizado e utilizado pelo usuário.
Câmera	Este componente permite tirar uma foto com a câmera do smartphone. A foto fica armazenada no dispositivo e pode ser visualizada e utilizada pelo usuário.
Escolhe Imagem	É um botão com um propósito especial. Quando é clicado, a galeria de imagens do dispositivo é aberta para que o usuário escolha uma imagem. A imagem que foi selecionada é armazenada no aplicativo.
Tocador	O tocador tem a função de executar um arquivo de áudio e pode controlar a vibração do dispositivo. Este componente é indicado para a execução de longos áudios, como músicas por exemplo.
Som	Este componente tem a capacidade de executar um arquivo de áudio ou também vibrar o dispositivos por um tempo indicado em milissegundos. Ele é indicado para execução de áudios curtos, como efeitos de som e pequenos toques por exemplo.
Gravador	É um componente capaz de gravar um áudio.
Reconhecedor de Voz	Este componente consegue reconhecer a fala do usuário e a transformar em um texto utilizando o reconhecimento de fala do Android.
Texto para Falar	Este componente tem a função de falar um dado texto em voz alta. A linguagem da fala pode ser modificada, mudando a pronúncia das palavras.
Reprodutor de Vídeo	O reprodutor de vídeo é um componente capaz de reproduzir um vídeo. Este componente tem as opções para o usuário iniciar, pausar, pular adiante e pular para trás o vídeo em reprodução.
Tradutor Yandex	Utilizado para a tradução de palavras e sentenças entre linguagens distintas. É um componente que necessita acesso à Internet.

Tabela III.4 - Detalhamento da categoria de Desenho e Animação (MIT, 2017e)

Desenho e Animação	
Componente	**Descrição**
Bola	Representa uma bola a qual reage a toques e arrastes do usuário e interage com outras *sprites*. A sua diferença para a imagem *sprite* é que este componente só representa visualmente uma bola, sendo possível apenas mudar sua cor e tamanho.
Pintura	É um painel retangular sensível ao toque onde *sprites* podem ser movidas e desenhos podem ser feitos. As posições na pintura são identificadas pelas coordenadas “X” (horizontal) e “Y” (vertical).
Imagem Sprite	Representa uma *sprite* que reage a toques e arrastes e interage com outras *sprites*. Este componente pode ter a aparência de uma imagem selecionada pelo usuário.

Tabela III.5 - Detalhamento da categoria de Mapas (MIT, 2017e)

Mapas	
Componente	**Descrição**
Círculo	É um componente que representa um círculo em uma dada latitude e longitude.
Coleção de Recursos	Este componente agrupa um ou mais recursos de um mapa. Qualquer evento que ocorra em um dos recursos também iniciará o evento nos outros recursos.
Sequência de Linhas	Permite o usuário desenhar uma contínua sequência de linhas em um mapa.
Mapa	Representa um espaço bidimensional que possui um mapa como imagem de fundo, possibilitando múltiplas marcações em cima deste.
Marcador	Este componente possibilita a indicação de pontos no mapa, como pontos de interesse. Estes pontos podem ser modificados de várias formas.
Polígono	O Polígono permite a exibição focada de uma área bidimensional. Pode ser utilizado para desenhar um

	perímetro, cidade, país, entre outros.
Retângulo	O Retângulo é um Polígono com latitudes e longitudes fixas para as bordas ao norte, sul, leste e oeste. Com o mudança de um vértice, o retângulo atualiza suas arestas de acordo.

Tabela III.6 - Detalhamento da categoria de Sensores (MIT, 2017e)

Sensores	
Componente	**Descrição**
Sensor Acelerômetro	É um componente que tem a capacidade de detectar movimentos e medir a aceleração.
Código de Barras	Componente capaz de ler um código de barras.
Temporizador	O Temporizador provê o tempo baseado no relógio interno do dispositivo. Também pode iniciar um contador de tempo para a geração de eventos, cálculos, manipulações e conversões.
Sensor Giroscópio	Este componente é capaz de medir a velocidade angular em três dimensões em unidades de graus por segundo. Para isto, o dispositivo deve ser um sensor de giroscópio.
Sensor de Localização	O Sensor de Localização informa ao usuário dados sobre a localização, como longitude, latitude, altitude, velocidade e endereço se forem suportados pelo dispositivo.
Campo próximo	Componente que provê comunicação entre dispositivos em um campo próximo (NFC - Near Field Communication).
Sensor de Orientação	Este componente permite determinar a orientação espacial do dispositivo.
Pedômetro	Baseado no Acelerômetro, o Pedômetro estima se um passo foi feito ou não. Pode calcular uma distância percorrida, por exemplo.
Sensor de Proximidade	O Sensor de Proximidade permite medir a proximidade de um objeto a tela do dispositivo, podendo ser utilizado para inúmeras funcionalidades.

Tabela III.7 - Detalhamento da categoria Social (MIT, 2017e)

Social	
Componente	**Descrição**
Escolhe Contato	É um botão que, ao ser clicado, exibe uma lista de contatos para o usuário escolher. Após a escolha, será informada uma série de informações sobre este contato.
Escolhe Email	Este componente se assemelha a Caixa de Texto. Enquanto o usuário começa a inserir o email ou nome de um contato, é mostrado um menu com opção a escolher que completam a entrada.
Ligação	O componente Ligação permite a realização de uma chamada telefônica a um número especificado em suas propriedades, tanto no Designer quanto nos Blocos. Este componente geralmente é utilizado com o Escolhe Contato.
Escolhe Número de Telefone	É um botão que, ao ser clicado, exibe uma lista de números de telefone para o usuário escolher. Após a escolha, será informada uma série de informações sobre este número.
Compartilhamento	Este componente permite o compartilhamento de arquivos ou mensagens entre o próprio *app* ou outros *apps* instalados no dispositivo. Ele exibe uma lista de aplicativos que podem receber tal arquivo ou mensagem e o usuário pode selecionar entre eles.
Mensagens SMS	É um componente capaz de enviar uma mensagem telefônica, enviando a mensagem que foi especificada.
Twitter	Este componente permite a comunicação com a rede social Twitter. Basta o usuário entrar em sua conta que ele tem acesso a várias funcionalidades da rede social.

