UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Talles Siqueira Ceolin

CODE RUSH: DESENVOLVIMENTO DE UM JOGO DE CARTAS FÍSICO PARA APRENDIZADO DE LÓGICA DE PROGRAMAÇÃO

Santa Maria, RS
2022

Talles Siqueira Ceolin

CODE RUSH: DESENVOLVIMENTO DE UM JOGO DE CARTAS FÍSICO PARA APRENDIZADO DE LÓGICA DE PROGRAMAÇÃO

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de **Bacharel em Ciência da Computação**

Orientador: Prof. Dr. Giovani Rubert Librelotto

Co-orientador: Prof. Henry Emanuel Leal Cagnini

505
Santa Maria, RS
2022

Siqueira Ceolin, Talles

Code Rush: desenvolvimento de um jogo de cartas físico para aprendizado de lógica de programação / por Talles Siqueira Ceolin. – 2022.

57 f.: il.; 30 cm.

Orientador: Giovani Rubert Librelotto

Co-orientador: Henry Emanuel Leal Cagnini

Trabalho de Conclusão de Curso - Universidade Federal de Santa Maria, Centro de Tecnologia, Bacharelado em Ciência da Computação , RS, 2022.

1. Pseudocódigo. 2. Computação desplugada. 3. Pensamento computacional. 4. Jogos educacionais. I. Rubert Librelotto, Giovani. II. Emanuel Leal Cagnini, Henry. III. Título.

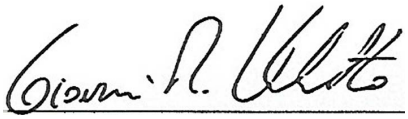
E-mail: tsceolin@inf.ufsm.br

Talles Siqueira Ceolin

CODE RUSH: DESENVOLVIMENTO DE UM JOGO DE CARTAS FÍSICO PARA APRENDIZADO DE LÓGICA DE PROGRAMAÇÃO

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de **Bacharel em Ciência da Computação**

Aprovado em 14 de fevereiro de 2022:



Giovani Rubert Librelotto, Dr. (UFSM)
(Presidente/Orientador)

Giliane Bernardi, Profa. (UFSM)

Roseclea Duarte Medina, Profa. (UFSM)

Santa Maria, RS

2022

RESUMO

CODE RUSH: DESENVOLVIMENTO DE UM JOGO DE CARTAS FÍSICO PARA APRENDIZADO DE LÓGICA DE PROGRAMAÇÃO

AUTOR: TALLES SIQUEIRA CEOLIN
ORIENTADOR: GIOVANI RUBERT LIBRELOTTO
CO-ORIENTADOR: HENRY EMANUEL LEAL CAGNINI

O ensino inicial sobre programação enfrenta desafios como grandes evasões e reprovações, assim como a introdução deste assunto para jovens que ainda não estudam sobre computação. A partir de teorias relacionadas ao pensamento computacional, a computação desplugada surge como forma alternativa para o aprendizado de lógica de programação. Diante destes conceitos, este trabalho apresenta não somente o desenvolvimento de um jogo de cartas físico como meio de aprender noções de lógica de programação e características específicas de linguagens de programação, como também uma aplicação para geração facilitada das imagens de cada carta. Os resultados obtidos por testes de jogabilidade com alunos do ensino médio exploram a efetividade do uso de jogos desplugados para aprendizado na área da computação.

Palavras-chave: Pseudocódigo. computação desplugada. pensamento computacional. jogos educacionais.

ABSTRACT

CODE RUSH: DEVELOPMENT OF A PHYSICAL CARD GAME FOR LEARNING PROGRAMMING LOGIC

AUTHOR: TALLES SIQUEIRA CEOLIN
ADVISOR: GIOVANI RUBERT LIBRELOTTO
COADVISOR: HENRY EMANUEL LEAL CAGNINI

The inital learnig of programming faces a few challenges such as evasion and failing subjects, just like the introduction of this subject for youngsters who do not study computer science. Based on theories related to computational thinking, computer science unplugged emerges as an alternative way of learning programming language. This paper presents not only the development of a physical card game as a programming logic and programming languages learning environment, as well as an application for generating each card's image. The results obtained through gameplay tests with high school students explore the effectivity of unplugged games as a form of learning in computer science.

Keywords: Pseudocode. unplugged computing. computational thinking. educational games.

LISTA DE FIGURAS

FIGURA 1 — Cartas do jogo Summon The JSON. 16
FIGURA 2 — Cartas do produto Coder Cards. 17
FIGURA 3 — Cartas e embalagem do jogo Coding Is Good. 17
FIGURA 4 — Logo e cartas do jogo AlgoCards. 18
FIGURA 5 — Etapas do modelo ADDIE. 20
FIGURA 6 — Esquema simulando as rodadas de um jogo. 24
FIGURA 7 — Leiaute de uma carta do jogo 25
FIGURA 8 — Mockup de cartas de diferentes baralhos. 26
FIGURA 9 — Fluxograma representando entradas e saídas do software. 27
FIGURA 10 — Interface gráfica do gerador de cartas. 27
FIGURA 11 — Exemplo de arquivo JSON e suas respectivas cartas geradas. 28
FIGURA 12 — JSON Schema para validar entrada de dados. 30
FIGURA 13 — Exemplo de páginas de um arquivo PDF gerado pelo programa. 31
FIGURA 14 — Foto dos alunos do CTISM jogando Code Rush. 33
FIGURA 15 — Foto do desenho da chave do campeonato de Code Rush. 34
FIGURA 16 — Resultados do questionário. 35
FIGURA 17 — Gráfico de média de avaliação por dimensão e subdimensão. 36
FIGURA 18 — Gráfico de frequência respostas do item “O jogo contribuiu para a minha aprendizagem na disciplina”. 36
FIGURA 19 — Gráfico de frequência respostas do item sobre JS “Descobri novos comandos e características da linguagem através do jogo” 37
FIGURA 20 — Gráfico de frequência respostas do item “Eu precisei aprender poucas coisas para poder começar a jogar o jogo” 37
FIGURA 21 — Gráfico de frequência respostas do item “Aprender a jogar este jogo foi fácil para mim” 38
FIGURA 22 — Gráfico de frequência respostas do item “O jogo não se torna monótono nas suas tarefas (repetitivo ou com tarefas chatas)” 38

LISTA DE TABELAS

TABELA 1 — Descrições e exemplos dos 4 pilares do pensamento computacional . . 13

TABELA 2 — Assuntos correspondentes às dificuldades das cartas de pseudocódigo. 22

LISTA DE ABREVIATURAS E SIGLAS

JS	JavaScript
PHP	Hypertext Preprocessor
CSV	Comma-separated-values (valores separados por vírgulas)
JSON	JavaScript Object Notation
PDF	Portable Document Format
ADDIE	Analyze, Design, Develop, Implement e Evaluate
CTISM	Colégio Técnico Industrial de Santa Maria
MEEGA	Model for the Evaluation of Educational Games
UX	Experiência do Usuário

SUMÁRIO

1 INTRODUÇÃO 10
1.1 OBJETIVOS 10
1.2 ESTRUTURA DO TRABALHO 11
2 REVISÃO DE LITERATURA 12
2.1 PENSAMENTO COMPUTACIONAL 12
2.2 COMPUTAÇÃO DESPLUGADA 14
2.3 PSEUDOCÓDIGO 14
2.4 JOGOS EDUCACIONAIS SOBRE PROGRAMAÇÃO 15
3 CODE RUSH 19
3.1 METODOLOGIA 19
3.1.1 Design instrucional 19
3.1.2 Definição das regras e design do jogo 21
3.1.3 Criação dos enunciados 21
3.2 PROJETO E IMPLEMENTAÇÃO 23
3.2.1 Regras do jogo 23
3.2.2 Cartas 24
3.2.3 Baralhos temáticos 25
3.2.4 Software de criação de baralhos 26
4 AVALIAÇÃO 32
4.1 TESTES COM USUÁRIOS 32
4.2 RESULTADOS OBTIDOS 34
5 CONCLUSÃO 40
5.1 TRABALHOS FUTUROS 41
REFERÊNCIAS 43
APÊNDICES 46

1 INTRODUÇÃO

Com o avanço da computação como parte do cotidiano de toda a população e sendo praticada em diversas áreas do conhecimento, certos conhecimentos desta área deixaram de ser restritos aos cientistas e profissionais do campo, e passaram a ter mais acesso por todos. Este avanço aumentou a procura e oferta de oportunidades de aprendizagem relacionadas a programação e computação em geral para um público mais jovem, como alunos de ensino médio (FONTES, 2021; SAMPAIO, 2021). Para introduzir este público ao assunto, técnicas lúdicas ligadas ao pensamento computacional (BRACKMANN, 2017) e áreas afins são aplicadas, como o uso de desafios de lógica ou jogos que estimulam a lógica similar a programação (ANDRÉ RAABA; BLIKSTEIN, 2020).

O aumento de oportunidades relacionadas a computação e a busca de profissionais na área (FONTES, 2021; SAMPAIO, 2021) incentiva os educadores a proporcionar novas formas de introduzir jovens com pouco ou nenhum conhecimento do assunto às noções de lógica e programação. A prática através de jogos lúdicos também pode ser uma forma eficaz ao trazer não somente melhorias no raciocínio como entretenimento aos jogadores (WING, 2016).

Além dos benefícios trazidos por estas formas de aprendizagem para aqueles que ainda não ingressaram no estudo da computação, há outro grupo a ser beneficiado: estudantes de disciplinas de algoritmos e programação, visto que estas possuem altos índices de evasão e reprovação (COSTA MORA, 2013). Considerando a extensão de conteúdos e informações inéditas para a maioria destes alunos, há uma necessidade de novos meios de aprendizado que auxiliem na memorização dos assuntos e na atenção para a disciplina.

1.1 OBJETIVOS

Considerando não somente os temas de computação desplugada (BLIKSTEIN, 2008) e pensamento computacional, mas também a realidade daqueles que não possuem acesso constante a meios eletrônicos como um computador ou celular, este trabalho apresenta a criação do Code Rush, um jogo de cartas em formato físico para auxiliar no desenvolvimento do raciocínio lógico e obtenção de conhecimentos de programação através de desafios rápidos envolvendo pseudocódigos ou linguagens de programação específicas. Além disso, este projeto disponibiliza um meio de automatizar a criação do leiaute de novas cartas e baralhos para aqueles que

desejam expandir os domínios de aplicação do jogo. Este trabalho possui os seguintes objetivos específicos:

- Desenvolver um jogo de cartas físico sobre programação que não necessita de computadores ou internet;
- Desenvolver uma aplicação para geração facilitada das cartas do jogo em formato de arquivo PDF pronto para impressão gráfica;
- Realizar uma sessão de testes de jogabilidade com pessoas de modo a coletar pontos a serem melhorados e avaliar as mecânicas e pontos fortes do jogo desenvolvido;
- Explorar a efetividade de jogos para aprendizado de lógica de programação e conhecimentos específicos de linguagens de programação;

1.2 ESTRUTURA DO TRABALHO

Este trabalho está organizado da seguinte forma. O Capítulo 2 apresenta conceitos da computação fundamentais para a construção deste projeto, como o pensamento computacional e a computação desplugada, assim como seus impactos no desenvolvimento de jogos com objetivos de aprendizado. Além disso, são analisados diversos exemplos de jogos educacionais semelhantes e relacionados a programação já existentes. O Capítulo 3 é dividido em duas seções, a primeira apresenta a metodologia utilizada durante o desenvolvimento do jogo de cartas, explorando as técnicas e recursos utilizados na elaboração das regras e elementos do mesmo. A segunda descreve as características do produto final, detalhando as regras de jogabilidade, o conteúdo de cada carta e o funcionamento do software de automatização na criação visual de cartas. O Capítulo 4 revela testes de jogabilidade feitos com pessoas, detalhando os instrumentos utilizados e seus resultados obtidos. Finalmente, o trabalho é concluído com o Capítulo 5, resumindo o projeto desenvolvido e seus benefícios gerados, além de apresentar oportunidades de trabalhos futuros.

2 REVISÃO DE LITERATURA

Nesta capítulo são apresentados conceitos teóricos fundamentais para a criação de jogos com intuito de aprendizado na área de computação, além de análises de projetos de jogos relacionados, apresentando não apenas os mecanismos, objetivos e contribuições de cada um para os conhecimentos de programação ou computação de modo geral, mas também a forma em que cada jogo contribuiu para o desenvolvimento do jogo apresentado no capítulo 3.

O resto deste capítulo está organizado da seguinte forma. Na Seção 2.1, é explicado o conceito e os pilares do pensamento computacional. A Seção 2.2 aborda a definição e vantagens da computação desplugada. A Seção 2.3 descreve o conceito de pseudocódigo e como pode ser usado para o aprendizado inicial de programação. Finalmente, a Seção 2.4 descreve os benefícios do uso de jogos para aprendizado, além de mostrar exemplos de projetos com um objetivo similar a este trabalho.

2.1 PENSAMENTO COMPUTACIONAL

O pensamento computacional representa uma habilidade universalmente aplicável que todos, não somente profissionais da computação, podem aprender e praticar. Ao longo dos seus trabalhos, Jeannette Wing, diretora do instituto de ciência de dados da Universidade de Columbia reconhecida mundialmente por promover o pensamento computacional, apresenta múltiplas definições como “uma metodologia para resolver problemas com o uso de fundamentos da computação e uma visão crítica” (WING, 2006), ou também como “o processo de pensamentos presentes ao formular problemas de forma que suas soluções eficazes possam ser realizados por uma máquina ou pessoa” (WING, 2014).

Considerando as décadas de sugestões de definições e representações, o pensamento computacional também é definido como uma capacidade criativa, crítica e estratégica usada pelo ser humano usa para identificar e resolver problemas considerando conhecimentos fundamentais da computação (BRACKMANN, 2017). Apesar da definição de pensamento computacional divergir bastante entre autores, e após diversas tentativas de conceituá-lo, ainda existem autores que acreditam que não saberemos o real significado ou como ter uma mensuração do mesmo (KURSHAN, 2016).

Como apresentado na Tabela 1, outra forma de melhor compreender o pensamento com-

putacional é através dos seus 4 pilares fundamentais. Visto que o seu objetivo principal está relacionado a resolução de problemas, estes pilares servem como guias de como organizar as etapas deste processo (TERRA, 2021).

