UNIVERSIDADE FEDERAL DO AMAPÁ

DEPARTAMENTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

COORDENAÇÃO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

Lorena Roberta Nunes Guimarães

REGEXMON: Um jogo de apoio ao ensino-aprendizagem de Expressões Regulares

Trabalho de Conclusão de Curso

Macapá

2026

Lorena Roberta Nunes Guimarães

REGEXMON: Um jogo de apoio ao ensino-aprendizagem de Expressões Regulares

Trabalho de Conclusão de Curso submetido à Banca Examinadora do Curso de Ciência da Computação da UNIFAP para a obtenção do Grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Julio Cezar Costa Furtado

Macapá
2026

Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca Central/UNIFAP-Macapá-AP
Elaborado por Cristina Fernandes – CRB-2 / 1569

Guimarães, Lorena Roberta Nunes.
G963r Regexmon: um jogo de apoio ao ensino-aprendizagem de expressões regulares / Lorena Roberta Nunes Guimarães. - Macapá, 2026.
1 recurso eletrônico.
62 f.

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Amapá, Coordenação do Curso de Ciência da Computação, Macapá, 2026.
Orientador: Julio Cezar Costa Furtado.
Coorientador: .

Modo de acesso: World Wide Web.
Formato de arquivo: Portable Document Format (PDF).

1. Jogos educacionais. 2. Expressões regulares. 3. RPG. I. Furtado, Julio Cezar Costa, orientador. II. Universidade Federal do Amapá. III. Título.

CDD 23. ed. – 793.93

GUIMARÃES, Lorena Roberta Nunes. **Regexmon**: um jogo de apoio ao ensino-aprendizagem de expressões regulares. Orientador: Julio Cezar Costa Furtado. 2026. 62 f. Trabalho de Conclusão de Curso (Graduação) - Ciência da Computação. Universidade Federal do Amapá, Macapá, 2026.

UNIVERSIDADE FEDERAL DO AMAPÁ
DEPARTAMENTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
COORDENAÇÃO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

ATA DE DEFESA DE TCC

Realizou-se, no dia 29 de maio de 2026, às 20h, via Google Meet, a defesa do TCC intitulado: Regexmon, Um Jogo de Apoio ao Ensino de Expressões Regulares, do(a) discente Lorena Roberta Nunes Guimarães (matrícula 2019003501). A Banca Examinadora foi composta pelo Prof. Dr. Julio Cezar Costa Furtado, presidente da banca e orientador; Prof. Dr. Rafael Pontes Lima e Prof. Me. Marco Antônio Leal da Silva, examinadores. Concluída a defesa, foram realizadas as arguições e comentários. Em seguida, procedeu-se o julgamento pelos membros da Banca Examinadora, tendo o trabalho sido APROVADO com nota 10.

E, para constar, eu, Prof. Dr. Julio Cezar Costa Furtado, orientador e presidente da Banca Examinadora, lavrei a presente ata que, após lida e achada conforme, foi assinada por mim e demais membros da Banca Examinadora.

Macapá-Ap., 29 de maio de 2026.

gov.br Documento assinado digitalmente
JULIO CEZAR COSTA FURTADO
Data: 12/06/2026 18:40:36-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Julio Cezar Costa Furtado
Orientador do TCC

gov.br Documento assinado digitalmente
RAFAEL PONTES LIMA
Data: 12/06/2026 10:32:59-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Rafael Pontes Lima
Examinador (UNIFAP)

gov.br Documento assinado digitalmente
MARCO ANTONIO LEAL DA SILVA
Data: 12/06/2026 18:35:40-0300
Verifique em https://validar.iti.gov.br

Prof. Me. Marco Antonio Leal da Silva
Examinador (UNIFAP)

AGRADECIMENTOS

Agradeço primeiro àquele que deu tudo por mim e quem mais acompanhou de perto essa jornada, ao meu Senhor e criador Jesus Cristo.

Agradeço de todo meu coração aos meus pais, que apesar do susto quando comentei a escolha do curso, sempre me apoiaram e sei o quanto lutaram para me dar o melhor, essa conquista é de vocês. Agradeço a minha irmã, que ouviu todas as minhas histórias do curso, riu, me aconselhou e apoiou. Agradeço por todas as caronas para a universidade!

Agradecimento especial ao meu noivo, que me ajudou a entender melhor o mundo dos jogos, mas também me auxiliou nos momentos de surtos na escrita do TCC e em muitos momentos da vida.

Agradeço aos meus amigos, vocês tornaram a caminhada mais leve, mais divertida e com memórias inesquecíveis, seja em conversas aleatórias sobre a vida ou na escuta em momentos difíceis.

Gostaria de agradecer também aos meus professores do curso, obrigada pela dedicação de vocês para ensinar mesmo em tempos difíceis que tivemos. Agradecimento especial também ao meu orientador Júlio Cezar, que acreditou no meu potencial desde o primeiro dia e aguentou todas as mensagens e dúvidas que tirei durante todo o processo do TCC. Agradeço por toda a paciência e parceria!

E por fim e não menos importante, aqueles que são homenageados neste TCC, meu avô Francisco e minha avó Rita, agradeço pela coragem que tiveram para sair da zona de conforto em busca de algo melhor para a família. Agradeço pelo legado que deixaram, ele continua perpetuando em cada um. A coragem de vocês reflete em mim e onde cheguei, agradeço de coração.

RESUMO

As expressões regulares são um conteúdo transversal nos cursos de Ciência da Computação, presente em disciplinas de Linguagens Formais, Engenharia de Software, compiladores e processamento de texto. Apesar de sua relevância prática, são consistentemente identificadas como um dos tópicos mais difíceis de ensinar, devido à notação compacta e sensível ao contexto, à ausência de mensagens de erro informativas na maioria dos interpretadores e à escassez de recursos educacionais dedicados. Este trabalho apresenta o RegexMon, um jogo RPG de mundo aberto desenvolvido na engine Godot 4, ambientado no campus universitário da UNIFAP, com o objetivo de apoiar o ensino de expressões regulares em cursos de graduação em Computação. O jogo apresenta uma narrativa original em que a jogadora, Mini me, assume o legado de seus avós, mestres de regex, para proteger o campus de criaturas chamadas Regexmons que fazem uso indevido dos padrões. O mecanismo central é um sistema de batalha em dois turnos: no turno de ataque, a jogadora deve construir uma string que corresponda ao padrão exibido pelo inimigo; no turno de defesa, deve sintetizar uma expressão regular que aceite um conjunto de strings válidas e rejeite outro. Essa dualidade operacionaliza as duas competências fundamentais do ensino de expressões regulares, reconhecimento e síntese de padrões, em uma única mecânica de jogo. A progressão pedagógica é estruturada em três ginásios correspondentes aos grupos de metacaracteres, representantes, quantificadores e âncoras, além de encontros aleatórios na grama e uma batalha final integradora. O jogo foi desenvolvido em um processo de pesquisa baseada em design por alunos de graduação sob supervisão docente, e encontra-se funcional com o mundo aberto completo, o sistema de batalha implementado e todos os desafios dos ginásios definidos. A avaliação empírica com estudantes está planejada como trabalho futuro.

Palavras-chave: jogos educacionais; expressões regulares; linguagens formais; RPG; aprendizagem baseada em jogos; ensino de computação.

ABSTRACT

Regular expressions are a cross-cutting topic in Computing curricula, present in formal languages, software engineering, compilers, and text processing courses. Despite their practical relevance, they are consistently identified as one of the most difficult topics to teach, due to compact and context-sensitive notation, the absence of informative error messages in most interpreters, and a scarcity of dedicated educational resources. This work presents RegexMon, an open-world RPG developed in the Godot 4 engine, set on the UNIFAP university campus, designed to support the teaching of regular expressions in undergraduate Computing courses. The game features an original narrative in which the player, Mini me, takes on the legacy of her grandparents, regex masters, to protect the campus from creatures called Regexmons that misuse patterns. The central mechanic is a two-turn battle system: in the attack turn, the player must construct a string that matches the pattern displayed by the opponent; in the defense turn, the player must synthesize a regular expression that accepts a set of valid strings and rejects another. This duality operationalizes the two fundamental competencies of regular expression instruction, pattern recognition and pattern synthesis, within a single game mechanic. Pedagogical progression is structured around three gymnasiums corresponding to metacharacter groups, representatives, quantifiers, and anchors, along with random grass encounters and a final integrative battle. The game was developed through a design-based research process by undergraduate students under faculty supervision, and is functional with a complete open world, an implemented battle system, and all gymnasium challenges defined. Empirical evaluation with students is planned as future work.

Keywords: educational games; regular expressions; formal languages; RPG; game-based learning; computing education.

LISTA DE FIGURAS

Figura 1. Exemplo de uso do metacaractere ponto. 20
Figura 2. Exemplo de uso do metacaractere lista. 20
Figura 3. Exemplo de uso do metacaractere lista negada. 21
Figura 4. Exemplo de uso do metacaractere opcional. 22
Figura 5. Exemplo de uso do metacaractere asterisco. 22
Figura 6. Exemplo de uso da expressão regular “coringa”. 23
Figura 7. Exemplo de uso do metacaractere mais. 23
Figura 8. Exemplo de uso do metacaractere ‘chaves’. 23
Figura 9. Exemplo de uso do metacaractere circunflexo. 24
Figura 10. Exemplo de uso do metacaractere cifrão. 25
Figura 11. Exemplo de uso do metacaractere borda. 25
Figura 12. Exemplo de uso do metacaractere escape. 26
Figura 13. Exemplo de uso do metacaractere ‘ou’. 27
Figura 14. Exemplo de uso do metacaractere grupo. 27
Figura 15. Exemplo de uso do metacaractere grupo com subgrupo. 27
Figura 16. Exemplo de uso do metacaractere retrovisor. 28
Figura 17. Overworld do RegexMon: campus universitário com zonas de grama, edificações e personagem principal. 37
Figura 18. Turno de ataque: o inimigo exibe um padrão regex e a jogadora deve digitar uma string correspondente. 38
Figura 19. Turno de defesa: a jogadora deve sintetizar uma regex que aceite as strings listadas em "Aceita" e rejeite as listadas em "Rejeita". 39

LISTA DE QUADROS

Quadro 1. Lista de Metacaracteres. 18
Quadro 2. Símbolos do grupo Representantes. 19
Quadro 3. Símbolos do grupo Quantificadores. 21
Quadro 4. Símbolos do grupo Âncoras. 24
Quadro 5. Símbolos do grupo Outros. 26
Quadro 6. Comparação entre os trabalhos relacionados e o RegexMon. 33

LISTA DE ABREVIATURAS E SIGLAS

RPG	*Role Playing Game*
GDD	Documento de Design do Jogo
DBR	Design-based research
UNIFAP	Universidade Federal do Amapá
IAQJEd	Instrumento de Avaliação da Qualidade de Jogos Digitais com Finalidade Educativa
ER	Expressões Regulares
LDB	Lei de Diretrizes e Bases da Educação Nacional
DCN	Diretrizes Curriculares Nacionais
RU	Restaurante Universitário
PCRE2	Perl Compatible Regular Expressions

SUMÁRIO

1 INTRODUÇÃO 11
1.1. Motivação, Justificativa e Contribuição à área 12
1.2. Objetivos 14
1.2.1. Objetivo Geral 14
1.2.2. Objetivos Específicos 14
1.3. Metodologia de Pesquisa 14
1.3.1. Tipo de Pesquisa 14
1.3.2. Pesquisa Baseada em Design 14
1.3.3. Fases do Desenvolvimento 15
1.4. Organização do Trabalho 16
2 CONTEXTUALIZAÇÃO E TERMINOLOGIAS DO TRABALHO 17
2.1. Expressões Regulares 17
2.2. Terminologia 18
2.2.1. Os metacaracteres 18
2.2.1.1. Representantes 19
2.2.1.1.1. O ponto (.) 19
2.2.1.1.2. Lista [] 20
2.2.1.1.3. Lista Negada [^] 20
2.2.1.2. Quantificadores 21
2.2.1.2.1. Opcional (?) 21
2.2.1.2.2. Asterisco (*) 22
2.2.1.2.3. Mais (+) 23
2.2.1.2.4. Chaves ({n,m}) 23
2.2.1.3. Âncoras 24
2.2.1.3.1. Circunflexo (^) 24
2.2.1.3.2. Cifrão ($) 24
2.2.1.3.3. Borda (\b) 25
2.2.1.4. Outros 25
2.2.1.4.1. Escape (\) 26
2.2.1.4.2. Ou (|) 26
2.2.1.4.3. Grupo (...) 27
2.2.1.4.4. Retrovisor (\1 … \9) 28
2.3. O uso de jogos no ensino 28
2.3.1. Os tipos de jogos 29
2.3.2. O uso de jogos na educação 30
2.3.3. O uso de jogos para ensino de programação 31
2.4. Trabalhos Relacionados 32
2.4.1. Labirinto Gramágico 32
2.4.2. Ferramentas web de prática de regex 33
2.4.3. Posicionamento do RegexMon 33
3 REGEXMON: O JOGO 35
3.1. Personagens 35
3.2. Mundo do Jogo 36
3.3. Mecânica de Batalha 37
3.4. Progressão Pedagógica 39
3.4.1. Encontros na Grama 39

3.4.2. Ginásio 1: Representantes (Quadra) 40
3.4.3. Ginásio 2: Quantificadores (RU) 40
3.4.3. Ginásio 3: Âncoras (Prédio) 40
3.4.5. Batalha Final (Bloco de Medicina) 41
3.5. Sistema de Feedback 41
3.6. Implementação Técnica 42
4 POTENCIAL EDUCACIONAL 44
4.1. Alinhamento Curricular 44
4.2. Princípios Pedagógicos 45
4.3. Validação Interna 46
5 CONSIDERAÇÕES FINAIS 48
REFERÊNCIAS BIBLIOGRÁFICAS 50
APÊNDICE A - DOCUMENTO DE DESIGN DO JOGO (GDD) 54
APÊNDICE B - ROTEIRO DO JOGO 66

1 INTRODUÇÃO

Com a influência da tecnologia em diversos âmbitos da vida humana, a educação também obteve novos caminhos de aprendizagem. Um deste novo caminho é o uso de jogos para auxiliar o ensino de matérias em instituições. No ambiente acadêmico, a presença dessa nova forma de aprendizagem ainda é pouco utilizada e dentro da área de Ciência da Computação sua presença se torna ainda mais ausente. Garozi et al. (2021), revela que os jogos para o ensino dentro de Ciência da Computação são voltados em sua maioria, para Engenharia de Software e Redes, enquanto áreas também importantes como Linguagem Formal possuem um ou nenhum jogo disponível.

A Linguagem Formal é a definição de uma linguagem com sintaxe bem definida e semântica precisa (Vieira, 2006). Estas linguagens são relevantes dentro da Ciência da Computação para a construção de compiladores para linguagens de programação. Toda linguagem formal possui um alfabeto associado e este é um conjunto finito não vazio de elementos que serão definidos através dos símbolos das Expressões Regulares (Vieira, 2006). As Expressões Regulares são difíceis de compreender devido a sua sintaxe diversificada e ao mesmo tempo compacta. Os significados específicos dos símbolos tornam sua combinação correta algo que envolve prática e atenção minuciosa, conforme a complexidade dos padrões aumenta, a dificuldade cresce na mesma proporção (Jargas, 2012). Os erros em uma expressão regular podem gerar resultados inesperados e em muitos casos, os interpretadores de regex não possuem mensagens de erros claros sobre a sua causa (Jargas, 2012). Além disso, o comportamento de uma expressão regular pode variar de acordo com a ferramenta ou linguagem de programação que está sendo usada, estas diferenças podem ser um obstáculo no aprendizado dos iniciantes no assunto e dificultar a transferência de conhecimentos entre contextos (GAROZI, 2021).

Outro fator que torna as Expressões Regulares um assunto complexo é a superficialidade na qual é explorado, apesar de haver inúmeras documentações disponíveis, o nível de conhecimento dentro deles é prévio e os exemplos utilizados abordam casos simples, levando um iniciante no assunto a enfrentar problemas ao se deparar com expressões mais complexas (Jargas, 2012). Um exemplo disto, é a dificuldade em aplicar expressões regulares aos problemas do mundo real, por conta dos dados dificilmente serem previsíveis e didáticos, é comum exigir expressões regulares complexas e com alto grau de flexibilidade (GAROZI, 2021).

Segundo Carneiro et al. (2015), os jogos como instrumento pedagógico estão presentes desde a Antiguidade e a sua função de ensino vai além do entretenimento. Kessler et al. (2010) afirma que os jogos educativos proporcionam aprendizagem aliadas ao desenvolvimento de competências como raciocínio, estratégia e reflexão de forma lúdica e divertida. Nesse contexto, uma abordagem amplamente utilizada é a dos jogos sérios, que utilizam o entretenimento para alcançar objetivos educacionais e de desenvolvimento de habilidades. Um exemplo é o jogo Labirinto Gramágico, criado por Henrique Garozi em seu trabalho "Labirinto Gramágico: Um jogo Educativo para o Ensino de Gramáticas Regulares". Nele, é abordado a relevância da Linguagem Formal e o registro do desenvolvimento de um jogo com tema no qual o jogador para sair do labirinto, deve abrir as portas através de feitiços que são nada mais que Expressões Regulares.

Pensando nisso, este trabalho desenvolveu um jogo sério ou "serious game" com objetivo de auxiliar o ensino de Expressões Regulares dentro das salas de aula, tornando o assunto mais didático e interessante aos discentes, além de combinar o elemento de entretenimento dos jogos com a habilidade de ensinar. O "Regexmon" consiste em um jogo do modelo RPG inspirado no jogo conhecido como "Pokémon". Ele usa de elementos como combates no campo e lutas em estádios para avaliar a evolução do jogador. O ensino de Expressões Regulares consiste nas batalhas em turnos presentes no jogo, onde o usuário para atacar deve construir strings que são aceitas pela expressão exposta pelo inimigo e para se defender, deve criar um padrão que aceite um conjunto de strings e rejeite outros. Além de descobrir novos símbolos no decorrer do seu progresso.

