

UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

**Uma Ferramenta de Simulações
Interativas para o Ensino de Conceitos de
Programação para Crianças**

Marília Takaguti Dicezare

MONOGRAFIA FINAL

MAC 499 — TRABALHO DE
FORMATURA SUPERVISIONADO

Supervisora: Prof.^a Dr.^a Kelly Rosa Braghetto

São Paulo
2024

*O conteúdo deste trabalho é publicado sob a licença CC BY 4.0
(Creative Commons Attribution 4.0 International License)*

Agradecimentos

*Quando a vida decepçiona qual é a solução?
Continue a nadar, continue a nadar, continue a nadar, nadar, nadar.
Pra achar a solução, nadar, nadar.
— Dory (Procurando Nemo)*

Primeiramente, gostaria de agradecer a Deus, que sempre está presente em minha vida. Ele me permite ter força e paciência para não desistir jamais.

Agradeço imensamente à minha família, especialmente, aos meus pais. Seu apoio e amor incondicionais me permitiram trilhar meus próprios caminhos e chegar até este momento, isso foi o mais importante.

Agradeço aos meus colegas do BCC, pela amizade, colaborações e aprendizados e pelos momentos de descontração que tornaram a graduação mais divertida. Aos professores e mentores, que nos promoveram oportunidades de crescimento pessoal e profissional. Em particular, ao meu primeiro mentor, João Daniel, que me apresentou oficialmente ao incrível mundo da Engenharia de *Software* e me ensinou os primeiros passos dentro dela. Ao professor Alfredo Goldman, que me ensinou que “*software* é sobre pessoas”. Uma lição importante que vivenciei e que levarei para a minha carreira.

Em especial, gostaria de agradecer profundamente a minha orientadora, professora Kelly Braghetto, pela paciência e pelos *insights* e *feedbacks* valiosíssimos durante o desenvolvimento, não somente desse trabalho de formatura, mas também durante minha pesquisa de Iniciação Científica.

Agradeço também ao professor Leônidas Brandão, que nos auxiliou na avaliação do protótipo proposto neste projeto, fornecendo o contato de um docente da Escola de Aplicação da FEUSP para que pudéssemos testar e validar nossa ideia na prática. Ao professor Henri Silva da Escola de Aplicação por disponibilizar uma aula para testarmos nossa ferramenta.

Por fim, agradeço a todos do Instituto de Matemática e Estatística da USP por proporcionarem um ambiente confortável e receptivo para os alunos.

Muito obrigada a todos que me acompanharam nessa jornada!

Resumo

Marília Takaguti Dicezare. **Uma Ferramenta de Simulações Interativas para o Ensino de Conceitos de Programação para Crianças.** Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2024.

O ensino de Computação para crianças e adolescentes é cada vez mais importante na Era Digital. O conhecimento de programação promove muitas competências fundamentais para o pensamento crítico e lógico desses jovens. Por isso, ele vem ganhando cada vez mais espaço nas escolas de educação básica brasileiras, tornando-se parte dos currículos escolares. Ferramentas de apoio pedagógico são fundamentais nesse processo de aprendizagem e o uso de simulações interativas na educação tem se mostrado eficaz em outras ciências. Nesse sentido, este Trabalho de Formatura teve como objetivo a criação e validação de um produto mínimo viável (MVP - *Minimum Viable Product*) de uma ferramenta *web* de simulações interativas de conceitos de lógica de programação para o ensino de Computação no ensino fundamental. A ferramenta considera conceitos de lógica de programação básicos, como variáveis, entrada e saída de dados, condicionais e laços de repetição, permitindo que os estudantes a explore livremente. Ademais, ela apresenta o pseudocódigo correspondente, a fim de promover o contato com a estrutura real de um código. Para realizar este trabalho, seguimos a metodologia *Design Science Research* (DSR) em Engenharia de Software, para construção de artefatos em contexto. Assim, investigamos o problema por meio de uma contextualização e, a partir disso, projetamos e validamos um artefato, o MVP. Também realizamos a implementação dele utilizando o *framework* Vue.js em Typescript. Por fim, avaliamos a usabilidade do MVP proposto e o aprendizado dos alunos através dele, em uma atividade com estudantes do ensino fundamental, por meio de um questionário de usabilidade (SUS - *System Usability Scale*) e de um questionário sobre o aprendizado dos conceitos de lógica de programação. A análise dos resultados dessas avaliações nos permitiu concluir que o uso de simulações interativas para ensino de Computação demonstrou potencial, principalmente quando aplicadas a crianças que já tiveram um contato anterior com programação, apresentando uma pontuação de usabilidade considerada boa. Considerando alunos que não têm conhecimento prévio em Computação, a usabilidade da ferramenta foi classificada como razoável segundo a pontuação do SUS e observamos a necessidade de melhorias na forma de introduzir os conceitos para esses usuários. Por meio da atividade de aprendizado, concluímos que a abordagem proposta não ajudou na compreensão dos conceitos de programação, uma vez que a maioria dos alunos não conseguiram identificá-los em trechos de pseudocódigo no questionário. Entretanto, observamos que algumas das confusões feitas pelos estudantes podem ter sido causadas pela formulação das questões avaliativas. Dessa maneira, foi possível notar a necessidade de melhorias na simulação e no pseudocódigo do MVP avaliado, implicando em uma nova interação nos ciclos de design e engenharia de DSR, o que seria um passo natural para este tipo de pesquisa.

Palavras-chave: simulações interativas. ensino de Computação. lógica de programação. MVP.

Abstract

Marília Takaguti Dicezare. **An Interactive Simulation Tool for Teaching Programming Concepts to Children**. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2024.

Teaching computer science to children and teenagers is increasingly important in the Digital Age. Programming knowledge promotes many fundamental skills for critical and logical thinking in these young people. Therefore, it gained more and more space in Brazilian basic education schools, becoming part of the school curricula. Pedagogical support tools are fundamental in this learning process and the use of interactive simulations in education has proven effective in other sciences. In this sense, this capstone project aimed to create and validate a minimum viable product (MVP) of a web tool for interactive simulations of programming logic concepts for teaching computing in elementary school. The tool considers basic programming logic concepts, such as variables, data input and output, conditionals, and loops, allowing students to explore it freely. Furthermore, it presents the corresponding pseudocode, to promote contact with the real structure of a code. To carry out this work, we followed the Design Science Research (DSR) methodology in Software Engineering, for building artifacts in context. Thus, we investigated the problem through contextualization and, based on that, we designed and validated an artifact, the MVP. We also implemented it using the Vue.js framework in Typescript. Finally, we evaluated the usability of the proposed MVP and the students' learning through it in an activity with elementary school students applying a usability questionnaire (SUS - System Usability Scale) and a questionnaire about the learning of programming logic concepts. Analysis of the results of these evaluations allowed us to conclude that using interactive simulations for teaching computing demonstrated potential, especially when applied to children who have had previous contact with programming, presenting a usability score considered good. Considering students who have no prior knowledge of computing, the tool's usability was classified as fair according to the SUS score and we observed the need for improvements in the way concepts are introduced to these users. Through the learning activity, we concluded that the proposed approach did not help in understanding programming concepts, since most students could not identify them in pseudocode snippets in the questionnaire. However, we observed that the formulation of the assessment questions may have caused some confusion among students. In this way, it was possible to note the need for improvements in the simulation and pseudocode of the evaluated MVP, implying a new interaction in the DSR design and engineering cycles, which would be a natural step for this type of research.

Keywords: interactive simulations. computer teaching. programming logic. MVP.

Lista de abreviaturas

BNCC	Base Nacional Curricular Comum
DS	<i>Design Science</i>
DSR	Pesquisa em <i>Design Science</i> (<i>Design Science Research</i>)
FEUSP	Faculdade de Educação da Universidade de São Paulo
IC	Iniciação Científica
IME	Instituto de Matemática e Estatística
MIT	Instituto de Tecnologia de Massachusetts (<i>Massachusetts Institute of Technology</i>)
MVP	Produto mínimo viável (<i>Minimum Viable Product</i>)
PhET	<i>Physics Education Technology</i>
SBC	Sociedade Brasileira de Computação
SI	Sistemas de Informação
SUS	<i>System Usability Scale</i>
TI	Tecnologia da Informação
UI	Interface do usuário (<i>User Interface</i>)
USP	Universidade de São Paulo
WCAG	Diretrizes de Acessibilidade para Conteúdo <i>Web</i> (<i>Web Content Accessibility Guidelines</i>)

Lista de figuras

1	Eixos da Computação (RIBEIRO <i>et al.</i> , 2019).	3
1.1	Interface do usuário da ferramenta <i>Scratch</i> (<i>Scratch - MIT</i> 2024).	8
1.2	Interface do usuário da ferramenta <i>App Inventor</i> para criar o design de um aplicativo (<i>MIT App Inventor</i> 2024).	8
1.3	Interface do usuário da ferramenta <i>App Inventor</i> para criar a lógica de interação dos elementos do aplicativo (<i>MIT App Inventor</i> 2024).	9
1.4	Interface do usuário do jogo <i>CodeCombat</i> (<i>CodeCombat</i> 2024).	9
1.5	Exemplo da utilização da linguagem do <i>Scratch</i> para programação do Arduino (MEDEIROS e WÜNSCH, 2019).	10
1.6	Panorama o kit educativo <i>Lego Mindstorms EV3</i> (<i>Lego Mindstorms</i> 2024). .	11
1.7	Descrição de um exemplo de atividade desplugada (BRACKMANN, 2017). .	11
1.8	Ilustração do objeto de aprendizagem de variáveis desenvolvido por FERNANDES <i>et al.</i> (2012).	12
1.9	Interface do usuário da simulação <i>Drone Blocks</i> (<i>Drone Blocks</i> 2024). . . .	13
2.1	Representação visual da escala de Likert (PUTNAM <i>et al.</i> , 2020).	21
3.1	Protótipo inicial da simulação “Quarto de Brinquedos”.	23
3.2	Protótipo inicial da simulação “Planejando a Festa”.	24
3.3	Protótipo da simulação “Guardando os Brinquedos” após validação. . . .	25
3.4	Protótipo da simulação “Planejando a Festa” após validação.	26
4.1	Tela inicial do MVP da simulação “Planejando a Festa”.	27
4.2	Representação visual do conceito de saída com a mensagem “impressa” na tela.	28
4.3	Escolhendo um item na simulação “Planejando a Festa”.	29
4.4	Destaques do pseudocódigo da simulação “Planejando a Festa” mostrando a separação em cores por conceito de lógica de programação e a indicação do nome deles ao passar o mouse por cima de cada um.	30

4.5	Caixa de diálogo contendo as definições dos conceitos de lógica de programação abordados na simulação “Planejando a Festa”. O menu lateral permite a escolha de um conceito, mostrando a sua definição e exemplos ao lado.	30
5.1	Representação visual da escala de Likert utilizada no formulário de usabilidade.	32
5.2	Trechos de pseudocódigo apresentados em cada pergunta da atividade de aprendizado referente à simulação “Planejando a Festa”.	33
5.3	Protótipo da simulação “Guardando os Brinquedos” apresentado na atividade de aprendizado para fornecer contexto para as questões.	34
5.4	Trechos de pseudocódigo apresentados em cada pergunta da atividade de aprendizado referente à simulação “Guardando os Brinquedos”.	35
6.1	Porcentagem do nível de concordância com as afirmações dos alunos que já tiveram contato anterior com programação.	38
6.2	Porcentagem do nível de concordância com as afirmações dos alunos que não tiveram contato anterior com programação.	38
6.3	Média dos níveis de concordância com cada afirmação dos alunos que já tiveram contato anterior com programação e dos que não tiveram. Cada raio do gráfico representa a média para uma afirmação do formulário. . .	39
6.4	Respostas dos alunos em relação à pergunta livre “O que você mais gostou da simulação?”, divididas em categorias.	41
6.5	Respostas dos alunos em relação à pergunta livre “O que você menos gostou da simulação?”, divididas em categorias.	42
6.6	Respostas dos alunos em relação à pergunta livre sobre comentários e sugestões da simulação, divididas em categorias.	43
6.7	Porcentagens das respostas dos alunos às perguntas da atividade de aprendizado relacionada à simulação “Planejando a Festa”. Para cada pergunta, apresentamos as respostas dos alunos que já tiveram contato anterior com Computação à esquerda e as dos que não tiveram à direita.	44
6.8	Porcentagens das respostas dos alunos às perguntas da atividade de aprendizado relacionada à simulação “Guardando os Brinquedos”. Para cada pergunta, apresentamos as respostas dos alunos que já tiveram contato anterior com Computação à esquerda e as dos que não tiveram à direita.	46

Lista de tabelas

2.1	Afirmativas em inglês do questionário de avaliação de usabilidade de sistemas SUS e adaptações propostas por PUTNAM <i>et al.</i> (2020) para crianças entre 7 e 11 anos, dividida em dois grupos.	21
6.1	Pontuação SUS calculada para cada aluno, com as médias para alunos que já tiveram contato anterior com programação e para os que não tiveram.	40

Sumário

Introdução	1
Ensino de Computação no Brasil e no mundo	1
O uso de ferramentas no ensino de Computação	3
Objetivos e contribuições	5
1 Ferramentas pedagógicas para ensino de Computação	7
1.1 Linguagem de programação visual em blocos	7
1.2 Jogos digitais	9
1.3 Kits de robótica	10
1.4 Computação desplugada	10
1.5 Simulações interativas	12
2 Metodologia de pesquisa	15
<i>Design Science Research</i>	15
2.1 Ciclo de <i>design</i>	17
2.1.1 Investigação do problema	17
2.1.2 Projeto do artefato	17
2.1.3 Validação do artefato	18
2.2 Ciclo de engenharia	18
2.2.1 Implementação da simulação	19
2.2.2 Avaliação da simulação	19
3 Criação e validação dos protótipos da simulação	23
4 Implementação da simulação	27
5 Avaliação da simulação	31
6 Resultados da avaliação da simulação	37
6.1 Formulário de usabilidade	37

Opiniões e sugestões sobre a simulação	40
6.2 Atividade de aprendizado	42
6.2.1 Perguntas sobre a simulação “Planejando a Festa”	43
6.2.2 Perguntas sobre a simulação “Guardando os Brinquedos”	45
7 Considerações finais	49
7.1 Conclusões	49
7.2 Limitações e trabalhos futuros	50
Referências	53

Introdução

O ensino da Computação estimula o desenvolvimento de importantes habilidades do mundo digital, como o raciocínio lógico, a análise e resolução de problemas, o pensamento crítico, a criatividade, a colaboração, entre outras. O pensamento computacional, que possibilita o desenvolvimento dessas habilidades, vai muito além de programar, sendo fundamental não somente para a Ciência da Computação, mas também para leitura, escrita e aritmética, proporcionando capacidades analíticas para crianças (WING, 2006).

Ensino de Computação no Brasil e no mundo

As habilidades desenvolvidas por meio do pensamento computacional são cada vez mais relevantes para crianças e adolescentes de diversas faixas etárias. Nesse sentido, muito tem sido discutido acerca da aprendizagem de Computação na educação básica nas últimas décadas. Escolas em vários países ao redor do mundo já introduziram o ensino de Computação. Alguns estudos revelam os desafios e as expectativas desta implementação, além de fatores importantes na educação em escolas k-12, as quais correspondem às séries do jardim de infância ao ensino médio.

HUBWIESER *et al.* (2015) mostraram a situação do ensino de Computação em escolas k-12 em 14 casos de estudo em estados de 12 países: Alemanha, Estados Unidos, Índia, Nova Zelândia, França, Coreia, Suécia, Reino Unido, Finlândia, Israel, Rússia e Itália. Eles identificaram diferenças em diversos aspectos nas implementações do ensino de Computação: encontraram cerca de 40 termos diferentes para descrever áreas relacionadas à Ciência da Computação; 24 categorias de objetivos de ensino; 19 de conteúdos de Computação, como conceitos de algoritmos, inteligência artificial, resolução de problemas, entre outros; 4 subcategorias de linguagens de programação, como sistemas baseados em hardware, ambientes de programação educacional com linguagem própria e baseados em outras linguagens, e linguagens de programação profissionais; e diferentes níveis de educação exigidos dos professores. Os autores ainda ressaltaram que as análises representam o estado global “instantâneo” (*snapshot*) no momento da pesquisa.

Outro trabalho, conduzido por FALKNER *et al.* (2019), analisou a implementação do ensino de Ciência da Computação no k-12, comparando os requisitos curriculares definidos por padrões (pretendidos) com o que realmente é implementado nas salas de aula pelos docentes. Foram analisados currículos em 7 países - Austrália, Inglaterra, Irlanda, Itália, Malta, Escócia e Estados Unidos -, com foco apenas em conceitos de Ciência da Computação e linguagens de programação. Eles observaram que os tópicos mais ensinados são algoritmos, programação, pensamento computacional e representação de dados. Alguns

tópicos, como inteligência artificial e robótica, apenas constam no currículo pretendido dos Estados Unidos e Malta, respectivamente, mas são abordados em sala de aula em diversos países. Ainda, eles descobriram que os educadores tendem a escolher as linguagens de programação baseados em fatores motivados pelos alunos ao invés de se basearem no currículo pretendido. O estudo mostrou que os professores utilizam tanto programação visual quanto baseada em texto no ensino fundamental, além de identificarem o uso frequente de atividades desplugadas, apesar destas não estarem explicitamente definidas no currículo padrão. Os autores também identificaram diferenças no alinhamento curricular entre os países, as quais acreditam que possam ajudar a guiar uma futura reforma curricular.

Ademais, [ZHU e WANG \(2023\)](#) realizaram uma pesquisa fenomenológica sobre fatores críticos na educação de Ciência da Computação em escolas k-12, a partir das perspectivas e visões de professores do ensino superior e de k-12. As entrevistas semiestruturadas conduzidas com os docentes indicaram que as habilidades de resolução de problemas pelos estudantes, por meio do pensamento computacional, são as competências mais importantes no ensino de Computação. Além disso, conhecimentos básicos em Matemática e programação também são bastante relevantes, enquanto que o ensino de linguagens de programação específicas não foram consideradas importantes.

No Brasil, pesquisadores, instituições de ensino e organizações juntaram esforços para implementar o ensino de Computação na educação básica. Segundo relatório do Conselho Nacional de Educação (CNE) ([BRASIL, 2021](#)), com a implantação da Base Nacional Comum Curricular (BNCC) em 2017, o CNE ficou responsável por elaborar as normas específicas sobre Computação. Entre 2019 e 2021, foram designados os membros da comissão para formular tais normas. Também houve colaborações de pesquisadores da Sociedade Brasileira de Computação (SBC), do Centro de Inovação para a Educação Brasileira (CIEB), do Ministério da Educação (MEC), dentre outras organizações. Finalmente, em 2022, o CNE homologou as Normas sobre Computação na Educação Básica - Complemento à BNCC.

Alguns estudos anteriores à homologação das normas mostram o panorama do processo de implementação da Computação na educação básica brasileira. Por exemplo, [FRANÇA e AMARAL \(2013\)](#) realizaram um mapeamento sistemático de artigos referentes ao ensino de Computação na educação básica publicados entre os anos de 2009 e 2012. Eles observaram um crescente interesse dos pesquisadores brasileiros no assunto, com destaque para as instituições de pesquisa das regiões Nordeste e Sul do país. Em outro estudo, [BORDINI et al. \(2016\)](#) apresentaram um levantamento de trabalhos relacionados ao pensamento computacional no ensino fundamental e médio realizados entre 2010 e 2015, com o intuito de mostrar o que já foi alcançado nessa área no Brasil. Eles identificaram diferenças nas metodologias de ensino, bem como nas ferramentas utilizadas e no público alvo dos estudos, mostrando conformidade com os trabalhos internacionais.

Ainda, [P. S. C. SANTOS et al. \(2018\)](#) analisaram a literatura sobre pensamento computacional e programação na educação básica brasileira no período de 2001 a 2016. Eles consideraram fatores como metodologia, ferramentas, público alvo e outros, mostrando novamente o crescente interesse dos pesquisadores no tema. Ademais, os autores apontaram algumas tendências e lacunas nessa área. [S. M. DOS SANTOS et al. \(2021\)](#) realizaram uma revisão sistemática sobre as diferentes abordagens de ensino de Computação no ensino fundamental em estudos publicados entre 2009 e 2020, em bases nacionais e internacionais.

Eles analisaram diferentes tipos de atividades voltadas ao ensino de Computação: plugadas, desplugadas e híbridas, apontando tendências e lacunas observadas nos trabalhos, como por exemplo, equidade e inclusão.

