Análise da Relação entre Fundamentos de Programação e Conceitos do Pensamento Computacional em Portfólios de Estudantes de um Curso EaD de Programação

Elias Dias de Araújo[1], **Deglaucy Jorge Teixeira** [2], **Maria Viviane de Menezes** [1]

[1]Universidade Federal do Ceará – Campus Quixadá
Av. José de Freitas Queiroz, 5003 – Cedro, Quixadá – CE – Brasil

[2]Fundação Demócrito Rocha
Av. Aguanambi, 282A - Joaquim Távora, Fortaleza – CE – Brasil

eliasdiasdearaujo@alu.ufc.br, deglaucy@fdr.org.br, vivianemenezes@ufc.br

Abstract. *The teaching of programming is fundamental for the development of abstraction and problem-solving skills in high school students and is related to the development of computational thinking. To facilitate the learning of programming concepts, block-based programming language tools have been developed in recent years, such as App Inventor. However, the scarcity of specialized teachers in the area poses a challenge to the widespread teaching of this type of content in conventional basic education. Distance education (EaD) is one of the alternatives to overcome this problem. This article presents an evaluation carried out with high school students from public schools in the state of Ceará, who participated in a distance programming course called "Fábrica de Programadores - Aprendendo a programar com Games". In this course, the App Inventor platform was used for the development of practical programming projects, and the evaluation proposed in this article was conducted through a rubric that relates programming fundamentals to levels of development in computational thinking by analyzing the portfolios of practical exercises submitted by the students participating in the course.In the analyzed module, 200 submissions were made, of which 123 were deemed suitable for analysis. Among these, 5 contained errors, highlighting the challenges faced by the students who submitted them, while 99 submissions, equivalent to* $80,48\%$ *of the submissions eligible for analysis, achieved the highest level in the computational thinking concepts addressed in the module. This demonstrates that using App Inventor as a programming learning tool for students in a distance learning format was effective.*

Resumo. *O ensino de programação é fundamental para o desenvolvimento da capacidade de abstração e resolução de problemas em alunos do ensino médio e está relacionado ao desenvolvimento do pensamento computacional. Para facilitar o aprendizado de conceitos de programação, ferramentas de linguagens de programação em blocos foram desenvolvidas nos últimos anos, tais como o App Inventor. Contudo, a escassez de professores especializados na área constitui um desafio para a popularização do ensino deste tipo de conteúdo do ensino básico na modalidade convencional, sendo a educação a distância (EaD) uma das alternativas para superar este problema. Esse artigo apresenta uma avaliação realizada com os estudantes de escolas públicas do ensino*

médio do estado do Ceará, participantes do curso de programação à distância denominado "Fábrica de Programadores - Aprendendo a programar com Games". Neste curso foi adotada a plataforma App Inventor para desenvolvimento dos projetos práticos de programação, sendo a avaliação proposta neste artigo realizada por meio de uma rubrica que relaciona os fundamentos de programação com níveis de desenvolvimento do pensamento computacional por meio da análise dos portfólios de exercícios práticos submetidos pelos alunos participantes do curso. No módulo analisado, 200 submissões foram realizadas e destas, 123 foram consideradas aptas para análise e dentre elas, existiram 5 com erros, evidenciando as dificuldades encontradas pelos estudantes que as enviaram e 99 submissões, equivalente a $80,48\%$ *das submissões aptas para análise, atingiram nível máximo nos conceitos do pensamento computacional abordados no módulo, constatando que o uso do App Inventor como ferramenta de aprendizado de programação pelos estudantes em um formato EAD foi eficaz.*

1. Introdução

O ensino de conceitos de programação está relacionado ao desenvolvimento da habilidade de abstração e à capacidade de resolver problemas por parte dos estudantes [Scherer et al. 2019]. Nesse contexto, a Base Nacional Comum Curricular [MEC 2017] e o seu complemento voltado a computação na educação básica definem as competências a serem cultivadas pelos alunos do ensino médio incluindo dentre elas o desenvolvimento do **Pensamento Computacional** (PC) que engloba a capacidade de compreender, analisar, definir, modelar, resolver, comparar e automatizar problemas e suas soluções de maneira metódica e sistemática [MEC 2022], seja na área da computação ou fora dela [Wing 2006].

O entendimento dos conceitos de pensamento computacional é essencial para que tais estudantes não se limitem apenas a serem usuários da tecnologia, mas sejam capazes de utilizá-la para resolver problemas de suas próprias vivências [Monroy-Hernández and Resnick 2008]. Assim sendo, interfaces de ensino de programação amigáveis a usuários iniciantes foram desenvolvidas ao longo dos anos. Dentre elas, destacam-se: *Scratch* [MIT 2023], *Code.org* [Code.org 2023] e *App Inventor* [MIT 2022a]. Essas ferramentas tem em comum que a programação é desenvolvida por meio de **linguagens de programação baseada em blocos** para que os conceitos fundamentais de programação possam ser aprendidos de forma lúdica. O *Scracth* tem o foco em desenvolvimento de jogos e personagens, o *Code.org* é um site com um conjunto de desafios para serem solucionados pelos alunos e o *App Inventor* objetiva o desenvolvimento de aplicativos móveis [MIT 2022a]. A programação em blocos utiliza elementos gráficos na criação dos programas, o que a distingue das formas de programação tradicionais com texto [Tsai 2019]. Nela, os códigos em textos são substituídos por blocos e as formas, cores e nomes desses blocos simplifica para os programadores identificarem quais são suas respectivas funcionalidades e se eles podem ou não encaixar-se em outros para formar diferentes sequências. Dessa maneira, a programação em blocos converge para a diminuição de erros sintáticos, direcionando a atenção do programador para a lógica do desenvolvimento da solução.

