INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO AMAZONAS
CAMPUS MANAUS CENTRO

LEONARDO HENRIQUE LIMA DE SOUZA

MY FRIEND LOGIC-X: UMA APLICAÇÃO GAMIFICADA PARA AUXÍLIO DO ENSINO DE LÓGICA DE PROGRAMAÇÃO PARA PESSOAS COM DISLEXIA

MANAUS - AM
2023

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO AMAZONAS
CAMPUS MANAUS CENTRO

LEONARDO HENRIQUE LIMA DE SOUZA

MY FRIEND LOGIC-X: UMA APLICAÇÃO GAMIFICADA PARA AUXÍLIO DO ENSINO DE LÓGICA DE PROGRAMAÇÃO PARA PESSOAS COM DISLEXIA

Trabalho de Conclusão de Curso apresentado à banca examinadora Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciências e Tecnologia do Amazonas – IFAM Campus Manaus Centro, como requisito para o cumprimento da disciplina TCC II – Projeto de Software.

Orientador: Prof. Dr. Renildo Viana Azevedo

MANAUS - AM
2023

Biblioteca do IFAM – Campus Manaus Centro

S729m Souza, Leonardo Henrique Lima de.
MY FRIEND LOGIC-X: uma aplicação gamificada para auxílio do ensino de lógica de programação para pessoas com dislexia / Leonardo Henrique Lima de Souza. – Manaus, 2023.
138 p. : il. color.

Monografia (Tecnologia em Análise e Desenvolvimento de Sistemas). – Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, *Campus* Manaus Centro, 2023.
Orientador: Prof. Dr. Renildo Viana Azevedo.

1. Desenvolvimento de sistemas. 2. Aprendizagem – lógica de programação. 3. Gamificação. I. Azevedo, Renildo Viana. (Orient.) II. Instituto Federal de Educação, Ciência e Tecnologia do Amazonas. III. Título.

CDD 005.3

Elaborada por Márcia Auzier CRB 11/597

LEONARDO HENRIQUE LIMA DE SOUZA

MY FRIEND LOGIC-X: UMA APLICAÇÃO GAMIFICADA PARA AUXÍLIO DO ENSINO DE LÓGICA DE PROGRAMAÇÃO PARA PESSOAS COM DISLEXIA

Trabalho de Conclusão de Curso apresentado à banca examinadora Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciências e Tecnologia do Amazonas – IFAM Campus Manaus - Centro, como requisito para o cumprimento da disciplina TCC II – Projeto de Software.
Orientador: Prof. Dr. Renildo Viana Azevedo

Aprovado em _____ de _____ de 2023

BANCA EXAMINADORA

Prof. Dr. Renildo Viana Azevedo (Orientador)
Instituto Federal do Amazonas

Prof.ª Dra. Viviane Gomes da Silva
Instituto Federal do Amazonas

Prof.ª MSc. Mirlem Rodrigues Ribeiro Pereira
Instituto Federal do Amazonas

SUMÁRIO

INTRODUÇÃO 8
PROBLEMATIZAÇÃO 8
JUSTIFICATIVA 9
OBJETIVOS 11
Objetivo geral 11
Objetivos específicos 11
METODOLOGIA E PROCESSO UTILIZADO 11
CAPÍTULO 1: MERGULHANDO NAS TEORIAS E PRÁTICAS APLICADAS 13
1.1. DISLEXIA, GAMIFICAÇÃO E APRENDIZAGEM DE LÓGICA DE PROGRAMAÇÃO 13
1.2. TÉCNICAS APLICADAS À SOLUÇÃO 17
1.2.1. Aprendizagem baseada em problemas 17
1.2.2. Feedback 18
1.2.3. Elementos audiovisuais 18
1.3. MODELO DE APLICAÇÃO 19
1.3.1. Aplicação educacional 19
1.3.2. Código aberto 20
CAPÍTULO 2: EXPLORANDO AS FERRAMENTAS UTILIZADAS 21
2.1. PROCESSO DE DESENVOLVIMENTO DE JOGOS 22
2.1.1. Concepção e planejamento 22
2.1.2. Fase de pré-produção 23
2.1.3. Fase de produção 24
2.1.4. Fase de pós-produção 25
2.2. GAMIFICAÇÃO 27
2.3. APRENDIZADO BASEADO EM PROBLEMAS (ABP) 30
2.4. AUDIOVISUAL 32
2.5. UNITY 35
2.6. TEXT-TO-SPEECH 40
CAPÍTULO 3: JORNADA DE CRIAÇÃO DO MY FRIEND LOGIC-X 44
3.1. CONCEPÇÃO DA IDEIA 44
3.2. PLANEJAMENTO DA APLICAÇÃO 45
3.3. DESIGN DA APLICAÇÃO 47

3.4. PRODUÇÃO E DESENVOLVIMENTO DA APLICAÇÃO 54
3.4.1. Telas da aplicação 54
3.4.2. Produção de roteiro textual da aplicação 63
3.4.3. Produção de áudio 68
3.4.4. Produção de vídeos 73
3.4.5. Desenvolvimento e implementação de scripts 77
CONSIDERAÇÕES FINAIS 89
REFERÊNCIAS 92
APÊNDICE 95
APÊNDICE A - Scripts desenvolvidos para a aplicação MyFriendLogic-X 95
1. ConfigurationData.cs 95
2. DataAllStructure.cs 95
3. FontTextSize.cs 95
4. InteractiveList.cs 98
5. LoadConfigurationData.cs 104
6. MenuScript.cs 106
7. PauseGame.cs 107
8. SaveStageData.cs 108
9. SceneCall.cs 109
11. SceneTimer.cs 110
12. StageInitialConfiguration.cs 111
13. TopicDataLoader.cs 112
14. TopicDataStruct.cs 115
15. TopicProgressDataLoader.cs 115
16. TransitionScript.cs 117
17. VideoPlayerStatus.cs 119
18. VoiceAudioPlay.cs 119
APÊNDICE B - DOCUMENTO DE DESIGN DE JOGO DA APLICAÇÃO MY FRIEND LOGIC-X 120
Visão Geral 120
Descrição Geral do Projeto 121
Mecânicas de Jogabilidade 121
História e Jogabilidade 122
1. Estrutura da sequência 122

2. Controle de Fluxo 126
3. Tipos de Dados 130
4. Operadores 133
5. Funções 135

RESUMO

Este trabalho discute a relação entre as dificuldades enfrentadas por pessoas com dislexia no ensino de lógica de programação e o uso da gamificação como uma abordagem para auxiliar no ensino. A dislexia é um distúrbio de aprendizagem e linguagem que afeta crianças e adultos em diferentes momentos da vida. A lógica de programação consiste em técnicas utilizadas no desenvolvimento de software. A gamificação é uma abordagem que incorpora elementos de jogos em contextos que não são jogos propriamente ditos. O objetivo central deste trabalho é desenvolver uma aplicação gamificada para auxiliar no ensino de lógica de programação para pessoas com dislexia. A aplicação utilizará técnicas visuais e sonoras, proporcionando uma experiência interativa e lúdica que facilite o aprendizado.

Palavras-chave: dislexia, lógica de programação, gamificação, ensino, aplicação.

ABSTRACT

This work discusses the relationship between the difficulties faced by individuals with dyslexia in learning programming logic and the use of gamification as an approach to assist in teaching. Dyslexia is a learning and language disorder that affects children and adults at different stages of life. Programming logic comprises techniques used in software development. Gamification is an approach that incorporates game elements into non-game contexts. The main objective of this work is to develop a gamified application to support the teaching of programming logic for individuals with dyslexia. The application will employ visual and auditory techniques, providing an interactive and playful experience that facilitates learning.

Keywords: dyslexia, programming logic, gamification, teaching, application.

INTRODUÇÃO

O presente trabalho tem como foco o estudo de uma solução para o auxílio do ensino de lógica de programação para pessoas com dislexia. A dislexia é um distúrbio de aprendizagem e da linguagem, que afeta desde crianças até adultos. A dislexia acompanha a pessoa por toda a vida, e para isso é necessário intervenções e abordagens que possam auxiliar na qualidade de vida da pessoa.

A lógica de programação é um conjunto de técnicas para o desenvolvimento de *software*, em que a mesma é abordada como disciplina em cursos de graduação, técnico ou cursos livres de programação. Neste trabalho, são apontados estudos que mostram as dificuldades enfrentadas pelos alunos de lógica de programação, especialmente na interpretação textual e abstração de problemas do mundo real, em que a aplicação das habilidades de lógica de programação é essencial. Essas dificuldades podem ser ainda mais desafiadoras para pessoas com dislexia, caso não haja uma abordagem adequada para auxiliá-las.

O trabalho tem como proposta de solução o desenvolvimento de uma aplicação com uso de gamificação para auxiliar o ensino de lógica de programação para pessoas com dislexia. A aplicação será um protótipo desenvolvido para auxiliar no ensino dos tópicos iniciais dos conhecimentos em lógica de programação, utilizando técnicas de gamificação como recompensas, desafios progressivos e *feedback*, além das técnicas de ensino para pessoas com dislexia baseadas em interação visual e sonora para atender o público alvo de estudantes que possuem ou não diagnóstico confirmado de dislexia.

De tal forma, para garantir a qualidade da solução proposta, será utilizada metodologia ágil de desenvolvimento de *software*, ferramentas adequadas para a implementação da aplicação, além de testes constantes e *feedback* dos usuários. Assim, o foco geral é desenvolver uma ferramenta que não apenas ajude no ensino de lógica de programação, mas também auxilie utilizando gamificação, na melhora da experiência de aprendizagem de pessoas com dislexia.

PROBLEMATIZAÇÃO

A presença da dislexia e as dificuldades enfrentadas no aprendizado podem ser fatores determinantes para a evasão ou abandono dos cursos de computação,

uma vez que os estudantes com esse transtorno podem enfrentar dificuldades significativas em disciplinas como a lógica de programação. Diante disso, surge a necessidade de uma aplicação que possa auxiliar esses alunos a superar essas dificuldades, tornando o processo de aprendizagem acessível e inclusivo.

Diante de uma realidade em que se verifica a dificuldade de aprendizado de lógica de programação, uma aplicação baseada em gamificação pode ser eficaz ao ensino de lógica de programação para estudantes com dislexia? Esta é a questão fundamental que este trabalho pretende responder.

JUSTIFICATIVA

Diante do crescente avanço tecnológico, o aprendizado de lógica de programação se torna fundamental para a formação de qualquer profissional na área de computação e afins. No entanto, muitos estudantes enfrentam dificuldades nesse processo, sejam elas originadas não só por metodologias de ensino mas como também por questões neurobiológicas, como a dislexia. De tal forma, surge a necessidade de uma aplicação que possa auxiliar no processo de ensino e aprendizagem da lógica de programação para pessoas que possuem dislexia, independente de qual fase da vida ela se encontra. Que de forma interativa e lúdica auxilie-os a aprender com técnicas visuais e sonoras a passagem do conhecimento.

A dislexia do desenvolvimento é um transtorno específico de aprendizagem que pode afetar o reconhecimento preciso e fluente das palavras, a habilidade de decodificação e a soletração, dificultando a compreensão de textos e, consequentemente, a aprendizagem de disciplinas como a lógica de programação.

Segundo a Associação Brasileira de Dislexia(2016), a dislexia é um transtorno específico de aprendizagem de origem neurobiológica. É caracterizada por dificuldades no reconhecimento preciso e/ou fluente das palavras, habilidade de decodificação e soletração. Essas dificuldades geralmente são causadas por um déficit no componente fonológico da linguagem e são inesperadas em relação à idade e outras habilidades cognitivas. De acordo com o Instituto ABCD(2021)[1], a dislexia afeta cerca de 10% da população mundial, onde a mesma calcula que

[1] INSTITUTO ABCD. **Página de perguntas e respostas**, c2021. Disponível em: <https://www.institutoabcd.org.br/perguntas-e-respostas/>. Acesso em: 08 de Novembro de 2023.

possam haver mais de 700 milhões de pessoas disléxicas no mundo. Enquanto no Brasil, estima-se que 4% da população possua dislexia, o que segundo o mesmo instituto, isso representa mais de 8 milhões de pessoas.

Diante desse entendimento, a dislexia pode afetar pessoas em qualquer momento da vida, na infância, adolescência e fase adulta, assim, levando a dificuldades de entendimento, leitura, expressão escrita e a aprendizagem de disciplinas como a lógica de programação, o que pode levar a mais dificuldades no aprendizado de novas habilidades. É de suma importância a investigação do tema não apenas para contribuir a minimizar lacunas e dificuldades enfrentadas por pessoas que possuem dislexia e que estudam a área tecnológica, bem como, ajudar a promover uma melhor qualidade no ensino da lógica de programação.

Com esse contexto, o uso da gamificação para desenvolver um aplicativo que possa auxiliar no ensino de lógica de programação pode trazer diversos benefícios. A utilização da gamificação, que segundo Raguze *et al*.(2016), são técnicas utilizadas em jogos e adaptadas para um contexto de não jogo, podem ajudar com estratégias visuais, sonoras e mecânicas lúdicas de recompensas, o ensino e aprendizado de lógica de programação.

As primeiras manifestações concretas da gamificação se deram em 2010, mas o conceito já é considerado como uma estratégia de engajamento e motivação em auxílio ao processo de aprendizagem em diversos setores do conhecimento humano, características estas encontradas nos jogadores quando em interação com os games (ORLANDI *et al*., 2018).

De acordo com Navarro(2013, *apud* ORLANDI *et al*., 2018), na gamificação, o jogo deixa de ser apenas uma distração e passa a ter seu conceito redefinido, assumindo um novo papel e importância na sociedade. Isso ocorre porque ele exerce influência no desenvolvimento sensorial, psicomotor e cognitivo do indivíduo. Nesse contexto, é necessário repensar o papel exclusivo de distração dos jogos.

Com isso, para ajudar a minimizar essas lacunas e promover um melhor ambiente educativo, a solução proposta neste trabalho tem como público-alvo pessoas que possuem dislexia e enfrentam dificuldades no aprendizado de lógica de programação. Este trabalho propõe o desenvolvimento de uma aplicação que utiliza a gamificação para auxiliar no ensino de lógica de programação para pessoas que possuem dislexia. Espera-se que a solução proposta neste trabalho possa contribuir para a melhoria da qualidade no ensino de lógica de programação, permitindo que

pessoas com dislexia possam adquirir conhecimentos sólidos e de qualidade, beneficiando-se de habilidades complexas posteriormente.

OBJETIVOS

Objetivo geral

Desenvolver uma aplicação com uso de gamificação para auxiliar no ensino de lógica de programação para pessoas com dislexia, utilizando técnicas visuais e sonoras que possam ajudar a promover o aprendizado de forma interativa e lúdica.

Objetivos específicos

- Identificar as principais dificuldades no aprendizado encontradas por pessoas com dislexia.
- Desenvolver uma aplicação com técnicas de gamificação para o ensino dos tópicos básicos de lógica de programação utilizando técnicas visuais e sonoras.

METODOLOGIA E PROCESSO UTILIZADO

Neste tópico é descrita a metodologia utilizada para o desenvolvimento deste trabalho e a solução proposta. Visando alcançar os objetivos gerais e específicos, a metodologia foi elaborada em três etapas: Pesquisa, Desenvolvimento e Avaliação.

A etapa de pesquisa tem como objetivo entender melhor as técnicas, abordagens, dificuldades em relação ao tema proposto, além de para o desenvolvimento da aplicação, de tal forma, foram realizados levantamentos bibliográficos utilizando a base de dados *Google Scholar*, além de pesquisas sistemáticas por artigos científicos que abordassem cada tópico específico do tema proposto neste trabalho. O método de pesquisa PICO(*Population, Intervention, Comparison e Outcome*), que segundo Santos *et al.*(2007), consiste na aplicação de práticas baseadas em evidências que envolve utilizar a mais solida base cientifica para embasar as decisões, foi utilizado para a busca de trabalhos relacionados com

tema, com o objetivo de identificar técnicas em conjunto para trabalhar com pessoas com dislexia utilizando a gamificação, além do que identificar *softwares* similares propostos nesses trabalhos.

Para o processo da etapa de desenvolvimento, será utilizado metodologias ágeis para auxiliar na construção da aplicação proposta, além de contar com a participação de pessoas que possuam ou não diagnóstico de dislexia a partir de institutos especializados em dislexia. A aplicação contará com técnicas de gamificação para tornar o processo de auxílio do ensino adequado ao contexto de aprendizado para pessoas com dislexia, com estratégias de ensino baseadas em interação visual e sonora.

Este processo de desenvolvimento será realizado com entregas parciais da aplicação para testes preliminares e avaliação dos usuários finais. Isso irá permitir maior flexibilidade e adaptabilidade durante o processo de desenvolvimento da aplicação, podendo garantir que a solução proposta atenda às necessidades dos usuários finais e seja efetiva no auxílio ao ensino de lógica de programação para pessoas com dislexia.

Para a etapa de avaliação, serão realizados testes da aplicação com um grupo de estudantes de cursos de computação que possuem ou não diagnóstico de dislexia, em que os mesmos serão responsáveis por avaliar a experiência de uso. Também será coletado dados quantitativos e qualitativos sobre o tempo de aprendizado, taxa de sucesso das atividades propostas na aplicação e o *feedback* dos usuários. Tais dados serão utilizados para concluir a efetividade da aplicação no auxílio ao ensino de lógica de programação para pessoas com dislexia.

CAPÍTULO 1: MERGULHANDO NAS TEORIAS E PRÁTICAS APLICADAS

Neste capítulo é feita uma discussão integrada sobre como a abordagem de gamificação pode ser aplicada e benéfica em relação para o auxílio de lógica de programação em relação a pessoas com dislexia, no mesmo, é apresentada as abordagens e técnicas utilizadas como a abordagem de aprendizado baseada em problemas e técnicas audiovisuais.

1.1. DISLEXIA, GAMIFICAÇÃO E APRENDIZAGEM DE LÓGICA DE PROGRAMAÇÃO

A dislexia é um distúrbio de aprendizagem que afeta muitas pessoas durante seu desenvolvimento acadêmico. A Associação Brasileira de Dislexia(2016) reconhece e classifica a dislexia como um transtorno específico de aprendizagem de origem neurobiológica. Manifesta-se através de dificuldades no reconhecimento preciso e/ou fluente das palavras, bem como na habilidade de decodificação e soletração. Essas dificuldades normalmente resultam de um déficit no componente fonológico da linguagem e são surpreendentes em relação à idade e outras habilidades cognitivas do indivíduo.

Segundo Silva[2] *et al.*(2016, *apud* Moura, 2013), os disléxicos possuem um funcionamento cerebral que difere em relação à recepção de informações, o que indica que seus cérebros são normais. No entanto, essas informações são processadas de forma inadequada devido a falhas nas conexões cerebrais. Consequentemente, enfrentam dificuldades no aprendizado da leitura, escrita e ortografia, pois têm dificuldade em assimilar as palavras devido a essas falhas no processo de leitura.

Dessa forma, uma pessoa com dislexia pode ter seu aprendizado prejudicado, já que todo o conteúdo é absorvido por meio da leitura através de textos. Como resultado, é comum que a pessoa com dislexia tenha dificuldades em assimilar palavras ou textos parecidos, o que pode gerar confusão e afetar seu aprendizado.

---

[2] SILVA, Nilza Sebastiana da. SILVA, Fabio José Antônio da. **A dislexia e a dificuldade na aprendizagem.** 2016.

De acordo com Montanari(2015, *apud* Rotta e Pedroso, 2006), frequentemente, os alunos começam a enfrentar dificuldades já no primeiro ano escolar. No entanto, se a escola não reconhecer essas dificuldades como algo preocupante, somente a partir do terceiro ano escolar é que elas começarão a ser percebidas, uma vez que nesse período há um aumento nas exigências em relação ao desempenho acadêmico.

Assunção(2018) afirma que se um indivíduo disléxico não recebe assistência profissional adequada, ele continuará enfrentando dificuldades na leitura e escrita. Contrariando o que se poderia supor, o problema não se agravará, mas, quando não tratado corretamente, pode resultar em consequências graves. Algumas características observadas incluem desempenho acadêmico deficiente, falta de noção do tempo, melhor desempenho em testes orais do que escritos e um sentimento de inferioridade.

De acordo com Montanari(2015), durante a leitura, os indivíduos disléxicos enfrentam dificuldades em compreender o conteúdo, encontrando obstáculos ao agrupar palavras e ao ler com pausas adequadas utilizando pontuação, além de ocorrerem inversões de sílabas. Consequentemente, a leitura se torna um processo lento, laborioso e individual para essas pessoas, prejudicando sua capacidade de compreender o texto, mesmo que tenham uma habilidade adequada de compreensão da língua falada.

Por fim, para trabalhar de forma efetiva com pessoas que possuem esse distúrbio, Montanari(2015, *apud*. Prado, 2010) afirma que deve ser feito um trabalho de forma integrada e contextualizada, envolvendo aspectos como: linguagem, raciocínio, concentração, percepção, esquemas corporal, orientação espacial, temporal e lateralidade. Mostra também que outra atividade que pode ser trabalhada é a estimulação auditiva e visual, visando trabalhar fonema-grafema, para este tipo de atividade.

Uma abordagem que pode ser empregada no trabalho com pessoas que possuem dislexia é a gamificação, uma técnica originada dos jogos eletrônicos. A gamificação traz consigo características dos jogos para contextos não relacionados a jogos, permitindo transmitir conhecimento de maneira lúdica e interativa.

Segundo Navarro(2013), a sobrevivência em si pode ser vista como uma forma de jogar com a vida, o que significa que a ideia de gamificação não é exatamente nova na sociedade. Para Raguze *et al*.(2016), a gamificação consiste

em utilizar elementos eficazes dos jogos, como estética, dinâmicas e mecânicas, com o objetivo de obter os mesmos benefícios proporcionados pela atividade de jogar. As mecânicas presentes nos jogos funcionam como mecanismos motivacionais para os indivíduos, contribuindo para o engajamento deles em diversos ambientes e aspectos.

Faria(2021, *apud* Busarello, 2014) afirma que, é possível aplicar a gamificação em atividades que requerem o estímulo do comportamento do indivíduo. O mesmo autor(2021, *apud* Alves, 2015) aponta que, em termos de aprendizagem, um dos maiores benefícios é o fato de que os jogos diminuem sensivelmente o tempo necessário para o aprendizado de um conceito.

Assim, trazendo para o contexto educacional, a gamificação pode ser uma ferramenta valiosa para o processo de aprendizagem, especialmente para pessoas com dislexia. Isso porque os elementos de jogos, como recompensas, desafios e *feedback* imediato, podem ajudar a manter o engajamento e a motivação dos alunos.

Ao utilizar a gamificação para ensinar habilidades e conceitos, os educadores podem tornar o processo de aprendizagem mais atraente e interativo. Faria(2021) afirma que a gamificação tem o poder de mudar o psicológico de uma pessoa, pois pode fazer com que entre no estado psicológico chamado de *flow*, onde é quando uma pessoa se sente totalmente engajada e concentrada na realização de uma determinada tarefa como foco para alcançar as suas metas.

Faria(2021) ainda afirma que a gamificação é um método viável para contornar o medo do erro, pelo fato de que ela trabalha como uma camuflagem. Assim permitindo que os alunos aprendam a partir de erros, sem sofrerem as consequências negativas da vida real. Isso pode ajudar a reduzir a ansiedade e o estresse associados ao aprendizado.

Raguze *et al*.(2016) mostra que, para a aplicação da gamificação ser efetiva, os aspectos aplicados devem representar algum significado para o jogador. Tanto no aspecto da perspectiva social ou no desafio de adquirir estes elementos, eles devem representar valor. O mesmo autor(2016) também afirma que, em um contexto educacional, os aspectos dos jogos são significativos para a aprendizagem. Aspectos como a repetição de experimentos, ciclos rápidos de resposta, níveis de dificuldade crescente, possibilidades de caminhos e recompensas, desempenham importantes papéis na aprendizagem.

Por fim, Raguze *et al*.(2016) aponta que, o engajamento do usuário é o objetivo central da gamificação. Para aplicá-la de forma adequada, é necessário ter conhecimentos específicos. Uma análise criteriosa permite identificar quais mecânicas são mais adequadas para atingir os objetivos do processo, levando em conta os perfis dos usuários identificados. Onde o processo de design é fundamental para o sucesso da gamificação. É essencial compreender as mecânicas dos jogos e as relações que os usuários terão com elas ao criar um processo de gamificação. Esses conhecimentos são requisitos primordiais a serem considerados durante o desenvolvimento do design da aplicação.

Levando em consideração os benefícios da gamificação no ensino, é uma abordagem eficaz utilizar essa técnica para transmitir o conhecimento de lógica de programação para pessoas com dislexia. O conteúdo da lógica de programação é predominantemente textual, tornando importante que os conceitos sejam transmitidos de maneira clara e sólida, permitindo que os indivíduos desenvolvam habilidades sólidas para futuros aprendizados.

Lógica de programação é a base para a criação de programas computacionais. De acordo com Leal(2014), a habilidade de pensar é um requisito fundamental para que os estudantes tenham sucesso ao aprender programação de computadores. A capacidade de pensar envolve o desenvolvimento do raciocínio e a habilidade de organizar o discurso de forma lógica, aplicando critérios como a busca de razões convincentes, inferências fundamentadas e a capacidade de apresentar explicações, descrições e argumentos coerentes.

Diante disso, as pessoas que não tiveram a oportunidade de desenvolver tais habilidades lógicas podem recorrer a diversas dificuldades ao aprender lógica de programação. Leal(2014) ainda afirma que é fundamental que os alunos desenvolvam a capacidade de abstração para resolver problemas e tenham sucesso nessa área. A abstração é um conceito importante na ciência da computação e é considerada um recurso essencial no processo de desenvolvimento de *software*.

Com isso, uma pessoa com dislexia, pode enfrentar dificuldades ao entrar em um curso de computação e tecnologias ou iniciar os estudos em programação. Isso ocorre porque na computação, o desenvolvimento de *software* é feito por meio da escrita de código, o que pode gerar dificuldades de assimilação de palavras, abstração, lógica, técnicas e compreensão dos termos técnicos referentes.

Na análise de Silva *et al*.(2019) a respeito das dificuldades no ensino de lógica de programação, mostra que 85,7% professores que participaram do estudo, apontaram que seus alunos tinham o desenvolvimento insuficiente da capacidade de leitura e interpretação de textos, ao lado das dificuldades de raciocínio lógico pouco desenvolvido e a baixa capacidade de abstração que foi apontada por 71,4% dos docentes. Por consequência, uma pessoa que possui dislexia e que está iniciando os estudos em lógica de programação pode enfrentar desafios significativos para aprender e desenvolver suas habilidades.

Dessa forma, Santos *et al*.(2014) mostra que o uso de jogos eletrônicos pedagógicos, têm se mostrado eficientes e eficazes no tratamento de disfunções motoras, sensoriais e perceptivas. A informática desempenha um papel valioso na construção das habilidades cognitivas, perceptivas e emocionais dos indivíduos disléxicos. O uso da informática estimula a percepção ao combinar imagens e textos, promove a orientação espaço-temporal e o controle dos movimentos. Além disso, a informática pode estimular a cognição por meio da capacidade de representação, simbolismo, resolução de problemas, imaginação, criatividade, leitura e escrita.

1.2. TÉCNICAS APLICADAS À SOLUÇÃO

Nesta seção serão descritas as técnicas utilizadas ao lado da gamificação para compor a solução proposta, com o objetivo de melhor atender e auxiliar no problema apresentado neste trabalho.

1.2.1. Aprendizagem baseada em problemas

Segundo Borochovicius *et al*.(2014), a Aprendizagem Baseada em Problemas(ABP) tem como objetivo principal capacitar os alunos a construírem o conhecimento conceitual, procedimental e atitudinal por meio da abordagem de problemas propostos, que os expõem a situações motivadoras e os preparam para o mundo.

Borochovicius *et al*.(2014, *apud* Ribeiro, 2008) ainda afirma que a Aprendizagem Baseada em Problemas possui objetivos educacionais abrangentes, estabelecendo uma base de conhecimentos estruturada em torno de problemas reais e integrando o desenvolvimento de habilidades de aprendizagem autônoma e

trabalho em equipe. Essa abordagem favorece a adaptabilidade a mudanças, a capacidade de resolver problemas em situações não rotineiras, o pensamento crítico e criativo, o trabalho em equipe, bem como o compromisso com a aprendizagem e o aperfeiçoamento contínuo.

Dessa forma, essa técnica foi escolhida para compor o desenvolvimento da solução proposta para incluir a apresentação de problemas do mundo real ou até mesmo situações hipotéticas, onde o usuário sendo o centro do ensino-aprendizado poderá relacionar e solucionar utilizando a lógica, o mesmo poderá interagir com a aplicação a ponto de solucionar os problemas propostos.

1.2.2. Feedback

De acordo com Fluminhan *et al*.(2013, *apud* Mason e Bruning, 2003) o objetivo do *feedback* é ajudar o aluno a identificar suas falhas e aprimorar seu desempenho, buscando maneiras de corrigir o que está incorreto e desenvolver seu potencial desejado.

Fluminhan *et al*.(2013, *apud* Vrasidas e McIsaac, 1999) ainda afirma que o *feedback* consiste nas respostas fornecidas pelo professor ao aluno em relação à correção de diversas atividades propostas, como tarefas de casa, trabalhos extras e contribuições em sala de aula. O mesmo autor(2013, *apud* Shute, 2007) mostra que no contexto educacional, o *feedback* desempenha um papel formativo. Definindo-o como informações comunicadas ao aprendiz com o objetivo de modificar seu pensamento ou comportamento para promover a aprendizagem. O principal objetivo do *feedback* formativo é aumentar o conhecimento, as habilidades e a compreensão do aluno em relação a um determinado conteúdo.

De tal forma, o *feedback* é a possibilidade de fornecer resultados ou informações ao usuário após concluir uma tarefa ou desafio. No contexto da solução proposta, esse resultado pode ser um conjunto de características provenientes da gamificação, como, pontuação, nível ou incentivos para continuar a aplicação.

1.2.3. Elementos audiovisuais

Affini e Américo(2007) afirmam que a utilização da produção audiovisual como uma alternativa metodológica ativa de ensino e aprendizagem permite que os

alunos se tornem cada vez mais autônomos, participativos, críticos e ativos na construção de seus conhecimentos. Isso ocorre porque os processos de criação de vídeos oferecem oportunidades para o desenvolvimento de habilidades, conhecimentos e atitudes que são essenciais para a vida profissional dos estudantes universitários, tais como proatividade, trabalho em equipe, aprendizagem colaborativa, pesquisa investigativa, criatividade, escrita, organização, tomada de decisões, aprendizagem significativa e autonomia.

Dessa forma, com a utilização dessa técnica, será possível aplicar elementos de áudio, como a narração, que ajudará a transmitir o conteúdo e auxiliar no foco e memorização do mesmo, e imagens para relacionar conteúdos textuais ou conceitos presentes no contexto de lógica de programação, facilitando o entendimento. Ambos, serão de extrema importância para auxiliar no ensino de lógica de programação para pessoas com dislexia, visto que o problema central das pessoas com esse transtorno envolve a dificuldade de relacionar elementos textuais.