Tabela 1 – Descrições e exemplos dos 4 pilares do pensamento computacional

Pilar	Descrição	Exemplo prático
Decomposição	Diante de uma situação, o problema é divido em fragmentos menores com intuito de facilitar a análise e resolução do mesmo.	Distribuição de capítulos e definição de objetivos no desenvolvimento de um projeto.
Reconhecimento de Padrões	Ao identificar compartilhamento de características, é possível perceber padrões e acelerar a resolução de problemas semelhantes.	Diagnosticar doenças a partir do reconhecimento dos sintomas e características do paciente.
Abstração	Criar representações do objetivo a ser resolvido, de modo a classificar e filtrar dados iniciais.	Representação de materiais através de cores em lixeiras de coleta seletiva.
Algoritmo	Sequência de etapas ou instruções, ou seja, o plano descrito em instruções, diagramas, pseudocódigo ou linguagens de programação para resolver o problema em questão.	Receita culinária onde demonstra instruções claras e concisas para cada etapa.

Fonte: (TERRA, 2021).

Devido à natureza do pensamento computacional em ser aplicável em áreas do conhecimento além da computação, já existem estudos e propostas (BARR; STEPHENSON, 2011) para agregá-lo ao currículo escolar K-12, modelo norte-americano do período correspondente ao ensino primário até o secundário (MASTERS, 2017). Uma implementação do pensamento computacional no sistema de ensino escolar exige esforços para mudanças em políticas educacionais e destinação de recursos para professores especializados, de modo a fornecer práticas apropriadas às idades dos estudantes. A concretização destas propostas pode trazer uma união entre a comunidade de ciência da computação e estudantes do ensino fundamental e médio, além de fornecer novos métodos de resolução de problemas aos estudantes. Os pilares mencionados e demais conceitos do pensamento computacional podem ser aplicados de modo a facilitar o aprendizado de disciplinas já existentes no currículo escolar. O estudo de Valerie Barr e Chris Stephenson apresenta um modelo com exemplos dessas aplicações, como os seguintes:

- **Abstração e matemática:** uso de variáveis de modo a identificar fatos essenciais em uma resolução de problema;
- **Estruturas de controle e estudos sociais:** escrever uma estória com ramificações;
- **Descomposição do problema e ciência:** realizar divisões e classificações de espécies;
- **Automação e linguagens:** recorrer a corretores ortográficos digitais;

2.2 COMPUTAÇÃO DESPLUGADA

A computação desplugada é definida como uma metodologia que busca o aprendizado de conhecimentos da computação de modo simples e interativo, porém sem a presença de computadores ou outros dispositivos de *hardware*, ou *software*. Através da aplicação do pensamento computacional, o raciocínio lógico pode ser estimulado através de desafios, jogos ou busca da solução de problemas. Embora esta habilidade possa ser comumente explorada de formas digitais, a abordagem de computação desplugada permite levar a prática de conceitos e ideias da computação para regiões e populações sem acesso a computadores e aparelhos eletrônicos semelhantes (SILVA RODRIGUES, 2017).

A interdisciplinaridade com estudantes do ensino fundamental ou médio é uma forma vantajosa de aplicar a computação desplugada, de modo a unir conhecimentos prévios de assuntos de disciplinas regulares com novos aprendizados da computação. Por exemplo, em um estudo em uma escola pública do estado da Bahia, alunos foram instruídos a unir o estudo de danças regionais de uma disciplina de artes, com a reprodução de algoritmos de ordenação (FERREIRA et al., 2015). A partir destes desafios e atividades, os participantes não apenas despertam novas ideias, mas também formam memórias marcantes de aprendizado, trazendo satisfação e revelando habilidades próprias que estavam apenas em desuso. É importante notar que os benefícios resultantes não são somente para aqueles que desejam seguir um caminho de estudo na computação, mas para todos, visto que os estímulos ao raciocínio lógico e resolução de problemas são aplicáveis em qualquer área do conhecimento (KAMINSKI; BOSCARIOLI, 2020; BELL, 2007).

2.3 PSEUDOCÓDIGO

Pseudocódigos se definem como uma forma genérica de expressar o raciocínio lógico em uma sequência de comandos, ou seja, um meio legível de descrever um algoritmo. O principal contraste do pseudocódigo em relação a linguagens de programação está na sua ausência de regras definidas: enquanto linguagens como JavaScript ou Python possuem regras precisas de sintaxe, o pseudocódigo busca para o usuário a compreensão do algoritmo e estudo da lógica. Devido a sua simplicidade e ausência de restrições, pseudolinguagens são comumente a primeira forma de aprendizado de lógica de programação em curso da computação. O estudo inicial através de pseudolinguagens introduzem conhecimentos essenciais presentes em lingua-

gens de programação, como o conceito de variáveis e seus tipos, estruturas de controle e laços de repetição (KRIGER, 2020; NOLETO, 2020). No Brasil, foi criado o Portugol, uma pseudo-linguagem baseada no idioma português, focada em facilitar o aprendizado de programação aos brasileiros, visto que linguagens de programação costumam possuir uma sintaxe com palavras do idioma inglês (MOTTA, 2019). Visto que pseudocódigos descartam sintaxes definidas em troca de um maior entendimento humano, esta é uma forma de apresentar lógica de programação em jogos educacionais.

2.4 JOGOS EDUCACIONAIS SOBRE PROGRAMAÇÃO

A criação de jogos envolve diversas decisões necessárias para o funcionamento do produto final, como a definição do gênero, plataforma (físico ou digital), mecânicas (ações individuais ou sistema de ações possíveis pelo jogador) (BOLLER, 2013), critérios para vencer, a presença ou não de uma história, personagens, artefatos (cartas, tabuleiros e outros componentes necessários para jogar), modos de interação entre os jogadores, entre outros. Para o desenvolvimento de um jogo educacional, existem critérios adicionais aos de jogos puramente lúdicos. Durante a concepção daquela categoria de jogo, é preciso estabelecer o objetivo de aprendizagem, ou seja, os conteúdos e assuntos que os jogadores deverão aprender ao jogar. Junto ao objetivo, deve haver a forma de *feedback* educacional para o jogador, que será como a informação será apresentada ao jogador, mostrando os seus erros e acertos (BATTISTELLA, 2016).

Considerando os aspectos envolvidos na criação de um jogo educacional, a computação desplugada pode ser aplicada de diversas formas, visando o aprendizado de conceitos de lógica, sintaxes de linguagens de programação ou outros assuntos da área de ciência da computação. A seguir serão apresentados alguns exemplos de jogos educacionais inseridos nestes objetivos de aprendizagem que serviram de referência e inspiração para o jogo desenvolvido neste trabalho.

- Summon the JSON: este jogo de até 4 jogadores, desenvolvido pela empresa Websoul, utiliza a característica que a mente humana possui em gravar informações através de imagens com mais facilidade e por mais tempo. Embora as mecânicas do jogo não estejam diretamente relacionadas ao aprendizado de lógica de programação, o criador reitera que a combinação de belíssimas ilustrações, visualizadas na Figura 1, com a apresentação de fragmentos de código auxiliam na memorização de funcionalidades específicas de lin-

guagens de programação. O jogo possui variações com baralhos temáticos das linguagens Python, JavaScript, C#, PHP, Java e C++, além de um específico para o *framework* React (SMYKOWSKI, 2021).

Figura 1 – Cartas do jogo Summon The JSON.

Fonte: (SMYKOWSKI, 2021).

- Coder Cards: este produto, apesar de não ser um jogo, trata-se de um baralho de cartas clássicas de poker onde os número e naipes de cada carta são representados tanto pelos seus símbolos originais, quanto por um nome de linguagem de programação e um trecho de código representado na linguagem mencionada. Esta apresentação, visualizada na Figura 2, traz curiosidade ao jogador e explora as diferenças de sintaxe entre linguagens e frameworks (MATHIEU LEGAULT, 2016).

- Coding Is Good: desenvolvido pela organização MathAndCoding, este jogo de cartas possui regras similares ao jogo "Super Trunfo" da empresa Grow. As cartas visualizadas na Figura 3 possuem desafios de lógica com trechos de código em Python, desta forma os dois jogadores devem responder os enunciados e em caso de acerto, adicionar a carta a uma pilha, assim o jogador com mais cartas na pilha de acertos vence. (MATHANDCODING, 2017).

- AlgoCards: disponível gratuitamente para download e impressão (BRACKMANN, 2021), este conjunto combina cartas com comandos de movimento como "gire à esquerda" e

Figura 2 – Cartas do produto Coder Cards.

Fonte: (MATHIEU LEGAULT, 2016).

Figura 3 – Cartas e embalagem do jogo Coding Is Good.

Fonte: (MATHANDCODING, 2017).

"para frente" com cartas de estruturas de lógica de repetição para permitir diversos jogos como "AlgoLabirinto", onde o jogador deve utilizar as cartas para criar instruções ao se mover num tabuleiro com obstáculos. Visualizado na Figura 4, estas cartas exploram os princípios básicos de lógica de programação de forma intuitiva e acessível.

O estudo das regras e mecânicas destes jogos educacionais servem como auxílio e referência para o processo de criação do jogo detalhado no próximo capítulo. A seguir será detalhado os aspectos em que cada jogo apresentado contribuiu para este processo. As variações do jogo "Summon The JSON" para diversas linguagens de programação ou *frameworks* contribuíram fundamentalmente para a ideia de baralhos temáticos para expansão do jogo. A forma de

Figura 4 – Logo e cartas do jogo AlgoCards.

Fonte: (BRACKMANN, 2021).

apresentar códigos em cartas teve inspiração nos produtos “Coder Cards” e “Coding is Good”, além do formato de jogo com perguntas e respostas presente no segundo. Finalmente, assim como no jogo “Algo Cards”, foi decidido que todos os elementos gráficos do jogo desenvolvido neste projeto seriam disponibilizados gratuitamente em arquivos prontos para impressão. Considerando o número de possíveis baralhos temáticos que ainda podem ser construídos no jogo “Summon The JSON”, e a falta de diversidade em linguagens apresentadas no jogo “Coding is Good”, este trabalho também busca solucionar esta limitação ao apresentar o desenvolvimento de um software que facilita a criação de novas cartas para qualquer tema. A partir da pesquisa destes jogos educacionais, foi possível coletar diversos elementos e mecânicas de jogo, de modo a utilizar as suas vantagens, prevenir limitações semelhantes e reforçar a eficácia dos objetivos de aprendizagem.

3 CODE RUSH

Com base nos pilares do pensamento computacional, na computação desplugada e nas limitações levantadas sobre jogos similares no Capítulo 2, este capítulo apresentará a metodologia e os detalhes do *Code rush*, uma proposta de jogo educacional na área da computação com objetivos de aprendizagem específicos para programação.

Este capítulo está organizado da seguinte forma. A Seção 3.1 apresenta a metodologia utilizada neste projeto, detalhando as etapas e estratégias escolhidas para construção dos elementos e regras do jogo desenvolvido, enquanto a Seção 3.2 detalha o produto final deste trabalho, o jogo de cartas *Code Rush*, descrevendo não apenas as cartas e baralhos elaborados, mas também um software implementado para facilitar o processo de criação do leiaute de cartas.

3.1 METODOLOGIA

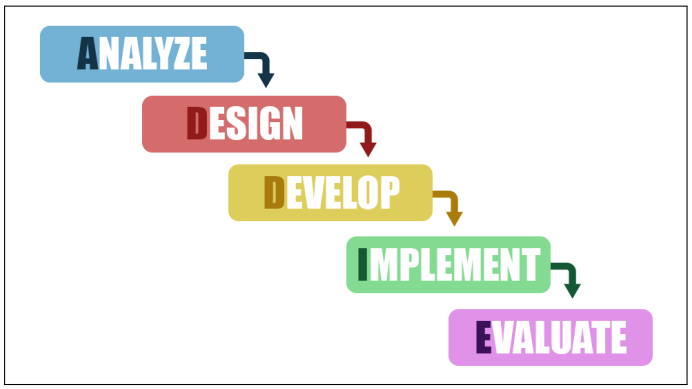
Esta seção apresenta a metodologia abordada durante as etapas de elaboração dos objetivos e regras do jogo, detalhando as referências utilizadas durante estes processos e suas conexões com o projeto implementado. Visto que o projeto se trata de um jogo de cartas que possuem perguntas e respostas sobre pseudocódigo e linguagens de programação específicas, esta Seção também detalha o processo de criação dos enunciados das cartas e suas respectivas classificações de complexidade. Além das descrições sobre a construção do jogo, também será relatado sobre o desenvolvimento do software para automatização de criação de leiaute de cartas.

3.1.1 Design instrucional

A construção do jogo, seus objetivos e regras seguiram uma metodologia inspirada no design instrucional (FILATRO, 2008), através do modelo ADDIE (*Analyze, Design, Develop, Implement* e *Evaluate*) (KURT, 2018) visualizado na Figura 5. O design instrucional se define como um processo sistemático de tradução de princípios de cognição e aprendizagem para o planejamento de materiais didáticos ou atividades, ou seja, possui como principal caso de uso o planejamento de atividades educacionais, porém neste projeto foi adaptado para o foco no recurso educacional desenvolvido, o jogo Code Rush. A partir do modelo ADDIE, inicia-se o

desenvolvimento do projeto pela etapa de análise com a definição dos objetivos de aprendizagem através da observação das necessidades e carências educacionais da área de conhecimento em questão.

Figura 5 – Etapas do modelo ADDIE.



Fonte: produção autoral.

Para este projeto foram analisados as formas de aprendizagem de lógica de programação e linguagens de programação, e assim foram definidos 2 objetivos de aprendizagem: reforçar conceitos básicos de lógica de programação através de pseudocódigos, e reforçar conhecimentos de sintaxe e funcionamento de linguagens de programação. Também nesta etapa, alunos com interesse em computação de ensino médio ou início do ensino superior foram definidos como principal público-alvo do jogo. A seguir, na etapa de design foi estabelecido as estratégias de ensino a serem aplicadas, as regras do jogo (conforme a Seção 3.1.3) e a definição dos objetivos da aplicação de automatização dos leiautes das cartas.

Durante o desenvolvimento, a partir das definições feitas nas etapas anteriores, foram elaborados os enunciados das cartas do jogo conforme a Seção 3.1.3 e o desenvolvimento do código da aplicação. Pela sua definição no modelo ADDIE, a fase de implementação é frequentemente caracterizada por ser um momento de treinar instrutores e facilitadores para o uso dos recursos educacionais desenvolvidos, assim como uma revisão do design criado até o momento, portanto neste projeto esta etapa é utilizada somente como momento de correções e preparações para a etapa de avaliação. Conforme o Capítulo 4, durante a etapa de avaliação é selecionado um guia de avaliação de jogos educacionais para aplicar os testes de jogabilidade, ou seja, sessões onde os convidados experimentam o jogo, na prática, e relatam suas experiências sobre a intuitividade e valor educacional do mesmo. Ao fim desta etapa também é realizado uma análise dos resultados obtidos pelos convidados.