No entanto, um dos principais entraves da popularização de conceitos de pen-

samento computacional na educação básica no Brasil é a formação de professores para esse fim [Oliveira and Cambraia 2020]. Assim, mesmo dispondo de ferramentas adequadas para o ensino de programação, a falta de professores especializados pode levar a uma abordagem menos eficaz no ensino desse conteúdo [Bordini et al. 2016]. Nesse contexto, a Educação a Distância (EaD) pode ser vista como uma alternativa para suprir a falta de professores especializados em computação no ensino básico e convencional [Kaminski and Boscarioli 2019, dos Santos and da Silva 2020].

A Fundação Demócrito Rocha é uma fundação com experiência na promoção de cursos de educação à distância nas mais diversas áreas e promoveu, em parceria com a Universidade Federal do Ceará, o curso denominado "*Fábrica de Programadores - Aprendendo a programar com Games*". Esse foi o primeiro curso promovido pela fundação na área de ensino de programação e sua proposta é ensinar os fundamentos de programação para alunos de escolas públicas do ensino médio do estado do Ceará por meio do uso da ferramenta *App Inventor* [fdr 2023]. O curso contém 12 módulos com conteúdos sobre fundamentos de programação, conceitos de jogos, ideação e apresentação de produto tecnológico.

Nos módulos práticos de programação, as atividades submetidas pelos alunos foram corrigidas manualmente por tutores, alunos de graduação de cursos da área de Tecnologia da Informação da UFC, e estes, supervisionados por professores universitários da mesma área. O curso foi avaliado por meio de formulário preenchido pelos alunos que concluíram a primeira turma. No entanto, **carece de uma avaliação mais estruturada no que diz respeito ao impacto das atividades práticas do curso no aprendizado dos fundamentos de programação ao relacioná-los com o desenvolvimento do pensamento computacional dos alunos participantes**.

Na literatura, há várias maneiras de avaliar o aprendizado de programação pelo desenvolvimento do PC. Trabalhos como os de [Tang et al. 2020], [Moreno-León et al. 2015], [von Wangenheim et al. 2018] elencam algumas dessas maneiras. Segundo [Tang et al. 2020] metodologias para medição de proficiência em PC consistem em coletas e análises de dados a partir de questões de múltiplas escolhas, entrevistas, pesquisas ou análises de portfólios. Essa última, está atrelada a criação de rubricas de associação entre as características de programação e uma relação direta destas com os conceitos do PC. Além disso, segundo o *Report of a Workshop on the Pedagocial Aspects of Computatcional Thinking* [Council et al. 2011], três são as razões para as quais devem ser pautados os objetivos de avaliar o Pensamento Computacional: (i) **julgar o currículo e material relacionado e pedagogia de ensino do pensamento computacional em determinada situação**; (ii) **julgar o progresso individual da pessoa que está aprendendo** e (iii) **gerenciar o treinamento e suporte do instrutor que está lecionando a respeito do PC**.

Assim, esse artigo tem o objetivo de apresentar uma análise, utilizando uma rubrica de avaliação que explora a relação entre os fundamentos de programação e os conceitos do pensamento computacional, a fim de avaliar: o nível de desenvolvimento dos conceitos de pensamento computacional exigidos nos exercícios de programação do curso e; a aquisição de conhecimentos em programação dos estudantes por meio da avaliação de seus portfólios, isto é, das respostas dos exercícios práticos submetidos pelos alunos.

O artigo está organizado como descrito a seguir: no Seção 2 será abordada a fundamentação teórica para o entendimento deste trabalho; na Seção 3 são apresentados os trabalhos relacionados ao tema; na Seção 4 são descritos os procedimentos metodológicos; na Seção 5 serão apresentados os resultados do trabalho e; na Seção 6 serão apresentadas as conclusões e as possibilidades de trabalhos futuros.

2. Fundamentação Teórica

Esta seção aborda a fundamentação teórica do trabalho. A subseção 2.1 aborda os conceitos sobre o *App Inventor*. A subseção 2.2 descreve o curso "*Fábrica de Programadores - Aprendendo a programar com Games*". E, finalmente, a subseção 2.3 apresenta as principais metodologias para análise do desenvolvimento do PC.

2.1. O App Inventor

O *App Inventor* é uma plataforma online, utilizada para o desenvolvimento de aplicações móveis, em que pode-se criar programas usando uma linguagem de programação em blocos [Patton et al. 2019]. Essa plataforma subdivide-se em duas seções principais, o painel de *Designer* e o painel de Blocos [Nascimento 2023]. Na seção do painel de *Designer* é possível configurar a interface de componentes de usuários, botões, legendas e outros. Esse painel também comporta componentes não visuais como sensores, componentes de mídia capazes de acessar funcionalidade do *smartphone*. Ademais, cada uma dessas categorias, possuí seus eventos, métodos e propriedades únicas para cada tipo de componente [da Cruz Alves and von Wangenheim 2023]. Já na seção do painel de Blocos é possível programar as funcionalidades do aplicativo. Os blocos representam conceitos de programação como estruturas de repetição, procedimentos, condicionais, funções, eventos e ações para um componente em particular da aplicação [da Cruz Alves and von Wangenheim 2023].

