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

Desenvolvimento de um Analisador de Design de Interface no Contexto do Ensino de Computação com o App Inventor

Karla Aparecida Justen

Florianópolis
2018
Universidade Federal de Santa Catarina
Departamento de Informática e Estatística

Desenvolvimento de um Analisador de Design de Interface no Contexto do Ensino de Computação com o App Inventor

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do Grau de Bacharelado em Ciências da Computação.

Autora: Karla Aparecida Justen

Orientadora: Christiane Gresse von Wangenheim
Co-Orientador: Jean Carlo R. Hauck

Florianópolis
2018.1

Karla Aparecida Justen

Desenvolvimento de um Analisador de Design de Interface no Contexto do Ensino de Computação 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, 17 de junho de 2019

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

Prof. Dr. Jean Carlo R. Hauck
Co-orientador
Universidade Federal de Santa Catarina

Msc. Nathalia da Cruz Alves
Avaliadora
Universidade Federal de Santa Catarina

Florianópolis
2018

Resumo

O uso cotidiano de tecnologias de informação e comunicação pela população tem se tornado um componente imprescindível, tanto na vida pessoal como na carreira profissional. Por isso, a inclusão do ensino dos princípios básicos da Ciência da Computação na Educação Básica acaba sendo conhecimento útil para o cidadão como um usuário de tecnologias de informação, mas também o torna capacitado para ser criador. De forma a incluir o ensino da computação na Educação Básica, há iniciativas que propõem a utilização de ambientes de programação visual baseado em blocos, como Scratch, Snap! e App Inventor. Para auxiliar estudantes a aprender competências básicas de computação incluindo o pensamento computacional por meio da programação, aprimorar a capacidade de raciocínio lógico para resolução de problemas, inovação e criatividade. Uma forma de ensinar computação é por meio de criação de aplicativos para dispositivos Android, com auxílio do App Inventor. Além da programação é essencial o design de interface, pois procura trazer aos usuários a melhor usabilidade. Isso torna importante ensinar não só a programação, mas também conceitos básicos de design de interface. Para proporcionar um processo de aprendizagem consistente é importante apresentar um *feedback* em forma de avaliação. Isso é uma tarefa complicada e trabalhosa para os professores. Visando facilitar a avaliação, existem ferramentas que analisam automaticamente os projetos de programação dos alunos e apresentam um *feedback* sobre a variedade e complexidade da programação realizada, como por exemplo Dr. Scratch e CodeMaster. Considerando projetos desenvolvidos pelo App Inventor, ainda não há ferramentas que avaliam o design de interfaces de apps desenvolvidos de forma automatizada. Assim, o objetivo deste trabalho é a aprimoração da ferramenta web CodeMaster, de forma a incluir a análise e avaliação automatizada de design de interface dos aplicativos desenvolvidos com App Inventor. Desta forma, o presente trabalho visa levantar o estado da arte, mapear as heurísticas de usabilidade a partir de um guia de design, elaborar uma rubrica, desenvolver o sistema e, por fim, realizar testes de corretude e validade. Assim, como resultado é apresentada uma rubrica que avalia o design de interface de aplicativos desenvolvidos no App Inventor. Também é apresentada a ferramenta CodeMaster, com a avaliação automatizada da rubrica. Com o objetivo de auxiliar professores da Educação Básica na avaliação de projetos e deste modo facilitar o ensino e popularização de computação e design de interfaces no Brasil.

Palavras chave: design de interface, App Inventor, avaliação automática.

LISTA DE FIGURAS

Figura 1 - Métodos Instrucionais, adaptado a partir de (SASKATCHEWAN 1991)..... 16

Figura 2 - Exemplo de uma Rubrica (GRESSE VON WANGENHEIM et al. 2018)..... 20

Figura 3 - Níveis de ensino de computação (CSTA, 2017)..... 23

Figura 4 - Diretriz de contraste cada nível WCAG para texto pequeno e grande...27

Figura 5 - Exemplo de linguagem de programação visual baseada em blocos (CSTA, 2017)..... 28

Figura 6 - Tela Blocos do App Inventor (APP INVENTOR, 2018)..... 29

Figura 7 - Tela "Designer" do App Inventor (APP INVENTOR, 2018)..... 30

Figura 8 - Opções para customizar cores no AppInventor…..... 32

Figura 9 - Tipos de famílias de fontes disponibilizadas no App Inventor…..…… 33

Figura 10 - Opções para definição da largura ou altura de um componente..... 33

Figura 11 - Processo de análise estática (STRIEWE, GOEDICKE, 2014)..... 34

Figura 12 - Visão do fluxo do verificador adaptado voltado ao ensino (SINTHSIRIMANA, PANJABUREE, 2013)..... 35

Figura 13 - Quantidade de publicações sobre avaliação do design de interface de smartphone mobile..... 41

Figura 14 - Questionário por ZAPATA et al (2014)..... 42

Figura 15 - Rubrica para avaliação de aplicativos desenvolvido pela Technovation (2014)..... 43

Figura 16 - Rubrica Parcial (SHERMAN, MARTIN, 2015)..... 43

Figura 17 - Lista de critérios para avaliação do design de interface (HOEHLE, ALJAFARI, VENKATESH, 2016)..... 44

Figura 18 - Lista de métricas de qualidade do design de interface (INES et al., 2017)..... 45

Figura 19 - Lista de defeitos do design de interface (INES et al., 2017)..... 45

Figura 20 - Rubrica parcial para avaliação de projetos App Inventor (GRESSE VON WANGENHEIM et al. 2018)……..... 46

Figura 21 - Checklist para avaliação do projeto visual e interação com usuário dos aplicativos Android (ANDROID, 2018)..... 47

Figura 22 - Lista Parcial do procedimentos de avaliação de aplicativos Android (ANDROID, 2018)..... 48

Figura 23 - Telas do app HealthyPlants..... 57

Figura 24 - Diagrama de Casos de Uso do CodeMaster……...66

Figura 25 - Diagrama de componentes (DEMETRIO, 2017) ...……..70

Figura 26 - Sequência da análise do projeto App Inventor ……....72

Figura 27 - Diagrama de classes simplificado ……..73

Figura 28 - Diagrama de classes do cadastro de administrador ……..... 75

Figura 29 - Diagrama de classes da geração da planilha XLSX ……...76

Figura 30 - Diagrama Entidade Relacionamento do BD CodeMaster……...78

Figura 31 - Menu principal do site ……..79

Figura 32 - Página referente ao caso de uso USC07…….79

Figura 33 - Página referente ao caso de uso USC09 ……79

Figura 34 - Página referente ao caso de uso USC01……..80

Figura 35 - Tela referente ao caso de uso USC010 …. ……..81

Figura 36 - Página referente ao caso de uso USC04 e USC05……...82

Figura 37 - Trecho de código final do método gradeProject().....83

Figura 38 - Trecho de código que interpreta o JsonArray em aluno_resultado_appinventor.jsp…….……83

Figura 39 - Classe DimensionamentoResponsivo.java ……..……84

Figura 40 - Trecho da classe ProjectSettings.java ...……..……..……84

Figura 41 - Trecho da classe AppInventorGradeDesign.java..……..…..……85

Figura 42 - Trecho de código que referência o critério dimensionamento responsivo no método analiseVisual() ……....85

Figura 43 - Código para cada critério na lista de escolha de avaliação pelo professor ……..86

Figura 44 - Método getProjectSettings() da classe Upload.java……....86

LISTA DE TABELAS

Tabela 1 - Métodos Instrucionais (SASKATCHEWAN, 1991)..... 16

Tabela 2 - Tipos de *feedback* instrucional classificados por abordagem (PHYE, BENDER, 1989)..... 18

Tabela 3 - Exemplos de tipos de notas expressadas em avaliações somativas..... 18

Tabela 4 - Competências básicas (CSTA, 2017) 22

Tabela 5 - Conceitos Fundamentais (CSTA, 2017)..... 22

Tabela 6 - Práticas Fundamentais (CSTA 2017)..... 22

Tabela 7 - Conceitos Transversais(CSTA, 2017)..... 23

Tabela 8 - Componentes de design de interface de usuário disponíveis no App Inventor (2018)..... 25

Tabela 9 - Meta-princípios da linguagem visual (SCHLATTER; LEVINSON, 2013)..26

Tabela 10 - Design de botões conforme o *Material Design*..... 26

Tabela 11 - Definição de texto pequeno e grande ……..... 27

Tabela 12 - Componentes do Designer do App Inventor…… 30

Tabela 13 - Termos de busca e respectivos sinônimos ou termos similares……. 36

Tabela 14 - *Strings* de buscas usadas nas bases de dados……..... 37

Tabela 15 - Quantidade de artigos resultantes em cada etapa da execução da busca……. 38

Tabela 16 - Especificação das informações extraídas……..…….. 39

Tabela 17 - Documentos resultantes da execução de busca……. 40

Tabela 18 - Checklist de usabilidade de aplicativos para smartphones v0.2 (GRESSE VON WANGENHEIM et al., 2018)..... 49

Tabela 19 - Métodos de desenvolvimento de cada proposta de avaliação……... 50

Tabela 20 - Métodos de Avaliação da qualidade da checklist/questionário/rubrica/ ferramenta…….. 51

Tabela 21 - Objetivos de aprendizagem da UI (MISSFELDT FILHO, 2018)..... 55

Tabela 22 - Plano de ensino da unidade instrucional (MISSFELDT FILHO, 2018).. 56

Tabela 23 - Rubrica CodeMaster UI Design - App Inventor ……...…. 58

Tabela 24 - Exemplificação de atribuição de nota pela rubrica CodeMaster UI Design - App Inventor ……..……..….. 62

Tabela 25 - Requisitos Funcionais……..……..….. 64

Tabela 26 - Requisitos Não Funcionais ……... 65

Tabela 27 - Critério Dimensionamento Responsivo ……...84

Tabela 28 - Resultados dos testes considerando o critério “Não usar itálico”..... 87

Tabela 29 - Aplicativos App Inventor selecionados para avaliação da corretude.... 87

Tabela 30 - Resultado da segunda etapa do teste de corretude…..……..……. 89

Tabela 31 - Resultados da análise de correlação policórica…...……..... 92

Tabela 32 - Resultados da análise de correlação item-total e alfa de Cronbach…...94

SUMÁRIO

1. Introdução 9
1.1 Contextualização 9
1.2 Objetivos 11
1.3 Metodologia de Pesquisa e Trabalho 12
1.4 Estrutura do Documento 14
2. Fundamentação Teórica 15
2.1 Ensino e Aprendizagem 15
2.1.1 Avaliação da aprendizagem do aluno 17
2.2 Ensino de Computação na Educação Básica 20
2.3 Design de Interface do Usuário 24
2.4 App Inventor 28
2.5 Avaliação automatizada de código 33
3. Estado da Arte e Prática 36
3.1 Definição do protocolo do mapeamento 36
3.2 Execução da Busca 38
3.3 Análise dos resultados 39
3.4 Discussão 52
4. Desenvolvimento do modelo de avaliação de design de interface de apps 54
4.1 Análise do contexto 54
4.2 Desenvolvimento da rubrica para avaliação do design visual 57
4.3 Feedback Instrucional 61
5. Automação da avaliação integrada ao CodeMaster 64
5.1 Análise de requisitos 66
5.2 Definição do Caso de Uso 66
5.3 Modelagem e Implementação do Sistema 70
5.3.1 Arquitetura e Implementação do Módulo de "Análise e Avaliação" 71
5.3.2 Arquitetura e Implementação do Módulo de “Apresentação” 74
5.3.3 Arquitetura e Implementação da “Camada de Persistência” 77
5.3.4 Implementação do Design de Interface 79
5.4 Teste de Corretude 87
6. Avaliação da confiabilidade e de validade 91
6.1 Definição e Execução da Validação 91
6.2 Análise dos dados 91
6.3 Discussão 96
7. Conclusão 98
Referências Bibliográficas 100
Apêndice A 112
Apêndice B 117
Apêndice C 123
Apêndice D 127
Apêndice E 145

1. Introdução

1.1 Contextualização

A difusão de tecnologias de informação e comunicação (TIC) em diversos setores econômicos tem causado mudanças organizacionais. Consequentemente, surgem empregos que exigem médio e alto nível de qualificação. Para suprir a carência de mão-de-obra qualificada é importante que todo cidadão tenha conhecimento dos princípios básicos da Ciência da Computação. Não apenas para saber utilizar os recursos tecnológicos atuais, mas também aprimorar a capacidade de raciocínio lógico para resolução de problemas, inovação e criatividade (CSTA, 2017), desenvolvendo assim o pensamento computacional (WING, 2006). O que acaba sendo conhecimento útil não apenas para aqueles que seguirão a profissão na área da Ciência da Computação, mas para todas as demais profissões.

Com esse propósito existem diferentes iniciativas (TECHNOVATION, 2017) (CSTA, 2017) (CODE.ORG, 2017) (COMPUTAÇÃO NA ESCOLA, 2019) que incentivam e motivam o ensino de programação para crianças da Educação Básica. Para isso, linguagens de programação baseada em blocos são tipicamente utilizadas, pois facilitam a focar na estrutura e na lógica do projeto, quando comparadas com linguagens baseadas em texto, que exigem conhecimento de sintaxe (KELLEHER; PAUSCH, 2005). Outra maneira de instigar o interesse, é utilizar da relação que as crianças têm com dispositivos móveis, baseada em entretenimento e diversão, para usar como ferramenta para aprimorar o pensamento computacional (DUDA; SILVA, 2015).

Deste modo, a Google desenvolveu a ferramenta de código-aberto App Inventor (2018), que é mantido pelo Instituto de Tecnologia de Massachusetts (MIT). App Inventor auxilia no desenvolvimento de aplicativos para dispositivos Android com suporte para design de interface e para a programação funcional do aplicativo. Essa ferramenta é um ambiente de programação em blocos, como por exemplo Scratch (2018) e Snap! (2018). Isso tem como objetivo proporcionar às pessoas com pouco ou nenhum conhecimento de programação a experiência de desenvolver aplicativos para dispositivos Android (2018) de forma intuitiva.

Nesse contexto, diferentes unidades instrucionais vêm aplicando esses recursos com seus alunos, tanto no cenário formal escolar como no informal, como *code clubs* e acampamentos de verão (OLIVEIRA et al., 2014) (CS FIRST, 2017). Essas unidades instrucionais abordam diferentes tipos de atividades variando em abordagens de problemas fechados, nos quais existe uma solução correta, como os exercícios da Hour of Code (2017). Até abordagens de aprendizado baseado em problemas[1], que permite produzir projetos para resolver problemas de forma livre não existindo soluções predefinidas (LYE, KOH, 2014), o que acaba dificultando a avaliação e o *feedback* (ESERYEL et al., 2013).

[1] Aprendizado baseado em problemas: em inglês *Problem-Based Learning*, sigla PBL

A avaliação e *feedback* instrucional são muito importantes para que o processo de ensino e aprendizagem seja bem sucedido (HATTIE; TIMPERLEY, 2007). Para isso a avaliação de desempenho pode ser usada para diferentes objetivos (GAGNÉ, BRIGGS, WAGER, 1992). Pode ser usada como forma de avaliar se a pessoa tem os conhecimentos ou pré-requisitos para aprender o que será ensinado. Outro motivo, seria avaliar o que o estudante está aprendendo durante uma unidade instrucional e remediar qualquer mal entendido antes de continuá-la, considerada avaliação formativa (GARRISON, EHRINGHAUS, 2013). Pode ser aplicada também ao final da unidade instrucional como forma de informar o progresso também aos pais, professores administradores, chamada de avaliação somativa (GARRISON, EHRINGHAUS, 2013).

Para operacionalizar a avaliação é importante definir o que e como será mensurado para auxiliar no que os alunos precisam focar (GAGNÉ; BRIGGS; WAGER, 1992). Para apresentar um *feedback* de um trabalho quando há uma resposta correta é simples, pois basta comparar as respostas do estudante com a solução correta (FORSYTHE; WIRTH, 1965). Porém, no contexto de aprendizagem baseada em problemas em que não existe um gabarito pré-definido, tipicamente se adotam rubricas. As rubricas apresentam uma lista de critérios e os níveis de desempenho alcançáveis em cada critério (BIAGIOTTI, 2005). Com base no peso de cada nível alcançado é possível apresentar uma pontuação e convertê-la em uma nota.

A realização de avaliações das atividades de programação é trabalhosa, pois é preciso analisar e corrigir a implementação desenvolvida por cada estudante. Também é necessário que haja competência do professor que avalia ter conhecimento de computação, porém atualmente faltam professores de computação na Educação Básica (TIC EDUCAÇÃO, 2016). Por isso professores de outras áreas e entusiastas com a iniciativa de ensinar computação têm se prontificado, o que acaba tornando mais desafiador o processo de avaliação das práticas (DELUCA; KLINGER, 2010). Outra desvantagem de uma avaliação manual pelo instrutor é a possibilidade de haver inconsistências, inexatidão e bias (ZEN et al., 2011).

Uma maneira de auxiliar professores a avaliarem projetos de uma turma inteira ao mesmo tempo é utilizando ferramentas que automatizam a avaliação de projetos de programação. Existem várias ferramentas para ensino superior para lidar com linguagens textuais (IHANTOLA et al., 2010). Porém existem poucas para linguagens de programação baseada em blocos, como Dr. Scratch (2017), focando na avaliação de projetos Scratch (2018); NinjaCode Village (OTA et al., 2016); e a ferramenta CodeMaster (2019) para avaliação de projetos App Inventor e Snap!.

Essas ferramentas analisam o projeto em duas etapas. A primeira etapa consiste em contabilizar a frequência de uso de blocos de implementação de cada tipo (por exemplo, condicional e repetição). A segunda etapa apresenta uma avaliação, com base numa rubrica, que gera uma nota. Assim podem ser utilizadas pelo professor, para avaliar os trabalhos de uma turma, e por alunos para avaliar com antecedência o progresso do seu projeto, antes de entregar ao professor (WILCOX, 2016).

Porém atualmente existe apenas uma ferramenta que automatiza a avaliação de projetos App Inventor focando exclusivamente na avaliação de conceitos de programação, a CodeMaster (2019). Mas, como parte do desenvolvimento de aplicativos de sucesso, uma competência importante é o design de interface.

O design de interface de usuário para dispositivos eletrônicos, no geral, tem como foco a maximização da usabilidade e a experiência do usuário (NIELSEN, 1993). O objetivo do design da interface do usuário é tornar a interação do usuário tão simples e eficiente quanto possível. Para obtenção desses resultados, o projeto de interface pode seguir por guias de estilo como o *Material Design* (2017). *Material Design* (2017) foca no design de interfaces para aplicativos Android fornecendo recomendações para o design de diversos elementos de interface, como cores, fontes, layout e componentes (PORTO; GRESSE VON WANGENHEIM; BARBOSA, 2017).

Assim unidades instrucionais ensinando o desenvolvimento de aplicativos *mobile* também devem abranger este tipo de competência. Por exemplo o currículo da Technovation (2017) e o curso “Faça o seu app”, oferecido pela Computação na Escola (2017). Consequentemente a avaliação da aprendizagem do design de interface também deve ser feita. Porém, atualmente não existe nenhuma ferramenta que fornece este suporte (PORTO; GRESSE VON WANGENHEIM; BARBOSA, 2017).

Assim, neste trabalho é proposta a aprimoração da ferramenta CodeMaster (2019), para incluir uma forma de avaliar o atendimento dos objetivos de aprendizagem no desenvolvimento de aplicativo em relação ao design de interface de aplicativos desenvolvidos com App Inventor. A avaliação é baseada no conjunto de heurísticas, apresentadas nas diretrizes definidas por guias de estilos (Material Design, 2017)(W3C, 2018). Como resultado espera-se fornecer uma ferramenta que auxilia professores da Educação Básica na avaliação de projetos. Com isso espera-se facilitar o ensino e popularização de computação e design de interfaces no Brasil.

1.2 Objetivos

Objetivo geral

O objetivo deste trabalho é aprimorar a ferramenta web CodeMaster, para fazer análise e avaliação automática de design de interface de aplicativos desenvolvidos com o App Inventor, a ser utilizada em unidades instrucionais no ensino de computação na Educação Básica.

Objetivos específicos

Os objetivos específicos deste trabalho são:

O1. Analisar a fundamentação teórica sobre ensino e aprendizagem, especificamente o ensino de computação na Educação Básica. Estudar a ferramenta

de desenvolvimento App Inventor. Analisar a importância do design de interface de aplicativos.

O2. Analisar o estado da arte sobre ferramentas que fazem análise de design de interface de aplicativos para *smartphones touchscreen* da plataforma Android ou desenvolvidos na ferramenta App Inventor.

O3. Aprimorar a ferramenta CodeMaster com o desenvolvimento da análise automatizada de design de interface de aplicativos, desenvolvidos em App Inventor.

O4. Avaliar a ferramenta criada em relação a confiabilidade e validade.

Premissas e restrições

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

A aprimoração a ser desenvolvida tem como foco a análise e avaliação de design de interface, somente de projetos desenvolvidos com a ferramenta de desenvolvimento App Inventor no ensino de computação na Educação Básica.

Este trabalho pretende tratar apenas os conceitos de design de interface de aplicativos para dispositivos Android. Sendo os conceitos trabalhados também restringidos aos recursos oferecidos pela ferramenta web App Inventor.

1.3 Metodologia de Pesquisa e Trabalho

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

Etapa 1 – Fundamentação teórica

Atividade focada em estudar, analisar e sintetizar os conceitos principais e a teoria referente aos temas a serem abordados neste trabalho. Nesta etapa são realizadas as seguintes subatividades:

A1.1 – Análise teórica sobre ensino, aprendizagem, unidades instrucionais e avaliação, especificamente o ensino de computação na Educação Básica.

A1.2 – Análise teórica sobre ambientes de programação em blocos, ambiente de desenvolvimento App Inventor.

Etapa 2 – Mapeamento Sistemático de Literatura

Nesta etapa é realizado um mapeamento sistemático de literatura seguindo o processo proposto por Petersen et al. (2008) para identificar e estudar analisadores de design de interface de ambientes de programação visual (baseado em blocos) atualmente sendo utilizados. O processo de mapeamento se inicia com a definição do escopo da pesquisa, pergunta de pesquisa, palavras-chaves e critérios. Com o protocolo de busca definido, é realizada a execução da busca. Durante a execução de busca o conjunto de palavras-chaves a partir dos resumos e conclusões dos

documentos são sempre reavaliados. Caso necessário, alterações no protocolo de busca são feitas e a execução reiniciada. Em seguida é realizada a extração das informações dos documentos, filtrando os documentos que respondem a pergunta de pesquisa definida no protocolo de busca. O mapeamento sistemático é finalizado com a análise das informações dos documentos restantes da etapa anterior. Resumidamente, esta atividade é dividida então subdividida em:

A2.1 – Definição do protocolo de busca.
A2.2 – Execução da busca.
A2.3 – Extração dos dados
A2.4 – Análise.

Etapa 3 – Desenvolvimento do modelo de avaliação

Para o desenvolvimento do modelo de avaliação do analisador automático de design de interface foi seguida a metodologia de design instrucional ADDIE[2] (Branch, 2009). Esse modelo é realizado em 5 fases. **Análise** é a primeira fase que consiste na análise do contexto do público alvo e a identificação das razões para o baixo desempenho no conteúdo proposto. Em seguida, na etapa **Projeto** é definida a performance que deseja ser alcançada e os métodos ideias para verificação. O objetivo da terceira etapa de **Desenvolvimento** é de gerar e/ou desenvolver todas as fontes de aprendizados necessárias para a unidade instrucional. Na próxima etapa, **Implementação** é preparado o ambiente de ensino e os estudantes para o processo de ensino da unidade instrucional. Por fim, a etapa **Avaliação** verifica a qualidade dos resultados alcançados e o processo realizado.

A3.1 – Análise do contexto e Projeto.
A3.2 – Desenvolver a rubrica.
A3.3 – Definir o *feedback* instrucional.

Etapa 4 – Desenvolvimento da ferramenta

Nesta etapa é desenvolvida a análise de design de interface e acrescentada à ferramenta web CodeMaster, com base no processo iterativo incremental de engenharia de software (Larman, Basili, 2003). Esta etapa é dividida nas seguintes atividades:

A4.1 – Análise de requisitos.
A4.2 – Design de interface.
A4.3 – Modelagem da arquitetura do sistema.
A4.4 – Modelagem detalhada e implementação.
A4.5 – Testes do sistema.

Etapa 5 – Avaliação da Rubrica

Nesta etapa a rubrica desenvolvida é avaliada em termos de confiabilidade e validade com base em um conjunto de apps desenvolvidos com App Inventor disponibilizados sob a licença de *creative commons* na galeria do App Inventor. Esta etapa é subdividida nas seguintes atividades:

[2] O modelo ADDIE é um acrônimo para as palavras: *Analyze*, *Design*, *Develop*, *Implement* e *Evaluate*, que em português significa respectivamente Análise, Projeto, Desenvolvimento, Implementação e Avaliação.

A5.1 – Avaliação da corretude da ferramenta.
A5.1.1 – Definição da avaliação da corretude e seleção dos apps.
A5.1.2 – Execução da avaliação da corretude.
A5.1.3 – Análise dos resultados.
A5.2 – Avaliação da confiabilidade e validade da rubrica.
A5.2.1 – Coletar os apps da Gallery do App Inventor
A5.2.2 – Definição da avaliação da confiabilidade e validade da rubrica.
A5.2.3 – Execução da avaliação da confiabilidade e validade da rubrica.
A5.2.4 – Análise dos resultados.

1.4 Estrutura do Documento

No capítulo 2 é apresentada a fundamentação teórica dos conceitos necessários que sustentam as propostas do trabalho. No capítulo 3 apresenta o mapeamento sistemático sobre os meios que existem para avaliação de design visual de interface de aplicativos móvel de *smartphones touchscreen* Android ou aplicativos desenvolvidos no App Inventor. O capítulo 4 apresenta o desenvolvimento do modelo de avaliação de design de interface de apps e a rubrica resultante, chamada de “CodeMaster UI Design - App Inventor”. O capítulo 5 apresenta as alterações realizadas no sistema CodeMaster para incluir a implementação da rubrica “CodeMaster UI Design - App Inventor” automatizada. O capítulo 5 também apresenta os testes de corretude feitos. O capítulo 6 apresenta a avaliação da confiabilidade e de validade. Para finalizar, o capítulo 7 apresenta a conclusão.

2. Fundamentação Teórica

2.1 Ensino e Aprendizagem

O **ensino** é a transmissão de competências sobre alguma coisa ou sobre como fazer algo (MICHAELIS, 2017). Essas competências podem incluir o Conhecimento, Habilidades e Atitudes[3] (KRAIGER, FORD, SALAS, 1993). O Conhecimento se refere a tomar consciência sobre um conjunto de variáveis e a relação entre elas. As habilidades focam no desenvolvimento de habilidades técnicas ou motoras (KRAIGER, FORD, SALAS, 1993). Também pode-se desenvolver atitudes mudando o estado interno (mental) de uma pessoa que influencia nas escolhas e consequentemente nas ações dela (GAGNÉ, BRIGGS, WAGER, 1992).

Com o ensino tem-se a expectativa de conseguir a aprendizagem de competências. A **aprendizagem** é o processo que provoca mudança no comportamento e no entendimento de um indivíduo, decorrentes de uma experiência (GAGNÉ, BRIGGS, WAGER, 1992). Essa mudança ocorre na aquisição de conhecimento, ao passar a compreender algo ou ao adquirir alguma habilidade ou mudança de atitude (MICHAELIS, 2017).

O ensino e a aprendizagem estão fortemente conectados entre si no processo de educação. A educação pode ser realizada de maneira formal e informal. A educação informal não é planejada, variando de acordo com o cotidiano, os valores e as culturas em que o indivíduo está inserido. Enquanto a formal é desenvolvida nas escolas e tem os conteúdos previamente demarcados (GOHN, 2006).

A educação formal tipicamente é estruturada por Unidades Instrucionais (UIs) com a intenção de auxiliar os instrutores a aplicar o conteúdo e os métodos de ensino (MARCONDES et al., 2009). Uma UI apresenta os objetivos de aprendizagem, o conteúdo a ser abordado, uma sequência lógica do conteúdo e como os alunos serão avaliados, podendo ser organizado como uma série de lições, exercícios ou atividades (HURWITZ, DAY, 2007).

A criação de UI é baseada em diferentes métodos instrucionais (Figura 1). A Tabela 1 apresenta os métodos instrucionais e suas descrições (SASKATCHEWAN, 1991).

[3] conhecimento, habilidades e atitudes, do inglês, *knowledge, skills e affectively - KSA*.

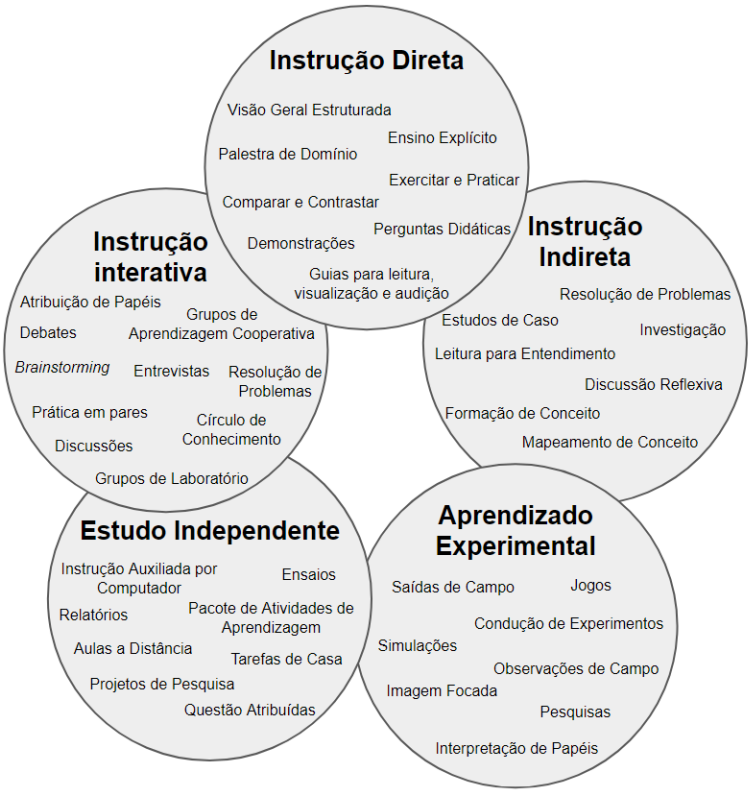


Figura 1: Métodos Instrucionais, adaptado a partir de (SASKATCHEWAN, 1991)

Tabela 1: Métodos Instrucionais (SASKATCHEWAN, 1991)

Método	**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 por meio 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 instrutor 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 instrutores 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 instrutor 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 instrutor 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.

Uma forma de aprendizagem experimental de forma ativa pode ser a **Aprendizagem Baseada em Problemas** (*Problem-Based Learning* - PBL). O PBL é uma abordagem instrucional centrada no aluno, objetivando incentivar os alunos a realizar pesquisas, integrar teoria com prática e aplicar conhecimentos e habilidades para desenvolver uma solução viável para um problema definido (SAVERY, 2006). Os problemas podem ser *well-structured* ou *ill-structured* (Jonassen, 1997). O problema *well-structured* pode ser traduzido do inglês como problemas bem estruturados ou problemas bem definidos. Esse tipo de problema tem uma solução conhecida e todas as informações necessárias para resolver o problema são providas, pois envolve conceitos e regras que são apresentadas no domínio de conhecimento ensinado. Isso torna as atividades com o desenvolvimento da solução previsível (Jonassen, 1997). Entretanto, o problema *ill-structured* possui elementos desconhecidos com variado grau de confiança, consequentemente tem diferentes meios de solucioná-lo ou até mesmo não ter solução. Mesmo havendo uma solução, geralmente não existe uma única solução correta (Jonassen, 1997).

2.1.1 Avaliação da aprendizagem do aluno

Uma parte essencial do ensino é a avaliação do aluno, pois o processo de avaliação consiste em verificar se os alunos por meio da UI conseguiram atingir completamente ou parcialmente o(s) objetivo(s) de aprendizagem da UI. Para isso, é preciso que a avaliação esteja alinhada com o conteúdo de ensino e as atividades realizadas durante a unidade instrucional (MACQUARIE, 2018)(NICOL, MCFARLANE-DICK, 2006).

As avaliações podem ser feitas de várias maneiras. Em contextos maiores (regionais, nacionais ou até internacionais), as avaliações são comumentes aplicadas por meio de testes padronizados[4], com perguntas múltiplas-escolhas, de completar sentenças, definir sentenças como verdadeiras ou falsas (HERRERA et al., 2007). Essa forma de avaliação é reconhecida como útil meio para comparar estudantes, unidades instrucionais e escolas (HERRERA et al., 2007), pois é simples apresentar o resultado da avaliação já que se há uma resposta correta, precisando apenas comparar as respostas do estudante com a solução correta (FORSYTHE; WIRTH, 1965). Porém, esse tipo de prova apresenta escopo muito limitado (MORRIS, 2011), restringe a criatividade e raramente tem contexto ou condições completamente realísticos (MACQUARIE, 2018). Outro problema é que a adoção deste tipo de teste pode causar um direcionamento do "ensinar para o teste"[5] (P. BLACK et al., 2011), focando exclusivamente em apenas na memorização do conteúdo para ter resultados bons nos testes, desviando o interesse de entender o conteúdo para agregar competências, evoluir e crescer (MORRIS, 2011).

Como resultado uma avaliação apresenta um *feedback* instrucional. Um *feedback* instrucional é a apresentação de observações sobre o desempenho do aluno com orientações para estimular o aluno a refletir sobre suas respostas e, se

[4] Testes padronizados, em inglês, *standardised tests*.
[5] "Ensinar para o teste", em inglês, 'teaching to the test'.

necessário, direcionar a melhorar o seu desempenho (GAGNÉ, BRIGGS, WAGER, 1992). Normalmente é apresentado como uma mensagem e/ou uma nota.

Existem diferentes abordagens para o *feedback* instrucional (Tabela 02) que podem ser classificadas de duas maneiras: por verificação ou por elaboração. A abordagem por verificação é definida como um julgamento se a resposta do aluno está correta ou não. Enquanto a abordagem por elaboração provê comentários e sugestões que buscam auxiliar que o próprio aluno identifique os pontos que precisam ser corrigidos (KRETLOW, BARTHOLOMEW, 2010)(BLACK, WILIAN, 1998).

Tabela 2: Tipos de *feedback* instrucional classificados por abordagem (PHYE, BENDER, 1989).

Tipo de *feedback* instrucional	**Descrição**
Abordagens por verificação	
Resposta correta	Apresenta ao aluno a resposta correta para um problema específico, sem apresentar qualquer informação adicional
Tente novamente	Informa ao aluno se a resposta está correta ou incorreta, e permite a ele uma ou mais tentativas para responder ao problema.
Marcação de erros	Destaca os erros presentes na solução apresentada pelo aluno, sem apresentar a resposta correta.
Abordagens por elaboração	
Isolação de atributo	Apresenta informações abordando individualmente os atributos centrais do conteúdo que está sendo estudado.
Dicas e sugestões	Orienta o aluno, indicando o que o ele deve fazer em seguida ou apresentando exemplos. Este tipo de *feedback* evita fazer uma apresentação explícita da solução correta
Análise de erros	Realiza a análise de erros e diagnósticos. Apresenta informações específicas sobre os erros cometidos pelos alunos e esclarece conceitos mal compreendidos (e.g. o que está errado e por que)
Tutorial	Realiza a marcação de erros, apresenta o informações relacionadas ao conteúdo, e provê dicas ou exemplos de como proceder. A resposta correta pode ou não ser apresentada.

O *feedback* instrucional tipicamente também inclui a alocação de uma nota ou conceito. A atribuição de nota pode se referir a um valor quantitativo ou qualitativo (Tabela 3). No Brasil não existe a definição de um esquema de notas para ser aplicado de forma padronizada em nível nacional (Alves, 2018). Segundo o artigo 7 da resolução CME Nº 02/2011 do Conselho Municipal de Educação de Florianópolis a nota pode ser expressa:

I - “através de parecer descritivo que revele o diagnóstico do processo de aprendizagem das respectivas competências e habilidades desenvolvidas pelos estudantes” (CME Florianópolis, 2011).

II - “através de numerais Indo-Arábicos variáveis de 1(um) a 10(dez)” (CME Florianópolis, 2011).

Tabela 3: Exemplos de tipos de notas expressadas em avaliações somativas.

Tipo Nota	**Melhor Nota**	**Notas intermediárias**	**Pior Nota**

Adjetivos	Excelente	Bom, Satisfatório	Ruim
Letras	A	B, C, D, E	F
Porcentagens	100%	99% até 1%	0%
Pontuação de um total	*n*, por exemplo 10	(*n-1*) até 1	0

É importante escolher o(s) tipo(s) de *feedback* instrucional(is), a fim de escolher aqueles mais aderentes ao contexto da UI e ao momento que será empregado (THURLINGS et al., 2013). É também importante que os *feedbacks* instrucionais sejam apresentados para o aluno em momentos convenientes para sua aprendizagem (NICOL, MCFARLANE-DICK, 2006). Caso contrário, o *feedback* instrucional pode se tornar incômodo, impactando negativamente na sua aprendizagem e motivação (SHUTE, 2007). As avaliações podem ser aplicadas em dois momentos durante uma unidade instrucional (BENNANI et al., 2012). Durante a UI para verificar o andamento e evolução da competência dos alunos, chamada de **avaliação formativa** (HERRERA et al., 2007). O *feedback* instrucional entregue durante a realização de alguma atividade precisa prover informações que indiquem ao aluno como melhorar sua resposta. Mas preferencialmente apenas de forma descritiva, sem atribuir notas, pois isto estimula o aluno a dominar a competência que está sendo ensinado (NICOL, MCFARLANE-DICK, 2006). Outro momento para aplicar uma avaliação é no final da UI, chamada de **avaliação somativa**, que tem o intuito de medir as competências que os alunos atingiram (LINN, MILLER, 2005). Normalmente é atribuído notas, comparando os resultados apresentados pelo aluno com os resultados esperados (BENNANI et al., 2012).

No contexto de aprendizagem baseado em problemas *ill-structured*, as atividades não tem um gabarito pré-definido (ADAMS, WEBSTER, 2012). Tipicamente o desempenho do aluno é avaliado com base no artefato criado, verificando a capacidade dos alunos de aplicar competências adquiridas em contextos *ill-structured* e trabalhar de forma colaborativa para resolver problemas complexos (Wiggins, 1993). Existem diversos tipos de avaliações no contexto da aprendizagem baseada em problemas, como por exemplo avaliações de desempenho, avaliação de portfólio, entrevistas baseadas nos artefatos criados e autoavaliações (Brennan e Resnick, 2012).

Uma maneira de realizar uma avaliação baseada em desempenho é por **rubricas**. As rubricas auxiliam a organizar os critérios e o quanto valem deixando bem específico ao estudante o que é preciso focar e ao professor a desenvolver uma avaliação mais objetiva e bem justificada (MCCAULEY, 2003). As rubricas usam medidas descritivas para separar níveis de desempenho na obtenção de resultados de aprendizagem, delineando os vários critérios associados a atividades de aprendizagem e indicadores descrevendo cada nível para avaliar o desempenho dos alunos (MCCAULEY, 2003). Os níveis de desempenhos são tipicamente escalas ordinais, como níveis “ruim”, “bom” e “muito bom” ou por pontuação, sendo possível alcançar apenas um nível de desempenho por critério e havendo um ordenamento de quão bem no critério. Por exemplo, a Figura 2 apresenta uma rubrica organizada em tabela, em que na primeira coluna estão listados os critérios a serem avaliados e

nas demais colunas é apresentado o que é preciso para conseguir alcançar o nível de desempenho.

NÍVEIS DE DESEMPENHOS

Critério	0 pontos	1 ponto	2 pontos	3 pontos
Telas	Apenas uma tela com componentes visuais e que seu estado não se altera com a execução do app (tela informativa).	Apenas uma tela com componentes visuais que se alteram com a execução do app.	Pelo menos duas telas e uma delas altera seu estado com a execução do app.	Duas ou mais telas e pelo menos 2 delas alteram seus estados com a execução do app.
Interface de Usuário	Utiliza apenas 1 elemento visual sem alinhamento.	Utiliza 2 ou mais elementos visuais sem alinhamento.	Utiliza 2 ou mais elementos visuais com 1 tipo de alinhamento.	Utiliza 5 ou mais elementos visuais com 2 ou mais elementos de alinhamento.
Nomeação: Variáveis e procedimentos	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.
Eventos	Nenhum manipulador de evento é usado (ex. On click).	1 tipo de manipuladores de eventos é usado.	2 tipos de manipuladores de eventos são usados.	Mais de 2 tipos de manipuladores de eventos são usados.

Figura 2: Exemplo de uma Rubrica (GRESSE VON WANGENHEIM et al., 2018)

Ao avaliar um projeto com uma rubrica, a partir dos níveis de desempenho alcançados com cada critério é possível inferir um *feedback* instrucional descritivo e também uma nota. Por meio das rubricas, para gerar uma nota é preciso atribuir o peso de cada critério e a pontuação possível conseguir em cada nível de desempenho. Tipicamente, é atribuído menos pontos aos níveis de desempenhos mais baixos e mais pontos para os níveis de desempenhos melhores. A pontuação mínima e máxima pode resultar diretamente na escala do tipo de nota utilizado, caso não, a pontuação alcançada numa atividade precisa ser convertida para o intervalo da nota.

2.2 Ensino de Computação na Educação Básica

Segundo o Ministério da Educação (2013), a Educação Básica é a formação escolar considerada como a:

> “Condição primeira para o exercício pleno da cidadania e o acesso aos direitos sociais, econômicos, civis e políticos. A educação deve proporcionar o desenvolvimento humano na sua plenitude, em condições de liberdade e dignidade, respeitando e valorizando as diferenças” (MINISTÉRIO DA EDUCAÇÃO, 2013).

A Educação Básica é formada pelas etapas de Ensino Infantil, Fundamental e Médio. O Ensino Fundamental é organizado em duas etapas, a primeira etapa engloba os 5 primeiros anos, chamado Ensino Fundamental I, e a segunda etapa engloba os quatro anos finais, do 6º ao 9º ano, chamado Ensino Fundamental II (MINISTÉRIO DA EDUCAÇÃO, 2013).

As diretrizes e bases da educação nacional são reguladas pela Lei 9.394/96 (Lei de Diretrizes e Bases da Educação – LDB) que divide a grade de ensino em um núcleo comum (Língua Portuguesa, História, Geografia, Matemática, Ciências, Educação Física e Arte) e um núcleo diversificado, que inclui uma segunda língua moderna e demais disciplinas. A partir do 5º ano, a escola é obrigada a oferecer aulas de segunda língua (LDB, Artigo 26, § 5º), sendo optada pela maioria a língua inglesa. Enquanto o ensino religioso é facultativo (LDB, Artigo 33). Porém, levando em consideração as mudanças constantes no mundo atual em relação às condições sociais, culturais e políticas, que refletem em alterações dos objetivos de instituições de ensino e de seus alunos. Assim currículos escolares precisam ser constantemente atualizados (BRUNER, 1977), não sendo diferente ao surgimento da cultura digital e o disseminado uso de tecnologias de informação e comunicação na sociedade (MINISTÉRIO DA EDUCAÇÃO, 2017)(CSTA, 2017). Para que a Educação Básica continue a cumprir o seu objetivo é preciso incluir em seu Currículo o ensino da Ciência da Computação:

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

Inclusive, a Base Nacional Comum Curricular (MINISTÉRIO DA EDUCAÇÃO, 2018) reconhece que os estudantes estão envolvidos com a cultura digital não apenas como consumidores, mas apresentam propostas e preocupações para educar participantes ativistas de ferramentas de comunicação e relacionamentos sociais. Além de incluir o pensamento computacional na área de matemática. Sendo o primeiro passo para a inserção da computação por meio do pensamento computacional na Educação Básica. Pois até então, no Brasil, esse conhecimento era restrito aos cursos de Educação Superior.

Para se promover o ensino de computação na Educação Básica a Sociedade Brasileira de Computação (2017), iniciou a definição de um guia de currículo. De acordo com esse guia o ensino de computação na Educação Básica deve abordar os principais eixos incluindo Pensamento Computacional, Mundo Digital e Cultura Digital. Junto com as competências e habilidades de cada um e como eles podem ser trabalhados ao longo da Educação Básica, da Educação Infantil até o Ensino Médio.

Internacionalmente existem diversas iniciativas de definição de currículo voltado ao ensino de computação na Educação Básica. Entre eles, o mais reconhecido é o *K-12 Computer Science Standards* (CSTA, 2017), que apresenta as competências de computação a serem aprendidas no contexto da Educação Básica incluem em Conceitos Fundamentais e Práticas Fundamentais (Tabela 4). Os Conceitos Fundamentais representam áreas de conteúdo chave em Ciência da

Computação (Tabela 5), enquanto as Práticas Fundamentais representam ações que os alunos usam para se envolver com os conceitos de maneiras ricas e significativas (Tabela 6).

Tabela 4: Competências básicas (CSTA, 2017)

Conceitos Fundamentais	Práticas Fundamentais
Sistema de Computação	Promover uma cultura de computação inclusiva
Redes e Internet	Colaborar acerca da computação
Dados e Análise	Reconhecer e definir problemas computacionais
Impactos da Computação	Desenvolver e usar abstrações
Algoritmos e Programação	Criar artefatos computacionais
	Testar e refinar artefatos computacionais
	Comunicar sobre a computação

Cada Conceito Fundamental é delineado por múltiplos subconjuntos que representam ideias específicas dentro de cada Conceito Fundamental. A aprendizagem para cada subconceito fornece um tópico que liga o aprendizado do aluno do jardim de infância ao último ano do ensino médio de forma progressiva, ou seja, evoluindo a dificuldade e a profundidade no assunto de acordo com o ano de ensino.

Tabela 5: Conceitos Fundamentais (CSTA, 2017)

Conceito	Descrição
Sistemas Computacionais	Os Sistemas Computacionais são formados por componentes físicos (hardware) e as instruções (software) que comunicam e processam informações em formato digital. A compreensão do hardware e do software é útil quando se soluciona um sistema de computação que não funciona como pretendido.
Redes e a Internet	Os dispositivos de computação tipicamente não operam em isolamento, havendo as redes para conectá-los permitindo compartilhar informações e recursos.
Dados e Análise	Existem sistemas de computação para processar dados. A quantidade de dados digitais gerados no mundo está se expandindo rapidamente, de modo que a necessidade de processar dados efetivamente é cada vez mais importante. Os dados são coletados e armazenados para que possam ser analisados para entender melhor o mundo e fazer previsões mais precisas.
Algoritmos e programação	Para criar programas significativos e eficientes envolve a escolha de quais informações usar, como processá-lo e armazená-lo, quebrando grandes problemas em menores, recombinando soluções existentes e analisando diferentes soluções. Assim sendo possível desenvolver um algoritmo, que é uma sequência de etapas projetadas para realizar uma tarefa específica.
Impactos da computação	A computação afeta muitos aspectos do mundo de forma positiva e negativa a nível local, nacional e global. Indivíduos e comunidades influenciam a computação através de seus comportamentos e interações culturais e sociais e, por sua vez, a computação influencia novas práticas culturais. Uma pessoa informada e responsável deve entender as implicações sociais do mundo digital, incluindo equidade e acesso à computação.

Tabela 6: Práticas Fundamentais (CSTA 2017)

Práticas Fundamentais	Descrição
Promover uma cultura de computação inclusiva	Considerar perspectivas dos outros. Atender as necessidades de diversos usuários.
Colaborar em torno da Computação	Cultivar relações de trabalho. Criar normas de equipe, expectativas e cargas de trabalho equitativas. Solicitar e incorporar comentários e fornecer *feedback*. Avaliar e selecionar as ferramentas tecnológicas
Reconhecer e Definir Problemas Computacionais	Decompor problemas complexos em subproblemas. Avaliar a viabilidade de resolver um problema computacionalmente.

Desenvolver e Usar Abstrações	Extrair características comuns de processos ou fenômenos. Avaliar e incorporar funcionalidades tecnológicas. Modelar e simular fenômenos.
Criar Artefatos Computacionais	Planejar, criar e modificar um artefato computacional
Testar Refinar Artefatos Computacionais	Testar artefatos. Identificar e corrigir erros. Avaliar e refinar um artefato.
Comunicar sobre a computação	Selecionar, organizar e interpretar conjuntos de dados. Descrever justificar e documentar soluções computacionais. Articular ideias.

Além dos Conceitos Fundamentais, CSTA (2017) apresenta conceitos que ilustram conexões entre os Conceitos Fundamentais, chamados de Conceitos Transversais[6]. Eles são integrados nos Conceitos Fundamentais, em vez de existir como uma dimensão independente da estrutura (Tabela 7).

Tabela 7: Conceitos Transversais (CSTA, 2017).

Conceitos Transversais	**Descrição**
Abstração	Uma abstração é o resultado de reduzir um processo ou conjunto de informações para um conjunto de características importantes para uso computacional. As abstrações estabelecem interações em um nível de complexidade reduzida, gerenciando os detalhes mais complexos abaixo do nível de interação. Uma abstração pode ser criada para generalizar uma variedade de situações escolhendo propriedades comuns menos detalhes específicos de implementação.
Relacionamentos do sistema	As partes de um sistema são interdependentes e organizadas para um propósito comum. Uma perspectiva de sistemas oferece a oportunidade de decompor problemas complexos em partes que são mais fáceis de entender, desenvolver, consertar e manter. Os princípios gerais de sistemas incluem *feedback*, controle, eficiência, modularidade, síntese, emergência e hierarquia.
Privacidade e segurança	A privacidade é a capacidade de isolar a informação e expressá-la seletivamente. Incluindo controles para a coleta, acesso, uso, armazenamento, compartilhamento e alteração de informações. A segurança suporta a privacidade mantendo proteções em torno dos sistemas de informação evitando roubo ou danos ao hardware, software e informações nos sistemas.
Comunicação e Coordenação	Os processos na computação são caracterizados pela troca confiável de informações entre agentes autônomos (comunicação) que cooperam para resultados comuns que nenhum agente poderia produzir sozinho (coordenação). A comunicação e a coordenação são processos distintos, mas não independentes.
Interação humano-computador	Os seres humanos interagem diretamente com computadores como laptops e smartphones, mas também outros dispositivos, como carros e eletrodomésticos, que possuem computadores incorporados. O desenvolvimento de interfaces de usuário efetivas e acessíveis envolve a integração do conhecimento técnico e das ciências sociais e engloba as perspectivas do designer e do usuário.

As competências de computação são segmentadas em objetivos de aprendizagem para ser abordada em cada nível de educação. Como esse *framework* divide o ensino de computação em 3 níveis podendo alinhar ao sistema de ensino no Brasil da seguinte maneira: ensino infantil e ensino fundamental I, o ensino fundamental II e ensino médio (Figura 3).

[6] Conceitos Transversais, em inglês, *crosscutting concepts.*

Figura 3: Níveis de ensino de computação (CSTA, 2017)

No Brasil já foram realizados vários cursos com unidades instrucionais baseadas nas diretrizes CSTA K-12 para se ensinar computação na Educação Básica (ALVES et al., 2017)(DANIEL et al., 2017). As atividades práticas destas unidades instrucionais envolvem na resolução de problemas *ill-structured*, como por exemplo, criação de jogos (ALVES et al., 2017) e de aplicativos (DANIEL et al., 2017).

2.3 Design de Interface do Usuário

No desenvolvimento de um produto é muito importante que sejam consideradas as características do usuário e como o usuário irá utilizá-lo. Isso é determinante para que o produto cause mínimos aspectos negativos na experiência do usuário, como frustrações e aborrecimentos, e muitos os aspectos positivos, como diversão, compromisso, facilidade (ROGERS; SHARP; PREECE, 2013)(SCHLATTER; LEVINSON, 2013). Para isso existe a área de Design de Interação, que se preocupa em:

> "Projetar produtos interativos para apoiar o modo como as pessoas se comunicam e interagem em seus cotidianos, seja em casa ou no trabalho. Em outras palavras, significa criar experiências de usuários que melhorem e ampliem a maneira como as pessoas trabalham, se comunicam e interagem" (ROGERS; SHARP; PREECE, 2013).

O Design de Interação subdivide-se em especialidades que enfatizam diferentes aspectos do que se está projetando (ROGERS; SHARP; PREECE, 2013). Uma dessas especialidades é o **Design de Interface do Usuário** (ROGERS; SHARP; PREECE, 2013). Design de interface de usuário é:

> "O design de interfaces de usuário para software ou máquinas, como a aparência de um aplicativo móvel, com foco na facilidade de uso e satisfação para o usuário. O design da interface do usuário geralmente se refere ao design de interfaces gráficas com o usuário - mas também pode se referir a outras, como interfaces de usuário natural e de voz" (Interaction Design Foundation, 2018).

Para o desenvolvimento de um aplicativo de sucesso é importante que a interface gráfica seja fácil, agradável de usar e exija pouco esforço para os usuários (ROGERS; SHARP; PREECE, 2013). Essa qualidade é chamada de usabilidade, que é o:

> "Grau com o qual um produto ou sistema pode ser usado por usuários específicos para atingir objetivos específicos com eficácia, eficiência e satisfação em um contexto de uso específico" (ISO/IEC 25010:2014).