3.1.2 Definição das regras e design do jogo

Com os objetivos de aprendizagem definidos e conforme a Seção 2.4, foram estudados outros jogos que tratam especificamente sobre programação. Considerando o objetivo de aprendizagem através de reforço a conhecimentos preestabelecidos de lógica ou linguagens de programação, foi escolhido o formato de perguntas e respostas como principal mecânica do jogo. O formato físico e desplugado foi preferido ao digital não somente pela capacidade de uso em lugares sem acesso a computadores, mas também por permitir uma fácil expansão dos assuntos do jogo através da aplicação detalhada na Seção 3.2.4. A partir dos objetivos de aprendizagem introduzidos no jogo foram definidos os elementos físicos, modos de interação dos jogadores, critérios para vencer e feito a modelagem do jogo (organização da sequência de todas jogadas realizadas pelos jogadores).

3.1.3 Criação dos enunciados

Para a elaboração do enunciado e a respectiva dificuldade de cada carta do jogo, alguns documentos preexistentes e relacionados ao tema de cada baralho foram estabelecidos como referência. No caso do baralho principal, cujo tema abrange pseudocódigos e lógica de programação, foram consultadas as ementas de "Lógica e Algoritmo[1]" e "Laboratório de Programação I[2]", disciplinas obrigatórias do primeiro semestre das graduações em "Bacharelado em Ciência da Computação" e "Bacharelado em Sistemas de Informação" da Universidade Federal de Santa Maria.

Conforme a Tabela 2, a dificuldade de cada carta deste baralho foi estabelecida a partir dos assuntos tratados no enunciado da carta e considerando a ordem de assuntos respectivos nas ementas mencionadas, de modo que os conteúdos iniciais são considerados de menor dificuldade e as cartas de maior complexidade possuem não apenas conteúdos mais avançados, como também uma combinação com os anteriores.

Para os baralhos específicos de linguagem de programação como Java e JavaScript, foram utilizados como referência websites de documentação destas linguagens como Mozilla[3], W3Schools[4] e DevDocs[5], seguindo a mesma relação anterior de complexidade das cartas e or-

[1] ufsm.br/ementario/disciplinas/ELC1064
[2] ufsm.br/ementario/disciplinas/ELC1065
[3] developer.mozilla.org
[4] w3schools.com
[5] devdocs.io

Tabela 2 – Assuntos correspondentes às dificuldades das cartas de pseudocódigo.

Nível de Dificuldade	Assuntos
1	Tipos de variáveis
2	Estruturas de controle (if) e operadores lógicos
3	Laços de repetição (while, for, do)
4	Vetores e múltiplos laços de repetição
5	Matrizes e combinações dos assuntos anteriores

Fonte: Produção autoral.

dem de assuntos nas documentações. Apesar de não ser possível definir precisamente quais linguagens de programação são mais essenciais de ter domínio, as duas linguagens definidas para possuírem baralhos específicos neste projeto foram escolhidas por estarem nas primeiras colocações nos índices de popularidade de linguagens de programação (YUGE, 2021) e possuírem diferentes casos de uso mais comuns (Javascript é frequentemente utilizado em desenvolvimento *web* enquanto Java está mais presente em aplicações *desktop* e *mobile*).

3.2 PROJETO E IMPLEMENTAÇÃO

A partir dos conceitos explorados no Capítulo 2 e da aplicação da metodologia apresentada na Seção 3.1, foi desenvolvido o “Code Rush”, um jogo de cartas físico de dois jogadores onde cada jogador deve responder uma sequência de perguntas de múltipla escolha sobre pseudolinguagens ou linguagens de programação visando acertar a resposta e assim obter mais pontos que o adversário.

Esta Seção aborda o jogo desenvolvido, detalhando suas regras e itens inclusos, como o leiaute das cartas e os baralhos temáticos. Além disso, apresenta-se uma aplicação de automatização na criação dos arquivos de imagens das cartas, buscando facilitar a expansão dos baralhos.

3.2.1 Regras do jogo

Para a preparação do jogo, as cartas do baralho deverão ser separadas em 5 pilhas conforme o nível de dificuldade de 1 a 5 representado em cada carta. Para iniciar o jogo deve-se decidir quem irá começar. Visto que não há vantagens em ser o primeiro ou segundo jogador, não há necessidade de realizar uma decisão por sorteio aleatório. O jogo possui 5 rodadas de dificuldade crescente onde cada jogador possui seu turno e após o fim da última rodada, em caso de empate de pontos, haverá também uma sequência de rodadas especiais denominada “Morte Súbita”.

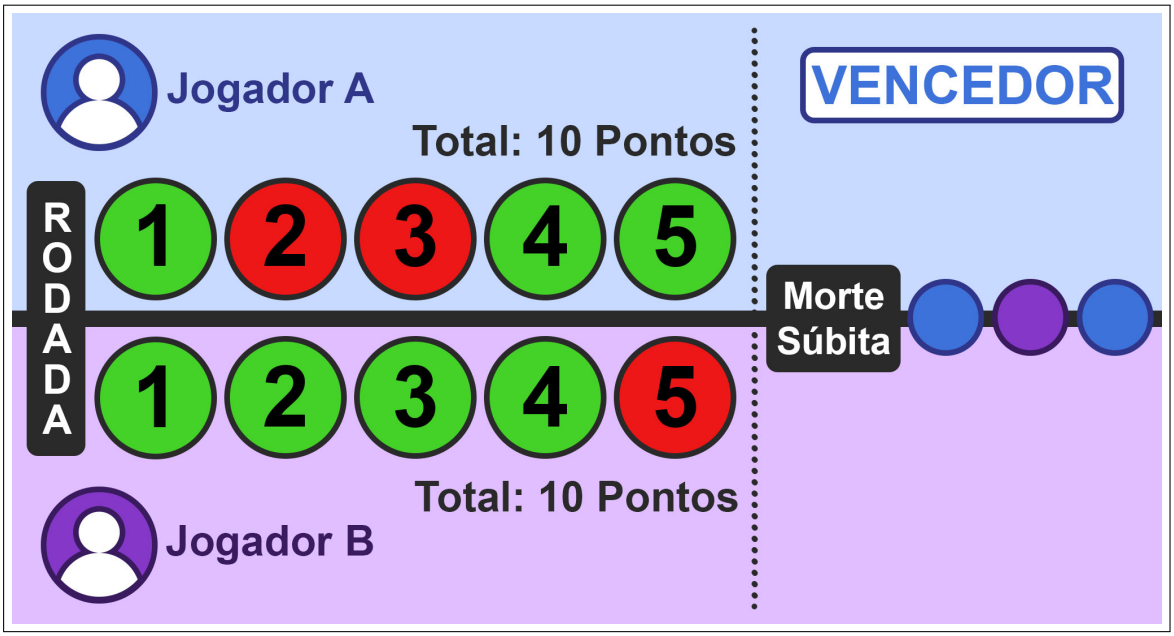
No início do turno, o jogador atual deverá sacar uma carta da pilha correspondente a dificuldade da rodada atual de forma que não observe a resposta no verso e em seguida deverá declarar a sua resposta para o enunciado da carta. Após a declaração, será revelada a resposta no verso da carta, em caso de acerto o jogador guarda a carta e recebe pontos equivalentes ao número da rodada atual, caso contrário não recebe e a carta é descartada. Uma versão inicial deste jogo usou limitações de tempo de resposta aos jogadores decrescentes conforme as rodadas, porém a mecânica foi descartada por interferir na efetividade dos objetivos de aprendizagem. O jogo então segue alternando entre jogadores até o fim da rodada 5, onde deverá ser comparado os pontos totais de cada jogador, caso não há empate o jogador com mais pontos é declarado o vencedor, caso contrário o jogo segue para a fase final.

Na fase de “Morte Súbita” haverá uma sequência de 3 rodadas utilizando as cartas de nível 5, onde cada jogador deve competir para responder o enunciado antes de seu adversário.

Caso o jogador que declarar a resposta primeiro estiver correto, ele recebe 1 ponto, caso contrário, o adversário recebe 1 ponto. O jogador que conseguir dois pontos nesta fase será declarado o vencedor da partida.

Conforme a Figura 6 é possível ver um exemplo de esquema simulando o resultado de um jogo. Neste exemplo o jogador A acerta durante as rodadas 1, 4 e 5, porém erra nas rodadas 2 e 3, enquanto o jogador B acerta todas exceto a rodada 5. Desta forma, ambos jogadores possuem 10 pontos, por consequência o jogo passa para a fase de "Morte Súbita", onde o jogador A acerta a primeira e terceira rodada, conseguindo assim a vitória da partida.

Figura 6 – Esquema simulando as rodadas de um jogo.



Fonte: Produção autoral.

3.2.2 Cartas

Conforme a Figura 7, onde a imagem a esquerda representa a frente e a direita o verso, cada carta possui os seguintes componentes:

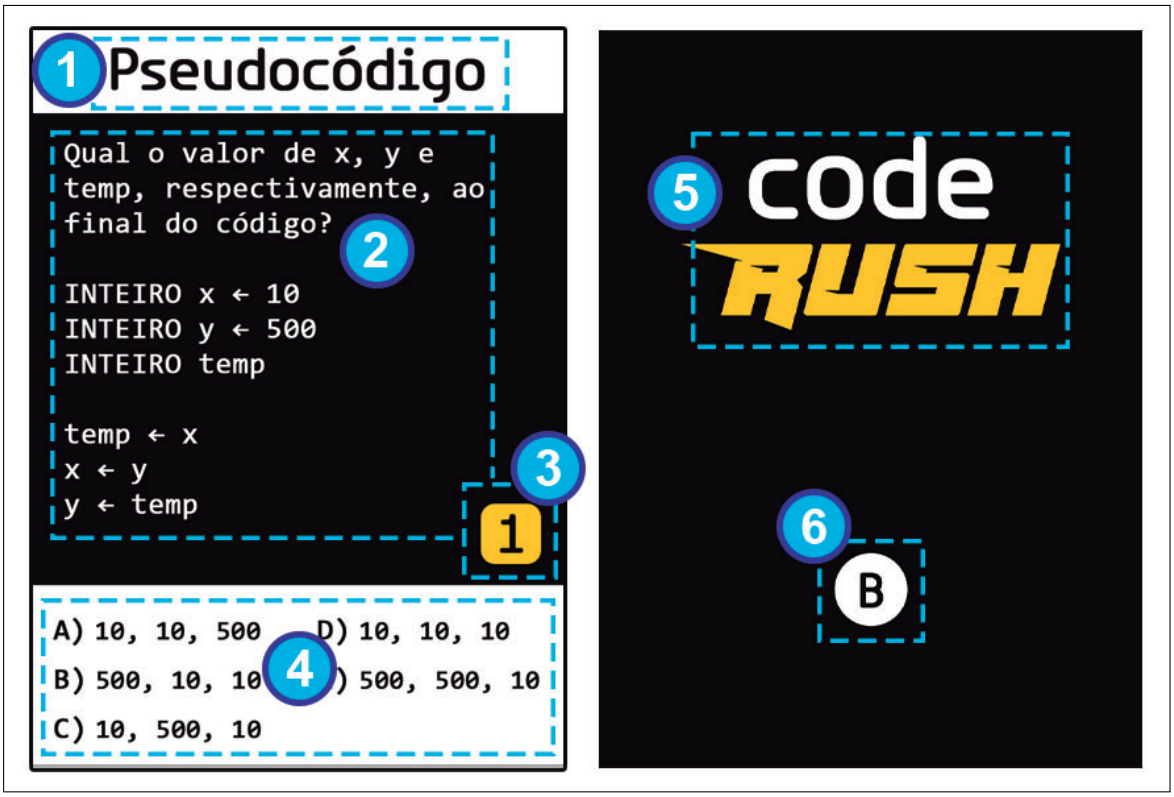
1. **Título:** nome do baralho temático a qual a carta pertence. Exemplos: Pseudocódigo, JavaScript, Java;
2. **Enunciado:** texto contendo a pergunta e código necessário;
3. **Dificuldade:** grau de dificuldade da carta, utilizado na rodada correspondente ao número. Seu número varia de 1 a 5;

4. **Alternativas:** opções de resposta ao enunciado da carta. Todas cartas possuem apenas 5 alternativas (“A”, “B”, “C”, “D”, e “E”);

5. **Logo:** título do jogo;

6. **Resposta:** letra correspondente a alternativa correta da carta.

Figura 7 – Leiaute de uma carta do jogo

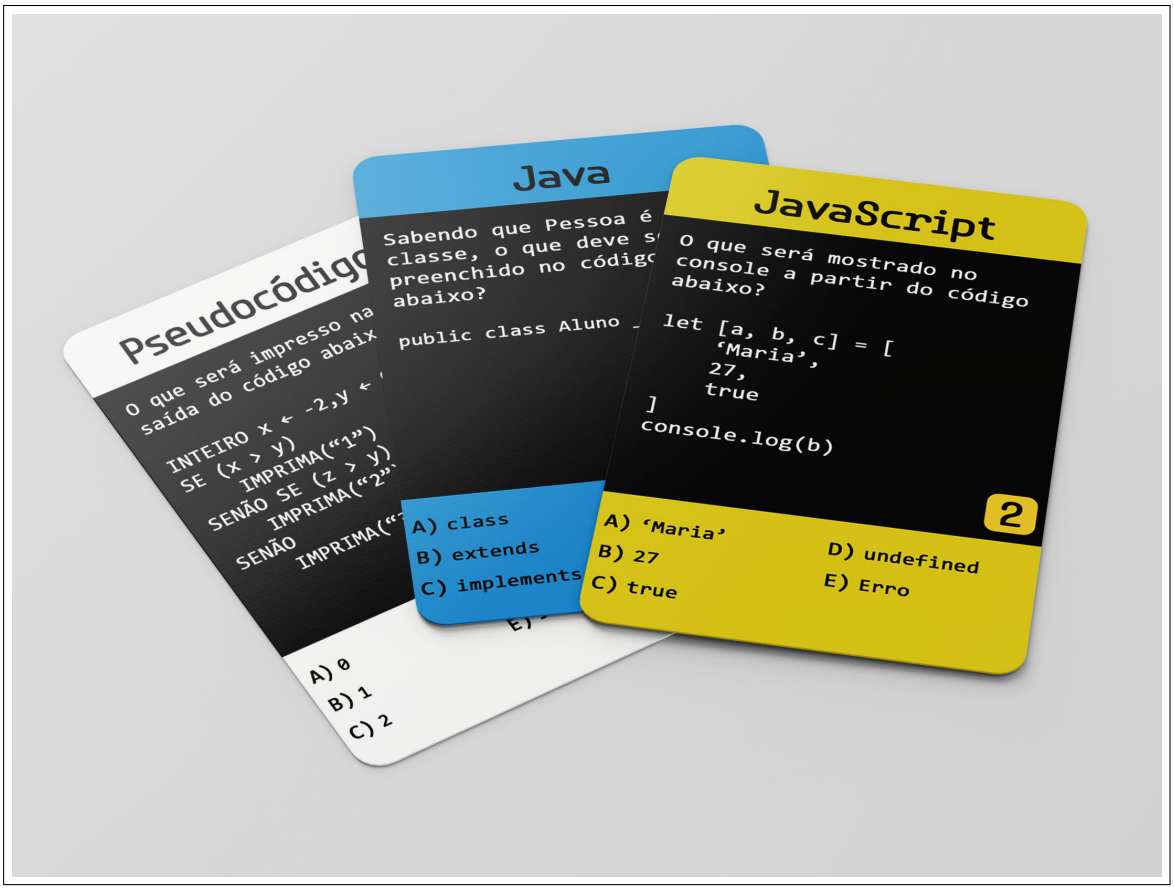


Fonte: Produção autoral.