Tabela III.8 - Detalhamento da categoria de Armazenamento (MIT, 2017e)

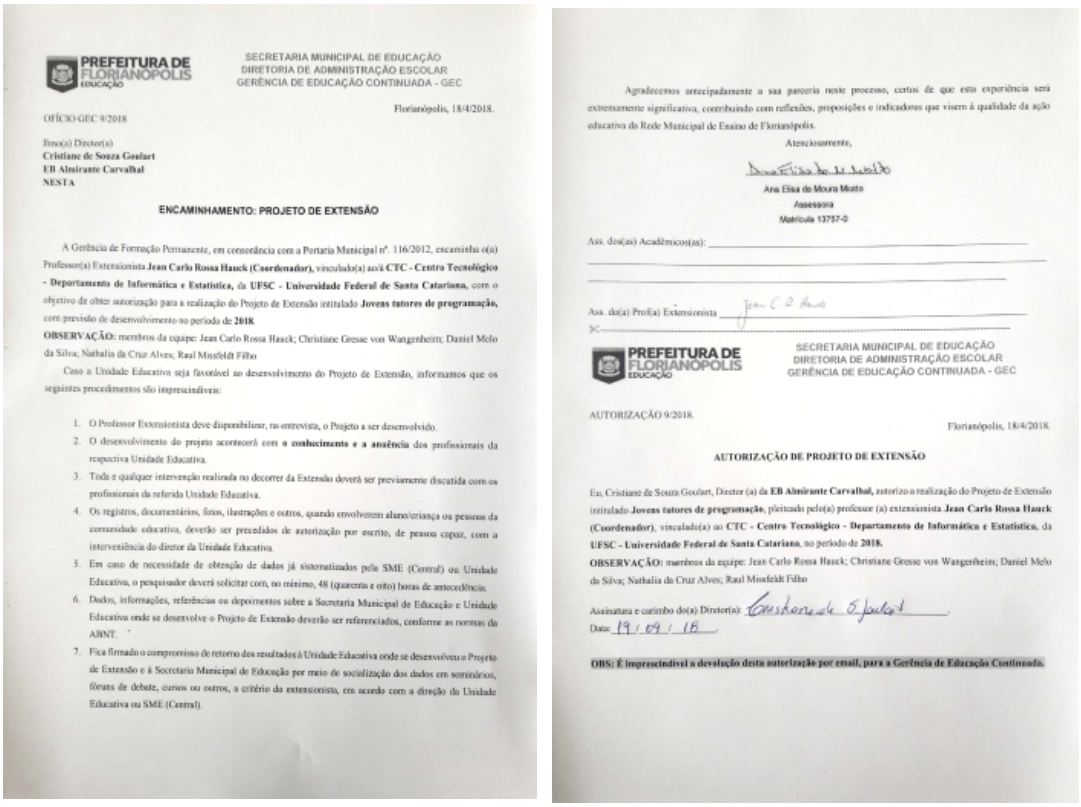
Armazenamento	
Componente	**Descrição**

Arquivo	É um componente para armazenamento e busca de arquivos. Permite a escrita ou leitura de arquivo do seu dispositivo.
Controle de *Fusiontables*	Este componente se comunica com o *Google Fusion Tables*, o qual permite que o usuário armazene, compartilhe, visualize tabelas de dados.
TinyDB	É um componente capaz de armazenar dados de um aplicativo no próprio dispositivo.
TinyWebDB	É um componente capaz de armazenar dados de um aplicativo utilizando um serviço *Web* para armazenar e buscar informações.

Tabela III.9 - Detalhamento da categoria de Conectividade (MIT, 2017e)

Conectividade	
Componente	**Descrição**
Iniciador de Atividades	Este componente é capaz de iniciar atividades, como por exemplo abrir a câmera do dispositivo, realizar um pesquisa na Internet e abrir um *browser* em uma certa página.
Cliente Bluetooth	Representa um cliente *Bluetooth*.
Servidor Bluetooth	Representa um servidor *Bluetooth*.
Web	Componente que provê funções para requisições HTTP GET, POST, PUT e DELETE.

Anexo IV



PREFEITURA DE FLORIANÓPOLIS
EDUCAÇÃO

SECRETARIA MUNICIPAL DE EDUCAÇÃO
DIRETORIA DE ADMINISTRAÇÃO ESCOLAR
GERÊNCIA DE EDUCAÇÃO CONTINUADA - GEC

Florianópolis, 18/4/2018.

OFÍCIO GEC 9/2018

Ilmo(a) Diretor(a)
Cristiane de Souza Goulart
EB Almirante Carvalhal
NESTA

ENCAMINHAMENTO: PROJETO DE EXTENSÃO

A Gerência de Formação Permanente, em consonância com a Portaria Municipal nº. 116/2012, encaminha o(a) Professor(a) Extensionista **Jean Carlo Rossa Hauck (Coordenador)**, vinculado(a) ao/à **CTC - Centro Tecnológico - Departamento de Informática e Estatística,** da **UFSC - Universidade Federal de Santa Catariana,** com o objetivo de obter autorização para a realização do Projeto de Extensão intitulado **Jovens tutores de programação,** com previsão de desenvolvimento no período de **2018**.

OBSERVAÇÃO: membros da equipe: Jean Carlo Rossa Hauck; Christiane Gresse von Wangenheim; Daniel Melo da Silva; Nathalia da Cruz Alves; Raul Missfeldt Filho

Caso a Unidade Educativa seja favorável ao desenvolvimento do Projeto de Extensão, informamos que os seguintes procedimentos são imprescindíveis:

1. O Professor Extensionista deve disponibilizar, na entrevista, o Projeto a ser desenvolvido.
2. O desenvolvimento do projeto acontecerá com **o conhecimento e a anuência** dos profissionais da respectiva Unidade Educativa.
3. Toda e qualquer intervenção realizada no decorrer da Extensão deverá ser previamente discutida com os profissionais da referida Unidade Educativa.
4. Os registros, documentários, fotos, ilustrações e outros, quando envolverem aluno/criança ou pessoas da comunidade educativa, deverão ser precedidos de autorização por escrito, de pessoa capaz, com a interveniência do diretor da Unidade Educativa.
5. Em caso de necessidade de obtenção de dados já sistematizados pela SME (Central) ou Unidade Educativa, o pesquisador deverá solicitar com, no mínimo, 48 (quarenta e oito) horas de antecedência.
6. Dados, informações, referências ou depoimentos sobre a Secretaria Municipal de Educação e Unidade Educativa onde se desenvolve o Projeto de Extensão deverão ser referenciados, conforme as normas da ABNT.
7. Fica firmado o compromisso de retorno dos resultados à Unidade Educativa onde se desenvolveu o Projeto de Extensão e à Secretaria Municipal de Educação por meio de socialização dos dados em seminários, fóruns de debate, cursos ou outros, a critério do extensionista, em acordo com a direção da Unidade Educativa ou SME (Central).

