UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Angélica Siqueira de Souza

UMA ABORDAGEM PARA AVALIAÇÃO DE ORIGINALIDADE DE DESIGN DE INTERFACE DE USUÁRIO DE APLICATIVOS MÓVEIS UTILIZANDO TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL PARA A EDUCAÇÃO BÁSICA

Florianópolis
2022

Angélica Siqueira de Souza

UMA ABORDAGEM PARA AVALIAÇÃO DE ORIGINALIDADE DE DESIGN DE INTERFACE DE USUÁRIO DE APLICATIVOS MÓVEIS UTILIZANDO TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL PARA A EDUCAÇÃO BÁSICA

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

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

Florianópolis

2022

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

Souza, Angélica Siqueira de
Uma abordagem para avaliação de originalidade de design de interface de usuário de aplicativos móveis utilizando técnicas de inteligência artificial para a educação básica / Angélica Siqueira de Souza ; orientador, Christiane Anneliese Gresse von Wangenheim, 2022.
119 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 2022.

Inclui referências.

1. Ciência da Computação. 2. Ciência da Computação. 3. Ensino de computação. 4. Criatividade. 5. Inteligência Artificial. I. von Wangenheim, Christiane Anneliese Gresse . II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

Angélica Siqueira de Souza

Uma abordagem para avaliação de originalidade de design de interface de usuário de aplicativos móveis utilizando técnicas de inteligência artificial para a educação básica

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca, em 21 de novembro de 2022, pela banca examinadora composta pelos seguintes membros:

Prof. Jean Carlo Rossa Hauck, Dr.
Universidade Federal de Santa Catarina

Prof. Jônata Tyska Carvalho, Dr.
Universidade Federal de Santa Catarina

Profa Vera Rejane Niedersberg Schuhmacher, Dra.
Universidade Federal de Santa Catarina

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

Documento assinado digitalmente
Mateus Grellert da Silva
Data: 02/01/2023 20:59:56-0300
CPF: ***.664.940-**
Verifique as assinaturas em https://v.ufsc.br

Coordenação do Programa de Pós-Graduação

Documento assinado digitalmente
Christiane Anneliese Gresse Von Wangenh
Data: 02/01/2023 11:58:40-0300
CPF: ***.218.819-**
Verifique as assinaturas em https://v.ufsc.br

Profa. Christiane Anneliese Gresse von Wangenheim, Dra.
Orientadora

Florianópolis, 2022.

Este trabalho é dedicado à minha família e a todos que me apoiaram durante todo o processo de desenvolvimento desta dissertação.

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus por ter me concedido a dádiva da vida, a saúde e tranquilidade mesmo quando as coisas pareciam estar perdidas no decorrer deste trabalho.

A minha orientadora, Prof.ª Dr. rer. nat. Christiane Gresse von Wangenheim, PMP pelas suas orientações, diligência, paciência e apoio durante todo o período de desenvolvimento desta dissertação. Professora, suas orientações me levaram a aprender muito.

Ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina, seus professores e técnico-administrativos, pela oportunidade, pelo auxílio e pelo conhecimento compartilhado.

Aos meus colegas do GQS (Grupo de Qualidade de Software), do INCoD (Instituto Nacional para Convergência Digital), da Iniciativa Computação na Escola, em especial à Nathalia Alves por toda tradução, ajuda, paciência e apoio durante todo o processo.

Agradeço aos professores que participaram como membros da banca pelos seus comentários e sugestões enriquecedoras.

Por último, mas não menos importante à minha família, que sempre contribuiu para meu desenvolvimento pessoal e profissional, em especial ao meu noivo, Lucas Melo da Silva, por compreender todas as nossas renúncias durante todo o processo, por sua ajuda, revisões, paciência, compreensão e apoio no decorrer desta dissertação.

Ensinar não é transferir conhecimento, mas criar possibilidades para a sua produção ou a sua construção. Quem ensina aprende ao ensinar e quem aprende ensina ao aprender.

Paulo Freire

RESUMO

Com o avanço da tecnologia no século XXI, é necessária uma adaptação da população ao novo modo de se viver, seja no trabalho ou na sua vida pessoal e convivência em sociedade. Isto faz com que seja necessário o desenvolvimento de novas habilidades do século XXI. Dentre estas novas habilidades destaca-se a criatividade, especialmente num mundo caracterizado pela automação. A criatividade tem diversas definições, mas em geral é definida como a capacidade de gerar ideias e soluções novas, originais ou surpreendentes. Uma das formas encontradas para estimular a criatividade é ensiná-la por meio da computação já na Educação Básica, como por exemplo na criação de aplicativos móveis utilizando design de interface de usuário (UI) originais, o aluno pode desenvolver criatividade, suas ideias e competências. Porém, se questiona como avaliar a originalidade especificamente em relação ao design de interface de usuário (UI) dos aplicativos móveis sendo criados pelo estudante para acompanhar o progresso da aprendizagem. Neste contexto, o objetivo deste trabalho é desenvolver um modelo de avaliação da originalidade de esqueleto do design de interface de usuário no nível de *wireframe* de apps Android, desenvolvidos com App Inventor, no contexto de avaliação de criatividade de aprendizagem do aluno na Educação Básica. O modelo desenvolvido possibilita a avaliação da originalidade dos apps desenvolvidos como resultados do processo de aprendizagem de forma automatizada utilizando medidas de similaridade que avaliam a originalidade de aplicativos criados no App Inventor, extraindo cada componente de UI presente nos aplicativos móveis e atribuindo uma nota para seu nível de originalidade. Assim, espera-se contribuir para melhorar a habilidade de criatividade, fornecendo feedback aos alunos e professores no processo de aprendizagem.

Palavras-chave: Criatividade, Originalidade, Ensino de computação, App Inventor, Inteligência Artificial.

ABSTRACT

With the advancement of technology in the 21st century, it is necessary for the population to adapt to the new way of living, whether at work or in their personal life and living in society. This makes it necessary to develop new 21st century skills. Among these new skills, creativity stands out, especially in a world characterized by automation. Creativity has several definitions, but in general it is defined as the ability to generate new, original or surprising ideas and solutions. One of the ways found to stimulate creativity is to teach it through computing already in Basic Education, such as in the creation of mobile applications using original user interface (UI) design, the student can develop creativity, their ideas and skills . However, the question is how to assess originality specifically in relation to the user interface (UI) design of mobile applications being created by the student to track learning progress. In this context, the objective of this work is to develop a model for evaluating the skeleton originality of user interface design at the wireframe level of Android apps, developed with App Inventor, in the context of evaluating student learning creativity in Basic Education. The developed model makes it possible to evaluate the originality of the apps developed as a result of the learning process in an automated way using similarity measures that evaluate the originality of apps created in App Inventor, extracting each UI component present in the mobile apps and assigning a grade to its originality level. Thus, it is expected to contribute to improve the ability of creativity, providing feedback to students and teachers in the learning process.

Keywords: Creativity, Originality, Computer Teaching, App Inventor, Artificial Intelligence.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

App	Aplicativo
AP	*Average Precision*
BNCC	Base Nacional Comum Curricular
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CNN	*Convolutional Neural Networks*
CSTA	*Computer Science Teachers Association*
CUDA	*Compute Unified Device Architecture*
ECD	*Evidence-Centered Assessment Design*
GPS	*Global Positioning System*
GPU	*Graphics Processing Unit*
GQM	*Goal Question Metric*
GUI	*Graphical User Interface*
IA	Inteligência Artificial
IOU	*Intersection over Union*
UI	Interface de Usuário
INCOD	Instituto Nacional para Convergência Digital
INE	Departamento de Informática e Estatística
INEP	Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira
LDA	*Latent Dirichlet Allocation*
LWCSE	*Largest Weighted Common Subtree Embeddings*
mAP	*mean Average Precision*
ML	*Machine Learning*
PADI	*Principled Assessment Designs for Inquiry*
PPGCC	Programa de Pós-Graduação em Ciência da Computação
QR	*Quick Response*
RCNN	*Region Based Convolutional Neural Networks*
RNN	*Recurrent Neural Network*
SBC	Sociedade Brasileira de Computação
TF-IDF	*Term Frequency–Inverse Document Frequency*
UFSC	Universidade Federal de Santa Catarina

UMC Use Modifique Crie

UnCP Universo de Comparação do Produto

UX *User Experience*

VGGNet *Very Deep Convolutional Networks*

WCAG *Web Content Accessibility Guidelines*

W3C *World Wide Web Consortium*

YOLO *You Only Look Once*

SUMÁRIO

1 INTRODUÇÃO

Esse capítulo apresenta uma contextualização do problema que se almeja solucionar neste trabalho e o método de pesquisa adotado para tal. São apresentados os objetivos gerais e específicos, bem como a delimitação do escopo deste trabalho.

1.1 CONTEXTUALIZAÇÃO

Habilidades de aprendizagem e inovação são o que separa os cidadãos que estão preparados para uma vida cada vez mais complexa e competitiva no mundo de hoje daqueles que não estão. Umas das habilidades importantes do século XXI é a criatividade (CAVALLO et al., 2016), especialmente num mundo caracterizado pela automação (WEF, 2020). Assim, é essencial promover o desenvolvimento da criatividade dos alunos já a partir da Educação Básica (CAVALLO et al., 2016).

Na Educação Básica a criatividade está tipicamente associada diretamente às atividades no ensino da arte, música e literatura, porém, o ensino da computação pode também apoiar neste processo importante e desafiador na vida dos cidadãos (ALVES et al., 2020a; YADAV; COOPER, 2017; CLEMENTS, 1995). A computação pode promover criatividade quando os cidadãos deixam de ser apenas consumidores de tecnologia e passa a construir artefatos computacionais que podem ter um impacto significativo na sociedade (YADAV; COOPER, 2017). Com a criação desses artefatos computacionais o aluno pode desenvolver criatividade com o objetivo de praticidade, expressão pessoal ou para abordar uma questão social (P21, 2017) desenvolvendo sua criatividade, suas ideias e competências (ALVES et al., 2020a). A criatividade também envolve um conjunto de processos de pensamento que segue os fundamentos da computação, como a observação, imaginação, visualização, abstração, criação e identificação de padrões, que podem apoiar no seu desenvolvimento (YADAV; COOPER, 2017).

Existem diversas definições da criatividade, mas em geral ela é definida como a capacidade de gerar ideias e soluções novas, originais ou surpreendentes (WALIA, 2019). Até hoje, a criatividade é classificada seguindo os 4Ps de Rhodes que é amplamente utilizada (ALVES et al., 2021b). Essa definição refere a criatividade em pessoa (personalidade, intelecto, temperamento, etc), produto (ideias tangíveis na forma de palavra, artefatos

computacionais etc), processo (motivação, percepção, aprendizagem, etc) e/ou ambiente (relação entre os seres humanos e o seu ambiente) (ALVES et al., 2021b). Focando no produto, a criatividade é tipicamente definida pelos subfatores de originalidade, utilidade e condensação, visando que um produto criativo deve ser original (novo), útil e apropriado (RUNCO; JAEGER, 2012).

Atualmente o ensino de computação na Educação Básica vem adotando metodologias ativas, levando aos alunos a criação de artefatos computacionais como aplicativos móveis (GROVER; PEA, 2013). Uma das estratégias para criar artefatos computacionais com criatividade no ensino de computação é o ciclo "Use-Modifique-Crie" (UMC) (LEE et al., 2011). A UMC é um ciclo que ocorre durante o estágio de criação, que é a primeira etapa do aprendizado, seja pelo uso e análise de um determinado artefato computacional, a segunda etapa seja recriando e modificando um artefato já existente e a terceira etapa seja criando um novo artefato computacional (LEE et al., 2011). Assim, principalmente durante a etapa de criação, os alunos desenvolvem a capacidade de gerar ideias, soluções originais e úteis (ALVES et al., 2020a).

Uma das formas de se ensinar conceitos e práticas de computação tem sido por meio de desenvolvimento de apps usando o App Inventor (MIT, 2020), que é um ambiente de programação visual baseado em blocos, que permite a quaisquer pessoas, de diferentes faixas etárias, criar apps para Android, estimulando as ideias, criatividade e o desenvolvimento tecnológico (PATTON et al., 2019). Com o App Inventor é possível programar um app funcional utilizando os conceitos de programação, como execução condicional, eventos, operadores matemáticos e lógicos, como também criar o design de interação e interface (PATTON et al., 2019). O App Inventor permite definir diversos designs de interação, como entrada de dados via digitação de texto, QR code ou GPS ou acelerômetro pelos sensores do celular. Referente ao design de interface de usuário (UI) permite em nível de esqueleto adicionar componentes visuais como botão, caixa de seleção, escolha de data, imagem, entre outros (MIT, 2020). Segundo Garrett (2011), o esqueleto é uma das dimensões de design de UI e determina a forma visual da tela, a apresentação e disposição de todos os elementos necessários para que a interação ocorra com uma funcionalidade existente na interface. Ainda segundo Garrett (2011), é no nível de esqueleto que são criados os *wireframes* que são representações hierárquicas da interface, apenas com informações básicas como tipos de

elementos e suas posições, sem detalhes de design visual. Assim, é possível configurar o posicionamento e alinhamento dos componentes no nível de design visual e também definir o tamanho, cor, fonte, etc (MIT, 2020). Desta forma, o App Inventor também suporta o desenvolvimento de design de interação e interface de forma criativa.

Visando o desenvolvimento da criatividade por meio do ensino de computação utilizando desenvolvimento de apps é importante dar um *feedback* ao aluno, avaliando o seu progresso de aprendizagem (ALVES et al., 2020a). Considerando que o ensino de computação é tipicamente feito adotando metodologias ativas[1] e tarefas abertas, a avaliação pode ser baseada em desempenho considerando o artefato computacional criado pelo aluno como resultado da aprendizagem (ALVES et al., 2020a). Esta avaliação baseada no desempenho é tipicamente definida por meio de rubricas confiáveis e válidas, que possibilitam avaliar a originalidade do artefato criado pelo aluno comparando com artefatos desenvolvidos por outros alunos no mesmo contexto educacional (ALVES et al., 2020a).

Na avaliação da criatividade enfocando na originalidade de um app, podem ser consideradas várias dimensões como objetivo, escopo (conteúdo e funcionalidade) e código-fonte, além da originalidade do design de interface do app (ALVES et al., 2020a). Porém, realizar a avaliação do artefato criado pelo aluno objetivando a criatividade e a originalidade não é algo trivial, pois o julgamento humano pode sofrer influências e preferências variáveis de acordo com sua vivência, tornando uma avaliação subjetiva, que não há uma “verdade absoluta” (ZEN; VANDERDONCKT, 2016; ALVES et al., 2020a).

Observando esta questão de falta de concordância entre avaliadores humanos e também para reduzir esforço e tempo dos professores para a realização deste tipo de avaliação na Educação Básica, visa-se automatizar este tipo de avaliação, permitindo que o aluno também tenha o acompanhamento em tempo real do *feedback* (ALVES et al., 2020a). Espera-se com a automação uma avaliação mais objetiva, com maior confiabilidade e validade, sem efeito de fatigas do professor e preferências injustas de certos alunos, possibilitando que a avaliação seja realizada com menos esforço, menos custo e com maior rapidez (ALVES et al., 2020a; BEATY; JOHNSON, 2020).

[1] “por meio das experiências os aprendizes conseguem construir modelos mentais, desenvolver ideias, concepções, conceitos e estratégias pessoais, ou seja, eles conseguem explorar seus processos cognitivos e as relações interpessoais, as interações do sujeito com os objetos, e dele com o mundo”. (FERREIRA, 2020, p. 49)

1.2 PROBLEMA

Atualmente existem apenas algumas abordagens para a avaliação de artefatos computacionais na Educação Básica que são automatizadas por meio de ferramentas de software (ALVES et al., 2019). De maneira geral, a maioria das ferramentas como por exemplo o Dr. Scratch (MORENO-LEÓN; ROBLES, 2015) e CodeMaster (ALVES et al., 2019) avaliam conceitos de algoritmos e programação.

Com relação a avaliação automática de qualidades do design de interface de usuário (UI) existem poucas ferramentas de software automatizadas incluindo a ferramenta CodeMaster UI Design (SOLECKI et al., 2020), que avalia a conformidade com diretrizes de design de apps Android e a ferramenta *Appsthetics* (MARTINS, 2019) que é um modelo que utiliza *deep learning* para avaliar automaticamente a estética visual de projetos desenvolvidos de apps Android. Outras abordagens propostas voltadas à avaliação do design de UI se referem a abordagens manuais (DENNER et al., 2012; FUNKE et al., 2017) que analisam componentes relacionados à usabilidade (ALVES et al., 2019). Algumas abordagens manuais analisam também componentes relacionados ao conteúdo do programa, como a criatividade e a estética (WERNER et al., 2012). A avaliação da criatividade, por se tratar de um aspecto complexo, é difícil de automatizar, o que se reflete no fato de que nenhuma abordagem automatizada com relação a esse critério foi encontrada (ALVES et al., 2019).

As ferramentas de software automatizadas para avaliação de artefatos computacionais na Educação Básica em geral, adotam a análise do código de maneiras diferentes, incluindo a análise estática ou dinâmica (ALVES et al., 2019). Mais recentemente começaram também a utilização de técnicas de *Machine Learning*, como redes neurais convolucionais para avaliar o grau da estética da interface de usuário (MARTINS, 2019). No entanto, nenhuma das ferramentas encontradas busca identificar o grau de originalidade de design de UI. Sendo assim, observa-se uma lacuna de abordagens para avaliar a originalidade de apps no contexto educacional.

Assim, o presente trabalho enfoca na pergunta de pesquisa se**:** É possível avaliar a originalidade do produto a partir de design de interface (componentes de UI) de usuário de apps criados com o App Inventor no contexto da Educação Básica de forma automatizada?

1.3 OBJETIVOS

Objetivo Geral. Desenvolver um modelo de avaliação da originalidade de esqueleto do design de interface de usuário no nível de *wireframe* de apps Android, desenvolvidos com App Inventor, no contexto de avaliação de criatividade de aprendizagem de alunos da Educação Básica. São adotadas técnicas de Inteligência Artificial, incluindo Medidas de Similaridade e *Deep Learning*, para automatizar a análise da originalidade. Enfoca-se neste trabalho especificamente na dimensão de esqueleto do design de interface e dentro dessa dimensão especificamente o grau de diferenças em relação aos componentes de UI com base no modelo de Garrett (2010) apresentado na Figura 1.

Figura 1 – Dimensões de Design de UI

Plano		Inclui
Objetivo		**Objetivos do sistema de software** Grau de diferença em relação ao objetivo do aplicativo.
Escopo		**Especificações funcionais** Grau de diferença em relação às funcionalidades dos aplicativos.
		Requisitos de conteúdo Grau de diferença em relação ao conteúdo dos aplicativos.
Design de UI	Estrutura	**Design de interação** Grau de diferenças com relação à entrada/saída de aplicativos (sensores, mídia, conectividade, etc).
		Arquitetura de informação Grau de diferenças em relação às informações apresentadas pelos aplicativos.
	Esqueleto	**Design de interface** Grau de diferenças em relação aos componentes da GUI, posicionamento e novas interfaces.
		Design de navegação Grau de diferenças em relação ao fluxo de navegação.
		Design de informação Grau de diferenças em relação à maneira como as informações são apresentadas.
	Aparência	**Design visual** Grau de diferenças com relação às cores, tipografia e imagens e ícones.
Código		**Código fonte** Grau de diferenças em relação ao JSON do projeto (representado como um vetor de recursos)

Fonte: Elaborado pela autora (2021) adaptando Alves et al. (2020a)

Objetivos Específicos. Os objetivos específicos do presente trabalho são:

Objetivo 1. Sintetizar a fundamentação teórica sobre originalidade, design de interfaces de usuário de aplicativos móveis e inteligência artificial.

Objetivo 2. Analisar o estado da arte em relação a análise automática da originalidade de design de interfaces de apps.

Objetivo 3. Analisar o contexto da avaliação de ensino da computação na Educação Básica.

Objetivo 4. Desenvolver um modelo automatizado de avaliação da originalidade de design de interface na dimensão de esqueleto de apps Android desenvolvidos com App Inventor.

Delimitações. O conceito de originalidade se limita ao contexto do construto de criatividade especificamente a respeito do produto, com foco exclusivamente nas características da originalidade. As demais características de criatividade referentes a adequação e condensação não são abordadas. Também não serão abordados outros tipos de criatividade como da pessoa, processo ou ambiente. Também não são considerados outros focos de originalidade como por exemplo na detecção de clones, plágio, entre outros.

O foco do presente trabalho é voltado ao design de interface de apps. Com relação ao design são abordados apenas aspectos do design de interface no nível de *wireframe*, outras dimensões do artefato como funcionalidade, escopo ou código não serão consideradas. Em relação à dimensão de esqueleto e enfocado somente no design de interface. São incluídos somente critérios que são verificáveis e automatizados para projetos criados com o ambiente de programação App Inventor para a plataforma Android. Não são considerados outros tipos de artefatos computacionais como jogos ou web sites.

1.4 METODOLOGIA

Este trabalho é classificado quanto à natureza como uma pesquisa exploratória aplicada e quanto à sua abordagem como uma pesquisa de multimétodos (SAUNDERS et al., 2019). Exploratória, pois é direcionada conforme os resultados já obtidos e de natureza aplicada, pois busca resolver questões específicas na análise automatizada de aplicativos criados com linguagens de programação baseadas em blocos. A pesquisa possui abordagem multimétodos, pois são utilizadas técnicas quantitativas e qualitativas, envolvendo pesquisa bibliográfica (CORDEIRO et al., 2007; GIL, 2010), mapeamento sistemático da literatura (PETERSEN et al., 2008; PETERSEN et al., 2015), entre outros.

A metodologia de pesquisa utilizada neste trabalho é dividida nas seguintes etapas conforme apresentado na Tabela 1.

Tabela 1 – Etapas e métodos de pesquisa

Etapa	Atividades	Métodos	Resultados
Etapa 1 Fundamentação Teórica	Analisar a teoria sobre design de interface de apps	Revisão narrativa (CORDEIRO et al., 2007; GIL, 2010)	Fundamentação Teórica

<table>
<tr><td colspan="2" rowspan="2"></td><td>Analisar a teoria sobre originalidade de apps no contexto da criatividade</td><td colspan="2" rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>Analisar a teoria sobre Machine Learning</td></tr>
<tr><td colspan="2" rowspan="3">Etapa 2
Estado da arte</td><td>Definir protocolo da revisão</td><td colspan="2" rowspan="3">Mapeamento sistemático da literatura
(PETERSEN et al., 2008; PETERSEN et al., 2015)</td><td rowspan="3">Análise do estado da arte</td></tr>
<tr><td>Executar busca e selecionar artigos relevantes</td></tr>
<tr><td>Extrair e analisar informações relevantes</td></tr>
<tr><td colspan="2">Etapa 3
Análise do domínio</td><td>Identificar orientações curriculares, análise do contexto educacional ferramentas e formas de representação na avaliação da criatividade de artefatos computacionais</td><td>Literatura Resenhas
(ALVES et al., 2021a; ALVES et al., 2021b)
Orientações Curriculares
(CSTA, 2016; MEC, 2022; SBC, 2022)
(MISLEVY, ALMOND e LUKAS, 2003)</td><td rowspan="4">ECD
(MISLEVY et al, 2003)</td><td>Background (Conhecimento)</td></tr>
<tr><td colspan="2">Etapa 4
Modelagem de domínio</td><td>Determinar as dimensões do produto de criatividade de avaliação a serem avaliadas</td><td>PADI (SEERATAN e MISLEVY, 2008).</td><td>Modelo de Domínio</td></tr>
<tr><td rowspan="3">Etapa 5
Estrutura de avaliação conceitual</td><td>Iteração 1</td><td>Criar modelo de evidência de originalidade utilizando medidas de similaridade.
Identificar e extrair os componentes de UI
Definir o Universo de Referência
Analisar e definir a medida de similaridade</td><td>Medida de Similaridade (POLAMURI, 2015)
Medida de Distância (SHIRKHORSHIDI; AGHABOZORGI; WAH, 2015)</td><td rowspan="3">Modelo de evidência</td></tr>
<tr><td>Iteração 2</td><td>Criar modelo de evidência de originalidade utilizando Deep Learning
Analisar os requisitos
Criar conjunto de dados para treinamento do modelo
Rotular os dados
Selecionar e treinar o modelo
Calcular o valor de similaridade
Avaliar o desempenho do cálculo do valor de similaridade de um app</td><td>Processo de desenvolvimento de modelo de Machine learning (LEARNING, 2020; AMERSHI, 2019; RAMOS et tal., 2020; WATANABE et al., 2019)</td></tr>
<tr><td colspan="3">Calcular a nota de originalidade</td></tr>
</table>

Fonte: Elaborado pela autora (2022)

Etapa 1 - Fundamentação teórica. Com base em estudos, análises e sínteses de conceitos a respeito das teorias que discordem sobre os temas a serem abordados nesse trabalho é apresentada a fundamentação teórica. A metodologia utilizada para a fundamentação de conceitos foi a revisão narrativa (CORDEIRO et al., 2007; GIL, 2010).

Etapa 2 – Estado da arte. Nesta etapa é realizado um mapeamento sistemático da literatura seguindo o processo proposto por Petersen et al. (2008) e Petersen et al. (2015) para identificar e analisar modelos de análise automatizado da originalidade de design de interfaces de usuário de apps atualmente sendo utilizados.

Etapa 3 - Análise do domínio. O domínio é analisado com base nos resultados da literatura resenhas (ALVES et al., 2021a; ALVES et al., 2021b) e orientações curriculares (CSTA, 2016; MEC, 2022; SBC, 2022) e a análise do contexto educacional com base em pesquisas existentes. Nesta etapa, reunimos informações sobre originalidade em computação, aspectos relevantes do aluno, aspectos relevantes das escolas brasileiras e aspectos dos professores brasileiros (MISLEVY et al., 2003).

Etapa 4 - Modelagem de domínio. Para modelar o domínio, expressamos a avaliação na forma narrativa usando os padrões de design do *Principled Assessment Designs for Inquiry (PADI)* (SEERATAN e MISLEVY, 2008). Aqui, foi definido o modelo do aluno considerando foco e conhecimentos, habilidades e habilidades adicionais; modelo de tarefa, definindo o produto de trabalho potencial, característica e variáveis características (MISLEVY; HAERTEL, 2006) considerando a originalidade que é uma das subdimensões da criatividade do produto.

Etapa 5 - Estrutura de avaliação conceitual. Seguindo a metodologia de *Evidence Centered Design* (ECD) (MISLEVY et al., 2003) para realizar a estrutura de avaliação conceitual. A estrutura de avaliação conceitual é dividida em duas iterações:

Iteração 1. Modelo de evidência utilizando medidas de similaridades tem como base no estado da arte de abordagens para a busca de componentes de UIs, incluindo a definição de formas alternativas de representação dos dados. São adotadas formas alternativas de representação do vetor de features (frequência binária, bruta, TF-IDF). Além disso, é feita a seleção de medidas de similaridade adequadas para as características neste contexto. Isso inclui também a implementação em *Python* usando as bibliotecas *scikit-learn* (SCIKIT-LEARN, 2021), pandas (PANDAS, 2021) e *numpy* (NUMPY, 2021) em *Notebooks Jupyter* no *Google Colab* (COLAB, 2022).

Iteração 2. Modelo de evidência para avaliação da originalidade de design de interfaces de apps abordando o processo de desenvolvimento de redes neurais/*deep learning*, inclui detectar automaticamente os componentes de UI utilizando *Deep Learning* para identificar a posição, tamanho e o tipo do componente de UI nos *screenshots* do novo app e identificar semelhança entre apps a partir do *layout* comparando os componentes de UI e o *layout* das telas do app novo com o universo de referência identificando um grau de similaridade *pairwise* com todos os apps no universo.

1.5 CONTRIBUIÇÕES

O presente trabalho está inserido na linha de pesquisa de Engenharia de Software do PPGCC/INE/UFSC, relacionado ao tema Interação Humano-Computador e qualidade de software especificamente a criatividade/originalidade de artefatos computacionais (PPGCC,

2020). A seguir, são apresentadas as contribuições deste trabalho no âmbito científico, tecnológico e social:

Contribuição científica. A principal contribuição científica é um modelo de avaliação de originalidade de design de interface de apps no contexto da avaliação da criatividade no contexto educacional. Também espera um conjunto de dados referente à originalidade e dados de avaliação comparando diversos tipos de técnicas de Inteligência Artificial.

Contribuição tecnológica. A contribuição tecnológica resultante deste trabalho é uma ferramenta que utiliza medidas de similaridades para avaliar a originalidade de aplicativos criados no App Inventor, extraindo cada componente de UI presente nos aplicativos móveis a partir do código de projetos App Inventor e atribuindo uma nota para seu nível de originalidade, contribuindo para a avaliação do desenvolvimento da criatividade de alunos na educação básica no contexto de desenvolvimento de aplicativos.

Contribuição social. O presente trabalho espera colaborar com o ensino e a avaliação da computação de forma ampla na educação básica. Considerando que é essencial fornecer um *feedback* por meio da avaliação para o processo de aprendizagem, espera-se que o presente trabalho auxilie na avaliação e no *feedback* da aprendizagem de competências de criatividade em relação ao design de interface, fornecendo um *feedback* no contexto do ensino da computação na Educação Básica. O presente trabalho também pode diminuir o esforço de avaliação dos educadores e designers instrucionais, bem como de professores, resolvendo também questões de preferência, favoritismo, etc. Além de colaborar com a inclusão digital, fornecendo conceitos e práticas da computação na educação básica, possibilitando que a maioria da população possa aprendê-los, favorecendo a preparação e o crescimento de diversas carreiras.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados conceitos essenciais Design de Interfaces de Usuário (UI) com foco em aplicativos móveis, seguido por uma análise teórica da originalidade de apps no contexto da criatividade. Este capítulo apresenta também uma análise teórica sobre técnicas da inteligência artificial, com foco em *deep learning*, medidas de similaridade e a avaliação no processo de aprendizagem.

2.1 DESIGN DE INTERFACE DE APPS

O Design de Interfaces de Usuário (UI), engloba princípios da organização, layout da tela, fluxos de trabalho, comportamentos interativos e linguagem visual, bem como design visual (FERREIRA et al., 2020). O Design Visual, por sua vez, tem a responsabilidade de expressar as potencialidades do sistema a partir de uma linguagem acessível ao usuário e está diretamente relacionado com a usabilidade (FERREIRA et al., 2020). Sendo assim, é fundamental que os elementos da linguagem visual de uma interface (cores, tipografia e imagens), estejam organizados levando em consideração os meta-princípios do design para atingir seu propósito, uma interface bonita e amigável (GARRETT, 2011; SCHLATTER; LEVINSON, 2013). Os meta-princípios do design são:

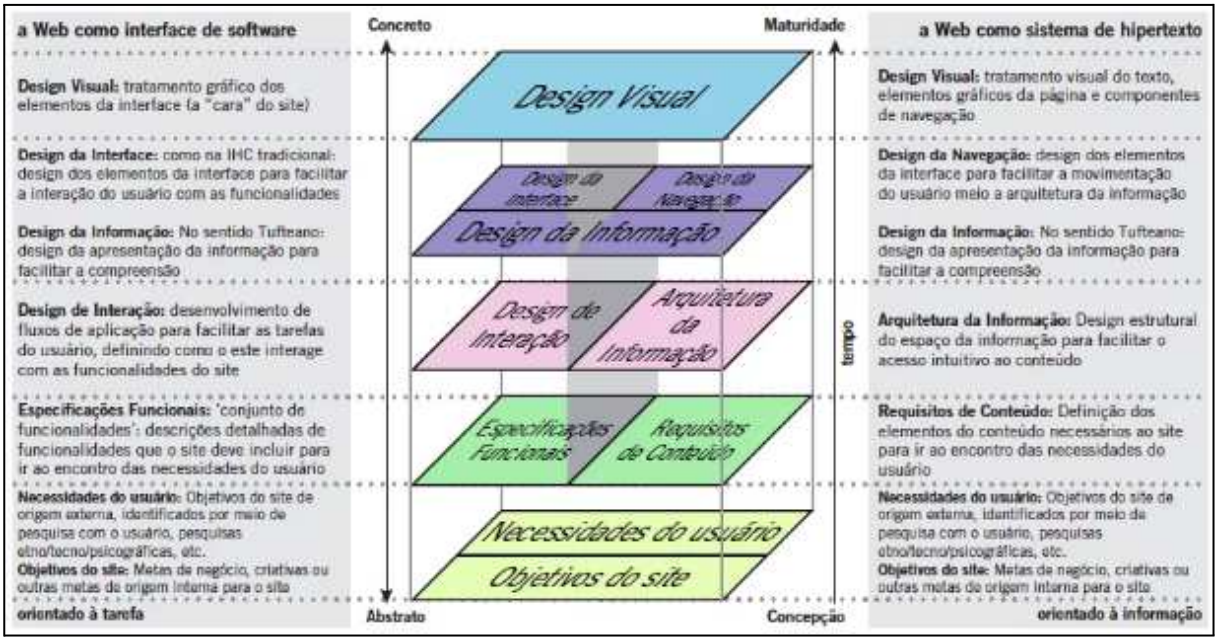
- Consistência: uso de convenções no posicionamento e tratamento dos componentes para facilitar o entendimento da interface;
- Hierarquia: destaque dos elementos mais importantes para que chamem mais atenção;
- Personalidade: efeitos na percepção dos usuários sobre o aplicativo.

