INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA

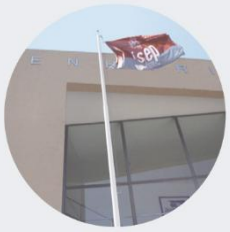



Desenvolvimento de Software para Auxílio na Aprendizagem do Pensamento Computacional em Comunidades em Situação de Vulnerabilidade Social na Amazônia

WENDERSON RONIERE BASTOS SILVA
Novembro de 2024

POLITÉCNICO
DO PORTO

INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Desenvolvimento de Software para Auxílio na Aprendizagem do Pensamento Computacional em Comunidades em Situação de Vulnerabilidade Social na Amazônia Legal Maranhense.

WENDERSON RONIERE BASTOS SILVA
Novembro de 2024

POLITÉCNICO
DO PORTO

Desenvolvimento de Software para Auxílio na Aprendizagem do Pensamento Computacional em Comunidades em Situação de Vulnerabilidade Social na Amazônia Legal Maranhense.

Wenderson Roniere Bastos Silva

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas Gráficos e Multimédia

Supervisor: Prof. Dr. Aristóteles de Almeida Lacerda Neto
Co-Supervisor: Prof(a). Dra. Paula Escudeiro

Santa Inês, Novembro 2024

Declaração de Integridade

Declaro ter conduzido este trabalho académico com integridade. Não plagiei ou apliquei qualquer forma de uso indevido de informações ou falsificação de resultados ao longo do processo que levou à sua elaboração. Portanto, o trabalho apresentado neste documento é original e de minha autoria, não tendo sido utilizado anteriormente para nenhum outro fim. Declaro ainda que tenho pleno conhecimento do Código de Conduta Ética do P. PORTO.
ISEP, Porto, 15 de Novembro de 2024.

RESUMO

Na Amazônia Legal do Maranhão, temos muitos problemas relacionados à educação. Isso é mais evidente em comunidades remotas e carentes, onde infelizmente vários fatores dificultam o acesso a uma educação de qualidade. É evidente que precisamos desenvolver um software para a área da educação, mais especificamente no campo da informática, para ensinar a este público lógicas básicas de programação. Nosso objetivo é em parte superar esses problemas e promover a inclusão digital.

Este presente trabalho tem o objetivo de democratizar o conhecimento e igualar as oportunidades educacionais entre os estudantes da região e de áreas mais favorecidas geograficamente e economicamente. Oferecemos um caminho de aprendizado de lógica de programação acessível e inclusivo. Como resultado, capacitando o público com habilidades em programação e informática para criar soluções tecnológicas para os desafios locais, o presente trabalho visa incentivar o pensamento crítico, promover o empreendedorismo local e contribuir para a sustentabilidade ambiental.

Palavras chave: Amazônia, programação, necessidade, desafio, software

ABSTRACT

In the Legal Amazon of Maranhão, we have many problems related to education. This is most evident in remote and underprivileged communities, where unfortunately various factors hinder access to quality education. It is clear that we need to develop software for education, more specifically in the field of computer science, to teach this public basic programming logic. Our aim is partly to overcome these problems and promote digital inclusion.

This work aims to democratize knowledge and equalize educational opportunities among students from the region and from more geographically and economically advantaged areas. We offer an accessible and inclusive way of learning programming logic. As a result, by empowering the public with programming and computer skills to create technological solutions to local challenges, this work aims to encourage critical thinking, promote local entrepreneurship and contribute to environmental sustainability.

Keywords: Amazon, programming, need, challenge, software

Agradecimentos

Quero agradecer ao Prof. Dr. Aristóteles de Almeida Lacerda Neto por todo apoio e acompanhamento durante esta jornada e a Prof(a). Dra. Paula Escudeiro pelo apoio e acompanhamento durante esta etapa de conclusão do projeto . Aos meus pais, irmão, namorada e amigos por me fornecerem todo apoio e compreensão durante essa caminhada . Aos meus colegas e companheiros de turma que me ajudaram e colaboraram para que esse momento fosse possível.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Gráficos

Acrónimos e Símbolos

Lista de Acrónimos

ITU	*International Telecommunication Union*
MIT	*Massachusetts Institute of Technology*
UNESCO	*United Nations Educational, Scientific and Cultural Organization*
UNICEF	*United Nations Children's Fund*
OECD	*Organisation for Economic Co-operation and Development*
BBC	*British Broadcasting Corporation*

1 Introdução

A tecnologia da informação(T.I) tem se firmado como uma das principais áreas de desenvolvimento e inovação na nossa atual sociedade. Em especial, a lógica de programação desempenha um papel muito importante ao proporcionar habilidades essenciais para a resolução de problemas, pensamento crítico e criatividade. Porém, a democratização do acesso a esse conhecimento infelizmente ainda encontra desafios significativos, especialmente em regiões mais isoladas e carentes de recursos educacionais, como a Amazônia Legal Maranhense.

A Amazônia Legal Maranhense, uma das áreas mais ricas em biodiversidade e cultura do Brasil, enfrenta diversas barreiras socioeconômicas e geográficas que dificultam a oferta de uma educação de qualidade. A falta de infraestrutura adequada, o acesso limitado a tecnologias modernas e a carência de profissionais qualificados são alguns dos obstáculos que impedem o desenvolvimento pleno do potencial educacional das crianças e jovens dessa região.

1.1 Contexto

Diante desse cenário, surge a necessidade de iniciativas inovadoras que possam superar essas barreiras e promover a inclusão digital. Este trabalho apresenta o desenvolvimento de um aplicativo educacional voltado para o ensino de lógica de programação, destinado especificamente às crianças e jovens da Amazônia Legal Maranhense. O objetivo é fornecer uma ferramenta acessível e eficiente que possibilite o aprendizado de conceitos fundamentais de programação, estimulando o interesse pela tecnologia e preparando esses jovens para os desafios do século XXI.

O aplicativo será concebido levando em consideração as particularidades culturais e sociais da região, bem como a necessidade de métodos de ensino adaptativos que possam engajar os alunos de forma lúdica e interativa.

Utilizando uma abordagem pedagógica baseada em jogos e atividades práticas, o aplicativo visa tornar o aprendizado da programação uma

experiência divertida e envolvente, ao mesmo tempo em que promove o desenvolvimento de competências essenciais para o futuro.

A presente tese detalha o processo de desenvolvimento do aplicativo, desde a concepção inicial até a implementação e os resultados obtidos em testes preliminares com estudantes da região. Além disso, são discutidas as implicações educacionais e sociais do projeto, bem como suas possíveis contribuições para a redução da desigualdade educacional e o empoderamento das comunidades locais.

1.2 Problema

A região da Amazônia Maranhense se notabiliza por ter uma grande extensão territorial aliada a uma grande diversidade cultural, encontra barreiras notáveis no quesito da educação, isso fica mais evidente em comunidades carentes e remotas. Um bom acesso a uma educação de qualidade e oportunidades de desenvolvimento são demasiadamente limitados, deixando uma quantidade importante de crianças e jovens em desvantagem em relação ao restante do país.

Deste modo, a ideia de desenvolver um aplicativo educacional focado no ensino de programação surge como um estratégia diferente e relevante, com um grande potencial de produzir uma transformação marcante, além disso, é fundamental ressaltar a relevância educacional intrínseca a esta pesquisa. A programação, no cenário do século XXI, emerge como uma habilidade essencial (OECD, 2021). O aumento da demanda de profissionais de tecnologia faz com que o aprendizado de programação não seja apenas desejável , mas também uma qualidade valiosa para a empregabilidade e para o pleno engajamento na sociedade digital.

O objetivo de encontrar a inclusão digital torna-se, assim, um dos pontos fortes dessa pesquisa. A concepção e desenvolvimento de um aplicativo educacional propõem um meio acessível e inclusivo para que crianças e jovens da região tenham acesso a conteúdo de qualidade em programação.

Não levando em conta nenhuma condição socioeconômica ou localização geográfica, essa proposta pretende disponibilizar a democratização do conhecimento, igualando as oportunidades educacionais entre estudantes da Amazônia maranhense com outras regiões mais favorecidas

Assim a está pesquisa busca trabalhar o pensamento crítico dos alunos “A programação não se limita a ensinar habilidades técnicas; ela também promove o desenvolvimento de habilidades cognitivas, como o pensamento lógico, a resolução de problemas e a criatividade" (Papert, 1993). Tais competências são primordiais para o sucesso acadêmico e profissional, preparando este público da Amazônia Legal Maranhense para enfrentar os desafios de um mundo cada vez mais digitalizado e complexo.

A nossa proposta almeja também promover o empreendedorismo e a criação de mais oportunidades locais. Ao capacitar crianças, jovens e adultos da Amazônia legal maranhense, com habilidades e conhecimentos em programação abre-se um leque de possibilidades no campo do empreendedorismo tecnológico. Ainda nesta linha a pesquisa pode contribuir para a sustentabilidade ambiental da região. Pensando em um perspetiva a longo prazo e no sucesso do projeto, o desenvolvimento de habilidades em programação permite que os jovens locais criem soluções tecnológicas voltadas para a resolução de questões ambientais locais, apoiando, assim, a preservação desse ecossistema único e essencial para o equilíbrio global.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo geral deste trabalho é desenvolver um aplicativo educacional eficaz e acessível que auxilie na aprendizagem de programação, visando capacitar crianças e jovens da Amazônia maranhense com habilidades tecnológicas e promover a inclusão digital, o pensamento crítico, o empreendedorismo e a contribuição para a sustentabilidade ambiental.

1.3.2 Objetivos Específicos

1.3.2.1 Avaliar a Eficácia do Aplicativo

Realizar uma avaliação rigorosa da eficácia do aplicativo em termos de aprendizagem de programação e desenvolvimento de habilidades cognitivas, como o pensamento lógico e a resolução de problemas, por meio de estudos de caso e testes práticos.

1.3.2.2 Promover a Implementação e a Adoção do Aplicativo

Estabelecer parcerias com escolas, organizações comunitárias e autoridades locais na Amazônia maranhense para garantir a implementação eficaz do aplicativo em ambientes educacionais e comunitários, incentivando sua adoção e uso regular por parte de crianças e jovens.

1.4 Metodologia

A metodologia desta pesquisa é estruturada em quatro fases principais: concepção do aplicativo, desenvolvimento técnico, criação de material didático, e aplicação e avaliação. Cada fase é detalhada a seguir:

1.4.1 Concepção do Aplicativo

Objetivo: Definir os requisitos e funcionalidades do aplicativo educacional, garantindo que ele atenda às necessidades específicas das crianças e jovens da Amazônia Legal Maranhense.

Atividades:

- **Análise de Necessidades:** Realização de entrevistas e questionários com professores, estudantes e especialistas em educação da região para identificar as principais dificuldades e necessidades educacionais.

- **Definição de Funcionalidades:** Com base na análise de necessidades, definição das funcionalidades essenciais do aplicativo, como jogos educativos, quizzes interativos, tutoriais de programação e fóruns de discussão.
- **Elaboração do Plano de Projeto:** Criação de um cronograma detalhado e definição dos recursos necessários para o desenvolvimento do aplicativo.

1.4.2. Desenvolvimento Técnico

Objetivo: Desenvolver o aplicativo utilizando o software Unity e a linguagem de programação C#, garantindo sua funcionalidade, usabilidade e adequação ao público-alvo.

Atividades:

- **Configuração do Ambiente de Desenvolvimento:** Instalação e configuração do software Unity e dos ambientes de desenvolvimento necessários para a programação em C#.
- **Desenvolvimento de Módulos:** Criação de módulos educacionais interativos, incluindo:

    - **Tutoriais Interativos:** Passo a passo para ensinar conceitos básicos de lógica de programação.
    - **Jogos Educativos:** Jogos que incentivem a prática de programação de forma lúdica.
    - **Quizzes e Exercícios:** Testes para avaliar o aprendizado e reforçar os conceitos ensinados.

- **Testes e Debugging:** Teste contínuo dos módulos para garantir a funcionalidade correta e resolução de bugs.

1.4.3. Criação de Material Didático

Objetivo: Desenvolver uma apostila complementar que ensine a usar o aplicativo e os conceitos básicos de lógica de programação.

Atividades:

- **Elaboração da Estrutura da Apostila:** Definição dos tópicos e organização do conteúdo, garantindo uma progressão lógica e gradual.
- **Desenvolvimento do Conteúdo:** Escrita dos capítulos da apostila.
- **Revisão e Edição:** Revisão do conteúdo da apostila para garantir clareza, precisão e adequação ao público-alvo.
- **Design e Formatação:** Formatação da apostila para impressão e distribuição digital, com a inclusão de ilustrações, gráficos e diagramas explicativos.

1.4.4. Aplicação e Avaliação

Objetivo: Implementar o aplicativo e a apostila em escolas da Amazônia Legal Maranhense e avaliar seu impacto educacional.

Atividades:

- **Piloto de Implementação:** Seleção de escolas para um projeto piloto, onde o aplicativo e a apostila serão introduzidos.
- **Monitoramento e Suporte:** Acompanhamento contínuo das escolas participantes, fornecendo suporte técnico e educacional.
- **Avaliação de Impacto:** Coleta de dados por meio de questionários, entrevistas e análises de desempenho dos estudantes para avaliar o impacto do aplicativo no aprendizado de programação.
- **Análise dos Resultados:** Análise qualitativa e quantitativa dos dados coletados para identificar pontos fortes, áreas de melhoria e o impacto geral do projeto.

Esta metodologia visa garantir um desenvolvimento estruturado e eficaz do aplicativo educacional, bem como uma implementação bem-sucedida nas escolas da Amazônia Legal Maranhense. Através de um processo iterativo e colaborativo, esperamos não apenas ensinar lógica de programação, mas também promover a inclusão digital e o desenvolvimento de habilidades essenciais para o século XXI.

2. Estado da Arte

A região da Amazônia Legal Maranhense, marcada por uma rica diversidade natural e cultural, também enfrenta desafios significativos no campo da educação. Crianças, jovens e adultos que vivem nessas comunidades, muitas vezes em situação de vulnerabilidade social, deparam-se com obstáculos que vão desde a falta de infraestrutura escolar até a escassez de recursos educacionais adequados. Essas barreiras impedem o pleno desenvolvimento cognitivo, técnico e criativo desses indivíduos, perpetuando um ciclo de desigualdade e exclusão.

A infraestrutura educacional precária, aliada à falta de investimentos governamentais e à escassez de programas educacionais personalizados para atender às necessidades específicas das comunidades amazônicas, contribui para uma disparidade no acesso ao conhecimento e às oportunidades. Enquanto em outras partes do mundo o pensamento computacional é reconhecido como uma habilidade essencial para o sucesso no século XXI, muitas dessas comunidades continuam enfrentando dificuldades para acessar recursos educacionais que promovam essa competência crucial.

Nesse contexto, o desenvolvimento de software educacional adaptado às particularidades da região da Amazônia Legal Maranhense surge como uma medida imperativa e urgente. Esse software deve ser projetado não apenas para ensinar conceitos de pensamento computacional, mas também para fazê-lo de maneira inclusiva, acessível e culturalmente relevante. É fundamental que essas ferramentas educacionais considerem não apenas as habilidades técnicas, mas também as experiências, valores e tradições das comunidades locais, garantindo assim sua eficácia e aceitação.

Portanto, o objetivo desta revisão é explorar o estado atual dos desenvolvimentos de softwares educacionais voltados para a promoção do pensamento computacional em comunidades em situação de vulnerabilidade.. Ao analisar iniciativas existentes, identificar lacunas e propor diretrizes para futuras pesquisas e intervenções, esperamos contribuir para a criação de soluções mais eficazes e inclusivas que promovam o desenvolvimento integral e sustentável dessas comunidades.

"A tecnologia pode ser uma aliada poderosa na promoção da educação inclusiva, oferecendo recursos e ferramentas adaptadas às necessidades individuais dos alunos" (E*uropean Commission*, 2021). A tecnologia possui o potencial de personalizar e individualizar o processo de ensino-aprendizagem, permitindo que cada aluno receba o suporte e os recursos necessários para seu desenvolvimento acadêmico e pessoal. Ao oferecer soluções acessíveis e adaptáveis, a tecnologia pode contribuir significativamente para a promoção da inclusão e para o fortalecimento da igualdade de oportunidades na educação.

"A educação digital é uma ferramenta poderosa para superar barreiras geográficas e sociais, ampliando o acesso à educação em comunidades remotas e marginalizadas" (*Schofield*, 2018). Em regiões onde o acesso à educação é limitado devido a barreiras geográficas ou sociais, a educação digital pode desempenhar um papel transformador, oferecendo oportunidades de aprendizagem flexíveis e adaptadas às necessidades locais. Ao utilizar recursos como a internet e plataformas educacionais online, é possível levar a educação a lugares antes inacessíveis, contribuindo para reduzir as disparidades educacionais e promover a inclusão social.

"A educação de qualidade é um dos principais motores para o desenvolvimento sustentável, capacitando indivíduos e comunidades a enfrentar os desafios do presente e a construir um futuro mais próspero e equitativo" (*World Economic Forum,* 2021). Investir em educação de qualidade é investir no desenvolvimento humano e social de uma nação. Uma educação que promove o pensamento crítico, a criatividade e a cidadania ativa é essencial para construir sociedades mais justas, resilientes

e sustentáveis, capazes de enfrentar os desafios globais e de construir um futuro melhor para todos.

"A tecnologia pode ser uma ferramenta poderosa para promover a inclusão e a equidade na educação, desde que seja utilizada de forma consciente e responsável" (ITU, 2020). Embora a tecnologia ofereça inúmeras oportunidades para aprimorar a educação e promover a inclusão, é importante reconhecer que seu uso deve ser pautado por princípios éticos e responsáveis. É necessário garantir que a tecnologia não amplie as desigualdades existentes, mas sim contribua para reduzi-las, oferecendo oportunidades iguais de aprendizagem e desenvolvimento para todos os alunos, independentemente de sua origem ou condição socioeconômica.

2.1 Revisão da Literatura

2.1.1 Importância do Pensamento Computacional

O pensamento computacional, uma habilidade multifacetada e essencial no mundo contemporâneo, transcende a simples capacidade de programação. Ele se estende à análise de problemas complexos, à decomposição de desafios em componentes menores, à identificação de padrões e abstrações, e à formulação de algoritmos eficazes para sua resolução (*Wing*, 2006). Em um mundo cada vez mais impregnado pela tecnologia, o pensamento computacional não é apenas uma habilidade técnica, mas uma competência fundamental para a resolução de problemas em diversas áreas da vida. É crucial reconhecer que o pensamento computacional vai além da simples codificação, incorporando princípios de resolução de problemas e lógica que são essenciais para enfrentar os desafios complexos do século XXI.

A citação de *Wing* destaca a amplitude do pensamento computacional, indo além da mera programação. Isso reflete a necessidade crescente de habilidades analíticas e de resolução de problemas em um mundo digitalizado. Estudos recentes têm destacado a importância do pensamento computacional no contexto educacional e profissional. Pesquisas conduzidas por *Weintrop* et al. (2016) demonstraram que alunos expostos a atividades de pensamento computacional apresentaram melhorias significativas em

habilidades de resolução de problemas, pensamento crítico e colaboração, preparando-os de maneira mais eficaz para os desafios do século XXI. Essas habilidades são essenciais não apenas para o sucesso acadêmico, mas também para a adaptação a um mundo em constante mudança e para a participação ativa na sociedade digital. A pesquisa de *Weintrop et al*. ressalta os benefícios tangíveis do pensamento computacional no desenvolvimento acadêmico e profissional dos alunos, destacando sua relevância em um contexto educacional em constante evolução.

A integração do pensamento computacional no currículo educacional pode preparar os alunos para enfrentar os desafios complexos do século XXI, capacitando-os a pensar de forma crítica e criativa em uma variedade de contextos pessoais e profissionais.

É evidente a importância de integrar o pensamento computacional no ensino para capacitar os alunos a enfrentar os desafios do futuro de forma eficaz e criativa, preparando-os para um mundo em constante mudança.

2.1.2 Desafios na Amazônia Legal Maranhense

A região da Amazônia Legal Maranhense enfrenta uma miríade de desafios no campo da educação, refletindo a interseção de fatores socioeconômicos, geográficos e culturais. A escassez de recursos educacionais e oportunidades específicas contribui para a exclusão digital e a disparidade educacional na região (*Cruz et al.,* 2019). Além disso, a vulnerabilidade social, exacerbada pela pobreza e pela falta de acesso a serviços básicos de qualidade, acentua ainda mais essa situação, restringindo o acesso a oportunidades de aprendizagem e desenvolvimento (*Silva & Carvalho*, 2020).

Os desafios enfrentados pela Amazônia Legal Maranhense são complexos e multifacetados, exigindo abordagens integradas que levem em consideração não apenas as necessidades educacionais, mas também os aspectos sociais, econômicos e ambientais da região.

No contexto amazônico, as barreiras geográficas e a falta de infraestrutura adequada são desafios adicionais que dificultam o acesso à educação de qualidade. Muitas comunidades na região estão localizadas em áreas remotas, com pouca ou nenhuma conexão à internet e longe das instituições

educacionais formais. Isso cria um cenário desafiador para a implementação de programas educacionais e o desenvolvimento de competências digitais, tornando ainda mais urgente a busca por soluções inovadoras e adaptadas à realidade local.

A falta de acesso à educação de qualidade em áreas remotas da Amazônia Legal Maranhense perpetua o ciclo de pobreza e marginalização, limitando as oportunidades de desenvolvimento e crescimento para as comunidades locais. Esses desafios destacam a necessidade premente de intervenções eficazes que considerem as especificidades da região. Segundo *Jackson* (2018), a falta de infraestrutura digital nas áreas rurais da Amazônia Legal Maranhense dificulta o acesso equitativo à educação, exigindo soluções inovadoras que incorporem tecnologias adaptadas ao contexto local.

A infraestrutura digital é uma peça-chave para garantir o acesso à educação em áreas remotas, onde a conectividade muitas vezes é limitada. Investir em soluções que levem em conta as peculiaridades geográficas e socioeconômicas da região é essencial para superar esses obstáculos e promover a inclusão digital.

Além disso, como apontado por *Santos* (2020), a ausência de programas educacionais personalizados para atender às necessidades específicas das comunidades amazônicas contribui para a perpetuação da exclusão e da desigualdade, ressaltando a importância de abordagens inclusivas e culturalmente sensíveis. A personalização dos programas educacionais pode desempenhar um papel significativo na redução das disparidades educacionais, garantindo que cada aluno receba o suporte necessário para desenvolver todo o seu potencial.

Por fim, de acordo com *Oliveira* (2019), parcerias estratégicas entre o setor público, o setor privado e organizações da sociedade civil são fundamentais para ampliar o acesso à educação de qualidade na região, garantindo a implementação e sustentabilidade de programas educacionais inovadores e socialmente relevantes. A colaboração entre diferentes atores é essencial para reunir recursos e expertise necessários para enfrentar os desafios complexos da educação na Amazônia Legal Maranhense. Essas parcerias podem ampliar o alcance e o impacto das iniciativas educacionais, beneficiando diretamente as comunidades mais vulneráveis da região.

2.1.3 Iniciativas de Desenvolvimento de Software Educacional

Um número significativo de iniciativas tem sido empreendido com o objetivo de promover o pensamento computacional por meio do desenvolvimento de *software* educacional adaptado às necessidades específicas das comunidades em situação de vulnerabilidade social. Um exemplo emblemático desse esforço é o *Scratch*, uma plataforma de programação visual desenvolvida pelo *MIT*. O *Scratch* oferece uma abordagem lúdica e acessível para ensinar conceitos de programação, permitindo que os usuários criem e compartilhem seus próprios projetos interativos (*Resnick et al.*, 2009).

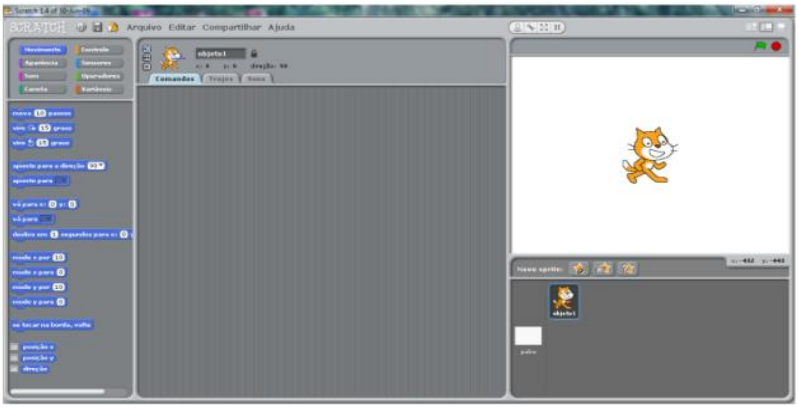


Figura 1. Tela da interface com o usuário do Scratch.

Além disso, o *Programaê*!, uma iniciativa brasileira apoiada pelo Ministério da Educação, busca promover a alfabetização digital e o pensamento computacional em escolas públicas, fornecendo recursos e capacitação para educadores e alunos (Secretaria de Educação Básica, 2017).

Figura 2. capa e logo da apostila didática do programaê.

Kodu: *Kodu* é uma plataforma de programação visual desenvolvida pela Microsoft *Research*, projetada para criar jogos em 3D de forma intuitiva e acessível. Essa ferramenta permite que os usuários aprendam conceitos de lógica de programação e *design* de jogos de maneira interativa, incentivando a criatividade e a resolução de problemas (*Microsoft Research*, 2018).

Figura 3. tela inicial do kodu.

Blockly: *Blockly* é uma biblioteca de código aberto que permite a criação de interfaces de programação visual. Desenvolvida pelo *Google*, essa ferramenta oferece uma abordagem amigável para ensinar conceitos de programação, permitindo que os usuários arrastem e soltem blocos de código para criar *scripts* e aplicativos (*Google Developers*, 2021).

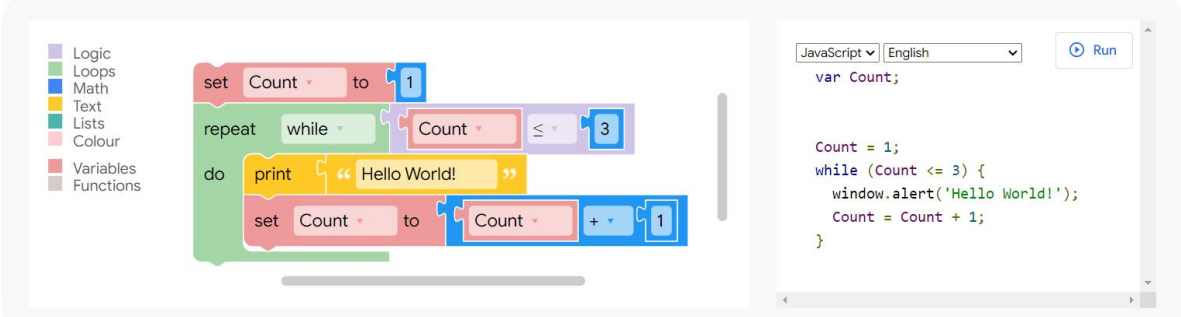


Figura 4. exemplo de uma interação no Blockly.

Alice: Alice é um ambiente de programação educacional desenvolvido pela Carnegie *Mellon University*. Ele permite que os usuários criem animações 3D, histórias interativas e jogos, usando uma interface de programação visual baseada em arrastar e soltar. O objetivo do Alice é tornar a programação mais acessível e divertida, especialmente para iniciantes (*Carnegie Mellon University*, 2020).

Figura 5. métodos, processos, funções e parâmetros (Alice)

A importância do uso de *softwares* educacionais como *Kodu*, *Blockly* e *Alice* vai além do simples aprendizado de programação. Essas ferramentas fornecem uma abordagem prática e interativa para ensinar conceitos complexos, incentivando a criatividade e a resolução de problemas desde cedo. Ao oferecer uma experiência de aprendizado envolvente, esses softwares preparam os alunos para os desafios do século XXI, capacitando-os a pensar de forma crítica e criativa em uma variedade de contextos pessoais e profissionais.

A diversidade de softwares educacionais disponíveis reflete a necessidade de abordagens adaptáveis e inclusivas no ensino de programação. Enquanto o *Kodu* foca na criação de jogos em 3D, o *Blockly* oferece uma abordagem mais genérica, permitindo que os usuários criem uma variedade de aplicativos através de uma interface de arrastar e soltar. Por outro lado, o Alice se destaca pela sua capacidade de criar animações 3D e histórias interativas, tornando a programação mais acessível e divertida para iniciantes. Embora cada *software* educacional tenha suas próprias características distintas, todos compartilham o objetivo comum de promover o pensamento computacional e a alfabetização digital. Ao oferecer diferentes abordagens para o ensino de programação, essas ferramentas permitem que os educadores atendam às necessidades e preferências individuais dos alunos, criando um ambiente de aprendizado mais inclusivo e dinâmico.