Agradecemos antecipadamente a sua parceria neste processo, certos de que esta experiência será extremamente significativa, contribuindo com reflexões, proposições e indicadores que visem à qualidade da ação educativa da Rede Municipal de Ensino de Florianópolis.

Atenciosamente,

Ana Elisa de Moura Mosto
Assessora
Matrícula 13757-0

Ass. dos(as) Acadêmicos(as): _____

Ass. do(a) Prof(a) Extensionista _____

PREFEITURA DE FLORIANÓPOLIS
EDUCAÇÃO

SECRETARIA MUNICIPAL DE EDUCAÇÃO
DIRETORIA DE ADMINISTRAÇÃO ESCOLAR
GERÊNCIA DE EDUCAÇÃO CONTINUADA - GEC

AUTORIZAÇÃO 9/2018.

Florianópolis, 18/4/2018.

AUTORIZAÇÃO DE PROJETO DE EXTENSÃO

Eu, Cristiane de Souza Goulart, Diretor (a) da **EB Almirante Carvalhal,** autorizo a realização do Projeto de Extensão intitulado **Jovens tutores de programação**, praticado pelo(a) professor (a) extensionista **Jean Carlo Rossa Hauck (Coordenador)**, vinculado(a) ao **CTC - Centro Tecnológico - Departamento de Informática e Estatística,** da **UFSC - Universidade Federal de Santa Catariana**, no período de **2018.**

OBSERVAÇÃO: membros da equipe: Jean Carlo Rossa Hauck; Christiane Gresse von Wangenheim; Daniel Melo da Silva; Nathalia da Cruz Alves; Raul Missfeldt Filho

Assinatura e carimbo do(a) Diretor(a): _____

Data: 19/04/18

OBS: É imprescindível a devolução desta autorização por email, para a Gerência de Educação Continuada.

Figura IV.1 Aprovação da Prefeitura de Florianópolis para o projeto Jovens Tutores de Programação 2018

Tabela IV.1. Aprovação da Prefeita de Florianópolis para o projeto ECOPET 2019

PREFEITURA DE FLORIANÓPOLIS
EDUCAÇÃO

SECRETARIA MUNICIPAL DE EDUCAÇÃO
DIRETORIA DE GESTÃO ESCOLAR
GERÊNCIA DE FORMAÇÃO CONTINUADA
GFC

OFÍCIO GFC 7 /2019

Florianópolis, 26/02/2019

Ilmo. Diretor (a)
Cristiane de Souza Goulart
EBM Almirante Carvalhal

ENCAMINHAMENTO: PESQUISA DE MESTRADO

A Gerência de Formação Continuada, em consonância com a Portaria Municipal nº. 116/2012, encaminha o (a) pesquisador (a) **Giselle Araújo e Silva Medeiros,** do **PPGE -Programa de Pós-graduação em Educação,** da **Universidade Federal de Santa Catarina - UFSC,** com o objetivo de obter autorização para a realização da pesquisa de Mestrado intitulada: **O desenvolvimento de aplicativos para dispositivos móveis na educação básica** na **EBM Almirante Carvalhal,** com previsão de desenvolvimento no período de: **2019.**

Caso a Unidade Educativa seja favorável à pesquisa, informamos que os seguintes procedimentos são imprescindíveis:

1. O pesquisador deve disponibilizar, na entrevista, carta de apresentação do professor orientador e projeto de pesquisa.
2. O desenvolvimento do projeto acontecerá com **o conhecimento e a anuência** dos profissionais da respectiva Unidade Educativa.
3. Toda e qualquer intervenção realizada pelo pesquisador deverá ser previamente discutida com os profissionais da referida Unidade Educativa.
4. Os registros, documentários, fotos, ilustrações e outros, quando envolverem aluno/criança ou pessoas da comunidade educativa, deverão ser precedidos de autorização por escrito, de pessoa capaz, com a interveniência do diretor da Unidade Educativa.
5. Em caso de necessidade de obtenção de dados já sistematizados pela SME (Central) ou Unidade Educativa, o pesquisador deverá solicitar com, no mínimo, 48 (quarenta e oito) horas de antecedência.
6. Dados, informações, referências ou depoimentos sobre a Secretaria Municipal de Educação deverão ser referenciados, conforme as normas da ABNT.
7. Fica firmado o compromisso de retorno dos resultados à Unidade Educativa onde se desenvolveu a pesquisa e à Secretaria Municipal de Educação por meio de socialização dos dados em seminários, fóruns de debate, cursos de extensão, a critério do pesquisador, em acordo com a direção da Unidade Educativa ou SME (Central).

Agradecemos antecipadamente a sua parceria nesse processo de investigação, certos de que esta experiência será extremamente significativa, contribuindo com reflexões, proposições e indicadores que visem à qualidade da ação educativa da Rede Municipal de Ensino de Florianópolis.