Os fundamentos de programação consistem na noção mais básica do que abrange programação, eles envolvem conceitos de algoritmos, soluções de problemas por meio de computadores [UFC 2022], ou seja, busca elencar os conceitos básicos e indispensáveis de programação e também concepção e desenvolvimento de algoritmos [USP 2017]. Os fundamentos de programação existentes no *App Inventor* são: variáveis, expressões, operadores controles de fluxo, entrada e saída de dados, cadeias de caracteres, procedimentos e listas.

Os projetos do App Inventor são automaticamente salvos no ambiente online. Porém, é possível exportá-los para usar em outros contextos e ambientes. Ao realizar esse processo, é gerado um arquivo *.aia* [MIT 2022b], ele por sua vez contém os arquivos de propriedades do projeto: os arquivos *.scm* contém uma estrutura em *JSON* (do inglês *JavaScript Object Notation*) responsável por conter os elementes visuais do painel do *Designer* e; os arquivos *.bky* possuem uma estrutura em *XML* (do inglês *Extensible Markup Language*) que inclui os blocos de programação usados para definir o comportamento do aplicativo [da Cruz Alves and von Wangenheim 2023].

2.2. Sobre o Curso

O "*Fábrica de Programadores - Aprendendo a programar com Games*" é um curso de introdução à programação, desenvolvido a partir de uma parceria entre a Fundação

Demócrito Rocha e a UFC. Ele é um curso ofertado para estudantes de escolas públicas de ensino médio, que busca fomentar conhecimentos sobre o desenvolvimento de aplicativos móveis. Neste curso, é utilizada a ferramenta *App Inventor* como linguagem visual para o aprendizado de programação. Os conteúdos do curso abrangem quatro categorias distintas: Ideação; Programação; Conceitos de Jogos e Apresentação de Produto Tecnológico. Essas categorias estão subdivididas e distribuídas em 12 módulos, e cada um desses módulos são compostos por ferramentas pedagógicas que incluem material teórico, videoaula teórica e videoaula de resolução de exercícios de programação.

A primeira edição do curso ocorreu em 2022 e a segunda edição ocorreu no ano de 2023. O conteúdo do curso foi desenvolvido por professores da UFC e o acompanhamento metodológico foi feito por uma equipe de tutores composta por estudantes dos cursos de graduação da área de Tecnologia da Informação que também são da UFC, supervisionados por tutores que também são professores da área da TI. Os módulos do curso são listados a seguir: Módulo 1 - Introdução a Programação; Módulo 2 - Conceitos de Jogos; Módulo 3 - Programação: O App Inventor; Módulo 4 - Ideação: Concebendo meu Jogo; Módulo 5 - Programação: Manipuladores de Eventos; Módulo 6 - Ideação: Projetando e Planejando meu Jogo; Módulo 7 - Ideação: Apresentação de Produto Tecnológico; Módulo 8 - Programação: Dados e Funções; Módulo 9 - Programação: Variáveis; Módulo 10 - Programação: Condicionais; Módulo 11 - Programação: Loops e Módulo 12 - Programação: Listas.

2.3. Metodologias para Análise da Proficiência em Pensamento Computacional

Segundo [Tang et al. 2020], os meios mais utilizados para validação do aprendizado do pensamento computacional, consistem em testes tradicionais que abrangem uma avaliação composta por questões de múltipla escolha e abertas; pesquisas; entrevistas e **análise de portfólio**. O trabalho conclui que a primeira e última forma são as mais frequentemente utilizadas.

Desse modo, uma forma de analisar esses projetos é criar uma rubrica de correspondência. O trabalho de [von Wangenheim et al. 2018] apresenta uma plataforma online, denominada *CodeMaster*, para analisar automaticamente projetos, que incluem os construídos com o *App Inventor*, definindo uma rubrica que trata da relação entre os critérios associados ao PC e o nível de performance, considerando os fundamentos do App Inventor. O trabalho de [Moreno-León et al. 2015] apresenta uma plataforma online semelhante, chamada de *Dr. Scratch*, que permite indivíduos analisarem automaticamente projetos concebidos utilizando a ferramenta *Scratch* atravéz também de uma rubrica que relaciona conceitos do PC as funcionalidades da aplicação que tem como base fundamentos de programação. A Tabela 1 exibe uma mostra da rubrica de avaliação de [von Wangenheim et al. 2018] para os seguintes critérios: Telas e decomposição de problema e Interface de usuário. Além desses, há também a análise dos critérios: Nomeando: Componentes; Variáveis; Procedimentos; Eventos; Abstração procedural; *Loops*; Condicionais; Operadores; Listas; Persistência de dados; Sensores; Mídia; Social; Conectividade; Desenho e animação. É importante ressaltar que essa rubrica é definida também com base nos conceitos do PC para dispositivos móveis elaboradas nos trabalho de [Sherman et al. 2014] e [Sherman and Martin 2015].