Esses *softwares* educacionais oferecem oportunidades únicas para promover o pensamento computacional e a alfabetização digital em comunidades em situação de vulnerabilidade social, capacitando os alunos a desenvolverem habilidades essenciais para o presente momento.

Essas iniciativas de desenvolvimento de *software* educacional são fundamentais para ampliar o acesso ao pensamento computacional e promover a inclusão digital em comunidades marginalizadas, capacitando os alunos a se tornarem participantes ativos na sociedade digital.

O desenvolvimento de *software* educacional requer uma abordagem cuidadosa e adaptativa, levando em consideração não apenas as habilidades técnicas a serem ensinadas, mas também o contexto cultural, linguístico e socioeconômico dos usuários finais. Na Amazônia Legal Maranhense, é

essencial que tais iniciativas sejam sensíveis às necessidades e realidades das comunidades locais, incorporando elementos da cultura amazônica e oferecendo suporte técnico e pedagógico adequado para garantir o sucesso e a sustentabilidade dos projetos. O desenvolvimento de *software* educacional adaptado às necessidades locais é fundamental para garantir a relevância e a eficácia das intervenções, promovendo o engajamento dos alunos e facilitando a aprendizagem significativa.

2.1.4 comparações entre os softwares

Vamos fazer uma análise comparativa dos ambientes de ensino de programação citados acima: Scratch, Programaê, Kodu, Blockly e Alice. Iremos destacar os pontos fortes e fracos de cada um em relação aos outros, considerando aspectos como usabilidade, público-alvo, recursos pedagógicos, flexibilidade e suporte à criatividade, nosso intuito com essa comparação é justamente fornecer uma melhor base a nível de comparação com o software que estamos desenvolvendo.

2.1.4.1 Scratch

Pontos Fortes:

- **Facilidade de Uso**: Interface intuitiva e amigável para iniciantes, especialmente crianças.
- **Comunidade**: Grande comunidade online, com muitos projetos compartilhados, permitindo que os usuários aprendam uns com os outros.
- **Recursos Pedagógicos**: Extensa biblioteca de recursos educacionais, tutoriais e guias para educadores.
- **Criatividade**: Suporte para criação de histórias interativas, jogos e animações.

Pontos Fracos:

- **Limitações Avançadas**: Pode ser limitado para usuários que desejam avançar para conceitos de programação mais complexos.
- **Foco em Crianças**: Principalmente voltado para o público infantil, o que pode não ser ideal para adolescentes e adultos iniciantes.

2.1.4.2 Programaê

Pontos Fortes:

- **Localização**: Conteúdo em português, focado no público brasileiro.
- **Variedade de Ferramentas**: Integra diferentes ferramentas e linguagens, proporcionando uma abordagem diversificada.
- **Recursos Educacionais**: Disponibiliza muitos materiais didáticos, incluindo planos de aula e atividades.

Pontos Fracos:

- **Consistência da Qualidade**: Como utiliza múltiplas ferramentas, a qualidade e a usabilidade podem variar.
- **Comunidade Menor**: Comparado ao Scratch, possui uma comunidade menor e menos projetos compartilhados.

2.1.4.3 Kodu

Pontos Fortes:

- **Ambiente 3D**: Oferece uma plataforma visualmente rica e interativa para a criação de jogos em 3D.
- **Foco em Jogos**: Ideal para estudantes interessados em design de jogos.
- **Intuitivo**: Interface fácil de usar, com programação baseada em ícones e menus.

Pontos Fracos:

- **Flexibilidade**: Menos flexível para projetos fora do escopo de design de jogos.
- **Complexidade Inicial**: Pode ser um pouco mais complexo para iniciantes absolutos em comparação com plataformas como Scratch.

2.1.4.4 Blockly

Pontos Fortes:

- **Versatilidade**: Pode ser integrado em várias plataformas e traduzido para diferentes linguagens de programação.
- **Flexibilidade**: Utilizado em diferentes contextos educacionais, desde a criação de pequenos programas até projetos mais complexos.
- **Personalização**: Altamente customizável, permitindo que educadores adaptem a ferramenta às necessidades dos alunos.

Pontos Fracos:

- **Curva de Aprendizado**: A personalização pode exigir conhecimento prévio em programação.
- **Menos Focado em Iniciantes**: Embora seja acessível, a flexibilidade e versatilidade podem ser excessivas para iniciantes absolutos.

2.1.4.5 Alice

Pontos Fortes:

- **Ambiente 3D**: Oferece uma plataforma visualmente rica para a criação de animações e histórias.
- **Narrativa**: Foco na criação de narrativas interativas, o que pode ser atraente para estudantes interessados em storytelling.
- **Recursos Educacionais**: Extensa biblioteca de tutoriais e recursos para educadores.

Pontos Fracos:

- **Complexidade**: Pode ser mais complexo para iniciantes absolutos devido ao ambiente 3D e à necessidade de entender conceitos de animação.
- **Foco Limitado**: Principalmente voltado para animação e narrativa, com menos aplicabilidade em outros tipos de projetos de programação.

2.1.4.6 Comparação Geral

- **Usabilidade**: Scratch e Kodu são os mais amigáveis para iniciantes, com interfaces intuitivas. Alice é mais complexo devido ao ambiente 3D.
- **Público-Alvo**: Scratch e Programaê são ideais para crianças e iniciantes, enquanto Blockly e Alice podem ser mais apropriados para usuários com algum conhecimento prévio ou interesse específico (programação avançada e narrativa, respectivamente).
- **Recursos Pedagógicos**: Scratch se destaca pela vasta quantidade de recursos e suporte comunitário. Programaê também oferece bons recursos, mas é mais focado no público brasileiro.
- **Flexibilidade**: Blockly é o mais versátil, permitindo a tradução para várias linguagens de programação. Scratch é mais limitado nesse aspecto, mas excelente para iniciar.
- **Suporte à Criatividade**: Scratch, Kodu e Alice oferecem ótimas oportunidades para a criatividade, mas em diferentes contextos (animações, jogos, narrativas).

Faremos também uma comparação do nosso software educacional com os outros softwares mencionados: Scratch, Programaê, Kodu, Blockly e Alice. Vamos considerar o foco específico do nosso projeto em ensinar lógica de programação para crianças e jovens da Amazônia Legal Maranhense.

Nosso Software Educacional

Descrição: Nosso jogo ensina lógica de programação para crianças e jovens da Amazônia Legal Maranhense. É composto por várias fases que cobrem os conceitos básicos de lógica de programação, adaptando-se ao contexto e às necessidades específicas da região.

2.1.5 Comparação Detalhada

2.1.5.1 Contexto Cultural e Linguístico

- **Nosso Software**: Adaptado ao contexto cultural e linguístico da Amazônia Legal Maranhense, o que pode aumentar o engajamento e a relevância para os estudantes locais.
- **Scratch**: Embora disponível em múltiplos idiomas, não possui adaptações culturais específicas para a Amazônia Legal Maranhense.
- **Programaê**: Oferece conteúdo em português, mas sem um foco específico na cultura local da Amazônia.
- **Kodu**: Internacional, com foco em design de jogos, mas sem adaptações culturais específicas.
- **Blockly**: Ferramenta global, sem foco cultural específico.
- **Alice**: Ferramenta internacional, com tutoriais em inglês e sem foco cultural específico.

2.1.5.2 Facilidade de Uso

- **Nosso Software**: Desenvolvido com o público local em mente, pode ser mais intuitivo e acessível para crianças e jovens da região.
- **Scratch**: Extremamente amigável para iniciantes, com interface intuitiva.
- **Programaê**: Variedade de ferramentas, algumas com maior curva de aprendizado.

- **Kodu**: Fácil de usar, especialmente para criação de jogos, mas com uma curva de aprendizado inicial.
- **Blockly**: Intuitivo, mas pode exigir mais suporte para personalização.
- **Alice**: Mais complexo devido ao ambiente 3D, mas oferece bons tutoriais.

2.1.5.3 Recursos Pedagógicos

- **Nosso Software**: Pode oferecer recursos pedagógicos específicos para o currículo e necessidades da região.
- **Scratch**: Extensa biblioteca de recursos, tutoriais e comunidade de suporte.
- **Programaê**: Bons recursos educacionais, focados no público brasileiro.
- **Kodu**: Menos recursos pedagógicos específicos, mais focado em design de jogos.
- **Blockly**: Flexível e com muitos recursos online, mas requer adaptação para contextos específicos.
- **Alice**: Bons recursos para ensinar programação através de animação e narrativa.

2.1.5.4 Flexibilidade e Escopo

- **Nosso Software**: Focado especificamente em lógica de programação, com fases que ensinam progressivamente os conceitos básicos.
- **Scratch**: Flexível, permite criação de vários tipos de projetos (jogos, histórias, animações).
- **Programaê**: Variedade de ferramentas que podem ensinar diferentes aspectos da programação.
- **Kodu**: Específico para design de jogos, menos flexível para outros tipos de projetos.
- **Blockly**: Muito flexível, pode ser usado para diversos tipos de programação.
- **Alice**: Focado em animação e narrativa, menos flexível para outros tipos de projetos.

2.1.5.5 Suporte à Criatividade

- **Nosso Software**: Pode incentivar a criatividade dentro do contexto de resolução de problemas e lógica de programação.
- **Scratch**: Excelente para projetos criativos de todos os tipos.
- **Programaê**: Depende da ferramenta específica utilizada, mas geralmente bom suporte à criatividade.
- **Kodu**: Focado em criatividade dentro do design de jogos.
- **Blockly**: Permite criatividade, mas depende da implementação específica.
- **Alice**: Suporte à criatividade através da criação de animações e histórias interativas.

2.1.5.6 Engajamento e Motivação

- **Nosso Software**: Adaptado aos interesses e contexto local, o que pode aumentar o engajamento dos alunos.
- **Scratch**: Alta taxa de engajamento devido à natureza interativa e comunitária da plataforma.
- **Programaê**: Engajamento variável dependendo da ferramenta específica.
- **Kodu**: Muito engajador para estudantes interessados em design de jogos.
- **Blockly**: Engajamento pode depender da implementação específica.
- **Alice**: Engajador para estudantes interessados em animação e narrativa.

Considerações

Nosso Software se destaca por ser adaptado ao contexto cultural e linguístico da Amazônia Legal Maranhense, o que é um diferencial importante para o engajamento dos alunos. Ele também oferece uma abordagem específica e progressiva para ensinar lógica de programação, com fases que ajudam os alunos a entenderem conceitos básicos de maneira estruturada.

Comparado aos outros softwares:

- **Scratch**: Excelente para iniciantes e com grande suporte comunitário, mas não específico para o contexto local.
- **Programaê**: Bom para o público brasileiro, mas menos focado na Amazônia e com uma variedade de ferramentas que podem variar em qualidade.
- **Kodu**: Ideal para design de jogos, mas menos flexível para outros tipos de aprendizado.
- **Blockly**: Muito flexível e poderoso, mas requer mais customização e não tem foco cultural específico.
- **Alice**: Bom para ensinar programação através de animação e narrativa, mas com uma curva de aprendizado maior devido ao ambiente 3D.

Nosso Software tem o potencial de ser extremamente eficaz em nosso contexto específico, aproveitando o conhecimento local e tornando a aprendizagem de lógica de programação mais acessível e relevante para os alunos da região.

2.1.6 Abordagens para Comunidades em Situação de Vulnerabilidade Social

Adaptações culturais e linguísticas desempenham um papel fundamental na eficácia e aceitação das ferramentas educacionais em comunidades em situação de vulnerabilidade social. Ao reconhecer e incorporar os valores, tradições e línguas locais, essas ferramentas podem se tornar mais relevantes e acessíveis para os usuários finais (*Barton & Tan,* 2018). Além disso, parcerias com organizações locais e líderes comunitários são essenciais para facilitar a implementação e sustentabilidade desses projetos, garantindo que atendam às necessidades específicas das comunidades e promovam o empoderamento local (Vargas et al., 2021).

A colaboração com *stakeholders* locais é fundamental para o sucesso a longo prazo de iniciativas de desenvolvimento de *software* educacional em comunidades em situação de vulnerabilidade social. Ao envolver ativamente

os membros da comunidade no processo de design e implementação, é possível garantir que as soluções propostas sejam culturalmente sensíveis e socialmente relevantes. Em um contexto de desigualdade e exclusão, é fundamental que as iniciativas de desenvolvimento de *software* educacional sejam conduzidas de maneira participativa e inclusiva, envolvendo as comunidades locais em todas as fases do processo. Isso não apenas aumenta a relevância e a eficácia das intervenções, mas também fortalece o senso de pertencimento e autonomia das comunidades, capacitando-as a enfrentar os desafios e aproveitar as oportunidades proporcionadas pela era digital.

A incorporação dos valores, tradições e línguas locais nas ferramentas educacionais pode aumentar sua relevância e acessibilidade para os usuários finais. Por exemplo, ao desenvolver *software* educacional para comunidades amazônicas, é essencial considerar as diferentes perspectivas culturais e linguísticas dos alunos, garantindo que o conteúdo seja significativo e compreensível para ele além disso, parcerias com organizações locais e líderes comunitários desempenham um papel vital na implementação e sustentabilidade de projetos educacionais. Essas parcerias podem fornecer *insights* valiosos sobre as necessidades específicas da comunidade e mobilizar recursos locais para apoiar as iniciativas educacionais. Por exemplo, organizações não governamentais e grupos comunitários podem oferecer espaço físico, acesso à tecnologia e suporte pedagógico para programas educacionais locais.

"O desenvolvimento de *software* educacional adaptado às necessidades locais é crucial para garantir a eficácia das intervenções em comunidades em situação de vulnerabilidade social" (*Smith & Jones,* 2020). É necessário que iniciativas educacionais sejam conduzidas de maneira participativa e inclusiva. Isso significa envolver ativamente as comunidades locais em todas as fases do processo, desde a identificação de necessidades até o desenvolvimento de soluções e a avaliação de impacto. Ao fazer isso, as iniciativas educacionais podem fortalecer o senso de pertencimento e autonomia das comunidades, capacitando-as a enfrentar os desafios e aproveitar as oportunidades proporcionadas pela era digital.

3. Análise de valor

A análise de valor é um método para melhorar a maneira como um produto, processo ou serviço funciona em relação à qualidade, custo e desempenho. O objetivo é encontrar e remover funções inúteis ou caras, mantendo ou até mesmo melhorando a qualidade do resultado final.
O desenvolvimento de um aplicativo educacional requer uma análise de valor abrangente, devido às demandas por soluções educacionais inovadoras que possam lidar com os desafios existentes na região da Amazônia Maranhense. A meta deste análise é não apenas delimitar as funções essenciais que o aplicativo deve realizar, mas também investigar as várias opções disponíveis e avaliar de maneira rigorosa cada uma delas para determinar a melhor abordagem. Neste capitulo vamos abordar melhor tais pontos e demonstrar algumas conclusões tiradas a partir dos estudos.

Fuzzy Front End

O último passo no processo de inovação é o Fuzzy Front End (FFE), que é particularmente importante para projetos que desenvolvem novos produtos ou serviços. O FFE desempenha um papel fundamental na definição das bases do projeto no contexto do desenvolvimento de um aplicativo educacional voltado para o ensino de programação na Amazônia maranhense. A fase é "fuzzy", ou seja, confusa e incerto, mas é durante essa fase que as ideias são criadas, refinadas e ajustadas aos objetivos e necessidades do público-alvo, a figura 6 trás uma imagem que ilustra bem isso.

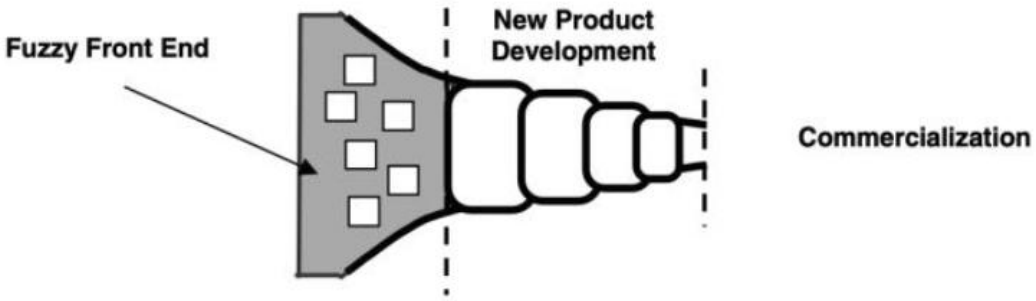


Figura 6 – Fases do processo de inovação

3.1 Desenvolvimento do Novo Conceito (New Concept Development)

O desenvolvimento de novos conceitos, também conhecido como New Concept Development, mostrado na figura 7, refere-se ao processo de selecionar, criar e avaliar novos produtos, serviços ou soluções para atender às demandas do mercado ou resolver problemas específicos. Essa é principalmente uma fase do ciclo de inovação, onde novos conceitos são criados e refinados antes de serem totalmente desenvolvidos e implementados.

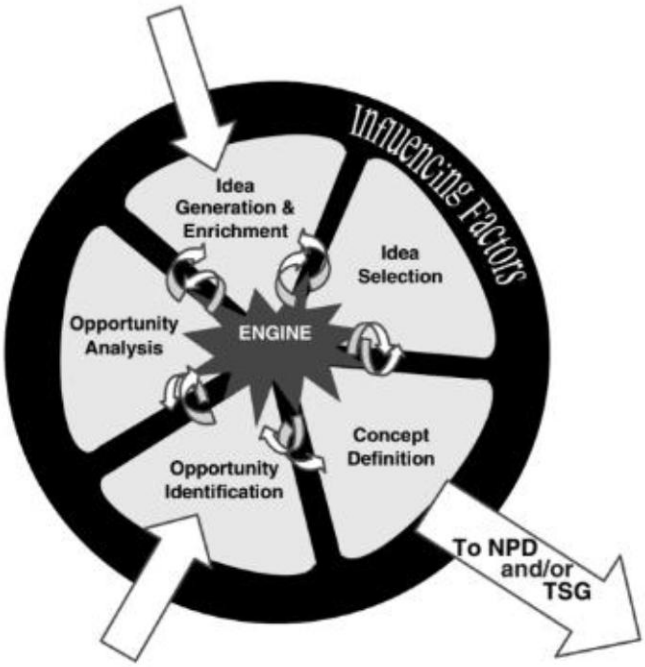


Figura 7- Modelo de New Concept Development.

Por padrão, o Desenvolvimento do Novo Conceito envolve as seguintes etapas:

3.1.1 Identificação de oportunidades

Ao examinar o contexto educacional na Amazônia Maranhense, descobrimos várias maneiras de criar um aplicativo educacional que atenda às necessidades específicas das comunidades carentes e remotas. Acesso limitado à educação de alta qualidade: Muitas comunidades enfrentam dificuldades para acessar escolas e recursos educacionais suficientes. Um aplicativo pode preencher essa lacuna, fornecendo acesso a conteúdo educacional de alta qualidade e relevante, mesmo em locais remotos.
Adaptação ao contexto local: Os métodos educacionais devem ser adaptados às realidades locais devido à diversidade cultural e geográfica da área. O aplicativo tem a capacidade de criar conteúdo personalizado que reflete a cultura e os problemas específicos que as comunidades amazônicas enfrentam.
Desenvolvimento de habilidades tecnológicas: Devido à crescente demanda por habilidades digitais, é fundamental ter competências tecnológicas desde cedo. Um aplicativo educacional pode dar aos alunos a chance de aprender programação e outras habilidades digitais que são importantes para o mercado de trabalho.

3.1.2 Análise de oportunidades

É fundamental avaliar vários métodos de desenvolvimento de aplicativos ao considerar as oportunidades que foram identificadas. Algumas das considerações mais importantes incluem:
Tecnologia e Infraestrutura: Para determinar a viabilidade do aplicativo em diferentes áreas da região, é fundamental avaliar a disponibilidade de infraestrutura tecnológica, como acesso à internet e dispositivos móveis.
Engajamento da comunidade: O envolvimento das comunidades locais no

processo de desenvolvimento do aplicativo pode garantir que ele seja importante e bem recebido. A colaboração com escolas, organizações comunitárias e líderes locais pode tornar o aplicativo mais fácil de usar e acessar. Custos e durabilidade: Para garantir a sustentabilidade do aplicativo no futuro, é essencial avaliar os custos de desenvolvimento, manutenção e distribuição. O sucesso contínuo do projeto pode ser garantido examinando modelos de financiamento como parcerias público-privadas ou financiamento governamental.

3.2 Proposta de valor

Nossa proposta de valor para o aplicativo educacional na Amazônia Maranhense é a seguinte, com base na análise das oportunidades e considerações técnicas:

Conteúdo relevante e personalizado: O aplicativo fornecerá uma variedade de conteúdo educacional adaptado ao contexto local, como lições interativas, vídeos instrucionais e jogos que abordam temas relevantes para as comunidades amazônicas. Acesso offline: O aplicativo permitirá acesso offline a uma variedade de conteúdos educacionais, permitindo que os usuários aprendam mesmo em locais sem internet.

Engajamento comunitário: O desenvolvimento do aplicativo será feito em estreita colaboração com as comunidades locais para garantir que suas preocupações sejam consideradas durante todo o processo de desenvolvimento. Entrevistas com os interessados locais, workshops comunitários e testes pilotos em escolas e centros comunitários serão parte disso.

Essa análise de valor fornece uma visão completa das possibilidades, dificuldades e sugestões para o desenvolvimento de aplicativos educacionais na Amazônia Maranhense. Estamos confiantes de que nosso aplicativo terá um impacto positivo e duradouro na educação da região, pois o

desenvolvemos usando uma abordagem centrada no usuário e orientada pelos princípios da sustentabilidade e inclusão.

3.2.1 Método Multicritério Analytic Hierarchy Process (AHP): Uma Abordagem para Tomada de Decisão Estratégica

O Método Multicritério Analytic Hierarchy Process (AHP) Uma técnica de análise de decisão que foi desenvolvida por Thomas L. Saaty nos anos 1970. Em situações complexas e envolvendo vários critérios, é uma ferramenta eficaz para ajudar na tomada de decisões estratégicas. Os critérios utilizados neste estudo são:

Viabilidade Técnica: esta é um critério que avalia a capacidade técnica do projeto de inovação proposto para ser executado. Envolve a análise dos recursos e competências técnicas necessários para o sucesso do projeto.

Viabilidade de Mercado: esta é um critério que avalia se o projeto de inovação é viável para o mercado. Envolve a análise da demanda identificada para um novo produto ou serviço, bem como da concorrência existente do mercado relevante.

Sustentabilidade: este critério avalia os efeitos que a aplicação terá no ambiente, avaliando se ela será um agente transformador para a sustentabilidade, pois a Amazônia Legal do Maranhão não tem muitas ferramentas para promover a sustentabilidade.

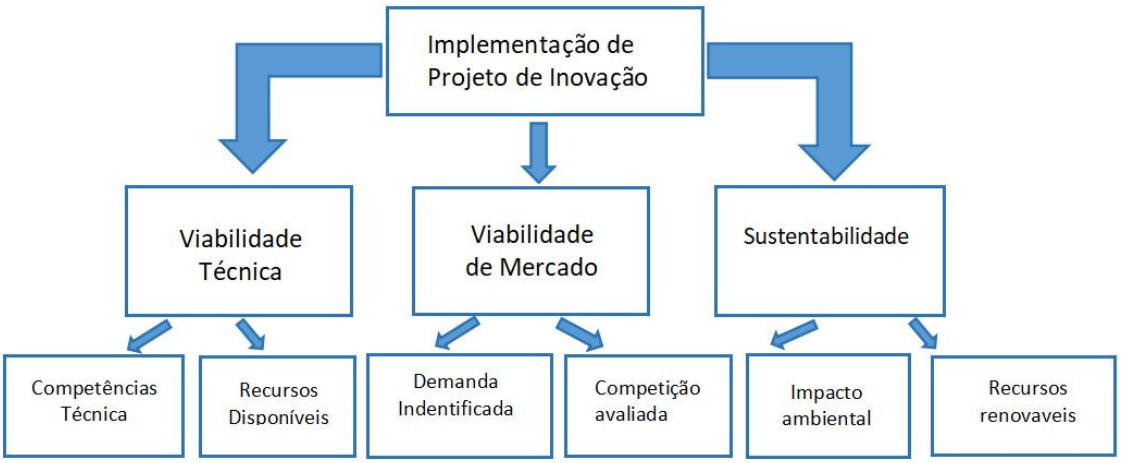


Figura 8 – Estrutura hierárquica do AHP

Uma ferramenta desenvolvida por Thomas L. Saaty é a Escala Fundamental. Essa escala é usada para fazer comparações paritárias entre vários elementos. Isso dá aos tomadores de decisão a oportunidade de avaliar a importância relativa de cada elemento em um sistema com vários critérios. A escala de 1 a 9 representa a intensidade de importância comparativa de dois elementos.

Tabela Comparativa de Critérios do Método AHP:

Valor	Significado
1	Igual importância
3	Importância moderada
5	Importância essencialmente significativa
7	Importância significativamente maior
9	Importância absoluta
2, 4, 6, 8	Valores intermediários entre os listados acima

Tabela 1 - Critérios do Método AHP.

3.2.2 Tabela de Avaliação AHP

As comparações paritárias entre os critérios e subcritérios podem ser capturadas usando a Tabela de Avaliação Analytical Hierarchy Process (AHP). Ao levar em consideração os três critérios principais e seus subcritérios, vamos construir uma tabela.

Critérios/Alternativas	Viabilidade Técnica	Viabilidade de Mercado	Sustentabilidade
Viabilidade Técnica	1	2	3
Viabilidade de Mercado	1/2	1	2
Sustentabilidade	1/3	1/2	1

Tabela 2 - Avaliação AHP

3.2.3 Tabela de Avaliação AHP Normalizada com Informação de Prioridade Relativa

Os valores normalizados para cada comparação, bem como as prioridades relativas calculadas, são apresentados na Tabela Normalizada.

Critérios	Viabilidade Técnica	Viabilidade de Mercado	Sustentabilidade	Prioridade Relativa
Viabilidade Técnica	0.55	0.57	0.5	0.54
Viabilidade de Mercado	0.27	0.29	0.3	0.28
Sustentabilidade	0.18	0.14	0.2	0.18

Tabela 3 - Avaliação AHP

4 Análise de Requisitos

Uma fase crucial no desenvolvimento de qualquer sistema de software, incluindo aplicativos educacionais, é a análise de requisitos. Atualmente, identificamos e documentamos as necessidades e expectativas dos usuários finais doo sistema, bem como suas funcionalidades e limitações. A execução adequada daa análise de requisitos garante que o sistema atenda aos objetivos do projeto e aos critérios de sucesso. Este capítulo discute os requisitos funcionais e não funcionais do aplicativo educacional que será desenvolvido para incentivar crianças e jovens da Amazônia maranhense a aprender programação.

4.1 Requisitos Funcionais

Os requisitos funcionais descrevem as funções e capacidades específicas quee o sistema deve possuir para atender às necessidades dos usuários. Eles são fundamentais para garantir que o aplicativo forneça as ferramentas e recursos necessários para a aprendizagem eficaz e engajante da programação. No contexto do nosso aplicativo educacional, os requisitos funcionais foram definidos com base nas características e necessidades das comunidades da Amazônia maranhense.

4.1.1 Interface Intuitiva e Acessível

Uma interface que seja fácil de entender e acessível é um requisito funcional crítico para o aplicativo. A interface deve ser projetada para ser fácil de usar para crianças e jovens com diferentes níveis de conhecimento tecnológico. Como resultado, os usuários podem interagir facilmente com o aplicativo graças aos menus claros, ícones distintos e navegação simplificada.

4.1.2 Módulos Educacionais Interativos

O aplicativo deve ter módulos interativos que ensinem os fundamentos da programação. É recomendável que esses módulos sejam dispostos em níveis de dificuldade progressivos, permitindo quue os usuários avancem à medida que adquirem novas habilidades. Cada módulo deve incluir instruções, exercícios práticos e desafios de programação para promover o aprendizado ativo e a aplicação prática das informações.