1.3. MODELO DE APLICAÇÃO

Nesta seção está presente a descrição do modelo ou tipo de aplicação que a solução proposta se enquadra.

1.3.1. Aplicação educacional

Segundo Menezes(2001), um *software* ou aplicação educacional é um programa de computador que tem como objetivo atender às necessidades e objetivos pedagógicos. Portanto, qualquer *software* pode ser considerado educacional, desde que seja utilizado dentro de um contexto e situação de ensino-aprendizagem, com uma metodologia que guie todo o processo.

No contexto da solução proposta, a aplicação educacional é um *software* interativo que auxilia no ensino de lógica de programação trazendo o conteúdo de forma gamificada para pessoas que possuem dislexia.

1.3.2. Código aberto

Segundo a Amazon Web Services(2023)[3], um *software* de código aberto é um tipo de *software* que possui o código-fonte disponível para que qualquer pessoa possa examinar, modificar e aprimorar. O código-fonte é a parte do *software* que os programadores podem manipular para alterar o funcionamento da aplicação ou adicionar novos recursos. Com acesso ao código-fonte de um *software*, qualquer pessoa pode melhorar ou personalizar a aplicação, adicionando recursos ou corrigindo erros existentes.

Para a aplicação desenvolvida neste trabalho, torna-lo uma aplicação de código aberto, traz a possibilidade de futuras melhorias por outros desenvolvedores.

[3] AMAZON WEB SERVICES. **What is Open Source?**, c2023. Disponível em: <https://aws.amazon.com/pt/what-is/open-source/>. Acesso em: 18 maio 2023

CAPÍTULO 2: EXPLORANDO AS FERRAMENTAS UTILIZADAS

Neste capítulo será apresentado a relevância e importância das técnicas, abordagens e ferramentas utilizadas para o desenvolvimento da aplicação proposta neste trabalho, assim como o desenvolvimento de ferramentas para o meio educacional. O objetivo é discutir a eficácia dos recursos selecionados para o contexto do ensino de lógica de programação para pessoas com dislexia, e enfatizar como os mesmos podem auxiliar no processo de ensino-aprendizagem.

O primeiro ponto abordado neste capítulo diz respeito a abordagem da gamificação, na qual introduz elementos típicos de jogos, mas aplicados em contextos que não são jogos. No ambiente educacional, essa abordagem tem sido amplamente utilizada e visa auxiliar e compor a aplicação ao lado dos outros métodos já citados.

Outro ponto chave é a abordagem em Aprendizagem baseada em Problemas (APB) onde será discutida sua utilidade ao lado da gamificação, referente a passagem de conhecimento. Este método propõe colocar pessoas diante de problemas reais, onde o mesmo deve incentivar a aplicar conhecimentos teóricos em cenários práticos.

Em seguida, será mostrado como as técnicas e abordagens audiovisuais podem auxiliar e compor a aprendizagem baseada em problemas, visto que, esse método é eficaz e recomendado para pessoas portadoras de dislexia.

Do ponto de vista prático, foi escolhido o software Unity para o desenvolvimento da aplicação. A escolha deste motor de jogo será justificada e será feita a comparação entre as demais ferramentas existentes e disponíveis atualmente. A discussão incluirá a adequação da Unity para o propósito específico deste trabalho, destacando suas funcionalidades que são úteis para o desenvolvimento de aplicações.

Por fim, o capítulo explora o uso da plataforma ElevenLabs, que é uma solução *text-to-speech* (TTS), no qual converte textos em fala sintetizada. O mesmo é de suma importância para compor e auxiliar a abordagem audiovisual da aplicação.

Cada uma das técnicas, abordagens e tecnologias citadas serão discutidas e mostradas as justificativas de utilização, no qual serão interligadas e mostradas como a combinação dos mesmos é de fato útil para o desenvolvimento da aplicação proposta neste trabalho.

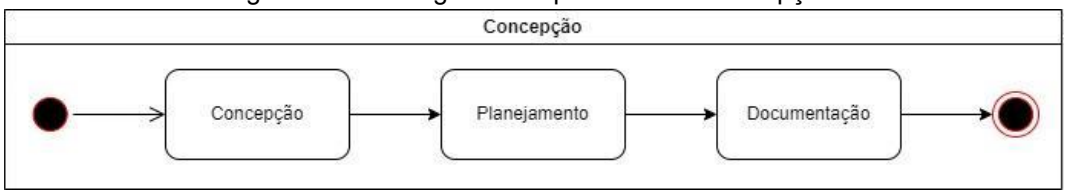
2.1. PROCESSO DE DESENVOLVIMENTO DE JOGOS

De acordo com Petrillo(2008, *apud* Rucker, 2002), o processo para o desenvolvimento de jogos é baseado no processo de modelagem em cascata, no qual é composto por etapas onde são executadas de modo sequencial, onde cada uma gera um produto e são independentes umas das outras. Cada uma dessas etapas que serão discutidas posteriormente cercam desde parte teórica conceitual até parte prática no qual é referente ao desenvolvimento do projeto. Apesar de utilizar como base o modelo em cascata, dependendo do planejamento e regras de negócio determinadas pela equipe, é possível e comum aplicar modificações no modelo de processo.

2.1.1. Concepção e planejamento

A primeira parte do processo de desenvolvimento, envolve elaboração de funcionalidades básicas, das quais posteriormente serão documentadas, na mesma é onde a história e enredo, custos e análise de mercado também são visualizadas e planejadas. Segundo NuES-UFRRJ(2020), na fase inicial da concepção de um jogo, diversas tarefas são empreendidas com o intuito de estabelecer aspectos fundamentais que irão orientar o desenvolvimento subsequente, é comum que muitas das ideias e conceitos que serão explorados no projeto já tenham surgido em discussões ou reflexões informais anteriores à implementação de um processo estruturado.

Figura 2.1 - Fluxograma do processo de concepção.



Fonte: Próprio autor.

Essas ideias preliminares, por hora, ainda não formalizadas são de suma importância, por constituírem a base criativa no qual o jogo será construído. É também nessa fase, que a equipe de desenvolvimento trabalha para aplicar esses conceitos e ideias iniciais em um processo conferente, assim, definindo metas e

objetivos claros que servirão de guia para as etapas subsequentes, no qual irá assegurar que o projeto seja original, viável e atraente para o público-alvo.

Ainda dentro deste processo, é feita a documentação inicial do projeto, no qual é chamado de *Game Design Document* (Documento de Design de Jogo) ou GDD, no qual contém as informações claras das mecânicas, design, história e qualquer outra informação detalhada a respeito do projeto. Segundo Petrillo (2008), o *Game Design Document* é o principal documento, e muita das vezes a única documentação de um jogo. No qual o objetivo do mesmo é detalhar as mecânicas do jogo, incluir principais componentes da história, cenários e outras informações detalhadas sobre o projeto. O mesmo autor afirma ainda que, a criação do documento sólido é visto tradicionalmente como o passo mais importante no processo de desenvolvimento de um jogo, no qual, as dificuldades encontradas na criação do mesmo são providas da natureza da tarefa e pelas ferramentas utilizadas na concepção.

De acordo com Main Leaf(2023), atualmente existem três tipos de documentos para o *Game Design Document*, o primeiro deles é o GDD de página-única, no qual é desenvolvido em uma única página onde é composto a visão geral do projeto de jogo, normalmente possui elementos visuais para que o estilo de jogo fique claro para os desenvolvedores. O segundo modelo é o GDD 10-páginas no qual possui detalhamento mais completo sobre o projeto de jogo, neste modelo são descritos os cenários, desafios, personagens, elementos de mecânica da jogabilidade, história e entre outros elementos detalhados sobre o jogo. Por fim, o terceiro e último modelo é referente ao GDD Bíblia, no qual pode ser maior que 15 páginas onde possui um grande detalhamento sobre o projeto de jogo, com esse modelo é possível deixar claro o tema e ambiente de jogo. Nesse modelo normalmente é descrito diálogos, diagramas e elementos importantes para o processo de desenvolvimento do jogo.

2.1.2. Fase de pré-produção

Sendo a segunda etapa do processo de desenvolvimento, a pré-produção se torna uma das etapas mais importantes por compor parte do refinamento do *Game Design Document*, onde são corrigidos e detalhados elementos do projeto de jogo.

Segundo Aleem *et al.* (2016), nesta etapa é desenvolvida a prototipação do projeto de jogo, no qual ajuda o desenvolvedor a clarear os fundamentos das mecânicas do jogo final. O mesmo afirma que o protótipo de jogo é de suma importância, pelo fato de que, é utilizado para ajudar a validar a experiência do usuário. O mesmo autor ainda afirma, que nesta etapa é onde é feito todo o processo de gerenciamento das regras de negócio de marketing e comércio, em que são estudados os riscos da produção do projeto de jogo e levantam pontos como originalidade, tempo de produção e orçamento.

Figura 2.2 - Fluxograma do processo de pré-produção.

Fonte: Próprio autor.

2.1.3. Fase de produção

A terceira etapa do processo é referente a produção, no caso, o desenvolvimento em si do projeto já planejado e documentado. Aleem *et al.*(2016) mostra que nesta etapa ocorre processo de criação de ativos, produção de *storyboard*, desenvolvimento de plataformas, criação de *scripts* ou programação.

Figura 2.3 - Fluxograma do processo de produção.

Fonte: Próprio autor.

O mesmo autor destaca a inovação para a criação de ativos, no qual diz respeito a objetos 3D, arquivos de áudio, imagens, texturas e quaisquer outros elementos que compõem o projeto de jogo. De acordo com Petrillo(2008), é nesta etapa que as equipes de arte, som e codificação trabalham juntas para o desenvolvimento contínuo das versões preliminares do ambiente do jogo, no qual são guiados pela documentação desenvolvida anteriormente.

Aleem *et al.*(2016) afirma que é nesta etapa em que é utilizado o motor de jogo, no qual é um software especializado que integra funcionalidades como som, física, inteligência artificial, animação e gerenciamento de memória. Segundo o autor, a complexidade de código de programação é cada vez maior em decorrência da incorporação de módulos mais complexos e técnicas de inteligência artificial, linguagens como C e C++ são comuns e ferramentas são propostas para facilitar o desenvolvimento.

2.1.4. Fase de pós-produção

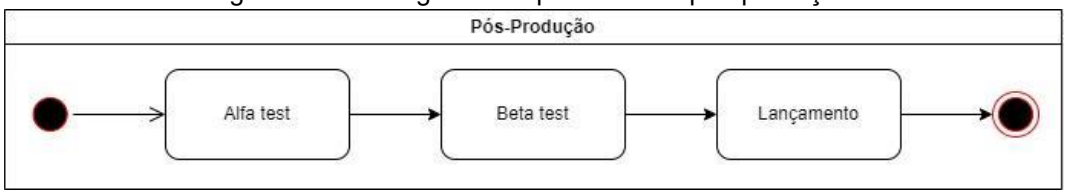
Nesta terceira e última etapa, foca na validação da produção do projeto de jogo, no qual é feito os testes relacionados para verificar a qualidade do projeto desenvolvido, no qual é representado pelo fluxograma da figura 2.4. Aleem *et al.*(2016) afirma que nesta etapa é onde o processo de garantia da qualidade é feito, no qual é essencial para validar o processo de desenvolvimento do jogo. O mesmo mostra que a coleta e avaliação dos dados de processo desde a pré-produção até a pós-produção ajudam a confirmar se o processo do desenvolvimento atingiu o objetivo definido.

Petrillo(2008) mostra que nesta etapa ocorre o estágio de teste alfa onde é feita a validação do projeto com as especificações do *Game Design Document*, o mesmo afirma que neste estágio ainda é possível um ciclo de atividade de construção, no qual as críticas e sugestões podem ser encaminhadas para os desenvolvedores.

Segundo Aleem *et al.*(2016), o mesmo afirma que nesta etapa ocorre o *beta testing* ou teste beta, no qual o jogo é liberado publicamente com propósito de teste, onde é feita a validação de funcionalidades e identificação de problemas que os desenvolvedores podem não ter encontrado. Petrillo(2008) afirma que neste estágio de teste beta, o objetivo é medir a receptividade dos usuários ao jogo, além de detectar eventuais problemas que possam ocorrer.

Por fim o estágio final de pós-produção do projeto é chamado de versão final ou *gold*, de acordo com Petrillo(2008), esse é o momento em que as principais sugestões dos usuários providas do teste beta foram implementadas no jogo, e o mesmo está pronto para ser lançado ao público geral.

Figura 2.4 - Fluxograma do processo de pós-produção.

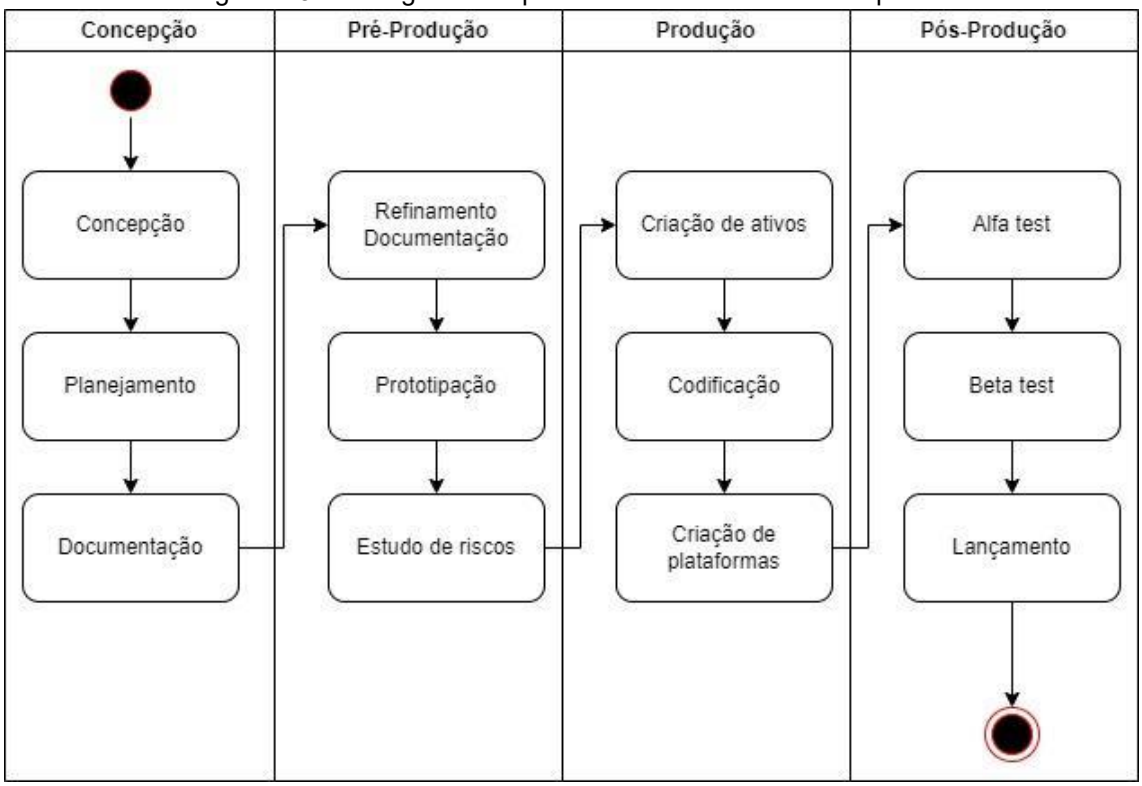


Fonte: Próprio autor.

Um ponto chave importante a ser citado, é sobre os tipos de testes aplicados na pós-produção. Aleem *et al.*(2016) afirma que nesta etapa podem ser aplicados dois tipos de testes, no qual o primeiro é o teste baseado em heurísticas que utilizam diretrizes de design para avaliar a jogabilidade, usabilidade e qualidade do jogo tanto por desenvolvedores quanto por usuários. O segundo tipo de teste, segundo o autor, são os testes empíricos, no qual são focados na qualidade e usabilidade mas que por outro lado são utilizados em jogos sérios, especialmente no contexto educacional.

Na figura 2.5 é possível visualizar o fluxograma completo da representação do processo de desenvolvimento baseado no modelo cascata, adaptado especificamente para este contexto. Como mencionado anteriormente, a adaptação do modelo cascata para o desenvolvimento de jogos não é uma abordagem rígida, mas sim flexível e suscetível a variações conforme as características específicas de cada equipe, projeto ou estudo. As modificações necessárias serão realizadas de acordo com as necessidades e particularidades do projeto em questão. Este fluxograma serve como um guia geral, oferecendo uma estrutura básica que pode ser ajustada para atender às demandas únicas de cada projeto de desenvolvimento de jogos, garantindo assim uma abordagem mais personalizada e eficaz.

Figura 2.5 - Fluxograma do processo modelo cascata adaptado.



Fonte: Próprio autor.

Neste tópico foi apresentado e discutido o processo de desenvolvimento de jogos no qual utiliza como base o modelo em cascata, visto que o mesmo é dividido em três etapas das quais possuem vários estágios, onde os mesmos são definidos pela equipe de desenvolvimento em decorrência da necessidade do projeto.

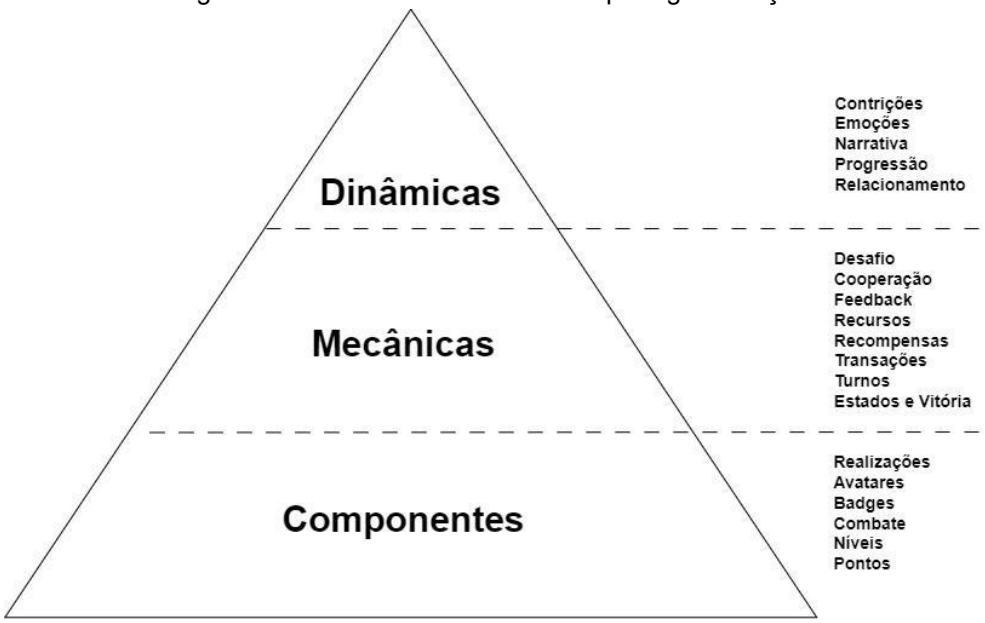
2.2. GAMIFICAÇÃO

Este tópico tratará da discussão sobre a justificativa da utilização da gamificação no contexto da aplicação proposta neste trabalho.

O uso de elementos de jogos em contextos que não são jogos é chamado de gamificação. No contexto educacional, a gamificação pode ser utilizada para melhorar a motivação, o engajamento e a aprendizagem dos alunos. Visto que em jogos, os mesmos são desenvolvidos para cativar o jogador e fazê-lo cumprir os objetivos propostos pelo mesmo. Elementos como objetivos e metas, *feedback*, recompensas, competição, progresso e nível são exemplos de elementos no qual podem ser utilizados nas aplicações educacionais para motivar o aluno em meio a passagem de conteúdo. É importante cativar o mesmo e fazê-lo sentir confortável

com o conteúdo e ambiente, quando os alunos estão motivados e engajados, os mesmos são mais propensos a aprender.

Figura 2.6 - Pirâmide de elementos para gamificação.



Fonte: Vancini *et al.* (2020).

As abordagens tradicionais de ensino tendem a ser centradas no professor, com o mesmo transmitindo informações aos alunos. Por outro lado, a gamificação centra o aluno, no qual assumem o papel mais ativo no aprendizado. Essa abordagem pode ser mais envolvente e motivadora do que as abordagens tradicionais. A gamificação não é uma abordagem milagrosa, é necessário escolher os elementos de jogos certos para o contexto e para o público alvo. Neste trabalho, foram utilizados elementos de pontuações, *feedback*, progresso e nível, no qual podem ajudar o aluno a se sentir mais envolvido no aprendizado.

A proposta para o elemento de pontuação no jogo, é justificado pelo qual o aluno poderá acompanhar seu progresso específico em cada tópico a partir de sua pontuação, pode verificar seu desempenho e isso poderá ajudar a motivar o mesmo a se dedicar em alcançar a pontuação máxima durante a utilização da aplicação. O processo de *feedback* é importante para mostrar ao aluno o desempenho atual em determinado momento, outro ponto que reforça a importância do *feedback* é a questão de que o retorno do mesmo é instantâneo, fazendo assim com que o aluno se assegure no momento da questão do seu desempenho. O progresso e nível é de

suma importância, atual como um tipo de *feedback* mas mostra ao aluno o quanto o mesmo decorreu em determinado contexto.

No quadro 2.1, é possível visualizar a comparação das principais características a respeito da abordagem de gamificação utilizada na aplicação proposta neste trabalho em relação a características das abordagens tradicionais de ensino.

Quadro 2.1 - Comparativo entre gamificação e abordagens tradicionais de ensino.

Características	**Gamificação**	**Abordagens Tradicionais de Ensino**
Foco	Engajamento e motivação através de elementos de jogos em contextos de aprendizagem.	Foco na transmissão de conteúdo e cumprimento de currículo.
Papel do Professor	Designer de experiência, facilitador e motivador.	Transmissor de conhecimento, autoridade em sala de aula.
Papel do Estudante	Participante ativo, buscando recompensas e reconhecimento.	Receptor de informações, passivo na aquisição de conhecimento.
Estrutura Curricular	Pode ser não-linear, com níveis, conquistas e desafios.	Linear e sequencial, baseado em um currículo estabelecido.
Avaliação	Baseada em pontos, emblemas, tabelas de classificação e conquistas.	Tradicionalmente baseada em testes, provas e trabalhos.
Desenvolvimento de Habilidades	Promove engajamento, solução de problemas e pensamento estratégico.	Foco no desenvolvimento acadêmico e preparação para avaliações.
Contexto de Aprendizagem	Ambiente lúdico que promove interatividade e competição saudável.	Ambiente mais formal e estruturado, com menos ênfase na interação lúdica.
Colaboração	Incentiva a cooperação e competição, dependendo do design do jogo.	Pode promover trabalho individual ou em grupo, mas geralmente menos integrado.
Flexibilidade	Altamente adaptável a	Menos flexível, seguindo

	diferentes conteúdos e dinâmicas de grupo.	uma estrutura pré definida e rígida.
Feedback	Imediato e contínuo, permitindo ajustes rápidos no processo de aprendizagem.	Pode ser menos frequente e menos imediato, dependendo do método de avaliação.
Motivação	Intrinsecamente motivadora através de recompensas e elementos de jogo.	Dependendo de fatores externos, como notas e aprovação.

Fonte: Próprio autor.

A abordagem de gamificação foi implementada na aplicação visando trazer benefícios para o usuário, tal abordagem utilizada traz a possibilidade do usuário repetir o conteúdo proposto quantas vezes lhe for necessário, de forma a não ter consequências em que o mesmo presenciaria na vida real com abordagens tradicionais de ensino. É importante reforçar que a aplicação da gamificação deve ser um trabalho minucioso, isso porque o desenvolvimento de uma aplicação com essa abordagem pode custar caro na questão elaborativa. Deve ser feita cuidadosamente a seleção dos elementos de jogos nos quais se encaixam para a aplicação, no qual os elementos apresentados acima, são básicos no qual podem auxiliar no processo de ensino e aprendizagem do aluno.

2.3. APRENDIZADO BASEADO EM PROBLEMAS (ABP)

Neste tópico será discutido a justificativa de escolha para a abordagem de Aprendizado Baseado em Problemas (ABP). A mesma é uma metodologia de ensino-aprendizagem focada na solução de problemas reais como ponto de partida para a aquisição e aplicação de novos conhecimentos. A mesma incentiva os alunos a serem o centro do processo de aprendizagem, incentivando-os a se tornarem aprendizes ativos e desenvolverem habilidades críticas de pensamento de resolução de problemas.

Em comparação com abordagens tradicionais de ensino, as mesmas são frequentemente centradas no professor e na memorização do conteúdo, enquanto o ABP é uma abordagem centrada no aluno.

Na aplicação proposta neste trabalho, a ABP foi utilizado e aplicado nos vídeos de passagem de conhecimento em que o problema é contextualizado e posteriormente o problema é lançado, focando no aluno para a resolução do mesmo. Os desafios foram elaborados de forma na qual não precisassem de contextualização, em que pudessem ser autoexplicativos visto a experiência anterior como os tópicos de passagem de conhecimento, isso reforça a questão do raciocínio lógico do aluno perante o desafio ou problema.

No quadro 2.2, é relacionado as principais características das quais são comparadas com abordagens tradicionais de ensino, a comparação tem a finalidade de demonstrar como o ABP pode ser benéfico em relação ao comparado.

Quadro 2.2 - Comparativo entre os principais motores de jogos disponíveis.

Características	**Aprendizado Baseado em Problemas (ABP)**	**Abordagens Tradicionais de Ensino**
Foco	No processo de aprendizagem através da resolução de problemas reais ou simulados.	Na transmissão de conteúdo pelo professor e memorização de informações pelos alunos.
Papel do Professor	Facilitador e orientador do processo de aprendizagem.	Autoridade principal, detentor e transmissor do conhecimento.
Papel do Estudante	Ativo, investigador e solucionador de problemas.	Passivo, receptor do conhecimento transmitido pelo professor.
Estrutura Curricular	Interdisciplinar e integrada em torno de problemas complexos.	Disciplinar, com conteúdos fragmentados e sequenciais.
Avaliação	Baseada no desempenho do processo de aprendizagem e habilidades de resolução de problemas.	Focada em testes e exames que medem a retenção de informações.
Desenvolvimento de Habilidades	Fomenta habilidades de pensamento crítico, trabalho em equipe, auto aprendizagem e aplicação prática do conhecimento.	Enfatiza a aprendizagem individual e o desenvolvimento de habilidades para testes e exames.
Aplicação do Conhecimento	Imediata, através da resolução de problemas	Muitas vezes teórica, com aplicação prática não

	práticos.	imediata ou desconectada do contexto de aprendizagem.
Contexto de Aprendizagem	Baseado em cenários reais que incentivam a relevância e o engajamento.	Contexto muitas vezes artificial, limitado ao que é apresentado nos livros e palestras.
Colaboração	Encoraja a colaboração e a aprendizagem em grupo.	Pode encorajar a competição e o desempenho individual.
Flexibilidade	Permite a adaptação a diferentes estilos de aprendizagem e ritmos.	Frequentemente rígido, seguindo um plano de ensino predefinido.

Fonte: Próprio autor.

Essa abordagem foi implementada de maneira a permitir que o usuário utilize seus próprios conhecimentos para solucionar problemas propostos pela aplicação. Esses problemas básicos estão relacionados ao conteúdo de lógica de programação, no qual faz analogias com situações comuns da vida real, possibilitando que o aluno assuma o papel de protagonista no contexto de ensino-aprendizagem.

Em suma, a abordagem do Aprendizado Baseado em Problemas coloca o aluno como foco central do cenário de ensino aprendizagem, sendo ele o ator principal, é possível focar atenção, motivar e engajar a solucionar problemas dos quais ele conhece e relacionar com o conteúdo a ser passado ao mesmo.

2.4. AUDIOVISUAL

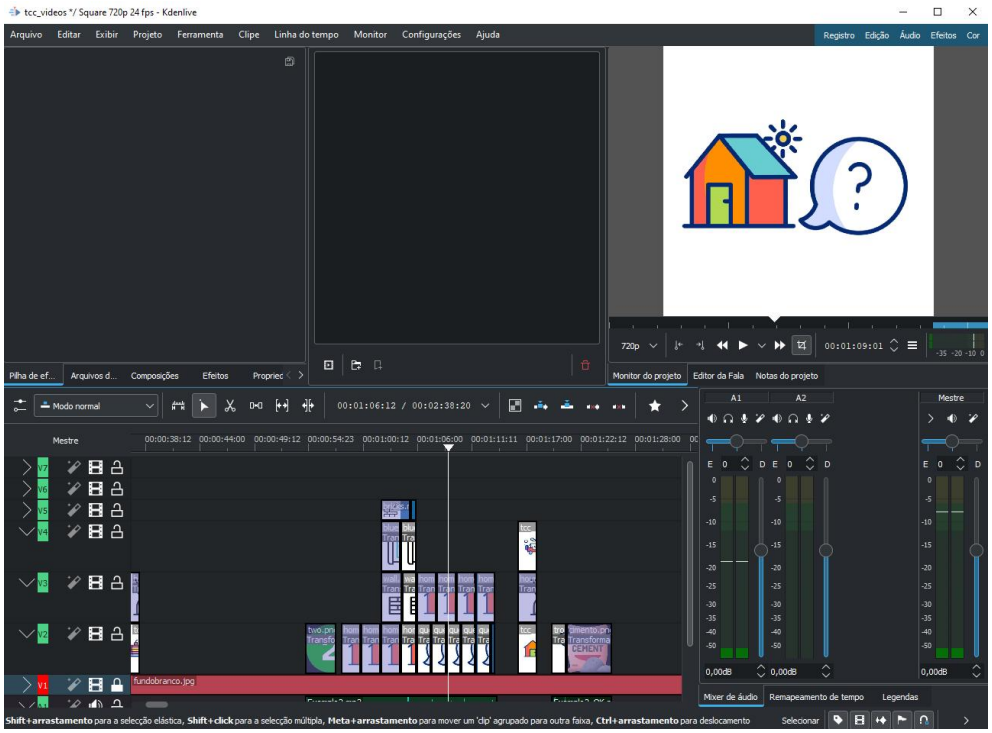
Neste tópico será discutido sobre a abordagem e técnicas audiovisuais elaboradas para a aplicação deste trabalho, assim como será justificado a utilização das mesmas.

A introdução dos meios audiovisuais envolve a compreensão de que somos seres multimodais no qual a nossa percepção é influenciada por múltiplos estímulos. A integração de texto, imagem e som tem o potencial de criar uma compreensão mais rica do que qualquer um desses elementos isoladamente. Esse fenômeno é reforçado pelas teorias de aprendizagem multimodal de Kress e Van Leeuwen(2001) no qual argumentam que, os significados são construídos e interpretados através de diversas formas semióticas e não apenas pela linguagem escrita.