Existem três características fundamentais que diferenciam o design de interface de aplicativos para dispositivos móveis do design para outras plataformas: espaço limitado da tela, interação com esses dispositivos e o contexto em que eles são utilizados. Essas características devem ser consideradas ao realizar o design de interface para que ele seja efetivo (WASSERMAN, 2010).

O *framework* criado por Garrett (2011) apresenta as diversas etapas relacionadas com o design de experiência do usuário (UX Design) e é estruturado em cinco planos ou etapas, como estratégia, escopo, estrutura, esqueleto e superfície, para entender os problemas da

experiência do usuário e as ferramentas utilizadas para resolvê-los. A Figura 2 aborda de forma resumida, a contribuição de cada plano, aplicados sequencialmente de baixo para cima, de práticas de projetos que resultam em artefatos abstratos para aquelas que resultam em artefatos concretos.

Figura 2 – Os planos do framework de Garrett



Fonte: Garrett (2011)

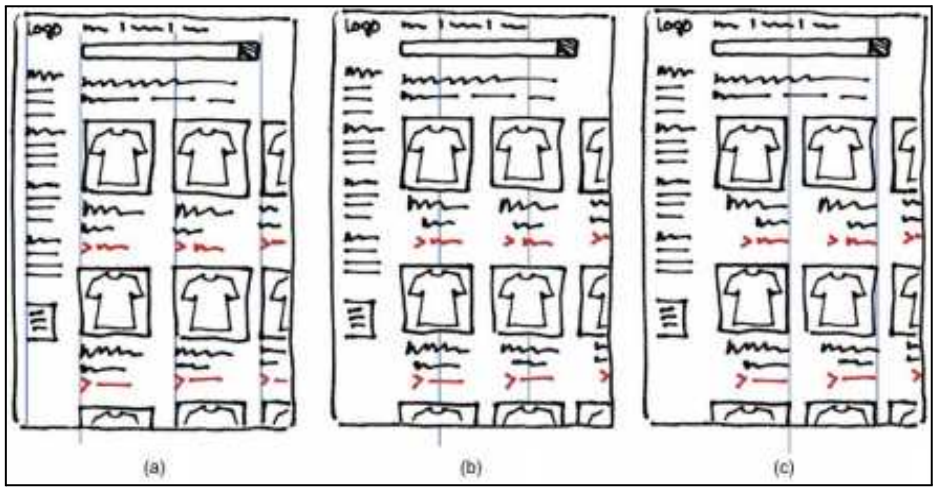
O design visual é a etapa final do design de UI, define detalhadamente a aparência do aplicativo, tornando o design da interface concreto e determinando como a informação na tela será apresentada visualmente (GARRETT, 2011). Além dos princípios de design visual, o design pode ser orientado por guias de estilo, que auxiliam nas diretrizes para facilitar o design de interfaces com usabilidade e acessibilidade, como por exemplo, para aplicativos Android, o Material Design (GOOGLE, 2021) se destaca como um guia de 24 estilos populares, sintetizando princípios clássicos de design. Para que haja consistência com o design de outros aplicativos da mesma plataforma, satisfazendo as expectativas dos usuários e facilitando o uso, é fundamental o cumprimento dessas diretrizes (WASSERMAN, 2010). O Material Design também fornece diretrizes com relação a aparência dos componentes da interface e como eles devem reagir a interações do usuário. Também auxilia nas diretrizes específicas para diversos outros componentes e aborda aspectos como layout, cor e tipografia. Para complementar essas diretrizes, as WCAG 2.1 (W3C, 2021a) fornecem recomendações para garantir a acessibilidade da interface, como por exemplo em relação ao contraste,

tamanho da letra, entre outros. Basicamente, o que usamos disso não é necessariamente para as pessoas com deficiência, mas para facilitar a legibilidade para qualquer pessoa.

A seguir, são apresentados os elementos do design de UI que envolvem o design visual e diretrizes do Material Design e das WCAG 2.1 e os princípios visuais referentes a esses elementos:

Layout: tem como objetivo posicionar os componentes da interface seguindo uma certa estrutura, ajudando os usuários a entenderem a interface. O *layout* define o tamanho dos componentes, a proximidade e o alinhamento entre eles, que afetam a percepção de quais componentes estão relacionados entre si e quais são mais importantes, conforme apresentado na Figura 3.

Figura 3 – Exemplo de uma mesma tela com diferentes alinhamentos



Fonte: Schlatter e Levinson (2013)

O *layout* depende da resolução da tela (largura e altura), que determina o espaço disponível para posicionar os componentes principalmente em dispositivos móveis, que possuem espaço de tela limitado. A resolução da tela é medida em pixels (px), o tamanho real de um pixel (o espaço físico que ele ocupa na tela do dispositivo) depende de cada dispositivo.

A resolução da tela define a proporção da tela (aspect ratio, em inglês) e a orientação do dispositivo. A proporção entre a largura e a altura define a proporção da tela, a mais utilizada entre os dispositivos móveis atualmente é 9:16 (DEVICEATLAS, 2021). Com relação a orientação da tela, pode ser retrato (se altura > largura) ou paisagem (se altura < largura) sendo que a maioria dos dispositivos móveis como smartphones e tablets, podem ser usados em ambas as orientações (W3C, 2021b), que é determinada pela posição em que o

usuário segura o dispositivo, exceto quando o aplicativo em uso força uma orientação específica.

O App Inventor disponibiliza em sua *Palette* opções de *layout* e opções de resolução de tela, também é possível alterar a orientação da tela apoiando a visualização de *layout*, conforme é apresentado na Figura 4.

Figura 4 – Opções de *layout* e resolução de tela do App Inventor

Fonte: MIT (2020)

Tipografia: define o formato das letras e demais caracteres, é composta por diversas fontes, com variações de um mesmo formato básico, como por exemplo, itálico e negrito, que não as variações mais comuns.

2.2 ORIGINALIDADE DE APPS NO CONTEXTO DA CRIATIVIDADE

A criatividade pode ser entendida de diferentes maneiras, como motivação, variação cega, pensamento divergente, mas de maneira geral diversos pesquisadores à definem como a capacidade de gerar ideias e soluções novas, originais ou surpreendentes, envolvendo a produção de ideias e artefatos novos e úteis (WALIA, 2019). A novidade é um dos fatores essenciais no processo criativo e reconhecê-la estimula a criatividade (ZHOU et al, 2017).

Uma proposta para o melhor entendimento da criatividade é classificar a definição baseado no Modelo de Criatividade dos 4Ps de Rhodes que até hoje é amplamente utilizada (ALVES et al., 2021b). Esse modelo define a criatividade em pessoa, produto, processo e ambiente, conforme apresentado na Tabela 2.

Tabela 2 – 4 Ps da criatividade

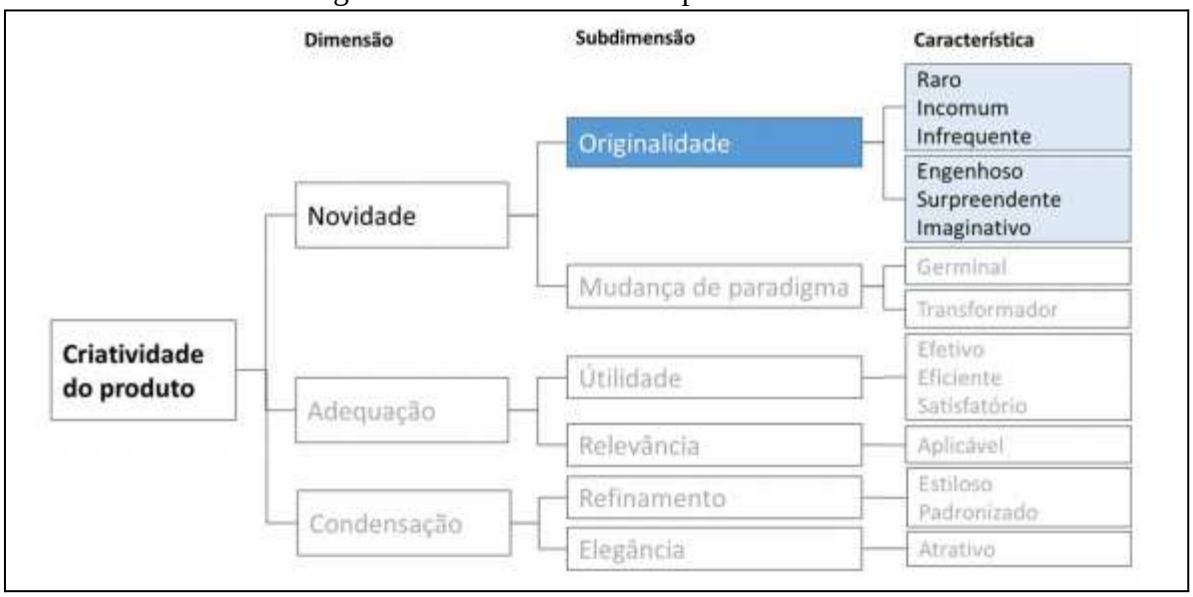
Criatividade

Pessoa	Processo	Ambiente	Produto
Personalidade; Intelecto; Temperamento; Físico; Traços; Atitudes; Autoconceito; Valores; Mecanismos de defesa; Comportamento.	Motivação por percepção; Aprendizagem; Pensamento; Comunicação.	Relacionamento entre os seres humanos e o seu ambiente.	Ideias tangíveis na forma de palavras, artefatos computacionais, tecido ou outros materiais.

Fonte: Elaborado pela autora com base em Alves et al (2021b).

Visando no presente trabalho um foco na avaliação do produto criado como resultado da aprendizagem, a criatividade com foco no produto ainda não tem um conjunto de características consolidado para sua definição (BESEMER; TREFFINGER, 1981; WALIA, 2019), porém, tipicamente a criatividade do produto é decomposto por novidade, adequação e condensação (ALVES et al., 2020a), conforme Figura 5.

Figura 5 – Características do produto criativo



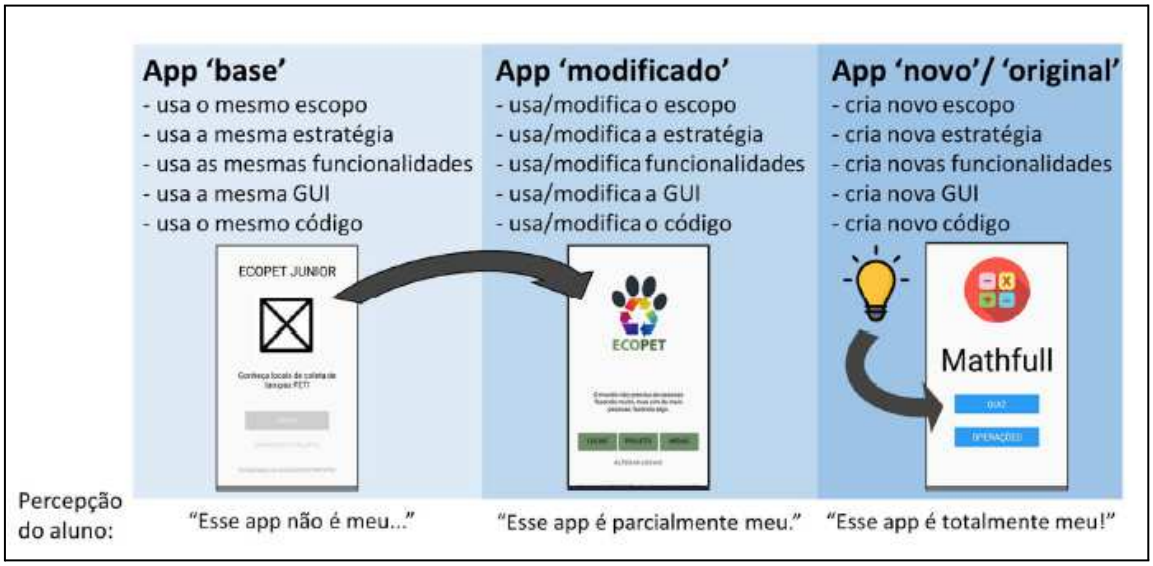
Fonte: Alves et al (2020a)

A novidade é uma dimensão da criatividade do produto, referente aos produtos originais que alteram um paradigma e é dividida entre originalidade e mudança de paradigma. A subdimensão originalidade é referente ao produto incomum ou pouco visualizado em um universo de produtos criados por pessoas com conhecimento e aprendizagem semelhantes (JACKSON; MESSICK, 1964), já a subdimensão mudança de paradigma se refere aos produtos que transformam e revolucionam uma área.

Ainda com relação à subdimensão originalidade, uma ideia ou produto criativo é considerado original se representa algo novo ou surpreendente que não existia antes (RUNCO; JAEGER, 2012). A originalidade é uma característica essencial para os produtos criativos (BESEMER; TREFFINGER, 1981), mesmo que sozinha não seja suficiente para classificar um produto como sendo criativo. Conforme Figura 1, a subdimensão originalidade tem como características um produto raro, incomum, infrequente, engenhoso, surpreendente e imaginativo (ALVES et al., 2020a).

O ciclo Use-Modifique-Crie (UMC) é utilizado no ensino de computação por meio do desenvolvimento de apps (ALVES et al., 2020a). Nesse contexto, como critério para medir a criatividade, podemos utilizar a originalidade (ALVES et al., 2020a), porém, um dos maiores desafios no desenvolvimento de apps, é distinguir projetos originais de projetos não originais (MUSTAFARAJ et al., 2017). Na etapa Use, que consiste no desenvolvimento de um app base, não se espera uma “criação” por parte do aluno. Desenvolvendo apps com o App Inventor, os usuários geralmente criam apps quase idênticos a sites populares do App Inventor (MUSTAFARAJ et al., 2017). Nesta etapa não se espera que sejam criados apps originais porque os alunos seguem um contexto já conhecido, sem ilustrar que projetaram e construíram por conta própria ou em grupos (MUSTAFARAJ et al., 2017). Já na etapa Modifique, é esperado que o aluno crie um app “parcialmente seu” sendo motivado a modificar alguma característica do app (ALVES et al., 2020a). Na etapa Crie, espera-se que o aluno tenha um nível maior de originalidade, apresentando um aplicativo criado bem diferente em relação ao aplicativo base e modificado nas etapas anteriores (ALVES et al., 2020a), conforme Figura 6.

Figura 6 – Abordagem UMC no contexto de desenvolvimento de apps



Fonte: Alves et al (2020a)

É nessa etapa que os alunos desenvolvem um aplicativo com base em suas próprias ideias e habilidades de programação atuais. Utilizando o App Inventor podemos identificar se os alunos estão aprendendo sobre programação e quais conceitos errôneos eles têm, nos concentrar em seus projetos originais e filtrando os projetos não originais (MUSTAFARAJ et al., 2017).

Exemplos de apps na etapa Crie são apresentados na Figura 7.

Figura 7 – Ilustração dos apps da etapa Crie

Fonte: Alves et al (2020a)

A avaliação de um app criado como resultado da aprendizagem, com relação à originalidade na etapa de Crie dentro do contexto de ensino de computação na Educação

Básica, envolve todas as suas dimensões (JACKSON; MESSICK, 1964). Um modelo de avaliação da originalidade de apps, deve considerar a importância das dimensões como objetivo e escopo que são incluídas no app, pois, apps com objetivo e escopo originais devem ser identificados em uma avaliação do produto criativo (ALVES et al., 2020a). As dimensões e itens baseados no modelo de Garrett (2010) para um modelo de avaliação da originalidade de apps são apresentados na Figura 8.

Figura 8 – Rubrica Modifique e Crie com o Universo de Comparação do Produto

Fonte: Alves et al (2020a)

Visando a avaliação por desempenho avaliando o design de UI criado pelo aluno neste modelo, uma rúbrica é definida para cada estágio, considerando diferentes expectativas com relação ao grau de originalidade na etapa Crie (ALVES et al., 2020a).

Ao avaliar a criatividade, é fundamental que a criatividade e seus subfatores se baseie em um universo de artefatos criados por pessoas com experiência e formação semelhantes (ALVES et al., 2020a) para que a avaliação seja de forma mais direta, comparando por exemplo, a ideia do aplicativo e seus componentes de UI, com os aplicativos existentes (JACKSON; MESSICK, 1964). O universo de referências com os quais os apps são comparados também requer definições (JACKSON; MESSICK, 1964), considerando o contexto educacional com foco no ensino do desenvolvimento de aplicativos com o App Inventor, pode ser representado, por exemplo, por aplicativos compartilhados na galeria do App Inventor (ALVES et al., 2020a). No entanto, ainda não se tem uma definição com relação a qual universo de comparação os avaliadores se baseiam de fato na hora de fazer suas

avaliações impossibilitando identificar se, na hora de avaliar, em vez de basear-se em aplicativos criados por alunos dentro de um contexto educacional, os avaliadores utilizaram seus conhecimentos e referências adquiridas ao longo de anos de uso de aplicativos semelhantes como parte de sua vida pessoal ou experiência de ensino (ALVES et al., 2020a).

Na etapa Crie é definido o Universo de Comparação do Produto (UnCP), considerando aplicativos criados por alunos com experiência e aprendizagem semelhantes, por exemplo, os aplicativos compartilhados na Galeria do App Inventor (ALVES et al., 2020a).

2.3 INTELIGÊNCIA ARTIFICIAL

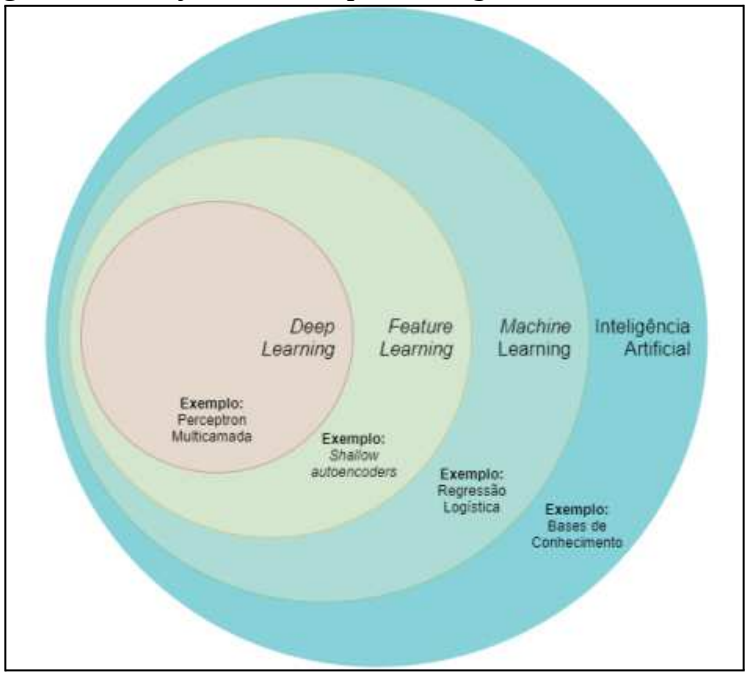
Para John McCarthy, reconhecido como pai da Inteligência Artificial (IA), a IA é “a ciência e a engenharia de criar máquinas inteligentes que tem habilidade para alcançar objetivos da mesma forma que os humanos” (INDIA, 2018). Em resumo, a Inteligência Artificial é a tentativa de representação por máquinas da inteligência humana. Um grande subcampo da área de IA é a *Machine Learning* (ML) que lida com o campo de estudos que dá a computadores a habilidade de aprender sem ser explicitamente programados (SAMUEL, 1959).

Machine Learning, aprendizado de máquina, compreende um conjunto de técnicas de ciência da computação para automatizar a aquisição de conhecimento, em vez de confiar em instruções humanas explícitas (MATHEWSON, 2019). É uma área de conhecimento que permite computadores aprenderem diretamente com exemplos e experiência na forma de dados (ROYAL SOCIETY, 2017).

Deep learning, aprendizado profundo, é uma técnica de computador que extrai e transforma os dados usando várias camadas de redes neurais, sua aplicação varia desde reconhecimento de fala humana a classificação de imagens de animais (HOWARD; GUGGER, 2020). Cada uma dessas camadas obtém suas entradas de camadas anteriores e as refina progressivamente, as camadas são treinadas por algoritmos que minimizam os erros e melhoram a precisão, assim a rede aprende a executar uma tarefa específica (HOWARD; GUGGER, 2020). *Deep learning*, aprende a representar o mundo hierarquicamente aninhada de conceitos, cada conceito é definido comparado a outro mais simples e com representações abstratas em termos de outros menos abstratas, por isso alcança grande poder e flexibilidade

(GOODFELLOW et al., 2016). *Deep learning* é um tipo de *feature learning*, que é um tipo de *machine learning* utilizado por várias abordagens de IA, mas não por todas, conforme ilustra a Figura 9.

Figura 9 – Relação entre *Deep Learning* e a área de IA



Fonte: Adaptado a partir de Goodfellow et al., 2016.

Deep learning utiliza redes neurais empilhadas, uma rede com múltiplas camadas de redes (GOODFELLOW et al., 2016).

2.3.1 Redes Neurais

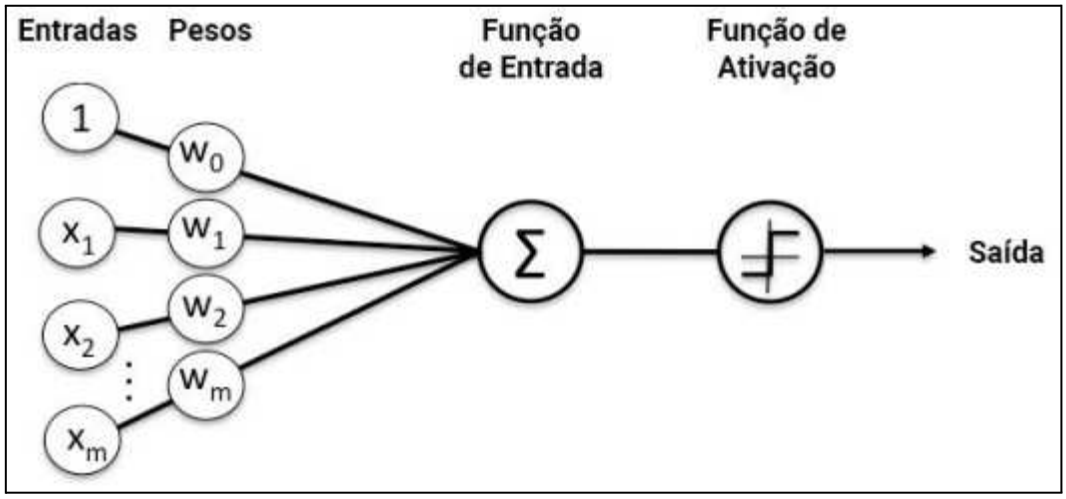
As redes neurais são uma abordagem a *Machine Learning* em que camadas de unidades computacionais estão conectadas a outra camada de uma forma que é inspirada por conexões entre neurônios no cérebro (ROYAL SOCIETY, 2017). Uma camada destas (unidades de entrada) é projetada para receber informações do mundo exterior, enquanto o outro lado da rede, uma saída camada, comunica uma decisão sobre os dados recebidos (ROYAL SOCIETY, 2017). Entre estas, outras camadas comunicam informações sobre elementos da entrada uns para os outros, que contribuem para a saída (ROYAL SOCIETY, 2017).

As redes neurais interpretam dados sensoriais por meio de uma espécie de percepção de máquina e categoriza ou agrupa entradas puras de dados, os padrões reconhecidos por elas

são numéricos, armazenados em vetores e todos os tipos de dados do mundo real, imagens, sons ou textos, precisam ser “traduzidos” (PATHMIND, 2021). Os nodos são os elementos que compõem uma rede neural, eles são conectados entre si por algum tipo de conexão que possuem peso (numérico) associado a eles (RUSSELL, 2015). Os pesos são métodos primários de memória a longo termo da rede e o processo de aprendizado são atualizações destes valores, os nodos podem ser chamados de nodos de entradas ou saída e são conectados ao ambiente externo (RUSSELL, 2015). Cada nodo, com a entrada recebida, faz a função de ativação, responsável por ativar ou não um neurônio com base em uma somatória de pesos e viés, introduzindo uma não-linearidade à saída dos neurônios, característica responsável pelo aprendizado e a realização de tarefas mais complexas (RUSSELL, 2015).

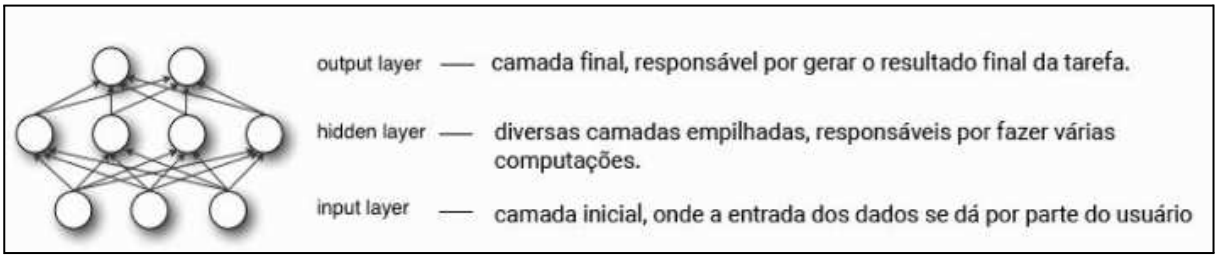
Uma camada é o conjunto das unidades que se parecem com neurônios e que “ligam e desligam” quando os dados são alimentados a eles como entrada e passam pela rede, as camadas tem como saída a entrada da camada subsequente, dando início a uma camada inicial que recebe os dados diretamente do usuário (RUSSELL, 2015). Na Figura 10 e na Figura 11, é possível visualizar um diagrama de exemplo de um nodo e a estrutura geral de uma rede, respectivamente.

Figura 10 – Diagrama exemplo de um nodo



Fonte: adaptado de PATHMIND, 2021.

Figura 11 – Diagrama exemplo de um nodo



Fonte: adaptado de SAS (2021).

Os métodos de aprendizados são:

- **Aprendizado não-supervisionado**: o computador é treinado com dados não pré-classificados por meio de tentativas de formar e/ou encontrar um padrão entre os dados contidos no conjunto de dados. Casos de uso principais são a detecção de padrões e modelagem descritiva (PATHMIND, 2021).
- **Aprendizado por reforço:** utiliza a observação e interação com o ambiente na sua tomada de decisão para maximizar o ganho ou minimizar a perda, o algoritmo aprende continuamente com suas ações, de maneira iterativa, até cobrir todos os possíveis estados, esse algoritmo é chamado de agente). Casos de uso incluem IAs para jogos de tabuleiro, mãos robóticas e automóveis autônomos (FUMO, 2017).
- **Aprendizado semi-supervisionado:** é a mescla entre os aprendizados supervisionados e não-supervisionados (PATHMIND, 2021).
- **Aprendizado supervisionado:** um conjunto de dados é classificado por um humano para que a rede neural possa começar seu processo de aprendizado por meio da análise e busca de relações entre todos os dados contidos no conjunto de dados (PATHMIND, 2021).

2.3.2 *Convolutional Neural Networks (*CNN*)*

Convolutional Neural Networks (CNN) é uma classe de rede neural profunda, mais comumente aplicada para analisar imagens visuais (VALUEVA et al., 2020). A CNN contém um extrator de recursos, que consiste em uma camada de convolução e um down-sampling camada (TIAN, 2020). Um neurônio está conectado apenas a uma parte dos neurônios na camada superior, e esta parte dos neurônios é chamada de campo receptivo local (TIAN, 2020). Uma camada convolucional geralmente contém vários mapas de características, cada mapa de características é composto de um número de neurônios, e os pesos dos neurônios são

compartilhados entre os mesmos mapas de recursos (TIAN, 2020). A maior característica do peso compartilhado é reduzir a conexão entre as várias camadas da rede, reduzir os parâmetros da rede e desempenhar um papel na prevenção do sobreajuste (TIAN, 2020). No processo de treinamento da rede, novos pesos são constantemente aprendidos e atualizados em tempo real até um peso razoável é finalmente aprendido (TIAN, 2020). A CNN tem três principais características: campo receptivo local, compartilhamento de peso e pooling (TIAN, 2020).

2.3.3 Medida de similaridade e distância

A medida de similaridade é a medida de quão parecidos são dois objetos de dados (POLAMURI, 2015). A medida de distância é um componente principal dos algoritmos de agrupamento baseados em distância. Se a distância for pequena, os recursos estão tendo um alto grau de similaridade. Considerando que uma grande distância será um baixo grau de semelhança (POLAMURI, 2015).

A semelhança é subjetiva e altamente dependente do domínio e da aplicação (POLAMURI, 2015). Geralmente, a similaridade é medida na faixa de 0 a 1 [0,1] (POLAMURI, 2015). No mundo do aprendizado de máquina, essa pontuação na faixa de [0, 1] é chamada de pontuação de similaridade (POLAMURI, 2015).

Existem as mais diversas medidas de similaridade/distância dependendo do tipo de objeto (texto, imagem, etc) sendo comparado por Shirkhorshidi et a., (2015), porém no foco do presente trabalho observa-se mais comumente o uso das medidas apresentadas na Tabela 3.

Tabela 3 – Similaridades

Medida de distância/ similaridade	Equação	Complexidade de tempo	Vantagens	Desvantagens	Aplicação
Euclidean	$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$	O(n)	Uso mais comum de medida de distância; Funciona bem com conjuntos de dados com clusters compactos ou isolados.	Sensível a outliers	Algoritmo K-means; Algoritmo Fuzzy C-means
Cosine	$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$	O(3n)	Muito eficiente para avaliar, especialmente para vetores esparsos.	Não é invariante à transformação linear	Usado principalmente em aplicativos de similaridade de documentos

Medida de distância/ similaridade	Equação	Complexidade de tempo	Vantagens	Desvantagens	Aplicação
Jaccard	$\frac{S_{com}}{s_1 + s_2 - S_{com}}$	-	medida da similaridade entre dois conjuntos	-	Comparar levantamentos de espécies em diferentes ambientes e em diferentes tempos

Fonte: Elaborado pela autora com base em Polamuri (2015) e Shirkhorshidi et al. (2015).

2.4 AVALIAÇÃO DA APRENDIZAGEM

A avaliação é um elemento central no processo de aprendizagem, que visa fornecer *feedback* ao aluno e/ou ao professor em relação à aprendizagem das competências pelo aluno (BRANCH, 2009). Ela é uma parte importante do processo de ensino e aprendizagem, pois avalia o desempenho de algo ou alguém. Sua função no processo de aprendizagem é servir para medir se o aluno sendo avaliado alcançou efetivamente os objetivos educacionais propostos (TYLER, 1949; MARTINS; GUISSO, 2019). Para o professor, a avaliação serve de acompanhamento de sua efetividade na transmissão de conhecimento, habilidades e atitudes. De acordo com Stegeman et al. (2016), o aluno precisa saber:

- o que um bom desempenho em uma tarefa;
- como seu próprio desempenho se compara com um bom;
- o que fazer para reduzir a lacuna entre os dois.

O planejamento de como a avaliação deve ser realizada decorre da definição das competências (conhecimento, habilidades e atitudes) a serem desenvolvidas pelos alunos como resultado da aprendizagem (MORAES; RODRIGUES, 2019). Utilizando a avaliação em ambientes de aprendizagem ativa uma das estratégias de desenvolver a aprendizagem é elaborar atividades baseadas em problemas, assim os resultados dessa experiência de aprendizagem prática desempenha um papel crucial e tem o potencial de ser altamente autêntico (BIALIK et al., 2016).

Segundo CSTA o objetivo de aprendizagem deve ser claro, observável e mensurável, assim, espera-se que o aluno entenda o que deve ser feito e que o professor tenha uma forma de verificar e medir o artefato/ação produzido pelo aluno (CSTA, 2016). De acordo com o design instrucional, uma avaliação pode ser classificada em: diagnóstica, formativa e somativa, conforme apresentado na Tabela 4.

Tabela 4 – Classificação de tipos de avaliação

Avaliação Diagnóstica	Avaliação Formativa	Avaliação Somativa
Realizada no início do processo e consiste em avaliar ou identificar os conhecimentos dos alunos acerca do tema antes da intervenção instrucional	Processo de coletar dados sobre a aprendizagem do aluno durante o ensino, de forma a revisar o processo de instrução para que este seja mais efetivo.	Processo de coleta de dados após o processo de ensino e tem caráter classificatório, tipicamente alocando-se uma nota a fim de determinar o grau de atingimento dos objetivos de aprendizagem

Fonte: Elaborado pela autora com base em Branch (2009).

A avaliação de atividades abertas de criação de artefatos computacionais tipicamente é realizada de forma somativa, sendo realizada ao final do processo instrucional, atribuindo-se uma nota sobre o artefato do aluno já totalmente desenvolvido.

O resultado da avaliação informado ao aluno tipicamente se dá pelo uso de notas, indicando seu desempenho de aprendizagem. A atribuição de nota pode ser feita com um valor quantitativo ou qualitativo (ALVES et al., 2019). Tipicamente na Educação básica no Brasil se adota o valor quantitativo, a nota é composta por uma escala numérica, na qual alunos que desempenharam a tarefa com máximo aproveitamento recebem a nota máxima.