Diante desse cenário nacional e mundial, a SBC elaborou as Diretrizes de Ensino de Computação na Educação Básica (RIBEIRO *et al.*, 2019), visando facilitar a implementação do ensino de Computação nos currículos das escolas brasileiras. Os conceitos de Computação são organizados em três eixos, como mostra a Figura 1:

- **Pensamento computacional:** refere-se à habilidade de analisar, compreender, modelar, comparar, solucionar e automatizar problemas e soluções de maneira sistemática, por meio de abstrações, análise de informações e da construção de algoritmos.
- **Mundo digital:** nele ocorre a codificação ou representação de informações, o processamento dos dados codificados e a distribuição dessas informações de forma segura.
- **Cultura digital:** envolve o conhecimento das tecnologias digitais, bem como as relações da Computação com outras áreas do conhecimento e a participação crítica, ética e responsável na sociedade do mundo digital.

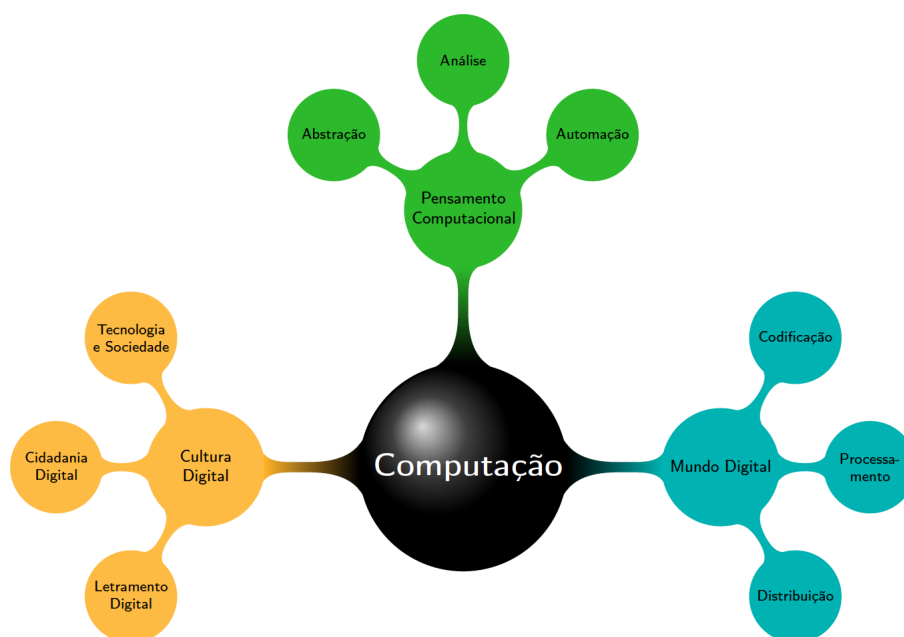


Figura 1: Eixos da Computação (RIBEIRO *et al.*, 2019).

O uso de ferramentas no ensino de Computação

É possível perceber que o ensino de Computação não se limita à aprendizagem de uma linguagem de programação. Ela é apenas uma ferramenta utilizada para aplicar estes conceitos de forma a chegar na solução de problemas, a partir do conhecimento de teoria da Computação e paradigmas de programação (BLATT *et al.*, 2017). Assim, diferentes

metodologias podem ser adotadas no ensino de programação para crianças e jovens e o uso de recursos didáticos apropriados, ferramentas ou aplicativos pedagógicos, é indispensável no aprendizado desses conceitos.

Há diversas maneiras de explorar a aprendizagem dos conceitos de Computação: jogos, programação visual, programação com blocos, kits de robótica, simulações, *storytelling*, entre outras. Dentre as ferramentas utilizadas no ensino fundamental, nota-se uma preferência pelas linguagens visuais e plataformas focadas no ensino dos fundamentos (e não no desenvolvimento) (GOMES *et al.*, 2017). A linguagem de programação em blocos é uma das mais citadas em estudos, apresentando diversas opções de ferramentas (BREZOLIN e SILVEIRA, 2021; SOUZA *et al.*, 2021).

Deste modo, *softwares* de apoio ao ensino de programação devem facilitar a compreensão das abstrações envolvidas nos conceitos de Computação, bem como, estimular o raciocínio lógico. Nesse sentido, o uso de simulações interativas facilita a visualização de conceitos abstratos, utilizando exemplos concretos para representá-los.

Alguns trabalhos mostram a efetividade das simulações no ensino em outras áreas. Em particular, a plataforma de simulações interativas PhET (*Physics Education Technology*) é um recurso bastante utilizado em várias disciplinas de ciências (KHATRI *et al.*, 2013). Criado em 2002 pelo ganhador do prêmio Nobel de Física em 2001, Carl Wieman, o projeto da Universidade de Colorado Boulder apresenta atualmente 170 simulações interativas distribuídas nas áreas de Física, Química, Matemática, Ciências da Terra e Biologia, destinadas tanto a alunos da educação básica como de ensino superior.

Entre 2012 e 2013, a plataforma realizou uma pesquisa com docentes que utilizavam a ferramenta, recebendo cerca de 2.000 respostas de escolas nos Estados Unidos (PRICE *et al.*, 2018). O estudo mostrou três aspectos que contribuem para a escolha da ferramenta na sala de aula. Primeiro, devido a sua flexibilidade, as simulações são utilizadas de diversas maneiras e com diferentes objetivos de aprendizagem, como entender conceitos, processos científicos e aumentar a motivação dos estudantes. Além disso, os docentes preferem que os estudantes tenham controle da simulação. Por fim, algumas propriedades percebidas das simulações foram a visualização, manipulabilidade e a capacidade de realizar demonstrações que não poderiam ser feitas em sala.

No Brasil, diversos estudos mostram pontos positivos da utilização dessa ferramenta como forma de agregar ao ensino tradicional. Uma pesquisa bibliográfica de publicações de autores brasileiros entre os anos de 2010 e 2020, realizada por RAMOS *et al.* (2020), mostrou que a utilização do simulador PhET, junto a uma metodologia de ensino, potencializou o aprendizado dos estudantes, além de torná-los participantes ativos desse processo.

Outro trabalho, conduzido por CRAVO e ESPARTOSA (2021), reforça essa conclusão. Eles avaliaram simulações de ciências e Biologia por meio de um protocolo de avaliação próprio e da realização de oficinas com futuros docentes, mostrando os desafios e potencialidade da ferramenta. Os autores concluíram que ela contribui positivamente no processo de ensino-aprendizagem, tornando-o mais didático e dinâmico, e que a atuação de professores como mediadores do ensino pode ajudar a contornar possíveis deficiências na sua utilização. Ainda, os trabalhos de ARAÚJO *et al.* (2021) e E. O. SANTOS e SILVA (2020) apresentam conclusões similares em relação ao uso de simulações no ensino de Física e Química,

respectivamente.

Assim, esta é uma abordagem interessante a ser explorada também no ensino da Computação, pois da mesma forma que as ciências da natureza e das humanidades ajudam a explicar o mundo real, a Ciência da Computação ajuda a explicar o mundo digital (RIBEIRO *et al.*, 2019). Portanto, é natural que busquemos ferramentas análogas para o estudo dessas ciências e, até onde sabemos, não existem simulações de conceitos de lógica de programação iguais às encontradas no PhET.

Objetivos e contribuições

Desse modo, este Trabalho de Formatura tem como objetivo a criação de um produto mínimo viável (MVP - *Minimum Viable Product*) de uma ferramenta de simulações interativas para o ensino de conceitos de lógica de programação para crianças no ensino fundamental. Para isso, desenvolvemos uma versão da aplicação com um conjunto mínimo de requisitos, contendo uma simulação envolvendo alguns conceitos de lógica de programação. Além disso, queremos que os usuários tenham contato com a estrutura real do código de programação correspondente a cada conceito simulado. Assim, integramos à simulação uma visualização do pseudocódigo associado.

Também aplicamos o MVP em sala de aula para obter *feedback* para um futuro desenvolvimento do sistema completo. Dessa forma, avaliamos a sua usabilidade com a ajuda de estudantes do ensino fundamental, por meio da aplicação de um questionário de usabilidade. Ademais, investigamos como representar os conceitos visualmente de forma que facilite a compreensão da lógica de programação por trás deles.

Capítulo 1

Ferramentas pedagógicas para ensino de Computação

Como vimos no Capítulo de Introdução, o ensino de Computação não está ligado a uma linguagem de programação específica, mas sim à abstração dos conceitos de programação e ao conjunto de habilidades que envolvem o pensamento computacional. Desse modo, diferentes ferramentas e tecnologias são utilizadas no apoio às aulas de Computação nas escolas. A seguir, discutimos algumas das abordagens mais utilizadas no ensino de programação para crianças e adolescentes.

1.1 Linguagem de programação visual em blocos

A linguagem de programação visual em blocos se baseia na construção de algoritmos usando blocos lógicos de maneira simples e intuitiva. Essa abordagem é amplamente utilizada em diversas ferramentas. Podemos citar como principal exemplo o *Scratch* (*Scratch - MIT 2024*), uma plataforma, que utiliza uma linguagem de programação visual baseada em blocos e permite criar histórias, jogos e animações. Criada inicialmente por um grupo do *MIT Media Lab* (Laboratório de Mídia do Instituto de Tecnologia de Massachusetts) e desenvolvida atualmente pela *Scratch Foundation*, ela é uma ferramenta gratuita e de código aberto, e está disponível em mais de 70 idiomas. Segundo seus desenvolvedores, ela promove o pensamento educacional, habilidades de resolução de problemas, criatividade, auto-expressão, colaboração e equidade.

A Figura 1.1 apresenta a interface do *Scratch*. O espaço à esquerda permite arrastar os blocos das diferentes categorias à área central da página, onde é montado o algoritmo. À direita, temos o resultado da execução do conjunto de blocos apresentado na forma de animações.

Outra ferramenta baseada em blocos que merece destaque é o *App Inventor* (*MIT App Inventor 2024*), originalmente criado pelo Google e atualmente mantido pelo MIT. A plataforma permite a criação de aplicativos para os sistemas operacionais Android e iOS. Ela apresenta duas telas para a construção de um aplicativo: uma de design (Figura 1.2), onde o visual do aplicativo pode ser montado, arrastando-se elementos visuais para a tela,

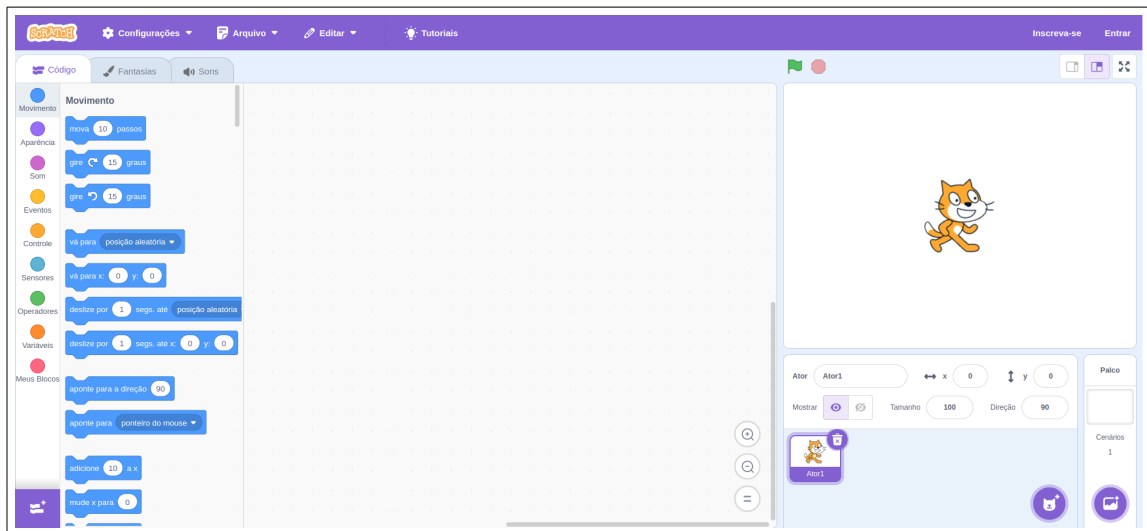


Figura 1.1: Interface do usuário da ferramenta Scratch (Scratch - MIT 2024).

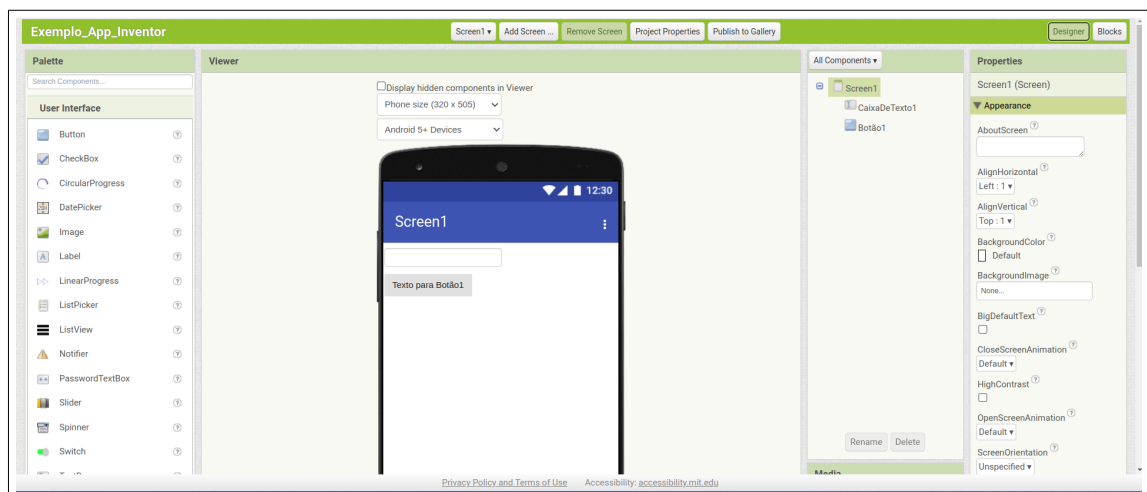


Figura 1.2: Interface do usuário da ferramenta App Inventor para criar o design de um aplicativo (MIT App Inventor 2024).

como botões, texto de entrada, caixas de seleção, entres outros; e uma de blocos (Figura 1.3), onde é definida a lógica de interação com os elementos visuais adicionados como, por exemplo, qual efeito deve ser aplicado a um botão se ele for clicado.

Diversos estudos apontam esta abordagem, principalmente com o uso do *Scratch*, como a mais utilizada no ensino de programação para crianças e adolescentes, apresentando resultados positivos (e.g. *BLATT et al., 2017*; *BORDINI et al., 2016*; *KHOURI et al., 2020*; *WERLICH et al., 2018*). Além disso, podemos observar que outras ferramentas, de jogos e robótica, por exemplo, também utilizam a linguagem visual em blocos em conjunto com sua metodologia principal. Alguns exemplos disso são: *Blocky Games* (*Blockly Games 2024*), *CodeCombat Junior* (*CodeCombat 2024*) e *Lego Mindstorms* (*Lego Mindstorms 2024*).

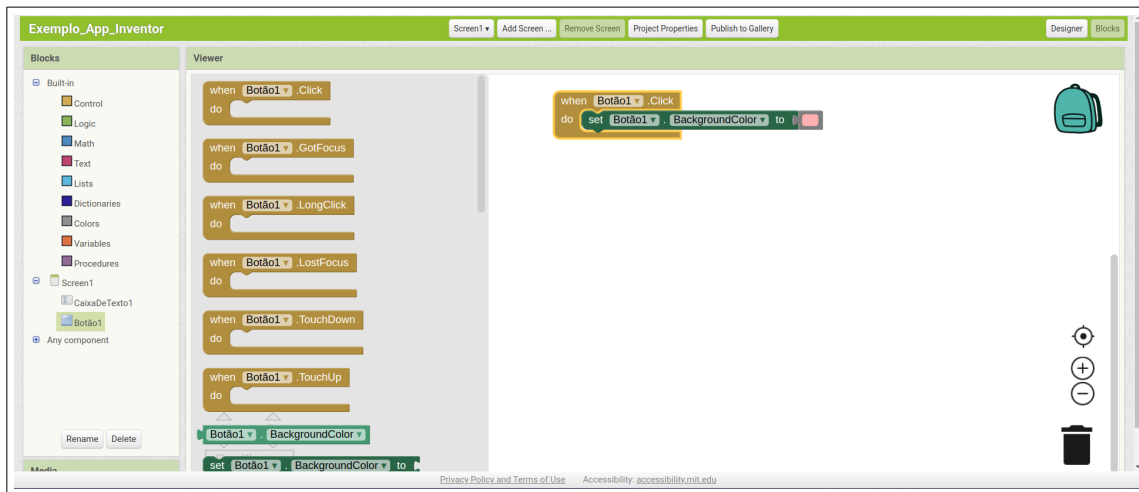


Figura 1.3: Interface do usuário da ferramenta App Inventor para criar a lógica de interação dos elementos do aplicativo (MIT App Inventor 2024).

1.2 Jogos digitais

A utilização de jogos digitais também pode estimular o pensamento computacional no ensino de programação. No geral, jogos fazem parte do cotidiano das crianças, além de aumentarem o engajamento e a motivação dos usuários. Um exemplo de jogo voltado para o ensino de programação é o *CodeCombat* (*CodeCombat* 2024), que permite que estudantes joguem e escrevam código utilizando uma linguagem baseada em texto com a sintaxe parecida com linguagens de programação. O jogo é gratuito e de código aberto, estando disponível em mais de 60 idiomas. Porém, a tradução para o português, muitas vezes, não é apresentada corretamente. A Figura 1.4 apresenta a interface utilizada no *CodeCombat*.



Figura 1.4: Interface do usuário do jogo CodeCombat (*CodeCombat* 2024).

Muitos desses jogos exploram comandos para orientar personagens, movendo-os para determinadas direções, e para realizar ações, utilizando de forma lúdica comandos com sintaxes muito similares a de funções das linguagens de programação.

1.3 Kits de robótica

Os kits de robótica combinam o uso de *hardware* com o de *software* para o ensino de lógica de programação. A abordagem envolve a utilização de placas com microcontroladores e a montagem de circuitos ou robôs para realizarem determinadas tarefas como, por exemplo, um circuito ligado a lâmpadas de LED que poderão ser acesas ou um robô que irá andar alguns centímetros sobre o chão. O *software*, então, é utilizado para criar um programa que determina os comandos de execução dessas ações.

Um kit muito utilizado devido ao seu menor custo é o do Arduino, que visa o controle de dispositivos, como LED e motores, ou a medição de variáveis, como temperatura e luminosidade. Ele requer conhecimentos em eletrônica por parte dos docentes e sua programação é feita através de um ambiente de desenvolvimento próprio baseado na linguagem de programação C (KLINCZAK e PAULA PINTO, 2024; MEDEIROS e WÜNSCH, 2019). Para facilitar a utilização deste tipo de ensino, algumas iniciativas criaram *softwares* baseados na linguagem visual em blocos do *Scratch* para escrever programas (Figura 1.5).

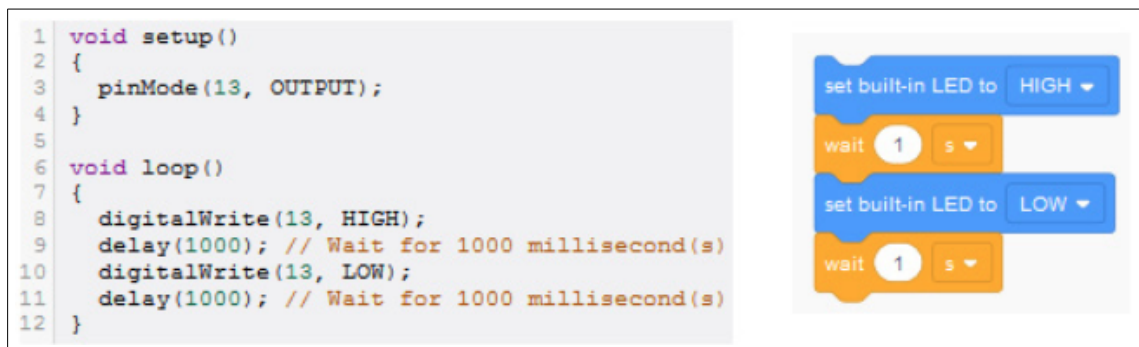


Figura 1.5: Exemplo da utilização da linguagem do Scratch para programação do Arduino (MEDEIROS e WÜNSCH, 2019).

Outro exemplo com um custo mais elevado é o Kit educativo *Lego Mindstorms*, o qual permite construir e programar um robô que pode andar, falar e pensar (Lego Mindstorms 2024). Sua montagem não requer conhecimentos de eletrônica, bastando apenas encaixar suas peças sem a necessidade de outras ferramentas (Figura 1.6). Também possui um ambiente próprio de programação visual, disponibilizado no site do fabricante.