Os diferentes estilos de interação do usuário causam grande impacto no design de interação para produtos eletrônicos (ROGERS; SHARP; PREECE, 2013). No contexto de aplicativos móveis existem vários fatores que precisam ser considerados ao tratar da qualidade da usabilidade. Como a tela pequena controlada por toque (em inglês, *touchscreen*), teclados virtuais, gestos, sensores e dados de localização. Além dos dispositivos possibilitarem o uso de apps em qualquer lugar e em qualquer momento (WASSERMAN, 2010), o que causa variação de iluminação e sons no local de uso. Esses fatores precisam ser considerados no design de interface de um aplicativo. Tipicamente o design de interface de aplicativos usa diversos componentes de interface conforme os utilizados no App Inventor, que são apresentados na Tabela 8.

Tabela 8: Componentes de design de interface de usuário disponíveis no App Inventor (2018)

Componente	**Descrição**
Botão	Componente com a capacidade de detectar cliques. A partir de cliques realiza alguma ação.
Caixa de seleção	Os componentes da caixa de seleção podem detectar toques do usuário e podem alterar seu estado booleano em resposta. Um componente de caixa de seleção gera um evento quando o usuário o toca.
EscolheData	Um botão que, quando clicado, inicia uma caixa de diálogo pop-up para permitir que o usuário selecione uma data.
Imagem	Componente para exibir imagens.
Legenda	Componente usado para mostrar texto.
EscolheLista	Um botão que, quando clicado, exibe uma lista de textos para o usuário escolher.
VisualizadorDeLista	Um componente visível que permite colocar uma lista de elementos na tela para exibir.
Notificador	O componente Notificador exibe caixas de diálogo de alerta, mensagens e alertas temporários
CaixaDeSenha	Os usuários inserem senhas em um componente de caixa de texto de senha, que oculta o texto que foi digitado nele.
Tela	Componente de nível superior contendo todos os outros componentes do programa.
Deslizador	Uma barra de progresso que adiciona um polegar arrastável. O usuário pode tocar no polegar e arrastar para a esquerda ou para a direita para definir a posição do polegar do controle deslizante.
ListaSuspensa	Esse componente exibe um pop-up com uma lista de elementos.
CaixaDeTexto	Componente em que os usuários inserem texto.
EscolheHora	Um botão que, quando clicado, inicia uma caixa de diálogo pop-up para permitir que o usuário selecione uma hora.

Dentro do universo de fatores que contribuem para boa qualidade do design visual, Schlatter e Levinson (2013) elencaram três que consideram afetar fortemente o design de um aplicativo. Esses três foram chamados de meta-princípios: consistência, hierarquia e personalidade (Tabela 9). Esses meta-princípios abrangem conceitos de coerência, consistência, uniformidade e hierarquia de elementos de cor, tipografia, imagens e controles.

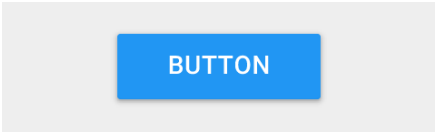
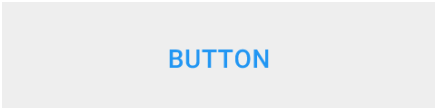
Tabela 9: Meta-princípios da linguagem visual (SCHLATTER; LEVINSON, 2013)

Critério	**Descrição**
Consistência	É o processo de definir e preservar expectativas do usuário, por meio da utilização de elementos com os quais ele está familiarizado. As expectativas provém tanto do que o usuário está visualizando quanto do que ele já viu no passado.
Hierarquia	Trata sobre a percepção e a interpretação da importância de cada objeto mostrado na tela. A percepção de hierarquia é influenciada pela posição, tamanho, cor, interface, tipo de controle (por exemplo, um botão versus um link) e tratamento dos elementos, bem como pela forma com que cada elemento se relaciona entre si. Dessa forma, observar a hierarquia ao construir um aplicativo significa definir onde colocar os elementos com base em sua importância relativa, pensando conscientemente em suas características e posições, para comunicar quais as prioridades do projeto.
Personalidade	Refere-se às impressões que são formadas pelo usuário, consciente ou inconscientemente, com base na aparência, no comportamento ou na satisfação de um aplicativo. Embora cada interação afete o modo como as pessoas interpretam e avaliam um app, o foco do estudo são os aspectos visuais da personalidade de um aplicativo.

Voltado ao design de interfaces de aplicativos Android, existem guias de estilo, como por exemplo, o *Material Design* (2018). O *Material Design* é um guia de estilo para design visual, que visa criar uma linguagem visual sintetizando princípios clássicos de um bom design com a inovação e a possibilidade de tecnologia e ciência (Google, 2017). O *Material Design* fornece recomendações para o design de diversos elementos de interface, por exemplo, em relação a cores, iconografia, tipografia, proporção, tamanhos, espaçamento, animação, acessibilidade, componentes, entre outros. Por exemplo, em relação ao componente Botão, o *Material Design* sugere o design de três tipos de botão (Tabela 9).

Tabela 10: Design de botões conforme o *Material Design*

Tipos Botão	**Descrição**	**Exemplo**
Botão de ação flutuante	Esse botão tem formato circular e precisa ter cor distinta do plano de fundo de onde o botão está. Recomenda-se usar esse botão quando o aplicativo exigir que as ações sejam persistentes e prontamente disponíveis.	

Botão realçado	Tem formato tipicamente retangular e precisa ter cor distinta do plano de fundo de onde o botão está. Podem ser utilizados *inline* e destacam-se mais que botões planos.	
Botão plano	Um botão sem fundo de destaque, precisando da cor da fonte ser diferente, para que o usuário reconheça como botão. Podem ser utilizados em caixas de diálogos, barras de ferramentas ou *inline*.	

As Diretrizes de Acessibilidade ao Conteúdo Web[7] (WCAG) 2.0 abrangem uma ampla gama de recomendações para tornar o conteúdo Web mais acessível. Essas diretrizes apresentam meios de atribuir acessibilidade ao conteúdo para usuários em geral e para ampla gama de pessoas com deficiências. Como pessoas com deficiências visuais, auditivas, físicas, de fala, cognitivas, de linguagem, de aprendizado e neurológicas.

WCAG 2.0 apresenta três níveis de conformidade para cada diretriz: A (menor), AA e AAA (maior). Cada nível atende às necessidades de diferentes grupos e situações diferentes. Sendo AAA o nível que permite acessibilidade à maior gama de pessoas com diferentes graus de deficiências. Por exemplo, WCAG 2.0 apresenta diretriz sobre o contraste mínimo necessário entre a cor do texto e a cor ao fundo do texto. Essa diretriz apresenta contraste ideal conforme o tamanho do texto e se é negrito ou não (Tabela 11). Como cálculo de contraste surgido por WCAG 2.0 pode resultar entre 1 e 21, quanto maior o valor resultante maior é o contraste entre cor do texto e a cor ao fundo do texto (Figura 4).

Tabela 11: Definição de texto pequeno e grande

Tipo texto	Definição
Texto pequeno	Texto não negrito com tamanho de fonte menor que 18pt. Ou negrito com tamanho de fonte menor que 14pt e é negrito.
Texto grande	Texto não negrito com tamanho de fonte maior ou igual a 18pt. Ou negrito com tamanho de fonte maior ou igual a 14pt é negrito.

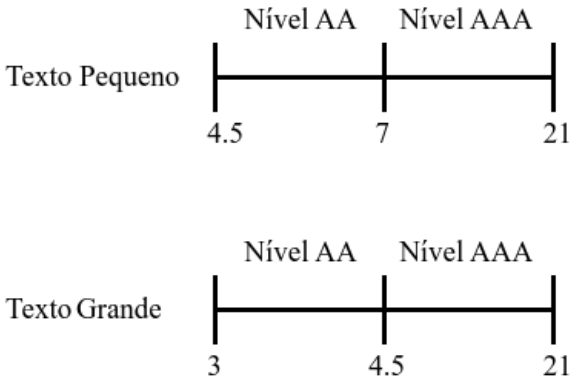


Figura 4: Diretriz de contraste cada nível WCAG para texto pequeno e grande

[7] Em inglês, *Web Content Accessibility Guidelines* (WCAG) 2.0.

2.4 App Inventor

Com a intenção de engajar crianças e jovens na área da computação no ensino fundamental e para facilitar a aprendizagem de computação, linguagens de programação visual baseadas em blocos são utilizadas. Como Scratch (2018), Snap! (2018) ou App Inventor (RESNICK et al., 2009). Linguagens de programação visual baseadas em blocos permitem criar sistemas de software ou aplicativos arrastando e encaixando blocos, em que todos os componentes estão definidos, normalmente, em forma de figuras ou blocos. O usuário deverá colocar esses blocos numa ordem que faça sentido ao programa e então a aplicação começa a ser desenvolvida (Figura 5). Desta maneira os alunos não precisam escrever linhas de código textuais, eliminando a necessidade de se entender e memorizar as sintaxes de linguagens textuais (CSTA, 2017) (PRICE, BARNES, 2017).

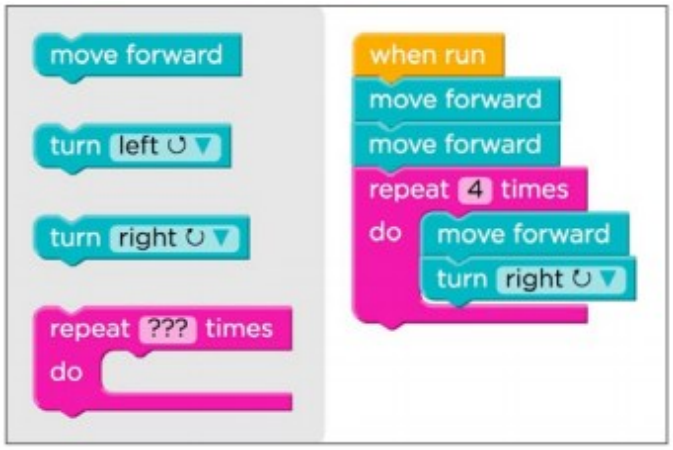


Figura 5: Exemplo de linguagem de programação visual baseada em blocos (CSTA, 2017).

O App Inventor (2018) é uma ferramenta de código-aberto, desenvolvida na Google sendo mantido pelo Instituto de Tecnologia de Massachusetts (MIT), que auxilia no desenvolvimento de aplicativos (ou app) para dispositivos Android com suporte para design de interface e para a programação funcional do aplicativo. Este ambiente de programação permite proporcionar às pessoas com pouco ou nenhum conhecimento de programação a experiência de desenvolver aplicativos para dispositivos Android de forma intuitiva. Em 2018, o App Inventor tem mais de 400 mil usuários ativos por mês de 195 países (APP INVENTOR, 2018).

O desenvolvimento da parte funcional do aplicativo é feito com programação visual baseada em blocos. Para isso, o App Inventor utiliza a biblioteca de código aberto Blockly (2018) para construir o vocabulário de blocos de programação que podem ser utilizados (PASTERNAK; FENICHEL; MARSHALL, 2017).

O App Inventor permite que o projeto de um aplicativo criado seja exportado em um arquivo compactado no formato aia. Na versão nb162a do App Inventor, este arquivo inclui todas as informações do app, como imagens e sons utilizados e os códigos produzidos pelos blocos da lógica e componentes visuais presentes no app. Os códigos referentes a parte visual do app ficam registrados em arquivos no

formato .scm que por sua vez encapsulam uma estrutura json com todos os componentes utilizados. Os códigos produzidos pela parte lógica do app ficam registrados em arquivos no formato .bky que por sua vez encapsulam estruturas xml com toda a lógica implementada.

O desenvolvimento de um aplicativo no App Inventor é dividido em duas grandes áreas: Designer e Blocos. A área Blocos (Figura 6) é utilizada para a programação da lógica para definir as ações da interface de usuário, como condicionais, loops, verificação de estado de sensores, etc. Para construir uma lógica para o aplicativo gerar alguma ação por alguma ação feita pelo usuário, é preciso escolher um bloco correspondente à ação que pretende executar. Inclusive dentre os blocos disponíveis para utilização, o App Inventor apresenta blocos que podem ser usados para cada componente adicionados nas telas (Apêndice A).

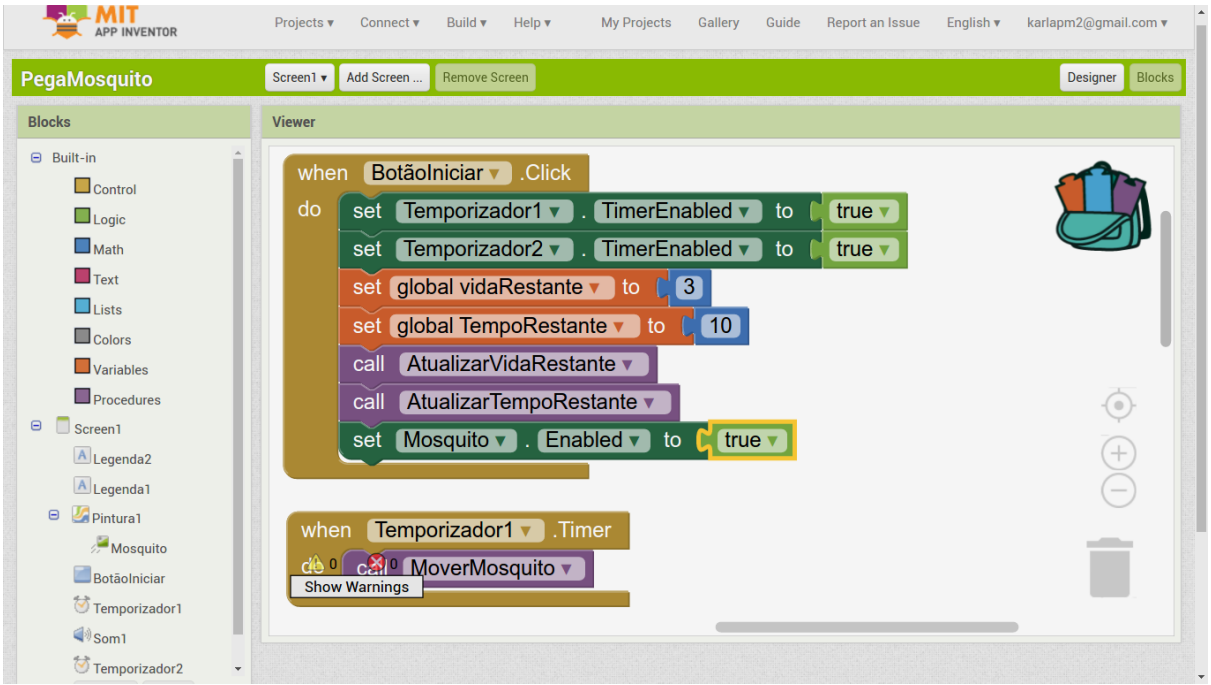


Figura 6: Tela Blocos do App Inventor (APP INVENTOR, 2018)

A área Designer é utilizada para a programação dos componentes referentes ao design de interface e de conectividade de recursos do celular (Figura 7), como botões, inserção de imagens, acelerômetro, sensor de localização, temporizadores, sons. Na Tabela 12 são apresentados os componentes da área Designer do App Inventor.

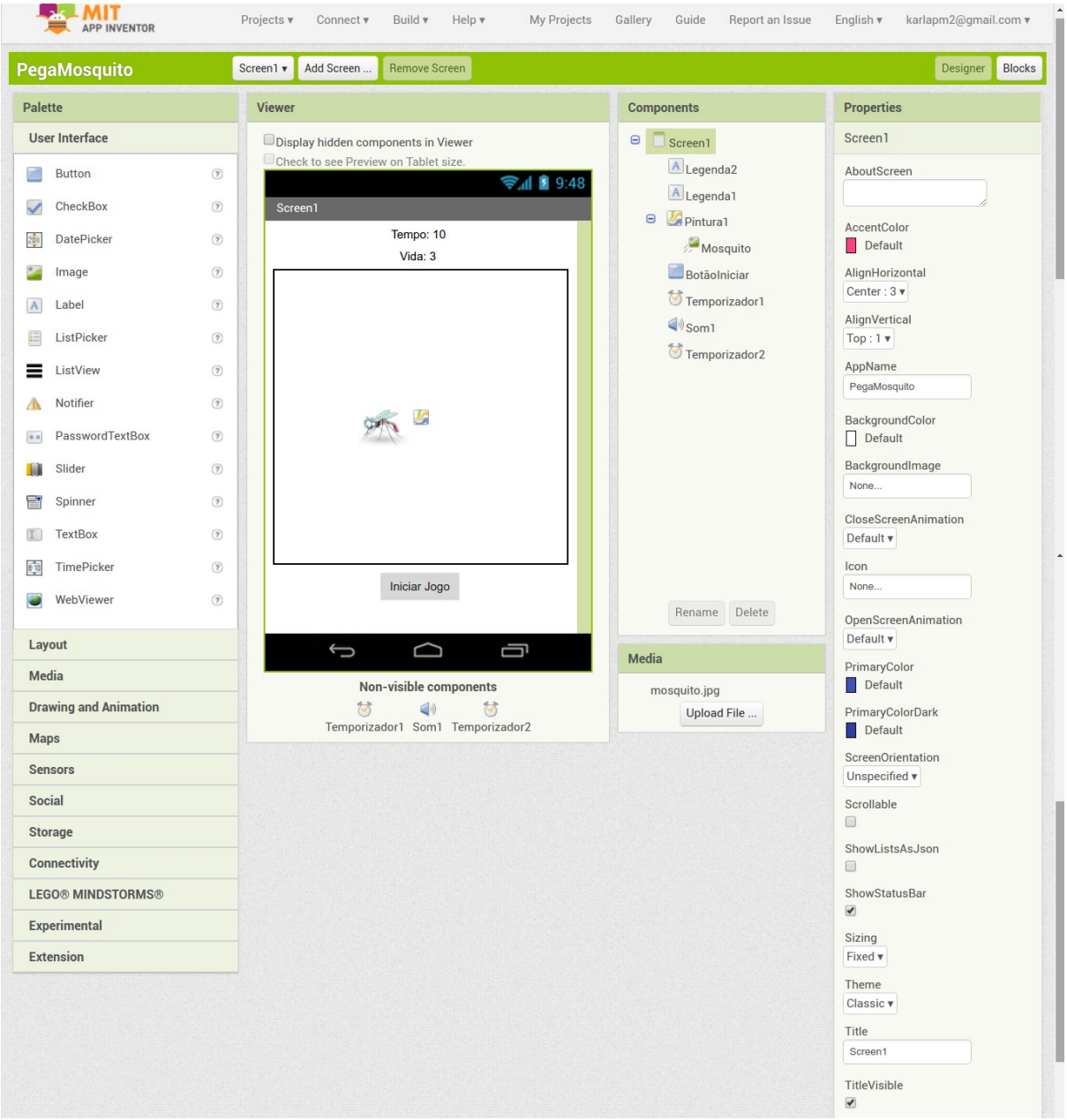


Figura 7: Tela “Designer” do App Inventor (APP INVENTOR, 2018)

Tabela 12: Componentes do Designer do App Inventor

Grupo	Componentes	Descrição
Interface de Usuário	Botão; Caixa de Seleção; Escolhe Data; Imagem; Legenda; Escolhe Lista; Visualizador de Listas; Notificador; Caixa de Senha; Deslizador; Lista Suspensa; Caixa de Texto; Escolhe Hora; Navegador Web.	Criação da parte visual do app. Todos os elementos visíveis da aplicação.
Organização	Organização Horizontal; Horizontal Scroll Arrangement; Organização em Tabela; Organização Vertical; Vertical Scroll Arrangement.	Auxilia na organização dos elementos visíveis de interface com usuário
Mídia	Câmera de Vídeo; Câmera; Escolhe Imagem; Tocador; Som; Gravador; Reconhecedor de Voz; Texto para Falar; Reprodutor de Vídeo; Tradutor Yandex	Apresenta componentes de mídia que podem ser usados na aplicação.
Desenho e Animação	Bola; Pintura; Sprite Imagem.	Componentes que permitem ao usuário desenhar e visualizar animações.
Mapas	Círculos; Feature Collection; LineString; Mapa; Marker; Polígonos;	Um dos Componentes é o mapa, que

	Retângulo.	inclui um mapa à interface. Os demais componentes personalizam esse mapa.
Sensores	Sensor acelerômetro; Código de barras; temporizador; Gyroscope; Localização; Near Field; Orientação; Pedômetro; Proximidade.	Componentes que obtém informações dos sensores presentes no dispositivo em que o app será instalado.
Social	Escolher contato; Escolher e-mail; Ligação; Escolher Número telefone; Compartilhamento; SMS; Twitter.	Componentes que permitem a comunicação da aplicação com outros aplicativos sociais
Armazenamento	Arquivo; Fusiontables; TinyDB; TinyWebDB.	Componentes que permitem a criação de bancos de dados para armazenar dados
Conectividade	Iniciador de Atividade; Cliente Bluetooth; Servidor Bluetooth; Web.	Componentes que permitem a conectividade da aplicação com outros dispositivos.

Os componentes do design de interface do aplicativo podem ser customizados, com base nas opções de propriedade de cada componente. Por exemplo, ao adicionar um botão no aplicativo, são apresentadas opções como:

- Escolher a cor de fundo,
- Formatar o texto apresentado (como itálico, negrito, tamanho da fonte, tipo da fonte, cor, alinhamento),
- Tamanho do componente (altura e largura),
- Habilitação se é inicialmente visível ou não, dentre outras.

Os componentes de design de interface podem ser customizados no App Inventor de três maneiras. Por exemplo, a cor de fundo e a cor de texto podem ser definidas:

- A primeira opção é selecionando diretamente a cor na Palheta (Figura 8.a);
- É possível customizar uma cor ao alterar a tonalidade e a transparência da cor escolhida inicialmente na palheta (Figura 8.b); e
- Também é possível customizar na área Blocos, selecionando um dos blocos de ajustes de cores do componente de interesse e escolher a cor (Figura 8.c).

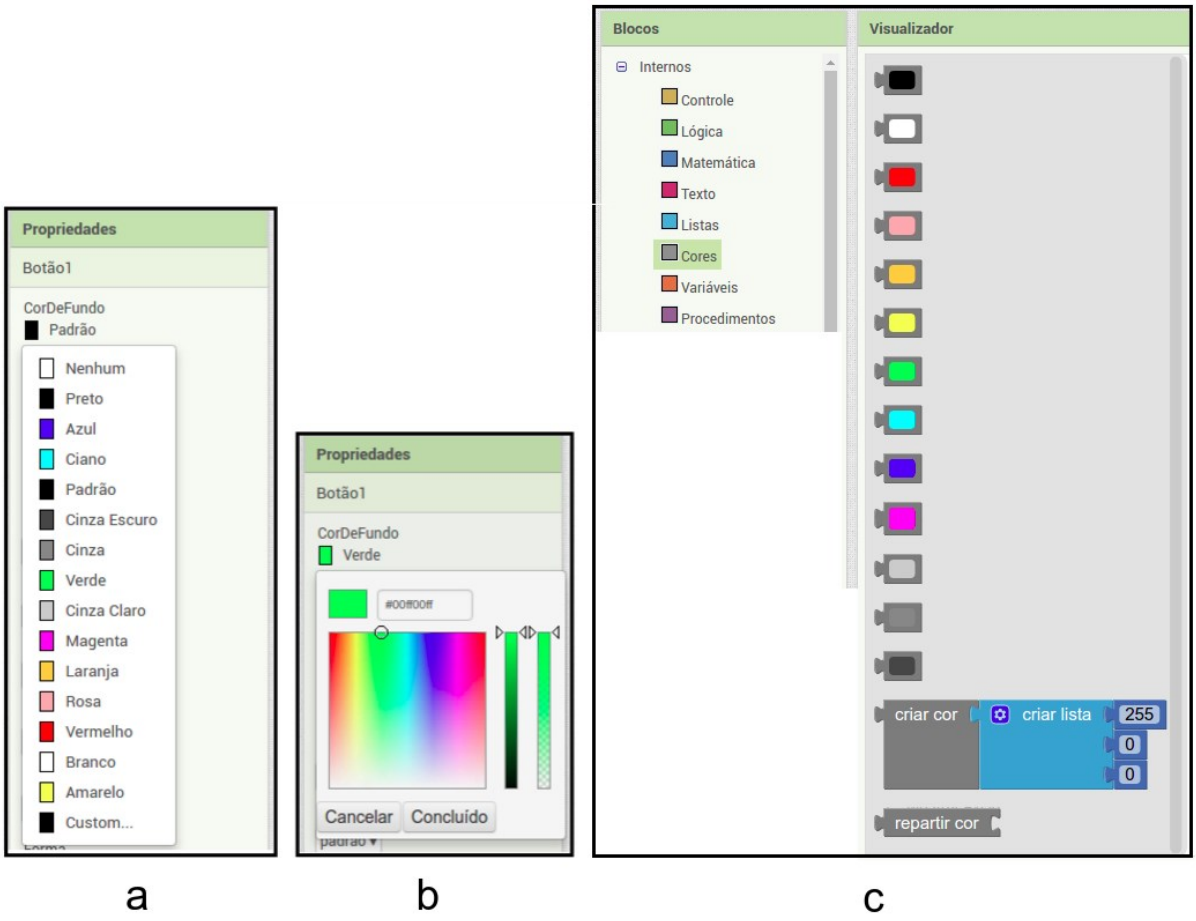


Figura 8: Opções para customizar cores no AppInventor
a. Palheta de cores para customização
b. Opção gradiente para alterar a tonalidade e a transparência da cor escolhida
c. Opção de programar a cor na área Blocos

Outra maneira de customizar alguns componentes é pela família da fonte (Figura 9), que define o design das letras. Também é possível definir a altura e largura do componente, como um botão, há quatro maneiras de definir (Figura 10a):

- Automático: o tamanho é escolhido pelo sistema. No caso de botões, por exemplo, a largura e altura são definidas pelo texto que apresenta.
- Preencher Principal: o componente procura preencher todo o espaço disponível em que está contido, podendo ser tela ou componente de organização.
- Pontos: define o tamanho especificado em pixels. A altura e largura em pontos também podem ser definidos na área Blocos com os blocos “.Altura” e “.Largura” (Figura 10b).
- Percentagem: define o tamanho especificado em percentagem do local em que o componente está contido, podendo ser tela ou componente de organização. A altura e largura em pontos também podem ser definidos na área Blocos com os blocos “.PercentualDeAltura” e “.PercentualDeLargura” (Figura 10b).

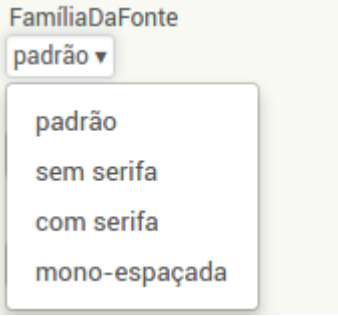


Figura 9: Tipos de famílias de fontes disponibilizadas no App Inventor

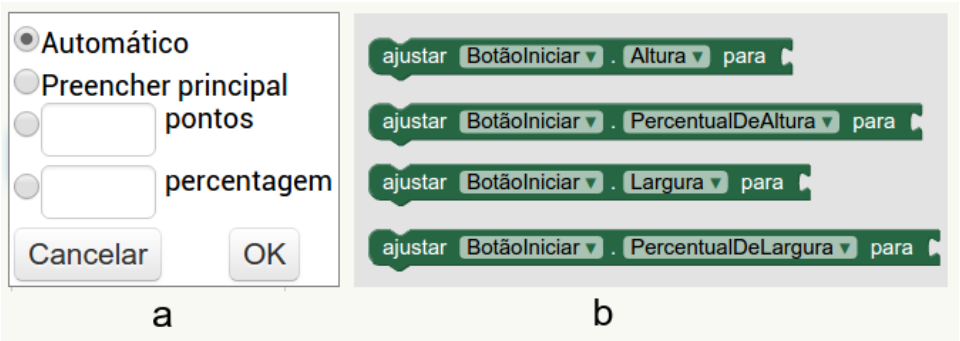


Figura 10: Opções para definição da largura ou altura de um componente
a. Opções para definição da largura ou altura de um componente via propriedade.
b. blocos de programação para definir largura ou altura de um botão.

Uma definição completa de formas de customização de cada componente de interface do App Inventor é apresentada no Apêndice B.

2.5 Avaliação automatizada de código

A aplicação de aprendizado baseado em problemas *ill-strucutured* no contexto de ensino de programação e desenvolvimento de apps em uma turma de alunos resulta em muitos trabalhos com distintas funcionalidades. Para que o avaliador consiga analisar e corrigir a implementação desenvolvida por cada estudante de forma consistente, com exatidão e evitando bias, requer muito tempo e dedicação do avaliador (ZEN et al., 2011). Isso considerando que o avaliador é formado no assunto avaliado, o que nem sempre é verdade, pois atualmente faltam professores de computação na Educação Básica (TIC EDUCAÇÃO, 2016). Comumente ocorre de professores de outras áreas e entusiastas com a iniciativa de ensinar computação têm se prontificado, o que acaba sendo mais desafiador o processo de avaliação das práticas (DELUCA; KLINGER, 2010).

Uma maneira de auxiliar professores a analisarem e avaliarem projetos de uma turma inteira é utilizando ferramentas de software que auxiliam na análise e avaliação do código de forma (semi-) automática (RASHKOVITS, LAVY, 2013), chamadas de analisadores de código ou *autograders*.

Um Analisador de Código é um sistema de software que averigua um código para determinar suas características, defeitos e/ou componentes utilizados (AHO,

SETHI, ULLMAN, 1986). São muito utilizados para compilar código, ou seja, no processo de tradução do código em linguagem de programação de alto nível em linguagem de máquina (baixo nível). Nesse caso a análise é dividida em três etapas (AHO, SETHI, ULLMAN, 1986):

- **Análise Léxica**: nessa etapa o analisador lê os caracteres do código e agrupa-os em tokens (identificadores) para uso nas etapas posteriores.
- **Análise Sintática**: é verificado se a cadeia de tokens resultados pelo analisador léxico é condizente com as regras da linguagem de alto nível do código analisado. Se sim, uma estrutura hierárquica dos tokens é gerada. E
- **Análise Semântica**: com base na estrutura hierárquica da análise sintática, identificam-se os operadores e operandos das expressões, enunciados e verifica se fazem sentido para linguagem de baixo nível.

A análise de código também pode ser usada para outras finalidades, como a análise da qualidade do código desenvolvido (RASHKOVITS, LAVY, 2013). Neste caso, podendo ser dividida em dois tipos de análise: a estática e a dinâmica (AHO, SETHI, ULLMAN, 1986). A análise dinâmica é o processo de executar o código e comparar os resultados gerados com uma saída de controle, conforme especificado em um caso de teste (BENFORD et al., 1995). Assim é verificado se a aplicação realmente resolve o problema proposto e está de acordo com sua especificação.

A análise estática é o processo de examinar o código-fonte sem executar o programa. Ela é usada para verificar a existência de erros de programação, como código-morto, checar a complexidade, detectar de palavras-chave para extração de informações e verificar vulnerabilidade da segurança (AHO, SETHI, ULLMAN, 1986). Por isso, é o tipo de análise mais comum para analisar e avaliar de forma automática os trabalhos práticos de programação (STRIEWE, GOEDICKE, 2014), sendo o foco do presente trabalho.

O processo de análise estática não é padronizado. Mas geralmente inicia-se com uma etapa chamada *parsing*, que é muito parecido com as análises léxica e sintática de um compilador, resultando numa estrutura de dados, como grafo ou árvore, com toda a informação de interesse. O verificador analisa e avalia todos os elementos presentes na estrutura de dados, com base em um conjunto de Regras de Análise, que informam ao verificador quais pontos e componentes do código devem ser analisados e de que forma (Figura 11).

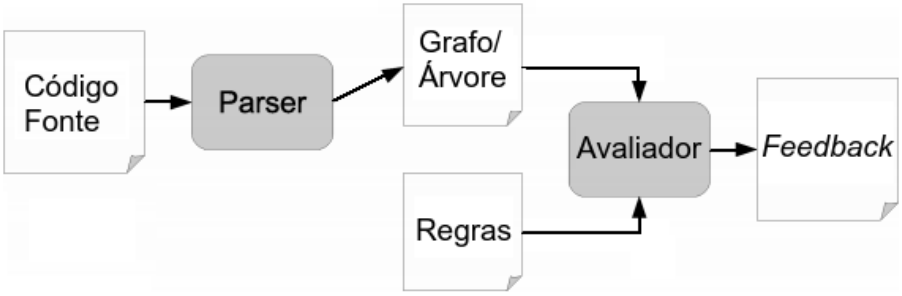


Figura 11: Processo de análise estática (STRIEWE, GOEDICKE, 2014).

O verificador pode ser subdividido em partes (Figura 12). A primeira etapa faz diversas análises sobre a estrutura de dados para verificar funções declaradas no

código, complexidade, paralelismo, entre outros. Na segunda etapa com auxílio de Regras de Análise são analisados os resultados da etapa anterior para quanto cada aspecto é significativo. Na terceira e última parte do verificador são concluídas, respectivamente, a nota e o *feedback* instrucional.

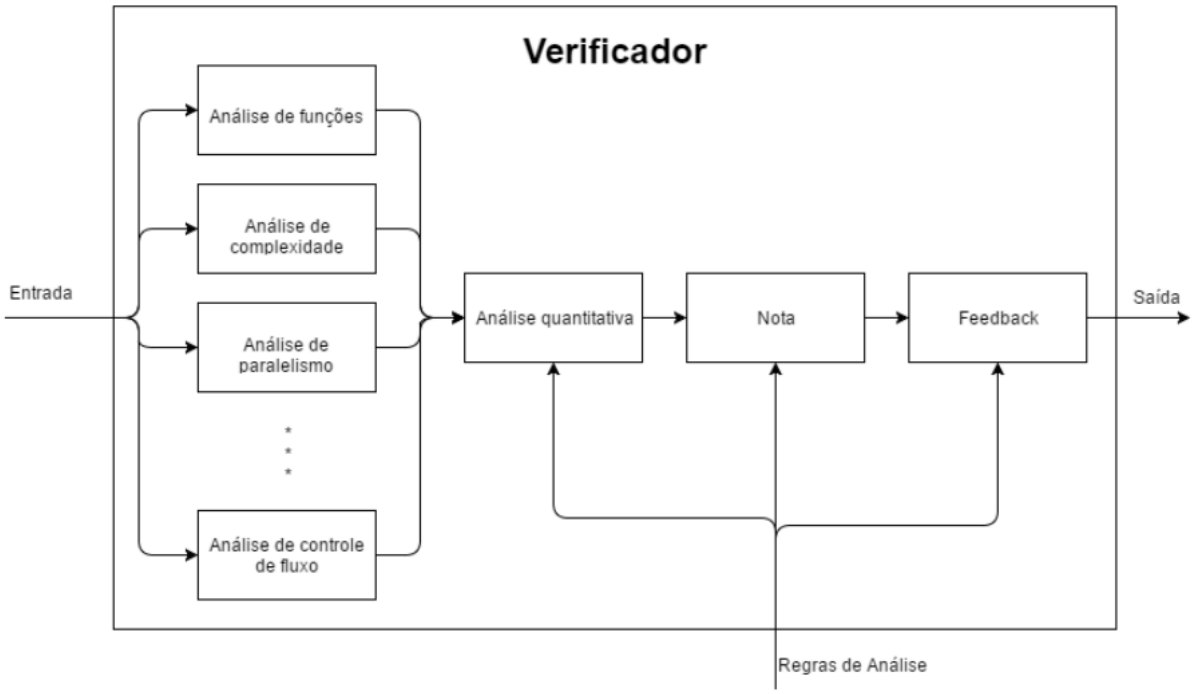


Figura 12: Visão do fluxo do verificador adaptado voltado ao ensino (SINTHSIRIMANA, PANJABUREE, 2013)

No contexto educacional, as ferramentas de análise estática de código podem ser utilizadas para contribuir para a aprendizagem do aluno, principalmente quando aplicado sobre aplicativos sem gabarito pré-definido. Isso permite o desenvolvimento de aplicativos, como atividades práticas de programação, sem uma assistência intensa do professor (STRIEWE; GOEDICKE, 2014).

3. Estado da Arte e Prática

A fim de levantar o estado da arte e prática são apresentados neste capítulo os resultados de um mapeamento sistemático da literatura de ferramentas de avaliação automática de design de interface de aplicativos desenvolvidos com o App Inventor. Este mapeamento é feito com base no processo definido por Petersen et al. (2008), atualizando e ampliando o mapeamento sistemático de Porto, Gresse von Wangenheim e Barbosa (2017).

3.1 Definição do protocolo do mapeamento

O objetivo desse mapeamento é de responder a seguinte pergunta de pesquisa: Quais meios existem para **avaliação de design visual de interface** de aplicativos móvel de *smartphones touchscreen* Android ou aplicativos desenvolvidos no App Inventor?

Esta questão de pesquisa é refinada nas seguintes perguntas de análise:
PA1. Quais modelos/ferramenta de avaliação de design visual existem?
PA2. Quais as características do modelo de avaliação do design visual?
PA3. Quais as características instrucionais do/da modelo/ferramenta?
PA4. Como o modelo/ferramenta foi desenvolvida?
PA5. Como a qualidade do modelo/ferramenta foi avaliada?

Para realizar as buscas foram escolhidas as principais bases de dados e bibliotecas digitais do campo da computação, que são ACM Digital Library, IEEE Xplore Digital Library, ScienceDirect, SpringerLink, Wiley e Scopus. Considerando publicações disponíveis via o Portal CAPES[8]. Além dessas bases, com o intuito de cobrir uma gama maior de publicações, também foram conduzidas pesquisas no Google Scholar, que indexa um grande conjunto de dados de diversas fontes de produção científicas distintas (HADDAWAY et al., 2015).

Com base na pergunta de pesquisa, para calibração da *string* de busca foram feitas buscas informais para auxiliar na definição a *string* de busca a serem aplicadas nas bases de dados (Tabela 13).

Tabela 13: Termos de busca e respectivos sinônimos ou termos similares

Termos de Buscas	**Sinônimos**
App Inventor	Android
Grading	Assessment
Mobile application	App
Usability	User experience, ux, user interface, design

[8] Um portal da internet para acesso ao conhecimento científico produzido internacionalmente, gerido pelo Ministério da Educação do Brasil e direcionado a instituições autorizadas, incluindo universidades, agências governamentais e empresas privadas (www.periodicos.capes.gov.br)

Após a calibração, definiu-se uma *string* de busca específica para aplicar nas bases de dados:

(“app inventor” OR android) AND (grading OR assessment) AND (app OR “mobile application”) AND (“user experience” OR ux OR “user interface” OR design OR usability)

Com a *string* de busca definida, ela foi adaptada para realizar a busca em cada uma das bases de dados consideradas (Tabela 14).

Tabela 14: *Strings* de buscas usadas nas bases de dados

Base de Dados	***String* de busca**
SpringerLink	("app inventor" OR android) AND (grading OR assessment) AND (app OR "mobile application") AND ("user experience" OR ux OR "user interface" OR design OR usability)
ScienceDirect	("app inventor" OR android) AND (grading OR assessment) AND (app OR "mobile application") AND ("user experience" OR ux OR "user interface" OR design OR usability)
Wiley	("app inventor" OR android) AND (grading OR assessment) AND (app OR "mobile application") AND ("user experience" OR ux OR "user interface" OR design OR usability)
IEEE	("app inventor" OR android) AND (grading OR assessment) AND (app OR "mobile application") AND ("user experience" OR ux OR "user interface" OR design OR usability)
ACM Digital	("app inventor" OR android) AND (grading OR assessment) AND (app OR "mobile application") AND ("user experience" OR ux OR "user interface" OR design OR usability)
Scopus	("app inventor" OR android) AND (grading OR assessment) AND (app OR "mobile application") AND ("user experience" OR ux OR "user interface" OR design OR usability)
GoogleScholar	("app inventor" OR android) (grading OR assessment) (app OR "mobile application") ("user experience" OR ux OR "user interface" OR design OR usability)

Critérios de inclusão e exclusão:

- São considerados artigos científicos e artefatos que apresentem rubricas/*checklist* para avaliação da qualidade do design de interface de apps.
- São incluídos apenas artefatos em inglês ou em português.
- São incluídos apenas artefatos que são acessíveis por meio do portal da CAPES
- Com a intenção de tornar a pesquisa mais ampla e abranger um maior resultado possível, esta pesquisa não restringe por ferramentas que façam avaliação automática e nem para aplicativos serem desenvolvidos no App Inventor.
- Também com a intenção de tornar a pesquisa mais ampla e abranger um maior resultado possível, a pesquisa não é limitada ao intuito de aplicação da avaliação de design de interface ser no contexto educacional.

Critérios de qualidade: Foram considerados apenas artigos que apresentam informações sobre os meios de avaliar o design visual de interface de aplicativos móvel de *smartphones touchscreen* Android ou aplicativos desenvolvidos no App Inventor.

3.2 Execução da Busca

A busca dos artigos foi realizada em maio e junho de 2018 pela autora do presente trabalho em conjunto com o mestrando Igor da Silva Solecki e com auxílio e orientação da Christiane Gresse von Wangenheim. A busca inicial resultou em 21532 artigos. Por isso nas bases de caso SpringerLink, Wiley e Scopus foram analisados apenas os artigos das disciplinas ou *subject* Ciência da Computação e Educação. No Google Scholar foram analisadas as duzentas publicações mais relevantes, pois a quantidade de publicações resultantes na busca no Google Scholar foi muito grande. Nas demais bases (Science Direct e ACM) foram analisados todos os artigos resultantes. Com os artigos analisados definidos, a execução da busca foi dividida em 3 etapas (Tabela 15):

- A partir dos resultados da busca, na primeira etapa 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 239 artigos potencialmente relevantes.
- Na segunda etapa foi relido o resumo e outras informações do artigo, como conclusão e análise das fotos; Essa etapa resultou em 63 artigos potencialmente relevantes.
- Na terceira os artigos foram lidos na íntegra. Resultou em 7 artigos relevantes para a pesquisa.

Tabela 15: Quantidade de artigos resultantes em cada etapa da execução da busca.

Base de dados	Quantidade de artigos resultantes da busca	Quantidade de artigos analisados	1º etapa	2ª etapa	3ª etapa
SpringerLink	950	314	51	10	0
ScienceDirect	647	647	78	19	2
Wiley	565	160	21	1	0
IEEE	0	0	0	0	0
ACM Digital	42	42	9	2	1
Scopus	1928+	1239	63	22	3
GoogleScholar	17400	200	17	9	1
Total	21532	2571	239	63	7

Durante a primeira etapa da busca, vários artigos foram desconsiderados porque apresentavam o desenvolvimento de aplicativos com o intuito de auxiliar no ensino de algo ou para auxílio na detecção ou serviço de saúde, tendo como avaliação se pessoas tiveram melhor rendimento com o app. Alguns apresentam pesquisas e análises da importância de utilizar processo de engenharia de software. Outros apresentam meios de análise para identificar apps maliciosos (comprometam segurança) e ou como auxílio para detecção/serviço de saúde.

Na segunda etapa 176 artigos foram excluídos, pois apesar de apresentarem a utilização de apps em sala de aula e alguns voltados em ensinar o desenvolvimento de apps, não apresentam a forma de avaliação do app desenvolvido, apenas o nível de satisfação dos alunos. Totalizando em 7 artefatos relevantes.

Como o mapeamento sistemático consiste na atualização e ampliação do mapeamento sistemático realizado por Porto, Gresse von Wangenheim e Barbosa (2017), havia a expectativa de encontrar todas as fontes citadas. Porém isso não ocorreu, pois artefatos de interesse como Technovation (2014) e Android (2018) não estavam dentre os 7 artefatos relevantes. Então esses artefatos foram incluídos na pesquisa por ter conteúdo relevante. Totalizando em 9 artefatos relevantes (Tabela 15).

3.3 Análise dos resultados

Para responder à questão de pesquisa, extraímos informações relevantes às perguntas de análise conforme especificado na Tabela 16.

Tabela 16: Especificação das informações extraídas.

Pergunta de análise	Dados a extrair	Descrição
PA1. Quais modelos/ferramenta de avaliação de design visual existem?	Nome	O nome ou o autor da UI
	Referência	Referência bibliográfica
PA2. Quais as características do modelo de avaliação do design visual?	Critérios sendo avaliados pelo modelo/ferramenta	
	Baseado em uma norma/guia de estilo	
	Tipo do modelo/ferramenta	checklist/rubrica
	Avaliação automatizada	sim/não. Se sim, apresentar descrição.
	Qual o contexto de aplicação?	Apps Android, etc.
PA3. Quais as características instrucionais da do modelo/ferramenta?	A proposta é usada em contexto educacional?	sim/não
	Descrição geral	Breve descrição geral da UI apresentando as suas principais características
	Ambiente de programação	Ambiente/linguagem de programação utilizado na UI
	Língua	A(s) língua(s) em qual a UI está disponível
	Licença	Licença de uso da UI
	Nível escolar	Nível escolar para qual a UI é projetada

PA4. Como o modelo/ferramenta foi desenvolvida?	Método de desenvolvimento da UI	Indicação do método adotado para o desenvolvimento da do modelo/ferramenta de avaliação
PA5. Como a qualidade do modelo/ferramenta foi avaliada?	Tipo de estudo de avaliação do modelo/ferramenta	Indicação do tipo de estudo (research design) da avaliação do modelo/ferramenta
	Fatores avaliados	Indicação dos fatores que foram avaliados
	Método de coleta de dados	Indicação do(s) método(s) utilizados para coleta de dados na avaliação do modelo/ferramenta
	Tamanho de amostra	Quantidade de pontos de dados utilizadas na a avaliação do modelo/ferramenta
	Método de análise de dados	Indicação do(s) método(s) de análise de dados utilizado(s) na avaliação do modelo/ferramenta
	Descobertas	Descrição dos principais resultados, pontos positivos e negativos identificadas na avaliação do modelo/ferramenta

Os artigos foram lidos de forma completa e os dados foram extraídos pelo Grupo de Qualidade de Software (GQS), mais especificamente pela autora em conjunto com o mestrando Igor da Silva Solecki e com auxílio e orientação da pesquisadora sênior Christiane Gresse von Wangenheim. Nos casos em que o artigo não apresenta nenhuma informação a ser extraída, indicamos a falta desta informação como não informada (NI).

PA1. Quais modelos/ferramenta de avaliação de design visual existem?

No total foram encontrados 9 artefatos que apresentam alguma forma de avaliação de design visual de interface de aplicativos móvel de *smartphones touchscreen* Android ou aplicativos desenvolvidos no App Inventor (Tabela 17).

Tabela 17: Documentos resultantes da execução de busca.

Citação	**Referência Bibliográfica**
(WAGNER et al., 2013)	WAGNER, A.; GRAY, J.; CORLEY, J.; WOLBER, D. **Using App Inventor in a K-12 Summer Camp**. SIGCSE '13 Proceeding of the 44th ACM technical symposium on Computer science education, Colorado, USA. 2013. p. 621-626.
(ZAPATA et al., 2014)	ZAPATA, B. C.; NIÑIROLA, A. H.; IDRI, A.; FERNÁNDEZ-ALEMÁN, J. L.; TOVAL, A. **Mobile PHRs Compliance with Android and iOS Usability Guidelines.** Journal of Medical Systems. vol. 38, n. 81. 2014.
(TECHNOVATION, 2014)	TECHNOVATION. **Teacher and Mentor Lesson Guide Lessons 1 - 6.** Fev. 2014. Disponível em: <https://technovationchallenge.org/wp-content/uploads/2014/01/TeacherMentorLessonGuide_L1_6.pdf>. Acessado em: Junho de 2018.
(SHERMAN, MARTIN, 2015)	SHERMAN, M.; MARTIN, F. **The Assessment of Mobile Computational Thinking**. Journal of Computing Sciences in Colleges. vol 30 n.6, 2015. p. 53-59.
(HOEHLE, ALJAFARI, VENKATESH, 2016)	HOEHLE, H.; ALJAFARI, R.; VENKATESH, V. **Leveraging Microsoft's mobile usability guidelines: Conceptualizing and developing scales for mobile application usability**. International Journal of Human-Computer Studies. vol 89, 2016, p.35-53
(GRESSE VON	GRESSE VON WANGENHEIM, C.; Witt, A. T.; Borgatto, A. F.;

WANGENHEIM et al., 2016)	Nunes, J. V.; Lacerda, T. C.; Krone, C.; Souza, L. O. **An Usability Score for Mobile Phone Applications based on Heuristics**. International Journal of Mobile Human Computer Interaction archive. vol 8. n. 1. 2016. p 23-58.
(INES et al., 2017)	INES, G.; MAKRAM, S.; MABROUKA, C.; MOURAD, A. **Evaluation of Mobile Interfaces as an Optimization Problem**. Procedia Computer Science vol 112, 2017, p. 235-248.
(GRESSE VON WANGENHEIM et al., 2018)	GRESSE VON WANGENHEIM, C.; HAUCK, J. C. R.; DEMETRIO, M. F. ; PELLE, R.; ALVES, N. C.; BARBOSA, H.; AZEVEDO, L. F. **CodeMaster – Automatic Assessment and Grading of App Inventor and Snap! Programs**. Informatics in Education, vol. 17, n. 1, 2018, p. 117–150.
(ANDROID, 2018)	ANDROID. Qualidade do aplicativo principal. Disponível em: <https://developer.android.com/docs/quality-guidelines/core-app-quality#ux>. Acessado em: Junho de 2018

Pode se observar que a preocupação em avaliar o design visual de interface de aplicativos móvel de *smartphones touchscreen* Android é algo existente desde 2013. Mas não há muitas pesquisas sobre isso, o que possibilita concluir que não há projetos que aprofundam nesse tema (Figura 13).

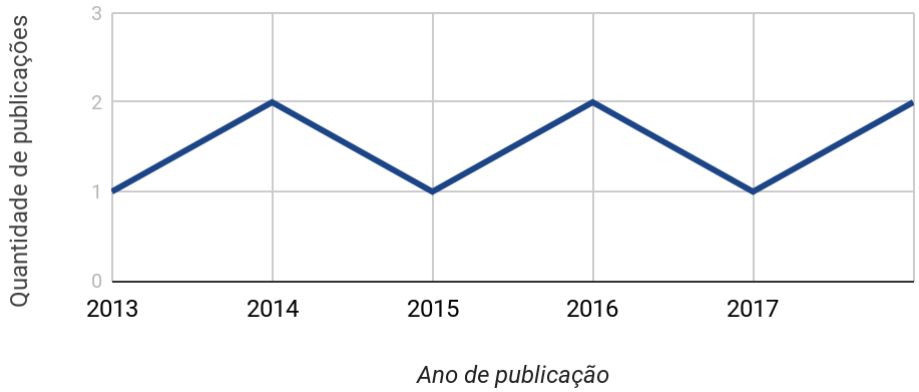


Figura 13: Quantidade de publicações sobre avaliação do *design* de interface de *smartphone mobile.*

PA2. Quais as características do modelo de avaliação do design visual?

Wagner et al (2013) propôs uma unidade instrucional para ensinar o desenvolvimento de aplicativos Android para alunos entre 9º e 12º ano, que avalia 11 objetivos de aprendizagem. Sendo que apenas o décimo corresponde ao design de interface de apps.

> 10. Criando uma interface de usuário gráfica (GUI) e codificando o componentes da GUI

Zapata et al. (2014) apresenta um questionário para aplicativos Android e iOS, pois segue as *guidelines* de ambos sistemas operacionais (Figura 14). Esse questionário foi desenvolvido para avaliar aplicativos de registros pessoais de saúde[9].

[9] Registros pessoais de saúde, em inglês, *Personal Health Records* (PHR).

Table 2 Questionnaire and survey results

Question	Survey
Q1. Style.	
Q1. 1. Is the writing style simple and informal and is the second person used to talk to the user?	3.64
Q1. 2. Are pictures used to explain ideas?	3.95
Q1. 3. Are pre-defined icons used for common actions?	4.18
Q1. 4. Does the app adapt to both horizontal and vertical orientations?	3.59
Q2. Behavior.	
Q2. 1. Are the user preferences learned over time?	3.45
Q2. 2. Do the elements react to the user's gestures by changing color or illumination?	3.64
Q2. 3. Are there confirming messages showing warning information related to actions that the user needs to consider?	3.55
Q2. 4. Are there acknowledging messages to let users know that the action they have invoked has been completed?	4.23
Q2. 5. Do long tasks show non-stationary activity indicators?	4.27
Q3. Structure.	
Q3. 1. Is the app loaded immediately without any splash screen or startup experience?	4.00
Q3. 2. Is the login delayed to allow the user to use a particular functionality first?	3.59
Q3. 3. Are suggested structure patterns used: action/tool bar, tab bar (top for Android, bottom for iOS), spinner or navigation drawer?	3.73
Q3. 4. Is the navigation consistent when moving between hierarchical screens?	4.23

Figura 14: Questionário por ZAPATA et al (2014).

Technovation (2014) apresenta uma rubrica em que somente um item (*User Interface*) é dedicado a avaliar aspectos da usabilidade do aplicativo (Figura 15). Tal item se restringe a definir, em uma escala de 1 a 4, quão intuitiva e fácil de usar é a interface do aplicativo. Há instruções ao avaliador, informando que é fazer a análise de acordo com seus conhecimentos, resultando numa avaliação um caráter subjetivo.