Este trabalho consiste neste capítulo introdutório, com resumo breve do conteúdo geral abordado, a motivação, os objetivos e a metodologia. No capítulo 2 é tratado sobre o referencial teórico, explicando a variação de símbolos das expressões regulares e os grupos que eles representam dentro do tema. O Capítulo 3 aborda o jogo, descrevendo sua narrativa, personagens, mecânica de batalha e progressão pedagógica. No Capítulo 4 o potencial pedagógico do jogo é discutido. O Capítulo 5 apresenta as considerações finais sobre o trabalho e futuros caminhos. E por fim as referências bibliográficas utilizadas ao longo do trabalho e o Apêndice A no qual contém o Documento de Design do Jogo (GDD).

1.1. Motivação, Justificativa e Contribuição à área

A educação é uma das áreas mais produtivas quanto à disseminação da tecnologia (Sommariva, 2012), entretanto ainda utiliza pouco os recursos disponíveis que ela proporciona para ensino. Devido a isto, o ensino, principalmente o ensino superior se torna

algo cansativo e desinteressante aos discentes. Isto é o que afirma Melo em seu estudo (Melo, 2017), ao buscar os diversos fatores de evasão de estudantes, as aulas entediantes e o ensino robotizado foram um dos fatores apontados. Tratando especificamente dos discentes do curso de Ciência da Computação, onde o uso da tecnologia é diretamente relacionado ao ensino, alguns recursos didáticos para suporte são pouco utilizados, como os jogos digitais. De acordo com Garozi et al. (2021), os poucos jogos educativos disponíveis são focados em Engenharia de Software e Redes de Computadores.

Apesar de jogos ainda serem vistos com o alvo de entretenimento e lazer (Sommariva, 2012), diversos projetos e estudos do uso de jogos eletrônicos na educação como meio de aprendizagem estão sendo conduzidos por instituições internacionais renomadas, dando assim credibilidade e legitimidade ao assunto (Lozza, 2017). A relevância do uso de jogos no ensino é o fato dele alcançar todos os gêneros e faixas etárias, contribuindo para o desenvolvimento de habilidades positivas, como atenção, criatividade e desempenho em sala de aula, além de promover a motivação do discente para o sucesso e engajamento (Monclar, 2018). De acordo com Battistella, "jogos são considerados uma ferramenta poderosa para ensino com um objetivo, eles promovem o 'aprendizado ativo' para alcançar um aprendizado profundo e eficaz dentro de um tempo de aula e instrução aceitável, além de um meio interessante de treino e prática. E por conta das suas características, como a competição, desafios e interações, os jogos conseguem tornar o aprendizado em uma experiência engajante e lúdica" (BATTISTELLA, 2016).

Em seu estudo sobre o uso de jogos para o ensino de computação em Educação Superior, Battistella et al. (2016) concluiu que existe uma certa quantidade de jogos digitais com esse objetivo, entretanto, sua grande maioria está focada no ensino de engenharia de software, como Garozi et al. (2021) afirma, ou no ensino dos fundamentos de programação. As outras áreas da grade curricular do curso, como Autômatos, Linguagens Formais e Expressões Regulares não apresentam um jogo ensinando os fundamentos da matéria e por sua vez, não oferecem um interesse aos alunos para estas temáticas.

Com estes fatores, o intuito deste trabalho é auxiliar a existência de conteúdo para o ensino de Expressões Regulares para os discentes do curso de Ciência da Computação. Além de contribuir para os estudos já existentes sobre a eficácia e relevância do uso de jogos na educação, principalmente no Ensino Superior, para tornar o ensino cada vez mais prazeroso e didático dentro das instituições.

1.2. Objetivos

1.2.1. Objetivo Geral

Desenvolver um jogo sério ou "serious game" no estilo RPG para apoiar o ensino-aprendizagem de expressões regulares em cursos de graduação em Ciência da Computação.

1.2.2. Objetivos Específicos

Para atingir o objetivo geral, os seguintes objetivos específicos foram definidos:

1. Identificar os grupos de metacaracteres de expressões regulares e mapear sua progressão pedagógica típica em cursos de graduação em Computação;
2. Projetar uma mecânica de batalha que operacionalize simultaneamente as competências de reconhecimento e síntese de padrões regex;
3. Desenvolver uma narrativa original que contextualiza o conteúdo de expressões regulares em um ambiente familiar ao público-alvo;
4. Implementar o jogo em Godot 4 com mundo aberto funcional, sistema de batalha e progressão pedagógica por ginásios;
5. Realizar validação interna das mecânicas e desafios implementados.

1.3. Metodologia de Pesquisa

1.3.1 Tipo de Pesquisa

Este trabalho caracteriza-se como uma pesquisa aplicada, pois tem como objetivo a produção de um artefato, o jogo RegexMon, voltado para a solução de um problema prático identificado na literatura e na prática docente: a escassez de recursos lúdicos para o ensino de expressões regulares em cursos de graduação em Computação. Quanto à abordagem, é qualitativa, uma vez que o desenvolvimento e a validação interna do jogo são orientados por critérios pedagógicos e decisões de design avaliadas por raciocínio analítico, e não por métricas quantitativas de desempenho acadêmico, que serão objeto de pesquisa futura. Quanto ao objetivo, é descritiva e exploratória: descritiva por documentar o processo de desenvolvimento e as características do jogo produzido; exploratória por investigar uma área com poucos trabalhos precedentes.

1.3.2 Pesquisa Baseada em Design

O procedimento metodológico adotado é a pesquisa baseada em design (design-based research, DBR). Nessa abordagem, a solução, neste caso o jogo, é desenvolvida e refinada iterativamente com base em critérios derivados da teoria pedagógica e em feedback obtido ao longo do processo de implementação. A DBR é particularmente adequada para o desenvolvimento de recursos educacionais, pois integra a produção do artefato com a investigação de princípios de design que podem ser generalizados para contextos similares.

A escolha da DBR fundamenta-se em três características do presente trabalho: a inexistência de um modelo consolidado de jogo para o ensino de expressões regulares que pudesse ser replicado; a necessidade de alinhar continuamente as decisões de design de jogo com os objetivos curriculares do conteúdo; e a natureza iterativa do desenvolvimento em Godot 4, que permitiu ciclos rápidos de implementação, teste e ajuste.

1.3.3 Fases do Desenvolvimento

O desenvolvimento do RegexMon foi organizado em três fases sequenciais: design pedagógico, implementação iterativa e validação interna.

Fase 1, Design pedagógico: nesta fase foram definidos os objetivos de aprendizagem a partir do currículo típico de expressões regulares em cursos de Ciência da Computação. Os quatro grupos de metacaracteres, representantes, quantificadores, âncoras e outros, foram mapeados em uma progressão pedagógica de três ginásios, cada um dedicado a um grupo de construtos. Para cada ginásio foram projetados os desafios de batalha, tanto de ataque quanto de defesa, garantindo que os padrões exigidos em cada desafio fossem os construtos do respectivo ginásio. A narrativa foi desenvolvida em paralelo, com o objetivo de contextualizar o conteúdo em uma história que ressoasse com o cotidiano universitário do público-alvo. O enredo, as regras, a dinâmica de combate e o design gráfico foram documentados em um Documento de Design do Jogo (GDD), disponível no Apêndice A.

Fase 2, Implementação iterativa: a implementação foi realizada em Godot 4, com a equipe de desenvolvimento composta por alunos de graduação sob supervisão docente. O mundo aberto foi construído com um sistema de tiles de 16x16 pixels, com múltiplas zonas navegáveis representando o campus da UNIFAP. O sistema de batalha foi implementado com dois modos de entrada: um para a fase de ataque, onde a jogadora digita uma string, e outro para a fase de defesa, onde ela digita uma expressão regular. A validação das entradas é realizada pelo motor de regex nativo do Godot 4. Os diálogos narrativos foram integrados às transições de cena e às batalhas dos ginásios. A progressão entre ginásios foi implementada

como uma sequência linear, com encontros aleatórios na grama disponíveis em toda a extensão do mundo aberto.

Fase 3, Validação interna: a validação interna consistiu em verificar que cada desafio de batalha era solucionável com os construtos definidos para o respectivo ginásio, que os padrões apresentados nas fases de ataque correspondiam às regras da liga ou ginásio, e que as strings aceitas e rejeitadas nas fases de defesa eram consistentes com a solução esperada. Nenhum estudo de usuário externo foi conduzido nesta fase; a avaliação empírica com estudantes de Ciência da Computação da UNIFAP, utilizando o instrumento IAQJEd, está planejada como trabalho futuro.

1.4 Organização do Trabalho

Este trabalho está organizado em cinco capítulos. O presente capítulo introdutório apresenta a motivação, a justificativa, os objetivos e a metodologia de pesquisa adotada. O Capítulo 2 aborda o referencial teórico, tratando das expressões regulares, seu histórico, a terminologia dos metacaracteres e seus quatro grupos funcionais, as dificuldades estruturais de seu ensino, os fundamentos da aprendizagem baseada em jogos e o uso de jogos sérios no ensino superior de Computação, além dos trabalhos relacionados. O Capítulo 3 apresenta o RegexMon em sua totalidade, cobrindo o conceito e a narrativa, os personagens, o mundo do jogo, a mecânica de batalha, a progressão pedagógica por ginásios e a implementação técnica. O Capítulo 4 analisa o potencial educacional do jogo, discutindo seu alinhamento curricular e os princípios pedagógicos que fundamentam seu design. O Capítulo 5 apresenta as considerações finais e as direções de trabalho futuras. Por fim, são listadas as referências bibliográficas utilizadas ao longo do trabalho, e o Apêndice A contém o Documento de Design do Jogo (GDD).

2 CONTEXTUALIZAÇÃO E TERMINOLOGIAS DO TRABALHO

Esta seção aborda o tópico base do trabalho, as expressões regulares e a sua terminologia, seus conceitos, grupos de metacaracteres e função de cada um. Além de apresentar a situação atual do uso de jogos na educação e os tipos de jogos utilizados para a eficácia do ensino. Por fim, também relata o uso de jogos para ensino de programação no nível superior, as dificuldades e benefícios da sua aplicação.

2.1. Expressões Regulares

O surgimento da ideia sobre Expressões Regulares aconteceu através da publicação do artigo “A logical calculus of the ideas immanent in nervous activity” por Warren McCulloch e Walter Pitts (1943). Uma das principais informações contidas no artigo foi a comparação de neurônios com um limite binário para a lógica booleana. No trabalho, os neurologistas associaram a atividade neuronal com a lógica proposicional, ou seja, modelaram a forma que os neurônios interagiam entre si através da lógica (MCCULLOCH, 1943).

Stephen Kleene (1951) padronizou algebricamente os modelos neurológicos descritos por McCulloch & Pitts no seu artigo “Representation of Events in Nerve Nets and Finite Automata”. A representação destes modelos foi realizada usando a notação matemática de Kleene chamada de “conjuntos regulares”, formando assim a álgebra de Kleene (KLEENE, 1951).

O cientista da computação Ken Thompson descreveu na publicação “Programming techniques: Regular expression search algorithm” (Thompson, 1968), um método de busca em texto que recebe como entrada uma determinada expressão regular. A construção desse mecanismo no editor de texto QED foi utilizando a álgebra de Kleene (Kleene, 1951). Dessa maneira, ele conseguiu demonstrar a implementação de um compilador para transformar uma expressão regular em um código compilado. O programador Henry Spencer (1986), lançou em um grupo de discussão a primeira biblioteca de expressões regulares para linguagem C, chamada regex (JARGAS, 2012).

A Expressão Regular (ER) é um método formal de especificar um padrão de texto (Jargas, 2012). Esse padrão é uma composição de símbolos e caracteres com funções que ao serem agrupados entre si e com outros caracteres literais, geram uma sequência que é uma expressão (Jargas, 2012). A expressão criada, através desse conjunto de caracteres, é considerada uma regra, que avalia se uma entrada de dados qualquer irá casar com ela e obedecer às suas condições.

Em outras palavras, as expressões regulares são padrões de texto compostos por símbolos e caracteres que, quando combinados, definem regras capazes de validar ou identificar sequências de dados. A flexibilidade proporcionada pela combinação desses elementos permite a construção de padrões com diferentes níveis de complexidade, adequados às mais diversas necessidades de processamento e validação de informações.

2.2. Terminologia

As Expressões Regulares (ER) são metacaracteres que casam um padrão. Segundo Jargas (2012), elas são formadas por símbolos e caracteres literais, os quais são chamados de metacaracteres. O termo casar ou 'match' relacionado a ER é o encontro ou combinação com um determinado padrão. Este padrão pode ser uma única palavra ou várias, uma linha vazia, um só número ou uma sequência. Ele pode ser o que quer que precise ser encontrado pela expressão regular (JARGAS, 2012).

Os metacaracteres possuem o objetivo de padronizar de maneira específica algo que é abrangente. De acordo com a definição de busca dada para a ER pode surgir uma lista infinita de casamento. Isto se dá pela utilidade das expressões regulares em buscar ou validar um padrão de texto que pode ser variável, como datas, nomes de pessoas, senhas, entre outros padrões.

2.2.1 Os metacaracteres

Na Expressão Regular, cada símbolo utilizado possui uma função específica que pode mudar dependendo do contexto no qual é inserido (Jargas, 2012). Além disso, ao ser agregado com outros símbolos e combinar suas funções, geram ERs mais complexas.

Ao todo são 14 metacaracteres padrão representados pelos símbolos no Quadro 1.

Metacaractere	Nome	Metacaractere	Nome
.	Ponto	^	Circunflexo
[]	Lista	$	Cifrão
[^]	Lista Negada	\b	Borda
?	Opcional	\	Escape
*	Asterisco	|	Ou
+	Mais	()	Grupo
{}	Chaves	\1	Retrovisor

Quadro 1: Lista de Metacaracteres.

Estes metacaracteres são divididos em quatro grupos distintos, separados de acordo com suas características comuns entre si. Os grupos são:

- Representantes: A função deste grupo é representar um ou mais caracteres, além disso eles casam a posição de um único caractere e não mais que um (JARGAS, 2012);
- Quantificadores: estes metacaracteres indicam o número de repetições permitidas para a entidade anterior, seja ela um metacaractere ou caractere (Jargas, 2012). Eles definem a quantidade de vezes que a entidade pode aparecer na ER. Os quantificadores não são quantificáveis, ou seja, utilizar dois deles seguidos em uma ER é um erro;
- Âncoras: estes metacaracteres marcam uma posição específica na linha (Jargas, 2012). Ele não possui função de representante e quantificadores, logo não sofre influência desses metacaracteres dentro de uma ER.
- Outros: os metacaracteres desse grupo possuem funções específicas e não estão relacionados entre si, logo não podem ser agrupados em outra classe (Jargas, 2012). O uso deles junto com os outros grupos torna as Expressões Regulares ainda mais complexas e robustas para a busca de palavras e textos.

2.2.1.1 Representantes

Nesta seção serão abordados acerca dos metacaracteres representantes com suas funções específicas e exemplo de como aplicá-los. O Quadro 2 apresenta os 3 símbolos presentes nesse grupo.

Metacaractere	Nome	Função
.	Ponto	Um caractere qualquer
[]	Lista	Lista de caracteres permitidos
[^]	Lista negada	Lista de caracteres proibidos

Quadro 2: Símbolos do grupo Representantes.

2.2.1.1.1 O ponto (.)

O caractere ponto é considerado o coringa dentre os metacaracteres das ERs. Este apelido é dado, pois o símbolo casa com qualquer outro caractere, seja um espaço em branco, número, letra ou até mesmo outro ponto. Na figura 1 temos o exemplo de funcionamento do metacaractere '.'.

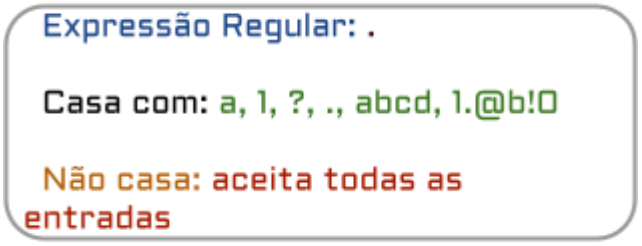

Figura 1: Exemplo de uso do metacaractere ponto.

2.2.1.1.2 Lista []

A lista, diferente do ponto, não é abrangente e não casa com qualquer caractere. Ela delimita quais caracteres aceitam para uma combinação, tornando assim, a ER mais específica, como demonstrado na Figura 2. Um detalhe importante da lista, é que quando um metacaractere se encontra dentro de uma, ele se torna um caractere normal e perde sua função.

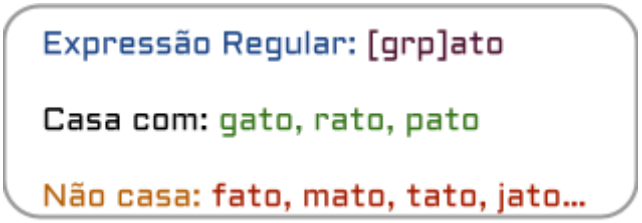

Figura 2: Exemplo de uso do metacaractere lista.

Dentro da lista, é possível inserir intervalos para evitar a listagem extensa de elementos que a compõem. Uma exceção a regra dos caracteres serem normais dentro da lista, é o traço, pois ele dentro de uma lista representa o intervalo entre dois caracteres. Um exemplo é a ER [0-9] que representa a lista [0123456789].

2.2.1.1.3 Lista Negada [^]

A lista negada funciona com as mesmas regras da lista, podendo ter também intervalos e caracteres literais. A diferença entre elas, é que ela funciona com a lógica inversa, ou seja, tudo aquilo que estiver dentro da lista negada é o que não será aceito para casar. O primeiro caractere é o circunflexo, ele é o que indica que uma lista é negada. Na Figura 3 há o exemplo

de uso da lista negada, onde os elementos a, b, c e d não são casados, pois estão dentro da lista negada.

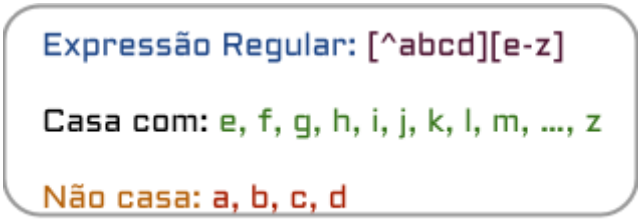

Figura 3: Exemplo de uso do metacaractere lista negada.