1.4 Computação desplugada

A Computação desplugada envolve atividades lúdicas que não necessitam do uso de computadores e podem abordar tópicos que ensinam conceitos de Computação, podendo ser realizadas por pessoas de quaisquer faixas etárias (BRACKMANN, 2017). Muitas vezes, as atividades envolvem ações de recortar, dobrar, colar, desenhar, pintar, resolver problemas,



Figura 1.6: Panorama o kit educativo Lego Mindstorms EV3 (Lego Mindstorms 2024).

e os estudantes podem trabalhar em conjunto, promovendo habilidades de criatividade e colaboração. A Figura 1.7 apresenta a descrição de um exemplo criado por BRACKMANN (2017) em sua pesquisa sobre o desenvolvimento do pensamento computacional por meio de atividades desplugadas na educação básica.

DESCRIÇÃO
Material necessário: • Um tabuleiro com os personagens da Turma da Mônica • Uma folha de resposta Objetivo: exercitar prioritariamente os pilares de Reconhecimento de Padrão e Algoritmos através da busca por trajetos entre dois pontos (personagens) e aprender uma forma de escrever resumidamente os mesmos comandos. Instruções: • Entregar uma folha para cada estudante • O objetivo é encontrar o menor caminho entre o ponto inicial (personagem 1) e o ponto final (personagem 2) descrito no lado esquerdo. • Registrar a rota escolhida através de flechas (instruções), indicando como o personagem deve se deslocar pelo tabuleiro, na linha indicada como “A”; • Após finalizados todos os trajetos “A”, os estudantes devem então abreviar suas instruções com o uso de multiplicadores (2x, 3x, 4x, etc.) na linha “B” de cada trajeto. Por exemplo: →→→→→↑↑↑↑↑↑↑↑↑↑ pode ser compactado como 5x→7x↑← O personagem não pode sobrepor a árvore durante o caminho. O rio não pode ser atravessado em qualquer ponto, neste caso deve-se usar a ponte.

Figura 1.7: Descrição de um exemplo de atividade desplugada (BRACKMANN, 2017).

1.5 Simulações interativas

Simulações interativas utilizam elementos visuais do dia-a-dia para representar as abstrações envolvidas nos conceitos de diversas ciências, os quais não podemos ver ou manipular. Elas auxiliam no aprendizado desses tópicos, uma vez que os estudantes conseguem ver e manipular as interações (PRICE *et al.*, 2018). Entretanto, ainda não existem muitas ferramentas que utilizem simulações interativas para o ensino de conceitos de lógica de programação. Em particular, não encontramos nenhuma ferramenta que se assemelhe a abordagem utilizada, de maneira bem-sucedida, pela plataforma PhET.

FERNANDES *et al.* (2012) criaram e avaliaram um objeto de aprendizagem na forma de animações e simulações no tema de variáveis de lógica de programação. O protótipo apresentava três partes: contextualização do conceito (sem interatividade), simulação de teste e fluxo de dados e uma atividade de fixação (Figura 1.8). Esse objeto de aprendizagem foi, então, avaliado por alunos e professores e os resultados obtidos demonstraram o potencial dessa abordagem. Porém, esse protótipo não foi encontrado disponível no link indicado no artigo.

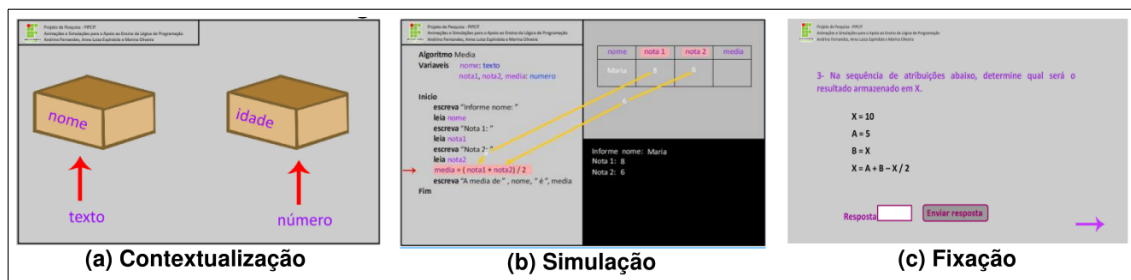


Figura 1.8: Ilustração do objeto de aprendizagem de variáveis desenvolvido por FERNANDES *et al.* (2012).

Um outro exemplo encontrado, foi a ferramenta *Drone Blocks* (Drone Blocks 2024), um simulador de drones com programas projetados para o ensino de fundamentos de Ciência da Computação, entre outras finalidades não relacionadas a educação. A interface da simulação criada para o ensino (Figura 1.9) é muito similar a do *Scratch*, a ferramenta utiliza a linguagem visual em blocos para construir instruções para o drone dentro de um cenário de sala de aula. Observamos como limitações dessa ferramenta o fato dela se restringir a simulações no contexto do drone na sala de aula e estar disponível apenas no idioma Inglês.

Dessa maneira, o MVP proposto neste projeto tem um caráter inovador, uma vez que não foram encontradas, até o momento, outras ferramentas com propostas similares. Além disso, ele difere das demais abordagens apresentadas, pois na maioria delas o usuário constrói um código para visualizar a execução dele em forma de jogos, animações, histórias e aplicações. Já no protótipo proposto, há a interação com os elementos visuais da simulação e o pseudocódigo correspondente pode ser observado através dessas interações. Ainda, optamos por utilizar o pseudocódigo ao invés da linguagem de programação visual em blocos a fim de promover o contato com a estrutura real de um código fonte.

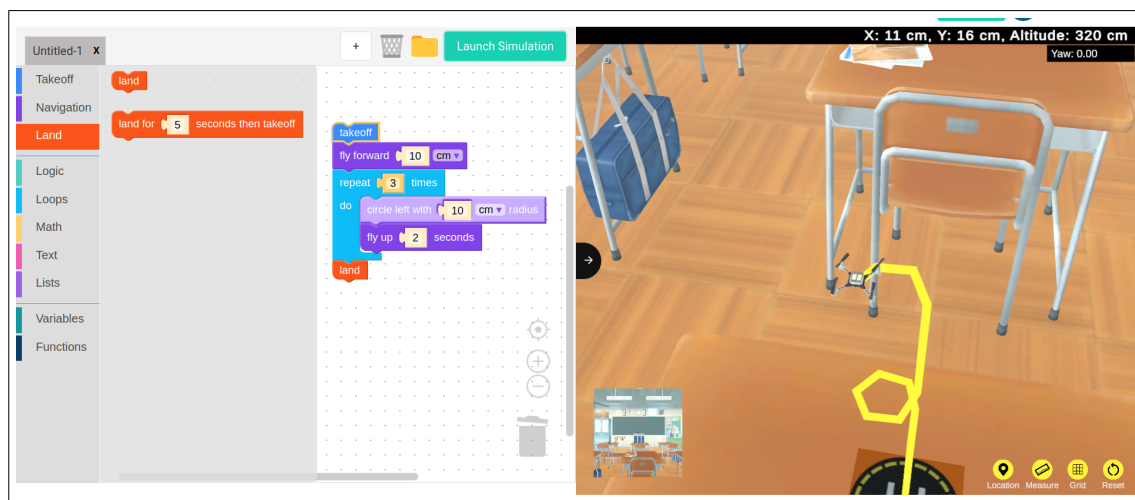


Figura 1.9: Interface do usuário da simulação Drone Blocks (Drone Blocks 2024).

Capítulo 2

Metodologia de pesquisa

Para criar e avaliar o MVP proposto, seguimos a metodologia de pesquisa de *Design Science*, que de forma geral envolve a elaboração de artefatos para solucionar problemas em um contexto. Neste Capítulo, explicamos o que é a *Design Science Research* e como ela foi conduzida neste projeto.

Design Science Research

A *Design Science Research* (DSR) é uma metodologia de pesquisa utilizada na elaboração de artefatos com propósitos práticos, constituindo um processo de resolução de problemas de domínio, em que o resultado deve ser avaliado pelo seu valor e utilidade. Esse paradigma é aplicado em diferentes áreas de pesquisa, como Sistemas de Informação, Gerenciamento de Negócios, Engenharia de *Software*, etc, podendo ser instanciado em variantes muito diferentes (DRESCH *et al.*, 2015; RUNESON *et al.*, 2020).

HEVNER *et al.* (2004) definiram sete diretrizes para a realização de DSR na área de Sistemas de Informação, que têm como princípio a construção e aplicação de um artefato para adquirir conhecimento sobre um problema de design e sua solução. Assim, uma pesquisa em *Design Science* requer a criação de um artefato com propósito para tecnologia da informação e com caráter inovador (diretriz 1), o qual deve considerar a solução de um problema de negócio relevante (diretriz 2). Além disso, o seu design deve ser avaliado rigorosamente considerando sua utilidade, qualidade e eficácia (diretriz 3). O artefato também deve ser inovador, proporcionando contribuições claras e verificáveis para pesquisa (diretriz 4), e sua construção e avaliação devem ser feitas de forma rigorosa (diretriz 5). O processo de design do artefato deve ser conduzido como um processo de pesquisa de uma solução efetiva para um problema (diretriz 6). Por fim, os resultados da pesquisa em *Design Science* devem ser comunicados de forma eficaz (diretriz 7).

WIERINGA (2014) estendeu a definição de *Design Science*, apresentando diretrizes para realizar pesquisas em Sistemas de Informação e Engenharia de *Software*. Para o autor, a interação entre o artefato e o contexto do problema contribui para chegar à solução do problema. Dessa forma, um projeto de DS itera sobre as atividades de design e investigação. A atividade de design é decomposta em três tarefas, designadas como ciclo de design. Esse

ciclo está inserido em um outro, de engenharia, no qual temos o resultado do ciclo de design sendo introduzido no contexto real e avaliado. As etapas de cada um dos ciclos estão descritas a seguir:

Ciclo de engenharia:

- **Ciclo de design:**
 - **Investigação do problema:** qual problema deve ser investigado e por quê?
 - **Design do tratamento (*Treatment design*):** projeto de um ou mais artefatos para tratar o problema.
 - **Validação do tratamento (*Treatment validation*):** análise para validar se esse projeto contribui para o tratamento do problema, caso seja implementado.
- **Implementação do tratamento (*Treatment implementation*):** tratamento do problema com um dos artefatos projetados.
- **Avaliação da implementação (*Implementation evaluation*):** avaliação do sucesso do tratamento. Ao final dessa etapa, podemos ter uma nova iteração no ciclo de engenharia.

O autor também destaca que projetos de pesquisa em *Design Science* estão relacionados apenas às três etapas do ciclo de *design*.

RUNESON *et al.* (2020) define etapas similares para o ciclo de engenharia, envolvendo a conceitualização do problema, o projeto da solução e a validação empírica. Os autores explicam que a *Design Science* abrange duas dimensões principais: problema-solução e teoria-prática. Eles descrevem as atividades de pesquisa que são realizadas de forma iterativa através das duas dimensões, são elas:

- **Conceitualização do problema:** descrição do problema.
- **Design da solução:** mapeamento do problema para uma solução geral.
- **Abstração:** identificação de decisões de *design* importantes para uma solução válida dentro de um escopo definido.
- **Instanciação:** implementação do artefato em contexto.
- **Validação empírica:** avaliação de como a solução implementada abordou o problema.

Ademais, projetos de pesquisa em *Design Science* devem considerar dois fatores importantes: a relevância da pesquisa para entidades na resolução de problemas reais e o rigor para que a pesquisa seja considerada válida e confiável, contribuindo para uma determinada área. Dessa forma, a DSR é importante tanto para a produção de conhecimento científico como para a resolução de problemas reais (DRESCH *et al.*, 2015; RUNESON *et al.*, 2020).

Portanto, a *Design Science* aborda problemas gerais através do estudo de instâncias específicas de problemas no contexto de pesquisa. Para realizar este Trabalho de Formatura, seguimos a metodologia de *Design Science Research* em Engenharia de Software proposta

por WIERINGA (2014), englobando o ciclo de *design* e de engenharia. Nas seções a seguir, descrevemos as etapas de cada ciclo referentes a este trabalho.

2.1 Ciclo de *design*

2.1.1 Investigação do problema

O problema investigado neste estudo foi o de uso de simulações interativas no ensino de conceitos de lógica de programação para crianças. Como apresentado no Capítulo 1, diferentes metodologias são adotadas para isso, utilizando recursos didáticos variados, sendo as linguagens visuais as mais utilizadas. Dessa maneira, investigamos o uso de simulações interativas para o ensino de Computação, uma vez que elas podem facilitar a visualização de ideias abstratas envolvidas nos conceitos de programação, utilizando exemplos concretos para representá-las. Ademais, essa metodologia se mostrou eficaz em outras ciências, com o uso da ferramenta PhET. Portanto, queremos expandi-la e testá-la em outros ambientes de aprendizado, como na área de Ciência da Computação.

Para investigar o problema, primeiramente, realizamos uma pesquisa sobre o estado atual do ensino de Computação para crianças em escolas no Brasil e no mundo. Em seguida, fizemos uma busca extensiva dos diversos tipos de ferramentas de apoio pedagógico existentes e similares a proposta neste trabalho, procurando entender as principais diferenças entre elas. Grande parte dessa pesquisa foi realizada durante o desenvolvimento do projeto de Iniciação Científica (IC) intitulado “Uma Ferramenta de Simulações Interativas para Ensino de Computação para Crianças”, realizado entre outubro de 2022 e outubro de 2023. O resumo dos principais resultados dessa investigação se encontra na Introdução e no Capítulo 1 desta monografia.

2.1.2 Projeto do artefato

O artefato projetado como tratamento do problema é um MVP de uma ferramenta de simulações interativas de conceitos de lógica de programação destinada a crianças do ensino fundamental. Um MVP (*Minimum Viable Product*) ou produto mínimo viável é um produto com funcionalidades suficientes para ser utilizado por usuários iniciais que possam fornecer *feedback* com o intuito de validar uma ideia.

No ensino da Computação, alguns conceitos introdutórios são fundamentais para o aprendizado da lógica de programação, tais como variáveis, entrada e saída de dados, operadores lógicos e aritméticos, condicionais, laços de repetição, funções, vetores, matrizes, entre outros. Desse modo, para testar a viabilidade do protótipo proposto, seu projeto deve compreender um conjunto mínimo desses conceitos, a fim de verificar o entendimento deles pelos usuários.

Assim, para projetar os artefatos ou protótipos do MVP, primeiramente, definimos alguns requisitos mínimos, listados a seguir:

- O MVP deve apresentar uma tela com uma área de trabalho e uma opção de ajuda com uma documentação explicativa de uso;

- A área de trabalho deve apresentar um espaço fixo para a simulação e outro para o pseudocódigo correspondente;
- A simulação deve envolver os seguintes conceitos de lógica de programação: variáveis, entrada, saída, operadores (aritméticos, lógicos e de comparação), condicionais e laços de repetição;
- A simulação deve apresentar um número limitado de interações possíveis com o usuário;
- A simulação deve encorajar os usuários a explorá-la livremente, com controles intuitivos e uma interface que possibilite boa usabilidade;
- O pseudocódigo deve ser gerado automaticamente conforme as interações com a simulação vão ocorrendo.

Além disso, também definimos e descrevemos os conceitos de lógica de programação que foram simulados e a sintaxe do pseudocódigo gerado. A partir dos protótipos iniciais, estudamos ideias similares e validamos os artefatos projetados para analisar alterações necessárias e as melhores opções de implementação. O Capítulo 3 apresenta em detalhes o desenvolvimento dos protótipos propostos.

2.1.3 Validação do artefato

A validação dos artefatos foi realizada com base nas opiniões de profissionais de ensino de Computação e matemática. Foram consultados: a professora Dra. Kelly Braghetto do Departamento de Ciência da Computação do IME, minha orientadora no trabalho de formatura, que também coordena o projeto CodificADAs USP, oferecendo cursos introdutórios de programação voltados para meninas do ensino médio; a estudante Victoria Nóvoa, do curso de Licenciatura em Educomunicação da USP, instrutora de tecnologia educacional de crianças no Colégio Santa Cruz; e o professor Henri Silva da Escola de Aplicação da Faculdade de Educação da USP, que leciona matemática para alunos do 8º ano do Ensino Fundamental II.

Em conversas com os avaliadores, foi possível obter *feedbacks* para verificar a usabilidade e utilidade do MVP a partir dos protótipos projetados, possibilitando o refinamento deles e a escolha de um artefato para implementação, além da definição do público alvo específico para testar a aplicação. No Capítulo 3, descrevemos o processo de validação e as alterações realizadas.

2.2 Ciclo de engenharia

Após a execução das etapas anteriores, completando uma iteração no ciclo de *design*, realizamos os demais passos do ciclo de engenharia, descritos a seguir.

2.2.1 Implementação da simulação

O MVP da simulação foi desenvolvido utilizando o *framework* de código aberto Vue.js,¹ que é muito utilizado para criar aplicações de página única (*single-page applications*), com a intenção de agilizar o desenvolvimento, permitindo a utilização de soluções existentes e reduzindo a necessidade de escrever códigos do zero. O projeto criado com o *framework* foi inicializado para ser utilizado com a linguagem de programação Typescript. Em conjunto com o Vue.js, utilizamos o Vuetify,² um *framework* de componentes de UI de código aberto, que contém diversos *layouts* prontos e componentes dinâmicos.

O código desenvolvido ao longo do projeto, com a implementação do artefato, está disponível no seguinte repositório: <https://github.com/mariliatd/logic-sims-mvp>, sob a licença MPL-2.0 (*Mozilla Public License Version 2.0*). Para desenvolver o MVP, utilizamos a organização em componentes que o *framework* Vue possibilita. Dessa maneira, foi possível definir cada conceito de lógica de programação como um componente isolado a ser reutilizado dentro de um *template* de simulação, conforme necessário. Também separamos os componentes referentes aos elementos visuais da simulação do pseudocódigo, criando maior modularidade no código.

Assim, com o artefato implementado, criamos uma página para disponibilizar o MVP, através da ferramenta de versionamento de códigos GitHub, para a aplicação e avaliação da ferramenta em sala de aula.

2.2.2 Avaliação da simulação

A avaliação do MPV da ferramenta de simulação foi realizada utilizando um formulário de usabilidade e uma atividade para verificação do aprendizado dos conceitos de lógica de programação. Primeiramente, por meio do questionário de usabilidade, métrica comum na avaliação de um protótipo ou sistema, o MVP foi analisado considerando a facilidade de uso, de aprendizado e satisfação. Em particular, utilizamos o questionário SUS (*System Usability Scale*), aplicado aos usuários finais.

Apesar de não haver medidas absolutas de usabilidade, é possível utilizar escalas gerais para comparar usabilidade em determinados contextos. O SUS representa uma escala de usabilidade com 10 itens que pode ser utilizada para avaliar sistemas (BROOKE *et al.*, 1996). Ele é baseado na escala Likert, a qual contém afirmações e os avaliadores indicam o grau de concordância ou discordância com cada uma, que varia de 1 a 5. Recomenda-se que as respostas para cada item sejam registradas de imediato, sem que os respondentes levem muito tempo pensando nelas.

Para obter o *score* de usabilidade do sistema a partir do questionário SUS, primeiramente, a contribuição de cada item é normalizada para uma escala de 0 a 4. Em seguida, a soma das pontuações dos itens é multiplicada por 2,5, obtendo um *score* em uma escala de 0 a 100. Entretanto, os autores não revelaram como analisar esta pontuação.

BANGOR *et al.* (2008) e BANGOR *et al.* (2009) realizaram um estudo onde avaliaram a

¹ <https://vuejs.org/>

² <https://vuetifyjs.com/en/>

usabilidade de diversos produtos e serviços utilizando o questionário SUS. Eles analisaram a média de pontuação do SUS em 273 estudos com cerca de 3.500 pesquisas individuais. Os autores realizaram uma interpretação dessa pontuação, comparando os quartis dos *scores*. Além disso, eles adicionaram uma afirmação ao questionário para avaliar a usabilidade do sistema através de uma escala de classificação de adjetivos, obtendo uma correlação entre eles e a média das pontuações. As médias que correspondem a cada adjetivo são as seguintes:

1. pior imaginável (*worst imaginable*): 12.5
2. horrível (*awful*) : 20.3
3. ruim (*poor*): 35.7
4. ok (*ok*): 50.9
5. bom (*good*): 71.4
6. excelente (*excellent*): 85.5
7. melhor imaginável (*best imaginable*): 90.9

Os autores ainda expressaram preocupações com o uso do adjetivo “ok”, uma vez que ele sugere uma experiência aceitável com o sistema, enquanto a média de pontuações na faixa dele sugere deficiências. Eles acreditam que o termo “razoável” (*fair*) seria melhor indicativo da usabilidade percebida pelos usuários.