Judging Rubric

DIRECTIONS: Evaluate these items objectively to the extent that you can. It is ok for every team to gain the highest score in each of these items. In fact if the team completed the entire Technovation curriculum, they should receive a perfect score on every item in this section.					
Objective items	**0**	**2**		**4**	**SCORE**
Did the girls identify a real problem in their community?	No	It somewhat, but not fully, addresses a local, real problem		Yes	
Does the app solve the problem that they identified?	No	It somewhat, but not fully, solves the problem		Yes	
Is the prototype they submitted fully functional? (It should contain at least 3 screens with all buttons and links functional and no obvious bugs.)	No, there are major defects.	Mostly, except for a few minor issues. I can still get the general idea.		Yes	
DIRECTIONS: Evaluate these items according to your expert, subjective judgment. These items should be measured relative to the quality of the other apps you judge in your pool. Every team should not be capable of receiving the highest score- these items should rank the different entries from ones that just fulfill the requirements to ones that are truly extraordinary.					
Subjective items	**2**	**4**	**6**	**8**	**SCORE**
Overall Pitch Quality. Is the Pitch compelling, and would you invest resources in this team?	Not at all	Possibly	Compelling, I would invest	Yes! Top of my investment portfolio	
	1	**2**	**3**	**4**	**SCORE**
Business Plan. Do they have a sound business plan? (thorough market analysis, viable marketing plan, etc.)	Not at all	A little	Quite a lot	Definitely	
Future Vision. Do they have a practical vision for extending the capabilities of their app beyond the prototype?	Not at all	A little	Quite a lot	Definitely	
Dynamic Functionality. Does the app go beyond static content and include dynamic functionality?	Not at all	A little	Quite a lot	Definitely	
User Interface. Is the app's interface intuitive and easy to use?	Not at all	A little	Quite a lot	Definitely	
Bonus Points! Does the App include the following?			**0**	**2**	**SCORE**
External Data Integration (calls data from an external API)			No	Yes	
An Especially Creative Design			No	Yes	
				TOTAL SCORE	

Figura 15: Rubrica para avaliação de aplicativos desenvolvido pela Technovation (2014)

Sherman e Martin (2015) apresentam uma rubrica para avaliar aplicativos desenvolvidos no App Inventor desenvolvidos por graduandos de diferentes cursos, ou seja, por alunos de diferentes cursos, além do curso de Ciência da Computação. Dentre os critérios de avaliação, o critério "*Screen Interface*" é o que caracteriza a complexidade da apresentação visual do aplicativo (Figura 16). A rubrica completa está em (SHERMAN et al., 2014).

	1 point	2 points	3 points	4 points
1. Screen Interface	Single screen with five or fewer visual components that do not programmatically change state.	Single screen with more than five visual components that do not programmatically change state.	Single screen, where some components programmatically change state based on user interaction with the app.	Two or more screens; screens may be implemented as screen components, or by programmatically changing visibility of groups of visual components.

Figura 16: Rubrica Parcial (SHERMAN, MARTIN, 2015)

Hoehle, Aljafari e Venkatesh (2016) apresentam uma lista de critérios baseada na *guideline* da Microsoft (Figura 17). O intuito é avaliar a interface gráfica de aplicativos de mídia social, como Facebook, LinkedIn, Twitter, My Space and Googleþ. de diferentes SO's: iPhone, BlackBerry, Android, Windows Mobile, Symbian, outros.

Axial codes	Subcategory	Open codes derived from Microsoft's guidelines
Aesthetic graphics	Well-designed and aesthetical graphics	• Images and graphics must enhance and support the user experience. • Graphics should be designed aesthetically and should not replace or overlap important textual content.
Color	Contrast and color	• The text of applications should have a good contrast with the background. • Color assists in the organization and grouping of information, helping to focus attention, convey differentiation, and establish relationships and visual hierarchies between elements. • Color can help readers scan information and quickly identify structural or functional elements, such as headers, menu items and hyperlinks. • When used incorrectly, however, color can easily distract attention from the task at hand. • If a color is being used to convey a specific meaning (for example, red to warn of danger or an error), chosen colors should be universally associated with the intended meaning and potential conflicts that result from cultural misinterpretation should be avoided.
Control obviousness	Consistent use of controls	• User controls should be obvious and interaction should be familiar, clear, and trustworthy. • The design should be consistent, logical, and coherent both within the application and within the target platform. • Controls and application features should be used consistently.
Entry point	Application accessibility and application entrance points	• Users should have several options to choose from if aiming to access an application. • The application should be designed in a way that it is accessible via direct controls or application menus, or a combination of both. • Well-designed applications should have several points to access a menu or an application.
Fingertip-size controls	Button size and control size	• Interface elements should not be smaller than the smallest average finger pad, that is, no smaller than 1 cm (0.4") in diameter or a 1 cm × 1 cm square. • The width of a finger limits the density of items on screen. If the items are too close, the user will not be able to choose a single one. • As the user is more likely to touch higher on the button by mistake than on either side, consider the height of your buttons and icons. • Essential information or features, such as a label, instructions, or sub-controls should be placed below an interface element that can be touched, as it may be hidden by the user's own body.
Font	Font style	• Font is an important consideration for designing applications because users appreciate well-chosen font styles. • Devices normally have one standard font style, which should be used as the application's default typeface.
Gestalt	Gestalt principles and proximity of interface elements	• Information and content should be organized in accordance with the Gestalt principles. • Each part of the application is affected by what surrounds it. • Users should be able to quickly make sense of the elements on-screen and understand what functionality or data they represent. • Elements that are close together are naturally perceived as being related. • Because of the small screen size, however, the use of proximity may be limited.
Hierarchy	Hierarchical menu structure and application navigation	• Drill-down views offer hierarchical navigation for applications that need to provide access to hierarchies of information. • The layout of the various views in the navigation chain is not restricted to lists, and should be optimized for the type of content and/or functionality. • In all cases, users navigate hierarchies in drill-down views by tapping items in a view to 'drill down' another level in the information hierarchy. • Users should also be able to move back toward the top or 'root' of the hierarchy and back commands should be also available. • Tapping 'back' at any level takes the user up to the previous level in the hierarchy.
Subtle animation	Animation use and simplicity of animated content	• Animations should be kept simple. • Avoid complex animations and, in particular, multiple simultaneous timeline-based motion animations. • Avoid unnecessary alpha effects or gradients and do not combine transitions with changes in transparency or other graphical effects because they are likely to slow down the animations.
Transition	Transition and flow of user interface elements	• Well-designed transitions help users and make the user interface more engaging. • Without transitions, the interaction feels less natural. • Transitions can be used to inform the users of what is going on. • Transitions should be used wisely and it is useful to test how users feel about them. • Transitions can easily create a WOW-factor to applications. • If every user interface element is twitching and turning wildly, it could as easily exhaust the user.

Figura 17: Lista de critérios para avaliação do design de interface (HOEHLE, ALJAFARI, VENKATESH, 2016)

Ines et al (2017) apresenta duas listas de avaliação de interface de aplicativos Android. Uma lista de métricas da qualidade do design de interface de um aplicativo (Figura 18). A segunda apresenta os defeitos possíveis no design de interface de um aplicativo (Figura 19). A proposta dessa publicação é uma ferramenta automatizada que avalie a interface gráfica de aplicativos *mobile* Android, a partir da lista de métricas da qualidade, e conclua quais defeitos dentre os apresentados em (Figura 19) o aplicativo apresenta.

Metrics	Description
Density	It represents the percentage of workload of MUIs. It refers the extent to which interface is covered with objects (Components).
Regularity	It measures the uniformity of MUI components. It is sensitive to the number of vertical and horizontal alignment and to the spacing between components.
Sequence	It represents the extent to which the MUI is ordered based on reading pattern of left-to-right, top-to-bottom that is common in Western cultures.
Grouping	It is the number of group on the MUI. It is the degree to which all elements seem belong together.
Simplicity	It measures the complexity of MUI by counting the number of rows and columns on the interface. It is sensitive to the number of elements on the screen.
Homogeneity	It measures the distribution of elements among the four quadrants of the MUI. Well homogeneity is achieved by an equal distribution of the elements among the four quadrants of the MUI.
Unity	It measures the coherence of MUI elements. Unity is grouping all MUI elements to appear like a one piece.
Symmetry	It gives by an equal distribution of the quantity of elements on the right and the left columns of mobile user interface.

Figura 18: Lista de métricas de qualidade do design de interface (INES et al., 2017).

Defects	Description
Incorrect layout of widgets	Incorrect layout of widgets: is related to the incorrect arrangement of MUI components (e.g., bad alignment, bad orientation, wrong position etc.)[14].
Overloaded	It is a bad density of MUI. In this case, the users find the MUI with high density and too crowded and not easy to read[6].
Complicated MUI	It is a measurement of structural sophistication of MUI layout. MUI includes more and more widgets to propose more features that are greater than the users' needs. So, this large number of features may lead to making MUIs complex[3].
Incorrect data presentation	It is an adaptiveness of the content which related to the incorrect extraction and presentation of information[22].
InCohesion of MUI :	It is a measure of the degree of interrelatedness of software component parts. In the MUI design, cohesion can be applied to show how the widgets are related to each other for a given MUI[2].
Difficult navigation	It is related to the lack of supplementary information and lack of descriptive labels that will be shown when a user selects an object (i.e. field). The latter problem is related to consistency [1].
Ineffective appearance of widgets	This problem occurs when MUI widgets follow an unexpected layout (e.g., wrong order). The presentation form and grouping of options does not match the mental model of the user so they react with confusion or cannot find the information although it is provided[6].
Imbalance of MUI	It is an unequal distribution of the quantity of interactive objects such as a button, a textfield and a textbox on the right and the left columns of a MUI.

Figura 19: Lista de defeitos do design de interface (INES et al., 2017).

Gresse von Wangenheim et al. (2018) apresenta uma rubrica para avaliação de aplicativos desenvolvidos no App Inventor, que foi baseada na rubrica apresentada por (Sherman et al., 2014). Dentre os critérios, apenas dois são do contexto de usabilidade da interface (Figura 20). Essa publicação também apresenta

a ferramenta web gratuita CodeMaster, que avalia automaticamente aplicativos com base nos critérios da rubrica.

Criteria	Level of Performance			
	0	1	2	3
Screens	Single screen with visual components that do not programmatically change state.	Single screen with visualcomponents that do programmatically change state.	Two screens with visual components and one screen with visual components that do programmatically change state.	Two or more screens with visual components and two or more screens with visual components that do programmatically change state.
User Interface	Uses one visual component without arrangement.	Uses two or more visualcomponents without arrangement.	Uses five or more visual components with one type of arrangement.	Uses five or more visual components with two or more types of arrangement.

Figura 20: Rubrica parcial para avaliação de projetos App Inventor (GRESSE VON WANGENHEIM et al. 2018)

Em Android (2018) é apresentado uma *checklist* para avaliação de qualidade de software de aplicativos Android. Este checklist é um meio para que desenvolvedores de aplicativos possam avaliar a qualidade de seus softwares. Os critérios além de avaliar o design de interface, também avaliam as funcionalidades e outros fatores de qualidade. Dentre os critérios apresentados, apenas os mostrados na figura 21 são referentes ao design visual e de interação com usuário. Android (2018) também apresenta um procedimento de avaliação manual, indicando quais passos o avaliador deve seguir para detectar incompatibilidades do aplicativo com a checklist (Figura 22).

Área	ID	Descrição	Testes
Design padrão	UX-B1	O aplicativo segue as diretrizes de Projeto para Android e usa padrões e ícones da interface do usuário comuns: a. O aplicativo não redefine a função esperada de um ícone do sistema (como o botão Voltar). b. O aplicativo não substitui um ícone do sistema por um ícone completamente diferente se acionar o comportamento da interface do usuário padrão. c. Se o aplicativo fornece uma versão personalizada de um ícone padrão do sistema, o ícone se assemelha muito ao ícone do sistema e aciona o comportamento padrão do sistema. d. O aplicativo não redefine nem usa incorretamente padrões da interface do usuário Android de forma que esses ícones ou comportamentos possam enganar ou confundir o usuário.	CR-all
Navegação	UX-N1	O aplicativo suporta a navegação do botão Voltar do sistema padrão e não utiliza nenhum prompt personalizado do "botão Voltar" na tela.	CR-3
	UX-N2	Todos os diálogos são descartados usando o botão Voltar.	CR-3
	UX-N3	Pressionar o botão Home em qualquer ponto direciona o usuário para a tela inicial do dispositivo.	CR-1
Notificações	UX-S1	As notificações seguem as diretrizes de Projeto para Android. Principalmente: a. Várias notificações são empilhadas em um único objeto de notificação, quando possível. b. As notificações são persistentes apenas se estiverem relacionadas a eventos contínuos (como reprodução de música ou uma chamada telefônica). c. As notificações não podem conter publicidade ou conteúdo não relacionado à função principal do aplicativo, a menos que o usuário tenha aceitado.	CR-11
	UX-S2	O aplicativo usa notificações apenas para: a. Indicar uma mudança no contexto relacionada pessoalmente ao usuário (como uma mensagem em entrada) ou b. Expor informações/controles relacionados a um evento contínuo (como a reprodução de música ou chamada telefônica).	CR-11

Figura 21: Checklist para avaliação do projeto visual e interação com usuário dos aplicativos Android (Android, 2018)

Tipo	Teste	Descrição
Conjunto principal	CR-0	Navegue por todas as partes do aplicativo — todas as telas, diálogos, configurações e fluxos do usuário. a. Se o aplicativo permitir edição ou criação de conteúdo, reprodução de jogos ou mídia, certifique-se de inserir esses fluxos para criar ou modificar conteúdo. b. Ao executar o aplicativo, realize mudanças temporárias na conectividade de rede, função da bateria, GPS ou disponibilidade de localização, carregamento do sistema e assim por diante.
	CR-1	Em cada tela do aplicativo, pressione a tecla Home do dispositivo e reinicie o aplicativo da tela All Apps.
	CR-2	Em cada tela do aplicativo, alterne para outro aplicativo em execução e volte para o aplicativo em teste usando a opção de troca de aplicativos Recents.
	CR-3	Em cada tela do aplicativo (e caixas de diálogo), pressione o botão Voltar.
	CR-5	Em cada tela do aplicativo, gire o dispositivo entre a orientação vertical e horizontal pelo menos três vezes.
	CR-6	Alterne para outro aplicativo para enviar o aplicativo de teste para o segundo plano. Vá para Settings e verifique se o aplicativo de teste está executando qualquer serviço enquanto está em segundo plano. No Android 4.0 e posterior, acesse a tela Apps e localize o aplicativo na guia "Running".
	CR-7	Pressione o botão liga/desliga para colocar o dispositivo em espera e pressione-o novamente para ligar a tela.
	CR-8	Configure o dispositivo para ser bloqueado quando o botão liga/desliga for pressionado. Pressione o botão liga/desliga para colocar o dispositivo em espera e pressione esse botão novamente para ligar a tela e desbloquear o dispositivo.
	CR-9	Para dispositivos com teclados virtuais, exiba e oculte o teclado pelo menos uma vez. Para dispositivos com teclados externos, conecte o dispositivo à entrada do teclado.
	CR-10	Para dispositivos com uma porta de monitor externa, conecte o monitor externo.
	CR-11	Acione e observe na gaveta de notificações todos os tipos de notificações que o aplicativo pode exibir. Expanda as notificações onde aplicável (Android 4.1 e superior) e toque em todas as ações disponíveis.
	CR-12	Verifique as permissões solicitadas pelo aplicativo em Settings > App Info.

Figura 22: Lista Parcial do procedimentos de avaliação de aplicativos Android.

Gresse von Wangenheim et al. (2016) apresenta a versão 0.2 da *checklist* MaTCH que avalia a usabilidade de aplicativos para *smartphones touchscreen* (Tabela 18). Cujo desenvolvimento foi baseado nas 10 heurísticas de Nielsen (1993).

Tabela 18: *Checklist* de usabilidade de aplicativos para *smartphones* v0.2 (GRESSE VON WANGENHEIM et al., 2016).

Usability Checklist for Applications of Touch-Screen Phones v0.2						
Inspector:				**Date:**		
Application name/Version:		**Operating System:**	**Device:**			
Heuristics	**Items**	**N**	**P**	**F**	**N/A**
Visibility of system status	1. For each user action, the application provides immediate and appropriate *feedback* on their status?				
	2. The selected items are clearly distinct from the others?				
	3. Messages related to critical and contextual information, such as, battery or network status are prioritized?				
	4. Messages about the status of the application use a clear and concise language?				
	5. All screens have an identification?				
	All screens keep menus and functions that are common to the application accessible?				
	7. Provides a status update for slower operations?				
	8. The application provides information about its version?				
	9. Offers other forms of feedback, beyond textual (light, sound, vibration)				
	10. When a message is displayed, the time is enough for reading?				
	Comments:				
Match between system and real world	11. The meaning of symbols and icons is understandable and intuitive?				
	12. Information is arranged in a logical and natural order?				
	Comments:				
User control and freedom	13. It is the user who initiates and terminates tasks and not the application?				
	14. It is possible to identify the number of steps required to perform a task?				
	15. The applicative can be closed, and no only minimized?				
	16. It is possible to predict the size of the screen through the size of the scrolling bar?				
	17. It is possible to return to the previous screen at any time?				
	18. If the applicative is associated to login register or e-mail accounts, it is easy to add more than one user?				
	19. The user can exit the application and return to the point where s/he stopped?				
	20. The user can cancel an action in progress (e.g., download)?				
	21. The user can undo an action?				
	22. The user can redo an action?				
	23. Applications with overlapping windows: are they transparent so that their context remains visible?				
	24. Relevant information such as text, buttons and controls are visible when the keyboard is open?				
	25. The application indicates clearly what's the next step to accomplish a task?				
	Comments:				
Consistency and standards	26. Screens with the same type of content have the same title?				
	27. Controls and buttons are distinguished from the rest of the layout, making it clear that they are clickable?				
	28. Items not clickable make it evident that they are not?				
	29. The name of the button/icon is consistent with the name of the screen that it opens?				
	30. All textual information from the application uses the same language?				
	31. Different functions are presented differently to the user?				
	32. Similar functions are presented in a similar way?				
	33. Controls that perform the same function are in similar positions on the screen?				
	34. The way of navigation is consistent between screens within the application?				
	35. Links are treated consistently between screens?				
	36. Textual information is presented in a standardized way? (font size, color)				
	Comments:				
Error prevention	*37. During the first user interactions with the application basic instructions are presented?*				
	8. Buttons and controls that trigger irreversible actions are located in areas of difficult access, or defiant gestures require user confirmation (e.g. a slide-to-unlock control used by Android and iOS)?				
	Comments:				
Recognition rather than recall	*39. Uses the name of the previous screen instead of "return" for names of buttons and links?*				
	40. In each menu option, the first word is the most important?				
	41. The titles of the screens are short?				
	42. Data and key messages are in the default position for applications of this platform?				
	43. There is a standardization of colors for the identification and marking of areas of the application?				
	44. The application uses in its texts and labels an understandable language to the user?				
	45. Titles of screens adequately describe their content?				
	46. Link labels adequately describe its contents?				
	47. Input fields make the need for data entry obvious?				
	Comments:				

Dentre as publicações encontradas apenas (INES et al., 2017) e (GRESSE VON WANGENHEIM et al., 2018) apresentam uma ferramenta que faz a análise automaticamente.

PA3. Quais as características instrucionais do/da modelo/ferramenta?

Apenas quatro publicações apresentam a avaliação do design de interface para o contexto educacional (SHERMAN, MARTIN, 2015) (GRESSE VON WANGENHEIM et al., 2018) (TECHNOVATION, 2014) (WAGNER et al., 2013).

Sherman e Martin (2015) têm o interesse de ensinar Pensamento Computacional através de *smartphone* com a ferramenta web App Inventor. A unidade instrucional foi aplicada em uma classe mista de graduandos, ou seja, uma turma com alunos de diferente cursos, além do curso de Ciência da Computação. O material é disponibilizado em inglês.

Gresse von Wangenheim et al. (2018) apresenta a ferramenta web CodeMaster, que foi desenvolvida com o intuito de auxiliar unidades instrucionais da Educação Básica que ensinam conceitos do Pensamento Computacional. A contribuição da ferramenta é em avaliar projetos Snap! (2018) e App Inventor (2018). Essa ferramenta é disponibilizada em português (Brasil) e em inglês.

Technovation (2017) é uma competição para garotas entre 10 e 18 anos com a intenção de ensinar STEM (*science*, *technology*, *engineering* e *math*) e motivar o interesse para nessa área. Com isso, pretende acabar com as diferenças de gênero no mercado de trabalho tecnológico. Em Technovation (2014) é apresentado um guia para mentores e professores sobre lições a serem ensinadas e como avaliar as estudantes. O material é em inglês. A primeira lição propõe o desenvolvimento de aplicativos com App Inventor.

Wagner et al. (2013) tem como contexto um curso aplicado num acampamento de verão de 3 semanas. A unidade instrucional tem como objetivo ensinar introdução a Java; robótica e App Inventor. O App Inventor é usado para iniciar a programação de apps. E o Java Bridge é usado, pois é uma API Java com os elementos do App Inventor.

PA4. Como o modelo/ferramenta foi desenvolvida?

Para que os aplicativos tenham o design de interface avaliado adequadamente, é importante que os critérios definidos sejam resultado de um estudo sistematizado (Tabela 19). Principalmente quando aplicado no contexto de ensino, pois é preciso que o *feedback* esteja alinhado com o conteúdo de ensino e as atividades realizadas durante a unidade instrucional (NICOL, MCFARLANE-DICK, 2006).

Tabela 19: Métodos de desenvolvimento de cada proposta de avaliação

Citação	Método de Desenvolvimento
(WAGNER et al., 2013)	NI
(ZAPATA et al., 2014)	O questionário proposto foi desenvolvido com base em: (1) padrões e recomendações de usabilidade conhecidos (ISO 9241–210) (W3C, 2010); (2) literatura relacionada (BROWN et al., 2013)(SEFFAH et al., 2006) (ABRAN et al., 2003) (NILSSON, 2009) e (3) as diretrizes oficiais de design para Android e iOS (ANDROID, 2018) (iOS, 2018).
(TECHNOVATION, 2014)	NI

(SHERMAN, MARTIN, 2015)	Para desenvolver a rubrica proposta, inicialmente foram analisadas as capacidades técnicas do App Inventor. Em seguida, foi promovido o desenvolvimento de aplicativos no App Inventor por um conjunto de alunos. Com base nos apps resultantes, a abrangência dos critérios de avaliação foram regulados até que correspondesse a maior parte do trabalho dos alunos. Mesmo assim, é fornecidos detalhes suficientes para distinguir projetos de tipos diferentes em vários graus de sofisticação computacional.
(HOEHLE, ALJAFARI, VENKATESH, 2016)	Inicialmente foi o autor revisou as diretrizes da Microsoft e codificou o conteúdo usando a análise linha-a-linha de Strauss e Corbin (1990, p. 119). Em seguida, usando codificação axial, os códigos abertos foram examinados para semelhanças ou diferenças e organizados como unidades conceituais. Depois os resultados foram organizados em uma matriz delineada por Miles e Huberman (1994). Organizar códigos em uma matriz de dados é útil para compactar informações codificadas e suporta conclusões de desenhos (Miles e Huberman, 1994). Posteriormente, um segundo autor revisou as diretrizes de usabilidade e os padrões de codificação associados. Em alguns casos, houve um desentendimento entre os autores, para alcançar um consenso foi realizada uma discussão para finalmente chegar no material proposto.
(GRESSE VON WANGENHEIM et al., 2016)	Foi utilizada a abordagem GQM - *Goal, Question, Metric* (BASILI, CALDIERA, ROMBACH, 1994). O GQM é uma abordagem popular para medir diversos atributos de qualidade de software, incluindo usabilidade (Hussain et al., 2012). GQM suporta a definição sistemática de métricas de uma perspectiva de cima para baixo e a análise e interpretação dos dados de medição de baixo para cima. Seguindo Clark e Watson (1995), o conjunto inicial de itens foi projetado em uma ampla e abrangente caminho. A fim de operacionalizar a medição dos itens, foi elaborada uma lista de verificação que define uma questão para cada um dos itens, seguindo uma abordagem sistemática para o desenho do questionário (Malhotra, 2006). Uma questão foi formulada cuidadosamente para cada um dos itens de medição seguindo Krosnick e Presser (2009).
(INES et al., 2017)	- A lista de métricas de qualidade do design interface mobile foi inspirada em (NGO, TEO, BYRNE, 2000) e baseada em vários critérios ergonômicos proposto por (BASTIEN, SCAPIN, 1993). - A lista de defeitos de design interface mobile foi construída com base na revisão literária feita, em que foi focado em defeitos que estão relacionados na avaliação da complexidade visual da interface. - Para desenvolver a ferramenta automatizada foi utilizada programação genética para gerar regras de avaliação e detectar defeitos das interfaces, levando em conta o contexto de uso e as métricas de qualidade.
(GRESSE VON WANGENHEIM et al., 2018)	- Para desenvolver a rubrica, foi seguido o procedimento proposto por Goodrich (1996). Primeiro foram analisadas as estruturas e rubricas existentes para objetivos típicos de aprendizagem e as estratégias instrucionais para a educação em computação K-12. De acordo com esses modelos existentes e os recursos específicos de ambos os ambientes de programação baseados em blocos (App Inventor e Snap!), foram identificado os critérios de avaliação. - Para o desenvolvimento do CodeMaster foi adotado uma abordagem de desenvolvimento de software iterativo e incremental (LARMAN, BASILI, 2003).
(ANDROID, 2018)	Os critérios foram definidos com base nas diretrizes de Qualidade do aplicativo para tablet (2018) e as Diretrizes de educação (2018).

PA5. Como a qualidade do modelo/ferramenta foi avaliada?

Essencial parte do processo do desenvolvimento de critérios para verificar a qualidade do design de interface de aplicativos, é a validação da *checklist*/questionário/rubrica. Na tabela 20 é apresentado como a publicação avaliou a *checklist*/questionário/rubrica/ferramenta proposta.

Tabela 20: Métodos de Avaliação da qualidade da *checklist*/questionário/rubrica/ferramenta.

Citação	**Método de Avaliação da Qualidade**
(WAGNER et al., 2013)	NI
(ZAPATA et al., 2014)	Foram feitas duas avaliações, uma verifica a aceitação das questões do questionário (Figura 13) proposto e a segunda para verificar a concordância das avaliações dos aplicativos entre dois autores. Na primeira avaliação, o questionário foi respondido por 22 pessoas, que responderam as questões selecionando números da "five-score Likert-type" , na escala de 1 a 5. Com isso foi avaliado o nível de **aceitação** das questões, cuja média ficou em 71.24 % (mean: 3.85). Na outra avaliação foi feita com dois autores do artigo. Cada um avaliou todos os 24 aplicativos selecionados de propósito PHR através do questionário. Cada pergunta foi respondida com pontuações, 1, ½ ou 0, de acordo com a porcentagem do quanto a pergunta é aplicável ao aplicativo. Com os resultados, foi analisada a **concordância** entre os autores com o coeficiente de Cohen's Kappa, que resultou em 0.97, indicando um nível um nível de quase perfeição de concordância.
(TECHNOVATION, 2014)	NI
(SHERMAN, MARTIN, 2015)	Com base na rubrica (SHERMAN et al., 2014), foram avaliados 45 apps independentemente pelos autores. Depois foram feitas comparações entre os resultados dos autores, os casos de que discordâncias foram resolvidas através de discussões feitas entre os autores.

(HOEHLE, ALJAFARI, VENKATESH, 2016)	Foi realizado um estudo piloto seguido de uma avaliação quantitativa da validade de conteúdo das escalas. Em seguida, foram feitas uma análise fatorial exploratória e a análise fatorial confirmatória em duas amostras (n = 404; n = 501) consistindo de consumidores alemães que usam aplicativos de mídia social móvel em seus smartphones. Para avaliar o modelo fatorial confirmatório, foi seguido um processo passo a passo avaliando unidimensionalidade, validade discriminante e confiabilidade. Para avaliar a validade nomológica da ferramenta, foi examinado o impacto da usabilidade de aplicativos móveis em dois resultados: **a intenção contínua de usar e a fidelidade à marca**. Os resultados confirmaram que a usabilidade de aplicativos móveis foi um bom preditor de ambos os resultados.
(GRESSE VON WANGENHEIM et al., 2016)	A *checklist* foi validada com base em um estudo empírico de 247 avaliações heurísticas. Os resultados foram analisados estatisticamente usando a Teoria da Resposta ao Item[10] (LORD, 1980). As conclusões causaram calibrações nos itens de medição e uma escala de medição padronizada foi construída.
(INES et al., 2017)	A avaliação teve como objetivo avaliar a **eficiência** da proposta para detecção de defeitos. Para isso foram avaliados 4 aplicativos (Duoling[11], Accuweather[12], Easy-loan-calculator[13] e Handicraft Women[14]) de duas maneira. A primeira maneira foi feita uma análise dos apps manualmente por 20 pessoas através de um questionário. Em seguida os 4 apps foram avaliados pela ferramenta proposta. Com os resultados das duas avaliações, comparações foram feitas através de duas equações. A primeira é calcula a **precisão**, que denota a fração de defeitos detectados corretamente entre o conjunto de todos os defeitos detectados. A segunda equação indica ***recall***, que é a fração de defeitos detectados corretamente entre o conjunto de todos os manualmente identificados pela base de defeitos. Os defeitos esperados (defeitos relatados no rastreamento) foram detectados com uma média de mais de 70% de precisão nos quatro aplicativos móveis testados.
(GRESSE VON WANGENHEIM et al., 2018)	Foram analisados a corretude e a qualidade da ferramenta CodeMaster em termos de utilidade, adequação funcional, eficiência de desempenho e usabilidade do ponto de vista de professores e alunos do ensino fundamental e médio no contexto da educação em informática. Baseado na ISO / IEC 25010 (2011), ISO / IEC 9241 (1998) TAM (Davis, 1989) e SUS (Brooke, 1996). Para testar a corretude, foram analisados 10 apps aleatórios da Galeria do App Inventor e comparados os resultados gerados pela ferramenta CodeMaster com os resultados da avaliação manual. Para testar as qualidades da ferramenta CodeMaster, 7 professores e 9 estudantes realizam uma tarefa pré-definida no CodeMaster e depois responderam um questionário.
(ANDROID, 2018)	NI

3.4 Discussão

O design de interface de aplicativos para *smartphones* precisa ser fácil, agradável de usar e exija pouco esforço para os usuários (ROGERS; SHARP; PREECE, 2013). Principalmente porque a qualidade da interface interfere diretamente na intenção dos usuários de continuar utilizando o app, como fortalece a fidelidade dos clientes à marca (HOEHLE, ALJAFARI, VENKATESH, 2016).

Para atingir um bom nível de qualidade o design de interface é muito importante avaliar a interface gráfica para guiar no desenvolvimento do aplicativo (GRESSE VON WANGENHEIM et al., 2016). Além de reduzir o custo de manutenção do app (INES et al., 2017).

Notou-se que 4 de 9 artefatos apresentam como meio de avaliação da qualidade do design de interface questionários ou *checklists*. Esses métodos são aplicados por um conjunto de pessoas do público alvo ou especialistas, sem um processo manual que consome bastante tempo e é propensa a erros, tornando em uma tarefa difícil (INES et al., 2017).

[10] Teoria da Resposta ao Item, em inglês, Item Response Theory (IRT).
[11] https://github.com/KartikTalwar/Duolingo
[12] https://github.com/AccuWeather
[13] https://play.google.com/store/apps/details?id=appinventor.ai_Newbebiko.Simulation
[14] https://github.com/mabroukachouchane/HandcraftWomen/blob/master/FemmeArtisan.rar

O método de avaliação por rubricas foi proposta por três artefatos analisados. Todas as rubricas encontradas são indicadas para o contexto educacional. Considerando a circunstância de unidades instrucionais com atividades *ill-structured* de desenvolvimento de aplicativos, rubrica é um ótimo meio de apresentar os critérios estabelecidos para avaliação. Mas o processo de realizar a análise manualmente torna-se árduo para os avaliadores, ainda mais se o avaliador não for especialista na área.

Assim é identificada a necessidade de automatizar a avaliação da qualidade do design de interface (Zapata et al., 2015). Dos 9 artefatos analisados apenas 2 propõem uma ferramenta automatizada (INES et al., 2017) (GRESSE VON WANGENHEIM et al., 2018), sendo que apenas Gresse von Wangenheim et al. (2018) é aplicado para o contexto educacional.

Ameaças à Validade da Revisão da Literatura

Como em qualquer mapeamento sistemático, existem algumas ameaças à validade dos resultados. Foram identificadas ameaças potenciais e aplicadas estratégias de mitigação para minimizar os impactos:

- **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, consideramos que os resultados dos artigos, sejam positivos ou negativos, têm apenas uma pequena influência sobre esse mapeamento sistemático, uma vez que foi buscado caracterizar os meios de avaliação da qualidade do design de interface de aplicativos.
- **Identificação de estudos**: Outro risco é a omissão de estudos relevantes. A fim de mitigar esse risco, foi construída cuidadosamente a *string* de busca para ser o mais abrangente possível, considerando não apenas os principais conceitos, mas também sinônimos.
- **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 realizaram a seleção juntos, discutindo a seleção até que o consenso fosse alcançado.

4. Desenvolvimento do modelo de avaliação de design de interface de apps

Com o objetivo de facilitar e reduzir o esforço necessário por parte dos instrutores para avaliar apps desenvolvidos pelos alunos no contexto do ensino de computação, este trabalho apresenta a rubrica “CodeMaster UI Design - App Inventor”. Essa rubrica é um meio de avaliar do design de interface. Este trabalho também apresenta um módulo que avalia automaticamente a rubrica “CodeMaster UI Design - App Inventor”, que foi incluída no CodeMaster.

4.1 Análise do contexto

Existem diversas unidades instrucionais para o ensino de computação na Educação Básica (FERREIRA et al, 2018). Como por exemplo, unidades instrucionais sendo desenvolvidas pela iniciativa Computação na Escola (MISSFELDT, 2017) (PINHEIRO, 2019) (FERREIRA, 2019). Essas unidades focam no ensino de conceitos de algoritmos e programação em conformidade ao guia de currículo CSTA (2017), englobando também competências de design de interface de aplicativos móveis.

A unidade instrucional “faça o seu app” apresentada por Missfeldt(2018) tem como objetivo ensinar computação por meio de programação de apps com App Inventor. No contexto de uma aprendizagem baseada em problemas *ill-structured*, ou seja, em problemas que se possuem uma solução, geralmente não existe uma única solução correta (Jonassen, 1997).

O público-alvo da unidade instrucional “faça o seu app” são alunos do Ensino Fundamental II das escolas brasileiras com idade entre 10 a 15 anos. Comumente, os alunos neste nível escolar são alfabetizados. Segundo Paiva (2018), 95% dos alunos entre 10 e 12 anos têm acesso a *smartphones* e essa proporção cresce conforme a idade. Sendo que nessa faixa etária 72% têm um smartphone próprio, enquanto 23% seguem usando aquele dos pais (PAIVA, 2018). E tipicamente não possuem nenhum conhecimento de conceitos de programação e computação em geral.

Como parte essencial do desenvolvimento de um app, a unidade instrucional “faça o seu app” também visa o ensino de competências de design de interface, que é indicado pelo objetivo de aprendizagem 6 (Tabela 21). E praticado por meio de uma atividade instrucional com problemas *ill-structured* desenvolvendo o seu próprio aplicativo com App Inventor.

O objetivo de aprendizagem 6 (Tabela 21) é lecionado na parte “2.5 - Criação e teste do design visual no App Inventor” do plano de ensino (Tabela 22). A finalidade dessa parte é de ensinar os aspectos sobre design visual de aplicativo, como tipografia, cores, menus, imagens e ícones. Para que os alunos sejam capacitados para aprimorar telas criadas em etapas anteriores do plano de ensino de acordo com as guidelines ensinadas.

Tabela 21: Objetivos de aprendizagem da UI (MISSFELDT FILHO, 2018)

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, 2017: 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, 2017: 2-AP-18)
OA4	Identificar e resolver problemas criando sistemas de software interativos	Algoritmo e programação / Engenharia de Usabilidade	(CSTA, 2017: 1B-CS-03, 3A-AP-13), (AIGA, 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, 2017: 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, 2017: 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, 2017: 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, 2017: 2-CS-02), (ACM/IEEE, 2013), (GARRET, 2011)
OA9	Modelar processos criando e seguindo algoritmos / mapas de navegação para concluir tarefas.	Algoritimo e programação	(CSTA, 2017: 1A-AP-08)
OA10	Usar fluxogramas, pseudocódigo e / ou mapas de navegação para resolver problemas complexos.	Algoritimo e programação	(CSTA, 2017: 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, 2017: 2-IC-21), (GARRET, 2011)*
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, 2017: 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.	Algoritmos and Programação/ Engenharia de Software/Engenharia de Usabilidade	(CSTA, 2017: 2-AP-15)
OA14	*Testar e refinar um sistema de software interativo para funcionalidade e usabilidade.*	*Engenharia de Software*	*(CSTA, 2017: 2-AP-17, 1B-AP-15, 3A-AP-21)*
OA15	*Recomendar melhorias no design de dispositivos de computação, com base nos resultados de verificação e validação.*	*Engenharia de Software*	*(CSTA, 2017: 2-CS-01)*
OA16	*Compartilhar o sistema de software interativo desenvolvido.*	*Algoritmo e programação*	*(CSTA, 2017: 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, 2017: 1B-AP-12)*
OA18	*Descrever escolhas tomadas durante o desenvolvimento do programa usando comentários, apresentações e demonstrações.*	*Impactos da computação*	*(CSTA, 2017: 1B-AP-17)*

Tabela 22: Plano de ensino da unidade instrucional (MISSFELDT FILHO, 2018)

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, o 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

Como resultado dessa atividade são desenvolvidos aplicativos como "HealthyPlants"[15], desenvolvido por alunas do 9º ano da Escola Básica Almirante Carvalhal e participantes do projeto Jovens Tutores de Programação 2018 (Figura 23).

[15] Mais informações em http://www.computacaonaescola.ufsc.br/?p=2854

Figura 23: Telas do app HealthyPlants

Considerando que no processo de ensino, os alunos precisam de *feedback* constantemente para saberem como está o seu desempenho. Fornecer esse *feedback* aos alunos referente a atividades *ill-structured* requer competência e dedicação dos professores para desenvolver o *feedback* para cada aluno de forma consistente, com exatidão e evitando bias (ZEN et al., 2011).

Porém a atual falta de professores na Educação Básica formados na área de computação (TIC Educação, 2016), é um fato que mobiliza professores de outras áreas do conhecimento a aplicar de forma interdisciplinar a computação. Nessa situação, os professores de outras áreas do conhecimento possuem maior dificuldade em realizar o processo de avaliação e *feedback* aos alunos. Além disso, quanto maior a turma, mais tempo requer para realizar um *feedback* consistente para todos os alunos. Tornando complexa a necessidade de dar *feedback* de forma instantânea durante o processo de aprendizagem.

Com base nesse cenário, notou-se a necessidade de alunos e professores terem suporte para avaliar a aprendizagem referente ao design visual de forma (semi-) automatizada.

4.2 Desenvolvimento da rubrica para avaliação do design visual

Para que o processo de ensino e aprendizagem seja bem sucedido é importante realizar avaliação e *feedback* instrucional (HATTIE; TIMPERLEY, 2007). Assim ao desenvolver um aplicativo com App Inventor no contexto educacional formal, conforme apresentado pela unidade instrucional Missfeldt (2019), é preciso avaliar o desempenho sobre design visual do app feito.

A avaliação do desempenho então é sistematicamente definia por meio da rubrica "CodeMaster UI Design - App Inventor". Essa rubrica foi desenvolvida em parceria com o bolsista de iniciação científica João Vitor Araújo Porto e o mestrando Igor da Silva Solecki (SOLECKI, 2019) da iniciativa Computação na Escola do GQS/InCod/INE/UFSC. A rubrica apresenta critérios objetivos para a avaliação do

design visual criado pelo aluno. Os critérios foram definidos a partir de princípios de design visual (SCHLATTER; LEVINSON, 2013)(*Material Design*, 2018)(WCAG 2.0, 2008)(W3C, 2018).

Um fator considerado para a inclusão de uma diretriz como critério na rubrica “CodeMaster UI Design - App Inventor” é a viabilidade de ser avaliado com os recursos oferecidos pelo App Inventor. Um caso de restringimento ocorreu com critérios que analisavam recursos que o App Inventor não oferece, como por exemplo, incluir no app botões flutuantes. Outro restringimento ocorreu com critérios que analisavam informações semânticas não fornecidas pelo App Inventor. Por exemplo, é possível alterar o tamanho da fonte do texto do componente “Legenda” e dispor no app de forma a criar legenda, corpo de texto, subtítulo 1, subtítulo 2, título, etc. Porém isso não é identificável através dos dados fornecidos pelo arquivo .aia, inviabilizando a análise automática dos tamanhos de fonte sugerido pelo Material Design para cada categoria de texto (legenda, corpo de texto, subtítulo 1, subtítulo 2, título, etc).

De acordo com os princípios e tipos de elementos de design de interface, a rubrica "CodeMaster UI Design - App Inventor” tem seus critérios organizados em categorias de:

- *Layout*: critérios que avalia como componentes são organizados e disposto na tela.
- Tipografia: compõem os critérios que define como os elementos textuais são apresentados.
- Escrita: engloba critérios que apresentam regra sobre como os elementos textuais precisam ser escritos.
- Cores: consiste em critérios que analisam as cores utilizadas no app.
- Imagens: são critérios que avaliam a qualidade de imagens e ícones.

Cada critério é avaliado por níveis de desempenho variando de 2 a 4 níveis em uma escala ordinal. O nível de desempenho 0 representa o pior desempenho, enquanto o maior nível representa o melhor desempenho. A rubrica "CodeMaster UI Design - App Inventor” é apresentada na Tabela 23.

Tabela 23: Rubrica CodeMaster UI Design - App Inventor

ID	Critério	Nível de performance	
		0	1
Layout			
L1	Dimensionamento responsivo	Valor da propriedade dimensionamento é “fixo”.	Valor da propriedade dimensio “responsivo”.
L2	Elementos de organização nas telas	Porcentagem de telas em todo o aplicativo que usam ao menos um dos componentes de organização pertence ao intervalo [0, 90).	Porcentagem de telas em todo o aplica menos um dos componentes de organi intervalo [90, 100].
L3	Tamanho mínimo de componentes alvos de toque	Há algum componente alvo de toque com largura ou altura menor que 48 pixels	Todos os componentes alvos de toqu altura de no mínimo 48 p

L4	Formato dos botões	Existem botões sem imagem com formatos diferentes	Todos os botões sem imagem do a mesmo formato
L5	Tamanho consistente de botões	Há botões dentro de um mesmo organizador ou diretamente na tela com altura ou largura definida de maneiras diferentes, ou com largura Automatic, ou com altura Automatic e tamanhos de fonte diferentes	Botões dentro do mesmo organizado na tela sempre têm altura e largur mesma forma, e a largura não é Au altura dos botões é Automatic, todo tamanho de fonte
L6	Densidade de elementos nas telas	Há alguma tela com 20 ou mais elementos ou com menos do que 2 elementos	Há alguma tela com entre 10 e 19 nenhuma com menos de 2 el
		Tipografia	
T1	Família da fonte	Porcentagem de componentes em todo o aplicativo que têm propriedade FamíliaDaFonte cujo valor é “sem serifa” ou “padrão” pertence ao intervalo [0, 90).	Porcentagem de componentes em t que têm propriedade FamíliaDaFonte serifa” ou “padrão” pertence ao inte
T2	Tamanho da fonte de texto de botões	Porcentagem, entre todos os botões (incluindo EscolheData etc.) com texto do aplicativo, que tem a propriedade TamanhoDaFonte igual a 14 pertence ao intervalo [0, 90).	Porcentagem, entre todos os botõ EscolheData etc.) com texto do aplic propriedade TamanhoDaFonte igual intervalo [90, 100].
T3	Tamanho da fonte e Tamanho do texto de componentes CheckBox, Label, ListView, CaixaDeSenha, CaixaDeTexto, Pintura e EscolheEmail	Porcentagem, entre todos os componentes (exceto botões) com propriedade Texto ou CadeiaDeElementos preenchida do aplicativo, que tem propriedade TamanhoDaFonte ou TamanhoDoTexto igual a 10, 12, 14, 16, 20, 24, 34, 48, 60 ou 96 pertence ao intervalo [0, 90).	Porcentagem, entre todos os compo botões) com propriedade Texto ou Ca preenchida do aplicativo, que tem TamanhoDaFonte ou TamanhoDoTex 14, 16, 20, 24, 34, 48, 60 ou 96 perte [90, 100].
T4	Não usar itálico	Há texto em itálico	Não há texto em itálic
T5	Botão Com Texto Centralizado	pelo menos um botão com texto não é centralizado	todos os botões com textos são c
		Escrita	
E1	Capitalização do texto de botões	Porcentagem de botões em todo o aplicativo que contêm texto com todas as letras em maiúsculo pertence ao intervalo [0, 90).	Porcentagem de botões em todo o contêm texto com todas as letras pertence ao intervalo [90,
E2	Capitalização de sentenças	Porcentagem de sentenças em todo o aplicativo que iniciam com letra maiúscula ou dígito pertence ao intervalo [0, 90).	Porcentagem de sentenças em todo iniciam com letra maiúscula ou dígi intervalo [90, 100].
E3	Componentes com propriedade Texto diferente do padrão, exceto os componentes EscolheEmail, CaixaDeTexto e CaixaDeSenha.	Porcentagem de componentes em todo o aplicativo que contêm texto diferente do padrão pertence ao intervalo [0, 90).	Porcentagem de componentes em t que contêm texto diferente do padr intervalo [90, 100].
E4	Componentes com propriedade Dica preenchida e diferente de default	Porcentagem de componentes com propriedade Dica preenchida e diferente de default pertence ao intervalo [0, 90).	Porcentagem de componentes com p preenchida e diferente de default per [90, 100].

E5	Não usa dois pontos ao final de legendas	Porcentagem de legendas em todo o aplicativo que não encerram com dois pontos pertence ao intervalo [0, 90).	Porcentagem de legendas em todo o a encerram com dois pontos pertence 100].
E6	Não usa ponto final para encerrar sentenças	Porcentagem de sentenças em todo o aplicativo que não encerram com ponto final pertence ao intervalo [0, 90).	Porcentagem de sentenças em todo não encerram com ponto final perte [90, 100].
E7	Quantidade de caracteres de texto do botão	O maior texto do botão tem mais de 14 caracteres	O maior texto do botão tem entre 8 [8,14] caracteres
E8	Evitar múltiplos pontos de exclamação/interrogação	Há algum texto com uma sequência de 2 ou mais pontos de exclamação e/ou interrogação, com ou sem espaços	Não há nenhum texto com uma sequê pontos de exclamação e/ou interroga espaços
	Cores		
C1	Sistema de cores	Utiliza no aplicativo nenhuma cor além das cores básicas (preto, branco e cinza). OU utiliza no aplicativo 4 ou mais cores além das cores básicas (preto, branco e cinza), podendo utilizar-se de outras variações tonais das cores escolhidas.	Utiliza no aplicativo 3 cores além da (preto, branco e cinza), podendo util variações tonais das cores es
C2	Contraste entre texto e fundo	O menor contraste entre texto e cor de fundo não satisfaz os requisitos das WCAG 2.0	O menor contraste entre texto e cor os requisitos do Nível AA das V
C3	Cores do Material Design	alguma cor das cores usadas não são “das cores propostas pelo MD ” ou apenas usa branco, preto e cinza	todas as cores usadas são “das cores MD ”
C4	Harmonia das cores	Há tonalidades de cores que não são harmônicas (complementares, análogas ou triádicas, seguindo o modelo RGB)	Apenas uma tonalidade de cor é usad tonalidades harmônicas (complement triádicas, seguindo o model
	Imagens		
I1	Ícones recomendados pelo Material Design	Porcentagem de ícones em todo o aplicativo que são semelhantes a algum ícone do Material Design pertence ao intervalo [0, 90).	Porcentagem de ícones em todo o ap semelhantes a algum ícone do Materia ao intervalo [90, 100]
I2	Pixelização de imagens	Porcentagem de imagens em todo o aplicativo que não apresentam pixelização (largura ou altura ampliada em 50% ou mais) pertence ao intervalo [0, 90).	Porcentagem de imagens em todo o a apresentam pixelização (largura ou al 50% ou mais) pertence ao interva
I3	Distorção de imagens em botões e pickers	Porcentagem de componentes Botão com imagem que têm largura e altura definidas da mesma forma, as duas usando Automático, as duas em Pixels ou as duas em Porcentagem, e que não apresentam	Porcentagem de componentes Botão têm largura e altura definidas da mesm usando Automático, as duas em Pixe Porcentagem, e que não apresenta

		distorção da imagem pertence ao intervalo [0, 90).	imagem pertence ao intervalo
I4	Não faz redimensionamento do componente Imagem	Porcentagem de componentes imagem em todo o aplicativo que têm a propriedade RedimensionarParaCaber não selecionada pertence ao intervalo [0, 90).	Porcentagem de componentes ima aplicativo que têm a propri RedimensionarParaCaber não selecio intervalo [90, 100].
I5	Imagem de fundo da tela	Há alguma imagem de fundo de tela distorcida	Não há nenhuma imagem de fundo d

Tipicamente os apps não apresentam todos os componentes avaliados na rubrica. Por exemplo, o critério “Tamanho da fonte e Tamanho do texto dos componentes CaixaDeSenha, CaixaDeTexto, Pintura e EscolheEmail” analisa o tamanho da fonte dos componentes:

- CaixaDeSenha
- CaixaDeTexto
- Pintura
- EscolheEmail

Se o app não tiver um desses componentes o critério não tem o que avaliar. Por isso foi criado o estado “não se aplica”, para sinalizar que o app não a tem o que é avaliado no critério. Consequentemente, o critério é desconsiderado do *feedback* instrucional.

4.3 *Feedback* Instrucional

O *feedback* instrucional tipicamente é representado por uma nota ou conceito. Por isso a avaliação feita com a rubrica "CodeMaster UI Design - App Inventor” (Capítulo 4.2) resulta em um *feedback* instrucional em forma de nota.

A atribuição de nota pode se referir a um valor quantitativo ou qualitativo. No Brasil não existe definido um esquema de notas para ser aplicado de forma padronizada em nível nacional (Alves, 2018). Então, para representar a nota resultante da avaliação com a rubrica é utilizada a representação quantitativa com valor entre 0 e 10, sendo 0 a pior nota e 10, a melhor. Pois é o padrão comum nas escolas regionais.

O processo de geração da nota de uma rubrica é baseado na Fórmula 1. A “*pontuação_maxima*” consiste na somatória dos maiores níveis de desempenho possíveis de atingir em cada critério. A “*pontuação_alcançada*” consiste na somatória dos níveis de desempenho atingido pelo app.

Fórmula 1: $$Nota = \frac{pontuacao\_alcancada}{pontuacao\_maxima}.10$$

Na rubrica "CodeMaster UI Design - App Inventor" os critérios podem entrar no estado "não se aplica", quando não possuem o componente avaliado pelo critério. Nesse caso, a "*pontuação_maxima*" e "*pontuação_alcançada*" não contabilizam esse critério. A Tabela 24 apresenta a avaliação do aplicativo "HealthyPlants" (Figura 23) realizada com a rubrica "CodeMaster UI Design - App Inventor".

A maior pontuação total possível de alcançar é de 32 pontos, se o aplicativo conter todos os componentes avaliados. O aplicativo "HealthyPlants" não tem todos os componentes avaliados (Figura 23) . Como no caso do critério "Componentes com propriedade Dica preenchida e diferente de default" que resultou em -1 (não se aplica) porque não tem componentes com propriedade Dica. Então conforme apresentado na tabela 24, a pontuação máxima possível de alcançar no contexto do aplicativo é 27 pontos. O aplicativo conseguir alcançar 20 pontos. Então conforme a fórmula sugere a nota resultante é 7,78.

Exemplo: $$Nota = \frac{pontuacao\_alcancada}{pontuacao\_maxima}.10 = \frac{21}{27}.10 = 7,78$$

Tabela 24: Exemplificação de atribuição de nota pela rubrica "CodeMaster UI Design - App Inventor"

ID	Critério	Pontuação recebida	Pontuação máxima
	Layout		
L1	Dimensionamento responsivo	0	1
L2	Elementos de organização nas telas	0	1
L3	Tamanho mínimo de componentes alvos de toque	1	1
L4	Formato dos botões	1	1
L5	Tamanho consistente de botões	1	1
L6	Densidade de elementos nas telas	2	2
	Tipografia		
T1	Família da fonte	1	1
T2	Tamanho da fonte de texto de botões	1	1
T3	Tamanho da fonte e Tamanho do texto de componentes CheckBox, Label, ListView, CaixaDeSenha, CaixaDeTexto, Pintura e EscolheEmail	0	1
T4	Não usar itálico	1	1
T5	Botão Com Texto Centralizado	1	1
	Escrita		
E1	Capitalização do texto de botões	-1	1
E2	Capitalização de sentenças, incluindo de botões	1	1

E3	Componentes com propriedade Texto diferente do padrão, exceto os componentes EscolheEmail, CaixaDeTexto e CaixaDeSenha.	1	1
E4	Componentes com propriedade Dica preenchida e diferente de default	-1	1
E5	Não usa dois pontos ao final de legendas	0	1
E6	Não usa ponto final para encerrar sentenças	1	1
E7	Quantidade de caracteres de texto do botão	2	2
E8	Evitar múltiplos pontos de exclamação/interrogação	1	1
	Cores		
C1	Sistema de cores	2	2
C2	Contraste entre texto e fundo	0	2
C3	Cores do Material Design	1	1
C4	Harmonia das cores	1	1
	Imagens		
I1	Ícones recomendados pelo Material Design	-1	1
I2	Pixelização de imagens	1	1
I3	Distorção de imagens em botões e pickers	-1	1
I4	Não faz redimensionamento do componente Imagem	1	1
I5	Imagem de fundo da tela	-1	1
	Pontuação Total	**21**	**27**

5. Automação da avaliação integrada ao CodeMaster

Um módulo de automatização da avaliação do design de interface de aplicativos Android foi desenvolvido com base na rubrica “CodeMaster UI Design - App Inventor" (capítulo 4.2). O processo de automatização da rubrica “CodeMaster UI Design - App Inventor" tem como objetivo reduzir o esforço do processo de avaliação para professores da Educação Básica, possibilitar maior exatidão na análise e evitar bias.