3.2.3 Baralhos temáticos

Para jogar Code Rush é possível utilizar um ou mais de seus baralhos temáticos simultaneamente, de forma que cada um possui um tema correspondente a uma linguagem de programação ou pseudolinguagem diferente. Conforme os exemplos de cartas visualizados na Figura 8, este projeto contém a criação de 3 baralhos: um sobre pseudocódigo (construído a partir da pseudolinguagem Portugol mencionada na Seção 2.3) para aprendizado de lógica de programação contendo 35 cartas, e outros 2 sobre as linguagens Java e JavaScript contendo 25 cartas cada, onde seus enunciados abordam características e funcionalidades específicas das linguagens. A lista de todas as cartas e suas características (e.g. baralho, enunciado, alternativas, dificuldade e resposta) são apresentadas no Apêndice A em formato de arquivo CSV.

Figura 8 – Mockup de cartas de diferentes baralhos.



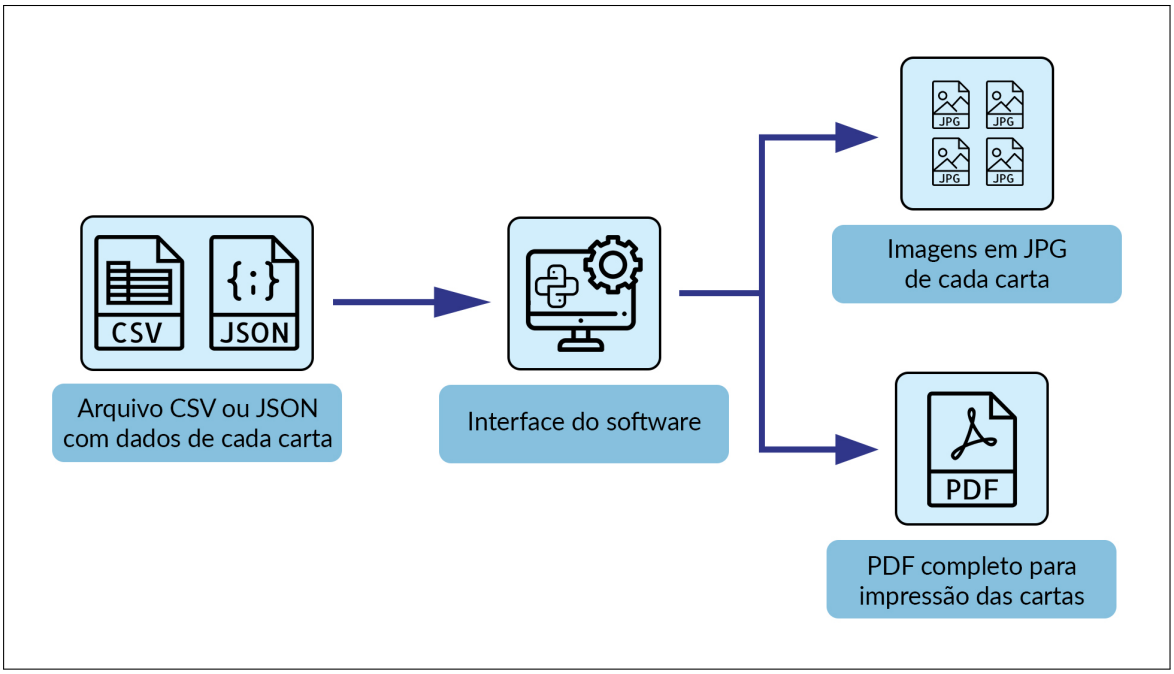
Fonte: Produção autoral.

3.2.4 Software de criação de baralhos

Para permitir uma fácil expansão do acervo de cartas e baralhos temáticos do jogo Code Rush por qualquer usuário interessado, foi desenvolvido um software capaz de gerar os arquivos necessários para a impressão de novas cartas seguindo o leiaute original das mesmas. Conforme o fluxograma visualizado na Figura 9 e a interface gráfica na Figura 10, o usuário deve preencher 3 campos de entrada:

- **Arquivo:** um arquivo em formato CSV ou JSON onde cada item deve possuir os campos “baralho”, “enunciado”, “alternativa_a”, “alternativa_b”, “alternativa_c”, “alternativa_d”, “alternativa_e”, “resposta” e “dificuldade”.
- **Pasta de destino:** o caminho do diretório no computador do usuário aonde os arquivos gerados serão gravados.
- **Cor de fundo:** através de um selecionador de cores é inserido o código hexadecimal da cor a ser utilizada no fundo da carta.

Figura 9 – Fluxograma representando entradas e saídas do software.



Fonte: Produção autoral.

Figura 10 – Interface gráfica do gerador de cartas.

Fonte: Produção autoral.

Para o envio dos arquivos em formato JSON ou CSV, todos os campos mencionados

anteriormente e exemplificados na Figura 11 devem possuir textos em formato de *string*. Para os campos de alternativas “A” até “E” e o de dificuldade, também é permitido o uso do tipo *number*, sem causar diferenças visuais nos arquivos gerados. Para os arquivos JSON, a validação destes campos e seus respectivos tipos são feitas através do JSON Schema visualizado na Figura 12, de modo que a entrada deverá ser um *Array* de objetos correspondentes às cartas com os campos mencionados. Em caso de envio de um arquivo com ausência de campos necessários ou possua campos preenchidos de modo inválido, o software emite um alerta sobre o erro para o usuário.

Para o campo “enunciado” foi estabelecido um limite de 30 caracteres por linha de texto e um máximo de 13 linhas, de forma que as quebras de linha devem ser declaradas explicitamente através do conjunto de caracteres especial “\n” (destacados nos enunciados do arquivo da Figura 11). Considerando estes limites e para garantir a fidelidade do leiaute esperado, foi utilizado no enunciado a fonte “Consolas”, uma fonte monoespaçada (todos os caracteres possuem a mesma largura). A mesma estratégia foi utilizada para os textos das alternativas, com um limite de 13 caracteres, e no título, com um limite de 14 caracteres utilizando a fonte “Typori Regular”. Os baralhos de pseudocódigo, JavaScript e Java utilizaram como cor de fundo os códigos hexadecimais “#FFFFFF”, “#DFCB2B” e “#3094D1” respectivamente.

Figura 11 – Exemplo de arquivo JSON e suas respectivas cartas geradas.

Fonte: Produção autoral.

Com todos os campos preenchidos, ao clicar no botão rotulado “Gerar” o programa irá fazer a leitura do arquivo com dados das cartas, criar um arquivo em formato JPG para cada carta e, conforme a Figura 13, gerar um arquivo PDF contendo até 9 imagens por página. Este arquivo PDF também possui páginas com os versos de cada carta dispostos de forma pronta para impressão gráfica em folhas de tamanho A3, incluindo as linhas de corte conforme a sangria de cada carta.

A aplicação foi desenvolvida utilizando a linguagem Python em sua versão 3.10 com o uso das bibliotecas *PySimpleGUI* e *Pillow* para criação da interface gráfica e geração dos arquivos de imagem, respectivamente. Ao final do desenvolvimento foi gerado um arquivo executável (formato *EXE*) através da biblioteca *py2exe* para garantir uma maior praticidade para aqueles que desejarem utilizar o programa. O código-fonte, o executável do programa e arquivos CSV e JSON de exemplo estão disponíveis no GitHub[6] junto a uma explicação de como utilizar o mesmo.

[6] https://github.com/TallesSC/CodeRush-card-generator

Figura 12 – JSON Schema para validar entrada de dados.

```
{
    "definitions": {},
    "\$schema": "http://json-schema.org/draft-07/schema\#",
    "title": "Root",
    "type": "array",
    "items":{
        "title": "Items",
        "type": "object",
        "required": ["baralho","enunciado","alternativa_a",
        "alternativa_b","alternativa_c","alternativa_d",
        "alternativa_e", "resposta", "dificuldade"],
        "properties": {
            "baralho": {
                "title": "Baralho",
                "type": "string"
            },
            "enunciado": {
                "title": "Enunciado",
                "type": "string"
            },
            "alternativa_a": {
                "title": "Alternativa_a",
                "type": ["string", "number"]
            },
            "alternativa_b": {
                "title": "Alternativa_b",
                "type": ["string", "number"]
            },
            "alternativa_c": {
                "title": "Alternativa_c",
                "type": ["string", "number"]
            },
            "alternativa_d": {
                "title": "Alternativa_d",
                "type": ["string", "number"]
            },
            "alternativa_e": {
                "title": "Alternativa_e",
                "type": ["string", "number"]
            },
            "resposta": {
                "title": "Resposta",
                "type": "string"
            },
            "dificuldade": {
                "title": "Dificuldade",
                "type": ["string", "integer"]
            }
        }
    }
}
```

Fonte: Produção autoral.

Figura 13 – Exemplo de páginas de um arquivo PDF gerado pelo programa.

Fonte: Produção autoral.

4 AVALIAÇÃO

Após a conclusão do desenvolvimento das regras e componentes do jogo de cartas, foram feitos testes de jogabilidades com o público-alvo, acompanhados de um questionário com o intuito de verificar a eficácia do jogo para os objetivos de aprendizagem. Neste capítulo, a Seção 4.1 descreve as técnicas utilizadas para esta avaliação somativa, ou seja, com intuito de apresentar os benefícios da versão final do jogo, enquanto a Seção 4.2 apresenta uma análise dos resultados obtidos durante esta etapa.

4.1 TESTES COM USUÁRIOS

Para a realização dos testes de jogabilidade, foram convidados alunos do primeiro e segundo ano do curso de Informática para Internet, integrado ao Ensino Médio do Colégio Técnico Industrial de Santa Maria (CTISM) e sob docência do professor Henry Cagnini. Os alunos do primeiro ano cursaram as disciplinas de Lógica e Algoritmos, onde a linguagem de programação C é ensinada, e Projeto e Desenvolvimento de Interfaces para Internet, que envolve programação em Javascript. Os alunos do segundo ano, além destas duas disciplinas, também cursaram Desenvolvimento de Sistemas para Internet, e Internet das Coisas, onde obtiveram experiência em programação em Python.

De um universo de 57 alunos convidados, 16 compareceram ao experimento no dia 6 de janeiro de 2022, realizado em uma sala de aula da escola, supervisionado pelo professor Henry Cagnini. Os alunos receberam folhas de papel com as instruções do jogo, disponível no Apêndice B, bem como foram permitidos a realizarem pelo menos uma partida de teste, para aprenderem as regras de forma prática. Foram utilizados quatro baralhos no experimento: três baralhos na temática pseudocódigo, e um baralho específico sobre JavaScript, distribuídos entre as partidas de forma aleatória e simultânea. A seguir, devido ao número de participantes e baralhos disponíveis, foi decidido que seria realizado um campeonato entre os participantes, onde o ganhador de cada partida iria progredir para as próximas etapas, em um formato de campeonato "mata-mata", apresentado na Figura 15. Os alunos também foram motivados a aprender o jogo pela gratificação com brindes (no caso bombons) conforme o progresso individual nas chaves do campeonato.

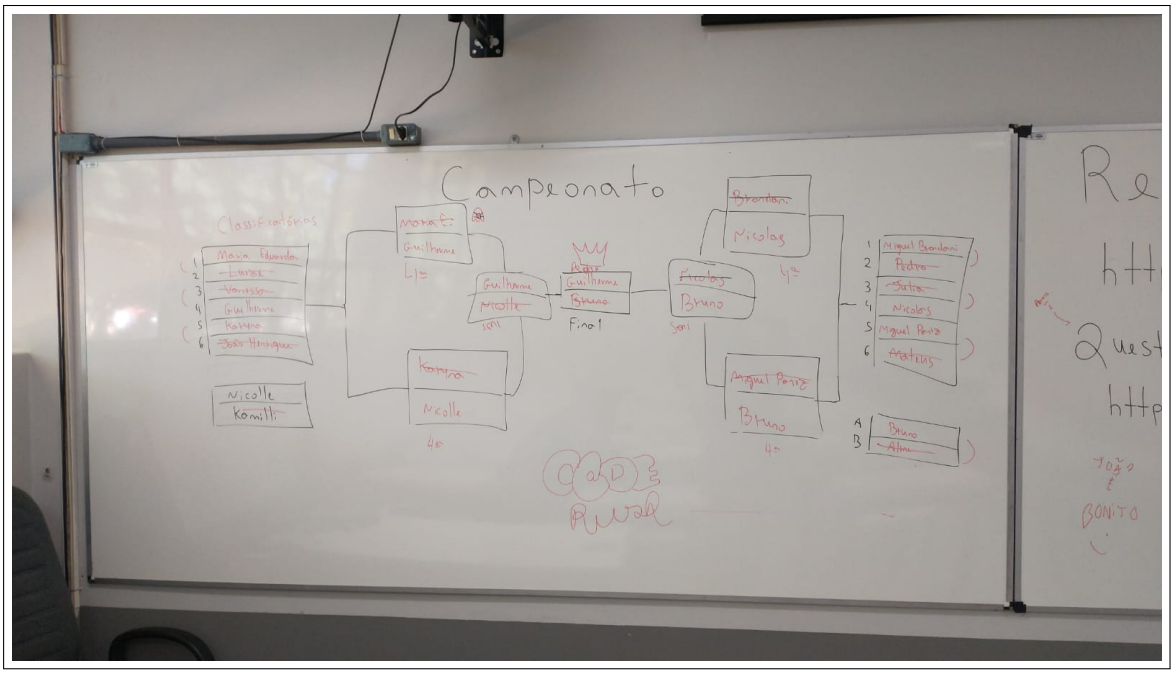
Após a sessão de partidas, os usuários foram solicitados a responder um questionário

construído através da plataforma *Google Forms*. As perguntas do questionário foram desenvolvidas a partir da metodologia MEEGA+ (*Model for the Evaluation of Educational GAmes*) (SAVI; WANGENHEIM; BORGATTO, 2011; PETRI; WANGENHEIM; BORGATTO, 2020), onde as alternativas para as respostas seguem uma escala *Likert* de 5 pontos de concordância (variando de "discordo totalmente" a "concordo totalmente") e englobam 9 dimensões: usabilidade (subdividido nas subdimensões de estética, aprendizibilidade, operabilidade e acessibilidade), confiança, desafio, satisfação, interação social, diversão, atenção focada, relevância e aprendizagem percebida. Visto que este é um jogo físico, e conforme as orientações do MEEGA+, itens da subdimensão "Proteção contra erros do usuário" foram descartados. Seguindo as orientações do MEEGA+ e visando investigar a contribuição do jogo para os objetivos de aprendizagem, além das perguntas voltadas às dimensões mencionadas, foram adicionadas perguntas referentes a estes conhecimentos específicos como parte da dimensão de aprendizagem percebida, e um campo de texto para críticas, sugestões ou elogios adicionais ao final. A listagem dos itens do questionário, junto a suas dimensões e subdimensões correspondentes estão disponíveis no Apêndice C deste trabalho.