Ademais, o SUS é considerado um questionário robusto e confiável, tendo sido utilizado em diversos projetos de pesquisa e avaliações na indústria (BROOKE *et al.*, 1996). Embora não tenha sido projetado considerando necessidades específicas de compreensão por crianças, o questionário tem sido utilizado em vários estudos de testes e avaliações de ferramentas com crianças de diversas faixas etárias com sucesso (i.e. WROŃSKA *et al.*, 2015; DEXHEIMER *et al.*, 2017; SÁNCHEZ-MORALES *et al.*, 2020; TASFIA *et al.*, 2023).

PUTNAM *et al.* (2020) realizaram uma adaptação e teste do SUS com crianças na faixa etária de 7 a 11 anos. Os autores adaptaram o questionário, com o auxílio de professores da educação básica, em um contexto de aplicativos móveis de jogos focados no ensino de programação e pensamento computacional. As afirmações foram ainda modificadas pensando na separação de dois grupos de faixa etária, entre 7 e 8 anos e entre 9 e 11 anos. A Tabela 2.1 mostra os enunciados do SUS original e as adaptações propostas para os dois grupos mencionados, em inglês. Além das simplificações das afirmações, os autores também utilizaram uma representação visual da escala de Likert (Figura 2.1), sugerida pelos docentes participantes do experimento. Os resultados obtidos nos experimentos mostraram que o questionário modificado juntamente com a escala visual foi compreendido pelas crianças participantes, necessitando apenas de clarificações mínimas. Eles ainda sugeriram alterações nas afirmações 6, 8 e 10 para melhorar a compreensão e a confiabilidade delas.

Dessa forma, utilizamos uma adaptação do questionário SUS para obter o *feedback* dos estudantes sobre o MVP para avaliá-lo em relação a sua usabilidade. Ademais, através da atividade de aprendizado dos conceitos de programação, analisamos o potencial do ensino de Computação utilizando simulações interativas. No Capítulo 5, descrevemos a

Afirmativa	SUS original	SUS adaptado: Grupo 9-11 anos	SUS adaptado: Grupo 7-8 anos
1	I think that I would like to use this system frequently.	If I had this [app] on my iPad, I think that I would like to play it a lot.	I would like to play [app] a lot more.
2	I found the system unnecessary complex.	I was confused many times when I was playing [app].	[app] was hard to play.
3	I thought the system was easy to use.	I thought [app] was easy to use.	I thought [app] was easy to use.
4	I think that I would need the support of a technical person to be able to use this system.	I would need help from an adult to continue to play [app].	I would need help to play [app] more.
5	I found the various functions in this system were well integrated.	I always felt like I knew what to do next when I played [app].	I knew what to do next when I played [app].
6	I thought there was too much inconsistency in the system.	Some of the things I had to do when playing [app] did not make sense.	Some things in [app] made no sense.
7	I would imagine that most people would learn to use this system very quickly.	I think most of my friends could learn to play [app] very quickly.	[app] would be easy for my friends to learn.
8	I felt the system was cumbersome to use.	Some of the things I had to do to play [app] were kind of weird.	To play [app] I had to do some weird things.
9	I felt very confident using the system.	I was confident when I was playing [app].	I was proud of how I played [app].
10	I needed to learn a lot of things before I could get going with this system.	I had to learn a lot of things before playing [app] well.	There was a lot to learn to play [app].

Tabela 2.1: Afirmativas em inglês do questionário de avaliação de usabilidade de sistemas SUS e adaptações propostas por *PUTNAM et al. (2020)* para crianças entre 7 e 11 anos, dividida em dois grupos.

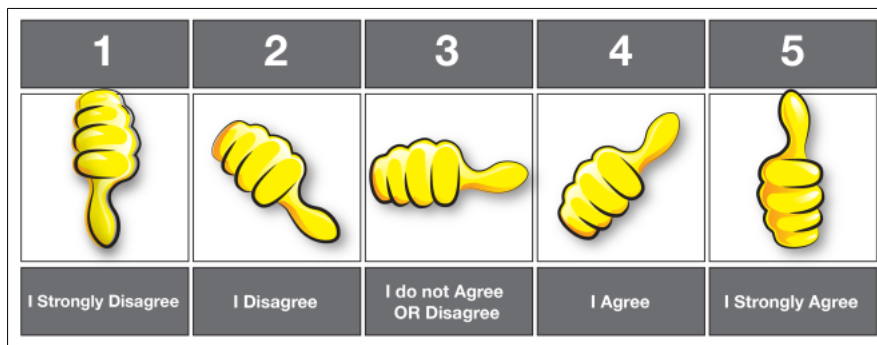


Figura 2.1: Representação visual da escala de Likert (*PUTNAM et al., 2020*).

atividade realizada em sala de aula com a aplicação dos dois formulários para avaliação do artefato implementado.

Capítulo 3

Criação e validação dos protótipos da simulação

A partir dos requisitos listados no Capítulo 2.1.2, criamos dois protótipos iniciais de simulações desenvolvidos no Figma,¹ uma plataforma *web* colaborativa para projetar interfaces. As Figuras 3.1 e 3.2 mostram esses protótipos.

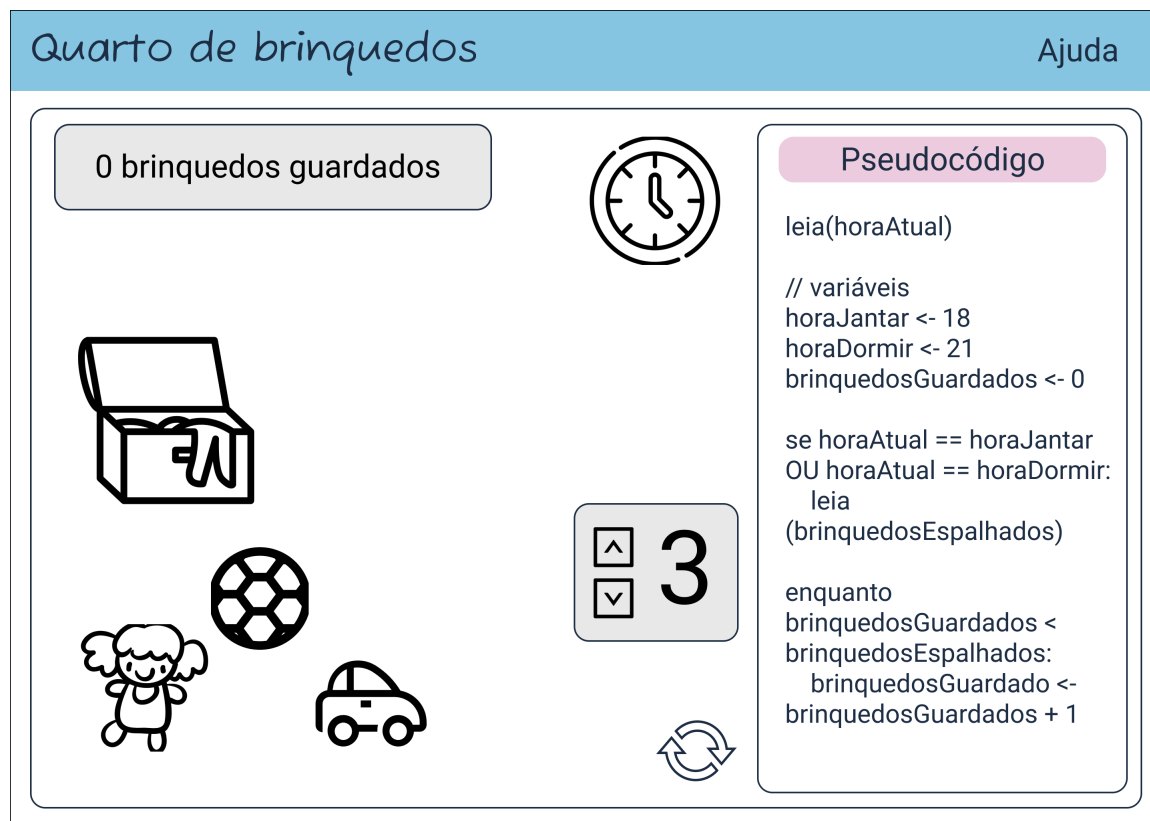


Figura 3.1: Protótipo inicial da simulação “Quarto de Brinquedos”.

¹ <https://www.figma.com/>

Na Figura 3.1, temos um cenário de quarto de brinquedos. O protótipo da simulação apresenta elementos visuais como um relógio de parede, um baú para guardar objetos e brinquedos espalhados pelo chão. Além disso, temos dois painéis, um controle para regular a quantidade de brinquedos espalhados e um contador que mostra quantos brinquedos foram guardados. Ainda, o protótipo apresenta um pseudocódigo associado à simulação.

A princípio, a ideia dessa simulação é apresentar duas interações possíveis: alterar a hora no relógio e guardar os brinquedos no baú de forma iterativa. Com isso, queremos abordar o conceito de variáveis, principalmente, com a ação de guardar os objetos em um contêiner; de entrada, com a informação recebida através da interação com o usuário ao alterar o horário; de operadores, na comparação de expressões simples e complexas com conectivos lógicos, e na operação de adição contida na ação de guardar os brinquedos; de condicionais, verificando se é o momento de organizar o quarto; e de laços de repetição, repetindo a ação de guardar cada objeto até que o quarto esteja organizado.



Figura 3.2: Protótipo inicial da simulação “Planejando a Festa”.

A Figura 3.2 apresenta um cenário de planejamento de uma festa. Nele há elementos visuais que representam itens que podem ser escolhidos para incorporar uma festa de aniversário. Ademais, temos um contador que mostra quantos itens referentes ao escolhido já estão preparados. Nessa simulação, o usuário pode ter duas interações: escolher um item para a festa e “prepará-lo” (incrementar um determinado item) até estar completo. Deste modo, nesse cenário, abordamos os conceitos de variáveis, por meio do total de objetos que devemos ter de acordo com a escolha do item; de entrada, a partir da escolha de um item para a festa; de operadores, na comparação de expressões simples, e na operação

de adição contida na ação de preparar um item; de condicionais, verificando qual item foi escolhido; e de laços de repetição, conforme incremento dos objetos até atingirem o total de acordo com cada elemento.

Com os protótipos iniciais projetados, foram realizadas algumas mudanças antes de apresentá-los, conforme sugestões da professora Kelly Braghetto. As Figuras 3.3 e 3.4 mostram as alterações sugeridas para os artefatos.

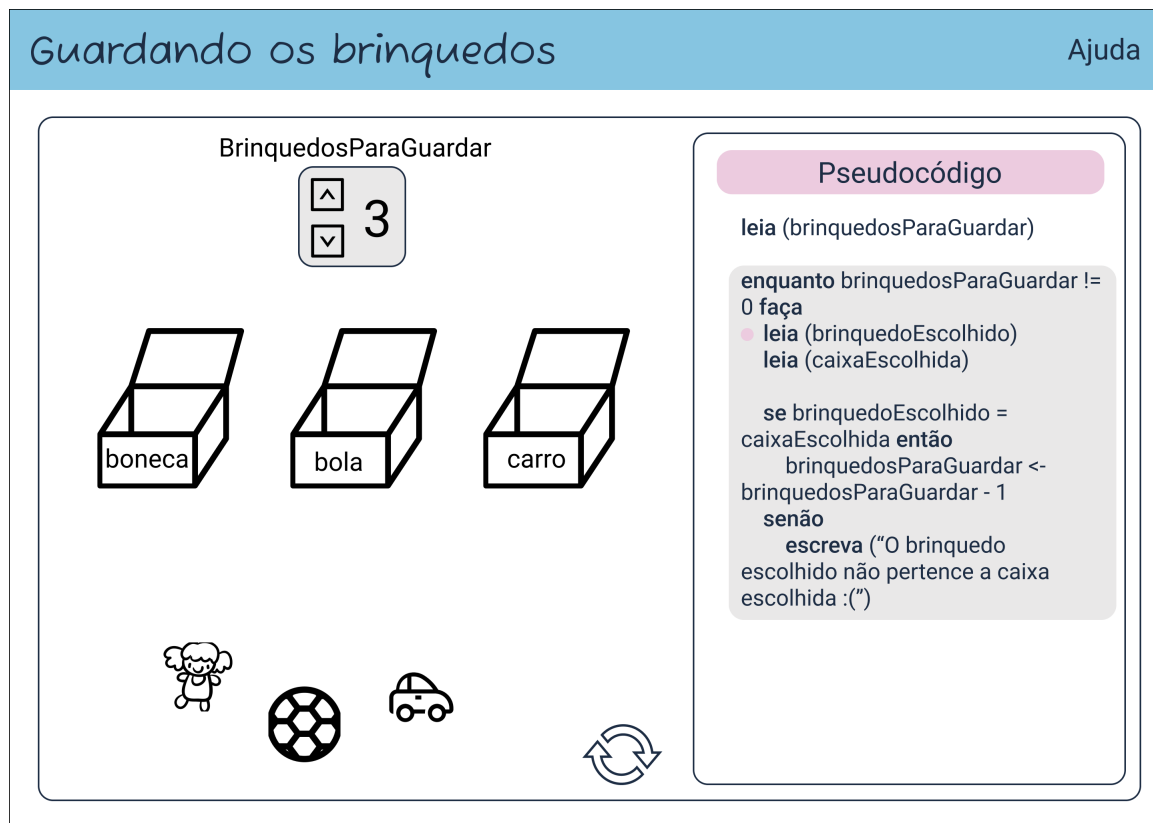


Figura 3.3: Protótipo da simulação “Guardando os Brinquedos” após validação.

Na Figura 3.3 podemos observar a nova simulação “Guardando os Brinquedos”, que teve sua lógica modificada para simplificar as interações com os elementos visuais e melhorar a usabilidade. Nele, há um contador que permite selecionar até três brinquedos para guardar em suas respectivas caixas. As interações possíveis nessa simulação são: incrementar o contador para escolher quantos brinquedos guardar e, após esta escolha, escolher brinquedo e caixa correspondente. A cada brinquedo guardado corretamente, o valor do contador deve ser subtraído de um. Nesse cenário, abordamos os conceitos de variáveis, na quantidade de brinquedos para guardar e nos itens (brinquedo e caixa) escolhidos; de entrada e saída, na leituras das variáveis e na escrita da mensagem quando o brinquedo e a caixa escolhida não correspondem; de operadores, na comparação de expressões simples e na operação de subtração contida na ação de guardar os brinquedos; de condicionais, verificando se o brinquedo e a caixa escolhida correspondem; e de laços de repetição, repetindo a ação de guardar cada objeto até não sobrar mais nenhum.

Já na nova versão da simulação “Planejando a Festa” (Figura 3.4), podemos observar a mudança no controle do incremento do contador que indica a quantidade de itens totais

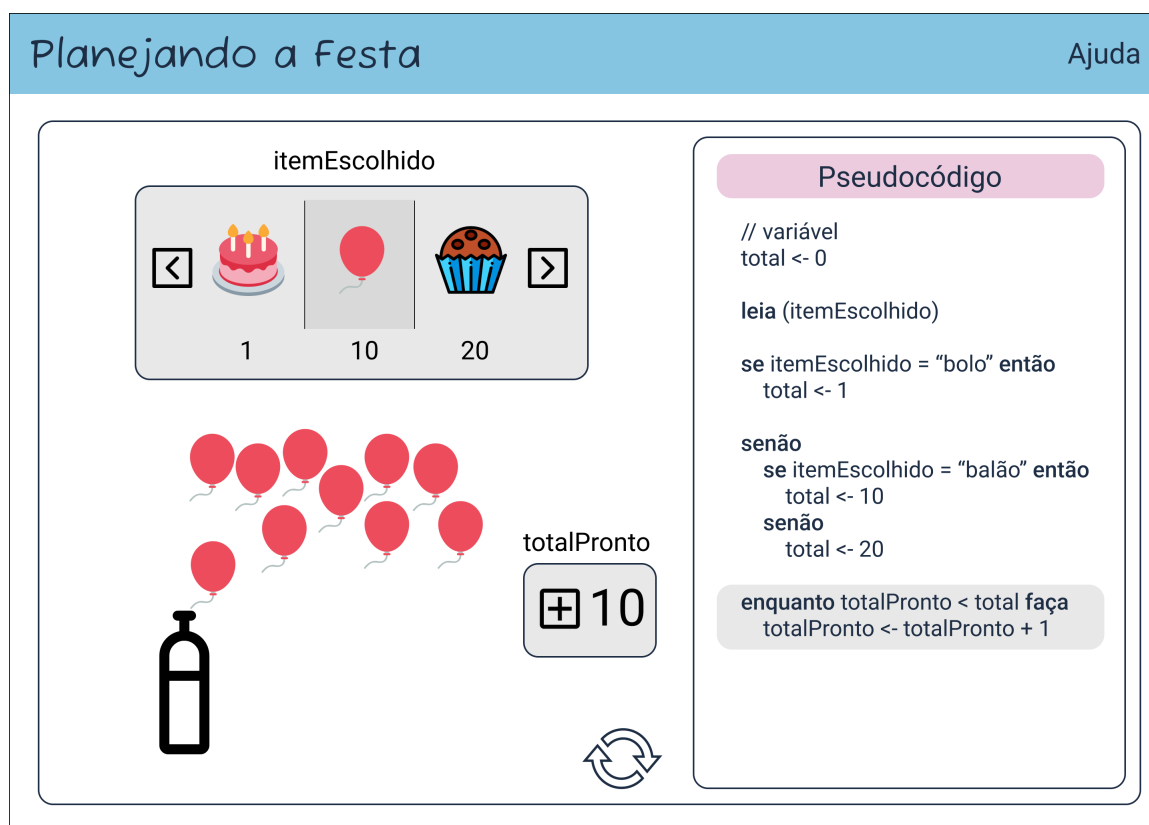


Figura 3.4: Protótipo da simulação “Planejando a Festa” após validação.

prontos. Além disso, no geral, as sugestões implementadas nos protótipos envolveram a melhoria nos nomes das variáveis, a incorporação do conceito de saída e o destaque na parte do pseudocódigo em que está ocorrendo a interação na simulação.

Em seguida, em conversas com os avaliadores, foi possível obter *feedbacks* para verificar a usabilidade e utilidade do MVP a partir dos protótipos projetados, possibilitando o refinamento deles e a escolha de um artefato para implementação. A simulação “Planejando a Festa” se mostrou mais interessante e com maiores possibilidades de interações e, por isso, foi escolhida como artefato a ser implementado. Ademais, o público alvo definido para a aplicação do MVP foram crianças no Ensino Fundamental II, a partir do 6º ano.

As demais sugestões recebidas pelos avaliadores levaram em conta, portanto, o protótipo escolhido e foram aplicadas diretamente na implementação do artefato. Dessa forma, a lógica da simulação “Planejando a Festa” foi alterada para melhorar a sua usabilidade e as interações com os elementos visuais. No Capítulo 4, descrevemos os detalhes da implementação com as modificações realizadas.

Capítulo 4

Implementação da simulação

O MVP da ferramenta de simulações interativas de conceitos de lógica de programação está disponível no link: <https://mariliatd.github.io/logic-sims-mvp/>. A tela inicial da simulação “Planejando a Festa” pode ser observada na Figura 4.1. Ela apresenta uma barra de navegação no topo da página, com o nome da simulação e um link de “ajuda”, o qual abre um diálogo contendo descrições dos conceitos de programação abordados. Abaixo dela, há o espaço da simulação que, inicialmente, apresenta um quadro com informações sobre a ferramenta, e o espaço do pseudocódigo à direita.

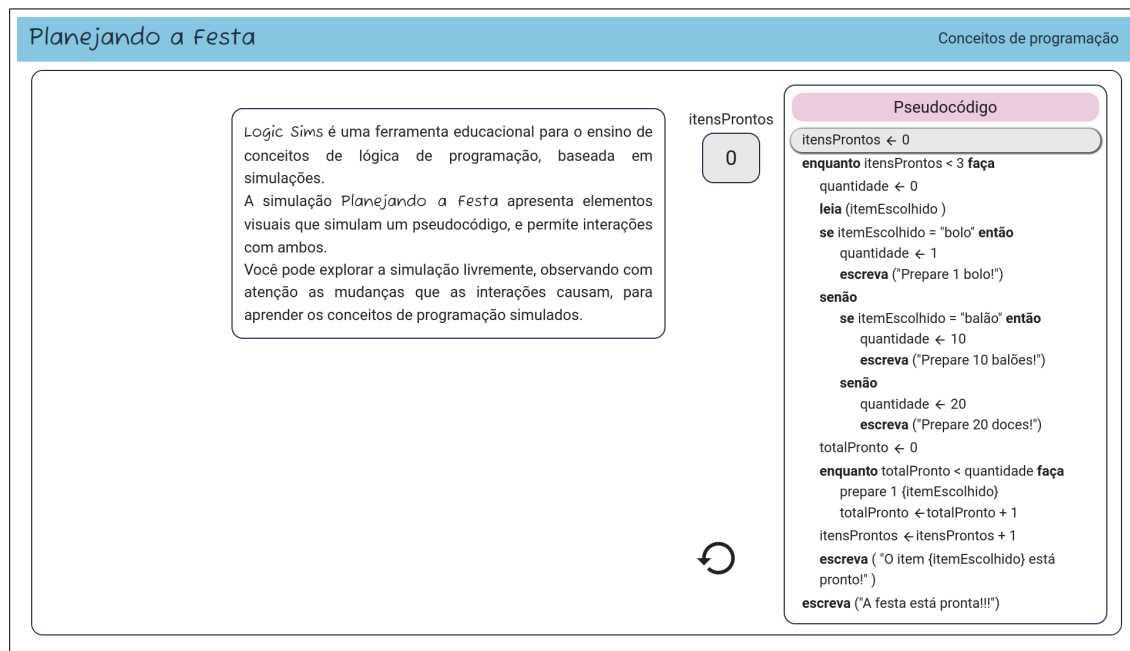


Figura 4.1: Tela inicial do MVP da simulação “Planejando a Festa”.