Esse módulo é integrado ao CodeMaster (DEMETRIO, 2017)(GRESSE VON WANGENHEIM et al., 2018)(ALVES, 2019), uma ferramenta web que analisa automaticamente projetos App Inventor de programação dos alunos e apresenta um *feedback* instrucional imediato sobre a variedade e complexidade da programação realizada.

5.1 Análise de requisitos

A versão da ferramenta CodeMaster apresentada neste trabalho mantém todos os requisitos originalmente definidos (ALVES, 2019) (DEMETRIO, 2017) (GRESSE VON WANGENHEIM et al., 2018). Os novos requisitos referenciam a análise automatizada da rubrica de design visual de interface (RF01, RF02 e RF03) e a inclusão de um novo usuário, o administrador.

O administrador é um usuário que possui acesso aos dados resultantes dos projetos analisados (App Inventor e/ou Snap!) pela ferramenta CodeMaster. Esse usuário surgiu da necessidade de permitir que interessados analisem os dados, com a facilidade e praticidade. Outro recurso disponível aos administradores é de filtrar os dados de interesse, conforme:

- Tipos de projeto: podendo ser App Inventor, Snap! ou ambos;
- Período de submissão;
- Quem enviou: por professores, alunos ou ambos;
- Nome da instituição do professor: especificado caso o interesse seja analisar projetos de professores de uma instituição específica;
- Nome do professor: especificado caso haja interesse em um professor em específico;

Requisitos funcionais (Tabela 25) e não funcionais (Tabela 26) foram identificados conforme a necessidade de automatizar a avaliação da rubrica "CodeMaster UI Design - App Inventor” e a necessidade de incluir as funcionalidades do novo usuário, o administrador.

Tabela 25: Requisitos Funcionais

ID	Requisito	Descrição	Artefato Entrada	Artefato Saída
RF01	Avaliar o design visual do Projeto App Inventor	A ferramenta deve avaliar cada critério (tabela 21) e apresentar o nível de desempenho alcançado em	arquivo .aia	Nível de desempenho alcançado em cada critério.

		cada critério		
RF02	Apresentar nota alcançada na rubrica	A ferramenta deve gerar uma nota para o desempenho alcançado na rubrica.	arquivo .aia	Nota de 0 a 10, representando o desempenho na rubrica.
RF03	Apresentar nota final	A ferramenta deve gerar uma nota final, apresentando o desempenho alcançado nas duas rubricas. Também é definida a cor da faixa do ninja.	arquivo .aia	Nota final de 0 a 10, representando o desempenho nas duas rubricas.
RF04	Realizar login de Administrador	A ferramenta deve permitir que o administrador autentique sua identidade por meio de login.	E-mail e senha.	Redirecionado para página principal do administrador ou interface com usuário exibe mensagem de erro caso não consiga autenticar.
RF05	Realizar logout de administrador	A ferramenta deve permitir que o administrador encerre a sessão atual de autenticação de sua identidade por meio de logout.	Login do administrador na sessão atual.	Encerramento da sessão atual e redirecionamento para pagina inicial da ferramenta.
RF06	Registrar novos administradores	A ferramenta deve permitir que qualquer administrador adicione outro administrador	Login do administrador	Sistema armazena em seu banco de dados o novo administrador
RF07	Editar administradores	Edite os dados de um admin	Login do administrador	Sistema edita em seu banco de dados o novo administrador
RF08	Deletar administradores	Deletar um admin	Login do administrador	Sistema exclui em seu banco de dados o novo administrador
RF09	Extrair dados do Banco de Dados	A ferramenta deve gerar uma planilha com os níveis de desempenho alcançados pelos aplicativos analisados	Informações de filtros para selecionar apps de interesse	Planilha

Tabela 26: Requisitos Não Funcionais

ID	**Requisito**	**Descrição**
RNF01	Transmitir informações de duas rubricas	Adaptar a comunicação do módulo “Análise e Avaliação” com o módulo “Apresentação” para que sejam transmitidas informações de duas rubricas, a rubrica “CodeMaster CT - App Inventor” e a rubrica proposta neste trabalho "CodeMaster UI Design - App Inventor”.
RNF02	Linguagem de	A ferramenta deve ser desenvolvida na linguagem de

	Programação Java	programação Java, pois é uma linguagem muito conhecida facilitando a manutenção do sistema por profissionais qualificados. E mantém homogeneidade com o sistema já existente
RNF03	Internacionalização	O sistema deve conter a opção de textos em português e inglês.
RNF04	A planilha deve ser Excel (.xlsx)	O administrador tem a funcionalidade de gerar uma planilha com os dados solicitados. Essa planilha tem de ser do formato .xlsx

5.2 Definição do Caso de Uso

Conforme o sistema existente do CodeMaster e as aprimorações apresentadas nos requisitos, foram elaborados os casos de uso conforme Wazlawick (2016). Existem elementos computacionais e usuários que interagem com o sistema. Esses elementos são os atores do sistema (Figura 23):

- Aluno – Envia atividade para avaliação, vê resultado da avaliação.
- Professor – Envia conjunto de projetos para avaliação, vê resultado das avaliações por turma.
- Administrador – Acessa as estatísticas de todas as avaliações presentes no banco de dados.

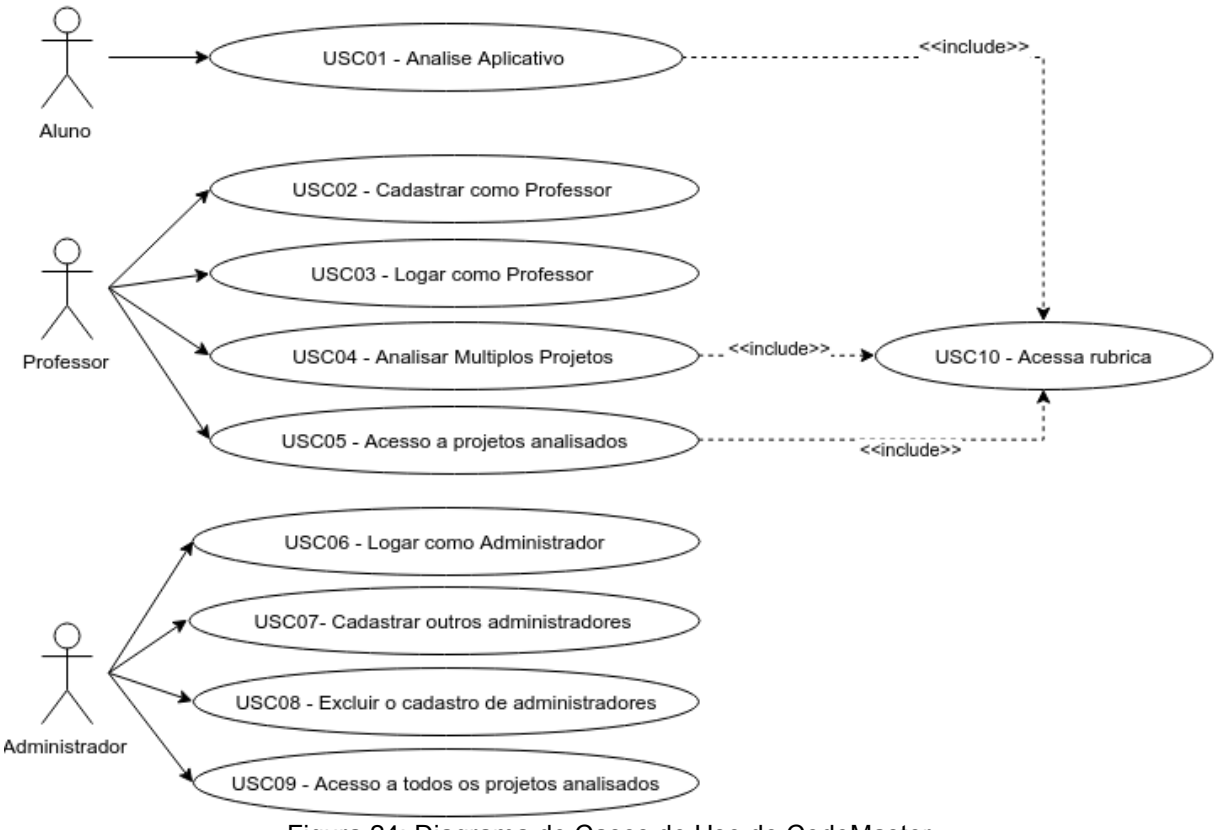


Figura 24: Diagrama de Casos de Uso do CodeMaster

Caso de Uso 1 – USC01 Análise Aplicativo
Ator: Aluno
Fluxo principal:

1. Aluno acessa o site do CodeMaster.
2. Aluno clica no item de menu “Aluno”.
3. Aluno clica em “Escolher arquivo” e escolhe projeto salvo em seu computador.
4. Aluno clica em “Avaliar” e sistema retorna o resultado da análise do projeto na aba da rubrica de programação.
5. Aluno clica em “Interface de Usuário” para visualizar o resultado da análise do projeto no contexto de design

Fluxo de exceção:

FE01: Sistema não consegue analisar o projeto enviado e uma página informando que um “Erro Ocorreu” com a opção de voltar para tela inicial é exibida.
FE02: Aluno clica em avaliar sem escolher projeto. Sistema informa uma mensagem: “Nenhum projeto foi selecionado para análise! ”.

Caso de Uso 2 – USC02 Cadastrar como Professor
Ator: Professor
Fluxo Principal:

1. Professor acessa site do CodeMaster
2. Professor clica no botão “Professor”, que fica no menu superior.
3. Professor clica na opção “Ainda não sou cadastrado” abaixo dos campos de login e senha.
4. Professor informa, nome, e-mail, instituição de ensino, disciplina, cidade, estado, país, senha e clica em “Cadastrar”.
5. Professor recebe e-mail para validar conta.
6. Professor abre link recebido no e-mail. O sistema abre a página de login de professor com uma mensagem de conta confirmada exibida.

Fluxo de Exceção:

4a. Professor não preenche um dos campos de texto e o sistema apresenta a mensagem “Todos os campos devem ser preenchidos”.
5a. E-mail informado já existe cadastrado, sistema apresenta a mensagem “Já existe uma conta com este e-mail, verifique o e-mail informado! ”

Caso de Uso 3 – USC03 Logar como Professor
Ator: Professor
Pré-condição: USC02
Fluxo Principal:

1. Professor acessa o site do CodeMaster.
2. Professor clica no botão “Professor” no menu superior da tela e informa seu e-mail e senha nos campos de login e senha e clica em “Login”.
3. O sistema autentica o professor e redireciona à página principal de submissão de múltiplos projetos.

Fluxo de Exceção:

FE02 - O e-mail e senha do professor não são identificados. O sistema informa uma mensagem: “Usuário e senha não identificados pelo sistema”.

Caso de Uso 4 – USC04 Análise de Múltiplos Projetos
Ator: Professor
Pré-condição: USC03
Fluxo Principal:

1. Professor digita o nome da turma, seleciona o tipo de projeto que deseja avaliar e clica em “Próximo”.
2. Professor seleciona os conceitos que deseja avaliar da rubrica de programação.
3. Professor seleciona os conceitos que deseja avaliar da rubrica de design.
4. Professor escolhe múltiplos arquivos de seu computador que correspondem a uma turma. Professor clica em “Avaliar”.
5. Professor recebe o resultado da análise de todos os projetos organizados em 3 tabelas. Uma aba com os notas finais, outra aba com as notas da rubrica “CodeMaster CT - App Inventor" e a última com os resultados da rubrica "CodeMaster UI Design - App Inventor”.

Fluxo de Exceção:

FE01 - Professor clica em avaliar sem escolher projetos. Sistema informa uma mensagem: “Nenhum projeto foi selecionado para análise!”.

Caso de Uso 5 - USC05 Acesso a projetos analisados
Ator: Professor
Pré-condição: USC03
Fluxo Principal:

1. Professor clica em “Turmas” no menu superior da área de professores.
2. Professor seleciona a turma que deseja ver as análises.
3. Sistema consulta no Banco de Dados e retorna o resultado das análises da turma selecionada.

Fluxo de Exceção:

FE01. Nenhuma turma foi previamente cadastrada pelo professor. Sistema apresenta mensagem: “Nenhuma turma foi encontrada”.

Caso de Uso 6 - USC06 Logar como Administrador
Ator: Administrador
Pré-condição: USC07
Fluxo Principal:

1. Administrador acessa o site do CodeMaster.
2. Administrador clica no botão “Admin” no menu superior da tela.
3. O sistema autentica o administrador e é redirecionado à página de cadastro de administradores.

Fluxo de Exceção:

FE02 - O e-mail e senha do administrador não são identificados. O sistema informa uma mensagem: “Usuário e senha não identificados pelo sistema”.

Caso de Uso 7 – USC07 Cadastrar outros administradores
Ator: Administrador
Pré-condição: USC06
Fluxo Principal:

1. Administrador logado, clica em “Cadastro”.
2. Administrador clica em “Adicionar”.
3. Administrador insere os dados do novo administrador e clica em “Salvar”.
4. O sistema atualiza a tabela de administradores, incluindo o novo administrador.
5. O sistema envia um email para o novo administrador, com link de confirmação e senha.
6. Novo administrador abre link recebido no e-mail e uma mensagem de conta confirmada é exibida.

Fluxo Alternativo:

4a. Administrador logado não preenche um dos campos de texto e o sistema apresenta a mensagem “Todos os campos devem ser preenchidos”.

Caso de Uso 8 – USC08 Excluir o cadastro de administradores
Ator: Administrador
Pré-condição: USC06
Fluxo Principal:

1. Administrador logado, clica em “Cadastro”.
2. Administrador clica no simbolo “⛔” da linha correspondente ao administrador que deseja excluir o cadastro.

Caso de Uso 9 – USC09 Acesso a todos os projetos analisados
Ator: Administrador
Pré-condição: USC06
Fluxo Principal:

1. Administrador logado, clica em “Estatística”.
2. Administrador define as informações de filtragem dos dados de interesse. E clica em “Gerar .xlsx”.
3. O sistema solicita um endereço da memória local para salvar a planilha gerada

Caso de Uso 10 – Acessa Rubrica
Ator: Professor ou Aluno
Pré-condição: USC01, USC04 ou USC05
Fluxo Principal:

1. Depende do caso de uso:
    a. Se a pré-condição for USC01: aluno clica em “Clique aqui para descobrir como melhorar sua pontuação!” na rubrica de programação.
    b. Se a pré-condição for USC04 ou USC05: professor clica em “Clique para visualizar a rubrica de avaliação” na rubrica de programação.

2. Usuário clica em “Interface de Usuário” para descobrir como melhorar sua pontuação na rubrica "CodeMaster UI Design - App Inventor”.

5.3 Modelagem e Implementação do Sistema

As aprimorações propostas na ferramenta CodeMaster (Capítulo 5.1) foram desenvolvidas mantendo o modelo arquitetônico existente (GRESSE von WANGENHEIM et al., 2018) (DEMETRIO, 2017) (ALVES, 2019). O modelo arquitetônico geral da ferramenta CodeMaster separa a apresentação dos resultados e a análise dos apps em diferentes módulos. A ferramenta é dividida em 3 grandes componentes (Figura 25). O objetivo da divisão em componentes é para permitir que a ferramenta seja escalável a longo prazo e permitir a conexão direta de outras aplicações no futuro. Os componentes são:

- Container Web 1: contém o módulo de apresentação e acesso ao banco de dados.
- Container Web 2: contém o módulo de análise e avaliação de código.
- Navegador de internet: consiste no navegador de internet do usuário, onde a interface gráfica do sistema é exibida.

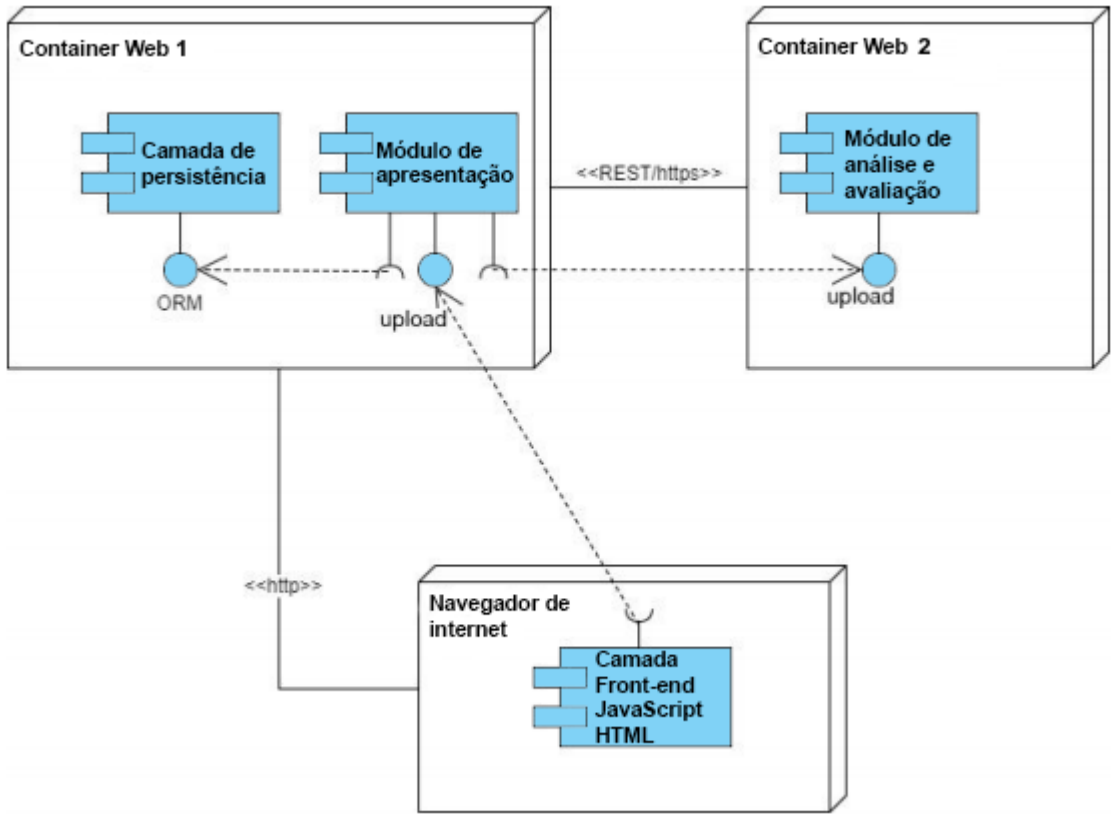


Figura 25: Diagrama de componentes (DEMETRIO, 2017).

O módulo “Apresentação” é responsável pelo controle da interface de usuário, o registro de professores e turmas, a submissão de projetos e a apresentação de resultados para professores e alunos.

O módulo “Front-end” tem como função o desenvolvimento das páginas web apresentadas no navegador de internet do usuário. Esse módulo utiliza JSP, Java Script, HTML5 e CSS3.

O módulo "Análise e avaliação" é responsável em analisar e avaliar o(s) aplicativo(s) conforme os critérios de interesse. O(s) aplicativo(s) e o arquivo de configuração são fornecidos pelo módulo “Apresentação”. E módulo "Análise e avaliação" envia para o módulo “Apresentação” o(s) *feedback(*s) instrucional(ais) e a pontuação alcançada em cada critério. Essa comunicação é feita por meio de um serviço web *Representational State Transfer* (REST). REST é uma arquitetura que permite a separação de sistemas web em módulos (RODRIGUEZ, 2008). Para implementar esse serviço o CodeMaster utiliza o framework Jersey (2019), pois abstrai os detalhes de baixo nível da implementação de comunicação entre os servidores e simplifica a implementação de um serviço REST. A escolha do framework REST foi devido a simplicidade de uso, por ser adequado ao que é proposto no CodeMaster e pela grande quantidade de material disponível (DEMETRIO, 2017).

5.3.1 Arquitetura e Implementação do Módulo de "Análise e Avaliação"

Após o usuário submeter um ou mais aplicativos no sistema, o módulo “Apresentação” envia para o módulo "Análise e Avaliação" o(s) aplicativo(s) e um arquivo de configurações. O arquivo de configurações apresenta um conjunto de booleanos que informa quais os critérios que o usuário tem interesse e, consequentemente, os que serão avaliados.

O aplicativo App Inventor é representado por um arquivo compactado com extensão .aia. O arquivo .aia contém um conjunto de arquivos .scm e .bky para cada tela do aplicativo. O arquivo .scm apresenta as informações sobre o design da tela, encapsuladas em uma estrutura json. Enquanto o arquivo .bky encapsula uma estrutura xml, com todos os blocos de programação utilizados na lógica do app.

O objeto de configurações (`ProjectSettings.java`) e cada arquivo .aia são enviados ao avaliador do App Inventor (`AppInventorGrader.java`) e o seguinte procedimento é realizado (Figura 26)(Figura 27):

1. O primeiro passo do avaliador é descompactar o arquivo .aia e obter apenas os arquivos com extensão bky e scm, que contém as informações de código relevantes para a análise. Cada arquivo scm é registrado como um JsonObject e incluído numa lista.
2. O avaliador `AppInventorGrader.java` executa dois métodos. Um método tem referências aos critérios da rubrica “CodeMaster CT - AppInventor”. Outro método referência os critérios da rubrica "CodeMaster UI Design - App Inventor” (Capítulo 4.2). Cada método retorna a nota alcançada em cada rubrica.

a. Cada critério da rubrica "CodeMaster UI Design - App Inventor” analisa todos os JsonObject do arquivo scm verificando os componentes visuais de interesse. Após a análise é definido o nível de desempenho alcançado. Caso não tenham os componentes visuais de interesse, o critério retorna um valor que corresponde ao estado “não se aplica”.
b. Os níveis de desempenho alcançados por cada critério são registrados em um objeto que estende Grade. No caso dos níveis de desempenho da rubrica "CodeMaster UI Design - App Inventor” é registrado em um objeto “AppInventorGradeDesign”. Os níveis de desempenho da rubrica “CodeMaster CT - AppInventor” é registrado em um objeto “AppInventorGradePrograma”.
c. Nos objetos Grades também são registradas variáveis que auxiliam na determinação dos níveis de desempenho alcançado (Apêndice C).

3. `AppInventorGrader.java` calcula a nota final com base nas notas alcançadas de cada rubrica conforme apresentado no capítulo 4.6. Com base na nota final é identificada a cor da faixa do ninja e registrada no objeto “AppInventorGradePrograma”.
4. `AppInventorGrader.java` gera um objeto JsonArray. Cada elemento é um é json gerado de um objeto Grade. O processo é finalizado com o objeto JsonArray sendo enviado para o módulo “Apresentação”.

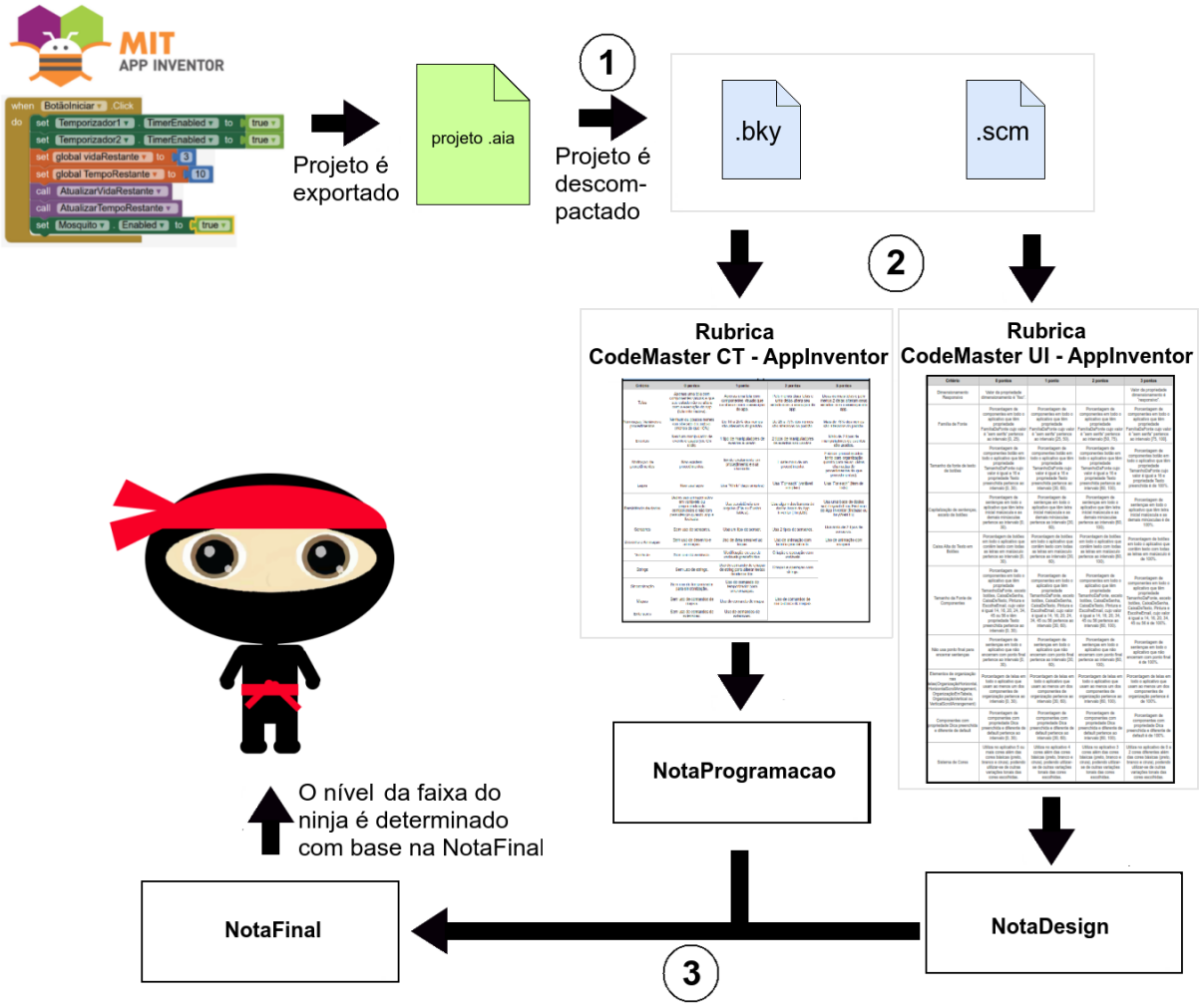


Figura 26: Sequência da análise do projeto App Inventor

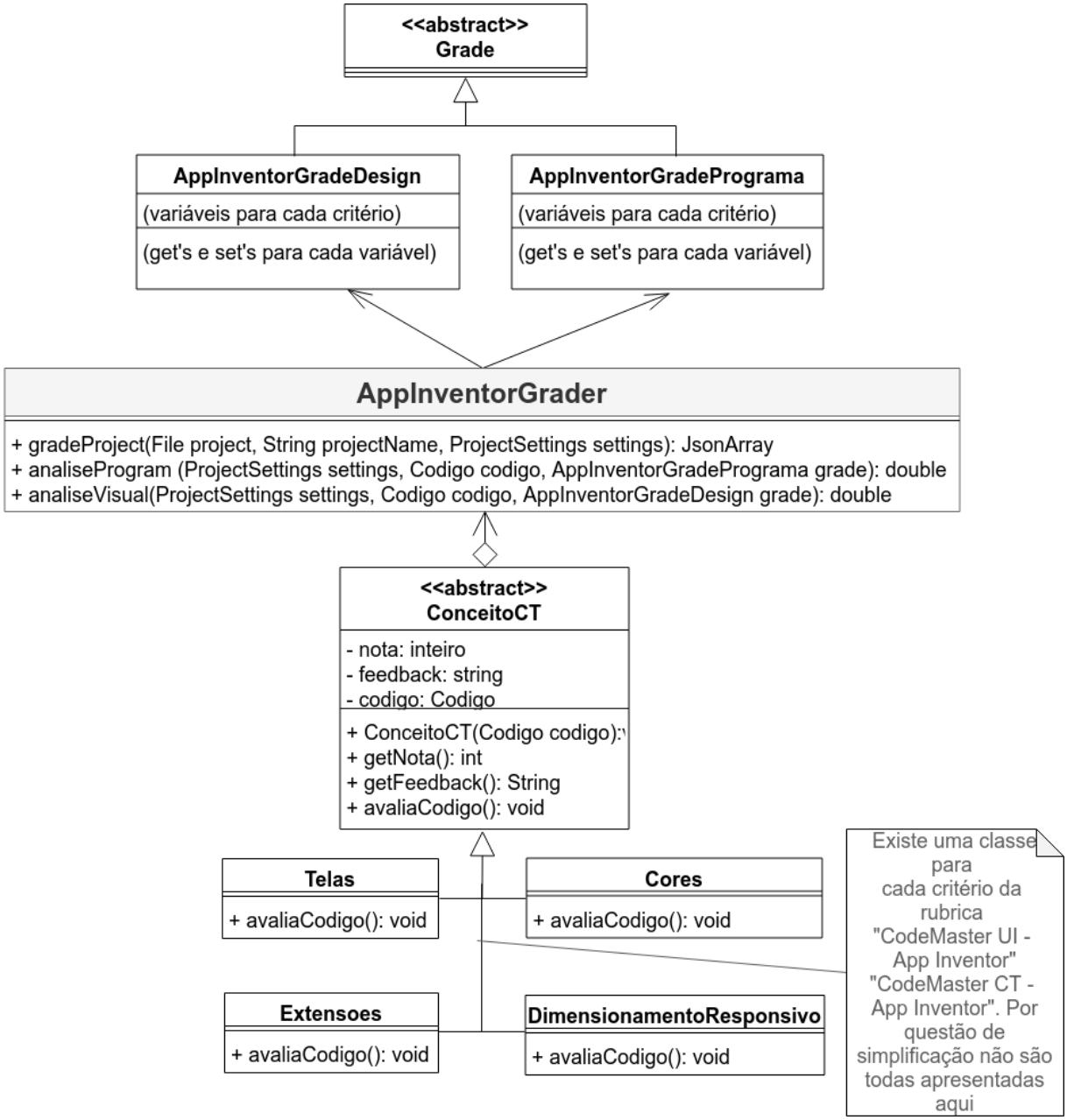


Figura 27: Diagrama de classes simplificado

5.3.2 Arquitetura e Implementação do Módulo de “Apresentação”

O módulo de “Apresentação” é responsável por receber o projeto do usuário por meio da interface web, enviar o projeto ao módulo "Análise e Avaliação" e apresentar o resultado da avaliação ao usuário. Este módulo também é responsável pelo cadastro de professores, a autenticação do professor no sistema, envio de e-mail para confirmação de cadastro, gerenciar o idioma e gerenciar todo o armazenamento de projetos, turmas, professores e administradores no banco de dados.

Para todos os fluxos que precisam acessar o Banco de Dados (BD), uma classe HTTPClient faz o envio da requisição e recebe uma resposta. O processo da requisição é intermediado por métodos da classe Control. Esses métodos referenciam objetos DAOs, que acessam ou alteram os dados do BD. Com os dados necessários, os métodos da classe Control tratam os dados, como encapsular em um objeto, e retornam à classe HTTPClient. Por exemplo, para cadastrar administradores a classe `CadastroAdmin.java` recebe a requisição de um novo cadastro. Para isso ela chama o método `salvarAdmin()` da classe `Control.java`. O método `salvarAdmin()` encapsula os dados em um objeto Administrador. O objeto Administrador é enviado como parâmetro para o método `salvar()` da classe AdminDAO (Figura 28). Após o cadastro concluído, um e-mail com o link de confirmação e senha é enviado ao endereço de email do administrador cadastrado.

Outro exemplo é a exportação dos projetos já analisados do Banco de Dados (BD) pelo administrador (Figura 29). A classe `MysqlToXls.java` recebe a requisição e chama o método `gerarXLS()`, que gera o arquivo .xlsx com auxílio das bibliotecas da Apache POI. O método `gerarXLS()` adquire os dados através de uma referência a `xlsDAO.java`. A classe `xlsDAO.java` coleta os dados conforme os parâmetros de filtragem definidos pelo usuário. Quando a planilha está montada, o sistema apresenta um pop-up para definir onde salvar a planilha localmente.

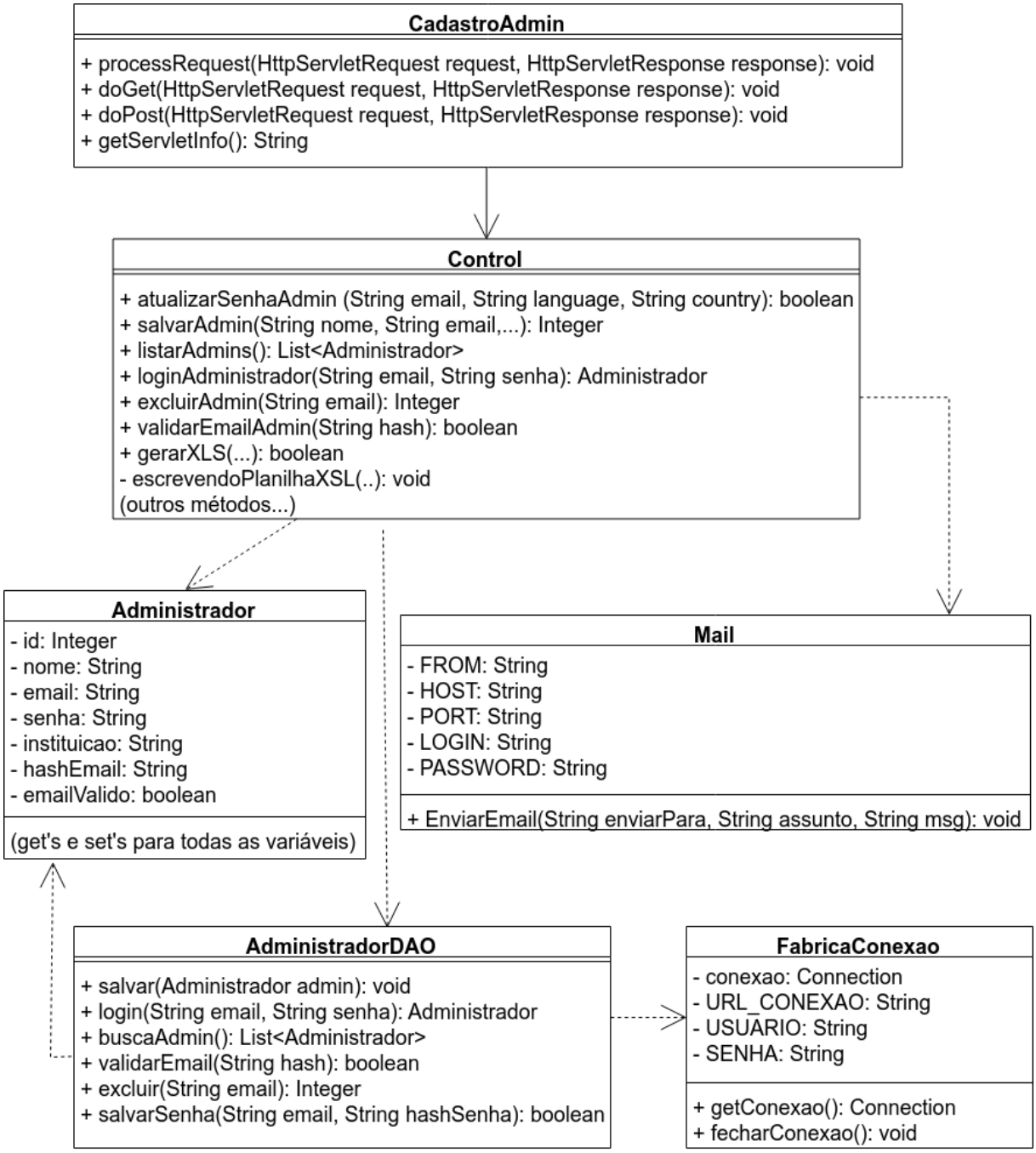


Figura 28: Diagrama de classes do cadastro de administrador

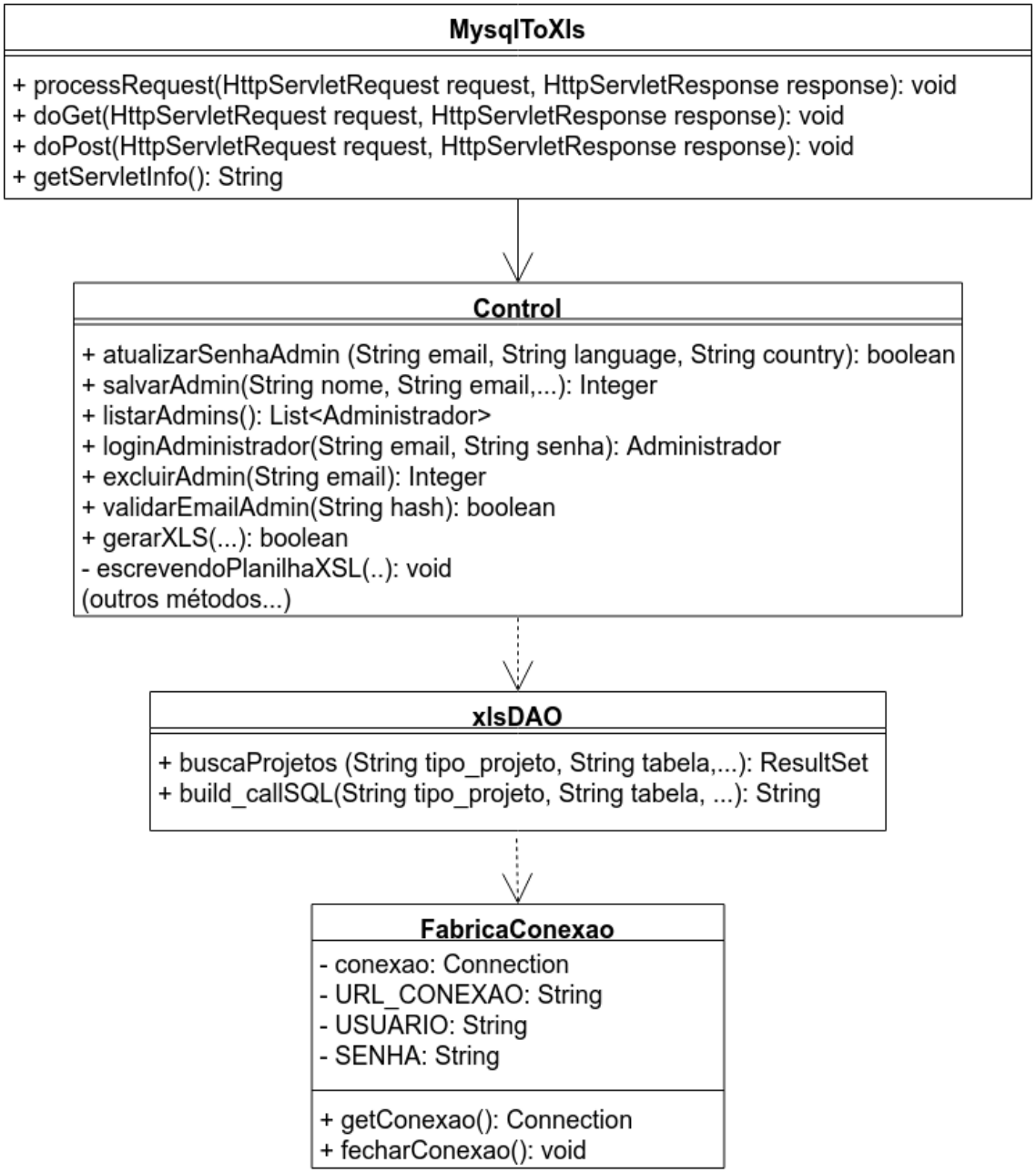


Figura 29: Diagrama de classes da geração da planilha xlsx

5.3.3 Arquitetura e Implementação da "Camada de Persistência"

A "Camada de Persistência" é um Banco de Dados Relacional (BD). Para gerenciar o BD é usado o sistema de gerenciamento de banco de dados MySQL. MySQL é um sistema Open Source e tem foco para sistemas online, sendo ideal para a ferramenta CodeMaster (MYSQL, 2019). A "Camada de Persistência" é acessada apenas pelo módulo de "Apresentação". A "Camada de Persistência" é formada pelas tabelas professor, turma, evento, grupo_de_alunos, projeto e administrador (Figura 30).

A **tabela professor** armazena os dados de cadastro de professores (nome, instituição, disciplina, etc) e os dados de acesso ao modo professor no CodeMaster (e-mail e senha).

A **tabela turma** registra as turmas criadas por cada professor. Nessa tabela é registrado o ID da turma criada, que é gerado automaticamente e é uma chave primária, ou seja, tem valor único para cada turma. O nome que o professor deu para a turma. E uma chave estrangeira que relaciona a tabela professor e informa qual é o professor da cada turma.

A **tabela projeto** armazena as informações de projetos avaliados. Nessa tabela foram feitas alterações nesta tabela a partir do proposto em Demetrio (2017). Inicialmente tinha a coluna `resultado_avaliacao`, que armazenava os dados da rubrica “CodeMaster CT - App Inventor". Com a inclusão da rubrica "CodeMaster UI Design - App Inventor” foi tomada a decisão de armazenar os resultados de cada rubrica em colunas distintas. Pois foi considerado mais viável organizacionalmente. Assim a versão da tabela de projeto agora contém duas colunas, a coluna `resultado_avaliacao_design` e a `resultado_avaliacao_programa`. Em que a coluna `resultado_avaliacao_design` armazena um JSON com os resultados da rubrica "CodeMaster UI Design - App Inventor” e a coluna `resultado_avaliacao_programa` um JSON os resultados da rubrica “CodeMaster CT - App Inventor" .

A estrutura JSON da coluna `resultado_avaliacao_design` corresponde ao de um objeto da classe `AppInventorGradeDesign.java`. Os objetos da classe `AppInventorGradeDesign.java` apresentam informações sobre:

- Níveis de desempenho alcançados em cada critério da rubrica "CodeMaster UI Design - App Inventor”.
- Variáveis auxiliares: são variáveis que auxiliaram na definição dos níveis de desempenho (Apêndice C). Essas variáveis são acessíveis apenas pelos administradores através da planilha gerada, podendo auxiliar na análise realizada com os dados.
- Nota da rubrica.

A **tabela administrador** apresenta os dados dos administradores cadastrados. (nome, instituição, etc) e os dados de acesso ao modo administrador no CodeMaster (e-mail e senha).

A **tabela evento** apresenta um recurso a ser adicionado futuramente no sistema. para que o professor defina eventos que uma determinada turma participa ao CodeMaster. Podendo a mesma turma realizar diferentes eventos.

A **tabela grupo_de_alunos** armazena a informação do nome do grupo de alunos ou o nome de um aluno em específico que criou determinado projeto. O nome do grupo de alunos ou aluno é determinado por meio do nome do arquivo de projeto enviado ao sistema. Portanto, no momento do envio do projeto para avaliação ao CodeMaster é importante que o nome do projeto contenha esta informação, por exemplo `MariaSilva.aia`. Essa informação permite acessar vários

projetos do mesmo aluno ou grupo de aluno. Porém, o recurso para acessar essa informação ainda não está implementada.

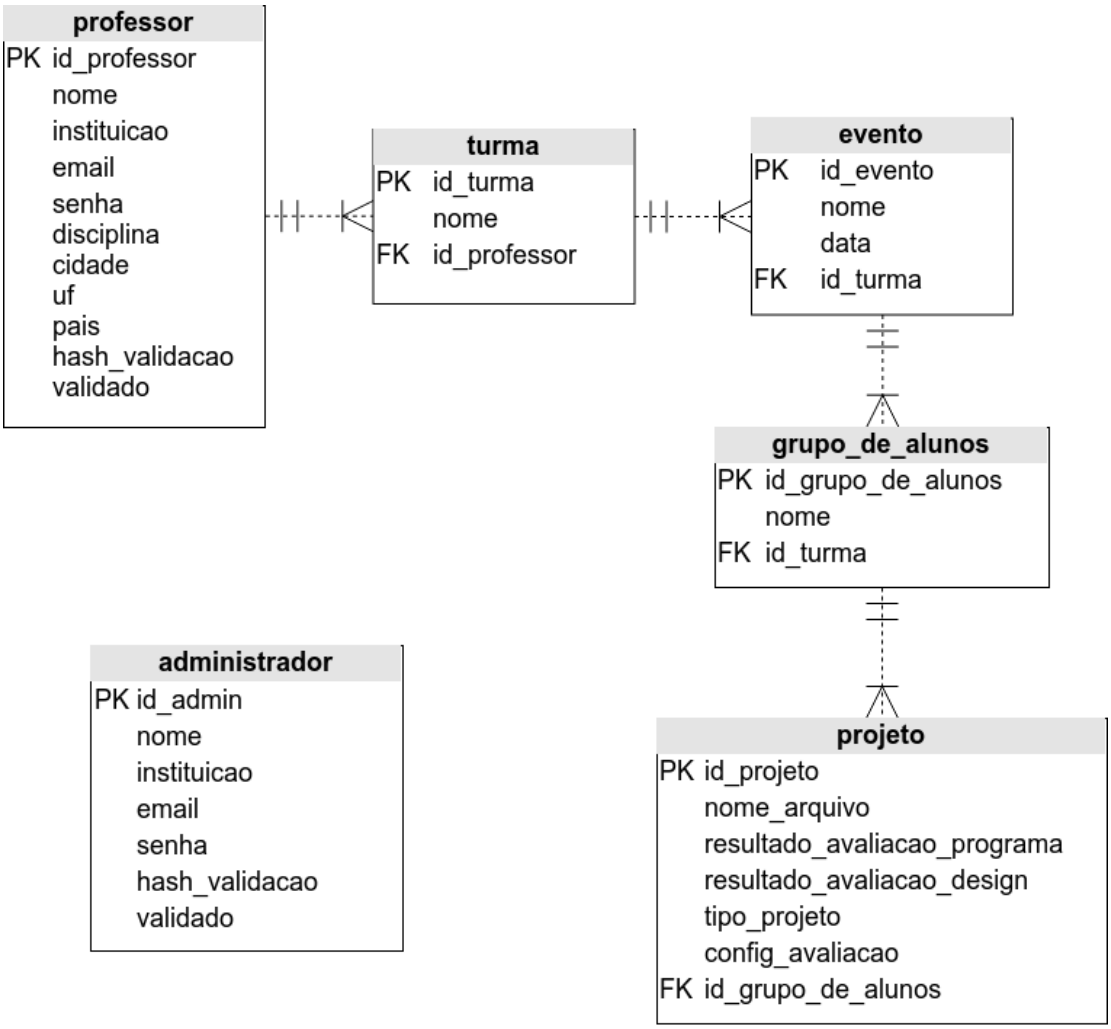


Figura 30: Diagrama Entidade Relacionamento do Banco de Dados CodeMaster

Por questão de segurança, as senhas e emails dos usuários cadastrados (professor e administrador) são armazenados em formato de um hash. Para gerar cada hash, é utilizado o algoritmo SHA-256. Assim é possível a partir de uma senha obter o hash, mas a partir do hash é impossível obter a senha. Isso dificulta a descoberta da senha dos usuários por pessoas mal intencionadas que possam vir a acessar o BD.

5.3.4 Implementação do Design de Interface

Conforme os recursos propostos (Capítulo 5.1), ajustes nas telas foram feitos e criação de novas telas também. A seguir são apresentadas as principais novidades. Para mais detalhes o Apêndice D apresenta o fluxo principal de cada caso de uso e as telas correspondentes.

Em relação ao novo usuário, o administrador, foi acrescentado no menu principal a opção aos recursos desse usuário (Figura 31). As telas sobre a funcionalidade de cadastrar outros administradores (Figura 32) e extrair planilha (Figura 33) também foram implementadas.

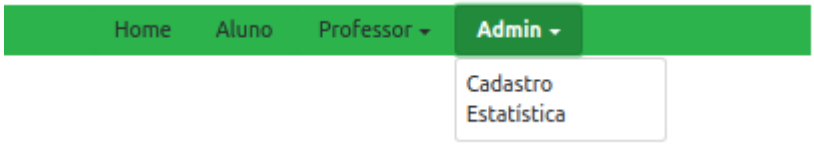


Figura 31: Menu principal do site

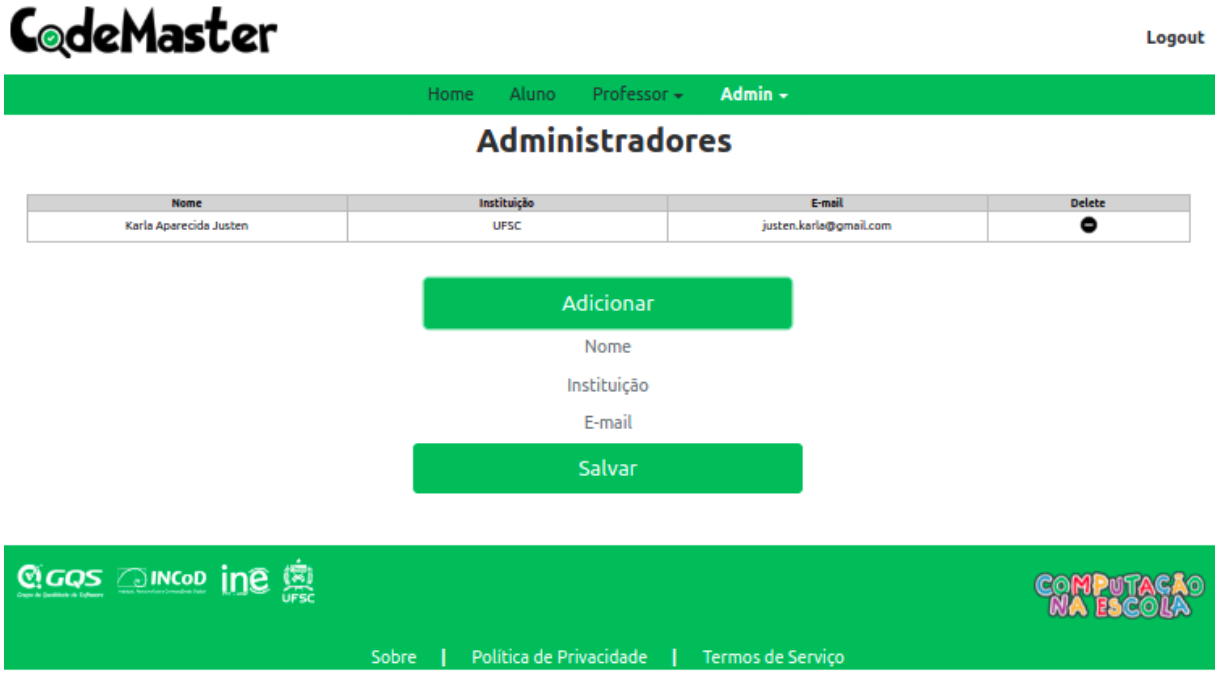


Figura 32: Página referente ao caso de uso USC07

Figura 33: Página referente ao caso de uso USC09

Devido à inclusão da rubrica "CodeMaster UI Design - App Inventor" foi necessário atualizar a tela de *feedback* dos alunos (Figura 34), a tela da rubrica dos projetos App Inventor (Figura 35), a tela de submissão de múltiplos apps para o professor e a tela *feedback* do professor (Figura 36). Em todas essas telas foram incluídas uma aba, chamada de "Interface de usuário", com o intuito de para apresentar os critérios da rubrica "CodeMaster UI Design - App Inventor".

Os critérios da rubrica "CodeMaster UI Design - App Inventor" são organizados em categorias são Layout, Tipografia, Personalização, Escrita e Cores. Para apresentar uma interface organizada e limpa, cada categoria é apresentada como um botão, que ao clicado alterna entre mostrar e ocultar os critérios correspondentes da categoria.