Uma característica sobre este metacaractere é que ele não casa o vazio, ou o "nada". Então se em uma ER há a lista negada [^0-9], que indica o casamento de qualquer coisa fora dos números, ela deve estar indicando a opção de casamento possível para a ER, seja letras, símbolos ou espaço em branco.

2.2.1.2 Quantificadores

Nesta seção serão abordados acerca dos quantificadores e suas respectivas funções, além de exemplos de como aplicá-los. O Quadro 3 apresenta os símbolos deste grupo e resumidamente as suas respectivas funções.

Metacaractere	Nome	Função
?	Opcional	Zero ou um
*	Asterisco	Zero, um ou mais
+	Mais	Um ou mais
{n,m}	Chaves	De n até m

Quadro 3: Símbolos do grupo Quantificadores.

2.2.1.2.1 Opcional (?)

É um quantificador que torna o caractere ou metacaractere que o antecede como opcional em uma ER, ou seja, pode não aparecer ou aparecer uma única vez, como no exemplo da Figura 4. Em qualquer um dos casos, ele se casa se o padrão estiver de acordo com o definido. Este quantificador é útil para procurar palavras no singular ou plural em documentos.

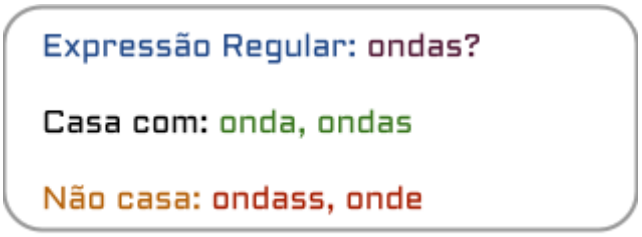

Figura 4: Exemplo de uso do metacaractere opcional.

2.2.1.2.2 Asterisco (*)

Diferente do opcional, onde a entidade pode aparecer uma vez ou não. O asterisco deixa o caractere que o antecede aparecer uma única vez, várias vezes ou infinitamente, ou seja, em qualquer quantidade e até não aparecer. No exemplo da Figura 5 é possível observar os diferentes 'matchs' utilizando o asterisco na lista com a e r.

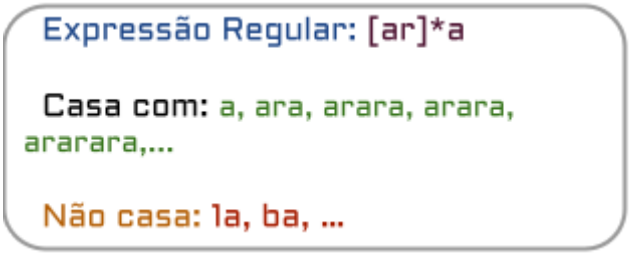

Figura 5: Exemplo de uso do metacaractere asterisco.

O asterisco e o ponto são dois metacaracteres abrangentes, com um aceitando qualquer quantidade e o outro qualquer caractere, respectivamente, como representado na Figura 6. A união desses dois metacaracteres (.*) permite a uma ER o casamento de qualquer caractere em qualquer quantidade, ou seja, aceita nada e tudo ao mesmo tempo (Jargas, 2012). O "nada" é devido ao casamento permitido pelo ponto de qualquer caractere ou nenhum também, dando a possibilidade de casar com uma linha vazia. O "tudo" é por conta de qualquer quantidade oferecida pelo asterisco, ou seja, buscará casar o máximo que conseguir. Por conta dessa versatilidade, a junção desses dois metacaracteres recebeu o nome de "coringa" entre as expressões regulares.

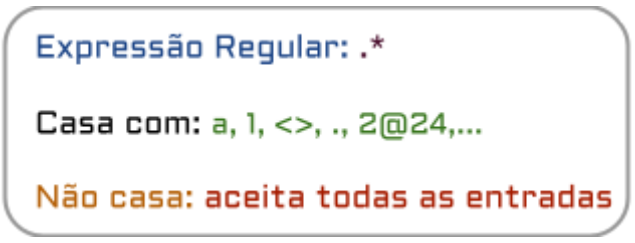

Figura 6: Exemplo de uso da expressão regular “coringa”.

2.2.1.2.3 Mais (+)

O funcionamento do mais (+) é semelhante ao asterisco, as regras aplicadas em um vale ao outro. A única diferença existente é que o mais não é opcional, a entidade anterior deve casar pelo menos uma vez ou várias vezes dentro das entradas para a ER. O exigido para este metacaractere é de no mínimo uma repetição da entidade especificada que o antecede (Jargas, 2012). No exemplo da Figura 7 revela que o número 3 deve aparecer pelo menos uma vez, quando ele não aparece, a string não é aceita.

Figura 7: Exemplo de uso do metacaractere mais.

2.2.1.2.4 Chaves ({n,m})

As chaves são a solução para uma quantificação mais controlada, especificando exatamente a quantidade de repetições da entidade anterior (Jargas, 2012). O seu modelo de representação é {n,m} que significa de n até m vezes. Na Figura 8, o número 9 pode ser casado quando representado de uma até 4 vezes, ao passar do valor delimitado, ele não casa.

Figura 8: Exemplo de uso do metacaractere ‘chaves’.

Dentro da representação das chaves existem outras sintaxes que podem omitir a quantidade final ou especificar exatamente um número desejado, sendo eles os metacaracteres opcional, asterisco e o mais. O uso deles na ER é correto e possui a mesma finalidade. A vantagem de utilizar os metacaracteres é a simplificação da leitura da ER.

2.2.1.3 Âncoras

Nesta seção serão abordados acerca dos metacaracteres das âncoras, suas funções e demonstração de uso dentro de uma ER. O Quadro 4 apresenta uma breve demonstração de cada símbolo e suas funções.

Metacaractere	Nome	Função
^	Circunflexo	Início da linha
$	Cifrão	Fim da linha
\b	Borda	Início ou fim de palavra

Quadro 4: Símbolos do grupo Âncoras.

2.2.1.3.1 Circunflexo (^)

O circunflexo marca o começo de uma linha, ou seja, especifica qual caractere deve estar no início de um padrão para casar com a ER (Jargas, 2012). Apesar de ser utilizado na lista negada [^], fora dela ele é uma âncora e com finalidade de encontrar linhas com um caractere único. No exemplo da Figura 9, a expressão regular terá match com todas as frases que iniciam com a letra ‘a’.

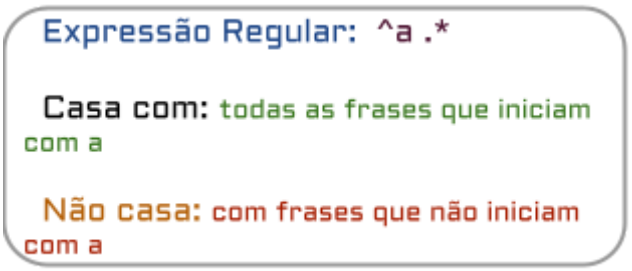

Figura 9: Exemplo de uso do metacaractere circunflexo.

2.2.1.3.2 Cifrão ($)

Este metacaractere é similar e complementar ao circunflexo (Jargas, 2012). A sua funcionalidade é marcar o fim de uma linha e seu uso só é válido no final de uma ER. A junção dele com o circunflexo (^$) significa a busca por uma linha vazia, algo comum para procurar dentro de documentos extensos. O exemplo da Figura 10 representa a busca de ceps que terminam com o número 45, demonstrando a utilidade da limitação dentro de uma regex.

Figura 10: Exemplo de uso do metacaractere cifrão.

2.2.1.3.3 Borda (\b)

O último metacaracteres do grupo das Âncoras é a borda, que marca o limite de uma palavra, ou seja, onde ela começa e termina. Sua representação é marcada pelo uso da contrabarra (\) e a letra b. É útil para casar com palavras exatas e não parte de palavras (Jargas, 2012), sendo o conceito de palavra o uso de letras, números e sublinhado, já os símbolos não são parte de uma palavra. Em alguns casos, pode ocorrer a confusão com a função do circunflexo e cifrão ($), importante ressaltar que um marca o início de uma linha e o outro o final, enquanto a borda marca o início ou fim de uma palavra. A Figura 11 demonstra o uso correto do metacaractere borda e as palavras que seriam casadas com o padrão estabelecido.

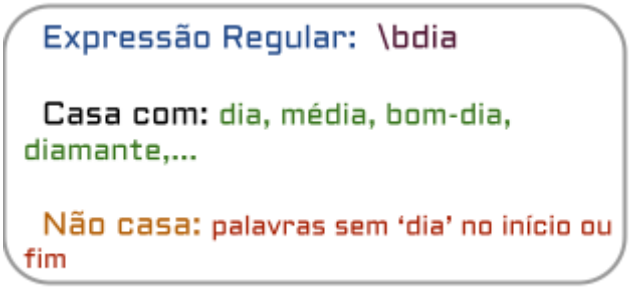

Figura 11: Exemplo de uso do metacaractere borda.

2.2.1.4 Outros

Nesta seção serão abordados sobre os metacaracteres denominados outros com suas funções específicas que não se relacionam entre si, como representado no Quadro 5. Além disso, há exemplos de aplicação dentro de uma expressão regular.

Metacaractere	Nome	Função
\c	Escape	Torna literal o caractere c
|	Ou	Ou um ou outro
(...)	Grupo	Delimita um grupo
\1…\9	Retrovisor	Texto casado nos grupos 1…9

Quadro 5: Símbolos do grupo Outros.

2.2.1.4.1 Escape (\)

Este metacaractere torna um outro metacaractere um caractere normal. Assim como a lista retira a função de uma metacaractere, o escape também se torna um metacaractere literal. Um detalhe acerca desse símbolo é que ele pode escapar a si próprio, ou seja, se tornar um caractere literal. No exemplo da Figura 12 observa-se que o metacaracter asterisco não possui sua função, na verdade, ele está na regex como um símbolo literal.

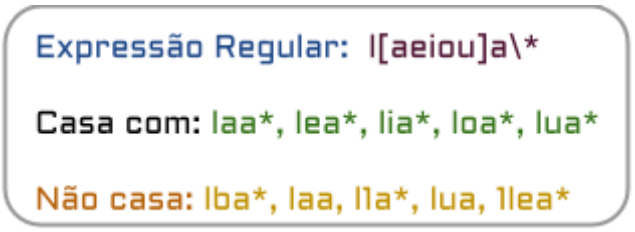

Figura 12: Exemplo de uso do metacaractere escape.

2.2.1.4.2 Ou (|)

O metacaractere ou, representado pela barra vertical (|) serve para casos em ERs onde há mais de uma alternativa possível. A lista, de certa forma, é uma espécie de ou, entretanto é apenas para uma letra, número ou símbolo. No caso do ou, ele assiste as palavras, como no caso de boa-tarde e boa-noite que pode variar uma palavra inteira (Jargas, 2012). O exemplo da Figura 13 revela o uso do ou para palavras.

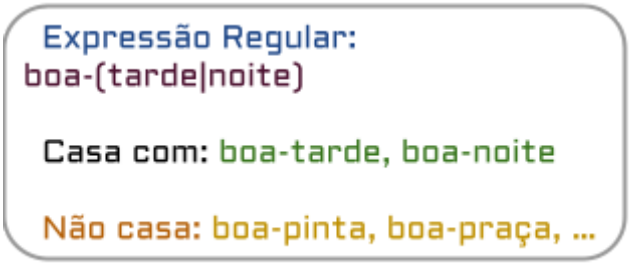

Figura 13: Exemplo de uso do metacaractere ‘ou’.

2.2.1.4.3 Grupo (...)

A função deste metacaractere é juntar vários objetos em um mesmo local. Dentro de um grupo pode ter um ou mais caracteres, metacaracteres ou até outros grupos. A representação dele é dada pelos parênteses “()” e seu conteúdo pode ser visto como um bloco na ER (JARGAS, 2012).

Os metacaracteres quantificadores unidos com os do grupos ‘outros’ ganham abrangência em sua função. Por outro lado, o metacaractere ‘ou’ se torna mais limitado, entretanto, acaba se tornando mais eficaz e específico, ou seja, seu funcionamento melhora. No exemplo da Figura 14, é possível observar o uso do metacaractere grupo e a facilidade da junção dele com outros metacaracteres.

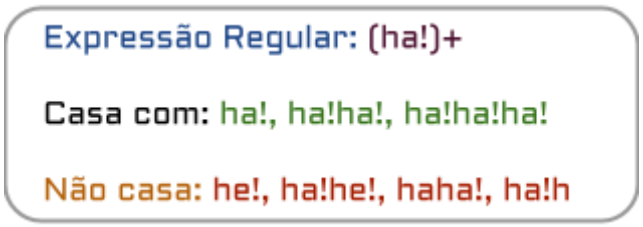

Figura 14: Exemplo de uso do metacaractere grupo.

O grupo não altera o sentido da ER, ele apenas serve como um marcador. Dentro dele também é possível criar subgrupos, como no exemplo da Figura 15, tornando uma expressão regular ainda mais complexa, bem como mais eficiente.

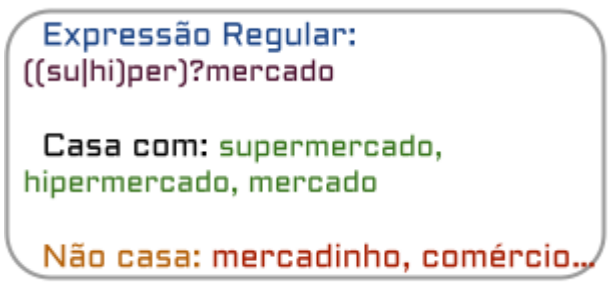

Figura 15: Exemplo de uso do metacaractere grupo com subgrupo.

2.2.1.4.4 Retrovisor (\1 … \9)

É um metacaractere conhecido como retrovisor por apenas englobar o trecho que está atrás. Seu uso é ideal para casar trechos repetidos em uma linha. O escape é utilizado para tornar uma caractere literal, entretanto, uma exceção a esta regra é que se depois do escape houver um número de 1 a 9, não se trata de um escape e sim um retrovisor (Jargas, 2012). Dessa forma, o máximo de retrovisores possíveis em uma ER são nove.

Sua funcionalidade é primordial para quando não se sabe exatamente qual texto o grupo irá casar, ou seja, pode ser qualquer palavra. Na Figura 16 há o exemplo do uso dos retrovisores contendo uma ordem para sua repetição, demonstrando o nível de complexidade no qual ele pode ser utilizado dentro de documentos.

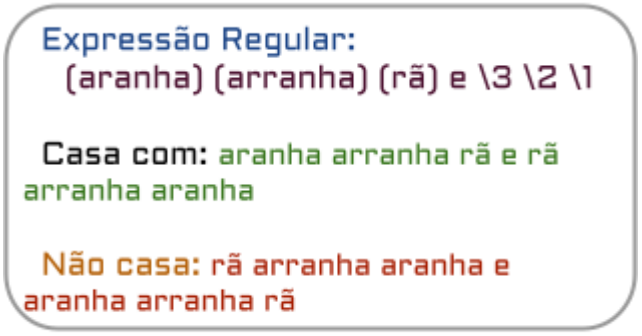

Figura 16: Exemplo de uso do metacaractere retrovisor.

A contagem dos retrovisores ocorre da esquerda para direita e este é o número do retrovisor, ou seja, o grupo um é o retrovisor \1 e assim sucessivamente. Um detalhe importante é que não é possível utilizar retrovisor sem a existência de grupos.

2.3. O uso de jogos no ensino

As instituições de ensino na busca de mudanças por estratégias metodológicas na educação padronizada, ficam atentas às novas exigências tecnológicas que surgem cotidianamente (Lozza, 2017). Os profissionais da área educacional utilizam de recursos pedagógicos, como jogos e brincadeiras para auxiliar a transposição de conteúdos aos alunos de forma que desperte o interesse e a curiosidade deles nas salas de aula (Dorneles, 2011). De acordo com Lozza (2017), aplicar novas estratégias de ensino é trazer o aluno para perto e fazer com que ele se sinta parte das instituições e do mundo.

A Lei de Diretrizes e Bases da Educação Nacional (LDB) (Brasil, 1996) e as Diretrizes Curriculares Nacionais (DCN) (Brasil, 2018) incentivam as instituições formadoras

a buscarem por um modelo de ensino-aprendizagem que possibilite a aproximação do aluno com realidade social para gerar uma reflexão sobre o meio que vive e adquirir um raciocínio crítico a fim de buscar transformar seu contexto (ALMEIDA, 2021).

Neste contexto, os jogos eletrônicos são uma opção viável para transformar o modelo de ensino-aprendizagem em uma educação mais dinâmica e lúdica aos alunos (Fialho, 2008). Os autores Silva e Brincher (2012) ressaltam que é possível perceber os indícios positivos da possibilidade de jogos no ensino devido a profusão de gêneros, plataformas, popularidade e o aumento significativo nos financiamentos de projetos dessa natureza. Ademais, eles concluem que os jogos oferecem a possibilidade de enriquecer o potencial de aprendizado de uma forma crítica e criativa (BRINCHER, 2012).

2.3.1 Os tipos de jogos

Um jogo pode ser definido como uma atividade estimulante e recreativa que envolve uma competição que apresenta um conjunto de regras, sendo estas executadas dentro de um ambiente restrito com envolvimento social em um espaço e tempo determinados (Zucarelli, 2013). Desse modo, existe uma variedade no tipos de jogos como:

- Board Games: conhecidos também como jogos de tabuleiro, são jogos que geralmente possuem uma grande interação entre os jogadores dentro de um tempo curto de jogo. Uma de suas características principais é a pouca influência da sorte e a competição por pontos ou recursos (do PRADO, 2018).
- Simulador: são jogos que permitem ao jogador aplicar seus conhecimentos na tomada de decisão, explorar estratégias alternativas e suas consequências, em um ambiente seguro e próximo do mundo real (da SILVA, 2016).
- Quiz: é um jogo com perguntas e respostas com o objetivo de obter um feedback dos jogadores sobre determinado assunto. Este jogo estimula a aprendizagem direta e auxilia no processo de autoavaliação (NASCIMENTO et al, 2022).
- Puzzle: são jogos com estrutura lógica aberta que encaminha para um processo reflexivo e resulta sobre a compreensão de um dado problema. Ele promove ao jogador a melhoria de formular, testar, questionar e resolver problemas (TONÉIS, 2016).