Nessa tela, é possível visualizar algumas das alterações sugeridas na etapa de validação dos protótipos. Essa nova versão da simulação também permite interações com o pseudocódigo, o qual tem as linhas destacadas com uma borda com uma animação que pisca quando é necessário interagir com ela. Esse recurso foi implementado para facilitar a visualização

da “execução” de cada passo do pseudocódigo, o qual foi modificado. Adicionamos à lógica da simulação um laço de repetição externo para realizar a iteração na preparação dos três itens disponíveis para compor a festa, que faltava no projeto do protótipo.

Assim, a simulação permite selecionar três itens em quantidades distintas e preparar cada um até atingir as quantidades correspondentes até que a festa esteja pronta. A seleção de um item pelo usuário representa a entrada ou leitura de um valor no pseudocódigo, que condiciona a quantidade de itens a serem preparados. Ao escolher um item ou finalizar o seu preparo e quando a festa está pronta são impressas mensagens na tela em uma janela representando a saída de dados no pseudocódigo, como mostra a Figura 4.2.

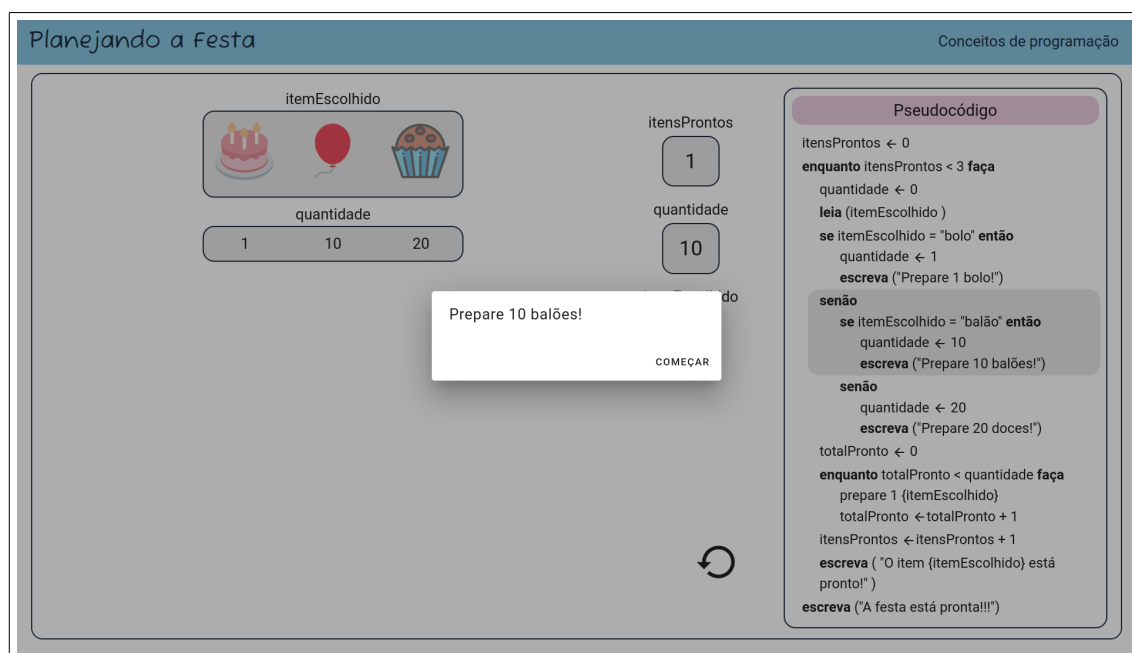


Figura 4.2: Representação visual do conceito de saída com a mensagem “impressa” na tela.

Quando o usuário seleciona um item para preparo, a escolha dos demais fica desabilitada, para que a interação com a simulação siga o fluxo do pseudocódigo e não permita que o usuário troque de item enquanto está dentro do laço de repetição interno, por exemplo. A Figura 4.3 mostra o cenário em que o item “bolo” foi selecionado e os demais estão sombreados, não sendo possível clicar neles. Ainda, após o preparo de um item, este fica desabilitado para escolha na iteração seguinte, limitando o preparo dos três itens (em qualquer ordem) para finalizar o preparo da festa e a simulação.

Na Figura 4.3 também podemos observar que cada variável está separada em um elemento visual próprio. Além disso, removemos o controle do incremento do contador que indica a quantidade total pronta de cada item, deixando essa interação apenas nas figuras de preparo de cada item (fogão, cilindro de gás e pratos) para representar o laço de repetição interno do pseudocódigo. Ainda, adicionamos o pseudocódigo para a execução das atividades de preparo do item escolhido dentro desse laço.

Outro recurso incorporado no pseudocódigo da simulação foi a adição de cores para diferenciar os conceitos de lógica de programação, o qual é comumente utilizado nas linguagens de programação em blocos. A escolha das cores foi feita através da ferramenta

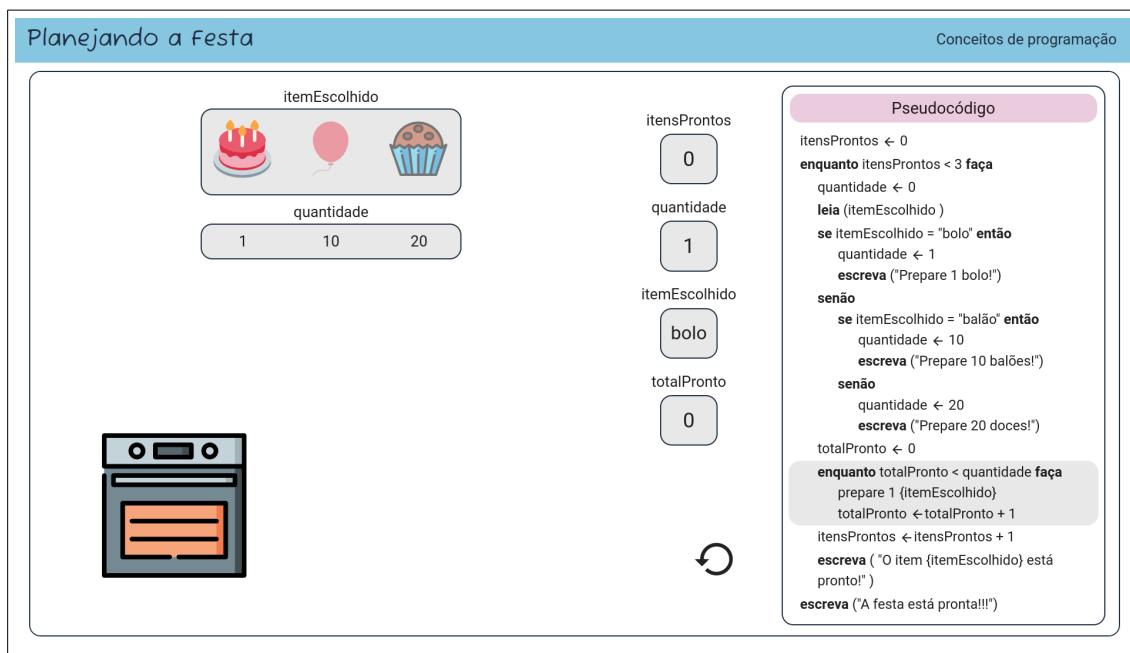


Figura 4.3: Escolhendo um item na simulação “Planejando a Festa”.

Coolors¹ que além de apresentar um gerador de paleta de cores, também disponibiliza um verificador de contraste. Ela segue as Diretrizes de Acessibilidade para Conteúdo Web (Web Content Accessibility Guidelines - WCAG), a qual define uma série de recomendações para tornar a web mais acessível (CALDWELL *et al.*, 2008). A categorização das cores foi feita da seguinte forma:

- Variáveis (amarelo)
- Entrada e saída (laranja)
- Operadores (verde)
- Condicionais (roxo)
- Laço de repetição (azul)

Além disso, a simulação mostra o nome de cada conceito de lógica de programação. Esse recurso pode ser visualizado ao passar o mouse em cima de cada conceito. A Figura 4.4 apresenta as alterações destacadas.

Por fim, o link para “Ajuda” na barra de navegação foi renomeado para “Conceitos de programação” e ele abre uma caixa de diálogo contendo definições de cada conceito, com figuras e exemplos (Figura 4.5). As definições contêm pseudocódigos de exemplo que seguem o esquema de cores descrito acima.

¹ coolors.co

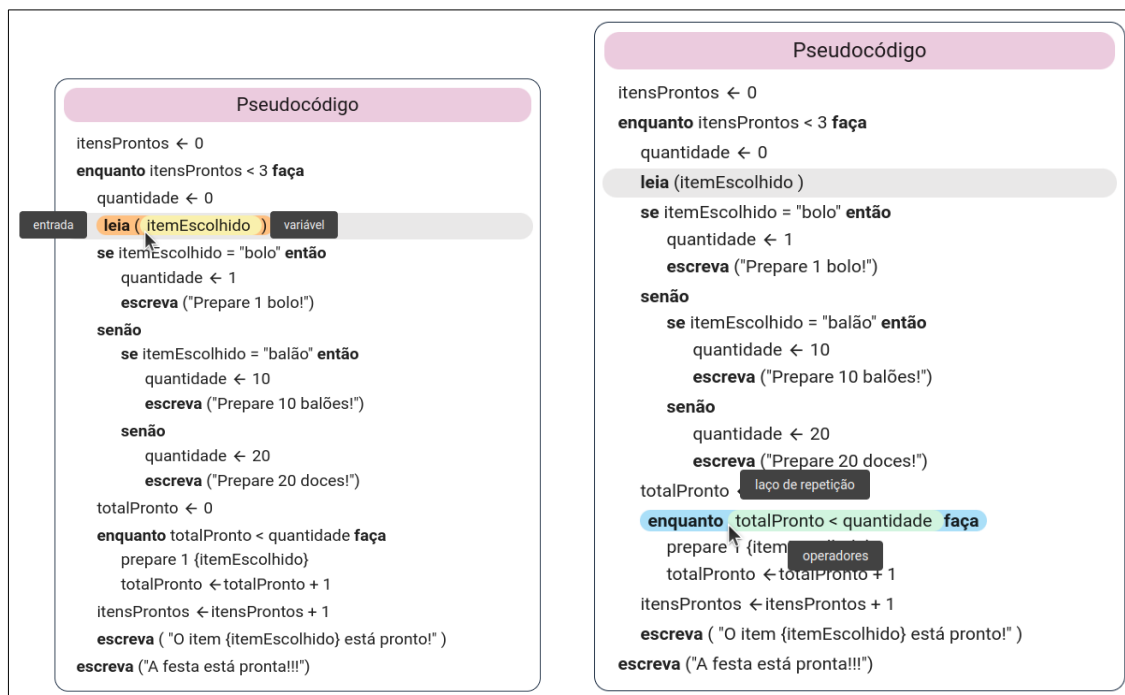


Figura 4.4: Destaques do pseudocódigo da simulação “Planejando a Festa” mostrando a separação em cores por conceito de lógica de programação e a indicação do nome deles ao passar o mouse por cima de cada um.

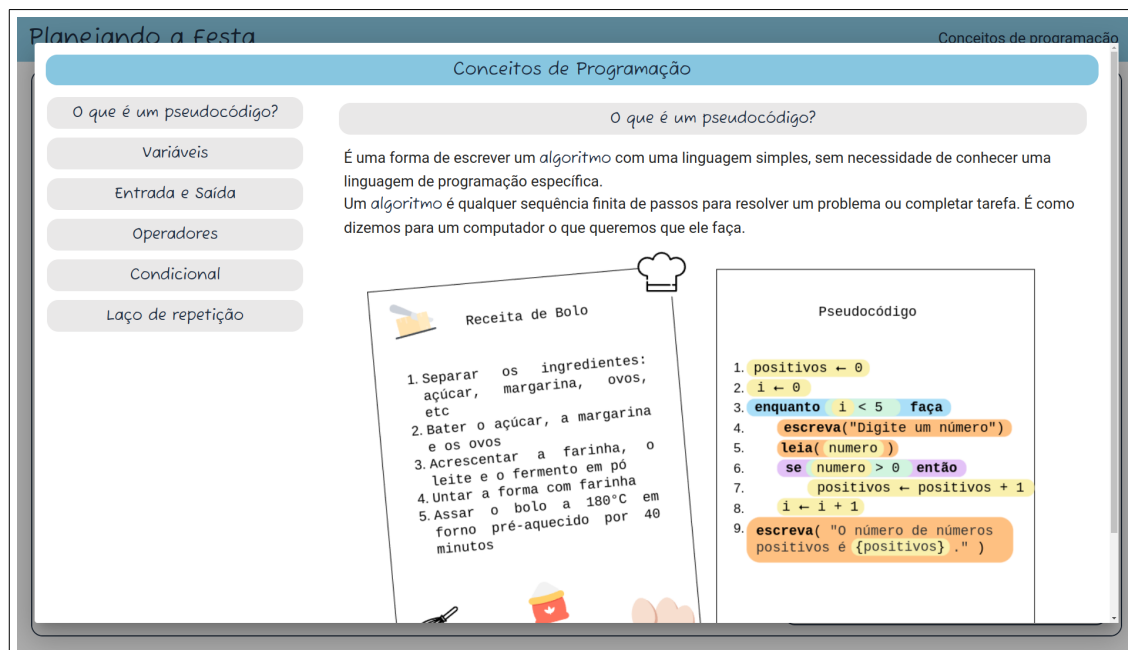


Figura 4.5: Caixa de diálogo contendo as definições dos conceitos de lógica de programação abordados na simulação “Planejando a Festa”. O menu lateral permite a escolha de um conceito, mostrando a sua definição e exemplos ao lado.

Capítulo 5

Avaliação da simulação

A avaliação da simulação por meio da escala de usabilidade (SUS) e da atividade de aprendizado foi realizada em sala de aula com os usuários finais. Para isso, realizamos uma atividade com a duração de uma aula de 50 minutos com uma turma de 26 alunos com idades entre 13 e 14 anos, do 8º ano do Ensino Fundamental II da Escola de Aplicação da Faculdade de Educação da USP (FEUSP). Durante a aula, contamos com o apoio do professor Henri Silva, que leciona matemática para a turma.

A atividade realizada em aula foi dividida em três partes: na primeira, apresentamos um exemplo de pseudocódigo e código em Python de um programa que, dados cinco números de entrada, calcula a quantidade de números pares e de números ímpares; na segunda parte, os alunos exploraram o MVP com a simulação de forma livre; e na terceira, eles preencheram dois formulários, o da escala de usabilidade e o da atividade de aprendizado, nessa ordem.

As 10 afirmações adaptadas do SUS e traduzidas para o português, apresentadas no formulário de usabilidade, estão enumeradas a seguir, enquanto a escala visual de Likert utilizada é mostrada na Figura 5.1.

1. Eu gostaria de explorar mais a simulação.
2. Eu fiquei confuso(a) muitas vezes enquanto explorava a simulação.
3. Eu achei que a simulação foi fácil de usar.
4. Eu precisaria de ajuda para conseguir explorar a simulação.
5. Eu sabia o que era preciso fazer quando explorei a simulação.
6. Algumas coisas que eu tive que fazer enquanto explorava a simulação não fizeram sentido.
7. Eu acho que a maioria dos meus amigos podem aprender a usar a simulação muito rápido.
8. Algumas das coisas que eu tive que fazer para explorar a simulação foram estranhas.
9. Eu me senti confiante enquanto explorava a simulação.

10. Eu tive que aprender muitas coisas antes de explorar a simulação.



Figura 5.1: Representação visual da escala de Likert utilizada no formulário de usabilidade.

Ainda, ao final do formulário foram feitas duas perguntas dissertativas sobre o que o usuário mais gostou na simulação e o que menos gostou, além de um espaço adicional para deixar quaisquer comentários ou sugestões sobre ela.

A atividade de aprendizado foi dividida em duas partes com 6 perguntas cada. Em cada pergunta, foi apresentado um trecho do pseudocódigo com um ou mais conceitos de lógica de programação e foi pedido para selecionar a opção que melhor representasse aquele pedaço de pseudocódigo, entre as seguintes opções: variáveis, entrada, saída, operadores, condicional e laço de repetição. A primeira parte da atividade continha perguntas sobre o pseudocódigo da simulação “Planejando a Festa”, explorada pelos estudantes. A Figura 5.2 mostra os trechos de pseudocódigo apresentados em cada questão. A segunda parte era referente à simulação “Guardando os Brinquedos”, a qual foi apresentada no formulário, para fornecer contexto para as questões, por meio de uma imagem (Figura 5.3). Essa simulação é uma versão modificada da criada na etapa de projeto dos artefatos, apresentada no Capítulo 3. A Figura 5.4 mostra os pedaços de pseudocódigo utilizados nessas questões.

O Capítulo 6 apresenta os resultados da avaliação do MVP feita a partir das respostas dos formulários.

```
se itemEscolhido = "bolo" então  
  quantidade ← 1  
  escreva ("Prepare 1 bolo!")  
senão  
  se itemEscolhido = "balão" então  
    quantidade ← 10  
    escreva ("Prepare 10 balões!")  
  senão  
    quantidade ← 20  
    escreva ("Prepare 20 doces!")
```

(a) Pseudocódigo referente à Pergunta 1

```
quantidade ← 0
```

(b) Pseudocódigo referente à Pergunta 2

```
enquanto totalPronto < quantidade faça  
  prepare 1 {itemEscolhido}  
  totalPronto ← totalPronto + 1
```

(c) Pseudocódigo referente à Pergunta 3

```
leia (itemEscolhido)
```

(d) Pseudocódigo referente à Pergunta 4

```
escreva ("A festa está pronta!!!")
```

(e) Pseudocódigo referente à Pergunta 5

```
itensProntos ← itensProntos + 1
```

(f) Pseudocódigo referente à Pergunta 6

Figura 5.2: Trechos de pseudocódigo apresentados em cada pergunta da atividade de aprendizado referente à simulação “Planejando a Festa”.