A avaliação do ensino de computação pode ser baseada no desempenho analisando um artefato computacional criado pelo aluno como resultado da aprendizagem, em termos de conceitos de algoritmo e programação (ALVES et al., 2020a) de apps, design visual (SOLECKI et al., 2020), (LYE; KOH, 2014), estética visual (MARTINS, 2019), entre outros.

A automação da avaliação permite apresentar um *feedback* rápido para os alunos, reduzindo a demanda de esforço e tempo dos avaliadores. Geralmente o número de atividades educacionais é consideravelmente maior do que o número de avaliadores, tornando impraticável avaliar todas as atividades, o que pode levar a um déficit de aprendizado para os alunos (GALAN et al., 2019). Já existem abordagens para automatizar a avaliação de artefatos computacionais referentes à aprendizagem de algoritmos e programação (ALVES et al., 2019). Há também ferramentas de automação de avaliação de design de interface de usuário (SOLECKI et al., 2020) e estética visual no contexto de desenvolvimento de aplicativos na Educação Básica (MARTINS, 2019), porém estes não avaliam o grau da originalidade de design de interface.

3 ESTADO DA ARTE

Nesta seção é apresentado o levantamento do estado de arte em relação às seguintes perguntas relacionadas ao foco do presente trabalho:

1) Quais abordagens existem para automaticamente analisar a originalidade de design de interfaces de usuário de apps, focando em soluções que adotam técnicas de IA.

2) Quais abordagens existem para automaticamente extrair os componentes de design de interfaces de aplicativos de aplicativos.

O levantamento é feito por meio de mapeamentos sistemáticos da literatura seguindo o processo proposto por Petersen et al. (2008) e Petersen et al. (2015), para identificar, analisar e comparar as abordagens existentes.

3.1 DEFINIÇÃO DO PROTOCOLO DE MAPEAMENTO

O objetivo do presente mapeamento sistemático da literatura é **identificar e analisar modelos de análise automatizada da originalidade de design de interfaces de usuário de apps, focando em soluções que adotam técnicas de IA**. O mapeamento visa responder à seguinte pergunta de pesquisa: Quais modelos/ferramentas existem para automaticamente avaliar a originalidade do produto a partir de design de interface de usuário de apps Android (criados com o App Inventor no contexto da Educação Básica) usando técnicas de IA?

A pergunta de pesquisa é refinada nas seguintes perguntas de análise:

PA1. Quais modelos/ferramentas de avaliação da originalidade do design de UI de apps Android existem?

PA2. Quais as características do modelo de avaliação da originalidade do design de interfaces?

PA3. Quais técnicas de IA são adotadas pelos modelos de avaliação?

PA4. Como foi feita a preparação do conjunto de dados e a rotulação?

PA5. Como é medido e qual é o desempenho das abordagens?

Critérios de inclusão e exclusão: Os artigos relevantes foram selecionados conforme os seguintes critérios de inclusão e exclusão:

- Incluir artigos científicos e artefatos que apresentam algum modelo de avaliação da originalidade em relação ao design de interface de usuário de aplicativos móveis. Com relação a similaridade, também são considerados artigos que não necessariamente visam a detecção do grau da originalidade, mas a similaridade entre designs de interfaces;
- Incluir apenas artigos em inglês no período de 2007, ano do lançamento do primeiro smartphone (MURTAZIN, 2010), até 2021 acessíveis via Portal CAPES;
- Incluir artigos que apresentam modelos de avaliação do design visual, não limitando a abordagens para uso educacional para se ter uma visão mais ampla do estado da arte.
- São excluídos artigos focados exclusivamente em questões de malware e/ou plágio.
- Para ampliar a visão geral, a pesquisa não é limitada ao contexto educacional, porém foram excluídos artigos focados exclusivamente em questões de malware e/ou plágio.
- Em termos de qualidade, foram incluídos apenas artigos contendo informações substanciais indicando como o design de UI é avaliado de acordo com sua originalidade.

Critérios de qualidade: São considerados apenas artigos que apresentam informações substanciais para se extrair informações referente às perguntas de análise. São excluídos artigos que apresentam, por exemplo, somente um resumo de uma proposta e para os quais não são encontradas mais informações detalhadas.

Bases de dados: As buscas foram realizadas nas principais bases de dados e bibliotecas digitais da área da computação. Pesquisamos no *Google Scholar* pois é considerado aceitável como uma fonte adicional visando a minimização do risco de omissão por buscar artigos de forma mais ampla de diversas áreas de conhecimento, (PIASECKI et al., 2018).

- IEEE Xplore (http://ieeexplore.ieee.org)
- ACM Digital Library (http://portal.acm.org)

- Scopus (https://www.scopus.com)
- arXiv.org E-print Archive (https://arxiv.org)
- Google Scholar (https://scholar.google.com)
- ScienceDirect (https://www.sciencedirect.com)

String de Busca: com base na pergunta de pesquisa, buscas informais foram utilizadas para calibrar a string de busca, a qual foi definida com base nos termos relevantes da pergunta de pesquisa do mapeamento, resultando nos termos de busca, sinônimos e traduções apresentados na Tabela 5.

Tabela 5 – Termos de busca relevantes, sinônimos e tradução (inglês)

Termo	**Sinônimos**	**Tradução (inglês)**
Aplicativos móveis	Android, App Inventor	*Mobile application, Android, App Inventor, App*
Design Visual	Design de interface de usuário	*User interface, UI design, Visual Languages*
Criatividade	Originalidade, Similaridade, novidade	*Creativity, Original, Similarity, Novel*
Deep Learning	Inteligência artificial, avaliação automática, redes neurais, machine learning	*Deep learning, Artificial Intelligence*

Fonte: Elaborado pela autora (2021)

Foi necessária a inclusão do sinônimo de visual languages considerando o retorno de alguns trabalhos relevantes nas buscas informais.

Após a realização das buscas informais, uma string de busca específica foi definida para se aplicar nas bases de dados mencionadas de modo a encontrar todos os artigos relevantes previamente conhecidos, e um número satisfatório de artigos possivelmente relevantes adicionais:

("*mobile application*" *OR* "*android*" *OR* "*app inventor*" *OR* "*app*") *AND* ("*user interface*" *OR* "*UI design*" *OR* "*visual languages*") *AND* ("*creativity*" *OR* "*original*" *OR* "*similarity*" *OR* "*novel*") *AND* ("*deep learning*" *OR* "*artificial intelligence*")

Após a definição da string de busca, foi realizada a sua adaptação para executar sua aplicação em cada base de dados utilizando a linguagem específica de cada uma, conforme apresentado na Tabela 6.

Tabela 6 – Strings de buscas utilizadas nas diferentes bases de dados)

Base de Dados	***String* de busca**
ACM Digital *Library*	[[All: "mobile application"] OR [All: "android"] OR [All: "app inventor"] OR [All: "app"]] AND [[All: "user interface"] OR [All: "ui design"] OR [All: "visual languages"]] AND [[All: "creativity"] OR [All: "original"] OR [All: "similarity"] OR [All: "novel"]] AND [[All: "deep learning"] OR [All: "artificial intelligence"]] AND [Publication Date: (01/01/2007 TO 04/30/2021)]
arXiv.org *E-print Archive*	*size*: 200; *date_range*: *from* 2007-01-01 *to* 2021-12-31; *include_cross_list*: *True*; *terms*: *AND all="mobile application" OR "android" OR "app inventor" OR "app"*; *AND all="user interface" OR "UI design" OR "visual languages"*; *AND all="creativity" OR "original" OR "similarity" OR "novel"*; *AND all="deep learning" OR "artificial intelligence"*
Google Scholar	*(“Mobile Application” OR “Android” OR “App Inventor”) AND (“User Interface” OR “UI Design” OR “Visual Languages”) AND (“Creativity” OR “Original” OR “Similarity”) AND (“Deep Learning” OR "Artificial Intelligence") Período específico 2007 - 2021*
IEEE *Xplore*	("Mobile application" OR "Android" OR "App Inventor" OR "App") AND ("User interface" OR "UI design" OR "Visual Languages") AND ("Creativity" OR "Original" OR "Similarity" OR "Novel") AND ("Deep learning" OR "Artificial Intelligence") *Filters Applied*: 2007 - 2021
ScienceDirect	("android” OR “app inventor" OR "app") AND ("user interface” OR “visual languages”) AND (“original” OR “similarity”) AND ("deep learning” OR “artificial intelligence") Year: 2007-2021
Scopus	ALL ("mobile application" OR "android" OR "app inventor" OR "app") AND ("user interface" OR "UI design" OR "visual languages") AND ("creativity" OR "original" OR "similarity" OR "novel") AND ("deep learning" OR "artificial intelligence") AND PUBYEAR > 2006 AND (LIMIT-TO (LANGUAGE , "English"))

Fonte: Elaborado pela autora (2021)

3.2 EXECUÇÃO DA BUSCA

A busca dos artigos foi realizada em março e abril de 2021 pela autora do presente trabalho e revisada pela orientadora. A busca inicial resultou em 19.738 artigos, dos quais foram selecionados artigos relevantes de acordo com os critérios de inclusão, exclusão e qualidade, conforme apresentado na Tabela 7.

Tabela 7 – Resultado da busca

Base de Dados	**Quantidade de artigos resultantes da busca**	**Quantidade de artigos analisados**	**Quantidade de artigos potencialmente relevantes (título e abstract)**	**Quantidade de artigos relevantes (com base no artigo na íntegra)**
ACM Digital Library	944	150	9	4
arXiv.org e-print archive	4	4	2	1
Google Scholar	14.200	150	6	3
IEEE Xplore	10	10	1	1
ScienceDirect	766	150	3	0
Scopus	3.814	150	22	11

Base de Dados	Quantidade de artigos resultantes da busca	Quantidade de artigos analisados	Quantidade de artigos potencialmente relevantes (título e abstract)	Quantidade de artigos relevantes (com base no artigo na íntegra)
Total				11 (sem duplicatas)

Fonte: Elaborado pela autora (2021)

Com base nos resultados iniciais, foi realizado um filtro inicial baseando-se apenas no título e resumo, os que mostraram maior potencial de abordar o tema deste artigo seguindo os critérios de inclusão e exclusão, estes então passaram por uma leitura completa para garantir que o tema dos artigos são relevantes para o este trabalho.

Alguns artigos encontrados nas buscas foram excluídos por falta de enfoque na originalidade/similaridade do artefato/app, como Jeong et al. (2020) ou por falta de enfoque no design de UI, como Ichinco e Kelleher (2018) e Basu (2019).

Vários artigos como Li et al. (2019), Lyu et al. (2016) e Soh et al. (2015), também foram excluídos pelo enfoque na análise de determinados tipos de plágios/clonagem, mas não na originalidade. Por impossibilidade de acesso ao texto na íntegra também foram excluídos os artigos Zhang et al. (2020) e Hu et al. (2020).

Após a aplicação de todos os critérios, foram identificados no total 11 artigos únicos relevantes sendo que 4 resultados apareceram como resultado em pelo menos 3 bases de dados.

3.3 ANÁLISE DOS RESULTADOS

Para responder à questão de pesquisa, as informações relevantes às perguntas de análise foram extraídas dos 11 artigos relevantes. Os artigos selecionados foram lidos de forma completa e os dados extraídos pela autora e revisados pela orientadora. Nos casos em que o artigo não apresenta nenhuma informação a ser extraída sobre um determinado dado, a falta desta informação é indicada como não informada (NI).

PA1. Quais modelos/ferramentas de avaliação da originalidade do design de UI de apps Android existem?

Foram encontradas somente 11 abordagens diferentes que contêm alguma forma de avaliação da originalidade do design de interfaces de apps Android, conforme apresentado na Tabela 8.

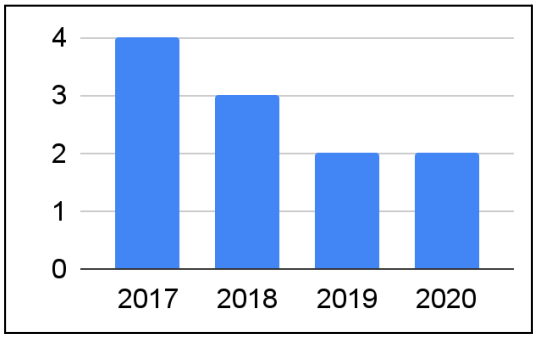
Tabela 8 – Dados extraídos para responder à pergunta de análise 1

Citação	**Referência bibliográfica**
(BEHRANG et al., 2018)	Behrang, F.; Reiss, S.P.; Orso, A. GUIfetch. **Proc. of the 5th Int. Conference on Mobile Software Engineering and Systems**, ACM, EUA, 2018.
(CHEN et al., 2020)	Chen, J.; Chen, C.; Xing, Z.; Xia, X.; Zhu, L.; Grundy, J.; Wang, J. Wireframe-based UI Design Search through Image Autoencoder. **ACM Transactions on Software Engineering and Methodology**, 29(3), 2020.
(CHEN et al., 2019)	Chen, X.; Zou, Q.; Fan, B.; Zheng, Z.; Luo, X. Recommending software features for mobile applications based on user interface comparison. **Requirements Engineering**, 24(4), 2018.
(DEKA et al., 2017)	Deka, B.; Huang, Z.; Franzen, C.; Hibschman, J.; Afergan, D.; Li, Y.; Nichols, J.; Kumar, R. Rico. **Proc. of the 30st Annual ACM Symposium on User Interface Software and Technology**, ACM, 2017.
(GE, 2019)	Ge, X. Android GUI Search Using Hand-Drawn Sketches. Proc. of the **41st Int. Conference on Software Engineering**, IEEE, 2019.
(HU et al., 2020)	Hu, Y.; Xu, G.; Zhang, B.; Lai, K.; Xu, G.; Zhang, M. Robust App Clone Detection Based on Similarity of UI Structure. **IEEE Access**, 8, 2020.
(MUSTAFARAJ et al., 2017)	Mustafaraj, E.; Turbak, F.; Svanberg, M. Identifying Original Projects in App Inventor. **Proc. of the Florida Artificial Intelligence Research Society Conference**, Marco Island, FL, EUA, 2017.
(TURBAK et al., 2017)	Turbak, F.; Mustafaraj, E.; Svanberg, M.; Dawson, M. Work in Progress: identifying and analyzing original projects in an open-ended blocks programming environment. **Proc. of the 23rd Int. Conference on Distributed Multimedia Systems**, Pittsburgh/EUA, 2017.
(SVANBERG, 2017)	Svanberg, M. Using feature vector representations to identify similar projects in App Inventor. **Proc. of the IEEE Blocks And Beyond Workshop**, IEEE, 2017.
(NGUYEN et al., 2018)	Nguyen, T. T.; Vu, P. M.; Pham, H. V.; Nguyen, T. T. Deep learning UI design patterns of mobile apps. **Proc. of the 40th Int. Conference on Software Engineering**, ACM, 2018.
(XIE et al., 2019)	Xie, Y.; Lin, T.; Xu, H. User Interface Code Retrieval: a novel visual-representation-aware approach. **IEEE Access**, 7, 2019.

Fonte: Elaborado pela autora (2021)

Voltando ao foco de identificar a similaridade de UI design foram encontradas poucas pesquisas. Observa-se que a maioria das pesquisas relacionadas à análise de similaridade de apps é voltada a identificação de clones e plágio. Pelo fato de que foram encontrados artigos somente de 2017 até 2020, observa -se que este foco de pesquisa surgiu somente recentemente, conforme apresentado na Figura 12.

Figura 12 – Distribuição de publicações relevantes pelo ano de publicação



Fonte: Elaborado pela autora (2021)

PA2. Quais as características do modelo de avaliação da originalidade do design de interfaces?

O foco dos critérios de avaliação do design de interface varia entre as abordagens encontradas. Portanto, as abordagens foram classificadas em uma das seguintes categorias.

Estrutura: Avalia a similaridade utilizando como base os *sketches* (esboços) para buscar aplicativos com design de interface semelhante comparando com os esboços.

Componentes/Blocos: Avalia a similaridade utilizando como base os componentes e blocos usados na tela.

Os dados extraídos que caracterizam o modelo de avaliação utilizado nas abordagens são apresentados na Tabela 9.

Tabela 9 – Dados extraídos para características do modelo de avaliação

Citação	**Critérios de originalidade**	**Escala de medição**	**Tipo dos Critérios**	**Dados de Entrada**	**Contexto de aplicação**
(BEHRANG et al., 2018)	Similaridade	Classificação de aplicativos semelhantes com base na pontuação de similaridade	Estrutura	*Sketch*	*Professional UI design*
(CHEN et al., 2020)	Similaridade	Principais UIs mais semelhantes	Estrutura	*Screenshot*	*Professional UI design*
(CHEN et al., 2019)	Similaridade	Classificação de UIs semelhantes	Estrutura	*Sketch*	*Professional UI design*
(DEKA et al., 2017)	Similaridade	Similaridade de layout de UIs	Estrutura	*Screenshot*	*Professional UI design*
(GE, 2019)	Similaridade	Classificação de UIs semelhantes	Estrutura	*Sketch*	*Professional UI design*
(HU et al., 2020)	Similaridade	Classificação de UIs semelhantes.	Estrutura	*Screenshot*	*Professional UI design*
(MUSTAFARAJ et al., 2017) (SVANBERG, 2017) (TURBAK et al., 2017)	Originalidade	Original ou não original	Componentes/blocos	componentes e blocos de programação	K-12 contexto educacional
(NGUYEN et al., 2018)	Similaridade	Classificação identificando os principais semelhantes	Estrutura	Texto descrevendo o tipo de tela desejado	*Professional UI design*
(XIE et al., 2019)	Similaridade	Classificação de UIs semelhantes	Estrutura	*Sketch*	*Professional UI design*

Fonte: Elaborado pela autora (2021)

As abordagens variam entre analisar a originalidade (MUSTAFARAJ et al., 2017; SVANBERG, 2017) e a similaridade (BEHRANG et al., 2018; NGUYEN et al., 2018), considerando o conceito da similaridade como o inverso da originalidade. Abordagens que analisam a originalidade, simplesmente avaliam se o design da UI é original ou não original. Por outro lado, abordagens que analisam a similaridade, utilizam escalas de medição de similaridade referente à semelhança entre design de UIs (CHEN et al., 2019) ou criam *rankings* de apps com base no seu grau de originalidade (GE, 2019; BEHRANG et al. 2018).

Em relação aos critérios utilizados, a maior parte das abordagens (8 de 11) avalia critérios relacionados à estrutura da UI, podendo analisar *screenshots* diretamente (ex.: Chen et al. (2020) ou *sketches* (ex.: Behrang et al. (2018)). Xie et al. (2019), transforma um *sketch* de UI como entrada em uma representação de árvore em que cada nó da árvore representa um *widget* ou *layout* e Ge (2019) em esqueletos da UI (UI *skeleton*). Por outro lado, Deka et al. (2017), Chen et al. (2020) e Hu et al. (2020), analisam posições e tamanhos dos elementos de cada UI a partir de um *screenshot*, diferenciando elementos textuais e não textuais e informações estruturais sobre o *layout*. Além disso, Nguyen et al. (2018) e Behrang et al. (2018), adotam a estratégia de analisar não somente a estrutura da UI a partir de uma tela, mas também outros elementos. Nguyen et al. (2018), por exemplo, analisa elementos avulsos, como uma parte da tela ou um elemento de UI e suas descrições e Behrang et al. (2018) utiliza o *sketch* e um conjunto de palavras-chave. Apenas Mustafaraj et al. (2017), Svanberg (2017) e Turbak et al. (2017) não analisam a estrutura da UI, considerando somente elementos avulsos, como componentes e blocos de programação a partir do código do aplicativo.

No contexto da avaliação da originalidade compara-se um artefato “novo” com um universo de referência, de forma a identificar se e com quantos artefatos desse universo de referência o artefato “novo” é similar. Neste contexto, Mustafaraj et al. (2017), Svanberg (2017) e Turbak et al. (2017), consideram projetos criados por alunos a partir de tutoriais globais ou exemplos específicos de cursos para aprender a desenvolver apps com App Inventor. Já a maioria das abordagens utilizam como universo de referência, apps Android disponibilizados em repositórios públicos, como Google Play Store (NGUYEN et al., 2018; HU et al., 2020; DEKA et al., 2017; GE, 2019), Apple App Store (XIE et al., 2019) e Github (BEHRANG et al., 2018).

Observou-se também que a maioria das abordagens encontradas são direcionadas ao uso profissional de design de UI (DEKA et al., 2017; NGUYEN et al., 2018). Somente as abordagens relatadas por Mustafaraj et al. (2017), Svanberg (2017) e Turbak et al. (2017) são voltadas especialmente ao contexto educacional.

PA3. Quais técnicas de IA são adotadas pelos modelos de avaliação?

Várias abordagens utilizam medidas de similaridade/distância, que permitem medir o quanto dois aplicativos são parecidos, conforme apresentado na Tabela 10.

Tabela 10 – Dados extraídos para responder à pergunta de análise 3

Citação	**Dados**	
	Abordagem	**Técnicas usadas**
(BEHRANG et al., 2018)	Medida de similaridade	Utilizam técnicas de deep learning para buscar em um conjunto de dados as UIs similares e calcular uma pontuação de similaridade entre o sketch e todas as UIs encontradas, apresentando um modelo que foi treinado para automaticamente comparar a pontuação de similaridade entre o sketch e a UI correspondente
(CHEN et al., 2020)	CNN	Treinam um CNN (CNN-based image autoencoder architecture) usando wireframes de design de IU não rotulados.
(CHEN et al., 2019)	GA	Utiliza algoritmos genéticos em conjunto com medidas de similaridade/distância a partir do texto extraído da UI para medir a similaridade dos componentes da UI usando o modelo Latent Dirichlet Allocation (LDA). O algoritmo genético é utilizado para otimizar a eficiência do tempo de determinar as correspondências quase ótimas de componentes da UI. Utilizam técnicas de deep learning para buscar em um conjunto de dados as UIs similares e calcular uma pontuação de similaridade entre o sketch e todas as IUs encontradas, apresentando um modelo que foi treinado para automaticamente comparar a pontuação de similaridade entre o sketch e a UI correspondente
(DEKA et al., 2017)	CNN	*Autoencoder*
(GE, 2019)	CNN	Faz a comparação entre árvores e o esboço dos esqueletos da UI correspondente, utilizando o algoritmo Largest Weighted Common Subtree Embeddings (LWCSE).
(HU et al., 2020)	NI	O processo de detecção pode ser dividido em quatro estágios. Em primeiro lugar, foram modificadas as declarações de atividades em AndroidManifest.xml geradas pela descompilação do Arquivo APK, portanto, cada atividade do aplicativo modificado pode ser iniciada pelo comando “AM”. Posteriormente, todas as atividades foram iniciadas e extraía as informações correspondentes da UI. Então, extrair os recursos de estrutura da IU de três dimensões introduzindo o conceito de "Camada". A semelhança de dois atividades é calculado com base em vários recursos estruturais. Finalmente, a similaridade de dois aplicativos baseados na porção de atividades semelhantes pode ser determinada.
(MUSTAFARAJ et al., 2017) (SVANBERG, 2017) (TURBAK et al., 2017)	Medidas de similaridade/distância	Euclidiana, *Cosine* e *Jaccard* em conjunto com clustering e métodos de agrupamento hierárquico para classificar automaticamente os projetos como não originais ou originais.
(NGUYEN et al., 2018)	RNN	Utilizam redes neurais recorrentes e redes adversárias geradoras para identificar a similaridade de screenshots de apps representados em forma descrição em linguagem natural, descrevendo o design da UI. Para isso, utilizam Deep-Visual, uma Recurrent Neural Network (RNN), que é treinada para aprender os padrões de design de UIs de apps.
(XIE et al., 2019)	CNNs	Convertem a UI em uma representação de árvore rotulada como entrada para treinar o modelo CNN (VGGNet). Utilizam técnicas de deep learning para buscar em um conjunto de dados as UIs similares e calcular uma pontuação de similaridade entre o sketch e todas as UIs encontradas, apresentando um modelo que foi treinado para automaticamente comparar a pontuação de similaridade entre o sketch e a UI correspondente

Fonte: Elaborado pela autora (2021)

Behrang et al. (2018), por exemplo, faz uma busca de similaridade, buscando *screenshots* similares a partir de um *sketch*. Algumas abordagens usam essas medidas de similaridade/distância em conjunto com métodos de *clustering*. Mustafaraj et al. (2017) e Svanberg (2017), extraindo vetores de frequência de comandos do código de apps, utilizam medidas de similaridade/distância, como Euclidiana, *Cosine* e *Jaccard* em conjunto com *clustering* e métodos de agrupamento hierárquico para classificar automaticamente os projetos como não originais ou originais.

Seguindo uma estratégia diferente, algumas abordagens utilizam *deep learning*, com modelos *Convolutional Neural Networks* (CNNs), comumente aplicados para analisar imagens visuais (VALUEVA et al., 2020). Xie et al. (2019), por exemplo, convertem a UI em uma representação de árvore rotulada como entrada para treinar o modelo CNN (VGGNet). Chen et al. (2020) treinam um CNN (CNN-*based image autoencoder architecture*) usando *wireframes* de design de UI não rotulados. Ge (2019) faz a comparação entre árvores e o esboço dos esqueletos da UI correspondente, utilizando o algoritmo *Largest Weighted Common Subtree Embeddings* (LWCSE).

Nguyen et al. (2018) utilizam redes neurais recorrentes e redes adversárias geradoras para identificar a similaridade de *screenshots* de apps representados em forma descrição em linguagem natural, descrevendo o design da UI. Para isso, utilizam a *Deep-Visual*, uma *Recurrent Neural Network* (RNN), que é treinada para aprender os padrões de design de UIs de apps.

Somente Chen et al (2019) utiliza algoritmos genéticos em conjunto com medidas de similaridade/distância a partir do texto extraído da UI para medir a similaridade dos componentes da IU usando o modelo *Latent Dirichlet Allocation* (LDA). O algoritmo genético é utilizado para otimizar a eficiência do tempo de determinar as correspondências quase ótimas de componentes da UI.

Várias abordagens utilizam técnicas de *deep learning* para buscar em um conjunto de dados as UIs similares e calcular uma pontuação de similaridade entre o *sketch* e todas as UIs encontradas, apresentando um modelo que foi treinado para automaticamente comparar a pontuação de similaridade entre o *sketch* e a UI correspondente (BEHRANG et al., 2018; XIE et al., 2019; CHEN et al., 2019).

PA4. Como foi feita a preparação do conjunto de dados e a rotulação?

Analisando os artigos relevantes, foi possível perceber a ausência de um conjunto de dados conhecido, com muitos dos artigos criando seu próprio conjunto de dados ou utilizando repositórios públicos, conforme apresentado na Tabela 11.

Tabela 11 – Dados extraídos para responder à pergunta de análise 4

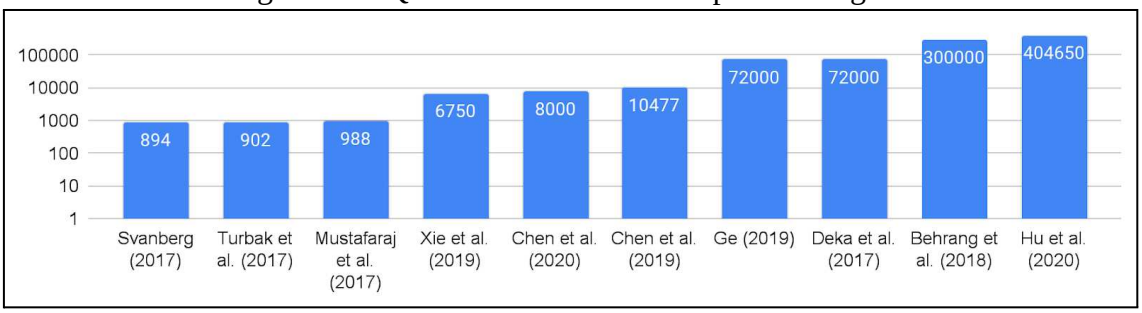
Citação	**Dados**				
	Conjunto de dados	**Rotulação**	**Pré-processamento**	**Formato**	**Referência**
(BEHRANG et al., 2018)	Repositórios públicos (Github)	NI	Parte de um sketch feito à mão.	Conjunto de palavras-chave e *sketch*	www.github.com e https://bitbucket.or g/
(CHEN et al., 2020)	NI	Não utiliza rotulação	Utilizam técnicas de deep learning para criar wireframe e encode em forma de vetor de característica (autoencoder)	Imagem em RGB 180x228	https://github.com/chenjshnn/WAE
(CHEN et al., 2019)	Repositórios públicos	NI	Utiliza a similaridade de texto na interface para medir a similaridade dos componentes da IU. Analisando a similaridade de texto, é utilizado o modelo latent dirichlet allocation (LDA) para modelar um documento e misturar as palavras, após misturar as palavras é utilizado duas ferramentas para calcular a similaridade de texto.	NI	NI
(DEKA et al., 2017)	Rico	NI	NI	Matriz 2D	http://interactionmining.org/rico
(GE, 2019)	Rico	NI	Pré-processamento e na sectio B *Deep-learning* baseado GUI *Skeleton Generation* sobre o pré-processamento transformando o *sketch* em um GUI *skeleton*.	*Sketch*	NI
(HU et al., 2020)	Conjunto de dados próprio	Elementos anotados manualmente	Foi utilizado *intents* explícitos, inserindo atributos para todas as atividades declaradas no Arquivo AndroidManifest.xml. Foi utilizado Apktool para descompacte e recompacte aplicativos, depois de obter o arquivo AndroidManifest.xml, foi inserido a tag "intent-filter" para cada nó de atividade, acompanhada por subtags "categoria" e "ação" para o "filtro de intenção" marcação. Reembalar e assinar os arquivos modificados no "novo" APK.	NI	NI
(MUSTAFARAJ et al., 2017) (SVANBERG, 2017) (TURBAK et al., 2017)	Conjunto de dados próprio	Elementos anotados manualmente	Utilizam componentes e blocos de programação. Extraem vetores de frequência de uso de comandos/blocos a partir do código de apps criados com App Inventor e rotulam manualmente os apps em original ou não original.	App Inventor code .aia	http:appinventor.mi t.edu
(NGUYEN et al., 2018)	Repositórios públicos	rotulado manualmente	Utilizam uma tela, uma parte da tela ou um elemento de UI em conjunto com suas descrições	*Screenshots*	NI
(XIE et al., 2019)	Conjunto de dados públicos	Comparando Árvores rotuladas	Parte de um sketch feito à mão e utilizam técnicas de deep learning para buscar em um conjunto de dados as UIs similares. A partir disso, é calculada uma pontuação de similaridade entre o sketch e todas as UIs encontradas. Converte o esboço em uma árvore de representação visual para então treinar o modelo CNN com um grande número de esboços rotulados.	*Sketch*	NI

Fonte: Elaborado pela autora (2021)

Várias abordagens utilizam como tipos de entradas *sketches* e/ou *screenshots*. Behrang et al. (2018) e Xie et al. (2019), por exemplo, partem de um *sketch* feito à mão. Algumas abordagens utilizam uma tela, uma parte da tela ou um elemento de UI em conjunto com suas descrições (NGUYEN et al., 2018). Outras abordagens usam componentes e blocos de programação, como as abordagens de Mustafaraj et al. (2017), Turbak et al. (2017) e Svanberg (2017). Somente uma das abordagens utiliza a similaridade de texto na interface para medir a similaridade dos componentes da IU (CHEN et al., 2019).

A maioria das abordagens realiza algum tipo de pré-processamento dos dados de entrada. Algumas abordagens utilizam técnicas de *deep learning* para criar *wireframe* e *encode* em forma de vetor de característica (*autoencoder*) (CHEN et al., 2020). Analisando a similaridade de texto, é utilizado o modelo *latent dirichlet allocation* (LDA) para modelar um documento e misturar as palavras, após misturar as palavras é utilizado duas ferramentas para calcular a similaridade de texto (CHEN et al., 2019). Por outro lado (MUSTAFARAJ et al., 2017; TURBAK et al., 2017; SVANBERG, 2017) extraem vetores de frequência de uso de comandos/blocos a partir do código de apps criados com App Inventor e rotulam manualmente os apps em original ou não original. Com relação aos conjuntos de dados utilizados para o treinamento de modelos de ML, observa-se que, em geral, a quantidade de instâncias varia entre menos de 1000 apps a quantidades que chegam próximo a meio milhão de apps, conforme apresentado na Figura 13.

Figura 13 – Quantidade de instâncias por abordagens



Fonte: Elaborado pela autora (2021)

PA5. Como é medido e qual é o desempenho das abordagens?

A maioria dos trabalhos apresenta avaliações do desempenho das abordagens. Porém, observa-se que são utilizados diferentes tipos de medidas, dependendo do tipo de avaliação adotado, para apresentar o desempenho das abordagens, conforme disponibilizado na Tabela 12.