As abordagens tradicionais de ensino têm sido frequentemente focadas e centradas no texto e na transmissão oral de conteúdo, seguindo modelo mais linear e sequencial, mas por outro lado, a abordagem audiovisual é mais dinâmica e interativa.

De tal forma, a abordagem foi implementada na aplicação proposta neste trabalho de forma a utilizar textos de maneira reduzida, mas junto ao áudio, na qual não pudessem ser representados apenas por imagem e áudio, nos vídeos de passagem do conteúdo os mesmos são combinados para fazer a relação do que seria um elemento textual, analisando elementos dos quais combinam texto, áudio e imagem, com o objetivo de melhorar e facilitar a compreensão do aluno perante a passagem de informação, é possível visualizar o processo na figura 2.7. Dessa forma, é possível fazer a integração da abordagem audiovisual junto a gamificação e a aprendizagem baseada em problemas, no qual juntos possuem o potencial de motivar, cativar, e prender a atenção do aluno perante o ensino-aprendizagem.

Figura 2.7 - Produção de material audiovisual.



Fonte: Próprio autor.

Com isso, a integração da abordagem audiovisual nesta aplicação reflete a adaptação às realidades de processamento de informação e apresenta novas

possibilidades para a motivação e aprendizagem do aluno. É uma abordagem dinâmica que evolui com as tecnologias e as compreensões de como aprendemos.

No quadro 2.3, foi relacionada as principais características que a abordagem audiovisual possui, no qual pode trazer benefícios para o ambiente de ensino aprendizagem. A mesma é comparada também com as abordagens tradicionais de ensino, visando mostrar as suas virtudes a serem usufruídas.

Quadro 2.3 - Comparativo entre abordagem audiovisual e abordagens tradicionais de ensino.

Características	**Abordagem Audiovisual**	**Abordagens Tradicionais de Ensino**
Foco	Utilização de mídias audiovisuais para facilitar o aprendizado.	Foco na instrução direta e no conteúdo textual.
Papel do Professor	Curador de conteúdo audiovisual e facilitador da discussão.	Fonte principal de conhecimento e instrução.
Papel do Estudante	Interativo e interpretativo, envolvendo-se com o conteúdo apresentado.	Passivo, ouvinte, focado em anotações e memorização.
Estrutura Curricular	Pode ser temática, baseada em narrativas ou estudos de caso apresentados em vídeos.	Baseada em um currículo estruturado e progressão sequencial dos tópicos.
Avaliação	Pode incluir análises críticas e discussões sobre o material audiovisual.	Tradicionalmente baseada em testes escritos e exames.
Desenvolvimento de Habilidades	Fomenta habilidades de análise crítica, compreensão e discussão.	Enfatiza a memorização e compreensão de informações e métodos.
Aplicação do Conhecimento	Encoraja a aplicação prática do conhecimento através de exemplos visuais e estudos de caso.	Muitas vezes teórica, com exemplos práticos menos frequentes.
Contexto de Aprendizagem	Dinâmico e muitas vezes mais envolvente devido ao estímulo visual e auditivo.	Mais estático, geralmente limitado à sala de aula e materiais de leitura.
Colaboração	Pode promover a	Colaboração limitada a

	discussão em grupo e projetos colaborativos baseados em conteúdo visual.	discussões em sala de aula e trabalhos em grupo.
Flexibilidade	Alta, com a possibilidade de acessar conteúdo audiovisual de qualquer lugar e a qualquer hora.	Menos flexível, dependente de horários de aula e acesso a material físico.
Engajamento Sensorial	Estimula vários sentidos simultaneamente, o que pode melhorar a retenção.	Frequentemente limitado ao estímulo visual e auditivo de palestras e leitura.
Tecnologia	Dependente de recursos tecnológicos para apresentação e acesso ao conteúdo.	Pode ou não incorporar tecnologia, muitas vezes dependente de livros e quadro-negro.

Fonte: Próprio autor.

A abordagem audiovisual foi implementada com a finalidade de trazer os benefícios apresentados no quadro 2.3, onde a presente abordagem foi combinada com a gamificação e aprendizado baseado em problemas, visando trazer o conjunto de benefícios dessas abordagens apresentadas e implementadas na aplicação proposta. Com isso, foi possível visualizar e comparar as características das quais tem alto índice benéfico para auxiliar o ensino, no qual podem ser aplicadas em qualquer aplicação educacional em vista de uma análise minuciosa para a sua implementação.

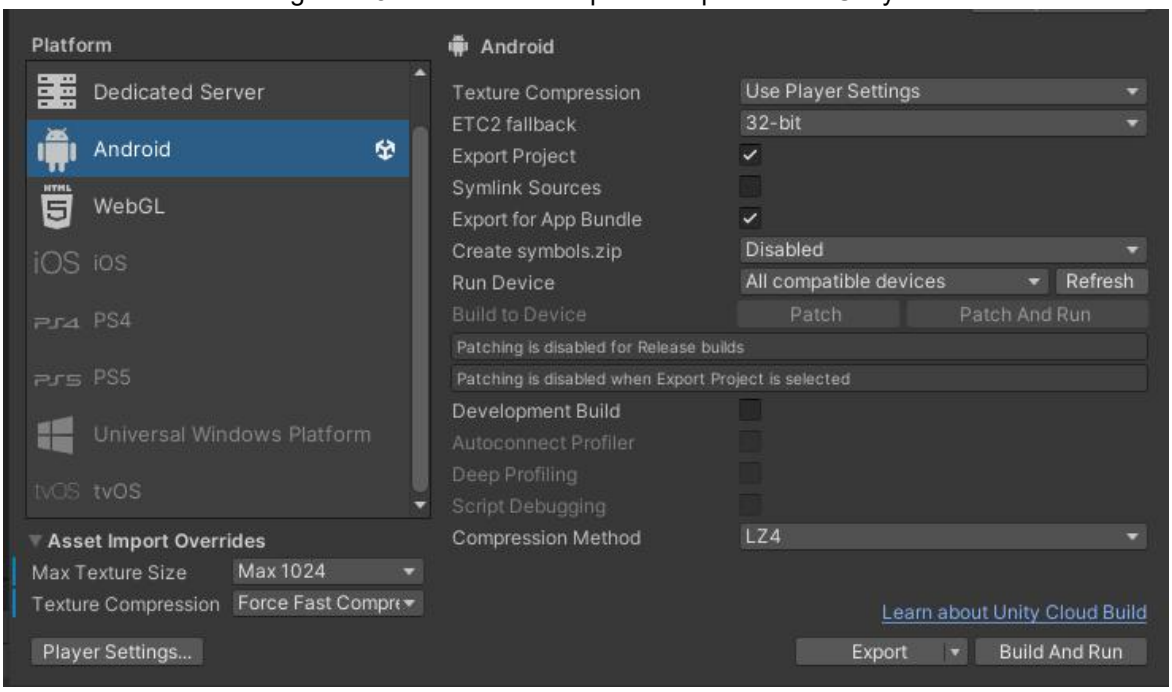
2.5. UNITY

Neste tópico será discutido a utilização da Unity como motor de desenvolvimento da aplicação proposta neste trabalho de pesquisa, assim como será justificada a utilização da mesma. Além disso, será feita uma comparação com os motores disponíveis atualmente.

Atualmente, a Unity tem a possibilidade de exportar projetos para diversas plataformas, desde consoles, dispositivos móveis que utilizam sistema Android e iOS, até outros dispositivos como Smart TVs e plataformas de realidade virtual. Assim, é possível desenvolver um único projeto e exportar para várias plataformas disponíveis

sem precisar reescrever ou desenvolver outro projeto do mesmo, isso faz com que o projeto seja multiplataforma implicitamente. É um ponto extremamente importante, visto que, ao desenvolver uma aplicação educacional é possível levar a mesma a diversas plataformas, tornando mais fácil o seu acesso.

Figura 2.8 - Plataformas suportadas pelo motor Unity.

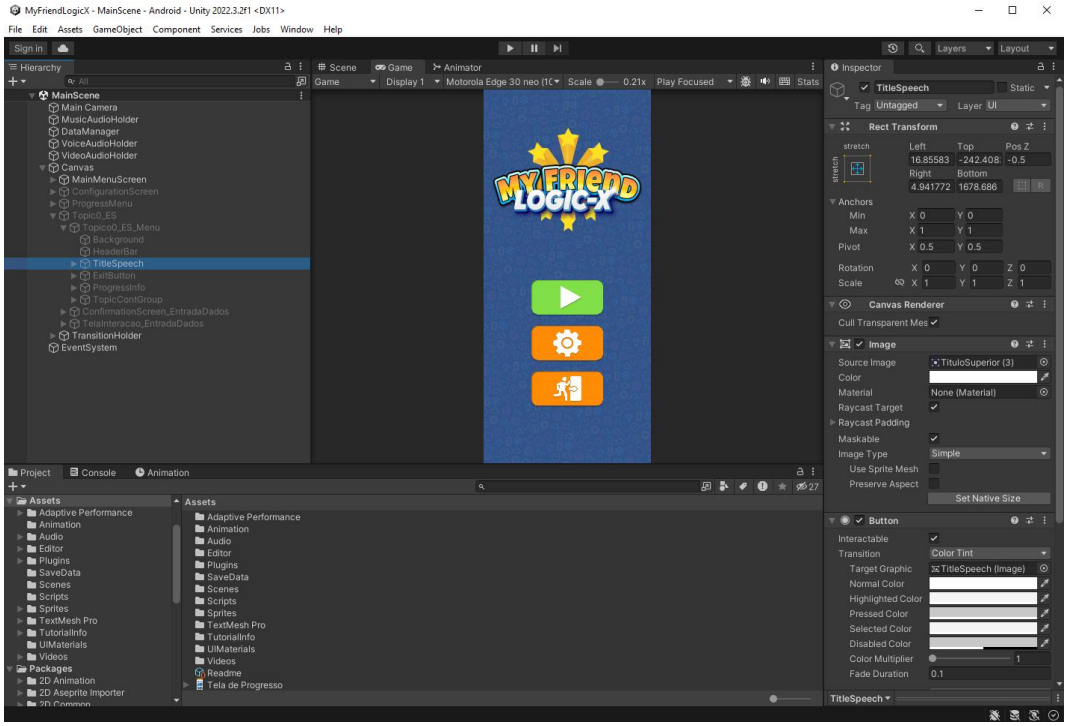


Fonte: Próprio autor.

A Unity também possui uma comunidade vasta, em que usuários compartilham conhecimentos, algoritmos, ferramentas personalizadas e melhorias para o fluxo de trabalho com o motor. A Unity também possui fórum em que os usuários do motor podem expor dúvidas e dicas para outros usuários e desenvolvedores, isso é extremamente importante, principalmente para uma pessoa que iniciou seus estudos recentemente com o *software*, o fato de ser uma vasta comunidade traz diversas gama de material disponível via internet, tanto vídeos como documentos.

Das ferramentas que a Unity disponibiliza, foi utilizada a *pipeline* 2D na qual possibilita o desenvolvimento de qualquer aplicação que utilize duas dimensões para trabalho. Do mesmo modo, foram implementados vídeos na aplicação, a Unity suporta vídeos em MP4 nativamente e sem nenhum tipo de configuração prévia, o que traz facilidade no desenvolvimento e importação com esse tipo de arquivo.

Figura 2.9 - Interface de desenvolvimento do motor Unity.



Fonte: Próprio autor.

Um ponto chave das funcionalidades utilizadas é o *Canvas*, no qual é possível desenvolver e manipular interface de usuário sem dificuldades, sendo possível criar botões, faixas, fundos e qualquer outro elemento que possa compor uma tela de uma aplicação 2D.

A Unity também possibilita a utilização de vários tipos de arquivos de áudio, a implementação de um sistema complexo de áudio para a aplicação depende da regra de negócio da aplicação, mas independente de qual seja, o desenvolvedor não encontrará dificuldades visto que o motor trabalha com objetos de jogo no qual cada um tipo tem seus recursos implementados por padrão, então independente se for um arquivo de áudio, imagem, vídeo, cada um desses elementos terão suas funcionalidades simplificadas de modo a facilitar o desenvolvimento da aplicação com os mesmos.

Para o desenvolvimento de *scripts* no qual é necessário manipular esses objetos de jogo, a Unity tem suporte nativo ao *C-Sharp*. É importante utilizar linguagem nativa, visto que o motor foi escrito com a mesma, assim é possível fazer uma fácil integração com o motor e compreender melhor como funciona cada biblioteca do mesmo.

Existem outros motores disponíveis atualmente, mas de fato a Unity é amplamente utilizada para aplicações de dispositivos móveis, no quadro 2.4 é possível visualizar os recursos utilizados da Unity na aplicação proposta neste trabalho em comparação com outros motores disponíveis.

Quadro 2.4 - Comparativo entre os principais motores de jogos disponíveis.

Características	**Unity**	**Unreal**	**CryEngine**	**Godot**
Linguagem	C#	C++, *Blueprints (visual scripting)*	C++, Lua	GDScript, C#, C++, *VisualScript*
Sistema de UI	Sim (uGUI)	*Sim (UMG - Unreal Motion Graphics)*	Sim (*Scaleform* para versões anteriores, agora proprietário)	Sim (sistema de UI dedicado)
Recursos de Áudio nativo	Motor de áudio integrado, suporta *middleware* de áudio	Motor de áudio integrado, suporta *middleware* de áudio	Motor de áudio integrado	Motor de áudio integrado
Recursos de vídeo nativo	Suporte a reprodução de vídeo	Suporte a reprodução de vídeo	Suporte a reprodução de vídeo	Suporte a reprodução de vídeo
Foco	Jogos, AR, VR, aplicações interativas	Jogos de alta qualidade, AR, VR	Jogos de alta qualidade, simulações	Jogos, aplicações interativas, educação

Custo de hardware	Baixo a Moderado	Alto (mais exigente para gráficos de alta qualidade)	Alto (especializado em gráficos de alta fidelidade)	Baixo a moderado
Complexidade	Baixa a Moderada	Alta (especialmente para C++)	Alta (devido a menos recursos e documentação)	Baixa a moderada (GDScript é amigável para iniciantes)
Licenciamento	Gratuito com *royalties* após um certo nível de receita	Gratuito com *royalties* após um certo nível de receita	Gratuito sem *royalties*	Completamente livre e open-source
Particularidade	Amplo suporte a plataformas; grande comunidade e ecossistema	Gráficos de ponta e sistema de física; filmagem virtual	Famoso por gráficos impressionantes; suporte a VR limitado	Motor leve e flexível; comunidade ativa open-source
Acesso ao Código Fonte	Disponível mediante pagamento	Totalmente livre e open-source	Parcialmente livre e open-source	Totalmente livre e open-source

Fonte: Próprio autor.

Com o quadro 2.4 é possível visualizar semelhanças entre as *engines* disponíveis, existe uma competitividade de mercado entre os principais motores que são Unreal e Unity, em que frequentemente estão atualizando suas ferramentas. Atualmente, a Unity é utilizada pela comunidade comumente para aplicações 2D,

que pode ser ou não para dispositivos móveis, enquanto Unreal é utilizada para aplicações com gráficos de alto nível.

A questão de desempenho da aplicação para os dois motores depende severamente de boas práticas de desenvolvimento, como, utilização de texturas de alta qualidade apenas em objetos dos quais necessitam do mesmo, otimização poligonal de objetos 3D para não haver alto processamento em parte da unidade de processamento gráfico(GPU), carregamento de objetos de forma dinâmica e otimização da interface de usuário(UI).

As boas práticas custam tempo para serem aplicadas no que resulta em maior tempo de desenvolvimento, das apresentadas, foram implementadas na aplicação proposta neste trabalho a otimização de texturas das quais não são superiores a 1080 *pixels*, visto que dificilmente um usuário executará em resoluções mais altas, isso também ajuda no tamanho em *megabytes* da imagem, no caso, quanto maior a imagem mais tempo a GPU *mobile* deverá ter para processar o arquivo e a otimização de UI para utilização dinâmica de objetos de tela.

A escolha do motor Unity para desenvolvimento foi apresentado acima, ao lado de seus recursos disponíveis que foram utilizados para o desenvolvimento da aplicação proposta neste trabalho de pesquisa, além de que o autor tem maior proficiência e conhecimento da mesma.

2.6. TEXT-TO-SPEECH

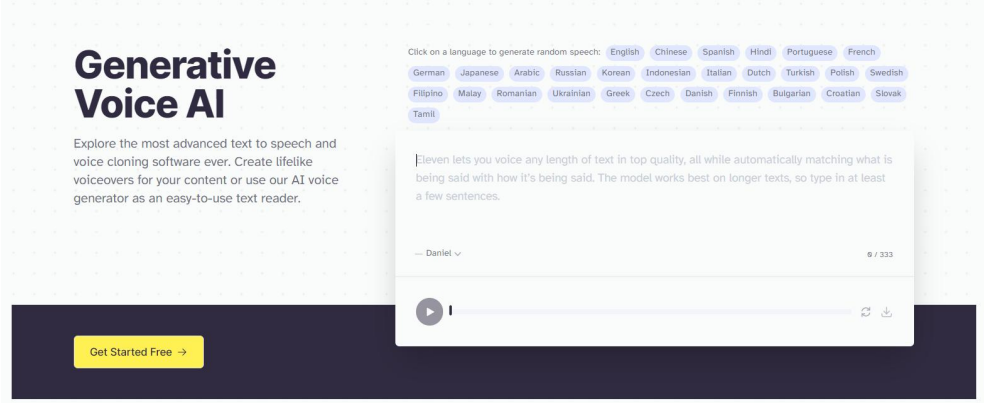
Neste tópico será discutido o uso da plataforma ElevenLabs como tecnologia *text-to-speech* para compor o desenvolvimento da aplicação proposta neste trabalho, assim como a justificativa da mesma.

O ElevenLabs é uma plataforma de inteligência artificial no qual possui um avançado recurso de texto para fala (*text-to-speech*), no qual é possível criar voz sintetizada a partir de textos, sendo possível ajustar o sentimento da fala processada. Com o mesmo é possível produzir fala sintetizada em até 29 idiomas disponíveis.

A plataforma foi utilizada a partir da elaboração do *Game Design Document* no qual detalha e especifica característica do projeto de jogo, o mesmo serve como mapa para todos os desenvolvedores ficarem alinhados perante as funcionalidades, recursos e particularidades da aplicação. No mesmo, ao finalizar o roteiro de cada

vídeo de passagem de conteúdo, foi implementada na plataforma o texto de roteiro e processada a voz sintetizada para a aplicação.

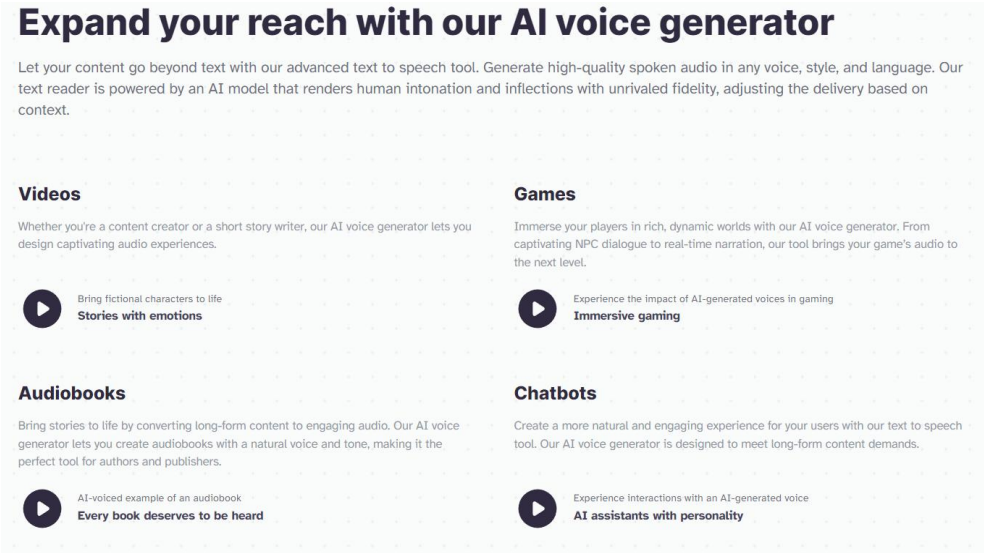
Figura 2.10 - Página inicial da plataforma ElevenLabs.



Fonte: elevenlabs.io.

O ElevenLabs pode ser utilizado tanto para fins educacionais quanto de entretenimento, sendo possível desenvolver aplicações audiovisuais e sonoras. A mesma disponibiliza uma API na qual pode ser implementada em qualquer aplicação em desenvolvimento, como jogos, *audiobook*, *chatbots* e assistentes virtuais.

Figura 2.11 - Contextos no qual a tecnologia *text-to-speech* pode ser utilizada.



Fonte: elevenlabs.io.

Para demonstrar como a tecnologia *text-to-speech* pode ser benéfica ao contexto educacional, foi elaborada uma comparação presente no quadro 2.5, no qual relaciona os principais benefícios do uso da tecnologia em relação ao não uso em aplicação das abordagens tradicionais de ensino.

Quadro 2.5 - Comparativo entre uso da tecnologia *text-to-speech* e abordagens tradicionais.

Características	**Tecnologia Text-to-Speech (TTS)**	**Abordagens Tradicionais de Ensino**
Foco	Conversão de texto em fala para melhorar o acesso e a compreensão do conteúdo.	Foco na leitura direta de textos e na explicação oral pelo professor.
Papel do Professor	Integrador de tecnologia para facilitar o aprendizado; menos foco na transmissão direta.	Transmissor principal de conhecimento e explicador de conteúdos.
Papel do Estudante	Ouvinte ativo; usuário da tecnologia para acessar informações escritas.	Leitor ativo; ouvinte passivo durante as explicações do professor.
Estrutura Curricular	Flexível, pode ser ajustada para incluir múltiplos formatos de conteúdo.	Estruturada, geralmente baseada em textos e materiais impressos.
Avaliação	Pode incluir avaliações adaptadas para estudantes com dificuldades de leitura.	Tradicionalmente baseada em leitura e escrita para testes e exames.
Desenvolvimento de Habilidades	Apoia habilidades de compreensão auditiva e pode beneficiar a aprendizagem de idiomas.	Focado no desenvolvimento de habilidades de leitura e escrita.
Aplicação do Conhecimento	Permite acessibilidade para estudantes com dificuldades visuais ou de leitura.	Exige que os alunos leiam e interpretem informações sem assistência adicional.
Contexto de Aprendizagem	Tecnológico e acessível, podendo ser utilizado em diferentes ambientes.	Confinado ao ambiente de sala de aula e aos materiais didáticos disponíveis.
Colaboração	Pode ser individualizado, mas também pode ser incorporado em atividades de grupo.	Pode variar entre aprendizado individual e projetos de grupo.
Flexibilidade	Alta, com possibilidade de ajustar a velocidade da fala e acessar em	Menos flexível, com dependência de materiais impressos e horários fixos

	múltiplos dispositivos.	de aula.
Inclusão	Alta, auxiliando alunos com deficiência visual, dislexia ou outras necessidades especiais.	Pode ser limitada para alunos com necessidades especiais que requerem adaptações.
Adaptação Tecnológica	Requer dispositivos e softwares específicos.	Geralmente não depende de tecnologia específica.

Fonte: Próprio autor.

Foi possível visualizar, que a tecnologia *text-to-speech* pode ser benéfica na área educacional, no qual a mesma pode atuar como um leitor e tradutor, visto que a plataforma possibilita a produção de voz sintetizada em vários idiomas.

Em suma, a plataforma traz funcionalidades simples em usabilidade que podem agregar acessibilidade em diversas áreas de desenvolvimento ou da educação.

Neste capítulo foram apresentadas as principais abordagens utilizadas no desenvolvimento da aplicação audiovisual gamificada My Friend Logic-X, onde foram discutidas as características principais, os recursos utilizados, ferramentas, e a justificativa para o uso das mesmas. É importante ressaltar que para cada tipo de aplicação, jogos ou afins é necessário fazer uma análise e concluir quais ferramentas e abordagens são necessárias para o desenvolvimento do mesmo, no meio dos jogos isso ocorre de forma recorrente com *game engines*, em que para aplicações 2D e *mobile* é comumente utilizado o motor Unity, enquanto para jogos com gráficos de alta qualidade é utilizado Unreal.

CAPÍTULO 3: JORNADA DE CRIAÇÃO DO MY FRIEND LOGIC-X

Neste capítulo será abordado o processo de desenvolvimento da aplicação educacional My Friend Logic-X, no qual tem o objetivo de auxiliar pessoas que possuem dislexia em disciplinas de lógica de programação utilizando da gamificação como abordagem. A aplicação aborda os principais tópicos de lógica de programação de forma audiovisual, para que dessa forma os elementos textuais possam ser minimizados, em que os mesmos são representados e adaptados para elementos visuais, visando a concepção sólida do conhecimento em questão para pessoas com dislexia. Como dito, a aplicação é voltada para pessoas que estudam lógica de programação e que possuem dislexia, visto que a dislexia é um distúrbio de aprendizado que afeta a leitura, com isso, as dificuldades enfrentadas por essas pessoas ao iniciar cursos ou seus estudos em lógica de programação podem ser muitos, no qual todas as linguagens e conteúdos relacionados são passados de forma escrita.

3.1. CONCEPÇÃO DA IDEIA

A escolha do tema surgiu em meio a observação de alunos no qual possuíam dificuldade em compreensão com a lógica de programação relacionada a escrita das linguagens utilizadas. Foi percebido que os mesmos não conseguiam aplicar conceitos básicos por não possuírem conhecimento sólido sobre o tema e também sobre a linguagem utilizada, neste caso, são dois problemas que se unem e formam um conflito para o aprendizado efetivo do tema.

Após estudo e pesquisa sobre o problema, foi relatado a dislexia como um distúrbio de aprendizado que pode afetar a leitura, compreensão e abstração de conteúdos que são passados de forma escrita. Feita a comparação das mesmas dificuldades, foi possível verificar que são semelhantes às que muitos alunos possuem durante o processo de aprendizagem de lógica de programação tradicional, utilizando linguagens de programação. Dessa forma, foi proposta uma aplicação no qual pudesse auxiliar o ensino de lógica de programação de forma audiovisual adequada para uma pessoa que possui a Dislexia, visando contribuir para a melhor compreensão dos conteúdos relacionados em lógica de programação.

3.2. PLANEJAMENTO DA APLICAÇÃO

A aplicação possui mecânicas simples visando que a complexidade pode afetar e confundir o usuário. Dessa forma, a mecânica de interação principal é a passagem de conteúdo em forma de vídeo interativo, em que o usuário poderá interagir com a aplicação no final de cada explicação. O usuário tem uma lista de objetos relacionados com o tema a fim de selecionar o objeto que mais faz sentido com a explicação passada pela aplicação. Para as telas de explicação de conteúdo, o tempo conta a partir do momento em que a lista é liberada para interação, a pontuação é representativa visto que nessa tela não é possível ser penalizado por errar o item associado, feito dessa forma, visa a compreensão e acerto da interação relacionada com o tema em questão. Para as telas de desafio, o conteúdo não é narrado, apenas uma representação relacionada ao tema é proposta e o contador de tempo é iniciado no momento que a lista é liberada para interação, nessa mesma tela a pontuação é penalizada caso o usuário erre o item relacionado ao tema, mas visto que os desafio são relativamente mais curtos que os tópicos de passagem de conhecimento, é possível o usuário reiniciar ou tentar novamente o desafio sem custo de tempo.

Toda a pontuação, tanto da tela de passagem de conhecimento quanto dos desafios são serializadas em arquivos JSON[4], no qual é utilizado o armazenamento interno do dispositivo para alocar o arquivo que contém a pontuação do usuário.

Os dados de configurações também são serializados em arquivos JSON e são alocados no armazenamento interno do dispositivo.

Inicialmente a plataforma selecionada foi a de dispositivos móveis com sistema operacional Android, visto que *smartphones* com o sistema são acessíveis em relação aos computadores e dispositivos com iOS, visando alcançar amplamente o público alvo.

Para a produção da aplicação foram selecionadas ferramentas e serviços para facilitar e agilizar o desenvolvimento da aplicação.

Inicialmente foi necessário o desenvolvimento de protótipo de telas da aplicação, seguindo os princípios e abordagens para desenvolver uma aplicação voltada a pessoas que possuem dislexia, dos quais as, telas que são compostas por

[4] JSON: Sigla para JavaScript Object Notation. Trata-se de um formato leve de intercâmbio de dados, em que sua estrutura simples baseada em pares chave-valor facilita a transmissão e interpretação de informações.

pouco texto e no lugar imagens auto explicativas no qual representem a opção ou elemento que poderia ser textual, cores que tenha amplo contraste e não possam se misturar ou confundir o usuário, de modo que, cada área possa representar uma funcionalidade ou elemento separado.

Para atender a produção da prototipagem de telas, foi utilizada a plataforma Figma, com ela foi possível montar as telas utilizando vetores, e imagens selecionadas a partir da plataforma Flat Icon[5]. Com isso foi possível desenvolver as telas que posteriormente seriam utilizadas com referência dentro da Unity para o desenvolvimento da aplicação.

Visando desenvolver em menos tempo a aplicação, foi utilizada a plataforma Flat Icon para selecionar os ícones necessários para a aplicação. A plataforma é gratuita e não possui nenhum tipo de taxa para o uso dos elementos baixados.

Para o desenvolvimento da aplicação foi utilizada a Unity, no qual se assemelha a uma IDE[6] convencional. Com a Unity foi possível montar *sprites*, logos, telas, códigos, mecânicas, a lógica por trás das telas, a tela de transição e desenvolver o sistema que tem a responsabilidade de salvar os dados do usuário no dispositivo em uso.

A aplicação utiliza sistemas de áudio para narrar elementos em geral presentes nas telas, assim como também, nos vídeos das telas de interação com usuário. Para isso, foi utilizada a plataforma Eleven Labs que conta com ferramenta de *text-to-speech* ou texto para fala, em que o usuário discorre um texto qualquer e a plataforma transforma em um áudio narrado por inteligência artificial.

Visto que a aplicação utiliza vídeos narrados interativos para a passagem de conhecimento dos temas, foi utilizado o editor de vídeo *open source* Kdenlive[7]. Com essa ferramenta foi possível montar os vídeos junto com os arquivos de áudio exportados do Eleven Labs e os ícones e logos selecionados na plataforma Flat Icon, para assim, compor o vídeo final de forma audiovisual sem que tenha a necessidade textual para passagem de conteúdo.

Para criar e editar códigos na Unity é necessário escolher um editor de código, dessa forma, foi selecionado o VSCode[8] por sua ampla compatibilidade e

---

[5] Flat Icon: Plataforma de ícones vetoriais com design plano, ideal para projetos gráficos e web.
[6] IDE: Ambiente de Desenvolvimento Integrado, facilita a codificação e depuração de software.
[7] Kdenlive: Software de edição de vídeo de código aberto e intuitivo.
[8] VSCode: Editor de código aberto da Microsoft, amplamente utilizado por desenvolvedores.

extensibilidade com demais ferramentas, com ele foi possível integrar à Unity com facilidade.