Figura 14 – Foto dos alunos do CTISM jogando Code Rush.

Fonte: Produção autoral.

Figura 15 – Foto do desenho da chave do campeonato de Code Rush.

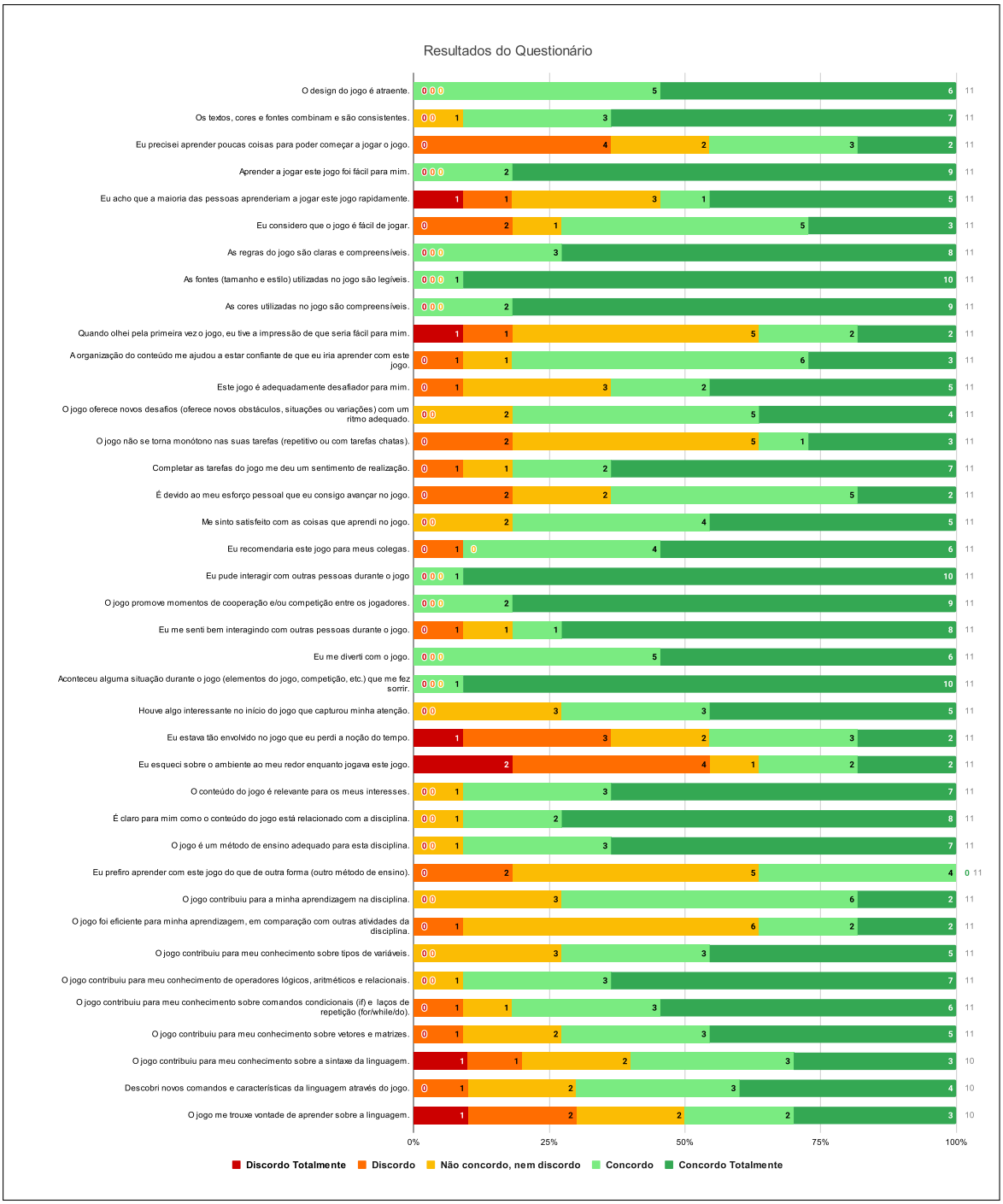


Fonte: Produção autoral.

4.2 RESULTADOS OBTIDOS

Dos 16 alunos participantes, foram obtidas 11 respostas no questionário. Considerando as respostas na escala Likert mencionada, e representando a total discordância como 1 ponto e total concordância como 5 pontos, podemos considerar que uma média superior a 4 significa uma boa avaliação, visto que a interpretação dos dados está diretamente ligada a construção do formato de resposta dos itens do formulário (SAVI; WANGENHEIM; BORGATTO, 2011). As frequências de cada resposta estão apresentadas na Figura 16.

Figura 16 – Resultados do questionário.

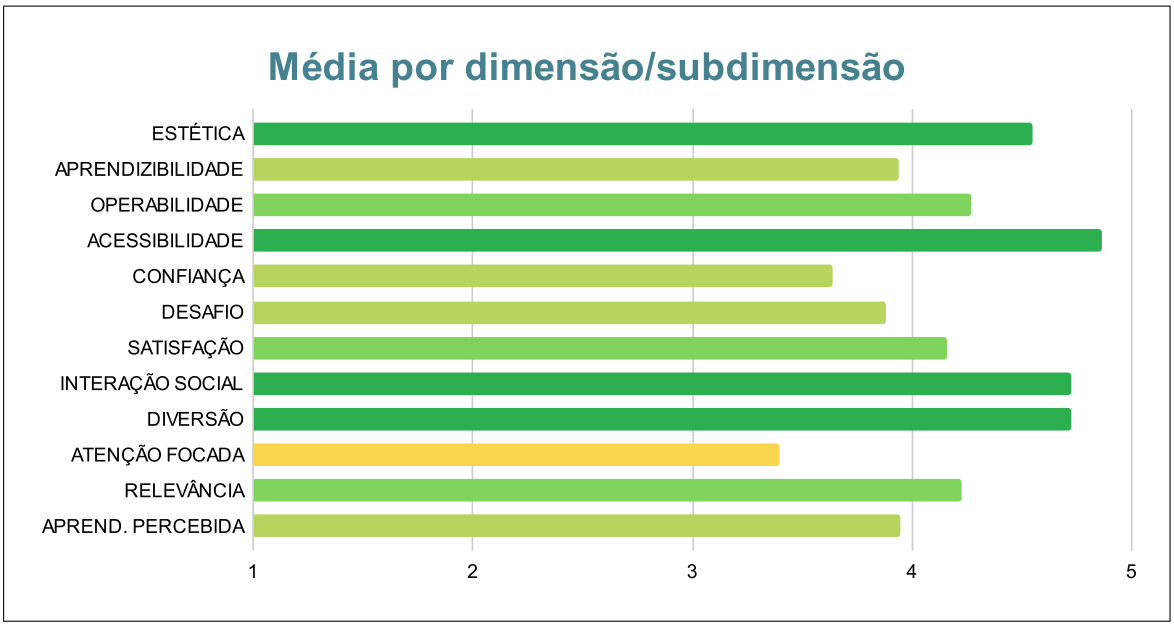


Fonte: Produção autoral.

Conforme os dados do gráfico na Figura 17, é possível visualizar o desempenho geral da avaliação em cada dimensão ou subdimensão. Através da média das respostas, nota-se o destaque das dimensões de estética, acessibilidade, interação social e diversão, seguidos pela boa avaliação dos campos ligados a operabilidade, satisfação e relevância. As dimensões de aprendizibilidade, confiança, desafio e aprendizagem percebida obtiveram um desempenho razoável,

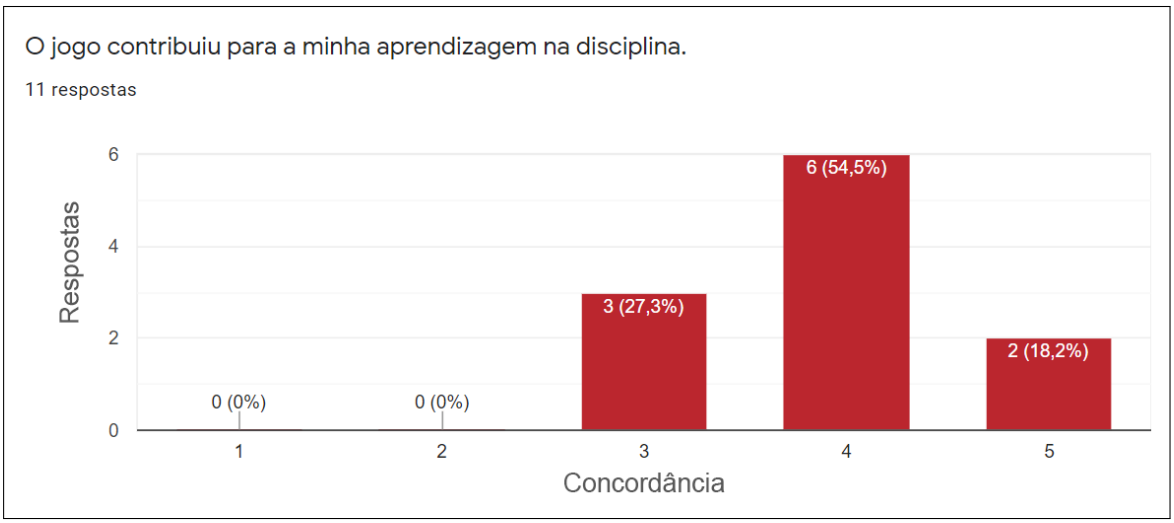
com uma média próxima aos 4 pontos. Finalmente, embora a dimensão de atenção focada ter obtido a menor média, a mesma está entre 3 e 4 pontos, considerando-se como regular. A partir dos resultados de avaliação sobre aprendizagem percebida e conhecimentos específicos de pseudocódigo e JavaScript visualizados nos gráficos das Figuras 17, 18 e 19, podemos concluir que os objetivos de aprendizagem do jogo obtiveram sucesso.

Figura 17 – Gráfico de média de avaliação por dimensão e subdimensão.



Fonte: Produção autoral.

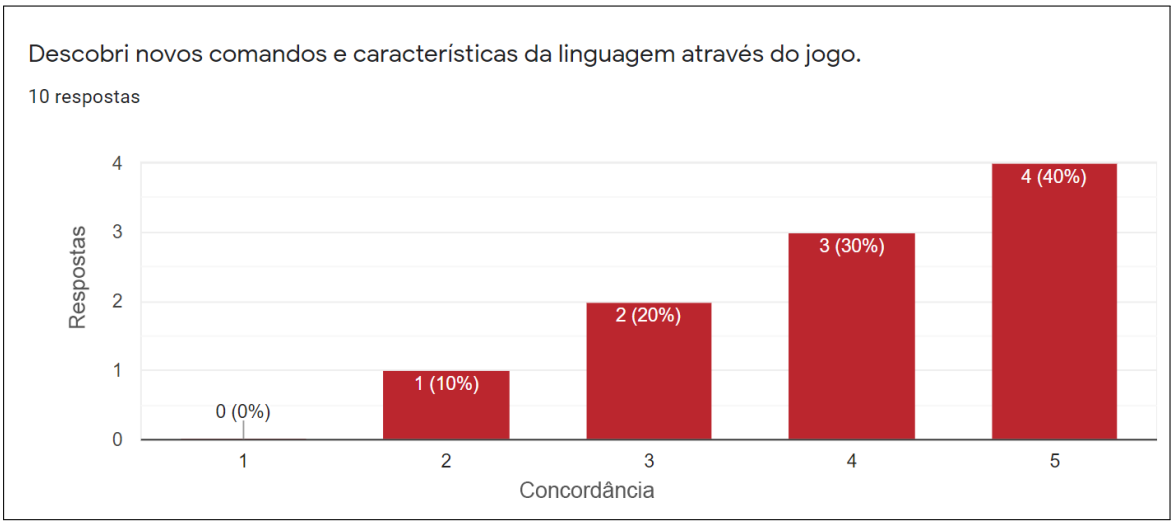
Figura 18 – Gráfico de frequência respostas do item "O jogo contribuiu para a minha aprendizagem na disciplina".



Fonte: Produção autoral.