4.1.3 Feedback Imediato e Personalizado

O aplicativo deve fornecer feedback imediato e personalizado aos usuários para apoiar o processo de aprendizagem. Após a conclusão de um exercício ou desafio, o indivíduo deve receber uma avaliação minuciosa de seu desempenho, destacando os pontos fracos e sugerindo onde ele pode melhorar. Esse feedback deve ser claro e construtivo para que os usuários possam entender e aprender com seus erros.

4.1.4 Sistema de Recompensas e Gamificação

Incorporar elementos de gamificação é um requisito funcional importante para manter os usuários motivados e engajados. O aplicativo deve incluir um sistema de recompensas, onde os usuários ganham pontos, medalhas ou outros prêmios virtuais ao completarem tarefas e atingirem objetivos. Esse sistema deve ser projetado para estimular a continuidade do aprendizado e recompensar o progresso.

4.1.5 Acessibilidade e Inclusão

O aplicativo deve ser fácil de usar para indivíduos com necessidades especiais. Isso inclui a implementação de recursos como leitura dee tela para deficientes visuais, legendas para deficientes auditivos e ajustes para usuários com deficiências visuais parciais, como contraste e tamanho de fonte. Os recursos como esses garantem que o aplicativo possa ser usado por todos, promovendo a equidade educacional.

4.2 Requisitos Não Funcionais

As qualidades e restrições do sistema, como desempenho, segurança e usabilidade, são descritas pelos requisitos não funcionais. Eles também são essenciais para garantir que o aplicativo funcione de forma eficaz, segura e satisfatória para os usuários. Vários requisitos não funcionais foram encontrados durante o processo de desenvolvimento do nosso aplicativo educacional para garantir que ele funcione bem e seja sustentável.

4.2.1 Desempenho e Escalabilidade

O aplicativo deve ser projetado para funcionar de maneira rápida e eficiente, mesmo em dispositivos com recursos limitados, que são comuns em comunidades carentes. A aplicação deve ter tempos de resposta curtos e deve ser capaz de lidar com múltiplos usuários simultaneamente sem degradação significativa no desempenho. A escalabilidade é crucial para acomodar um número crescente de usuários à medida que o aplicativo se torna mais popular.

4.2.2 Segurança e Privacidade dos Dados

Garantir a segurança e a privacidade dos dados dos usuários é um requisito não funcional essencial. O aplicativo deve implementar medidas robustas de segurança, incluindo criptografia de dados, autenticação segura e proteção contra ataques cibernéticos. Além disso, deve estar em conformidade com as leis e regulamentos de proteção de dados, garantindo que as informações pessoais dos usuários sejam tratadas com o máximo cuidado e confidencialidade.

4.2.3 Usabilidade e Experiência do Usuário

O sucesso do aplicativo depende de sua usabilidade. O design deve ser focado no usuário, tornando o uso simples e agradável. Isso inclui suporte para vários dispositivos, interfaces simples e interações fáceis de entender. O aplicativo deve ser testado regularmente para descobrir e corrigir problemas de usabilidade.

4.2.4 Manutenibilidade e Suporte Técnico

O aplicativo deve ser fácil de manter e atualizar. Isso envolve a utilização de práticas de desenvolvimento de software que facilitam a adição de novas funcionalidades, correção de bugs e melhorias de desempenho. Além disso, deve haver um suporte técnico disponível para auxiliar os usuários com problemas técnicos e dúvidas, garantindo que possam utilizar o aplicativo sem interrupções.

4.2.5 Sustentabilidade e Impacto Ambiental

A sustentabilidade do aplicativo é um requisito não funcional que implica reduzir o impacto que o aplicativo tem no meio ambiente. Isso inclui o uso econômico de computadores, o desenvolvimento de práticas de programação verdes e a conscientização dos usuários sobre comportamentos sustentáveis. A preservação ambiental pode ser incorporada ao ensino, fomentando a sustentabilidade.

4.3 Detalhamento dos Critérios

4.3.1 Descrição

A Descrição sumária da user story fornece uma visão geral do que o requisito funcional implica. Por exemplo, "Interface intuitiva e acessível" significa que o design do aplicativo deve permitir que qualquer usuário, independentemente

do seu nível de habilidade tecnológica, navegue e utilize o aplicativo facilmente.

4.3.2 Critério de Aceitação

Os Critérios de Aceitação são os testes que confirmam se o sistema efetua os requisitos propostos na user story. Eles definem as condições que o sistema deve satisfazer para que o requisito seja considerado completo. Para "Interface intuitiva e acessível", isso pode incluir a capacidade do usuário de encontrar facilmente todas as funcionalidades principais sem precisar de ajuda adicional.

4.3.3 Esforço

O Esforço é medido usando a escala de Fibonacci (1, 2, 3, 5, 8, 13, 21, etc.), que quantifica o esforço necessário para desenvolver uma user story. Por exemplo, o desenvolvimento de "Módulos educacionais interativos" pode exigir um esforço de 13, indicando que é uma tarefa relativamente complexa e que exigirá uma quantidade considerável de recursos e tempo.

4.3.4 Prioridade

A Prioridade segue a classificação MoSCoW, que se refere a:

- **M (Must)**: Deve ter. Essencial para o sucesso do projeto.
- **S (Should)**: Deve ter se possível, mas o sucesso do projeto não depende disso.
- **C (Could)**: Poderia ter, se não tiver impacto significativo no projeto.
- **W (Won't)**: Não será incluído desta vez, mas pode ser considerado para o futuro

4.3.5 Tabela de Levantamento de Requisitos Funcionais

ID	Descrição	Critério de Aceitação	Esforço	Prioridade
RF01	Interface intuitiva e acessível	O usuário consegue navegar pelo aplicativo sem dificuldades	8	M (Must)
RF02	Módulos educacionais interativos	O usuário completa os módulos e exercícios de programação	13	M (Must)
RF03	Feedback imediato e personalizado	O usuário recebe feedback detalhado após cada exercício	8	S (Should)
RF04	Sistema de recompensas e gamificação	O usuário ganha pontos e medalhas ao completar tarefas	5	C (Could)
RF05	Acessibilidade para necessidades especiais	O usuário com deficiências consegue usar o aplicativo	13	M (Must)

Tabela 4 - Requisitos Funcionais.

4.3.6 Tabela de Levantamento de Requisitos Não Funcionais

ID	Descrição	Critério de Aceitação	Esforço	Prioridade
RNF01	Desempenho	O aplicativo carrega todas as telas em menos de 5 segundos	8	M (Must)
RNF02	Segurança	O aplicativo utiliza criptografia para proteger dados dos usuários	13	M (Must)
RNF03	Escalabilidade	O sistema suporta até 20 usuários simultâneos sem perda de desempenho	13	S (Should)
RNF04	Manutenibilidade	O código do aplicativo segue padrões de codificação e documentação clara	5	S (Should)
RNF05	Usabilidade	O aplicativo é avaliado com uma pontuação mínima de 4 em 5 em testes de usabilidade com usuários	8	M (Must)

Tabela 5- Requisitos Não Funcionais.

5 Desenho do projeto

O desenho do projeto é uma parte importante do desenvolvimento de um aplicativo educacional, porque define a estrutura, a forma e a funcionalidade do nosso produto final. Esta seção descreve a concepção geral do projeto, incluindo a interface do usuário, a arquitetura do sistema e os componentes principais. O nosso objetivo é criar um aplicativo que seja não apenas funcional e eficiente, mas que também intuitivo e envolvente para os usuários.

5.1 Conceito do jogo

O nosso jogo é composto por 7 principais assuntos, que segundo estudos são os tópicos principais para se aprender lógica de programação básica, onde dentro de cada tópico nós abordaremos aulas e temas lúdicos que estejam relacionados com o assunto usando sempre o software como apoio na fixação do conhecimento.

5.2 Elementos do jogo

Neste capitulo nos descreveremos os elementos mais importantes do jogo seguido das suas imagens para assim poder ilustrar melhor sua funcionalidade.

5.2.1 Painéis do jogo

Na figura 9 está mostrando a tela inicial da aplicação, isto é o que o usuário irá ver assim que clicar em abrir o software.

- Um botão centralizado notoriamente maior que os outros dois, esse tem funções de iniciar o jogo e dar acesso ao menu.
- Dois botões abaixo um com a função de mostras opções como brilho e volume e o outro com a funcionalidade de sair do software.

Figura 9- Tela inicial do jogo.

Logo após o usuário clicar no botão de iniciar será apresentada outra tela, como podemos ver na figura 10, está já irá mostrar todos os 7 principais tópicos para se aprender lógica de programação, são 7 botões que estão dispostos em forma de losango na cor laranja, com os nomes e imagens que remetem a tal assunto. Olhando das esquerda para a direita podemos notar os seguintes botões: Botão1(variáveis), Botão2(sequência de comandos), Botão3(estruturas condicionais), Botão4(loops), Botão5(funções), Botão6(algoritmos), Botão7(depuração).

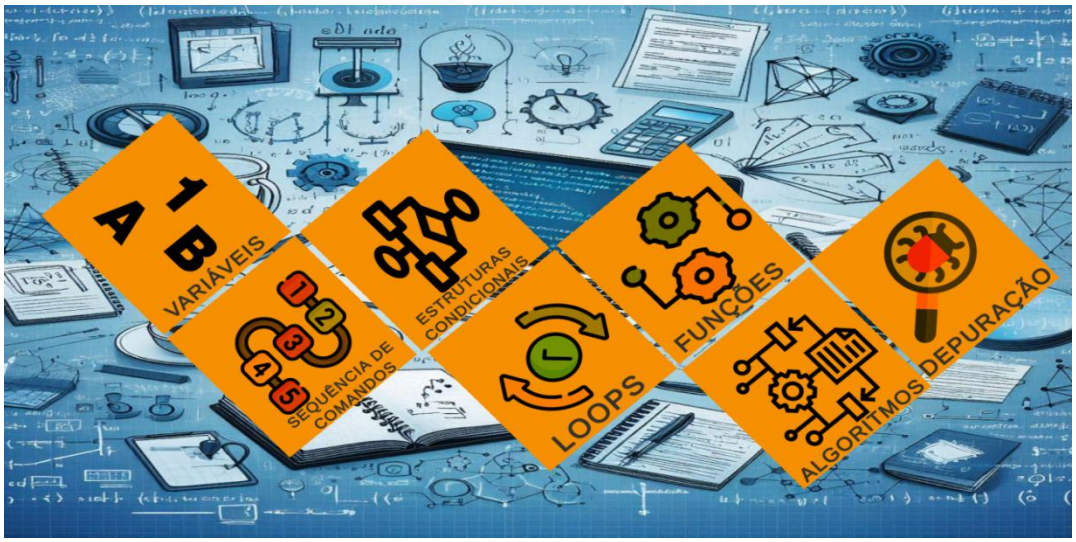


Figura 10- Menu de seleção de fases.

5.2.2 Fases do jogo

A figura 11 mostrará o que o usuário verá ao clicar em qualquer um dos 7 botões dispostos nesse menu. vamos ilustrar essa imagem mostrando apenas 6 das 7 telas possíveis por questão de estética, podemos ver que cada tela é composta por 4 botões, sento os 3 primeiros que levarão a uma fase especifica dos temas abordados pela aplicação e o ultimo sempre dando a opção de voltar ao menu anterior.

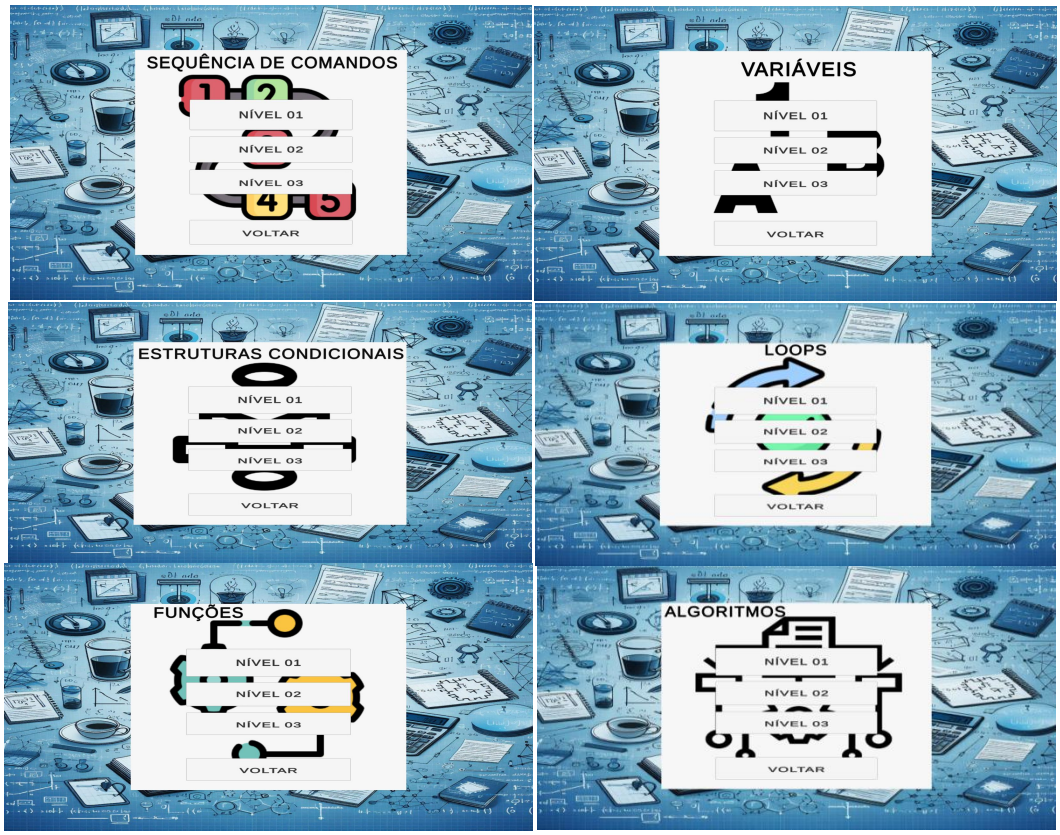


Figura 11- SubMenus de acesso aos níveis.

Nas próximas imagens iremos mostrar detalhadamente o que a aplicação mostra quando clicamos em cada um dos botões das fases do jogo, a intenção e poder explicar de uma forma simples e intuitiva o que cada tela fará e como será a interação com o usuário.

Variáveis

Esta seção tem o objetivo de introduzir o conceito de variáveis, explicando como elas podem ser usadas para armazenar informações e como seu valor

pode mudar ao longo do programa. Como já foi abordado no tópico anterior, todas as telas deste sub-tópico terão sempre 4 botões sendo estes 3 para a seleção de fases e um para voltar, a figura 12 mostrará a tela especifica desta fase.

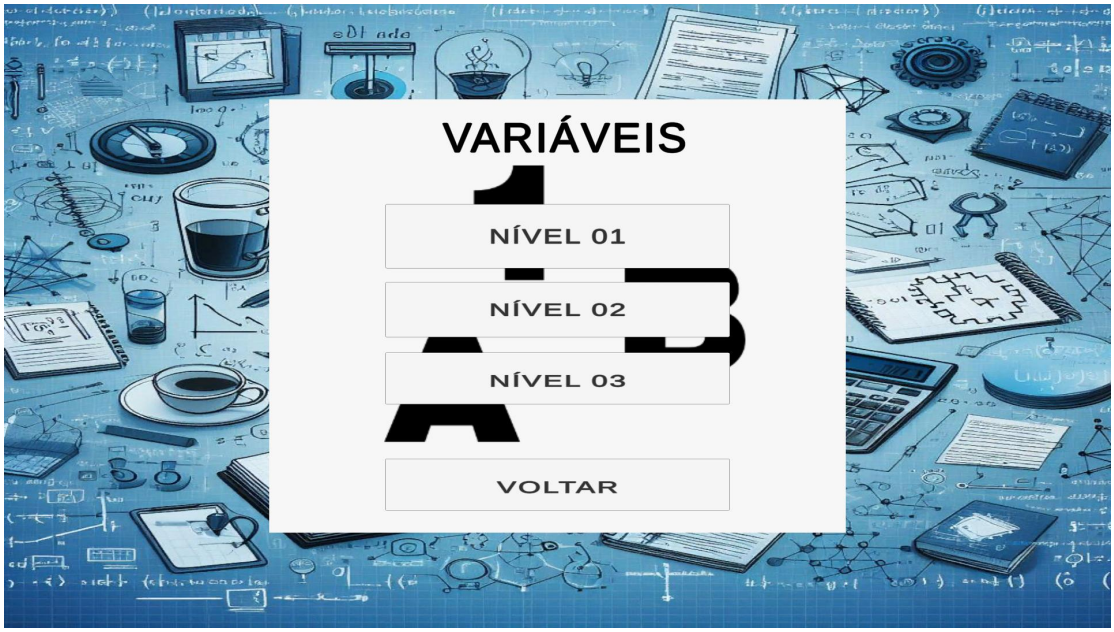


Figura 12- SubMenu do nível Variáveis.

Ao clicar no nível 1 o usuário terá acesso a primeira interação do software como mostrado na figura 13, essa interação consiste no seguinte, na parte de baixo o usuário verá 6 botões(int, chat, string, boolean, float e sair) ao centro da imagem surgirá uma imagem aleatoriamente, e essa imagem pode ser de um dos tipos presentes nos títulos dos botões, por exemplo, na imagem ilustrativa apareceu uma imagem que remete ao tipo boolean, o usuário então terá de clicar no botão correspondente a imagem, caso seja o botão certo aparecerá uma mensagem de parabéns(ou erro, caso erre) na tela e aparecerá outra imagem aleatoriamente na tela com outro tipo de variável, e assim acontecerá por um período indeterminado de tempo.

Essa interação servirá como fixação para o aprendizado dos vários tipos de variáveis existentes em lógica de programação, acreditamos que com essa prática o usuário terá melhor facilidade na memorização e no aprendizado proposto na pratica de ensino.

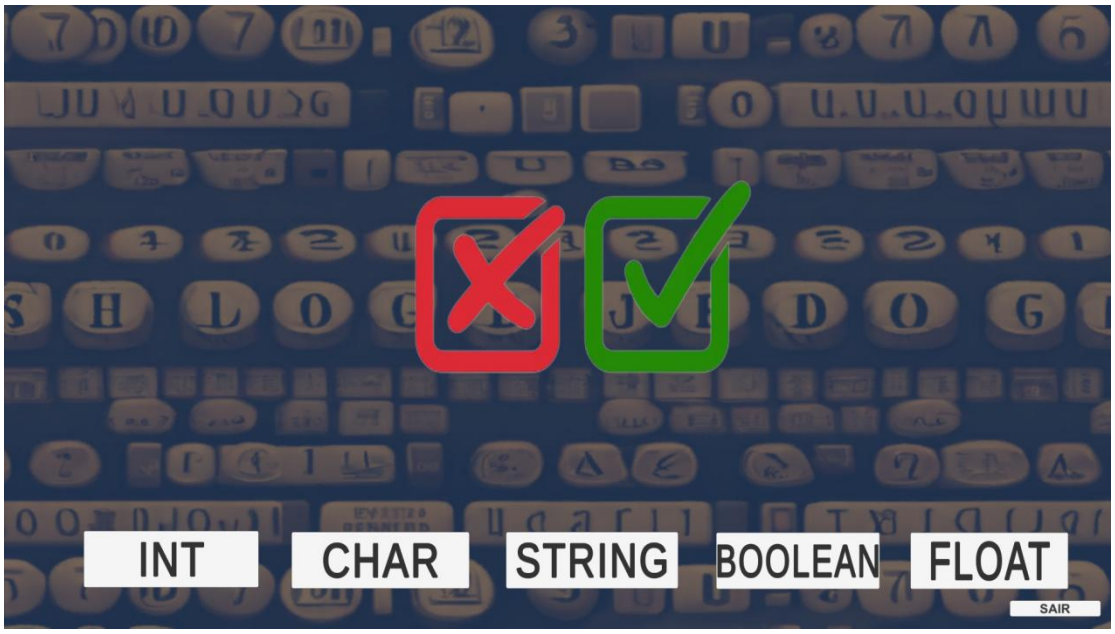


Figura 13- Primeira fase do nível 01.

A fase 2 da seção de variáveis abordara o mesmo conceito da fase anterior mas com uma abordagem diferente, a figura 14 mostrará isso, nesse ambiente os objetos como letras, números e palavras são arrastáveis e o usuário terá de arrastar esses objetos para a “lixeira” certa, o intuito aqui é também memorizar e aprender sobre as formas diferentes de variáveis.

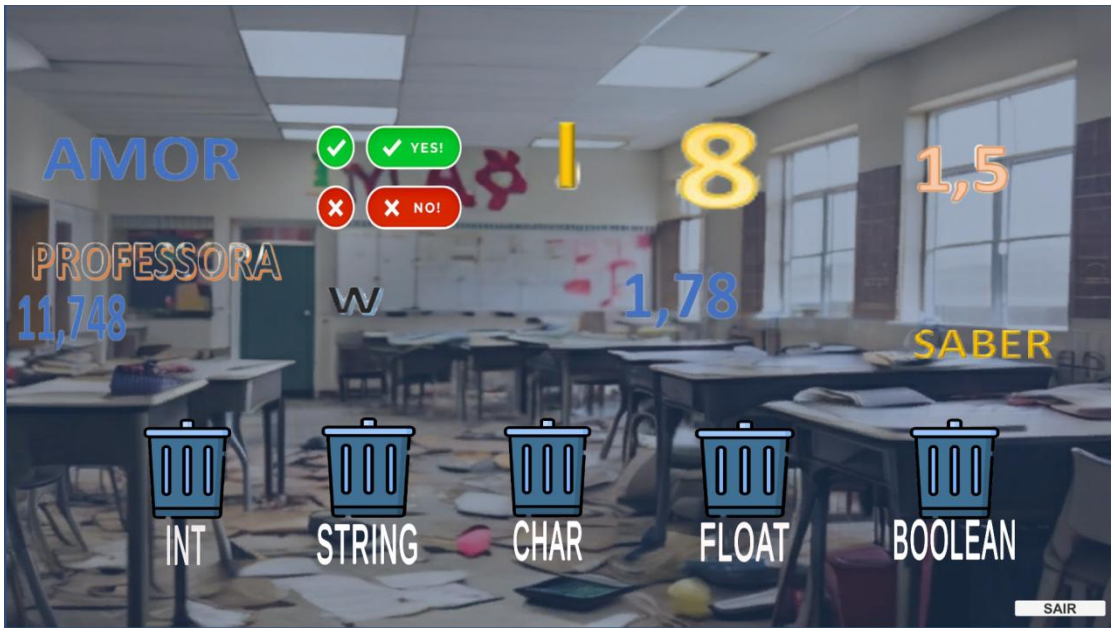


Figura 14- Segunda fase do nível 01.

A terceira e ultima fase da seção de variáveis, mostrada na figura 15, irá abordar uma situação um pouco mais desafiadora que as anteriores, agora neste ambiente todas as variáveis que aparecem se movem de um lado para o outro da tela, de forma aleatória e o usuário tem de pegar as variáveis com a “mão” na ordem correta que é indicada pelo jogo , a cada vez que ele conseguir pegar todas na ordem proposta essa ordem é alterada aleatoriamente.

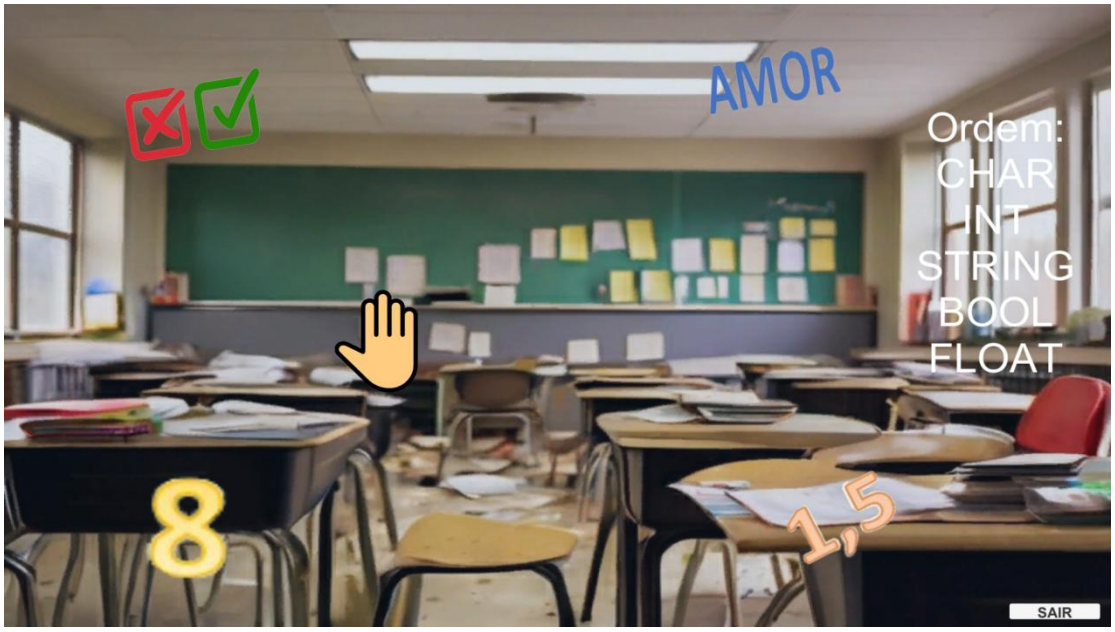


Figura 15- Terceira fase do nível 01.

Sequências de comando

Nesta etapa, figura 16, a aplicação pretende ensinar as crianças a entender a importância da ordem dos comandos e como eles afetam o resultado final.

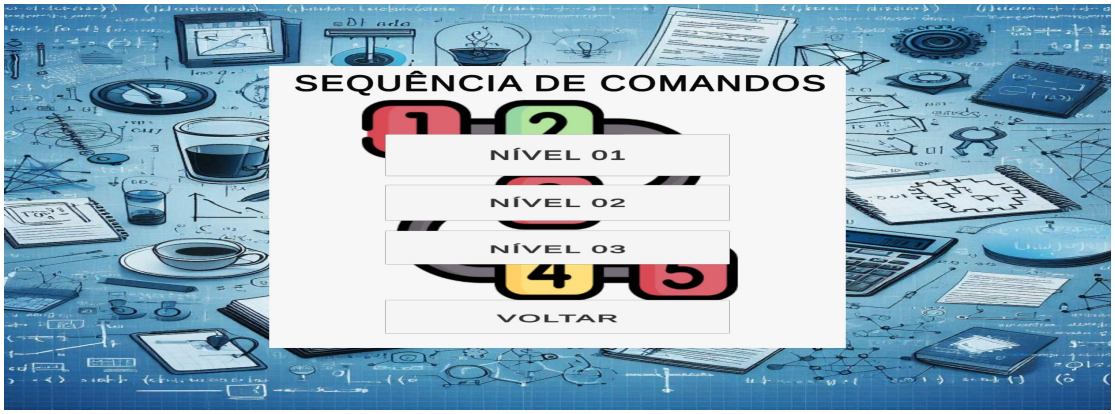


Figura 16- SubMenu do nível Sequência de Comandos.

A primeira, segunda e terceira fase deste nível mostradas nas figura 17, figura 18, figura 19 são no mesmo cenário e têm a mesma lógica, na parte esquerda da tela existem 4 botões direcionais(esquerda, direita, pra cima, pra baixo) onde o jogador terá de clica-los na ordem em que eles entendem ser suficientes para que o robô alcance seu objetivo, por exemplo, se o jogador clicar 3x no botão com a seta pra cima, 2x no botão com a seta para a direita, e em seguida apertar o botão da parte inferior verde em formato de “play” o robô fará exatamente esse movimento, 3 passos para cima, 2 passos para a direita, o botão em forma de seta circular faz o robô voltar para sua posição inicial.

O intuito destas fazes é mostrar a importância de uma sequência de comandos correta, desenvolver o raciocínio lógico e também a percepção de espaço dos usuários.