Tabela 12 – Dados extraídos para responder à pergunta de análise 5

Referência	**Medida/método de análise**	**Resultado**
BEHRANG et al. (2018)	Tau de Kendall	TAU = Valores variando de 0.6 a 1 (p variando de 0.015 a 0.142)
	Rho de Spearman	RHO = Valores variando de 0.7 a 1 (p variando de 0.037 a 0.188)
CHEN et al. (2020)	Precision@k (Pre@k) (k=1)	Precision@1= 70% (component scaling ratio 20%) Precision@1= 84.6% (component-removal ratio 20%) Precision@1= 70.6% (fully connected neural network baseline ratio 20%) Precision@1= 47.6% (fully connected neural network baseline ratio 30%) Precision@1= 0.5 (scaling-10% and removal-20%)
	Mean Reciprocal Rank (MRR)	MRR = 0.73 (component scaling ratio 20%) MRR = 0.88 (component-removal ratio 20%) MRR = 0.77 (fully connected neural network baseline ratio 20%) MRR = 0.57 (fully connected neural network baseline ratio 30%) MRR = 0.62 (scaling-10% and removal-20%)
	Relevância usando Precision@k (Pre@k) (k=1, 5 e 10) e *Mean Reciprocal Rank* (MRR)	Relevância precision@1 = 0.44 Relevância precision@5 = 0.40 Relevância precision@10 = 0.38 MRR = 0.59
CHEN et al. (2019)	*Average Precision* (AP) para 30 apps	AP = 0.42
	Mean Average Precision (MAP) para 30 apps	MAP (30) = 0.418
DEKA et al. (2017)	Não relatado	Não relatado
GE (2019)	Comparação de um escore de similaridade com *ranking*	A abordagem encontra as 6 GUIs mais similares a uma GUI de consulta
HU et al. (2020)	Eficácia (razão falso-positivos (FPR) e razão falso-negativos (FNR))	FPR = 0.02% FNR = 0.42%
	Eficiência (Tempo em segundos (T))	T = 0.053s para comparar um par de apps
MUSTAFARAJ et al. (2017)	Acurácia	Acurácia = 0.89 (considerando uma distância de Jaccard a um threshold de 0.4)
TURBAK et al. (2017)	Não relatado	Não relatado
SVANBERG (2017)	Comparação de medidas por meio de revocação (*recall*)	Revocação > 0.8 (*Cosine*, *Jaccard*, Euclidiana e *Cityblock* sem grandes diferenças)
	Comparação de seleção de *features* por meio de revocação (*recall*)	Revocação > 0.9 (apenas blocos ou blocos com componentes) Revocação > 0.5 (apenas componentes)
	Comparação de normalização de *features* por meio de revocação (*recall*)	Revocação > 0.7 (TF-IDF e Count) Revocação > 0.5 (Binário)
NGUYEN et al. (2018)	Não relatado	Não relatado
XIE et al. (2019)	*Intraclass Correlation Coefficient* (ICC)	ICC = 0.725 a 0.889, média de 0.822 ($p < 0.0001$)
	Coeficiente de Correlação de Pearson (PPC)	PPC = 0.883
	Mean Squared Error (MSE)	MSE = 0.027

Fonte: Elaborado pela autora (2021)

Abordagens que propõem um *ranking* automatizado de originalidade de vários apps utilizam medidas de estatística para avaliar o desempenho da correlação entre *rankings*, como Rho de Spearman, Tau de Kendall, Coeficiente de Correlação de Pearson, etc. (GE, 2019; XIE et al., 2019; BEHRANG et al., 2018). Por outro lado, abordagens que propõem a avaliação do grau de originalidade de um app, utilizam medidas como acurácia (MUSTAFARAJ et al.,

2017), revocação (SVANBERG, 2017), eficácia (HU et al., 2020), etc. Cabe ressaltar que apenas uma abordagem também realiza uma avaliação da eficiência, buscando apresentar como o uso de um algoritmo genético auxilia na rapidez do processamento da análise de originalidade (HU et al., 2020).

De forma geral os resultados apresentam bom desempenho, tanto em medidas de correlação, como de acurácia, revocação, etc. No entanto, algumas abordagens apresentam uma comparação cujos resultados denotam que algumas configurações podem ser melhores/piores, como os resultados de comparação apresentados por Svanberg (2017) em que apenas blocos ou blocos com componentes apresentam resultados superiores a utilizar somente componentes do App Inventor.

4 DESENVOLVIMENTO DA SOLUÇÃO

O presente trabalho visa propor uma abordagem para automaticamente analisar o grau de originalidade referente aos componentes de design de interface de usuário de um app desenvolvido com App Inventor, com desempenho adequado, que pode ser utilizado para contribuir na avaliação da criatividade no ensino de computação na Educação Básica.

Com o objetivo de desenvolver uma avaliação, foi adotado a metodologia de *Evidence Centered Design* (ECD) (MISLEVY et al., 2003) para analisar o contexto e os modelos conceituais de projeto. O ECD organiza o trabalho de projeto e implementação em termos de camadas, vê a avaliação como um processo de coleta de evidências para apoiar um argumento sobre o que um aluno sabe e pode fazer, melhorar a coerência das avaliações ao relacionar explicitamente com as características de tarefas, evidência o desempenho dos alunos nessas tarefas e o conhecimento e as habilidades implicadas por essa evidência.

De acordo com Mislevy e Haertel (2006), o ECD é dividido em 5 camadas:

- Análise de Domínio: identificar e analisar informações sobre o domínio de interesse que tem implicações diretas para a avaliação, por exemplo: como o conhecimento é construído adquirido, usado e comunicado.
- Modelagem de Domínio: descrever o argumento de avaliação em forma narrativa com base em informação de análise de domínio.
- Estrutura de avaliação conceitual: descrever argumentos de avaliação em estruturas e especificações para tarefas e testes, procedimentos de avaliação, modelos de medição.
- Implementação de avaliação: implementar avaliação, incluindo tarefas prontas para apresentação e calibradas modelos de medição.
- Entrega de avaliação: coordenar interações alunos e tarefas, aplicando a avaliação na prática, coletando os dados e relatando os resultados.

As mesmas serão apresentadas nas próximas seções.

4.1 ANÁLISE DE DOMÍNIO

Nesta seção, são apresentados as definições e detalhes sobre como avaliar a originalidade do produto considerando artefatos computacionais no contexto do ensino de computação.

4.1.1 Originalidade em computação

A computação é uma disciplina fundamentalmente criativa (FINCHER; ROBINS 2019) quando estudantes deixam de ser apenas consumidores de tecnologia e passam a construir artefatos computacionais criativos que podem ter um impacto significativo na sociedade (YADAV; COOPER, 2017). Nesse contexto, a criatividade é uma das chaves para responder aos desafios do desenvolvimento de artefatos computacionais hoje, bem como preparar os alunos para aprender a criar, testar e refinar artefatos computacionais (SHUTE et al., 2017).

Várias diretrizes curriculares para o ensino de computação incluem alguns aspectos da criatividade na criação de artefatos computacionais, juntamente com as práticas computacionais. O CSTA (2016) *framework* inclui explicitamente a criatividade como parte do processo de criação de um artefato, considerando também, que o professor pode apresentar um problema explícito e solicitar que o aluno crie soluções criativas. O processo de desenvolvimento de artefatos computacionais abrange tanto a expressão criativa quanto a exploração de ideias para criar protótipos e resolver problemas computacionais. Os alunos criam artefatos que são pessoalmente relevantes ou benéficos para sua comunidade e além. Os artefatos computacionais podem ser criados combinando e modificando artefatos existentes ou desenvolvendo novos artefatos, como p.ex. e aplicativos móveis (CSTA, 2016, p. 80).

A Base Nacional Comum Curricular (MEC, 2022) do ensino fundamental, está organizada em cinco áreas do conhecimento, são elas linguagens, matemática, ciências da natureza, ciências humanas e ensino religioso. Já a BNCC do Ensino Médio está organizada por quatro áreas do conhecimento que são linguagens e suas tecnologias, matemática e suas tecnologias, ciências da natureza e suas tecnologias, ciências humanas e sociais aplicadas. Ambas as fases, incluem a criatividade como parte das competências gerais da educação básica:

- Exercitar a curiosidade intelectual e recorrer à abordagem própria das ciências, incluindo a investigação, a reflexão, a análise crítica, a imaginação e a criatividade, para investigar causas, elaborar e testar hipóteses, formular e resolver problemas e criar soluções (inclusive tecnológicas) com base nos conhecimentos das diferentes áreas.
- Compreender, utilizar e criar tecnologias digitais de informação e comunicação de forma crítica, significativa, reflexiva e ética nas diversas práticas sociais (incluindo as escolares) para se comunicar, acessar e disseminar informações, produzir conhecimentos, resolver problemas e exercer protagonismo e autoria na vida pessoal e coletiva.

Segundo a Sociedade Brasileira de Computação (SBC) (2022), os conhecimentos da área de computação podem ser divididos em três eixos:

- Pensamento computacional: refere-se à capacidade de compreender e solucionar problemas através da construção de algoritmos.
- Mundo Digital: refere-se aos veículos de comunicação e interação baseados em tecnologia digital, com o ensino a distância.
- Cultura Digital: refere-se a padrões de comportamento e novos questionamentos morais e éticos na sociedade que surgiram em decorrência do Mundo Digital.

Na educação básica, os alunos devem ser introduzidos a conceitos relacionados às estruturas abstratas necessárias à resolução de problemas para que o aluno tome consciência do processo de resolução de problemas e compreenda a importância de ser capaz de descrever a solução em forma de algoritmo (SBC, 2022). Após dominar esses conceitos, espera-se que os estudantes sejam capazes de selecionar e utilizar modelos e representações adequados para descrever informações e processos, bem como dominem as principais técnicas para construir soluções algorítmicas. Uma forma de aprender esses conceitos é por meio da criação de aplicativos (ARAÚJO et al., 2020). Na educação básica, ambientes de programação baseados em blocos como o App Inventor dão a oportunidade para os alunos exercitarem e aprenderem conceitos de computação.

Considerando que a criatividade pode ser definida por meio do artefato criado, conforme definido na seção 2, a originalidade é uma das subdimensões da criatividade do produto

(JACKSON; MESSICK, 1964). Com relação à subdimensão originalidade, uma ideia ou produto criativo é considerado original se representa algo novo ou surpreendente que não existia antes (RUNCO; JAEGER, 2012). A subdimensão originalidade tem como características um produto raro, incomum, infrequente, engenhoso, surpreendente e imaginativo (ALVES et al., 2020a).

Contudo, o desenvolvimento de aplicativos envolve não somente competências de programação, mas também de design de interface de usuário (UI), um conceito transversal da computação (CSTA, 2017). O design de interface visa maximizar a usabilidade e a experiência do usuário, tornando a interação do usuário com o aplicativo eficaz, eficiente e satisfatória (WASSERMAN, 2010).

O App Inventor, conforme apresentado na seção 2, é um ambiente de programação visual com uma comunidade online, que permite a qualquer pessoa (inclusive crianças) facilmente criar e compartilhar aplicativos móveis. O ensino de computação é tipicamente feito adotando metodologias ativas e tarefas abertas que inspiram os alunos a criarem seus resultados engajando-os na construção de artefatos digitais e tangíveis por meio do uso de tecnologias (RODE et al., 2015; BRENNAN et al., 2019). Uma das estratégias para o ensino da computação por meio do desenvolvimento de apps é a abordagem “Use-Modifique-Crie” (UMC) (LYTLE et al., 2019; LEE et al., 2011). Na UMC os alunos aprendem primeiro “usando” e analisando um determinado artefato computacional, “remixando” e modificando um artefato já existente e, eventualmente, “criando” um novo (LEE et al., 2011). Assim, os alunos desenvolvem a capacidade de gerar ideias e soluções originais e úteis (WALIA, 2019), especialmente durante a fase de criação, na qual, para estimular a criatividade, as atividades de aprendizagem são muitas vezes colocadas como problemas abertos e mal definidos em um contexto adotando uma abordagem baseada em problemas.

4.1.2 Aspectos relevantes do aluno

A avaliação da originalidade de aplicativos móveis criados é voltada para alunos do Ensino Fundamental (anos finais - 6º ao 9º ano) e Ensino Médio das escolas brasileiras.

Ao longo do Ensino Fundamental (anos finais), os estudantes se deparam com desafios de maior complexidade, sobretudo devido à necessidade de se apropriarem das diferentes lógicas de organização dos conhecimentos relacionados às áreas (MINISTÉRIO

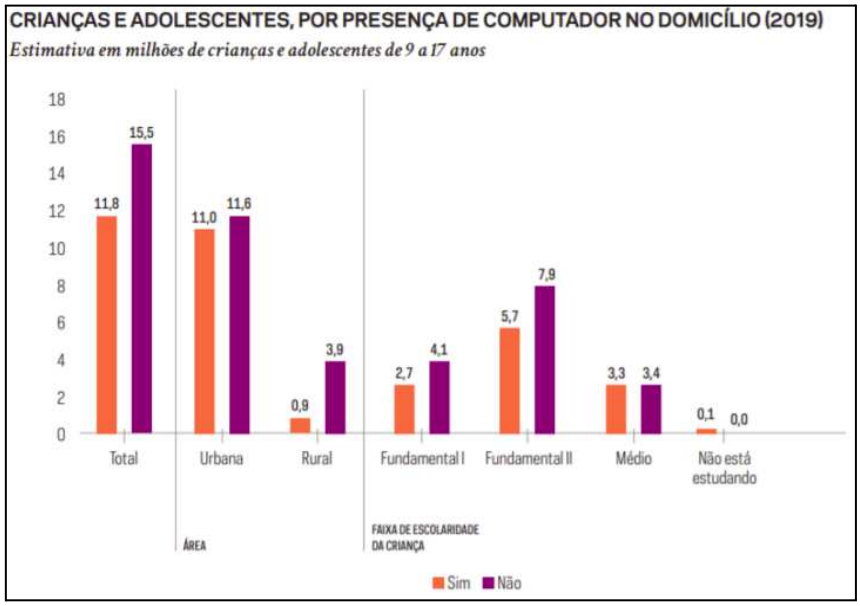
DA EDUCAÇÃO, 2022). Nessa fase, do Ensino Fundamental (anos finais), também é necessário considerar que a cultura digital promove mudanças sociais significativas nas sociedades contemporâneas. Em decorrência do avanço e da multiplicação das tecnologias de informação e comunicação e do crescente acesso a elas pela maior disponibilidade de computadores, telefones celulares, *tablets* e afins, os estudantes estão dinamicamente inseridos nessa cultura, não somente como consumidores (MINISTÉRIO DA EDUCAÇÃO, 2022). Os jovens têm se engajado cada vez mais como protagonistas da cultura digital, envolvendo-se diretamente em novas formas de interação multimidiática e multimodal e de atuação social em rede, que se realizam de modo cada vez mais ágil.

O Ensino Médio corresponde à última fase da Educação Básica, com duração de três anos e alunos na faixa etária regular de 15 a 18 anos. Considera-se que nessa faixa os alunos possuem fluência na língua portuguesa do Brasil e conhecimentos básicos de línguas estrangeiras como o inglês e espanhol (MINISTÉRIO DA EDUCAÇÃO, 2022). É esperado que os estudantes dessa fase consolidem e aprofundem os conhecimentos adquiridos no Ensino Fundamental, obtenham preparo básico para o mercado de trabalho, além de compreender fundamentos científico-tecnológicos dos processos produtivos, relacionando a teoria com a prática no ensino de cada disciplina (MINISTÉRIO DA EDUCAÇÃO, 2022). No Ensino Médio, o MEC (2022), define como foco o reconhecimento das potencialidades das tecnologias digitais para a realização de uma série de atividades relacionadas a todas as áreas do conhecimento, a diversas práticas sociais e ao mundo do trabalho.

4.1.3 Aspectos relevantes das escolas brasileiras

De acordo com o censo escolar de 2021 (INEP, 2021), 95,7% das escolas federais disponibilizam computadores de mesa para alunos do ensino fundamental. Esse número é relativamente menor, se comparado à disponibilidade de computadores de mesa para alunos do ensino médio, que corresponde a 99,5% nas escolas federais, conforme apresentado na Tabela 13. Segundo dados da pesquisa Brasil (2019), no Brasil 11,8 milhões dos alunos possuem computador no domicílio, conforme Figura 14.

Figura 14 – Crianças e adolescentes, por presença de computador no domicílio



Fonte: Brasil (2019)

Já os domicílios que continham tablet, a pesquisa aponta um percentual de apenas 11,3% (EDUCA, 2022). Em 4,7% das residências não havia nenhum tipo de telefone fixo. Por outro lado, a parcela dos domicílios em que havia aparelho celular alcançou 94% (EDUCA, 2022). Da população com 10 anos ou mais de idade, 81% tinham telefone móvel celular para uso pessoal (EDUCA, 2022).

Segundo dados do censo escolar de 2021 (INEP, 2021), 100,0% das escolas municipais do ensino fundamental no Brasil possuem Internet. Esse número é quase igual se comparado ao ensino médio das escolas municipais, onde 99,8% delas têm acesso à internet (INEP, 2021). A disponibilidade (%) de acesso a internet nas escolas do ensino fundamental e do ensino médio, nas escolas brasileiras é apresentada na Tabela 13.

Tabela 13 – Disponibilidade (%) de recursos tecnológicos nas escolas brasileiras por dependência administrativa, segundo recurso - 2021

Recurso	Dependência Administrativa			
	Federal (Pública)	Estadual (Pública)	Municipal (Pública)	Privada
Internet Ensino Fundamental	100,0%	92,0%	69,8%	98,4%
Internet Ensino Médio	99,8%	95,0%	93,6%	99,5%
Internet Banda Larga Ensino Fundamental	100,00%	76,1%	56,4%	89,5%
Internet Banda Larga Ensino Médio	98,3%	81,0%	77,7%	93,4%
Internet para Alunos Ensino Fundamental	89,4%	65,2%	27,8%	53,1%
Internet para Alunos Ensino Médio	98,8%	68,2%	54,3%	73,1%
Internet para Uso Administrativo Ensino Fundamental	100,00%	90,8%	66,3%	94,9%

Recurso	Dependência Administrativa			
	Federal (Pública)	Estadual (Pública)	Municipal (Pública)	Privada
Internet Ensino Fundamental	100,0%	92,0%	69,8%	98,4%
Internet Ensino Médio	99,8%	95,0%	93,6%	99,5%
Internet Banda Larga Ensino Fundamental	100,00%	76,1%	56,4%	89,5%
Internet para Uso Administrativo Ensino Médio	99,2%	93,5%	90,4%	96,8%
Internet para Ensino e Aprendizagem Ensino Fundamental	89,4%	74,1%	39,8%	70,6%
Internet para Ensino e Aprendizagem Ensino Médio	91,8%	74,6%	66,0%	84,0%
Lousa Digital Ensino Fundamental	55,3%	29,8%	10,8%	15,5%
Lousa Digital Ensino Médio	55,6%	31,3%	24,5%	29,4%
Projetor Multimídia Ensino Fundamental	95,7%	79,1%	55,4%	73,3%
Projetor Multimídia Ensino Médio	98,8%	81,5%	82,4%	86,9%
Computador de Mesa para Alunos Ensino Fundamental	95,7%	76,9%	39,2%	66,9%
Computador de Mesa para Alunos Ensino Médio	99,5%	78,8%	76,1%	79,6%
Computador Portátil para Alunos Ensino Fundamental	63,8%	37,7%	25,8%	50,4%
Computador Portátil para Alunos Ensino Médio	50,8%	40,7%	42,6%	57,1%
Tablet para Alunos Ensino Fundamental	34,0%	12,6%	6,6%	26,7%
Tablet para Alunos Ensino Médio	34,8%	13,2%	8,0%	32,9%

Fonte: Elaborado pela autora com base no (INEP, 2021).

No entanto, com relação ao acesso à internet para o ensino e aprendizagem, essa porcentagem diminuiu para 89,4% no ensino fundamental e 91,8% no ensino médico das escolas federais. Em geral, a disponibilidade de recursos tecnológicos no ensino médio é maior do que nas escolas do ensino fundamental.

O número de escolas públicas federais no Brasil é muito pequeno em comparação ao número de escolas públicas estaduais e municipais, a maioria das escolas possui contexto com menos recursos e maiores dificuldades na realização de atividades que utilizam recursos digitais, conforme é possível visualizar na Tabela 14.

Tabela 14 – Número de escolas por dependência administrativa, segundo a localização - 2020

Localização da Escola	Dependência Administrativa					
	Total	Pública	Federal	Estadual	Municipal	Privada
Total	179.533	138.487	700	29.888	107.899	41.046

Fonte: INEP (2020)

Essas dificuldades também são relatadas pelos professores ao tentar usar as tecnologias nas atividades pedagógicas. Segundo TIC Educação (2019), a maioria dos professores relata que o número de computadores por aluno, bem como computadores

conectados à internet são insuficientes e dificultam muito o uso de algumas tecnologias no contexto educacional.

Com relação a duração das aulas no Brasil, as aulas no Ensino Fundamental e Médio têm duração geralmente de 45 minutos e as turmas possuem em média 30 alunos INEP (2020).

No Brasil, não há uma padronização para a atribuição de notas. Cada Conselho Municipal de Educação possui autonomia para definir os instrumentos avaliativos, embora usualmente se dê pela alocação de notas de forma quantitativa por meio de números inteiros variáveis de 0 (desempenho totalmente insatisfatório) a 10 (desempenho totalmente satisfatório) (CME, 2011).

4.1.4 Aspectos dos professores brasileiros

Para ensinar computação os professores precisam ter conhecimento de conteúdo em computação (especialmente algoritmos e programação), e também conhecimento tecnológico (para instalar e manter a infraestrutura de TI necessária) (ALVES et al., 2020c). No entanto, visando a introdução ampla do ensino de computação em escolas brasileiras, atualmente é caracterizada pela falta de professores formados na área de computação e sem tempo livre no contexto de turmas grandes.

A maioria das escolas não têm professores com formação em informática (ALVES et al., 2016). Uma das razões é que o número de graduados com licenciaturas com foco em computação é muito inferior, por exemplo, ao número de licenciados com licenciatura em disciplinas específicas, como matemática e língua nativa – português (INEP, 2020).

Isso pode tornar mais difícil, por exemplo, avaliar a aprendizagem do aluno como professores de computação formalmente treinados na Educação Básica são escassos. Muitas iniciativas visam formação de professores em serviço de outras áreas, abrangendo também competências básicas de informática como conhecimento pedagógico e tecnológico para possibilitar o ensino de computação em um multidisciplinar em suas respectivas disciplinas (ALVES et al., 2020c).

A maioria das escolas não têm professores com formação em informática. Um dos motivos é que o número de graduados com licenciaturas com foco em computação é muito menor do que, por exemplo, o número de graduados com licenciatura para disciplinas tradicionais, como matemática e língua nativa – português INEP (2020). Consequentemente, a introdução da computação no ensino fundamental e médio no Brasil depende de professores formados em outras áreas. Muitas iniciativas visam à formação de professores em serviço de outras áreas, abrangendo competências básicas de informática, bem como conhecimentos pedagógicos e tecnológicos para viabilizar o ensino de computação de forma multidisciplinar em suas respectivas disciplinas (ALVES et al., 2020c). Isso pode tornar mais difícil, por exemplo, avaliar o aprendizado dos alunos, pois os professores de computação formalmente treinados no K-12 são escassos (ALVES et al., 2020c).

4.2 MODELAGEM DE DOMÍNIO

Nesta seção, é indicado os elementos que são necessários na avaliação de criatividade de artefatos computacionais no contexto do ensino de computação usando os padrões de design do *Principled Assessment Designs for Inquiry* (PADI) (SEERATAN; MISLEVY, 2008). Com base na análise do domínio foi definido um padrão de projeto, especificando parte do domínio do construto em termos das competências a serem medidas e os dados que podem ser usados como evidência dessas competências, bem como as atividades educacionais nas quais esses dados podem ser usados.

O padrão de projeto é criado considerando o conhecimento focal e habilidades (MISLEVY; HAERTEL, 2006), bem como outros conhecimentos/habilidades que podem ser necessários no contexto da originalidade do artefato computacional. Foram definidas observações potenciais, ou seja, algumas coisas possíveis que alguém poderia ver os alunos fazendo que dariam evidências sobre os conhecimentos/habilidades (MISLEVY; HAERTEL, 2006). Para a tarefa modelo foi descrito os potenciais produtos de trabalhos, características que evidenciam aspectos de situações de avaliação que provavelmente evocam a evidência desejada e características variáveis para descrever aspectos de situações de avaliação que podem ser variados para mudar a dificuldade ou foco (MISLEVY; HAERTEL, 2006), conforme apresentado na Tabela 15.

Tabela 15 – Padrão de Design

Elemento	Descrição

Conhecimento Focal, Habilidades e Outros Atributos	- Criar aplicativos móveis originais em termos de esqueleto do design de interface.
Conhecimentos, Habilidades e Atributos Adicionais	– Criar protótipos de sistemas de software interativos em diferentes níveis (esboços, baixa fidelidade, alta fidelidade, funcional) **incluindo o design de interface (esqueleto)** (CSTA, 2016: 2-AP-19, 1B-AP-13, 2- AP-13), (ACM/IEEE, 2013). – Criar soluções tecnológicas usando a abordagem adequada à ciência, incluindo pesquisa, reflexão, análise crítica, imaginação e **criatividade (**MEC, 2018).
Observações potenciais	Design de interface (esqueleto) de aplicativos móveis criado por aluno.
Potenciais Produtos de Trabalhos	Aplicativo móvel (arquivo .aia), screenshot.
Potencial Rubrica(s)	–
Recursos característicos	Contexto de aprendizagem ativa com foco principalmente na etapa de criar do ciclo Use-Modifique-Crie: - criar: criar um *layout* de UI de aplicativo móvel para resolver um problema identificado pelo próprio aluno.
Recursos Variáveis	- Criar: o aluno desenvolve UI a partir do zero.

Fonte: Elaborado pela autora (2022)

Com relação aos conhecimentos, habilidades e atributos adicionais foram apresentados os citados na tabela acima considerando o design de interface no nível de *layout* focando na criatividade no contexto da originalidade do artefato computacional.

4.2.1 Modelo do aluno

Considerando uma estratégia de UMC, no nível de criação, o objetivo geral de aprendizagem é que o aluno deve ser capaz de criar um aplicativo móvel que seja original em termos de esqueleto com foco no design de interface. Os objetivos de aprendizagem são detalhados na Tabela 16.

Tabela 16 – Objetivos de aprendizagem

ID	Objetivo de aprendizagem	Fonte
OA1 - Criar	- Criar protótipos de sistemas de software interativos em diferentes níveis (esboços, baixa fidelidade, alta fidelidade, funcional) **incluindo o design de interface (esqueleto)**.	(CSTA, 2016: 2-AP-19, 1B-AP-13, 2- AP-13), (ACM/IEEE, 2013)
OA2-Criar	- Criar soluções tecnológicas usando a abordagem adequada à ciência, incluindo pesquisa, reflexão, análise crítica, imaginação e **criatividade**.	(MEC, 2018)

Fonte: Elaborado pela autora (2022)

4.2.2 Modelo de tarefa

Para o modelo de tarefa, o código do potencial artefato de trabalho é o aplicativo móvel (arquivo .aia) ou *screenshot* criado pelo aluno usando o App Inventor . Com base neste artefato, a evidência é extraída para obter observações potenciais definidas no Modelo de evidência.

Considerando os aspectos do modelo de tarefa, os traços característicos compreendem o desenvolvimento de uma aplicação móvel dentro de um contexto de aprendizagem ativa durante a fase de Criar do ciclo UMC em uma estratégia baseada em problemas. Considerando que criação de um aplicativo móvel, não é apenas programar, mas também acompanhar todo o processo de design de UI, conforme apresentado na Tabela 17.

Tabela 17 – Tarefa/Objetivos de aprendizagem

Tarefa	Objetivos de aprendizagem
A partir de um *Wireframe* disponibilizado sem design de interface de UI pronto, criar o design de interface de UI, p. ex. para uma turma do Ensino Fundamental	- Criar protótipos de sistemas de software interativos em diferentes níveis (esboços, baixa fidelidade, alta fidelidade, funcional) **incluindo o design de interface (esqueleto)**.
Criar *wireframe* com design de interface de UI, p. ex. para uma turma do Ensino Médio	- Criar soluções tecnológicas usando a abordagem adequada à ciência, incluindo pesquisa, reflexão, análise crítica, imaginação e **criatividade**.

Fonte: Elaborado pela autora (2022)

Algumas características do modelo de tarefa podem variar para alterar a dificuldade ou foco da tarefa, por exemplo, o professor pode querer uma avaliação mais abrangente, comparando todos os aplicativos móveis criados por alunos em uma aula ou até mesmo de uma galeria pública, como a App Inventor Gallery.

4.2.3 Modelo de evidência

Considerando o foco na originalidade de aplicativos móveis, potenciais observações contam com a análise dos aplicativos móveis dos alunos. O objetivo da análise é determinar o grau da dimensão de originalidade do artefato com base em evidências de que o aluno adquiriu o conhecimento, habilidades e habilidades focais e adicionais necessários (Tabela 15).

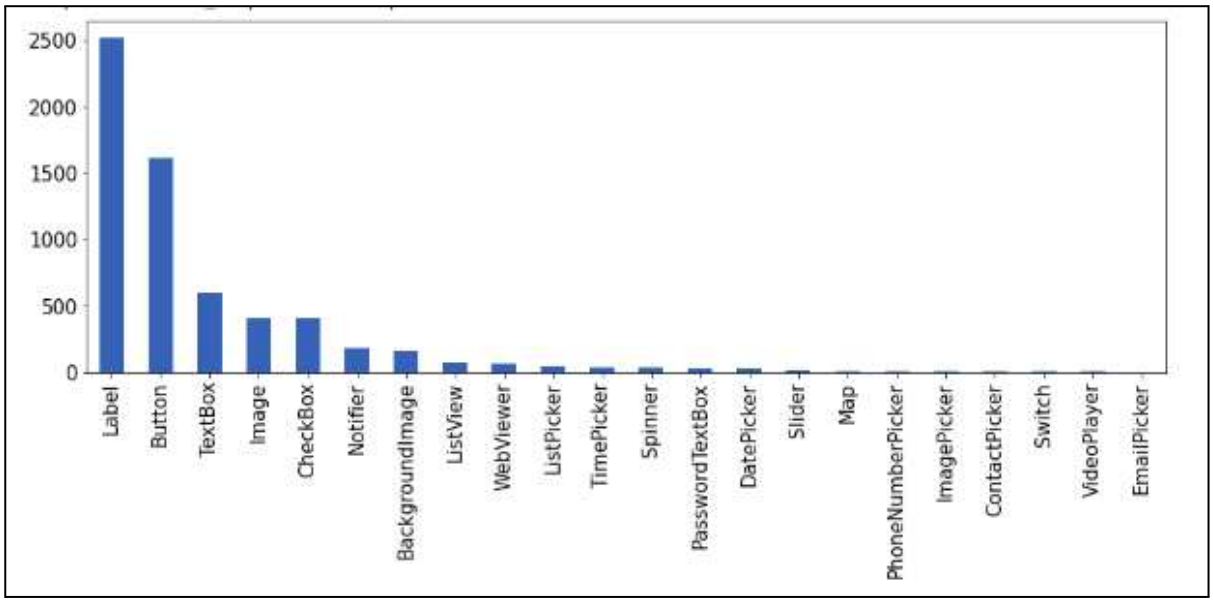
A originalidade do artefato é analisada considerando sua natureza de acordo com as camadas (planos) do *framework* de Garrett (2010). A dimensão originalidade, considera

apenas a camada esqueleto com foco no design de interface, pois estamos trabalhando em uma interface estática abordagem e design de informação e análise de design de navegação dependem da dinâmica execução do aplicativo móvel.

4.2.3.1 Modelo de evidência de originalidade utilizando medidas de similaridade

A maioria dos aplicativos do App Inventor inclui alguns componentes de interface, com total de 93% de 88.606 projetos (Alves et al., 2019). Geralmente, os componentes mais comuns da interface do usuário usados em aplicativos do App Inventor são *Label*, *TextBox*, *CheckBox*, *Image* e *Button* com base em uma análise de larga escala de 88 mil aplicativos. Para a presente pesquisa, foram considerados apenas componentes de UIs visíveis ao usuário, conforme apresentado na Figura 15.