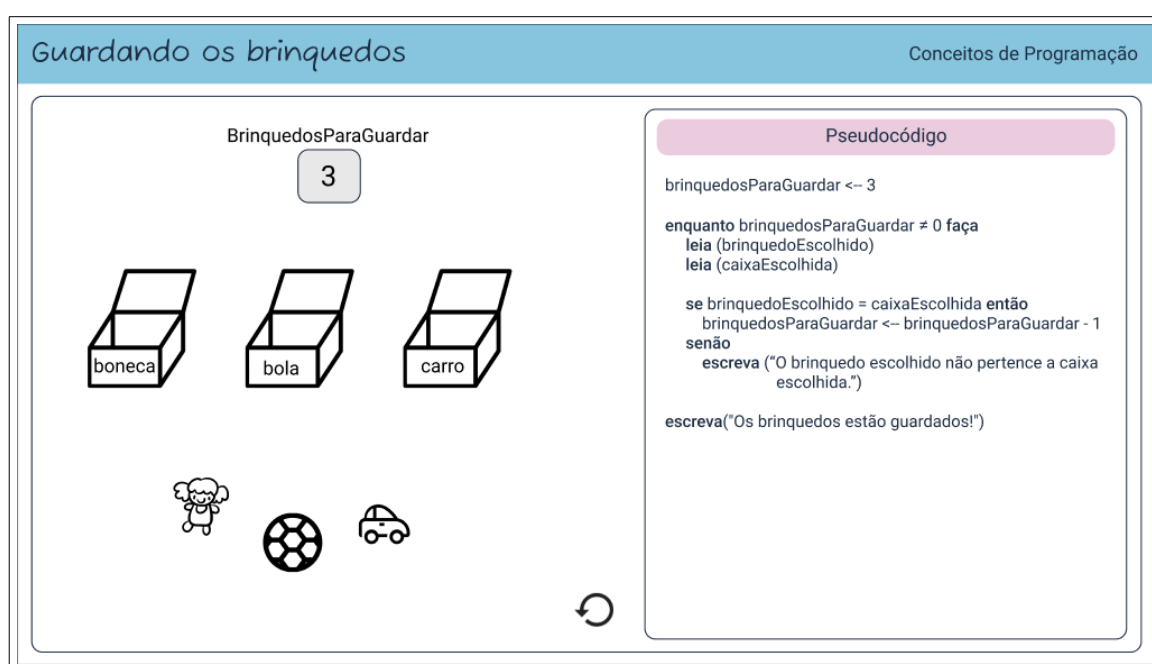


Figura 5.3: Protótipo da simulação “Guardando os Brinquedos” apresentado na atividade de aprendizado para fornecer contexto para as questões.


```
se brinquedoEscolhido = caixaEscolhida então  
  brinquedosParaGuardar <-- brinquedosParaGuardar - 1  
senão  
  escreva ("O brinquedo escolhido não pertence a caixa  
    escolhida.")
```

(a) Pseudocódigo referente à Pergunta 1

```
escreva("Os brinquedos estão guardados!")
```

(b) Pseudocódigo referente à Pergunta 2

```
enquanto brinquedosParaGuardar ≠ 0 faça  
  leia (brinquedoEscolhido)  
  leia (caixaEscolhida)  
  
  se brinquedoEscolhido = caixaEscolhida então  
    brinquedosParaGuardar <-- brinquedosParaGuardar - 1  
  senão  
    escreva ("O brinquedo escolhido não pertence a caixa  
      escolhida.")
```

(c) Pseudocódigo referente à Pergunta 3

```
brinquedosParaGuardar <-- brinquedosParaGuardar - 1
```

(d) Pseudocódigo referente à Pergunta 4

```
brinquedosParaGuardar <-- 3
```

(e) Pseudocódigo referente à Pergunta 5

```
leia (brinquedoEscolhido)  
leia (caixaEscolhida)
```

(f) Pseudocódigo referente à Pergunta 6

Figura 5.4: Trechos de pseudocódigo apresentados em cada pergunta da atividade de aprendizado referente à simulação “Guardando os Brinquedos”.

Capítulo 6

Resultados da avaliação da simulação

Os resultados da avaliação da simulação se encontram divididos em duas seções. A primeira se refere ao formulário de usabilidade, no qual obtivemos 26 respostas dos alunos presentes. A segunda é referente às respostas da atividade de aprendizado, na qual tivemos 13 respostas. Em virtude do tempo limitado durante a aula, alguns dos alunos não conseguiram terminar de preencher o segundo formulário, da atividade de aprendizado, por isso, obtivemos menos respostas nele.

6.1 Formulário de usabilidade

O formulário de usabilidade adaptado do questionário SUS foi preenchido pelos alunos logo após eles terminarem de explorar a simulação. Dos 26 alunos que o responderam, 12 não haviam tido contato anterior com programação e 14 já haviam tido. Para analisar as respostas, separamos os resultados entre esses dois grupos de estudantes.

A Figura 6.1 apresenta as respostas dos alunos que haviam tido contato anterior com programação. Dentro desse grupo de alunos, a maioria respondeu que gostaria de explorar mais a simulação (85,8%). Apenas 14,3% ficaram confusos com a ferramenta, sendo que 64,3% dos estudantes concordaram que ela foi fácil de usar. Ademais, 64,3% desses alunos acreditam que não precisariam de ajuda para explorar o MVP e 21,4% precisariam. Em consonância, 64,3% responderam que sabiam o que era preciso fazer para usar a simulação. Também podemos observar que somente 21,4% das crianças acharam que algumas coisas na simulação não fizeram sentido e 14,3% delas, que tiveram que fazer algumas coisas estranhas na ferramenta. Além disso, a maioria dos usuários (78,6%) acredita que seus amigos poderiam aprender a usar a simulação muito rápido e se sentiram confiantes enquanto a exploravam. Por fim, 28,6% dos alunos acham que tiveram que aprender muitas coisas antes de usar o MVP, enquanto a maioria discorda (57,2%).

Já na Figura 6.2, podemos observar as respostas dos alunos que não tiveram contato anterior com programação. Dentre eles, mais da metade (58,4%) respondeu que gostaria de explorar mais a simulação, porém um grande número (41,7%) não concordou nem discordou da afirmação. Ainda, 50% dos alunos desse grupo relatou que ficou confuso muitas vezes enquanto explorava o MVP e 33,3% não concordou nem discordou disso. A maioria (66,7%)

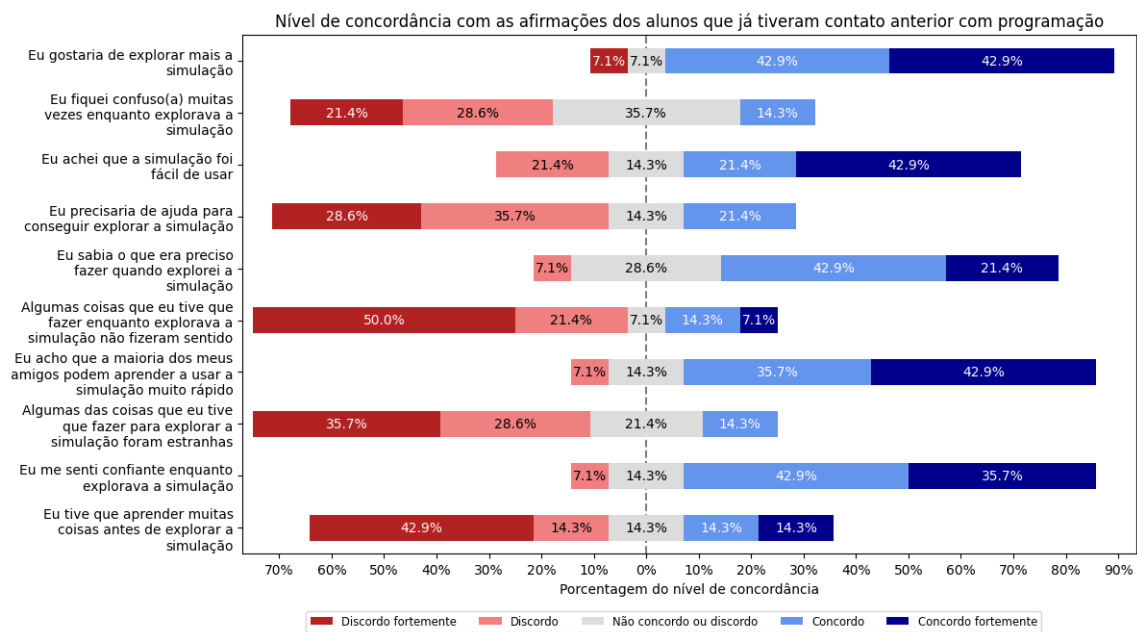


Figura 6.1: Porcentagem do nível de concordância com as afirmações dos alunos que já tiveram contato anterior com programação.

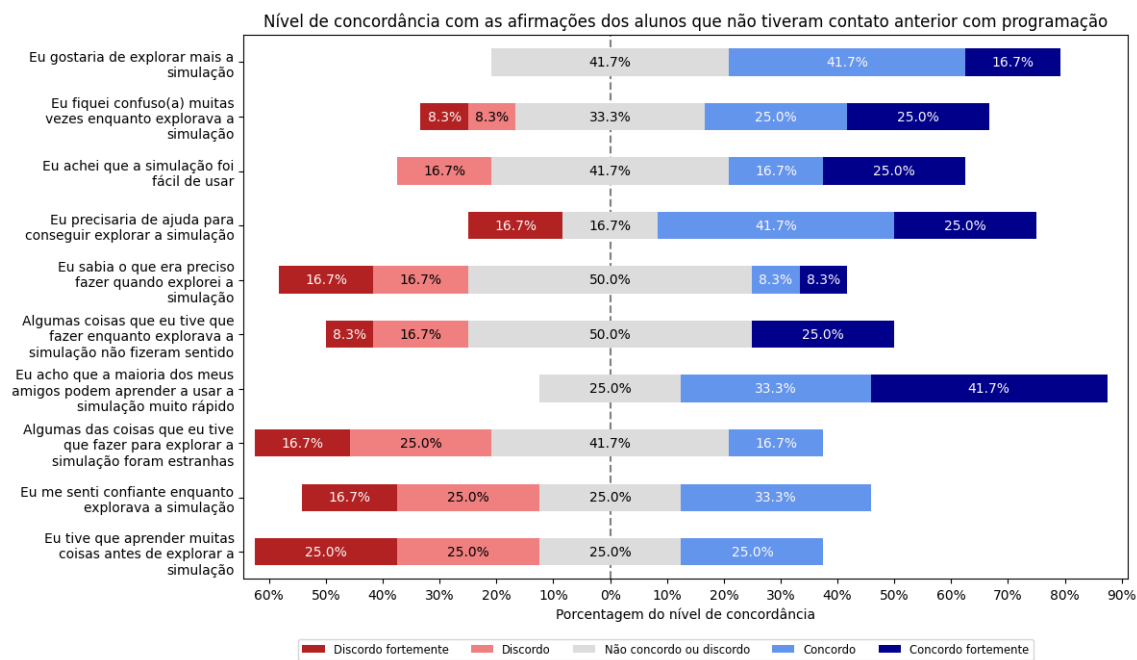


Figura 6.2: Porcentagem do nível de concordância com as afirmações dos alunos que não tiveram contato anterior com programação.

também achou que precisaria de ajuda para usar a simulação e apenas 16,6% sabia o que era preciso fazer quando usou a ferramenta. Neste último caso, 50% não concordou nem discordou da afirmação. De forma contraditória, 41,7% acharam que a ferramenta foi fácil de usar e apenas 16,7% discordaram. Ademais, metade dos alunos não concordaram nem discordaram de que algumas coisas que tiveram que fazer não fizeram sentido, e apenas

16,7% acharam que algumas coisas foram estranhas, enquanto 41,7% não concordaram nem discordaram. Apesar das respostas anteriores, a maioria dos usuários (72%) acredita que seus amigos poderiam aprender a usar a simulação muito rápido e apenas 25% acha que teve que aprender muitas coisas antes de explorá-la. Por fim, 33,3% das crianças relataram que se sentiram confiantes ao explorar o MVP enquanto 41,7% discordaram.

Ademais, a Figura 6.3 apresenta a média dos níveis de concordância com cada afirmativa dos dois grupos de alunos.

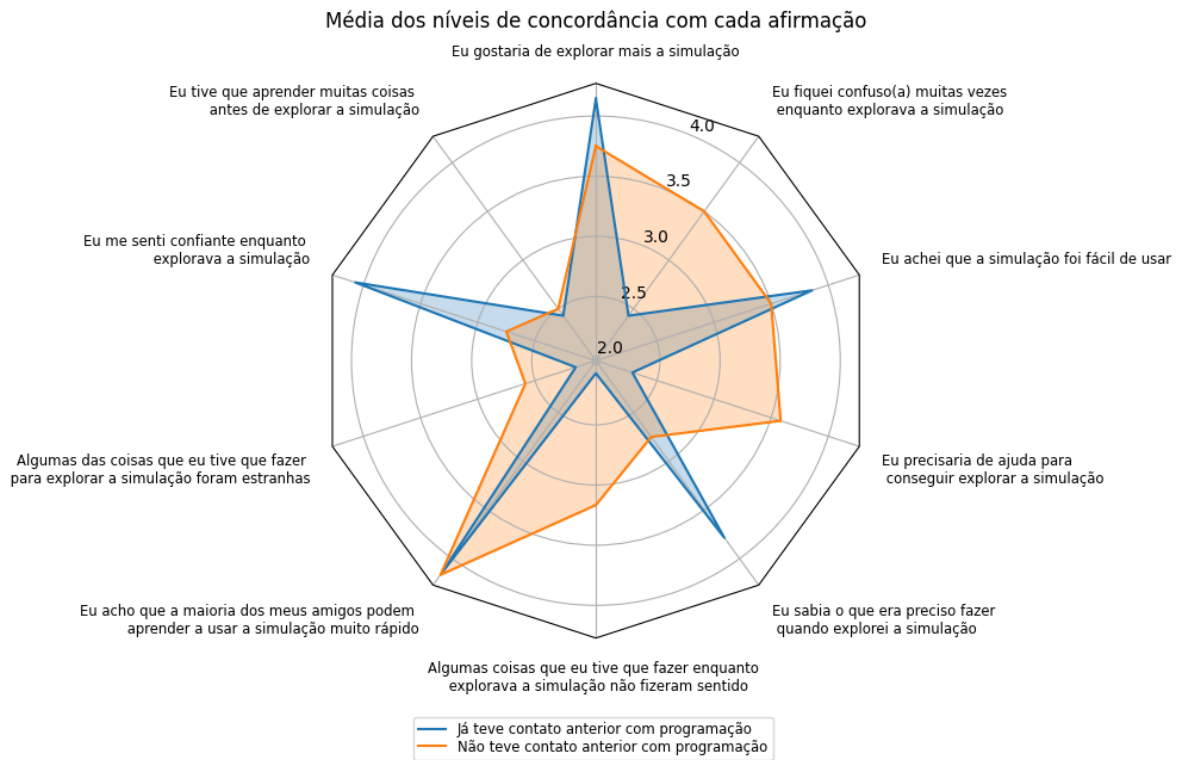


Figura 6.3: Média dos níveis de concordância com cada afirmação dos alunos que já tiveram contato anterior com programação e dos que não tiveram. Cada raio do gráfico representa a média para uma afirmação do formulário.

Podemos observar que as afirmações do formulário SUS apresentam duas conotações contrárias, tendo algumas delas um sentimento positivo em relação a ferramenta avaliada e outras um sentimento negativo. No questionário, as questões estão intercaladas, de forma que a primeira apresenta um sentimento positivo, a segunda apresenta um sentimento negativo, e assim por diante. Assim, analisando as respostas dos alunos que já tinham um contato prévio com programação, podemos notar que elas estão dentro do esperado, ou seja, em média, os estudantes concordam com as afirmações de caráter positivo e discordam daquelas de caráter negativo. Isso indica que o MPV projetado demonstrou potencial para a continuação ou complementação do aprendizado dos conceitos de lógica de programação das crianças.

Em relação às respostas dos alunos que não tinham contato prévio com programação, observamos algumas inconsistências entre os níveis de concordância com as afirmações de caráter positivo e negativo. Encontramos muitas respostas concordando com afirma-

ções complementares de conotações contrárias. Além disso, em pelo menos metade das afirmações, grande parte dos estudantes permaneceu neutro, optando por não concordar ou discordar, o que não aconteceu com o grupo anterior. Porém, apesar de obtermos muitas respostas concordantes com as afirmações com sentimento negativo em relação ao MVP, a maioria dos estudantes concordou que gostaria de explorar mais a simulação e que seus amigos poderiam aprender a usá-la muito rápido, mostrando o potencial da ferramenta. A partir dessas respostas, verificamos a necessidade de melhorar a forma de introduzir os conceitos de lógica de programação para aqueles que nunca tiveram contato com Computação.

Ademais, calculamos as pontuações SUS, na escala de 0 a 100, para as respostas dos dois grupos de alunos. Para os alunos que já tiveram contato anterior com programação, obtivemos uma média de *score* de 71,6, indicando uma boa usabilidade para o MVP, segundo [BANGOR et al. \(2009\)](#). Para aqueles que não tinham contato anterior com Computação, a média de pontuação calculada foi de 53,9, o que demonstra usabilidade razoável para a ferramenta de acordo com os autores.

	Já tiveram contato anterior com programação	Não tiveram contato anterior com programação
	82,5	55
	75	50
	100	47,5
	65	47,5
	75	65
	77,5	30
	65	87,5
	67,5	50
	60	57,5
	62,5	40
	72,5	50
	37,5	67,5
	97,5	-
	65	-
média	71,6	53,9

Tabela 6.1: Pontuação SUS calculada para cada aluno, com as médias para alunos que já tiveram contato anterior com programação e para os que não tiveram.

Opiniões e sugestões sobre a simulação

Em relação as questões dissertativas sobre opiniões e sugestões da simulação, apresentadas ao final do formulário de usabilidade, quando questionado o que mais gostaram na ferramenta, no geral, as crianças relataram que gostaram da experiência. Dividimos as respostas em quatro categorias baseadas nos temas que se destacaram no *feedback*: recursos específicos, interatividade, experiência de aprendizado e experiência geral. A Figura 6.4 apresenta os relatos dos estudantes divididos nessas categorias.

Assim, dentre o que as crianças mais gostaram, muitos relataram recursos específicos,

O que você mais gostou da simulação?

Recursos específicos	Interatividade
1. das variáveis 2. gostei da festa 3. Brigadeiro 4. da festa 5. montar a festa 6. do exemplo da festa 7. das bexigas 8. De fazer brigadeiro	1. A forma em que as imagens entram em cena 2. eu gostei dos joguinho da festa pois é bem interativo 3. do joguinho da festa, super interativo 4. eu gostei mais de como apareceu, as quantidades e variáveis. Porque é um modelo muito divertido para quem ainda não entende nada sobre computação. 5. Eu gostei porque é bem interativo e legal de fazer 6. como ela limita ou varia cada item para ter o total além do ciclo
Experiência de aprendizado	Experiência geral
1. eu gostei de aprender um pouco da programação 2. Achei bem didático e resumido 3. eu gostei que o pseudocódigo é muito simples e fácil de entender 4. O que eu mais gostei foi a experiência de testar o código 5. Simples e muitas das coisas que você tinha que clicar eram indicadas 6. Eu gostei que os resultados são bem diretos	1. Eu gostei da experiência 2. Gostei de poder mexer 3. gostei de poder ter esse contato com ela por que tem como fazer jogos com ela etc... 4. como podemos fazer uma joguinho muito simples e rápido

Figura 6.4: Respostas dos alunos em relação à pergunta livre “O que você mais gostou da simulação?”, divididas em categorias.

como preparar um determinado item para a festa. Além disso, algumas respostas ressaltaram a interatividade da simulação como ponto positivo, como os elementos visuais se apresentavam. Outros relatos destacaram boas experiências de aprendizado e no geral, descrevendo o MVP como didático, simples e direto.

Sobre o que menos gostaram da simulação, observamos outras quatro categorias em que se encaixavam os temas de respostas dos alunos: experiência positiva geral, confusão ou dificuldade, *design* ou problemas de interação, e conteúdo ou tarefas. A Figura 6.5 apresenta os relatos dos estudantes divididos nessas categorias. Muitos alunos não observaram pontos negativos na experiência geral com a simulação. Por outro lado, muitos encontraram dificuldades e relataram que ficaram confusos, principalmente com o pseudocódigo, ou que acharam complicado. Algumas crianças também apontaram alguns problemas de *design* ou interatividade no MVP e limitações nas possíveis tarefas.