Atenciosamente,

Ariane F Minussi Dutra
Assessora
Matr'cula: 27008-3

Assinatura do Pesquisador: _____

PREFEITURA DE FLORIANÓPOLIS
EDUCAÇÃO

SECRETARIA MUNICIPAL DE EDUCAÇÃO
DIRETORIA DE GESTÃO ESCOLAR
GERÊNCIA DE FORMAÇÃO CONTINUADA
GFC

AUTORIZAÇÃO 7/2019

AUTORIZAÇÃO DE PESQUISA DE MESTRADO

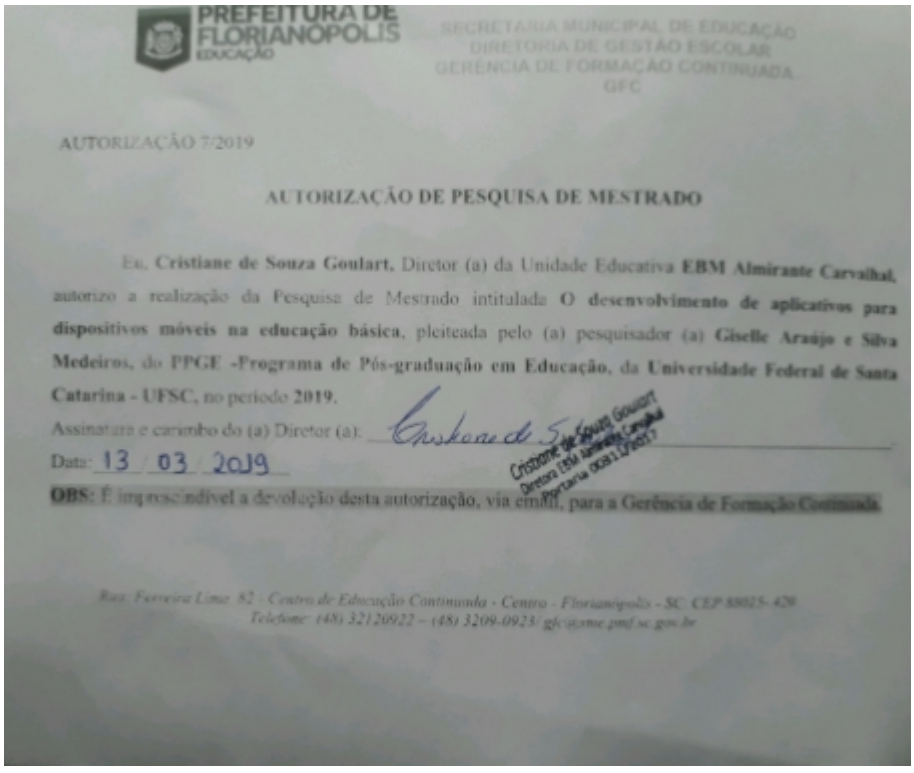
Eu, **Cristiane de Souza Goulart,** Diretor (a) da Unidade Educativa **EBM Almirante Carvalhal,** autorizo a realização da Pesquisa de Mestrado intitulada **O desenvolvimento de aplicativos para dispositivos móveis na educação básica**, pleiteada pelo (a) pesquisador (a) **Giselle Araújo e Silva Medeiros,** do **PPGE -Programa de Pós-graduação em Educação,** da **Universidade Federal de Santa Catarina - UFSC,** no período **2019.**

Assinatura e carimbo do (a) Diretor(a): _____.

Data:_____/_____/_____.

OBS: É imprescindível a devolução desta autorização, via email, para a Gerência de Formação Continuada.

Rua: Ferreira Lima, 82 - Centro de Educação Continuada - Centro - Florianópolis - SC. CEP 88015- 420
Telefone: (48) 32120922 – (48) 3209-0923/ gfc@sme.pmf.sc.gov.br


PREFEITURA DE FLORIANÓPOLIS
EDUCAÇÃO

SECRETARIA MUNICIPAL DE EDUCAÇÃO
DIRETORIA DE GESTÃO ESCOLAR
GERÊNCIA DE FORMAÇÃO CONTINUADA
GFC

AUTORIZAÇÃO 7/2019

AUTORIZAÇÃO DE PESQUISA DE MESTRADO

Eu, **Cristiane de Souza Goulart,** Diretor (a) da Unidade Educativa **EBM Almirante Carvalhal,** autorizo a realização da Pesquisa de Mestrado intitulada **O desenvolvimento de aplicativos para dispositivos móveis na educação básica**, pleiteada pelo (a) pesquisador (a) **Giselle Araújo e Silva Medeiros,** do **PPGE -Programa de Pós-graduação em Educação,** da **Universidade Federal de Santa Catarina - UFSC,** no período **2019.**

Assinatura e carimbo do (a) Diretor (a): _____

Data: 13 / 03 / 2019

Cristiane de Souza Goulart
Diretora EBM Almirante Carvalhal
Portaria 0081/2017

OBS: É imprescindível a devolução desta autorização, via email, para a Gerência de Formação Continuada.

Rua: Ferreira Lima, 82 - Centro de Educação Continuada - Centro - Florianópolis - SC. CEP 88015- 420
Telefone: (48) 32120922 – (48) 3209-0923/ gfc@sme.pmf.sc.gov.br

Anexo V

Tabela V.1. Artigo do TCC

DESENVOLVIMENTO DE UMA UNIDADE INSTRUCIONAL PARA ENSINAR O DESENVOLVIMENTO DE APPS NO ENSINO FUNDAMENTAL COM O APP INVENTOR

Raul M. Filho[1], Christiane G. von Wangenheim[1], Jean R. Hauck[1], Giselle A. S. Medeiros[1]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina - Florianópolis, SC - Brazil

raul_mf96@hotmail.com, c.wangenheim@ufsc.br, jean.hauck@ufsc.br, gisellearaujo.ufsc@gmail.com

Abstract. *Computing has become increasingly influential nowadays, helping and improving people's lives. For this reason, computing should be popularized so that students can learn key computer ideas and how to code so that they can solve problems in the easiest and most efficient possible way in their interest areas. While there are currently several initiatives to teach computing in elementary school, there are basically no instructional units covering other important concepts such as interface design. In this context, is systematically developed an instructional unit to teach programming and apps interface design as an important competency in the area of Computing following ACM/CSTA K-12 Computer Science Framework and SBC curricula in Elementary Schools. This instructional unit covers the teaching of app programming by following a software process, design thinking and interface design through the development of apps for Android phones, using the App Inventor visual programming environment. The results confirm the unit's success, achieving the learning objectives, facilitating learning and arousing interest in the students.*