- Digital Games: é uma atividade lúdica exercitada por ações e decisões que resultam em uma condição final. Estas decisões são limitadas por um conjunto de regras e pelo universo em que o jogo se ambienta (LUCCHESE, 2009).
- Jogos Sérios (Serious Game): são aplicações de software ou hardware desenvolvidas a partir dos princípios do *game design*, onde o objetivo final não é o entretenimento (Richvoldsen, 2009). Estes jogos são utilizados para ensinar, treinar e promover hábitos saudáveis e mudanças sociais (Ritterfield, 2009). Os jogos sérios possuem mais do que apenas história, arte e software. Eles combinam pedagogia e atividades que transmitem conhecimento ou habilidade durante o jogo (ZYDA, 2005).
- RPG (Role Playing Game): é um jogo de representação de papéis, onde a cooperação e a criatividade são os principais elementos. Nele, o jogador controla o seu personagem dentro de um ambiente e dependendo das suas escolhas, os atributos do personagem podem se alterar e assim construir dinamicamente uma história (GRANDO et al, 2008).

2.3.2 O uso de jogos na educação

Os jogos engajam e atraem seus espectadores de forma tão sutil que quase não é perceptível. Isto ocorre, por conta dos elementos chave que possui, os quais são: regras, objetivos, resultados, desafio, interação e história (Lozza, 2017). Os jogos digitais educacionais possuem os mesmos elementos, além de auxiliar os estudantes a analisar seu desempenho com o uso do feedback sobre sua performance e trazer informações sobre um determinado assunto, para que o aluno continue aprendendo durante o jogo (COVOS, 2018).

A introdução de jogos na sala de aula deve ser marcada por uma relação com a aprendizagem, onde o professor e o aluno tenham uma experiência harmônica na construção de conhecimento (Lozza, 2017). A interação com seus pares na troca, no conflito e no surgimento de novas ideias, possibilita a construção de representações sobre algo (Montibeller, 2003). Na aplicação dos jogos, é importante realizar uma testagem antes de sua utilização, além de efetuar uma breve explicação sobre o conteúdo que o jogo irá abranger e por fim, elaborar atividades pedagógicas relacionadas ao jogo para comprovar sua eficácia como ferramenta de ensino (LOZZA, 2017).

O aprendizado utilizando jogos podem ser de maneiras distintas, sendo duas delas através da História e a outra pelas mecânicas de jogo. Os jogos com ensino através da história

possuem conteúdos a serem abordados em sua narrativa e são apresentados ao jogador pela vivência do personagem principal (Lozza, 2017). Por outro lado, os jogos que ensinam através da sua mecânica, transmitem o conteúdo a ser ensinado através do simples fato do jogador estar jogando.

Nesse contexto, é possível observar a possibilidade de transmitir conhecimento de forma lúdica, moderna e que converse com a nova geração de alunos. Assim, os jogos se apresentam como alternativas importantes no processo de aprendizagem, independente da faixa etária, visando o envolvimento dos estudantes e o ensino eficiente (KESSLER, 2010).

2.3.3 O uso de jogos para ensino de programação

A procura por cursos da área de Computação tem crescido nos últimos anos, impulsionada pela transformação digital e pela crescente demanda do mercado por profissionais qualificados (Alvim, 2024). Esse cenário tem ampliado o interesse por formações em Ciência da Computação, Sistemas de Informação e áreas correlatas. Entretanto, em contramão da demanda do mercado, dentro destes cursos há uma alta taxa de desistência e reprovação (PITEIRA, 2018).

Uma das principais razões para este fator, é a dificuldade no ensino e aprendizagem de programação, no qual muitos alunos veem como uma atividade cansativa e entediante, já que precisam aprender a sintaxe e a semântica de uma linguagem de programação junto com a habilidade de resolver problemas (Lindoo, 2018). Na procura de alternativas a este problema, Souza (2016) afirma que o uso de jogos com propósito é uma das soluções mais sugeridas para despertar o interesse e melhorar o aprendizado sobre programação.

O aprendizado através de jogos é muito mais eficiente e pode ser aplicado para todas as idades. Isto ocorre pela presença de componentes do cotidiano nos jogos, o que por sua vez, desperta o interesse do aprendiz e o torna um sujeito ativo do processo (Silva et al, 2017). Os pesquisadores Uzunca e Jansen (2016), afirmam que os jogos são uma nova forma de aprender, pois proporcionam o aprendizado em uma realidade virtual, o que encoraja o jogador, já que há a possibilidade de recomeçar, além de possibilitar a capacidade de encontrar e usar informações sem a necessidade de memorização.

Os jogos sérios ou serious game, são caracterizados pela relação entre o jogo em si e o conteúdo educacional a ser aprendido (Monclar, 2018). Eles utilizam o entretenimento como meio para alcançar objetivos além da diversão. Esses objetivos podem estar relacionados à educação, treinamento, saúde, políticas públicas ou comunicação estratégica (ZYDA, 2005).

Os jogos sérios são caracterizados pela integração entre entretenimento e objetivos educacionais, permitindo que os jogadores desenvolvam conhecimentos e habilidades durante a experiência de jogo. Segundo Lindoo (2018), esses jogos podem ser fundamentados no conceito de computação humana, uma abordagem que utiliza as capacidades cognitivas dos indivíduos para a resolução de problemas. Além disso, proporcionam experiências de aprendizagem de baixo custo e baixo risco, contribuindo para o desenvolvimento de habilidades como atenção e percepção visuo-espacial (MITAMURA, 2012).

Além de elementos comuns aos jogos digitais, como narrativa, arte e software, os jogos sérios incorporam componentes pedagógicos voltados à transmissão de conhecimentos e ao desenvolvimento de competências. Entretanto, a dimensão educacional não deve substituir o caráter lúdico do jogo. O entretenimento permanece como elemento central da experiência, enquanto a pedagogia é integrada às mecânicas e à narrativa, permitindo que o aprendizado ocorra de maneira natural durante a interação do jogador com o ambiente virtual. (ZYDA, 2005).

2.4. Trabalhos relacionados

Esta seção analisa trabalhos que compartilham a intenção de design ou o escopo de conteúdo mais próximos do RegexMon, situando o jogo no panorama atual de recursos educacionais para o ensino de expressões regulares e de conteúdos formais de Computação.

2.4.1 Labirinto Gramágico

O Labirinto Gramágico (Garozi et al., 2021) é o trabalho mais próximo do RegexMon em termos de conteúdo instrucional. Trata-se de um jogo digital de labirinto desenvolvido em Unity, destinado a estudantes de graduação em Ciência da Computação em disciplinas de Linguagens Formais. No jogo, o jogador percorre um labirinto e deve abrir portas construindo expressões regulares que correspondam ao padrão exibido em cada porta.

O trabalho representa uma contribuição relevante por ser um dos poucos jogos dedicados especificamente ao conteúdo de expressões regulares no contexto da educação em Computação. No entanto, sua cobertura pedagógica é restrita à competência de síntese: o jogador sempre escreve uma regex para corresponder a um padrão dado, nunca é desafiado a construir uma string a partir de um padrão. Além disso, os desafios são fixos, limitando a reusabilidade do jogo em múltiplas sessões. O gênero de labirinto oferece pouca estrutura de

progressão narrativa ou curricular, e a ausência de feedback além do binário corresponde ou não corresponde torna o diagnóstico de erros difícil para estudantes iniciantes.

O RegexMon compartilha o mesmo público-alvo e conteúdo, mas difere em todos os demais aspectos: gênero RPG de mundo aberto com narrativa original, abordagem das duas competências em cada batalha, geração procedural de desafios e progressão pedagógica estruturada em ginásios alinhados ao currículo.

2.4.2 Ferramentas web de prática de regex

Ferramentas como Regex Golf, RegexOne e Regex Crossword oferecem ambientes de exercício baseados em repetição nos quais o usuário constrói padrões para corresponder ou diferenciar conjuntos de strings. Essas ferramentas são amplamente utilizadas em contextos de autoestudo e apresentam algumas qualidades pedagógicas relevantes: o Regex Golf incentiva a otimização de padrões pela menor quantidade de caracteres; o RegexOne oferece uma progressão tutorial com explicações conceituais; o Regex Crossword propõe desafios combinatórios que exigem raciocínio sobre interseção de padrões.

No entanto, essas ferramentas não constituem jogos no sentido pedagógico pleno identificado por Lozza (2017), não possuem narrativa, progressão com consequências, estrutura de engajamento que sustente o uso ao longo do tempo, nem mecanismo de feedback além do binário de correspondência ou destaque visual. Abordam exclusivamente a competência de síntese e são concebidas para aprendizes autodirigidos com motivação intrínseca, não para uso estruturado em sala de aula com estudantes que ainda estão construindo essa motivação. A ausência de progressão curricular significa que não há garantia de que o estudante seja exposto a todos os grupos de metacaracteres de forma ordenada.

2.4.3 Posicionamento do RegexMon

O Quadro 6 resume a comparação entre os trabalhos analisados e o RegexMon em seis dimensões de design e pedagógicas.

Dimensão	Labirinto Gramágico	Ferramentas web	RegexMon
Meio	Digital (Unity)	Web (navegador)	Digital (Godot 4)
Gênero	Labirinto / puzzle	Ferramenta de prática	RPG de mundo aberto
Conteúdo	Expressões regulares	Expressões regulares	Expressões regulares
Progressão	Conjunto fixo de desafios	Manual ou fixo	3 ginásios + procedural

Competência abordada	Apenas síntese	Apenas síntese	Reconhecimento + Síntese
Feedback	Corresponde / não corresponde	Destaque visual	Barras de HP + resultado
Público-alvo	Graduação em CC	Geral / profissional	Graduação em CC

Quadro 6. Comparação entre os trabalhos relacionados e o RegexMon.

O RegexMon ocupa uma posição distinta entre os trabalhos analisados. É o único jogo digital dedicado a expressões regulares que aborda simultaneamente as duas competências centrais do domínio, reconhecimento e síntese, dentro de uma única mecânica de jogo. É também o único que combina progressão pedagógica curricular com geração procedural de desafios, garantindo reusabilidade e variedade. A narrativa original ambientada no campus da UNIFAP diferencia o jogo das ferramentas de prática web e do Labirinto Gramágico, aproximando-o do perfil de jogos sérios identificados por Battistella et al. (2016) como mais eficazes para o engajamento em disciplinas de Computação no ensino superior.

3 REGEXMON: O JOGO

O RegexMon é um jogo RPG de mundo aberto ambientado no campus universitário da Universidade Federal do Amapá (UNIFAP). O jogo foi concebido com uma narrativa original que entrelaça a história das expressões regulares com elementos de fantasia e aventura, contextualizando o conteúdo em um ambiente reconhecível pelo público-alvo.

A narrativa tem início quando Mini me acorda misteriosamente no bloco de Ciência da Computação, sem saber como chegou ali. Ao entrar em uma sala do bloco, ela encontra seu avô, conhecido como Vô Chico, que revela a história das expressões regulares por meio de uma metáfora fantástica: dois magos-neurologistas descobriram que os impulsos neurais obedeciam à lógica proposicional e registraram esse conhecimento em um pergaminho encantado; séculos depois, um professor de Matemágica encontrou esse pergaminho e percebeu que criaturas chamadas Regexmons habitavam padrões de linguagem, criando assim os fundamentos das expressões regulares.

Vô Chico e Vó Rita revelam ser os guardiões do campus: mestres de regex que, com o envelhecimento, perderam a força para conter os Regexmons que fazem uso indevido dos padrões. Eles escolheram Mini me para assumir o legado da família e proteger o campus. Mini me aceita a missão, e a jornada tem início.

A escolha de situar a narrativa no campus da UNIFAP é deliberada. Ao reconhecer os espaços do jogo, como o bloco de Ciência da Computação, o campo de futebol, o restaurante universitário (RU), o bloco de artes e os ginásios, os estudantes estabelecem uma conexão imediata entre o universo ficcional e seu ambiente real de estudo. Essa estratégia narrativa reforça a pertinência do conteúdo e favorece o engajamento do público-alvo.

3.1 Personagens

Mini me: a personagem principal controlada pelo jogador. Estudante de Computação que desperta no campus sem saber como chegou ali, Mini me é escolhida pelos avós para assumir o legado familiar e proteger o campus. Ao longo da jornada, ela aprende sobre expressões regulares e as utiliza como "feitiços" nas batalhas contra os Regexmons.

Vô Chico: avô de Mini me e ex-guardião do campus. Aparece no bloco de Ciência da Computação no início do jogo e introduz Mini me ao conceito de expressões regulares e ao contexto da missão. Explica a história dos metacaracteres representantes antes de enviar Mini me ao campo de futebol para obter seus primeiros símbolos.

Vó Rita: avó de Mini me e co-guardiã do campus. Aparece no bloco de artes e ensina Mini me a ler e interpretar expressões regulares antes da primeira batalha. Utiliza a metáfora de uma dança para descrever a leitura sequencial de um padrão regex: âncoras marcam o começo e o fim, representantes indicam os participantes, quantificadores definem o ritmo e grupos criam os pares.

Vilão: antagonista do jogo, responsável por controlar os Regexmons para fins maliciosos. Aparece apenas na batalha final, no bloco de medicina, confrontando Mini me com os desafios mais complexos que combinam todos os grupos de metacaracteres aprendidos ao longo da jornada.

Regexmons: criaturas que habitam padrões de linguagem, presentes tanto nas zonas de grama do mundo aberto quanto nos ginásios. Cada Regexmon apresenta um conjunto de feitiços, expressões regulares ou strings, que a jogadora deve decifrar para vencê-lo. Os Regexmons que aparecem na grama são gerados proceduralmente, com nomes derivados de padrões regex, os de ginásio e batalha final têm desafios fixos, definidos pelos rivais e pelo vilão.

3.2 Mundo do Jogo

O mundo do RegexMon é um mapa aberto que representa o campus universitário, composto por múltiplas zonas navegáveis conectadas. A jogadora se move pelo mapa usando controles direcionais, com o personagem animado em um sistema de tiles de 16x16 pixels que produz um visual estilizado inspirado nos jogos de RPG clássicos de plataformas portáteis.

As principais zonas do mapa incluem: a área externa do campus, com calçadas, grama, árvores e veículos estacionados; o bloco de Ciência da Computação, onde a narrativa tem início; o campo de futebol, onde Mini me obtêm seus primeiros símbolos; o bloco de artes, onde Vó Rita ensina a leitura de expressões regulares; o restaurante universitário (RU), que abriga o segundo ginásio; e o bloco de medicina, cenário da batalha final.

As zonas de grama alta são as áreas onde ocorrem os encontros aleatórios com Regexmons. A cada passo da jogadora nessas zonas, há uma probabilidade de disparar uma batalha aleatória. O nível de dificuldade do Regexmon encontrado é sorteado, expondo a jogadora a diferentes grupos de metacaracteres conforme ela explora o campus. Essa mecânica de encontros aleatórios é o mecanismo de prática repetida do jogo: além dos desafios estruturados dos ginásios, a jogadora acumula experiência com expressões regulares

por meio de encontros variados durante a exploração livre. A Figura 17 apresenta o overworld do RegexMon.

Figura 17. Overworld do RegexMon: campus universitário com zonas de grama, edificações e personagem principal.

3.3 Mecânica de Batalha

O sistema de batalha é o mecanismo pedagógico central do RegexMon. Cada batalha, seja um encontro aleatório na grama ou um desafio de ginásio, é composta por dois turnos sequenciais que operacionalizam as duas competências fundamentais do domínio de expressões regulares. A especificação completa da mecânica de batalha, incluindo todos os desafios dos ginásios e da batalha final, encontra-se documentada no GDD (Apêndice A).

Turno de ataque (reconhecimento): o Regexmon inimigo exibe uma expressão regular na tela de batalha, acompanhada de um campo de entrada. A jogadora deve digitar uma string que corresponda ao padrão exibido. A entrada é validada pelo motor de regex nativo do Godot 4: se a string fornecida satisfaz o padrão, o ataque é bem-sucedido e o Regexmon perde metade de seu HP. Uma resposta incorreta decrementa o HP da jogadora em um passo. Após três respostas incorretas distribuídas entre os dois turnos, a batalha termina em derrota e a jogadora retorna ao overworld no ponto onde o encontro ocorreu, sem penalidade persistente.

Turno de defesa (síntese): o Regexmon exibe três strings que devem ser aceitas e três strings que devem ser rejeitadas, todas derivadas de uma expressão regular com solução que permanece oculta. A jogadora deve digitar uma expressão regular que aceite simultaneamente todas as strings do conjunto de aceitas e rejeite todas as strings do conjunto de rejeitadas. A validação utiliza correspondência de string completa, garantindo que o padrão seja preciso e não excessivamente permissivo. Uma resposta correta causa o dano restante ao Regexmon e encerra a batalha; uma resposta incorreta decrementa o HP da jogadora e a fase se repete com os mesmos conjuntos de strings.

A dualidade ataque-defesa é uma escolha pedagógica deliberada. O turno de ataque desenvolve a capacidade de interpretar um padrão regex e raciocinar sobre as strings que ele aceita, competência exigida ao ler regras de validação, analisar código existente ou depurar um padrão que produz correspondências inesperadas. O turno de defesa desenvolve a capacidade de construir uma expressão regular a partir de especificações comportamentais, competência exigida ao escrever validadores de entrada, padrões de busca ou restrições de formato. Como ambos os turnos são obrigatórios em cada batalha, nenhuma das duas competências pode ser evitada pela escolha de estilo de jogo.

A Figura 18 apresenta a tela de batalha no turno de ataque, e a Figura 19 apresenta o turno de defesa.

Figura 18. Turno de ataque: o inimigo exibe um padrão regex e a jogadora deve digitar uma string correspondente.

Figura 19. Turno de defesa: a jogadora deve sintetizar uma regex que aceite as strings listadas em "Aceita" e rejeite as listadas em "Rejeita".

A jogadora entra em cada batalha com pontos de HP suficientes para três erros combinados entre os dois turnos. A barra de HP da jogadora decrementa visivelmente a cada resposta incorreta, transitando de verde para amarelo e de amarelo para vermelho conforme os erros se acumulam. No terceiro erro, a batalha encerra em derrota e a jogadora retorna ao overworld. Essa consequência é suficientemente significativa para incentivar o raciocínio cuidadoso antes de cada submissão, mas suficientemente baixa para que o fracasso permaneça um evento informativo, e não punitivo.