Tabela 1. Parte da Rubrica do App Inventor.

Critério	Nível 0	Nível 1	Nível 2	Nível 3
Telas e decomposição de problema	Tela única com componentes visuais estáticos	Tela única com componentes visuais que mudam de estado	Duas telas com componentes visuais e uma tela com componentes que mudam de estado	Duas telas ou mais telas com componentes visuais e duas ou mais telas com componentes que mudam de estado
Interface de usuário	Usa um componente visual sem arranjo	Usa dois ou mais componentes visuais sem arranjo	Usa cinco ou mais componentes visuais com um do tipo arranjo	Usa cinco ou mais componentes visuais com dois ou mais do tipo arranjo
Eventos	Nenhum uso de componente de eventos	Usa um componente de eventos	Usa dois componentes de eventos	Usa mais que dois componentes de eventos
Sensores	Nenhum uso de sensores	Usa um componente do tipo sensor	Usa dois componentes do tipo sensor	Usa mais de dois componentes do tipo sensor

3. Trabalhos Relacionados

Nesta seção são apresentados três trabalhos relacionados a esse artigo. O trabalho de [Park and Shin 2019] tem o objetivo de comparar o *Scratch* e o *App Inventor* a partir de uma rubrica para avaliar ambas as ferramentas no que está relacionado ao pensamento computacional. Os tipos de blocos de cada projeto foram colocados em análise com base na rubrica definida. A rubrica criada atribui três pontos para cada conceito do pensamento computacional. Os critérios para atribuição dos pontos são específicos para cada uma das duas ferramentas. Ao todo foram escolhidos 524 projetos do *Scratch* e 379 projetos do App Inventor. Os conceitos que são considerados na rubrica são: abstração e decomposição de problemas; Paralelismo; Pensamento lógico; Sincronização; Controle de fluxo; Interatividade do usuário e Representação de dados. Por fim, foi concluído a partir dos resultados que os projetos em Scratch tiveram pontuações maiores em paralelismo, sincronização e controle de fluxo, enquanto o App Inventor tiveram pontuações maiores em interatividade do usuário e representação de dados.

O artigo de [da Cruz Alves et al. 2020] tem como objetivo entender como os projetos desenvolvidos com o *App Inventor* são programados. Para isso, é realizada uma análise de 88606 projetos que estão na galeria oficial da plataforma *MIT App Inventor Gallery*. O tamanho do projeto foi medido a partir de uma contagem do número de blocos existentes nele, inferindo conceito ou abstração da programação que ele está relacionado. Foi constatado que existem projetos com mais de sessenta mil blocos e foi encontrado uma média de 162.5 blocos por projeto. Também foram analisados todos os blocos que estavam associados diretamente a algum tipo de componente. Portanto, foi determinado que existe um uso expressivo de blocos relacionados a eventos de componentes no *App Inventor*, blocos relacionados a outros conceitos computacionais como sequências e con-

dicionais não foram usados com alta frequência e conceitos que incluem mídia, sensores, redes sociais e conectividade são menos usados devido à sua necessidade apenas em tipos específicos de aplicativos.

O trabalho de [Sousa et al. 2020] propõe a realização de um levantamento sobre o uso da programação em blocos no ensino do PC. A pesquisa trata-se de um mapeamento sistemático da literatura que busca entender os diversos cenários em que a programação em blocos está sendo usada para auxiliar a aprendizagem do PC. Os autores concluem que o *Scratch* apareceu em 30% dos estudos e o *App Inventor* ficou em segundo lugar, correspondendo a 10% dos trabalhos. Outra constatação foi que 20% dos artigos não informaram nenhuma ferramenta de avaliação. Apenas 3,4% dos estudos foram feitos no Brasil, indicando que é uma área com forte potencial de estudo no país.

4. Metodologia

A metodologia de pesquisa deste trabalho consiste em efetuar uma análise do ensino de programação utilizando o ambiente do App Inventor em um cenário de ensino a distância, utilizando a rubrica de avaliação proposta por [von Wangenheim et al. 2018]. O cenário em questão é o curso "*Fábrica de Programadores - Aprendendo a programar com Games*" que tem como público-alvo estudantes do ensino médio de escolas públicas do estado do Ceará. Devido a restrições de espaço, optou-se em realizar a análise dos exercícios do "Módulo 03" do curso por se tratar do primeiro módulo de programação do curso. O objetivo deste módulo é apresentar os ambientes da ferramenta (*Designer* e Blocos) e, também, os primeiros fundamentos de programação tais como entrada e saída de dados. O módulo contém três exercícios de programação que são baseados no aplicativo "*Talk-ToMe*", para usuários iniciantes na utilização do *App Inventor* [MIT 2022a], os exercícios apresentam componentes de interação com usuários, a saber:

- Exercício 01. Propõe que o estudante crie um aplicativo que leia o texto "Olá! Este é meu primeiro aplicativo!", quando o usuário clicar em um botão.
- Exercício 02. Propõe que o estudante crie um aplicativo que leia o texto "Olá! Este é meu primeiro aplicativo" quando o usuário clicar em um botão e também leia o texto "Pare de me sacudir!" quando o usuário sacudir o celular.
- Exercício 03. Propõe que o estudante crie um aplicativo que leia um texto qualquer, digitado pelo usuário, quando o usuário clicar em um botão e leia o texto "Pare de me sacudir!" quando o usuário sacudir o celular.