Resumo. *A computação vem se tornando cada vez mais influente nos dias atuais, auxiliando e melhorando a vida das pessoas. Por esse motivo, a partir do Ensino Fundamental, a computação deve ser popularizada para que estudantes possam aprender ideias-chave de computação e a programação resolvendo problemas da maneira mais fácil e eficiente possível em suas áreas de aplicação. Mesmo existindo atualmente várias iniciativas para ensinar computação no Ensino Fundamental, basicamente não existem unidades instrucionais abrangendo também outros conceitos importantes, como o design de interface. Neste contexto, é desenvolvida uma unidade instrucional de maneira sistemática para ensinar a programação e design de interface de apps como competência importante na área de Computação alinhado aos currículos de referência do ACM/CSTA K-12 Computer Science Framework e da SBC no Ensino Fundamental. Esta unidade instrucional contempla o ensino de programação de apps seguindo um processo de software, design thinking e design de interface por meio do desenvolvimento de apps para celulares Android, utilizando o ambiente de programação visual App Inventor. Os resultados confirmam o sucesso da unidade, atingindo os objetivos de aprendizagem, facilitando a aprendizagem e despertando o interesse nos alunos.*

1. Introdução

A influência da computação é sentida por todos individualmente, em sociedade e também de forma global [CSTA 2016]. A área de computação vem possibilitando grandes inovações em todos os campos de estudo e também na vida diária das pessoas. Diversos tipos de informações podem ser transformados e processadas pelos computadores para criar apps, jogos, carros autônomos, robôs inteligentes e muito mais [CSTA 2016]. Portanto, é muito importante que as pessoas já aprendam desde pequenas os conceitos da computação.

Com o aprendizado da Computação, o estudante desenvolve seu raciocínio lógico, pensamento algorítmico, design e uma estruturada forma de resolução de problemas, conceitos e habilidades que podem ser usados muito além das salas de aula [CSTA 2011]. Essas competências proporcionam ao aluno a capacidade de implementar, testar e implantar uma solução, lidando com as restrições do mundo real, e essas habilidades são praticáveis em muitos contextos [CSTA 2011]. Esse cenário enfatiza a necessidade de oferecer uma educação que envolva diferentes conhecimentos de computação, de forma que computadores devem ser vistos muito além de ferramentas, mas, sim, um meio acessível de expressar ideias e expor a criatividade [Cambraia e Scaico 2013].

O ensino de conceitos básicos de computação nas escolas é fundamental para desenvolver o raciocínio computacional e lógico das crianças, pelo seu caráter transversal às demais ciências [Nunes 2011]. É necessário o desenvolvimento de habilidades computacionais na educação básica, pois se caracteriza como algo importante intelectualmente. Pode promover múltiplos caminhos profissionais futuros, desenvolver a capacidade de resolução de problemas, apoiar e relacionar-se com outras ciências e motivar os estudantes [CSTA 2011]. O raciocínio lógico deveria ser ensinado desde cedo, pois aumenta a capacidade de dedução e conclusão de problemas [Sica 2011]. Porém, o ensino dessas competências é reservado muitas vezes somente para o ensino superior que optam pelos cursos relacionados, mas as diversas outras áreas do saber ficam deficientes neste quesito [Sica 2011]. Atualmente o que é ensinado em escolas é o que se chama de "computer literacy" [CSTA 2016], que se trata do ensino do uso de aplicativos tais como editores de texto, imagens, apresentações, etc. O objetivo é que seja ensinado nas escolas a proficiência digital, "IT fluency" [CSTA 2011], que se refere a possibilitar os alunos a terem a capacidade de aprender e aplicar novas tecnologias de forma produtiva.

Já existem diversas iniciativas mundialmente focando no ensino de computação explorando diversas maneiras de ensino, como unidades escolares, *code clubs*, *summer camps*, etc. [MIT(c) 2019], focando principalmente no ensino de computação via programação de jogos, robôs ou *apps*. Tipicamente, neste estágio escolar, a forma como é ensinada a programação é via linguagens de programação baseadas em blocos, como *Scratch* [Scratch 2019] por exemplo.

O *App Inventor* [MIT(a) 2019] é um ambiente de programação visual baseado em blocos e permite qualquer pessoa, até mesmo uma criança, começar a programar e construir aplicativos completos para dispositivos *Android*. Criando *apps* com o *App Inventor*, estudantes aprendem a pensar criativamente, a trabalhar de forma colaborativa e a pensar de forma sistemática na solução de problemas.

Já existem unidades instrucionais voltadas ao ensino de programação de *apps* com *App Inventor*. A maioria envolve unidades instrucionais ensinando a programação de um determinado tipo de *app* por meio de vídeos explicativos [MIT(b) 2019] e tutoriais *online*, como o "Magic 8-ball" que faz previsões do futuro [MIT(d) 2019] ou oficinas [Daniel 2016]. A grande parte destes são em Inglês, com poucas exceções, como por exemplo o jogo do mosquito [Daniel 2016]. Por outro lado, surgiram também iniciativas como a *Technovation*, que estimula meninas a aprenderem a programar por meio de um desafio, que dispõe um currículo e material didático guiando o desenvolvimento de um app incluindo conceitos de negócio e de design [Technovation 2019]. Contudo, grande parte das iniciativas ensina somente a programação dos *apps*, muitas vezes sem abordar outras áreas importantes da computação como engenharia de *software*, *design thinking*, e design de interfaces. O *design thinking* fornece um processo dinâmico, participativo, e criativo para empatizar, definir, idealizar, prototipar e testar um sistema de software [Razzouk e Shute 2012] focando na resolução de problemas que valoriza e explora várias perspectivas de um problema [Chen e Huang 2017].

Desta forma, este projeto desenvolve uma unidade instrucional voltada ao ensino de computação integrando conceitos de engenharia de *software*, *design thinking* e design de interfaces para alunos do Ensino Fundamental no Brasil. A unidade é alinhada ao *framework* de currículo da CSTA [CSTA 2016] e do guia de currículo da SBC [SBC 2018]. A unidade ensina a computação por meio de programação de *apps* com *App Inventor* no contexto de uma aprendizagem baseada em problemas, abrangendo desde a parte do levantamento do problema, definição de solução, design de interface até o desenvolvimento de um aplicativo funcional. Espera-se que esta unidade instrucional criada possa ser aplicada amplamente em escolas do Brasil no ensino fundamental, popularizando não só competências de programação, mas também os conceitos de design e engenharia de *software* como elementos do aprendizado de computação.