Depois de selecionar e separar as ferramentas, assim como o desenvolvimento dos elementos iniciais, foi desenvolvido um documento chamado *Game Design Document*[9], onde o objetivo do mesmo é documentar as mecânicas básicas de jogabilidade, modelo de aplicação, plataformas, escopo do projeto, história e jogabilidade, descrição geral e qualquer outra informação referente a aplicação, foi necessário a elaboração este documento visto que a aplicação utiliza da gamificação como abordagem principal.

3.3. DESIGN DA APLICAÇÃO

A plataforma Figma foi utilizada inicialmente para criar conceitos e modelos de tela modulares no qual pudesse atender e ser reutilizada por sub módulos e entre outros tipos de telas. Se diz modular, para a tela que utiliza elementos no qual podem ser retirados e incrementados sem alterar a estrutura ou funcionamento da aplicação. As telas e mecânicas desenvolvidas visam a produção de elementos que pudessem ser reutilizados independente de que seja sua atribuição.

Foi abordado os princípios e meios para desenvolver a aplicação adaptada para pessoas que possuem dislexia, do qual, as telas desenvolvidas são compostas por ícones e logos que traduzem em forma de imagem o que poderia ser textual, para auxiliar a compreensão das representações cada elemento possui uma áudio narrado que foi desenvolvido utilizando o Eleven Labs, no qual o usuário ao tocar no elemento o áudio é disparado narrando o que seria aquele item selecionado.

A figura 3.1 representa os princípios aplicados, em que o primeiro botão com a função de iniciar, no qual ao ser selecionado, inicie a tela de progresso em que o usuário poderá selecionar os tópicos da aplicação e seus sub tópicos referentes.

O segundo botão possui o formato de engrenagem no qual representa as configurações da aplicação que é compostas por opções que possam auxiliar a pessoa com dislexia no ajuste personalizado da aplicações em certos elementos, é um fato que a aplicação se faz necessário elementos textuais em certos itens, e desse modo foi elaborado e implementado opções para aumento e espaçamento da

[9] GDD: Game Design Document, documento crucial que define aspectos essenciais do jogo, como mecânicas, história e estética, usado para orientar no processo de desenvolvimento.

fonte a fim de auxiliar e facilitar a leitura desses elementos textuais, como é mostrado na figura 3.1.

Cada opção ou elemento presente no sistema, que é representado visualmente por uma imagem ou logo, é cuidadosamente precedido por um arquivo de áudio. Este elemento de áudio tem a função essencial de narrar e explicar a funcionalidade associada àquela opção específica, proporcionando assim um entendimento mais claro e detalhado.

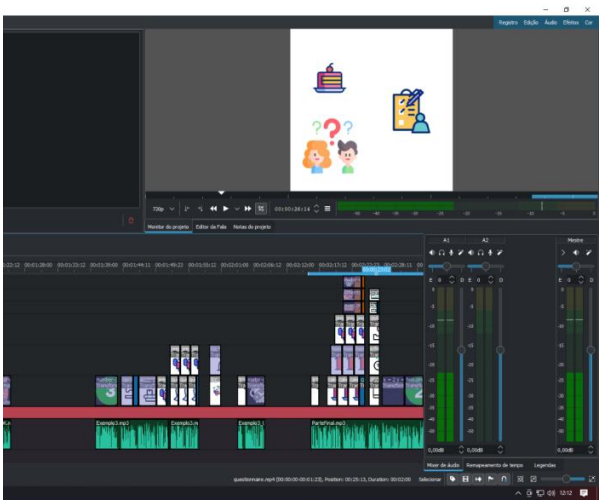
Este método de associação de áudio com representações visuais garante uma experiência de usuário intuitiva e acessível, além disso, essa abordagem multimodal, combinando estímulos auditivos e visuais, reforça o aprendizado e a compreensão das funções de cada elemento dentro do sistema, contribuindo para uma interação eficiente e gratificante.

Figura 3.1 - Tela inicial e configurações.

Fonte: Próprio autor.

A aplicação utiliza cores que distinguem cada elemento na tela, de forma que, os mesmos possam não confundir o usuário e assim levar a clareza e certeza de cada item individualmente, no qual, as cores foram selecionadas para ter contraste em relação às demais, assim como a utilização de espaçamento favorece a distinção de cada elemento na tela. A cor de fundo da tela inicial foi atribuída ao azul claro, no qual transmite um ambiente agradável e ainda mantém o contraste de distinção de elementos para a composição da tela.

Figura 3.2 - Timeline no software Kdenlive.

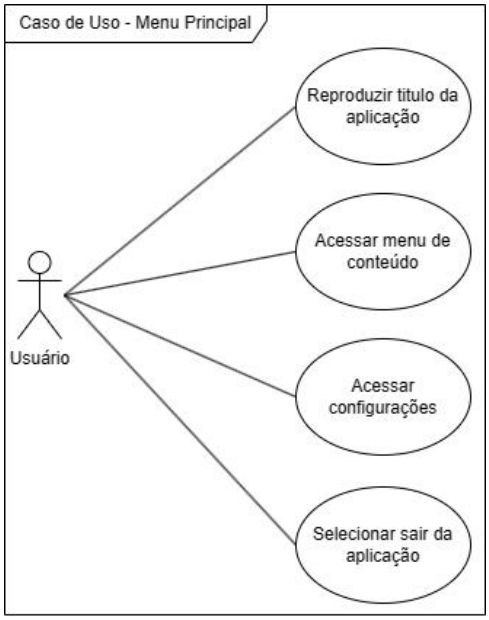


Fonte: Próprio autor.

O *storyboard* da aplicação foi feito proceduralmente junto a produção de vídeos, ao lado da produção do GDD(*Game Design Document*) que se encontra todo o roteiro e instruções de desenvolvimento dos vídeos de conteúdo do tema referente, na figura 3.2 é possível analisar a produção da linha do tempo lógica de um tópico dentro do editor de vídeo Kdenlive.

Para o fluxograma de utilização, foram identificados 3 fluxo de cenários nos quais o usuário poderá executar durante o uso da aplicação. Dos quais foi utilizado e observado a partir do diagrama de caso de uso da figura 3.3.

Figura 3.3 - Diagrama de caso de uso do menu principal.

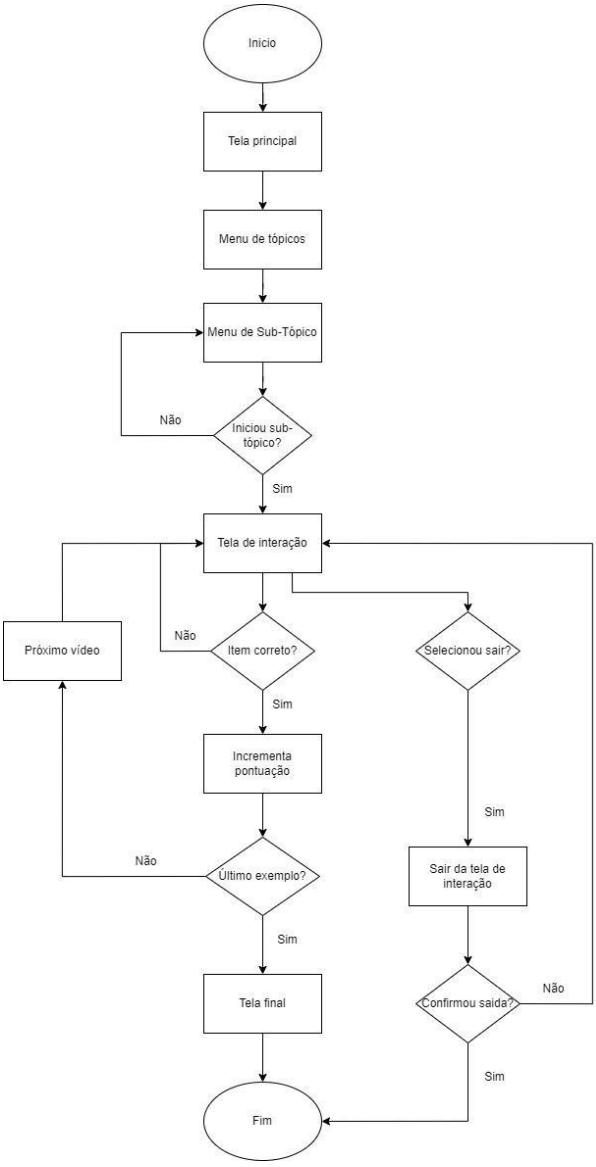


Fonte: Próprio autor.

O primeiro cenário representado pela figura 3.4, mostra o fluxo de uso quando o usuário executa a utilização da aplicação para iniciar um tópico de conteúdo. O mesmo inicia a aplicação, navega pela tela de tópicos, sub-tópicos e seleciona o sub-tópico ou conteúdo de preferência, ao selecionar uma tela de confirmação de conteúdo é exibida, local que o usuário poderá confirmar se quer realmente iniciar o conteúdo ou voltar a tela de sub-tópicos.

Iniciando o conteúdo, o usuário poderá selecionar os itens após a reprodução dos vídeos de passagem de conteúdo, nesse cenário, o usuário não é penalizado caso selecione uma opção errada, onde poderá aprender com erros e sanar dúvidas entre itens fornecidos pela lista.

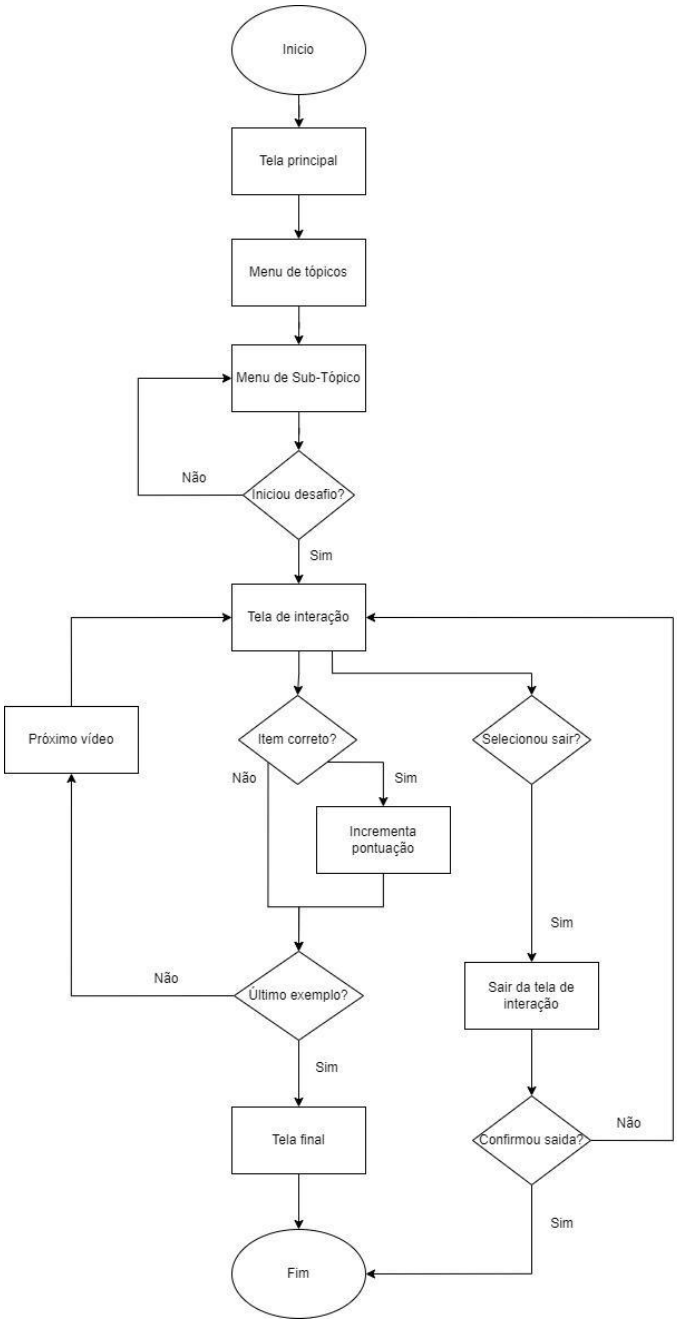
Figura 3.4 - Fluxograma para iniciar tópico de conteúdo.



Fonte: Próprio autor.

O segundo cenário é representado pela figura 3.5, no qual representa o fluxo para a execução da utilização da aplicação para iniciar um desafio. O usuário inicia a aplicação assim como na figura 3.4, a principal diferença é na tela de interação no qual o usuário pode ser penalizado, que seria não ganhar os pontos referentes quando errar um item selecionado relacionado ao tema aplicado.

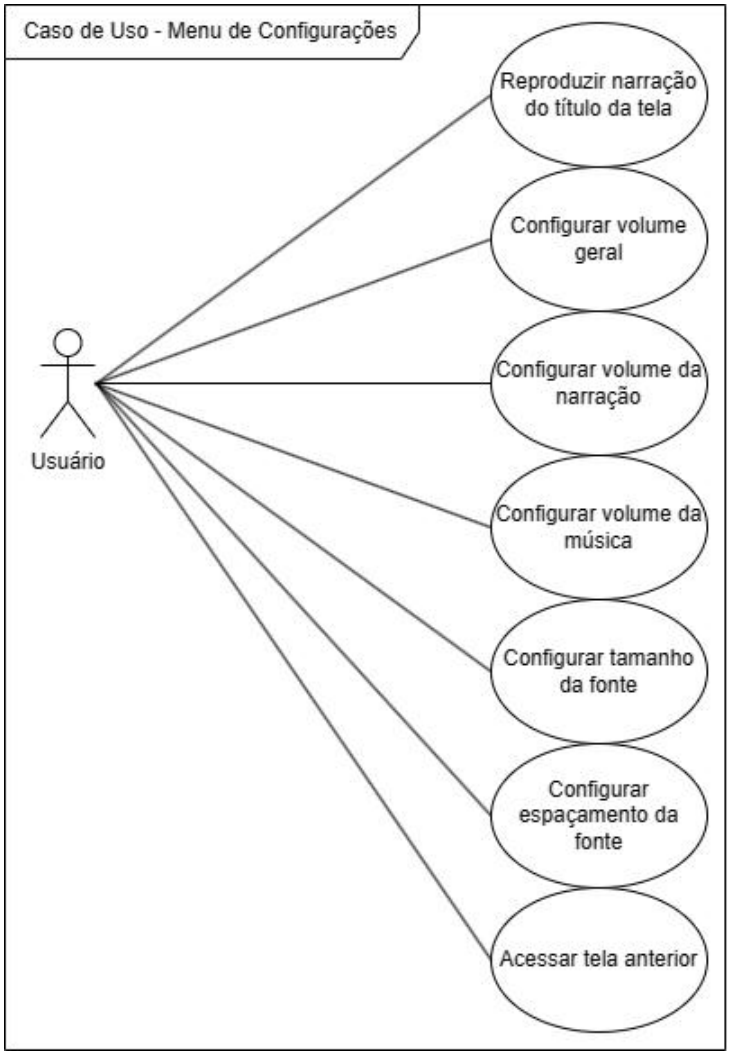
Figura 3.5 - Fluxograma para iniciar desafio.



Fonte: Próprio autor.

O terceiro e último cenário é referente a utilização da aplicação para iniciar ajustes na tela de configuração, em que para o mesmo foi baseado no diagrama de caso de uso representado na figura 3.6, o fluxo criado pode ser visualizado na figura 3.7.

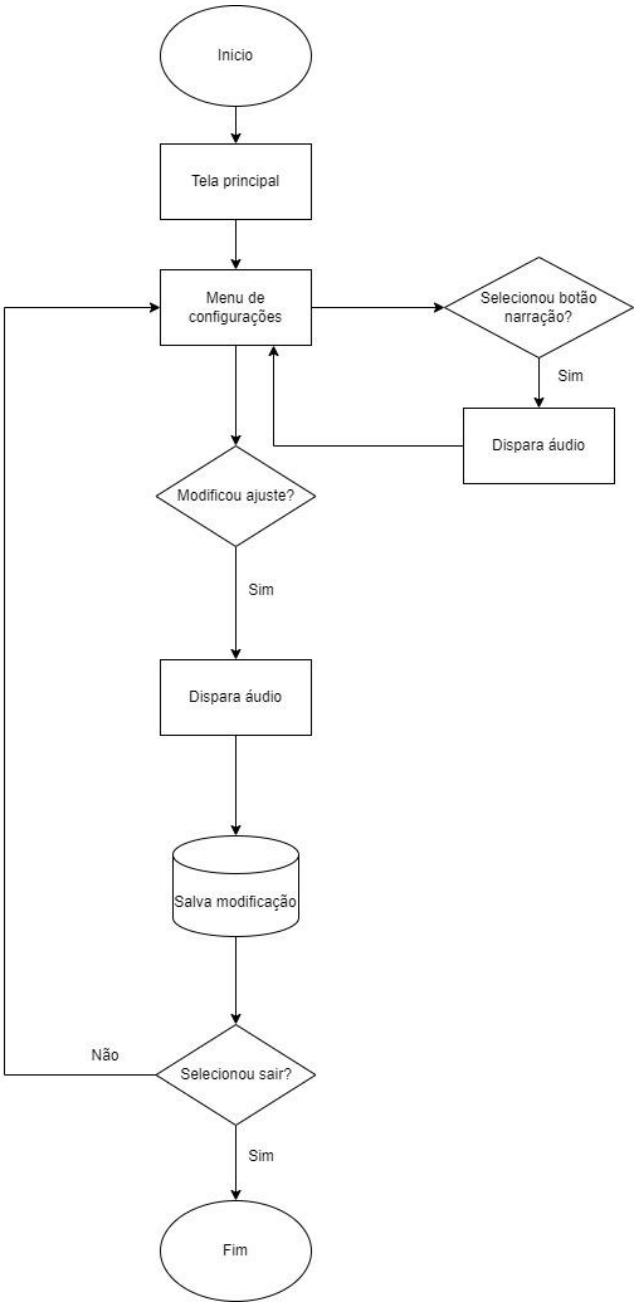
Figura 3.6 - Diagrama de caso de uso do menu de configurações.



Fonte: Próprio autor.

Para os caso de uso deste cenário foram identificados 7 casos da interação do usuário com a aplicação. Os mesmos são representados pelo fluxograma da figura 3.7, no qual todas as preferências de ajuste do usuário são salvas em um arquivo do tipo JSON para serem utilizadas para carregar os dados quando iniciar a aplicação novamente.

Figura 3.7 - Fluxograma para iniciar tela de configurações.



Fonte: Próprio autor.

O usuário inicia a aplicação e na tela inicial seleciona a opção para mostrar a tela de configurações, na mesma quando o usuário ajustar uma opção, um áudio referente a opção deve ser disparado narrando qual opção foi ajustada e posteriormente deve salvar os ajustes serializando em um arquivo JSON e utilizando o armazenamento do dispositivo do usuário para guardar o arquivo.

3.4. PRODUÇÃO E DESENVOLVIMENTO DA APLICAÇÃO

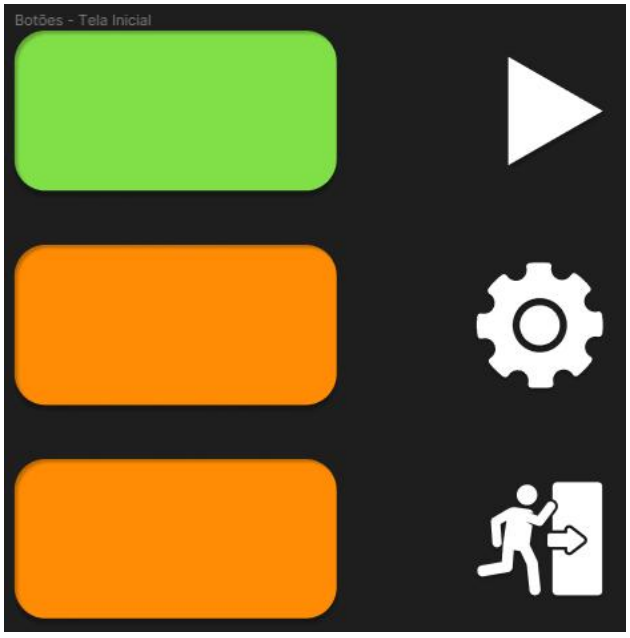
No presente tópico será discutido e mostrado o processo de produção da aplicação proposta neste trabalho. Será abordado o processo de design das telas, produção textual de roteiro, produção de áudio, vídeos e *scripts* para implementação das mecânicas necessárias da aplicação.

3.4.1. Telas da aplicação

Nesta seção será mostrada como as telas definitivas da aplicação foram desenvolvidas utilizando as ferramentas anteriormente propostas.

Todas as telas foram reutilizadas da plataforma Figma no qual oferece a possibilidade de criar objetos vetorizados, com isso, foi possível exportar *sprites*, imagens, montagens que pudessem compor a tela da aplicação dentro da Unity.

Figura 3.8 - Componentes de um botão.



Fonte: Próprio autor.

Na figura 3.8 é possível visualizar como um botão foi montado e por quantos elementos ele é composto. Foi possível criar formas simples e atribuir imagens ou ícones a essas formas, formando o que seria um botão para a tela da aplicação.

Figura 3.9 - Botões finalizados.

Fonte: Próprio autor.

Na figura 3.9 é possível visualizar como é composto os botões da tela inicial montados. Essa mesma abordagem se utiliza para outros elementos, como cabeçalhos e faixas, na figura 3.10 é possível visualizar essa abordagem em outro elemento.

Figura 3.10 - Estrutura de botões faixa.

Fonte: Próprio autor.

Atualmente a proporção de tela dos *smartphones* atuais é de 16:9, dessa forma todas as telas foram produzidas levando em consideração a resolução dessa

atual proporção como mostra a figura 3.11, visando ser compatível com a maioria dos dispositivos disponíveis no mercado e em alcance do público alvo.

Figura 3.11 - Proporção de tela

Fonte: Próprio autor.

Todas as telas produzidas são compostas por faixas, ícones e botões em que foi utilizado o processo de montagem exemplificado acima. Foram desenvolvidas no total de 10 interfaces de *mockup* para demonstrar o planejamento de telas da aplicação. Dos mesmos 4 telas utilizam estrutura similar no qual serão reaproveitadas de forma modular dentro da Unity no processo definitivo de montagem.

Utilizando a Unity para desenvolvimento definitivo das telas, foi necessário exportar cada elemento dentro do Figma em formato de imagem PNG[10] para que o arquivo mantivesse o Canal *Alpha*[11] de transparência. Todos os elementos foram exportados obedecendo o processo da regra de negócio de cada elemento. Na

[10] PNG: Formato de imagem com suporte a transparência, comum na web e design gráfico.
[11] Canal Alpha: Em imagens, controla a transparência de pixels para sobreposição suave de elementos.

figura 3.12 é possível visualizar, como o botão foi exportado obedecendo a sua regra de montagem dentro da Unity, nesse caso, o mesmo deve ser exportado sem elementos textuais pelo fato de que dentro da Unity os textos serão inseridos manualmente para obedecer a estrutura do módulo que será reaproveitado para produzir outro elemento.

Figura 3.12 - Botão faixa exportado.

Fonte: Próprio autor.

Como mencionado anteriormente, todos os *sprites* devem ser exportados em formato de imagem PNG para que possam manter o canal *Alpha* de transparência, isso se faz necessário em botões, faixas ou elementos que necessitam de transparência para mostrar o fundo de imagem da tela. Na figura 3.13 é possível visualizar como é representado um *sprite* no qual contém o canal *Alpha*, é possível verificar não só a transparência como a semi-transparência aplicada ao troféu presente na imagem.

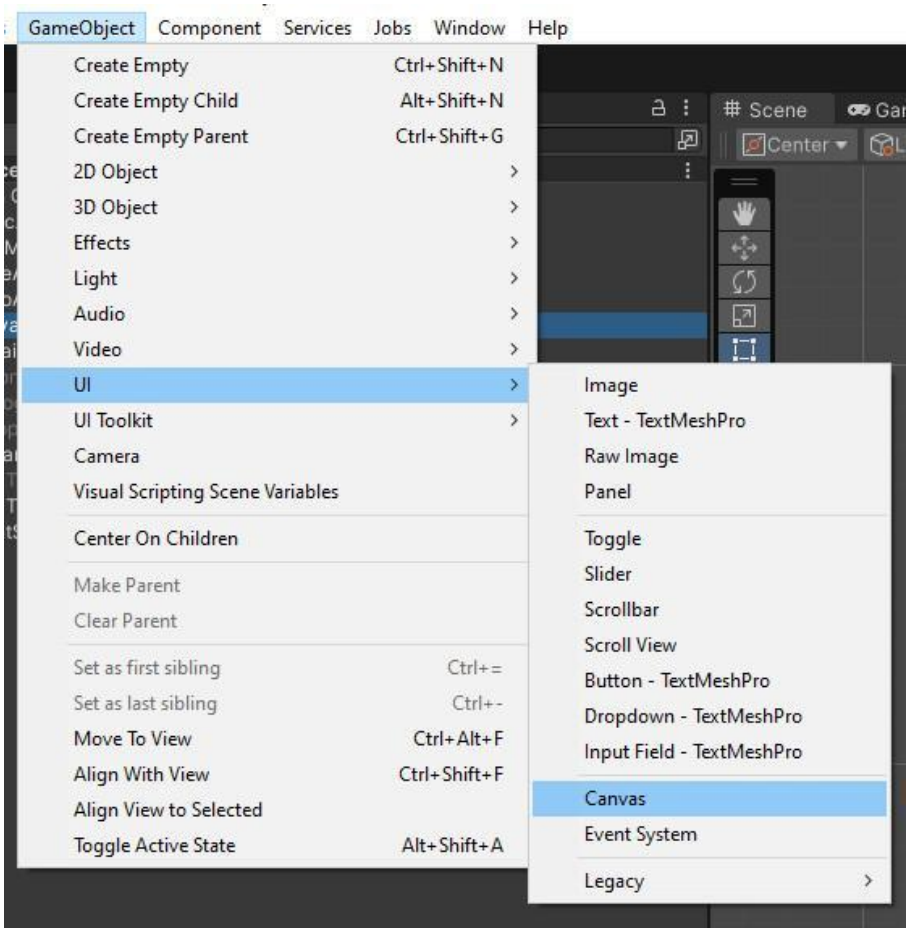
Figura 3.13 - *Sprite* com transparência.

Fonte: Próprio autor.

Com todos os elementos exportados, os mesmos foram direcionados e importados dentro da Unity no qual foi criado um projeto para a aplicação proposta neste trabalho de pesquisa.

No primeiro momento é necessário criar um modelo *canvas*, no qual é uma área onde todos os elementos da interface de usuário devem estar dentro, na figura 3.14 é possível visualizar o caminho dentro da Unity para criação de um *canvas*.

Figura 3.14 - *Sprite* com transparência.

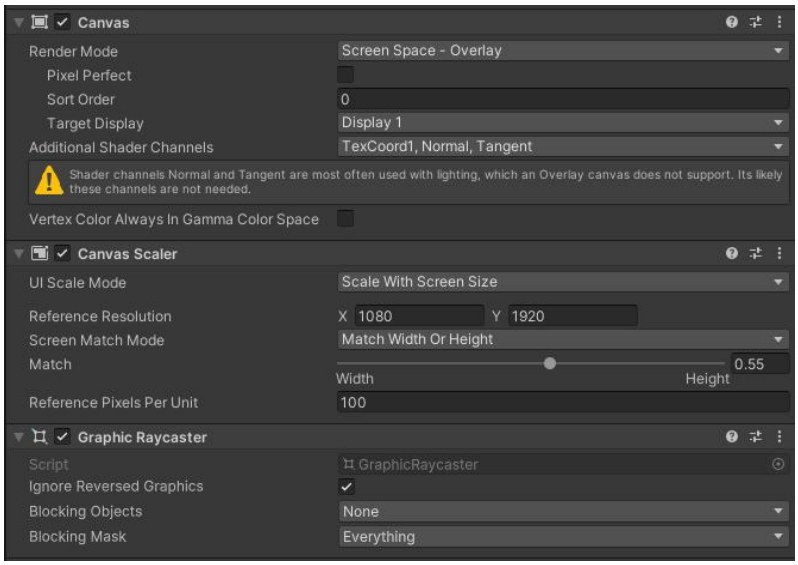


Fonte: Próprio autor.

O *canvas* é um objeto de jogo com um componente de mesmo nome, todos os elementos da interface de usuário devem ser filhos desse *canvas*. O *canvas* também possui outros componentes nos quais são dependentes do componente *canvas*, o *canvas scaler* é necessário para configurar a proporção interface de usuário para a tela do dispositivo no qual a aplicação está sendo desenvolvida, no contexto da aplicação proposta, é possível visualizar na figura 3.15 que o *canvas scaler* está configurado para uma resolução de referencia de 1080 por 1920 pixel, no

qual foi definida e mostrada anteriormente, além disso o tipo de escala está configurado para altura e largura, o que dá a possibilidade de configurar com essa proporção.

Figura 3.15 - Componentes de um *canvas*.



Fonte: Próprio autor.

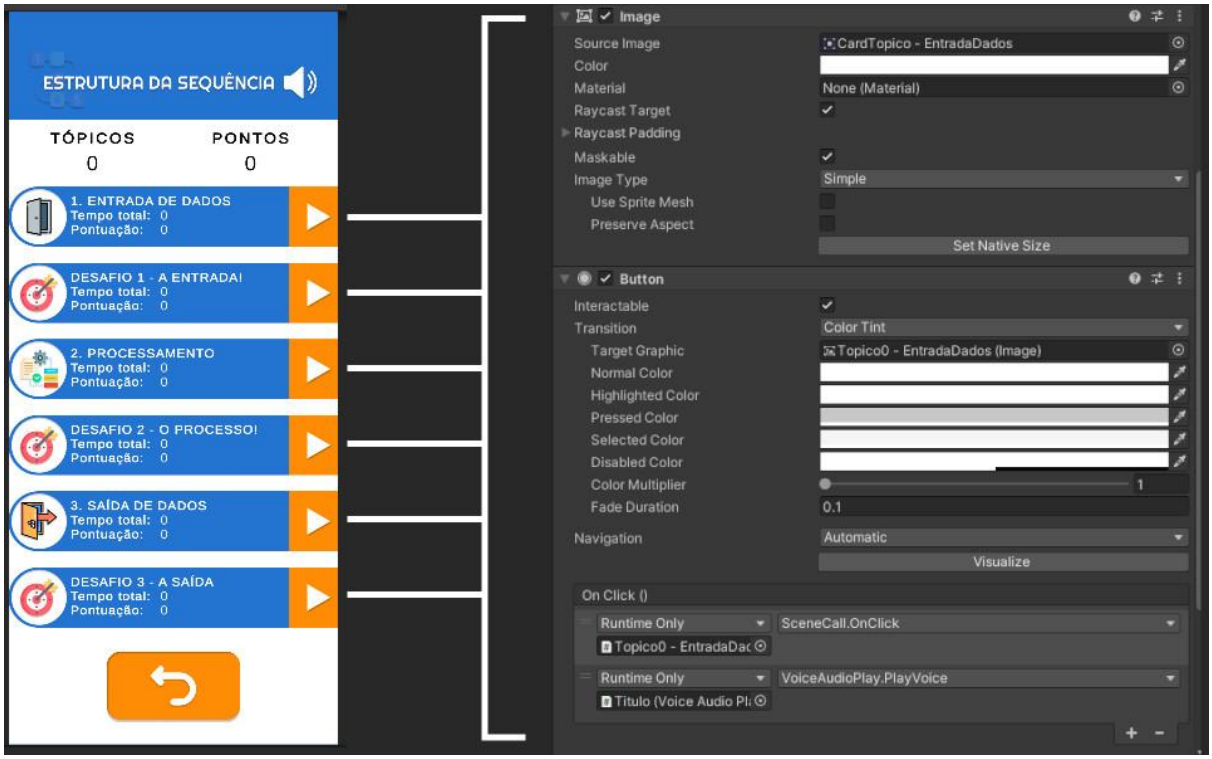
Em resultado, dentro da Unity, a área de trabalho para os elementos da interface de usuário ficam com o seguinte formato apresentado na figura 3.16, assim, a proporção atende ao padrão encontrado nos smartphones atuais.