A avaliação proposta tem como objetivos: (i) classificar os modelos de resposta dos exercícios de programação deste módulo, conforme os critérios e níveis de conceitos do pensamento computacional definidos na rubrica de [von Wangenheim et al. 2018] e; (ii) avaliar os portfólios dos alunos, classificando se estes conseguiram atingir os níveis de desenvolvimento do pensamento computacional esperado para cada exercício. Os modelos de resposta dos exercícios são listados a seguir:

- Exercício 01 - Modelo de Resposta (I) - 1 bloco de evento de componente, 1 bloco de chamada de procedimento associado a 1 componente do tipo mídia e 1 bloco de texto.
- Exercício 02 - Modelo de Resposta (II) - 2 blocos de evento de componente em que 1 deles está associado a 1 componente do tipo sensor, 2 blocos de chamada de procedimento associados a 1 componente de mídia e 2 blocos de texto.

- Exercício 03 - Modelo de Resposta (III) - 2 blocos de evento de componente, 2 blocos de chamada de procedimento associados a 1 componente do tipo mídia, 1 bloco de texto e 1 bloco de modificador de propriedade de Componente associado a 1 componente de caixa de texto que muda de estado.

A aplicação da rubrica de avaliação nos modelos de respostas é um processo que consiste em relacionar cada um dos conceitos da rubrica aos níveis do pensamento computacional nela definidos. Um passo mais elaborado é a avaliação dos portfólios dos alunos, uma vez que estamos trabalhando em um universo de muitas submissões e precisamos automatizar esse processo. Assim, será realizada uma contagem automática da quantidade e tipos de blocos submetidos nos exercícios (analisando o arquivo `.xml` correspondente a cada submissão). Para realização dessa tarefa foram desenvolvidos alguns *scripts* em *Python* e outro em *Typescript* que juntos tornam possível o processo de contagem dos blocos[1]. Os *scripts* em *Python* são:

- gerar_submissoes_aceitas.py. Esse programa é responsável por verificar todas as submissões de um determinado módulo e categorizá-las de acordo com a presença ou ausência do arquivo '*.aia*'. As submissões que não contém um arquivo nesse formato são consideradas inválidas.
- gerar_bky.py. Esse programa percorre o conjunto de submissões válidas e extrai o conteúdo dos arquivos .bky, enviando-os para o *script analise_bky.py* para efetuar-se a contagem dos blocos, após isso, o é retornado o conteúdo da contagem e é criado um *.csv* respectivo para cada *.bky* analisado e ao final é gerado o *.csv* final que é o somatório de todos os outros.
- analise_bky.py. Esse programa possui uma função que recebe um texto *XML* derivado dos arquivos *.bky* gerados pelo *script gerar_bky.py*, ao receber o texto é feita a contagem da ocorrência de cada tipo de bloco e então é retornado os valores da contagem para o *script gerar_bky.py*.

Já o *script* em *Typescript* denominado *comparar.ts* é responsável por efetuar a comparação entre cada um dos arquivos *.csv* finais de cada exercício de cada submissão e comparar com os modelos de resposta que também vão estar no formato *.csv*, para assim validar quem fez o envio correto de cada exercício.

Além disso, os dados referentes às submissões dos exercícios em cada um dos módulos, foram oriundos do Moodle (Ambiente Virtual de Aprendizagem) do curso e também todas as informações referentes a identidade dos estudantes foram anonimizadas para garantir a privacidade dos mesmos. Dessa forma, após realizada a contagem dos blocos, procede-se para a análise, também automática, desta contagem. É verificado quantas das submissões foram iguais ou maiores em quantidade por tipo de bloco, quando comparado com os modelos de resposta do respectivo exercício. Para as submissões que falham na comparação com os modelos de resposta, verificou-se manualmente o projeto para identificar o erro, ou possível estruturação do programa que não foi mapeada nos modelos. Assim, é possível saber quantos estudantes conseguiram desenvolver os conceitos esperados em determinado exercício e quantos apresentaram algum *deficit*, de acordo com os erros mapeados.

[1]Os *scripts* podem ser acessados em `https://github.com/Elias-Dias-De-Araujo/app_inventor_analise`

5. Resultados

Primeiramente, vale ressaltar, que todos os exercícios possuem apenas uma tela e que caso possuam algum bloco modificador de propriedade de componente no seu modelo de resposta, então irá significar que algum componente visual da aplicação irá mudar de estado, ou seja, não será estático, dessa forma pontuando nível 1 no conceito de *telas e decomposição de problema*, caso não possua, então irá pontuar nível 0 neste conceito. Ademais, para o conceito *interface de usuário*, ter um componente visual de arranjo é o equivalente a um componente de organização de tela, que será responsável por organizar a interface de diferentes maneiras, dessa forma, todos os componentes que estiverem em seu escopo estarão então fazendo parte de um componente de arranjo, caso estejam fora do escopo, então não estarão fazendo parte.