2. Design Instrucional

O objetivo geral desta experiência relatada é o ensino de conceitos de computação por meio da programação de apps para dispositivos móveis Android utilizando a ferramenta App Inventor no Ensino Fundamental de escolas públicas brasileiras. Dessa maneira, o contexto de aplicação é formado pelo público alvo de alunos do Ensino Fundamental de escolas brasileiras com idade entre dez e quinze anos. Atualmente, os alunos nesta idade tipicamente possuem capacidade de utilizar e controlar dispositivos eletrônicos, sobretudo *smartphones* [D'Angelo 2017]. O ambiente de aplicação são escolas públicas municipais de Florianópolis que possuem uma sala informatizada com computadores e acesso à Internet. Estas sala possuem um professor responsável que normalmente possuem formação em alguma licenciatura com alguma especialização em tecnologias na educação [PMF 2019], mas não costumam ter conhecimentos na área de computação.

Os objetivos de aprendizagem envolvem principalmente os conceitos de programação e o pensamento computacional, mas também os de engenharia de *software* e design de interfaces, conteúdos focados nesta experiência. Adicionalmente, também são incluídos objetivos de aprendizagem relacionados à sustentabilidade, pelo caráter interdisciplinar da unidade instrucional com a disciplina de Ciências.

São criadas duas versões da unidade instrucional proposta, possibilitando a aplicação de forma interdisciplinar em outras disciplinas da educação básica e/ou no contra turno ou como atividade extracurricular. A versão mais curta é projetada para a disciplina de Ciências com o tema desenvolvimento e consumo sustentável, o qual compreende ensino de conceitos de computação ao mesmo tempo que os assuntos de sustentabilidade são envolvidos. A unidade em sua versão longa é projetada para ser aplicada durante um tempo mais prolongado, com diversas aulas, cada uma abordando conceitos específicos do processo de desenvolvimento de apps. As versões da UI são projetadas com o objetivo de serem ministradas de forma presencial, sendo coordenadas por um professor da educação básica com auxílio ou não de um mentor capacitado nos assuntos envolvidos na unidade. Há também a possibilidade da unidade ser aplicada à distância.

O conteúdo da unidade instrucional inclui o ensino competências de Engenharia de *Software*, de programação, de design thinking e design de interface, por meio do desenvolvimento de aplicativos com a ferramenta App Inventor. As versões da unidade são projetadas para a aplicação em uma duração de 30 e 10 horas/aula nas versões longa e curta, respectivamente. O material instrucional envolve slides, um jogo, tarefas de casa, workbooks para a documentação dos artefatos criados, vídeos, etc. (Figura 1).

Figura 1. Materiais instrucionais

3. Metodologia

O objetivo deste trabalho é o desenvolvimento, aplicação e avaliação de uma unidade instrucional com o intuito de ensinar conceitos de computação no Ensino Fundamental. Dessa forma, é realizado um estudo sobre os resultados da unidade por meio de um caso de estudo exploratório para compreender os fatos observados e coletados durante as aplicações, possibilitando a orientação para trabalhos futuros.

O estudo de caso é realizado conforme os procedimentos propostos por Wohlin et al. (2012) e Yin (2013).

Definição do estudo: a definição do estudo é realizada com base na proposição da pergunta de pesquisa, hipóteses e design de pesquisa. Da pergunta de pesquisa são, então, derivadas quatro perguntas de análise as quais são sistematicamente decompostas em medidas para a coleta de dados conforme a abordagem GQM [Basili et al. 1994]. A coleta de dados é realizada por meio sobretudo de questionários.

Execução do estudo: como abordagem de design instrucional é utilizado o modelo ADDIE [Branch 2009], o qual possibilita o desenvolvimento sistemático da unidade. Aqui são realizadas duas etapas, a primeira se caracteriza pelo design da unidade, levantando as características dos aprendizagem e do ambiente. Com isso, são definidos e/ou produzidos os objetivos de aprendizagem, planos de ensino, materiais instrucionais, métodos e estratégias instrucionais a serem utilizados. Na segunda etapa, a unidade instrucional é aplicada em suas duas versões, coletando os dados sobre sua realização com as formas definidas para tal.

Análise e interpretação do estudo: os dados coletados das duas aplicações da unidade instrucional são, então, analisados e interpretados com base em métodos quantitativos e qualitativos, gerando os resultados e conclusões finais.

4. Resultados

Foram realizadas duas aplicações das versões da unidade instrucional nos anos de 2018 e 2019 na escola Almirante Carvalhal na cidade de Florianópolis. A quantidade de respostas e os dados demográficos podem ser observados na Tabela 1.

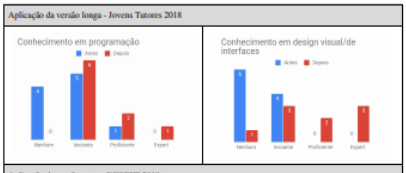
Tabela 1. Informações das duas aplicações realizadas

Aplicação da versão longa - Jovens Tutores 2018	
Quantidade de respostas	9
Quantidade de meninas	5
Quantidade de meninos	5
Média de idade	14 anos
Aplicação da versão curta - ECOPET 2019	
Quantidade de respostas	24
Quantidade de meninas	12
Quantidade de meninos	14
Média de idade	10 anos

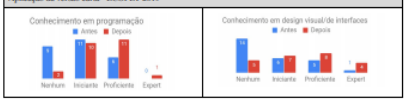
Os resultados da análise dos dados coletados durante as duas aplicações indicam uma avaliação positiva da unidade instrucional (Tabela 2). Em ambas, os alunos seguiram todo o processo de desenvolvimento proposto e obtiveram um acréscimo em seu aprendizado e um ótimo resultado nos apps desenvolvidos. Foi notável que a maioria dos alunos estava motivada e interessada em aprender mais sobre os conceitos de computação, o que acarretou em uma grande experiência para eles. A percepção do aumento do aprendizado foi algo notório pela grande maioria dos alunos. Os resultados indicam, para as duas aplicações, que antes e depois dos cursos há uma significativa diferença na aprendizagem percebida pelos alunos. Ter havido alunos que disseram ter nível "expert" em certos conteúdos, faz crer que os resultados foram ainda melhor do que o esperado e que estes alunos têm confiança de que o que aprenderam pode ser fielmente reproduzido por eles.