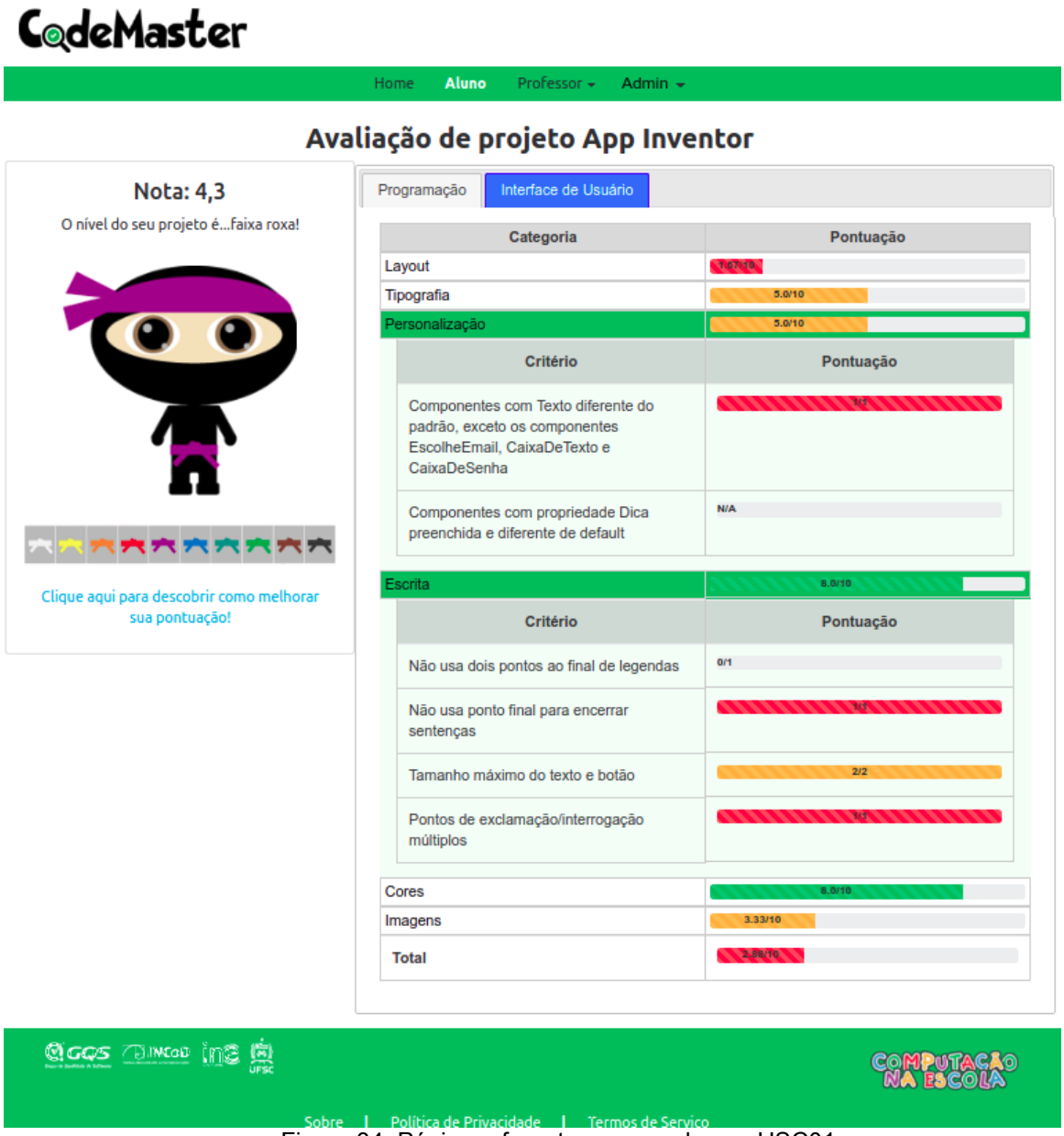


Figura 34: Página referente ao caso de uso USC01

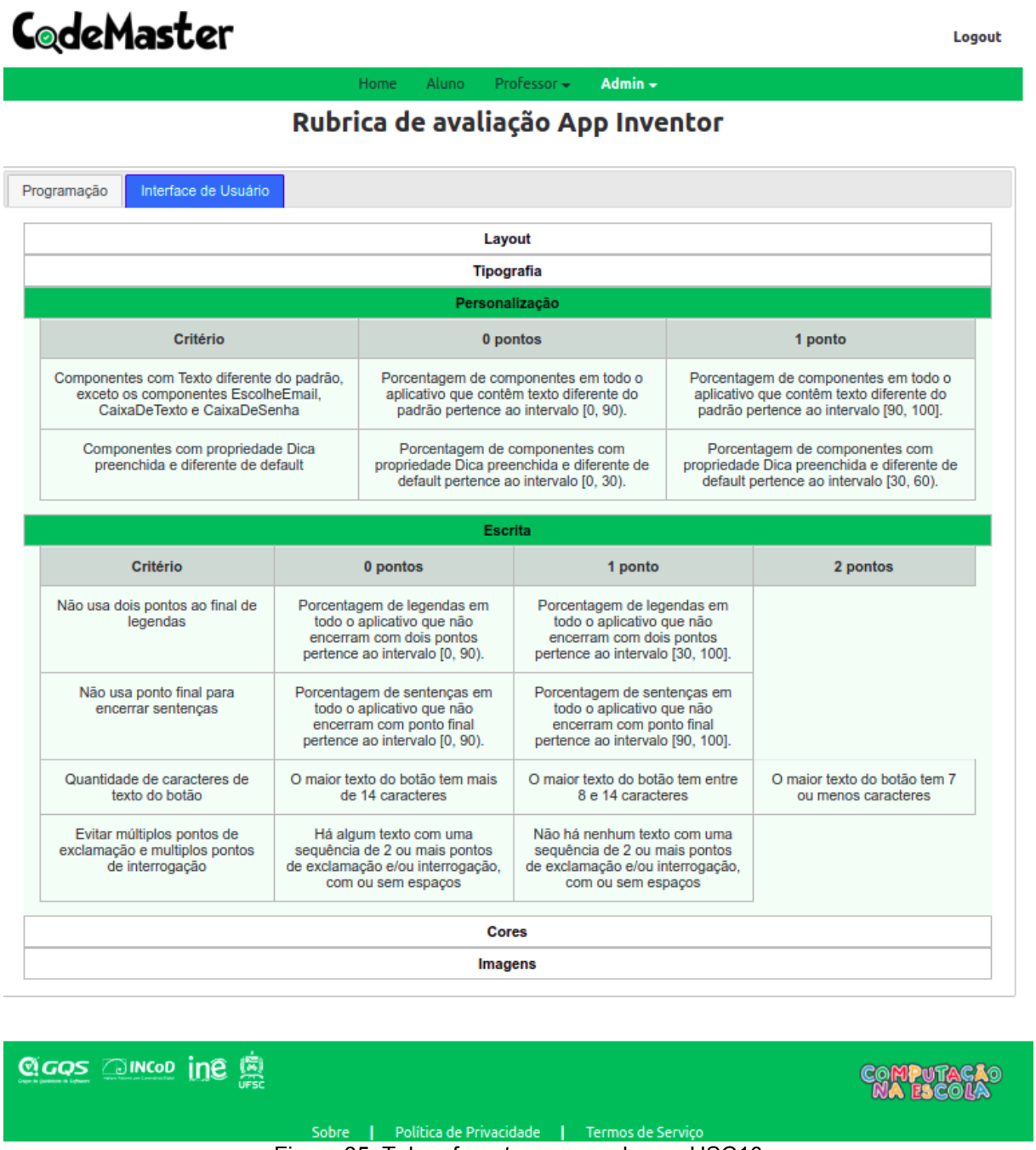


Figura 35: Tela referente ao caso de uso USC10

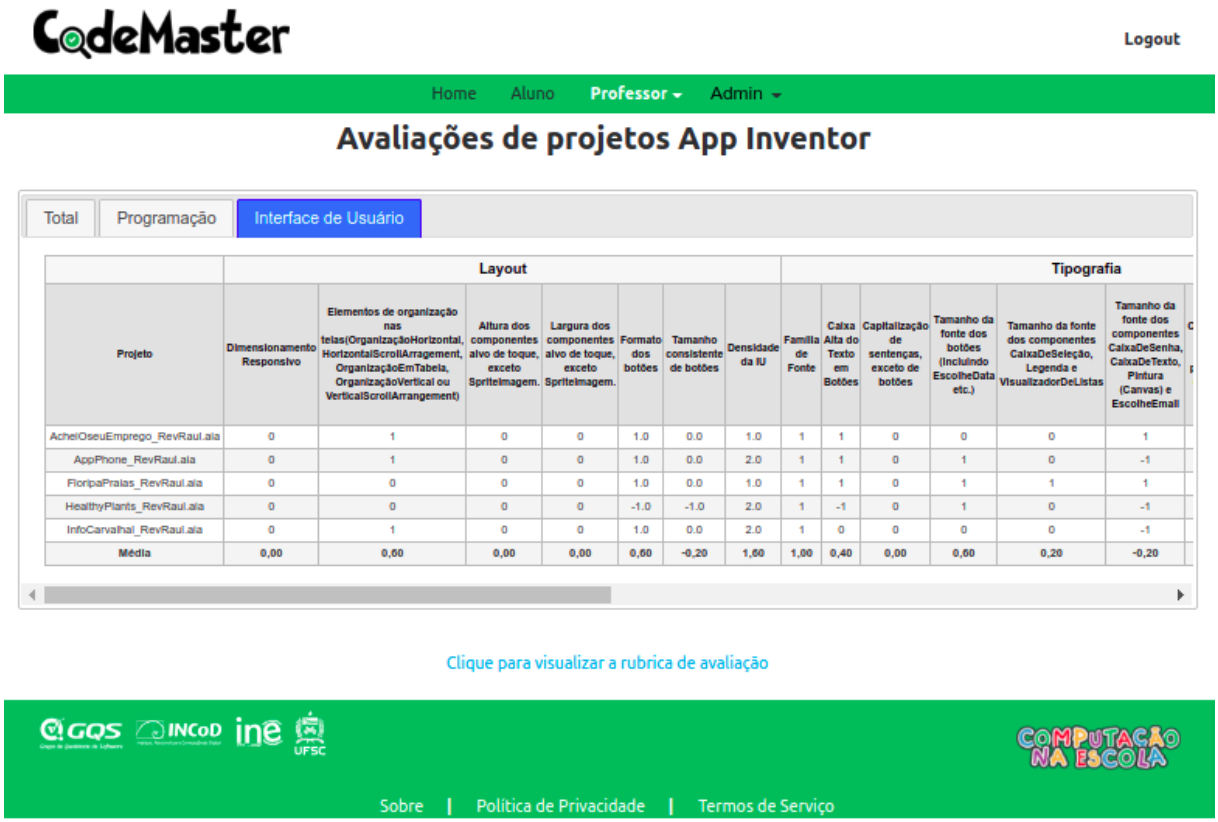


Figura 36: Página referente ao caso de uso USC04 e USC05

A ferramenta CodeMaster v1.0 está disponível online em: http://apps.computacaonaescola.ufsc.br:8080.

5.3.5 Expandindo o Codemaster

O CodeMaster foi projetado para que as expansões sejam fáceis de serem implementadas. Nesta seção é apresentado o processo de implementação para adicionar uma nova rubrica ao analisador de projetos App Inventor e como adicionar um novo critério à uma rubrica.

Adicionando nova rubrica

A inclusão de uma rubrica no analisador do App Inventor requer alterações no módulo “Apresentação” e no módulo “Análise e Avaliação”.

No módulo “Análise e Avaliação” é preciso:

1. criar um objeto Grade.
2. incluir na classe `AppInventorGrader.java` um método que referencia os critérios da nova rubrica. Nesse método o objeto grade é preenchido com as informações de interesse.
3. Referenciar o método criado no passo anterior no método `gradeProject()` da classe `AppInventorGrader.java`

4. Ao final do método `gradeProject()` um objeto JsonArray é criado. Cada elemento é baseado no objeto Grade de cada rubrica (Figura 37). Então é preciso incluir um json com o Grade da nova rubrica.

```
JsonArray arrayGrades = new JsonArray();
Gson gson = new GsonBuilder().create();
arrayGrades.add(gson.toJsonTree(gradePrograma,AppInventorGradePrograma.class));
arrayGrades.add(gson.toJsonTree(gradeDesign,AppInventorGradeDesign.class));
return arrayGrades;
```

Figura 37: Trecho de código final do método gradeProject()

É importante saber claramente qual elemento do JsonArray representa qual rubrica. Pois quando o JsonArray for enviado para o módulo “Apresentação”, cada elemento json será convertido para o objeto Grade correspondente.

O módulo “Apresentação” precisa ser preparado para interpretar corretamente o JsonArray a ser recebido e saber onde apresentar as informações. Para isso os seguintes passos precisam ser realizados:

1. Incluir uma aba para apresentar os critérios da nova rubrica nas páginas de feedback do aluno (`aluno_resultado_appinventor.jsp`), feedback do professor (`professor_resultado_appinventor.jsp`), escolha de critérios quando o professor está cadastrado turmas (`professor.jsp`) e na página que apresenta as rúbricas descritas (`rubrica_appinventor.jsp`).
2. Criar um objeto Grade da nova rubrica nas páginas de feedback do aluno (`aluno_resultado_appinventor.jsp`), feedback do professor (`professor_resultado_appinventor.jsp`). Esse objeto grade precisa ser instaciado por um elemento do JsonArray (Figura 38).

```
JsonParser jsonParser = new JsonParser();
JsonArray arrayGrades = (JsonArray)jsonParser.parse(msg);
AppInventorGradePrograma gPrograma = gson.fromJson(arrayGrades.get(0), AppInventorGradePrograma.class);
AppInventorGradeDesign gDesign = gson.fromJson(arrayGrades.get(1), AppInventorGradeDesign.class);
```

Figura 38: Trecho de código que interpreta o JsonArray em aluno_resultado_appinventor.jsp

3. A tabela projeto do banco de dados precisa ser atualizada. É preciso incluir uma coluna para o resultado da rubrica nova. Consequentemente, é necessário atualizar a classe Projeto.java, o método `salvarProjetoBd()` da classe `Control.java` e a classe `ProjetoDAO`.

Com a rubrica incluída no sistema, é possível então popular a rubrica com critérios.

Adicionando novo critério

A inclusão da avaliação de um novo critério em uma das rubricas requer que os módulos “Análise e avaliação” e “Apresentação” sejam alterados. A seguir é explicado o passo a passo e para exemplificar é usado o processo de inclusão do critério dimensionamento responsivo no CodeMaster (Tabela 27).

Tabela 27: Critério Dimensionamento Responsivo

Critério	Níveis de Desempenhos	
	0	
Dimensionamento responsivo	Valor da propriedade dimensionamento é “fixo”.	Valor da pro

No módulo “Análise e Avaliação”, o processo de adicionar um critério para avaliar projetos App Inventor inicia com a implementação de uma classe que estenda ConceitoCT. As classes que estendem ConceitoCT são salvos no pacote `br.ufsc.cne.codemaster.restgrader.appinventorgrader.conceitos`.

Conforme o exemplo, é criada uma classe chamada `DimensionamentoResponsivo.java` (Figura 39). Na classe nova é preciso implementar o método `avaliaCódigo()`. O objetivo do método `avaliaCódigo()` é definir o nível de desempenho alcançado no critério. Para isso é preciso saber como encontrar as informações de interesse e se essa informação está no arquivo scm ou bky.

```
public class DimensionamentoResponsivo extends ConceitoCT {

    public DimensionamentoResponsivo(Codigo codigo) {
        super(codigo);
        avaliaCodigo();
    }

    @Override
    public void avaliaCodigo() {

        JSONObject properties = (JSONObject) codigo.getcodigosTelasVisual().get(0).get("Properties");
        if (properties.containsKey("Sizing")) {
            this.nota = 1;
        } else {
            this.nota = 0;
        }
    }
}
```

Figura 39: Classe DimensionamentoResponsivo.java

Em seguida é preciso adicionar na classe `ProjectSettings.java` métodos *get* e *set* de uma variavel boolena. O objetivo desses métodos é saber se o critério é para ser avaliado (Figura 40).

```
public Boolean getDimensionamentoResponsivoIsRelevant() {
    return dimensionamentoResponsivoIsRelevant;
}

public void setDimensionamentoResponsivoIsRelevant(Boolean dimensionamentoResponsivo) {
    this.dimensionamentoResponsivoIsRelevant = dimensionamentoResponsivo;
}
```

Figura 40: Trecho da classe ProjectSettings.java

É preciso adicionar métodos *get* e *set* para cada variável que pretende armazenar no banco de dados em uma classe `Grade.java`. A classe `Grade.java` a ser usada é a que referência à rubrica em que o critério pertence. Como o critério dimensionamento responsivo é para ser incluído na rubrica "CodeMaster UI Design -

App Inventor", a classe representante é `AppInventorGradeDesign.java` (Figura 41).

```
public Integer getDimensionamentoResponsivo() {
    return dimensionamentoResponsivo;
}

public void setDimensionamentoResponsivo(Integer dimensionamentoResponsivo) {
    this.dimensionamentoResponsivo = dimensionamentoResponsivo;
}
```

Figura 41: Trecho da classe AppInventorGradeDesign.java

Em seguida, é necessário referenciar a análise do critério na classe `Grader.java`. No exemplo tratado, o projeto avaliado é um App Inventor, então é preciso referenciar o critério na classe `AppInventorGrader.java`.

Na classe `AppInventorGrader.java` tem o método `analiseVisual()`, que referencia os critérios da rubrica "CodeMaster UI Design - App Inventor", então é nela que as referências do critério Dimensionamento Responsivo precisa ser incluído (Figura 42).

```
// Dimensionamento Responsivo
if(settings.getDimensionamentoResponsivoIsRelevant()) {
    ConceitoCT dimResp = new DimensionamentoResponsivo(codigo);
    grade.setDimensionamentoResponsivo(dimResp.getNota());
    if (grade.getDimensionamentoResponsivo() >= 0)
    {
        qtdConceitosVisuaisAnalisados++;
        totalVisual += grade.getDimensionamentoResponsivo();
    }
    log.info("Sua nota em DimensionamentoResponsivo foi: " + dimResp.getNota() +
            " | Feedback: " + dimResp.getFeedback());
}else {
    grade.setDimensionamentoResponsivo(0);
}
```

Figura 42: Trecho de código que referência o critério Dimensionamento Responsivo no método analiseVisual()

No módulo "Apresentação" também é preciso atualizar as classes `ProjectSettings.java` e `AppInventorGradeDesign.java` conforme feito com essas classes que estão no módulo "Análise e Avaliação" (Figura 40 e 41). Essas classes ficam no pacote `br.cne.codemaster.duplicated`.

As telas de feedback apresentadas para os usuários, alunos e professores, precisam ser atualizadas. Essas atualizações consistem em adicionar uma lacuna para apresentar o nível de desempenho alcançado no novo critério. Por exemplo, o critério dimensionamento responsivo precisa ser adicionado junto com os critérios da rubrica "CodeMaster UI Design - App Inventor". Ou seja, na aba "Interface de Usuário" da tela `aluno_resultado_appinventor.jsp` (Figura 34) e da tela `professor_resultado_appinventor.jsp` (Figura 36).

É preciso atualizar a lista de critérios em que permite o professor definir quais critérios devem ser avaliados. Para acrescentar o novo critério na lista é preciso

acrescentar um checkbox na classe `professor.jsp` (Figura 43), especificamente na aba da rubrica. Conforme o exemplo, é na aba “Interface de Usuário”.

```
<div class="checkbox">
    <label><input type="checkbox" name="checkDimenResp" value="" checked>
    <%=messages.getString("ConceitoDimensionamentoResponsivo")%>
    </label>
</div>
```

Figura 43: Código para cada critério na lista de escolha de avaliação pelo professor

Após a alteração da classe `professor.jsp`, foi atualizado o método `getProjectSettings()` da classe Upload.java. É necessário verificar se o *checkbox* do critério foi selecionado ou não pelo professor e atribuir essa informação no objeto ProjectSettings (Figura 44). Esse objeto ProjectSettings será posteriormente enviado para o módulo “Análise e Avaliação” .

```
private ProjectSettings getProjectSettings(HttpServletRequest request) {
    ProjectSettings settings = new ProjectSettings();
    ...
    settings.setDimensionamentoResponsivoIsRelevant(request.getParameter("checkDimenResp") != null);
    return settings;
}
```

Figura 44: Método getProjectSettings() da classe Upload.java

O próximo passo é acrescentar a descrição do critério na página da rubrica correspondente. Conforme o exemplo a inclusão do critério é na aba “Interface do Usuário” da página `rubrica_appinventor.jsp`.

Como a ferramenta web oferece a opção do site ser apresentado em português ou em inglês, para todas as páginas jsp alteradas do módulo “Apresentação” é preciso acrescentar as informações necessárias nos arquivos Bundle_en_US.properties e Bundle_pt_BR.properties correspondentes. Por exemplo, a figura 43 apresenta `ConceitoDimensionamentoResponsivo`, pois se o site for apresentado em português, será acessado o arquivo `ProfessorBundle_pt_BR.properties` e nesse arquivo “`ConceitoDimensionamentoResponsivo = Dimensionamento Responsivo`”. Mas se o site for apresentado em inglês, será acessado o arquivo `ProfessorBundle_en_US.properties` e nesse arquivo “`ConceitoDimensionamentoResponsivo = Responsive Sizing`”.

Com a implementação do novo critério, é necessário realizar testes de corretude.

5.4 Teste de Corretude

O teste de corretude consiste em verificar se a ferramenta CodeMaster, ao avaliar um aplicativo do App Inventor, apresenta o nível de desempenho alcançado correto para cada critério da rubrica "CodeMaster UI Design - App Inventor”.

O teste de corretude foi realizado em duas etapas. Na primeira etapa foram desenvolvidos no total 321 aplicativos App Inventor para testar a corretude dos

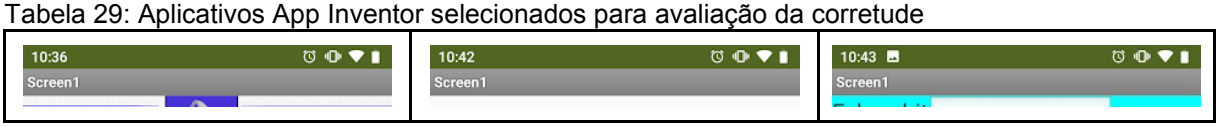
critérios implementados. Os aplicativos foram desenvolvidos pelo bolsista de iniciação científica João Vitor Araújo Porto da iniciativa Computação na Escola do GQS/InCod/INE/UFSC. Cada aplicativo foi avaliado manualmente, depois submetido no CodeMaster e verificada a igualdade entre os níveis de desempenho. Por exemplo, o critério T4 (Não usar itálico) (Tabela 28). Caso houvesse discordância entre a avaliação manual e a automática, uma análise era feita para entender a causa e, se necessário, correções na implementação eram feitas. Esse processo foi realizada de maneira iterativa até que a avaliação manual e a automática resultassem em níveis de desempenhos iguais.

Tabela 28: Resultados dos testes considerando o critério “Não usar itálico”.

Nome do aplicativo	Nível de performance esperado (manual)	Nível de performance alcançado (automática)	Descrição
A_T4_0_0	0	0	Aplicativo contém 10 sentenças (de 10) cuja propriedade FonteItálico tem valor verdadeiro. Nenhuma sentença é vazia.
A_T4_0_9	0	0	Aplicativo contém 9 sentenças (de 10) cuja propriedade FonteItálico tem valor falso. Nenhuma sentença é vazia.
A_T4_1_10	1	1	Aplicativo contém 10 sentenças (de 10) cuja propriedade FonteItálico tem valor falso. Nenhuma sentença é vazia.

Na segunda etapa do teste de corretude foi realizada a comparação entre a avaliação manual e a avaliação automática de 9 aplicativos aleatórios publicados na Galeria do App Inventor (Tabela 29). Conforme a tabela 30 os resultados são os iguais, ou seja, o módulo de avaliação automática da rubrica "CodeMaster UI Design - App Inventor” passou no teste.

Tabela 29: Aplicativos App Inventor selecionados para avaliação da corretude



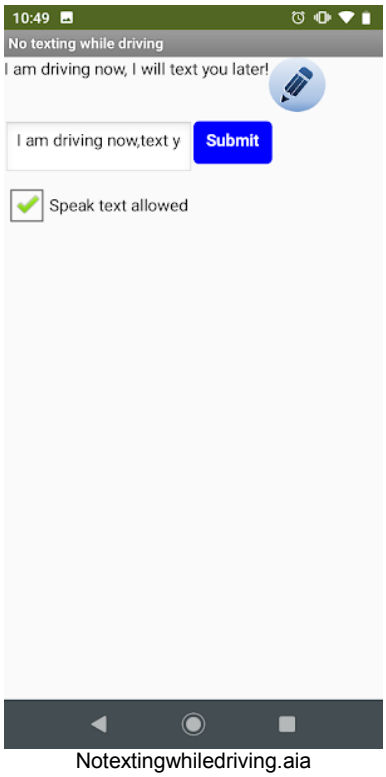
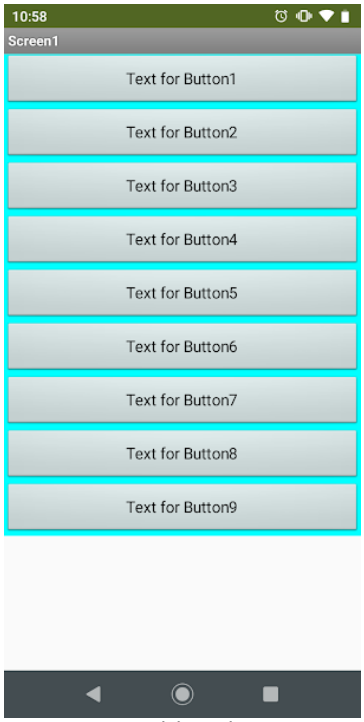
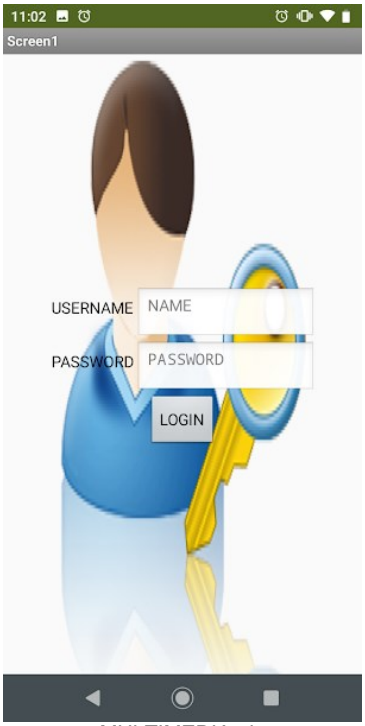
DASAPaintItTemplate.aia	HomeworkClock.aia	Fahrenheit2Celcius.aia
 Notextingwhiledriving.aia	 colorrandomizerspark2.aia	 LIFEORDEATH.aia
 LionattracterThomasSPARK1.aia	 musicbox.aia	 MULTIMEDIA.aia

Tabela 30: Resultado da segunda etapa do teste de corretude

apps		Critérios																									
		L1	L2	L3	L4	L5	L6	T1	T2	T3	T4	T5	E1	E2	E3	E4	E5	E6	E7	E8	C1	C2	C3	C4	I1	I2	I3	I4	I5
Notextingwhiledriving.aia	Aut.	0	1	0	1	1	2	1	1	1	1	1	0	1	1	1	1	1	2	1	2	2	0	1	0	1	0	-1	-1
	Man.	0	1	0	1	1	2	1	1	1	1	1	0	1	1	1	1	1	2	1	2	2	0	1	0	1	0	-1	-1
HomeworkClock.aia	Aut.	0	1	1	1	0	2	1	1	0	1	1	0	1	1	-1	1	1	2	1	2	2	0	1	-1	-1	-1	-1	-1
	Man.	0	1	1	1	0	2	1	1	0	1	1	0	1	1	-1	1	1	2	1	2	2	0	1	-1	-1	-1	-1	-1
Fahrenheit2Celcius.aia	Aut.	0	1	1	1	0	2	1	0	1	1	1	0	1	1	0	1	1	2	1	2	2	0	1	-1	0	-1	0	-1
	Man.	0	1	1	1	0	2	1	0	1	1	1	0	1	1	0	1	1	2	1	2	2	0	1	-1	0	-1	0	-1
LionattracterThomasSPARK1.aia	Aut.	0	0	1	-1	1	2	1	-1	0	1	-1	-1	0	1	-1	1	1	-1	1	2	2	0	0	0	1	0	-1	0
	Man.	0	0	1	-1	1	2	1	-1	0	1	-1	-1	0	1	-1	1	1	-1	1	2	2	0	0	0	1	0	-1	0
musicbox.aia	Aut.	0	1	1	1	1	2	1	1	-1	1	1	0	1	0	-1	-1	1	0	1	2	2	0	1	-1	-1	-1	-1	-1
	Man.	0	1	1	1	1	2	1	1	-1	1	1	0	1	0	-1	-1	1	0	1	2	2	0	1	-1	-1	-1	-1	-1
LIFEORDEATH.aia	Aut.	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	-1	1	0	1	2	0	0	1	-1	0	-1	-1	0
	Man.	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	-1	1	0	1	2	0	0	1	-1	0	-1	-1	0
MULTIMEDIA.aia	Aut.	0	0	0	1	0	2	1	1	1	1	1	0	0	1	0	1	1	0	1	0	0	0	-1	-1	0	-1	-1	0
	Man.	0	0	0	1	0	2	1	1	1	1	1	0	0	1	0	1	1	0	1	0	0	0	-1	-1	0	-1	-1	0
colorrandomizerspark2.aia	Aut.	0	0	1	-1	-1	2	1	-1	1	1	-1	-1	0	1	0	-1	1	-1	1	0	2	0	-1	-1	-1	-1	-1	-1
	Man.	0	0	1	-1	-1	2	1	-1	1	1	-1	-1	0	1	0	-1	1	-1	1	0	2	0	-1	-1	-1	-1	-1	-1
DASAPaintItTemplate.aia	Aut.	0	1	0	-1	1	2	-1	-1	1	1	-1	-1	1	-1	-1	-1	1	-1	1	0	-1	0	-1	0	1	0	-1	-1
	Man.	0	1	0	-1	1	2	-1	-1	1	1	-1	-1	1	-1	-1	-1	1	-1	1	0	-1	0	-1	0	1	0	-1	-1

6. Avaliação da confiabilidade e de validade

Foram realizadas duas avaliações na rubrica "CodeMaster UI Design - App Inventor" implementada. Uma avaliação sobre a confiabilidade e outra sobre a validade de construto. As avaliações foram feitas em parceria com o bolsista de iniciação científica João Vitor Araújo Porto e o mestrando Igor da Silva Solecki (SOLECKI, 2019) da iniciativa Computação na Escola do GQS/InCod/INE/UFSC.

6.1 Definição e Execução da Validação

O objetivo deste estudo é analisar a rubrica "CodeMaster UI Design - App Inventor" em termos de confiabilidade e validade de construto para a verificação da conformidade com os guias de design visual no contexto do ensino de computação na Educação Básica. Para avaliar a confiabilidade da rubrica, a seguinte questão de análise foi elaborada:

Q1: Há evidência de consistência interna na rubrica CodeMaster UI Design - App Inventor?

A validade de construto é analisada por meio da validade convergente e discriminante (TROCHIM, DONNELLY, 2008). A validade convergente testa se os conceitos ou medidas que deveriam supostamente estar relacionados estão de fato relacionados. Validade discriminante diz respeito ao grau em que dois critérios de fatores de qualidade que não devem estar relacionados estão de fato não relacionados (TROCHIM, DONNELLY, 2008). Assim, a seguinte questão de análise foi elaborada:

Q2: Há evidência de validade convergente e discriminante na rubrica CodeMaster UI Design - App Inventor?

Para realizar as análises, foram baixados 978 aplicativos disponíveis publicamente na Galeria do App Inventor (MIT, 2019) em maio de 2018. Esses aplicativos foram avaliados automaticamente utilizando a ferramenta CodeMaster. Os resultados das avaliações foram exportados para uma planilha xlsx com os níveis de desempenho alcançados em cada critério da rubrica "CodeMaster UI Design - App Inventor".

6.2 Análise dos dados

Usando os resultados das avaliações, foram realizadas análises estatísticas para responder às questões de análise. O procedimento seguido nesta etapa foi o proposto por DeVellis (2003).

6.2.1 Há evidência de consistência interna na rubrica CodeMaster UI Design - App Inventor?

A confiabilidade da rubrica “CodeMaster UI Design - App Inventor” foi analisada medindo sua consistência interna por meio do coeficiente alfa de Cronbach (1951). O coeficiente alfa de Cronbach (α) indica o grau em que um conjunto de critérios mede um único fator de qualidade (CRONBACH, 1951). No caso do trabalho proposto, o fator de qualidade consiste na conformidade do design de interface com as diretrizes de design visual.

O coeficiente alfa de Cronbach consiste em um valor entre 0 e 1. Existe um consenso de que os valores de alfa de Cronbach no intervalo de 0,70 a 0,95 são considerados aceitáveis, ou seja, indica consistência interna (DEVELLIS, 2003). Analisando os 28 critérios da rubrica “CodeMaster UI Design - App Inventor”, foi obtido α = 0,71. Sendo considerado um valor satisfatório por estar no intervalo de 0,70 a 0,95.

6.2.2 Há evidências de validades convergente e discriminante na rubrica CodeMaster UI Design - App Inventor?

Para obter evidências de validades convergente e discriminante dos critérios da rubrica de design do CodeMaster, foram calculadas as intercorrelações entre critérios e a correlação critério-total (WOHLIN et al. 2012).

Intercorrelações entre critérios

A intercorrelação entre critérios baseia na organização de critérios em dimensões (TROCHIM, DONNELLY, 2008) (CARMINES, ZELLER, 1982). As dimensões neste caso são as categorias (e.g., tipografia, cor, etc.) em que a rubrica está organizada (Capítulo 4.2). Espera-se que os critérios da mesma dimensão tenham uma alta correlação, para que a rubrica tenha validade convergente (TROCHIM, DONNELLY, 2008) (CARMINES, ZELLER, 1982). Por outro lado, para se obter evidência de validade discriminante, espera-se que critérios de diferentes dimensões tenham baixa correlação.

Para analisar as intercorrelações entre critérios de uma mesma dimensão foi usada a correlação policórica (Tabela 31), pois é considerada a mais apropriada para variáveis ordinais observadas (OLSSON, 1979). A Tabela 31 mostra o coeficiente de correlação, indicando o grau de correlação entre dois critérios ordinais (pares de critérios). Os coeficientes de correlação entre os critérios dentro da mesma dimensão estão coloridos.

Tabela 31: Resultados da análise de correlação policórica

		Layout						Tipografia					Escrita								Cores				Imagens				
		L1 - dimensionamentoResponsivo	L2 - telasComOrganizadores	L3 - tamanhoDeAlvosDeToque	L4 - formatoDosBotoes	L5 - tamanhoConsistenteDeBotoes	L6 - densidadeDaIU	T1 - familiaFonte	T2 - tamanhoFonteBotoes	T3 - tamanhoDaFonte	T4 - naoUsaItalico	T5 - botaoComTextoCentralizado	E1 - caixaAltaBotao	E2 - capitalizacaoSentenca	E3 - textoDefault	E4 - dicaPreenchidaEDiferenteDefault	E5 - doisPontosLegenda	E6 - pontoFinalSentenca	E7 - tamanhoMaximoTextoBotao	E8 - exclamacaoMultipla	C1 - sistemaDeCores	C2 - contrasteEntreTextoEFundo	C3 - coresMaterialDesign	C4 - coresHarmonicas	I1 - iconesMD	I2 - pixelizacao	I3 - imagensEmBotoes	I4 - componenteImagem	I5 - imagensDeFundo
Layout	L1 - dimensionamentoResponsivo	1.00																											
	L2 - telasComOrganizadores	0.09	1.00																										
	L3 - tamanhoDeAlvosDeToque	-0.29	0.06	1.00																									
	L4 - formatoDosBotoes	-0.11	**0.30**	0.24	1.00																								
	L5 - tamanhoConsistenteDeBotoes	-0.02	0.11	0.25	0.18	1.00																							
	L6 - densidadeDaIU	-0.16	**0.37**	**0.40**	**0.30**	**0.39**	1.00																						
Tipografia	T1 - familiaFonte	0.03	**0.34**	**0.33**	**0.54**	0.15	**0.44**	1.00																					
	T2 - tamanhoFonteBotoes	-0.23	0.14	0.28	**0.65**	0.18	0.29	**0.57**	1.00																				
	T3 - tamanhoDaFonte	-0.06	0.25	0.08	0.27	0.19	**0.36**	**0.42**	**0.46**	1.00																			
	T4 - naoUsaItalico	-0.16	**0.36**	**0.36**	**0.50**	**0.38**	**0.52**	**0.65**	**0.48**	**0.39**	1.00																		
	T5 - botaoComTextoCentralizado	-0.04	0.27	0.26	**0.80**	0.04	0.17	**0.62**	**0.75**	0.24	**0.49**	1.00																	
Escrita	E1 - caixaAltaBotao	0.20	0.13	0.07	**0.32**	-0.14	-0.17	**0.30**	0.02	-0.11	0.17	**0.56**	1.00																
	E2 - capitalizacaoSentenca	0.03	-0.07	-0.17	-0.05	-0.04	-0.06	-0.28	-0.23	-0.15	-0.08	-0.22	0.20	1.00															
	E3 - textoDefault	0.13	**0.38**	0.15	**0.55**	0.28	**0.48**	**0.53**	**0.49**	**0.38**	**0.50**	**0.47**	-0.04	-0.12	1.00														
	E4 - dicaPreenchidaEDiferenteDefault	0.27	0.14	-0.03	0.15	0.01	0.14	0.01	0.20	0.19	0.13	**0.30**	0.06	0.01	**0.38**	1.00													
	E5 - doisPontosLegenda	0.12	0.13	0.03	-0.03	0.09	0.13	**0.36**	-0.12	0.02	0.25	0.08	0.21	-0.13	**0.30**	0.12	1.00												
	E6 - pontoFinalSentenca	0.11	-0.03	-0.06	-0.02	-0.12	-0.05	-0.04	0.01	0.02	0.08	0.00	0.14	0.11	-0.12	-0.18	-0.22	1.00											
	E7 - tamanhoMaximoTextoBotao	-0.09	0.28	0.15	**0.64**	0.23	**0.29**	**0.42**	**0.53**	0.18	**0.48**	**0.74**	**0.41**	-0.15	**0.47**	0.08	-0.02	0.17	1.00										
	E8 - exclamacaoMultipla	0.01	0.21	-0.17	0.06	0.17	**0.32**	0.08	0.02	0.08	0.00	-0.10	-0.06	0.21	0.05	-0.03	-0.09	**0.33**	0.15	1.00									
Cores	C1 - sistemaDeCores	0.19	0.16	0.00	0.18	0.11	0.11	**0.32**	-0.12	0.03	0.25	0.06	0.09	-0.01	0.22	-0.14	0.23	0.00	0.00	0.16	1.00								
	C2 - contrasteEntreTextoEFundo	-0.17	0.14	**0.31**	0.28	**0.49**	**0.46**	**0.50**	**0.40**	**0.31**	**0.52**	0.23	-0.17	-0.13	**0.48**	0.04	0.04	-0.01	**0.30**	0.05	-0.06	1.00							
	C3 - coresMaterialDesign	0.09	0.05	-0.01	0.19	-0.04	-0.07	**0.50**	-0.05	-0.06	0.17	0.15	**0.35**	0.01	-0.09	-0.55	0.25	-0.16	0.07	-0.04	**0.63**	-0.06	1.00						
	C4 - coresHarmonicas	0.12	0.15	0.03	0.17	0.16	0.20	0.22	-0.07	0.11	**0.29**	0.05	-0.04	-0.06	0.15	-0.24	0.15	0.03	0.02	0.09	**0.90**	0.01	**0.57**	1.00					
Imagens	I1 - iconesMD	**0.42**	0.17	-0.50	0.12	0.13	0.09	0.05	-0.19	0.02	0.09	-0.11	0.02	**0.50**	**0.38**	0.18	0.21	-0.17	0.03	**0.34**	0.50	-0.08	0.26	0.36	1.00				
	I2 - pixelizacao	0.21	0.04	0.00	-0.04	0.00	0.03	0.18	-0.08	-0.14	0.13	-0.01	-0.05	-0.01	0.27	0.09	0.12	-0.12	-0.10	0.01	0.16	0.01	0.05	0.06	0.29	1.00			
	I3 - imagensEmBotoes	0.18	0.03	0.01	-0.21	0.12	0.08	0.04	-0.25	-0.15	0.19	-0.26	-0.14	0.04	0.18	0.06	0.10	-0.14	-0.17	0.04	0.22	0.10	0.00	0.08	**0.36**	**0.76**	1.00		
	I4 - componenteImagem	0.03	-0.08	-0.15	0.12	-0.20	-0.17	0.08	0.01	-0.08	0.02	0.15	0.17	-0.03	0.28	0.15	0.21	-0.22	-0.18	-0.11	0.05	-0.15	-0.02	-0.02	0.12	**0.46**	0.09	1.00	
	I5 - imagensDeFundo	0.12	-0.01	0.01	-0.24	-0.16	-0.06	-0.15	-0.12	-0.13	-0.19	-0.14	-0.02	-0.13	-0.04	0.11	-0.08	**0.60**	-0.17	**0.45**	-0.09	-0.11	-0.45	-0.17	-0.23	0.26	0.17	0.25	1.00

De acordo com Cohen (1998), uma correlação entre os critérios é considerada satisfatória se o coeficiente de correlação for maior que 0,29, indicando que há uma correlação média ou alta entre os critérios. Na Tabela 31, os resultados em negrito são os que apresentam correlações satisfatórias, ou seja, maior que 0,29.

A correlação entre os critérios da dimensão Layout não é satisfatória entre todos. O critério L6 tem correlação com quase todos os outros critérios dessa categoria. Enquanto o critério L1 tem baixa correlação com todos os critérios dessa dimensão. O critério L1 demonstra boa correlação apenas com o critério I1, que é de outra dimensão.

Os critérios que pertencem à dimensão Tipografia mostram um grau aceitável de correlação entre si. Especialmente os critérios T2 e T5 apresentam um valor de correlação muito alto (0,75). Apenas um par (critérios T3 e T5) apresentou 0,24, sendo considerada uma correlação insatisfatória.

Algumas correlações entre os critérios da dimensão Escrita são satisfatória. Porém, outras demonstram correlações negativa, o que é muito ruim. Também se observa que o critério E3 tem alta correlação com todos os itens de tipografia. Isso indica que quando o desenvolvedor do aplicativo se preocupa em trocar o nome, também se preocupa com todos os aspectos de Tipografia.

A maioria dos critérios da dimensão Cores apresentam alta correlação entre si. Sendo a melhor correlação entre os critérios C1 e C4. As correlações do critério C2 com os demais critérios dessa dimensão são ruins. Enquanto o critério C2 apresenta boas correlações com critérios de outras dimensões (Layout, Tipografia e Escrita). Assim, é possível que esse critério se enquadre melhor na categoria Tipografia sendo relacionado a questões da cor de texto.

No que se refere à dimensão Imagens, somente o critério I5 apresenta baixa correlação com outros critérios da mesma dimensão. Mas esse critério (I5) apresenta alta correlação com dois critérios da Tipografia. A correlação mais alta dentre os critérios da dimensão Imagens ocorre com os critérios I3 e I2.

Como vários critérios de diferentes dimensões mostram alto grau de correlação, a rubrica não apresenta validade discriminante. Também não é possível garantir a validade convergente, pois existem correlações entre critérios da mesma dimensão com grau baixo. Essas baixas correlações entre os critérios da mesma dimensão indicam que os critérios não estão medindo o mesmo fator dimensional. Consequentemente, a validade da conformidade do design visual das interfaces dos aplicativos do App Inventor com as diretrizes não é totalmente garantida pela rubrica. Portanto, esses critérios precisam ser revisados em análises futuras e potencialmente reagrupados ou eliminados da rubrica.

Correlação critério-total

A correlação critério-total consiste em avaliar a correlação de um critério com todos os outros critérios (DEVELLIS, 2003). Para essa análise, o método da correlação critério-total corrigida foi utilizado. Esse método compara um critério com todos os outros critérios da rubrica, exceto com ele mesmo (Tabela 32). Espera-se que haja uma correlação média ou alta, pois isso indica que os critérios apresentam consistência em comparação com os outros critérios. Para isso o coeficiente de correlação precisa ser maior que 0,29 (COHEN, 1998). Caso resulte em baixa correlação (menor que 0,29) significa que o critério enfraquece a validade da rubrica e, portanto, deve ser eliminada.

Além disso, foi analisado o alfa de Cronbach da rubrica excluindo apenas um dos critério. A expectativa é de que nenhum critério cause uma diminuição significativa no alfa de Cronbach (DEVELLIS, 2003)(Tabela 32).

Tabela 32: Resultados da análise de correlação item-total e alfa de Cronbach.

Item	Correlação item-total	Alfa de Cronbach com item removido
L1. dimensionamentoResponsivo	-0.01	0.71
L2. telasComOrganizadores	**0.30**	0.70
L3. tamanhoDeAlvosDeToque	0.25	0.70
L4. formatoDosBotoes	**0.49**	0.69
L5. tamanhoConsistenteDeBotoes	0.27	0.70
L6. densidadeDaIU	**0.44**	0.68
T1. familiaFonte	**0.58**	0.68
T2. tamanhoFonteBotoes	**0.38**	0.69
T3. tamanhoDaFonte	0.28	0.70
T4. naoUsaItalico	**0.56**	0.68
T5. botaoComTextoCentralizado	**0.47**	0.69
E1. caixaAltaBotao	0.09	0.71
E2. capitalizacaoSentenca	-0.13	**0.73**
E3. textoDefault	**0.56**	0.69
E4. dicaPreenchidaEDiferenteDefault	0.08	0.71
E5. doisPontosLegenda	0.14	0.71
E6. pontoFinalSentenca	-0.03	0.72
E7. tamanhoMaximoTextoBotao	**0.37**	0.69
E8. exclamacaoMultipla	0.07	0.71
C1. sistemaDeCores	0.23	0.71
C2. contrasteEntreTextoEFundo	**0.36**	0.70
C3. coresMaterialDesign	0.08	0.71
C4. coresHarmonicas	0.28	0.70
I1. iconesMD	0.07	0.71
I2. pixelizacao	0.10	0.71
I3. imagensEmBotoes	0.06	0.71
I4. componenteImagem	-0.01	0.72
I5. imagensDeFundo	-0.02	0.71

O valor do alfa de Cronbach mostra um pequeno aumento somente para o item E2. Os demais itens não apresentam aumento do alfa da rubrica quando excluídos, indicando que contribuem para a avaliação de forma geral.

Em relação à correlação item-total, apenas 10 critérios de um total de 28 critérios apresenta correlação acima de 0,29. No entanto, alguns itens apresentam uma correlação baixa, como I3 e I1 ou até mesmo negativa, como I4 e E2. Isso

indica que esses critérios não estão significativamente correlacionados com os demais critérios da rubrica e possivelmente devem ser reformulados ou até mesmo excluídos.

6.3 Discussão

Os resultados obtidos, em geral, indicam confiabilidade aceitável e validade de construto minimamente suficiente da rubrica “CodeMaster UI Design - App Inventor” para a avaliação da conformidade do design de IU de aplicativos Android com guias de estilo e de acessibilidade. No entanto, a análise também aponta a necessidade de revisão e reestruturação da rubrica em relação a algumas questões. As baixas correlações entre critérios de uma mesma dimensão podem indicar que os critérios não estão medindo o mesmo fator, sendo necessária a reorganização desses critérios nas diferentes dimensões. Além disso, as análises indicam que alguns critérios têm baixa correlação com os demais critérios da rubrica, sendo necessária a reformulação ou até mesmo exclusão dos mesmos da rubrica. Em decorrência desses resultados, é necessário revisar os critérios por meio de análises estatísticas adicionais com amostras de dados maiores.

No entanto, em geral, os resultados da análise mostram que a rubrica “CodeMaster UI Design - App Inventor” representa um instrumento confiável. Isso indica que a rubrica pode ser usada para uma avaliação válida do design da interface do usuário dos aplicativos do App Inventor em relação à conformidade com as diretrizes de design visual. No entanto, é importante salientar que a rubrica representa apenas uma alternativa para avaliar o design da IU e que uma avaliação mais abrangente deve ser completada por outros métodos de avaliação, como entrevistas, revisões por pares, testes de usabilidade, etc.

Ameaças à validade

Este trabalho está sujeito a várias ameaças à validade. Portanto, foram identificadas ameaças potenciais e estratégias de mitigação foram aplicadas para minimizar seu impacto na pesquisa proposta. A fim de mitigar as ameaças relacionadas ao design do estudo, foram definidos e documentos uma metodologia sistemática usando a abordagem GQM (BASILI, CALDIERA, ROMBAC, 1994).

Outra ameaça refere-se à padronização da avaliação do dados de cada aplicativo. Como o estudo é limitado exclusivamente à avaliação da rubrica “CodeMaster UI Design - App Inventor” de forma automatizada, esse risco é eliminado.

Outra questão refere-se ao agrupamento de dados de diferentes contextos. Os aplicativos da amostra analisada foram coletados da Galeria do App Inventor e

nenhuma informação sobre o nível de conhecimento dos criadores dos aplicativos é disponibilizado. Ou seja, aplicativos desenvolvidos por inexperiente ou por profissionais são avaliados igualmente. No entanto, como o objetivo é analisar a validade da rubrica de maneira independente do contexto, isso não é considerado algo a ser tratado neste momento.

A escolha dos métodos estatísticos para a análise dos dados sempre pode representar ameaças à pesquisa. Para minimizar essa ameaça, foi realizada uma análise estatística baseada na abordagem para a construção de escalas de medida conforme proposta por DeVellis (2003), que está alinhada com procedimentos para avaliação da consistência interna e validade de construto de instrumentos de medida.

7. Conclusão

O objetivo principal do presente trabalho foi de desenvolver um módulo de avaliação automatizada do design de interface de aplicativos desenvolvidos com o App Inventor. E incluir esse módulo na ferramenta web CodeMaster. Com a intenção de ser utilizada em unidades instrucionais no ensino de computação na Educação Básica. Inicialmente foi realizada a análise da fundamentação teórica e diretrizes que definissem meios de verificar a qualidade dos designs (O1). Também foi analisado o estado da arte por meio de um mapeamento sistemático sobre ferramentas que fazem análise de design de interface de aplicativos para *smartphones touchscreen* da plataforma Android ou desenvolvidos na ferramenta App Inventor (O2). Foram analisados vários trabalhos e identificou-se a ausência de ferramentas que avaliassem automaticamente o design de interface de aplicativos no contexto educacional. A rubrica "CodeMaster UI Design - App Inventor" foi desenvolvida de acordo com os meta-princípios da linguagem visual (SCHLATTER; LEVINSON, 2013), com as diretrizes WCAG 2.0 (2008) e *Material Design* (2018). Também foi desenvolvida a implementação da avaliação automatizada da rubrica "CodeMaster UI Design - App Inventor", que foi incluída na ferramenta CodeMaster (O3). A ferramenta foi testada conforme a corretude, para assegurar o possível que a implementação feita corresponde ao proposto pela rubrica. Com a intenção de avaliar a confiabilidade e a validade da rubrica "CodeMaster UI Design - App Inventor" foram calculados o coeficiente alfa de Cronbach (1951), as intercorrelações entre critérios e a correlação critério-total (O4). Os resultados foram satisfatórios e disponibilizaram informações para auxiliar na aprimoração da rubrica.

Em termos de impacto científico é criado um conhecimento inovador em relação a automatização de avaliação de design de interface de aplicativos Android no contexto do ensino de desenvolvimento de aplicativos móveis no Ensino Básico. É criado também como resultado tecnológico um software inovador que permite a análise do design de interface de apps de forma automatizada. Em termos sociais, com a disponibilidade de um modelo conceitual desses, espera-se contribuir com a melhoria do ensino de computação inserido nas escolas Brasileiras em relação a aprendizagem de computação e a popularização deste conhecimento na sociedade. Espera-se também pela introdução do design de interface no Ensino Básico motivar mais alunos a procurarem como formação superior cursos da área de computação e combatendo dessa forma as altas taxas atuais de evasão nesses cursos.

Como trabalhos futuros, sugere-se aprimorar o desempenho da avaliação de aplicativos pelo CodeMaster, para reduzir o tempo de espera da execução da avaliação. Uma solução possível seria paralelizar a execução da avaliação das rubricas "CodeMaster CT - App Inventor" e "CodeMaster UI Design - App Inventor". Da mesma forma que há o estado "não se aplica" para quando o aplicativo não tenha o componente avaliado pelo critério, é proposto adicionar o estado "não avaliado". O estado "não avaliado" seria correspondido aos critérios em que o professor solicitou que não fossem avaliado. Consequentemente, não apresentar a coluna do critério "não avaliado" no feedback dado ao professor. Recomenda-se também utilizar os dados resultantes da avaliação de confiabilidade e de validade

como base para realizar aprimorações na rubrica "CodeMaster UI Design - App Inventor”.

Referências Bibliográficas

ABRAN A.; KHELIFI A.; SURYN W.; SEFFAH A. **Consolidating the ISO Usability Models**. In: Proceedings of the 11th International Software Quality Management Conference and 8th Annual INSPIRE Conference (2003)

ADAMS, J. C.. WEBSTER, A. R. **What do students learn about programming from game, music video, and storytelling projects?** In: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, Raleigh, NC, USA, 2012.

AHO, A. V.; SETHI, R.; ULLMAN, J. D. **Compilers, Principles, Techniques**. Boston: Addison Wesley. 2ª ed. 1986.

ALLEN INTERACTIONS Disposnível em: <https://www.alleninteractions.com/sam-process>. Acessado em: novembro de 2017.

ALVES, N. C. **Modelo de Análise de Código de Linguagens de Programação Visuais no Ensino Básico**. Dissertação (Mestrado em Ciência da Computação) - Curso de Pós-graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, 2019.

ALVES, N. D. C.; RODRIGUES, P. E., BORGATTO, A. F., GRESSE VON WANGENHEIM, C., HAUCk J. C. R. **Ensino de Computação de Forma Multidisciplinar em Disciplinas de História no Ensino Fundamental - Um Estudo de Caso**. Revista Brasileira de Informática na Educação, vol. 24, no. 3, 2017, p. 31-46.

ANDROID. Qualidade do aplicativo principal. Disponível em: <https://developer.android.com/docs/quality-guidelines/core-app-quality#ux>. Acessado em: Junho de 2018.

App Inventor, About us.. Disponível em: <http://appinventor.mit.edu/explore/about-us.html>. Acessado em fevereiro de 2018.