A Tabela 2 ilustra a relação entre os modelos de resposta propostos e os níveis do PC que são avaliados para cada um dos exercícios do Módulo 3, segundo a rubrica de [von Wangenheim et al. 2018]. Para o critério *telas e decomposição de problema*, os modelos I e II não apresentaram mudança nas propriedades de componentes, logo se mantiveram no nível 0, já o modelo III possui um componente que muda de estado, pois suas propriedades mudam; Para o critério *interface de usuário*, os modelos I e II usam apenas um componente visual sem arranjo que é um botão por isso eles se mantém no nível 0, já para o modelo III são usados dois, em que um é um botão e o outro é uma caixa de texto que estão fora de arranjo também, mantendo-se no nível 1; Para o critério *eventos* o modelo I possui um bloco de manipulação de evento, mantendo-se no nível 1, já os modelos II e III possuem dois blocos de manipulação de evento cada, mantendo-se no nível 2; Para o critério *sensores*, apenas o modelo I não possui um componente de sensor, mantendo-se nível 0, já os outros dois modelos possuem 1 sensor cada, mantendo-se no nível 1; Para o critério *mídia*, todos os modelos tem um componente de mídia, sendo assim, mantendo-se todos no nível 1. Para todos os outros critérios, os modelos ficaram no nível 0, e a demonstração da análise não será aqui exibida por restrições de espaço e o modelo de resposta III é o que contém os maiores níveis nos conceitos do PC deste módulo.

Dessa forma, para o módulo analisado segundo rubrica de avaliação, espera-se que os alunos evoluam do nível 0 para o nível 1 no conceito *telas e decomposição de problema*, *interface do usuário* e *sensores*; evoluam do nível 1 para o nível 2 no conceito de *eventos*; atinjam o nível 1 no conceito de *mídia*.

Tabela 2. Relacionamento entre Modelo de Resposta e Conceito do PC.

Critério	Níveis - Modelo de Resposta (I)	Níveis - Modelo de Resposta (II)	Níveis - Modelo de Resposta (III)
Telas e decomposição de problema	0	0	1
Interface de usuário	0	0	1
Eventos	1	2	2
Sensores	0	1	1
Mídia	1	1	1

A avaliação dos portfólios dos estudantes incluiu utilizar os *scripts* de contagem e

analise automática de blocos no universo de submissões dos exercícios do Módulo 3. A Figura 1 consta que após a análise dos dados: 200 alunos realizaram alguma submissão neste primeiro módulo de programação. Dentre elas, as submissões de 77 alunos foram consideradas inválidas para nossa análise, pois foram submissões que continham arquivos que eram diferentes do formato `.aia` esperado, o qual possibilita realizar uma análise automática com o *script* desenvolvido. Portanto, esta pesquisa conta com um universo de 123 alunos que realizaram submissões válidas.

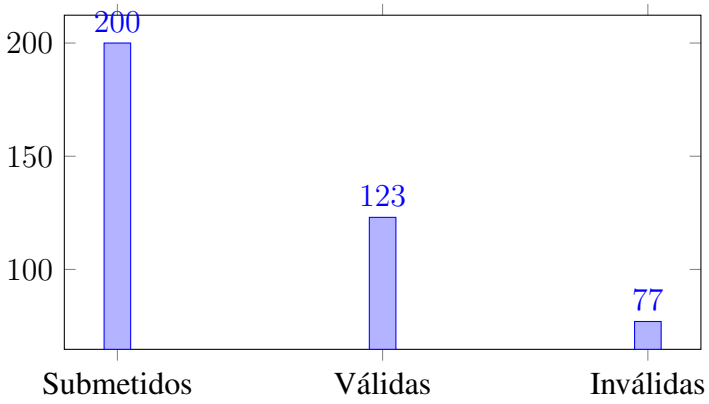

Figura 1. Quantidade de alunos e a caracterização de suas submissões.

Em virtude disto, foi efetuada a contagem dos blocos nas submissões válidas, comparando-as com o modelo de resposta equivalente. A Figura 2 exibe as submissões constatadas para cada um dos exercícios do Módulo 3. As "*submissões corretas*" correspondem aos envios que foram equivalentes ao modelo de resposta, ou que foram constatados como corretos de maneira manual, as "*submissões incorretas*" correspondem aos envios que não foram iguais ao modelo de resposta e que foram constatados como incorretos de maneira manual.

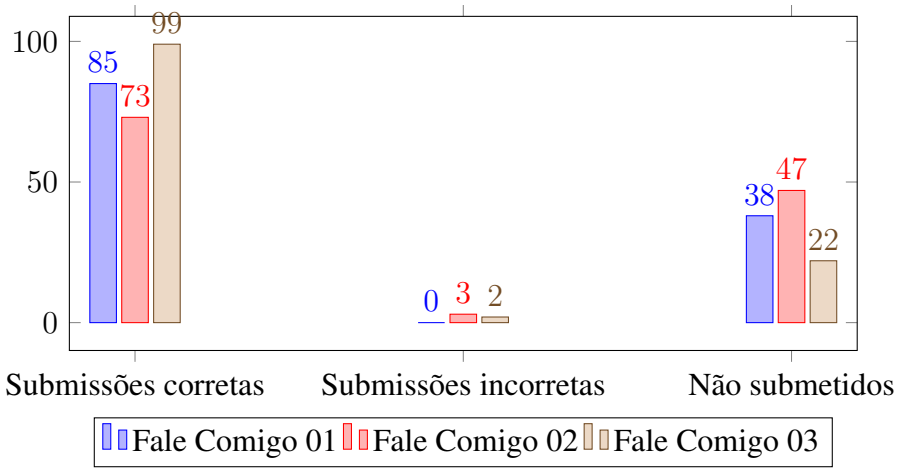

Figura 2. Corretude das submissões válidas dos exercícios do módulo 03.