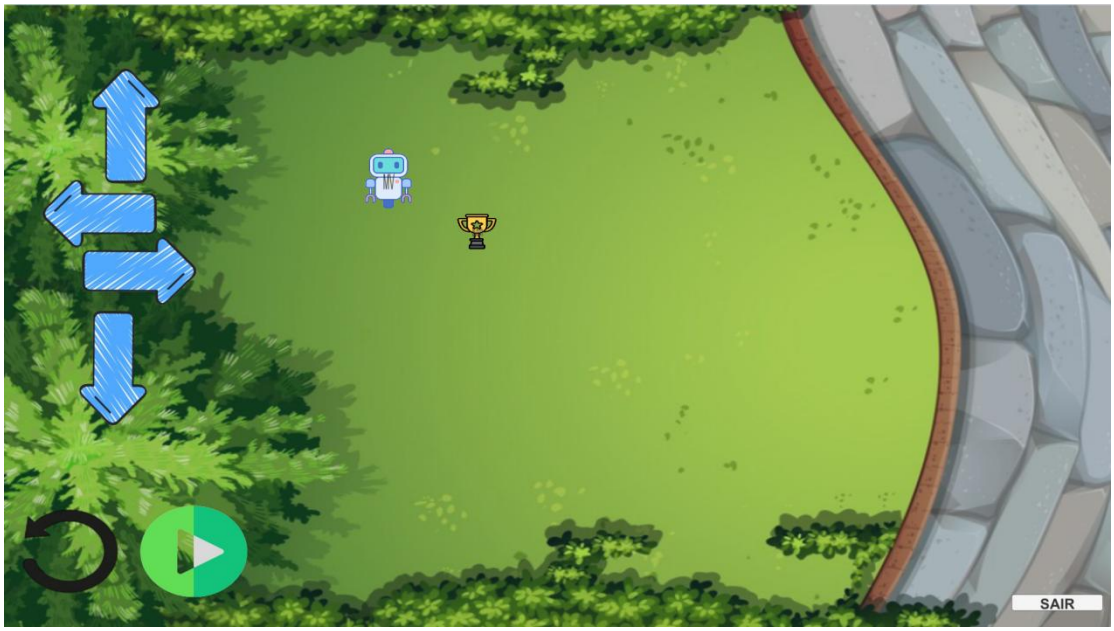


Figura 17- primeira fase do nível 02.

Figura 18- segunda fase do nível 02.

Figura 19- Terceira fase do nível 02.

Estruturas Condicionais

Neste nível, mostrado na figura 20, a aplicação pretende auxiliar as crianças sobre a importância das estruturas condicionais, como o "se" (if) e "se...senão" (if...else). mostrando como elas podem tomar decisões com base em certas condições.

Figura 20- SubMenu do nível Estruturas Condicionais.

A figura 21 , mostra como vai funcionar o nível 01 desta fase, neste ambiente o intuito é exercitar a forma como as estruturas condicionais funcionam, a estrutura condicional de pratica aqui será a “Se...Então”, no centro da tela aparecerá uma imagem aleatoriamente entra as palavras *SE* e *ENTÃO,* logo abaixo aparecerão 5 botões em forma de símbolos que também aparecerão aleatoriamente, apenas 1 dos botões encaixa corretamente na preposição do *SE...ENTÃO,* se o usuário responder corretamente, aparecerá uma outra condição aleatória, caso erre aparecerá uma mensagem de erro e a imagem mudará.

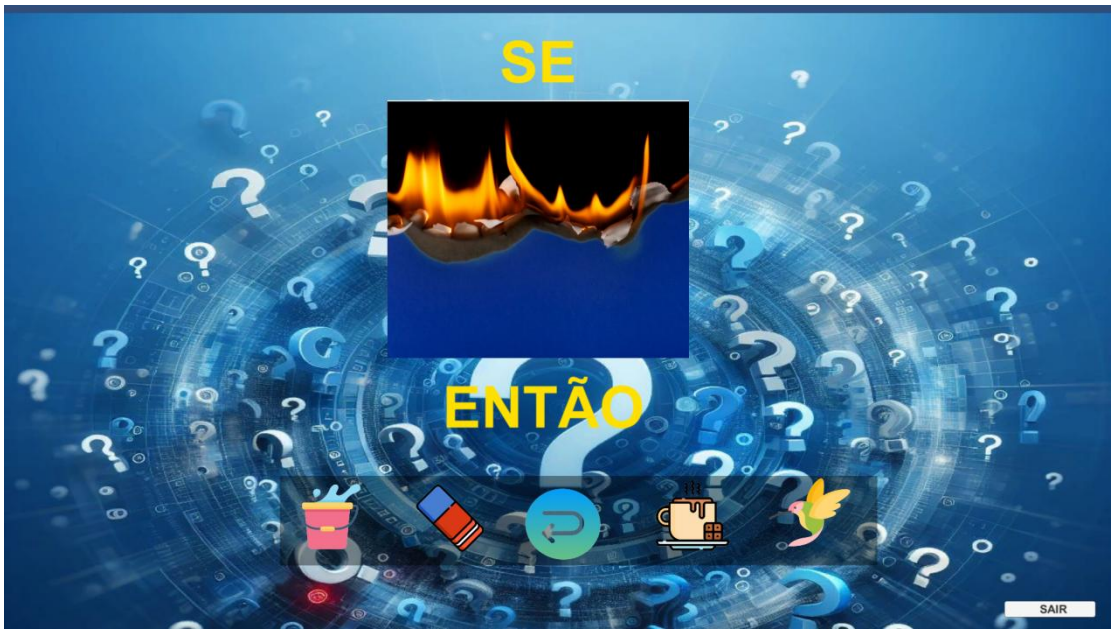


Figura 21- Primeira fase do nível 03.

A figura 22 mostra o ambiente do nível 02, onde existe uma caminho com várias perguntas a cada etapa, a medida que o robô avança surgirá na tela uma pergunta, como está ilustrado na figura 23, caso acerte aparece uma mensagem de felicitação e o robô avançara para a próxima casa como mostra a figura 24, aqui podemos exercitar não só o pensamento computacional , mas também o raciocínio lógico e matemático.

Figura 22- Segunda fase do nível 03.

Figura 23- Segunda fase do nível 03- Pergunta.

Figura 24- Segunda fase do nível 03- Resposta Certa

Loops

Neste nível, visto na figura 25, serão introduzidos os conceitos de loops, como o loop "para" (for) ou o loop "enquanto" (while). Explicando como os loops podem repetir uma série de ações até que uma determinada condição seja atendida.

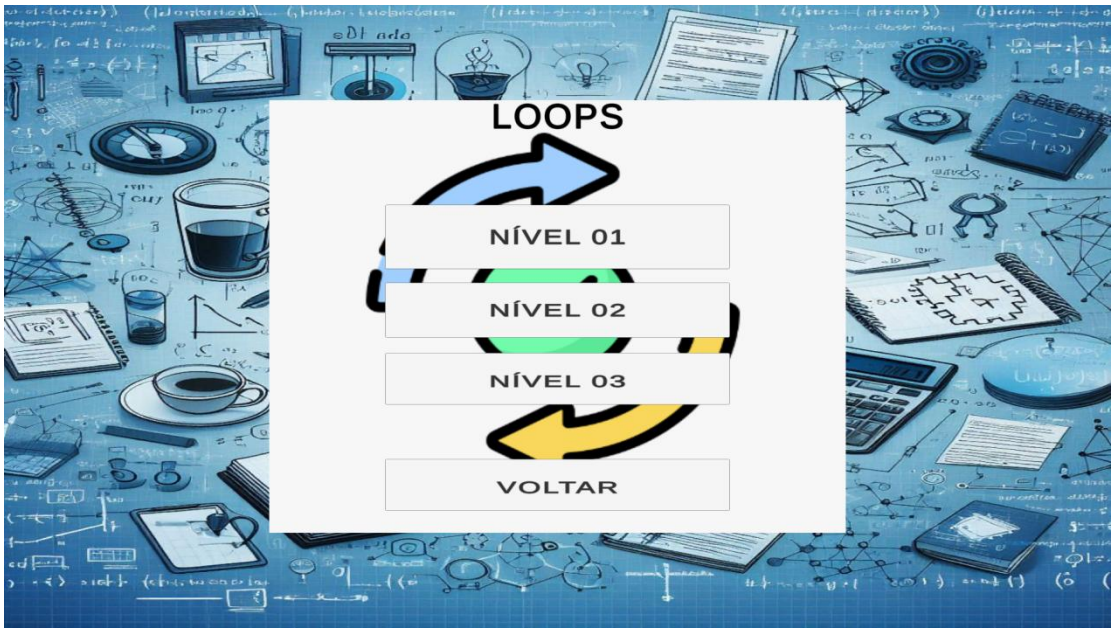


Figura 25- SubMenu do nível Loops.

A primeira fase deste nível é ilustrado pelas figuras 26 e 27 onde no seguinte cenário temos um percurso que precisa ser percorrido, porem, é necessário que se diga a direção e a quantidade de passos que o personagem precisa dar, o usuário poderá fazer isso usando *textboxs* que estão na parte superior da tela, nelas ele colocará os números das casas que ele deseje por onde vai andar e a direção, ele poderá escolher através das setas direcionais azuis, nesta fase ele exercitará o uso da estrutura de repetição *For* e entenderá como montar e usar corretamente esta estrutura.

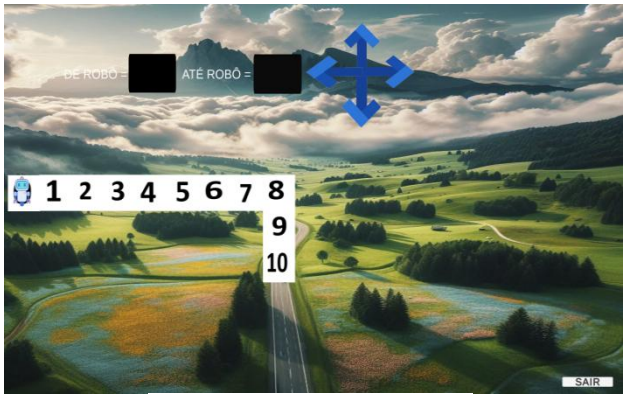


Figura 26- primeira fase do nível 04.

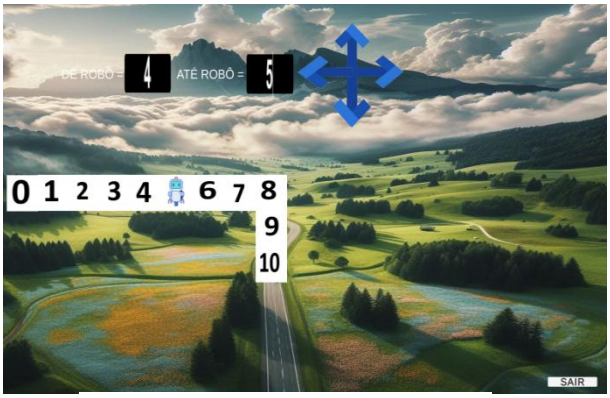


Figura 27- primeira fase do nível 04, após uma ação.

Caso por algum motivo o usuário digite um valor e/ou uma direção que faça o robô sair do trajeto pré-determinado, aparecerá na tela uma mensagem de erro dizendo que as condição digitadas não estão de acordo com o caminho proposto.

Funções

Neste nível, pretendemos Mostrar como as funções podem agrupar um conjunto de comandos para serem reutilizados várias vezes.Nas fases mostradas na figura 28 vamos explicar a importância de escrever funções e sua importância dentro de um programa.

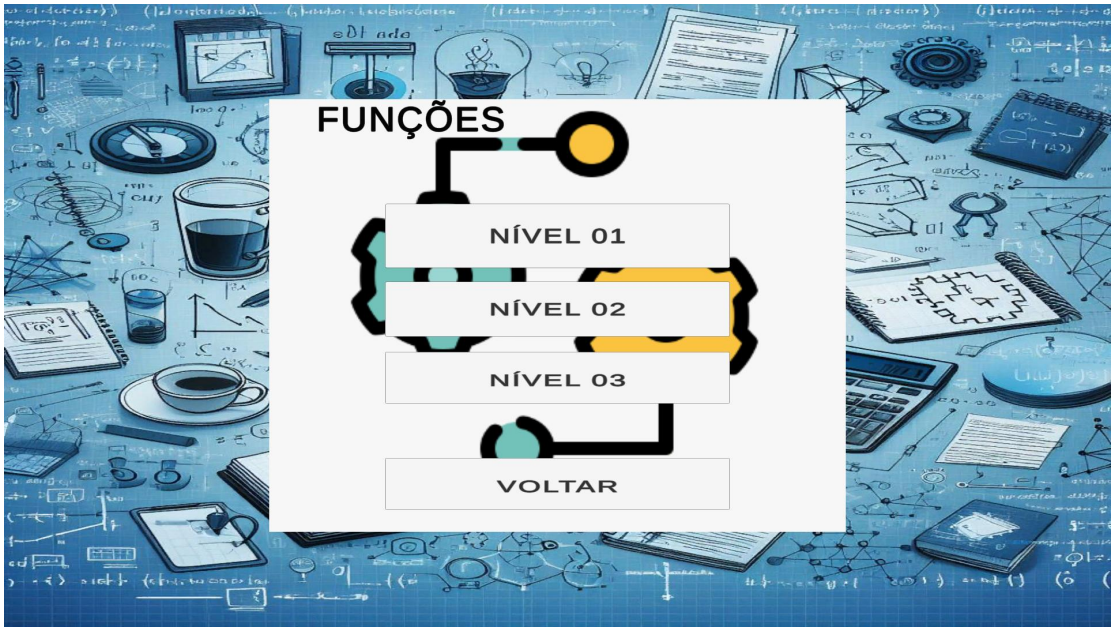


Figura 28- SubMenu do nível Funções.

A figura 29 ilustra o cenário da fase 01 deste nível, o usuário encontrar a seguinte situação, na parte inferior tempos 3 quadrados coloridos que são arrastáveis, de mesma dimensão na parte superior temos 3 quadrados laranjas que servirão como locais para se encaixar os quadrados coloridos, um pouco a esquerda da imagem podemos ver um troféu que está envolto por uma parede amarela, o intuito desta fase é fazer o jogador descobrir qual a sequencia certa de cores que farão a caixa com o troféu abrir. Com essa prática podemos mostrar que cada quadrado colorido é uma função deferente que tem sua importância e que elas precisam estar no local certo e na ordem

certa para que possam funcionar corretamente, as figuras 30 e 31 vão mostrar o funcionamento tanto para o acerto quanto para o erro do usuário.

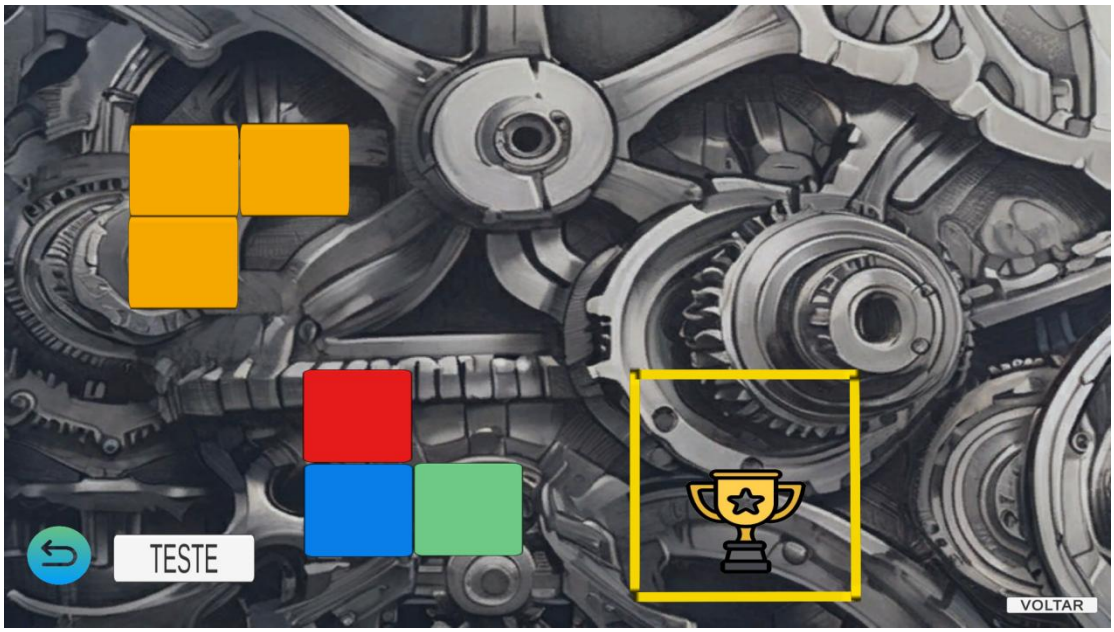


Figura 29- Primeira fase do nível 05.

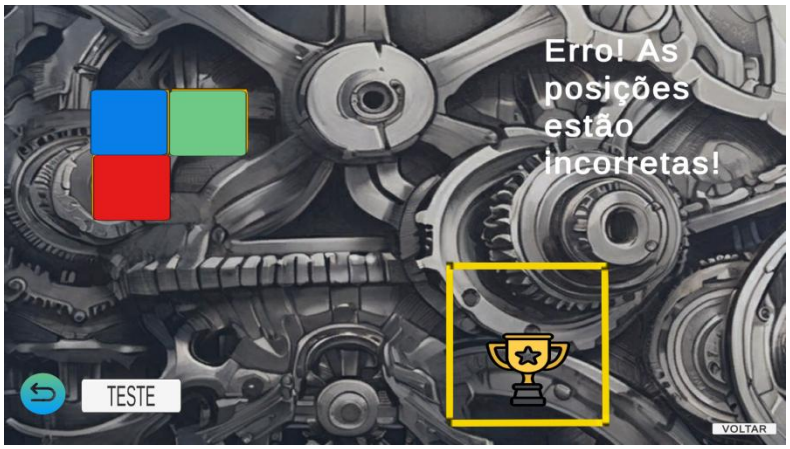


Figura 30- Primeira fase do nível 05- Case de erro.

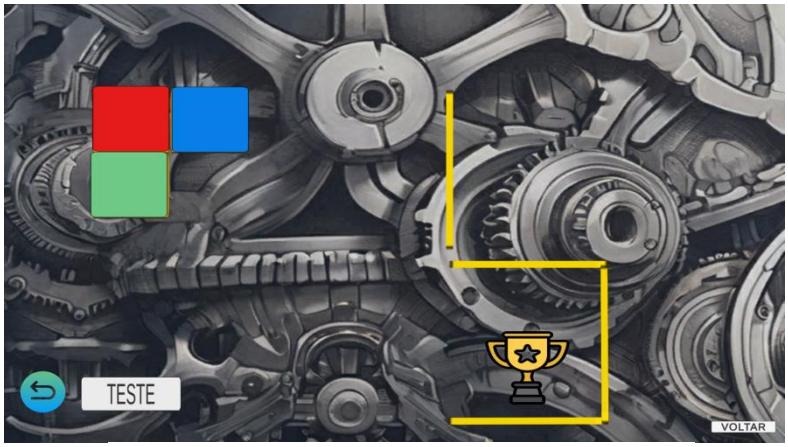


Figura 31- Primeira fase do nível 05- Case de acerto.

Depuração

Neste nível, mostrado na figura 32, pretende-se ensinar as crianças a identificar e corrigir erros em seus programas. Incentiva-las a resolver problemas e a entender como os erros podem ser úteis para aprender.

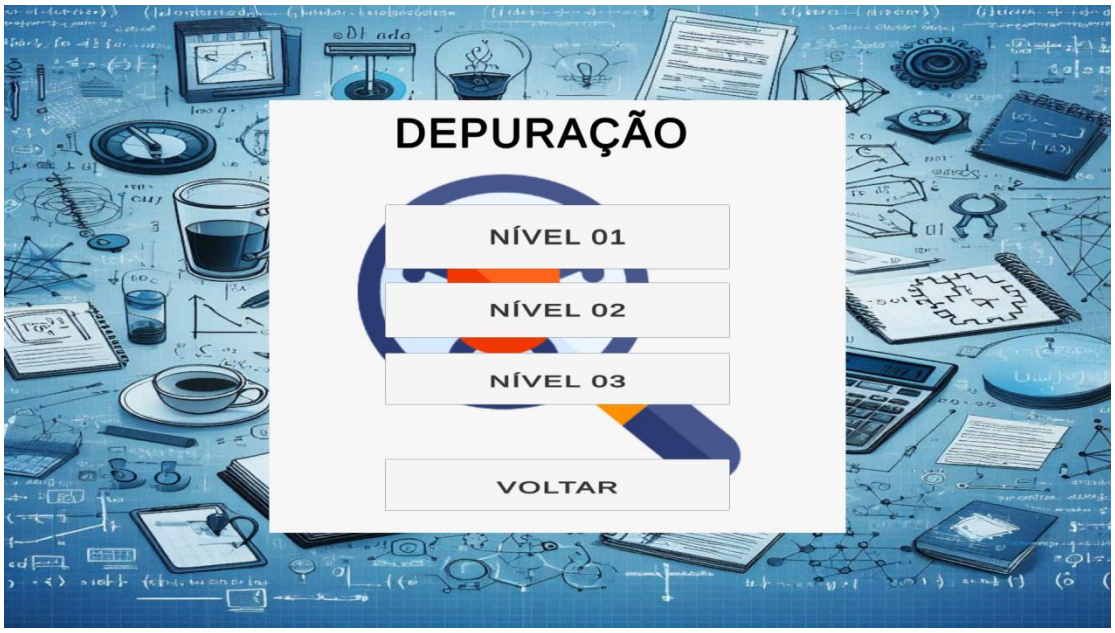


Figura 32- SubMenu do nível Depuração.

A figura 33 mostra o ambiente do nível 01 onde surgirá um trecho de pseuso código aleatoriamente, nesses trechos terão lacunas em branco que o usuário precisará preencher com a palavra que ele julgar adequada para estar ali, após preencher todos os códigos com as palavras ele apertará no botão abaixo com o nome testar , se as palavras estiverem certas irá aparecer uma mensagem de parabéns e outro pseudo código aparecerá para ser preenchido, caso contrario uma mensagem de erro aparecerá.

Figura 33- Primeira fase do nível 07.

Atualização e manutenção

Como este software ainda está em uma versão beta, algumas fases podem apresentar falhar na execução ou na compilação, e algumas ainda não foram desenvolvidas completamente, deste modo, para essas fases teremos a seguinte tela, ilustrada na figura 34.

Figura 34- Tela de Atualizações.

6 Arquitetura e Implementação

A arquitetura e a implementação são componentes fundamentais para o desenvolvimento de um aplicativo educacional eficaz e robusto. Neste capítulo nós detalharemos a estrutura técnica do nosso aplicativo, incluindo a arquitetura de software, os componentes principais e o processo de implementação. Nosso objetivo é garantir que o aplicativo não só atenda aos requisitos funcionais e não funcionais, mas também seja escalável, manutenível e eficiente.

A arquitetura de software do aplicativo educacional foi projetada para suportar uma plataforma interativa e responsiva, capaz de fornecer uma experiência de usuário envolvente e educativa. A arquitetura é composta por vários módulos que interagem de maneira coesa para oferecer funcionalidades completas e integradas.

6.1 Estrutura

Toda a aplicação foi desenvolvida no Unity, separamos as seções por pastas para deixar tudo o mais organizado possível a figura 35 mostra a organização do diretório principal.

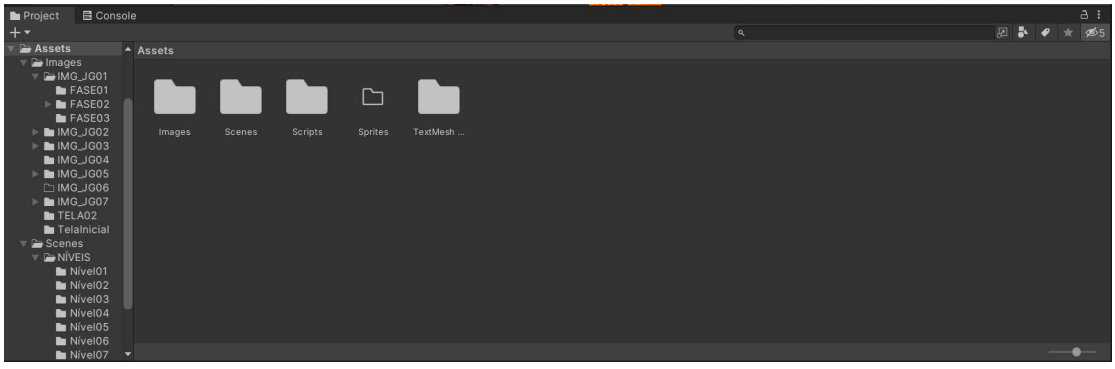


Figura 35- Diretório principal.

Dessa maneira separamos todos os arquivos, que foram também separados por seus respectivos níveis e fases, isso deixa a construção da aplicação mais prática e produtiva , a figura 36 demonstra melhor como é a organização dentro do diretório principal.

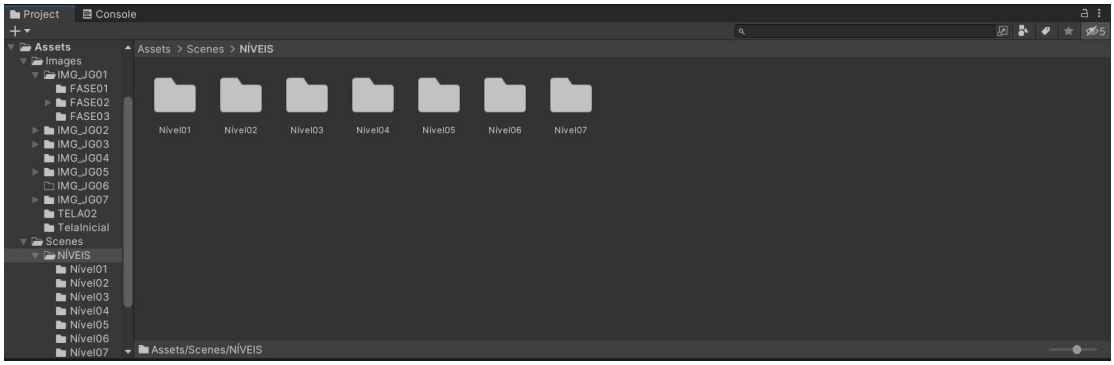


Figura 36- Organização dos níveis no diretório principal.

Cada imagem, script e objeto necessário para a construção do aplicativo foi devidamente agrupado em seu respectivo nível, essa boa prática de programação é fundamental para futuras atualizações e também torna fácil a navegação por futuros programadores que não participaram da construção inicial da aplicação.

6.2 Menus e trocas de tela

Existem inúmeras telas em nossa aplicação, que estão divididas entre telas de menu, telas de transições entre menus e os níveis propriamente ditos, a figura 37, mostra como a tela te Menu funciona, na imagem abaixo percebemos a disposição dos botões(JogarButton, SairButton, OpcoesButton), dentro de cada um desses botões há um script(MenuManeger) que determinará o funcionamento e pra onde cada botão levará após ser clicado.

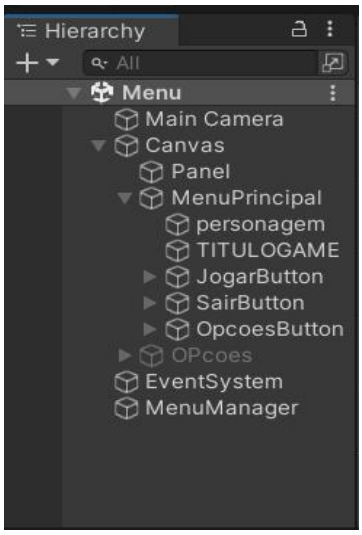


Figura 37- Hierarchy tela Menu.

A figura 38 mostra o funcionamento do script MenuManeger, nesse script temos as seguintes funções: *AbrirOpcoes(), FecharOpcoes(), SairJogo() e fase()*. Essas funções são chamadas dentro dos seus respectivos botões, o menu de opções é um painel que não está visível na tela e aparece assim que o usuário clica no botão opções, dentro de cada função temos a interação de desativação de um dos dois painéis existentes(painelMenu, painelOpcoes), essa lógica fica a critério da função do botão onde um painel será desativado e o outra ativado.

A função *fase()* vai ficar responsável pela lógica de trocar as telas, esta função não está presente apenas no menu mas também em todas as interações de troca de tela da aplicação, as figuras 39 e 40 vão mostrar como essa função é chamada nos botões.