3.4 Progressão Pedagógica

A progressão pedagógica do RegexMon é organizada em camadas complementares: encontros aleatórios na grama, que oferecem prática variada e não estruturada ao longo de toda a jornada; ginásios, que introduzem cada grupo de metacaracteres de forma estruturada e controlada; e a batalha final, que integra todos os construtos aprendidos em desafios de alta complexidade. Os padrões e respostas esperadas de cada desafio estão detalhados no Apêndice A.

3.4.1 Encontros na Grama

Os encontros aleatórios na grama são o mecanismo de prática contínua do jogo. À medida que Mini me explora o campus, cada passo nas zonas de grama alta tem probabilidade de disparar uma batalha contra um Regexmon gerado proceduralmente. O nível de dificuldade do encontro é sorteado, expondo a jogadora a diferentes grupos de construtos conforme ela avança na narrativa. Esses encontros não ensinam novos conceitos, essa função é reservada aos ginásios, mas reforçam os construtos já introduzidos por meio de repetição em contextos variados.

3.4.2 Ginásio 1: Representantes (Quadra)

O primeiro ginásio está localizado na quadra esportiva do campus. Antes de acessar o ginásio, Mini me já recebeu do Vô Chico uma introdução ao conceito de expressões regulares e aos metacaracteres representantes, e realizou batalhas de treino com a Vó Rita e encontros na grama envolvendo listas e listas negadas.

No ginásio, Mini me enfrenta um rival que desafia seu domínio dos representantes. Os dois desafios do ginásio cobrem o uso combinado do ponto (.), que casa com qualquer caractere, e da lista negada ([^...]), que exclui caracteres específicos. Após vencer o rival, o treinador do ginásio concede à jogadora os símbolos dos quantificadores, desbloqueando o próximo grupo de metacaracteres para os encontros na grama e para o próximo ginásio.

3.4.3 Ginásio 2: Quantificadores (RU)

O segundo ginásio está localizado no Restaurante Universitário (RU). Antes de acessá-lo, Mini me já realizou encontros na grama com cada um dos quatro quantificadores principais: o opcional (?), o asterisco (*), o mais (+) e as chaves ({n,m}).

O ginásio apresenta três desafios que combinam quantificadores com representantes, exigindo que a jogadora aplique as duas famílias de metacaracteres em conjunto. O rival do ginásio anuncia que seus padrões "podem se repetir infinitamente", sinalizando o tema central dos quantificadores. Após a vitória, o treinador concede os símbolos das âncoras.

3.4.4 Ginásio 3: Âncoras (Prédio)

O terceiro ginásio está localizado em um prédio do campus. Os encontros na grama anteriores ao ginásio introduziram os três metacaracteres âncora: o circunflexo (^), que marca

o início da linha; o cifrão ($), que marca o fim da linha; e a borda (\b), que marca o limite entre uma palavra e um não-caractere de palavra.

O ginásio apresenta três batalhas de crescente complexidade. A primeira combina o circunflexo com representantes e quantificadores para exigir padrões que começam ou terminam de forma específica. A segunda utiliza a borda para exigir correspondências com palavras completas de comprimento exato. A terceira combina borda, lista negada e quantificadores em um padrão que exige raciocínio integrado sobre posição, composição e quantidade. Após a vitória, o treinador anuncia a batalha final.

3.4.5 Batalha Final (Bloco de Medicina)

A batalha final ocorre no bloco de medicina e é o clímax da jornada de Mini me. Ao entrar, ela encontra o vilão responsável pelo uso indevido dos Regexmons no campus. Os três desafios da batalha final combinam todos os grupos de metacaracteres aprendidos ao longo da jornada, com padrões de alta complexidade que incluem grupos de captura, âncoras de início e fim, quantificadores com intervalos precisos e listas com ranges.

Os exemplos da batalha final incluem padrões que validam estruturas do mundo real: endereços no formato CEP, endereços de e-mail com formato específico e números de CPF. Esses exemplos reforçam a mensagem pedagógica central do jogo: as expressões regulares não são uma abstração acadêmica, mas uma ferramenta concreta aplicada cotidianamente no desenvolvimento de software.

Após a derrota do vilão, os avós de Mini me aparecem para celebrar a conquista, e o jogo encerra com uma cena de abraço e um texto final que resume a jornada da personagem.

3.5 Sistema de Feedback

O sistema de feedback do RegexMon opera em dois níveis complementares. O primeiro e mais imediato é o resultado da correspondência em si: a string digitada pela jogadora ou corresponde ao padrão do inimigo ou não corresponde, e esse resultado é comunicado imediatamente pelo diálogo de batalha e pela animação da barra de HP. Não há mensagens de erro, avisos de sintaxe ou explicações textuais: a jogadora deve inferir a causa de uma falha examinando a relação entre a string submetida e o padrão exibido, realizando exatamente o raciocínio diagnóstico exigido na depuração profissional de expressões

regulares. Pérez et al. (2018) classificam esse tipo de retorno como feedback intrínseco: a consequência da decisão está embutida no próprio ambiente da tarefa.

O segundo nível é a barra de HP. A barra da jogadora decrementa visivelmente a cada resposta incorreta, transitando de verde para amarelo e de amarelo para vermelho conforme os erros se acumulam em direção ao limite de três. Esse custo visível incentiva o raciocínio cuidadoso antes de cada submissão. A barra do inimigo anima em dois passos distintos, 50% após o turno de ataque e 50% após o turno de defesa, tornando a estrutura de dois turnos visível durante o jogo e fornecendo à jogadora uma representação espacial de seu progresso na batalha.

A ausência de feedback explicativo é uma escolha pedagógica intencional, fundamentada em Hosseini et al. (2019), que identificaram que jogos com mecanismos de feedback intrínseco produzem resultados de aprendizagem mais fortes e duráveis, porque exigem que o aprendiz construa um modelo causal do sistema em vez de simplesmente responder a sinais externos. Ao não fornecer a resposta correta após um erro, o RegexMon mantém a jogadora no ciclo de experimentação, observação e revisão que Kolb (1984) identifica como condição para a eficiência do aprendizado.

3.6 Implementação Técnica

O RegexMon foi desenvolvido em Godot 4, engine de desenvolvimento de jogos de código aberto que oferece suporte nativo a múltiplas plataformas e inclui uma implementação integrada do motor de expressões regulares baseado em PCRE2. A escolha do Godot 4 foi motivada pela sua licença aberta, pela qualidade do suporte a jogos 2D, pela disponibilidade de documentação em português e pela familiaridade da equipe de desenvolvimento com a ferramenta.

O jogo é organizado em torno de módulos funcionais distintos. O gerenciador de cenas coordena as transições entre o overworld e as telas de batalha, com animação de fade que sinalizam a mudança de contexto. O sistema de batalha implementa a lógica dos dois turnos, incluindo a validação de entrada, o controle de HP e a lógica de derrota. O gerador de encontros determina a probabilidade e o nível dos encontros aleatórios na grama. O sistema de diálogo exibe as falas dos personagens durante as cenas narrativas e as introduções dos ginásios.

O mapa do overworld foi construído com um sistema de tiles de 16x16 pixels, utilizando tilesets estilizados que reproduzem a estética visual dos jogos RPG clássicos. Os

Regexmons possuem sprites originais, produzidos especificamente para o projeto, e seus nomes são derivados de padrões regex gerados proceduralmente pelo sistema de encontros aleatórios.

4 POTENCIAL EDUCACIONAL

4.1 Alinhamento Curricular

A progressão pedagógica do RegexMon cobre o currículo padrão de metacaracteres de expressões regulares presente em disciplinas de Linguagens Formais e Autômatos de cursos de graduação em Computação (Garozi et al., 2021). O primeiro ginásio abrange os representantes, ponto e listas, que constituem o ponto de entrada de qualquer instrução em expressões regulares. O segundo ginásio cobre os quantificadores, os construtos mais frequentemente avaliados em exercícios introdutórios e mais comumente compreendidos de forma equivocada devido à sua interação com os representantes. O terceiro ginásio abrange as âncoras, que exigem um raciocínio sobre a posição das correspondências dentro de uma string em vez de sobre caracteres individuais, uma mudança conceitual que corresponde a um ponto de dificuldade documentado no ensino de expressões regulares. A batalha final integra todos os grupos em desafios de alta complexidade que incluem padrões aplicados a estruturas do mundo real, como validação de CEP, endereço de e-mail e CPF.

O turno de defesa, em particular, oferece uma prática difícil de replicar no ensino convencional: a jogadora deve construir uma expressão regular que seja simultaneamente permissiva o suficiente para aceitar um conjunto de strings válidas e restritiva o suficiente para rejeitar um conjunto de strings inválidas. Essa dupla restrição é exatamente o modelo de especificação utilizado em testes de software profissional e validação de entrada de dados, e sua prática repetida no contexto do jogo proporciona uma aprendizagem situada (Lave e Wenger, 1991) que conecta a notação abstrata à sua aplicação prática.

A disciplina de Linguagens Formais e Autômatos foi escolhida como contexto de aplicação do RegexMon por ser o componente curricular em que o estudo de expressões regulares é tradicionalmente abordado nos cursos de graduação em Computação. Nessa disciplina, os estudantes têm contato com conceitos fundamentais para a compreensão de linguagens formais, autômatos e compiladores, sendo as expressões regulares um dos primeiros tópicos apresentados. Entretanto, a natureza abstrata desses conceitos e a sintaxe compacta das expressões regulares podem dificultar o processo de aprendizagem, especialmente para estudantes iniciantes.

Para viabilizar sua utilização nesse contexto acadêmico, o RegexMon foi desenvolvido como uma aplicação desktop utilizando a engine Godot 4 e executada no sistema operacional Windows. O jogo não requer conexão com a internet para seu

funcionamento, sendo necessário apenas um computador compatível com os requisitos mínimos da engine utilizada. Essa característica facilita sua adoção em laboratórios de informática e em computadores pessoais dos estudantes, permitindo sua utilização tanto em sala de aula quanto em atividades extraclasse.

4.2 Princípios Pedagógicos

A arquitetura pedagógica do RegexMon fundamenta-se em três teorias mutuamente reforçadoras, cada uma mapeada sobre um elemento estrutural específico do design do jogo.

O ciclo de aprendizagem experiencial de Kolb (1984) corresponde diretamente a cada batalha. A jogadora submete uma string ou expressão regular (experimentação ativa), observa se ela corresponde ao padrão (experiência concreta), diagnostica a causa do sucesso ou da falha examinando a relação entre a entrada submetida e o padrão exibido (observação reflexiva) e modifica sua abordagem para a próxima tentativa (conceituação abstrata). O ciclo se completa em segundos e pode ser repetido quantas vezes a jogadora necessitar dentro de uma única batalha, proporcionando a prática de ciclos curtos e repetíveis que Kolb identifica como condição para a eficiência do aprendizado.

A teoria da aprendizagem situada de Lave e Wenger (1991) é operacionalizada pela progressão por ginásios. Cada novo grupo de construtos é introduzido por meio de encontros nos quais esses construtos aparecem nos padrões do inimigo antes de qualquer instrução explícita. A jogadora encontra um quantificador em uma batalha antes de receber qualquer definição do que são quantificadores, e deve raciocinar sobre seu comportamento a partir do resultado da correspondência. Essa é a inversão estrutural que a aprendizagem situada prescreve: o problema precede e motiva o conceito, que adquire significado por meio do problema específico que resolve. Os diálogos dos treinadores dos ginásios, que explicam os construtos após a vitória, funcionam como uma formalização do conhecimento que a jogadora já construiu empiricamente durante as batalhas.

A zona de desenvolvimento proximal de Vygotsky (1978) é operacionalizada pela estrutura de progressão por ginásios e encontros. Dentro de cada ginásio, os desafios são rigorosamente limitados aos construtos definidos para aquela etapa, garantindo que a jogadora nunca seja confrontada com um elemento de padrão que não tenha encontrado em etapas anteriores. O limite de três erros por batalha fornece uma margem de segurança que mantém a tarefa dentro do intervalo aprendível sem eliminar a consequência do erro. Hosseini et al. (2019) identificaram que jogos com mecanismos de feedback intrínseco à tarefa produzem

resultados de aprendizagem mais fortes e duráveis porque exigem que o aprendiz construa um modelo causal do sistema em vez de simplesmente responder a sinais externos, que é exatamente o que o feedback baseado em correspondência do RegexMon demanda.

4.3 Validação Interna

Nesta seção é descrito o procedimento de avaliação interna conduzido sobre o RegexMon, realizado previamente à avaliação empírica com estudantes, que está planejada como trabalho futuro. A avaliação interna foi conduzida por um especialista com atuação nas áreas de Educação em Computação e de expressões regulares, docente com experiência no ensino do conteúdo em disciplinas de graduação. A escolha por uma avaliação por especialista nesta etapa fundamenta-se na abordagem de pesquisa baseada em design adotada neste trabalho, na qual o artefato é refinado iterativamente a partir de critérios derivados da teoria pedagógica antes de ser exposto ao público-alvo, reduzindo o risco de submeter aos estudantes uma versão com inconsistências conceituais ou pedagógicas.

A avaliação foi organizada em duas dimensões complementares. A primeira, de natureza técnico-conceitual, examinou a correção dos conteúdos de expressões regulares presentes no jogo, considerando a distinção entre os dois tipos de desafio existentes. Os desafios dos ginásios e da batalha final são roteirizados, definidos manualmente no roteiro pedagógico e apresentados uma única vez à jogadora ao longo da progressão. Por constituírem um conjunto finito, foram verificados de forma exaustiva: para cada um deles, o especialista confirmou que o padrão apresentado na fase de ataque era sintaticamente válido e semanticamente coerente com os construtos do respectivo ginásio, que o desafio de defesa admitia ao menos uma solução construível apenas com os construtos já introduzidos até aquela etapa, e que os conjuntos de strings aceitas e rejeitadas eram consistentes com a solução esperada, sem ambiguidades que permitissem soluções triviais ou excessivamente permissivas. Os encontros na grama, por sua vez, têm seus desafios gerados proceduralmente, de modo que a verificação incidiu sobre as regras de geração: os conjuntos de construtos permitidos em cada nível, a lógica de composição dos padrões a partir desses construtos, o procedimento de construção das strings aceitas e as operações de mutação utilizadas na geração das strings rejeitadas. Complementarmente, foram avaliadas amostras de desafios gerados em cada nível, confirmando que as propriedades verificadas nos desafios roteirizados também se sustentavam nas instâncias produzidas pelo gerador.

A segunda dimensão, de natureza pedagógica, examinou a adequação do jogo enquanto recurso de ensino. Nesse exame foram considerados o alinhamento entre os desafios e os objetivos de aprendizagem definidos na fase de design pedagógico, a coerência da progressão de dificuldade dentro de cada ginásio e entre ginásios, a manutenção dos desafios dentro do intervalo aprendível discutido na Seção 4.2, à luz da zona de desenvolvimento proximal, e a clareza do feedback fornecido à jogadora após cada tentativa, dado que o feedback intrínseco à tarefa constitui um dos fundamentos pedagógicos do design. Vale ressaltar que, no caso dos encontros na grama, essa análise contemplou também a função pedagógica específica desse mecanismo, a prática contínua por repetição em contextos variados, verificando que os desafios gerados reforçaram construtos já introduzidos sem antecipar conteúdos reservados a ginásios posteriores.

O procedimento foi conduzido por meio de sessões de jogo completas, do início do mundo aberto até a batalha final, combinadas com a inspeção analítica do roteiro pedagógico dos ginásios, das regras de geração procedural dos encontros na grama e do Documento de Design do Jogo (GDD). As inconsistências identificadas, tais como padrões roteirizados que admitiam soluções fora do escopo do ginásio ou mutações que produziam strings rejeitadas redundantes, foram registradas e corrigidas em ciclos sucessivos de implementação e reavaliação, em conformidade com a natureza iterativa da pesquisa baseada em design. Ao final do processo, constatou-se que todos os desafios roteirizados e os desafios gerados nas amostras avaliadas eram solucionáveis com os construtos definidos para cada etapa e que a progressão de dificuldade era consistente com a estrutura pedagógica planejada.

Cabe destacar que esta avaliação possui limitações inerentes ao seu caráter interno. Por ter sido conduzida por um único especialista, sem a participação de usuários externos, seus resultados indicam a consistência conceitual e pedagógica do artefato, mas não permitem afirmações sobre a experiência de jogo, a usabilidade ou os ganhos de aprendizagem junto ao público-alvo. Além disso, no caso dos desafios procedurais, a verificação por amostragem não garante a correção de todas as instâncias possíveis do gerador, embora a validação automática de cada mutação contra o padrão, descrita no GDD, mitigue parte desse risco. Desta forma, os achados desta etapa devem ser interpretados como uma condição necessária, porém não suficiente, para a qualidade educacional do jogo, cuja verificação empírica, com estudantes de Ciência da Computação da UNIFAP e o instrumento IAQJEd, constitui a primeira direção de trabalho futuro apresentada no Capítulo 5.

5 CONSIDERAÇÕES FINAIS

Este trabalho apresentou o RegexMon, um jogo RPG de mundo aberto desenvolvido em Godot 4 para apoiar o ensino de expressões regulares em cursos de graduação em Ciência da Computação. O jogo foi concebido a partir de uma lacuna concreta identificada na literatura: a escassez de recursos educacionais dedicados ao ensino de expressões regulares, em particular jogos que abordem simultaneamente as competências de reconhecimento e síntese de padrões.

A contribuição central do RegexMon é seu sistema de batalha em dois turnos, que operacionaliza ambas as competências dentro de uma única mecânica de jogo. No turno de ataque, a jogadora interpreta um padrão e constrói uma string correspondente; no turno de defesa, ela sintetiza uma expressão regular a partir de um conjunto de strings aceitas e rejeitadas. Como ambos os turnos são obrigatórios em cada batalha, nenhuma das duas competências pode ser evitada. Essa integração diferencia o RegexMon das ferramentas de prática de regex existentes, que abordam exclusivamente a síntese, e do único trabalho precedente com conteúdo similar, o Labirinto Gramágico, que também se restringe à síntese com desafios fixos.