BASILI, V. R., CALDIERA G., ROMBACH, H. D. **Goal Question Metric Paradigm**. In: Encyclopedia of Software Engineering. John Wiley & Sons, vol 1, 1994, p. 528-532.

BASTIEN, J. C.; SCAPIN, D. L. Ergonomic criteria for the evaluation of human-computer interfaces. Technical report N° 156, Inria Rocquencourt, França 1993.

BENNANI, S.; IDRISSI, M. K.; FADOULI, N.; YASSINE, B. T.; OUGUENGAY, Y. A. **Online project based learning driven by competencies: A systematic strategy proposal for assessment**. In: Proceedings of the Int. Conf. on 255 Interactive Mobile and Computer Aided Learning, Amman, Jordan, 2012, p. 92-97.

BENFORD, S.D., BURKE, E.K., FOXLEY, E., HIGGINS, C. **The Ceilidh system for the automatic grading of students on programming courses**. In: Proceedings of the 33rd ACM-SE Annual on Southeast Regional Conference, Clemson, SC , USA, p. 176 – 182. 1995.

BIAGIOTTI, L. C. M. **Conhecendo e Aplicando Rubricas Em Avaliações**. In: Proceedings of the 12º Congresso Internacional de Educação a Distância, Florianópolis, Brasil, 2005.

BLACK, P., BURKHARDT; H., DARO; P., LAPPAN, G.; PEAD, D.; STEPHENSON, M. High-stakes Examinations that Support Student Learning: Recommendations for the design, development and implementation of the SBAC assessments: International Society for Design and Development in Education Working Group on Examinations and Policy. 2011.

BLACK, P., WILIAN, D. **Assessment and classroom learning**. Assessment in Education: Principles, Policy & Practice. vol. 5 n.1. 1998. p. 7–74.

BLOCKLY, 2018. Disponível em: https://developers.google.com/blockly/. Acesso em: fevereiro de 2018.

BRANCH, R. M. **Instructional Design: The ADDIE Approach**. New York: Springer, 2009.

BROWN, W.; YEN, P.-Y.; ROJAS, M.; SCHNALL, R. **Assessment of the Health IT Usability Evaluation Model (Health-ITUEM) for evaluating mobile health (mHealth) technology**. Journal Biomed Inform, vol. 46, 2013, p. 1080–1087.

BRUNER, J. S. **The Process of Education.** Harvard University Press. 1977. Disponível em: http://edci770.pbworks.com/w/file/fetch/45494576/Bruner_Processes_of_Education.pdf. Acessado em: Fevereiro de 2018.

CARMINES, E. G. ; ZELLER, R. A. **Reliability and validity assessment** (5th ed.). Beverly Hills: Sage Publications Inc. 1982.

CLARK, L.A., WATSON, D.B. **Constructing validity: Basic issues in objective scale development**. Psychological Assessment, vol. 7, 1995, p. 309-319.

CME Florianópolis. RESOLUÇÃO CME Nº02/2011. Conselho Municipal de Educação de Florianópolis. 2011.

CodeHS. Disponível em: https://codehs.com/info/curriculum. Acessado em: março de 2018.

Code.org. Disponível em: <https://code.org/>. Acessado em: outubro de 2017.

CODE. **Computer Science Fundamentals Courses A-F**. 2017. Disponível em: https://code.org/files/CSF_CoursesA-F_Curriculum_Guide.pdf Acessado em: Fevereiro de 2018.

CODEMASTER, About us. Disponível em: <http://apps.computacaonaescola.ufsc.br:8080/sobre.jsp>. Acessado em outubro de 2019.

COHEN, J. **Statistical Power Analysis for the Behavioral Sciences**. Routledge Academic. 1998.

COMPUTAÇÃO NA ESCOLA. Disponível em: <http://www.computacaonaescola.ufsc.br/> Acessado em: maio de 2019.

CRONBACH, L. J. **Coefficient alpha and the internal structure of tests**. Psychometrika, vol 16, n. 3, 1951.

CS FIRST. Disponível em: <https://www.cs-first.com/en/home>. Acessado em: outubro de 2017.

CSTA. ACM. **CSTA K –12 Computer Science Standards**, 2016. Disponível em: <http://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf>. Acesso em: outubro de 2017.

DANIEL, G. T; GRESSE VON WANGENHEIM, C.; MEDEIROS, G. A. S.; ALVES, N. C. **Ensinando a Computação por meio de Programação com App Inventor**., In: Proceedings of the VIII Computer on the Beach. Florianópolis, Brasil. 2017.

DELUCA, C.; KLINGER, D. A. Assessment literacy development: identifying gaps in teacher candidates' learning. Assessment in Education: Principles, Policy & Practice, vol 17, n. 4, 2010, p. 419-438.

DEMETRIO, M. F. **Desenvolvimento de um analisador e avaliador de código de App Inventor para ensino de computação**. 2017. Trabalho de Conclusão do Curso de Bacharel em Ciências da Computação da Universidade Federal de Santa Catarina, Florianópolis, Brasil.

DEVELLIS, R. F. **Scale development: theory and applications**. SAGE Publications. 2003.

Diretrizes de educação. Disponível em: <https://developers.google.com/edu/guidelines>. Acessado em: Junho de 2018.

DR. SCRATCH. Disponível em: <http://www.drscratch.org/>. Acessado em: outubro de 2017.

DUDA, R.; SILVA, S. C. R.. **Desenvolvimento de Aplicativos para android com uso do app inventor: uso de novas tecnologias no processo de ensino-aprendizagem em matemática.** Revista Conexão, vol.11, n.3, 2015, p. 310-323.

ESERYEL, D.; IFENTHALER, D.; Xun, G. **Validation study of a method for assessing complex ill-structured problem solving by using causal representations**. Educational Technology Research. vol. 61, 2013, p. 443–463.

ERTMER, P. A.; NEWBY, T. J. **Behaviorism, Cognitivism, Constructivism: Comparing Critical Features From an Instructional Design Perspective**. Performance Improvement Quarterly, vol.6 n.4, 1993, p. 50–72.

FEE, S. B.; HOLLAND-MINKLEY, A. M. **Teaching Computer Science through Problems, not Solutions**. Computer Science Education, v. 2, 2010, p. 129-144.

Ferreira, Miriam Nathalie Fortuna. **Design de Interface para Apps: uma Unidade Ensino no Contexto do Ensino Fundamental**. Mestrado, Curso de Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis. Em andamento. 2019.

FERREIRA, M. N. F.; PINHEIRO, F.; MISSFELDT FILHO, R; GRESSE VON WANGENHEIM, G.; GONÇALVES, B. S. **Ensinando Design de Interface de Usuário na Educação Básica: Um Mapeamento Sistemático do Estado da Arte e Prática**. Relatório Técnico. INCoD/GQS.09.2018.P. Outubro 2018.

FORSYTHE, G. E.; WIRTH, N. **Automatic Grading Programs**. Communications of the ACM, vol. 8, n. 5, 1965, p. 275–278.

GAGNÉ, R. M.; BRIGGS, L. J. E; WAGER, W. W. **Principles of Instruction Design**. Orlando: Harcourt Brace & Company. 4ª ed, 1992.

GARRISON, C.; EHRINGHAUS, M. **Formative and Summative Assessments in the Classroom**. Association for Middle Level Education, Westerville, OH, EUA, 2013.

GOHN, M. **Educação não-formal, participação da sociedade civil e estruturas colegiadas nas escolas**. Rio de Janeiro, Brasil. vol.14, n. 50, 2006, p. 27-38.

GOODRICH, H. **Understanding Rubrics**. Educational Leadership, vol. 54, n. 4, 1996, p. 14–18.

Google. Introduction - Material Design. Disponível em: <https://material.io/guidelines/>. Acesso em: abril de 2018.

GRESSE VON WANGENHEIM, C.; Witt, A. T.; Borgatto, A. F.; Nunes, J. V.; Lacerda, T. C.; Krone, C.; Souza, L. O. **An Usability Score for Mobile Phone**

Applications based on Heuristics. International Journal of Mobile Human Computer Interaction. vol 8. n. 1. 2016. p 23-58.

GRESSE VON WANGENHEIM, C.; HAUCK, J. C. R.; DEMETRIO, M. F.; PELLE, R.; ALVES, N. C.; BARBOSA, H.; AZEVEDO, L. F. **CodeMaster v1.0 - An Overview**. Relatório Técnico, NOVEMBRO - 2017. Disponível em: <http://www.incod.ufsc.br/wp-content/uploads/2017/04/INCoD-RelatorioTecnicos-CodeMaster-v10E.pdf>. Acessado em: dezembro de 2017.

GRESSE VON WANGENHEIM, C.; HAUCK, J. C. R.; DEMETRIO, M. F. ; PELLE, R.; ALVES, N. C.; BARBOSA, H.; AZEVEDO, L. F. **CodeMaster – Automatic Assessment and Grading of App Inventor and Snap! Programs**. Informatics in Education, vol. 17, n. 1, 2018, p. 117–150.

GRESSE VON WANGENHEIM, C.; ALVES, N. C; WEBER, A. R. **Resumo do K-12 Computer Science Standards (Versão 2017)**. Relatório Técnico INCoD/GQS.04.2017.P., INCoD, INE, UFSC, Florianópolis, Brasil, 2017.

HADDAWAY, N. R., COLLINS, A. M., COUGHLIN, D., KIRK, S. **The Role of Google Scholar in Evidence Reviews and Its Applicability to Grey Literature Searching**. PLOS ONE, v. 10, n. 9, set. 2015.

HATTIE, J.; TIMPERLEY, H. **The power of feedback**. Review of educational research, vol. 77, n.1, 2007, p. 81–112.

HERRERA, S.G.; MURRY, K. G.; CABRAL, R.M. **Assessment accommodations for classroom teachers of culturally and linguistically diverse students**. Boston, MA: Pearson Education Inc. 2007.

HOEHLE, H.; ALJAFARI, R.; Venkatesh, V. **Leveraging Microsoft's mobile usability guidelines: Conceptualizing and developing scales for mobile application usability**. International Journal of Human-Computer Studies. vol 89, 2016, p.35-53.

HOUR OF CODE. Disponível em: <https://hourofcode.com>. Acessado em: outubro de 2017.

HUSSAIN, A.; KUTAR, M.; MUTALIB, A.A.; KAMAL, F. M. **Modeling Subjective Metrics for Mobile Evaluation**. Journal of Research and Innovation in Information Systems, vol. 1, 2012, p. 11-20.

HURWITZ, A.; DAY, M. **Children and their art: Methods for the elementary school**. 8 ed. Belmont, CA: Thomson Wadsworth, 2007. Disponível em: <https://ia601608.us.archive.org/23/items/childrentheirart00gait_0/childrentheirart00gait_0.pdf>. Acessado em: dezembro de 2017.

IHANTOLA, P.; AHONIEMI, T.; KARAVIRTA, V.; SEPPÄLÄ, O. **Review of recent systems for automatic assessment of programming assignments**. Proceedings

of the 10th Koli Calling International Conference on Computing Education Research, Koli, Finland, 2010, p. 86–93.

INES, G.; MAKRAM, S.; MABROUKA, C.; MOURAD, A. **Evaluation of Mobile Interfaces as an Optimization Problem**. Procedia Computer Science vol 112, 2017, p. 235-248.

INEP, 2010-2017. Instituto Nacional De Estudos E Pesquisas Educacionais Anísio Teixeira. Sinopses Estatísticas da Educação Superior. Disponível em: <http://portal.inep.gov.br/web/guest/sinopses-estatisticas-da-educacao-superior>. Acessado em: Abril de 2019.

Interaction Design Foundation. **User Interface (UI) Design - What is User Interface (UI) Design?** Disponível em: <https://www.interaction-design.org/literature/topics/ui-design>. Acessado em: maio de 2018.

iOS Developer Library iOS Human Interface Guidelines. Dsiponível em: <https://developer.apple.com/design/human-interface-guidelines/>. Acessado em: Junho 2018.

ISO 9241–210, International Standard: Ergonomics of Human–System Interaction – Part 210: Human-Centred Design for Interactive Systems. 2010.

ISO/IEC25010. ISO/IEC 25010. 2014. Disponível em: <http://iso25000.com/index.php/en/iso-25000-standards/iso-25010?limitstart=0>. Acessado em: maio de 2018.

ITU - International Telecommunication Union. **The World in 2011 – Facts and Figures 2011**. 2011.

JERSEY, RESTful Web Services in Java. 2017. Disponível em: <https://jersey.github.io/>. Acessado em: Maio de 2019.

JONASSEN, D. H. **Instructional design models for well-structured and ill-structured problem-solving learning outcomes.** Educational Technology Research and Development, vol. 45. p. 65-90. 1997.

KELLEHER, C.; PAUSCH, R. **Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers**. ACM Computing Surveys, vol. 37, n.2, 2005, p. 83–137.

KRAIGER, K., FORD, J., SALAS, E. **Application of cognitive, skill-based, and affective theories of learning outcomes to new methods of training evaluation**. 1993. Journal of Applied Psychology, vol. 78, p. 311-328.

KRETLOW, A., BARTHOLOMEW, C. **Using coaching to improve the fidelity of evidence-based practices: A review of studies**. Teacher Education and Special Education, vol 33, n.4, 2010. p. 279–299.

KROSNICK, J.A., PRESSER, S. **Question and Questionnaire Design**. In: Wright, J. D., Marsden, P. V. (eds.) Handbook of Survey Research (2nd ed.). San Diego: Elsevier. 2009.

LARMAN, C.; BASILI, V. R. **Iterative and Incremental Development: A Brief History**. Computer, vol. 36, n.6, 2003, p. 47-56.

Lei de Diretrizes e Bases da Educação Nacional - Lei Nº 9.394. BRASÍLIA, BRASIL. 1996.

LINN, R. L.; MILLER, M. D. **Measurement and assessment in teaching**. Upper Saddle River: Prentice Hall. 9ª ed. 2005.

LORD, F.M. **Applications of item response theory to practical testing problems**. Mahwah: Lawrence Erlbaum Associates, Inc. 1980.

LYE, S. Y.; KOH, J. H. L. **Review on teaching and learning of computational thinking through programming: What is next for K-12?**. Computers in Human Behavior, vol. 41, 2014, p. 51-61.

MICHAELIS. 2017. Disponível em: <http://michaelis.uol.com.br/>. Acessado em: dezembro de 2017.

MACQUARIE UNIVERSITY AUSTRÁLIA. **Evaluation: Assessing Student Achievement Of Learning Outcomes**. Disponível em: https://staff.mq.edu.au/teaching/evaluation/resources_evaluation/developing_unit/assess_achievement/. Acessado em: Fevereiro de 2018.

GOOGLE, Material Design Google, 2017. Disponível em: <https://material.io/>. Acessado em: outubro de 2017.

MARCONDES, M. E. R. et al. **Materiais instrucionais numa perspectiva CTSA: uma análise de unidades didáticas produzidas por professores de química em formação continuada**. Investigações em Ensino de Ciências, vol. 14, n. 2, 2009.

Ministério da Educação. **Diretrizes Curriculares Nacionais para Educação Básica**. Brasília, 2013. Disponível em: http://portal.mec.gov.br/docman/julho-2013-pdf/13677-diretrizes-educacao-basica-2013-pdf/file. Acessado em Fevereiro de 2018.

MISSFELDT FILHO, R. **DESENVOLVIMENTO DE UMA UNIDADE INSTRUCIONAL PARA ENSINAR O DESENVOLVIMENTO DE APPS NO ENSINO FUNDAMENTAL COM O APP INVENTOR**. Monografia (Trabalho de Conclusão de Curso em Ciência da Computação) - Curso de Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, 2018.

MCCAULEY, R. **Rubrics as Assessment Guides.** Newsletter ACM SIGCSE Bulletin, vol. 35, n.4, p. 17-18, 2003

MILES, M.B., HUBERMAN, A.M. **Qualitative Data Analysis: A Sourcebook of New Methods**. 2nd edition. Thousand Oaks: Sage Publications. 1994.

Ministério da Educação. **Base Nacional Comum Curricular**. 2018. Dísponivel em: http://basenacionalcomum.mec.gov.br/images/BNCC_EI_EF_110518_versaofinal_site.pdf. Acessado em: Junho de 2019.

MIT. "Join the App Inventor Community Gallery!", https://appinventor.mit.edu/explore/blogs/shay/2013/04/join-app-inventor-community-gallery.html. Acessado em: Maio de 2019.

MORRIS, A. **Student Standardised Testing: Current Practices in OECD Countries and a Literature Review**, OECD Education Working Papers, n.65, OECD Publishing, Paris. 2011.

MORRISON, G. R.; ROSS, S. M.; KEMP, J. E.; KALMAN, H. **Designing effective instruction**. John Wiley & Sons. 6 ed. 2010.

MYSQL. Disponível em: <https://www.mysql.com/>. Acessado em: Maio de 2019.

NGO, D. C. L.; TEO, L. S.; BYRNE, J. G. **Formalising guidelines for the design of screen layouts**. Displays, vol. 21, n. 1, 2000, p. 3- 15.

NILSSON, E. G. **Design patterns for user interface for mobile applications**. Adv Eng Softw, vol. 40, 2009, p. 1318–1328.

NICOL, D.; MCFARLANE-DICK, D. **Formative assessment and self-regulated learning: A model and seven principles of good feedback practice**. Studies in higher education, vol. 31, n.2, 2006, p. 199–218.

NIELSEN, J. **Usability engineering**. Boston: AP Professional, 2ª ed, 1993, p. 362.

OLIVEIRA, M. L. S.; Souza, A. A.; Barbosa, A. F.; Barreiros, E. F. S. **Ensino de lógica de programação no ensino fundamental utilizando o Scratch: um relato de experiência**. In: Anais do XXII Workshop sobre Educação em Computação no Congresso a Sociedade Brasileira da Computação, Brasília, Brasil, 2014.

OLSSON, U. **Maximum likelihood estimation of the polychoric correlation coefficient**. Psychometrika, 44(4). 1979.

OpenCV Library. Disponível em: <https://opencv.org/>. Acessado em: junho de 2018.

OTA, G.; MORIMOTO, Y.; KATO, H.. **Ninja code village for scratch: Function samples/function analyser and automatic assessment of computational**

thinking concepts. Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, Cambridge, UK. 2016.

PASTERNAK, E.; FENICHEL, R.; MARSHALL, A. N. **Tips for Creating a Block Language with Blockly**. IEEE Blocks and Beyond Workshop. Raleigh, NC, USA. 2017.

Paiva, F. **Panorama Mobile Time/Opinion Box - Crianças e smartphones no Brasil**. Outubro de 2018. Disponivel em: https://panoramamobiletime.com.br/. Acessado em: Abril de 2019.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. **Systematic Mapping Studies in Software Engineering**. In: Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering, Bari, Italy, 2008, p. 68-77.

PHYE, G., BENDER, T. **Feedback complexity and practice: Response pattern analysis in retention and transfer**. Contemporary Educational Psychology, vol. 14, n.2, 1989, p. 97–110.

PINHEIRO, FERNANDO da C. **Modelo instrucional para o ensino de Engenharia de Software e Usabilidade voltado à Educação Básica**. Dissertação de Mestrado, Curso de Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis. Em andamento. 2019.

PORTO, J. V. A.; Gresse VON WANGENHEIM, C.; BARBOSA, H. **Heurísticas de usabilidade de aplicativos Android desenvolvidos com App Inventor: Uma Revisão Sistemática da Literatura.** Relatório Técnico INCoD/GQS.07.2017.P, UFSC, Florianópolis, Brasil, 2017.

PRICE, T. W.; BARNES, T. **Position paper: Block-based programming should offer intelligent support for learners**. Blocks and Beyond Workshop. Raleigh, NC, USA. 2017.

Qualidade do aplicativo para tablet. Disponivel em: <https://developer.android.com/docs/quality-guidelines/tablet-app-quality>. Acessado em: junho de 2018.

RASHKOVITS, R; LAVY, I. **FACT: A Formative Assessment Criteria Tool for the Assessment of Students' Programming Tasks**. In: Proceedings of the World Congress on Engineering. Londres. UK. 2013.

RESNICK, M., MALONEY, J., MONROY-HERNANDEZ, A., RUSK, N., EASTMOND, E., BRENNAN, K. **Scratch: Programming for all**. Communications of the ACM, vol. 52, n.11, 2009, p. 60–67.

RODRIGUEZ, A. **Restful web services: The basics**. IBM developerWorks, 2008.

ROGERS, Y.; SHARP, H; PREECE, J. **Design de Interação: Além da Interação Homem-Computador**. Bookman, 3ª ed. 2013.

SASKATCHEWAN EDUCATION. **Instructional Approaches: A Framework for Professional Practice.** Canada: Saskatchewan Education, 1991. Disponível em: <https://wikieducator.org/images/e/e2/Instructional-Approaches_Handbook.pdf>. Acessado em: Maio de 2018.

SAVERY, J. R. **Overview of problem-based learning: Definitions and distinctions**. Interdisciplinary Journal of Problem-based Learning. vol 1. 2006.

SBC, **Plano de Gestão para a SBC Biênio Agosto 2015 - Julho 2017**. 2015. Disponível em: http://www.sbc.org.br/documentos-da-sbc/send/135-eleicoes/999-plano-de-gestao-para-a-sbc-bienio-agosto-2015-julho-2017 Acessado em: Fevereiro de 2018.

SCHLATTER, T.; LEVINSON, D. **Visual usability**: Principles and practices for designing digital applications. Newnes, 2013.

Scratch. Disponível em: https://scratch.mit.edu/. Acessado em: Fevereiro de 2018.

SEFFAH, A.; DONYAEE, M.; KLINE, R. B.; PADDA, H. K. **Usability measurement and metrics: A consolidated model**. Software Quality Journal, vol. 14, 2006, p.159–178.

SHERMAN, M.; F. MARTIN, M.; BALDWIN,L.; DEFILIPPO, J. **App Inventor Project Rubric — Computational Thinking through Mobile Computing**. 2014. Disponível em: <http://nsfmobilect.wordpress.com/evaluation/>. Acessado em: junho de 2018.

SHERMAN, M.; MARTIN, F. **The Assessment of Mobile Computational Thinking**. Journal of Computing Sciences in Colleges. vol 30 n.6, 2015. p. 53-59.

SHUTE, V. **Focus on formative feedback**. Princeton: Educational Testing Service (ETS). 2007.

SINTHSIRIMANA, S.; PANJABUREE, P. **A Development of Computer Programming Analyzer: A Case Study on PHP Programming Language Institute for Innovative Learning**. Journal of Industrial and Intelligent Information. vol. 1, n. 3, 2013.

Sociedade Brasileira de Computação. **Referenciais de Formação em Computação: Educação Básica**. 2017. Dísponível em: http://www.sbc.org.br/files/ComputacaoEducacaoBasica-versaofinal-julho2017.pdf. Acessado em: Fevereiro de 2018.

SOLECKI, I. S.. **Desenvolvimento de uma abordagem para avaliação de qualidade de código de linguagens de programação baseadas em blocos**.

Dissertação de Mestrado, Curso de Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis. Em andamento. 2019.

SNAP!. Disponível em:https://snap.berkeley.edu/. Acesso em: Fevereiro de 2018.

STRAUSS, A.; CORBIN, J. **Basics of Qualitative Research: Grounded Theory Procedures and Techniques**. Newbury Park: Sage Publications.1990.

STRIEWE, M.; GOEDICKE, M. **A review of static analysis approaches for programming exercises**. Communications in Computer and Information Science. vol. 439, 2014, p. 100-113.

TECHNOVATION, 2017. Disponível em: <http://technovationchallenge.org/> Acessado em: outubro de 2017.

TECHNOVATION. Teacher and Mentor Lesson Guide Lessons 1 - 6. Fev. 2014. Disponível em:<https://technovationchallenge.org/wp-content/uploads/2014/01/TeacherMentorLessonGuide_L1_6.pdf>. Acessado em: Junho de 2018.

THEODORIDIS, S., KOUTROUMBAS, K. **Pattern Recognition**. 4 ed. 2009.

THURLINGS, M., VERMEULEN, M., BASTIAENS, T., & STIJNEN, S. **Understanding feedback: A learning theory perspective**. Educational Research Review, vol. 9, n.1. 2013. p. 1-15.

TIC EDUCAÇÃO. **Marco Referencial Metodológico para a Medição do Acesso e Uso das Tecnologias de Informação e Comunicação (TIC) na Educação**. Centro Regional de Estudos para o Desenvolvimento da Sociedade da Informação. 2016. Disponível em: <http://cetic.br/media/docs/publicacoes/8/marco-referencial-metodologico-para-a-medicao-do-acesso-e-uso-das-tecnologias-de-informacao-e-comunicacao-na-educacao.pdf>. Acessado em: outubro de 2017.

TROCHIM, W. M. K.; DONNELLY, J. P. **Research methods knowledge base** (3rd ed.). Mason, OH: Atomic Dog Publishing. 2008.

W3C. **Mobile Web Application Best Practices**. 2010. Disponível em: <http://www.w3.org/TR/mwabp/>. Acessado em: Junho de 2018.

WAGNER, A.; GRAY, J.; CORLEY, J.; WOLBER, D. **Using App Inventor in a K-12 Summer Camp**. In: Proceeding of the 44th ACM technical symposium on Computer science education, Denver, CO, USA. 2013. p. 621-626.

WARD, J.D.; LEE, C.L. **A review of problem-based learning**. Journal of Family and Consumer Sciences Education, vol. 20, n.1,2002, p. 16-26.

WASSERMAN, A. I. **Software Engineering Issues for Mobile Application Development**. In: Proceedings of the FSE/SDP workshop on Future of software engineering research, Santa Fe, NM, , USA. 2010, p. 397-400.

WAZLAWICK, R. S. **Análise e Design Orientados a Objetos para Sistemas de Informação: Modelagem com UML, OCL e IFML**. Editora Campus. 3ª edição. 2016.

Web Content Accessibility Guidelines (WCAG) 2.0. W3C Recommendation 11 December 2008. Disponível em: <https://www.w3.org/TR/2008/REC-WCAG20-20081211/>. Acessado em: Abril de 2019.

WILCOX, C. **Testing Strategies for the Automated Grading of Student Programs**. In: Proc. of ACM Technical Symposium on Computing Science Education, Memphis, TN, USA, 2016, p. 437-442.

WIGGINS, G. P. **The Jossey-Bass education series. Assessing student performance: Exploring the purpose and limits of testing**. San Francisco: Jossey-Bass. 1993.

WING, J. M. **Computational Thinking**. Communications of the ACM, vol. 49, n. 3, 2006, p. 33-36.

WOHLIN, C.; Runeson, P.; Höst, M.; Ohlsson, M. C.; Regnell, B.; Wesslén, A. **Experimentation in Software Engineering**. Berlin: Springer, 235. 2012.

ZAPATA, B. C.; NIÑIROLA, A. H.; IDRI, A.; FERNÁNDEZ-ALEMÁN, J. L.; TOVAL, A. **Mobile PHRs Compliance with Android and iOS Usability Guidelines**. Journal of Medical Systems. vol. 38, n. 81. 2014.

Zapata, B. C.; Fernández-Alemán, J. L.; Idri, A; Toval, A. **Empirical Studies on Usability of mHealth Apps: A Systematic Literature Review**. Journal of Medical Systems, vol 399, n. 1, 2015.

ZEN, K.; ISKANDAR, D. N. F A., LINANG, O. **Using Latent Semantic Analysis for automated grading programming assignments.** In: Proc. of the International Conference on Semantic Technology and Information Retrieval, Putrajaya, Malaysia, 2011, p. 82 – 88.

Apêndice A

A ferramenta web App Inventor é organizada em duas grandes áreas: Designer e Blocos. A área Designer apresenta os recursos necessária para a construção do design de interface (como botões) e da seleção dos sensores (como acelerômetro). A área Blocos é utilizada para a programação da lógica, como condicionais, loops, verificação de estado de sensores, etc. Para construir uma lógica para o aplicativo reagir por alguma ação feita pelo usuário, é preciso escolher um bloco correspondente à ação que pretende executar.

App Inventor disponibiliza blocos genéricos, que são disponível a todo momento independente de quais componentes estejam incluído no design do app. No item **controle** são listados os comandos responsáveis pelas funções de controle da execução da aplicação, como laços, condicionais e iterações em listas (Tabela 1). No item **lógica** contém os componentes responsáveis pelas operações lógicas sobre as variáveis (Tabela 2). Em **matemática** são apresentados componentes responsáveis por operações matemáticas sobre variáveis (Tabela 3). No item **texto** Componentes responsáveis pela manipulação de textos, onde estão incluídas operações de junção de texto, verificação de partes de um texto, divisão de texto, entre outros (Tabela 4). No item **lista** tem os componentes responsáveis pela criação e manipulação de listas de dados, que são utilizados em aplicações com volume de informações é grande e se torna inviável armazenar em variáveis (Tabela 5). No item de **cores** há os componentes responsáveis por determinar as cores do aplicativo, podendo ser utilizadas cores padrões ou cores personalizadas pelo desenvolvedor (Tabela 6). Em **variáveis** contém os componentes responsáveis pela criação e manipulação das variáveis da aplicação. As variáveis podem ser definidas globalmente ou localmente (Tabela 7). Já em **procedimentos** tem os componentes responsáveis pelos procedimentos da aplicação, também são chamados na computação de métodos ou funções (Tabela 8).

Além dos blocos genéricos, o App Inventor disponibiliza um conjunto de blocos para cada componente que esteja no design do app. Por exemplo, caso na tela de design tenha um botão, então serão disponibilizados os blocos apresentado na Tabela 9.

Tabela 1: Blocos de Controle do App Inventor

Bloco	**Descrição**
if e if else	Testa uma condição dada. Se a condição for verdadeira, executa uma sequência de blocos, senão, ignora esta sequência de blocos.
For each – from – to	Executa os blocos na seção para cada valor numérico no intervalo - a partir de – e - terminando em -, incrementando o número a cada vez executado.
For each list item	Executa uma sequência de blocos para cada item de uma lista.
While	Testa uma condição repetidas vezes, enquanto for verdadeira executa uma sequência de blocos.
If then else	Testa uma condição, se for verdadeira, executa os blocos do then, se for falsa, executa os blocos do else.
Do	Executa um bloco e retorna um resultado.
evaluate but ignore result	Usado como bloco fictício, o bloco em que este esteja conectado será executado, mas seu retorno

	será ignorado.
open another screen	Abre uma tela com o nome dado
open another screen with start value	Abre uma tela com o nome e um valor passado.
get start value	Retorna o valor atual dado a uma tela
close screen	Fecha a tela atual.
close screen with value	Fecha a tela atual e retorna o valor passado a ela.
close application	Fecha a aplicação.
get plain start text	Retorna o texto que foi passado a tela quando foi iniciada por outro app.
close screen with plain text	Fecha a tela e passa o texto ao app que iniciou ela.

Tabela 2: Blocos de Lógica do App Inventor

Bloco	**Descrição**
True, false	Representa a constante com valor verdadeiro ou falso.
Not	Negação lógica.
Igualdade e Diferença	Testa se os argumentos são iguais ou diferentes.
And	Testa se todas as condições lógicas são verdadeiras.
Or	Testa se alguma condição lógica é verdadeira.

Tabela 3: Blocos de Matemática do App Inventor

Bloco	**Descrição**
Basic number block	Representa qualquer número positivo ou negativo.
=, ≠, >, ≥, <, ≤	Blocos com as operações de igualdade ou desigualdade.
min, max	Retorna o maior ou menor valor.
sqrt, abs, -, log, e^, round, ceiling, floor	Operações unárias padrões.
modulo of, remainder of, quotient if	Operações de módulo, resto e quociente.
sin, cos, tan, asin, acos, atan	Operações trigonométricas.
convert radians to degrees, convert degrees to radians	Conversão de graus em radianos e radianos em graus.
+, -, *, /	Operações de soma, subtração, multiplicação e divisão.
Integer random	Retorna número inteiro entre valores passados.
Random fraction	Retorna uma fração randômica
Random set seed to	Define a semente para a geração de números randômicos.
Format as decimal	Formata um número decimal em uma quantidade de casas decimais definidas
Is a number?	Retorna verdadeiro se o objeto passado é um número.
Conver number	Converte um número em binário, hexadecimal ou decimal

Tabela 4: Blocos de Texto do App Inventor

Bloco	**Descrição**
String “”	Contém um texto (string)
Join	Concatena strings.
length	Retorna quantidade de caracteres da string.
Is empty	Retorna verdadeiro se a string não contém caracteres.
compare texts < > =	Compara strings, se é “maior” ou “menor” alfabeticamente, ou string igual.
Trim	Remove qualquer espaço que exista na string e retorna.
Upcase	Retorna a string passada formatada em letras maiúsculas.
Downcase	Retorna a string passada formatada em letras minúsculas.
Starts at	Retorna a posição em que começa a string passada em um texto.
Contains	Retorna true se uma parte aparece no texto.
Split at first	Divide o texto em duas partes usando a localização da primeira ocorrência do ponto de divisão.
Split at first of any	Divide o texto em dois itens de lista.
Segment	Retorna o trecho do texto de início e tamanho passados por parâmetro.
Replace all	Retorna um novo texto em que foram substituídas todas as ocorrências de determinada substring.
Obfuscated text	o texto inserido neste bloco é tratado como secreto, tendo um pouco de segurança para distribuir a informação.
is a string?	retorna se a variável for uma palavra ou algum texto.

Tabela 5: Blocos de Lista do App Inventor

Bloco	**Descrição**
create empty list	Cria uma lista vazia.
make a list	Cria uma lista a partir dos blocos definidos, se não definir blocos, cria uma lista vazia.
add items to list	Adiciona itens ao final da lista.
is in list?	Retorna verdadeiro se o elemento está na lista.
length of list	Retorna a quantidade de itens na lista.
is list empty?	Retorna verdadeiro se a lista está vazia.
pick a random item	Obtém um item qualquer da lista
index in list	Retorna a posição que determinado item está na lista.
select list item	Seleciona o item que está em determinada posição na lista.
insert list item	Insere item na lista na posição informada.

replace list item	Substitui item da lista por novo na posição informada.
remove list item	Remove item da lista na posição informada.
append to list	Adiciona itens da segunda lista no final da primeira.
copy list	Faz uma cópia da lista incluindo sublistas.
is a list?	Retorna verdadeiro se o objeto é uma lista.
list to csv row	Interpreta a lista como uma linha e retorna um arquivo CSV.
list from csv row	Captura um texto em CSV formatado em uma linha e cria uma lista.
list to csv table	Interpreta a lista como uma tabela e retorna um arquivo CSV.
list from csv table	Captura um texto em CSV formatado em uma tabela e cria uma lista.
lookup in pairs	Usado para pesquisar informações em uma estrutura em pares parecida com um dicionário.

Tabela 6: Blocos de Cores do App Inventor

Bloco	**Descrição**
basic color blocks	Bloco de cores básicas.
Make color	Recebe uma lista de 3 ou 4 valores que representam o valor RGB ou RGBA da cor.
Split color	Retorna uma lista com os valores do RGB.

Tabela 7: Blocos de Variáveis do App Inventor

Bloco	**Descrição**
initialize global name to	Usado para criar uma variável global definindo um nome.
get	Uma forma de obter qualquer variável que tenha sido criada no escopo atual.
set to	Uma forma de atribuir um valor a uma variável criada.
initialize Local name to - in (do)	Permite a criação de uma variável que será usada apenas no DO de um método.
initialize Local name to - in (return)	Permite a criação de uma variável que será usada apenas no RETURN de um método.

Tabela 8: Blocos de Procedimentos do App Inventor

Bloco	**Descrição**
procedure do	Agrupa uma sequência de blocos e não retorna uma resposta ao final da sua execução.
procedure result	Agrupa uma sequência de blocos e retorna uma resposta ao final da sua execução.

Tabela 9: Blocos relacionados ao componente Botão do App Inventor

Blocos	**Descrição**
when click do	Agrupa um conjunto de ações para realizar quando o botão for pressionado e solto.
when got focus do	Quando o cursor passou por cima do botão é habilitado o botão para ser pressionado.
when long click do	Agrupa um conjunto de ações para realizar quando o botão estiver sendo pressionado.
when touch up do	Agrupa um conjunto de ações para realizar quando o botão for solto.

when touch down do	Agrupa um conjunto de ações para realizar quando o botão for pressionado.

Apêndice B

Neste apêndice são apresentados como cada componente de interface pode ser customizado na ferramenta web App Inventor.

Os componente do Tipo Mapas, Sensores, Armazenamento, Conectividade que não são de interface ou não possuem propriedades para customização.

Tabela 01: Opções para Customizar o Componente Tela (ref)

Opções	**Descrição**
TelaSobre	Informações sobre a tela. Aparece quando "Sobre este aplicativo" é selecionado no menu do sistema. Use-o para informar os usuários sobre seu aplicativo. Em vários aplicativos de tela, cada tela tem suas próprias informações.
Accent Color	Essa é a cor de destaque usada para realces. Os componentes afetados por essa propriedade incluem diálogos criados pelo Notificador, o DateTimePicker e outros.
Alinhamento Horizontal	Número que codifica como o conteúdo da tela é alinhado horizontalmente. As opções são: 1 = alinhado à esquerda, 2 = centrado na horizontal, 3 = alinhado à direita.
Alinhamento Vertical	Um número que codifica como o conteúdo do arranjo é alinhado verticalmente. As opções são: 1 = alinhado no topo, 2 = verticalmente centrado, 3 = alinhado no fundo. O alinhamento vertical não tem efeito se a tela for rolável.
Nome do App	Este é o nome de exibição do aplicativo instalado no telefone. Se o nome do aplicativo estiver em branco, ele será definido para o nome do projeto quando o projeto é criado.
Cor de Fundo	Cor apresentada no fundo da tela.
Imagem de Fundo	Se definida uma imagem de fundo, ela será apresentada no fundo da tela.
Animação Fechamento De Tela	A animação para fechar a tela atual e retornar à tela anterior. As opções válidas são default, fade, zoom, slider horizontal, slider vertical e nenhum.
Icone	O ícone apresentado como atalho para acessar o aplicativo quando instalado no telefone
Animação Abertura De Tela	A animação para mudar para outra tela. Opções válidas são padrão, fade, zoom, slide horizontal, slide vertical e none.
Cor Primária	Essa é a cor principal usada como parte do tema Android, incluindo a cor da barra de título da tela.
Cor Primária Escura	This is the primary color used when the Theme property is specified to be Dark. It applies to a number of elements, including the screen's title bar.
Orientação Da Tela	A orientação da tela solicitada, especificada como um valor de texto. Os valores mais usados são paisagem, retrato, sensor, usuário e não especificado. Consulte a documentação do desenvolvedor do Android para ActivityInfo.Screen_Orientation para obter a lista completa de configurações possíveis.
Rolável	Quando marcada, haverá uma barra de rolagem vertical na tela e a altura do aplicativo pode exceder a altura física do dispositivo. Quando desmarcada, a altura do aplicativo é restrita à altura do dispositivo.
Mostrar Lista como Json	Se false, as listas serão convertidas em strings usando notação Lisp, ou seja, como símbolos separados por espaços, por exemplo, (a 1 b2 (cd). Se true, as listas aparecerão como em Json ou Python, por exemplo ["a", 1 , "b", 2, ["c", "d"]]. Esta propriedade aparece apenas na Tela 1, e o valor da Tela 1 determina o comportamento de todas as telas. A propriedade é padronizada como "false", o que significa que o aplicativo App Inventor deve defini-lo explicitamente como "true" se a sintaxe JSON / Python for desejada. Em algum momento no futuro, alteraremos o sistema para que novos projetos sejam criados com essa propriedade definida como "true" por padrão. O programador do App Inventor também pode defini-lo de volta para "false" nos projetos mais novos, se desejado.
Mostrar Barra De Estado	A barra de status é a barra superior na tela. Esta propriedade informa se a barra de status está visível.
Dimensionamento	Se definido como fixo, os layouts de tela serão criados para uma única tela de tamanho fixo e com escalonamento automático. Se definido como responsivo, os layouts de tela usarão a resolução real do dispositivo. Consulte a documentação sobre design responsivo no App Inventor para obter mais informações. Essa propriedade aparece somente na tela 1 e controla o tamanho de todas as telas no aplicativo.
Tema	Seleciona o tema para o aplicativo. O tema só pode ser definido em tempo de compilação e o Companion irá aproximar as alterações durante o desenvolvimento ao vivo. As opções possíveis são o Classic, que é o mesmo

	que versões anteriores do App Inventor; Device Default, que dá o mesmo tema da versão do Android rodando no dispositivo e usa PrimaryColor para o ActinBar e possui botões de luz; Texto de Título em Preto, que é o tema Padrão do Dispositivo, mas com texto de título em preto, e Escuro, que é uma versão escura do tema Padrão do Dispositivo usando PrimaryColorDark e tendo componentes cinza escuro.
Título	A legenda do *Form*, que aparece na barra de título
Título Visivel	A barra de título é a barra cinza superior na tela. Esta propriedade informa se a barra de título está visível.
Tutorial URL	Um URL que será aberto no painel do lado esquerdo (que pode ser alternado quando estiver aberto). Isso é destinado a projetos que possuem um tutorial in-line como parte do projeto. Por motivos de segurança, somente os tutoriais hospedados em http://appinventor.mit.edu ou vinculados a partir do nosso encurtador de URL (http://appinv.us) podem ser usados aqui. Outras URLs serão ignoradas silenciosamente.
Versão Do Código	Um valor inteiro que deve ser incrementado sempre que um novo arquivo de pacote de aplicativos Android (APK) for criado para a Google Play Store.
Nome Da Versão	Uma string que pode ser alterada para permitir que os usuários da Google Play Store façam a distinção entre diferentes versões do aplicativo.

Tabela 02: Opções para Customizar Componentes do tipo Interface do Usuário

	Botão	Caixa de Seleção	Escolhe Data	Imagem	Legenda	Escolhe Lista	Visualizador de Listas	Notificador	Caixa de Senha	Deslizador	Lista Suspensa	Caixa de Texto	Escolhe Hora	Navegador Web
Cor de Fundo	x	x	x		x	x	x	x	x			x	x	
Ativado	x	x	x			x			x			x	x	
Fonte Negrito	x	x	x		x	x			x			x	x	
Fonte Itálico	x	x	x		x	x			x			x	x	
Tamanho da Fonte	x	x	x		x	x	x		x			x	x	
Família da Fonte	x	x	x		x	x			x			x	x	
Altura	x	x	x	x	x	x	x		x			x	x	x
Largura	x	x	x	x	x	x	x		x	x	x	x	x	x
Imagem	x		x	x		x							x	
Forma	x		x			x							x	
Mostrar *Feedback*	x		x			x							x	
Texto	x	x	x		x	x			x			x	x	
Alinhamento do Texto	x		x		x	x			x			x	x	
Cor do Texto	x	x	x		x	x	x	x	x			x	x	
Visível	x	x	x	x	x				x	x	x	x	x	x
Marcado		x												
Ângulo de Rotação				x										
Redimensionar para Caber				x										
HTML Formato					x									

Tem Margem					x									
Cadeia de Elementos						x	x				x			
Cor de Fundo do Item						x								
Seleção						x	x				x			
Mostrar barra de filtragem						x	x							
Título						x								
Cor de Seleção							x							
Tamanho do Texto							x							
Tamanho do Notificador								x						
Dica									x			x		
Cor à Esquerda										x				
Cor à Direita										x				
Valor Máximo										x				
Valor Minimo										x				
Indicador Ativado										x				
Posição do Indicador										x				
Mesangem											x			
MultiLinha												x		
SomenteNúmeros												x		
Seguir Links														x
UrlInicial														x
IgnorarErrosSsl														x
Pedir Autorização														x
UsaLocalização														x

Tabela 03: Opções para Customizar Componentes do tipo Organização

	Organização Horizontal	Arranjo de Rolamento Horizontal	Organização em Tabela	Organização Vertical	Arranjo de Rolamento Vertical
Alinhamento Horizontal	x	x		x	x
Alinhamento Vertical	x	x		x	x
Cor de Fundo	x	x		x	x
Altura	x	x	x	x	x
Largura	x	x	x	x	x
Imagem	x	x		x	x

Visível	x	x	x	x	x
Colunas			x		
Linhas			x		

Dentres os componentes de Organização (Tabela 04) tem alguns que não possuem propriedades para customização, que são CâmeradeVídeo, Câmera, ReconhecedorDeVoz e Tradutor Yandex.

Tabela 04: Opções para Customizar Componentes do tipo Mídia

	Escolhe Imagem	Tocador	Som	Gravador	Texto para Falar	Reprodutor de Vídeo
Cor de Fundo	x					
Ativado	x					
Fonte Negrito	x					
Fonte Itálico	x					
Tamanho da Fonte	x					
Família da Fonte	x					
Altura	x				x	x
Largura	x					x
Imagem	x					
Forma	x					
Mostrar Feedback	x					
Texto	x					
Alinhamento do Texto	x					
Cor do Texto	x					
Visível	x					x
Repetir		x				
TocarSomenteEmPrimeiroPlano		x				
Fonte		x	x			x
Volume		x				x
Intervalo			x			
Gravação Salva				x		
País					x	
Idioma					x	
Velocidade de Fala					x	

Tabela 5: Opções para Customizar Componentes do tipo Desenho e Animação

	Bola	Pintura	SpriteImagem
Ativado	x		x
Direção	x		x
Intervalo	x		x
Cor de Pintura	x	x	
Raio	x		
Velocidade	x		x
Visível	x	x	x
X	x		x
Y	x		x
Z	x		x
Cor de Fundo		x	
Imagem de Fundo		x	
Tamanho da Fonte		x	
Altura		x	x
Largura		x	x
Largura da Linha		x	
Alinhamento do Texto		x	
Imagem			x
Rodar			x

Dentres os componentes Social (Tabela 6) tem alguns que não possuem propriedades para customização ou nenhuma dela lidam com usabilidade, que são Ligação, Compartilhamento, Mensagens SMS.

Tabela 6: Opções para Customizar Componentes do tipo Social

	Escolhe Contato	Escolhe Email	Escolhe Número de Telefone
Cor de Fundo	x	x	x
Ativado	x	x	x
Fonte Negrito	x	x	x
Fonte Itálico	x	x	x
Tamanho da Fonte	x	x	x
Família da Fonte	x	x	x
Altura	x	x	x
Largura	x	x	x
Imagem	x		x

Forma	x		x
Mostrar *Feedback*	x		x
Texto	x	x	x
Alinhamento do Texto	x	x	x
Cor do Texto	x	x	x
Visível	x	x	x
Dica		x	

Apêndice C

Junto com a nota alcançada na rubrica “CodeMaster UI Design - App Inventor” e com os níveis de desempenhos alcançados em cada critério. No Banco de Dados (BD) também são registrados variáveis que são importantes para a definição do nível de desempenho alcançado.

A Tabela 1 apresenta as variáveis salvas em BD para cada critério e a descrição do que cada variável significa.