Conforme o gráfico visualizado na Figura 20, houve uma divisão nas respostas em relação a aprendizibilidade do jogo. Neste item, 45,5% declararam que precisam aprender poucas

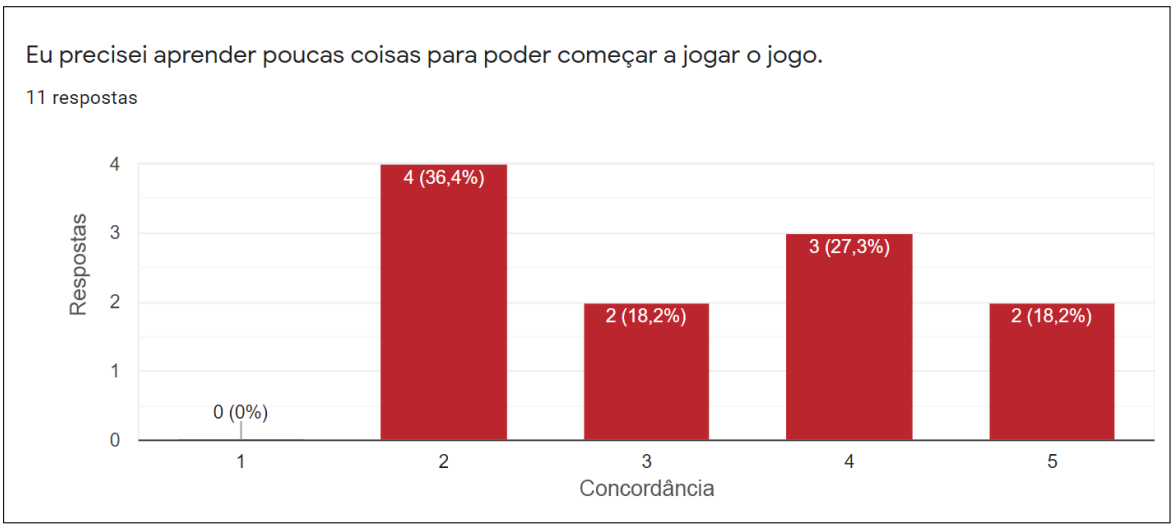
Figura 19 – Gráfico de frequência respostas do item sobre JS “Descobri novos comandos e características da linguagem através do jogo”.



Fonte: Produção autoral.

coisas para poder começar a jogar o jogo, enquanto 36,4% discordou. Entretanto, segundo o gráfico da Figura 21, todos os participantes concordaram que aprender as regras do jogo foi uma tarefa fácil, portanto as regras do jogo possuem uma intuitividade satisfatória quando consideramos apenas jogadores pertencentes ao público-alvo definido, visto que é necessário um conhecimento prévio extenso para jogar o Code Rush.

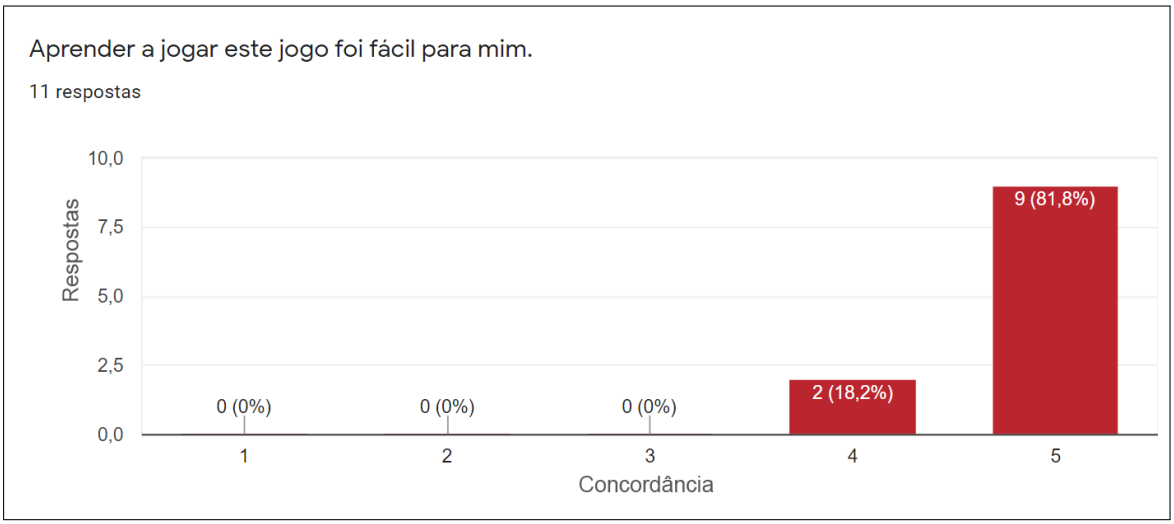
Figura 20 – Gráfico de frequência respostas do item “Eu precisei aprender poucas coisas para poder começar a jogar o jogo”.



Fonte: Produção autoral.

Como visualizado no gráfico da Figura 22, 18,2% discordou sobre o jogo não se tornar monótono, enquanto 45,5% respondeu que não concordam nem discordam. Considerando estes dados e a resposta de campo de texto livre ao fim do questionário, onde o participante declara

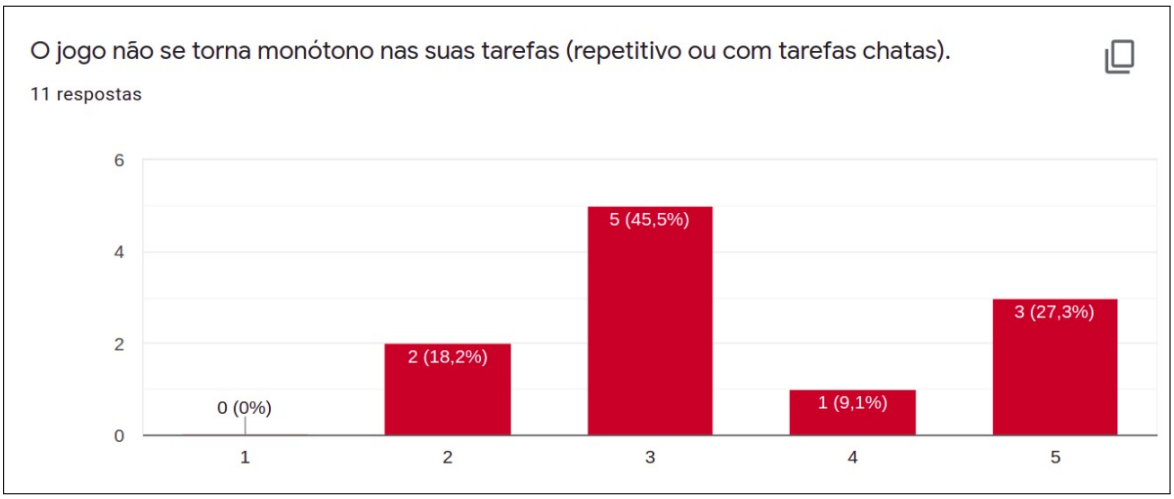
Figura 21 – Gráfico de frequência respostas do item “Aprender a jogar este jogo foi fácil para mim”.



Fonte: Produção autoral.

“As perguntas se tornam repetitivas depois de algumas rodadas”, é notável que o jogo se torna repetitivo ao jogar diversas partidas, visto que os participantes memorizavam as respostas de cartas encontradas em partidas anteriores. Felizmente, esta queixa que impacta diretamente na dimensão de desafio do jogo, é facilmente solucionada ao expandir o acervo de cartas e baralhos temáticos com o uso da aplicação detalhada na Seção 3.2.4.

Figura 22 – Gráfico de frequência respostas do item “O jogo não se torna monótono nas suas tarefas (repetitivo ou com tarefas chatas)”.



Fonte: Produção autoral.

Durante o experimento e conforme os resultados do campeonato, foi possível observar, embora subjetivamente, uma relação entre o desempenho e a divisão dos alunos de primeiro e segundo ano. Nesta relação, foi notável que os alunos do segundo ano do ensino médio

obtiveram mais sucesso durante as partidas, e por consequência chegaram, em sua maioria, mais próximos do primeiro lugar no campeonato. Dos 16 participantes do campeonato, 9 (56,25%) eram do segundo ano e 7 (43,75%) do primeiro, porém nas quartas e semifinal, a proporção de alunos do primeiro caiu para 25%, e a final foi disputada por 2 alunos do segundo ano. A hipótese provável sobre o ocorrido, levantada pelo professor supervisor, está ligada ao fato dos alunos mais veteranos já possuírem mais conhecimentos sobre programação.

Outra observação está em relação à preferência dos baralhos temáticos, os alunos, em sua maioria, consideraram o baralho de JavaScript como mais difícil, principalmente devido aos conhecimentos mais específicos que eram necessários para responder corretamente diversas cartas deste conjunto. Embora esta percepção por parte dos alunos ser válida, já que o principal objetivo dos baralhos de linguagens de programação é reforçar os conhecimentos de sintaxe da mesma, é proposital que este baralho seja desenvolvido para ser mais complexo. Caso os enunciados das cartas de baralhos de linguagens não tratassem de regras e sintaxes de forma detalhista e específica, haveriam muitas semelhanças entre os enunciados de cada baralho e consequentemente prejudicaria o objetivo de aprendizagem destes baralhos.

5 CONCLUSÃO

Os altos índices de reprovação e evasão em disciplinas introdutórias em cursos da computação, junto ao aumento na demanda de profissionais na área de informática, incentivaram o aumento de esforços e inovações nas técnicas de aprendizagem e ensinamento. Diante deste contexto, a aplicação de jogos educacionais para introduzir ou reforçar conceitos de programação a jovens iniciando estudos de computação têm se mostrado uma opção viável.

A partir desta justificativa, explorando os conceitos do pensamento computacional e da computação desplugada, este trabalho apresentou o desenvolvimento de uma nova proposta de jogo educacional voltado ao ensino de conceitos da computação. O jogo Code Rush foi construído para reforçar os conhecimentos de lógica e funcionalidades específicas de linguagens de programação através do formato de cartas com perguntas e respostas, ou seja, de forma simples, competitiva e divertida.

Além da construção do jogo, o desenvolvimento do software para geração de cartas (descrito na Seção 3.2.4) é outra contribuição deste trabalho, permitindo que o leitor interessado possa gerar novas cartas e baralhos temáticos, no formato da metodologia proposta neste trabalho.

Para validar os objetivos de aprendizagem propostos pelo Code Rush, foi realizado uma seção de testes de jogabilidade com alunos do CTISM, de modo a coletar a opinião dos participantes sobre o impacto do jogo no conhecimento dos seus assuntos abordados. A partir do questionário de avaliação, onde foi utilizado o modelo MEEGA+, foi revelado uma boa recepção do jogo e uma grande contribuição pro aprendizado dos alunos. Estética, acessibilidade, interação social e diversão foram dimensões de destaque nesta avaliação, de modo que o restante dos aspectos também demonstraram resultados satisfatórios. Considerando os resultados obtidos na avaliação deste projeto e a definição de um jogo educacional de qualidade como aquele possui objetivos bem definidos, motiva os jogadores a estudarem e fornece aprendizado de conteúdos através da diversão (SAVI; WANGENHEIM; BORGATTO, 2011), podemos considerar que o jogo Code Rush obteve sucesso em sua qualidade e propósito. A análise dos testes de jogabilidade também comprovaram positivamente o objetivo de estudar jogos educacionais como ferramenta de aprendizado na computação. Embora muitos destes jogos necessitam algum conhecimento prévio sobre o assunto do objetivo de aprendizagem, como o Code Rush, os mesmos se mostram como um grande reforço ao sistema de aprendizado clássico.

Finalmente, considerando o desenvolvimento do jogo, do software de geração de cartas, e os resultados obtidos na avaliação com usuários, é seguro dizer que este trabalho atingiu todos os seus objetivos propostos.

5.1 TRABALHOS FUTUROS

Embora este trabalho detalhe a construção de um jogo de cartas pronto para ser distribuído e experimentado, o processo de desenvolvimento permitiu o surgimento de possíveis trabalhos futuros para acrescentar melhorias ao jogo Code Rush e aprofundar os estudos da eficácia dos jogos educacionais. A seguir, serão apresentados algumas premissas iniciais para serem exploradas a partir deste trabalho.

Apesar do foco do jogo Code Rush na área de programação, as regras e elementos do jogo permitem expansão além do acervo de cartas e baralhos de outras linguagens de programação e *frameworks*. Com o uso do software de geração de cartas disponibilizado, é possível aplicar o jogo em qualquer área do conhecimento, como, por exemplo, em línguas estrangeiras, ao ensinar novas palavras e expressões, ou até mesmo em uma área não-correlacionada às linguagens, como a Medicina, no auxílio da memorização da correlação entre sintomas e doenças.

Considerando cartas com enunciados envolvendo códigos com variáveis e seus respectivos valores, é possível expandir o software de gerador dos arquivos para uma expansão do acervo de cartas ainda mais eficaz. Nesta expansão os códigos poderiam possuir componentes gerados aleatoriamente, como o valor de números inteiros, impactando numa quantidade infinita de variações de cada carta. Desta forma, a partir da aleatoriedade nos enunciados, as alternativas de resposta também deverão corresponder a lógica do código gerado. Esta abordagem também permitiria o uso de expressões ou gramáticas regulares na construção de *templates* para estes enunciados, de modo que uma pequena quantia de modelos conseguiriam gerar uma grande quantidade de cartas, eliminando consideravelmente o fator de repetitividade no jogo.

Finalmente, como forma de expandir o estudo da eficácia de jogos educacionais, a aplicação de novos testes de jogabilidade, com diferentes turmas, revelariam mais dados e conclusões. Considerando o ensino remoto sendo empregado na maioria das escolas e universidades durante os anos 2020 e 2021 (período em que este trabalho foi conduzido), a realização de testes com usuários pós-pandemia permitiria analisar o impacto deste formato de ensino nos conhecimentos e interesses dos alunos.

REFERÊNCIAS

ANDRÉ RAABA, A. Z.; BLIKSTEIN, P. **Computação na Educação Básica**: fundamentos e experiências. [S.l.]: Penso, 2020.

BARR, V.; STEPHENSON, C. Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community? **ACM Inroads**, [S.l.], v.2, 03 2011.

BATTISTELLA, P. E. **ENgAGED**: um processo de desenvolvimento de jogos para ensino em computação. 2016. Master's Thesis — UFSC.

BELL, T. **Computer Science Unplugged**. Acessado em janeiro/2022, "https://www.csunplugged.org/".

BLIKSTEIN, P. **O pensamento computacional e a reinvenção do computador na educação**. 2008.

BOLLER, S. **Learning Game Design**: game mechanics. Acessado em janeiro/2022, "http://www.theknowledgeguru.com/learning-game-design-mechanics/".

BRACKMANN, C. P. **Desenvolvimento do pensamento computacional através de atividades desplugadas na educação básica**. 2017. Master's Thesis — UFRGS.

BRACKMANN, C. P. **AlgoCards - Pensamento Computacional**. Acessado em janeiro/2022, "https://www.computacional.com.br/#AlgoCards".