Das submissões corretas na Figura 2, podemos constatar que:

- Para o exercício *Fale Comigo 01*, considerou-se que 85 alunos alcançaram os níveis de pensamento computacional esperados para este exercício, conforme

mostrado na Tabela 1. No entanto, os 15 alunos que entregaram apenas este exercícios, e não mais os outros, não avançaram nos níveis do pensamento computacional nos conceitos de *telas e decomposição de problema*, *eventos* e *sensores*;

- Para o exercício *Fale Comigo 02*, considerou-se que 73 alunos atingiram os níveis de pensamento computacional esperados para este exercício, conforme mostrado na Tabela 1. No entanto, os 6 alunos que resolveram apenas esse exercício, e nenhum outro, não conseguiram subir de nível 0 para nível 1 em *telas e decomposição de problema* e *interface de usuários*.
- Para o exercício *Fale Comigo 03*, considerou-se que as 99 submissões atingiram o nível máximo dos conceitos esperado para o módulo introdutório do curso. 28 alunos enviaram apenas esse exercício, todavia, por ser o exercício que continha nível máximo em todos os conceitos do módulo, então eles não tiveram nenhum prejuízo de aprendizado.

Para as submissões incorretas, verificou-se que para o exercício *Fale Comigo 01*, não houve este tipo de submissão. Assim, todas as submissões enviadas atingiram os níveis do pensamento computacional esperado para esse exercício. Para os demais exercícios, existiram 5 submissões com erro no total: 1 aluno duplicou um evento de componente para o exercício *Fale Comigo 02*, o que impossibilitou a execução do evento, fazendo com que ele não atingisse o nível 2 em *eventos* exigido para esse exercício; 1 aluno resolveu apenas o exercício *Fale Comigo 01* e tentou submeter os demais dois exercícios, todavia não criou a quantidade de eventos necessários para resolvê-los, ou seja, não evoluiu o critério de *eventos* para o nível 2; 1 aluno submeteu apenas os exercícios *Fale Comigo 02* e *Fale Comigo 03*, todavia, o exercício *Fale Comigo 03* estava errado pois ele não conseguiu criar a quantidade de blocos de chamada de procedimento e de texto, não conseguindo evoluir para os níveis do pensamento computacional exigidos por este exercício; 1 aluno fez a submissão incorreta do exercício *Fale Comigo 02*, pois não conseguiu criar a quantidade de blocos de texto necessários, todavia, resolveu corretamente o exercício *Fale Comigo 01* e o *Fale Comigo 03*. atingindo assim o nível mais alto nos critérios esperados para o módulo.

6. Conclusões e Trabalhos Futuros

Este artigo apresentou um estudo da aquisição de conhecimento em programação de estudantes de escolas públicas do ensino médio do estado do Ceará, participantes do curso "*Fábrica de Programadores - Aprendendo a programar com Games*", por meio de uma análise a partir da relação entre os fundamentos de programação e os conceitos do pensamento computacional destacadas na rubrica de [von Wangenheim et al. 2018]. O artigo concentrou-se na análise dos exercícios do módulo 3 do curso, que é o módulo introdutório dos fundamentos de programação. Realizamos as seguintes análises: (i) analisamos os modelos de respostas dos exercícios do curso para avaliar que níveis do pensamento computacional, segundo rubrica de avaliação, era exigido para cada um dos exercícios e; (ii) analisamos os portfólios dos alunos, isto é, os exercícios práticos de programação submetidos pelos participantes do curso, com o objetivo de verificar se estes atingiram os níveis do pensamento computacional esperado para cada exercício. Para análise automática dos portfólios dos alunos, desenvolvemos *scripts* em *Python* e em *Typescript*.

Com a análise dos exercícios do módulo 3, segundo rubrica de avaliação, espera-

se que os alunos: evoluam do nível 0 para o nível 1 no conceito *telas e decomposição de problema*, *interface do usuário* e *sensores*; evoluam do nível 1 para o nível 2 no conceito de *eventos* e; que atinjam o nível 1 no conceito de *mídia*. A análise dos portfólios revelou que o resultado foi satisfatório, já que de 123 alunos, 99 conseguiram atingir o nível máximo nos conceitos propostos para o módulo, totalizando $80,48\%$. Esse fator constata que o aprendizado de programação por parte dos estudantes ao utilizar o App Inventor como ferramenta de aprendizado de programação no formato de ensino EAD foi adequado.

Como trabalhos futuros, pretendemos: (i) avaliar os outros módulos de programação da primeira edição do curso e (ii) avaliar as submissões referentes a segunda edição do curso e (iii) realizar um comparativo das abordagens da primeira e segunda edição para avaliar se ocorreu uma melhora na quantidade de alunos que adquiram os níveis dos conceitos do pensamento computacional esperados.