Tabela 1: Tabela com as variáveis salvas em BD

ID	Critério	Variável	Descrição
Layout			
L1	Dimensionamento responsivo	dimensionamentoResponsivo	apresenta pontuação alcançada no critério.
L2	Elementos de organização nas telas	qtdTelas	quantidade de telas que o app tem
		qtdTelasComOrganizadores	quantidade de telas que tem pelo menos um componente do tipo organizador
		telasComOrganizadores	apresenta pontuação alcançada no critério.
L3	Tamanho mínimo de componentes alvos de toque	tamanhoDeAlvosDeToque	Nota do critério
		menorLarguraDeAlvoDeToque	Menor largura de componentes alvos de toque
		menorAlturaDeAlvoDeToque	Menor altura de componentes alvos de toque
L4	Formato dos botões	formatoDosBotoes	Nota do critério
		qtdBotoesSemImagem	Quantidade de botões sem imagem no aplicativo
L5	Tamanho consistente de botões	tamanhoConsistenteDeBotoes	Nota do critério
		qtdAgrupamentosDeBotoes	Quantidade de agrupamentos de botões (2 ou mais botões diretamente na tela ou em um organizador)
		usoDeBotoesNaTela	Se botões diretamente na tela sempre são consistentes (-1: não há botões diretamente na tela; 0: não são; 1: sempre são)
		usoDeBotoesNaHorizontal	Se botões em organizadores horizontais sempre são consistentes (-1: não há botões em organizadores horizontais; 0: não são; 1: sempre são)
		usoDeBotoesNaVertical	Se botões em organizadores verticais sempre são consistentes (-1: não há botões em organizadores verticais; 0: não são; 1: sempre são)
		usoDeBotoesEmGrade	Se botões em organizadores em grade/tabela sempre são consistentes (-1: não há botões em organizadores em grade/tabela; 0: não são; 1: sempre são)
L6	Densidade de elementos nas telas	densidadeDaIU	Nota do critério
		maiorDensidadeIU	Densidade (quantidade de elementos) da tela com mais elementos
		menorDensidadeIU	Densidade (quantidade de elementos) da tela com menos elementos
		qtdTelasComDensidadeOK	Quantidade de telas com densidade média ou baixa (< 10)
		qtdTelasComDensidadeAlta	Quantidade de telas com densidade alta (de 10 a 19)
		qtdTelasComDensidadeMuitoAlta	Quantidade de telas com densidade muito alta (>= 20)
Tipografia			
T1	Família da fonte	qtdFontTypeface	quantidade de componente que tiver a propriedade Typeface e não tiver Texto vazio.
		qtdSansSerif	quantidade de componentes que tiver a propriedade Typeface default ou igual a um (1) e não tiver Texto vazio.
		familiaFonte	apresenta pontuação alcançada no critério.
T2	Tamanho da fonte de texto de	qtdBotaoComTexto	quantidade de botões (incluindo pickers) com a propriedade Texto preenchida.
		qtdFontBotaoTamanho14	quantidade de botões (incluindo pickers) com a propriedade Texto preenchida e tamanho da fonte igual a 14.

	botões	tamanhoFonteBotoes	apresenta pontuação alcançada no critério.
T3	Tamanho da fonte e Tamanho do texto de componentes CheckBox, Label, ListView, CaixaDeSenha, CaixaDeTexto, Pintura e EscolheEmail	qtdComponentesComTamFonte	quantidade de componentes CheckBox, Label e ListView com texto e CaixaDeSenha, CaixaDeTexto, Pintura e EscolheEmail
		qtdComponentesComTamFonteCorreto	quantidade de componentes com propriedade Tamanho da fonte ou Tamanho do texto que satisfaz o critério.
		tamanhoDaFonte	apresenta pontuação alcançada no critério.
T4	Não usar itálico	qtdSentencaPodeSerItalico	quantidade de componentes que tem a propriedade "FontItalic"
		qtdSentencaNaoItalico	quantidade de componentes que tem a propriedade "FontItalic" e não está itálico
		naoUsaItalico	pontuação alcançada no critério
T5	Botão Com Texto Centralizado	qtdBotaoComTextoCentralizado	quantidade de botões cujo texto está centralizado
		botaoComTextoCentralizado	pontuação alcançada no critério
Escrita			
E1	Capitalização do texto de botões	qtdBotaoComTodoTextoCaixaAlta	quantidade de botões no app cujo texto é todo em Caixa Alta.
		caixaAltaBotao	apresenta pontuação alcançada no critério.
E2	Capitalização de sentenças	qtdSentenca	quantidade de sentenças que o app tem, em diferentes propriedades de diferentes componentes.
		qtdSentencaComCapitalizacaoCorreta	quantidade de sentenças que iniciam com letra maiúscula ou dígito
		capitalizacaoSentenca	apresenta pontuação alcançada no critério.
E3	Componentes com propriedade Texto diferente do padrão, exceto os componentes EscolheEmail, CaixaDeTexto e CaixaDeSenha.	qtdComponentesComTexto	quantidade de componentes no app que contém a propriedade Texto e o Texto não está vazio.
		qtdCompComTextoIgualPadrao	quantidade de componentes no app com a propriedade Texto **igual** ao texto padrão gerado .automaticamente pelo AppInventor.
		textoDefault	apresenta pontuação alcançada no critério.
E4	Componentes com propriedade Dica preenchida e diferente de default	qtdCompComDicaIgualPadraoOuVazio	quantidade de componentes que tem a propriedade Dica e texto está igual ao padrão do App Inventor ou vazio.
		qtdComponentesComDica	quantidade de componentes que tem a propriedade Dica preenchida.
		dicaPreenchidaEDiferenteDefault	apresenta pontuação alcançada no critério.
E5	Não usa dois pontos ao final de legendas	qtdLegenda	quantidade de componente legenda no app que tem propriedade texto diferente de vazio.
		qtdLegendaSemDoisPontos	quantidade de componente legenda no app que tem propriedade texto diferente de vazio e não termina com dois pontos (:).
		doisPontosLegenda	apresenta pontuação alcançada no critério.
E6	Não usa ponto final para encerrar sentenças	qtdSentencaCriterioPontoFinal	quantidade de sentenças em todo app que pode ter sentenças (foi testado e há casos em que essa quantidade se diferencia da "qtdSentença" do critério 4)
		qtdSentencaSemPontoFinal	quantidade de sentenças que não termina com ponto final.
		pontoFinalSentenca	apresenta pontuação alcançada no critério.
E7	Quantidade de caracteres de texto do botão	maiorTextoDeBotaoEmCaracteres	tamanho do maior texto de botão do app em caracteres (contabiliza espaço)
		tamanhoMaximoTextoBotao	pontuação alcançada no critério
E8	Evitar múltiplos pontos de exclamação/interrogação	exclamacaoMultipla	Nota do critério
		qtdTextos	Quantidade de textos avaliados no critério
		qtdTextosComExclamacaoMultipla	Quantidade de textos avaliados no critério que têm exclamações/interrogações múltiplas
Cores			
C1	Sistema de cores	qtdCores	quantidade de cores (com distintas tonalidades) que o app tem sem contabilizar preto,

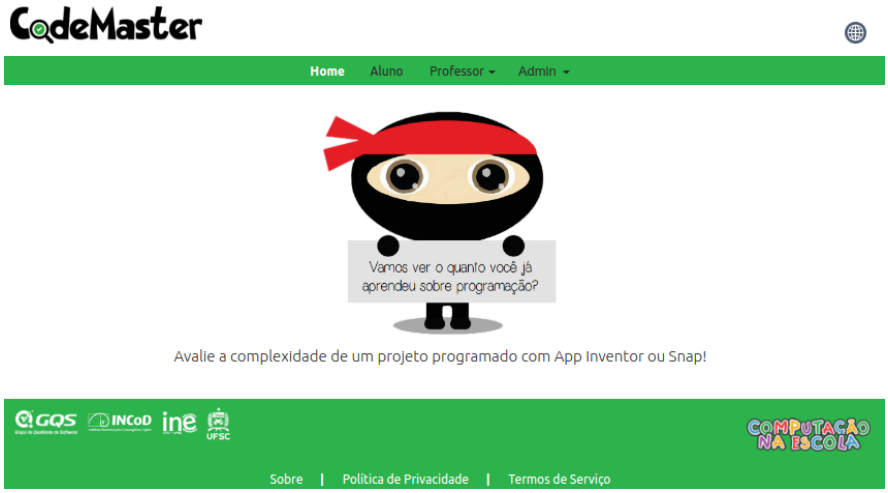
			branco e cinza
		sistemaDeCores	apresenta pontuação alcançada no critério.
C2	Contraste entre texto e fundo	contrasteEntreTextoEFundo	Nota do critério
C3	Cores do Material Design	qtdCoresNoApp	quantidade de cores que o aplicativo tem na fonte texto ou ao fundo do componente.
		qtdCoresIguaisAoMD	quantidade de cores que o aplicativo tem na fonte texto ou ao fundo do componente E que é igual ao recomendado pelo Material Design[16]
		coresMaterialDesign	pontuação alcançada no critério
C4	Harmonia das cores	coresHarmonicas	Nota do critério
		qtdCoresUnicas	Quantidade de cores, não considerando cores parecidas como a mesma
Imagens			
I1	Ícones recomendados pelo Material Design	qtdTotalIcones	quantidade de componentes considerados ícones
		qtdIconesDoMD	quantidade de ícones detectados como do Material Design
		iconesMD	pontuação alcançada no critério
I2	Pixelização de imagens	pixelizacao	pontuação alcançada no critério
		qtdTelasComFundo	quantidade de telas que têm imagem de fundo
		qtdTelasComFundoSemPixelizacao	quantidade de telas que têm imagem de fundo que não pixeliza
		qtdComponentesImagemComImagem	quantidade de componentes Imagem com imagem definida
		qtdComponentesImagemSemPixelizacao	quantidade de componentes Imagem cuja imagem não pixeliza
		qtdBotoesComImagem (repetida em Distorção de imagens em botões e pickers)	quantidade de botões ou pickers que têm imagem
		qtdBotoesComImagemSemPixelizacao	quantidade de botões ou pickers que têm imagem que não pixeliza
		maiorPixFundo	maior pixelização em imagens de fundo
		maiorPixCompImagem	maior pixelização em componentes Imagem
		maiorPixBotao	maior pixelização em botões
I3	Distorção de imagens em botões e pickers	imagensEmBotoes	pontuação alcançada no critério
		qtdBotoesComImagem (mesma do critério Pixelização)	quantidade de botões (incluindo pickers) que têm imagem
		qtdBotoesComImagemSemDistorcao	quantidade de botões (incluindo pickers) que têm imagem que não distorce
I4	Não faz redimensionamento do componente Imagem	qtdComponentesImagem	quantidade de componentes do tipo Imagem no aplicativo inteiro
		qtdImagensSemRedimensionar	quantidade de componentes do tipo Imagem sem RedimensionarParaCaber
		componenteImagem	pontuação alcançada no critério
I5	Imagem de fundo da tela	imagensDeFundo	pontuação alcançada no critério
		qtdTelasIrrestritaGrave	quantidade de telas sem restrição de orientação e com imagem de fundo com distorção sempre grave
		qtdTelasRestritaGrave	quantidade de telas com restrição de orientação e com imagem de fundo com distorção grave
		qtdTelasIrrestritas	quantidade de telas (com imagem de fundo) com orientação indefinida
		qtdTelasRestritas	quantidade de telas (com imagem de fundo) com orientação definida
		qtdTelasRestritaLeve	quantidade de telas com restrição de orientação, mas com imagem de fundo com distorção leve
		maiorDistIrrestrita	maior distorção em telas irrestritas (considerando sempre a orientação em que a imagem fica melhor)
		maiorDistRestrita	maior distorção em telas restritas

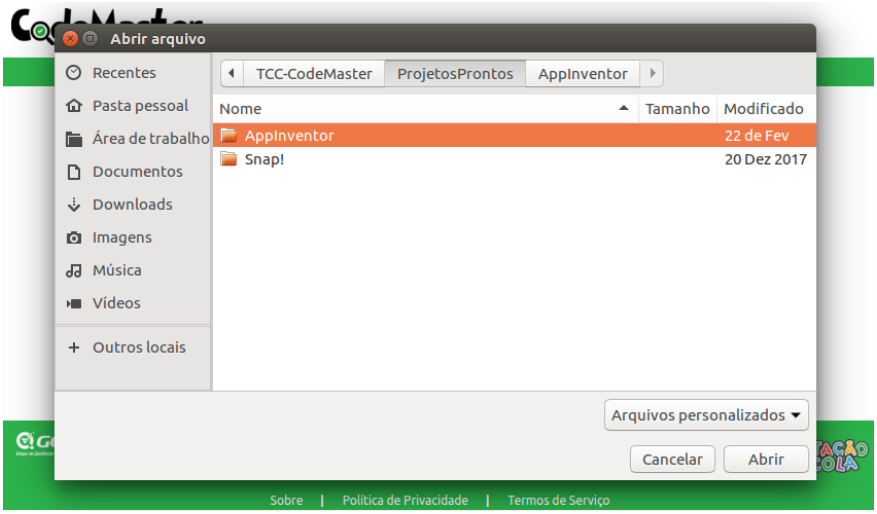
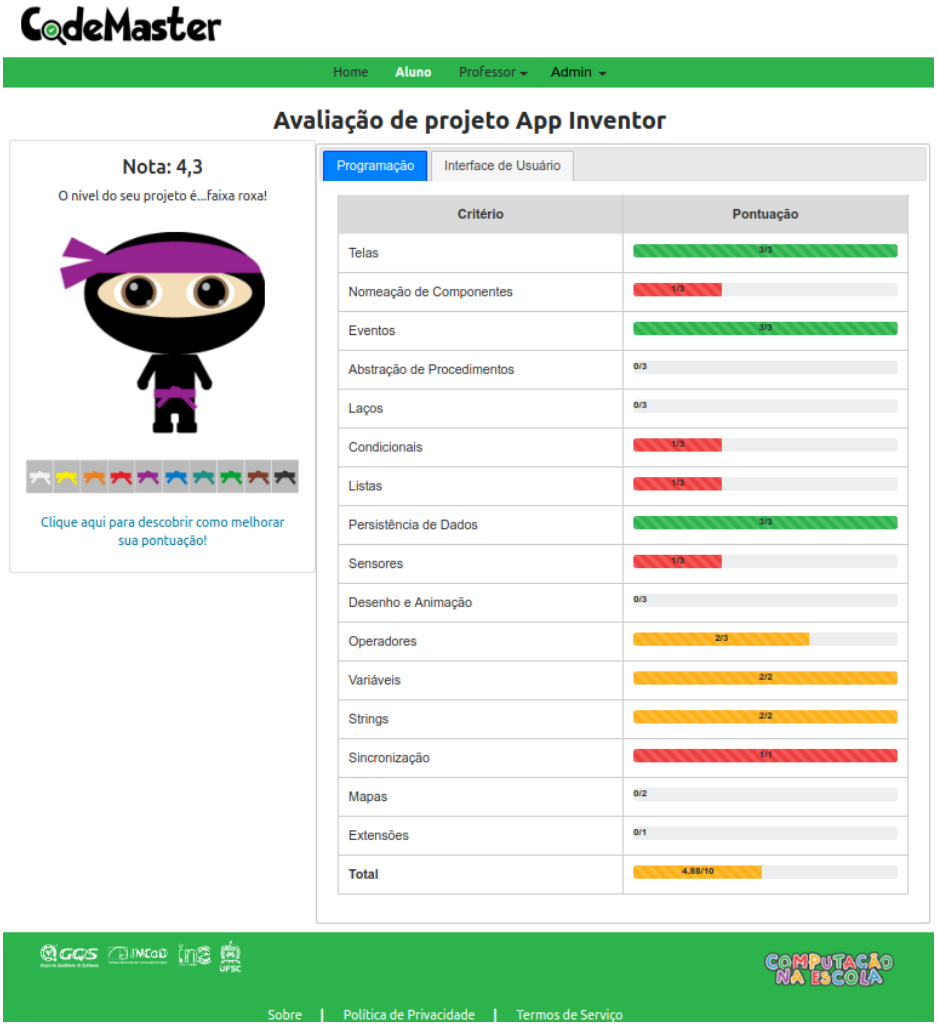
[16] *Material Design color palettes:* https://material.io/design/color/the-color-system.html#tools-for-picking-colors

Apêndice D

Design de interface resultante do sistema de acordo com o fluxo principal de execução dos casos de uso (Capítulo 5.2).

Tabela 1: Interfaces gráficas do USC01

01. Aluno acessa o site do CodeMaster	
2. Aluno clica no item de menu “Aluno”	

3. Aluno clica em “Escolher arquivo” e escolhe projeto salvo em seu computador.	
4. Aluno clica em “Avaliar” e sistema retorna o resultado da análise do projeto na aba da rubrica de programação	

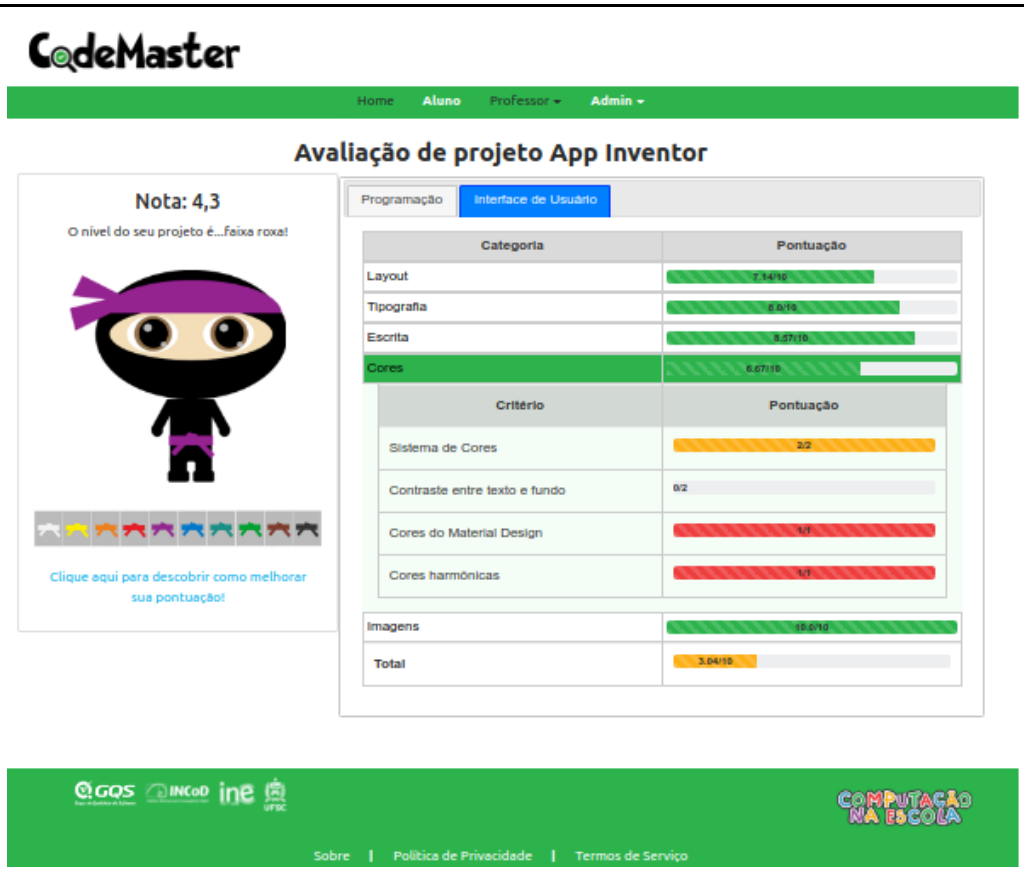
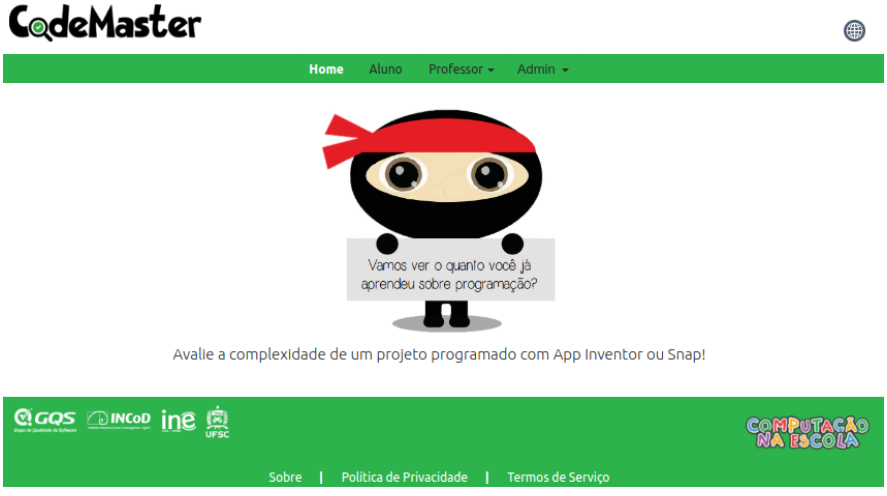
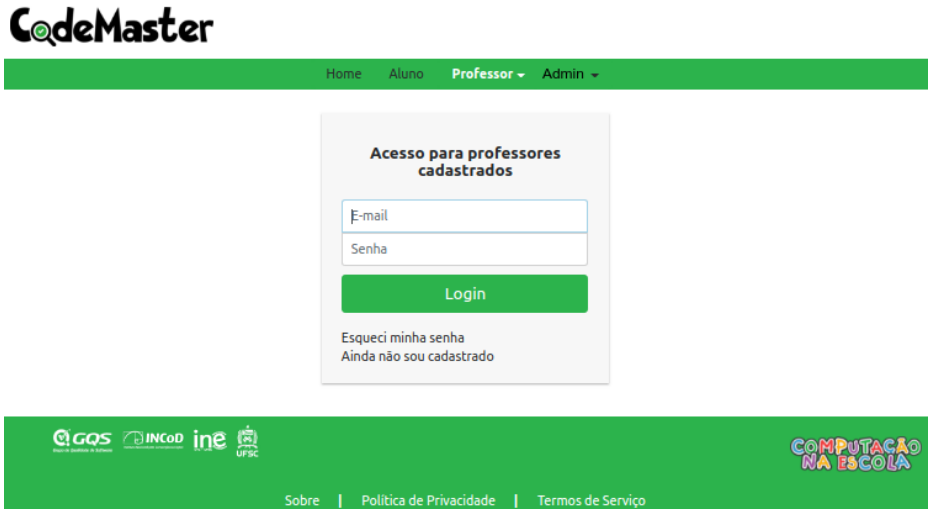
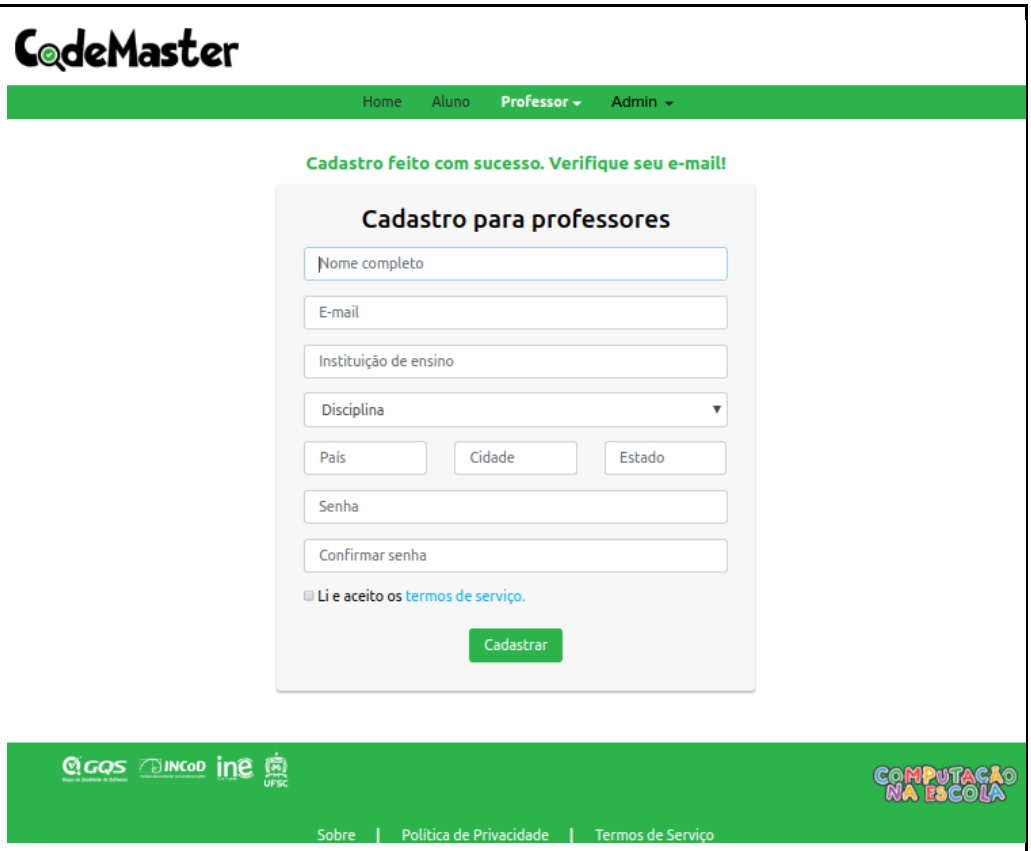
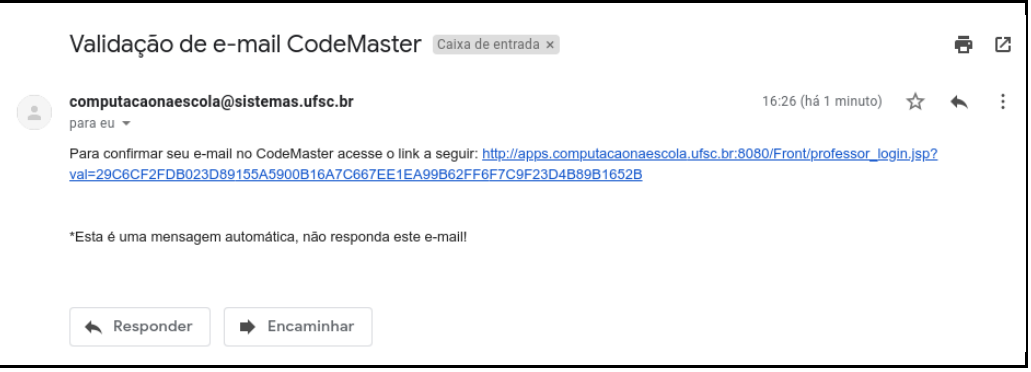
5. Aluno clica em “Interface de Usuário” para visualizar o resultado da análise do projeto no contexto de design	

Tabela 2: Interfaces gráficas do USC02 - Cadastrar como professor

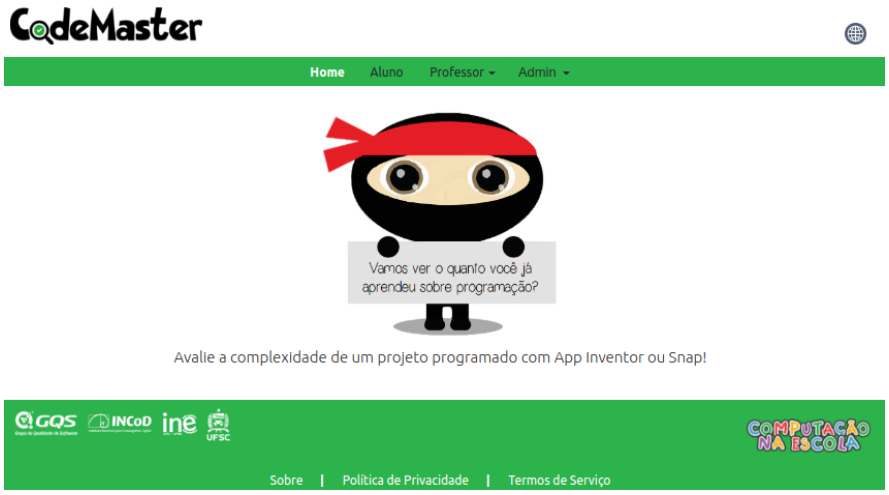
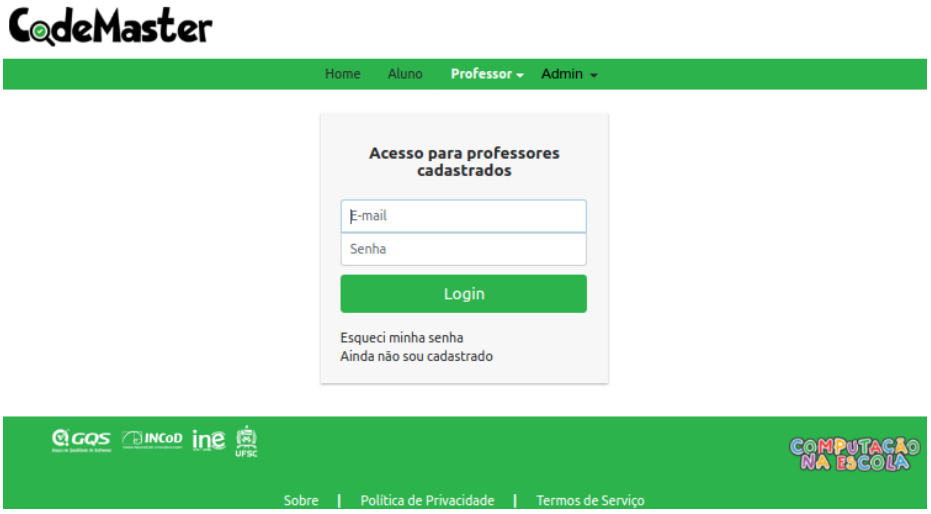
1. Professor acessa site do CodeMaster.	

2. Professor clica no botão “Professor”, que fica no menu superior.	
3. Professor clica na opção de “Ainda não sou cadastrado” abaixo dos campos de login e senha.	

4. Professor informa, nome, e-mail, instituição de ensino, disciplina, cidade, estado, país, senha e clica em “Cadastrar”.	
5. Professor recebe e-mail com link para validar conta.	

6. Professor abre link recebido no e-mail. O sistema abre a página de login de professor com uma mensagem de conta confirmada exibida.	

Tabela 3: Interfaces gráficas do USC03 - Logar como Professor

1. Professor acessa o site do CodeMaster	
2. Professor clica no botão “Professor” no menu superior da tela e informa seu e-mail e senha nos campos de login e senha e clica em “Login”.	

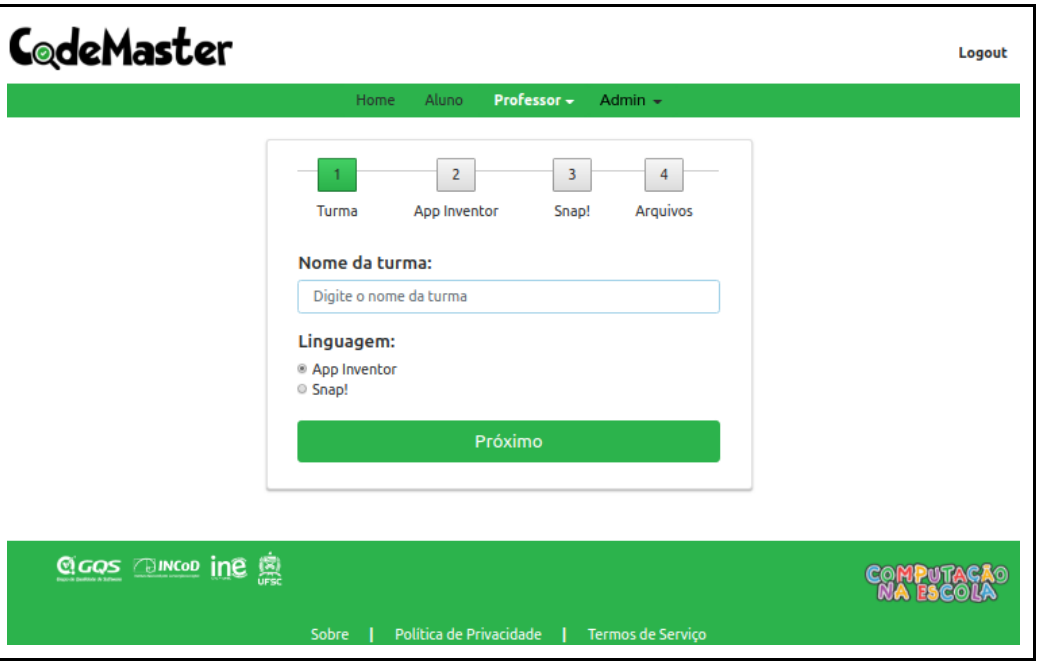
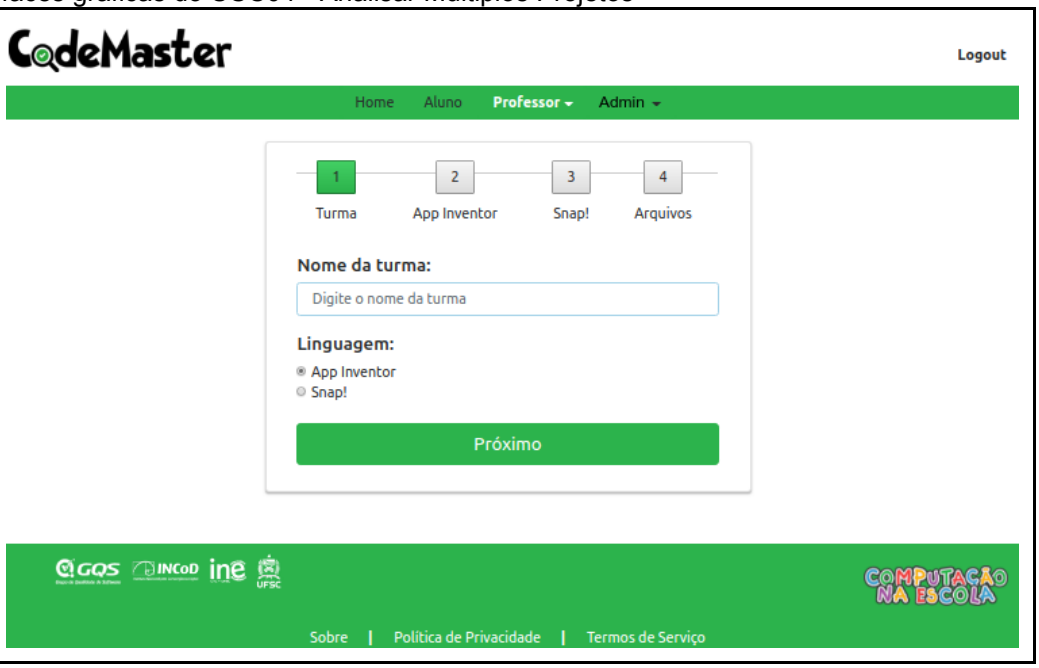
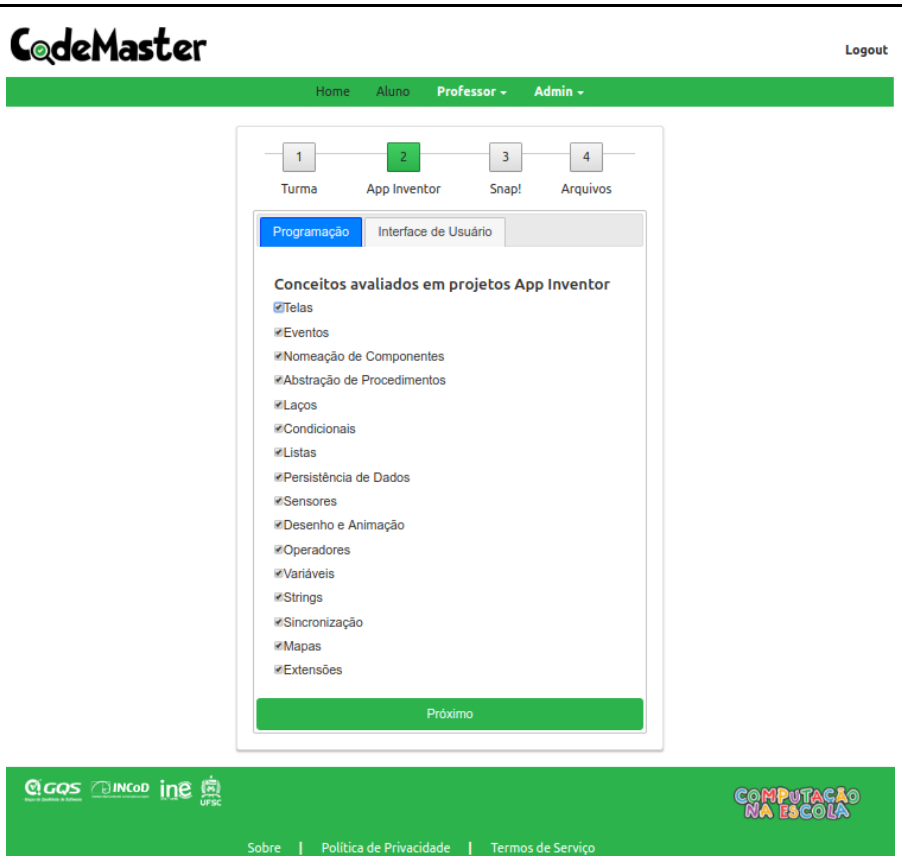
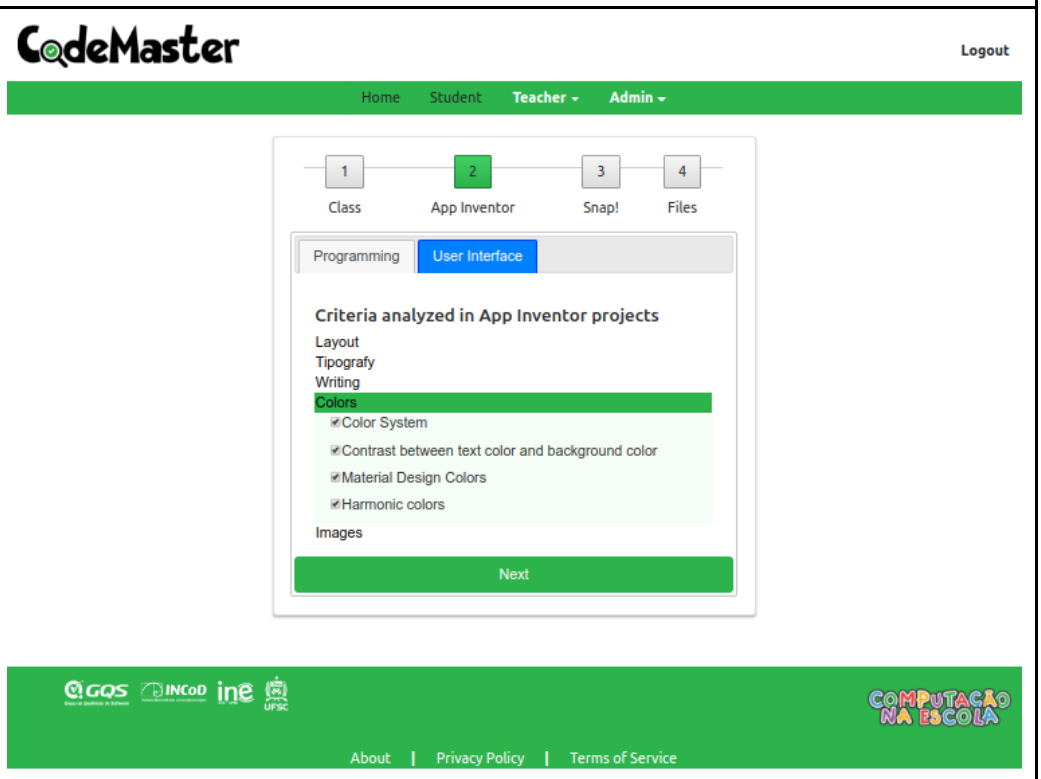
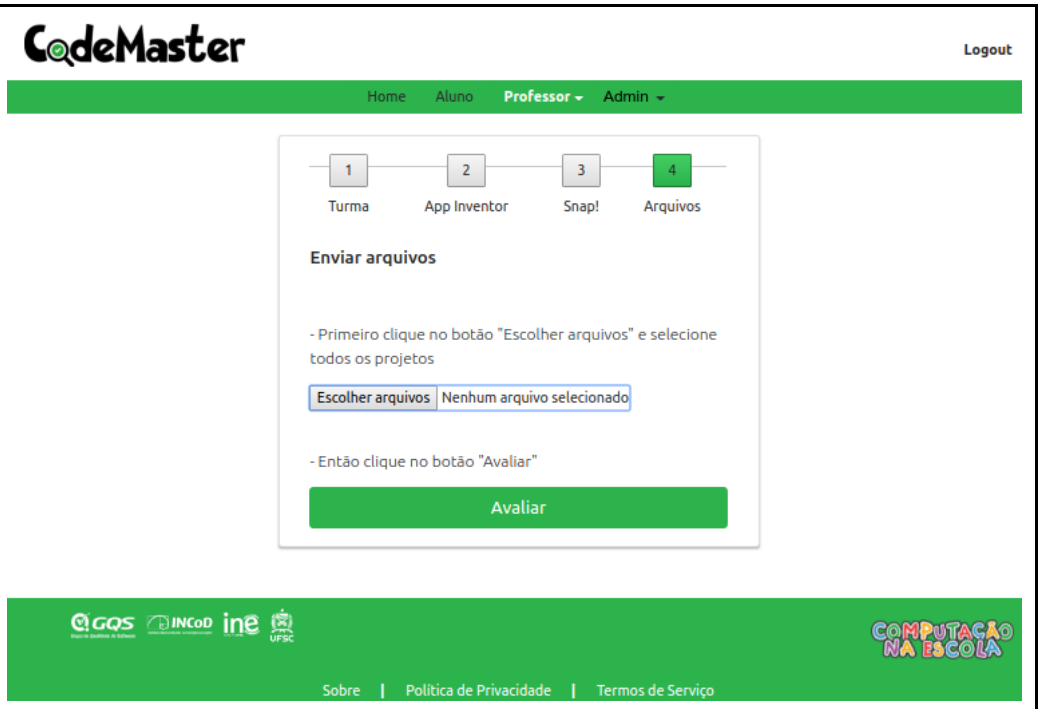
3. O sistema autentica o professor e redireciona à página principal de submissão de múltiplos projetos.	

Tabela 4: Interfaces gráficas do USC04 - Analisar Múltiplos Projetos

1. Professor digita o nome da turma e seleciona o tipo de projeto que deseja avaliar e clica em “Próximo”. Obs.: Para essa fluxo de execução é para ser selecionado “App Inventor”	

2. Professor seleciona os conceitos que deseja avaliar da rubrica de programação	
3. Professor seleciona os conceitos que deseja avaliar da rubrica de design	

4. Professor escolhe múltiplos arquivos de seu computador que correspondem a uma turma. Professor clica em avaliar.	
5a. Professor recebe o resultado da análise de todos os projetos em em 3 tabelas. Aba Total	

Projeto	Nota em Programação	Nota em Interface de Usuário	Nota Final	Nível
FloripaPraias_RevRaul.aia	5.12	2.56	4,35	faixa roxa
HealthyPlants_RevRaul.aia	4.55	2.55	4,25	faixa roxa
InfoCarvalhal_RevRaul.aia	4.63	1.55	3,50	faixa vermelha
AcheiOseuEmprego_RevRaul.aia	5.55	2.22	4,76	faixa roxa
AppPhone_RevRaul.aia	4.39	2.72	3,59	faixa vermelha
Média	**20,40**	**19,80**	**4,22**	

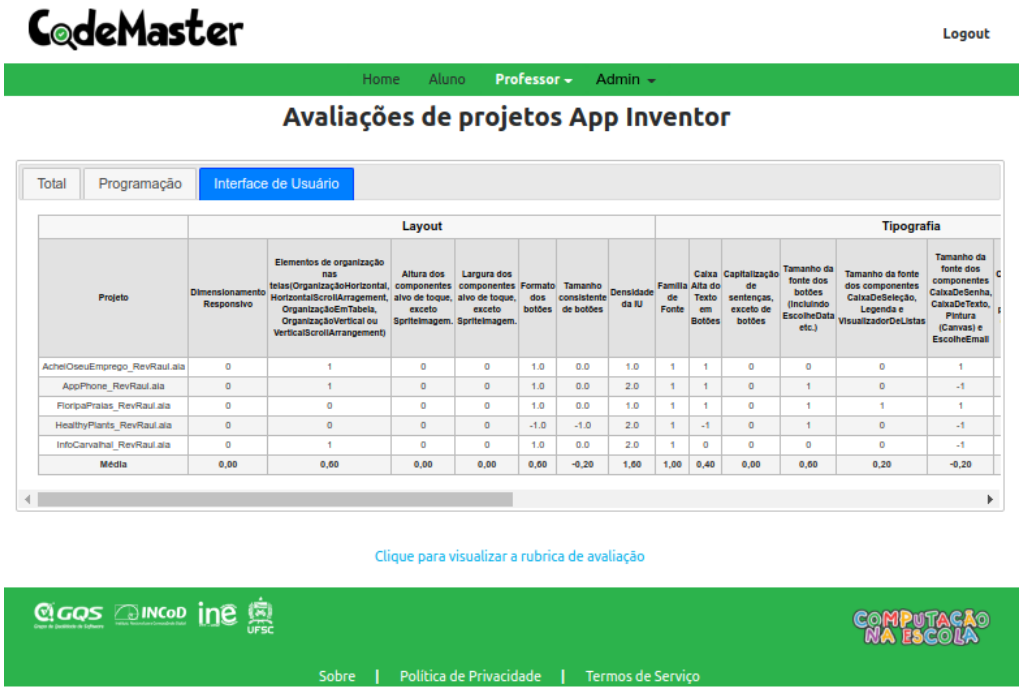
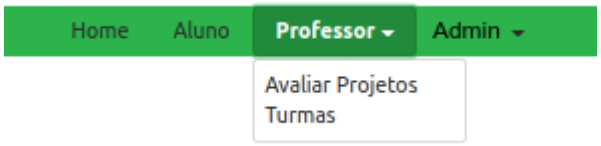
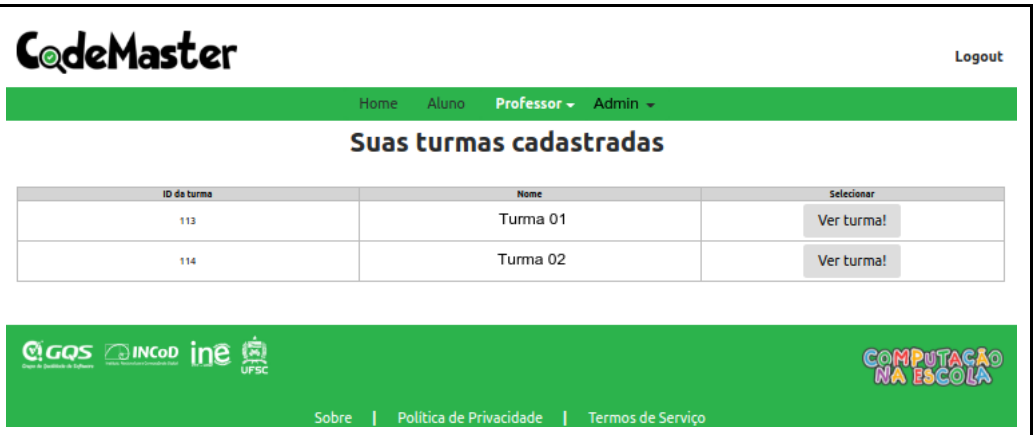
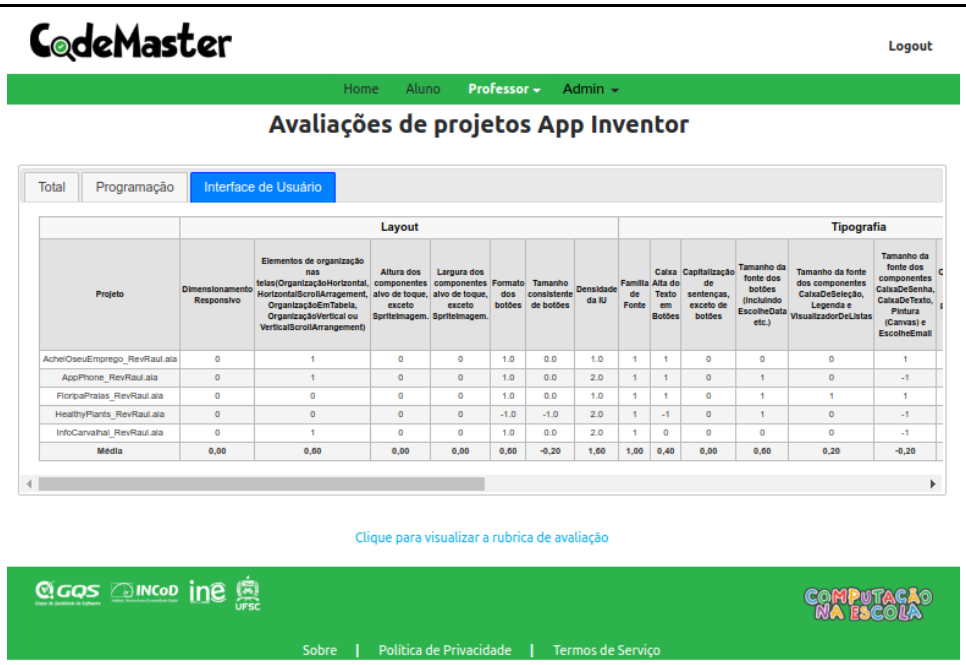
5b. Professor recebe o resultado da análise de todos os projetos em em 3 tabelas. Aba Programação	
5c. Professor recebe o resultado da análise de todos os projetos em em 3 tabelas. Aba Interface de Usuário	

Tabela 5: Interfaces gráficas do USC05 - Acesso a projetos analisados

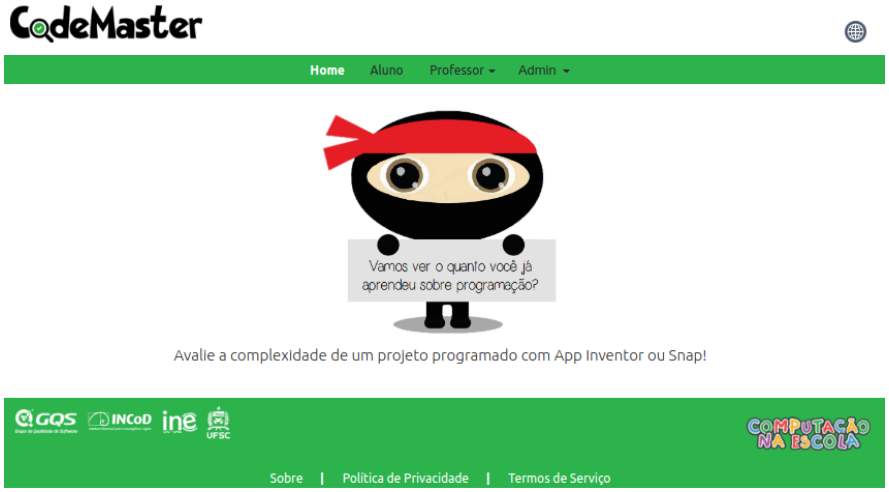
1. Professor clica em “Turmas” no menu superior da área de professores	

2. Professor seleciona a turma que deseja ver as análises.	
3a. Sistema consulta no Banco de Dados e retorna o resultado das análises da turma selecionada. Aba Total.	
3b. Sistema consulta no Banco de Dados e retorna o resultado das análises da turma selecionada. Aba Programação	

3c. Sistema consulta no Banco de Dados e retorna o resultado das análises da turma selecionada. Aba Interface de Usuário	

Projeto	Layout							Tipografia						
	Dimensionamento Responsivo	Elementos de organização nas telas(OrganizaçãoHorizontal, HorizontalScrollArragement, OrganizaçãoEmTabela, OrganizaçãoVertical ou VerticalScrollArrangement)	Altura dos componentes alvo de toque, exceto SpriteImagem.	Largura dos componentes alvo de toque, exceto SpriteImagem.	Formato dos botões	Tamanho consistente de botões	Densidade da IU	Família de Fonte	Caixa Alta do Texto em Botões	Capitalização de sentenças, exceto de botões	Tamanho da fonte dos botões (incluindo EscolheData etc.)	Tamanho da fonte dos componentes CaixaDeSeleção, Legenda e VisualizadorDeListas	Tamanho da fonte dos componentes CaixaDeSenha, CaixaDeTexto, Pintura (Canvas) e EscolheEmail	C
AcheiOseuEmprego_RevRaul.aia	0	1	0	0	1.0	0.0	1.0	1	1	0	0	0	1	
AppPhone_RevRaul.aia	0	1	0	0	1.0	0.0	2.0	1	1	0	1	0	-1	
FloripaPraias_RevRaul.aia	0	0	0	0	1.0	0.0	1.0	1	1	0	1	1	1	
HealthyPlants_RevRaul.aia	0	0	0	0	-1.0	-1.0	2.0	1	-1	0	1	0	-1	
InfoCarvalhal_RevRaul.aia	0	1	0	0	1.0	0.0	2.0	1	0	0	0	0	-1	
Média	**0,00**	**0,60**	**0,00**	**0,00**	**0,60**	**-0,20**	**1,60**	**1,00**	**0,40**	**0,00**	**0,60**	**0,20**	**-0,20**	

Tabela 6: Interfaces gráficas do USC06 - Logar como Administrador

1. Administrador acessa o site do CodeMaster.	

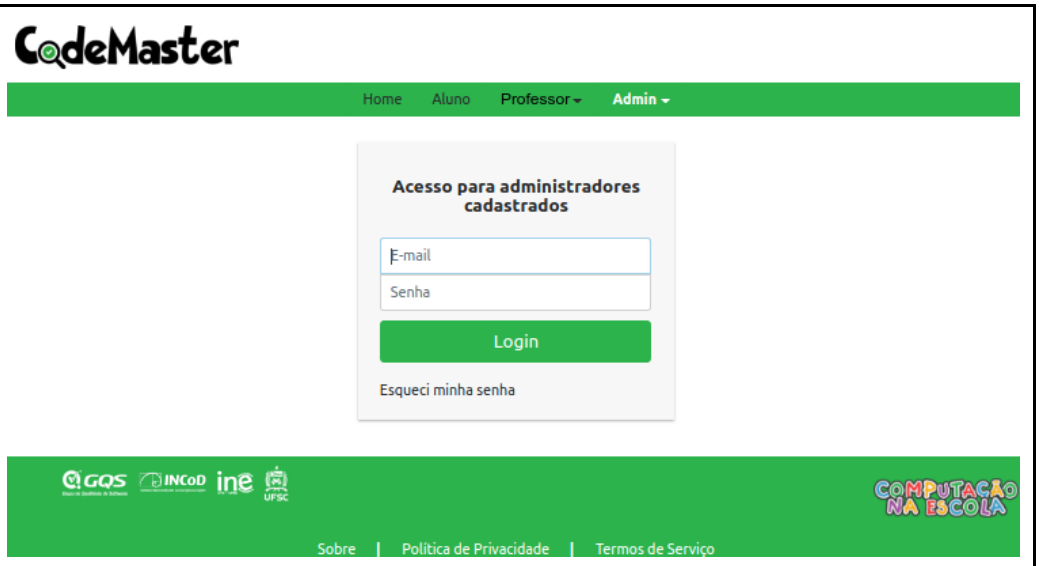
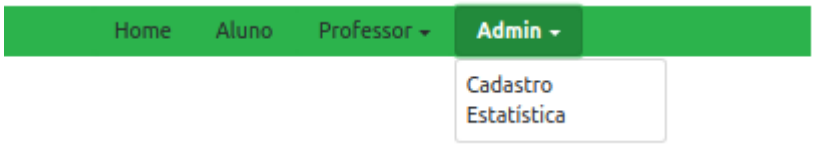
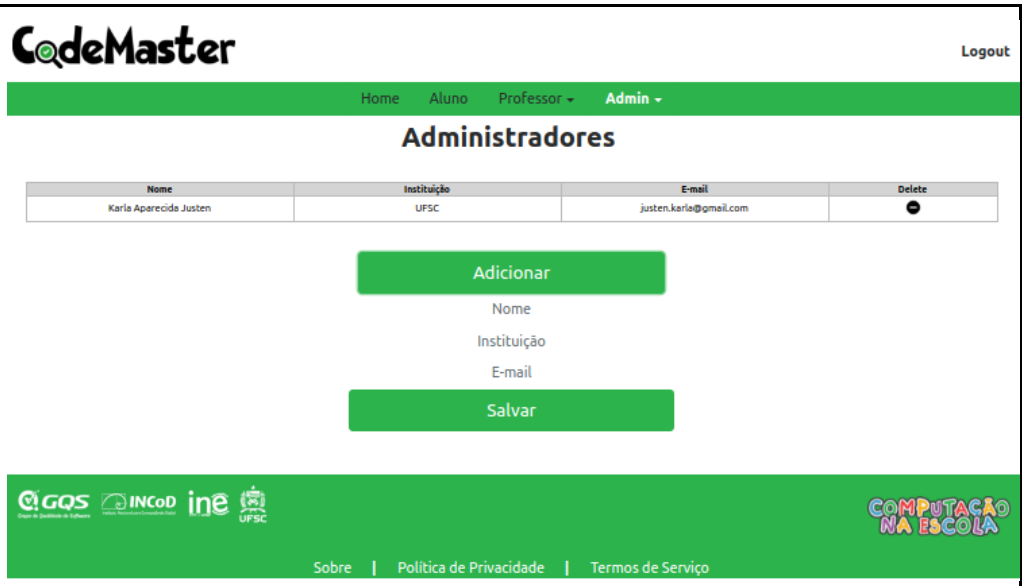
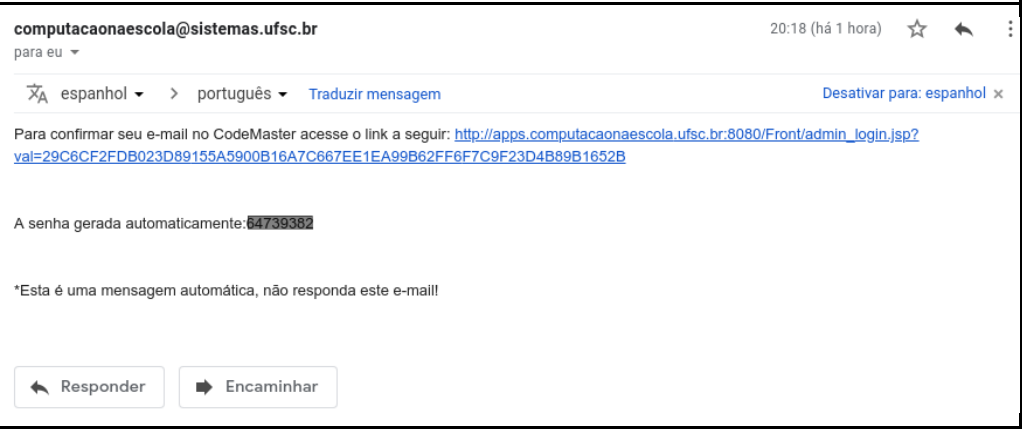
2. Administrador clica no botão “Admin” no menu superior da tela.	
3. O sistema autentica o administrador e é redirecionado à página de cadastro de administradores.	

Tabela 7: Interfaces gráficas do USC07 - Cadastrar outros administradores

1. Administrador logado, clica em “Cadastro”.	
2. Administrador clica em “Adicionar”.	

3. Administrador insere os dados do novo administrador e clica em “Salvar”	
4. O sistema atualiza a tabela de administradores, incluindo o novo administrador	
5. O sistema envia um email para o novo administrador, com link de confirmação e senha.	