Tabela 2. Autoavaliação sobre o conhecimento nos conceitos de computação

Aplicação da versão longa - Jovens Tutores 2018



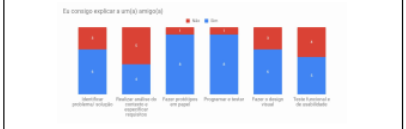
Aplicação da versão curta - ECOPET 2019



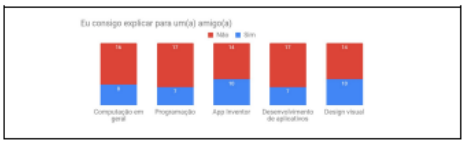
Enquanto na primeira aplicação os alunos, em geral, se sentem capacitados a explicar a maioria dos conteúdos, na segunda, os participantes não demonstram habilidade e segurança para ensinar os conteúdos ensinados (Tabela 3). Uma das respostas para esta diferença é que os alunos na primeira aplicação tiveram mais tempo para aprender os conteúdos e também um contato mais direto com os instrutores, por serem em menor quantidade. Também, na segunda aplicação os alunos eram em média 4 anos mais novos e, tipicamente, não é esperado que alunos nessa idade tenham a habilidade de transferir conhecimentos, ainda mais algo tão novo e complexo para eles.

Tabela 3. Capacidade percebida pelos alunos em ensinar os conceitos vistos nas aulas

Aplicação da versão longa - Jovens Tutores 2018



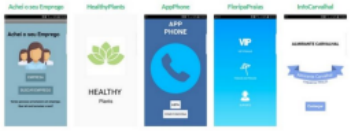
Aplicação da versão curta - ECOPET 2019



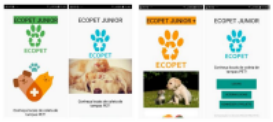
O desempenho dos alunos na primeira aplicação foi algo que chamou muito a atenção, superando expectativas pela qualidade dos *apps* criados (Tabela 4). Este resultado demonstra que a utilização de um processo de desenvolvimento de apps bem definido, juntamente aos conceitos de *design thinking*, permite construir aplicativos úteis e com uma excelente interface com o usuário. Pela avaliação dos artefatos desenvolvidos com base nos *workbooks*, é possível perceber que os alunos conseguiram compreender e produzir as etapas necessárias no processo de desenvolvimento de uma maneira adequada, comprovando a qualidade do resultado final. Em relação aos conceitos de engenharia de *software* como histórias de usuário e testes funcionais os alunos demonstram um significativo grau de conhecimento, demonstrando que foram conteúdos bem aceitos por eles. Já os conteúdos envolvendo casos de uso e funcionamento das tarefas dos *apps*, os alunos tiveram grande dificuldade para aplicar e mesmo com certa ajuda os resultados não foram satisfatórios. Na segunda aplicação, o desempenho dos alunos em relação aos *apps* desenvolvidos não foi tão satisfatória quanto na primeira (Tabela 4). Os alunos demonstraram mais dificuldade em aplicar os conteúdos estudados em sala e o resultado foi percebido nos *apps* construídos. Um dos fatores que influenciou este resultado negativo é a quantidade de alunos em relação ao tamanho da sala. Pelo fato da sala estar mais cheia, para prender a atenção dos alunos é muito mais difícil e distrações afetam o aprendizado dos conteúdos ensinados. Outro fato que fez a diferença é a idade dos alunos, diversos fatores podem ser afetados por causa disso. O interesse dos alunos se torna mais limitado, a capacidade de prestar atenção é afetada, a experiência em resolver problemas e a preferência pela diversão em vez de um bom resultado. Como o tempo geral do curso também foi reduzido em relação à primeira aplicação, as aulas tiveram que ser comprimidas ou não ensinadas, justificando os resultados inferiores.

Tabela 4. Apps desenvolvidos nas duas aplicações

Aplicação da versão longa - Jovens Tutores 2018



Aplicação da versão curta - ECOPET 2019

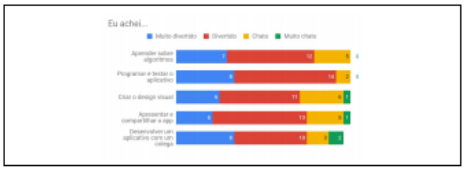


A participação nas aplicações da unidade instrucional se mostrou algo muito prazeroso para os alunos, que acreditam ter sido uma experiência divertida (Tabela 5). Este fato demonstra que as aulas foram aplicadas de uma forma agradável e o nível de ensino dos conteúdos foi adequado a maioria dos alunos. A percepção do tempo durante as aulas foi outro parâmetro que indica a boa experiência do curso. Boa parte dos estudantes aponta que não sentiu a passagem do tempo durante o ensino dos conteúdos em sala, o que demonstra que eles voltaram seu foco na instrução e estavam interessados no aprendizado dos conceitos apresentados. Outro ponto que chama atenção é a apreciação pelo desenvolvimento dos apps com um colega, apresentada pela grande fração dos alunos. O desenvolvimento em equipe estimula a discussão entre os alunos, sempre buscando a melhor maneira de resolver os problemas ou na hora de decidir decisões de projeto. Além disto, a construção do app fica sujeita a análise por duas pessoas, o que leva a um resultado com mais qualidade e menos defeitos.

Tabela 5. Experiência de aprendizado nas duas aplicações

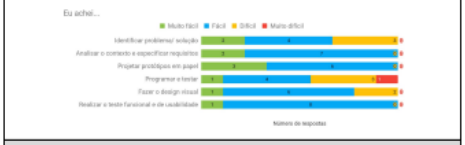
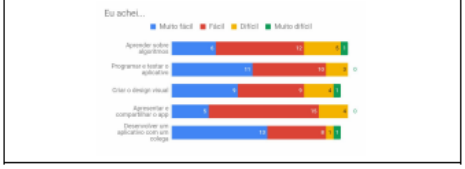
Aplicação da versão longa - Jovens Tutores 2018

Aplicação da versão curta - ECOPET 2019



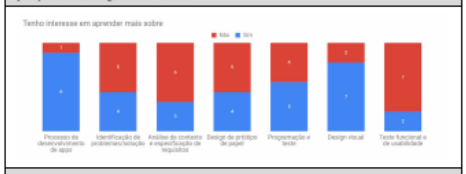
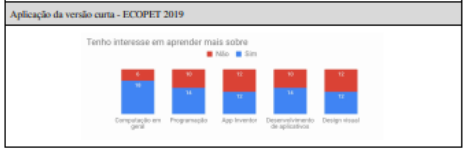
A utilização da ferramenta *App Inventor* e a forma como as aulas foram planejadas auxiliou para que os alunos avaliassem o aprendizado dos conteúdos como fáceis (Tabela 6). Por mais que a maioria dos conceitos ensinados tenham um grau de complexidade elevado, os participantes das aplicações sentiram um nível adequado de aprendizagem, caracterizando com um certo grau de facilidade.