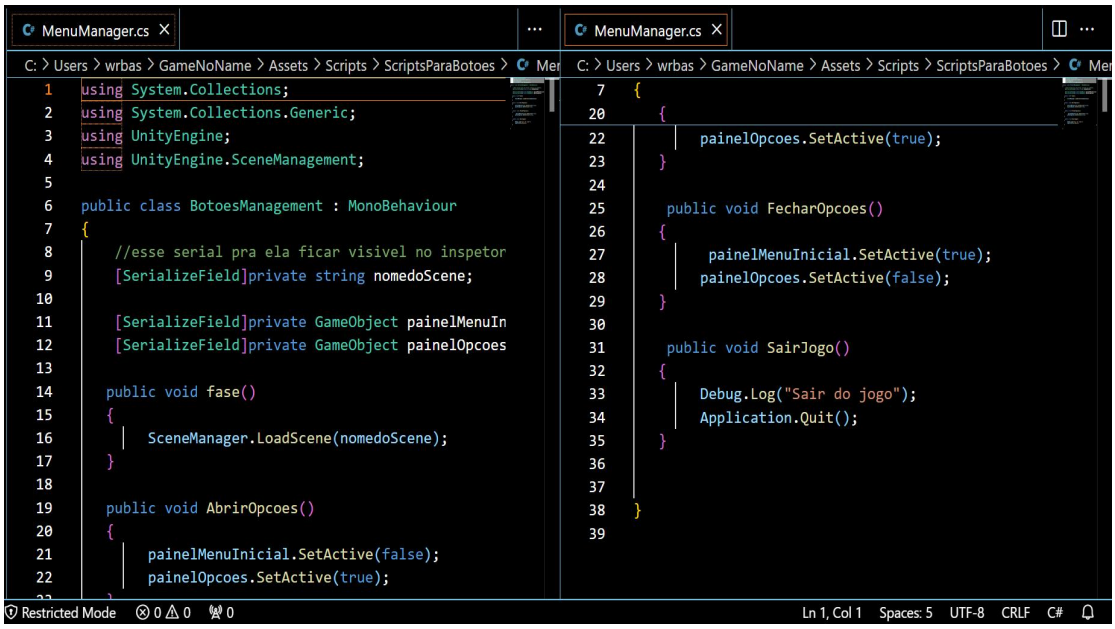


Figura 38- Script MenuManeger.

Podemos ver nas imagens o botão(JogarButton) chamando a função *fase()* no parâmetro *On Click()* e na imagem da direita o funcionamento do script MenuManeger dentro deste botão indicando qual tela abrirá após o botão ser clicado.

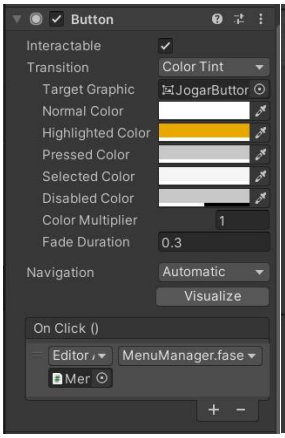


Figura 39- Configuração padrão do botão

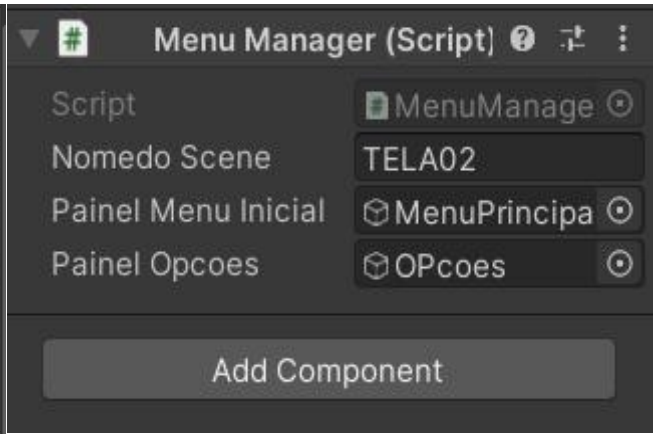


Figura 40- Script MenuManeger.

6.3 Fases

A figura 41 mostrará a organização dos componentes da primeira fase, no painel hierarchy podemos ver todos os componentes disponíveis na tela da primeira fase no Panel temos 5 tipos de botão diferentes*(int, char,string, float*

e bool) e todas as imagens que estão nomeadas com seu tipo seguido de um numero, exemplo: bool01, char03, int02, etc.

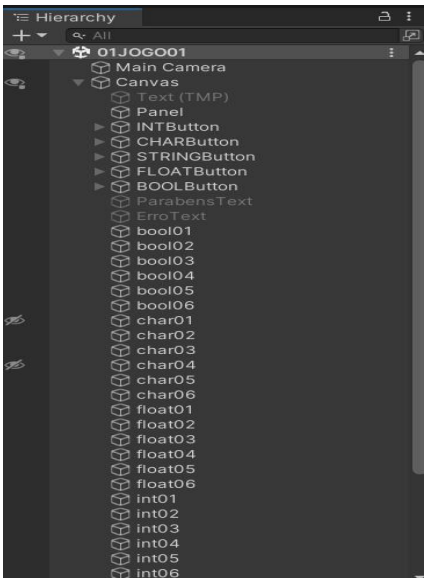


Figura 41- Organização hierarchy 01jogo01.

6.3.1 Fase01 jogo01

O *sript* principal que vai controlar tudo que acontecerá nessa fase é o *GM01, mostrado na figura 42, ele controlará as duas principais variáveis(variável botão, variável imagem) e as formas como elas vão interagir com o controle do usuário.*

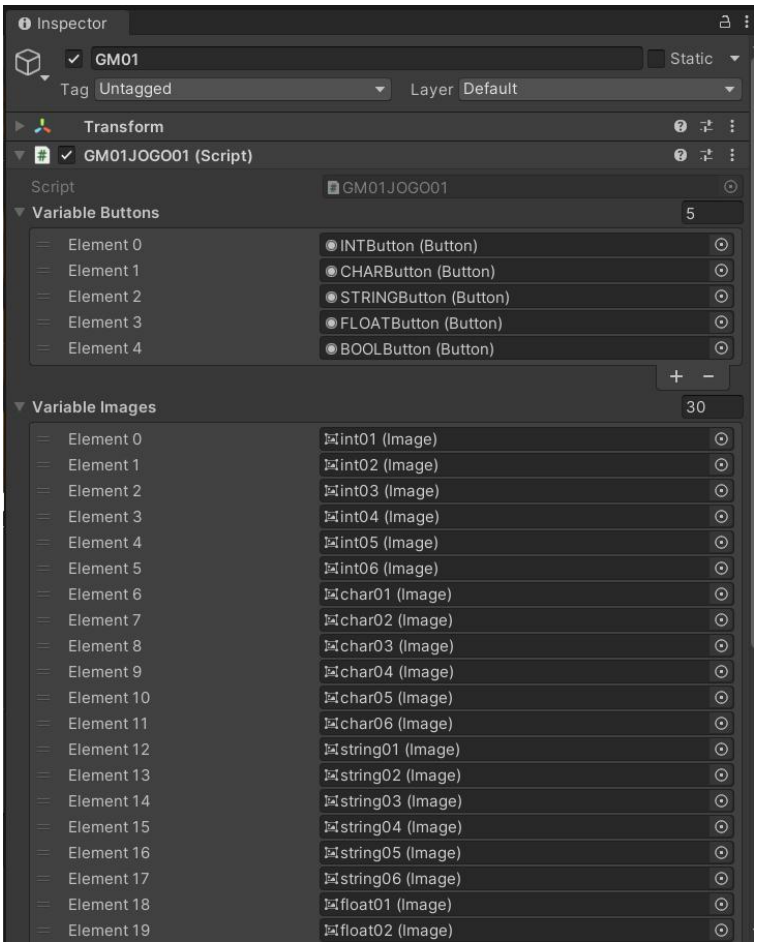


Figura 42- Organização do script GM01jogo01.

No script mostrado na figura 43 podemos ver a declaração das variáveis e seus tipos(*Button, Image, GameObjec*t) cada tipo tem uma finalidade diferente dentro do jogo o código em C# está todo comentado para facilitar o entendimento de possíveis estudos e alterações, mas dentre todas as funções gostaríamos de destacar é a função *HideAllImagens()* que esconde todas as imagens e mostra apenas uma aleatoriamente, isto é importante pois a partir dessa lógica toda a fase funciona e gera uma interação proveitosa com o usuário.

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

public class GM01JOGO01 : MonoBehaviour
{
    public Button[] variableButtons; // Array dos b
    public Image[] variableImages; // Array das ima
    public GameObject parabensText; // Objeto de te
    public GameObject erroText; // Objeto de texto

    private int currentButtonIndex; // Índice do bc

    private void Start()
    {
        parabensText.SetActive(false); // Desativa
        erroText.SetActive(false); // Desativa o ob

        HideAllImages(); // Esconde todas as imagen

        // Adiciona um listener de clique para cada
        for (int i = 0; i < variableButtons.Length;
        {
            int buttonIndex = i; // Cria uma variav
            variableButtons[i].onClick.AddListener(
        }

        ShowRandomImage(); // Mostra a primeira ima
    }

    private void ShowRandomImage()
    {
        // Esconde todas as imagens
        HideAllImages();

        // Seleciona aleatoriamente um novo botão
        currentButtonIndex = Random.Range(0, variab

        // Calcula os índices das imagens correspon
        int startImageIndex = currentButtonIndex *
        int endImageIndex = startImageIndex + 5;

        // Exibe uma imagem aleatória dentro do con
        int randomImageIndex = Random.Range(startIm
        variableImages[randomImageIndex].gameObject
    }

    private void HideAllImages()
    {
        // Esconde todas as imagens
        foreach (Image image in variableImages)
        {
            image.gameObject.SetActive(false);
        }
    }

    private void OnVariableButtonClick(int buttonIn
    {
        // Verifica se o botão clicado corresponde
        if (buttonIndex == currentButtonIndex)
        {
            parabensText.SetActive(true); // Exibe

            // Aguarda 2 segundos e chama a função
            Invoke("HideParabensTextAndCurrentImage
            Invoke("ShowRandomImage", 2f);
        }
        else
        {
            erroText.SetActive(true); // Exibe o te

            // Aguarda 2 segundos e oculta o texto
            Invoke("HideErrorText", 2f);
        }
    }

    private void HideParabensTextAndCurrentImage()
    {
        parabensText.SetActive(false); // Oculta o texto de parabens

        // Oculta todas as imagens
        HideAllImages();
    }

    private void HideErrorText()
    {
        erroText.SetActive(false); // Oculta o texto de erro
    }
}
```

Figura 43- Código do script GM01jogo01.

6.3.2 Fase01 jogo02

Na figura 44 temos acesso a hierarquia de objetos presentes na cena desta fase do lado direito podemos ver os objetos tipo caixa e os objetos tipo botão e os textos necessários, já no lado esquerdo da imagem usamos como exemplo o *IntButton* para mostrar o conteúdo que está dentro dele mas ressaltamos que todos os botões tem este mesmo padrão de comandos e o mesmo script chamado *Dropzone,* este script é responsável por gerir a lógica de identificação das imagens presentes na cena. Na figura 45 podemos ver mais componentes presentes nos objetos arrastáveis e damos destaque ao parâmetro *Box Collider 2D* que permite a interação e a detecção de contato entre os objetos na cena.

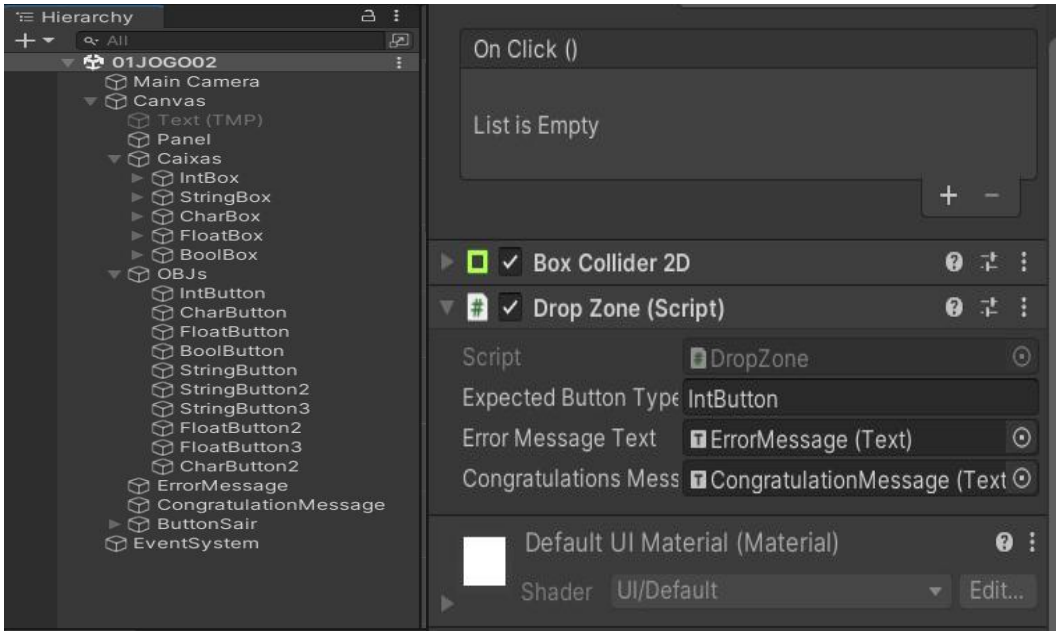


Figura 44- Hierarchy 01Jogo02 e disposição do script dropZone.

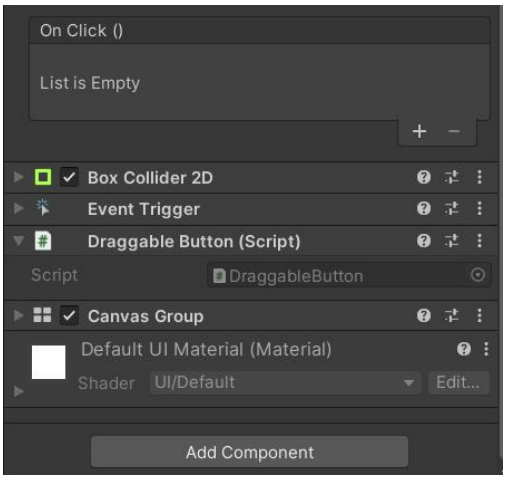


Figura 45- Disposição do script Dreggable usado nos objetos da cena.

Na figura 46 podemos ver o código presente no script *DraggbleButton*, esse script é responsável por permitir que as imagens sejam arrastadas para qualquer lugar da tela, essas funções já são nativas do Unity, logo, não precisamos criar nada, apenas foi necessário chamar as funções corretas.

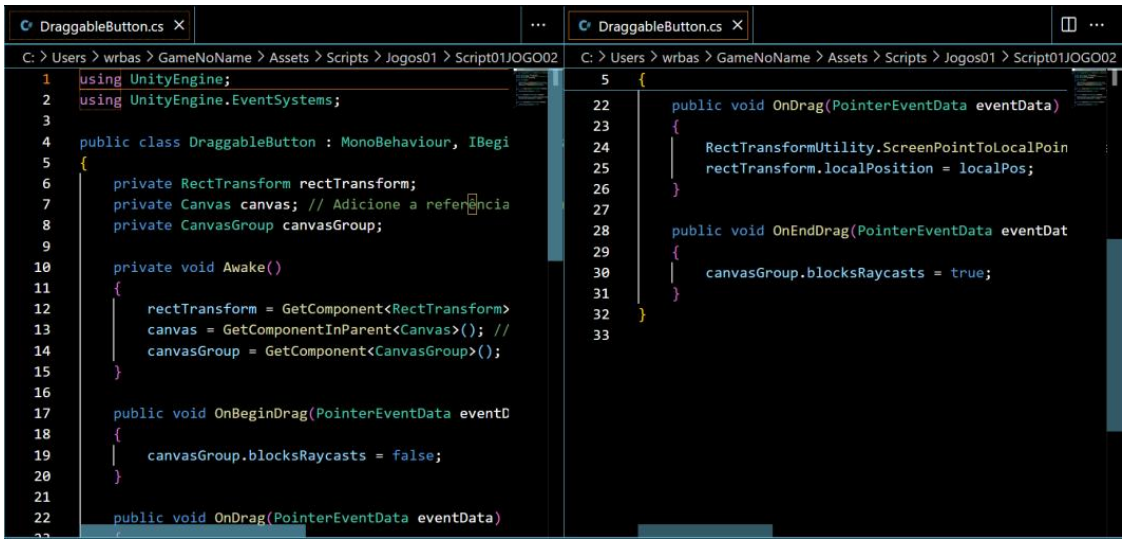


Figura 46- Código script DraggbleButton.

O segundo script presente nesta fase é o *DropZone*, mostrado na figura 47 e na figura 48 , ele está presente nas “caixas” visíveis na tela, e sua função é detectar quando um objeto é colocado sobre ele, essa detecção é fundamental para todas a lógica do jogo.

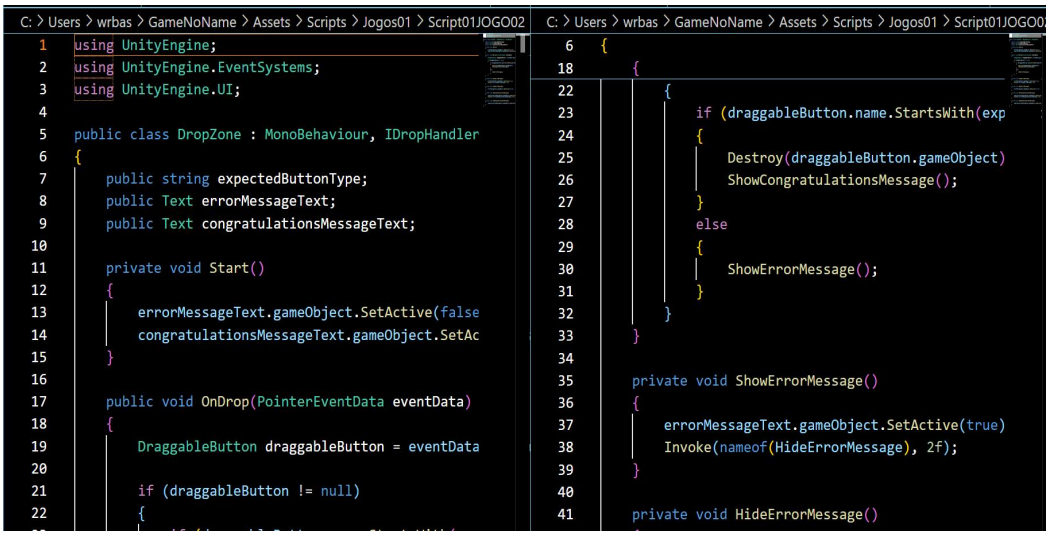


Figura 47- Código script DropZone.

Figura 48- Continuação código script DraggbleButton.

6.3.3 Fase01 jogo03

A seguir temos 2 figuras (49 e 50) a primeira figura mostrará a hierarquia de objetos presentes na cena, o lado direito da mesma imagem mostra o conteúdo presente em todas os botões, vamos dar destaque dessa vez ao componente Rigidbody 2D que vai permitir simular a gravidade nos objetos, já a segunda imagem localizada ao lado esquerdo mostrará o padrão presente nas imagens que aparecem na cena, ambas a imagens também contem diferentes scripts cada qual fundamental para o funcionamento da lógica.

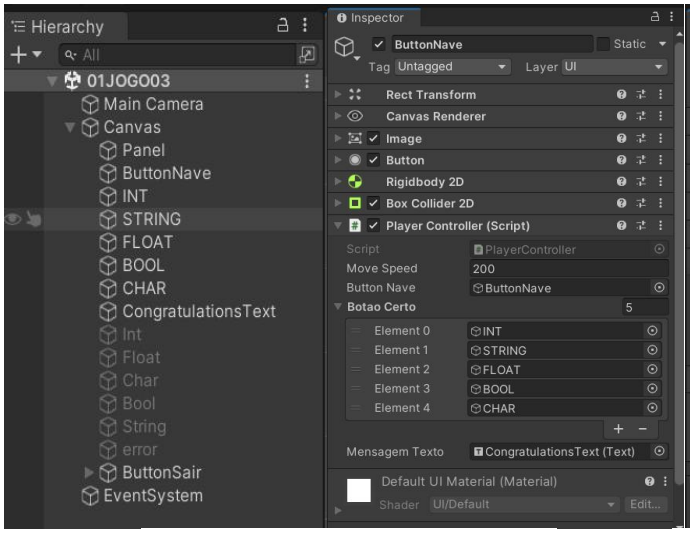


Figura 49- Hierarquia de ordens da cena 01Jogo03

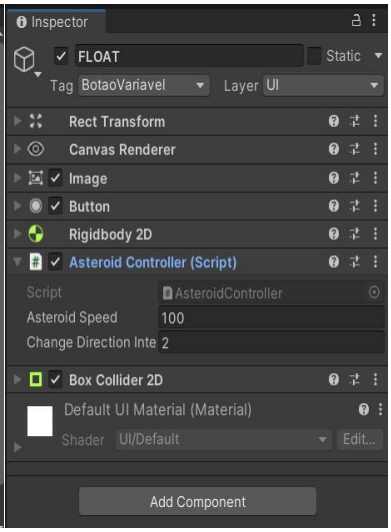


Figura 50- Padrão de configuração da cena 01Jogo03

O script presente nas imagens ´o AsteroidController, como podemos ver nas figuras 51, 52, 53 e 54, este script é responsável por dar o comportamento de “asteroides” para as imagens, fazendo com que elas circulem pelo cenário de forma aleatória e automática e com uma velocidade constante. A função *Update()* é a responsável por atualizar constantemente a posição da imagem e mudar seu deslocamento aleatoriamente.

C: > Users > wrbas > GameNoName > Assets > Scripts > Jogos01 > Script01Jogo03 > AsteroidController.cs

```
1   using UnityEngine;
2
3   public class AsteroidController : MonoBehaviour
4   {
5       public float asteroidSpeed = 1.0f; // Velocidade fixa dos asteroides
6       public float changeDirectionInterval = 2.0f; // Intervalo para mudar a direção em segundos
7
8       private Vector2 screenSize;
9       private Vector3 targetDirection;
10      private float timeUntilChange;
11
12
13
14
15      void Start()
16      {
17          screenSize = new Vector2(Screen.width, Screen.height);
18          timeUntilChange = changeDirectionInterval;
19          Respawn();
20      }
21
22      void Update()
```

Figura 51- Parte-1 script AsteroidController.

```
23      {
24          // Move o asteroide na direção atual
25          Vector3 newPosition = transform.position + targetDirection * asteroidSpeed * Time.deltaTime;
26          transform.position = WrapAroundScreen(newPosition);
27
28          // Conta o tempo até a próxima mudança de direção
29          timeUntilChange -= Time.deltaTime;
30
31          // Verifica se é hora de mudar de direção
32          if (timeUntilChange <= 0)
33          {
34              // Gera uma nova direção aleatória
35              targetDirection = Random.insideUnitCircle.normalized;
36
37              // Define um novo intervalo de mudança de direção aleatório
38              changeDirectionInterval = Random.Range(1.0f, 4.0f);
39              timeUntilChange = changeDirectionInterval;
40          }
41      }
42
```

Figura 52- Parte-2 script AsteroidController.

```
42
43      Vector3 WrapAroundScreen(Vector3 position)
44      {
45          // Mantenha os objetos dentro da tela, fazendo-os aparecer do outro lado
46          position.x = Mathf.Repeat(position.x, screenSize.x);
47          position.y = Mathf.Repeat(position.y, screenSize.y);
48          return position;
49      }
50
51      void Respawn()
52      {
53          // Reposicione o asteroide em uma posição aleatória na borda da tela
54          float randomX = Random.Range(0, screenSize.x);
55          float randomY = Random.Range(0, screenSize.y);
56          float side = Random.Range(0, 4); // 0: topo, 1: direita, 2: baixo, 3: esquerda
57
58          switch (side)
59          {
60              case 0:
61                  transform.position = new Vector3(randomX, screenSize.y, 0);
```

Figura 53- Parte-3 script AsteroidController.