Por fim, recebemos comentários referentes a apreciação e divertimento em relação à experiência com a simulação, e complexidade dela, além de sugestões para organizar os

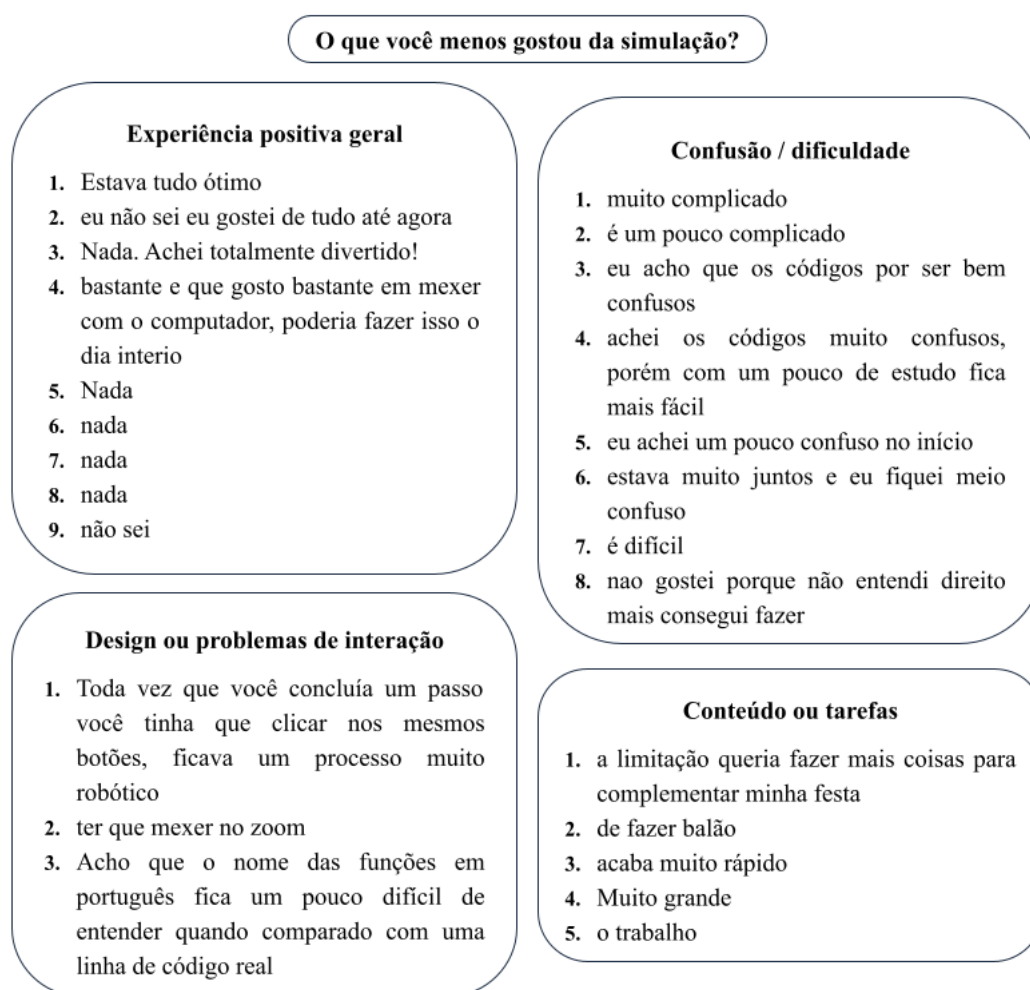


Figura 6.5: Respostas dos alunos em relação à pergunta livre “O que você menos gostou da simulação?”, divididas em categorias.

códigos, como mostra a Figura 6.6. A partir das opiniões e sugestões recebidas nas três questões livres, podemos também constatar a necessidade de algumas modificações nas interações da simulação, bem como na apresentação do pseudocódigo correspondente para tentar diminuir a complexidade encontrada por alguns e melhorar a usabilidade da ferramenta.

6.2 Atividade de aprendizado

O questionário com a atividades de aprendizado foi respondido por 13 alunos após o preenchimento do formulário anterior. Das 13 respostas, 6 foram de alunos que nunca tiveram contato com programação e 7 de alunos que já tiveram. Ainda, dos 6 alunos, apenas 3 responderam a segunda parte da atividade, referente à simulação “Guardando os Brinquedos”. Para analisar as respostas, separamos os resultados entre os dois grupos de estudantes.

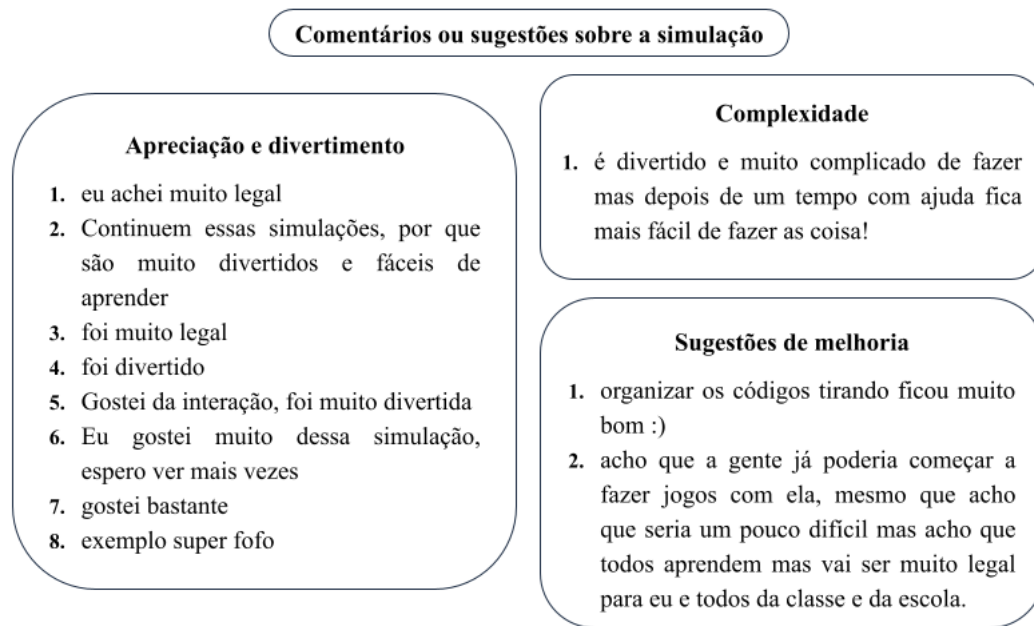


Figura 6.6: Respostas dos alunos em relação à pergunta livre sobre comentários e sugestões da simulação, divididas em categorias.

6.2.1 Perguntas sobre a simulação “Planejando a Festa”

A Figura 6.7 apresenta as respostas dos alunos às questões relacionadas à simulação “Planejando a Festa”. A pergunta 1 tinha como resposta o conceito de ‘condicional’. Considerando os alunos que já tinham um contato anterior com programação, 42,9% acertou esta questão, outros 42,9% escolheram o conceito de ‘variáveis’ e 14,3%, o de ‘saída’. Dentre os demais alunos, que nunca tiveram contato com Computação, 33,3% escolheu a resposta certa, enquanto 50% respondeu ‘variáveis’ e 14,3% escolheu ‘saída’. Podemos observar que houve confusão entre o conceito de ‘condicional’ e o de ‘variáveis’ e ‘saída’ pelos dois grupos de estudantes. Entretanto, no trecho de pseudocódigo apresentado nessa questão (Figura 5.2a) temos a repetição dos dois conceitos que foram confundidos como resposta na execução dentro dos condicionais, o que poderia explicar o engano.

No caso da questão 2, que tinha como resposta ‘variáveis’, observamos que houve uma confusão entre esse conceito e o de ‘entrada’, que também foi notada no geral. Nessa pergunta, 57,1% dos alunos com conhecimento prévio de Computação acertaram a resposta enquanto 42,9% escolheram ‘entrada’. Dos alunos que não tinham conhecimento prévio, tivemos um número menor de acertos (33,3%) e o restante também respondeu ‘entrada’.

Já na pergunta 3, a resposta correta era ‘laço de repetição’. Das crianças que tiveram contato anterior com programação, 42,9% acertaram a questão e os demais responderam ‘saída’ (42,9%) e ‘operadores’ (14,3%). Já entre os alunos que não tiveram contato anterior com Computação, 33,3% escolheram a resposta correta, enquanto 33,3% escolheram o conceito de ‘saída’, 16,7% de ‘operadores’ e outros 16,7% de ‘condicional’. Nesta questão tivemos muitas respostas divergentes, assim como na pergunta 1. O trecho de pseudocódigo apresentado nela (Figura 5.2c) trazia além do laço de repetição, o conceito de operadores

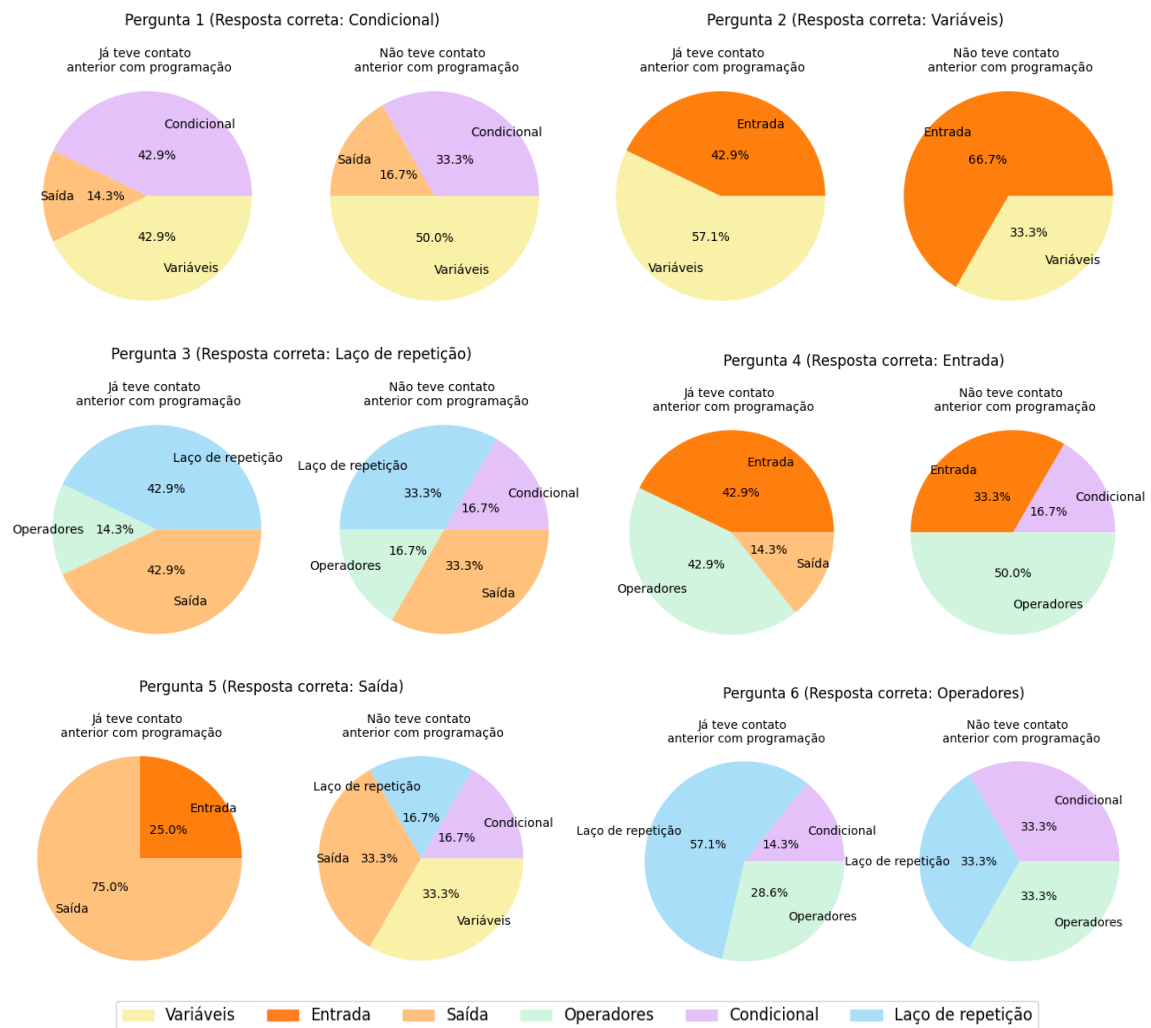


Figura 6.7: Porcentagens das respostas dos alunos às perguntas da atividade de aprendizado relacionada à simulação “Planejando a Festa”. Para cada pergunta, apresentamos as respostas dos alunos que já tiveram contato anterior com Computação à esquerda e as dos que não tiveram à direita.

representado na condição do laço e dentro da execução dele, o que poderia justificar a escolha desta resposta por alguns alunos. Além disso, acreditamos também que a ação de preparar um item, a qual ocorre dentro do laço, possa ter sido confundida com o conceito de ‘saída’, respondido por outros. Entretanto, a escolha do conceito de ‘condicional’ por alguns estudantes indica a falta de compreensão deles.

Nas três últimas questões, também observamos que não houve entendimento da maior parte dos alunos em relação aos conceitos de lógica de programação. A quarta pergunta do questionário tinha como resposta o conceito de ‘entrada’. Nela, observamos que 42,9% dos alunos com conhecimento em programação acertaram a questão. Outras respostas foram ‘operadores’ (42,9%) e ‘saída’ (14,3%). Dos alunos sem conhecimento prévio em Computação, 33,3% escolheram a resposta certa, enquanto 50% responderam ‘operadores’ e 16,7% ‘condicional’.

Na pergunta 5, que tinha como resposta o conceito de ‘saída’, a maioria das crianças com conhecimento anterior de programação acertou (75%) e apenas 25% confundiram

com de ‘entrada’. Dos alunos sem conhecimento anterior 33,3% responderam corretamente e os demais alunos responderam ‘variáveis’ (33,3%), ‘condicional’ (16,7%) e ‘laço de repetição’ (16,7%).

Por fim, na questão 6 a resposta correta era ‘operadores’ e poucos alunos acertaram o conceito apresentado. Entre as crianças que já tiveram contato com Computação, apenas 28,6% acertou a resposta. A maior parte das respostas foi ‘laço de repetição’ (57,1%) e os demais responderam ‘condicional’ (14,3%). Entre os estudantes que nunca tiveram contato com programação, as respostas foram igualmente escolhidas entre ‘operadores’, ‘laço de repetição’ e ‘condicional’.

6.2.2 Perguntas sobre a simulação “Guardando os Brinquedos”

A Figura 6.8 apresenta as respostas dos alunos às questões relacionadas à simulação “Guardando os Brinquedos”, com a qual não houve interação. As crianças apenas visualizaram a imagem de um estado da simulação e do pseudocódigo completo. No geral, podemos observar que as crianças com conhecimento prévio em programação fizeram mais confusões em relação aos conceitos de programação abordados do que os demais alunos.

A primeira pergunta tinha como resposta ‘condicional’ e, dos alunos que já tiveram contato com Computação, 85,7% acertaram, enquanto 14,3% responderam ‘variáveis’. Entre aqueles que não conheciam programação, 66,7% escolheram a resposta correta e 33,3% responderam o conceito de ‘saída’. Nesta questão, novamente, observamos que o trecho de pseudocódigo apresentado (Figura 5.4a) continha também os demais conceitos escolhidos como resposta por alguns estudantes, o que poderia explicar a confusão.

Já na questão 2, cuja resposta era ‘saída’, houve problemas de compreensão dos conceitos por parte da maioria das crianças. Entre aquelas que já conheciam programação, apenas 28,6% acertaram a resposta, enquanto 42,9% escolheram ‘operadores’ e 28,6% escolheram ‘entrada’. Entre os alunos que não conheciam Computação, 33,3% deles acertaram e 66,7% escolheram ‘entrada’.

Na pergunta 3, em que a resposta correta era ‘laço de repetição’, as duas crianças que não tinham contato anterior com Computação e responderam a questão, acertaram. Dentre os demais alunos, que já tiveram contato, 28,6% acertaram a resposta e o restante escolheu os conceitos de ‘saída’ (28,6%), ‘condicional’ (28,6%) e ‘entrada’ (14,3%). Porém, ressaltamos que todos os conceitos citados estavam presentes no trecho de pseudocódigo apresentado para eles na pergunta (Figura 5.4c).

Já a pergunta 4 tinha como resposta o conceito de ‘operadores’. Entre os alunos que conheciam Computação, apenas 14,3% acertaram a questão, e a maioria (42,9%) respondeu ‘variáveis’. Outros escolheram ‘laço de repetição’ (28,6%) e ‘condicional’ (14,3%). Entre os alunos que não tinham conhecimento, a maioria (66,7%) acertou a resposta e 33,3% respondeu ‘laço de repetição’. Nessa questão, o trecho de pseudocódigo apresentado (Figura 5.4d) trazia uma operação sendo atribuída a uma variável. Assim, a confusão com o conceito de ‘variáveis’ pode ter ocorrido por isso. Entretanto, a escolha das demais respostas mostrou, novamente, a falta de compreensão dos conceitos de programação por parte das crianças.

Ademais, nas duas últimas questões, também observamos que não houve entendimento

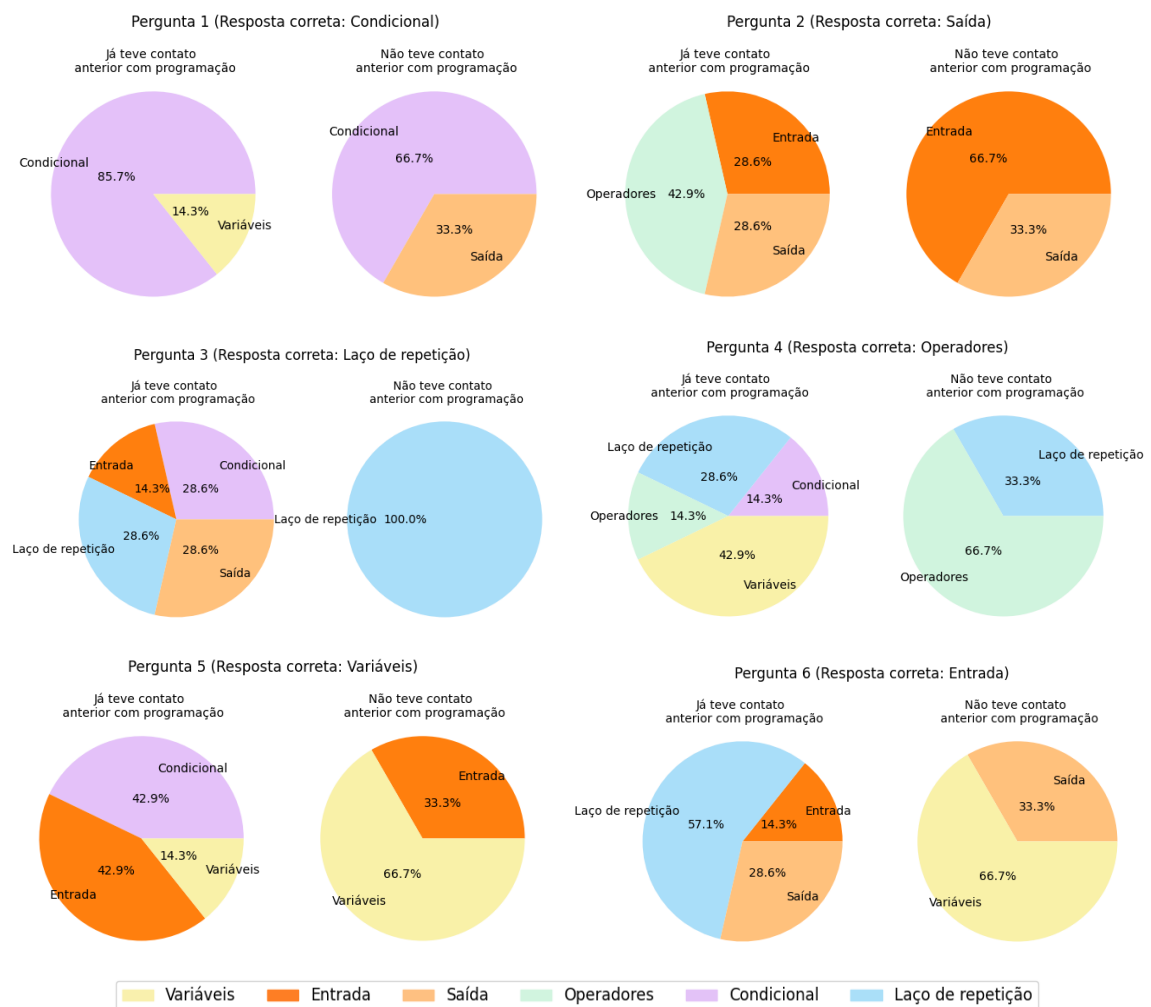


Figura 6.8: Porcentagens das respostas dos alunos às perguntas da atividade de aprendizado relacionada à simulação “Guardando os Brinquedos”. Para cada pergunta, apresentamos as respostas dos alunos que já tiveram contato anterior com Computação à esquerda e as dos que não tiveram à direita.

dos alunos em relação aos conceitos. Na questão 5, em que a resposta era ‘variáveis’, dos alunos que tiveram contato anterior com programação, apenas 14,3% acertaram, enquanto 42,9% responderam ‘entrada’ e 42,9% responderam ‘condicional’. Dos estudantes sem conhecimento prévio, 66,7% respondeu ‘variáveis’ e 33,3% ‘entrada’.

Finalmente, na pergunta 6, a resposta correta era ‘entrada’. Entre as crianças que conheciam Computação, apenas 14,3% acertou a resposta, 57,1% apontaram de forma errada o conceito de ‘laço de repetição’ e 28,6% o de ‘saída’. Dentre os três alunos que nunca tiveram contato com programação, nenhum acertou essa questão, 66,7% deles confundiu o conceito com o de ‘variáveis’ e 33,3% escolheu ‘saída’.

A partir desses resultados, observamos que o uso das simulações não cumpriu a finalidade de ensinar os conceitos de lógica de programação para as crianças. Entretanto, como observado, algumas confusões dos conceitos de programação pelos alunos podem ter sido causadas pela formulação das questões da atividade de aprendizado. A presença

de muitos conceitos, que inclusive se repetiam, nos trechos de pseudocódigo apresentados em algumas questões pode ter gerado ambiguidade nas respostas. Ainda assim, aqueles conceitos que foram apresentados de forma isolada, como o trecho de ‘variáveis’, não poderia ter sido confundido com o de ‘condicional’, por exemplo, sendo completamente desrelacionado. Isso também indica a falta de entendimento do conteúdo apresentado.