A progressão pedagógica do jogo é estruturada em três ginásios correspondentes aos grupos de metacaracteres, representantes, quantificadores e âncoras, complementados por encontros aleatórios na grama para prática contínua e por uma batalha final integradora. A narrativa, ambientada no campus da UNIFAP com personagens originais, contextualiza o conteúdo em um ambiente reconhecível pelo público-alvo e estabelece uma conexão entre a ficção do jogo e o uso prático das expressões regulares no desenvolvimento de software, evidenciada pelos exemplos da batalha final: validação de CEP, e-mail e CPF.

O jogo encontra-se funcional com o mundo aberto completo, o sistema de batalha implementado e todos os desafios dos ginásios e da batalha final definidos e validados internamente. A validação interna confirmou que todos os desafios são solucionáveis com os construtos definidos para cada etapa e que a progressão de dificuldade é consistente com a estrutura pedagógica planejada.

Três direções de trabalho futuro são prioritárias. A primeira é a avaliação empírica com estudantes de Ciência da Computação da UNIFAP, utilizando o instrumento IAQJEd para avaliar a qualidade do jogo e pré e pós-testes para medir ganhos de aprendizagem nas competências de reconhecimento e síntese de padrões. Essa avaliação constituirá o objeto de uma pesquisa subsequente. A segunda é a expansão da progressão pedagógica para cobrir o

quarto grupo de metacaracteres, o grupo "outros", que inclui escape, alternância, grupos de captura e retrovisores, com um quarto ginásio e novos encontros na grama. A terceira é a implementação de um modo de batalha de ginásio com desafios adaptativos baseados no desempenho da jogadora, movendo o jogo na direção de sistemas de aprendizagem adaptativa.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, F. S., de Oliveira, P. B., & dos Reis, D. A. (2021). *A importância dos jogos didáticos no processo de ensino aprendizagem: Revisão integrativa. Research, Society and Development, 10(4), e41210414309-e41210414309.*

ALVIM, Ícaro Vasconcelos; BITTENCOURT, Roberto A.; DURAN, Rodrigo Silva. Evasão nos cursos de graduação em Computação no Brasil. *In: SIMPÓSIO BRASILEIRO DE EDUCAÇÃO EM COMPUTAÇÃO (EDUCOMP), 4., 2024, São Paulo. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2024. p. 309-320.*

BATTISTELLA, P. E.; VON WANGENHEIM, C. G.; FERNANDES, J. M.; PACHECO, H. F. Games for teaching computing in higher education: a systematic review. *IEEE Technology and Engineering Education*, v. 9, n. 1, p. 8-30, 2016.

BRASIL, G., *Cursos de tecnologia: pesquisa mostra que a busca cresceu 39 https://noticias.gs1br.org/busca-por-cursos-de-tecnologia-cresceu-39/, Março.*

BRASIL. Lei nº 9.394, de 20 de dezembro de 1996. *Estabelece as diretrizes e bases da educação nacional. Brasília, DF: Diário Oficial da União, 1996.*

BRASIL. Conselho Nacional de Educação. Câmara de Educação Básica. Resolução CNE/CEB nº 3, de 21 de novembro de 2018. *Atualiza as Diretrizes Curriculares Nacionais para o Ensino Médio. Brasília, DF: Ministério da Educação, 2018.*

CARNEIRO, K. T. (2015). *Por uma memória do jogo: a presença do jogo na infância de octogenários e nonagenários (Doctoral dissertation, Tese (Doutorado em Educação Escolar). UNESP-Universidade Estadual Paulista, 273f).*

COVOS, J. S., Covos, J. F., Rodrigues, F. R., & Ouchi, J. D. (2018). *O novo perfil de alunos no ensino superior, e a utilização de jogos lúdicos para facilitação do ensino aprendizagem. Revista Saúde em Foco, 1, 63-74.*

da SILVA, R. R. L., ZATTAR, I. C., CLETO, M. G., & STEFANO, N. M. (2016). *O uso de jogos e simulação como métodos alternativos de ensino em engenharia no Brasil: uma revisão bibliográfica. Revista ESPACIOS| Vol. 37 (Nº 05) Ano 2016.*

DCN: *Parecer CNE/CEB nº 3/2018, aprovado em 8 de novembro de 2018 - Atualização das Diretrizes Curriculares Nacionais para o Ensino Médio, observadas as alterações introduzidas na LDB pela Lei nº 13.415/2017.*

do PRADO, L. L. (2018). *Educação lúdica: os jogos de tabuleiro modernos como ferramenta pedagógica. Revista Eletrônica Ludus Scientiae, 2(2).*

DORNELES, R. M. C. T., & EDUCAÇÃO, A. L. N. (2011). *Uma atitude pedagógica 2 ed. Ver., atual e ampl. Curitiba: Ibepex.*

GAROZI, H.; VON WANGENHEIM, C. G.; HAUCK, J. C. R. Labirinto Gramágico: Um Jogo Educativo para o Ensino de Gramáticas Regulares. In: *Anais do XX Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames 2021)*, 2021.

GEE, J. P. *What Video Games Have to Teach Us About Learning and Literacy.* New York: Palgrave Macmillan, 2003.

GRANDO, A., & TAROUCO, L. M. R. (2008). *O uso de jogos educacionais do tipo RPG na educação. Revista Novas Tecnologias na Educação, 6(1).*

HOSSEINI, H.; HARTT, M.; MOSTAFAPOUR, M. Learning is child's play: game-based learning in computer science education. *ACM Transactions on Computing Education*, v. 19, n. 3, p. 1-18, 2019.

JARGAS, A. M. *Expressões Regulares: Uma Abordagem Divertida.* 3. ed. São Paulo: Novatec, 2012.

KESSLER, D. M.; MCLEOD, A. J.; SRINIVASAN, P. *Educational games in computing. In: Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE 2010), Milwaukee, WI, p. 555, 2010.*

KLEENE, S. C. *Representation of events in nerve nets and finite automata. In: SHANNON, C.; MCCARTHY, J. (Ed.). Automata Studies. Princeton: Princeton University Press, 1951. p. 3-41.*

KOLB, D. A. *Experiential Learning: Experience as the Source of Learning and Development.* Englewood Cliffs, NJ: Prentice-Hall, 1984.

LAVE, J.; WENGER, E. *Situated Learning: Legitimate Peripheral Participation.* Cambridge: Cambridge University Press, 1991.

LDB: *(BRASIL, Lei Nº 9.394, de 20 de Novembro de 1996. Dispõe sobre as diretrizes e bases da educação nacional).*

LINDOO, E. (2018). *Back to the basics in an effort to improve student retention in intro to programming classes. J. Comput. Sci.*

LOZZA, A. B (2017). O uso de jogos como metodologia de ensino. *Revista Educação Pública*, v. 17, n. 10, 2017.

LUCCHESE, F., & Ribeiro, B. (2009). *Conceituação de jogos digitais. São Paulo, 7.*

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, v. 5, p. 115-133, 1943.

MELO, G. F., & NAVES, M. L. D. P. (2017). *Retenção e evasão: desafios para a gestão da educação superior. Acesso em, 20.*

MITAMURA, T., Suzuki, Y., & Oohori, T. (2012, October). Serious games for learning programming languages. In *2012 IEEE international conference on systems, man, and cybernetics (SMC)* (pp. 1812-1817). IEEE.

MONCLAR, R. S., Silva, M. A., & Xexéo, G. (2018). Jogos com Propósito para o Ensino de Programação. *Anais do XVII Simpósio Brasileiro de Jogos e Entretenimento Digital–SBGames*, 1132-1140.

NASCIMENTO, J. M. T. de S, Lima, N. D. J. F., Lima, C. D., dos Santos, L. B. P., Pires, C. J., da Silva, V. L. R., ... & Nunes, M. D. J. M. (2022). *Quiz: Um jogo on-line como ferramenta no ensino remoto de Biologia. Research, Society and Development, 11(15), e116111536706-e116111536706.*

PÉREZ, E. del M.; DUQUE, A. P. G.; GARCÍA, L. C. F. Game-based learning: increasing the logical-mathematical, naturalistic, and linguistic learning levels of primary school students. *Journal of New Approaches in Educational Research*, v. 7, n. 1, p. 31-39, 2018.

PRENSKY, M. *Digital Game-Based Learning.* New York: McGraw-Hill, 2001.

PITEIRA, M. D. R. F. (2018). *Aprendizagem da programação no Ensino Superior: a adoção de cursos" online" gamificados.*

RICHVOLDSEN Richvoldsen, H. (2009). *Serious Gaming: Serious content in an entertaining framework* (Master's thesis, Institutt for elektronikk og telekommunikasjon).

RITTERFELD, U. (2009). *Serious Games: Mechanisms and Effects*. Routeledge.

SILVA, L. F. D., Silva, M. I. M. D., Figueredo, M. D. F. N., Santos, M. A. D., & Sousa, R. B. D. (2017). *A contribuição dos jogos no ensino da matemática, nas séries iniciais.*

SOMMARIVA, L. W. (2012). *UsabilityGame: jogo simulador para apoio ao ensino de usabilidade.*

SPENCER, H. *regexp.* Usenet: comp.sources.unix, 1986.

THOMPSON, K. Programming techniques: regular expression search algorithm. *Communications of the ACM*, v. 11, n. 6, p. 419-422, 1968.

TONÉIS, C. N. (2016). *O design de Puzzles nos jogos digitais. SBC–Proceedings of SBGames, 404-411.*

VIEIRA, N. J. (2006). *Introdução aos fundamentos da computação: linguagens e máquinas. Pioneira Thomson Learning.*

VYGOTSKY, L. S (1978). *Mind in Society: The Development of Higher Psychological Processes.* Cambridge: Harvard University Press, 1978.

ZUCARELLI, I; COUTO. L (2013). *Jogo de tabuleiro em incentivo à alimentação infantil. Universidade São Judas Tadeu, São Paulo, 2013.*

ZYDA, MICHAEL (2005). *From visual simulation to virtual reality to games. Computer, v. 38, n. 9, p. 25–32, 2005.*

APÊNDICE A

DOCUMENTO DE DESIGN DO JOGO (GDD)

REGEXMON: Um jogo de apoio ao ensino-aprendizagem de Expressões Regulares

Versão 1.0

2025

1. VISÃO GERAL DO JOGO

Título: RegexMon
Gênero: RPG de mundo aberto, batalha por turnos
Plataforma: PC (Windows, Linux, macOS), com possibilidade de exportação para Web
Engine: Godot 4
Público-alvo: Estudantes de graduação em Ciência da Computação, especialmente em disciplinas de Linguagens Formais, Autômatos e Teoria da Computação
Objetivo pedagógico: Apoiar o ensino-aprendizagem de expressões regulares, desenvolvendo as competências de reconhecimento de padrões (interpretação de uma regex e construção de uma string correspondente) e síntese de padrões (construção de uma regex a partir de um conjunto de strings aceitas e rejeitadas)
Perspectiva: Visão superior (top-down), estilo dos RPGs clássicos de plataformas portáteis
Modo de jogo: Um jogador

O RegexMon é um jogo sério no estilo RPG de mundo aberto ambientado no campus universitário da UNIFAP. A jogadora controla Mini me, uma estudante de Computação que assume o legado de seus avós, mestres de regex, para proteger o campus de criaturas chamadas Regexmons que fazem uso indevido de padrões de linguagem. O jogo ensina expressões regulares por meio de batalhas por turnos em que a jogadora deve aplicar os construtos da notação para vencer os inimigos.

2. NARRATIVA

2.1. Premissa

Em um mundo onde ciência e magia caminham lado a lado, dois magos-neurologistas descobriram que os impulsos neurais obedeciam à lógica proposicional. Séculos depois, um professor de Matemágica encontrou os registros desse conhecimento e percebeu que criaturas chamadas Regexmons habitavam padrões de linguagem. Esses seres mágicos aparecem sempre que uma sequência de símbolos se repete com regularidade, e seu comportamento pode ser previsto e controlado por meio das expressões regulares.
Vô Chico e Vó Rita, avós de Mini me, foram grandes mestres de regex e guardiões do campus da UNIFAP. Com o envelhecimento, perderam força para conter os Regexmons que passaram

a usar seus poderes para fins maliciosos, controlados por um vilão. Eles escolheram Mini me para assumir o legado e proteger o campus.

2.2. Personagens

Mini me

Papel: Protagonista controlada pelo jogador
Descrição: Estudante de Ciência da Computação que acorda misteriosamente no bloco de CC da UNIFAP. Determinada e corajosa, aceita a missão de seus avós e aprende a dominar as expressões regulares ao longo da jornada.

Vô Chico

Papel: Mentor inicial, avô de Mini me
Descrição: Ex-guardião do campus. Aparece no bloco de CC no início do jogo, apresenta o universo das expressões regulares por meio de uma metáfora histórica e explica os metacaracteres representantes antes de enviar Mini me ao campo de futebol.

Vó Rita

Papel: Mentora de combate, avó de Mini me
Descrição: Co-guardiã do campus. Aparece no bloco de artes e ensina Mini me a ler expressões regulares utilizando a metáfora de uma dança: âncoras marcam começo e fim, representantes indicam os participantes, quantificadores definem o ritmo e grupos criam os pares. Conduz a primeira batalha de treino.

Vilão

Papel: Antagonista principal
Descrição: Responsável por controlar os Regexmons para fins maliciosos. Aparece apenas na batalha final, no bloco de medicina, com os desafios mais complexos do jogo.

Regexmons

Papel: Inimigos

Descrição: Criaturas que habitam padrões de linguagem. Os Regexmons selvagens aparecem nas zonas de grama e são gerados proceduralmente, com nomes derivados de padrões regex. Os Regexmons de ginásio e batalha final têm desafios fixos, definidos pelos rivais e pelo vilão. Possuem 10 sprites originais produzidos para o projeto.

2.3. Estrutura Narrativa

Prólogo: Mini me acorda no bloco de CC. Encontra Vô Chico, que explica o universo dos Regexmons e os metacaracteres representantes. Mini me recebe seus primeiros símbolos no campo de futebol.

Ato 1: Mini me encontra Vó Rita no bloco de artes, aprende a ler expressões regulares e realiza sua primeira batalha de treino. Explora o campus, enfrenta Regexmons selvagens na grama e conquista o Ginásio 1 (representantes) na quadra esportiva.

Ato 2: Mini me aprende os quantificadores por meio de encontros na grama e conquista o Ginásio 2 (quantificadores) no RU. Em seguida, aprende as âncoras e conquista o Ginásio 3 (âncoras) no prédio.

Clímax: Os avós aparecem para encorajar Mini me antes da batalha final. No bloco de medicina, Mini me enfrenta o vilão em três desafios que integram todos os grupos de metacaracteres.

Epílogo: O vilão é derrotado e desaparece. Mini me abraça seus avós. Texto final: "E assim, Mini me alcança o fim de sua jornada no mundo do Regex, carregando com bravura o legado da sua família."

3. GAMEPLAY

3.1. Controles

A jogadora controla Mini me por meio das teclas direcionais (setas ou WASD) para movimentação no overworld. A interação com NPCs e objetos é realizada com a tecla de confirmação (Z ou Enter). Na tela de batalha, a navegação entre opções é feita pelas teclas direcionais e a seleção pela tecla de confirmação. A entrada de texto (strings e expressões regulares) é realizada por meio de um campo de digitação ativado ao selecionar a opção de ataque.

3.2. Loop Principal de Jogo

O loop principal do RegexMon alterna entre dois modos: exploração do overworld e batalha.

Exploração: Mini me navega pelo campus em visão superior, interage com personagens e objetos, e acessa as zonas de grama e ginásios. Encontros aleatórios são disparados a cada passo nas zonas de grama alta, com probabilidade configurável.
Batalha: Cada encontro dispara a tela de batalha. A batalha é composta por dois turnos sequenciais obrigatórios: turno de ataque (reconhecimento) e turno de defesa (síntese). A jogadora possui HP para três erros combinados. No terceiro erro, a batalha encerra em derrota e a jogadora retorna ao overworld sem penalidade persistente.

3.3. Sistema de Batalha

Turno de Ataque (Reconhecimento)

O Regexmon inimigo exibe uma expressão regular gerada pelo sistema. A jogadora deve digitar uma string que satisfaça o padrão. A entrada é validada pelo motor de regex do Godot 4 por correspondência parcial (search). Uma resposta correta causa 50% de dano ao inimigo. Uma resposta incorreta decrementa o HP da jogadora em um passo.

Turno de Defesa (Síntese)

O Regexmon exibe três strings que devem ser aceitas e três strings que devem ser rejeitadas, derivadas de uma expressão regular solução oculta. A jogadora deve digitar uma expressão regular que aceite todas as strings do conjunto de aceitas e rejeite todas as do conjunto de rejeitadas. A validação utiliza correspondência de string completa (fullmatch com âncoras). Uma resposta correta causa os 50% restantes de dano e encerra a batalha. Uma resposta incorreta decrementa o HP da jogadora e a fase se repete.

HP e Derrota

A jogadora inicia cada batalha com HP suficiente para cometer três erros. A barra de HP é visual e transita de verde para amarelo e de vermelho conforme os erros se acumulam. Ao terceiro erro, a batalha termina em derrota e a jogadora retorna ao ponto do encontro no overworld.

3.4. Geração Procedural de Desafios

Os encontros aleatórios na grama geram desafios proceduralmente. O nível do encontro é sorteado aleatoriamente entre os níveis disponíveis. Para cada nível, o sistema seleciona construtos permitidos e compõe um padrão regex a partir deles. As strings aceitas são construídas interpretando o padrão token a token. As strings rejeitadas são geradas por quatro operações de mutação aplicadas às strings aceitas: truncamento, adição de caractere especial, substituição por dígito e duplicação da string. Cada mutação é verificada contra o padrão para confirmar a não-correspondência antes da inclusão.
Os desafios dos ginásios e da batalha final são fixos, definidos manualmente no roteiro pedagógico, garantindo coerência com o currículo de cada etapa.

4. MUNDO DO JOGO

4.1. Mapa Geral

O overworld representa o campus universitário da UNIFAP em visão superior, construído com tiles de 16x16 pixels. O mapa é contínuo e navegável, com múltiplas zonas conectadas por caminhos e passagens.