Figura 3.16 - Componentes de um *canvas*.

Fonte: Próprio autor.

A Unity possui um objeto de jogo para configurações de botões, esses objetos são clicáveis no qual ao ser clicado podem executar qualquer comando ou função. O botão tem um *background* que é utilizado para preenchimento da área definida para o mesmo, dessa forma, é utilizado os *sprites* exportados do figma sem o conteúdo textual. Todos os objetos de botões possuem os componentes representados na figura 3.17. No componente *button* na sessão *OnClick*() no qual é um função desse componente, é possível associar um objeto no qual possui uma função ou definições de entidade como estar ativo ou não, e ao clicar no botão a função ou ação associado ao *OnClick*() é executada. A figura abaixo mostra o objeto Topico0 - EntradaDados associado, onde é chamado do script *SceneCall* a função *OnClick* que carrega a tela referente a entrada de dados.

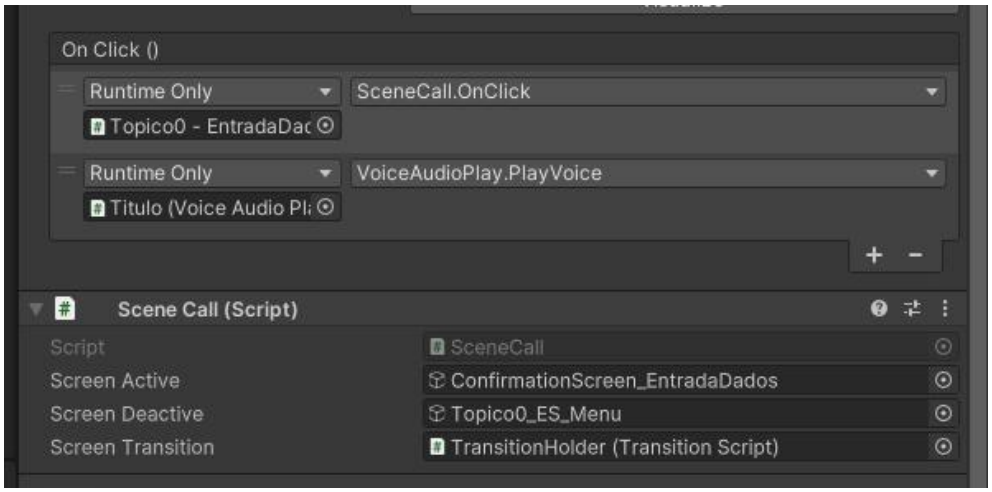
Figura 3.17 - Componentes de cada botão faixa.



Fonte: Próprio autor.

Na figura 3.18 é mostrado o script *SceneCall* que tem a função de chamar uma tela referente, existe um campo para um objeto que será desativado, e um ativado, assim que a função for chamada o processo de transição é feito. O *script* pode ser reaproveitado para qualquer botão, sendo que os objetos são associativos e não constantes.

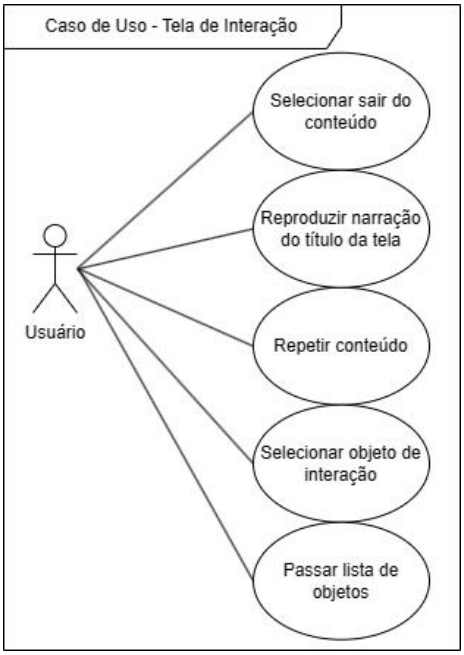
Figura 3.18 - *Script SceneCall* associado a um botão.



Fonte: Próprio autor.

A tela de interação com o usuário que é feita de forma a ser reutilizada para os demais tópicos propostos posteriormente. Na tela, será transmitido o vídeo de conteúdo para o usuário, no mesmo é apresentado um problema do mundo real e no fim é feita uma pergunta de qual item da lista pode solucionar o problema apresentado, posteriormente é explicado o por que aquele item é o ideal e como o mesmo é relacionado ao conteúdo passado. A utilização dessa interface foi baseada no caso de uso representado no diagrama da interface de interação no qual pode ser visualizado na figura 3.19.

Figura 3.19 - Diagrama de caso de uso da tela de interação.



Fonte: Próprio autor.

Na figura 3.20, é representada a tela de interação no tópico de estrutura da sequência para entrada de dados dentro da Unity, no vídeo é apresentado um processo de fabricação de bolo dos quais possuem entrada de dados que são os ingredientes, o usuário deve selecionar na lista qual dos item é um ingrediente que representa a entrada de dados.

Figura 3.20 - Tela de interação na Unity.

Fonte: Próprio autor.

Dessa forma é feita a passagem de conteúdo apresentando um problema cotidiano, questionando qual o item relacionando a solução do mesmo, e por fim, explicando a relação da solução com os conteúdos iniciais de lógica de programação.

Nesta seção foram apresentados os principais processos para abordagens e criação das telas da aplicação proposta neste trabalho, além de como funciona o escalonamento de proporção e a chamada de telas.

3.4.2. Produção de roteiro textual da aplicação

Esta seção tem como objetivo mostrar a importância do texto de roteiro implementado no *Game Design Document*, assim como o processo de planejamento, desenvolvimento e sua respectiva estruturação. O Documento de Design de Jogo ou *Game Design Document* é um documento altamente detalhado que tem a finalidade de servir como guia de referência para o desenvolvimento de um jogo. Ele é utilizado por toda a equipe de desenvolvimento, como por programadores, artistas, produtores e qualquer outro membro da equipe para garantir que todos estejam no mesmo nível em relação à visão e aos detalhes do projeto de jogo. Dado isso, é de suma importância que o roteiro e os detalhes do mesmos estejam implementados no GDD(*Game Design Document)*[12].

Abaixo é possível visualizar a estrutura utilizada para detalhar o roteiro de um tópico da aplicação, no qual cada sub tópico é precedido de um desafio que propõe uma atividade relacionada ao mesmo tema mas sem narração, com o objetivo de exercitar o raciocínio lógico referente ao problema atribuído. A aplicação passa o vídeo de conteúdo elaborado como o roteiro desenvolvido, na finalização é explicada a relação dos itens selecionados com o conteúdo de lógica de programação.

Estrutura da sequência - Tópicos:

- Entrada de dados
- Desafio 1 - A Entrada
- Processamento
- Desafio 2 - O processo
- Saída de dados
- Desafio 3 - A saída

Exemplos para ensino:

- **Bolo**
    - Entrada: Ingredientes
    - Processamento: Equipamentos de preparo
    - Saída: Bolo

[12] GDD: Game Design Document, documento crucial que define aspectos essenciais do jogo, como mecânicas, história e estética, usado para orientar no processo de desenvolvimento.

- **Construção**
    - Entrada: Materiais de construção
    - Processamento: Pedreiro/Máquinas
    - Saída: Casa
- **Veículo**
    - Entrada: Peças
    - Processamento: Montagem
    - Saida: Moto
- **Lista**: Ovos, Cimento, Motor, Batedeira, Betoneira, Máquina industrial, Bolo, Casa, Moto.

Roteiro:

Tópico 1 - Entrada de Dados:

A entrada de dados é a primeira etapa da estrutura da sequência, é nessa etapa que os dados necessários para uma operação são fornecidos. Dependendo do contexto, qualquer tipo de dados pode ser fornecido.

Vamos ao primeiro exemplo da entrada de dados! Imagine que você quer preparar um bolo especial, em primeiro momento é necessário que você separe os ingredientes. Consegue me dizer qual item da lista abaixo pode compor os ingredientes necessários para preparar o bolo?
Isso mesmo! O item selecionado faz parte dos ingredientes para preparar o bolo.

Vamos para o segundo exemplo da entrada de dados! Na construção de uma casa, são necessários vários tipos de materiais, onde cada um tem um objetivo específico. Consegue me dizer qual dos itens da lista é um material que pode compor a lista de materiais necessários para a construção da casa?
Isso mesmo! O item selecionado faz parte dos materiais necessários para a construção da casa.

Vamos para o terceiro e último exemplo da entrada de dados! Em uma fábrica onde são montados motos e veículos, cada um tem um tipo de peça que vai ser utilizada e vai compor o veículo final.
Consegue me dizer qual item da lista pode ser uma peça para compor o veículo?

Isso mesmo! O item selecionado faz parte das peças que compõem o veículo.

Finalização:
Excelente!!! Você entendeu perfeitamente que para algo ser preparado, construído ou montado, devem ser fornecidos os materiais correspondentes e necessários. Assim é a entrada de dados, dados são fornecidos para posteriormente serem utilizados na segunda etapa da estrutura da sequência que é chamada de processamento, que veremos no próximo tópico.

Desafio 1: Utiliza a mesma mecânica mas com exemplos diferentes do explicado e sem narração.

Tópico 2 - Processamento:
O processamento é a segunda etapa da estrutura da sequência, é nessa etapa que os dados fornecidos na entrada são operados. Dependendo do contexto, qualquer tipo de atividade pode ser feita com os dados fornecidos.

Vamos ao primeiro exemplo do processamento! Para preparar um bolo, foram fornecidos vários materiais, como ovos, farinha e fermento. Mas os ingredientes não vão se misturar sozinhos, são necessárias ferramentas para iniciar o processo de preparo do bolo.
Consegue me dizer qual item da lista é uma ferramenta necessária para preparar a mistura do bolo?
Isso mesmo! A batedeira vai ser utilizada no processamento dos ingredientes fornecidos para preparar a mistura do bolo.

Vamos para o segundo exemplo do processamento! Para construir uma casa, foram entregues os materiais necessários para compor a casa, mas a casa não vai ser construída sozinha. São necessárias pessoas capacitadas e essas pessoas precisam de ferramentas para iniciar o processo de construção da casa.
Consegue me dizer qual item da lista é uma ferramenta para os trabalhadores iniciarem a construção?
Isso mesmo! A betoneira vai ser utilizada no processo de construção da casa, utilizando os materiais fornecidos para preparar o cimento.

Vamos para o terceiro e último exemplo do processamento! Para montar veículos, foram entregues peças para compor os mesmos, mas essas peças não se encaixam sozinhas, são necessárias ferramentas para iniciar o processo de montagem dos veículos.
Consegue me dizer qual item da lista é uma ferramenta que pode iniciar o processo de montagem dos veículos?
Isso mesmo! Esse robô industrial vai dar início ao processo de montagem dos veículos utilizando as peças fornecidas.

Finalização:
É isso aí! Você entendeu perfeitamente que para alcançar um objetivo, resultado ou um produto, é necessário algum processo de preparo, construção ou qualquer outra ação que possa utilizar os dados de entrada fornecidos. Dessa forma, após o processamento teremos um resultado final que é a saída de dados, que veremos no próximo tópico.

Desafio 2: Utiliza a mesma mecânica mas com exemplos diferentes do explicado e sem narração.

Tópico 3 - Saída
A saída de dados é a terceira e última etapa da estrutura da sequência, é nesta parte onde temos o retorno do processamento de dados. A etapa é simples, mas ela tem uma condição, ao fornecer dados para serem processados para um resultado esperado, a saída deve ser exatamente esse resultado esperado. Caso contrário, existe algo errado no processamento.

Dessa forma, vamos para o primeiro exemplo de saída de dados! Após fornecer ingredientes para preparar um bolo, o processo de preparo foi iniciado utilizando morangos, no fim o bolo foi levado ao forno.
Consegue me dizer qual item da lista representa o bolo preparado?
Isso mesmo! Um bolo de morango foi o resultado, bem simples, né?

Vamos para o segundo exemplo de saída de dados! Após fornecer o material de construção dos quais foram tijolos, cimento e outros materiais. Uma casa incrível foi levantada.
Consegue me dizer qual item da lista representa o tipo de casa construída?
Isso mesmo! Uma casa de alvenaria. Bem tranquilo, né?

Vamos para o terceiro e último exemplo de saída de dados! A fábrica estava a todo vapor montando veículos após as peças serem fornecidas. Mas o tipo de peças fornecidas não pareciam ser de carros.
Consegue me dizer qual item da lista representa o tipo de veículo montado?
Isso mesmo! Uma moto, essa foi mais fácil, né?

Finalização:
Você conseguiu! Entendeu perfeitamente como funciona a estrutura da sequência, composta por entrada, processamento e saída de dados. Agora, que tal um desafio? Essa é só com você, o desafio sobre saída de dados te aguarda.

Desafio 3: Utiliza a mesma mecânica mas com exemplos diferentes do explicado e sem narração.

A estruturação e desenvolvimento do roteiro acima foi elaborada a partir do estudo do tópico proposto na aplicação, dessa forma, foram feitas relações com problemas do mundo real no qual o usuário deve interagir com a aplicação para solucioná-los. No quadro abaixo é possível visualizar os tópicos e subtópicos dos mesmos propostos para aplicação, qual para cada um foi produzido texto de roteiro e detalhes referentes.

Quadro 3.1 - Material de roteiro produzido.

Tópico	Subtópicos
Estrutura da sequência	Entrada de dados; Processamento; Saída de Dados.
Controle de fluxo	Estrutura de controle condicional; Estrutura de repetição.
Tipo de dados	Inteiro, flutuante e texto; Booleano, caractere e nulo.
Variáveis	Tipo e Valor.
Operadores	Operadores aritméticos; Operadores de comparação.
Funções	Parâmetros, valor de retorno e corpo da função.

Fonte: Próprio autor.

Nesta seção foi mostrado a estrutura do texto de roteiro dos tópicos propostos para a aplicação, assim como a utilidade e finalidade do *Game Design Document* onde no qual o roteiro se encontra.

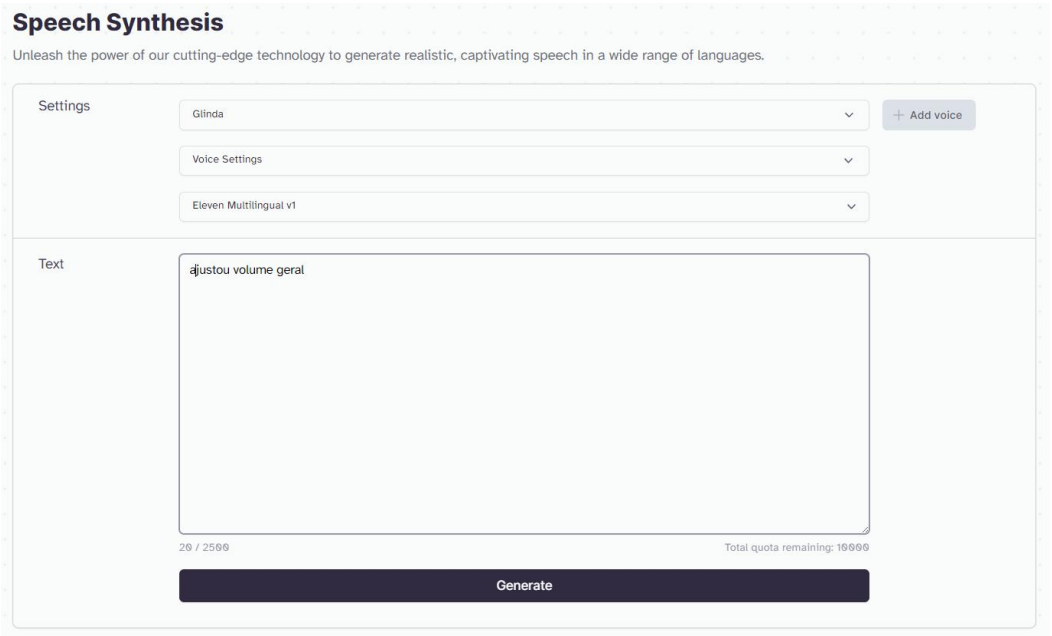
3.4.3. Produção de áudio

Nesta seção é apresentado o processo de produção de áudio utilizando as ferramentas propostas neste trabalho. Para a produção da narração da aplicação, foram propostos dois momentos, um para a produção de arquivos de áudio da narração dos elementos da tela do usuário e os arquivos de áudio dos vídeos de passagem de conteúdo.

Para a produção geral de áudio foi utilizada a plataforma Eleven Labs que conta com ferramenta *text-to-speech*(Texto para Fala) ou s*peech synthesis*(Fala Sintética), no qual o usuário implementa um texto no campo relacionado, ajusta que tipo de voz sequer necessário para seu contexto e pode também ajustar o sentimento da voz, após isso é gerado um arquivo de áudio com todas as

características selecionadas anteriormente pelo usuário, no qual poder ser visualizada na figura 3.21.

Figura 3.21 - Produção de arquivos de áudio para elementos narrados.



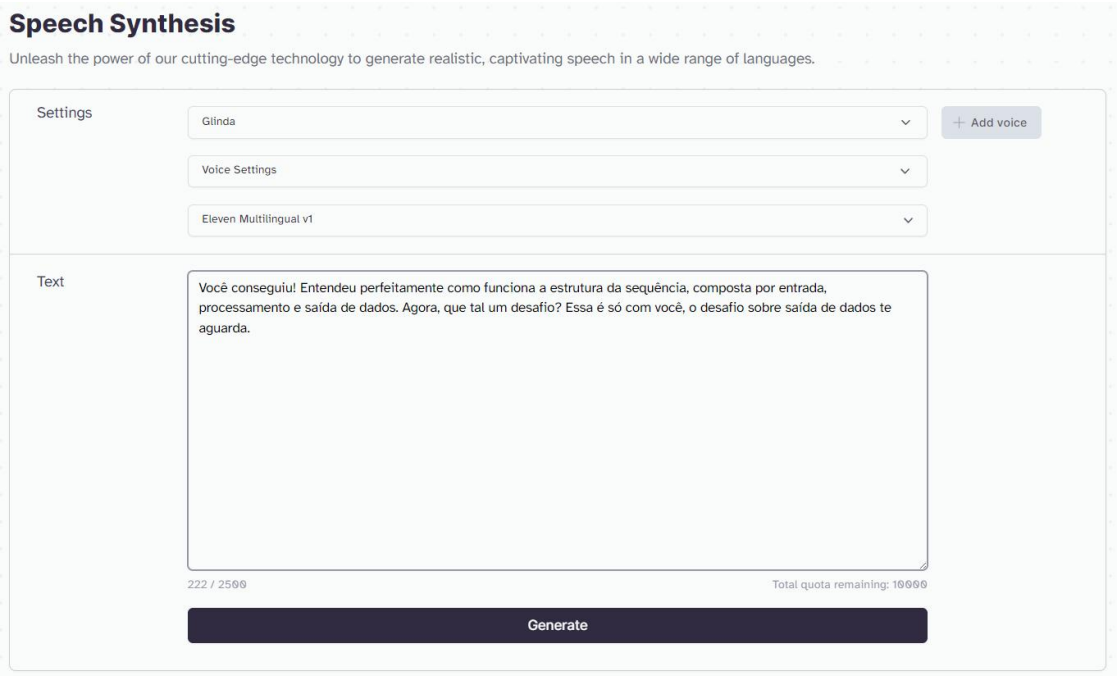
Fonte: Próprio autor.

Para a produção dos arquivos de áudio da narração dos elementos, foram identificados quais elementos que deveriam ser narrados, a figura 3.1 no tópico 3.3 de design da aplicação neste capítulo, mostrou a tela de configurações no qual todos os elementos da tela possuem arquivos de áudio para auxiliar na compreensão do conteúdo textual ou do contexto do elemento. Opções como Áudio Geral, são compostas por áudio que uma voz narra exatamente o que o usuário faz, no caso, o ajuste é para o áudio geral da aplicação. Essa regra de negócio é aplicada para todos os elementos na aplicação que precisam ser narrados e o contexto explicado.

Para a produção dos arquivos de áudio dos vídeos de passagem de conteúdo, primeiramente foi realizada a produção de um documento chamado GDD ou *Game Design Document*, no qual contém informações gerais e específicas sobre o projeto de jogo. Isso se faz necessário pelo fato de que a proposta deste trabalho é o desenvolvimento de uma aplicação gamificada em que a gamificação são características dos jogos em contextos que não são jogos. Ao desenvolver uma aplicação que possui tais características, se torna conveniente desenvolver tal documento mencionado.

De tal forma, foi elaborado o roteiro de todos os cenários de passagem de conteúdo, no qual posteriormente foi utilizado a plataforma ElevenLabs para a produção da voz sintetizada a partir do texto de roteiro produzido, como mostra a figura 3.22.

Figura 3.22 - Tela da plataforma Eleven Labs.



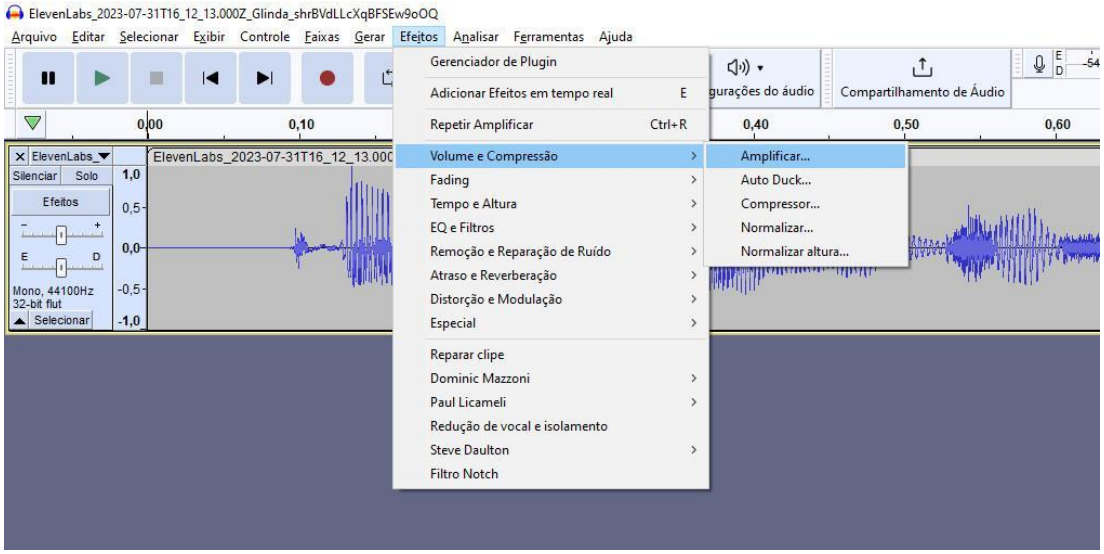
Fonte: Próprio autor.

Para todos esses cenário de áudio para funcionalidades e para os vídeos de passagem de conteúdo, se fez necessário o ajuste de volume dos mesmos, visto que a voz sintetizada produzida é gerada a partir de configuração de sentimentos, no qual certas partes do áudio podem vir com oscilações de volume, no caso, algumas partes a voz sintetizada pode falar mais baixo.

Para o ajuste do volume dos áudio produzidos foi utilizado o software Audacity[13], os áudio foram importados um a um e foi ajustado o volume pela opção Amplificar. A figura 3.23 mostra o processo de amplificação dos áudio produzidos. Após o processo de amplificação, cada áudio foi exportado e importado para a Unity onde foram trabalhados juntamente com *scripts* escritos para disparar os áudios e com os botões ao serem utilizados.

[13] Audacity: Editor de áudio de código aberto, versátil e acessível.

Figura 3.23 - Ajuste de volume em arquivos de áudio no Audacity.

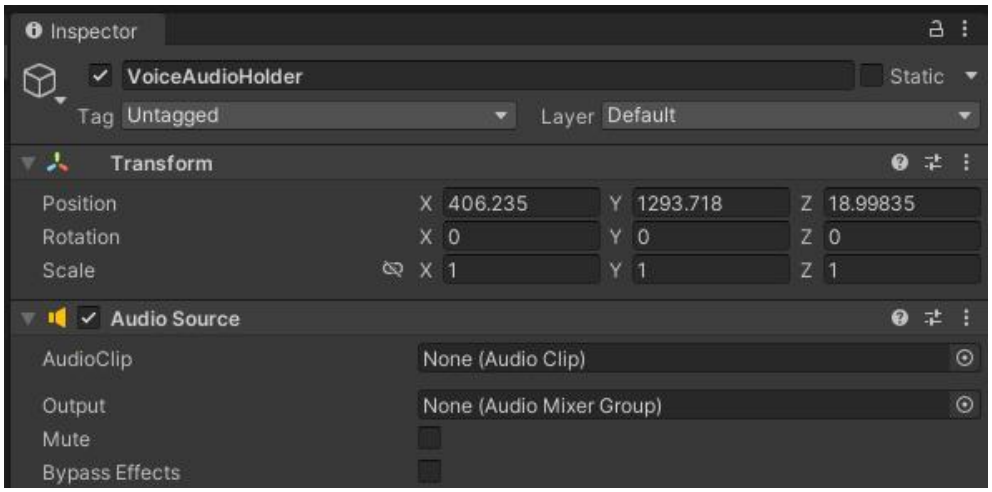


Fonte: Próprio autor.

A aplicação foi planejada para ser modular e reutilizável em que um objeto ou componente pudesse ser utilizado por outro sem ter problemas, no caso do áudio, foi criado um único *player* de áudio que seria responsável por executar todos os arquivos de áudio, mas visto que poderia ser executado apenas um de cada vez. Isso se faz pelo fato de que o usuário poderia utilizar mais de uma opção da mesma vez ou em um curto período de tempo onde não seria suficiente para o áudio disparado finalizar e o outro começar, assim, utilizando essa abordagem um áudio é disparado por vez, se o usuário utilizar uma nova opção, o áudio anterior é interrompido e o novo é disparado.

Dentro da Unity e para a aplicação, esse *player* ou objeto de áudio foi chamado de *VoiceAudioHolder*, no qual tem atribuído o trabalho descrito acima. Na figura 3.24 é possível visualizar a estrutura deste objeto, no qual apenas possui um *AudioSource* ou fonte de áudio, nesse componente existe um campo chamado *AudioClip* onde é atribuído o arquivo de áudio a ser executado, quando um botão é utilizado, o *script* associa o áudio do botão ao *audio source* e dispara o áudio, assim se houver outro áudio em execução anteriormente, ele é parado para o novo áudio ser executado, evitando assim, dois áudios executando ao mesmo tempo.

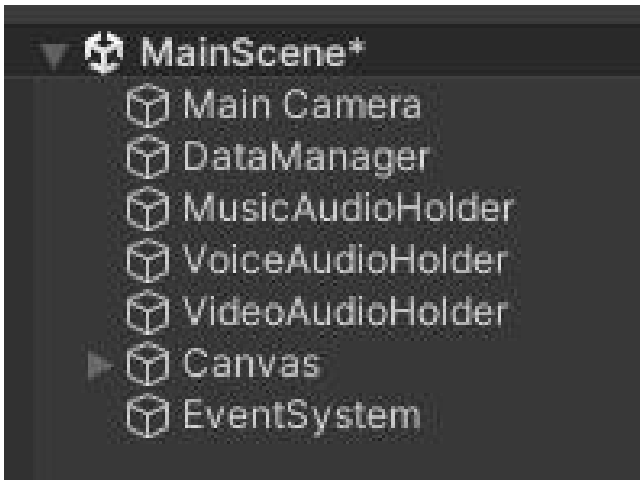
Figura 3.24 - Componente em *VoiceAudioHolder*



Fonte: Próprio autor.

De tal forma, o mesmo ocorre para os outros objetos que são responsáveis pela execução de áudio, que são o *MusicAudioHolder* e *VideoAudioHolder*, onde respectivamente o contexto atribuído é a execução do arquivo de áudio das músicas da aplicação e do áudio dos vídeos de passagem de conteúdo, no qual é possível visualizar na figura 3.25.

Figura 3.25 - Lista de *players* de áudio da aplicação.



Fonte: Próprio autor.

Nesta seção foi apresentado o processo de desenvolvimento dos arquivos de áudio para narração de elementos e vídeo da aplicação proposta neste trabalho, visando atribuir a abordagem audiovisual à aplicação de modo a auxiliar pessoas que possuem dislexia.

3.4.4. Produção de vídeos

Nesta seção será abordado o processo de desenvolvimento dos vídeos de passagem de conteúdo no qual são utilizados na aplicação proposta neste trabalho.

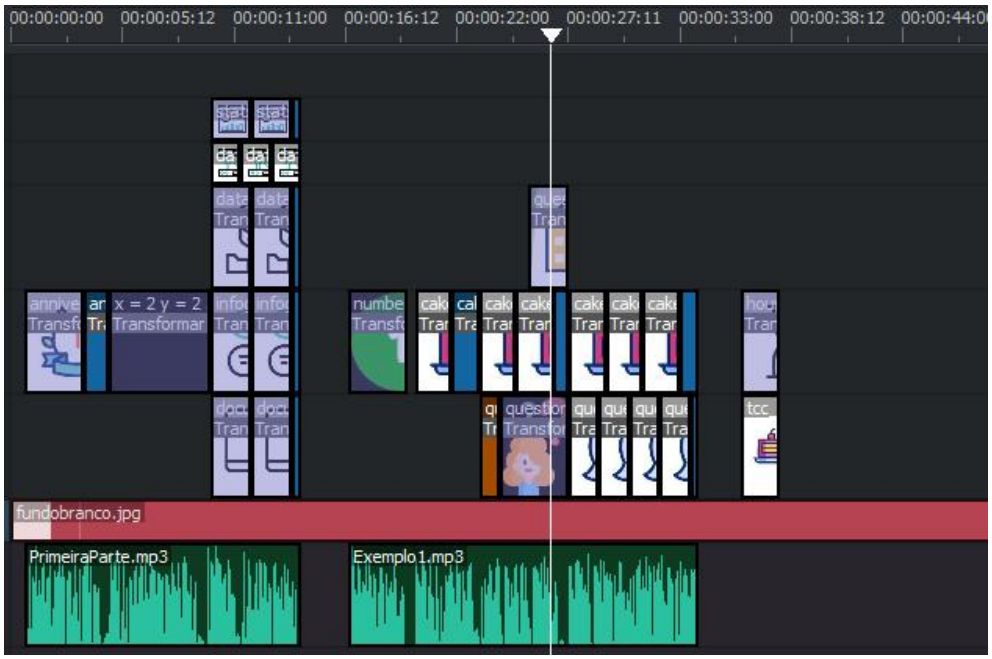
Em primeiro momento, foi necessário desenvolver os roteiros de cada tópico, tema mencionado anteriormente. Dessa forma cada roteiro foi elaborado de forma a atender a abordagem audiovisual onde os exemplos pudessem ser claros e objetivos em relação ao tópico atribuído.