Tabela 6. Facilidade de aprendizado nas duas aplicações

Aplicação da versão longa - Jovens Tutores 2018

Aplicação da versão curta - ECOPET 2019


A adoção de um processo de desenvolvimento de *apps* com *design thinking* envolvendo conceitos design visual serviu como incentivo para que os alunos se interessassem mais pelos conteúdos, tornando-os mais criativos e participativos (Tabela 7). Também permitiu a construção de diversos *apps* úteis que resolvem problemas da sociedade, permitindo que os alunos se envolvam na mudança do mundo em que vivem de maneira prática. A percepção da importância dos conteúdos ensinados também comprova que os alunos compreendem que no futuro essa aprendizagem se tornará necessária a diferença no mundo em que vivem.

Tabela 7. Interesse em aprender mais sobre os conteúdos nas duas aplicações

Aplicação da versão longa - Jovens Tutores 2018

Aplicação da versão curta - ECOPET 2019


5. Conclusão

A finalidade deste trabalho foi o desenvolvimento de uma unidade instrucional para ensinar o desenvolvimento de aplicativos para alunos do ensino fundamental de escolas brasileiras com a ferramenta App Inventor, envolvendo conceitos de engenharia de software e design de interfaces. A unidade foi aplicada em dois momentos diferentes, em uma versão longa/completa e outra mais curta.

A versão longa envolveu 10 alunos do ensino fundamental, com idade entre 13 e 15 anos, na produção de um aplicativo funcional. A avaliação desta aplicação permitiu notar que as expectativas foram cumpridas e os resultados foram muito além do esperado. A maioria dos conteúdos ensinados foram bem aceitos pelos alunos e as competências e o interesse dos alunos tiveram um acréscimo notável. Na aplicação mais curta, foram envolvidos 28 alunos do ensino fundamental, com idade entre 9 e 11 anos, no aprimoramento de um aplicativo já desenvolvido focando mais na construção do design de interfaces de maneira interdisciplinar com a disciplina de Ciências com o tema de sustentabilidade. A avaliação desta aplicação mostrou que o tempo mais curto, a idade e a quantidade de alunos tiveram forte influência para que os resultados não fossem tão bons quanto na primeira aplicação. Entretanto, em geral, os alunos se mostraram interessados em aprender mais sobre os conteúdos e acreditam que foi uma experiência divertida ter participado do curso.

Como resultado deste trabalho, é disponibilizada uma unidade instrucional para ensinar o processo de desenvolvimento de apps por meio do desenvolvimento de aplicativos abrangendo conceitos de engenharia de software e design de interfaces. Indica-se que esta unidade seja aplicada para alunos a partir do ensino fundamental, com idade acima de 14 anos de idade com ou sem experiência prévia em computação/programação. A ideia geral desta unidade é capacitar os alunos à produção de *apps* utilizando um processo de desenvolvimento adequado, desenvolvendo habilidades de programação e design. Sua aplicação pode ser inserida tanto de forma interdisciplinar quanto extracurricular e, até mesmo, a distância.

Referências

Basili, V. R. et al. (1994) "The Goal Question Metric Approach", http://www.cs.umd.edu/~mvz/handouts/gqm.pdf, jul. 2019.

Branch, R. M. (2009) Instructional Design: The ADDIE Approach. New York, New York, USA, Springer Science & Business Media.

Cambraia, A. C. e Scaico, P. D. (2017) "Os desafios da Educação em Computação no Brasil: um relato de experiências com Projetos PIBID no Sul e Nordeste do país". Revista Espaço Acadêmico, n. 194.

CSTA. (2016) "K-12 Computer Science Framework", http://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf, jul. 2019.

CSTA. (2011) "K-12 Computer Science Standards" https://www.csteachers.org/page/standards, jul. 2019.

D'Angelo, P. (2017) "Pesquisa sobre smartphones: a relação dos pais e das crianças com smartphones no Brasil", https://blog.opinionbox.com/pesquisa-smartphones-criancas/, jul.. 2019.

Daniel, G. T. (2016) Design de unidade instrucional de desenvolvimento de aplicativos para o ensino fundamental, Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis.

MIT (a). (2019) "About us", http://appinventor.mit.edu/explore/about-us.html, jul. 2019.

MIT (b). (2019) "Beginner Video Tutorials", http://appinventor.mit.edu/explore/ai2/beginner-videos.html, jul. 2019.

MIT (c). (2019) "News & Events", http://appinventor.mit.edu/explore/news-events.html, jul. 2019.

MIT (d). (2019) "Magic 8-Ball Tutorial", http://appinventor.mit.edu/explore/teach/magic-8-ball.html, jul. 2019.

Nunes, D. J. (2012) Ciência da Computação na Educação Básica. Anais do XX Workshop sobre Educação em Computação (WEI), Curitiba, Paraná.

PMF. (2019) "Processo Seletivo De Substitutos - Edital nº 004/2018", http://substituto2019.fepese.org.br/?go=download&arquivo=2018_PMF_Substitutos_Ed_04.pdf, jul. 2019.

SBC. (2017) "Referenciais de Formação em Computação: Educação Básica", http://www.sbc.org.br/files/ComputacaoEducacaoBasica-versaofinal-julho2017.pdf, jul. 2019.

Scratch. (2019) "Acerca do Scratch", https://scratch.mit.edu/about, jul. 2019.

Sica, C. (2011) "Ciência da Computação no Ensino Básico e Médio", http://www.odiario.com/blogs/carlossica/2011/10/07/ciencia-da-computacao-no-ensino-medio/, jul. 2019.

Technovation. (2019) "Currículo", http://www.technovationbrasil.org/currículo, jul. 2019.

Wohlin, C. et al. (2012) Experimentation in Software Engineering. Berlin: Springer-Verlag Berlin Heidelberg, p. 235.

Yin, R. K. (2009) Case Study Research: Design and Methods. SAGE, 4 ed., p. 219.