```
61                  transform.position = new Vector3(randomX, screenSize.y, 0);
62                  break;
63              case 1:
64                  transform.position = new Vector3(screenSize.x, randomY, 0);
65                  break;
66              case 2:
67                  transform.position = new Vector3(randomX, 0, 0);
68                  break;
69              case 3:
70                  transform.position = new Vector3(0, randomY, 0);
71                  break;
72          }
73      }
74  }
75
```

Figura 54- Parte-4 script AsteroidController.

O segundo script presente nesta fase é o *PlayerControler ele pode ser visto nas* figuras 55 e 56*, esse código controla "mão" presente n cena, mão essa que é controlada pelo usuário dentro da cena do jogo. A função que vamos destacar dessa vez é a função CheckCollisionWithCorrectButton()* que vai justamente fazer a interpretação do toque e comparar se o usuário tocou a imagem certa, mostrando assim na tela uma mensagem de parabéns ou erro.

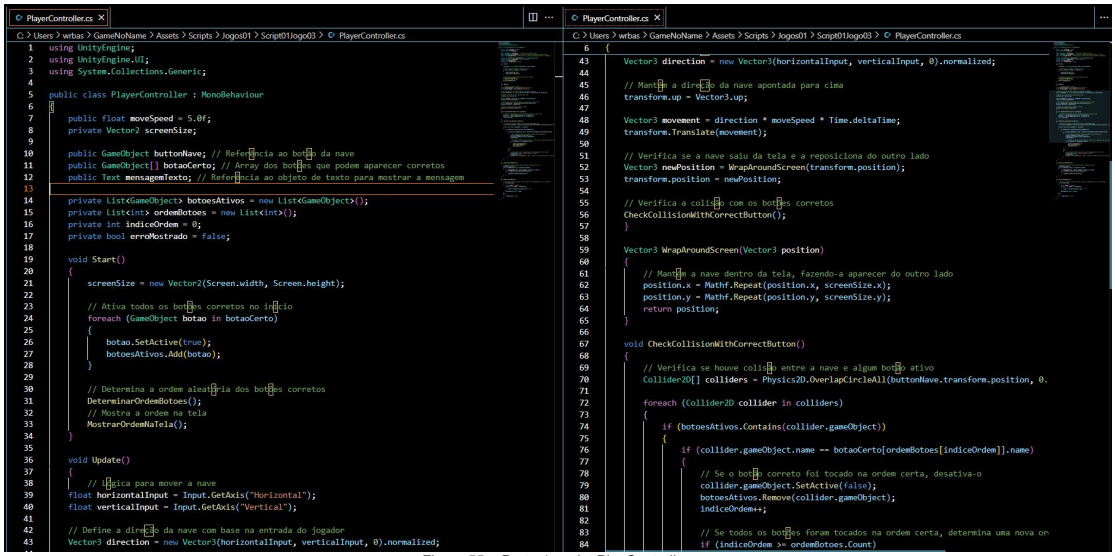


Figura 55- Parte-1 script PlayController.

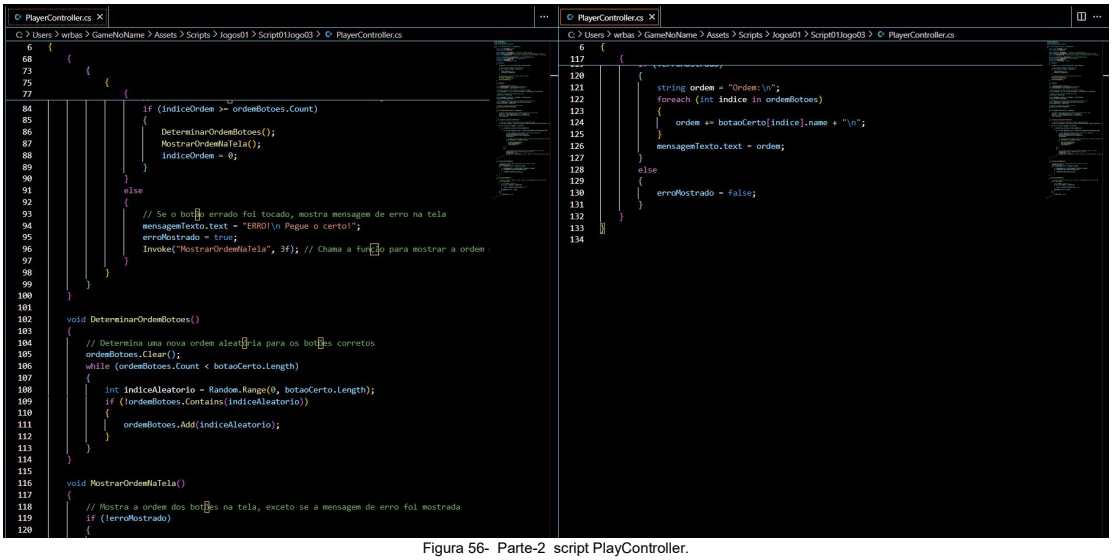


Figura 56- Parte-2 script PlayController.

6.3.4 Fase02 jogo01

Os jogos 01, 02 e 03 deste nível usam a mesma lógica, mesmo cenário e objetos com a diferença apenas de que a dificuldade é aumentada se incrementando mais objetos a cena. Usaremos então por base, o objetos e scripts do jogo01, a figura 57 no lado esquerdo mostra a hierarquia dos objetos presentes na cena enquanto ao lado direito vamos usar por base os componentes de um dos botões assumindo que todos têm o mesmo padrão e contém e mesmo script *BotaoAcao.*

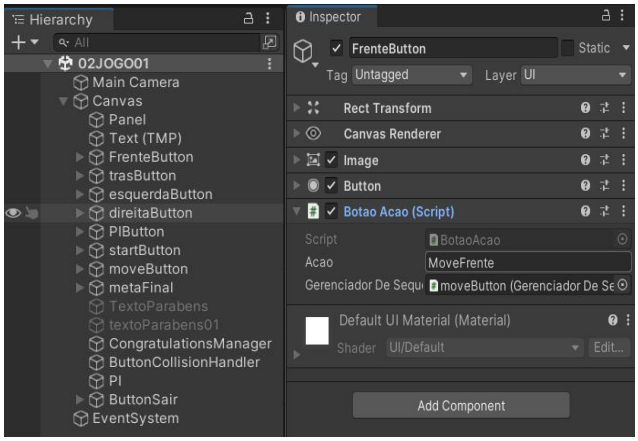


Figura 57- Hierarquia 02Jogo01 e componentes dos botões e movimento.

Na figura 58 temos mas 2 inspectors desta vez o da esquerda mostrará todos os atributos presentes no objeto *metaFinal*, que é o objeto central de conclusão desta fase, ao lado esquerdo da imagem temos o *moveButton* que é o objeto que se move pelo cenário e é contrtolado pelo usuário, detalhe que o *moveButton* tem um script especial que é encarregado pela lógica de sua movimentação.

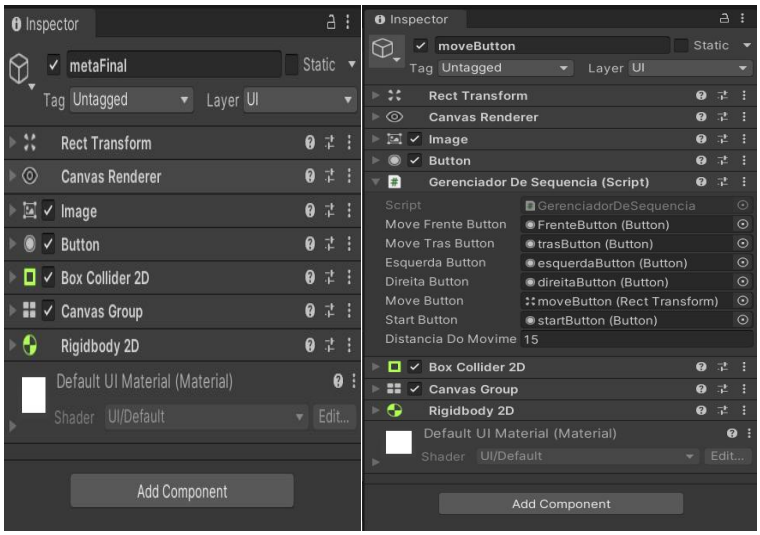


Figura 58- Atributos do objeto metaFinal e configuração padrão MoveButton.

A figura 59 mostra o script responsável por dar *start* no jogo, este script inicializa todas as outras lógicas e dá inicio as sequencias de comandos.

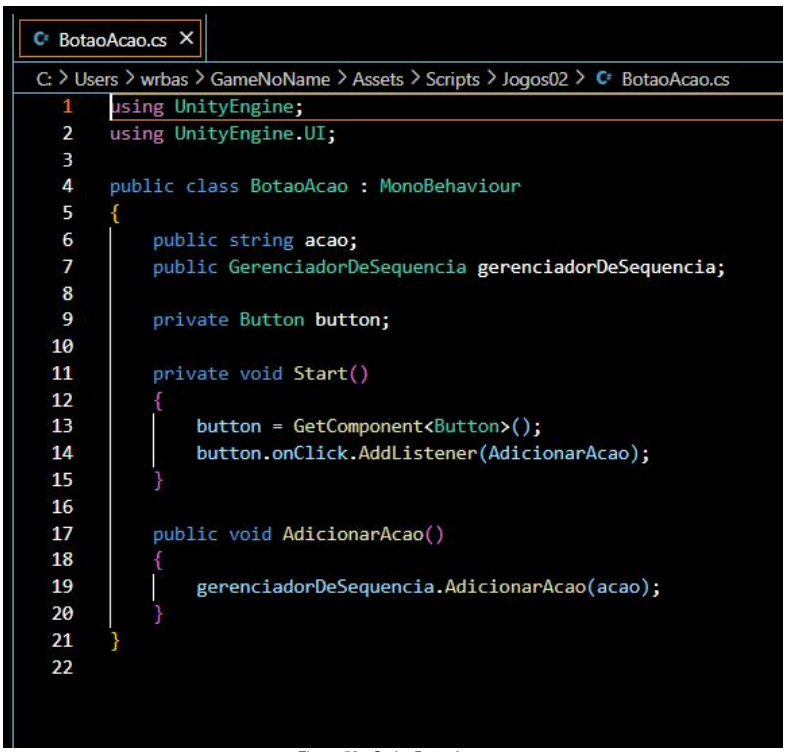


Figura 59- Script BotaoAcao.

A figura 60 mostra o script *BotaoPi,* esse script vai *resetar* o personagem da fase para a sua posição inicial de jogo, caso por algum motivo o usuário erre ou simplesmente queira tentar de novo.

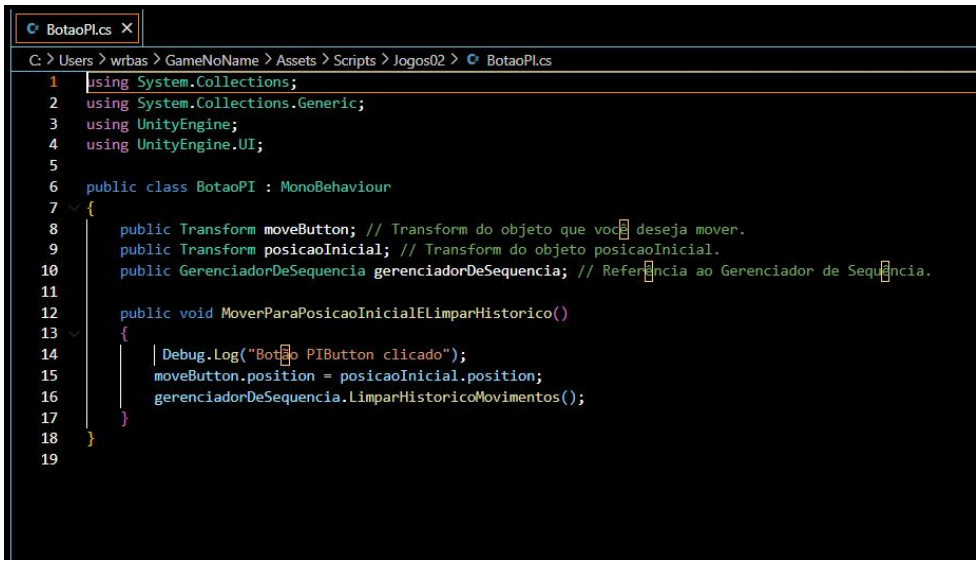


Figura 60- Script BotaoPI.

Este script mostrado na figura 61 detecta as colisões entre o personagens e os objetos colocados na cena, através dele nós identificamos se o nosso personagem bateu em alguma coisa ou não.

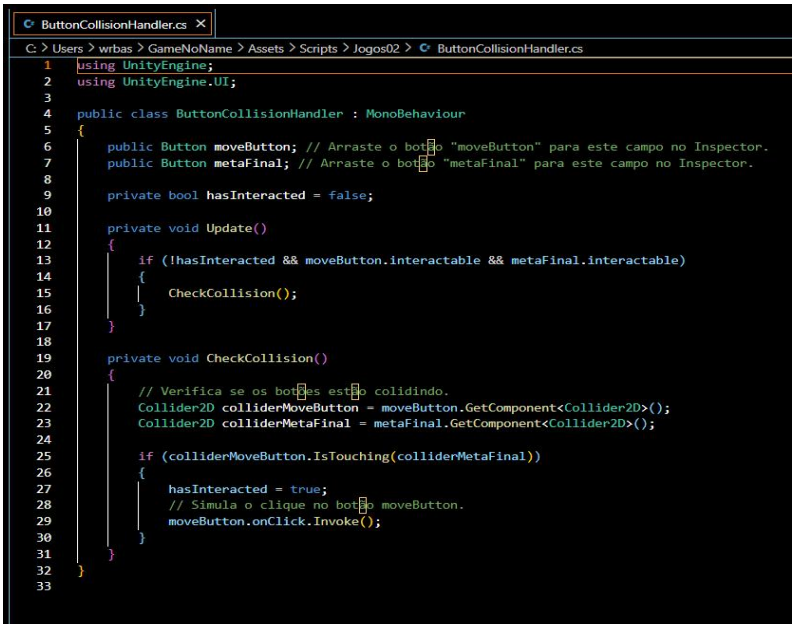


Figura 61- Script BurronCollisionHandler.

Existe também nesta fase um script somente para mensagens de parabéns, caso o usuário consiga alcançar o objetivo da fase, ele será acionado única e exclusivamente quando a fase foi concluída, a figura 62 mostra seu código.

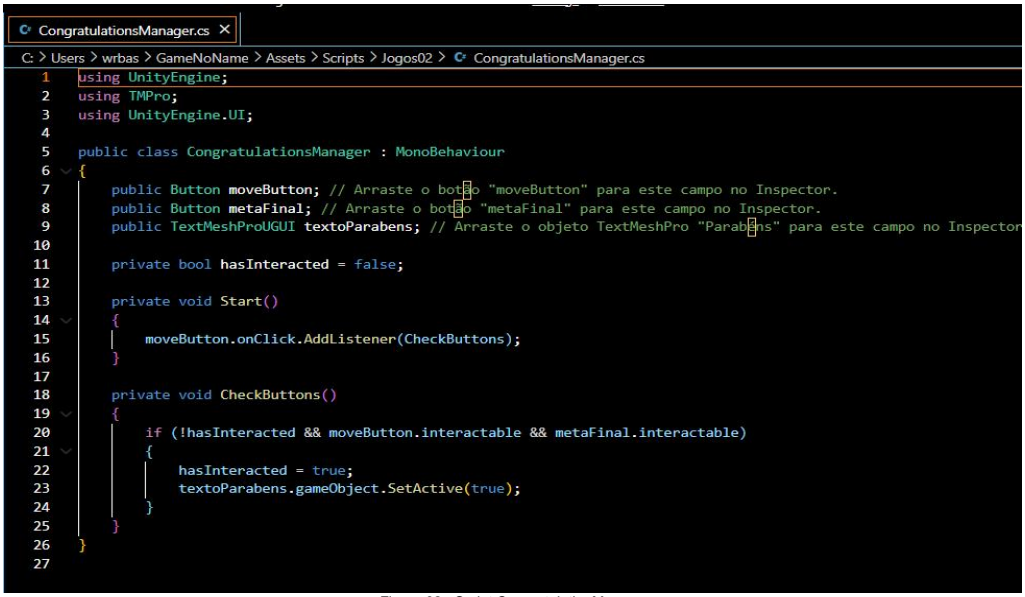


Figura 62- Script CongratulationManger.

Por último a figura 63 mostra o script responsável por armazenar as sequências digitadas pelo usuário e transferi-las para o personagem da fase esse script recebe os comandos enviados por meio das teclas de direção e envia esses comandos para o personagem dentro do jogo.

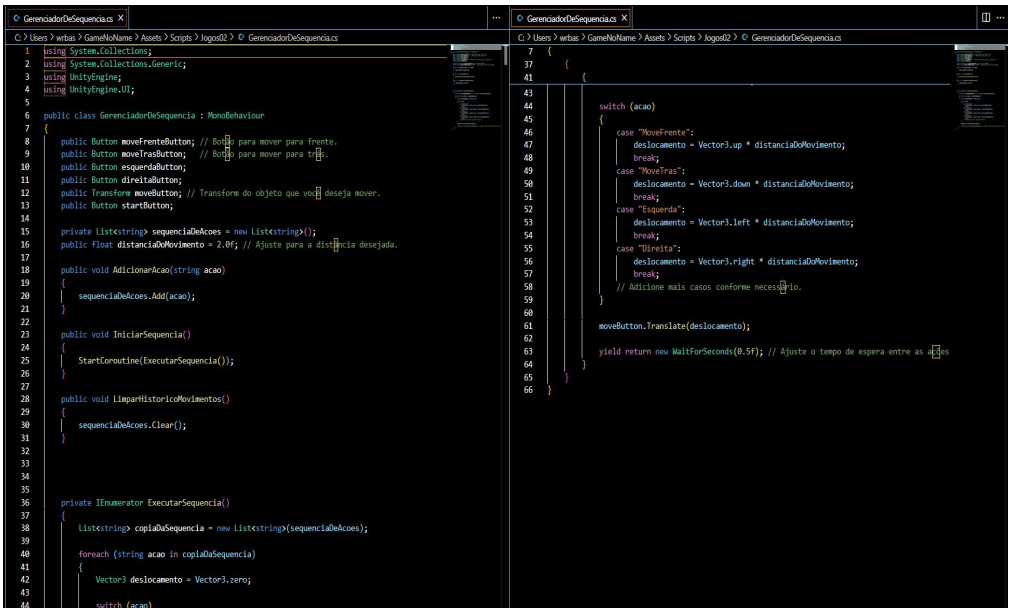


Figura 63- Script GerenciadorDeSequencia.

6.3.5 Fase03 jogo01

Na figura 64 temos a hierarquia dos objetos desta fase, painéis, imagens e botões necessários para o funcionamento do nível, já na figura 65 podemos observar a organização desses objetos no script *ControladorJogo,* podemos notar a forma como objetos e imagens são organizados em forma de vetores e cada um tem um referencia especifica.

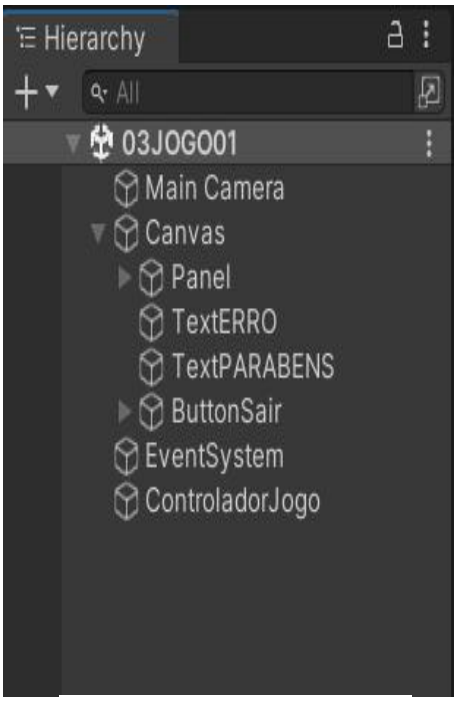


Figura 64- Hierarchy 03Jogo01

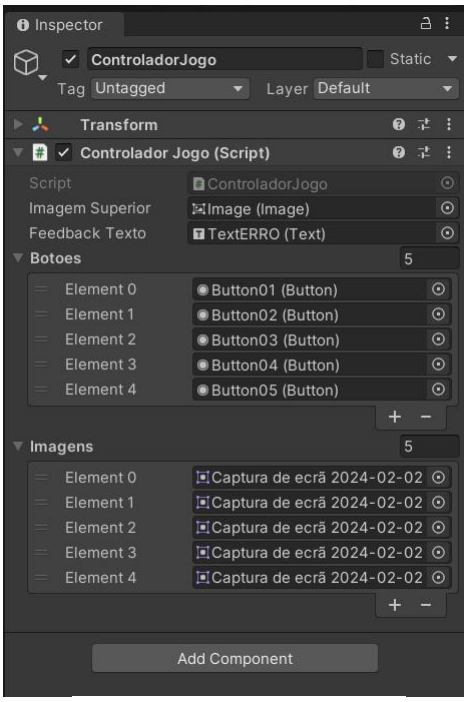


Figura 65- Organização script ControladorJogo.

O script *controladorJogo*, mostrado na figura 66, é o responsável por gerir toda a lógica desta fase dentro dele tempos 2 funções principais: IniciarJogo(), que será responsável por habilitar e desabilitar as telas e botões e a função *VerificarResposta()* que faz a checagem da resposta enviada pelo usuário e em caso de acerto permite que o usuário avance e caso haja erro informa com uma mensagem e da a opção de tentar novamente.

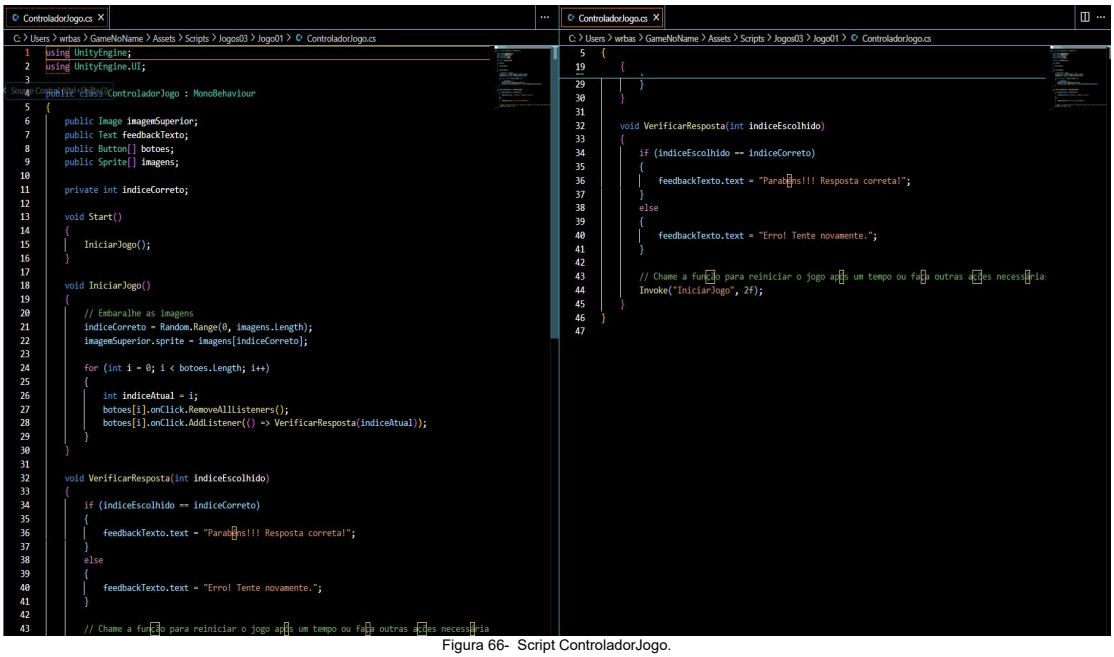


Figura 66- Script ControladorJogo.

6.3.6 Fase03 jogo02

A figura 67 mostra a sequência de objetos necessários para formar a cena e a sua organização tal como as parte ocultas que só aparecerão mediante a condições predefinidas nos scripts, este cenário é composto por diversos painéis que serão chamados mediante algumas condições.

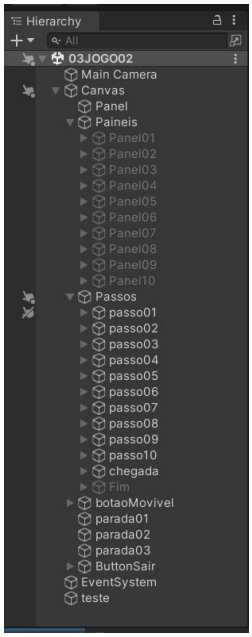


Figura 67- Objetos presentes na cena 03Jogo02.

O script mostrado na figura 68 é o responsável por gerênciar todos os botões, painéis ativos e não ativos e os textos que serão exibidos na tela, ele é fundamental para o bom funcionamento e organização do código.

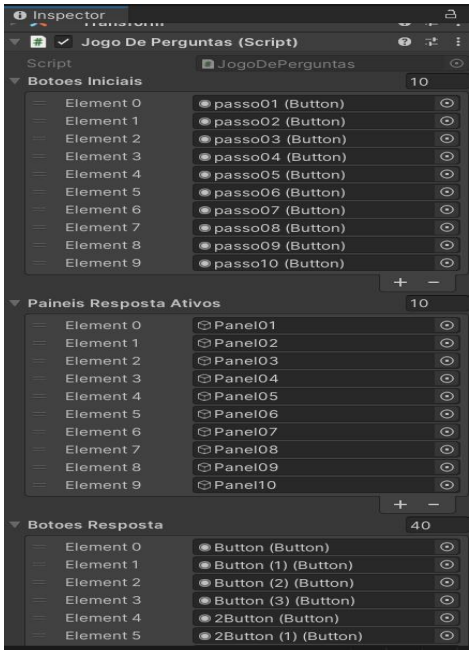


Figura 68- Organização do script JogoDePerguntas.

O script AtivarBotao como o próprio nome já diz tem a finalidade de habilitar os botões que ficam por padrão desativados na tela, para que toda a lógica funcione este script é fundamental, a figura 69 mostrará ele na integra.

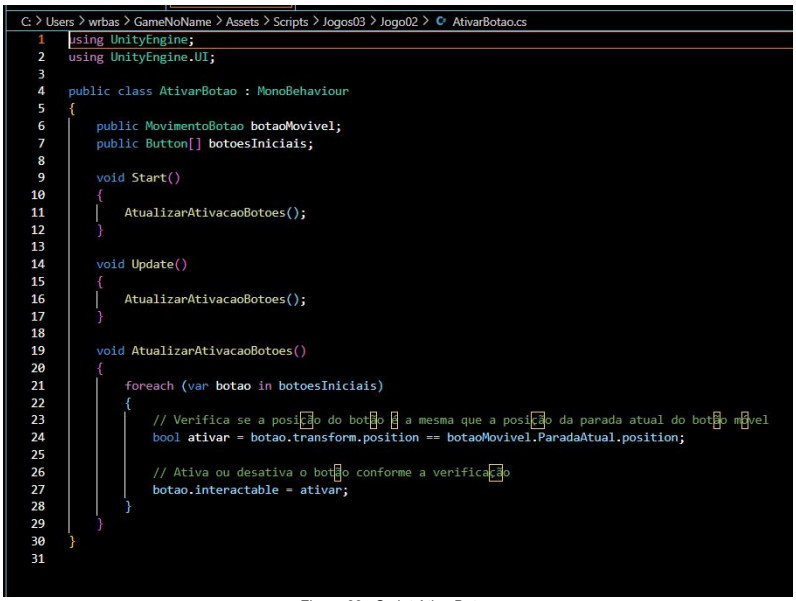


Figura 69- Script AtivarBotao.

As figura 70 e 71 mostram a seguir o principal script desta fase, pois este é responsável por gerenciar toda aquela organização mostrada na figura 68, dentre todas as funções vamos aqui destacar a função *AbrirPainelPerguntas()* que usa um foreach para fazer uma varredura sobre os painéis desativados habilitando assim apenas um painel e deixando ativo.

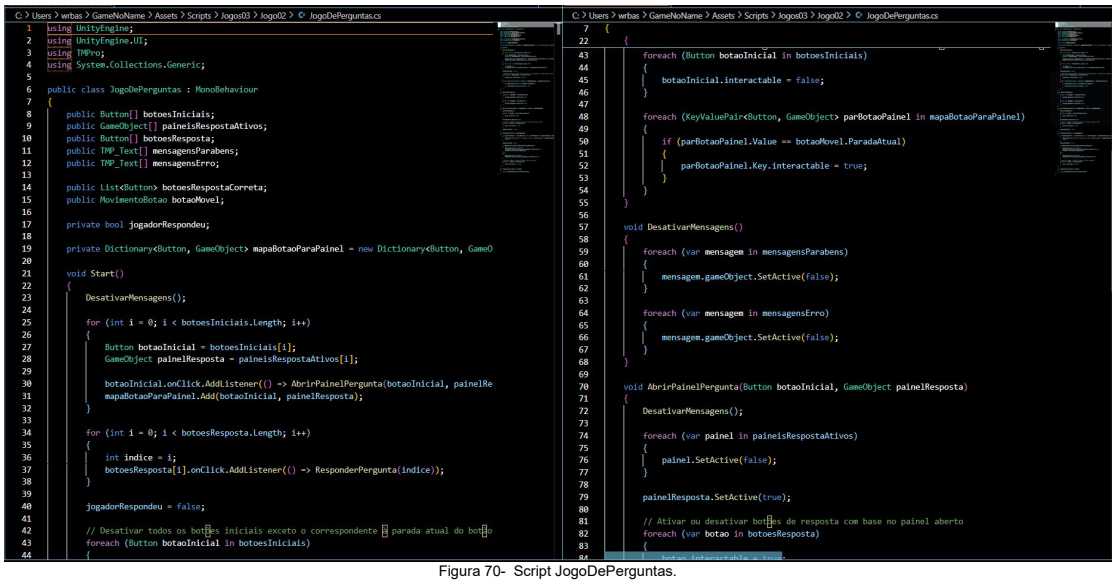


Figura 70- Script JogoDePerguntas.

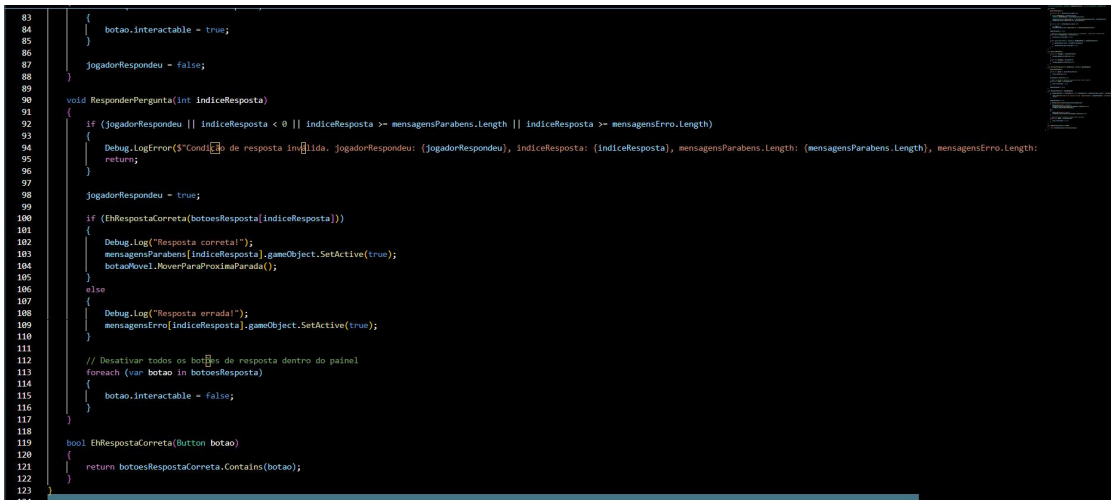


Figura 71- Script JogoDePerguntas- parte final.

O ultimo script presente neste nível é mostrado na figura 72 , sua lógica é simples porém fundamental, ele será o encarregado de proporcionar o movimento do personagem, mudando a posição conforme o desempenho do usuário.

C: > Users > wrbas > GameNoName > Assets > Scripts > Jogos03 > Jogo02 > MovimentoBotao.cs