Visto isso, foi necessário coletar os recursos para compor os vídeos de forma visual, deixando de lado qualquer tipo de elemento textual, mas em alguns casos, como exemplificação de equações, foi necessário utilizar o texto. Dessa forma, foi utilizada a plataforma Flat Icon[14] como base de dados para coletar os recursos necessários para atender o roteiro do tópico do tema da aplicação proposta neste trabalho.

Após a conclusão do texto do roteiro, procedeu-se à coleta dos arquivos de áudio necessários para a produção dos vídeos, obtidos através da plataforma Eleven Labs. A figura 3.26 ilustra a montagem da linha do tempo de um tópico específico. Esta linha do tempo, detalhada na mesma figura, representa o tópico relacionado à entrada de dados em estrutura de sequência. Utilizando o software Kdenlive, é possível criar e editar vídeos de maneira eficiente. Este software oferece ao usuário uma interface intuitiva, com uma linha do tempo organizada em canais ou camadas. Cada camada, quando posicionada acima de outra, permite a sobreposição de objetos, criando uma montagem complexa e detalhada. Além disso, o Kdenlive disponibiliza canais distintos para vídeo e áudio, nos quais elementos visuais e sonoros podem ser adicionados e manipulados livremente, permitindo uma grande flexibilidade e criatividade no processo de edição. Esta capacidade de sobreposição e a facilidade de manipulação de diferentes camadas de áudio e vídeo tornam o Kdenlive uma ferramenta essencial para a produção de vídeos de alta qualidade, como demonstrado no exemplo do projeto em questão.

[14] Flat Icon: Plataforma de ícones vetoriais com design plano, ideal para projetos gráficos e web.

Figura 3.26 - Linha do tempo no Kdenlive.



Fonte: Próprio autor.

Para atender o roteiro e a regra de negócio da aplicação da forma como é executado os vídeos, foram planejados e produzidos quatro tipos de vídeos nos quais cada um possui uma responsabilidade diferente na linha do tempo do tópico de cada tema. No quadro 3.2 é possível visualizar as características de cada tipo de vídeo.

Quadro 3.2 - Relação de tipos de vídeos para cada tópico.

Vídeo	**Responsabilidade**
Passagem de conteúdo	Responsável pela passagem de conteúdo dos tópicos relacionados, onde deve, de forma visual passar o conteúdo atribuído em vez de utilizar elementos textuais.
Timer para interação	O timer de interação é um curto vídeo que é repetido até que o usuário selecione a resposta correta.
Resposta correta	Este vídeo mostra a resposta correta

	após o usuário selecionar a mesma. No qual é feito um breve comentário sobre o item selecionado e o presente contexto
Finalização de tópico	Este vídeo é responsável por fazer um resumo dos exemplos passados no tópico de forma clara, para que assim possa fazer uma conclusão do conteúdo passado.

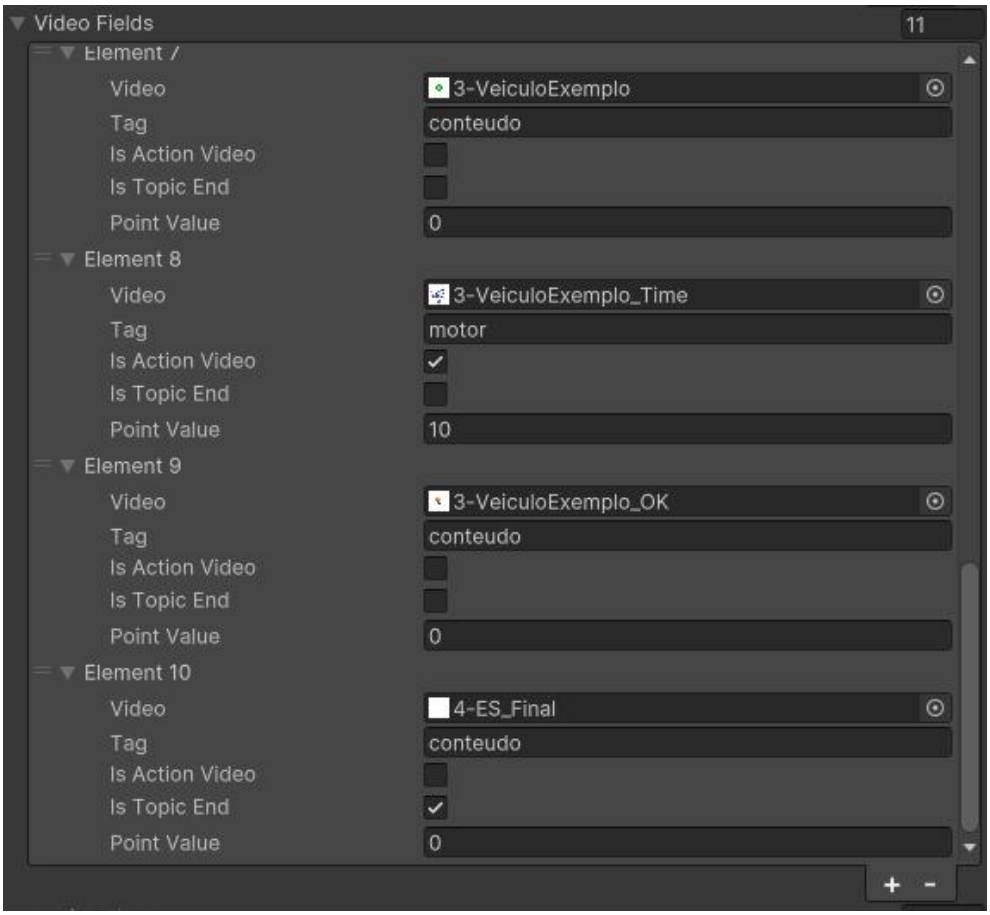
Fonte: Próprio autor.

O próximo passo foi a importação dos materiais de vídeos produzidos para a Unity, dessa forma os materiais foram atribuídos ao *script* de interação no qual foi montado da seguinte forma demonstrado na figura 3.27.

Vídeos de conteúdo não precisam de *flags*, onde são executados normalmente. Vídeos de tempo são marcados como *IsActionVideo* no qual a *flag* é necessária para repetir o vídeo até que o usuário selecione a resposta correta, também é atribuído a pontuação de resposta correta. Os vídeos de resposta correta não são marcados com nenhuma *flag* visto que devem ser executados normalmente. O vídeo de finalização de conteúdo é marcado com a *flag IsTopicEnd* para que o algoritmo possa verificar o último vídeo e chamar a tela de finalização de conteúdo no qual mostra a pontuação e tempo do usuário.

Para todos os vídeos são necessários *tags*, pelo fato que foi criada uma estrutura universal para verificar no momento da interação com usuário se a *tag* do item selecionado é igual ao video de tempo marcado com *IsActionVideo*, na figura 3.27 o elemento 8 é atribuída a *tag* motor, para que no momento que o usuário seleciona o item com a *tag* motor, a resposta correta possa ser validada e passar para o vídeo de resposta correta.

Figura 3.27 - Estrutura *VideoFields* do *script* InteractiveList no editor da Unity.



Fonte: Próprio autor.

Dessa forma, esse processo foi utilizado para a produção dos demais materiais de vídeos obedecendo o roteiro de processo abaixo.

1. Desenvolvimento de roteiro para os vídeo de passagem de conteúdo;
2. Coleta de recursos na plataforma FlatIcon para compor os vídeos;
3. Extração de arquivos de áudio da plataforma ElevenLabs utilizando o texto de roteiro produzido;
4. Montagem e produção dos vídeos seguindo a regra de negócio da aplicação dentro do software Kdenlive;
5. Importação para Unity e montagem lógica seguindo a regra de negócio da aplicação.

Nesta seção foi apresenado o processo de produção dos materiais de vídeo utilizados na aplicação proposta neste trabalho, foi mostrado que o objetivo de cada

vídeo planejado é a passagem de conteúdo de forma relacional ao tema proposto mas de forma visual, deixando de lado elementos textuais.

3.4.5. Desenvolvimento e implementação de scripts

Para o desenvolvimento das funcionalidades e mecânicas da aplicação foi utilizada a linguagem de programação *C Sharp* que é nativa da Unity. Como mencionado anteriormente, a aplicação foi planejada de forma que seja modular e possa aproveitar telas e códigos para serem reutilizados independente da sua aplicação dentro do contexto proposto.

Para a aplicação foram desenvolvidos 18 *scripts* para atender as necessidades do contexto proposto, como mostra o quadro abaixo, no qual o *script InteractiveList* é o principal responsável pelas mecânicas na tela de interação com o usuário.

Quadro 3.3 - *Scripts* implementados para atender contexto da aplicação.

Script	Descrição
ConfigurationData	Classe de estrutura para objeto de arquivo de dados JSON.
DataAllStructure	Classe de estrutura para objeto de arquivo de dados JSON.
FontTextSize	*Script* para definir tamanho do texto da aplicação a partir de valores de 0 à 1 salvos em arquivo JSON de configurações.
InteractiveList	Responsável pelas mecânicas de interação do usuário com a aplicação.
LoadConfigurationData	*Script* responsável por salvar e carregar os dados referentes aos ajustes de configurações.

MenuScript	Classe na qual executa a animação de rolamento de imagem em *background*, é utilizada na tela inicial.
PauseGame	Pausa a aplicação definindo o tempo para 0 ou 1 para resumir.
SaveStageData	Estrutura responsável pelas características do tópico no qual também é responsável por salvar os dados em um arquivo JSON referente ao tópico, esse *script* é utilizado em *TopicDataLoader.*
SceneCall	*Script* no qual ativa e desativa telas em execução com a transição de tela.
SceneLocker	Determina uma *flag* para que se um tópico é desbloqueado
SceneTimer	Trabalha como contador de tempo no momento de interação do usuário com a aplicação nos tópicos em passagem de conteúdo e desafios.
StageInitialConfiguration	Define as configurações iniciais de música para a aplicação.
TopicDataLoader	Responsável por carregar e salvar os dados referentes aos tópicos para tela de subtópicos.
TopicDataStruct	Determina a estrutura dos arquivos JSON para armazenamento.
TopicProgressDataLoader	*Script* responsável por carregar os dados e formatar para a tela de

	progresso.
TransitionScript	*Script* no qual é responsável por executar a animação de transição entre telas.
VideoPlayerStatus	Classe de estrutura que determina o estado do *player* de vídeo.
VoiceAudioPlay	*Script* responsável por disparar arquivos de áudio, é utilizado quando botões são clicados.

Fonte: Próprio autor.

Visto a descrição e responsabilidade de cada *script* desenvolvido, será discorrido a construção e funcionamento dos *scripts* mais importantes no contexto da aplicação.

O *script InteractiveList* é responsável pela execução do conteúdo em vídeo, sistema de pontuação, sistema de erro, sistema para verificar quando o tópico é finalizado, também é responsável por organizar e conter a lista de *sprites* que servem como itens na lista interativa, lista de vídeos que devem ser executados ao decorrer do tópico, e lista de botões nos quais são utilizados para interação com a aplicação.

Para agregar a estrutura de vídeos e suas funcionalidades, dentro do arquivo de *script InteractiveList* foi implementado uma estrutura serializável no qual contém as características necessárias para atender a regra de negócio de cada arquivo de vídeo que deve ser executado. A mesma possui uma variável do tipo *VideoClip* que é importada da API da Unity, no qual é possível atribuir arquivos de vídeos. Possui também um campo de texto chamado *tag* no qual deverá ser atribuído um nome a estrutura de vídeo para ser comparada ao item selecionado pelo usuário posteriormente. Duas variáveis booleanas servindo de *flag* para serem comparadas se é um vídeo de interação ou um vídeo final e por fim o campo de pontuação do tipo inteiro, no qual é atribuído ao vídeo de interação onde o mesmo deve retornar a

pontuação definida quando o usuário retornar a resposta correta do vídeo de interação. Toda essa representação pode ser visualizada na figura 3.28.

Figura 3.28 - Estrutura de *VideoField*.

```
[Serializable]
2 references
public struct VideoField{
    2 references
    public VideoClip video;
    1 reference
    public string tag;

    4 references
    public bool isActionVideo;

    2 references
    public bool isTopicEnd;

    0 references
    public int pointValue;

}
```

Fonte: Próprio autor.

Em resultado, foi criado uma lista de objetos chamada *videoFields* do mesmo tipo da estrutura, cada objeto dentro dessa lista tem os mesmos campos definidos na estrutura *VideoField*, no qual pode ser observado na figura 3.29.

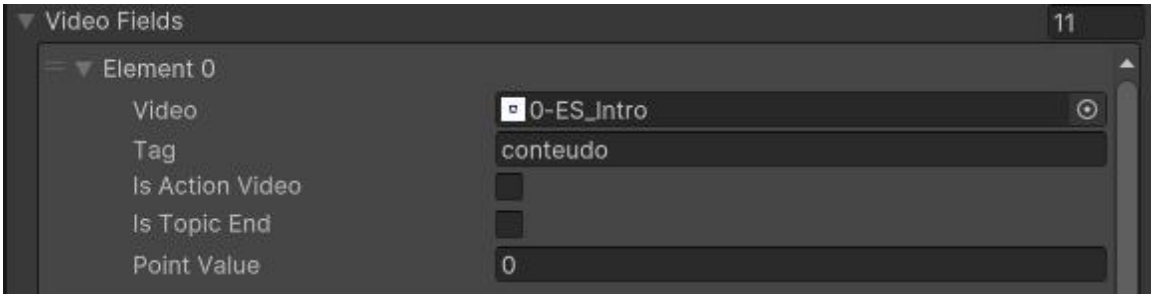
Figura 3.29 - Estrutura de *VideoField*.

```
[Header("Object lists configuration:")]
[SerializeField]
1 reference
private List<Button> buttonFields = new List<Button>();

[SerializeField]
9 references
private List<VideoField> videoFields = new List<VideoField>();

[SerializeField]
12 references
private List<PrefabField> prefabFields = new List<PrefabField>();
```

Fonte: Próprio autor.

Dentro do editor da Unity, o componente criado pode ser visualizado da seguinte forma como mostra a figura 3.30, onde contém uma lista com vários objetos que contém os campos definidos na estrutura.

Figura 3.30 - Representação de *videoFiels* no editor da Unity.



Fonte: Próprio autor.

A figura 3.31, mostra como funciona a verificação para se caso o usuário acertar ou errar um item que a lista oferece em relação ao vídeo proposto para interação.

Em primeiro momento é feita uma comparação entre a *tag* do item selecionado, no caso, é utilizada *actionButton* onde é associado o botão da tela de interação no qual contém a imagem e *tag* do item selecionado, então é feita a comparação com a *tag* do *videoPlayer* para quem foi associado os dados do vídeo de interação e também é verificado se para o *vídeoPlayer* foi associado uma condição verdadeira para um vídeo de interação.

Caso toda essa validação seja verdadeira, o *player* é passado para o próximo vídeo que no caso é de resposta correta, o *StatusButton* muda para *false* onde o botão de interação é desativado pelo fato de que o usuário acertou a resposta e não está mais em um vídeo de interação, o próximo passo é parar o contador de tempo, em seguida é acrescentado o valor de pontos padrão que é igual a 10 e posteriormente é atribuído ao texto em tela a nova pontuação. Por fim é disparado o áudio de resposta correta, onde é atribuído o arquivo de áudio para o áudio *player*.

Figura 3.31 - Método *OnActionButton* em *InteractiveList*.

```
public void OnActionButton(){
    if(actionButton.tag == videoPlayer.tag && videoPlayer.GetComponent<VideoPlayerStatus>().isActionVideo){
        NextVideo();
        StatusButton(false);
        timerObject.GetComponent<SceneTimer>().IsCounting = false;
        scenePoints += 10;
        pointText.SetText(scenePoints.ToString());
        audioSource.clip = okAudioClip;
        audioSource.Play();
    }else{
        audioSource.clip = erroAudioClip;
        erroItemActive = true;
        actionButton.GetComponent<RawImage>().texture = erroSprite;
        audioSource.Play();
    }
}
```

Fonte: Próprio autor.

Se caso a validação for falsa, é disparado o arquivo de áudio de resposta errada, a variável booleana *erroItemActive* é dita com verdadeira para que na função *Update* possa ser verificado quando terminar o áudio para que o botão volte a ter a imagem e atributos do item selecionado anteriormente. Isso se deve que ao errar um item, no mesmo bloco *else* é atribuído à imagem de item errado para o botão.

A figura 3.32 mostra o método *FixedUpdate*(), no qual é executado em tempo real, todo bloco ou linha de código implementado dentro dessa função será sempre executado em modo parecido com um *loop*, mas esse *loop* só para quando a aplicação é finalizada. Dito isso, dentro deste método foi implementada verificações para verificar se o vídeo executado no momento é um vídeo de interação e se os botões de interação estão desativados, se caso for verdadeiro, os botões são ativados e o contador de tempo é iniciado.

Figura 3.32 - Método *FixedUpdate* em InteractiveList.

```
public void FixedUpdate(){
    if(videoPlayer.GetComponent<VideoPlayerStatus>().isActionVideo && !isButtonActive){
        StatusButton(true);
        timerObject.GetComponent<SceneTimer>().IsCounting = true;
    }
```

Fonte: Próprio autor.

Outra implementação para o método *FixedUpdate*, é a verificação se o vídeo executado no momento é um vídeo de passagem de conteúdo, se caso for, é verificado se os botões estão ativados, caso estejam eles são desativados e então é

feita a verificação para se um vídeo não é um vídeo final, caso seja verdadeira o próximo vídeo é executado, como mostra a figura 3.33.

Figura 3.33 - Método *FixedUpdate* em *InteractiveList*, segundo bloco de verificação.

```
if(videoPlayer.isPaused && !videoPlayer.GetComponent<VideoPlayerStatus>().isActionVideo){
    if(isButtonActive){
        StatusButton(false);
    }

    if(!videoPlayer.GetComponent<VideoPlayerStatus>().isTopicEnd){
        NextVideo();
    }
}
```

Fonte: Próprio autor.

A terceira implementação do método é responsável por verificar se o botão de interação está sendo usado como um item de erro, caso seja verdadeiro, o botão é atribuído novamente para os atributos do item selecionado anteriormente, onde pode ser visualizado na figura 3.34.

Figura 3.34 - Método *FixedUpdate* em *InteractiveList*, terceiro bloco de verificação.

```
if(erroItemActive && !audioSource.isPlaying){
    actionButton.GetComponent<RawImage>().texture = prefabFields[i].sprite;
    erroItemActive = false;
}
```

Fonte: Próprio autor.

Por fim, a última implementação do método *FixedUpdate* diz respeito a verificação se o vídeo executado é um vídeo final, para que assim possa chamar a tela final no qual mostra o resultado do usuário em relação ao tópico referente, no qual pode ser visualizada na figura 3.35.

Figura 3.35 - Método *FixedUpdate* em *InteractiveList*, quarto bloco de verificação.

```
if(videoPlayer.isPaused && videoPlayer.GetComponent<VideoPlayerStatus>().isTopicEnd && finalCalled){
    FinalScreenUpdate();
}
```

Fonte: Próprio autor.

Ainda dentro do InteractiveList, a figura 3.36 mostra como é a implementação do método *NextVideo*, no qual tem a responsabilidade de se associar ao *videoPlayer* os atributos do próximo vídeo dentro da lista de vídeos a partir de um contador chamado *videoIndex*, ainda dentro desse método é feita a verificação para caso o seja um vídeo final atribuir a variável *finalCalled* como verdadeira para ser utilizada posteriormente em *FixedUpdate* como mostrado anteriormente.

Figura 3.36 - Método *NextVideo* para avançar arquivos de vídeo.

```
public void NextVideo(){
    videoIndex++;
    videoPlayer.clip = videoFields[videoIndex].video;
    videoPlayer.GetComponent<VideoPlayerStatus>().isActionVideo = videoFields[videoIndex].isActionVideo;
    videoPlayer.GetComponent<VideoPlayerStatus>().isTopicEnd = videoFields[videoIndex].isTopicEnd;
    videoPlayer.isLooping = videoFields[videoIndex].isActionVideo;
    videoPlayer.tag = videoFields[videoIndex].tag;
    videoPlayer.Play();

    if(videoPlayer.GetComponent<VideoPlayerStatus>().isTopicEnd){
        finalCalled = true;
    }
}
```

Fonte: Próprio autor.

Em seguida, o composto de *scripts* desenvolvidos de suma importância dentro do contexto proposto é o armazenamento dos dados proeminentes da interação com os tópicos finalizados, diante da interação do usuário com a aplicação.

A lógica e estrutura para armazenamento de dados foi dividida em dois contextos, o primeiro é para o armazenamento dos dados referentes às configurações personalizadas que o usuário determina na tela de configurações. A segunda é para os dados referentes aos tópicos finalizados, salvando pontuação e tempo.

Para armazenar os dados das configurações, assim com realizar a leitura, foram implementados os *scripts LoadConfigurationData* e *ConfigurationData* que é uma estrutura para a serialização dos dados salvos em um arquivo JSON no armazenamento local do dispositivo do usuário.

Na figura 3.37 é possível visualizar a implementação do método *Save* dentro de *LoadConfigurationData*, no qual é responsável por salvar os dados referentes aos ajustes feitos pela tela de configurações. Os objetos do tipo *sliders* são associados às variáveis do *script* no qual são acessados seus valores. Um *slider* tem valor de 0 a 1, assim como o volume de um *player* de áudio. Esses valores são utilizados para

configurar, por exemplo, volume geral, volume de voz e volume de música. Quando esses valores são acessados, eles são associados a um objeto do mesmo tipo da estrutura *ConfigurationData* no qual posteriormente é serializada para um arquivo JSON utilizando o caminho persistente de onde a aplicação foi instalada no dispositivo do usuário.

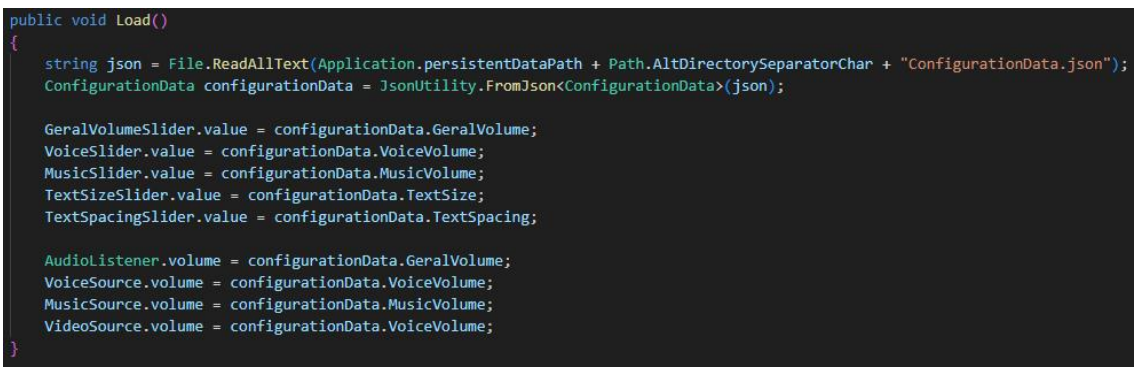
Figura 3.37 - Método para salvar dados em *LoadConfigurationData*.

```
public void Save()
{
    ConfigurationData configurationData = new ConfigurationData
    {
        GeralVolume = GeralVolumeSlider.value,
        VoiceVolume = VoiceSlider.value,
        MusicVolume = MusicSlider.value,
        TextSize = TextSizeSlider.value,
        TextSpacing = TextSpacingSlider.value
    };

    string json = JsonUtility.ToJson(configurationData, true);
    File.WriteAllText(Application.persistentDataPath + Path.AltDirectorySeparatorChar + "ConfigurationData.json", json);
    Debug.Log(Application.persistentDataPath + Path.AltDirectorySeparatorChar + "ConfigurationData.json");
}
```

Fonte: Próprio autor.

Um método chamado *Load* é implementado em *LoadConfigurationData* no qual pode ser visualizado na figura 3.38, este método tem a responsabilidade de carregar e serializar os dados do arquivo JSON para os objetos de jogo, dessa forma os valores salvos são associados aos componentes de jogo quando a aplicação é iniciada. O processo é feito carregando o arquivo no qual deve ter o mesmo nome e caminho do que foi salvo anteriormente, em seguida é feita a serialização um objeto do tipo da estrutura *ConfigurationData* e por fim é feita a associação dos valores do objeto para os componentes de jogo, que são os *Sliders* e os *players* de áudio.

Figura 3.38 - Método para carregar dados em *LoadConfigurationData*.



```
public void Load()
{
    string json = File.ReadAllText(Application.persistentDataPath + Path.AltDirectorySeparatorChar + "ConfigurationData.json");
    ConfigurationData configurationData = JsonUtility.FromJson<ConfigurationData>(json);

    GeralVolumeSlider.value = configurationData.GeralVolume;
    VoiceSlider.value = configurationData.VoiceVolume;
    MusicSlider.value = configurationData.MusicVolume;
    TextSizeSlider.value = configurationData.TextSize;
    TextSpacingSlider.value = configurationData.TextSpacing;

    AudioListener.volume = configurationData.GeralVolume;
    VoiceSource.volume = configurationData.VoiceVolume;
    MusicSource.volume = configurationData.MusicVolume;
    VideoSource.volume = configurationData.VoiceVolume;
}
```

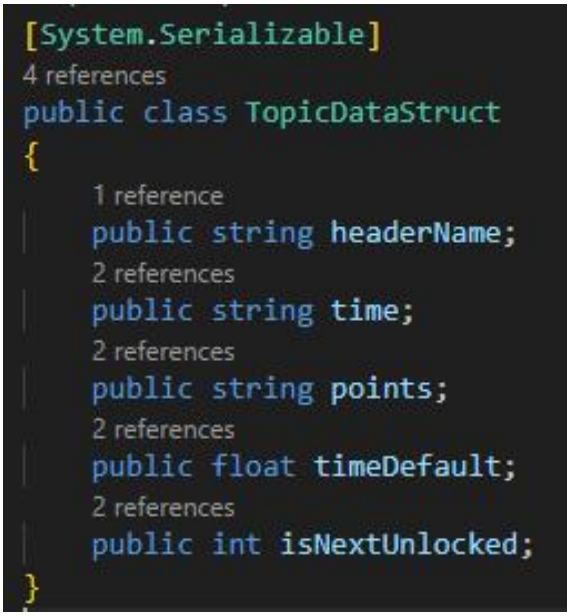
Fonte: Próprio autor.

Para o contexto de armazenamento de dados para a pontuação e tempo do usuário em relação ao tópico em referência, foi criada uma estrutura para agregar e ser serializado para um arquivo JSON.

Para cada tópico foi definido que seria salvo um arquivo JSON, para que pudesse ser encontrado e mais facilmente testado e validado os dados referentes a cada tópico, levando em consideração abordagem computacional no qual divide um problema em pedaços e soluciona os pequenos pedaços um de cada vez, tomando essa abordagem, acaba facilitando a análise de dados perante o teste com o público alvo.

Na figura 3.39 é possível observar a estrutura implementada no qual possui campo para titulo do tópico, tempo, pontuações, tempo original sem formatação para minutos e segundos, e uma variável int que serve como *flag* no qual tem a responsabilidade de verificar se o tópico foi finalizado e o próximo deve ser desbloqueado.

Figura 3.39 - Estrutura de dados dos tópicos para serialização do arquivo JSON.



Fonte: Próprio autor.

Para salvar os dados relacionados aos tópicos, foi criado um *script* chamado *SaveStageData* no qual serializa os dados de tempo, pontuação e outros para o objeto do tipo *TopicDataStruct*. Utilizando a mesma lógica os dados são salvos em um arquivo JSON no armazenamento interno do dispositivo do usuário.

Figura 3.40 - Método para salvar dados relacionados aos tópicos.

```
public void Save()
{
    TopicDataStruct topicDataStruct = new TopicDataStruct
    {
        headerName = HeaderName,
        time = timeText.text,
        points = pointsText.text,
        isNextUnlocked = isNextUnlocked,
        timeDefault = timer.getTime()
    };
    string json = JsonUtility.ToJson(topicDataStruct, true);
    HeaderName = HeaderName.Replace(" ", "");
    File.WriteAllText(Application.persistentDataPath + Path.AltDirectorySeparatorChar + HeaderName + ".json", json);
    Debug.Log(Application.persistentDataPath + Path.AltDirectorySeparatorChar + HeaderName + ".json");
}
```

Fonte: Próprio autor.

Os dados dos tópicos são utilizados na tela de tópicos e subtópicos, no qual é feita a formatação e cálculo para a soma total de pontos em cada tópico e tempo, e é associada os dados brutos para cada subtópico.

Visto esse contexto foi criado o *script TopicProgressDataLoader*, no qual tem a responsabilidade de carregar todos os dados e associar aos subtópicos e fazer a formatação e associar aos tópicos. Na figura 3.41 é possível visualizar o bloco de código no qual realiza a formatação e associação dos dados para a tela de tópico ou tela de progresso.

Figura 3.41 - Método para carregar todos os dados relacionados aos tópicos.

```
public void LoadAllData()
{
    isFinished = 0;
    totalPoints = 0;

    foreach(TopicProgressStruct t in topicProgressStructs){
        t.LoadData();
        totalPoints += int.Parse(t.PointsText.text);
        if(t.IsFinished){
            isFinished += 1;
        }
    }
    progressTopicsText.text = isFinished.ToString();
    progressPointsText.text = totalPoints.ToString();
}
```

Fonte: Próprio autor.

Como dito, é necessário que cada subtópico tenha seus dados referentes de quando o usuário finalizou o mesmo, dessa forma, para o *script TopicProgressDataLoader* é implementado um método no qual é responsável a atribuir esses dados para cada campo de cada subtópico com seus respectivos

dados salvos e carregados a partir do arquivo JSON, na figura 3.42, o método *LoadData* é chamado no momento da iteração da lista em *LoadAllData*, dessa forma, todos os tópicos e subtópicos dentro da lista tem seus dados carregados a partir do armazenamento interno do dispositivo.

O método *SetTime* tem a funcionalidade de formatar o tempo bruto em segundos para minutos e segundos, como mostra a figura 3.42.