4.2. Zonas Principais

Zona	Tipo	Função narrativa/pedagógica
Bloco de CC	Interior	Início da narrativa; encontro com Vô Chico
Campo de futebol	Exterior com grama	Mini me obtém os primeiros símbolos; encontros aleatórios
Bloco de Artes	Interior	Encontro com Vó Rita; primeira batalha de treino
Área externa geral	Exterior com grama	Encontros aleatórios de todos os níveis
Quadra (Ginásio 1)	Interior	Batalhas de representantes; desbloqueio de quantificadores
RU (Ginásio 2)	Interior	Batalhas de quantificadores; desbloqueio de âncoras
Prédio (Ginásio 3)	Interior	Batalhas de âncoras; preparação para batalha final
Bloco de Medicina	Interior	Batalha final contra o vilão

4.3. Zonas de Grama Alta

As zonas de grama alta estão presentes no campo de futebol e nas áreas externas do campus. A cada passo da jogadora nessas zonas, há uma probabilidade de disparar um encontro aleatório. O nível do encontro é sorteado de modo a expor a jogadora a construtos variados conforme ela avança na narrativa.

5. PROGRESSÃO PEDAGÓGICA

5.1. Estrutura Geral

Etapa	Local	Conteúdo introduzido
Prólogo	Bloco CC / Bloco Artes	Conceito de regex, representantes (ponto, lista, lista negada), leitura de padrões
Grama (rep.)	Campus geral	Prática com lista e lista negada
Ginásio 1	Quadra	Ponto (.) e lista negada combinados; 2 batalhas
Grama (quant.)	Campus geral	Opcional (?), asterisco (*), mais (+), chaves ({n,m})
Ginásio 2	RU	Quantificadores combinados com representantes; 3 batalhas
Grama (âncoras)	Campus geral	Circunflexo (^), cifrão ($), borda (\b)
Ginásio 3	Prédio	Âncoras combinadas com quantificadores e representantes; 3 batalhas
Batalha Final	Bloco Medicina	Todos os grupos integrados; 3 desafios de alta complexidade

5.2. Desafios dos Ginásios

Ginásio 1, Representantes (Quadra)

Nº	Fase	Feitiço / Desafio	Resposta esperada
1	Ataque	1.d.s — aceita qualquer caractere entre 1, d e s	1zd3s (qualquer string com um caractere entre cada literal)
1	Defesa	Aceita: abc, a1c, a-c / Não aceita: ac, abbc	a.c

2	Ataque	[^abc]7[^xyz] — literal 7 no meio, sem a/b/c antes e sem x/y/z depois	d7k, m7p, z7a
2	Defesa	Aceita: d1e4hg / Não aceita: d123e4hg	d[1-5]e[^5-9][^f]g

Ginásio 2, Quantificadores (RU)

Nº	Fase	Feitiço / Desafio	Resposta esperada
1	Ataque	c[iou]?p — aceita zero ou uma vogal entre i, o, u	cp, cip, cop, cup
1	Defesa	Aceita: bat, bet, bit, bt	b[aei]?t
2	Ataque	[zym]+[0-9]{2,4} — uma ou mais letras de [zym] e 2 a 4 dígitos	zym1234, zzzz9832, ym5869
2	Defesa	Aceita: xa, xya, yyzaa, zzzaa	[xyz]+[a]{1,3}
3	Ataque	b[aei]{1,2}t[xyz]? — 1 a 2 vogais e xyz opcional	baet, bat, beitz
3	Defesa	Aceita: son, soun, souun, sounk	s[ou]{1,3}n[km]?

Ginásio 3, Âncoras (Prédio)

Nº	Fase	Feitiço / Desafio	Resposta esperada
1	Ataque	^pro[a-z]* — começa com "pro" e aceita letras minúsculas depois	projeto, progresso
1	Defesa	Aceita: 123fim, 9fim, 2025fim	[0-9]+fim$
2	Ataque	\b[a-z]{3}\b — palavra isolada de exatamente 3 letras	lua, mar, sol, paz
2	Defesa	Aceita: pa, ir, pé, ar	\b[a-z]{2}\b
3	Ataque	\b[a-z]*ing\b — palavras terminadas em "ing"	coding, testing, learning
3	Defesa	Aceita: bc123ion, ql89ion, dp890ion	\b[^aeiou]{2,3}[0-9]+ion\b

Batalha Final (Bloco de Medicina)

Nº	Fase	Feitiço / Desafio	Resposta esperada

1	Ataque	^([a-z]{1,2}[0-9]+){2,3}$ — 1-2 letras + dígitos, repetido 2-3 vezes	ab11cd34de11, aa0987zz1234
1	Defesa	Aceita: 54675-000, 81340-079, 45455-003 (CEP)	^[0-9]{5}-[0-9]{3}$
2	Ataque	^[a-z]{2,4}(-[a-z]{2,}){2,4}$ — palavras separadas por hífen	mago-poder-vitoria
2	Defesa	Aceita: teste00@email.com, az44@email.com (e-mail)	^[a-z]{2,9}[0-9]{2,4}@email.com
3	Ataque	^([0-9][a-z]{1,2}[0-9]{2,3}){2}$ — padrão numérico duplo	1aa1111aa111, 9z116d33
3	Defesa	Aceita: 041.234.885-00, 180.778.253-08 (CPF)	\b[0-9]{3}.[0-9]{3}.[0-9]{3}-[0-9]{2}\b

6. ARTE E ÁUDIO

6.1. Estilo Visual

O RegexMon adota um estilo visual pixel art inspirado nos RPGs clássicos de plataformas portáteis dos anos 1990 e 2000. Os tiles do overworld são de 16x16 pixels, com paleta de cores saturadas e elementos reconhecíveis do campus universitário: edificações, calçadas, vegetação, veículos e interiores de salas. A tela de batalha reproduz a interface característica do gênero, com o sprite do inimigo exibido em destaque, as barras de HP de ambos os lados e o painel de diálogo na parte inferior.

6.2. Personagens e Regexmons

Mini me é representada por um sprite de personagem jogável com animações de movimento nas quatro direções e animação de idle. Os 10 Regexmons possuem sprites originais produzidos especificamente para o projeto, com designs que remetem visualmente à ideia de padrões e sequências. As 154 imagens de Pokémon originais presentes no repositório de código são utilizadas como fallback para encontros durante o desenvolvimento e não integram a versão final do jogo.

6.3. Interface

Overworld: HUD mínimo, sem exibição permanente de informações. O acesso ao menu de party e opções é realizado por tecla dedicada.
Batalha: Painel inferior com nome e HP do inimigo, HP da jogadora, diálogo de batalha, campo de entrada de texto e botões de ação (ATACAR, VOLTAR).
Fonte: Fonte estilizada no padrão dos RPGs clássicos para os diálogos de batalha e narrativos.

6.4. Áudio

O projeto prevê trilha sonora em estilo chiptune para o overworld e para as batalhas, com efeitos sonoros para acerto, erro, transição de cena e derrota. A implementação de áudio constitui entrega futura.

7. ARQUITETURA TÉCNICA

7.1. Engine e Linguagem

O jogo foi desenvolvido em Godot 4, utilizando GDScript como linguagem de programação. A escolha da engine é motivada por sua licença de código aberto (MIT), pelo suporte nativo a jogos 2D com sistema de tiles, pela inclusão de um motor de expressões regulares baseado em PCRE2 e pela disponibilidade de documentação em português.

7.2. Módulos Principais

Módulo	Responsabilidade
Gerenciador de cenas	Coordena transições entre overworld e batalha, com animações de fade e preservação do estado do mapa
Sistema de batalha	Implementa a lógica dos dois turnos, validação de entrada, controle de HP e condição de derrota
Gerador de padrões	Gera expressões regulares proceduralmente a partir das regras configuradas por nível
Gerador de strings	Constrói conjuntos de strings aceitas (por interpretação do padrão) e rejeitadas (por mutação)
Validador de entrada	Valida strings (correspondência parcial) e expressões regulares (correspondência completa) usando o motor de regex do Godot 4
Sistema de diálogo	Exibe falas dos personagens durante cenas narrativas, introduções de ginásios e encerramento de batalhas
Controlador do jogador	Gerencia movimento tile-a-tile, animações direcionais e detecção de colisão com zonas de grama e portas
Gerador de encontros	Determina probabilidade e nível dos encontros aleatórios nas zonas de grama

7.3. Fluxo de uma Batalha Aleatória

1. Jogadora pisa na grama alta.
2. Sistema sorteia se haverá encontro (probabilidade configurável).
3. Se sim, sistema sorteia o nível do encontro.
4. Gerador de padrões produz uma expressão regular para o nível sorteado.
5. Gerador de strings produz 3 strings aceitas e 3 strings rejeitadas.
6. Gerenciador de cenas transiciona para a tela de batalha com fade.

7. Turno de ataque: jogadora digita string, validador verifica correspondência parcial.
8. Turno de defesa: jogadora digita regex, validador verifica correspondência completa para cada string.
9. Resultado: vitória (inimigo com 0 HP) ou derrota (jogadora com 0 HP).
10. Gerenciador de cenas retorna ao overworld com fade.

APÊNDICE B

ROTEIRO DO JOGO

REGEXMON: Um jogo de apoio ao ensino-aprendizagem de Expressões Regulares

Versão 1.0

2025

Roteiro do Jogo

Início do Jogo

- Mini me acorda no bloco de Ciência da Computação (na frente do bloco)
    - “QUE? Que lugar é esse?? Como eu vim parar aqui??? TEM ALGUÉM AÍ?? Calma, acho que conheço esse lugar”
- Entra em uma sala do bloco
    - Entra na sala e vê a pessoa: “Oi, tudo bem? Você pode me ajudar?” - vai andando na direção da pessoa - É que sou nova aqui e estou um pouco, quer dizer, muito perdida.
        - O avô vira e mini me fala: VÔ CHICO??? Como o senhor está aqui? Como que eu vim parar aqui???
        - Vô: Oi minha filha! Se acalme, vou explicar tudo pra você, sente-se aqui!

Explicação do Avô:

Era uma vez, em um mundo distante onde ciência e magia caminhavam lado a lado, dois magos estudiosos — também conhecidos como neurologistas — tornaram-se fascinados pela complexidade dos neurônios e por como sua comunicação parecia quase mágica.

Movidos pela curiosidade, passaram anos mergulhados em grimórios e experimentos, tentando entender como os impulsos mágicos — ou elétricos — viajavam de um neurônio a outro. Foi então que descobriram algo extraordinário: esses sinais obedeciam a um tipo de raciocínio encantado, muito parecido com a **lógica proposicional**. Eles podiam ser descritos por afirmações que eram verdadeiras ou falsas, como feitiços com respostas binárias.

Empolgados com a descoberta, os magos registraram tudo em um antigo pergaminho encantado. Ali, mostraram que os neurônios se comportavam como proposições mágicas, conectando-se por meio de regras lógicas — uma sinfonia de *e*, *ou*, *se... então* — e assim, formularam a base para um novo tipo de feitiçaria lógica.

Séculos depois, um professor de **Matemágica**, ao explorar a Biblioteca da cidade em busca de artefatos esquecidos, encontrou esse pergaminho. Encantado com o conteúdo, dedicou-se a estudá-lo e levou a teoria adiante. Durante suas pesquisas, o professor passou a observar criaturas curiosas chamadas **Regexmons** — seres que habitavam padrões de linguagem.

Esses animais mágicos apareciam sempre que uma sequência de símbolos, letras ou sons se repetia com regularidade. O professor, intrigado, percebeu que podia prever o comportamento dos Regexmons usando os mesmos princípios da lógica proposicional descrita no pergaminho. Assim, ele criou diagramas, padrões e fórmulas que capturavam o comportamento dessas criaturas, o que mais tarde passou a ser conhecido como os **conjuntos regulares** — ou *regular sets* — base fundamental das expressões regulares (*regex*).

E foi assim que a antiga magia dos neurônios, traduzida em lógica, encontrou um novo lar nas linguagens mágicas da programação.

Mini me: “Isso é surreal vô, é encantador também, mas qual o motivo de eu estar aqui? Como o senhor sabe de tudo isso?”

Avô Chico: “Há muito tempo atrás minha filha, eu e sua avó fomos grandes mestres de Regex e assim como todos os mestres, cada um tinha seu campo para proteger, o meu e a sua avó era este local, esta Universidade. Com o passar do tempo, fomos enfraquecendo e não conseguimos exterminar os animais que faziam uso dessa magia para o mal. Como queríamos deixar um legado, escolhemos você para finalizar a missão. Sabemos que você é capaz de ajudar este lugar! Mas antes, você precisa aprender sobre os conceitos de regex e como usá-lo. E então minha filha? Você aceita essa missão? Este lugar precisa da sua ajuda!”

Mini me: “Que honra mais maluca e admirável vô! Aceito carregar o legado de vocês e farei o meu melhor para me tornar a maior mestre de Regex!”

Avô Chico: “Que ótimo minha filha. Deixa eu explicar para você o que são as expressões regulares:

- Expressão regular é um método formal de especificar um padrão de texto. Esse padrão é uma composição de símbolos e caracteres com funções que ao serem agrupados entre si e com outros caracteres literais, geram uma sequência que é uma expressão.
- A expressão criada, através desse conjunto de caracteres, é considerada uma regra, que avalia se uma entrada de dados qualquer irá casar com ela e obedecer às suas condições.”

Mini me: “Tá, acho que entendi o conceito, mas como assim composição de símbolos e caracteres? Que símbolos??”

Avô Chico: “O professor Matemátigo conseguiu identificar os símbolos usando nas expressões, como eles funcionam e quais os seus grupos.

- “Cada símbolo possui uma função dentro da magia e que dependendo do contexto que for inserido, pode mudar e gerar outro feitiço. Apesar de delicados, quando são agrupados, podem gerar diversos feitiços com funções complexas”.
- “Dentro dos símbolos, há pequenas divisões: representantes, quantificadores, âncoras e metacarecteres especiais. Eles são divididos de acordo com suas características comuns.
- Por enquanto, explicarei somente sobre os Representantes, ao decorrer de sua jornada, você aprenderá o restante…
- Dentro do grupo de representantes temos:
    - Representantes: sua função é representar um ou mais caracteres
    - . (ponto) = sua função é encontrar qualquer caractere, ou seja, qualquer feitiço.
    - [...] (lista) = ela demonstra quais caracteres são permitidos, ou seja, quais serão usados para gerar o feitiço

- [^...] (lista negada) = ela demonstra quais caracteres são proibidos, ou seja, quais não poderão ser usados para gerar um feitiço.

Mini me: “Certo, consegui entender! Mas, por onde começo?”

Avô Chico: “Vá para o campo de futebol e pegue a luz brilhante que está por lá, ela lhe dará alguns símbolos e caracteres para você começar! Boa sorte minha filha, contamos com você!”

- Depois da conversa, ela sai do bloco e vai em direção ao Campo de Futebol (lá terá algo brilhante e ela ganhará os símbolos dos representantes)

- Já estando com os símbolos, ela vê uma luz brilhando no Bloco de Artes. Ao entrar, encontra sua avó sentada no banco do corredor

Mini me: “Vó Rita?? Você também está aqui? Bem que o vovô falou que vocês são os guardiões desse lugar. Tinha uma luz brilhando aqui, você a viu?”

Vó Rita: “Era eu tentando chamar sua atenção com uma lanterna. Vi que você pegou alguns símbolos no campo de futebol. Já sabe como irá usá-los, o que seu avô lhe explicou?”

Mini me: “Não vó, estou um pouco perdida. O vô Chico explicou sobre os símbolos e os regex, mas como irei lutar contra os regexmons assim?

Vó Rita: “Vamos por partes. Primeiro você precisa aprender a ler uma Regex

Pense nelas como uma dança…

Primeiro, analise **os limites**, âncoras como `^` (começo) e `$` (fim) indicam onde o feitiço começa e termina.

Depois, verifique quem participa da dança, pontos, listas ou listas negadas mostram quais caracteres podem ou não aparecer.

Em seguida, se atente ao ritmo, quantificadores como `*`, `+` ou `{n,m}` dizem quantas vezes algo pode se repetir.

E por fim, note os pares, parênteses criam grupos, e o símbolo `|` oferece caminhos alternativos, como ‘ou este, ou aquele’. “

Mini me: “Uau vó, entendi. Acho que estou pronta para o combate”

Vó Rita: “Então vamos para sua primeira batalha…”

- E então começa a 1° batalha de Mini me

BATALHAS DO JOGO

1ª Batalha – Treino com a Avó

Feitiço: `.`
👉 Aceita qualquer caractere.

Resposta: `aaa`
✅ O padrão é aceito.

Resultado: O feitiço é quebrado.

REPRESENTANTES

🎈 1ª Batalha – Na Grama com o Regexmon (USO DE LISTA)

ATAQUE:

O Regexmon surge na grama, soltando rugidos ameaçadores e mostrando seus símbolos estranhos pelo corpo.

Fala: "Ahhh… tente me decifrar se for capaz!"

Explicação:
As listas `[ ]` servem para indicar quais caracteres são aceitos. Por exemplo, `[abc]` significa que apenas `a`, `b` ou `c` podem ser usados.

Feitiço: `[xyz]`
👉 Aceita apenas "x", "y" ou "z".

Resposta: `y`
✅ O padrão é aceito.

Resultado: O feitiço acerta o Regexmon! Ele recua, rugindo ameaçadoramente, mas o treino continua.

DEFESA:

Regexmon se posiciona e diz:
"açaí bacaba abacaxi"

Explicação:
Mini me deve usar um feitiço de lista para aceitar apenas palavras que comecem com "a" ou "b".

Feitiço: `[ab]`
👉 Aceita apenas palavras que comecem com “a” ou “b”.

Resposta: `b`
✅ O padrão é aceito.

Resultado:
O feitiço acerta o Regexmon! Ele recua, bufando e resmungando:
Fala: “Argh… você está me desafiando demais!”

🎈2ª Batalha – Na Grama com o Regexmon (USO DE LISTA NEGADA)

ATAQUE:

Explicação:
A **lista negada** `[^...]` mostra quais caracteres **não são permitidos**. Ou seja, tudo o que estiver dentro dos colchetes será rejeitado.
Exemplo: `[^abc]` significa que qualquer caractere é aceito, **menos** “a”, “b” ou “c”.

Feitiço: `[^egmz]`
👉 Não aceita “e”, “g”, “m” ou “z”.

Resposta: `i`
✅ O padrão é aceito, pois “i” não está na lista proibida.

Resultado: O feitiço é quebrado.

DEFESA:

Ataque do Regexmon:
Frase lançada → 123789

Feitiço do Regexmon: `[^456]`
👉 Esse feitiço só aceita **caracteres que não sejam números**.