```
1   using UnityEngine;
2
3   public class MovimentoBotao : MonoBehaviour
4   {
5       public Transform[] paradas;
6       private int paradaAtualIndex = 0;
7
8       public Transform ParadaAtual
9       {
10          get { return paradas[paradaAtualIndex]; }
11      }
12
13      void Start()
14      {
15          // Inicialmente, mova o botão para a primeira parada
16          transform.position = paradas[0].position;
17      }
18
19      public void MoverParaProximaParada()
20      {
21          // Move o botão para a próxima parada
22          paradaAtualIndex = (paradaAtualIndex + 1) % paradas.Length;
23          transform.position = paradas[paradaAtualIndex].position;
24      }
25  }
26
```

Figura 72- Script MovimentoBotao.

6.3.7 Fase04 Jogo01

A hierarquia de objetos da próxima fase está mostrada na figura 73, esta fase é composta por botões , imagens e textos, já a figura 74 podemos ver a organização do principal script desta fase(MovimentoAndante) e a forma como os elementos estão dispostos para o funcionamento do código.

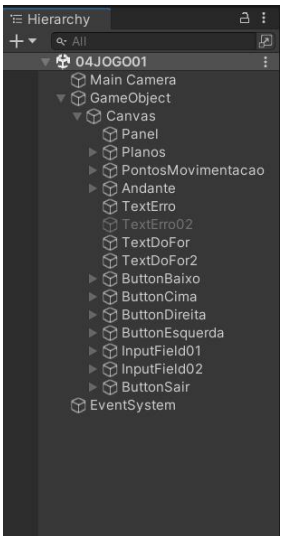


Figura 73- Hierarquia cena 04Jogo01.

Figura 74- Organização script MovimentoAndante.

Como todos os botões têm as mesmas funções e o mesmo script vamos mostrar na figura 75 apenas o inspector de um botão, tendo em vista que todos são padrões, na imagem em questão podemos ver o script MovimentoAndante sendo chamado.

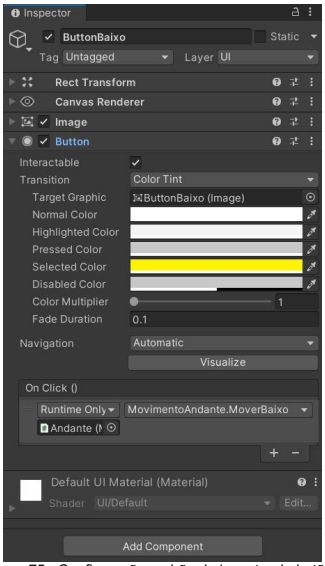
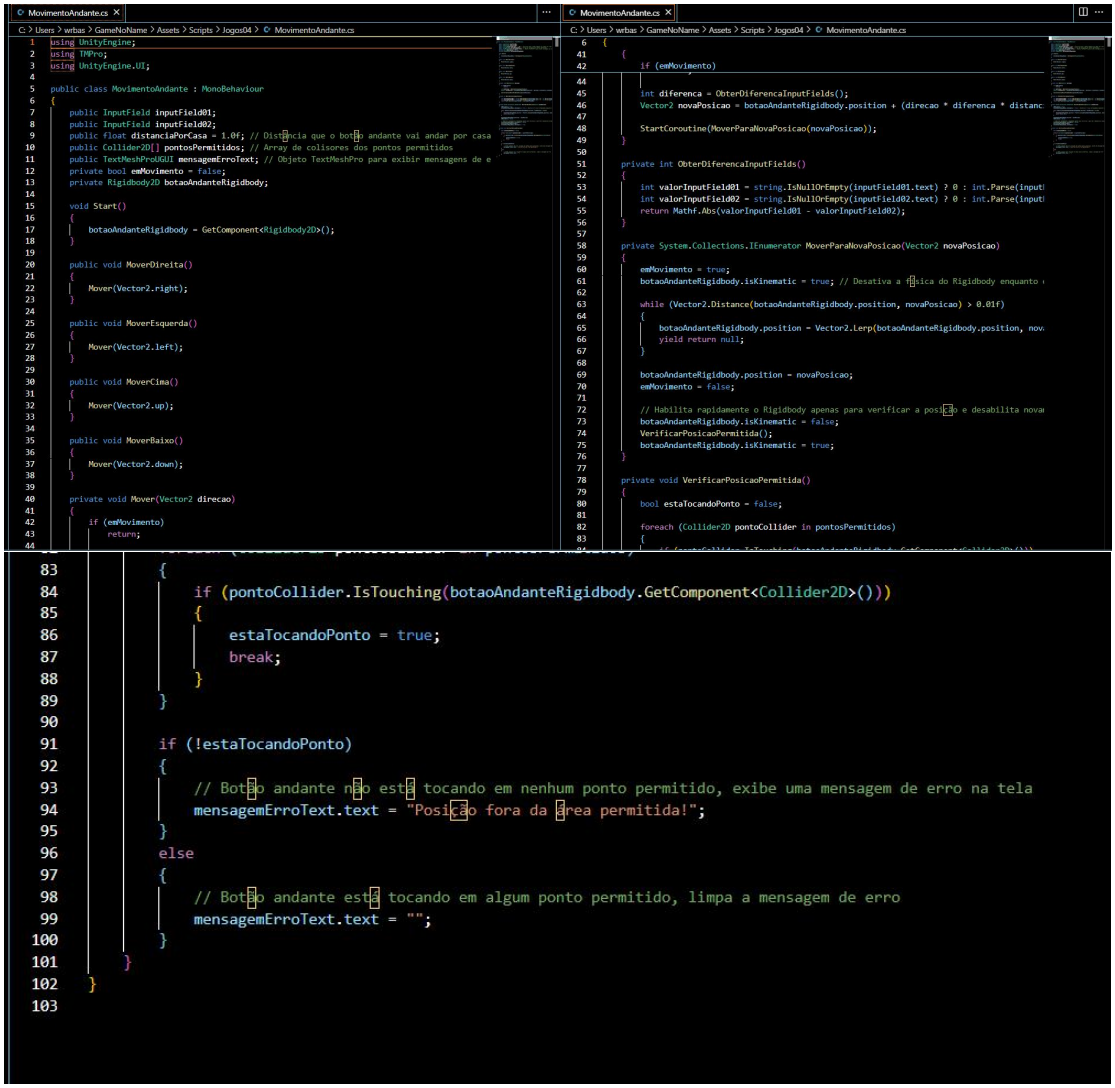


Figura 75- Configuração padrão do inspetor do botões.

A figura 76 mostrará em três partes o funcionamento do script MovimentoAndante e todas as suas funções detalhadas, este script é responsável por fazer o movimento do boneco na cena baseado em comando digitados pelo usuário, muitas funções já são nativas do sistema Unity o que torna mais fácil a criação do código, um exemplo disso são os métodos para movimentos laterais e diagonais, as bibliotecas instaladas já incluem essa funcionalidade.



```
83          {
84              if (pontoCollider.IsTouching(botaoAndanteRigidbody.GetComponent<Collider2D>()))
85              {
86                  estaTocandoPonto = true;
87                  break;
88              }
89          }
90
91          if (!estaTocandoPonto)
92          {
93              // Botão andante não está tocando em nenhum ponto permitido, exibe uma mensagem de erro na tela
94              mensagemErroText.text = "Posição fora da área permitida!";
95          }
96          else
97          {
98              // Botão andante está tocando em algum ponto permitido, limpa a mensagem de erro
99              mensagemErroText.text = "";
100         }
101     }
102 }
103
```

Figura 76- Script MovimentoAndante.

6.3.8 Fase05 Jogo01

As figuras 77 e 78 mostram os objetos presentes na cena e sua hierarquia e o conteúdo de um objeto chamado função, este objeto é padrão e existem vários outro que são idênticos a ele espalhados na cena, no objeto função podemos ver a disposição do script *BotaoArrastavel* que faz com que estes possam ser clicados e arrastados pela cena. Na figura 79 é mostrada a organização do script GameController, que é o script principal desta fase, ele tem a finalidade de controlar todos os objetos e interações que acontecem no cenário.

Figura 77- Hierarchy cena 05Jogo01.

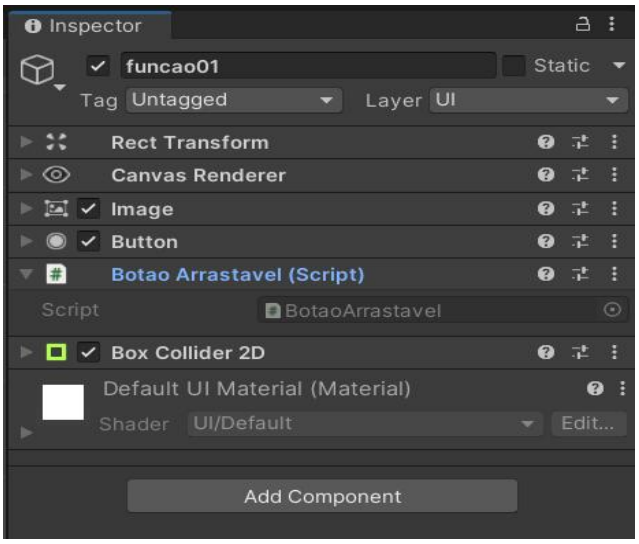


Figura 78- Conteúdo do objeto Função.

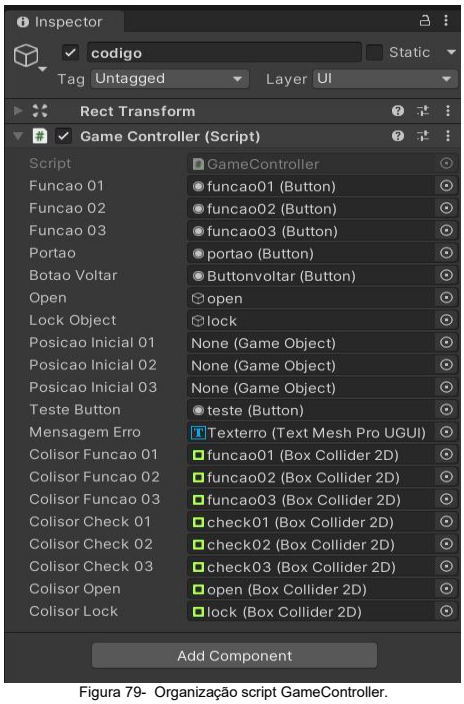


Figura 79- Organização script GameController.

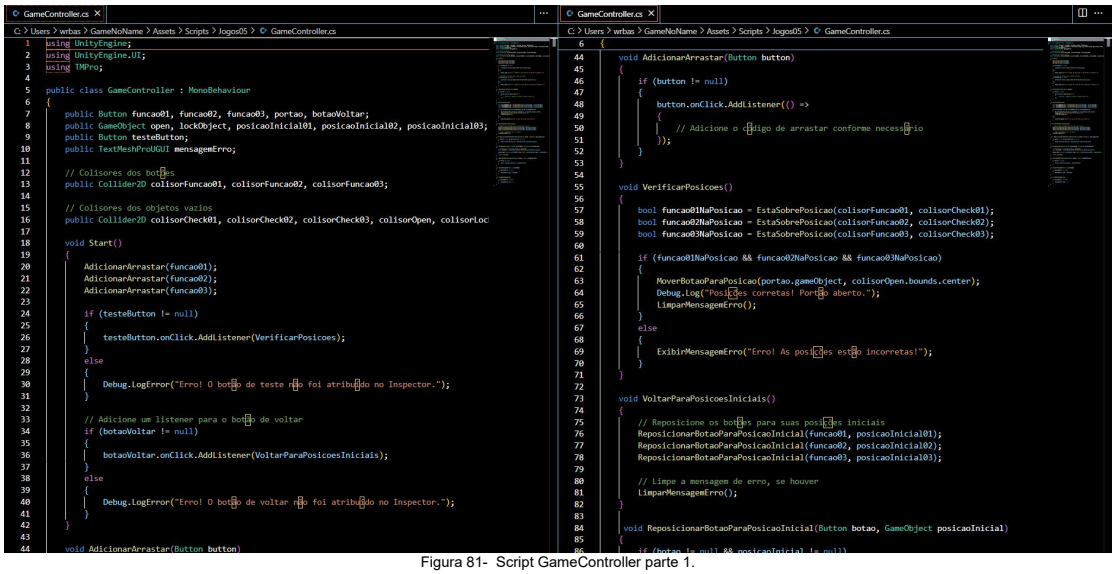
Este script mostrado na figura 80 tem uma função bem simples mas essencial para o bom funcionamento da lógica, ele irá permitir que alguns objetos que usarem ele sejam arrastados para qualquer parte da tela, nele é criado uma lógica com if/else para para permitir que o objeto seja arrastado ou não.

C: > Users > wrbas > GameNoName > Assets > Scripts > Jogos05 > BotaoArrastavel.cs

```
1  using UnityEngine;
2  using UnityEngine.UI;
3  using UnityEngine.EventSystems;
4
5  public class BotaoArrastavel : MonoBehaviour, IPointerDownHandler, IDragHandler, IPointerUpHandler
6  {
7      private bool arrastando = false;
8      private RectTransform botaoTransform;
9
10     public void OnPointerDown(PointerEventData eventData)
11     {
12         arrastando = true;
13         botaoTransform = GetComponent<RectTransform>();
14         botaoTransform.SetAsLastSibling(); // Para garantir que o botão seja desenhado por cima de outros objetos
15     }
16
17     public void OnDrag(PointerEventData eventData)
18     {
19         if (arrastando)
20         {
21             botaoTransform.anchoredPosition += eventData.delta;
22         }
23     }
24
25     public void OnPointerUp(PointerEventData eventData)
26     {
27         arrastando = false;
28     }
29 }
30
```

Figura 80- Script BotaoArrastavel.

As figuras 81 e 82 mostram o código do script GameController e todas as funções necessárias para o seu bom funcionamento, esse script faz a verificação das posições dos objetos e exibe mensagens de êxito ou erro conforme o desempenho do usuário, a função que vamos destacar aqui será a *verificarPosicoes()* que analisa se os objetos estão em suas posições corretas e exibe uma mensagem de erro ou acerto, a lógica é toda construida em cima da condicional if/else.



```
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class GameController : MonoBehaviour
{
    public Button funcao01, funcao02, funcao03, portao, botaoVoltar;
    public GameObject open, lockObject, posicaoInicial01, posicaoInicial02, posicaoInicial03;
    public Button testeButton;
    public TextMeshProUGUI mensagemErro;

    // Colisores dos botões
    public Collider2D colisorFuncao01, colisorFuncao02, colisorFuncao03;

    // Colisores dos objetos vazios
    public Collider2D colisorCheck01, colisorCheck02, colisorCheck03, colisorOpen, colisorLoc

    void Start()
    {
        AdicionarArrastar(funcao01);
        AdicionarArrastar(funcao02);
        AdicionarArrastar(funcao03);

        if (testeButton != null)
        {
            testeButton.onClick.AddListener(VerificarPosicoes);
        }
        else
        {
            Debug.LogError("Erro! O botão de teste não foi atribuído no Inspector.");
        }

        // Adicione um listener para o botão de voltar
        if (botaoVoltar != null)
        {
            botaoVoltar.onClick.AddListener(VoltarParaPosicoesIniciais);
        }
        else
        {
            Debug.LogError("Erro! O botão de voltar não foi atribuído no Inspector.");
        }
    }

    void AdicionarArrastar(Button button)
```

```
{
    void AdicionarArrastar(Button button)
    {
        if (button != null)
        {
            button.onClick.AddListener(() =>
            {
                // Adicione o código de arrastar conforme necessário
            });
        }
    }

    void VerificarPosicoes()
    {
        bool funcao01NaPosicao = EstaSobrePosicao(colisorFuncao01, colisorCheck01);
        bool funcao02NaPosicao = EstaSobrePosicao(colisorFuncao02, colisorCheck02);
        bool funcao03NaPosicao = EstaSobrePosicao(colisorFuncao03, colisorCheck03);

        if (funcao01NaPosicao && funcao02NaPosicao && funcao03NaPosicao)
        {
            MoverBotaoParaPosicao(portao.gameObject, colisorOpen.bounds.center);
            Debug.Log("Posições corretas! Portão aberto.");
            LimparMensagemErro();
        }
        else
        {
            ExibirMensagemErro("Erro! As posições estão incorretas!");
        }
    }

    void VoltarParaPosicoesIniciais()
    {
        // Reposicione os botões para suas posições iniciais
        ReposicionarBotaoParaPosicaoInicial(funcao01, posicaoInicial01);
        ReposicionarBotaoParaPosicaoInicial(funcao02, posicaoInicial02);
        ReposicionarBotaoParaPosicaoInicial(funcao03, posicaoInicial03);

        // Limpe a mensagem de erro, se houver
        LimparMensagemErro();
    }

    void ReposicionarBotaoParaPosicaoInicial(Button botao, GameObject posicaoInicial)
    {
        if (botao != null && posicaoInicial != null)
```

Figura 81- Script GameController parte 1.

```
        {
            botao.transform.position = posicaoInicial.transform.position;
        }
    }

    bool EstaSobrePosicao(Collider2D colisorBotao, Collider2D colisorPosicao)
    {
        // Verifica se os colisores estão se sobrepondo
        bool resultado = colisorBotao.bounds.Intersects(colisorPosicao.bounds);

        Debug.Log($"Colisor {colisorBotao.name} sobre {colisorPosicao.name}: {resultado}");

        return resultado;
    }

    void MoverBotaoParaPosicao(GameObject button, Vector3 targetPosition)
    {
        if (button != null)
        {
            button.transform.position = targetPosition;
        }
    }

    void ExibirMensagemErro(string mensagem)
    {
        if (mensagemErro != null)
        {
            mensagemErro.text = mensagem;
        }
    }

    void LimparMensagemErro()
    {
        if (mensagemErro != null)
        {
            mensagemErro.text = "";
        }
    }
}
```

Figura 82- Script GameController parte 2.

6.3.9 Fase07 Jogo01

A primeira fase deste nível é composto basicamente por inputs(entradas de texto) e textos, a figura 83 mostra a organização desses itens e sua hierarquia, já a figura 84 nos mostra todos os elementos presentes no botão responsável por executar a ação do jogo juntamente com o script TestePalavras que será responsável por receber e processar as entradas de texto.

Figura 83- Hierarquia Cena 07Jogo01.

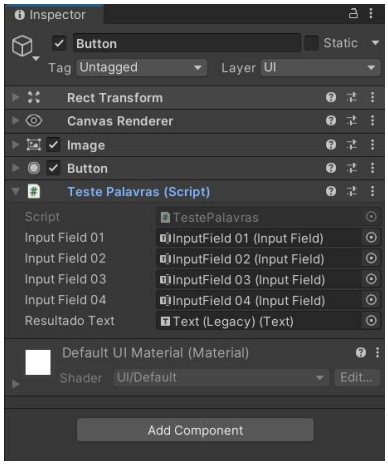


Figura 84- Organização do Script TestePalavras .

O script TestePalavras tem a lógica simples de comparar strings ele faz uma varredura pela palavra comparando letra por letra digitada e dependendo do grau de aceitabilidade, que no nosso caso e´80%, ele emite uma mensagem na tela de erro ou acerto a função *CalcularSimilaridade()* faz este trabalho *e* podemos observar tal código na figura 85.

```
using UnityEngine;
using UnityEngine.UI;

public class TestePalavras : MonoBehaviour
{
    public InputField inputField01;
    public InputField inputField02;
    public InputField inputField03;
    public InputField inputField04;
    public Text resultadoText;

    // Palavras predefinidas
    private string palavra01 = "int";
    private string palavra02 = "i";
    private string palavra03 = "if";
    private string palavra04 = "}";

    public void TestarPalavras()
    {
        // Obter as palavras digitadas pelo jogador
        string entrada01 = inputField01.text.ToLower();
        string entrada02 = inputField02.text.ToLower();
        string entrada03 = inputField03.text.ToLower();
        string entrada04 = inputField04.text.ToLower();

        // Verificar se as palavras são iguais (comparação case-insensitive)
        float semelhanca01 = CalcularSimilaridade(entrada01, palavra01);
        float semelhanca02 = CalcularSimilaridade(entrada02, palavra02);
        float semelhanca03 = CalcularSimilaridade(entrada03, palavra03);
        float semelhanca04 = CalcularSimilaridade(entrada04, palavra04);

        // Verificar se a média de semelhança é maior ou igual a 80%
        float mediaSemelhanca = (semelhanca01 + semelhanca02 + semelhanca03 + semelhanca04) /

        // Exibir mensagens com base na média de semelhança
        if (mediaSemelhanca >= 0.8f)
        {
            resultadoText.text = "Parabéns! Palavras corretas.";
        }
        else
        {
            resultadoText.text = "Erro! Tente novamente.";
        }
    }
    {
    }

    private float CalcularSimilaridade(string palavraDigitada, string palavraPredefinida)
    {
        int comprimentoMinimo = Mathf.Min(palavraDigitada.Length, palavraPredefinida.Length);
        int caracteresIguais = 0;

        for (int i = 0; i < comprimentoMinimo; i++)
        {
            if (palavraDigitada[i] == palavraPredefinida[i])
            {
                caracteresIguais++;
            }
        }