Figura 3.42 - Método para carregar todos os dados relacionados aos tópicos.

```
1 reference
public void LoadData(){
    PointsText.text = topicDataLoader.GeneralPointText.text;
    SetTime();

    if(int.Parse(topicDataLoader.GeneralTopicText.text) >= toFinished){
        IsFinished = true;
    }
}

1 reference
public void SetTime(){

    float time = topicDataLoader.TimeDefaultTotal;

    int minutes = Mathf.FloorToInt(time/60);
    int seconds = Mathf.FloorToInt(time%60);

    timeText.text = string.Format("{0:00}:{1:00}", minutes, seconds);
}
```

Fonte: Próprio autor.

Neste capítulo foi apresentado todo o processo de desenvolvimento abordado para a produção da aplicação. Foi exemplificado e mostrado o planejamento e desenvolvimento de cada material necessário para compor a aplicação proposta neste trabalho. Como dito, a aplicação foi planejada para ser desenvolvida modularmente e reutilizável, onde de fato o processo exemplificado se aplica para o desenvolvimento de cada item presente na aplicação.

CONSIDERAÇÕES FINAIS

Em suma, a ausência de testes práticos não diminui a relevância teórica e as técnicas da aplicação desenvolvida. Com este trabalho de pesquisa foi possível abordar técnicas valiosas das quais podem ser aplicadas no desenvolvimento de software educacional, incluindo concepção de ideia, planejamento, design, implementação e os devidos processos para os mesmos. A documentação detalhada desenvolvida pode vir a servir como recursos para possíveis trabalhos futuros de pesquisa na área.

Com os resultados de Assunção(2018) foi possível abordar que pessoas portadoras da dislexia podem não ter o acompanhamento adequado para superar dificuldades provindas da necessidade educacional específica, no qual Montanari(2015, *apud*. Prado, 2010) afirma que as técnicas e abordagens audiovisuais são uma excelente meio de auxílio para os mesmos.

O uso da gamificação é notório por ser uma abordagem universal podendo ser aplicada em qualquer área de desenvolvimento, com esta pesquisa foi possível justificar que essa abordagem é valiosa ao ser aplicada na área educacional, no qual segundo os resultados de Faria(2021), serve de engajamento para o aluno tornando-o o centro e personagem principal no ensino-aprendizagem, em resultado, cativando a concentração no processo de aprendizagem, no qual foi possível afirmar com os resultados de Raguze *et al*.(2016) que o engajamento do usuário é o objetivo central da gamificação.

Com o referente trabalho de pesquisa foi possível notar que a abordagem tradicional do ensino de lógica de programação é feito de forma textual, com os resultados de Silva *et al*.(2019) foi possível notar as dificuldades que uma pessoa pode enfrentar durante a jornada de aprendizagem, as mesmas dificuldades podem ser intensificadas para uma pessoa que possui dislexia, no mais, as frustrações que um dislexo pode enfrentar podem ser contornadas ao aplicar abordagens audiovisuais, tratando e adaptando o conteúdo ensinado.

Com o presente trabalho foi possível entender a diferença entre um jogo e uma aplicação na qual utiliza da abordagem de gamificação, de acordo com Raguze *et al*.(2016), a mesma são características eficazes dos jogos no qual se fazem comprovadas a motivar o usuário no processo de utilização do jogo ou aplicação, no

qual uma aplicação educacional gamificada utiliza esses recursos dos jogos para cativar o aluno no ensino-aprendizagem.

Este trabalho de pesquisa também colabora para o desenvolvimento de aplicações gamificadas que possam auxiliar em qualquer área do conhecimento no meio educacional a fim de ajudar alunos que possuem necessidades educacionais específicas. O conhecimento, técnicas e abordagens apresentadas podem ser de extrema importância para implementação e implantação de projetos nos quais atuam no auxílio desses alunos.

Mesmo diante de tais resultados, foram presenciadas limitações no decorrer do desenvolvimento deste trabalho de pesquisa, as dificuldades em conhecimento para trabalhos de pesquisa e o referente tempo para elaboração, produção da pesquisa e desenvolvimento foi de grande impacto. O presente trabalho não se fez apenas no processo de desenvolvimento, mas também no aprendizado da realização de um trabalho de pesquisa, no qual o mesmo consome tempo para realização efetiva e correta do mesmo. O tempo limitado atingiu diretamente o processo de pesquisa pelo público-alvo e a realização dos testes para a avaliar a aplicação, as técnicas e abordagens utilizadas no desenvolvimento.

Devido aos resultados obtidos em conjunto com as limitações presenciadas, são abertas várias possibilidades para trabalhos futuros em relação a aplicação e a área de pesquisa referente, no qual primeiramente, se faz necessário a aplicação de testes práticos para análise e validação dos resultados obtidos a partir do uso da aplicação desenvolvida, visando o melhoramento e estudo das dificuldades encontradas pelos alunos que utilizaram a mesma.

Sugere-se que para a aplicação, deve ser desenvolvida e aplicada novas abordagens perante a passagem de conhecimento, no qual atualmente utiliza-se a mesma estratégia de interação para todos os tópicos. Sugere-se também, a inclusão de problemas clássicos de lógica de programação, como torre de hanói, das quais podem ser inseridas ao final do último conteúdo, fazendo assim, a garantia que o usuário consuma o conhecimento de todos tópicos antes de iniciar o problema clássico. Por fim, foi observado que o uso de inteligência artificial pode ser benéfico para prover a adaptabilidade do grau de dificuldade da aplicação em relação ao aluno, isso faz proveito em determinar e adaptar os desafios presentes em relação às necessidades específicas de cada usuário.

Dessa forma, os trabalhos posteriores a este podem se fazer benéficos para o público alvo ao aplicar os pontos apresentados, os mesmo podem se fazer superiores em relação à adaptabilidade das dificuldades específicas de intelecto, adaptação e carência de conhecimento anterior que é necessário para o processo de aprendizagem da lógica de programação.

Diante da produção deste trabalho de pesquisa foi adquirida a experiência pelo processo de aprendizado do desenvolvimento do mesmo, na qual foi possível realizar as etapas de pesquisa e desenvolvimento. A mesma foi adquirida desde a proposta de projeto utilizando o método de investigação PICO(*Population, Intervention, Comparison e Outcome*) e posteriormente foram aplicados os resultado obtidos no desenvolvimento da solução proposta, com isso, diante da experiência presenciada, o processo de aprendizado das normas e etapas para produção deste trabalho foi evoluindo no decorrer do tempo. Outro ponto chave foi a integração dos conhecimentos e resultados obtidos através da pesquisa, na qual foi possível interligar abordagens e técnicas distintas em prol de um único objetivo.

As dificuldades presenciadas foram por conta da falta de conhecimento em relação ao desenvolvimento de trabalhos de pesquisa, no processo de desenvolvimento da proposta de projeto essas dificuldades se mostraram mais aparentes e tiveram um impacto maior, o curto tempo em conjunto a outras responsabilidades do curso acadêmico intensificaram ainda mais tais dificuldades presenciadas. Mas em resultado, tais dificuldades presenciadas foram superadas e com elas foi adquirida a experiência e conhecimento em relação a produção de trabalhos de pesquisa.

REFERÊNCIAS

AFFINI, Letícia Passos. AMÉRICO, Marcos. **Aprendizagem baseada em problemas: uma abordagem interdisciplinar para o ensino do audiovisual.** Bauru, 2007.

ALEEM, Saiqa. et al. **Game development software engineering process life cycle: a systematic review**. Abu Dhabi, 2016.

AICINEMA. **O que é Audiovisual?**, 2021. Disponível em: <https://www.aicinema.com.br/o-que-e-audiovisual/>. Acesso em: 18 maio 2023.

AMAZON WEB SERVICES. **What is Open Source?**, c2023. Disponível em: <https://aws.amazon.com/pt/what-is/open-source/>. Acesso em: 18 maio 2023.

ASSOCIAÇÃO BRASILEIRA DE DISLEXIA. **O que é dislexia?**, 2016. Disponível em: <https://www.dislexia.org.br/o-que-e-dislexia/>. Acesso em: 18 maio 2023.

ASSUNÇÃO, Gabriele Silva. **A dislexia e os desafios no processo de aprendizagem da língua portuguesa.** Santo Antônio de Jesus, 2018.

AUDACITY. **Sobre o Audacity**, c2023. Disponível em: <https://www.audacityteam.org/about/>. Acesso em: 18 maio 2023.

BATISTA, Jeize de Fatima. et al. **Estimugame: um software de estímulo para a leitura de alunos diagnosticados com dislexia.** Santa Catarina, 2019.

BOROCHOVICIUS, Eli. TORTELLA, Jussara Cristina Barboza. **Aprendizagem baseada em problemas: um método de ensino-aprendizagem e suas práticas educativas.** Rio de Janeiro, 2014.

FARIA, Alexandre Ferreira de. **Gamificação na educação.** Goiânia, 2021.

FLUMINHAN, Carmem Silva Lima. et al. **A importância do feedback como ferramenta pedagógica na educação a distância.** Presidente Prudente, 2013.

GIMP. **Página inicial do GIMP**, [s.d]. Disponível em: <https://www.gimp.org/>. Acesso em: 18 maio 2023.

INKSCAPE. **Sobre o Inkscape**, [s.d]. Disponível em: <https://inkscape.org/pt-br/sobre/>. Acesso em: 18 maio 2023.

INSTITUTO ABCD. **Página de perguntas e respostas**, c2021. Disponível em: <https://www.institutoabcd.org.br/perguntas-e-respostas/>. Acesso em: 08 de Novembro de 2023.

KITA, Daniela Mayumi. **Iara App e o paradigma da inclusão: desafios no desenvolvimento de um aplicativo acessível.** Rio Claro, 2022.

KRESS, Gunther; LEEUWEN, Theo van. **Multimodal discourse: the modes and media of contemporary communication.** Nova York, 2001.

LEAL, Alexis Vinicius de Aquino. **Ensino de programação no ensino médio integrado.** Goiânia, 2014.

MAIN LEAF. **GDD exemplo e guia completo para o criar o seu jogo**, 2023. Disponível em: <https://mainleaf.com/pt/gdd-exemplo-e-guia-completo-para-criar-o-seu-jogo/>. Acesso em: 08 de Novembro de 2023.

MENDES, João Francisco Soares. **"Canto das Histórias" - Jogo sério para inclusão de crianças com perturbações do desenvolvimento.** Lisboa, 2018.

MENEZES, Ebenezer Takuno de. **Verbete software educacional.** Dicionário Interativo da Educação Brasileira - Educa Brasil. São Paulo: Midiamix Editora, 2001. Disponível em <https://www.educabrasil.com.br/software-educacional/>. Acesso em 18 maio 2023.

MICROSOFT. **Tour of C#**, 2023. Disponível em: <https://learn.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/>. Acesso em: 18 maio 2023.

MICROSOFT. **Unity: Developing Your First Game with Unity and C#**, 2015. Disponível em: <https://learn.microsoft.com/pt-br/archive/msdn-magazine/2014/august/unity-developing-your-first-game-with-unity-and-csharp>. Acesso em: 18 maio 2023.

MICROSOFT. **Visual Studio Code**, c2023. Disponível em: <https://visualstudio.microsoft.com/pt-br/#vscode-section>. Acesso em: 18 maio 2023.

MONTANARI, Rafaela. **Uma análise sobre dislexia na escola.** Rio Claro, 2015.

MOURA, Suzana Paula Pedreira Tavares de. **A dislexia e os desafios pedagógicos**. Universidade Cândido Mendes, Niterói, 2013.

NAVARRO, Gabrielle. **Gamificação: a transformação do conceito do termo jogo no contexto da pós-modernidade.** São Paulo, 2013.

NuES-UFRRJ. **Núcleo de Práticas de Engenharia de Software Aplicadas a Games**, 2020. Disponível em: <https://r1.ufrrj.br/nuesgames/index.php>. Acesso em: 08 de Novembro de 2023.

ORLANDI, Tomás Roberto Cotta. et al. **Gamificação: uma nova abordagem multimodal para a educação.** Brasília, 2018.

PETRILLO, Fábio dos Santos. **Práticas ágeis no processo de desenvolvimento de jogos eletrônicos**. Porto Alegre, 2008.

RAGUZE, Tiago. SILVA, Régio Pierre da. **Gamificação aplicada a ambientes de aprendizagem.** Novo Hamburgo, 2016.

SANTOS, Cristina Mamédio da Costa. et al. **A estratégia PICO para a construção da pergunta de pesquisa e busca de evidências.** USP, 2007.

SANTOS, Rafaela Correia dos. et al. **Jogos educativos na dislexia.** Campina Grande, 2014.

SILVA, Danilo Nogueira da. et al. **Lógica de programação: Dificuldades de ensino-aprendizagem, métodos e ferramentas computacionais.** Anápolis, 2019.

SILVA, Diogo Vinícius de S.. et al. **Os benefícios do uso de kanban na gerência de projetos de manutenção de software.** Teresina, 2012.

SILVA, Elton Martins. **Jogos digitais como ferramenta de desenvolvimento de habilidades em crianças com transtornos do espectro autista.** Surubim, 2022.

SILVA, Nilza Sebastiana da. SILVA, Fabio José Antônio da. **A dislexia e a dificuldade na aprendizagem.** 2016.

SPEAKTOR. **Como funciona a síntese de fala?**, 2022. Disponível em: <https://speaktor.com/speech-synthesis/>. Acesso em: 08 junho 2023.

UML DIAGRAMS. **The Unified Modeling Language**, c2023. Disponível em: <https://www.uml-diagrams.org/>. Acesso em: 08 junho 2023.

VANCINI, Bruna Elisabete. et al. **A utilização da gamificação como elemento de engajamento de estudantes do ensino fundamental**. Passo Fundo, 2020.

APÊNDICE

APÊNDICE A - Scripts desenvolvidos para a aplicação MyFriendLogic-X

1. ConfigurationData.cs

```
[System.Serializable]
public class ConfigurationData
{
    public float GeralVolume;
    public float VoiceVolume;
    public float MusicVolume;
    public float TextSize;
    public float TextSpacing;
}
```

2. DataAllStructure.cs

```
[System.Serializable]
public class DataAllStructure
{
    public int points = 0;
    public int topics = 0;
    public float totalTime = 0;
}
```

3. FontTextSize.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using System.IO;
public class FontTextSize : MonoBehaviour
{
```

```
    [SerializeField] private float textSize0;
    [SerializeField] private float textSize1;
    [SerializeField] private float textSize2;
    [SerializeField] private float textSize3;
    [SerializeField] private float textSize4;

    [SerializeField] private float[] textSpacing = new float[5];
    public TMP_Text text;

    public void Start(){
        text = GetComponent<TMP_Text>();
    }

    public void LateUpdate(){
        string json = File.ReadAllText(Application.persistentDataPath +
Path.AltDirectorySeparatorChar + "ConfigurationData.json");
        ConfigurationData configurationData =
JsonUtility.FromJson<ConfigurationData>(json);
        SetTextSize(configurationData.TextSize);
        SetTextSpacing(configurationData.TextSpacing);
    }

    public void SetTextSize(float size)
    {
        float input = size;
        switch (input){
            case 0:
                text.fontSize = textSize0;
                break;
            case 1:
                text.fontSize = textSize1;
                break;
            case 2:
```

```
            text.fontSize = textSize2;
            break;
        case 3:
            text.fontSize = textSize3;
            break;
        case 4:
            text.fontSize = textSize4;
            break;
    }
}

public void SetTextSpacing(float spacing)
{
    float input = spacing;
    switch (input){
        case 0:
            text.characterSpacing = textSpacing[0];
            break;
        case 1:
            text.characterSpacing = textSpacing[1];
            break;
        case 2:
            text.characterSpacing = textSpacing[2];
            break;
        case 3:
            text.characterSpacing = textSpacing[3];
            break;
        case 4:
            text.characterSpacing = textSpacing[4];
            break;
    }
}
}
```

4. InteractiveList.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Video;

[Serializable]
public struct PrefabField{
    public Texture sprite;
    public string tag;
}

[Serializable]
public struct VideoField{
    public VideoClip video;
    public string tag;

    public bool isActionVideo;

    public bool isTopicEnd;

    public int pointValue;

}

public class InteractiveList : MonoBehaviour
{

    [Header("Interactive elements configuration:")]
```

```
[SerializeField]
private VideoPlayer videoPlayer;

[SerializeField]
private GameObject actionButton;

[SerializeField]
private AudioClip okAudioClip;

[Header("Item error configuration:")]
[SerializeField]
private AudioClip erroAudioClip;

[SerializeField]
private Texture erroSprite;

[SerializeField]
private AudioSource audioSource;

[Header("Point configuration:")]
[SerializeField]
private GameObject timerObject;

[SerializeField]
private TextMeshProUGUI pointText;

[Header("Final screen configuration:")]

[SerializeField]
private GameObject finalScreen;

[SerializeField]
private TextMeshProUGUI finalPointText;
```

```
[SerializeField]
private TextMeshProUGUI timeText;

[SerializeField]
private TextMeshProUGUI finalTimeText;
[SerializeField]
private AudioClip finalAudioClip;

private bool finalCalled = false;

[Header("Object lists configuration:")]
[SerializeField]
private List<Button> buttonFields = new List<Button>();

[SerializeField]
private List<VideoField> videoFields = new List<VideoField>();

[SerializeField]
private List<PrefabField> prefabFields = new List<PrefabField>();

private int i = 0;
private int videoIndex = 0;
private bool isButtonActive = false;
private bool erroItemActive = false;
private int scenePoints = 0;

public void Start()
{
    i = 0;
    timeText.text = "Tempo: 00:00";
    pointText.text = "0";
    scenePoints = 0;
    timerObject.GetComponent<SceneTimer>().IsCounting = false;
    timerObject.GetComponent<SceneTimer>().Time = 0f;
```

```
        finalScreen.SetActive(false);
        finalCalled = false;

        System.Random rand = new System.Random();

        for (int k = prefabFields.Count - 1; k > 0; k--)
        {
            int j = rand.Next(0, k + 1);
            PrefabField temp = prefabFields[k];
            prefabFields[k] = prefabFields[j];
            prefabFields[j] = temp;
        }

        actionButton.GetComponent<RawImage>().texture = prefabFields[0].sprite;
        actionButton.tag = prefabFields[0].tag;
        videoIndex = 0;
        videoPlayer.clip = videoFields[videoIndex].video;
        videoPlayer.GetComponent<VideoPlayerStatus>().isActionVideo =
videoFields[videoIndex].isActionVideo;
        videoPlayer.GetComponent<VideoPlayerStatus>().isTopicEnd =
videoFields[videoIndex].isTopicEnd;
        videoPlayer.isLooping = videoFields[videoIndex].isActionVideo;
        StatusButton(false);
    }

    public void FixedUpdate(){
        if(videoPlayer.GetComponent<VideoPlayerStatus>().isActionVideo
&& !isButtonActive){
            StatusButton(true);
            timerObject.GetComponent<SceneTimer>().IsCounting = true;
        }

        if(videoPlayer.isPaused
&& !videoPlayer.GetComponent<VideoPlayerStatus>().isActionVideo){
```

```
            if(isButtonActive){
                StatusButton(false);
            }

            if(!videoPlayer.GetComponent<VideoPlayerStatus>().isTopicEnd){
                NextVideo();
            }
        }

        if(erroItemActive && !audioSource.isPlaying){
            actionButton.GetComponent<RawImage>().texture = prefabFields[i].sprite;
            erroItemActive = false;
        }

        if(videoPlayer.isPaused &&
videoPlayer.GetComponent<VideoPlayerStatus>().isTopicEnd && finalCalled){
            FinalScreenUpdate();
        }

    }

    public void OnTouchAction(bool actionFoward){
        erroItemActive = false;
        if(actionFoward){
            if(i >= prefabFields.Count-1){
                i = 0;
            }else{
                i++;
            }

        }else{
            if(i <= 0){
                i = prefabFields.Count-1;
            }else{
```

```
                i--;
            }
        }
        actionButton.GetComponent<RawImage>().texture = prefabFields[i].sprite;
        actionButton.tag = prefabFields[i].tag;
    }

    public void StatusButton(bool status){
        isButtonActive = status;
        foreach(Button b in buttonFields){
            b.interactable = status;
        }
        Debug.Log("Fui modificado");
    }
    public void OnActionButton(){
        if(actionButton.tag == videoPlayer.tag &&
videoPlayer.GetComponent<VideoPlayerStatus>().isActionVideo){
            NextVideo();
            StatusButton(false);
            timerObject.GetComponent<SceneTimer>().IsCounting = false;
            scenePoints += 10;
            pointText.SetText(scenePoints.ToString());
            audioSource.clip = okAudioClip;
            audioSource.Play();
        }else{
            audioSource.clip = erroAudioClip;
            erroItemActive = true;
            actionButton.GetComponent<RawImage>().texture = erroSprite;
            audioSource.Play();
        }
    }

    public void NextVideo(){
        videoIndex++;
```

```
        videoPlayer.clip = videoFields[videoIndex].video;
        videoPlayer.GetComponent<VideoPlayerStatus>().isActionVideo =
videoFields[videoIndex].isActionVideo;
        videoPlayer.GetComponent<VideoPlayerStatus>().isTopicEnd =
videoFields[videoIndex].isTopicEnd;
        videoPlayer.isLooping = videoFields[videoIndex].isActionVideo;
        videoPlayer.tag = videoFields[videoIndex].tag;
        videoPlayer.Play();

        if(videoPlayer.GetComponent<VideoPlayerStatus>().isTopicEnd){
            finalCalled = true;
        }
    }

    public void FinalScreenUpdate(){
        audioSource.clip = finalAudioClip;
        audioSource.Play();
        finalScreen.SetActive(true);
        finalPointText.text = pointText.text;
        finalTimeText.text = timeText.text.Replace("Tempo: ", "");
        finalCalled = false;
    }
}
```

5. LoadConfigurationData.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using UnityEngine;
using UnityEngine.UI;

public class LoadConfigurationData : MonoBehaviour
{
```

```
    [SerializeField] private Slider GeralVolumeSlider;
    [SerializeField] private Slider VoiceSlider;
    [SerializeField] private Slider MusicSlider;
    [SerializeField] private Slider TextSizeSlider;
    [SerializeField] private Slider TextSpacingSlider;
    [SerializeField] private AudioSource MusicSource;
    [SerializeField] private AudioSource VoiceSource;
    [SerializeField] private AudioSource VideoSource;

    void Start(){
        string json = File.ReadAllText(Application.persistentDataPath +
Path.AltDirectorySeparatorChar + "ConfigurationData.json");
        ConfigurationData configurationData =
JsonUtility.FromJson<ConfigurationData>(json);

        GeralVolumeSlider.value = configurationData.GeralVolume;
        VoiceSlider.value = configurationData.VoiceVolume;
        MusicSlider.value = configurationData.MusicVolume;
        TextSizeSlider.value = configurationData.TextSize;
        TextSpacingSlider.value = configurationData.TextSpacing;

        AudioListener.volume = configurationData.GeralVolume;
        VoiceSource.volume = configurationData.VoiceVolume;
        MusicSource.volume = configurationData.MusicVolume;
        VoiceSource.Stop();
    }

    public void Save()
    {
        ConfigurationData configurationData = new ConfigurationData
        {
            GeralVolume = GeralVolumeSlider.value,
            VoiceVolume = VoiceSlider.value,
            MusicVolume = MusicSlider.value,
```

```
            TextSize = TextSizeSlider.value,
            TextSpacing = TextSpacingSlider.value
        };

        string json = JsonUtility.ToJson(configurationData, true);
        File.WriteAllText(Application.persistentDataPath +
Path.AltDirectorySeparatorChar + "ConfigurationData.json", json);
        Debug.Log(Application.persistentDataPath + Path.AltDirectorySeparatorChar +
"ConfigurationData.json");
    }

    public void Load()
    {
        string json = File.ReadAllText(Application.persistentDataPath +
Path.AltDirectorySeparatorChar + "ConfigurationData.json");
        ConfigurationData configurationData =
JsonUtility.FromJson<ConfigurationData>(json);

        GeralVolumeSlider.value = configurationData.GeralVolume;
        VoiceSlider.value = configurationData.VoiceVolume;
        MusicSlider.value = configurationData.MusicVolume;
        TextSizeSlider.value = configurationData.TextSize;
        TextSpacingSlider.value = configurationData.TextSpacing;

        AudioListener.volume = configurationData.GeralVolume;
        VoiceSource.volume = configurationData.VoiceVolume;
        MusicSource.volume = configurationData.MusicVolume;
        VideoSource.volume = configurationData.VoiceVolume;
    }
}
```

6. MenuScript.cs

```
using System.Collections;
using System.Collections.Generic;
```

```
using UnityEngine;
using UnityEngine.UI;

public class MenuScript : MonoBehaviour
{

    [SerializeField] private RawImage imgBackground;
    //[SerializeField] private Image imgTitle;
    [SerializeField] private float x, y; //fillSpeed;

    void Update()
    {
        //imgTitle.fillAmount += fillSpeed * Time.deltaTime;
        imgBackground.uvRect = new Rect(imgBackground.uvRect.position + new
Vector2(x, y) * Time.deltaTime, imgBackground.uvRect.size);
    }
}
```

7. PauseGame.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PauseGame : MonoBehaviour
{
    public void Pause()
    {
        Time.timeScale = 0;
    }

    public void Resume()
    {
        Time.timeScale = 1;
    }
```

```
}
```

8. SaveStageData.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using UnityEngine;
using TMPro;

public class SaveStageData : MonoBehaviour
{
    [SerializeField]
    private string headerName;
    [SerializeField]
    private TextMeshProUGUI timeText;
    [SerializeField]
    private TextMeshProUGUI pointsText;
    [SerializeField]
    private SceneTimer timer;
    [SerializeField]
    public int isNextUnlocked = 1;

    public string HeaderName { get => headerName; set => headerName = value; }

    public void Save()
    {
        TopicDataStruct topicDataStruct = new TopicDataStruct
        {
            headerName = HeaderName,
            time = timeText.text,
            points = pointsText.text,
            isNextUnlocked = isNextUnlocked,
            timeDefault = timer.getTime()
```

```
        };
        string json = JsonUtility.ToJson(topicDataStruct, true);
        HeaderName = HeaderName.Replace(" ", "");
        File.WriteAllText(Application.persistentDataPath +
Path.AltDirectorySeparatorChar + HeaderName + ".json", json);
        Debug.Log(Application.persistentDataPath + Path.AltDirectorySeparatorChar +
HeaderName + ".json");
    }
}
```

9. SceneCall.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SceneCall : MonoBehaviour
{
    [SerializeField] private GameObject screenActive;
    [SerializeField] private GameObject screenDeactive;
    [SerializeField] private TransitionScript screenTransition;

    public void OnClick()
    {
        screenTransition.SetPosition(screenActive, screenDeactive);
    }
}
```

10. SceneLocker.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SceneLocker : MonoBehaviour
{
```

```
    private bool isLocked = false;

    void Locker(bool isLocked)
    {
        this.isLocked = isLocked;
    }
}
```

11. SceneTimer.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;
using UnityEngine.UI;

public class SceneTimer : MonoBehaviour
{
    private float time = 0f;

    [SerializeField]
    private TextMeshProUGUI timeText;

    [SerializeField]
    private bool isCounting = false;
    public bool IsCounting { get => isCounting; set => isCounting = value; }
    public float Time { get => time; set => time = value; }

    void Update()
    {
        if(IsCounting){
            CoutingTime();
        }
    }
```

```
    public void CoutingTime(){
        Time += UnityEngine.Time.deltaTime;

        int minutes = Mathf.FloorToInt(Time/60);
        int seconds = Mathf.FloorToInt(Time%60);

        timeText.SetText(string.Format("Tempo: {0:00}:{1:00}", minutes, seconds));
    }

    public float getTime(){
        return Time;
    }
}
```

12. StageInitialConfiguration.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class StageInitialConfiguration : MonoBehaviour
{
    [SerializeField]
    private List<AudioClip> stageAudioClip = new List<AudioClip>();
    [SerializeField]
    private AudioClip originalAudioClip;
    [SerializeField]
    private AudioSource audioSource;
    [SerializeField]
    private float delayWaitTime = 0.5f;
    private bool isStage = false;

    private int index;
    IEnumerator SetMusicStage()
```

```
    {
        yield return new WaitForSeconds(delayWaitTime);
        if(isStage){
            audioSource.clip = stageAudioClip[index];
        }else{
            audioSource.clip = originalAudioClip;
        }
        audioSource.Play();
    }
    public void MusicStage(bool isStage){
        this.isStage = isStage;
        index = Random.Range(0, stageAudioClip.Count);
        StartCoroutine(SetMusicStage());
    }
}
```

13. TopicDataLoader.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using TMPro;
using UnityEngine;
using UnityEngine.UI;

[Serializable]
public class TopicStructLoader{
    [SerializeField]
    private string headerName;
    [SerializeField]
    private int maximumPoints;
    [SerializeField]
    private TextMeshProUGUI topicScreenPointText;
    [SerializeField]
```

```
    private TextMeshProUGUI timeText;
    [SerializeField]
    private TextMeshProUGUI pointText;
    [SerializeField]
    private SaveStageData saveStageData;
    private float timeDefault = 0f;
    public float TimeDefault { get => timeDefault; set => timeDefault = value; }
    private int isNextUnlocked = 0;
    public int IsNextUnlocked { get => isNextUnlocked; set => isNextUnlocked =
value; }
    public TextMeshProUGUI PointText { get => pointText; set => pointText = value; }
    public string HeaderName { get => headerName; set => headerName = value; }

    public void Load()
    {
        string json = File.ReadAllText(Application.persistentDataPath +
Path.AltDirectorySeparatorChar + saveStageData.HeaderName + ".json");
        TopicDataStruct configurationData =
JsonUtility.FromJson<TopicDataStruct>(json);

        timeText.text = configurationData.time;
        PointText.text = configurationData.points;
        IsNextUnlocked = configurationData.isNextUnlocked;
        TimeDefault = configurationData.timeDefault;
        topicScreenPointText.text = pointText.text + "/" + maximumPoints.ToString();
    }
}

public class TopicDataLoader : MonoBehaviour
{
    [SerializeField]
    private string headerName;
    [SerializeField]
    private TextMeshProUGUI generalTopicText;
```

```
    public TextMeshProUGUI GeneralTopicText { get => generalTopicText; set =>
generalTopicText = value; }
    [SerializeField]
    private TextMeshProUGUI generalPointText;
    public TextMeshProUGUI GeneralPointText { get => generalPointText; set =>
generalPointText = value; }
    [SerializeField]
    private List<TopicStructLoader> topicStructLoaders = new
List<TopicStructLoader>();
    private float timeDefaultTotal = 0f;
    public float TimeDefaultTotal { get => timeDefaultTotal; set => timeDefaultTotal =
value; }

    public void LoadData(){
        foreach(TopicStructLoader t in topicStructLoaders){
            t.Load();
        }

        SaveAllData();
        string json = File.ReadAllText(Application.persistentDataPath +
Path.AltDirectorySeparatorChar + headerName +"-AllData"+ ".json");
        DataAllStructure configurationData =
JsonUtility.FromJson<DataAllStructure>(json);
        GeneralPointText.text = configurationData.points.ToString();
        GeneralTopicText.text = configurationData.topics.ToString();
        TimeDefaultTotal = configurationData.totalTime;
    }

    public void SaveAllData(){
        DataAllStructure dataAllStructure = new DataAllStructure();

        foreach(TopicStructLoader t in topicStructLoaders){
            dataAllStructure.points += int.Parse(t.PointText.text);
            dataAllStructure.topics += t.IsNextUnlocked;
```

```
            dataAllStructure.totalTime += t.TimeDefault;
        }

        string json = JsonUtility.ToJson(dataAllStructure, true);
        string HeaderName = headerName.Replace(" ", "");
        File.WriteAllText(Application.persistentDataPath +
Path.AltDirectorySeparatorChar + HeaderName +"-AllData"+ ".json", json);
        Debug.Log(Application.persistentDataPath + Path.AltDirectorySeparatorChar +
HeaderName +"-AllData"+ ".json");
    }
}
```