Figura 15 – Componentes de UI utilizados por aplicativos



Fonte: Elaborado pela autora (2022)

Visando a identificação da originalidade de esqueleto do design de interface utilizando o código de apps como entrada, normalmente, vetores de frequência de recursos extraídos de arquivos de aplicativos móveis são usados (MUSTAFARAJ et al., 2017). Os vetores de frequência de recursos podem ser representados de diferentes maneiras:

- Frequência: frequência de cada token dentro de cada documento.
- Binário: indicação se um token está ou não no documento.
- TF-IDF: *Term Frequency–Inverse Document Frequency* é um método de ponderação que leva em conta todos os documentos do corpus de textos dando

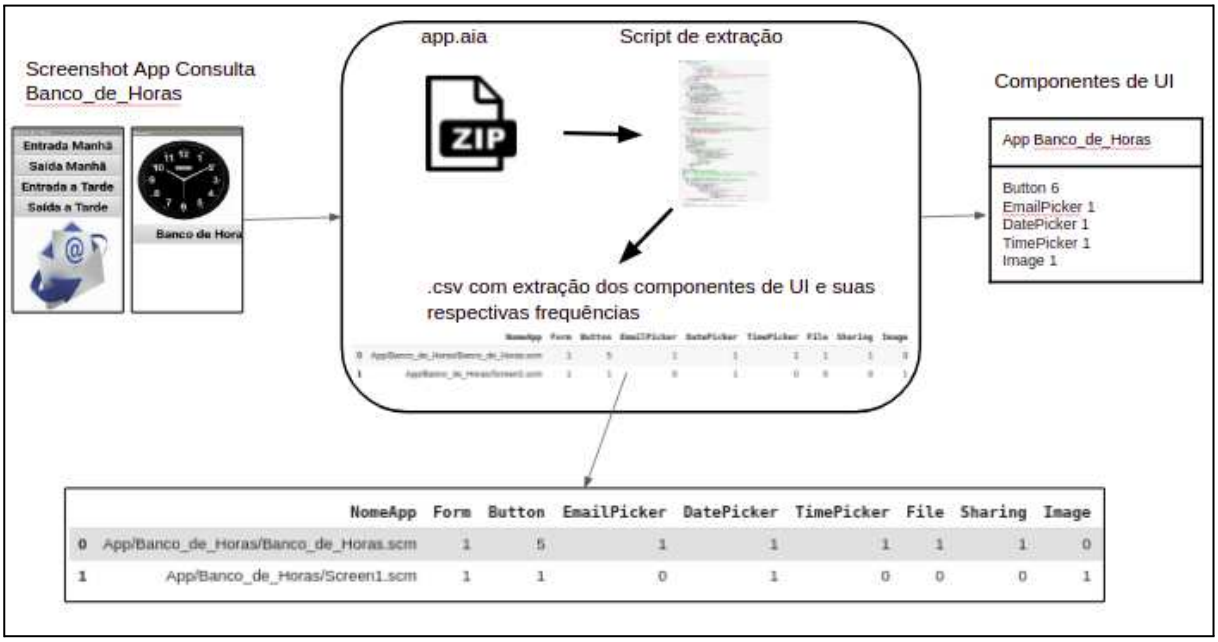
> pesos maiores para termos raros com alta capacidade de diferenciação de outros textos e menos peso para termos mais comuns encontrados em todo o corpus (MANNING et al., 2008).

Para a extração automática é realizado um *parse* de palavras do código de projeto do App Inventor (.aia) verificando a presença dos respectivos componentes. Para representar a presença de componentes de UI em um app, adotou-se um modelo de *Code BOW*, também chamado de modelo espaço vetorial, que é uma representação simples usada em processamento de linguagem natural e recuperação de informação (SALTON et al., 1975).

O vetor de features indicando a presença de componentes de UI design em um app é gerado automaticamente a partir do código do app (.aia) identificando a frequência de uso dos componentes (GRESSE VON WANGENHEIM et al., 2018), conforme apresentado na Figura 16. O Script de telas do App Inventor analisa o código do projeto desenvolvido com o App Inventor (arquivo .aia) em três etapas:

1. O código do projeto (arquivo .aia) é descompactado, lido, processado e convertido em uma string para ser mais facilmente manipulado.
2. Uma análise léxica é feita na string resultante, convertendo a sequência de caracteres em uma sequência de tokens (strings com um significado atribuído).
3. Por fim, a ferramenta passa pela lista de tokens contando a frequência de cada token, criando uma tabela de tokens e suas frequências de uso.

Figura 16 – Processo de extração componentes de UI



Fonte: Elaborado pela autora (2022)

Assim, foram obtidos os vetores com 22 features/componentes de UI, criando um vetor esparso de alta dimensionalidade em que cada entrada representa um componente específico do App Inventor e o valor da entrada representa a presença/quantidade de componentes de UI no código do app, conforme apresentado da Tabela 18.

Tabela 18 – Componentes de UI extraídos de apps do App Inventor

COMPONENTE DE UI			
DESIGN DE INTERFACE	**MULTIMÍDIA**	**MAPS**	**SOCIAL**
Button CheckBox DatePicker Image BackgroundImage Label ListPicker ListView Notifier PasswordTextBox Slider Spinner Switch TextBox TimePicker WebViewer	ImagePicker VideoPlayer	Map	ContactPicker EmailPicker PhoneNumberPicker

Fonte: Elaborado pela autora (2022)

Universo de Referência. Foi obtido o código fonte (arquivo .aia) de 2.822 aplicativos da galeria do App Inventor e da iniciativa Computação na Escola. Desses foram removidos apps de jogos, por ter uma interface bem distinta em termos de design de interface, focando

somente em apps em português e que tenham pelo menos, um componente de UI entre os apresentados na Figura 15. Ao final, o conjunto de dados utilizado nesta pesquisa é composto de um total de 208 aplicativos da Galeria do App Inventor e aplicativos criados no contexto da iniciativa Computação na Escola, conforme demonstrado na Figura 17.

Figura 17 – Exemplos de screenshots de apps da Galeria do App Inventor e criados no contexto da iniciativa Computação na Escola

Fonte: Elaborado pela autora (2022)

Analisando a composição do universo de referência observa-se a distribuição de frequências de componentes individuais conforme apresentado na Tabela 19.

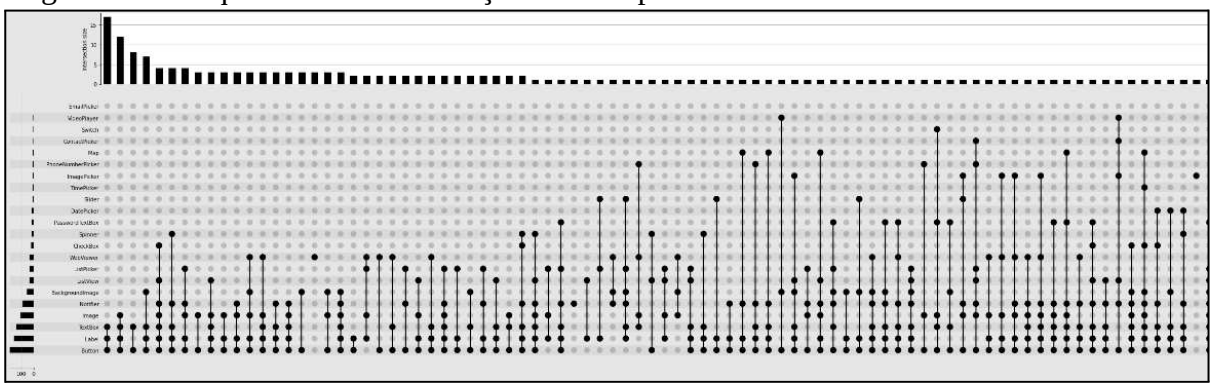
Tabela 19 – Frequência de cada componentes no Universo de Referência

COMPONENTE DE UI	QUANTIDADE	COMPONENTE DE UI	QUANTIDADE
Label	2.521	Spinner	35
Button	1.609	PasswordTextBox	25
TextBox	604	DatePicker	23
Image	409	Slider	13
CheckBox	404	Map	11
Notifier	185	PhoneNumberPicker	11
BackgroundImage	158	ImagePicker	9
ListView	75	ContactPicker	7
WebViewer	61	Switch	2
ListPicker	45	VideoPlayer	2
TimePicker	37	EmailPicker	0

Fonte: Elaborado pela autora (2022)

A distribuição de frequências em relação a combinações de componentes de UIs presentes em conjunto nos aplicativos do universo de referência é apresentada na Figura 18.

Figura 18 – Frequências de combinações de componentes de UIs no Universo de Referência



Fonte: Elaborado pela autora (2022)

É possível perceber a ocorrência maior de aplicativos contendo apenas os componentes de UI *Button*, *Label* e *TextBox*. No Apêndice B apresenta a distribuição de frequências, em relação a combinações de componentes de UIs presentes em conjunto nos aplicativos do universo de referência, por completo.

Medidas de Similaridade. A medida de similaridade mede o quanto duas instâncias são parecidas: quanto mais parecidas, maior o valor. É tipicamente determinado por medidas de similaridade dando, como resultado, um valor numérico geralmente na faixa de 0 a 1, onde 0 significa completamente diferente e 1 significa totalmente similar. Quanto mais apps similares existirem no universo de referência, menos o app é considerado original (SVANBERG 2017), podendo assim ser utilizado para medir originalidade. Desse modo, o grau de originalidade se refere à quantidade de aplicativos cuja similaridade é medida dentro do intervalo [0,1] em relação a um universo específico (p.ex., apps na Galeria do App Inventor) esteja acima de um *threshold*.

Por outro lado, medidas de distância são definidas como funções que satisfazem as propriedades métricas e correspondem a calcular a proximidade de um ponto no espaço para outro, retornando um número real positivo. Considerando medidas de distância no contexto da análise da originalidade do esqueleto do design de interface, a **originalidade** é definida pela distância entre os apps, quanto maior a distância entre os apps, maior a originalidade.

Para avaliar um aplicativo (aplicativo que irá ser avaliado pelo modelo), os respectivos componentes de UI do aplicativo são extraídos e comparados com os componentes de UIs extraídos de aplicativos do universo de referência (208 aplicativos). A comparação inclui quantas vezes cada componente de UI aparece no aplicativo vs. no universo de referência e

qual(is) combinação(ões) ocorre(m) no aplicativo vs. no universo de referência, utilizando de uma medida de similaridade/distância.

Durante a pesquisa exploratória, foram experimentadas diferentes medidas de similaridades/distâncias incluindo *Cosine*, *Euclidean* e *Jaccard*.

***Cosine*.** Na medida de similaridade *cosine*, é avaliada a semelhança entre dois vetores diferentes de zero, medindo o *cosine* do ângulo entre eles. Na pesquisa exploratória foi utilizado o *Cosine* com os vetores de Frequência, Binário e TF-IDF. Após coletar o saco-de-palavras de cada app é aplicada a métrica de *Cosine* para medir a semelhança entre os dois vetores, medindo o *cosine* do ângulo entre eles, projetados em um espaço multidimensional.

***Euclidean*.** Na medida de distância *Euclidean* é baseado na distância relativa a dois vetores distintos representados no espaço bi-dimensional. É calculado a diferença entre os dois pontos projetados em um plano. Na pesquisa exploratória foi utilizada esta medida de distância para os vetores de frequência, binário e TF-IDF.

***Jaccard*.** Para a medida de similaridade *jaccard* é comparado dois vetores usando conjuntos, em que é possível definir quais elementos são compartilhados e quais são distintos entre si. Sendo assim, é aplicado apenas o vetor binário utilizando valores entre 0 e 1, sendo que o valor mais próximo a 1 representa uma semelhança maior entre os dois conjuntos analisados.

Além dessas medidas, foi criada uma medida customizada *Combined similarity* para percorrer um aplicativo para os componentes de UIs presentes no universo de referência, adicionando as contagens de combinações de componentes de UIs presentes no aplicativo e somando com o número de ocorrência dos componentes de UI, conforme detalhado abaixo:

Combined similarity = (Frequência normalizada de ocorrência de elementos + Frequência normalizada de combinações de elementos) / 2, sendo:

- **Frequência normalizada de ocorrência de elementos:** número de ocorrência dos componentes de UI do app. Esse valor é normalizado em relação ao número de

ocorrência do universo de referência, coletando o valor total das ocorrências para o componente em questão e dividindo pelo total de apps no universo de referência. Esses valores normalizados são somados a todos os componentes do app com o valor normalizado e dividido pela quantidade de componentes do app.

- **Frequência normalizada de combinações de elementos:** número de combinações de componentes de UI iguais ao app de consulta. Esse valor é normalizado em relação ao número de combinações do universo de referência, coletando o valor total de combinações iguais para o app em questão e dividindo pelo total de combinações no universo de referência.

Comparação do desempenho das diferentes medidas de similaridade/distância. Para avaliar o desempenho das alternativas das medidas de similaridade/distância, foi conduzido um estudo exploratório comparando os resultados gerados automaticamente com um *ranking* manual criado por pesquisadores na área de design de interface. Para esta avaliação exploratória, foi selecionado um conjunto de 208 projetos de apps do App Inventor da Galeria do App Inventor e da iniciativa Computação na Escola/INCOD/INE/UFSC para compor o universo de referência. Foi selecionado um conjunto de testes com 10 aplicativos escolhidos da Galeria do App Inventor e da iniciativa Computação na Escola/INCOD/INE/UFSC para servir como apps de consulta. Este conjunto de apps consulta foi montado inicialmente selecionando 100 apps aleatoriamente, dos quais foram selecionados 10 apps para incluir tanto apps muito originais como também pouco originais para assegurar uma boa representatividade em relação a diferentes graus de originalidade. A lista dos 10 apps de consulta, juntamente com a lista de componentes de UI/combinação e frequência dos componentes de UI para cada app, está documentada no Apêndice A.

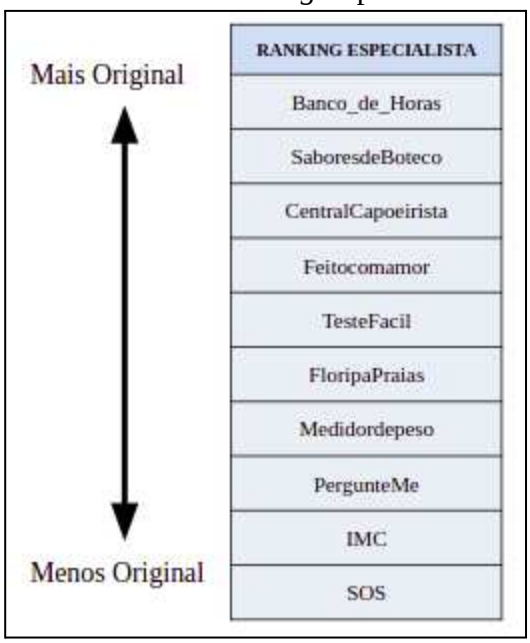
Os 10 apps consulta foram manualmente ranqueados em termos de originalidade considerando apenas a dimensão esqueleto do design de interface, conforme as seguintes regras, nesta ordem:

1. Grau de frequência da combinação dos componentes do app;
2. Grau de frequência dos componentes individuais;

3. Se tem só um ou muitos componentes em comum é considerado menos original.

O *ranking* foi realizado por três pesquisadores (especialistas em design de interface) da iniciativa Computação na Escola/INCoD/INE/UFSC em conjunto, até que um consenso foi obtido, conforme apresentado na Tabela 20.

Tabela 20 – *Ranking* Especialista



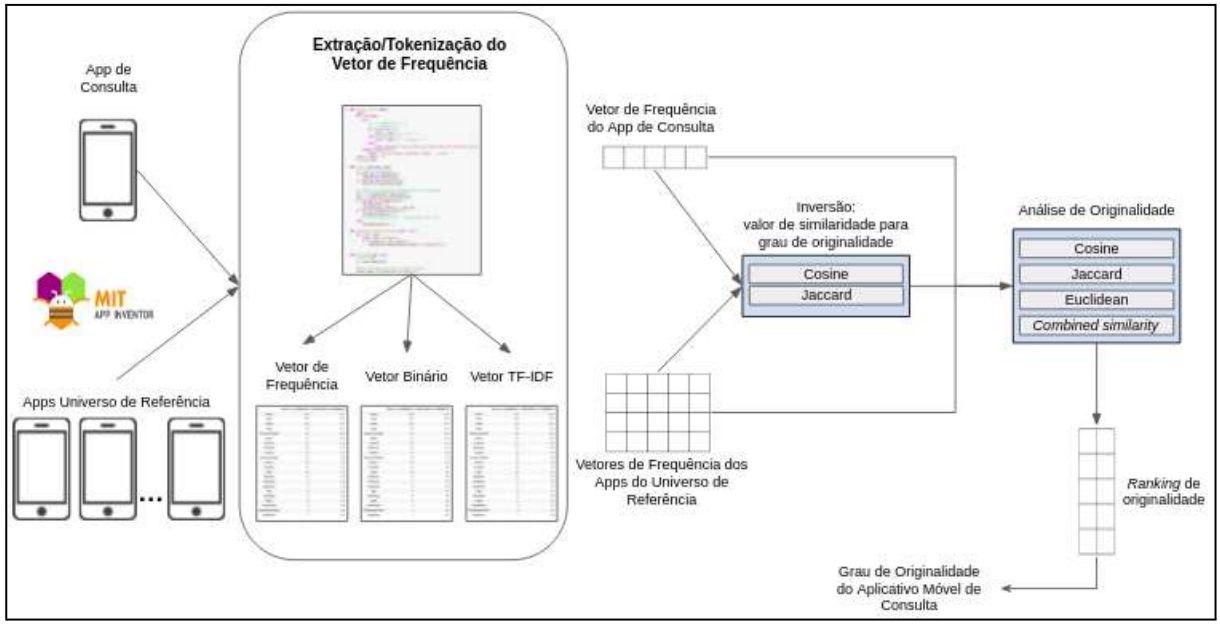
Fonte: Elaborado pela autora (2022)

Aplicando a abordagem proposta a todos os apps, foram extraídas automaticamente os componentes de UI. Em seguida foi calculado automaticamente o grau de originalidade de cada um dos 10 apps de consulta em relação ao universo de referência. Visando identificar a originalidade de esqueleto do design de interface utilizando o código de apps como entrada, a análise de originalidade é feita com base nos vetores de frequência de *features* extraídas dos códigos do projeto .aia do App Inventor, representando as quantidades de componentes de design em arquivos .scm.

Adotando uma abordagem por medidas de similaridade/distância para identificar aplicativos originais em um universo de referência para um aplicativo móvel de consulta (aplicativo móvel criado por um aluno como resultado do processo de aprendizagem) o aplicativo móvel de consulta é comparado com cada um dos aplicativos móveis de um universo de referência calculando o grau de originalidade.

Para as medidas de similaridade, *Cosine* e *Jaccard*, é realizada a inversão do valor de similaridade para o grau de originalidade. O valor da similaridade é sempre um valor entre 1 (maior valor de uma medida de similaridade) e 0, onde 1 significa que dois apps são iguais e 0 significa que dois apps são diferentes. Nesse caso, para obter o grau de originalidade realizamos a inversão do valor de similaridade para o grau de originalidade diminuindo o valor da similaridade por 1. Essa mesma abordagem não é aplicada para a medida de distância *Euclidean* e para a medida *Combined Similarity*. Para a medida de distância *Euclidean* não se faz necessário a inversão, pois quanto maior a distância entre os apps, maior a originalidade. Com relação a medida *Combined Similarity* o cálculo é proporcional à similaridade, ou seja, quanto maior a similaridade com outros apps existentes maior é o valor *Combined Similarity*. Porém, não é possível determinar um valor dentro do intervalo 0 e 1, impossibilitando identificar o valor máximo para realizar a inversão do valor similaridade para o grau de originalidade. Dessa forma, para a medida *Combined Similarity* é utilizado valor original. A partir dos resultados do grau de originalidade, é criado um *ranking* ordenado do app mais original para o app menos original, considerando que os aplicativos móveis mais similares são considerados menos originais e vice-versa. O processo da análise de originalidade é apresentado na Figura 19.

Figura 19 – Processo da análise de originalidade



Fonte: Elaborado pela autora (2022)

Os resultados preliminares comparando os cálculos de originalidade com o *ranking* criado por especialistas, sendo ordenado do app mais original para o menos original, é apresentado na Tabela 21.

Tabela 21 – Resultados preliminares de originalidade ordenado do app mais original para o menos original

	***Ranking* Especialistas**	***Ranking Combined Similarity***	***Ranking Cosine Binary***	***Ranking Cosine Frequency***	***Ranking Cosine TF-IDF***	***Ranking Eucledian Binary***	***Ranking Eucledian Frequency***	***Ranking Eucledian TF-IDF***	***Ranking Jaccard Binary***
Mais Original	Banco_de_Horas	Feitocomamor	Feitocomamor	Feitocomamor	Feitocomamor	Feitocomamor	Feitocomamor	Feitocomamor	Feitocomamor
	SaboresdeBoteco	Banco_de_Horas	Banco_de_Horas	Banco_de_Horas	sos	Banco_de_Horas	Banco_de_Horas	sos	Banco_de_Horas
	CentralCapoeirista	**CentralCapoeirista**	sos	sos	Banco_de_Horas	sos	sos	Banco_de_Horas	sos
	Feitocomamor	SaboresdeBoteco	CentralCapoeirista	TesteFacil	TesteFacil	CentralCapoeirista	TesteFacil	TesteFacil	CentralCapoeirista
	TesteFacil	FloripaPraias	SaboresdeBoteco	IMC	IMC	SaboresdeBoteco	IMC	IMC	SaboresdeBoteco
	FloripaPraias	TesteFacil	**FloripaPraias**	Medidordepeso	Medidordepeso	**FloripaPraias**	Medidordepeso	Medidordepeso	**FloripaPraias**
	Medidordepeso	**Medidordepeso**	TesteFacil	CentralCapoeirista	CentralCapoeirista	TesteFacil	SaboresdeBoteco	CentralCapoeirista	TesteFacil
Menos Original	PergunteMe	**PergunteMe**	Medidordepeso	FloripaPraias	FloripaPraias	Medidordepeso	FloripaPraias	FloripaPraias	Medidordepeso
	IMC	**IMC**	**IMC**	SaboresdeBoteco	PergunteMe	**IMC**	CentralCapoeirista	SaboresdeBoteco	**IMC**
	sos	**sos**	PergunteMe	PergunteMe	SaboresdeBoteco	PergunteMe	PergunteMe	PergunteMe	PergunteMe

Fonte: Elaborado pela autora (2022)

Observa-se que o *ranking* da medida *Combined Similarity* é o que mais se aproxima do *ranking* criado por especialistas, com cinco apps na mesma posição destacados em azul, sendo que o app **Feitocomamor** é o app mais distante do *ranking* criado por especialistas. Com relação às medidas de similaridade, as medidas *Cosine Binary e Jaccard Binary*, obtiveram os melhores resultados, com dois apps na mesma posição do *ranking* criado por especialistas, também destacado em azul. O mesmo ocorre com a medida de distância *Eucledian Binary*.

Para analisar a correlação entre o *ranking* de originalidade automaticamente calculado e o *ranking* criado por especialistas foi analisada a correlação entre os *rankings* utilizando a correlação de Spearman. O coeficiente de correlação de *Spearman*, rs, pode ter valores entre +1 e -1. Um rs de +1 indica uma associação perfeita entre os *rankings*, um rs de 0 indica nenhuma associação dos *rankings* e um rs de -1 indica uma perfeita associação negativa dos *rankings*. Quanto mais próximo o rs está de zero, mais fraca é a associação entre as duas notas. Os resultados preliminares da Correlação de *Spearman*, com destaque em azul para as correlações fortes (RUMSEY, 2021), são apresentados na Tabela 22.

Tabela 22 – Correlação de *Spearman*

	Correlação de *Spearman*
Medida de Similaridade Combined Similarity	**-0.918**
Medida de Similaridade Cosine Binary	**0.547**
Medida de Similaridade Cosine Frequency	0.128

	Correlação de *Spearman*
Medida de Similaridade Cosine TF-IDF	-0.055
Medida de Distância Eucledian Binary	**-0.547**
Medida de Distância Eucledian Frequency	-0.164
Medida de Distância Eucledian TF-IDF	-0.018
Medida de Similaridade Jaccard Binary	**0.547**

Fonte: Elaborado pela autora (2022)

Observa-se que as medidas de similaridade *Combined Similarity*, *Cosine Binary* e *Jaccard Binary* demonstram uma mais forte correlação (RUMSEY, 2021), sendo que a medida *Combined similarity* é a que mais se aproxima de uma correlação perfeita negativa. Conforme mencionado anteriormente, o cálculo da medida *Combined Similarity* é proporcional à similaridade, ou seja, quanto maior a similaridade com outros apps existentes maior é o valor *Combined Similarity*, então a correlação negativa com a nota de originalidade humana é considerada melhor, de maneira que quanto **maior** a nota de originalidade humana, **menor** é a similaridade com outros apps.

Já com relação às medidas de similaridade *Cosine Binary* e *Jaccard Binary* a correlação positiva significa que os *rankings* estão na mesma direção, ou seja, quanto **maior** o grau de originalidade entre dois apps, **maior** a nota de originalidade humana. Para a medida *Eucledian Binary*, a correlação negativa não é o melhor resultado, pois quanto **maior** o valor de originalidade **menor** será a nota de originalidade humana, indicando que os *rankings* não estão na mesma direção. A medida de distância, *Eucledian Binary* também obteve uma correlação forte (RUMSEY, 2021), com o mesmo valor das medidas de similaridade, porém, a correlação foi negativa.

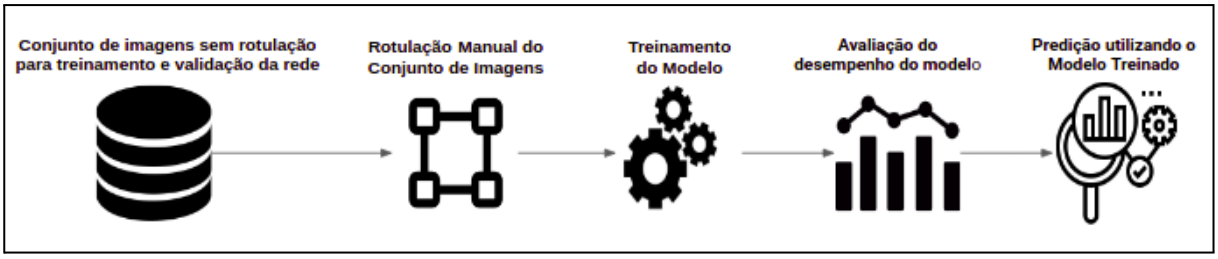
Observando o processo de análise da originalidade utilizado nesta abordagem e o valor em termos absolutos de correlação mais alto, foi selecionada a medida *Combined Similarity* (-0.918) para a análise da originalidade de *layout* de apps. Um dos pontos fracos da medida selecionada, podemos destacar quando um app novo tem apenas um componente de UI raro (com pouca ou nenhuma frequência no universo de referência). Nesses casos, o grau de originalidade é maior se comparado a nota de originalidade humana, p. ex., o app do conjunto de testes Feitocomamor considerado o mais original para a medida *Combined Similarity*, porém, na avaliação dos especialistas o app não é o mais original.

4.2.3.2 Modelo de evidência de originalidade utilizando Deep Learning

Visando identificar a originalidade de esqueleto do design de interface, uma alternativa é analisar a possibilidade de utilizar *Deep Learning* detectando automaticamente o componente de UI nos *screenshots* do novo app. Dessa maneira pode-se identificar a semelhança entre apps a partir do *layout* comparando os componentes de UI e o *layout* dos *screenshots* do app novo com o universo de referência identificando um grau de similaridade com todos os apps no universo.

Inicialmente é realizado o processo de *Machine Learning* (ML) para detecção automática dos componentes e *layout* de UI, selecionando um conjunto de imagens (*screenshots*) para ser utilizado como dados para treinamento e avaliação do desempenho do modelo de ML. Em seguida é feita a rotulação manual das imagens. O treinamento da rede neural artificial com base nas imagens já rotuladas, conforme apresentado na Figura 20.

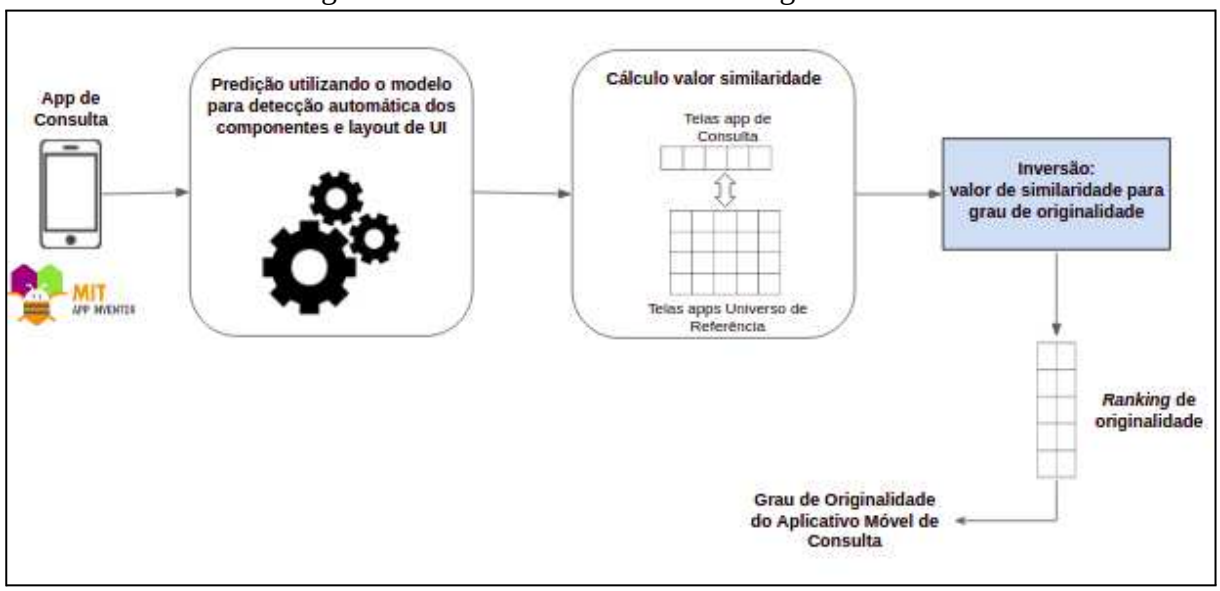
Figura 20 – Processo de machine learning para detecção automática dos componentes e *layout* de UI



Fonte: Elaborado pela autora (2022)

Após o processo de *Machine Learning*, é realizada a predição da originalidade, que tem como entrada um conjunto de *screenshots* de um app “novo” criado por um estudante como resultado de processo de aprendizagem com App Inventor e tem como resultado calcular a nota de originalidade do esqueleto do design de interface desse app em comparação com um universo de referência, conforme é apresentado na Figura 21.

Figura 21 – Processo da análise de originalidade



Fonte: Elaborado pela autora (2022)

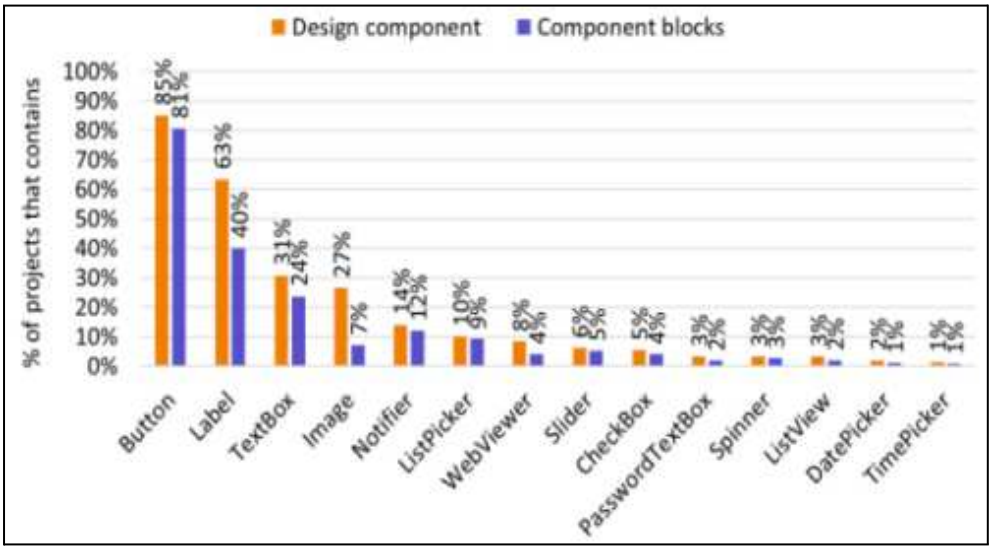
Análise dos Requisitos. O objetivo dessa etapa do presente trabalho é desenvolver um modelo de *Deep Learning* que aprenda com a experiência E para alguma classe de tarefas T e medida de desempenho P, de forma que seu desempenho para executar as tarefas em T, medido por P, melhore com E. Assim:

- Uma tarefa em T é a detecção dos componentes de UI de quais e quantos componentes cada *screenshot* possui e sua posição.
- A experiência E é um conjunto de dados rotulados de 300 *screenshots* de aplicativos desenvolvidos no App Inventor em contextos semelhantes, com um ou mais rótulos, sendo que cada anotação representa um componente de UI presente nos *screenshots* e a sua posição.
- O desempenho P é avaliado pela mAP[2] (*mean Average Precision*) do modelo, sendo que sua taxa do modelo deve ser acima de 90%.

As entradas do modelo são *screenshots* de aplicativos Android desenvolvidos com o App Inventor e rotulados manualmente por avaliadores especialistas. Devido ao grande número de componentes de interface de usuário disponibilizados pelo App Inventor, foram selecionados aqueles que aparecem com maior frequência em aplicativos do App Inventor, com base em Alves et al. (2020b), conforme apresentado na Figura 22.

[2] O mean AP médio (mAP) de todas as categorias de objetos é geralmente usado como métrica final de desempenho (ZOU et al. 2019).

Figura 22 – Fluxo do modelo de evidência



Fonte: ALVES et al. 2020b.

Porém, considerando o foco do presente trabalho em relação aos componentes de UI, observa-se que alguns dos componentes não aparecem visualmente na tela (*Notifier*) e outros são visualmente idênticos (*Button*, *ListVierw*, *DatePicker*, *TimePicker*). Sendo mais recentes, alguns componentes de UI visualmente importantes como *Map*, *Slider*, não foram considerados no levantamento feito por Alves et al. (2020b). Com base nessa análise foram selecionados um conjunto de componentes de UI considerados relevantes e possíveis de serem detectados e classificados automaticamente para o presente modelo: *Label, Button, TextBox, Image, CheckBox, ListPicker, Slider, Map* e *Switch*. A representação de todos esses componentes selecionados no App Inventor é apresentada na Figura 23.