COSTA MORA, L. M. M. G. e Michael da. Evasão na disciplina de algoritmo e programação: um estudo a partir dos fatores intervenientes na perspectiva do aluno. **Tercera Conferencia sobre el Abandono en la Educación Superior**, [S.l.], 2013.

FERREIRA, A. C. et al. Experiência Prática Interdisciplinar do Raciocínio Computacional em Atividades de Computação Desplugada na Educação Básica. **Anais do WIE**, [S.l.], 2015.

FILATRO, A. **Design Instrucional na Prática**. [S.l.]: Pearson, 2008.

FONTES, H. **Projeto Codifique inscreve alunos do ensino médio e monitores para curso de programação**. Acessado em Outubro/2021, "https://jornal.usp.br/universidade/projeto-codifique-inscreve-alunos-do-ensino-medio-e-monitores-para-curso-de-programacao/".

KAMINSKI, M. R.; BOSCARIOLI, C. Práticas de computação desplugada como introdução ao desenvolvimento do pensamento computacional nos anos iniciais do ensino fundamental. **#Tear: Revista de Educação, Ciência e Tecnologia**, [S.l.], v.9, n.2, dez. 2020.

KRIGER, D. **O que é Pseudocódigo, por que é importante e como escrever?** Acessado em janeiro/2022, "https://kenzie.com.br/blog/pseudocodigo/".

KURSHAN, B. **Thawing from a Long Winter in Computer Science Education**. Acessado em novembro/2021, "https://www.forbes.com/sites/barbarakurshan/2016/02/25/thawing-from-a-long-winter-in-computer-science-education/?sh=577c4716284d".

KURT, S. **ADDIE Model**: instructional design. Acessado em dezembro/2021, https://educationaltechnology.net/the-addie-model-instructional-design/.

MASTERS, N. **What is K-12 Curriculum**. Acessado em janeiro/2022, https://study.com/academy/lesson/what-is-k-12-curriculum.html.

MATHANDCODING. **Coding Is Good**. Acessado em outubro/2021, "http://www.mathandcoding.org/".

MATHIEU LEGAULT, P. L. **Coder Cards**. Acessado em dezembro/2021, "https://coderlife.io/".

MOTTA, D. **Portugol**: que idioma é esse? Acessado em janeiro/2022, "https://douglasmotta.com.br/2019/01/24/portugol-que-idioma-e-esse/".

NOLETO, C. **Pseudocódigo**: o que é e como é usado na programação? Acessado em janeiro/2022, "https://blog.betrybe.com/tecnologia/pseudocodigo/".

PETRI, G.; WANGENHEIM, C. Gresse von; BORGATTO, A. MEEGA+: um modelo para a avaliação de jogos educacionais para o ensino de computação. **Revista Brasileira de Informática na Educação**, [S.l.], v.27, p.52, 01 2020.

SAMPAIO, K. **USP oferece curso de programação para mulheres do ensino médio**. Acessado em outubro/2021, "https://agenciabrasil.ebc.com.br/educacao/noticia/2021-09/usp-oferece-curso-de-programacao-para-mulheres-do-ensino-medio".

SAVI, R.; WANGENHEIM, C. G. v.; BORGATTO, A. F. A Model for the Evaluation of Educational Games for Teaching Software Engineering. In: BRAZILIAN SYMPOSIUM ON SOFTWARE ENGINEERING, 2011. **Anais...** [S.l.: s.n.], 2011. p.194–203.

SILVA RODRIGUES, S. da. **Computação Desplugada no Ensino Fundamental I**: uma experiência metodológica numa escola pública na paraíba. 2017. Dissertação (Mestrado em Ciência da Computação) — .

SMYKOWSKI, T. **Summon The JSON**. Acessado em dezembro/2021, "https://summonthejson.com/products/summon-the-json-javascript-deck".

TERRA, R. **Pensamento Computacional e seus 4 pilares**. Acessado em janeiro/2022, https://www.makerzine.com.br/educacao/pensamento-computacional-e-seus-4-pilares/.

WING, J. Computational Thinking. **Communications of the ACM**, [S.l.], v.49, p.33–35, 03 2006.

WING, J. M. Computational Thinking with Jeannette Wing. **Columbia Journalism School**, [S.l.], 2014.

WING, J. M. PENSAMENTO COMPUTACIONAL – Um conjunto de atitudes e habilidades que todos, não só cientistas da computação, ficaram ansiosos para aprender e usar. **Revista Brasileira de Ensino de Ciência e Tecnologia**, [S.l.], v.9, n.2, 2016.

YUGE, C. **Qual é a linguagem de programação mais usada no mundo atualmente?** Acessado em janeiro/2022, https://canaltech.com.br/mercado/qual-e-a-linguagem-de-programacao-mais-usada-no-mundo-atualmente-200857/.

APÊNDICES

APÊNDICE A – Tabela de cartas

Baralho	Enunciado	A	B	C	D	E	RESPOSTA	Dificuldade
Pseudocódigo	Qual é a saída do código abaixo? INTEIRO i ← 5 ENQUANTO (i >= 0) FAÇA SE (i == 0) IMPRIMA(“*”) SENÃO SE (i % 2 == 0) IMPRIMA(“#”) SENÃO IMPRIMA(“_”) i ← i - 1	_#_#_*	*_#_#_	_#_#_#	#_#_#*	#_#_*	A	4
Pseudocódigo	Qual é a saída do código abaixo? INTEIRO i, j PARA (i ← 0; i <2; i++) PARA (j ← 0; i <2; j++) SE (i >j) IMPRIMA(“_”) SENÃO IMPRIMA(“#”)	###_	##_#	#__#	_#_#	###_	B	4
Pseudocódigo	Para que o código abaixo imprima “012345”, qual trecho está faltando? INTEIRO i ← 0 ENQUANTO (_____) FAÇA IMPRIMA(i) i ← i + 1	i >0	i <5	i <= 4	i >4	i <= 5	E	3
Pseudocódigo	Complete o código que identifica se um número é par ou ímpar. INTEIRO num LEIA (num) SE (_____) IMPRIMA(“PAR”) SENÃO IMPRIMA(“ÍMPAR”)	num / 2 == 1	num / 2 == 0	num * 2 == 1	num % 2 == 1	num % 2 == 0	E	2
Pseudocódigo	Complete o código que identifica se um triângulo de lados a/b/c é equilátero (lados iguais). REAL a, b, c LEIA (a, b, c) SE (_____) IMPRIMA(“É equilátero.”)	a != b && b != c	a == b && b == c	a >b && b >c	a != b || b != c	a == b || b == c	B	3
Pseudocódigo	Qual das expressões abaixo resulta em Verdadeiro considerando que A ← 0 e B ← 2 ?	(5 <A) || (B <3)	(5 >A) && (B <2)	(A >0) || (B <0)	(A >2) && (B >0)	(5 >A) && (B >2)	A	2
Pseudocódigo	Qual o tipo da variável x no código abaixo? _____ x ← Verdadeiro SE (x) IMPRIMA(“Sim”) SENÃO IMPRIMA(“Não”)	Inteiro	Real	Lógico	Caractere	Vetor	C	1

Pseudocódigo	Qual o tipo das variáveis raio e pi no código abaixo? _____ raio ← 5.75 _____ pi ← 3.14159 IMPRIMA(pi * raio * raio)	Inteiro	Real	Lógico	Caractere	Vetor	B	1
Pseudocódigo	Qual valor deverá ser preenchido para que o programa abaixo não entre em um loop infinito? INTEIRO num ← 5 ENQUANTO (_____) FAÇA IMPRIMA(num) num ← num - 1	Verdadeiro	n <= 5	n <6	n >0	n >0 || n <5	D	3
Pseudocódigo	Qual o valor de x, y e temp, respectivamente, ao final do código? INTEIRO x ← 10, y ← 500, temp temp ← x x ← y y ← temp	10, 10, 500	500, 10, 10	10, 500, 10	10, 10, 10	500, 500, 10	B	1
Pseudocódigo	Qual é a saída do código abaixo? INTEIRO i, num ← 4 PARA (i ← 0; i <5; i++) SE (i <num) IMPRIMA("-") SENÃO SE (i == num) IMPRIMA("=") SENÃO IMPRIMA("+")	+=—	–=++	—=+	—++	—-=	E	5
Pseudocódigo	O que será impresso no código abaixo? INTEIRO a ← 3, b ← 0, c ← 7 SE (a >b && a >c) IMPRIMA(a) SENÃO SE (b >a && b >c) IMPRIMA(b) SENÃO IMPRIMA(c)	0	3	7	10	Nada	C	2
Pseudocódigo	Qual a saída do código abaixo? INTEIRO vetor[4] ← [3, 1, 9, 8] INTEIRO maior ← 0 PARA (i ← 0; i <4, i++) SE (vetor[i] >maior) maior = vetor[i] IMPRIMA(maior)	3	0	9	8	4	C	5
Pseudocódigo	Qual a saída do código abaixo? INTEIRO vetor[4] ← [6, 7, 3, 1] INTEIRO menor ← 0 PARA (i ← 0; i <4, i++) SE (vetor[i] <menor) menor = vetor[i] IMPRIMA(menor)	6	7	3	1	0	E	5
Pseudocódigo	Qual número deverá ser preenchido no código abaixo para ser impresso "4321"? INTEIRO num ← ___ ENQUANTO (num >0) FAÇA IMPRIMA(num) num ← num - 1	0	1	2	3	4	E	3

Pseudocódigo	Qual número deverá ser preenchido no código abaixo para ser impresso “02468”? INTEIRO num ← 0 ENQUANTO (num <= ____) FAÇA IMPRIMA(num) num ← num + 2	0	2	6	8	10	D	3
Pseudocódigo	O que será impresso na saída do código abaixo? LÓGICO x ← Verdadeiro, y ← Falso SE (x && !y) IMPRIMA(“1”) SENÃO SE (y ‖ !x) IMPRIMA(“2”) SENÃO IMPRIMA(“3”)	1	2	3	Verdadeiro	Falso	A	2
Pseudocódigo	O que será impresso na saída do código abaixo? LÓGICO x ← Falso, y ← Verdadeiro SE (x && y) IMPRIMA(“1”) SENÃO SE (x ‖ !y) IMPRIMA(“2”) SENÃO IMPRIMA(“3”)	1	2	3	Verdadeiro	Falso	C	2
Pseudocódigo	Qual será o valor impresso ao final do código abaixo? INTEIRO i, fatorial ← 1 PARA (i ← 1; i <= 3; i++) fatorial ← fatorial * i IMPRIMA(fatorial)	6	5	3	1	0	A	3
Pseudocódigo	Qual será o valor impresso no código abaixo? INTEIRO vetor[4] ← [3, 6, 2, 7] IMPRIMA(vetor[1] + vetor[3])	3	13	9	5	67	B	4
Pseudocódigo	Qual será o valor impresso no código abaixo? INTEIRO vetor[3] ← [4, 1, 3, 8] IMPRIMA(vetor[1] * vetor[3])	20	1	15	8	4	D	4
Pseudocódigo	Qual será o valor impresso no código abaixo? INTEIRO vetor[4] ← [2, 8, 1, 5] IMPRIMA(vetor[0] - vetor[2])	0	-1	1	2	3	C	4
Pseudocódigo	Qual o valor impresso ao final do código abaixo? INTEIRO soma ← 0 INTEIRO vetor[5] = [2, 1, 5, 0, 3] PARA (i ← 0; i <5; i++) soma ← soma + vetor[i] IMPRIMA(soma)	6	8	10	11	13	D	5
Pseudocódigo	Qual o valor impresso ao final do código abaixo? INTEIRO soma ← 0 INTEIRO vetor[5] = [2, 0, -2, 4, 1] PARA (i ← 0; i <5; i++) soma ← soma + vetor[i] IMPRIMA(soma)	7	5	3	0	-2	B	5

Pseudocódigo	O que será impresso na saída do código abaixo? INTEIRO x ← 5, y ← 2, z ← 9 SE (x >z) IMPRIMA(“1”) SENÃO SE (z <y) IMPRIMA(“2”) SENÃO IMPRIMA(“3”)	1	2	3	5	9	C	2
Pseudocódigo	O que será impresso na saída do código abaixo? INTEIRO x ← -2, y ← 0, z ← 5 SE (x >y) IMPRIMA(“1”) SENÃO SE (z >y) IMPRIMA(“2”) SENÃO IMPRIMA(“3”)	0	1	2	3	5	C	2
Pseudocódigo	Para que o código abaixo imprima “789”, qual trecho está faltando? INTEIRO i ← 0 ENQUANTO i <10 FAÇA SE (_____) IMPRIMA(i) i ← i + 1	i <7	i >7	i <= 7	i >= 7	i != 7	D	3
Pseudocódigo	Qual é a saída do código abaixo? INTEIRO i ← 0 ENQUANTO (i <6) FAÇA SE (i <3) IMPRIMA(“a”) SENÃO IMPRIMA(“b”) i ← i + 1	aaabbb	aaaabb	bbaaaa	ababab	bbbaaa	A	3
Pseudocódigo	Qual é a saída do código abaixo? INTEIRO i, num ← 2 PARA (i ← 0; i <5; i++) SE (i <= num) IMPRIMA(“A”) SENÃO IMPRIMA(“B”)	AAAAB	AABBB	ABABA	ABBBB	AAABB	E	3
Pseudocódigo	Qual o tipo da variável x no código abaixo? _____ x ← ‘a’ SE (x == ‘b’) IMPRIMA(“b”) SENÃO IMPRIMA(“a”)	Inteiro	Real	Lógico	Caractere	Vetor	D	1
Pseudocódigo	Qual o valor de x, y e z, respectivamente, ao final do código? INTEIRO x ← 50, y ← 25, z z ← x + y x ← y y ← z - x	25, 50, 75	75, 25, 50	25, 75, 50	50, 50, 75	25, 25, 50	A	1