14. TopicDataStruct.cs

```
[System.Serializable]
public class TopicDataStruct
{
    public string headerName;
    public string time;
    public string points;
    public float timeDefault;
    public int isNextUnlocked;
}
```

15. TopicProgressDataLoader.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;

[Serializable]
public class TopicProgressStruct{
    [SerializeField]
    private string headerName;
```

```
    [SerializeField]
    private TopicDataLoader topicDataLoader;
    [SerializeField]
    private TextMeshProUGUI timeText;
    [SerializeField]
    private TextMeshProUGUI pointsText;
    [SerializeField]
    private int toFinished;
    [SerializeField]
    private bool isFinished = false;

    public bool IsFinished { get => isFinished; set => isFinished = value; }
    public TextMeshProUGUI PointsText { get => pointsText; set => pointsText =
value; }

    public void LoadData(){
        PointsText.text = topicDataLoader.GeneralPointText.text;
        SetTime();

        if(int.Parse(topicDataLoader.GeneralTopicText.text) >= toFinished){
            IsFinished = true;
        }
    }

    public void SetTime(){

        float time = topicDataLoader.TimeDefaultTotal;

        int minutes = Mathf.FloorToInt(time/60);
        int seconds = Mathf.FloorToInt(time%60);

        timeText.text = string.Format("{0:00}:{1:00}", minutes, seconds);
    }
}
```

```
public class TopicProgressDataLoader : MonoBehaviour
{
    [SerializeField]
    private TextMeshProUGUI progressPointsText;
    [SerializeField]
    private TextMeshProUGUI progressTopicsText;
    [SerializeField]
    private List<TopicProgressStruct> topicProgressStructs = new
List<TopicProgressStruct>();
    private int isFinished;
    private int totalPoints;
    public void LoadAllData()
    {
        isFinished = 0;
        totalPoints = 0;

        foreach(TopicProgressStruct t in topicProgressStructs){
            t.LoadData();
            totalPoints += int.Parse(t.PointsText.text);
            if(t.IsFinished){
                isFinished += 1;
            }
        }
        progressTopicsText.text = isFinished.ToString();
        progressPointsText.text = totalPoints.ToString();
    }
}
```

16. TransitionScript.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

```
public class TransitionScript : MonoBehaviour
{

    [SerializeField] private RectTransform transitionScreen;
    [SerializeField] private GameObject transitionScreenObject;
    [SerializeField] private RectTransform[] transitionPostion;
    [SerializeField] private GameObject screenActive;
    [SerializeField] private GameObject screenDeactive;
    [SerializeField] private float speed = 30;
    private int i = 1;
    //left: -1414, right: 1414
    void Start()
    {
        i = 1;
        speed = 30;
    }

    // Update is called once per frame
    void Update()
    {
        if(transitionScreenObject.activeSelf){
            transitionScreen.position = Vector3.MoveTowards(transitionScreen.position,
transitionPostion[i].position, (speed * Time.deltaTime)*100);
            if(transitionScreen.position == (transitionPostion[0].position +
transitionPostion[2].position) / 2){
                i++;
                screenActive.SetActive(true);
                screenDeactive.SetActive(false);
            }
            if(transitionScreen.position == transitionPostion[2].position){
                transitionScreenObject.SetActive(false);
            }
        }
```

```
    }

    public void SetPosition(GameObject screenActive, GameObject screenDeactive){
        this.screenActive = screenActive;
        this.screenDeactive = screenDeactive;
        transitionScreen.position = transitionPostion[0].position;
        i = 1;
        transitionScreenObject.SetActive(true);
    }
}
```

17. VideoPlayerStatus.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class VideoPlayerStatus : MonoBehaviour
{
    public bool isActionVideo = false;
    public bool isTopicEnd = false;
}
```

18. VoiceAudioPlay.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class VoiceAudioPlay : MonoBehaviour
{
    [SerializeField] private AudioClip audioClip;
    [SerializeField] private AudioSource audioSource;

    public void PlayVoice()
    {
```

```
        audioSource.clip = audioClip;
        audioSource.Play();
    }
}
```

APÊNDICE B - DOCUMENTO DE DESIGN DE JOGO DA APLICAÇÃO MY FRIEND LOGIC-X

O documento de design de jogo contempla capa, contracapa e demais imagens e diagramas presentes no capitulo 3 deste trabalho, visto isso, neste apêndice é apresentado o conteúdo escrito do documento no qual detalha a aplicação gamificada proposta.

Visão Geral

Tema

Aplicação gamificada para o auxílio em lógica de programação

Mecânicas básicas de jogabilidade

- Cutscene/Vídeos explicativos com desafios interativos no qual o jogador/usuário deverá responder corretamente tópicos e desafios relacionados.
- O usuário utilizará uma barra inferior com lista de opções para interagir com a aplicação.

Plataformas

Android(5.0 ou superior)

Modelo de aplicação

Open Source

Escopo do Projeto

- Custo e Prazo: Sem custo, 3 meses
- Equipe: Leonardo Henrique Lima de Souza
    - Responsabilidade: Programador, Designer, Audio Designer.

Descrição Geral do Projeto

My Friend Logic X é uma aplicação que utiliza da gamificação para engajar e auxiliar no ensino de lógica de programação para pessoas com dislexia. O objetivo é passar de maneira sólida o conceito e lógica por trás dos tópicos relacionados à lógica de programação sem utilizar linguagens de programação, para que assim, correlacionando a problemas do mundo real o usuário consiga consumir o conteúdo e utilizar de maneira eficaz.

A aplicação deve possuir um tela para configurações, onde o usuário poderá ajustar as opções referentes a acessibilidade para pessoas que possuem dislexia, opções como: tamanho da fonte, espaçamento, volume de áudio e voz. A aplicação deve conter telas para organizar os tópicos principais da lógica de programação assim como seus sub tópicos e desafios relacionados ao tema. Deve possuir tela para passagem de conhecimento, onde o usuário poderá interagir com a aplicação.

A aplicação utiliza armazenamento local para guardar os dados do desempenho do usuário, como, pontuação e tempo no tópico referente. A aplicação deve mostrar os dados do usuário nas telas relacionadas. Todos os dados devem ser guardados e carregados durante a utilização da aplicação.

Mecânicas de Jogabilidade

- Interação com a aplicação
    - A aplicação possui uma lista como uma barra inferior onde conterá uma lista de opções no qual poderá selecionar e interagir com a aplicação.
- Vídeo/Cutscene
    - O conteúdo é passado em forma de vídeo/cutscene na área central da tela de passagem de conteúdo/interação, dessa forma, o usuário poderá consumir o conteúdo e interagir com o mesmo.

História e Jogabilidade

Temas:

1. Estrutura da Sequência
2. Controle de Fluxo
3. Tipo de dados
4. Operadores
5. Funções

1. Estrutura da sequência

- Entrada de dados
- Desafio 1 - A Entrada
- Processamento
- Desafio 2 - O processo
- Saída de dados
- Desafio 3 - A saída

Exemplo de ensino:

- **Bolo**
    - Entrada: Ingredientes
    - Processamento: Equipamentos de preparo
    - Saída: Bolo

- **Construção**
    - Entrada: Materiais de construção
    - Processamento: Pedreiro/Máquinas
    - Saída: Casa

- **Veículo**
    - Entrada: Peças
    - Processamento: Montagem
    - Saida: Moto

- **Lista**: Ovos, Cimento, Motor, Batedeira, Betoneira, Máquina industrial, Bolo, Casa, Moto.

Roteiro:

Tópico 1 - Entrada de Dados:
A entrada de dados é a primeira etapa da estrutura da sequência, é nessa etapa que os dados necessários para uma operação são fornecidos. Dependendo do contexto, qualquer tipo de dados pode ser fornecido.

Vamos ao primeiro exemplo da entrada de dados! Imagine que você quer preparar um bolo especial, em primeiro momento é necessário que você separe os ingredientes. Consegue me dizer qual item da lista abaixo pode compor os ingredientes necessários para preparar o bolo?

Isso mesmo! O item selecionado faz parte dos ingredientes para preparar o bolo.

Vamos para o segundo exemplo da entrada de dados! Na construção de uma casa, são necessários vários tipos de materiais, onde cada um tem um objetivo específico. Consegue me dizer qual dos itens da lista é um material que pode compor a lista de materiais necessários para a construção da casa?

Isso mesmo! O item selecionado faz parte dos materiais necessários para a construção da casa.

Vamos para o terceiro e último exemplo da entrada de dados! Em uma fábrica onde são montados motos e veículos, cada um tem um tipo de peça que vai ser utilizada e vai compor o veículo final.
Consegue me dizer qual item da lista pode ser uma peça para compor o veículo?

Isso mesmo! O item selecionado faz parte das peças que compõem o veículo.

Excelente!!! Você entendeu perfeitamente que para algo ser preparado, construído ou montado, devem ser fornecidos os materiais correspondentes e necessários.

Assim é a entrada de dados, dados são fornecidos para posteriormente serem utilizados na segunda etapa da estrutura da sequência que é chamada de processamento, que veremos no próximo tópico.

Desafio 1: Utiliza a mesma mecânica mas com exemplos diferentes do explicado e sem narração.

Tópico 2 - Processamento:

O processamento é a segunda etapa da estrutura da sequência, é nessa etapa que os dados fornecidos na entrada são operados. Dependendo do contexto, qualquer tipo de atividade pode ser feita com os dados fornecidos.

Vamos ao primeiro exemplo do processamento! Para preparar um bolo, foram fornecidos vários materiais, como ovos, farinha e fermento. Mas os ingredientes não vão se misturar sozinhos, são necessárias ferramentas para iniciar o processo de preparo do bolo.
Consegue me dizer qual item da lista é uma ferramenta necessária para preparar a mistura do bolo?

Isso mesmo! A batedeira vai ser utilizada no processamento dos ingredientes fornecidos para preparar a mistura do bolo.

Vamos para o segundo exemplo do processamento! Para construir uma casa, foram entregues os materiais necessários para compor a casa, mas a casa não vai ser construída sozinha. São necessárias pessoas capacitadas e essas pessoas precisam de ferramentas para iniciar o processo de construção da casa.
Consegue me dizer qual item da lista é uma ferramenta para os trabalhadores iniciarem a construção?

Isso mesmo! A betoneira vai ser utilizada no processo de construção da casa, utilizando os materiais fornecidos para preparar o cimento.

Vamos para o terceiro e último exemplo do processamento! Para montar veículos, foram entregues peças para compor os mesmos, mas essas peças não se encaixam

sozinhas, são necessárias ferramentas para iniciar o processo de montagem dos veículos.
Consegue me dizer qual item da lista é uma ferramenta que pode iniciar o processo de montagem dos veículos?

Isso mesmo! Esse robô industrial vai dar início ao processo de montagem dos veículos utilizando as peças fornecidas.
É isso aí! Você entendeu perfeitamente que para alcançar um objetivo, resultado ou um produto, é necessário algum processo de preparo, construção ou qualquer outra ação que possa utilizar os dados de entrada fornecidos. Dessa forma, após o processamento teremos um resultado final que é a saída de dados, que veremos no próximo tópico.

Desafio 2: Utiliza a mesma mecânica mas com exemplos diferentes do explicado e sem narração.

Tópico 3 - Saída

A saída de dados é a terceira e última etapa da estrutura da sequência, é nesta parte onde temos o retorno do processamento de dados. A etapa é simples, mas ela tem uma condição, ao fornecer dados para serem processados para um resultado esperado, a saída deve ser exatamente esse resultado esperado. Caso contrário, existe algo errado no processamento.

Dessa forma, vamos para o primeiro exemplo de saída de dados! Após fornecer ingredientes para preparar um bolo, o processo de preparo foi iniciado utilizando morangos, no fim o bolo foi levado ao forno.

Consegue me dizer qual item da lista representa o bolo preparado?
Isso mesmo! Um bolo de morango foi o resultado, bem simples, né?

Vamos para o segundo exemplo de saída de dados! Após fornecer o material de construção dos quais foram tijolos, cimento e outros materiais. Uma casa incrível foi levantada.
Consegue me dizer qual item da lista representa o tipo de casa construída?

Isso mesmo! Uma casa de alvenaria. Bem tranquilo, né?

Vamos para o terceiro e último exemplo de saída de dados! A fábrica estava a todo vapor montando veículos após as peças serem fornecidas. Mas o tipo de peças fornecidas não pareciam ser de carros.
Consegue me dizer qual item da lista representa o tipo de veículo montado?

Isso mesmo! Uma moto, essa foi mais fácil, né?

Você conseguiu! Entendeu perfeitamente como funciona a estrutura da sequência, composta por entrada, processamento e saída de dados. Agora, que tal um desafio? Essa é só com você, o desafio sobre saída de dados te aguarda.

Desafio 3: Utiliza a mesma mecânica mas com exemplos diferentes do explicado e sem narração.

2. Controle de Fluxo

- Estrutura de Controle Condicional (Se, Senão, Então Se)
- Desafio 1.
- Estrutura de Repetição(Para, Enquanto)
- Desafio 2.

Exemplo de ensino:

- **Estrutura de Controle Condicional (Se, Senão, Então Se)**
    - Furto(Se): Se o ladrão entrar na caasa o alarme deve ser acionado;
    - Tráfego(senão): Se o sinal estiver verde, os veículos devem trafegar, senão, eles devem parar;
    - Restaurante(Então Se): Se o cliente quiser uma carne, ofereça a carne, então se o cliente quiser um prato vegano, ofereça um prato vegano;

- **Estrutura de Repetição(Para, Enquanto)**

    - Produção de Bolo (For): Produção de bolo com condição de parada da repetição;
    - Produção de Bolo (Enquanto): Enquanto houver ingrediente produza bolo;

- **Lista**: Alarme, Sinal verde, Sinal vermelho, Prato carne, Prato vegano, Número de inicio, Numero de parada, ingrediente de bolo(sabor);

Roteiro:

Tópico 1 - Estrutura de Controle Condicional (Se, Senão, Então Se)

O controle de fluxo é uma parte fundamental da lógica de programação que permite que um programa execute diferentes instruções em diferentes situações, com base em condições específicas. Ele ajuda a controlar a ordem em que as instruções são executadas e a tomar decisões dentro do programa. O controle de fluxo é composto por estrutura de controle condicional e estrutura de repetição.

Vamos à primeira estrutura de controle condicional, onde utilizaremos a estrutura “SE” ou “IF” em inglês. A estrutura IF ou SE permite que você execute um bloco de código se uma condição for verdadeira.

Imagine uma pessoa que está dormindo tranquilamente, quando de repente um invasor adentra sua casa. Mas a casa dessa pessoa tem alarme e sensor de movimento. Nesse caso “SE” o invasor entrar o que o sistema de segurança deve fazer? Selecione da lista abaixo, o item que corresponde a ação que deve ser realizada.

Isso mesmo! Se caso o invasor entre na casa, o sistema de segurança deve soar o alarme!

Agora vamos para a segunda estrutura de controle condicional, como falado anteriormente, a estrutura SE ou IF permite que você execute um bloco de código se uma condição for verdadeira. Mas o que acontece se não for verdadeira?

Nesse caso, a estrutura SE pode ser seguida por uma cláusula SENÃO ou ELSE para executar uma ação alternativa se a condição não for verdadeira.

Imagine um tráfego de veículos onde tem um semáforo, SE o semáforo estiver “verde” os carros podem seguir, mas caso contrário, se o semáforo não estiver verde o que eles devem fazer?

Isso mesmo! Os veículos devem parar, isso porque a condição que verifica para os veículos seguirem o caminho é falsa.

Agora vamos para a terceira e última estrutura de controle condicional. A estrutura de controle condicional pode ainda ser seguida por uma cláusula SE ou IF, após o SENÃO, que ficaria algo como, SENÃO SE ou ELSE IF.

Dessa forma, será executada uma ação ou processo alternativo mas que obedece uma condicional específica diferente da verificada anteriormente.

Desafio 1: Utiliza a mesma mecânica mas com exemplos diferentes do explicado e sem narração.

Tópico 2 - Estrutura de Repetição(Para, Enquanto)

Uma estrutura de repetição, também conhecida como loop ou laço, é uma construção fundamental na lógica de programação que permite executar uma ação ou processo várias vezes com base em uma condição específica. Essa condição é avaliada a cada iteração do laço, e o processo ou ação dentro do loop é executado repetidamente até que a condição se torne falsa.

Para(For)

Agora vamos ver a primeira estrutura de repetição chamada “For” ou “Para”. No qual é utilizado para executar uma ação ou processo diversas vezes mas com uma condição definida.

Vamos montar juntos essa condição para a estrutura de repetição FOR. A estrutura de repetição FOR é composta por condição de início, condição de parada e incremento. A condição de parada é feita utilizando a variável auxiliar de iteração ou contador, a partir dela onde será possível verificar também a condição de parada. A condição de parada, como dito anteriormente, utiliza a mesma variável auxiliar, no momento que o valor da variável auxiliar atender a condição de parada, o laço de repetição estará concluído e não será mais executado. Por fim, o incremento, onde é

definido qual será o tipo atribuição de valor para a variável auxiliar, se é incremento onde o valor é adicionado de um em um, ou decremento, se o valor é subtraído de um em um.

Agora, imagine uma fábrica de bolos, no qual deve preparar 30 bolos de morango, primeiro devemos entender que todos os bolos devem ser preparados, dessa forma vamos definir o valor da propriedade de início do laço.

Consegue me dizer qual dos itens da lista abaixo definem que o laço possa iniciar a partir do primeiro bolo a ser produzido?

Isso mesmo, X = 0. Deve iniciar do zero, por que nenhum bolo foi produzido, no momento que o laço for completo o iterador será igual a 1 mostrando que uma iteração foi realizada.

Agora para a condição de parada, qual dos itens abaixo define quando o laço deve parar ao alcançar 30 bolos produzidos?

Isso mesmo, quando for X > 30! Isso porque o valor 30 será alcançado após o processo do bolo 30 ser realizado. Dessa forma o laço irá parar quando o contador chegar em 30.

Finalmente vamos definir o nosso incrementador. Selecione na lista abaixo se nosso laço deve incrementar o contador ou decrementar para que o mesmo possa iterar 30 vezes para produzir bolos.

Isso mesmo! O item X += 1 incrementa a variável auxiliar que é nosso contador.

Você entendeu perfeitamente como funciona a estrutura de repetição FOR ou PARA, conseguiu entender o que é a condicional de início, parada e o incrementador da estrutura. Muito bem!

Enquanto(While)

A estrutura de repetição ENQUANTO ou WHILE, é bem simples, ela irá executar o laço de repetição ENQUANTO uma condição for verdadeira.

Vamos utilizar o exemplo da fábrica de bolos novamente. O laço deve ser executado para produzir os 30 bolos de morango, no momento que os bolos foram

produzidos, o laço deve parar. Então, o laço deve ser executado ENQUANTO todos os bolos ainda não forem produzidos. No nosso exemplo, o nosso incrementador é incrementado dentro do laço de repetição em X += 1.

Consegue me dizer qual item da lista abaixo pode definir a condição de execução para a estrutura de repetição?

Isso mesmo, enquanto X < 30. Dessa forma o laço de repetição será executado enquanto os 30 bolos não forem produzidos.

E é exatamente assim que a estrutura de repetição ENQUANTO(while) funciona, você entendeu perfeitamente!

Desafio 2: Utiliza a mesma mecânica mas com exemplos diferentes do explicado e sem narração.

3. Tipos de Dados

- Inteiro, Flutuante e Texto;
- Desafio 1;
- Booleano, Caractere e Nulo;
- Desafio 2;

Exemplo de ensino:

- Inteiro, Flutuante e Texto;
  Associar um inteiro a uma variável do tipo int
  Associar um número flutuante ao uma variável float
  Associar um texto a uma variável do tipo string

- Booleano, Caractere e Nulo;
  Associar verdadeiro ou falso a uma variável do tipo bool
  Associar um Caractere a uma variável do tipo Char
  Associar Nulo a uma variável que não recebeu valor

- **Lista**: Número inteiro, Número Decimal, Um nome, Alarme ligado (verde), Qualquer texto, Suporte de papel higiênico vazio.

Roteiro:

Tópico 1 - Inteiro, Flutuante e Texto

Na lógica de programação utilizamos vários tipos de dados para fazer vários tipos de operações, nesse tópico abordaremos os tipos de dados mais utilizados.

Nessa primeira parte abordaremos tipos inteiros, flutuantes e textos.
Então vamos ao nosso primeiro exemplo.
Imagine que temos um contador de um laço de repetição, qual tipo de dado podemos utilizar para nossa variável que deve ser contada de 1 a 10?

Isso mesmo, do tipo inteiro! Como queremos que faça a contagem de 1 a 10, ela deve ser incrementada 10 vezes. Números inteiros são aqueles tipos de número que não possuem vírgula ou não são decimais. Os números inteiros são úteis quando queremos utilizar um contador por inteiro.

Vamos ao segundo exemplo, imagine que você tem uma maçã mas quer dividir ela duas partes iguais, uma parte para você e outra para seu amigo. Qual tipo de dado representa o resultado numérico da divisão da maçã por dois?

Isso mesmo! O resultado é um flutuante que é um número decimal, a maçã é representada por 1 e o resultado é 0,5. Dessa forma, ela foi dividida com precisão para duas pessoas. Um número inteiro não poderia representar essa divisão por não ser um número decimal.

Vamos ao terceiro e último exemplo, vamos utilizar os exemplos anteriores. Imagine duas situações, o contador e a divisão da maçã. Nessa situação você está contando em quanto tempo consegue dividir a maçã para duas pessoas e esse resultado e tempo e divisão deve ser armazenado em um único tipo de dados. Consegue me dizer qual tipo de dado pode representar ambos os tipos de resultados de uma vez?

Isso mesmo! O tipo de texto pode representar o resultado. O tipo de texto aceita qualquer tipo de dado, mas ele funciona de forma a ser representativa, se você atribuir um número para um tipo texto, posteriormente você não conseguirá fazer operações matemáticas com esse número.

Tópico 2 - Booleano, Caractere e Nulo

Nesta segunda parte de tipos de dados, veremos os tipos booleano que é um tipo que aceita apenas dois valores, se é verdadeiro ou falso, o tipo caractere que aceita individualmente qualquer tipo de caracteres e nulo que é quando não é atribuído nenhum tipo de valor a uma variável.

Vamos ao primeiro exemplo, imagine um sistema de segurança no qual deve estar ativado quando for necessário. Você consegue me responder se é verdadeiro ou falso o sistema de segurança ficar ativado a noite?

Isso mesmo! É verdadeiro que o sistema de segurança deve estar ativado à noite. O valor booleano é apenas verdadeiro ou falso, é utilizado principalmente em condicionais, que foi o caso do exemplo, onde um alarme sempre deve estar ativado a noite.

Vamos ao segundo exemplo, os caracteres são a composição de um texto, eles podem ser qualquer coisa mas o tipo de de dados e operações feitas neles são diferentes, uma expressão matemática pode ser representada por caracteres mas a operação pode não ser feita diretamente nele. Dessa forma, consegue me dizer qual item da lista é um caracter?

Isso mesmo! Qualquer item da lista é um caracter!

Vamos ao terceiro e último exemplo. Imagine uma situação inusitada no banheiro, onde uma pessoa com dor de barriga foi correndo para o banheiro, ao chegar lá ela olhou para o suporte de papel higiênico e viu que o mesmo tinha valor nulo. Qual item da lista pode representar a situação nula para o suporte de papel higiênico?

Isso mesmo! O suporte vazio, isso por que um valor nulo é quando há uma variável que no caso é o suporte mas não foi instalado nenhum item ou valor, nesse caso é nulo. Se caso ainda houvesse rolo de papel higiênico sem o papel, seria vazio.

4. Operadores

- Operadores aritméticos
- Desafio 1.
- Operadores de comparação
- Desafio 2.

Exemplo de ensino:

- Associar operadores aritméticos à funções e equações matemáticas(adição, subtração, divisão e multiplicação);
- Associar operadores de comparação a condicional(==, >=, <=, !=).
- **Lista**: Sinal de adição, Sinal de subtração, sinal de multiplicação, sinal de divisão e subtração, Sinal maior igual, Sinal de diferença, Sinal de igualdade.

Roteiro:

Tópico 1 - Operadores aritméticos(Adição, Subtração, Multiplicação e Divisão)

Operadores em lógica de programação são os mesmos usados na matemática, nesse primeiro tópico, vamos abordar sobre operadores aritméticos, dos quais são, adição, subtração, multiplicação e divisão. Esse tópico será bem simples e não terá nenhum tipo de assunto novo, porque provavelmente você já deve ter visto esse tipo de atribuição.

Vamos ao primeiro exemplo, em um programa de computador temos vários processos, nesse primeiro processo é passado como valor o número 2 e o número 4, no qual será retornado o resultado 6. Consegue me dizer qual operador deve ser usado para retornar o valor 6 como resultado?

Isso mesmo, o operador de adição.

Agora vamos para o segundo exemplo. Dentro de um outro processo é feita uma equação, no qual retorna o valor 7, consegue me dizer quais os operadores que compõem essa equação para que o resultado retornado seja 7?

Isso mesmo, operador de adição e divisão!

Nosso último exemplo, na equação 5-10*2, tem o resultado igual a 15, consegue me dizer quais operadores devem ser utilizados para o resultado seja 15?

Isso mesmo, o operador de subtração e multiplicação!

Viu só, você conseguiu entender como utilizar operadores, onde provavelmente você já sabia. Em linguagens de programação o sinal de adição e subtração permanecem os mesmo usados convencionalmente, enquanto de multiplicação e divisão são representados por asteriscos e uma barra.

Você conseguiu acertar tudo, parabens!

Desafio 1: Utiliza a mesma mecânica mas com exemplos diferentes do explicado e sem narração.

Tópico 2 - Operadores de comparação(==, >=, <=, !=)

Neste tópico abordaremos os operadores de comparação que são elementos fundamentais na programação que permitem comparar valores ou expressões e determinar se uma determinada relação entre eles é verdadeira ou falsa. Eles são amplamente utilizados em construções condicionais, como instruções "if", "else if" e "while loops", para tomar decisões com base em condições.

A igualdade é representada pelo operador ==, para maior igual >=, para menor igual <= e diferente de !=.

Vamos ao primeiro exemplo. Imagine uma verificação em um bloco condicional if. Onde é necessário verificar o valor de uma variável. Valor atribuído a variável é 7 e para o bloco condicional ser verdadeiro e ser executado o valor deve

ser maior ou igual a 5. Consegue me dizer qual operador podemos utilizar para atender os requisitos da verificação?

Isso mesmo! O operador de maior ou igual >=.

Vamos ao segundo exemplo, Impinge um bloco condicional onde o valor atribuído a uma variável deve ser sempre diferente de zero, para que a divisão dentro do bloco possa ser executada. Qual operador podemos utilizar para atender os requisitos da verificação?

Isso mesmo! O operador de diferente de !=;

Vamos ao terceiro e último exemplo. Imagine agora uma senha, essa senha será definida em uma variável de texto, e para entrar no bloco condicional a senha deve ser idêntica a cadastrada no sistema. Consegue me dizer qual operador podemos utilizar para atender os requisitos da verificação?

Isso mesmo! O operador de igualdade ==;

Desafio 2: Utiliza a mesma mecânica mas com exemplos diferentes do explicado e sem narração.

5. Funções

- Parâmetros, Valor de retorno e Corpo da função.
- Desafio 1.

Exemplo de ensino:

- Associar valores ao tipo de parâmetro;
- Associar operadores para o processamento no corpo da função;
- Associar valor ao tipo do valor de retorno da função.
- **Lista**: Decimais(Float), X+Y, Outros itens para compor a lista.

Roteiro:

Tópico 1 - Parâmetros, Valor de retorno e Corpo da função

Funções em lógica de programação são blocos de código que são projetados para realizar uma tarefa específica. Elas são fundamentais na organização e modularização do código, permitindo que tarefas repetitivas ou complexas sejam encapsuladas em um único bloco de código, o qual pode ser chamado múltiplas vezes a partir de diferentes partes do programa. Neste tópico iremos montar nossa função por partes, passaremos o valor de parâmetro, o corpo da função e o valor de retorno.

Vamos iniciar, a nossa função faz parte de um programa de calculadora, onde a responsabilidade da função é somar dois números.

Primeiramente iremos chamar o nome da nossa função de Soma. Para recebermos os valores diretamente para a função, utilizamos variável que chamamos de parâmetros da função, na qual serve de caminho para a passagem de valores por outras partes do código ou sistema. Nesse caso precisamos somar qualquer tipo de número. Consegue me dizer qual tipo de variável da lista abaixo é um tipo no qual podemos atribuir valor da soma de números decimais?

Isso mesmo! Utilizaremos números flutuantes, dessa forma poderemos somar números decimais e não limitar a inteiros.

Vamos a segunda parte da nossa função. No corpo da função é necessário somar os dois números que estão sendo passados por parâmetro, dos quais serão armazenados em uma variável de mesmo tipo, consegue me dizer qual item da lista abaixo representa a equação que atende os requisitos da nossa função?

Isso mesmo! A equação é x+y!

Agora vamos a terceira e última parte da nossa função. Até agora pegamos os valores e somamos eles, mas precisamos retornar esse resultado. O retorno deve ser correspondente ao tipo de valor gerado. Consegue me dizer qual item da lista abaixo representa o tipo de valor retornado pela nossa função?

Isso mesmo! A nossa função retorna um valor do tipo flutuante!

Viu só? Foi bem fácil montar a nossa função, passamos por todas as partes principais, que são os valores de parâmetro, o corpo da função e o tipo de retorno. É exatamente assim que funciona uma função, valores são atribuídos e passados a ela, e no corpo da função é feito o processo de ação da mesma, você pode implementar verificações, equações e qualquer tipo de processo dentro de uma função. Uma função pode retornar ou não um valor, mas lembre que ao retornar, o tipo deve ser igual ao tipo do resultado processado pela função.

Desafio 1: Utiliza a mesma mecânica mas com exemplos diferentes do explicado e sem narração.