Defesa de Mini me:
Resposta: `[^456]`
Padrão encontrado na frase → `os números 4,5 e 6 não aparecem`

Resultado: O ataque é defendido com sucesso!

GINÁSIO 1 - QUADRA DA UNIFAP:

⚔️ Ginásio 1 – Batalhas com Representantes

Mini me encontra o seu rival no Ginásio e ele diz:

- "Veremos se você sabe mesmo usar os **representantes**. Prepare-se!"

🔮 Feitiço 1 – Uso do ponto (.)

ATAQUE:

Feitiço: `1.d.s`
👉 Aceita qualquer caractere, mas somente um entre: “1”, “d” e “s”.

Resposta: `1zd3s`
✅ O padrão é aceito, pois há somente um caractere, seja qual for, entre “1”, “d” e “s”

DEFESA:

Feitiço: `abc`, `a1c`, `a-c`
👉 Aceita apenas um caractere qualquer entre “a” e “c”.

Resposta: `a.c`
✅ O padrão é aceito.

Aceita: `abc`, `a1c`, `a-c`
Não aceita: `ac`, `abbc`, `adcde`

🔮 Feitiço 2 – Uso da lista negada [^...]

ATAQUE:

Feitiço: `[^abc]7[^xyz]`
👉 Explicação: `[^abc]` → qualquer caractere **exceto** a, b ou c / `7` → literal / `[^xyz]` → qualquer caractere **exceto** x, y ou z

Resposta: `d7k`, `m7p`, `z7a`
✅ O padrão é aceito, pois respeita as regras de não começar com “a”, “b” ou “c”, ter o 7 no meio e não terminar com “x”, “y” ou “z”

DEFESA:

Feitiço: `d1e4hg,d4e3ag`
👉 Aceita um caractere entre “d” e “e” que seja um número entre 1-5, aceita qualquer caractere que não seja o f e o literal g no final

Resposta: `d[1-5]e[^5-9][^f]g ou d[1-5]e[^5-9].g`
✅ O padrão é aceito.
Não aceita: `d123e4hg`,`d4e3g`…

Rival:
“Eu acreditava que minha lista era inquebrável… mas você a decifrou com facilidade. Tenho que admitir, sua habilidade é impressionante. Parabéns pela vitória.”

- Mini me segue em direção à saída, ainda sentindo a energia da batalha. Antes que pudesse deixar o ginásio, o treinador se aproxima.

Treinador:
“Excelente trabalho, Mini me! Você dominou o poder dos representantes e provou sua força. Como reconhecimento, receba agora os símbolos dos quantificadores. Eles vão ampliar ainda mais o alcance dos seus feitiços.”

QUANTIFICADORES

🥊 1° Encontro – ? (Opcional)

ATAQUE:

Regexmon: “Prepare-se, minha forma está escondida entre símbolos!”

Feitiço do Regexmon: `colou?r`
👉 A letra `u` é opcional.

Explicação: O `?` significa que o elemento pode aparecer **zero ou uma vez,** o caractere deve aparecer atrás do caractere que ele deseja ser opcional.

Resposta: `color` e `colour`

✅ Mini me defende o ataque entendendo que a letra `u` pode ou não estar presente.

DEFESA:

Feitiço do Regexmon: `"https://", “http://”`
👉 A letra `s` é opcional.

Resposta: `https?://`

✅ Mini me defende o ataque entendendo que a letra `s` pode ou não estar presente.

🥊 2° Encontro – * (Zero, um ou mais)

ATAQUE:

Fala: “HAHAHAH, Quero ver se você entende meu padrão!”

Explicação: O * permite que o elemento apareça **nenhuma, uma ou várias vezes**.

Feitiço: **h*a***

👉 A letra a e h permitem zero, uma ou várias repetições

Resposta: **ha, haha, haahaa,hahaha, hahahaha**

✅ Mini me defende com sucesso, permitindo todas as variações de h e a.

DEFESA:

Feitiço: **a, ab, abb, abbbc, abbcc, abcccc**

👉 A letra b e c permitem zero, uma ou várias repetições

Resposta: **ab*c***

✅ Mini me defende com sucesso, permitindo todas as variações de b e c.

🥊 3° Encontro – Quantificador + (Um ou mais)

ATAQUE:

Fala: “Você vai precisar mais do que sorte para decifrar isso!”

Explicação: O + exige que o elemento apareça **pelo menos uma vez**.

Feitiço: go+

👉 A letra o deve aparecer pelo menos uma vez.

Resposta: go, goo, gooo…

✅ **Não aceita só g**, pois precisa de pelo menos um o.

DEFESA:

Feitiço: sollllllllll, sooooooool, sooollll, sool, sol

👉 As letras “o”, “l” devem aparecer pelo menos uma vez.

Resposta: `so+l+`

✅ Mini me descobre que o feitiço só funciona se houver ao menos um "o" e ao menos um "l", em qualquer quantidade maior.

🥊 4° Encontro – Quantificador `{n,m}` (Faixa de repetições)

ATAQUE:

Fala: “Atenção: cada caractere conta, não erre!”

Explicação: As chaves determinam a quantidade mínima e máxima de vezes que o elemento pode aparecer.

Feitiço do Regexmon: `[0-9]{3,5}-[0-9]{1,3}`

👉 No início devem aparecer no mínimo 3 e máximo 5 números de 0-9, um - literal e 1 à 3 números de 0-9

Resposta: `12345-000, 67934-096…`

✅ Mini me defende o ataque, reconhecendo a faixa correta.

DEFESA:

Feitiço do Regexmon: AB123, A5, B45430, AB123456, 123

👉 No início devem aparecer ou não a letra “A” ou “B” e aparecer números de 1-9, sendo o mínimo 1 e máximo 6 números

Resposta: `A?B?[0-9]{1,6}`

✅ Mini me defende o ataque, reconhecendo o padrão da regex.

GINÁSIO 2 -RU:

🔮 Feitiço 1

Mini me encontra o seu rival no Ginásio e ele diz:

- "Parabéns por chegar até aqui, mas daqui você não irá passar, meus padrões podem se repetir infinitamente"

Feitiços do treinador:

ATAQUE

Feitiço: `c[iou]?p`

👉 O feitiço aceita zero ou uma vogal entre “i”, “o” ou “u”

Resposta: `cp, cip, cop, cup`

✅ Ataque defendido.

DEFESA:

Feitiço: `bat`, `bet`, `bit`, `bt`
👉 O feitiço aceita zero ou uma vogal entre “a”, “e” ou “i”

Resposta: `b[aei]?t`
✅ Ataque defendido.

🔮 Feitiço 2

ATAQUE:

Feitiço: `[zym]+[0-9]{2,4}`
👉 O feitiço aceita uma ou mais letras da lista [zym] e no mínimo 2 e máximo 4 números de 0-9.

Resposta: `zym1234, zzzz9832, ym5869`
✅ Ataque defendido.

DEFESA:

Feitiço: `xa, xya, yyzaa, zzzaa, xxyyzzzzzzaaa`
👉 O feitiço aceita xyz várias vezes e de 1-3 a vogal “a”.

Resposta: `[xyz]+[a]{1,3}`
✅ Ataque defendido.

🔮 Feitiço 3

ATAQUE:

Feitiço: `b[aei]{1,2}t[xyz]?`
👉 O feitiço aceita uma ou mais letras da lista [zym] e no mínimo 2 e máximo 4 números de 0-9.

Resposta: `baet, bat, beitz`
✅ Ataque defendido.

DEFESA:

Feitiço: **`son, soun, souun, sounk, soounm`**
👉 O feitiço aceita xyz várias vezes e de 1-3 a vogal “a”.

Resposta: `s[ou]{1,3}n[km]?`
✅ Ataque defendido.

Rival fala: “Ok, você venceu dessa vez… mas não vai ser sempre!”

- Mini me segue em direção à saída, ainda sentindo a energia da batalha. Antes que pudesse deixar o ginásio, o treinador se aproxima.

Treinador:

- "Parabéns, Mini me! Você provou seu domínio sobre os quantificadores, controlando repetições e padrões com maestria. Como recompensa, você agora desbloqueia os símbolos das âncoras. Prepare-se: os próximos desafios serão mais complexos, exigindo ainda mais atenção e habilidade para marcar com precisão o início e o fim de cada padrão!"

ÂNCORAS:

⚓ 1° Encontro

ATAQUE:

Fala: “Eu marco o **começo** e não deixo ninguém passar!”

Explicação: O símbolo `^` é uma **âncora** que marca o **início da linha ou da palavra**.

Feitiço: `^c`
👉 O feitiço exige que a palavra **comece com a letra “c”**.

Resposta: `carro, casa, cachorro`

✅ Por respeitar as regras, Mini me acerta o feitiço

DEFESA:

Feitiço: `gato, gelo, girafa, gorro, guia`
👉 O feitiço exige que a palavra **comece com a letra “c”**.

Resposta: `^g[aeiou].+`

✅ Por respeitar as regras, Mini me acerta o feitiço

⚓ 2° Encontro

ATAQUE:

Fala: “Você chegou ao fim… esse é o seu destino!”

Explicação: O símbolo `$` é uma âncora que marca o **fim da linha ou da palavra.**

Feitiço: `a$`
👉 O feitiço exige que a palavra termine com a letra “a”.

Resposta: `rosa, roda, bola, água, jarra`

✅ Por respeitar as regras, Mini me acerta o feitiço

DEFESA:

Feitiço: `pele, doce, forte, filme`
👉 O feitiço exige que a palavra termine com a letra “e”.

Resposta: `.+e$`

✅ Por respeitar as regras, Mini me acerta o feitiço

⚓ 3° Encontro

ATAQUE:

Fala: “Meu feitiço é simples: começo aqui, termino ali… e você no meio cai!”

Explicação: O `\b` representa uma fronteira de palavra. Ele marca o espaço entre um caractere de palavra (`\w`: letras, números ou `_`) e um caractere que não seja de palavra (como espaço, pontuação ou o início/fim da linha).

Feitiço: \bmar\b
👉 Esse feitiço só aceita **a palavra isolada "mar"**, e não palavras que a contenham dentro (como *remar* ou *submarino*).

Resposta: `mar`

✅ Por respeitar as regras, Mini me acerta o feitiço

DEFESA:

Feitiço: sol, sal, sul, selo
👉 O `\b` garante que a palavra esteja isolada. Dentro da lista `[ao]`, o feitiço aceita tanto sal quanto sol, mas não “selo” ou “sul”.

Resposta: `\b s[ao]l \b`

✅ Por respeitar as regras, Mini me acerta o feitiço

GINÁSIO 3 - PRÉDIO:

Mini me entra no ginásio e o seu rival diz:

- "Ah… você chegou até aqui, Mini me. Vejo que aprendeu bastante, mas não se engane, eu controlo o início e o fim de tudo — nada começa sem minha aprovação, e nada termina sem minha marca! Se você tentar escapar pelo meio, encontrará apenas armadilhas. Prepare-se, porque o campo de batalha é meu, e o começo e o fim decidirão seu destino!"

🔹 Batalha 1

ATAQUE:

Feitiço: ^pro[a-z]*

👉 O `^` exige que a frase **comece com "pro"** e `[a-z]*` aceita zero ou mais letras minúsculas depois.

Resposta: `projeto, progresso…`

✅ Mini me acerta o padrão e quebra o feitiço

DEFESA:

Feitiço: 123fim, 9fim, 2025fim

👉 A regra `[0-9]+` exige **um ou mais dígitos numéricos** e `fim$` garante que a palavra **termine exatamente em "fim"**.

Resposta: `[0-9]+fim$`

✅ Mini me entende o padrão e a exigência da regex e consegue se defender.

🔹 Batalha 2

ATAQUE:

Feitiço: **\b[a-z]{3}\b**

👉 O `\b` garante que a correspondência seja uma palavra isolada e `[a-z]{3}` exige **exatamente 3 letras minúsculas**.

Resposta: `lua, mar, sol, paz`

✅ Mini me acerta o padrão e quebra o feitiço.

DEFESA:

Feitiço: pa, ir, pé, ar

👉 O `\b` garante que a correspondência seja uma palavra isolada e `[a-z]{2}` exige **exatamente 2 letras minúsculas**.

Resposta: \b[a-z]{2}\b

✅ Mini me entende o padrão e a exigência da regex e consegue se defender.

🔹 Batalha 3

ATAQUE:

Feitiço: \b[a-z]*ing\b

👉 A regra `\b` delimita a palavra, `[a-z]*` aceita zero ou mais letras antes e `ing` exige que a palavra termine em "ing".

Resposta: coding, testing, learning

✅ Mini acerta as palavras aceitas e quebra o feitiço.

DEFESA:

Feitiço: bc123ion, ql89ion, dp890ion

👉 O `\b` garante que seja uma palavra completa, `[^aeiou]{2,3}` começa com **2 a 3 consoantes seguidas** (nenhuma vogal), `[0-9]+` determina **um ou mais números**, o `ion` determina que a a palavra deve terminar com **"ion"** e `\b` fecha garantindo o fim da palavra.

Resposta: \b[^aeiou]{2,3}[0-9]+ion\b

✅ Mini me entende o padrão e a exigência da regex e consegue se defender.

Rival:
“Naaaaão, minhas âncoras… foram arrancadas pelo seu poder! Não consigo acreditar! Você realmente é muito esperta, infelizmente tenho que dizer: parabéns!

- Mini me segue em direção à saída, animada por conquistar mais um ginásio. Já chegando no portão, ela aguarda o treinador para lhe dar os parabéns.
- **Treinador:**
  "Parabéns, Mini me! Você conquistou todos os ginásios e dominou cada desafio com coragem e sabedoria. Agora, chegou o momento de encarar o desafio final. Sua

batalha final reunirá todo o poder dos padrões que você enfrentou até aqui. Vá em frente e dê o seu melhor… e que a sorte esteja ao seu lado nessa batalha decisiva!"

Ao caminhar para a Batalha Final, os avós de Mini me aparecem:

- Vô Chico: Parabéns minha filha, você já tem nos dado muito orgulho! Você enfrentou todos os desafios e conseguiu vencer com sabedoria! Nosso legado está vivo em você!
- Vó Rita: Minha filha, como você nos surpreende, em pouco tempo aprendeu tudo! Parabéns por suas conquistas! Agora vá em frente e vença sua batalha final e garanta a proteção desse lugar. Acreditamos em você!

⚔️ BATALHA FINAL - Bloco de Medicina:

Serão 3 feitiços utilizando tudo o que foi aprendido

- Mini me chega no Ginásio Final e ao entrar se depara com o vilão que controlava os regexmon

Vilão: “Então você finalmente chegou… achei que nunca chegaria até aqui. Preciso lhe informar que todos os ginásios foram apenas um aquecimento… agora enfrentará meu verdadeiro poder! Eu domino cada padrão, cada repetição… você está pronta para falhar? Seus queridos avôs falharam, pelo visto seu legado é esse, falhar… Então vamos logo acabar com isso!”

🛡️Desafio 1

ATAQUE:

Feitiço: `^([a-z]{1,2}[0-9]+){2,3}$`

👉 Toda a linha deve ter **1-2 letras seguidas de 1 ou mais números**, repetido 2-3 vezes.

Resposta: `ab11cd34de11, aa0987zz1234, …`

✅ Mini entende o padrão da regra e quebra o feitiço.

DEFESA:

Feitiço: `54675-000, 81340-079, 45455-003`

👉 O início da regex deve ter 5 números entre 0-9, um ‘-’ literal e no fim 3 números de 0-9.

Resposta: `^[0-9]{5}-[0-9]{3}$`

✅ Mini me compreende o padrão do início e fim da regex e consegue se defender.

🛡️Desafio 2

ATAQUE:

Feitiço: ^[a-z]{2,4}(-[a-z]{2,}){2,4}$

👉 A regex aceita palavras separadas com o ‘hífen’, a primeira com entre 2-4 letras de a-z, a 2° deve ter no mínimo 2 letras de a-z sem máximo e esse padrão deve repetir no mínimo de 2 até 4 vezes para ser aceito.

Resposta: `mago-poder-vitoria, mar-chuva-ar-vento`

✅ Mini me compreende o padrão do início e fim da regex e consegue se defender.

DEFESA:

Feitiço: **`teste00@email.com, abcdefghi9876@email.com, az44@email.com`**

👉 O início da regex deve ter no mínimo 2 e máximo 9 letras de a-z, em seguida números de 0-9, no mínimo 2 e máximo 4 e ‘@email.com’ no final.

Resposta: `^[a-z]{2,9}[0-9]{2,4}@email.com`

✅ Mini me compreende o padrão e quebra o feitiço.

🛡️Desafio 3

ATAQUE:

Feitiço: ^([0-9][a-z]{1,2}[0-9]{2,3}){2}$

👉 A regex aceita um número de 0-9 no ínicio, 2 letras de a-z e no fim de 2-3 números de 0-9 e esse padrão deve se repetir 2 vezes

Resposta: `1aa1111aa111, 9z116d33`

✅ Mini me compreende o padrão do início e fim da regex e consegue se defender.

DEFESA:

Feitiço: **`041.234.885-00, 180.778.253-08, 140.594.925-06`**

👉 O início da regex deve ter no mínimo 3 seguida de ponto literal, em seguida de 3 números, um ponto literal de novo, o hífen literal e 2 dígitos no final.

Resposta: `\b[0-9]{3}.[0-9]{3}.[0-9]{3}-[0-9]{2}\b`

✅ Mini me compreende o padrão e quebra o feitiço.

🏆VITÓRIA DA MINI ME:

Vilão:

- "Nãaaao… como você conseguiu me derrotar? Eu dominava cada padrão, cada linha, cada repetição… nada deveria ter me alcançado! Você… você realmente dominou tudo que aprendeu… Isso não pode estar acontecendo…! Você pode me ter ganhado hoje, mas quem sabe no futuro?
    - Então o vilão desaparece

TEXTO FINAL:

"E assim, Mini me alcança o fim de sua jornada no mundo do Regex, carregando com bravura o legado da sua família. Com garra, determinação e coragem, ela protegeu o campus e mostrou que o esforço de seus avós nunca será esquecido. Agora, em paz, eles podem descansar tranquilos, sabendo que o legado que construíram continua vivo e protegido."

- *Cena final Mini me abraçando seus avôs.*