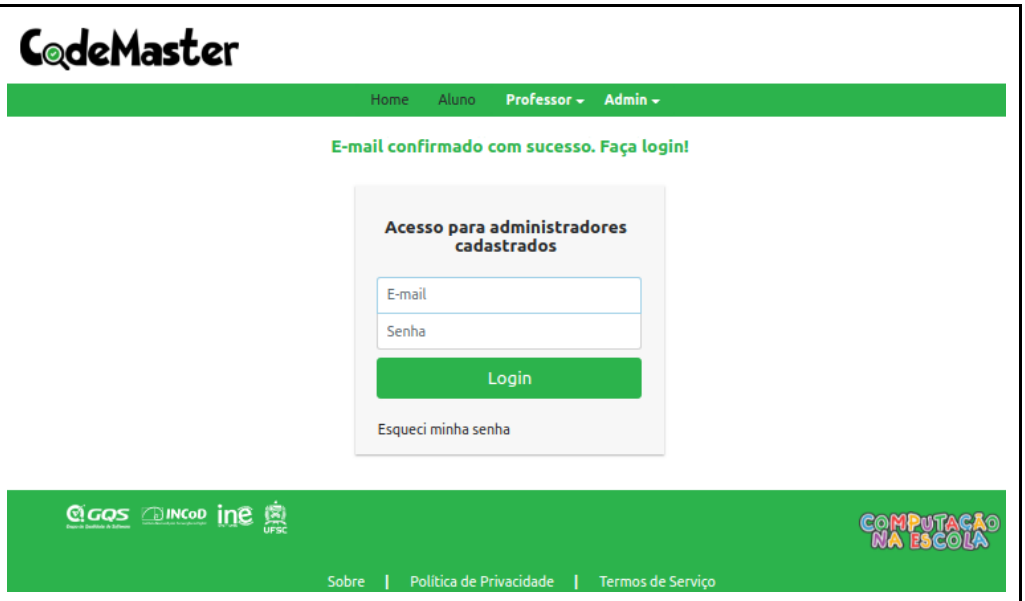
6. Novo administrador abre link recebido no e-mail e uma mensagem de conta confirmada é exibida.	

Tabela 8: Interfaces gráficas do USC08 - Excluir o cadastro de administradores

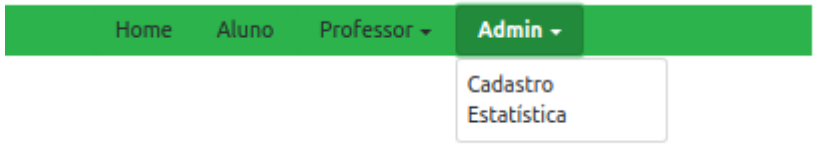
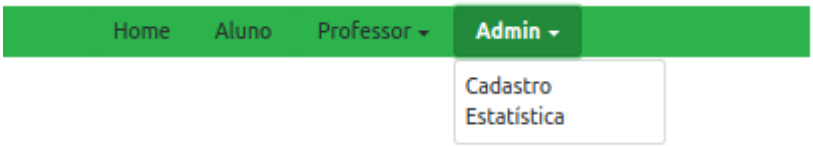
1. Administrador logado, clica em “Cadastro”	
2. Administrador clica no simbolo “⛔” da linha correspondente ao administrador que deseja excluir o cadastro.	

Tabela 9: Interfaces gráficas do USC09 - Acesso a todos os projetos analisados

1. Administrador logado, clica em “Estatística”.	

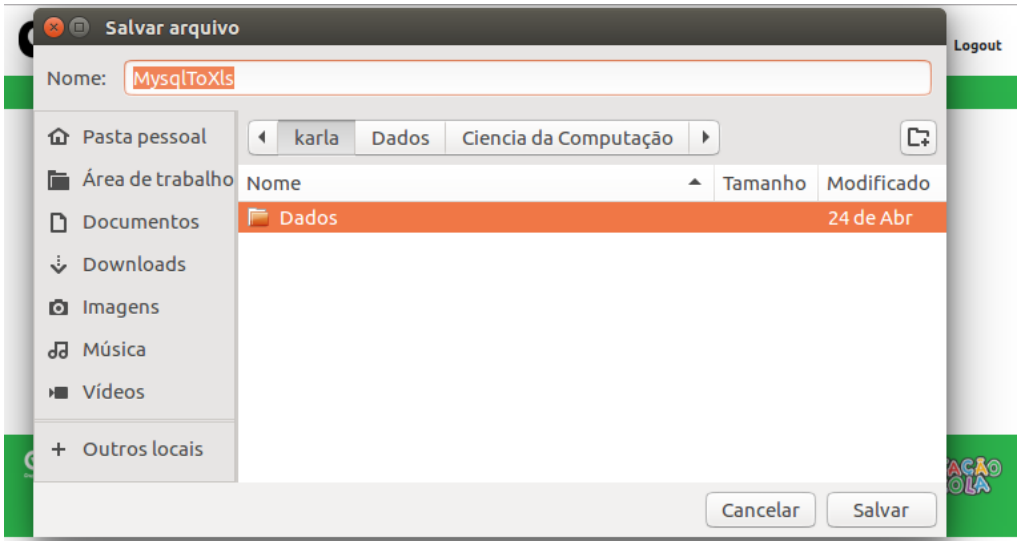
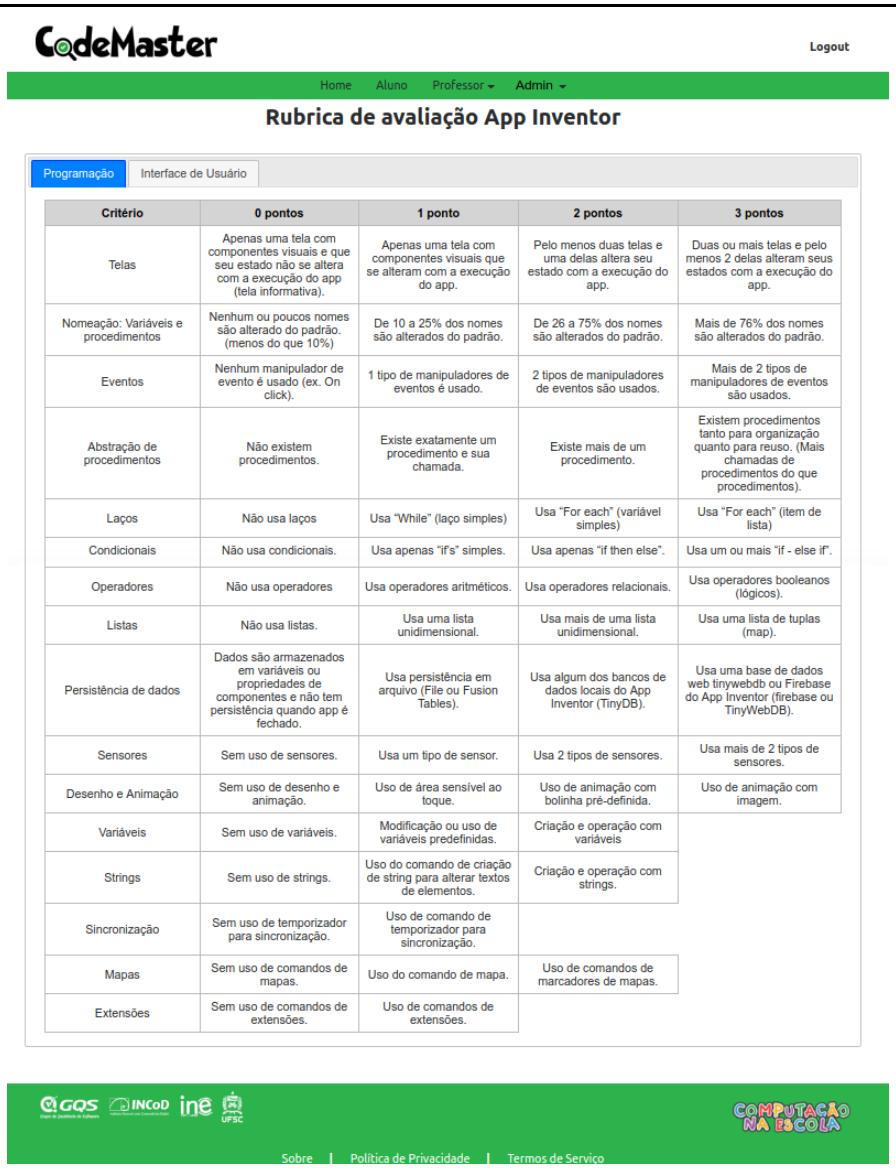
2. Administrador define as informações de filtragem dos dados de interesse. E clica em “Gerar .xlsx”	
3. O sistema solicita um endereço da memória local para salvar a planilha gerada	

Tabela 10: USC10 Acessa Rubrica

1. Se a pré-condição for USC01: aluno clica em “Clique aqui para descobrir como melhorar sua pontuação!” na rubrica de programação. OU Se a pré-condição for USC04 ou USC05: professor clica em “Clique para visualizar a rubrica de avaliação” na rubrica de programação.	

CodeMaster

Logout

Home Aluno Professor Admin

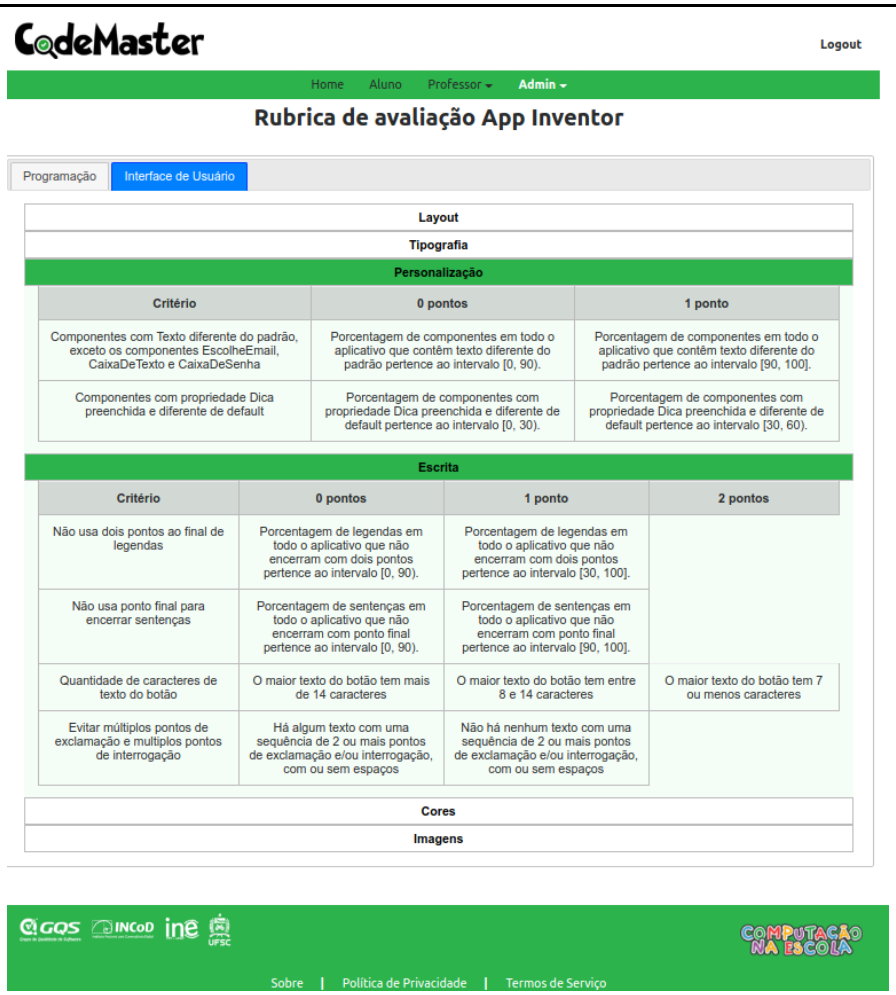
Rubrica de avaliação App Inventor

Programação | Interface de Usuário

Critério	0 pontos	1 ponto	2 pontos	3 pontos
Telas	Apenas uma tela com componentes visuais e que seu estado não se altera com a execução do app (tela informativa).	Apenas uma tela com componentes visuais que se alteram com a execução do app.	Pelo menos duas telas e uma delas altera seu estado com a execução do app.	Duas ou mais telas e pelo menos 2 delas alteram seus estados com a execução do app.
Nomeação: Variáveis e procedimentos	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.
Eventos	Nenhum manipulador de evento é usado (ex. On click).	1 tipo de manipuladores de eventos é usado.	2 tipos de manipuladores de eventos são usados.	Mais de 2 tipos de manipuladores de eventos são usados.
Abstração de procedimentos	Não existem procedimentos.	Existe exatamente um procedimento e sua chamada.	Existe mais de um procedimento.	Existem procedimentos tanto para organização quanto para reuso. (Mais chamadas de procedimentos do que procedimentos).
Laços	Não usa laços	Usa “While” (laço simples)	Usa “For each” (variável simples)	Usa “For each” (item de lista)
Condicionais	Não usa condicionais.	Usa apenas “if’s” simples.	Usa apenas “if then else”.	Usa um ou mais “if - else if”.
Operadores	Não usa operadores	Usa operadores aritméticos.	Usa operadores relacionais.	Usa operadores booleanos (lógicos).
Listas	Não usa listas.	Usa uma lista unidimensional.	Usa mais de uma lista unidimensional.	Usa uma lista de tuplas (map).
Persistência de dados	Dados são armazenados em variáveis ou propriedades de componentes e não tem persistência quando app é fechado.	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).
Sensores	Sem uso de sensores.	Usa um tipo de sensor.	Usa 2 tipos de sensores.	Usa mais de 2 tipos de sensores.
Desenho e Animação	Sem uso de desenho e animação.	Uso de área sensível ao toque.	Uso de animação com bolinha pré-definida.	Uso de animação com imagem.
Variáveis	Sem uso de variáveis.	Modificação ou uso de variáveis predefinidas.	Criação e operação com variáveis	
Strings	Sem uso de strings.	Uso do comando de criação de string para alterar textos de elementos.	Criação e operação com strings.	
Sincronização	Sem uso de temporizador para sincronização.	Uso de comando de temporizador para sincronização.		
Mapas	Sem uso de comandos de mapas.	Uso do comando de mapa.	Uso de comandos de marcadores de mapas.	
Extensões	Sem uso de comandos de extensões.	Uso de comandos de extensões.		

GQS INCoD ine UFSC COMPUTAÇÃO NA ESCOLA

Sobre | Política de Privacidade | Termos de Serviço

7. Usuário clica em “Interface de Usuário” para descobrir como melhorar sua pontuação na rubrica "CodeMaster UI Design - App Inventor”.	

CodeMaster

Logout

Home Aluno Professor Admin

Rubrica de avaliação App Inventor

Programação | Interface de Usuário

Layout

Tipografia

Personalização

Critério	0 pontos	1 ponto
Componentes com Texto diferente do padrão, exceto os componentes EscolheEmail, CaixaDeTexto e CaixaDeSenha	Porcentagem de componentes em todo o aplicativo que contêm texto diferente do padrão pertence ao intervalo [0, 90).	Porcentagem de componentes em todo o aplicativo que contêm texto diferente do padrão pertence ao intervalo [90, 100].
Componentes com propriedade Dica preenchida e diferente de default	Porcentagem de componentes com propriedade Dica preenchida e diferente de default pertence ao intervalo [0, 30).	Porcentagem de componentes com propriedade Dica preenchida e diferente de default pertence ao intervalo [30, 60).

Escrita

Critério	0 pontos	1 ponto	2 pontos
Não usa dois pontos ao final de legendas	Porcentagem de legendas em todo o aplicativo que não encerram com dois pontos pertence ao intervalo [0, 90).	Porcentagem de legendas em todo o aplicativo que não encerram com dois pontos pertence ao intervalo [30, 100].	
Não usa ponto final para encerrar sentenças	Porcentagem de sentenças em todo o aplicativo que não encerram com ponto final pertence ao intervalo [0, 90).	Porcentagem de sentenças em todo o aplicativo que não encerram com ponto final pertence ao intervalo [90, 100].	
Quantidade de caracteres de texto do botão	O maior texto do botão tem mais de 14 caracteres	O maior texto do botão tem entre 8 e 14 caracteres	O maior texto do botão tem 7 ou menos caracteres
Evitar múltiplos pontos de exclamação e multiplos pontos de interrogação	Há algum texto com uma sequência de 2 ou mais pontos de exclamação e/ou interrogação, com ou sem espaços	Não há nenhum texto com uma sequência de 2 ou mais pontos de exclamação e/ou interrogação, com ou sem espaços	

Cores

Imagens

GQS INCoD ine UFSC COMPUTAÇÃO NA ESCOLA

Sobre | Política de Privacidade | Termos de Serviço

Apêndice E

Desenvolvimento de um Analisador de Design de Interface no Contexto do Ensino de Computação com o App Inventor

Karla Aparecida Justen[1], João V. A. Porto[1], Igor d. S. Solecki[1], C. Gresse von Wangenheim[1], Jean C. R. Hauck[1]

[1] Departamento de Informatica e Estatística, Universidade Federal de Santa Catarina (UFSC), Florianópolis, Brasil

{karla.justen,joao.porto}@grad.ufsc.br, igor.solecki@posgrad.ufsc.br, {c.wangenheim,jean.hauck,adriano.borgatto}@ufsc.br

Abstract. *The diffusion of information technologies in society makes it necessary for everyone to be familiar with these technologies. One way to address these needs is by teaching computational thinking in K-12 through the development of mobile applications with App Inventor covering also user interface design, being an important quality. Yet, assessing the application's interface design in the educational context is quite arduous. In order to facilitate this process, this article presents a rubric and its automation through a web-based tool. An initial evaluation of the rubric indicated an acceptable reliability (Cronbach* α *= 0,71) and validity, yet its construct validity could not completely been demonstrated pointing out for further research on this issue.*

Resumo. *A difusão das tecnologias de informação na sociedade torna necessário que todos estejam familiarizados com essas tecnologias. Uma forma de abordar essa necessidade é ensinando o pensamento computacional na Educação Básica por meio do desenvolvimento de aplicativos móveis App Inventor abordando também o design da interface do usuário, pois é uma qualidade importante. No entanto, avaliar o design da interface do aplicativo no contexto educacional é bastante árduo. A fim de facilitar este processo, este artigo apresenta uma rubrica e sua automação em uma ferramenta web. Uma avaliação inicial da rubrica indicou uma consistência interna aceitável (alfa de Cronbach = 0,71) e validade, mas sua validade de construto não foi completamente satisfatória, apontando para futuras pesquisas sobre esta questão.*

1. Introdução

O uso cotidiano de tecnologias de informação e comunicação pela população tem sido imprescindível, tanto na vida pessoal como na carreira profissional. Por isso, a inclusão do ensino dos princípios básicos da computação na Educação Básica acaba sendo conhecimento útil para o cidadão como um usuário de tecnologias de informação, mas também o torna capacitado para ser criador [Wing, 2006]. Uma maneira de incluir o ensino da computação na Educação Básica é aproveitar a relação que os estudantes têm com dispositivos móveis utilizados para entretenimento e diversão como ferramenta para ensinar o pensamento computacional [Duda e Silva, 2015] por meio de criação de aplicativos App Inventor.

O App Inventor (2019) é uma ferramenta de código-aberto, desenvolvida na Google sendo mantido pelo Instituto de Tecnologia de Massachusetts (MIT). O App Inventor auxilia no desenvolvimento de aplicativos para dispositivos Android suportando design de interface e a programação funcional do aplicativo. Esse ambiente de programação permite proporcionar às

pessoas com pouco ou nenhum conhecimento de programação a experiência de desenvolver aplicativos para dispositivos Android.

Uma das principais qualidades de um aplicativo é a sua usabilidade. E como a qualidade da interface interfere diretamente na intenção dos usuários de continuar utilizando o aplicativo [Hoehle, Aljafari e Venkatesh, 2016]. O design de interface de aplicativos para smartphones precisa ser fácil, agradável de usar e exija pouco esforço para os usuários [Rogers, Sharp e Preece, 2013]. Atualmente já existem vários cursos e tutoriais de programação (p. ex. MIT (2019), Daniel et al. (2017) e AppInventor.org (2019)). Porém, esses geralmente enfocam somente na parte da programação fornecendo nenhum suporte (ou somente de forma muito superficial) para design de interface [Ferreira et al. 2019].

Para acompanhar o processo de aprendizagem é essencial avaliar o design de interface de aplicativos criados pelos alunos e fornecer *feedback*. No contexto de aprendizagem baseada em problemas em que não existe um gabarito pré-definido, tipicamente são adotadas rubricas. Rubricas definem um conjunto de critérios e níveis de desempenho alcançáveis em cada critério [Biagiotti, 2005]. Representando os nível níveis de desempenho numa escala ordinal pode ser indicado uma nota geral.

Com o objetivo de auxiliar professores da Educação Básica na avaliação de projetos de forma consistente, com exatidão e evitando *bias*, minimizando o esforço do avaliador foi automatizada a avaliação da rubrica e incluído na ferramenta web CodeMaster [Gresse Von Wangenheim et al. 2018]. Os resultados apresentados nesse artigo podem ser adotados por professores da Educação Básica no ensino de computação nas suas disciplinas e também por pesquisadores dessa área para análise de dados e evolução da ferramenta.

2. Metodologia

A metodologia de pesquisa utilizada neste trabalho é dividida em cinco etapas principais. A primeira etapa foi a fundamentação teórica, que consiste em estudar, analisar e sintetizar os conceitos principais e a teoria referente aos temas abordados em detalhes em Justen (2019). A segunda etapa foi realizado o estado da arte (Capítulo 3). Nessa etapa é realizado um mapeamento sistemático de literatura seguindo o processo proposto por Petersen et al. (2008) para identificar e estudar analisadores de design de interface de ambientes de programação visual (baseado em blocos) atualmente sendo utilizados. A terceira etapa foi realizado o desenvolvimento da rubrica seguindo a metodologia de design instrucional ADDIE [Branch, 2009]. Na quarta etapa foi desenvolvida a análise de design de interface e acrescentada à ferramenta web CodeMaster, com base no processo iterativo incremental de engenharia de software [Larman e Basili, 2003]. Por fim, a quinta etapa a rubrica foi avaliada em termos de consistência interna e validade com base em um conjunto de apps desenvolvidos com App Inventor disponibilizados sob a licença de creative commons na galeria do App Inventor.

3. Estado da Arte

Com a intenção de identificar e estudar analisadores de design de interface de ambientes de programação visual (baseado em blocos) atualmente sendo utilizados, foi realizado o mapeamento sistemático de literatura seguindo o procedimento proposto por Petersen et al. (2008). O objetivo é responder a seguinte pergunta de pesquisa: Quais meios existem para avaliação de design visual de interface de aplicativos móvel de smartphones touchscreen Android ou aplicativos desenvolvidos no App Inventor?

As buscas foram realizadas nas principais bases de dados e bibliotecas digitais do campo da computação, que são (ACM Digital Library, IEEE Xplore, ScienceDirect, SpringerLink, Wiley e Scopus). Considerando publicações disponíveis via o Portal CAPES. Com o intuito de cobrir uma gama maior de publicações, também foram conduzidas pesquisas no Google Scholar, que indexa um grande conjunto de dados de diversas fontes de produção científicas distintas [Haddaway et al. 2015].

No total foram encontrados 2571 artefatos nas buscas (maiores detalhes estão disponíveis em Justen (2019)). Analisando os resultados das buscas foram encontrados somente 9 artefatos que satisfazem os critérios de inclusão e exclusão.

Notou-se que 4 de 9 artefatos apresentam como meio de avaliação da qualidade do design de interface questionários ou *checklists* voltado ao desenvolvimento de apps de forma geral [Android, 2018] [Gresse von Wangenheim et al. 2016] [Zapata et al. 2014]. O método de avaliação por rubricas foi proposta por 4 artefatos analisados [Technovation, 2014] [Hoehle, Aljafari e Venkatesh, 2016] [Wagner et al. 2013] [Sherman e Martin, 2015]. No contexto educacional não existem automatizações desses instrumentos, o que pode consumir um esforço considerável sendo propensa a erros, tornando a avaliação uma tarefa difícil [Ines et al., 2017].

Foram encontradas também 2 soluções que abordam a avaliação de design de interface de forma automatizada [Ines et al., 2017] [Gresse von Wangenheim et al., 2018]. Gresse von Wangenheim et al (2018) apresentam a ferramenta CodeMaster v1.0 voltado ao ensino de pensamento computacional no contexto educacional. Porém nessa versão da ferramenta a qualidade do design de interface é abordado somente de forma superficial por meio de 2 critérios. Um referenciava à quantidade de telas e outro à quantidade de componentes visuais. Enquanto Ines et al (2017) apresenta duas listas de avaliação de interface de aplicativos Android. Uma lista de métricas da qualidade e outra de defeitos do design de interface de um aplicativo Android. Porém as duas listas não não especificam os limites quantitativos para a avaliação de cada métrica. Por exemplo, a métrica de densidade avalia quando uma aplicativo contém muitos componentes em uma tela e não é definido quando é considerado baixa ou alta densidade, sendo um meio inviável para o contexto educacional.

Essa falta de soluções para avaliar aplicativos Android desenvolvidos no App Inventor no contexto educacional evidencia então claramente a necessidade de pesquisa nessa área.

4. Rubrica "CodeMaster UI Design - App Inventor"

Com esse objetivo foi desenvolvido a rubrica "CodeMaster UI Design - App Inventor" que visa avaliar a qualidade do design de interface de apps Android a partir de conceitos básicos de design visual com base em Schlatter e Levinson (2013) e guias como o Material Design (2019) e Web Content Accessibility Guidelines 2.0 (2018). Foram também levado em consideração somente elementos visuais que podem ser criados os recursos do App Inventor.

De acordo com os princípios e tipos de elementos de design de interface, a rubrica foram organizados em nas seguintes categorias:

- Layout: critérios que avaliam como componentes são organizados e dispostos na tela.
- Tipografia: critérios que definem como os elementos textuais são apresentados.
- Escrita: critérios que definem como os elementos textuais precisam ser escritos.
- Cores: critérios que analisam as cores utilizadas no app e a sua combinação
- Imagens: critérios que avaliam a qualidade de imagens e ícones.

Cada critério é avaliado por níveis de desempenho variando de 2 a 4 níveis em uma escala ordinal. O nível de desempenho 0 representa o pior desempenho, enquanto o maior nível representa o melhor desempenho. A rubrica "CodeMaster UI Design - App Inventor" é apresentada na Tabela 4.

Tabela 1. Rubrica "CodeMaster UI Design - App Inventor"

Critério	Níveis de desempenho			
	0 pt	1 pt	2pt	3pt
Layout				
L1. O dimensionamento é fixo ou responsivo?	Fixo	Responsivo		
L2. Quantas telas usam organizadores?	Menos de 90%	90% ou mais		
L3. Todos os componentes alvos de toque têm largura e altura maior ou igual a 48 pixels?	Não	Sim		
L4. Todos os botões têm o mesmo formato?	Não	Sim		
L5. Botões agrupados na interface sempre têm o mesmo tamanho?	Não	Sim		
L6. Quantos elementos têm a tela com menos elementos (min) e a tela com mais elementos (max)?	$min < 2$ ou $max \geq 20$	$min \geq 2$ e $max \in [10, 19]$	$min \geq 2$ e $max \leq 9$	
Tipografia				
T1. Quantos componentes têm família da fonte sem serifa?	Menos de 90%	90% ou mais		
T2. Quantos botões com texto têm tamanho da fonte igual a 14?	Menos de 90%	90% ou mais		
T3. Quantos componentes (exceto botões) têm tamanho da fonte igual a um dos tamanhos recomendados?	Menos de 90%	90% ou mais		
T4. Há texto em itálico?	Sim	Não		
T5. O texto de todos os botões é centralizado?	Não	Sim		
Escrita				
E1. Quantos botões têm o texto todo em maiúsculo?	Menos de 90%	90% ou mais		
E2. Quantas sentenças começam com letra maiúscula ou dígito?	Menos de 90%	90% ou mais		
E3. Quantos componentes têm texto diferente do padrão (p. ex., "Texto para Botão1")?	Menos de 90%	90% ou mais		
E4. Quantas caixas de texto têm uma dica definida?	Menos de 90%	90% ou mais		
E5. Quantos rótulos (*labels*) **não** terminam com ":" (dois pontos)?	Menos de 90%	90% ou mais		
E6. Quantas sentenças **não** terminam com "." (ponto)?	Menos de 90%	90% ou mais		
E7. O botão com texto mais longo tem quantos caracteres?	15 ou mais	de 8 a 14	7 ou menos	
E8. Há texto com múltiplos sinais de exclamação/interrogação ("!!", "?!")?	Sim	Não		
Cores				
C1. Quantas cores são usadas no aplicativo (além de preto, branco e cinza)?	5 ou mais	4	3	1 ou 2
C2. Qual o nível WCAG do aplicativo em relação ao contraste do texto?	Insuficiente	Nível AA	Nível AAA	
C3. Usam-se apenas cores das paletas sugeridas pelo Material Design?	Não	Sim		
C4. As tonalidades de cores usadas são harmônicas entre si (complementares, análogas ou triádicas)?	Não	Sim		
Imagens				
I1. Quantos ícones usados no aplicativo são do Material Design?	Menos de 90%	90% ou mais		
I2. Existem imagens pixelizadas?	Sim	Não		
I3. Existem botões com uma imagem distorcida?	Sim	Não		
I4. Existem componentes do tipo Imagem distorcidos?	Sim	Não		
I5. Existem imagens de fundo de tela distorcidas?	Sim	Não		

5. Automatização da Avaliação da Rubrica

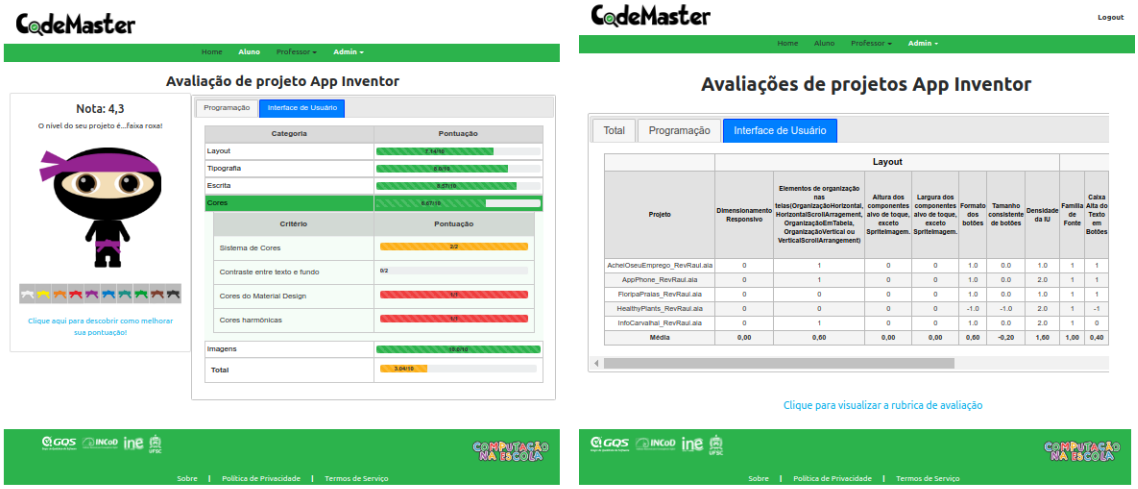
Um módulo de automatização da avaliação do design de interface de aplicativos App Inventor foi desenvolvido com base na rubrica "CodeMaster UI Design - App Inventor". Esse módulo foi integrado a ferramenta CodeMaster [Gresse Von Wangenheim et al. 2018] complementando de forma mais completa a avaliação do design de interface além de conceitos de programação.

O módulo desenvolvido realiza uma análise estática no arquivo do aplicativo (.aia) submetido para extrair as informações de interesse e identificar o nível de desempenho alcançado

em cada critério da rubrica. Com os níveis de desempenho alcançado definido para todos os critérios uma nota de 0 a 10 é calculada.

Com base nas notas de cada rubrica é apresentada uma nota final e nível ninja em que a cor da faixa muda conforme a nota final, sendo uma forma de gamificação para estimular os alunos a aprimorarem os seus aplicativos. A ferramenta pode ser utilizado individualmente pelo próprio estudante para acompanhar o seu processo de aprendizagem (Figura 1a) ou pelo professor avaliando ao mesmo tempo todos os projetos de uma turma (Figura 1b).

A ferramenta está sendo disponibilizada gratuitamente no site da iniciativa computação na escola (http://apps.computacaonaescola.ufsc.br:8080).



a b

Figura 1. Exemplo de avaliação do design de UI.
1a Usuário como aluno. 1b Usuário como professor

O módulo foi testado conforme a corretude, para assegurar o possível que a implementação feita corresponde ao proposto pela rubrica.

6. Avaliação de Consistência e Validade da Rubrica

Importante para assegurar uma avaliação precisa, é que a rubrica utilizada é confiável e válida [DeVellis, 2003]. Assim, foi realizada uma avaliação inicial da rubrica "CodeMaster UI Design - App Inventor" em termos de consistência interna e validade de construto. Para avaliar a consistência da rubrica, a seguinte questão de análise foi elaborada:

Q1: Há evidência de consistência interna na rubrica CodeMaster UI Design - App Inventor?

A validade de construto é analisada por meio da validade convergente e discriminante [Trochim e Donnelly, 2008]. A validade convergente testa se os conceitos ou medidas que deveriam supostamente estar relacionados estão de fato relacionados. Validade discriminante diz respeito ao grau em que dois critérios de fatores de qualidade que não devem estar relacionados estão de fato não relacionados [Trochim e Donnelly, 2008]. Assim, a seguinte questão de análise foi elaborada:

Q2: Há evidência de validade convergente e discriminante na rubrica CodeMaster UI Design - App Inventor?

Para realizar as análises, foram baixados 978 aplicativos disponíveis publicamente na Galeria do App Inventor [MIT, 2019] em maio de 2018. Esses aplicativos foram avaliados automaticamente utilizando a ferramenta CodeMaster. Os resultados das avaliações foram exportados para uma planilha xlsx com os níveis de desempenho alcançados em cada critério da rubrica "CodeMaster UI Design - App Inventor".

6.1 Análise

Nessa seção, analisamos cada questão de análise.

6.1 Há evidência de consistência interna na rubrica CodeMaster UI Design - App Inventor?

A confiabilidade da rubrica "CodeMaster UI Design - App Inventor" foi analisada medindo sua consistência interna por meio do coeficiente alfa de Cronbach (1951). O coeficiente alfa de Cronbach indica o grau em que um conjunto de critérios mede um único fator de qualidade [Cronbach, 1951]. O coeficiente alfa de Cronbach consiste em um valor entre 0 e 1. Existe um consenso de que os valores de alfa de Cronbach no intervalo de 0,70 a 0,95 são considerados aceitáveis, ou seja, indica consistência interna [Devellis, 2003]. Analisando os 28 critérios da rubrica "CodeMaster UI Design - App Inventor", foi obtido Cronbach $\alpha = 0{,}71$, sendo considerado então um valor satisfatório.

6.2 Há evidências de validades convergente e discriminante na rubrica CodeMaster UI Design - App Inventor?

Para obter evidências de validades convergente e discriminante dos critérios da rubrica de design do CodeMaster, foram calculadas as intercorrelações entre critérios e a correlação critério-total [Wohlin et al. 2012].

Intercorrelações entre critérios. Para analisar as intercorrelações entre critérios foi usada a correlação policórica (Tabela 2), a mais apropriada para variáveis ordinais observadas [Olsson, 1979]. Para validade convergente, espera-se que os critérios da mesma categoria tenham uma alta correlação [Trochim e Donnelly, 2008]. Por outro lado, para se obter evidência de validade discriminante, espera-se que critérios de diferentes categorias tenham baixa correlação. De acordo com Cohen (1998), uma correlação entre os critérios é considerada satisfatória se o coeficiente de correlação for maior que 0,29, indicando que há uma correlação média ou alta entre os critérios.

Tabela 2. Resultados da análise de correlação policórica

		Layout						Tipografia					Escrita								Cores				Imagens				
		L1	L2	L3	L4	L5	L6	T1	T2	T3	T4	T5	E1	E2	E3	E4	E5	E6	E7	E8	C1	C2	C3	C4	I1	I2	I3	I4	I5
Layout	L1. dimensionamentoResponsivo	1.00																											
	L2 telasComOrganizadores	0.09	1.00																										
	L3 tamanhoDeAlvosDeToque	-0.29	0.06	1.00																									
	L4 formatoDosBotoes	-0.11	**0.30**	0.24	1.00																								
	L5 tamanhoConsistenteDeBotoes	-0.02	0.11	0.25	0.18	1.00																							
	L6 densidadeDaIU	-0.16	**0.37**	**0.40**	**0.30**	**0.39**	1.00																						
Tipografia	T1 familiaFonte	0.03	**0.34**	**0.33**	**0.54**	0.15	**0.44**	1.00																					
	T2 tamanhoFonteBotoes	-0.23	0.14	0.28	**0.65**	0.18	0.29	**0.57**	1.00																				
	T3 tamanhoDaFonte	-0.06	0.25	0.08	0.27	0.19	**0.36**	**0.42**	**0.46**	1.00																			
	T4 naoUsaItalico	-0.16	**0.36**	**0.36**	**0.50**	**0.38**	**0.52**	**0.65**	**0.48**	**0.39**	1.00																		
	T5 botaoComTextoCentralizado	-0.04	0.27	0.26	**0.80**	0.04	0.17	**0.62**	**0.75**	0.24	**0.49**	1.00																	
Escrita	E1 caixaAltaBotao	0.20	0.13	0.07	**0.32**	-0.14	-0.17	**0.30**	0.02	-0.11	0.17	**0.56**	1.00																
	E2 capitalizacaoSentenca	0.03	-0.07	-0.17	-0.05	-0.04	-0.06	-0.28	-0.23	-0.15	-0.08	-0.22	0.20	1.00															
	E3 textoDefault	0.13	**0.38**	0.15	**0.55**	0.28	**0.48**	**0.53**	**0.49**	**0.38**	**0.50**	**0.47**	-0.04	-0.12	1.00														
	E4 diccaPreenchidaEDiferenteDefault	0.27	0.14	-0.03	0.15	0.01	0.14	0.01	0.20	0.19	0.13	**0.30**	0.06	0.01	**0.38**	1.00													
	E5 doisPontosLegenda	0.12	0.13	0.03	-0.03	0.09	0.13	**0.36**	-0.12	0.02	0.25	0.08	0.21	-0.13	**0.30**	0.12	1.00												
	E6 pontoFinalSentenca	0.11	-0.03	-0.06	-0.02	-0.12	-0.05	-0.04	0.01	0.02	0.08	0.00	0.14	0.11	-0.12	-0.18	-0.22	1.00											
	E7 tamanhoMaximoTextoBotao	-0.09	0.28	0.15	**0.64**	0.23	**0.29**	**0.42**	**0.53**	0.18	**0.48**	**0.74**	**0.41**	-0.15	**0.47**	0.08	-0.02	0.17	1.00										
	E8 exclamacaoMultipla	0.01	0.21	-0.17	0.06	0.17	**0.32**	0.08	0.02	0.08	0.00	-0.10	-0.06	0.21	0.05	-0.03	-0.09	**0.33**	0.15	1.00									
Cores	C1 sistemaDeCores	0.19	0.16	0.00	0.18	0.11	0.11	**0.32**	-0.12	0.03	0.25	0.06	0.09	-0.01	0.22	-0.14	0.23	0.00	0.00	0.16	1.00								
	C2 contrasteEntreTextoEFundo	-0.17	0.14	**0.31**	0.28	**0.49**	**0.46**	**0.50**	**0.40**	**0.31**	**0.52**	0.23	-0.17	-0.13	**0.48**	0.04	0.04	-0.01	**0.30**	0.05	-0.06	1.00							
	C3 coresMaterialDesign	0.09	0.05	-0.01	0.19	-0.04	-0.07	**0.50**	-0.05	-0.06	0.17	0.15	**0.35**	0.01	-0.09	-0.55	0.25	-0.16	0.07	-0.04	**0.63**	-0.06	1.00						
	C4 coresHarmonicas	0.12	0.15	0.03	0.17	0.16	0.20	0.22	-0.07	0.11	**0.29**	0.05	-0.04	-0.06	0.15	-0.24	0.15	0.03	0.02	0.09	**0.90**	0.01	**0.57**	1.00					
Imagens	I1 iconesMD	**0.42**	0.17	-0.50	0.12	0.13	0.09	0.05	-0.19	0.02	0.09	-0.11	0.02	**0.50**	**0.38**	0.18	0.21	-0.17	0.03	**0.34**	0.50	-0.08	0.26	0.36	1.00				
	I2 pixelizacao	0.21	0.04	0.00	-0.04	0.00	0.03	0.18	-0.08	-0.14	0.13	-0.01	-0.05	-0.01	0.27	0.09	0.12	-0.12	-0.10	0.01	0.16	0.01	0.05	0.06	0.29	1.00			
	I3 imagensEmBotoes	0.18	0.03	0.01	-0.21	0.12	0.08	0.04	-0.25	-0.15	0.19	-0.26	-0.14	0.04	0.18	0.06	0.10	-0.14	-0.17	0.04	0.22	0.10	0.00	0.08	**0.36**	**0.76**	1.00		
	I4 componenteImagem	0.03	-0.08	-0.15	0.12	-0.20	-0.17	0.08	0.01	-0.08	0.02	0.15	0.17	-0.03	0.28	0.15	0.21	-0.22	-0.18	-0.11	0.05	-0.15	-0.02	-0.02	0.12	**0.46**	0.09	1.00	
	I5 imagensDeFundo	0.12	-0.01	0.01	-0.24	-0.16	-0.06	-0.15	-0.12	-0.13	-0.19	-0.14	-0.02	-0.13	-0.04	0.11	-0.08	**0.60**	-0.17	**0.45**	-0.09	-0.11	-0.45	-0.17	-0.23	0.26	0.17	0.25	1.00

Os critérios que pertencem à dimensão tipográfica mostram um grau aceitável de correlação em 9 de 10 correlações. Especialmente os critérios T2 e T5 apresentam um valor de correlação muito alto. Apenas o par T3 e T5 apresenta um valor abaixo de 0,29.

Dentre as 6 correlações entre os critérios de Cores, 3 foram satisfatórias. O critério C2 não tem correlação satisfatória com os demais critérios da categoria Cores. Porém C2 apresenta uma boa correlação com 4 de 5 critérios da categoria Tipografia. Isso sugere que esse critério se enquadre melhor na categoria de Tipografia sendo relacionado a questões da cor de texto. Situação semelhante ocorre com o critério E3, pois não tem correlação satisfatória com os critérios da mesma categoria.

Como vários critérios de diferentes dimensões mostram alto grau de correlação, a rubrica não apresenta validade discriminante. Também não é possível garantir a validade convergente, pois existem correlações entre critérios da mesma dimensão com grau baixo. Essas baixas correlações entre os critérios da mesma dimensão indicam que os critérios não estão medindo o mesmo fator dimensional. Consequentemente, a validade da conformidade do design visual das interfaces dos aplicativos do App Inventor com as diretrizes não é totalmente garantida pela rubrica. Portanto, esses critérios precisam ser revisados em análises futuras e potencialmente reagrupados ou eliminados da rubrica.

Correlação critério-total. A correlação critério-total consiste em avaliar a correlação de um critério com todos os outros critérios [DeVellis, 2003]. Para essa análise foi utilizado o método da correlação critério-total corrigida. Esse método compara um critério com todos os outros critérios da rubrica, exceto com ele mesmo (Tabela 3). Espera-se que haja uma correlação média ou alta, pois isso indica que os critérios apresentam consistência em comparação com os outros critérios. Para isso o coeficiente de correlação precisa ser maior que 0,29 [Cohen, 1998]. Além disso, foi analisado o alfa de Cronbach da rubrica excluindo apenas um dos critério. A expectativa é de que nenhum critério cause uma diminuição significativa no alfa de Cronbach [DeVellis, 2003].

Tabela 3. Resultados da análise de correlação item-total e alfa de Cronbach.

Item	Correla. item-total	Alfa de Cronbach com item removido
L1. dimensionamentoResponsivo	**-0.01**	0.71
L2. telasComOrganizadores	0.30	0.70
L3. tamanhoDeAlvosDeToque	0.25	0.70
L4. formatoDosBotoes	0.49	0.69
L5. tamanhoConsistenteDeBotoes	0.27	0.70
L6. densidadeDaIU	0.44	0.68
T1. familiaFonte	0.58	0.68
T2. tamanhoFonteBotoes	0.38	0.69
T3. tamanhoDaFonte	0.28	0.70
T4. naoUsaItalico	0.56	0.68
T5. botaoComTextoCentralizado	0.47	0.69
E1. caixaAltaBotao	**0.09**	0.71
E2. capitalizacaoSentenca	**-0.13**	**0.73**
E3. textoDefault	0.56	0.69
E4. dicaPreenchidaEDiferenteDefault	**0.08**	0.71
E5. doisPontosLegenda	**0.14**	0.71
E6. pontoFinalSentenca	**-0.03**	0.72
E7. tamanhoMaximoTextoBotao	0.37	0.69
E8. exclamacaoMultipla	**0.07**	0.71
C1. sistemaDeCores	0.23	0.71
C2. contrasteEntreTextoEFundo	0.36	0.70
C3. coresMaterialDesign	**0.08**	0.71
C4. coresHarmonicas	0.28	0.70

I1. iconesMD	**0.07**	0.71
I2. pixelizacao	**0.10**	0.71
I3. imagensEmBotoes	**0.06**	0.71
I4. componenteImagem	**-0.01**	0.72
I5. imagensDeFundo	**-0.02**	0.71

Em relação à correlação item-total, apenas 10 critérios do total de 28 critérios apresenta correlação acima de 0,29. No entanto, alguns itens apresentam uma correlação baixa, como I3 e I1 ou até mesmo negativa, como I4 e E2. Isso indica que esses critérios não estão significativamente correlacionados com os demais critérios da rubrica e possivelmente devem ser reformulados ou até mesmo excluídos.

O valor do alfa de Cronbach mostra um pequeno aumento somente para o item E2. Os demais itens não apresentam aumento do alfa da rubrica quando excluídos, indicando que contribuem para a avaliação de forma geral.

6.3 Ameaças à validade

Este trabalho está sujeito a várias ameaças à validade. Portanto, foram identificadas ameaças potenciais e estratégias de mitigação foram aplicadas para minimizar seu impacto na pesquisa proposta. A fim de mitigar as ameaças relacionadas ao design do estudo, foram definidos e documentos uma metodologia sistemática usando a abordagem GQM [Basili, Caldiera e Rombac, 1994].

Outra questão refere-se ao agrupamento de dados de diferentes contextos. Os aplicativos da amostra analisada foram coletados da Galeria do App Inventor e nenhuma informação sobre o nível de conhecimento dos criadores dos aplicativos é disponibilizado. Ou seja, aplicativos desenvolvidos por inexperiente ou por profissionais são avaliados igualmente. No entanto, como o objetivo é analisar a validade da rubrica de maneira independente do contexto, isso não é considerado algo a ser tratado neste momento.

A escolha dos métodos estatísticos para a análise dos dados sempre pode representar ameaças à pesquisa. Para minimizar essa ameaça, foi realizada uma análise estatística baseada na abordagem para a construção de escalas de medida conforme proposta por DeVellis (2003), que está alinhada com procedimentos para avaliação da consistência interna e validade de construto de instrumentos de medida.

7. Conclusão

Esse artigo apresenta resultados de pesquisa voltada a automatizar a avaliação do design de interface de aplicativos criados com App Inventor na Educação Básica. Os resultados da avaliação inicial da rubrica indicou uma consistência interna aceitável (alfa de Cronbach = 0,71) e validade, mas sua validade de construto não foi completamente satisfatória, apontando para futuras pesquisas sobre esta questão.

Em termos de impacto científico foi criado uma rubrica de avaliação de design de interface de apps criados com App Inventor, representando um instrumento inédito. Em termos tecnológicos foi criado um módulo de um sistema web para automatizar a avaliação dessa rubrica. Em termos sociais espera-se contribuir com a popularização de computação na sociedade. Espera-se também pela abordagem do design de interface motivar minorias, como meninas, procurarem uma carreira nessa área, além de oferecer um ensino de computação mais completo.

Referências

Android (2018) Qualidade Do Aplicativo Principal. Disponível Em: <Https://Developer.Android.Com/Docs/Quality-guidelines/Core-app-quality#Ux>. Acessado em: Junho de 2018

App Inventor (2019), About us. Disponível em: http://appinventor.mit.edu/explore/about-us.html. Acessado em: Maio de 2019.

AppInventor.org (2019). Course In A Box, http://www.appinventor.org/content/CourseInABox/Intro, Acessado em: Maio de 2019.

Basili, V. R., Caldiera G., Rombach, H. D. (1994) Goal Question Metric Paradigm. In: Encyclopedia of Software Engineering. John Wiley & Sons, 1, p. 528-532.

Biagiotti, L. C. M. (2005) Conhecendo e Aplicando Rubricas Em Avaliações. In: Proc. of the 12º Congresso Internacional de Educação a Distância, Florianópolis, Brasil.

Branch, R. M. (2009)Instructional Design: The ADDIE Approach. New York: Springer.

Cohen, J. (1998). Statistical Power Analysis for the Behavioral Sciences. Routledge Academic.

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. Psychometrika, 16(3).

Daniel, G. T. et al (2017). Ensinando a Computação por meio de Programação com App Inventor. In: Proc. of Computer on the Beach, Florianópolis/Brazil.

Deluca, C.; Klinger, D. A. (2010) Assessment literacy development: identifying gaps in teacher candidates' learning. Assessment in Education: Principles, Policy & Practice, 17 (4), p. 419-438.

DeVellis, R. F. (2003). Scale development: theory and applications. SAGE Publications.

Duda, R.; Silva, S. C. R. (2015) Desenvolvimento de Aplicativos para android com uso do app inventor: uso de novas tecnologias no processo de ensino-aprendizagem em matemática. Revista Conexão, 11(3), p. 310-323.

Ferreira, M. N. F. et al. (2019). Learning user interface design and development of mobile applications in middle school. ACM Interactions 26(4), 2019.

Gresse Von Wangenheim, C. et al. (2018) Codemaster – Automatic Assessment And Grading Of App Inventor And Snap! Programs. Informatics In Education, 17(1), 117–150.

Gresse von Wangenheim, C. et al. (2016) An Usability Score For Mobile Phone Applications Based on Heuristics. International Journal of Mobile Human Computer Interaction Archive. 8(1), 23-58.

Haddaway, N. R. et al. (2015) The Role of Google Scholar in Evidence Reviews and Its Applicability to Grey Literature Searching. PLOS ONE, 10(9).

Hoehle, H.; Aljafari, R.; Venkatesh, V. (2016) Leveraging Microsoft's Mobile Usability Guidelines: Conceptualizing and Developing Scales For Mobile Application Usability. International Journal of Human-Computer Studies. 89, p. 35-53.

Ines, G .et al. (2017) Evaluation of Mobile Interfaces as an Optimization Problem. Procedia Computer Science 112, p. 235-248.

Justen, K. A. (2019) Desenvolvimento de um Analisador de Design de Interface no Contexto do Ensino de Computação com o App Inventor. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Universidade Federal de Santa Catarina, Florianópolis.

Larman, C.; Basili, V. R. (2003) Iterative and Incremental Development: A Brief History. Computer, 36(6) p. 47-56.

Material Design. Disponível em: <https://material.io/guidelines/>. Acesso em: Maio de 2019.

MIT (2019). "Tutorials for App Inventor", http://appinventor.mit.edu/explore/ai2/tutorials.html. Acessado em: Maio de 2019.

Olsson, U (1979). Maximum likelihood estimation of the polychoric correlation coefficient. Psychometrika, 44(4).

Petersen, K. et al. (2008) Systematic Mapping Studies in Software Engineering. In: Proc. of the 12th Int. Conference on Evaluation and Assessment in Software Engineering, Bari, Italy, p. 68-77.

Rashkovits, R; Lavy, I. (2013) FACT: A Formative Assessment Criteria Tool for the Assessment of Students' Programming Tasks. In: Proc. of the World Congress on Engineering. Londres, UK.

Rogers, Y.; Sharp, H; Preece, J. (2013) Design de Interação: Além da Interação Homem-Computador. Bookman, 3ª ed.

Schlatter, T.; Levinson, D. Visual (2013) Usability: Principles and practices for designing digital applications. Newnes.

Sherman, M.; Martin, F. (2015) The Assessment of Mobile Computational Thinking. Journal Of Computing Sciences In Colleges. 30(6), 53-59.

Technovation (2014) Teacher And Mentor Lesson Guide Lessons 1 - 6. Disponível em: <https://technovationchallenge.org/wp-content/uploads/2014/01/TeacherMentorLessonGuide_L1_6.pdf>. Acessado em: Junho de 2018.

Tic Educação (2016) Marco Referencial Metodológico para a Medição do Acesso e Uso das Tecnologias de Informação e Comunicação (TIC) na Educação. Centro Regional de Estudos para o Desenvolvimento da Sociedade da Informação.

Trochim, W. M. K.; Donnelly, J. P. (2008) Research methods knowledge base (3rd ed.). Mason, OH: Atomic Dog Publishing.

Wagner, A. et al. (2013) Using App Inventor In A K-12 Summer Camp. In: Proc. of the 44th ACM Tech. Symposium on Computer Science Education, Colorado, USA.

Web Content Accessibility Guidelines 2.0. (2008) W3C Recommendation. Disponível em: <https://www.w3.org/TR/2008/REC-WCAG20-20081211/>. Acessado em: Abril de 2019.

Wing, J. M. (2006) Computational Thinking. Communications of the ACM, 49(3), 33-36.

Wohlin, C. et al. (2012) Experimentation in Software Engineering. Berlin: Springer.

Zapata, B. C. et al. (2014) Mobile Phrs Compliance With Android And Ios Usability Guidelines. Journal of Medical Systems. 38(81).

Zen, K.; Iskandar, D. N. F A., Linang, O. (2011) Using Latent Semantic Analysis for automated grading programming assignments. In: Proc. of the Int. Conf. on Semantic Technology and Information Retrieval, Putrajaya, Malaysia.