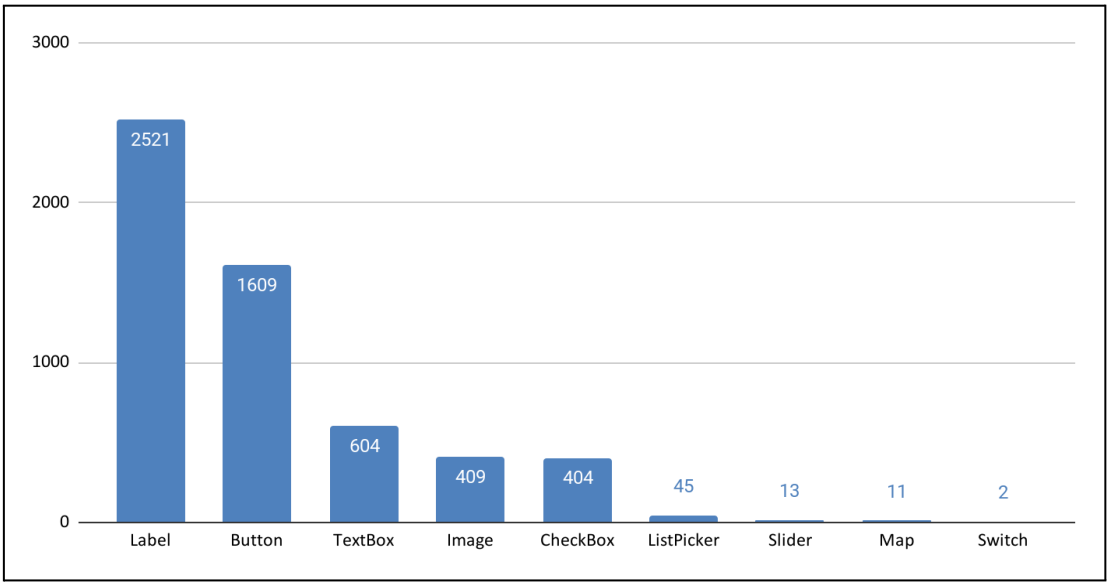
Figura 23 – Visualização dos componentes selecionados no App Inventor

Fonte: Captura de tela do App Inventor (MIT, 2020)

Conjunto de dados para treinamento do modelo. Foram selecionados 350 *screenshots* aleatoriamente e manualmente anotados por pesquisadores da iniciativa Computação na Escola.

Analisando a distribuição da aparência dos componentes UIs nas telas, observou-se que alguns componentes de UI aparecem com uma frequência muito diferente dos outros, conforme apresentado na Figura 24.

Figura 24 – Frequência dos componentes de interface no universo de referência



Fonte: Elaborado pela autora (2022)

Nota-se que há uma grande diferença na frequência dos componentes, principalmente os componentes de UI *Button* e *Switch*. Foi realizada uma busca em apps com os componentes de UI menos frequentes, mas como esses apps também contém os outros componentes de UI mais frequentes, a inclusão dessas telas não ia alterar significativamente essa distribuição. Assim, como a distribuição corresponde com a frequência de uso dos componentes, apresentado na Figura 15 se manteve o conjunto de dados criado.

Rotulação dos dados. Para o treinamento do modelo foi adotado a aprendizagem supervisionada, criando manualmente as anotações referentes ao *screenshot*. As anotações foram criadas utilizando a ferramenta *YoloLabel* (https://github.com/developer0hye/Yolo_Label). Visualmente quando a anotação de uma imagem é feita por meio do *YoloLabel*, é apresentado o *bounding box* com os "retângulos" em volta dos componentes capturados. Essa ferramenta de rotulação gera como saída um arquivo .txt, com os seguintes atributos: *class*, *center_x*, *center_y*, *width*, *height* identificando as anotações com as classes de componentes de UI e as posições dos componentes, conforme apresentado na Figura 25.

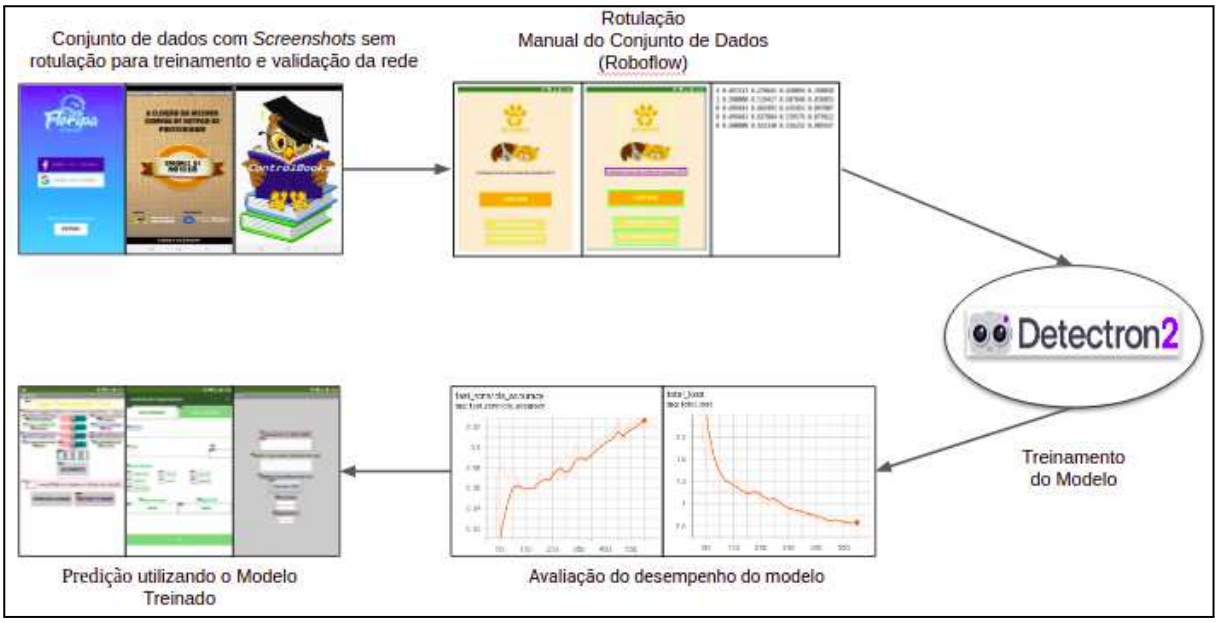
Figura 25 – Exemplo de *screenshot* e arquivo com o conjunto de *labels*

Fonte: Elaborado pela autora (2022)

Após rotular as imagens, as mesmas foram importadas para a ferramenta *Roboflow* (https://roboflow.com/), para a exportação no formato .COCO, formato padrão utilizado pelo modelo selecionado.

Seleção/Treinamento do Modelo. Foi adotada uma abordagem de aprendizagem supervisionada utilizando de detecção de objetos, conforme apresentado na Figura 26.

Figura 26 – Processo de treinamento do modelo para detecção automática dos componentes e *layout* de UI



Fonte: Elaborado pela autora (2022)

Para detectar automaticamente os componentes e *layout* de UI em *screenshots* de apps foi utilizado a detecção de objetos *Detetron2* com o algoritmo *faster_rcnn_X_101_32x8d_FPN_3x* (HONDA, 2020). *Detectron2*, é uma biblioteca mantida e disponibilizada pela Meta AI, que se coloca como *estado da arte* para detecção e segmentação de imagens (WU et al., 2019a). É a biblioteca de próxima geração do *Facebook* AI *Research* que fornece algoritmos de detecção e segmentação de última geração e tem como sucessor do *Detectron* e *maskrcnn-benchmark* (WU et al., 2019a). *Detectron2* é implementada em *PyTorch* , é flexível e extensível, é capaz de fornecer treinamento rápido em um ou vários servidores GPU. *Detectron2* inclui implementações de alta qualidade de algoritmos de detecção de objetos de última geração, incluindo *DensePose*, redes de pirâmide de recursos panópticos e inúmeras variantes do pioneiro *Mask R-CNN* família de modelos também desenvolvida pela *FAIR* (WU et al., 2019b).

O treinamento foi realizado utilizando pesos pré-treinados disponibilizados pelo próprio *Detectron2* e adaptando os arquivos necessários para alcançar o melhor resultado, modificando o número de épocas, *base.lr* e *batch* conforme apresentado na Tabela 23.

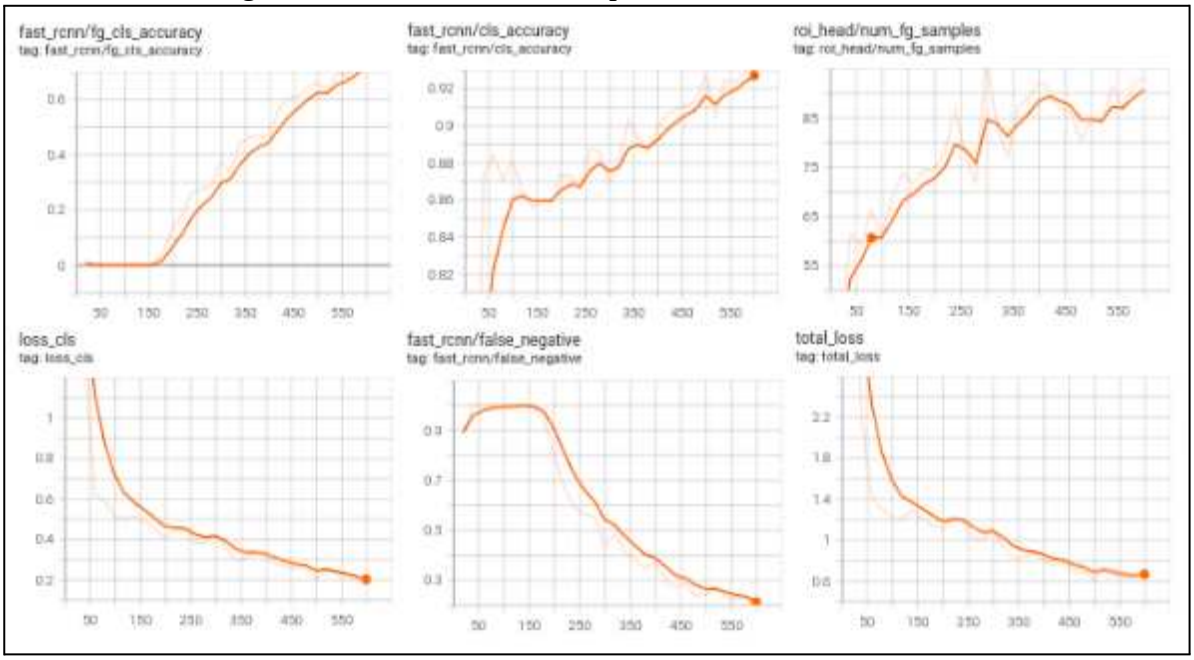
Tabela 23 – Parâmetros utilizados no treinamento do modelo com *Detectron2*

Configuração	Valor
Quantidade de épocas	O treinamento foi iniciado com 50 épocas, porém, à medida que o número de épocas eram aumentadas, era identificado melhora do desempenho do modelo, a quantidade de épocas foi aumentada para 600.
Tamanho do *batch*	4
Taxa de Aprendizagem	0.001
Quantidade de classes utilizadas	9
Quantidade total de imagens usadas para treinamento e validação	365 imagens
Divisão do conjunto de dados	O conjunto de dados foi dividido aleatoriamente em um conjunto de treinamento com 292 imagens (80%) e um conjunto de validação com 73 imagens (20%).

Fonte: Elaborado pela autora (2022)

O treinamento da rede foi realizado utilizando um *Jupyter Notebook* no *Google Colab*. Na avaliação do treinamento do modelo de detecção de objetos várias medidas foram analisadas, disponibilizadas pelo próprio *Detecron2*, conforme apresentado na Figura 27.

Figura 27 – Métricas de desempenho do treinamento da rede



Fonte: Elaborado pela autora (2022)

Pode-se notar um alto crescimento na métrica da acurácia conforme aumenta o número de iterações, já com relação ao total de *loss* pode-se observar que a medida em que as iterações aumentam exponencialmente, a média da probabilidade prevista diminui.

Avaliação do desempenho do modelo de detecção de objetos. Para avaliar o desempenho na detecção de objetos, a medida mais comum é a Precisão Média (*Average Precision* - AP[3])

[3] AP (*Average Precision*) é a avaliação mais utilizada para detecção de objetos. AP é definido como a

(ZOU et al., 2019). A Tabela 24 apresenta todas as precisões da rede e a Tabela 25 apresenta as precisões por componente de UI após o treinamento da rede com 600 épocas.

Tabela 24 – Precisões do modelo treinado

AP (Precisão Média)	AP50 (Taxa de Sobreposição de 50%)	AP75 (Taxa de Sobreposição de 75%)	mAP (média de AP)
28.963	47.207	33.646	29.074

Fonte: Elaborado pela autora (2022)

Para definir os valores de mAP devem ser considerados os intervalos de confiança de IOU[4] (*Intersection over Union*), que calcula um valor para saber se a *bounding box* prevista é compatível com a *bounding box* previamente rotulada (ZOU et al., 2019). Usa-se tradicionalmente 0,5, com uma taxa de sobreposição de 50%, que indica o quanto a *bounding box* prevista é compatível com a *bounding box* previamente rotulada (quanto elas se sobrepõem). Para o cálculo do AP é usado os valores de 50% a 95% com amostragem de 5% em 5%, com isso temos um resultado de 10 valores de APs, então é calculada uma média para fornecer um valor único. Nota-se que o AP (IOU = 50) atinge uma precisão de 47,207%. O AP (IOU 75%) atinge uma precisão de 33,646%.

Tabela 25 – Precisões por componente de UI

Class	Images	Labels	AP
todos	**73**	**644**	**28.963**
Button	73	171	61.894
Map	73	5	0
Label	73	224	41.134
TextBox	73	99	33.309
Image	73	41	40.907
Slider	73	21	0
CheckBox	73	32	52.024
Switch	73	15	31.403
ListPicker	73	36	0

Fonte: Elaborado pela autora (2022)

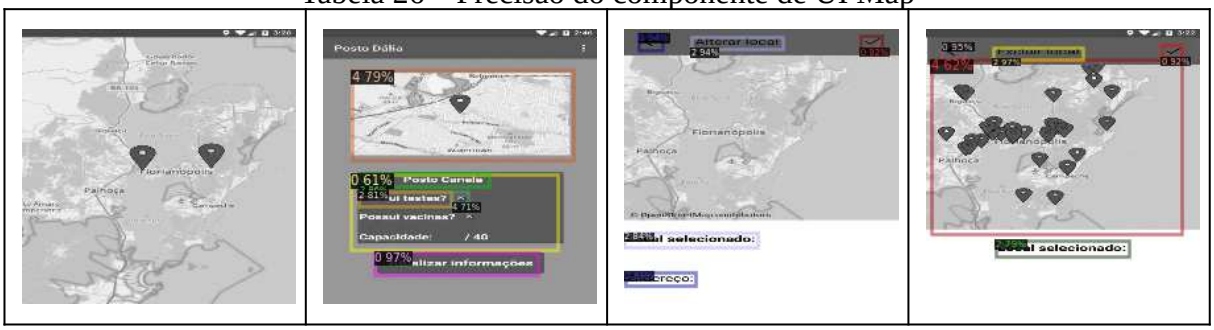
Analisando os componentes individualmente pode-se observar que, de forma geral, o AP de alguns componentes está acima de 20%, com apenas dois componentes de UI atingindo um valor acima de 50%. O AP dos componentes de UI *Map*, *Slider* e *ListPicker* atingiram

precisão de detecção média em diferentes *recalls* (ZOU et al. 2019).

[4] IOU (*Intersection over Union*) métrica utilizada para avaliar algoritmos de *Deep Learning*, estimando quão bem uma máscara ou *bounding box* prevista corresponde aos dados de verdade (HASTY, 2022).

0%, provavelmente pelo conjunto de testes conter poucos componentes desses tipos, conforme é possível observar o componente *Map* na Tabela 26.

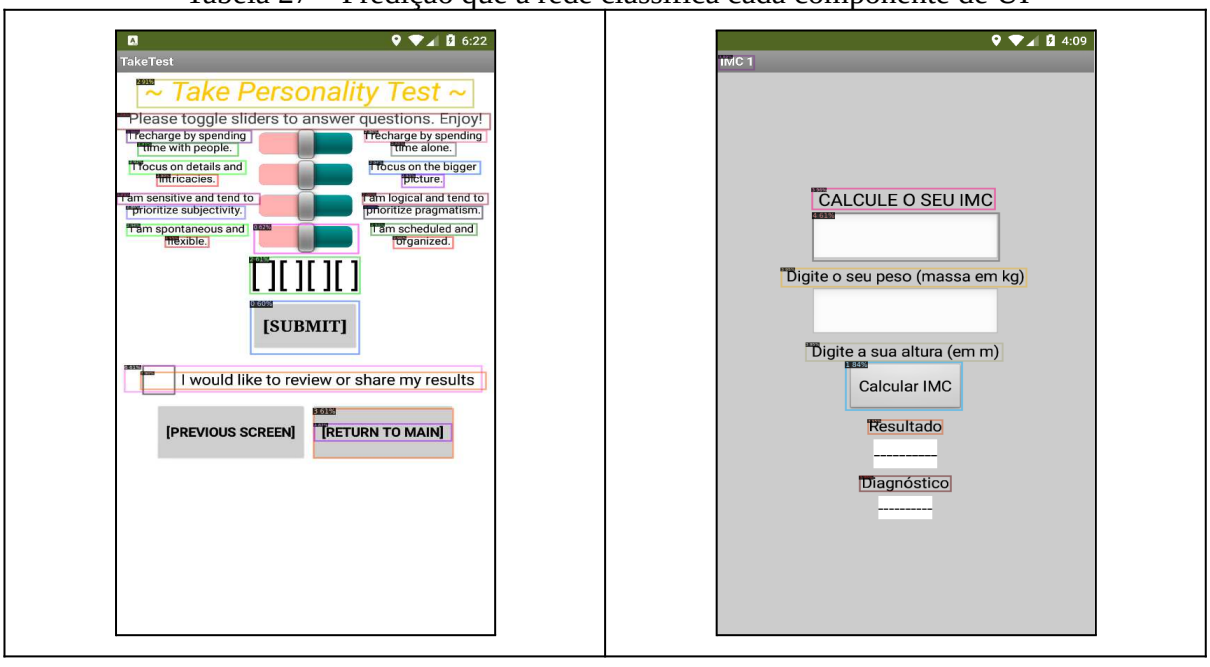
Tabela 26 – Precisão do componente de UI Map



Fonte: Elaborado pela autora (2022)

Além da análise dos resultados da validação, foi realizado também uma avaliação de desempenho utilizando um conjunto de testes composto de 10 imagens novas que não foram utilizadas anteriormente no treinamento/validação. Os resultados desse teste com a predição utilizando o modelo treinado estão apresentados na Tabela 27.

Tabela 27 – Predição que a rede classifica cada componente de UI

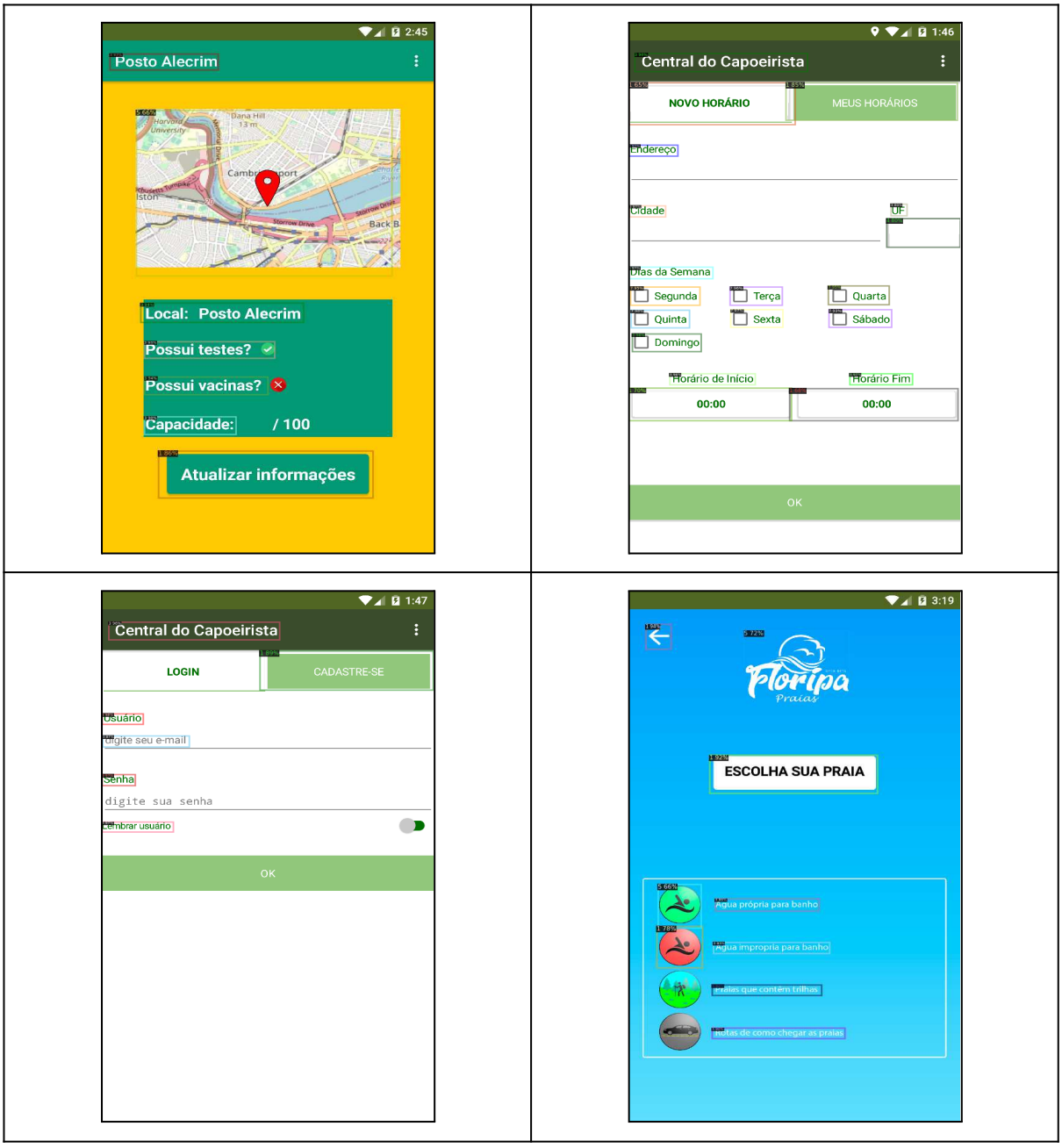



Concurso Sabores de Boteco - Login
Entre ou cadastre-se!
Digite seu e-mail:
Digite seu E-mail
Digite sua senha:
Digite a sua senha:
ENTRAR
CADASTRAR


Banco_de_Horas
Entrada Manhã
Saída Manhã
Entrada a Tarde
Saída a Tarde


Screen1
Em que galáxia fica o planeta Terra?
Sua Resposta:
Hint for TextBox1
Certo / Errado
Enviar Resposta
Próxima Pergunta


Floripa
Praias
VIP
Selecione uma praia para aparecer a descrição abaixo!
Selecionar Praia
LEGENDA
Agua própria para banho
Agua impropria para banho
Praias que contêm trilhas
Rotas de como chegar as praias



Fonte: Elaborado pela autora (2022)

Nota-se na predição e após testar a rede que o componente de *Button* foi o que obteve melhor precisão nos resultados com AP de 61%. Já o componente de *Slider* foi detectado com um AP 0% e ao realizar o teste da rede é possível observar que dos quatro componentes de UI *Slider* apresentados nas imagens acima, apenas um componente de *Slider* foi identificado. O componente de *Switch*, foi detectado com uma AP de 31%, porém, ao realizar o teste com imagens que contém o componente, o mesmo não foi identificado, o contrário aconteceu com o componente de UI *Map*, o componente foi detectado com um valor 0% na precisão do componente, porém na validação da imagem o componente foi identificado. Concluímos

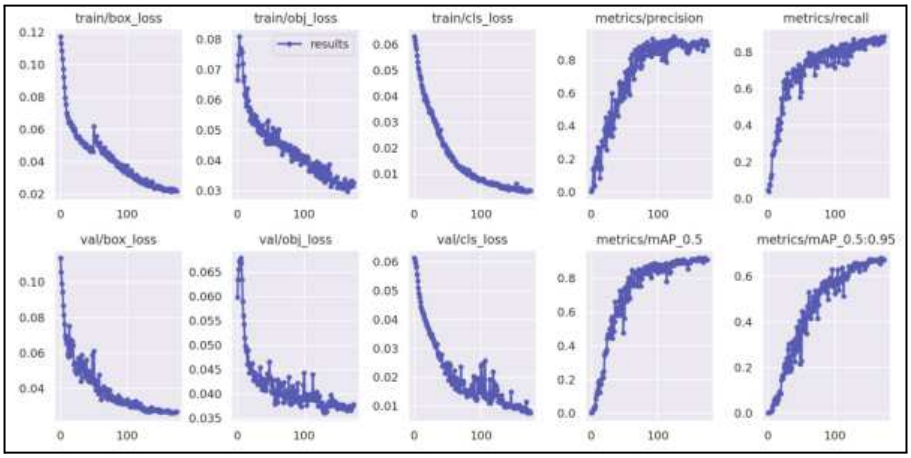
então que o modelo treinado não obteve resultados satisfatórios, não detectando corretamente os componentes de UI.

4.2.3.3 Comparação do desempenho com modelos de Deep Learning alternativos

Foi realizada uma comparação do desempenho com o modelo *Detectron2* e o *Yolov5* (KREUCH, 2022), ambos os modelos utilizaram o mesmo conjunto de dados para treinamento e avaliação do desempenho.

Modelo utilizando *YOLOv5*. Para a detecção automática dos componentes e *layout* de UI em *screenshots* de apps, Kreuch (2022) utilizou a detecção de objetos utilizando *Deep Learning*. Para o treinamento do modelo foi adotado a aprendizagem supervisionada, criando manualmente as anotações referentes ao *screenshot*, utilizando a ferramenta *YoloLabel*. Foi utilizado o *YOLOv5* e dentro da família dos modelos utilizou a versão *YOLOv5s*. O *YOLO* é um dos algoritmos de detecção de objetos mais famosos devido à sua velocidade e precisão (ULTRALYTICS, 2021). Sendo assim, optou-se pelo *YOLOv5s* por ser a versão atual e também por motivos tecnológicos, uma vez que os outros modelos exigem mais memória e CUDA para treinar o modelo e são muito mais lentos para executar o treinamento. O treinamento foi realizado utilizando pesos pré-treinados disponibilizados pelo próprio *YOLOv5s* e adaptando os arquivos necessários, modificando o número de classes, *epochs* e *batch_size* (KREUCH, 2022). Um alto crescimento das métricas de precisão, recall e mAP conforme aumenta o número de iterações, aumenta a precisão da classificação e diminui as taxas de erros, conforme é possível observar na Figura 28.

Figura 28 – Métricas de desempenho do treinamento da rede utilizando *YOLOv5s*



Fonte: Kreuch (2022)

Avaliação do desempenho dos modelos. O desempenho do modelo de detecção de objetos do *YOLOv5s* é documentado na Tabela 28.

Tabela 28 – Desempenho *YOLOv5s*

Classe	Imagens *YOLOv5s*	*Labels YOLOv5s*	mAP *YOLOv5s*
Todos	**73**	**644**	**91.4**
Button	73	171	94.2
Map	73	5	79.9
Label	73	224	90.2
TextBox	73	99	92.4
Image	73	41	84.3
Slider	73	21	92.0
CheckBox	73	32	95.7
Switch	73	15	99.5
ListPicker	73	36	94.6

Fonte: Elaborado pela autora (2022

Kreuch (2022) apresenta a mAP de cada classe (componentes de UI), já as métricas disponibilizadas pelo modelo *Detectron2* são a AP, mesmo sendo métricas distintas, é possível observar uma superioridade considerável maior do modelo *YOLOv5s*. Com relação a mAP de todas as classes, o *YOLOv5s* apresentou o resultado de 91,4 superando o resultado 29,074 *Detectron2* apresentado no tópico Avaliação do desempenho do modelo de detecção de objetos referente ao *Detectron2*.

Uma comparação foi realizada com relação a validação das imagens, após o treinamento dos modelos, conforme apresentado na Tabela 29.

Tabela 29 – Grau de predição que das redes classificadas para cada componente de uma tela

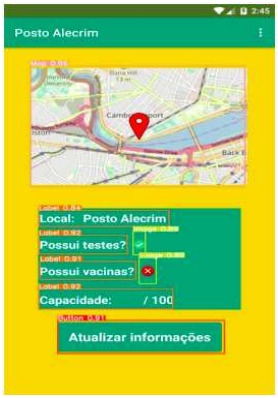
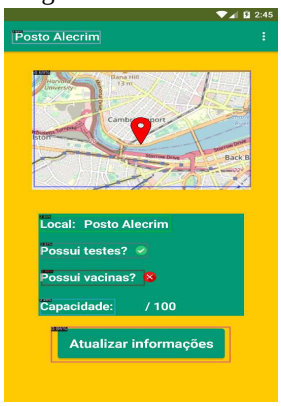
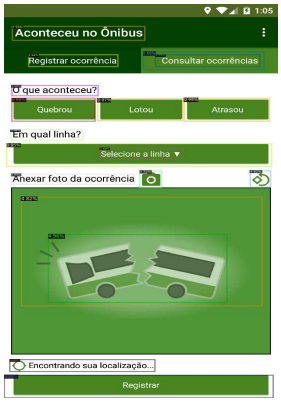
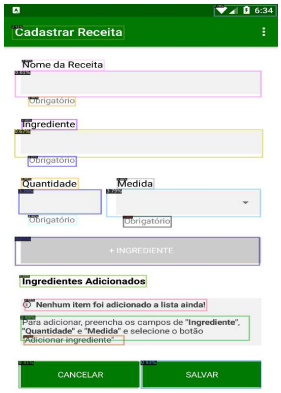
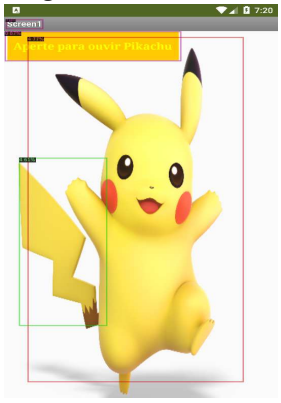
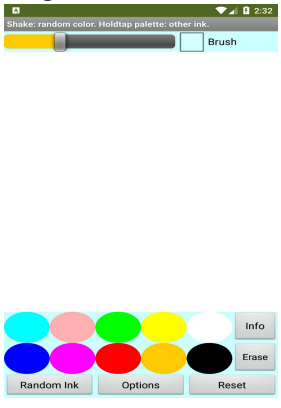
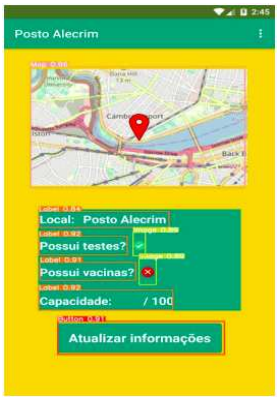
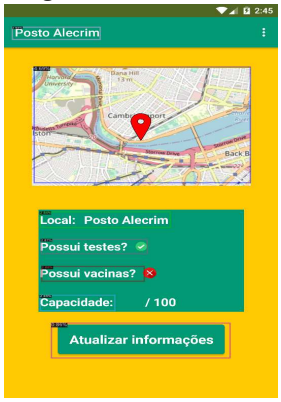
Imagem 1 - *YOLOv5s*	Imagem 1 - *Detectron2*	Imagem 2 - *YOLOv5s*	Imagem 2 - *Detectron2*
			
Imagem 3 - *YOLOv5s*	Imagem 3 - *Detectron2*	Imagem 4 - *YOLOv5s*	Imagem 4 - *Detectron2*
			
Imagem 5 - *YOLOv5s*	Imagem 5 - *Detectron2*	Imagem 6 - *YOLOv5s*	Imagem 6 - *Detectron2*
			
Imagem 7 - *YOLOv5s*	Imagem 7 - *Detectron2*	Imagem 8 - *YOLOv5s*	Imagem 8 - *Detectron2*

Imagem 1 - *YOLOv5s*	Imagem 1 - *Detectron2*	Imagem 2 - *YOLOv5s*	Imagem 2 - *Detectron2*
			

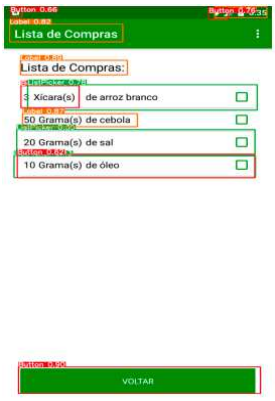
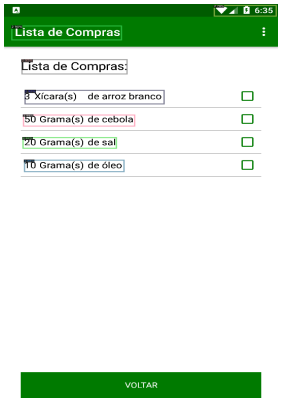
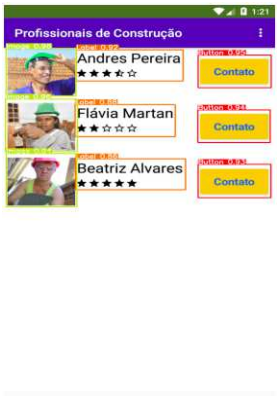
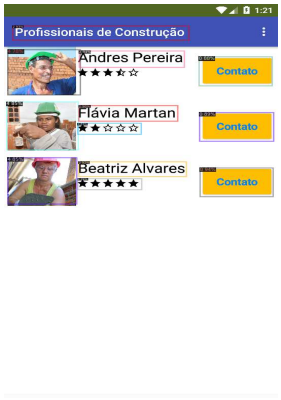

	

Imagem 9 - *YOLOv5s*	Imagem 9 - *Detectron2*	Imagem 10 - *YOLOv5s*	Imagem 10 - *Detectron2*
			

Fonte: Elaborado pela autora (2022)

Observa-se que na Imagem 4 o *YOLOv5s* identificou corretamente o componente de *Slider*, já o *Detectron2* não identificou o componente e também, não identificou os *buttons* presentes no *screenshot*. Na Imagem 7 o *Detectron2* identificou o componente *label* ao invés do componente *checkbox*, como identificou o corretamente o *YOLOv5s*.