Ademais, notamos que os alunos sem conhecimento prévio em programação tiveram um maior número de acertos nas questões relacionadas à simulação “Guardando os Brinquedos”, com a qual eles não interagiram. Isso poderia sugerir a necessidade de alterações e melhorias nas interações implementadas na simulação “Planejando a Festa”. Por fim, acreditamos que, no geral, os resultados indicaram a necessidade de diversas reformulações para o MVP, implicando em uma nova iteração nos ciclos de *design* e engenharia de DSR.

Capítulo 7

Considerações finais

Neste trabalho criamos e avaliamos um MVP de uma ferramenta de simulações interativas para o ensino de conceitos de programação para crianças do ensino fundamental. Utilizamos a metodologia de *Design Science Research* proposta por [WIERINGA \(2014\)](#) para conduzir a pesquisa deste projeto, seguindo os ciclos de engenharia e *design*. Dessa forma, realizamos uma investigação do problema, *design* e validação do artefato, implementação e avaliação do MVP. As conclusões referentes aos resultados obtidos a partir da avaliação de usabilidade da ferramenta e do aprendizado dos alunos, bem como, as limitações do projeto e sugestões para trabalhos futuros são descritos nas seções a seguir.

7.1 Conclusões

Em relação à análise de usabilidade do MVP, concluímos que o uso de simulações interativas para ensinar conceitos de lógica de programação demonstrou potencial e despertou o interesse dos usuários iniciais que testaram a ferramenta. No geral, os alunos do 8º ano do ensino Fundamental mostraram curiosidade em explorar mais a simulação e acreditam que seus colegas poderiam aprender a usá-la muito rápido.

Principalmente para crianças que já tinham um contato anterior com programação, o MVP apresentou potencial para continuação ou complementação do aprendizados delas. A análise das respostas do formulário de usabilidade mostrou que, em média, estes alunos concordaram com as afirmações de sentimento positivo em relação à ferramenta e discordaram das afirmações de sentimento negativo. Ademais, a média da pontuação SUS obtida por meio do *feedback* deste grupo de alunos foi de 71,6, o que indica uma boa usabilidade da simulação.

Para o grupo de crianças que não tinham conhecimento prévio de Computação, observamos mais concordância com as afirmações de caráter negativo. Muitos alunos alegaram que ficaram confusos ao explorar o MVP e que precisariam de ajuda para continuar a usá-lo, por exemplo. Assim, verificamos a necessidade de melhorar a maneira de introduzir os conceitos de lógica de programação para aqueles que terão um primeiro contato com esse conteúdo por meio da ferramenta. Ainda, a média do *score* de usabilidade calculada

através das respostas desses estudantes foi de 53,9, indicando uma usabilidade razoável da ferramenta.

Além das respostas do questionário SUS, também coletamos opiniões e sugestões livres sobre o MVP. Por meio delas, confirmamos o potencial da abordagem proposta, com relatos positivos de experiência e constatamos a demanda de algumas alterações na interatividade da simulação e apresentação do seu pseudocódigo, com o intuito de diminuir a dificuldade e confusão enfrentadas pelos alunos e de melhorar a usabilidade geral da ferramenta.

Em relação à análise da atividade de aprendizado, concluímos que, em um primeiro momento, a simulação não ajudou na compreensão dos conceitos de lógica de programação apresentados. Dentre os alunos que já tinham um conhecimento anterior em Computação, 42,9% acertaram todas ou quase todas as questões referentes à simulação “Planejando a Festa” e apenas 14,3% acertaram todas ou quase todas as questões sobre a simulação “Guardando os Brinquedos”. Dos alunos que não tinham contato anterior com programação, 33,3% acertaram todas ou quase todas as questões sobre a simulação “Planejando a Festa” e a mesma quantidade em relação às questões da simulação “Guardando os Brinquedos”.

Por meio dessa análise, também notamos que os alunos sem conhecimento prévio em programação tiveram um maior número de acertos nas questões relacionadas à simulação “Guardando os Brinquedos”, com a qual eles não interagiram. Logo, caso tenha de fato ocorrido maior compreensão da “simulação” sem interações, isso poderia sugerir a necessidade de alterações e melhorias nas interações implementadas na simulação “Planejando a Festa”.

Porém, as análises referentes à atividade de aprendizado apresentam algumas ressalvas. Observamos que algumas confusões dos conceitos de programação feitas pelos alunos podem ter sido causadas pela formulação das questões da atividade. Algumas das perguntas apresentavam trechos de pseudocódigo com muitos conceitos, que inclusive se repetiam, podendo ter gerado ambiguidade no entendimento dessas perguntas. Além disso, houve limitações em relação ao tempo e finalização dessa atividade, discutidas na próxima seção.

Contudo, em outras questões, os conceitos foram apresentados de forma isolada, sendo completamente desrelacionados com algumas das respostas escolhidas, indicando a falta de compreensão do conteúdo apresentado. Dessa forma, verificamos a necessidade de reformulações para o MVP, implicando em uma nova iteração nos ciclos de *design* e engenharia de DSR, o que é um passo natural para esse tipo de pesquisa.

7.2 Limitações e trabalhos futuros

Durante o projeto, um dos desafios enfrentados foi encontrar escolas para realizar a aplicação do MVP em sala de aula com alunos do Ensino Fundamental. Um dos motivos para isso foi termos esperado até o mês de Setembro para entrar em contato com as escolas, quando o planejamento das aulas já havia sido concluído. Entretanto, queríamos garantir que o MVP estivesse implementado e funcional antes de prometer uma aula para as turmas. Com a ajuda do professor Dr. Leônidas Brandão do Departamento de Ciência da Computação do IME-USP, conseguimos o contato do professor Henri Silva da Escola de Aplicação da FEUSP, que nos permitiu participar de uma de suas aulas em uma turma do 8º do Ensino Fundamental para realizar uma atividade, aplicando o MVP

e coletando o *feedback* dos alunos.

Outra limitação se refere ao tempo de duração da aula em que aplicamos o MVP, que foi de 50 minutos. Nesse sentido, metade da turma não conseguiu terminar de preencher o formulário que continha a atividade de aprendizado, o qual foi apresentado por último. Além disso, conforme os colegas iam terminando a atividade, observamos que algumas crianças sentiram pressa para também finalizar o preenchimento do questionário. Com isso, houve uma certa propagação das respostas entre os alunos, que compartilharam o que responderam com os demais.

Para a continuação deste trabalho sugerimos algumas alterações para o protótipo aplicado, como a introdução dos conceitos de programação na forma de tutorial antes de iniciar a interação com a simulação. O *link* referente aos “Conceitos de Programação” que abre a caixa de diálogo com as definições desses poderia ser utilizado para compor esse tutorial, adicionando ainda animações e interações para dividi-lo em etapas, as quais os alunos deveriam concluir antes de explorar a simulação. Ainda, sugerimos uma melhoria na indicação dos conceitos de programação na apresentação do pseudocódigo, uma vez que as cores e os nomes deles só eram visualizados ao passar o mouse em cima dos trechos de código. Ademais, recomendamos uma reformulação da atividade de aprendizado, modificando os trechos de pseudocódigo apresentado a fim de remover a ambiguidade.

Por fim, também propomos apresentar cada conceito de lógica de programação em simulações individuais, como foi proposto no projeto de Iniciação Científica (IC) intitulado “Uma Ferramenta de Simulações Interativas para Ensino de Computação para Crianças”(<https://github.com/mariliatd/monografia/tree/main/ic>), realizado entre outubro de 2022 e outubro de 2023. Desse modo, os usuários poderão explorar as interações envolvidas em cada conceito separadamente, o que pode levar a melhor compreensão de cada um deles por parte das crianças e mais autonomia para explorar as simulações fora da sala de aula ou sem a ajuda de um docente.

Referências

- [ARAÚJO *et al.* 2021] Evando Santos ARAÚJO, João Lucas Barbosa do NASCIMENTO, Jucilene Carvalho SILVA e Caiala Fonseca Andrade BIM. “O uso de simuladores virtuais educacionais e as possibilidades do phet para a aprendizagem de física no ensino fundamental”. *Revista de Ensino de Ciências e Matemática* 12.3 (2021), pp. 1–25 (citado na pg. 4).
- [BANGOR *et al.* 2008] Aaron BANGOR, Philip T. KORTUM e James T. MILLER. “An empirical evaluation of the system usability scale”. *Intl. Journal of Human–Computer Interaction* 24.6 (2008), pp. 574–594 (citado na pg. 19).
- [BANGOR *et al.* 2009] Aaron BANGOR, Philip T. KORTUM e James T. MILLER. “Determining what individual sus scores mean: adding an adjective rating scale”. *Journal of usability studies* 4.3 (2009), pp. 114–123 (citado nas pgs. 19, 40).
- [BLATT *et al.* 2017] Lucas BLATT, Valdecir BECKER e Alexandre FERREIRA. “Mapeamento sistemático sobre metodologias e ferramentas de apoio para o ensino de programação”. In: *Workshop de Informática na Escola (WIE)*. SBC. 2017, pp. 815–824 (citado nas pgs. 3, 8).
- [Blockly Games 2024] Blockly Games. URL: <https://blockly.games/>. acesso em 14/12/2024. 2024 (citado na pg. 8).
- [BORDINI *et al.* 2016] Adriana BORDINI *et al.* “Computação na educação básica no brasil: o estado da arte”. *Revista de Informática Teórica e Aplicada* 23.2 (2016), pp. 210–238 (citado nas pgs. 2, 8).
- [BRACKMANN 2017] Christian Puhlmann BRACKMANN. “Desenvolvimento do pensamento computacional através de atividades desplugadas na educação básica”. Tese de dout. Universidade Federal do Rio Grande do Sul, 2017 (citado nas pgs. viii, 10, 11).
- [BRASIL 2021] BRASIL. *Normas sobre Computação na Educação Básica - Complemento à BNCC*. Parecer Normativo, n. 23001.001050/2019-18. Ministério da Educação. Conselho Nacional de Educação. Brasília, DF, abr. de 2021. 49 pp. (citado na pg. 2).

- [BREZOLIN e SILVEIRA 2021] Carmen Vera Scorsatto BREZOLIN e Milene Selbach SILVEIRA. “Panorama brasileiro de uso de ferramentas para desenvolvimento do pensamento computacional e ensino de programação”. In: *Workshop sobre Educação em Computação (WEI)*. SBC. 2021, pp. 398–407 (citado na pg. 4).
- [BROOKE *et al.* 1996] John BROOKE *et al.* “Sus-a quick and dirty usability scale”. *Usability evaluation in industry* 189.194 (1996), pp. 4–7 (citado nas pgs. 19, 20).
- [CALDWELL *et al.* 2008] Ben CALDWELL *et al.* “Web content accessibility guidelines (wcag) 2.0”. *WWW Consortium (W3C)* 290.1-34 (2008), pp. 5–12 (citado na pg. 29).
- [CodeCombat 2024] CodeCombat. URL: <https://codecombat.com/home>. acesso em 14/12/2024. 2024 (citado nas pgs. viii, 8, 9).
- [CRAVO e ESPARTOSA 2021] Andreia Regina CRAVO e Karina Dias ESPARTOSA. “Avaliação de simulações interativas em ciências da plataforma on-line “phet” por meio de parâmetros de avaliação e de oficinas com futuros docentes”. *Revista de Ensino de Biologia da SBEnBio* (2021), pp. 658–679 (citado na pg. 4).
- [DEXHEIMER *et al.* 2017] Judith W. DEXHEIMER *et al.* “Usability evaluation of the smart application for youth with mtbi”. *International journal of medical informatics* 97 (2017), pp. 163–170 (citado na pg. 20).
- [DRESCH *et al.* 2015] Aline DRESCH, Daniel Pacheco LACERDA e José Antônio Valle Antunes JR. *Design science research*. Springer, 2015 (citado nas pgs. 15, 16).
- [Drone Blocks 2024] Drone Blocks. URL: <https://droneblocks.io/>. acesso em 14/12/2024. 2024 (citado nas pgs. viii, 12, 13).
- [FALKNER *et al.* 2019] Katrina FALKNER *et al.* “An international comparison of k-12 computer science education intended and enacted curricula”. In: *Proceedings of the 19th Koli calling international conference on computing education research*. 2019, pp. 1–10 (citado na pg. 1).
- [FERNANDES *et al.* 2012] Andrino FERNANDES, Anna Luiza ESPÍNDOLA e Marina Pinheiro OLIVEIRA. “Animações e simulações para apoio ao ensino da lógica de programação”. *Revista Técnico-Científica do IFSC* (2012), pp. 266–266 (citado nas pgs. viii, 12).
- [FRANÇA e AMARAL 2013] Rozelma Soares de FRANÇA e Haroldo José Costa do AMARAL. “Ensino de computação na educação básica no brasil: um mapeamento sistemático”. In: *Anais do XXI Workshop sobre Educação em Computação*. SBC. 2013, pp. 426–431 (citado na pg. 2).
- [GOMES *et al.* 2017] Vitor GOMES, Renata PONTES, Carlos CAMELO, Gilvonaldo CAVALCANTI e Mirko PERKUSICH. “Ensino de programação para crianças e adolescentes: um estudo exploratório”. In: *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*. Vol. 6. 1. 2017, p. 490 (citado na pg. 4).

- [HEVNER *et al.* 2004] Alan R. HEVNER, Salvatore T. MARCH, Jinsoo PARK e Sudha RAM. “Design science in information systems research”. *MIS quarterly* (2004), pp. 75–105 (citado na pg. 15).
- [HUBWIESER *et al.* 2015] Peter HUBWIESER *et al.* “A global snapshot of computer science education in k-12 schools”. In: *Proceedings of the 2015 ITiCSE on working group reports*. ACM Digital Library, 2015, pp. 65–83 (citado na pg. 1).
- [KHATRI *et al.* 2013] Raina KHATRI, Charles HENDERSON, Renee COLE e Jeffrey FROYD. “Over one hundred million simulations delivered: a case study of the phet interactive simulations”. In: *Physics Education Research Conference*. Citeseer. 2013, pp. 205–208 (citado na pg. 4).
- [KHOURI *et al.* 2020] Cátia Mesquita Brasil KHOURI, Gidevaldo Novais dos SANTOS e Maria Silva Santos BARBOSA. “Mapeamento sistemático em metodologias de ensino-aprendizagem de programação”. *Revista de Ciência da Computação* 2.1 (2020), pp. 13–27 (citado na pg. 8).
- [KLINCZAK e PAULA PINTO 2024] Marjori KLINCZAK e Jose Simao de PAULA PINTO. “Estudo comparativo de kits de robótica voltados ao ensino de pensamento computacional”. *Anais CIET: Horizonte* (2024) (citado na pg. 10).
- [Lego Mindstorms 2024] Lego Mindstorms. URL: <https://kids.lego.com/pt-br/sets/mindstorms/ev3-0729302d5ae04b5d88a30c2dd6d7afba>. acesso em 14/12/2024. 2024 (citado nas pgs. viii, 8, 10, 11).
- [MEDEIROS e WÜNSCH 2019] Luciano Frontino de MEDEIROS e Luana Priscila WÜNSCH. “Ensino de programação em robótica com arduino para alunos do ensino fundamental: relato de experiência”. *Revista Espaço Pedagógico* 26.2 (2019), pp. 456–480 (citado nas pgs. viii, 10).
- [MIT App Inventor 2024] MIT App Inventor. URL: <https://appinventor.mit.edu/>. acesso em 14/12/2024. 2024 (citado nas pgs. viii, 7–9).
- [PRICE *et al.* 2018] Argenta M. PRICE, Katherine K. PERKINS, N. G. HOLMES e Carl E. WIEMAN. “How and why do high school teachers use phet interactive simulations”. *Learning* 33 (2018), p. 37 (citado nas pgs. 4, 12).
- [PUTNAM *et al.* 2020] Cynthia PUTNAM, Melisa PUTHENMADOM, Marjorie Ann CUERDO, Wanshu WANG e Nathaniel PAUL. “Adaptation of the system usability scale for user testing with children”. In: *Extended abstracts of the 2020 CHI conference on human factors in computing systems*. 2020, pp. 1–7 (citado nas pgs. viii, x, 20, 21).
- [RAMOS *et al.* 2020] Marcos Coelho RAMOS, Kitawann Tayrone de SOUSA NUNES CARDOSO e Maria do CARMO SILVA CARVALHO. “O ensino de ciências com o uso da ferramenta digital simulador phet por meio da estratégia investigativa nos anos finais do ensino fundamental ii”. *Anais CIET: Horizonte* (2020) (citado na pg. 4).

- [RIBEIRO *et al.* 2019] Leila RIBEIRO *et al.* *Diretrizes da sociedade brasileira de computação para o ensino de computação na educação básica*. Rel. técn. 2019 (citado nas pgs. viii, 3, 5).
- [RUNESON *et al.* 2020] Per RUNESON, Emelie ENGSTRÖM e Margaret-Anne STOREY. “The design science paradigm as a frame for empirical software engineering”. *Contemporary empirical methods in software engineering* (2020), pp. 127–147 (citado nas pgs. 15, 16).
- [S. M. DOS SANTOS *et al.* 2021] Aline de S. M. DOS SANTOS, Wellington G. PEREIRA e Rozelma Soares de FRANÇA. “Como ensinar ciência da computação para crianças? tendências e lacunas de pesquisa na área”. In: *Anais do XXIX Workshop sobre Educação em Computação*. SBC. 2021, pp. 298–307 (citado na pg. 2).
- [SÁNCHEZ-MORALES *et al.* 2020] A. SÁNCHEZ-MORALES, J. A. DURAND-RIVERA e C. L. MARTÍNEZ-GONZÁLEZ. “Usability evaluation of a tangible user interface and serious game for identification of cognitive deficiencies in preschool children”. *International Journal of Advanced Computer Science and Applications* 11.6 (2020) (citado na pg. 20).
- [E. O. SANTOS e SILVA 2020] Elisângela Oliveira SANTOS e Ivanderson Pereira da SILVA. “Revisão acerca do tema simulações computacionais no ensino de química (2008–2017)”. *Debates em Educação* 12.27 (2020), pp. 841–855 (citado na pg. 4).
- [P. S. C. SANTOS *et al.* 2018] Priscila S. C. SANTOS, Luis Gustavo J. ARAUJO e Roberto A. BITTENCOURT. “A mapping study of computational thinking and programming in brazilian k-12 education”. In: *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE. 2018, pp. 1–8 (citado na pg. 2).
- [Scratch - MIT 2024] Scratch - MIT. URL: <https://scratch.mit.edu/>. acesso em 14/12/2024. 2024 (citado nas pgs. viii, 7, 8).
- [SOUZA *et al.* 2021] Franciely Alves de SOUZA, Taciana Pontual FALCÃO e Rafael Ferreira MELLO. “O ensino de programação na educação básica: uma revisão da literatura”. *Anais do XXXII Simpósio Brasileiro de Informática na Educação* (2021), pp. 1265–1275 (citado na pg. 4).
- [TASFIA *et al.* 2023] Sheikh TASFIA, Muhammad Nazrul ISLAM, Syeda Ajbina NUSRAT e Nusrat JAHAN. “Evaluating usability of ar-based learning applications for children using sus and heuristic evaluation”. In: *Proceedings of the Fourth International Conference on Trends in Computational and Cognitive Engineering: TCCE 2022*. Springer. 2023, pp. 87–98 (citado na pg. 20).
- [WERLICH *et al.* 2018] Claudia WERLICH, Avanilde KEMCZINSKI e Isabela GASPARINI. “Pensamento computacional no ensino fundamental: um mapeamento sistemático”. In: *XXIII Congreso Internacional de Informática Educativa*. Nuevas Ideas en Informática Educativa (TISE) Chile. 2018, pp. 375–384 (citado na pg. 8).

REFERÊNCIAS

- [WIERINGA 2014] Roel J. WIERINGA. *Design science methodology for information systems and software engineering*. Springer, 2014 (citado nas pgs. 15, 17, 49).
- [WING 2006] Jeannette M. WING. “Computational thinking”. *Communications of the ACM* 49.3 (2006), pp. 33–35 (citado na pg. 1).
- [WROŃSKA *et al.* 2015] Natalia WROŃSKA, Begonya GARCIA-ZAPIRAIN e Amaia MENDEZ-ZORRILLA. “An ipad-based tool for improving the skills of children with attention deficit disorder”. *International journal of environmental research and public health* 12.6 (2015), pp. 6261–6280 (citado na pg. 20).
- [ZHU e WANG 2023] Meina ZHU e Cheng WANG. “Core competencies of k-12 computer science education from the perspectives of college faculties and k-12 teachers.” *International Journal of Computer Science Education in Schools* 6.2 (2023), n2 (citado na pg. 2).