Pseudocódigo	Qual o valor impresso ao final do código abaixo? INTEIRO soma ← 0 INTEIRO matriz[2][2] = [[2, 6], [1, 4]] PARA (i ← 0; i <2; i++) PARA (j ← 0; j <2; j++) soma ← soma + matriz[i][j] IMPRIMA(soma)	10	13	8	12	11	B	5
Pseudocódigo	Qual o valor impresso ao final do código abaixo? INTEIRO soma ← 0 INTEIRO matriz[2][2] = [[0, 7], [5, 2]] PARA (i ← 0; i <2; i++) PARA (j ← 0; j <2; j++) soma ← soma + matriz[i][j] IMPRIMA(soma)	7	5	17	16	14	E	5
Pseudocódigo	Qual o valor impresso ao final do código abaixo? INTEIRO matriz[3][3] = [[5, 1, 2], [0, 2, 3], [6, 4, 0]] IMPRIMA(matriz[0][2] + matriz[1][1])	6	8	4	3	10	C	5
Pseudocódigo	Qual o valor impresso ao final do código abaixo? INTEIRO matriz[3][3] = [[1, 2, 0], [9, 3, 6], [2, 1, 5]] IMPRIMA(matriz[1][2] + matriz[2][0])	6	8	7	15	10	B	5
JS	Qual o tipo das variáveis nome e idade, respectivamente, no código abaixo? let nome = “Fulano” let idade = 27	Boolean e Number	String e Number	String e String	Number e Object	String e Null	B	

JS	Qual o tipo das variáveis endereco e pessoa,
	respectivamente, no código abaixo?
	
	const endereco = “Av. Roraima”
	const pessoa = {
	nome: “Fulano”,
	sobrenome: “De Tal”
	};
		String e Boolean	Number e Object	String e String	String e Object	String e Null	D	

JS	Qual o tipo das variáveis notas e aprovado, respectivamente, no código abaixo? let notas = [7, 9, 6, 10] let aprovado = true	Boolean e String	Number e Boolean	Object e Boolean	Object e String	Number e Object	C	

JS	O que será mostrado no console a
	partir do código abaixo?
	
	const pessoa = {
	nome: “Fulano”,
	sobrenome: “De Tal”
	
	};
	console.log(pessoa.nome)
		pessoa.nome	pessoa	Fulano De Tal	Fulano	nome	D	

JS	O que será mostrado no console a partir do código abaixo?
	
	function print(a = 5){
	console.log(a)
	}
	print()
	print(42)
		42 e 5	a e 42	5 e 42	e 42	5 e	C	

JS	O que deverá ser preenchido no código
	abaixo para não ocorrer um erro?
	
	if (true){
	_____ a = 5
	}
	console.log(a)
		let	var	const	function	object	B	

JS	O que será mostrado no console a partir do código abaixo?
	
	let a = 2
	if (true) {
	let a = 5
	console.log(a)
	}
	console.log(a)
		5 e 5	5 e 2	2 e 2	2 e 5	5 e "a"	B	

JS	O que será mostrado no console a partir do código abaixo? const idade = 42 idade = 45 console.log(idade)	42	45	idade	const	Erro	E	

JS	O que será mostrado no console a partir do código abaixo?
	
	const vetor = [3, 5, 7]
	vetor.foo = 'Olá'
	for (let i in vetor) {
	console.log(i)
	}
		0, 1, 2, foo	0, 1, 2, 3	3, 5, 7, 'Olá'	3, 5, 7, foo	0, 1, 2, 'Olá'	A	

JS	O que será mostrado no console a partir do código abaixo?
	
	const vetor = [0, 3, 9]
	for (let i of vetor) {
	console.log(i)
	}
		0, 1, 2	0, 3, 9	1, 2, 3	3, 2 ,1	9, 3, 0	B	

JS	O que será mostrado no console a partir do código abaixo? const a = [1, 2] const b = [3, 4] const c = [...a, ...b] console.log(c)	[1, 2, 3, 4]	[[1, 2], [3, 4]]	[a, b]	undefined	Erro	A	
JS	Qual será o valor de x e y a partir do código abaixo? const jogo = 'Code Rush' x = jogo.includes('Code') y =jogo.includes('rush')	tru e true	false e false	false e true	true e false	"Code"e "rush"	D	
JS	O que será mostrado no console a partir do código abaixo? console.log('Abc'.repeat(3))	AbcAbcAbc	AbcAbcAbcAbc	Abc	Abc3	Abc333	A	
JS	O que será mostrado no console a partir do código abaixo? const obj1 = {a: 1} const obj2 = {b: 2} const obj3 = Object.assign({}, obj1,obj2) console.log(obj3)	{a: 1, b: 2}	{}	{1, 2}	12	undefined	A	

JS	O que será mostrado no console a partir do código abaixo? const vetor1 = [0, 3, 7] const vetor2 = [...vetor1, 9] console.log(vetor2)	[0, 3, 7]	[[0, 3, 7], 9]	[0, 3, 7, 9]	[9]	undefined	C	
JS	O que será mostrado no console a partir do código abaixo? let [a, b, c] = [‘Maria’, 27, true] console.log(b)	‘Maria’	27	true	undefined	Erro	B	
JS	O que será mostrado no console a partir do código abaixo? const nome = “Ana” console.log(nome + 2 * 2)	Ana22	Ana4	Ana2Ana2	AnaAnaAnaAna	NaN	B	

JS	O que será mostrado no console a partir do código abaixo?
	
	if(0)
	console.log(‘a’)
	else if (“0”)
	console.log(‘b’)
	else
	console.log(‘c’)
		"a"	"b"	"c"	0	"0"	B	

JS	O que será mostrado no console a partir do código abaixo? let [a, b, ...resto] = [25, true, ‘Fulano’, false] console.log(resto)	"Fulano"	false e false	["Fulano", false]	[]	undefined	C	

JS	O que será mostrado no console a partir do código abaixo?
	
	let a = true;
	let b = false;
	[a, b] = [b, a];
	console.log(a)
	console.log(b)
		false e false	true e true	false e true	true e false	[true, false]	C	

JS	Qual o valor de nome a partir do código abaixo? const nome = “Fulano” nome = nome.slice(1, 3)	"Ful"	"Fu"	"ul"	"ula"	"ulan"	C	
JS	Qual será o valor de nome a partir do código abaixo? const nome = “Fulano” nome = nome.substr(2, 4)	"ula"	"ulano"	"la"	"lano"	"Fu"	D	
JS	Qual o valor da variável x no código abaixo? let x = !!“false” === !!“true”;	true	false	"true"	"false"	null	A	

JS	O que deve ser preenchido no código abaixo?
	
	_____ function f() {
	return 1;
	}
	
	f().then(alert);
		static	return	promise	await	async	E	

JS	Qual o valor da variável resultado no código abaixo? const soma = (a, b) =>a + b const resultado = soma(2, 6)	2	4	6	8	undefined	D	
Java	Sabendo que Animal é uma interface, o que deve ser preenchido no código abaixo? public class Gato _____ Animal	class	extends	implements	static	public	C	
Java	Sabendo que Pessoa é uma classe, o que deve ser preenchido no código abaixo? public class Aluno _____ Pessoa	class	extends	implements	static	public	B	

Java	O que deve ser preenchido no código abaixo?
	
	public _____ imprimeNumero(int n){
	System.out.println(“n = ” + n);
	}
		int	static	boolean	String	void	E	

Java	O que deve ser preenchido no código abaixo?
	
	_____ maiorQue(int x, int y){
	return x >y;
	}
		int	String	boolean	float	void	C	

Java	Qual será o valor de vetor ao fim do código abaixo? int vetor[] = {4, 8, 0, 2, 5}; Arrays.sort(vetor);	{4, 8, 0, 2, 5}	{5, 2, 0, 8, 4}	{8, 5, 4, 2, 0}	{0, 2, 4, 5, 8}	{2, 4, 5, 8, 0}	D	

Java	O que deve ser preenchido no código abaixo?
	
	_____ idade = 25;
	_____ altura = 1.76;
		String, boolean	char, boolean	char, int	String, char	text, boolean	E	

Java	O que será impresso no código abaixo?
	
	for (int i = 0; i <5; i++) {
	
	if (i == 3) {
	
	break;
	
	}
	
	System.out.print(i);
	
	}
		012345	01234	0124	0123	012	A	

Java	O que será impresso no código abaixo?
	
	for (int i = 0; i <5; i++) {
	if (i == 3) {
	continue;
	}
	
	System.out.print(i);
	}
		01245	0124	012	01234	012345	E	

Java	O que deve ser preenchido no código abaixo?
	
	public class Pessoa {
	
	int idade;
	
	public Pessoa(int idade){
	
	_____ = idade;
	}
	}
		Pessoa	int	this.idade	idade	Pessoa.idade	E	

Java	Preencha o código abaixo para que o método e atributo sejam acessíveis apenas dentro de sua classe.
	
	class Conta{
	____ double saldo;
	
	____ void sacar(double v){
	this.saldo -= v;
	}
	}
		public	private	protected	default	static	C	

Java	O que será impresso no código abaixo?
	
	ArrayList<String>carros = new ArrayList<String>();
	carros.add("Volvo");
	carross.add("BMW");
	carros.add("Ford");
	carros.add("Mazda");
	System.out.println(cars.get(2));
		carros	Volvo	BMW	Ford	Mazda	D	

Java	O que deverá ser preenchido no espaço abaixo?
	
	____ {
	int[] v = {1, 2, 3};
	System.out.print(v[5]);
	} catch (Exception e) {
	System.out.print("erro");
	}
		if	while	try	catch	finally	D	

Java	O que deverá ser preenchido no espaço abaixo?
	
	try {
	int[] v = {1, 2, 3};
	System.out.print(v[5]);
	} catch (Exception e) {
	System.out.print("erro");
	} _____ {
	System.out.print("mensagem");
	}
		if	while	try	catch	finally	E	

APÊNDICE B – Documento de regras do jogo

CODE RUSH

Regras de Jogo

Número de jogadores por partida: 2 (1 Vs 1)
Baralhos disponíveis: 3 de Pseudocódigo e 1 de JavaScript

Para preparar o jogo, selecione um ou mais dos baralhos disponíveis e separe as cartas em 5 pilhas conforme o nível de dificuldade representado na carta (quadrado amarelo com número de 1 a 5 na frente).

Antes de começar, decida qual jogador irá começar jogando (não há vantagens ou desvantagens em ser o primeiro a jogar).

O primeiro jogador começa sacando uma carta da pilha de primeira dificuldade (número 1) e deverá responder o enunciado com uma resposta de acordo com uma das alternativas na parte inferior da carta. Após declarar sua resposta o jogador revela a parte traseira da carta onde contém a resposta, caso esteja correto o jogador guarda a carta e recebe o número de pontos correspondente ao nível da rodada (neste caso 1 ponto), caso contrário não recebe pontos e a carta é descartada.

Em seguida o segundo jogador repete os passos do primeiro jogador. Após os dois realizarem seus turnos, os jogadores deverão subir para a próxima dificuldade e usar o próximo baralho (níveis 2, 3, 4 e por último 5). Os jogadores alternam os turnos sacando e respondendo cartas de cada nível até o nível 5, se um jogador possuir mais pontos que o outro ao fim desta rodada o jogador com mais pontos vence. Em caso de empate o jogo passa para a fase final de “Morte Súbita” ou “Rush”.

Nesta etapa final os jogadores deverão competir em uma “melhor de 3”, de forma que será sacado uma carta de nível 5 e ambos jogadores deverão responder a mesma carta, o jogador que responder primeiro corretamente guarda a carta e recebe um ponto. Caso um jogador responda errado, o jogador adversário recebe o ponto. O jogador que ganhar dois pontos nesta fase é declaro como o vencedor.

***OBS:** O baralho de JavaScript está sendo testado sem a mecânica de dificuldade em cada carta, portanto considere usar o baralho inteiro como um só.

APÊNDICE C – Questionário de avaliação

Dimensão/Subdimensão		Descrição
Usabilidade	Estética	O design do jogo é atraente.
		Os textos, cores e fontes combinam e são consistentes.
	Aprendizibilidade	Eu precisei aprender poucas coisas para poder começar a jogar o jogo.
		Aprender a jogar este jogo foi fácil para mim.
		Eu acho que a maioria das pessoas aprenderiam a jogar este jogo rapidamente.
	Operabilidade	Eu considero que o jogo é fácil de jogar.
		As regras do jogo são claras e compreensíveis.
	Acessibilidade	As fontes (tamanho e estilo) utilizadas no jogo são legíveis.
		As cores utilizadas no jogo são compreensíveis.
Confiança		Quando olhei pela primeira vez o jogo, eu tive a impressão de que seria fácil para mim.
		A organização do conteúdo me ajudou a estar confiante de que eu iria aprender com este jogo.
Desafio		Este jogo é adequadamente desafiador para mim.
		O jogo oferece novos desafios (oferece novos obstáculos, situações ou variações) com um ritmo adequado.
		O jogo não se torna monótono nas suas tarefas (repetitivo ou com tarefas chatas).
Satisfação		Completar as tarefas do jogo me deu um sentimento de realização.
		É devido ao meu esforço pessoal que eu consigo avançar no jogo.
		Me sinto satisfeito com as coisas que aprendi no jogo.
		Eu recomendaria este jogo para meus colegas.
Interação Social		Eu pude interagir com outras pessoas durante o jogo
		O jogo promove momentos de cooperação e/ou competição entre os jogadores.
		Eu me senti bem interagindo com outras pessoas durante o jogo.
Diversão		Eu me diverti com o jogo.
		Aconteceu alguma situação durante o jogo (elementos do jogo, competição, etc.) que me fez sorrir.
Atenção focada		Houve algo interessante no início do jogo que capturou minha atenção.
		Eu estava tão envolvido no jogo que eu perdi a noção do tempo.
		Eu esqueci sobre o ambiente ao meu redor enquanto jogava este jogo.
Relevância		O conteúdo do jogo é relevante para os meus interesses.
		É claro para mim como o conteúdo do jogo está relacionado com a disciplina.
		O jogo é um método de ensino adequado para esta disciplina.
		Eu prefiro aprender com este jogo do que de outra forma (outro método de ensino).
Aprendizagem percebida		O jogo contribuiu para a minha aprendizagem na disciplina.
		O jogo foi eficiente para minha aprendizagem, em comparação com outras atividades da disciplina.
		O jogo contribuiu para meu conhecimento sobre tipos de variáveis.
		O jogo contribuiu para meu conhecimento de operadores lógicos, aritméticos e relacionais.
		O jogo contribuiu para meu conhecimento sobre comandos condicionais (if) e laços de repetição (for/while/do).
		O jogo contribuiu para meu conhecimento sobre vetores e matrizes.
		O jogo contribuiu para meu conhecimento sobre a sintaxe da linguagem.
		Descobri novos comandos e características da linguagem através do jogo.
		O jogo me trouxe vontade de aprender sobre a linguagem.
Comentários adicionais		Este espaço é para você escrever críticas, sugestões ou elogios sobre qualquer aspecto do jogo. Sua sinceridade é essencial e contribui muito para este projeto, obrigado pela participação!