Foi realizada também uma comparação com relação a quantidade de componentes de UI utilizados no presente trabalho, conforme apresentado na Tabela 30.

Tabela 30 – Comparação em relação a quantidade de componentes de UI identificados

Imagem	Componente de UI	Elementos existentes	*Detectron2*	*YOLOv5s*
Imagem 1	Button	4	3	3
	Label	8	4	2
	Image	1	1	1
Imagem 2	Button	1	1	1
	Map	1	1	1
	Label	16	5	4
	Image	2	0	2
Imagem 3	Button	8	5	7
	Label	9	6	5
	Image	2	2	3
	ListPicker	1	1	0
Imagem 4	Button	3	3	5
	Label	12	13	8
	TextBox	3	4	5
Imagem 5	Button	1	2	1
Imagem 6	Button	15	3	15
	Slider	1	0	1
	CheckBox	1	0	0
	ListPicker	1	0	0
Imagem 7	Button	1	0	5
	Label	1	6	3
Imagem 8	Button	3	3	3
	Label	7	1	1
	Image	2	1	3
Imagem 9	Button	1	1	1
	Label	6	2	1
	Image	2	2	1
Imagem 10	Button	4	3	3
	Label	4	4	3
	Image	3	3	3

Fonte: Elaborado pela autora (2022)

Primeiro foi identificado a quantidade de componentes em cada imagem e na sequência foi contabilizada a quantidade de componentes que cada identificou. É possível observar que para o componente *Label*, o modelo *Detectron2* identificou uma maior quantidade de componentes, já para o componente *Button*, o *YOLOv5s* obteve o melhor

resultado. Com relação ao componente *Slider,* o mesmo foi identificado apenas com o modelo *YOLO5s.*

Discussão dos modelos apresentados. Conforme é possível observar e já mencionado nas comparações acima, o modelo *YOLOv5s* obteve a média da precisão (mAP) total 91.4, enquanto o *Detectron2* obteve a média da precisão (mAP) 29.074. Com relação ao componente de Map, o resultado do *YOLOv5s,* utilizando a mesma quantidade de imagens para o treinamento, apresentou uma precisão média de 79.9, enquanto o *Detectron2* não alcançou nenhum resultado, apresentando 0 (zero) como precisão. O *Detectron2* também não apresentou nenhum resultado para os componentes de UI *Slider* e *ListPicker*, sendo que o *YOLOv5s* obteve como resultado da precisão média 92 e 94.6, respectivamente. Apenas os componentes de UI *Button* e *CheckBox* obtiveram resultado acima de 50 de precisão média no *Detectron2*, sendo mais precisamente 61.89 para *Button* e 52.02 para *CheckBox*, sendo que o *YOLOv5s* apresentou previsão média acima de 90 para 7 dos 10 componentes de UI utilizando o mesmo conjunto de dados.

Observa-se nas comparações acima que o *YOLOv5s* obteve os melhores resultados, utilizando o mesmo conjunto de dados e a mesma quantidade de classes e imagens. Conclui-se que para o presente trabalho o modelo que parece ser mais adequado é o *YOLOv5s*, considerando o tipo de imagens utilizado e o tamanho do conjunto de dados.

4.2.3.4 Análise do grau de similaridade de um screenshot utilizando YOLOv5s

Utilizando a informação dos componentes de UI e o *layout* dos *screenshots* como entrada calcula-se nesse passo um grau de similaridade dos *screenshots* de um app em relação ao universo de referência. Para essa análise determina-se primeiro o grau de similaridade de um *screenshot* criado (novo) pelo aluno como resultado de aprendizagem em relação a um universo de referência e ao final um grau de similaridade considerando todos os *screenshots* de um app. Vale ressaltar que para obter as informações dos componentes de UI e o *layout* dos *screenshots* foi utilizado os mesmos pesos da rede treinada no *YOLOv5s* (Kreuch (2022).

Cálculo do valor de similaridade. Para calcular o valor de similaridade de um novo app, utilizando o modelo de detecção de objetos treinado para identificar os componentes de UI e sua estrutura, em relação a cada um dos apps no universo de referência, foi utilizada a alternativa proposta por Mao et al. (2018) e por Kreuch (2022). Essa abordagem compara a

similaridade entre dois apps utilizando os recursos: tipo do componente de UI, a posição (coordenadas x, coordenadas y) e o tamanho (altura e largura).

Após obter essa lista de recursos sobre os apps, é observado a similaridade composto e similaridade de tipo, similaridade de posição, similaridade de tamanho e para cada uma dessas similaridades é realizado um cálculo para calcular as taxas de similaridade entre os componentes de dois apps, conforme apresentado na Tabela 31.

Tabela 31 – Conjunto de fórmulas para calcular as taxas de similaridade

Similaridade do tipo	Dados dois componentes c1 = {c1.t, c1.p, c1.s} e c2 = {c2.t, c2.p, c2.s}, a similaridade do tipo s.t entre c1 e c2
	Cálculo: $s.t$ = { 0 $c1.t \neq c2.t$ 1 $c1.t = c2.t$
Similaridade de posição	Dados dois componentes c1 e c2 com suas posições correspondentes c1.p e c2.p
	Cálculo: s.p=(1−‖c1.p−c2.p‖‖‖max‖)×exp[−(‖c1.p−c2.p‖‖‖max‖)] Onde $\|\|c1.p - c2.p\|\|$ é a distância Euclidiana entre os dois componentes, e ‖max‖ é o tamanho da diagonal da tela
Similaridade de tamanho	Dados dois componentes c1 e c2 com seu tamanho correspondente (c1.sw, c1.sh) e (c2.sw, c2.sh)
	Cálculo: Similaridade entre a largura s.sw (1) e a altura s.sh (2) s.sw=(1−|c1.sw−c2.sw|maxc1.sw,c2.sw)×exp[−(|c1.sw−c2.sw|maxc1.sw,c2.sw)] (1) s.sh=(1−|c1.sh−c2.sh|maxc1.sh,c2.sh)×exp[−(|c1.sh−c2.sh|maxc1.sh,c2.sh)] (2)
Similaridade entre o tamanho dos componentes de UI	Cálculo: s.s=√s.sw×s.sh
Similaridade dos componentes	Cálculo: s=s.t×√s.p×s.s

Fonte: Elaborado pela autora (2022)

Para refletir a importância na representação visual, utiliza-se o tamanho para derivar o peso da pontuação de similaridade. A pontuação de similaridade com o peso é calculada da seguinte forma:

$$sW=s\times c1.sw\times c1\sum.shc1i\in F1c1i.sw\times c1i.sh$$

Após calculada a taxa de similaridade de cada componente de UI, é comparado a semelhança entre pares para obter uma matriz de similaridade dos componentes de UI do app

novo e os *screenshots* do universo de referência. Então, para obter o maior valor de taxa de similaridade entre dois *screenshots* é utilizado o algoritmo húngaro de Munkres (MUNKRES, 1957), para tratar a matriz de similaridade como uma matriz de custo e obter a melhor correspondência. Como resultado, é determinado um valor de similaridade de uma tela em relação a todas as telas incluídas no universo de referência.

Avaliação do desempenho do cálculo do valor de similaridade de um app. Para avaliar o desempenho dessa abordagem foi utilizado a abordagem proposta por Kreuch (2022), bem como o mesmo conjunto de teste e o mesmo universo de referência utilizado na seção 4.2.3.1. Duas alternativas foram avaliadas, na primeira alternativa para cada *screenshot* dos apps foi calculado o valor de similaridade e em seguida foi calculada uma média ponderada de todos os *screenshots* de um app. Após calcular a média ponderada, é criado um *ranking* de similaridade utilizando como alternativas o valor de similaridade. Para calcular a média, soma-se o valor de similaridade de uma tela comparada com todas as telas do universo de referência e divide-se pelo número total de telas do universo de referência.

Na segunda alternativa foi utilizado um *threshold*, sendo utilizado como valor a mediana com o valor central de 50%. Como resultado é identificado o valor de similaridade de um app levando em consideração todos os seus *screenshots.* Para calcular o valor de similaridade de cada app, é realizado a soma da quantidade total de telas que tem uma taxa de similaridade maior que 50% (*threshold*), então é feito um *ranking* das 10 telas/apps que obtiveram mais quantidade de telas/apps similares. A definição das alternativas e valores do *threshold*, foram definidas em experimentos empíricos.

É realizada a inversão do valor de similaridade para o grau de originalidade. O valor da similaridade é sempre um valor entre 1 (maior valor de uma medida de similaridade) e 0, onde 1 significa que dois apps são iguais e 0 significa que dois apps são diferentes. Nesse caso, para obter o grau de originalidade realizamos a inversão do valor de similaridade para o grau de originalidade diminuindo o valor da similaridade por 1. A partir dos resultados do grau de originalidade, é criado um *ranking* ordenado do app mais original para o app menos original, considerando que os aplicativos móveis mais similares são considerados menos originais e vice-versa. Assim, essas medidas são comparadas com um *ranking* alocado por especialistas conforme seção 4.2.3.1. A comparação dos *rankings* de originalidade é apresentada na Tabela 32.

Tabela 32 – Comparação dos *rankings* de originalidade (10 mais original | 1 menos original)

Apps	Cálculo automático			**Média das avaliações manuais feita por especialistas**
	Originalidade média calculada automaticamente com universo de referência	*Ranking* baseado na média de similaridade	*Ranking Threshold 0,5*	
Banco_de_Horas	7.174	5	3	10
SaboresdeBoteco	7.232	7	9	9
CentralCapoeirista	7.297	3	1	8
Feitocomamor	7.299	1	6	8
TesteFacil	7.6357	8	8	7,5
FloripaPraias	7.6977	2	4	6,5
Medidordepeso	7.8164	9	10	5
PergunteMe	7.9388	4	5	4,5
IMC	7.9871	6	7	3,5
sos	8.0847	10	2	1

Fonte: Elaborado pela autora (2022)

Para avaliar a correlação das alternativas utilizou-se coeficientes de correlação. Foi aplicado o coeficiente de correlação de Spearman (SPEARMAN, 1904). O coeficiente de correlação de Spearman mede a força e a direção da associação entre duas variáveis classificadas, que resulta em um valor entre -1 e 1. Quanto mais uma variável aumenta, a outra também aumenta, então temos uma correlação positiva. Ou quanto mais uma variável aumenta, a outra diminui, resultando numa correlação negativa. Os resultados dos coeficientes de Spearman do conjunto de 10 apps de teste, para obter a correlação entre o *ranking* criado por especialistas com os rankings de média e *threshold* são apresentados na Tabela 33.

Tabela 33 – Coeficientes de correlação de Spearman

Correlação *ranking* especialistas com média de similaridade	Correlação *ranking* especialistas com *threshold*
-0.152	0.146

Fonte: Elaborado pela autora (2022)

Pode-se notar que a correlação com *threshold* foi positiva, o que indica que os valores de *threshold* estão indo em direção ao *ranking* criado por especialistas. Porém, obteve uma correlação fraca (0.146) (RUMSEY, 2021). Já a correlação com originalidade foi negativa, o que indica que os valores com originalidade não estão indo em direção ao *ranking* criado por

especialistas. Nota-se também que a correlação com originalidade obteve também uma correlação fraca (RUMSEY, 2021).

4.2.3.5 Comparação utilizando medidas de similaridade e Deep Learning

Observa-se uma diferença entre os valores de correlação entre os modelos de evidências de originalidade utilizando medidas de similaridade e *Deep Learning*. Enquanto no modelo de evidência utilizando medidas de similaridade obtivemos correlação forte (*Combined similarity* - correlação -0.918), no modelo de evidência utilizando *Deep Learning*, as duas correlações foram fracas (Correlação com Similaridade -0.152 e Correlação com *threshold* 0.146).

Outro fator a ser levado em consideração é que para criar o *ranking* manual foram considerados 22 componentes de UI, os mesmos utilizados no modelo de evidência de originalidade utilizando medidas de similaridade enquanto para o modelo de evidência de originalidade utilizando *Deep Learning* foram utilizados apenas 9 componentes de UI, conforme apresentado na Tabela 34.

Tabela 34 – Componentes de UI utilizados

Componentes de UI utilizados no *Ranking* Manual	**Componentes de UI utilizados no modelo de evidência de originalidade utilizando medidas de similaridade**	**Componentes de UI utilizados no modelo de evidência de originalidade utilizando *Deep Learning***
Button **CheckBox** DatePicker **Image** BackgroundImage **Label** **ListPicker** ListView Notifier PasswordTextBox **Slider** Spinner **Switch** **TextBox** TimePicker WebViewer ImagePicker VideoPlayer **Map** ContactPicker EmailPicker PhoneNumberPicker	**Button** **CheckBox** DatePicker **Image** BackgroundImage **Label** **ListPicker** ListView Notifier PasswordTextBox **Slider** Spinner **Switch** **TextBox** TimePicker WebViewer ImagePicker VideoPlayer **Map** ContactPicker EmailPicker PhoneNumberPicker	**Button** **CheckBox** **Image** **Label** **ListPicker** **Slider** **Switch** **TextBox** **Map**

Fonte: Elaborado pela autora (2022)

Essa divergência na quantidade de componentes utilizados entre os modelos de evidência pode ter impactado consideravelmente nos valores da correlação, uma vez que

utilizamos o mesmo *ranking* manual e o mesmo universo de referência para ambos os modelos de evidência. Tendo em vista que o objeto do presente trabalho é fornecer um método para avaliar o aluno, podemos observar que essa quantidade menor de componentes de UI no modelo de evidência de originalidade utilizando *Deep Learning* já é considerado uma restrição do modelo. Já o modelo de evidência utilizando medidas de similaridade tem a vantagem de permitir analisar uma maior quantidade de componentes de UI, uma outra vantagem desse modelo está relacionado a facilidade do cálculo e por consequência a performance em disponibilizar a nota para o aluno.

Como os valores de correlação utilizando o modelo de evidência de originalidade utilizando *Deep Learning* foram fracos e o modelo teve uma restrição da quantidade de componentes, optou-se em utilizar o modelo de evidência de originalidade utilizando medidas de similaridade para criar uma nota de originalidade considerando uma análise maior da quantidade de componentes de UI, a facilidade do cálculo da nota e por consequência a performance em disponibilizar a nota para o aluno.

4.2.3.6 Cálculo da nota de originaldiade

Usando o grau de similaridade medido por *Combined Similarity* (-0.918) como entrada é definido o cálculo da nota de originalidade. Observa-se que a originalidade representa o inverso da similaridade. A nota de originalidade é definida no intervalo de [0 .. 10] conforme as notas tipicamente utilizadas na educação básica em escolas brasileiras. Dessa forma a nota 10 representa um aplicativo muito original e a nota 0 a um aplicativo nada original em relação à dimensão esqueleto do design de interface.

A transformação do valor da medida de *Combined similarity* de um aplicativo em uma nota de originalidade é feita por uma transformação usando uma função. A função foi definida com base numa alocação manual de notas de originalidade por três especialistas utilizando os 10 apps de consulta e os valores de originalidade calculados automaticamente. Para calcular os valores de similaridades foi utilizado o universo de referência apresentado na seção 4.2.3.1, tópico Universo de Referência. Os valores de similaridade e notas de originalidade para o conjunto de testes são apresentados na Tabela 35.

Tabela 35 – Valores de similaridade e notas de originalidade para o conjunto de testes

Apps	**Valor de similaridade calculado automaticamente com *Combined Similarity***	**Nota de originalidade manualmente alocado por especialistas** Nota 10 para o app mais original Nota 1 para o app menos original
Banco_de_Horas	0.999	10
SaboresdeBoteco	1.493	9
CentralCapoeirista	1.378	8
Feitocomamor	0.159	8
TesteFacil	1.972	7,5
FloripaPraias	1.635	6,5
Medidordepeso	2.969	5
PergunteMe	3.141	4,5
IMC	3.864	3,5
sos	3.868	1

Fonte: Elaborado pela autora (2022)

Para identificar a função de transformação foram experimentados vários *Fit Method* da ferramenta do site *MycurveFit* (https://mycurvefit.com), para comparar alternativas de funções. Para os valores comparativos, serão usados os valores da medida como X e as médias da nota manual dos especialistas como Y. As funções alternativas são avaliadas pelo *Goodness Measure* coeficiente de determinação (R^2). Essa é uma medida estatística que representa quão bem as previsões de regressão se aproximam dos pontos de dados reais (MYCURVEFIT, 2022; ORIGINLAB, 2022). Um valor R^2 mais próximo à 1 indica que as previsões de regressão se ajustam perfeitamente aos dados, por outro lado, um R^2 mais próximo à 0 indica um ajuste imperfeito (MYCURVEFIT, 2022; ORIGINLAB, 2022). Foram analisados vários modelos *Fit Method* para identificar qual se aproxima mais de 1 com relação a medida R^2, conforme apresentado na Tabela 36.

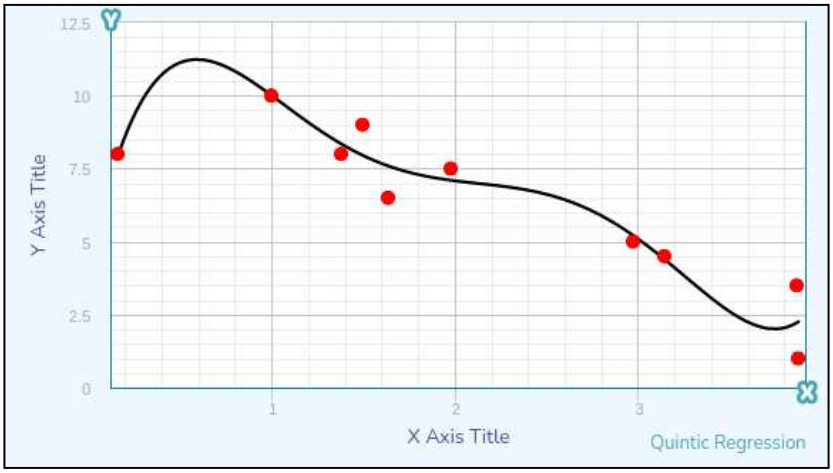
Tabela 36 – Análise dos modelos *Fit Method* e valores R^2

4PL	Michaelis Menten	Exponential	Power	Linear Regression	**Quintic Regression**	Quartic Regression	Quadratic Regression	Cubic Regression
0.8746	-3.538	0.7901	0.3119	0.7901	**0.9166**	0.9077	0.8757	0.8858

Fonte: Elaborado pela autora (2022)

Conforme apresentado na Tabela 36, o modelo que mais se aproxima do ajuste perfeito (1) é o modelo *Quintic Regression*. A curva da função de transformação com base na medida de *Combined Similarity* é apresentada na Figura 29.

Figura 29 – Função de transformação com base na medida de *Combined Similarity*



Fonte: Elaborado pela autora (2022)

Os valores da *Goodness Measures* gerada pela medida de *Combined similarity* é apresentada na Figura 30.

Figura 30 – *Goodness Measures* gerada pela medida de *Combined similarity*

Goodness Measures What's this?		
R^2		0.9166
aR^2		0.8123
P		0.02793
SE		1.2
F		8.792
AIC		34.87
BIC		36.68
DWS		2.924
DoF		4
AICc		62.87
Coefficients		
a	4.201136	± 1.418
b	30.26846	± 3.415
c	−43.83095	± 4.067
d	25.16233	± 7.525
e	−6.426354	± 1.982
f	0.5995241	± 2.853
Equation		
$y = 4.201136 + 30.26846x - 43.83095x^2 + 25.16233x^3 - 6.426354x^4 + 0.5995241x^5$		

Fonte: Elaborado pela autora (2022)

Dessa forma o valor de originalidade automaticamente calculado usando *Combined Similarity* é transformado em uma nota de originalidade utilizando a seguinte fórmula:

Nota de originalidade = 4.201136 + 30.26846*CombinedSimilarity* − 43.83095*CombinedSimilarity*^2+ 25.16233*CombinedSimilarity*^3 − 6.426354*CombinedSimilarity*^4 + 0.5995241*CombinedSimilarity*^5

5 CONCLUSÃO

Tendo em vista o desenvolvimento de um modelo de avaliação da originalidade de componente de UI de aplicativos móveis desenvolvidos com App Inventor na educação básica, foi feita a síntese em relação aos conceitos de originalidade, design de interfaces de usuário, medidas de similaridade e *deep learning* (Objetivo 1). A análise do estado da arte foi realizada em relação a análise automática da originalidade de design de interfaces de apps (Objetivo 2). Os resultados do objetivo 2 demonstram que o tema de avaliação de originalidade enfocando no design de UI de apps Android é recente e ainda pouco abordado. A maioria das abordagens encontradas avaliam a originalidade utilizando como base a estrutura do *layout* do aplicativo (*screenshots* ou *sketches*). Visando a automatização da avaliação da originalidade de apps desenvolvidos no contexto do ensino de computação observou-se ainda uma grande lacuna apontando a necessidade de mais pesquisas nesta área. A maioria das abordagens são utilizadas no contexto profissional, sendo que poucas estão relacionadas ao contexto educacional. Várias abordagens focam no processo de desenvolvimento da solução e não em detalhes do modelo resultante, resultando na falta de informações sobre a avaliação do desempenho, como fatores avaliados, métodos de coleta de dados, tamanho da amostra utilizada e suas respectivas descobertas.

Seguindo o ECD (MISLEVY et al., 2003) foi realizada a análise do contexto da avaliação de ensino da computação na Educação Básica (Objetivo 3), detalhando os objetivos de aprendizagem (modelo do aluno) e o potencial artefato de trabalho (modelo de tarefa). Após isso, foram experimentados dois modelos de evidência de originalidade, um modelo de avaliação da originalidade em relação aos componentes de UI presentes em um aplicativo do App Inventor utilizando medidas de similaridades e outro modelo de originalidade para automaticamente avaliar a originalidade do esqueleto de design de interface de um app criado também utilizando o App Inventor (Objetivo 4). Como resultado observou-se um desempenho melhor do modelo de avaliação da originalidade utilizando medidas de similaridades. Como principal resultado científico foi desenvolvido um modelo de avaliação utilizando medidas de similaridades que avaliam a originalidade de aplicativos criados no App Inventor, extraindo cada componente de UI presente nos aplicativos móveis e atribuindo uma nota para seu nível de originalidade no intervalo de [0 .. 10] conforme as notas tipicamente utilizadas na educação básica em escolas brasileiras, sendo atribuído a nota 10 para um aplicativo muito

original e a nota 0 a um aplicativo nada original em relação à dimensão esqueleto do design de interface. Com isso, espera-se contribuir para a avaliação do desenvolvimento da criatividade de alunos na educação básica no contexto de desenvolvimento de aplicativos. Com relação ao modelo de evidência utilizando *Deep Learning*, os resultados obtiveram correlações fracas (Correlação com Similaridade -0.152 e Correlação com *threshold* 0.146) o que impossibilitou a utilização da mesma como ferramenta para avaliar a originalidade de aplicativos móveis.

No decorrer deste trabalho, foi realizada a seguinte publicação científica:

- SOUZA, A. S, DA CRUZ ALVES, N., GRESSE VON WANGENHEIM, C., KREUCH, L. Análise Automatizada da Originalidade de Design de Interfaces de Usuário no Contexto Educacional: Um Mapeamento da Literatura. SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 32. , 2021, Online. Porto Alegre: Sociedade Brasileira de Computação, 2021. p. 1140-1151. DOI: https://doi.org/10.5753/sbie.2021.217542. **Qualis B1 - Ciência da Computação.**

Para trabalhos futuros para a abordagem sugere-se uma análise com maior ênfase em identificar outros possíveis componentes de UI contidos em um aplicativo no contexto do App Inventor, assim, podendo avaliar a originalidade de um aplicativo de forma mais ampla. Com relação a avaliação, é sugerida a possibilidade de apresentar um *feedback* qualitativo ao aluno com foco nas competências que precisa desenvolver ou se está desenvolvendo-as como deseja. Também é sugerida a integração deste modelo na ferramenta *CodeMaster*.

REFERÊNCIAS

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

ALVES, N. d. C. , GRESSE VON WANGENHEIM, C., RODRIGUES, P. E., HAUCK, J. C. R., BORGATTO, A. F. Ensino de Computação de Forma Multidisciplinar em Disciplinas de História no Ensino Fundamental – Um Estudo de Caso. **Revista Brasileira de Informática na Educação**, 24(3), 2016.

ALVES, N. C. **CodeMaster: um modelo de avaliação do pensamento computacional na educação básica através da análise de código de linguagem de programação visual**. Dissertação (Programa de Pós-Graduação em Ciência da Computação (PPGCC)) – Universidade Federal de Santa Catarina, 2019.

ALVES, N. da C., GRESSE VON WANGENHEIM, C., HAUCK, J. C. R. **Um Modelo de Avaliação do Pensamento Computacional na Educação Básica através da Análise de Código de Linguagem de Programação Visual**. Anais do Concurso Alexandre Direne de Teses, Dissertações e Trabalhos de Conclusão no VIII Congresso Brasileiro de Informática na Educação, Brasília/DF, 2019.

ALVES, Nathalia da Cruz; VON WANGENHEIM, Christiane Gresse; ALBERTO, Matheus; MARTINS-PACHECO, Lúcia Helena. Uma Proposta de Avaliação da Originalidade do Produto no Ensino de Algoritmos e Programação na Educação Básica. **Anais do Xxxi Simpósio Brasileiro de Informática na Educação (Sbie 2020)**, [S.L.], p. 41-50, 24 nov. 2020a. Sociedade Brasileira de Computação. http://dx.doi.org/10.5753/cbie.sbie.2020.41.

ALVES, N. da C., GRESSE VON WANGENHEIM, C., HAUCK, J. C. R. A Large-scale Analysis of App Inventor Projects. **XV Conferencia Latinoamericana de Tecnologias de Aprendizaje (LACLO)**, 2020b.

ALVES, N. da C., KRETZER, F., GRESSE VON WANGENHEIM, C., FERREIRA, M. N. F.; HAUCK, J. C. R. Formação Continuada de Professores da Educação Básica para o Ensino de Algoritmos e Programação. In: **Anais do Simpósio Brasileiro de Informática na Educação, Natal, Brasil**, 2020c.

ALVES, N. D. C.; GRESSE VON WANGENHEIM, C.; MARTINS-PACHECO, L. H. Assessing Product Creativity in Computing Education: A Systematic Mapping Study. **Informatics in Education**, v. 20, n. 1, p. 19-45, 2021a.

ALVES, Nathalia da Cruz; VON WANGENHEIM, Christiane Gresse; MARTINS-PACHECO, Lúcia Helena; BORGATTO, Adriano Ferreti. Existem concordância e confiabilidade na avaliação da criatividade de resultados tangíveis da aprendizagem de computação na Educação Básica? **Anais do I Simpósio Brasileiro de Educação em Computação (Educomp 2021)**, p. 12-22, 26 abr. 2021b. Sociedade Brasileira de Computação. http://dx.doi.org/10.5753/educomp.2021.14467.

AMERSHI, Saleema; BEGEL, Andrew; BIRD, Christian; DELINE, Robert; GALL, Harald; KAMAR, Ece; NAGAPPAN, Nachiappan; NUSHI, Besmira; ZIMMERMANN, Thomas.

Software Engineering for Machine Learning: a case study. **2019 Ieee/acm 41St International Conference On Software Engineering**: Software Engineering in Practice (ICSE-SEIP), [S.L.], p. 291-300, maio 2019. IEEE. http://dx.doi.org/10.1109/icse-seip.2019.00042.

ARAÚJO, G., DE MEDEIROS, S., BERGMANN, J. C. F., & VON WANGENHEIM, C. G. Práticas pedagógicas com o desenvolvimento de aplicativos para dispositivos móveis por estudantes da Educação Básica. TEXTURA-Revista de Educação e Letras. 2020, p.22-49.

BASU, Satabdi. Using Rubrics Integrating Design and Coding to Assess Middle School Students' Open-ended Block-based Programming Projects. **Proceedings Of The 50Th Acm Technical Symposium On Computer Science Education**, [S.L.], p. 1211-1217, 22 fev. 2019. ACM. http://dx.doi.org/10.1145/3287324.3287412.

BEATY, Roger; JOHNSON, Dan Richard. Automating Creativity Assessment with SemDis: an open platform for computing semantic distance. **Psyarxiv**, [S.L.], p. 1-29, 14 fev. 2020. Center for Open Science. http://dx.doi.org/10.31234/osf.io/nwvps.

BESEMER, Susan P.; TREFFINGER, Donald J.. Analysis of Creative Products: review and synthesis*. **The Journal Of Creative Behavior**, [S.L.], v. 15, n. 3, p. 158-178, set. 1981. Wiley. http://dx.doi.org/10.1002/j.2162-6057.1981.tb00287.x.

BIALIK, M. et al. **Avaliações em Evolução para a Educação do Século XXI**. Centro de Redesenho Curricular. [Sl]. 2016.

BRANCH, Robert Maribe. **Instructional Design**: the addie approach. New York: Springer, 2009.

BRASIL, Tic Kids Online. **[Microdados] TIC Kids Online Brasil - 2019 Pais e responsáveis**. 2019. Disponível em: https://cetic.br/pt/arquivos/kidsonline/2019/pais/. Acesso em: 10 jan. 2022.

BRENNAN, K.; BALCH, C.; CHUNG, M. Creative computing 3.0, 2019. Disponível em: http://scratched.gse.harvard.edu/guide/. Acesso em: 20 Abr. 2020.

CAVALLO, David; SENGER, Helena; GOMES, Alex Sandro; BITTENCOURT, Ig Ibert; SILVEIRA, Ismar Frango. Inovação e Criatividade na Educação Básica: Dos conceitos ao ecossistema. **Revista Brasileira de Informática na Educação**, v. 24, n. 2. 2016.

CAPES. **Portal de Periódicos CAPES/MEC**. Disponível em: http://www-periodicos-capes-gov-br.ezl.periodicos.capes.gov.br/index.php?. Acesso em: 02 mar. 2021.

CLEMENTS, Douglas H.. Teaching creativity with computers. **Educational Psychology Review**, [S.L.], v. 7, n. 2, p. 141-161, jun. 1995. Springer Science and Business Media LLC. http://dx.doi.org/10.1007/bf02212491.

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

COLAB. **Conheça o Colab**. Disponível em: https://colab.research.google.com/#scrollTo=5fCEDCU_qrC0. Acesso em: 06 ago. 2022.

CORDEIRO, Alexander Magno; OLIVEIRA, Glória Maria de; RENTERÍA, Juan Miguel; GUIMARÃES, Carlos Alberto. Revisão sistemática: uma revisão narrativa. **Revista do Colégio Brasileiro de Cirurgiões**, [S.L.], v. 34, n. 6, p. 428-431, dez. 2007. FapUNIFESP (SciELO). http://dx.doi.org/10.1590/s0100-69912007000600012.

CSTA. K-12 Computer Science Framework, 2016. Disponível em: <https://k12cs.org/>. Acesso em: 04 abr. 2022.

CSTA (2017). “Standards - CSTA K-12 Computer Science Standards”. Disponível em: https://www.csteachers.org/page/standards. Acesso em: 12 de abril, 2022.

DENNER, Jill; WERNER, Linda; ORTIZ, Eloy. Computer games created by middle school girls: can they be used to measure understanding of computer science concepts?. **Computers & Education**, [S.L.], v. 58, n. 1, p. 240-249, jan. 2012. Elsevier BV. http://dx.doi.org/10.1016/j.compedu.2011.08.006.

DEVICEATLAS. **Most used smartphone screen resolutions in 2019**. Disponível em: https://deviceatlas.com/blog/most-used-smartphone-screen-resolutions. Acesso em: 21 jan. 2021.