Referências

Bordini, A., Avila, C. M. O., Weisshahn, Y., da Cunha, M. M., da Costa Cavalheiro, S. A., Foss, L., Aguiar, M. S., and Reiser, R. H. S. (2016). Computaçao na educaçao básica no brasil: o estado da arte. *Revista de Informática Teórica e Aplicada*, 23(2):210–238.

Code.org (2023). Code.org. Acessado em 13 de outubro de 2023.

Council, N. R. et al. (2011). *Report of a workshop on the pedagogical aspects of computational thinking*. National Academies Press. Acessado em 30 de setembro de 2023.

da Cruz Alves, N. and von Wangenheim, C. G. (2023). Uma análise em larga-escala das funcionalidades de aplicativos criados com app inventor. In *Anais do III Simpósio Brasileiro de Educação em Computação*, pages 27–36. SBC.

da Cruz Alves, N., von Wangenheim, C. G., and Hauck, J. C. R. (2020). Teaching programming to novices: A large-scale analysis of app inventor projects. In *2020 XV Conferencia Latinoamericana de Tecnologias de Aprendizaje (LACLO)*. IEEE. Acessado em 16 de setembro de 2023.

dos Santos, V. G. and da Silva, S. L. (2020). Educação tecnológica: o ensino da programação para crianças do ensino fundamental através do ambiente code. org. *Conecte-se! Revista Interdisciplinar de Extensão*, 4(7):23–39.

fdr (2023). Aprendendo a programar com games. Acessado em 7 de outubro de 2023.

Kaminski, M. R. and Boscarioli, C. (2019). Uso do ambiente code. org para ensino de programação no ensino fundamental i-uma experiência no desafio hora do código. *Revista ENCITEC*, 9(1):63–76.

MEC (2017). Base nacional comum curricular educação é a base. pages 477–544. Acessado em 16 de setembro de 2023.

MEC (2022). Computação na educação básica - complemento à bncc. Acessado em 24 de junho de 2024.

MIT (2022a). Mit app inventor. Acessado em 12 de agosto de 2024.

MIT (2022b). Sharing and remixing apps. Acessado em 23 de novembro de 2023.

MIT (2023). Scratch. Acessado em 13 de outubro de 2023.

Monroy-Hernández, A. and Resnick, M. (2008). Empowering kids to create and share programmable media. *interactions*, 15(2):50–53. Acessado em 14 de agosto de 2024.

Moreno-León, J., Robles, G., and Román-González, M. (2015). Dr. scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *RED. Revista de Educación a Distancia*, (46):1–23. Acessado em 30 de setembro de 2023.

Nascimento, R. d. S. (2023). O uso da ferramenta app inventor no ensino de programação em cursos de ti. Acessado em 2 de outubro de 2023.

Oliveira, W. and Cambraia, A. C. (2020). Desafios na formação de professores de computação: Reflexões e ações em construção. In *Anais do XXVI Workshop de Informática na Escola*, pages 319–328. SBC.

Park, Y. and Shin, Y. (2019). Comparing the effectiveness of scratch and app inventor with regard to learning computational thinking concepts. *Electronics*, 8(11):1269. Acessado em 16 de setembro de 2023.

Patton, E. W., Tissenbaum, M., and Harunani, F. (2019). Mit app inventor: Objectives, design, and development. *Computational thinking education*, pages 31–49. Acessado em 2 de outubro de 2023.

Scherer, R., Siddiq, F., and Viveros, B. S. (2019). The cognitive benefits of learning computer programming: A meta-analysis of transfer effects. *Journal of Educational Psychology*, 111(5):764–792. Acessado em 9 de setembro de 2023.

Sherman, M. and Martin, F. (2015). The assessment of mobile computational thinking. *Journal of Computing Sciences in Colleges*, 30(6):53–59. Acessado em 27 de novembro de 2023.

Sherman, M., Martin, F., Baldwin, L., and DeFilippo, J. (2014). App inventor project rubric–computational thinking through mobile computing. *https://nsfmobilect.files.wordpress.com/2014/09/mobile-ct-rubric-for-app-inventor-2014-09-01.pdf*. Acessado em 27 de novembro de 2023.

Sousa, L. D. L., Farias, E. J., and de Carvalho, W. V. (2020). Programação em blocos aplicada no ensino do pensamento computacional: Um mapeamento sistemático. In *Anais do XXXI Simpósio Brasileiro de Informática na Educação (SBIE 2020)*. Sociedade Brasileira de Computação.

Tang, X., Yin, Y., Lin, Q., Hadad, R., and Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education*, 148:103798. Acessado em 1 de outubro de 2023.

Tsai, C.-Y. (2019). Improving students' understanding of basic programming concepts through visual programming language: The role of self-efficacy. *Computers in Human Behavior*, 95:224–232. Acessado em 2 de outubro de 2023.

UFC (2022). Fundamentos de programação. Acessado em 13 de outubro de 2023.

USP (2017). Fundamentos de linguagem de programação. Acessado em 13 de outubro de 2023.

von Wangenheim, C. G., Hauck, J. C., Demetrio, M. F., Pelle, R., da Cruz Alves, N., Barbosa, H., and Azevedo, L. F. (2018). Codemaster–automatic assessment and grading of app inventor and snap! programs. *Informatics in Education*, 17(1):117–150. Acessado em 30 de setembro de 2023.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3):33–35. Acessado em 12 de agosto de 2024.