EDUCA, Ibge. **Uso de Internet, televisão e celular no Brasil**. Disponível em: https://educa.ibge.gov.br/criancas/brasil/2697-ie-ibge-educa/jovens/materias-especiais/20787uso-de-internet-televisao-e-celular-no-brasil.html. Acesso em: 17 jul. 2022.

FERREIRA, Miriam Nathalie Fortuna. **Um modelo para uma unidade de ensino sobre princípios do design visual de interfaces de app: um estudo de caso**. 2020. 106 f. Dissertação (Mestrado) - Curso de Design, Universidade Federal de Santa Catarina, Florianópolis, 2020.

FERREIRA, Miriam Nathalie Fortuna; PINHEIRO, Fernando da Cruz; VON WANGENHEIM, Christiane Gresse; MISSFELDT FILHO, Raul; HAUCK, Jean Carlo R.. Ensinando Design de Interface de Usuário de Aplicativos Móveis no Ensino Fundamental. **Revista Brasileira de Informática na Educação**, [S.L.], v. 28, p. 48-72, 16 fev. 2020. Sociedade Brasileira de Computacao - SB. http://dx.doi.org/10.5753/rbie.2020.28.0.48.

FINCHER, S. A. .; ROBINS, A. V. **The Cambridge handbook of computing education research**. Cambridge: University Press, 2019.

FUMO, David. **Types of Machine Learning Algorithms You Should Know**. 2017. Disponível em: https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861. Acesso em: 14 fev. 2021.

FUNKE, Alexandra; GELDREICH, Katharina; HUBWIESER, Peter. Analysis of scratch projects of an introductory programming course for primary school students. **2017 Ieee Global Engineering Education Conference (Educon)**, [S.L.], p. 1-8, abr. 2017. IEEE. http://dx.doi.org/10.1109/educon.2017.7943005.

GALAN, D.; HERADIO, R.; VARGAS, H.; ABAD, I.; CERRADA, J. A. Automated Assessment of Computer Programming Practices: The 8-Years UNED Experience. IEEE Access, 7(1), pp. 130113-130119, 2019.

GARRETT, Jesse James. **The Elements of User Experience**: user-centered design for the web and beyond. 2. ed. Thousand Oaks: New Riders Publishing, 2010.

GARRETT, Jesse James. **The Elements of User Experience**: user-centered design for the web and beyond. 2. ed. Berkeley: New Riders Publishing, 2011.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa**. 5. ed. São Paulo: Atlas, 2010.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. Cambridge: The Mit Press, 2016.

GOOGLE. **Roboto**. Disponível em: https://fonts.google.com/specimen/Roboto. Acesso em: 21 jan. 2021.

GOOGLE. **Material Design**. Disponível em: https://material.io/. Acesso em: 21 jan. 2021.

GRESSE VON WANGENHEIM, C. et al. CodeMaster – Automatic Assessment and Grading of App Inventor and Snap! Programs. **Informatics in Education**, v. 17, n. 1, p. 117-150, 2018.

GROVER, Shuchi; PEA, Roy. Computational Thinking in K–12. **Educational Researcher**, [S.L.], v. 42, n. 1, p. 38-43, jan. 2013. American Educational Research Association (AERA). http://dx.doi.org/10.3102/0013189x12463051.

HASTY. **Intersection over Union (IoU)**. Disponível em: https://hasty.ai/docs/mp-wiki/metrics/iou-intersection-over-union. Acesso em: 03 set. 2022.

HONDA, Hiroto. **Digging into Detectron 2 — part 1**. 2020. Disponível em: https://medium.com/@hirotoschwert/digging-into-detectron-2-47b2e794fabd. Acesso em: 17 jul. 2022.

HOWARD, Jeremy; GUGGER, Sylvain. **Deep Learning for Coders with Fastai and Pytorch**: ai applications without a phd. Sebastopol: O'Reilly Media, Incorporated, 2020.

HU, Rui; CHEN, Mingang; CAI, Lizhi; CHEN, Wenjie. Detection and Segmentation of Graphical Elements on GUIs for Mobile Apps Based on Deep Learning. **Lecture Notes Of The Institute For Computer Sciences, Social Informatics And Telecommunications Engineering**, [S.L.], p. 187-197, 2020. Springer International Publishing. http://dx.doi.org/10.1007/978-3-030-64214-3_13.

ICHINCO, Michelle; KELLEHER, Caitlin. Semi-automatic suggestion generation for young novice programmers in an open-ended context. **Proceedings Of The 17Th Acm Conference On Interaction Design And Children**, [S.L.], p. 405-412, 19 jun. 2018. ACM. http://dx.doi.org/10.1145/3202185.3202762.

INDIA, U. Difference between Machine Learning, Deep Learning and Artificial Intelligence, 2018. Disponível em: https://medium.com/@UdacityINDIA/differencebetween-machine-learning-deep-learning-and-artificial-intelligence-e9073d43a4c3. Acesso em: março 2022.

INEP. **Censo da Educação Superior 2020**. Disponível em: https://www.gov.br/inep/pt-br/areas-de-atuacao/pesquisas-estatisticas-e-indicadores/censo-da-educacao-superior. Acesso em: abril.2022.

INEP. **Resumo técnico censo escolar da base educação básica 2021**. Disponível em: https://download.inep.gov.br/publicacoes/institucionais/estatisticas_e_indicadores/resumo_tecnico_censo_escolar_2021.pdf. Acesso em: 17 jul. 2021.

JACKSON, Philip W.; MESSICK, Samuel. THE PERSON, THE PRODUCT, AND THE RESPONSE: conceptual problems in the assessment of creativity1. **Ets Research Bulletin Series**, [S.L.], v. 1964, n. 2, p. 0-27, dez. 1964. Wiley. http://dx.doi.org/10.1002/j.2333-8504.1964.tb00695.x.

JEONG, Jongwook; KIM, Neunghoe; IN, Hoh Peter. Detecting usability problems in mobile applications on the basis of dissimilarity in user behavior. **International Journal Of Human-Computer Studies**, [S.L.], v. 139, p. 102364, jul. 2020. Elsevier BV. http://dx.doi.org/10.1016/j.ijhcs.2019.10.001.

KREUCH, Leonardo. **Desenvolvimento de um Modelo de Avaliação da Originalidade do Esqueleto de Design de Interface de Aplicativos Android**. 2022. 83 f. TCC (Graduação) - Curso de Ciência da Computação, Universidade Federal de Santa Catarina. Florianópolis, Florianópolis, 2022.

LEARNING, Partnership For 21St Century (P21). **Our mission is to realize the power and promise of 21st century learning for every student—in early learning, in school, and beyond school—across the country and around the globe**. 2017. Disponível em: http://www.p21.org/news-events/p21blog/2128-computerscience-a-playground-for-21st-century-skills. Acesso em: 31 out. 2020.

LEE, Irene; MARTIN, Fred; DENNER, Jill; COULTER, Bob; ALLAN, Walter; ERICKSON, Jeri; MALYN-SMITH, Joyce; WERNER, Linda. Computational thinking for youth in practice. **Acm Inroads**, [S.L.], v. 2, n. 1, p. 32-37, 25 fev. 2011. Association for Computing Machinery (ACM). http://dx.doi.org/10.1145/1929887.1929902.

LI, Li; BISSYANDÉ, Tegawendé F.; WANG, Hao-Yu; KLEIN, Jacques. On Identifying and Explaining Similarities in Android Apps. **Journal Of Computer Science And Technology**, [S.L.], v. 34, n. 2, p. 437-455, mar. 2019. Springer Science and Business Media LLC. http://dx.doi.org/10.1007/s11390-019-1918-8.

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**, v. 41, p. 51–61, 2014.

LYTLE, N. et al. **Use, Modify, Create:** Comparing Computational Thinking Lesson Progressions for STEM Classes. Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '19). New York, NY, USA,: [s.n.]. 2019.

LYU, Fang; LIN, Yapin; YANG, Junfeng; ZHOU, Junhai. SUIDroid: an efficient hardening-resilient approach to android app clone detection. **2016 Ieee Trustcom/Bigdatase/Ispa**, [S.L.], p. 511-518, ago. 2016. IEEE. http://dx.doi.org/10.1109/trustcom.2016.0104.

MANNING, Christopher D.; RAGHAVAN, Prabhakar; SCHÜTZE, Hinrich. **Introduction to Information Retrieval**. Cambridge University Press, 2008.

MAO, J., et al. Robust Detection of Android UI Similarity. School of Electronic and Information Engineering, Beihang University, 2018.

MATHEWSON, Kory W.. A Human-Centered Approach to Interactive Machine Learning. **Arxiv.Org**, arXiv:1910.05528 [cs.LG], 2019.

MARTINS, Osvaldo Paulo Heiderscheidt Roberge. **Desenvolvimento de um Modelo para Avaliação da Estética Visual de Interfaces de Usuários de Aplicativos Usando Deep Learning**. 2019. 116 f. TCC (Graduação) - Curso de Ciência da Computação, Universidade Federal de Santa Catarina. Florianópolis, Florianópolis, 2019.

MARTINS, L. DA C. G. F.; GUISSO, L. F. **Avaliação: um desafio no processo de ensino-aprendizagem na educação - revisão de literatura**. Revista Eletrônica Acervo Saúde, 24, e379. 2019.

MEC. Base Nacional Comum Curricular. Ministério da Educação - Brasil. Brasília. 2018.

MEC. Base Nacional Comum Curricular. Ministério da Educação - Brasil. Brasília. 2022.

MEC. **Base Nacional Comum Curricular**. Disponível em: http://basenacionalcomum.mec.gov.br/images/BNCC_EI_EF_110518_versaofinal_site.pdf. Acesso em: 17 jul. 2022.

MORAES, J. B.; RODRIGUES, M. S. **Competências profissionais: um estudo comparativo entre os cursos superiores de tecnologia em processos gerenciais do Instituto Federal de São Paulo**. Latin American Journal of Business Management, 10(2), 2019.

MINISTÉRIO DA EDUCAÇÃO. **Base Nacional Comum Curricular**. Disponível em: http://basenacionalcomum.mec.gov.br/images/BNCC_EI_EF_110518_versaofinal_site.pdf. Acesso em: 26 mar. 2022.

MISLEVY, R. J.; ALMOND, R. G.; LUKAS, J. F. A brief introduction to evidence- centered design. **ETS Research Report Series**, v. 1, p. i-29, 2003.

MISLEVY, R. J.; HAERTEL, G. D. Implications of evidence-centered design for educational testing. **Educational measurement: issues and practice**, v. 25, n. 4, p. 6-20, 2006.

MIT. **MIT App Inventor**. Disponível em: http://appinventor.mit.edu/. Acesso em: 20 out. 2020.

MORENO-LEÓN, Jesús; ROBLES, Gregorio. Dr. Scratch. **Proceedings Of The Workshop In Primary And Secondary Computing Education**, [S.L.], p. 132-133, 9 nov. 2015. ACM. http://dx.doi.org/10.1145/2818314.2818338.

MUNKRES, J. Algorithms for the Assignment and Transportation Problems. **Journal of the Society for Industrial and Applied Mathematics**, v. 5, n. 1, p.32–38, mar. 1957.

MUSTAFARAJ, Eni; TURBA, Franklyn; SVANBERG, Maja. Identifying Original Projects in App Inventor. **Florida Artificial Intelligence Research Society Conference**, [s. l], p. 567-572, 2017.

MURTAZIN, Eldar. **Apple's Phone: From 1980s' Sketches to iPhone. Part 3**. 2010. Disponível em: https://mobile-review.com/articles/2010/iphone-history3-en.shtml. Acesso em: 19 mar. 2021.

MYCURVEFIT. **Goodness Measures**. Disponível em: https://mycurvefit.com/. Acesso em: 04 set. 2022.

NUMPY. Numpy. Disponível em: https://numpy.org/. Acesso em: 20 nov. 2021.

ORIGINLAB. **Linear and Polynomial Regression**. Disponível em: https://www.originlab.com/doc/Origin-Help/Linear-Polynomial-Regression. Acesso em: 07 set. 2022.

PANDAS. Pandas. Disponível em: https://pandas.pydata.org/. Acesso em: 20 nov. 2021.

PATHMIND. **A Beginner's Guide to Neural Networks and Deep Learning**. Disponível em: https://wiki.pathmind.com/neural-network. Acesso em: 14 fev. 2021.

PATTON, Evan W.; TISSENBAUM, Michael; HARUNANI, Farzeen. MIT App Inventor: objectives, design, and development. **Computational Thinking Education**, [S.L.], p. 31-49, 2019. Springer Singapore. http://dx.doi.org/10.1007/978-981-13-6528-7_3.

PETERSEN, Kai; VAKKALANKA, Sairam; KUZNIARZ, Ludwik. Guidelines for conducting systematic mapping studies in software engineering: an update. **Information And Software Technology**, [S.L.], v. 64, p. 1-18, ago. 2015. Elsevier BV. http://dx.doi.org/10.1016/j.infsof.2015.03.007.

PETERSEN, Kai; FELDT, Robert; MUJTABA, Shahid; MATTSSON, Michael. Systematic Mapping Studies in Software Engineering. **Systematic Mapping Studies In Software Engineering**, [s. l], p. 68-77, 2008.

PIASECKI, Jan; WALIGORA, Marcin; DRANSEIKA, Vilius. Google Search as an Additional Source in Systematic Reviews. **Science And Engineering Ethics**, [s. l], v. 24, p. 809-810, 2018.

POLAMURI, Saimadhu. **FIVE MOST POPULAR SIMILARITY MEASURES IMPLEMENTATION IN PYTHON**. 2015. Disponível em: https://dataaspirant.com/five-most-popular-similarity-measures-implementation-in-python/. Acesso em: 11 jun. 2021.

PPGCC. **Linhas de Pesquisa**. Disponível em: https://ppgcc.ufsc.br/linhas-de-pesquisa-2/. Acesso em: 20 nov. 2020.

RODE, J. A. et al. **From computational thinking to computational making**. Proc. of the Int. Joint Conference on Pervasive and Ubiquitous Computing. Osaka: ACM. 2015. p. 239-250.

ROYAL SOCIETY. **Machine learning**: the power and promise of computers that learn by example. the power and promise of computers that learn by example. 2017. Disponível em: https://royalsociety.org/~/media/policy/projects/machine-learning/publications/machine-learning-report.pdf. Acesso em: 14 fev. 2021.

RUMSEY, D.J. How to Interpret a Correlation Coefficient r. In: Dummies, a very brand [S.I.], 08 jul 2021. Disponível em: https://www.dummies.com/article/academics-thearts/math/statistics/how-to-interpret-a-correlation-coefficient-r-169792. Acesso em: 31 ago. 2022

RUNCO, Mark A.; JAEGER, Garrett J.. The Standard Definition of Creativity. **Creativity Research Journal**, [S.L.], v. 24, n. 1, p. 92-96, jan. 2012. Informa UK Limited. http://dx.doi.org/10.1080/10400419.2012.650092.

RUSSELL, Stuart. **Artificial Intelligence**: a modern approach. India: Pearson, 2015.

SALTON, G.; WONG, A.; YANG, C. S.. A vector space model for automatic indexing. **Communications Of The Acm**, [S.L.], v. 18, n. 11, p. 613-620, nov. 1975. Association for Computing Machinery (ACM). http://dx.doi.org/10.1145/361219.361220.

SAMUEL, A.L. Some Studies in Machine Learning Using the Game of Checkers. IBM Journal of Research and Development, 3(3), 210-229, 1959.

SAS. (2021) **Redes neurais: o que são e qual sua importância?** Disponível em: https://www.sas.com/pt_br/insights/analytics/neural-networks.html#technical. Acesso em: 10 nov. 2021.

SAUNDERS, Mark N.K.; LEWIS, Philip; THORNHILL, Adrian. **Research Methods for Business Students**. 8. ed. Edinburgh Gate: Pearson, 2019.

SCHLATTER, Tania; LEVINSON, Deborah. **Visual Usability**: principles and practices for designing digital applications. San Francisco: Morgan Kaufmann, 2013.

SCIKIT-LEARN. **Scikit-learn**. Disponível em: https://scikit-learn.org/stable/. Acesso em: 20 nov. 2021.

SEERATAN, K.; MISLEVY, R. **Design patterns for assessing internal knowledge representations**. SRI International. Menlo Park, Califórnia. 2008.

SHIRKHORSHIDI, Ali Seyed; AGHABOZORGI, Saeed; WAH, Teh Ying. A Comparison Study on Similarity and Dissimilarity Measures in Clustering Continuous Data. **Plos One**, [S.L.], v. 10, n. 12, p. 1-20, 11 dez. 2015. Public Library of Science (PLoS). http://dx.doi.org/10.1371/journal.pone.0144059.

SHUTE, V. J.; SUN, C.; ASBELL-CLARKE, J. Demystifying computational thinking. **Educational Research Review**, v. 22, p. 142-158, 2017.

Sociedade Brasileira de Computação, Diretrizes para ensino de Computação na Educação Básica (SBC). Disponível em: <https://www.sbc.org.br/educacao/diretrizes-para-ensino-de-computacao-naeducacao-basica>. Acesso em: 10 abr. 2022.

SOH, Charlie; TAN, Hee Beng Kuan; ARNATOVICH, Yauhen Leanidavich; WANG, Lipo. Detecting Clones in Android Applications through Analyzing User Interfaces. **2015 Ieee 23Rd International Conference On Program Comprehension**, [S.L.], p. 163-173, maio 2015. IEEE. http://dx.doi.org/10.1109/icpc.2015.25.

SOLECKI, Igor; PORTO, João; ALVES, Nathalia da Cruz; VON WANGENHEIM, Christiane Gresse; HAUCK, Jean; BORGATTO, Adriano Ferreti. Automated Assessment of the Visual Design of Android Apps Developed with App Inventor. **Proceedings Of The 51St Acm Technical Symposium On Computer Science Education**, [S.L.], p. 51-57, 25 fev. 2020. ACM. http://dx.doi.org/10.1145/3328778.3366868.

SOWMIYA, C.; SUMITRA, P.. Analytical study of heart disease diagnosis using classification techniques. **2017 Ieee International Conference On Intelligent Techniques In Control, Optimization And Signal Processing (Incos)**, [S.L.], p. 1-5, mar. 2017. IEEE. http://dx.doi.org/10.1109/itcosp.2017.8303115.

SPEARMAN, C.E. (1904) General Intelligence, Objectively Determined and Measured. **American Journal of Psychology**, 15, 201-292.

SVANBERG, M. Using feature vector representations to identify similar projects in app inventor. In: **Proc. Of IEEE Blocks and Beyond Workshop**, p. 117–18, 2017.

TIAN, Youhui. Artificial Intelligence Image Recognition Method Based on Convolutional Neural Network Algorithm. **Ieee Access**, [S.L.], v. 8, p. 125731-125744, 2020. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/access.2020.3006097.

TIC EDUCAÇÃO. **TIC Educação**. Cetic. São Paulo. 2019.

TYLER, R. W. **Basic Principles of Curriculum and Instruction**. Chicago: University of Chicago Press, 1949.

ULTRALYTICS. YOLOv5 Documentation, [s.d.]. Disponível em: https://docs.ultralytics.com/. Acesso em: 20 nov. 2021.

VALUEVA, M.V.; NAGORNOV, N.N.; LYAKHOV, P.A.; VALUEV, G.V.; CHERVYAKOV, N.I.. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. **Mathematics And Computers In Simulation**, [S.L.], v. 177, p. 232-243, nov. 2020. Elsevier BV. http://dx.doi.org/10.1016/j.matcom.2020.04.031.

W3C. **Web Content Accessibility Guidelines 2.1**. Disponível em: https://www.w3.org/TR/2018/REC-WCAG21-20180605/. Acesso em: 21 jan. 2021a.

W3C. **Understanding Success Criterion 1.3.4: Orientation**. Disponível em: https://www.w3.org/WAI/WCAG21/Understanding/orientation.html. Acesso em: 21 jan. 2021b.

WALIA, Chetan. A Dynamic Definition of Creativity. **Creativity Research Journal**, [S.L.], v. 31, n. 3, p. 237-247, 3 jul. 2019. Informa UK Limited. http://dx.doi.org/10.1080/10400419.2019.1641787.

WASSERMAN, Anthony I.. Software engineering issues for mobile application development. Proceedings Of The Fse/sdp **Workshop On Future Of Software Engineering Research - Foser '10**, [S.L.], p. 397-400, 2010. ACM Press. http://dx.doi.org/10.1145/1882362.1882443.

WEF. **The Future of Jobs Report 2020**. 2020. Disponível em: https://www.weforum.org/reports/the-future-of-jobs-report-2020. Acesso em: 06 ago. 2022.

WERNER, L.; DENNER, J.; CAMPE, S.; KAWAMOTO, D. C. The Fairy Performance Assessment: Measuring Computational Thinking in Middle School. **Proceedings of the 43rd ACM technical symposium on Computer Science Education**, Raleigh, North Carolina, USA, 2012. p. 215-220.

WU, Yuxin; KIRILLOV, Alexander; MASSA, Francisco; LO, Wan-Yen; GIRSHICK, Ross. **Detectron2**. 2019a. Disponível em: https://github.com/facebookresearch/detectron2. Acesso em: 17 jul. 2022.

WU, Yuxin; KIRILLOV, Alexander; MASSA, Francisco; LO, Wan-Yen; GIRSHICK, Ross. **Detectron2: A PyTorch-based modular object detection library**. 2019b. Disponível em: https://ai.facebook.com/blog/-detectron2-a-pytorch-based-modular-object-detection-library-/. Acesso em: 17 jul. 2022.

YADAV, Aman; COOPER, Steve. Fostering creativity through computing. **Communications Of The Acm**, [S.L.], v. 60, n. 2, p. 31-33, 23 jan. 2017. Association for Computing Machinery (ACM). http://dx.doi.org/10.1145/3029595.

ZEN, Mathieu; VANDERDONCKT, Jean. Assessing User Interface Aesthetics based on the Inter-subjectivity of Judgment. **International Bcs Human Computer Interaction Conference**, [S.L.], p. 1-12, 1 jul. 2016. BCS Learning & Development. http://dx.doi.org/10.14236/ewic/hci2016.25.

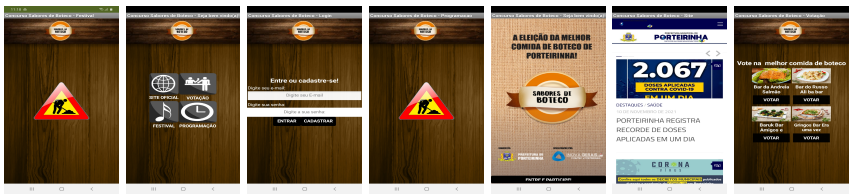
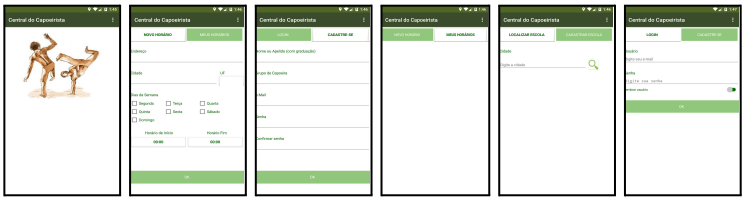
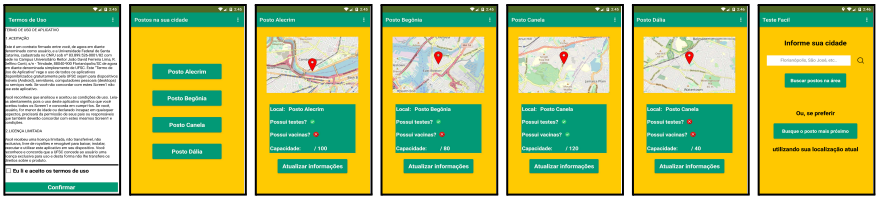
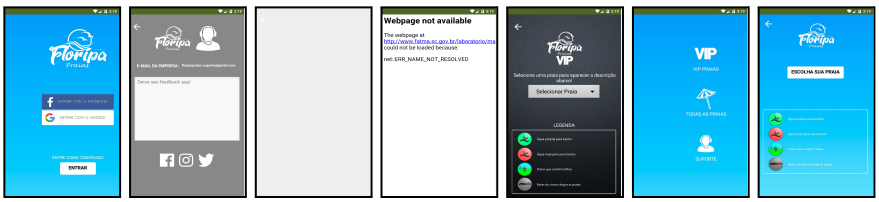
ZHANG, Tao; LIU, Ying; GAO, Jerry; GAO, Li Peng; CHENG, Jing. Deep Learning-Based Mobile Application Isomorphic GUI Identification for Automated Robotic Testing. **Ieee**

Software, [S.L.], v. 37, n. 4, p. 67-74, jul. 2020. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/ms.2020.2987044.

ZHOU, Jing; WANG, Xiaoye May; SONG, Lynda Jiwen; WU, Junfeng. Is it new? Personal and contextual influences on perceptions of novelty and creativity. **Journal Of Applied Psychology**, [S.L.], v. 102, n. 2, p. 180-202, fev. 2017. American Psychological Association (APA). http://dx.doi.org/10.1037/apl0000166.

ZOU, Z. et al. Object Detection in 20 Years: A Survey, **arXiv:1905.05055v2** [cs.CV]. 2019.

APÊNDICE A - APPS DE CONSULTA

COMPONENTE DE UI-COMBINAÇÃO	COMPONENTE DE UI- FREQUÊNCIA	ID/NOMES/SCREENSHOTS
Button EmailPicker DatePicker TimePicker Image	6 1 1 1 1	**6105807452241920.aia/Banco_de_Horas** 
Button Image BackgroundImage Label Notifier PasswordTextBox WebViewer EmailPicker	11 6 7 12 3 1 1 1	**5054792942223360.aia/SaboresdeBoteco** 
Button CheckBox Image Label ListPicker ListView PasswordTextBox Switch TextBox TimePicker	14 7 1 50 1 2 3 1 7 2	**CentralCapoeirista.aia/CentralCapoeirista** 
Button CheckBox Image Label Notifier TextBox Map	11 1 9 80 6 1 4	**TesteFacil.aia/TesteFacil** 
Button Image BackgroundImage Label ListPicker Spinner TextBox WebViewer	14 11 4 46 1 1 1 1	**FloripaPraias.aia/FloripaPraias** 

COMPONENTE DE UI-COMBINAÇÃO	COMPONENTE DE UI- FREQUÊNCIA	ID/NOMES/SCREENSHOTS
WebViewer	1	**4663417046695936.aia/Feitocomamor** 
Button Label TextBox BackgroundImage	1 4 2 1	**5420291046768640.aia/Medidordepeso** 
Image TextBox Label Button	1 1 3 2	**5992873907191808.aia/PergunteMe** 
Button Textbox Label	1 2 7	**4990570013130752.aia/IMC** 
Button	1	**sos.aia/SOS** 

APÊNDICE B - FREQUÊNCIA DE COMBINAÇÕES DE COMPONENTES DE UIs NO UNIVERSO DE REFERÊNCIA

Combinação	Nº de Frequência	Combinação	Nº de Frequência
Button Label TextBox	17	Button Image Label WebViewer	1
Button Image Label TextBox	12	Button Image Label ListView Notifier Map	1
Button TextBox	8	Button DatePicker Image Label ListPicker ListView TextBox TimePicker	1
Button BackgroundImage Label TextBox	7	Button Image BackgroundImage Label Notifier PasswordTextBox TextBox ImagePicker	1
Button Image Label ListPicker Notifier	4	Button Image BackgroundImage Label ListView Notifier PasswordTextBox TextBox Map	1
Button Label Notifier Spinner TextBox	4	Button Label Notifier Map	1
Button CheckBox Image Label ListView Notifier TextBox	4	Button Image Label Notifier WebViewer ImagePicker	1
Button Image BackgroundImage Label Notifier TextBox	3	Button Label ListPicker ListView TextBox	1
Button Image Label Notifier TextBox	3	Button Image Label Notifier PasswordTextBox TextBox Map	1
Button Image Label	3	Button Image Label WebViewer ImagePicker	1
Button Label Notifier TextBox	3	Notifier	1
WebViewer	3	Button CheckBox DatePicker Image Label ListView Notifier TextBox TimePicker	1
Button Image Label TextBox WebViewer	3	Button CheckBox Label ListView Notifier Spinner TextBox	1
Button Image BackgroundImage Label WebViewer	3	Button Image Label ListView Notifier Spinner TextBox	1
Button Image BackgroundImage	3	Button DatePicker Image BackgroundImage Label Spinner TextBox TimePicker	1
Button Image Label Notifier	3	Button CheckBox DatePicker Label Notifier TextBox	1
Button Image Label ListView TextBox	3	Button CheckBox Image Label ListView	1
Button BackgroundImage	3	Button Label Notifier TextBox Map	1
Button Image	3	Button DatePicker Image Label Notifier PasswordTextBox TextBox TimePicker	1
Button Label ListView TextBox	2	Button DatePicker Label Notifier Spinner TextBox	1
Button Image Label ListView	2	Button Image BackgroundImage Slider ContactPicker PhoneNumberPicker	1
Button Image Label ListPicker Notifier TextBox	2	Button CheckBox Label ListPicker Notifier TextBox PhoneNumberPicker	1
Button BackgroundImage TextBox	2	Button Label Notifier TextBox PhoneNumberPicker	1
Button Label ListPicker	2	Button Image Label TextBox PhoneNumberPicker	1
Button CheckBox Label Notifier Spinner TextBox	2	Button Image BackgroundImage Label	1
Button Label TextBox WebViewer	2	Button Image BackgroundImage Label TextBox	1
Button TextBox WebViewer	2	Button DatePicker Image BackgroundImage Label ListPicker Notifier TextBox TimePicker	1
Button Label ListPicker Notifier	2	Button BackgroundImage Label	1
Button Label	2	Image TextBox WebViewer PhoneNumberPicker	1
Button WebViewer	2	Button BackgroundImage Label Notifier TextBox WebViewer	1
Image Label ListPicker WebViewer	2	Button DatePicker Label Notifier TextBox	1
Button ListPicker Notifier	2	ImagePicker	1
Button Image TextBox	2	BackgroundImage Label ListPicker Notifier Slider TextBox	1
Button CheckBox Label Notifier TextBox	1	Button DatePicker Image Label ListPicker ListView Notifier TextBox	1
Button Image BackgroundImage Label ListPicker Notifier	1	Button CheckBox Label ListView Notifier PasswordTextBox	1

Combinação	Nº de Frequência	Combinação	Nº de Frequência
TextBox ContactPicker		TextBox	
Button CheckBox Image Label ListView Notifier Spinner Switch TextBox TimePicker	1	BackgroundImage Notifier WebViewer	1
Button CheckBox Image Label Spinner WebViewer	1	Button Image Label Notifier TextBox WebViewer	1
Button Image BackgroundImage Label ListPicker ListView Notifier PasswordTextBox TextBox	1	Label ListView	1
Button DatePicker BackgroundImage Label TextBox	1	Label ListPicker Slider	1
Button BackgroundImage VideoPlayer	1	Button Image Label Notifier PasswordTextBox TextBox	1
Button BackgroundImage Label Slider	1	Button Label Spinner TextBox	1
Button Image PasswordTextBox Switch TextBox	1	Button BackgroundImage Label ListView TextBox ImagePicker	1
Button Image Label Notifier TextBox WebViewer ImagePicker	1	Button CheckBox Image BackgroundImage ListView Notifier TextBox WebViewer	1
Button ListView Notifier ImagePicker VideoPlayer ContactPicker	1	Button DatePicker Image BackgroundImage Label Slider	1
Image Label ListPicker	1	Button Image BackgroundImage Label Notifier TextBox TimePicker	1
Button CheckBox Image BackgroundImage Label Notifier TextBox TimePicker Map	1	Button BackgroundImage Label Notifier PasswordTextBox TextBox	1
Button Image PhoneNumberPicker	1	Image BackgroundImage WebViewer	1
Button Image ListPicker Notifier	1	Button BackgroundImage PasswordTextBox TextBox	1
Button CheckBox BackgroundImage Label ListPicker Notifier Slider TextBox	1	Button DatePicker BackgroundImage Label ListPicker Notifier Spinner TextBox	1
Button Label ListPicker Notifier PasswordTextBox TextBox	1	Button BackgroundImage Label ListPicker ListView Notifier TextBox	1
Button Label Notifier PasswordTextBox TextBox WebViewer	1	Button BackgroundImage ListPicker Notifier PasswordTextBox TextBox	1
Button Image BackgroundImage Label Notifier PasswordTextBox TextBox	1	Button DatePicker Image	1
Button Label Notifier	1	Button Image BackgroundImage Label PasswordTextBox TextBox	1
Button CheckBox Label TextBox	1	Button CheckBox Image BackgroundImage Label Notifier TextBox	1
Button Label ListView Notifier TextBox	1	Button Image BackgroundImage Label Notifier Slider TextBox WebViewer	1
Button CheckBox Image Label ListView Notifier	1	Button Image BackgroundImage Label ListPicker PasswordTextBox TextBox	1
Button CheckBox Image Label Notifier Spinner	1	Button Image Notifier Slider ImagePicker	1
Button Image Label ListView Notifier Spinner TextBox ImagePicker	1	Button Label Notifier TextBox WebViewer	1
Image Label ListPicker ListView	1	Button Image ListPicker ListView Notifier ContactPicker PhoneNumberPicker	1
Button ListView Spinner	1	Button DatePicker Image BackgroundImage Label Notifier TextBox TimePicker	1