        return (float)caracteresIguais / comprimentoMinimo;
    }
}
```

Figura 85- Script TestePalavras.

7 Testes

7.1 Testes Funcionais

Objetivo: Validar que o nosso software atenda aos requisitos funcionais especificados, garantindo que cada funcionalidade funcione conforme o planejado inicialmente por nós.

Procedimento: Testar funcionalidades específicas do sistema em relação aos critérios de aceitação definidos. Nos testes específicos utilizam,os o emulador do Unity para avaliar tais critérios

Para demonstrarmos os resultados utilizaremos as tabelas de requisitos que usamos no capitulo 4 fazendo apenas uma pequena alteração adicionando uma coluna a mais chamada de resultado onde nela colocaremos se o requisito foi aprovado ou não.

Tabela de Testes Funcionais

ID	Descrição	Critério de Aceitação	Esforço (Fibonacci)	Prioridade (MoSCoW)	Resultado
TF01	Interface intuitiva e acessível	Usuário pode navegar pelo aplicativo sem dificuldades e encontrar facilmente as funcionalidades	5	Must Have	Passou
TF02	Módulos educacionais interativos	Módulos apresentam conteúdo interativo, incluindo quizzes e exercícios práticos	8	Must Have	Passou
TF03	Feedback imediato e personalizado	Sistema fornece feedback instantâneo e personalizado após cada atividade	8	Should Have	Passou
TF04	Sistema de recompensas e gamificação	Usuário recebe recompensas e incentivos através de um sistema de gamificação	5	Could Have	Passou
TF05	Acessibilidade para necessidades especiais	Sistema é acessível para usuários com deficiências, incluindo opções de leitura em voz alta e contrastes de cores	13	Must Have	Pendente

Tabela 6 - Testes Funcionais.

7.2 Testes Não Funcionais

Objetivo: Avaliar de forma qualitativa o desempenho do sistema em áreas como usabilidade, desempenho, segurança, entre outros, que não estão diretamente ligados às funcionalidades específicas.

Procedimento: Realizar testes em aspectos não funcionais do sistema para garantir que ele atenda aos requisitos de qualidade estabelecidos.

Tabela de Testes Não Funcionais

ID	Descrição	Critério de Aceitação	Esforço (Fibonacci)	Prioridade (MoSCoW)	Resultado
TNF01	Desempenho	Sistema deve responder a todas as solicitações em menos de 2 segundos	8	Must Have	Passou
TNF02	Segurança	Sistema deve garantir a proteção de dados dos usuários e prevenir acessos não autorizados	13	Must Have	Passou
TNF03	Escalabilidade	Sistema deve suportar um grande número de usuários simultâneos sem degradação significativa	13	Should Have	Passou
TNF04	Manutenibilidade	Código deve ser documentado e estruturado para facilitar manutenção e atualização	8	Should Have	Passou
TNF05	Usabilidade	Sistema deve ser intuitivo e fácil de usar para todos os usuários	8	Must Have	Passou

Tabela 7 - Testes não Funcionais.

Os testes funcionais e não funcionais realizados nos mostraram que o nosso aplicativo educacional atende bem aos requisitos especificados, tanto em termos de funcionalidades quanto de qualidade e desempenho. As funcionalidades principais , como aa interface intuitiva, módulos educacionais interativos, e o sistema de gamificação, foram validadas com êxito. De igual modo, os testes não funcionais garantiram que o sistema opere eficientemente, mantém a segurança dos dados, e pode ser escalado para acomodar um número crescente dee usuários. Essas validações são necessárias para garantir uma experiência de usuário agradável e o sucesso contínuo do aplicativo.

8. Experimentação e Resultados

Qualquer projeto de desenvolvimento de software educacional deve passar pela fase de experimentação e resultados, especialmente em um ambiente tão específico e desafiador como o da Amazônia maranhense. Esta parte do processo inclui a implementação prática do aplicativo desenvolvido. Em seguida, é realizada uma avaliação de seu impacto e eficácia. As hipóteses desenvolvidas durante as fases de planejamento e desenvolvimento devem ser validadas por meio da experimentação; os resultados da experimentação fornecem informações úteis sobre como melhorar continuamente e como justificar um projeto.

8.1 Divulgação da Experiência

A experiência foi divulgada por meio de várias atividades com o objetivo de apresentar o aplicativo educacional às comunidades alvo na Amazônia maranhense e coletar feedback inicial dos usuários. Isso exigirá colaboração de escolas locais, organizações não governamentais e líderes comunitários. A campanha de publicidade será organizada em várias fases:

Workshops e apresentações: Para demonstrar o uso do aplicativo, workshops serão realizados nas comunidades locais. Demonstrações práticas, sessões de perguntas e respostas e atividades interativas estão planejadas para envolver o público nessas sessões.

Material Promocional: Preparamos uma apostila, vídeos explicativos e cartazes para distribuir nas comunidades para aumentar a conscientização sobre o aplicativo e seus benefícios educacionais.

Redes Sociais e Mídia Local: Para divulgar o projeto, usaremos as plataformas de mídia local e redes sociais. Artigos em jornais locais, entrevistas em rádios comunitárias e publicações em redes sociais ajudarão a atingir um público maior.

Feedback inicial: Vamos coletar feedback dos participantes durante as apresentações e workshops por meio de entrevistas e questionários. Isso será muito importante para entender as primeiras impressões e descobrir áreas de melhoria.

8.2 Primeira Fase de Simulações

Nosso objetivo principal na primeira fase das simulações será testar a funcionalidade básica do aplicativo e determinar sua usabilidade para usuários iniciais. Um grupo piloto composto por cerca de 15 indivíduos de várias comunidades na Amazônia maranhense será usado para conduzir essa fase em um ambiente controlado.

Configuração do Teste: Os testes serão realizados em parte no ambiente da comunidade e em parte nos laboratórios de informática do IFMA. Na primeira área, os alunos terão acesso ao aplicativo, tanto em computadores no laboratório quanto pela pessoa que está ministrando a aula no local.

Metodologia: A metodologia de observação direta será usada para incentivar os alunos a usar o aplicativo registrando problemas e comportamentos de uso. Além disso, questionários estruturados serão usados para coletar as opiniões dos usuários sobre a experiência.

Resultados iniciais: Os resultados devem demonstrar como a maioria dos alunos achou o aplicativo simples e fácil de usar. Mas sabemos que alguns problemas técnicos, como erros de software e falhas de usabilidade, serão encontrados e corrigidos antes da próxima fase de simulações.

8.3 Segunda Fase de Simulações

O objetivo da segunda fase de simulações será avaliar a eficácia do aplicativo na aprendizagem de programação e no desenvolvimento de habilidades cognitivas por um período de tempo mais longo. Aproximadamente cem participantes estão participando desta fase.

Avaliação de Desempenho: O progresso dos alunos em programação e pensamento lógico será avaliado por meio de avaliações pré e pós-teste. Além disso, sessões de feedback com os colaboradores ajudarão a

determinar quanto o aplicativo funciona em um ambiente de aprendizagem real.

Intervenções Adaptativas: O conteúdo e a abordagem pedagógica do aplicativo serão modificados para atender às necessidades dos alunos com base nos resultados iniciais. Ao serem integradas ao aplicativo, as ferramentas de análise de dados ajudarão a monitorar o progresso individual dos alunos e ajustar o conteúdo de acordo com suas necessidades.

8.4 Análise de Resultados

A análise de resultados se concentrará em avaliar a eficácia geral do aplicativo e descobrir melhorias no aprendizado dos alunos. Desempenho Acadêmico: Os resultados dos testes pré e pós mostram se os alunos melhoraram significativamente em sua programação e pensamento lógico. Elaboraremos gráficos de desempenho usando uma média. Engajamento e Satisfação: Os dados coletados dos questionários revelarão quantos alunos estão engajados e satisfeitos. Aumentaremos o número de alunos que disseram que o aplicativo tornou o aprendizado de programação mais atraente e fácil, e diminuiremos o número de alunos que disseram que o aplicativo n não melhorou seu aprendizado.

Feedback Qualitativo: Avaliações realizadas com alunos e colaboradores fornecerão informações úteis sobre a experiência de uso do aplicativo. Levaremos em consideração as opiniões de muitos sobre o quão importantes são a interface intuitiva e as funcionalidades interativas para facilitar o aprendizado.

Impacto na Comunidade: Esperamos que a adoção do aplicativo tenha um efeito positivo na comunidade escolar, pois motivará os professores a usar tecnologias digitais em suas aulas e despertará o interesse dos alunos em carreiras tecnológicas.

8.4.1 Gráficos de comparação relativos aos questionários

Nesta atual fase encontramos algumas dificuldades para implementar o projeto, a priori iríamos aplicar o projeto em uma aldeia indígena aqui da nossa região(Aldeia Genuária) mas a escola onde eles têm aula estava em obras, e assim permaneceu até o presente momento, ficando acordado assim que faríamos este projeto com eles apenas no próximo ano, como a apresentação será agora, entramos em contato com outra comunidade para que o projeto pudesse ser aplicado. A outra comunidade disponível trata-se de um bairro carente de um interior do Maranhão chamado Zé-Doca.

O publico desta comunidade não era tão grande como o da Aldeia Genuária, pois lá teríamos um publico de aproximadamente 50 crianças e jovens, já nesta comunidade tivemos uma baixa significativa, trabalhamos com um público que girava em torno de 30 crianças. Na oportunidade tivemos um total de 10 encontros que aconteceram em uma média de 2 encontros por semana. Elaboramos questionários relacionados a cada tema abordados com eles e a intenção era avaliar se havia evolução na aprendizagem e se os conceitos estavam ou não sendo abordados, a estratégia que utilizamos foi aplicar questionários no inícios dos encontros para medir o nível de conhecimento que eles já tinham e no encerramento dos encontros aplicar questionário com o nível semelhante aos iniciais mas com questões diferentes para medir o nível de absorção do conhecimento. Infelizmente não conseguimos aplicar o questionário com as 30 crianças pois nem todas sabiam ler/escrever, mesmo já estando na faixa etária capaz para tal, as crianças tinham entre 10 e 13 anos. Logo abaixo demonstraremos os gráficos comparativos entre os questionários aplicados nesse intervalo de tempo, a seguir serão mostrados a comparação entre dois questionários, a esquerda os questionários aplicados no inícios do projeto, e a direita os aplicados ao fim dos encontros.

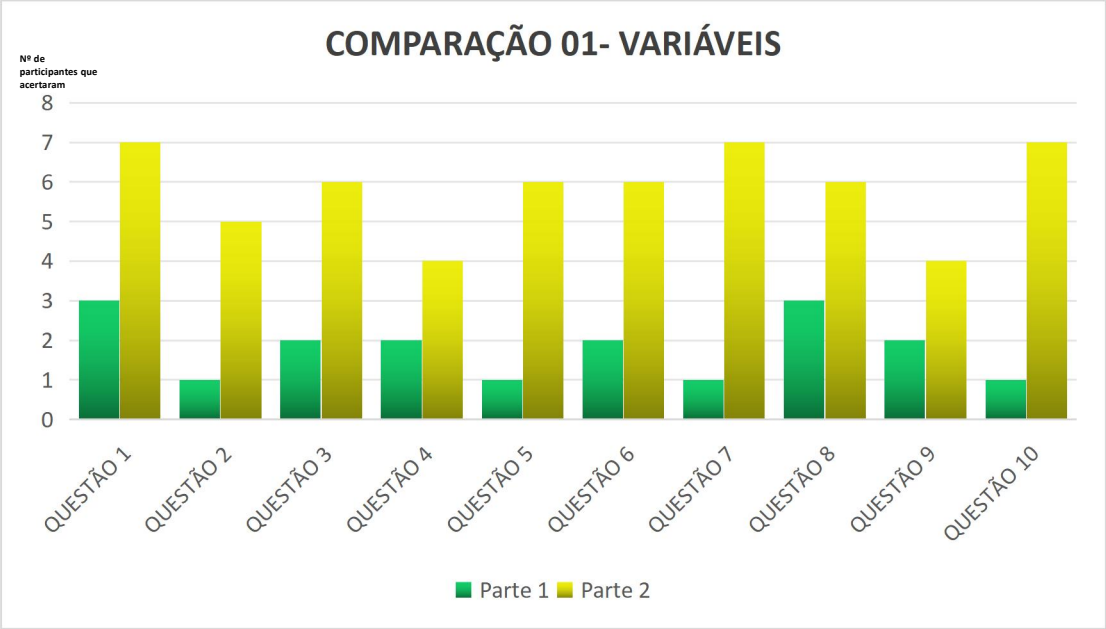


Gráfico 01 - Comparação dos questionários relacionados ao tema Variáveis.

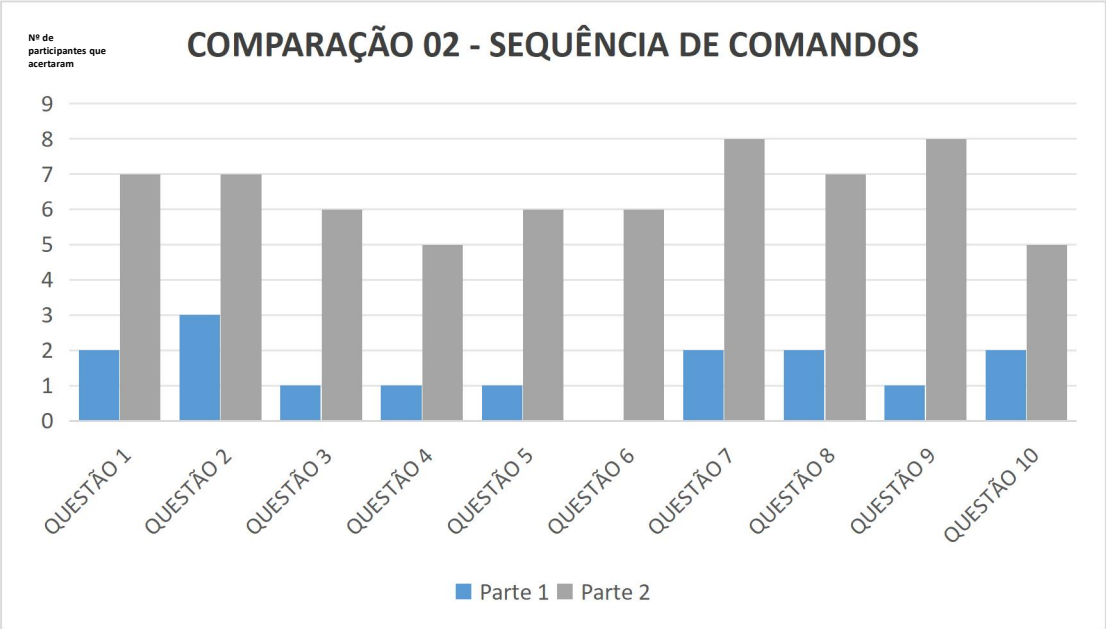


Gráfico 02 - Comparação dos questionários relacionados ao tema Sequência de comandos.

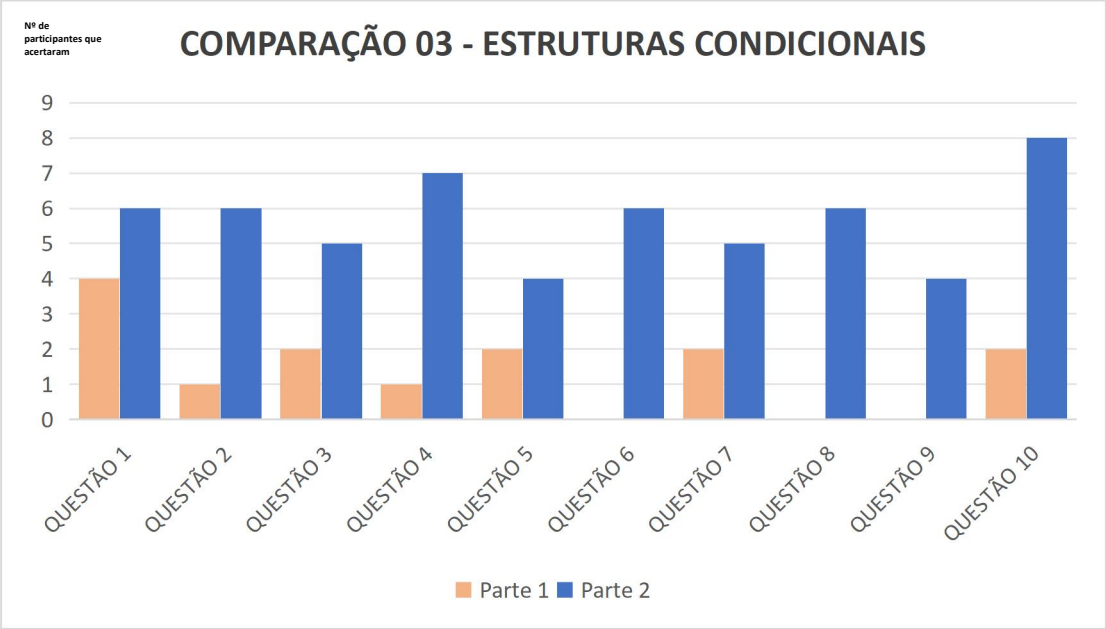


Gráfico 03 - Comparação dos questionários relacionados ao tema Estruturas Condicionais.

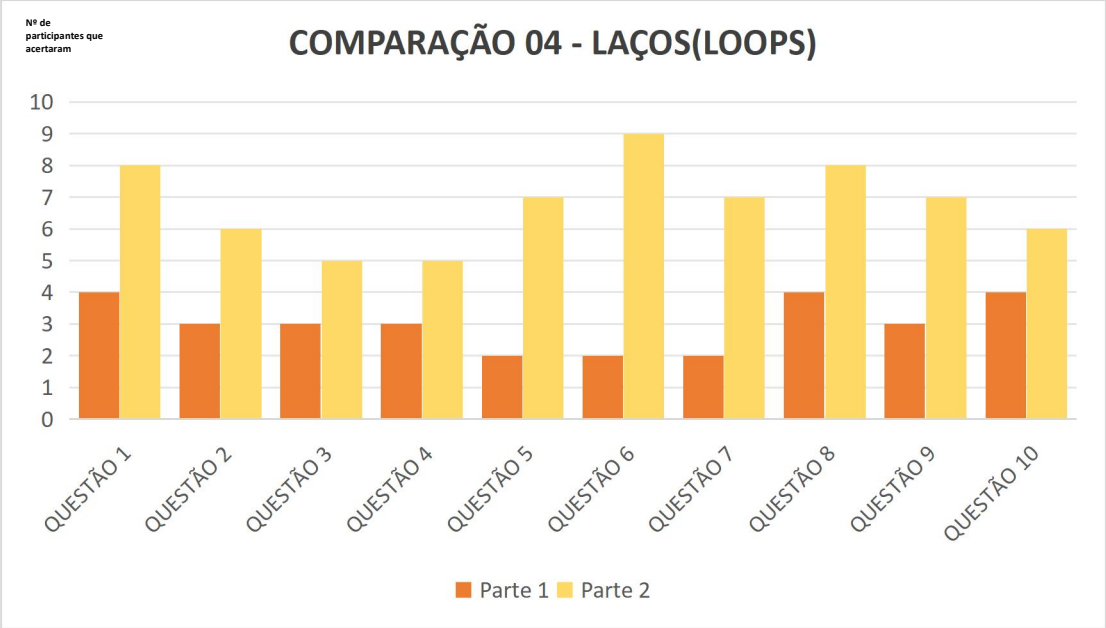


Gráfico 04 - Comparação dos questionários relacionados ao tema Laços(Loops).

9 Conclusão

O desenvolvimento de *software* educacional para auxiliar na aprendizagem do pensamento computacional em comunidades em situação de vulnerabilidade social na Amazônia Legal Maranhense é uma medida crucial para reduzir as disparidades educacionais e ampliar as oportunidades de aprendizagem para essa população. Ao adaptar abordagens existentes e considerar as necessidades específicas da região, é possível criar soluções eficazes e sustentáveis que promovam o desenvolvimento cognitivo e técnico dos indivíduos.

Neste contexto, é fundamental reconhecer a importância de abordagens inclusivas e culturalmente relevantes no desenvolvimento de *software* educacional. Como destacado por *Barton e Tan* (2018), a adaptação de ferramentas educacionais para refletir as experiências e valores das comunidades locais é essencial para garantir sua eficácia e aceitação. Além disso, parcerias com organizações locais e líderes comunitários, conforme sugerido por *Vargas et al.* (2021), podem facilitar a implementação e sustentabilidade desses projetos.

É necessário também investir em pesquisa adicional para entender melhor as necessidades específicas das comunidades amazônicas e desenvolver soluções mais eficazes e personalizadas. A pesquisa de *Papert* (1980) sobre o uso de linguagens de programação na educação demonstrou os benefícios de uma abordagem construtivista para o ensino de computação, destacando a importância de promover a autonomia e a criatividade dos alunos.

Além disso, é importante destacar as contribuições de iniciativas internacionais, como o projeto *Code.org* nos Estados Unidos e o programa *BBC Micro:bit* no Reino Unido. Esses programas oferecem recursos e currículos gratuitos para promover a educação em ciência da computação e a programação entre os jovens em todo o mundo, demonstrando o potencial transformador da tecnologia educacional.

Em suma, o desenvolvimento de *software* educacional para promover o pensamento computacional em comunidades em situação de vulnerabilidade social na Amazônia Legal Maranhense é um desafio complexo, mas essencial. Ao colaborar com as partes interessadas locais, investir em

pesquisa e adaptação cultural e aproveitar as lições aprendidas com iniciativas globais, podemos criar soluções mais eficazes e inclusivas que capacitam os indivíduos e transformam comunidades inteiras.

Referências

Applications/Pages/Digital-Inclusion.aspx [Acesso em 16 de fevereiro de 2024].

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? ACM Inroads, 2(1), 48-54.

Barton, K. C., & Tan, E. (2018). Culturally Responsive Computing: A Theory Revisited. Journal of the Learning Sciences, 27(3), 379–409.

Carnegie Mellon University. (2020). Alice. Recuperado de https://www.alice.org/.[Acesso em 20 de fevereiro de 2024].

Cruz, F., et al. (2019). A importância da tecnologia educacional para a inclusão social. Revista Thema, 16(1), 227-240.

European Commission. (2021). Digital Education Action Plan. [Online] Disponível em: https://ec.europa.eu/education/education-in-the-eu/digital-education-action-plan_en [Acesso em 16 de fevereiro de 2024].

European Commission. (2021). Digital Inclusion Strategies for Education. Recuperado de https://ec.europa.eu/digital-single-market/en/digital-inclusion-education

Gee, J. (2007). Good Video Games and Good Learning: Collected Essays on Video Games, Learning, and Literacy. Peter Lang Publishing.

Gee, J. P. (2007). What Video Games Have to Teach Us About Learning and Literacy. Macmillan.

Google Developers. (2021). Blockly. Recuperado de https://developers.google.com/blockly.

Gupta, R. K., Sharma, S. K., & Kaushal, R. (2017). Role of ICT in Rural Development. International Journal of Computer Applications, 160(5), 1-5.

Gupta, R., et al. (2017). Impact of Digital Technologies on Rural Communities: Case Studies from Developing Countries. International Journal of Information Management, 37(5), 450-459.

International Telecommunication Union (ITU). (2020). Digital Inclusion. [Online] Disponível em: https://www.itu.int/en/ITU-D/ICT-

ITU. (2020). Digital Technologies for Inclusive Education. Recuperado de https://www.itu.int/en/ITU-D/ICT-Applications/Pages/Inclusive-Education.aspx.

Jackson, A. (2018). Digital Infrastructure in Rural Areas: Challenges and Opportunities. Journal of Rural Studies, 64, 182-195.

Johnson, L. (2019). The Transformative Role of Technology in Education. Publisher.

Jonassen, D. (2000). Computers as Mindtools for Schools: Engaging Critical Thinking. Prentice Hall.

Jonassen, D. H. (2000). Toward a Design Theory of Problem Solving. Educational Technology Research and Development, 48(4), 63-85.

Kafai, Y. B., & Burke, Q. (2015). Connected Code: Why Children Need to Learn Programming. MIT Press.

Kafai, Y., & Burke, Q. (2015). Connected Code: Why Children Need to Learn Programming. MIT Press.

learning-for-the-21st-century. [Acesso em 20 de fevereiro de 2024].

Mann, D., Kumar, V., & Singh, A. (2020). Localized Educational Technology Solutions for Marginalized Communities: Challenges and Opportunities. International Journal of Educational Technology in Higher Education, 17(1), 1-23.

Mann, J., et al. (2020). Leveraging Educational Technology for Development: Lessons from the Field. Journal of Education for Sustainable Development, 14(1), 50-65.

Microsoft Research. (2018). Kodu. Recuperado de https://www.microsoft.com/en-us/research/project/kodu/. Acesso em 17 de fevereiro de 2024].

Nussbaum, M. (2010). Not for Profit: Why Democracy Needs the Humanities. Princeton University Press.

OECD. (2021). "Skills for the 21st Century: What Should Students Learn?" OECD Publishing.

OECD. (2021). Digital Education for All: A Blueprint for Inclusion. Recuperado de https://www.oecd.org/education/digital-education-for-all-a-blueprint-for-inclusion.html.

OECD. (2021). Digital Inclusion in Education. Recuperado de https://www.oecd.org/education/digital-inclusion-in-education.html.

OECD. (2021). Ensuring Equitable Access to Digital Education. Recuperado de https://www.oecd.org/education/digital-education/ensuring-equitable-access-to-digital-education.htm.

Oliveira, R. (2019). Strategic Partnerships for Quality Education in the Legal Amazon: Perspectives and Challenges. Journal of Amazonian Development, 5(1), 78-91.

Papert, S. (1993). "Mindstorms: Children, Computers, and Powerful Ideas." Basic Books.

PAPERT, Seymour. A Máquina das Crianças: Repensando a escola na era da informática. Porto Alegre: Artmed, 1994.

Resnick, M., et al. (2009). Scratch: Programming for All. Communications of the ACM, 52(11), 60-67.

Santos, C. (2020). Customized Educational Programs for Amazonian Communities: Addressing Challenges of Exclusion. Amazonian Education Journal, 12(2), 45-58.

Schofield, D. (2018). Digital Education in Remote Communities: Challenges and Opportunities. Academic Press.

Schofield, D. (2018). Digital Education in Remote Communities: Challenges and Opportunities. International Journal of Information and Education Technology, 8(9), 640-644.

Secretaria de Educação Básica. (2017). Programaê!: Caderno do Professor. Brasília: Ministério da Educação.

Silva, A. C., & Carvalho, C. V. (2020). Tecnologias digitais na educação: exclusão e vulnerabilidade social. Revista Brasileira de Informática na Educação, 28(3), 1-10.

Smith, A. B., & Jones, C. D. (2020). Desenvolvimento de software educacional adaptado: Uma abordagem essencial para comunidades vulneráveis. Journal of Educational Technology, 8(3), 45-60.

UNESCO. (2019). Education for the Most Marginalized: Access to Information and Communication Technology (ICT) in Brazilian Indigenous Communities.

UNESCO. (2020). Education for Sustainable Development Goals: Learning Objectives.

Recuperado de https://unesdoc.unesco.org/ark:/48223/pf0000374667.

UNESCO. (2020). Lifelong Learning for Sustainable Development. Recuperado de https://unesdoc.unesco.org/ark:/48223/pf0000374667.

UNICEF. (2020). Education for Every Child: The Right to Education. Recuperado de https://www.unicef.org/education/right-to-education [Acesso em 18 de fevereiro de 2024].

Vargas, E. J., et al. (2021). Promovendo o acesso à educação digital em comunidades vulneráveis: um estudo de caso na região amazônica. Anais do Congresso Brasileiro de Informática na Educação.

Vygotsky, L. (1978). Mind in Society: The Development of Higher Psychological Processes. Harvard University Press.

Wang, S., et al. (2016). Computational Thinking in Education: Where Does it Fit? A Systematic Literary Review. In: International Conference on Technology in Education (ICTE 2016).

Wang, S., Wang, H., & Wang, H. (2016). Computational Thinking in Education: A Review. Springer International Publishing.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. Journal of Science Education and Technology, 25(1), 127–147.

Wing, J. M. (2006). Computational Thinking. Communications of the ACM, 49(3), 33–35.

World Bank. (2018). Digital Dividends: Sustainable Development Goals and Digital Technologies in Rural Areas.

World Economic Forum. (2021). The Future of Education: Transforming Learning for the 21st Century. Recuperado de https://www.weforum.org/reports/the-future-of-education-transforming

World Economic Forum. (2021). The Future of Jobs Report. [Online] Disponível em: https://www.weforum.org/reports/the-future-of-jobs-report-2021 [Acesso em 16 de fevereiro de 2024].

Anexo A - Questionários

01 Variáveis

Primeira Parte (Antes da Aula)

1- O que é uma variável?

a) Um tipo de comida
b) Um lugar para guardar informações
c) Um jogo de tabuleiro
d) Um nome de pessoa

2- Qual dos seguintes itens pode ser guardado em uma variável?

a) Um número, como 10
b) Uma palavra, como "gato"
c) Um valor como "verdadeiro" ou "falso"
d) Todos os anteriores

3- Por que usamos variáveis em programação?

a) Para lembrar o nome dos nossos amigos
b) Para guardar informações que podemos usar depois
c) Para desenhar figuras
d) Para mudar a cor da tela

4- Como é o nome de uma variável?

a) Um número
b) Uma cor
c) Uma palavra que escolhemos
d) Um símbolo

5- Se você quiser lembrar a idade de alguém, o que você pode usar?

a) Uma caneta
b) Um papel
c) Uma variável
d) Um lápis

6- O que acontece quando você coloca um valor em uma variável?

a) A variável guarda esse valor
b) O valor desaparece
c) Nada acontece
d) A variável explode

7- Uma variável pode guardar...

a) Apenas números
b) Apenas palavras
c) Números e palavras
d) Apenas figuras

8- Qual seria um bom nome para uma variável que guarda a altura de uma pessoa?

a) animal
b) cor_favorita
c) altura
d) casa

9- O que você pode fazer com uma variável depois que ela guarda um valor?

a) Jogar fora
b) Usar esse valor mais tarde
c) Esconder a variável
d) Desenhar com ela

10- Se uma variável guarda o nome de uma pessoa, como podemos chamá-la?

a) Uma caixa de números
b) Uma caixa de palavras
c) Uma caixa de figuras
d) Uma caixa de cores

GABARITO

1-Resposta correta: b) Um lugar para guardar informações
2-Resposta correta: d) Todos os anteriores
3-Resposta correta: b) Para guardar informações que podemos usar depois
4-Resposta correta: c) Uma palavra que escolhemos
5-Resposta correta: c) Uma variável
6-Resposta correta: a) A variável guarda esse valor
7-Resposta correta: c) Números e palavras
8-Resposta correta: c) altura
9-Resposta correta: b) Usar esse valor mais tarde
10-Resposta correta: b) Uma caixa de palavras

02 Sequência de comandos

Primeira Parte (Antes da Aula)

1-O que é uma sequência de comandos?

a) Uma lista de compras
b) Uma ordem de instruções para o computador seguir
c) Um tipo de música
d) Uma receita de bolo

2-Por que é importante seguir uma sequência de comandos corretamente?

a) Para o computador entender o que fazer
b) Para o computador ficar feliz
c) Para deixar o computador bonito
d) Para ganhar pontos

3-Qual das opções abaixo é um exemplo de sequência de comandos?

a) Acordar, escovar os dentes, tomar café
b) Contar até 10
c) Fazer uma soma de números
d) Todas as anteriores

4-O que acontece se um comando estiver fora de ordem?

a) Nada muda
b) O computador pode fazer algo errado
c) O computador faz tudo mais rápido
d) O computador muda de cor

5-Como podemos garantir que uma sequência de comandos funcione bem?

a) Colocando os comandos na ordem certa
b) Colocando os comandos em qualquer ordem
c) Deixando alguns comandos de fora
d) Usando apenas um comando

6-Qual das seguintes atividades é parecida com uma sequência de comandos?

a) Seguir uma receita de bolo
b) Correr em um campo
c) Desenhar um círculo
d) Pintar uma parede

7-O que podemos fazer se a sequência de comandos não funcionar?

a) Verificar a ordem dos comandos
b) Ignorar o problema

c) Desligar o computador
d) Mudar o nome dos comandos

8-Se queremos que o computador faça uma tarefa específica, o que precisamos dar a ele?

a) Um desenho
b) Uma sequência de comandos
c) Um filme
d) Uma música

9-O que acontece se deixarmos um comando importante de fora da sequência?

a) O computador faz tudo corretamente
b) O computador pode não entender o que fazer
c) O computador vai adivinhar o comando
d) O computador vai ficar mais rápido

10-Qual é o resultado de uma sequência de comandos bem executada?

a) O computador faz o que esperamos
b) O computador trava
c) O computador faz outra coisa
d) O computador apaga tudo

GABARITO

1-Resposta correta: b) Uma ordem de instruções para o computador seguir
2-Resposta correta: a) Para o computador entender o que fazer
3-Resposta correta: d) Todas as anteriores
4-Resposta correta: b) O computador pode fazer algo errado
5-Resposta correta: a) Colocando os comandos na ordem certa
6-Resposta correta: a) Seguir uma receita de bolo
7-Resposta correta: a) Verificar a ordem dos comandos
8-Resposta correta: b) Uma sequência de comandos
9-Resposta correta: b) O computador pode não entender o que fazer
10-Resposta correta: a) O computador faz o que esperamos

03 Estruturas condicionais

Primeira Parte (Antes da Aula)

1-O que é uma estrutura condicional?

a) Um comando que permite escolher entre duas ou mais opções
b) Um tipo de animal que vive na floresta

c) Uma cor para pintar a tela do computador
d) Um desenho animado famoso

2-Quando usamos uma estrutura condicional?

a) Quando queremos fazer um desenho
b) Quando precisamos tomar uma decisão baseada em uma condição
c) Quando precisamos somar dois números
d) Quando queremos escrever uma carta

3-O que acontece se a condição em uma estrutura condicional for verdadeira?

a) Nada acontece
b) O computador ignora a condição
c) O computador executa o comando ligado a essa condição
d) O computador desliga

4-Se uma condição em uma estrutura condicional for falsa, o que acontece?

a) O computador pula para o próximo comando
b) O computador repete o comando
c) O computador mostra uma mensagem de erro
d) O computador trava

5-Qual das opções a seguir é um exemplo de estrutura condicional?

a) Se estiver chovendo, use guarda-chuva; senão, não use.
b) Escrever seu nome na tela
c) Somar dois números
d) Desenhar um círculo

6-Para que serve a palavra "senão" em uma estrutura condicional?

a) Para repetir um comando
b) Para indicar o que fazer se a condição não for verdadeira
c) Para desenhar na tela
d) Para somar números

7-O que é uma condição em uma estrutura condicional?

a) Uma regra que deve ser verdadeira ou falsa
b) Um número que devemos somar
c) Um desenho que aparece na tela
d) Uma música que toca no computador

8-Qual das situações a seguir usa uma estrutura condicional?

a) Se a luz estiver acesa, desligue; se estiver apagada, ligue.
b) Escrever uma carta para um amigo
c) Desenhar uma casa
d) Contar até dez

9-Quando uma estrutura condicional não é executada?

a) Quando a condição é verdadeira
b) Quando a condição é falsa
c) Quando o computador não entende o comando
d) Quando o comando não está na sequência

10-O que precisamos fazer para uma estrutura condicional funcionar corretamente?

a) Escrever uma condição que o computador possa verificar
b) Desenhar uma figura na tela
c) Tocar uma música antes de executar o comando
d) Desligar o computador

GABARITO

1-Resposta correta: a) Um comando que permite escolher entre duas ou mais opções
2-Resposta correta: b) Quando precisamos tomar uma decisão baseada em uma condição
3-Resposta correta: c) O computador executa o comando ligado a essa condição
4-Resposta correta: a) O computador pula para o próximo comando
5-Resposta correta: a) Se estiver chovendo, use guarda-chuva; senão, não use.
6-Resposta correta: b) Para indicar o que fazer se a condição não for verdadeira
7-Resposta correta: a) Uma regra que deve ser verdadeira ou falsa
8-Resposta correta: a) Se a luz estiver acesa, desligue; se estiver apagada, ligue.
9-Resposta correta: b) Quando a condição é falsa
10-Resposta correta: a) Escrever uma condição que o computador possa verificar

04 Laços (loops)

Primeira Parte (Antes da Aula)

1-O que é um laço (loop) em programação?

a) Um comando que repete uma ação várias vezes
b) Um desenho circular na tela do computador
c) Uma música que toca em repetição
d) Um tipo de quebra-cabeça

2-Para que usamos um laço (loop)?

a) Para fazer uma ação uma única vez
b) Para repetir uma ação várias vezes
c) Para desenhar na tela
d) Para tocar uma música diferente

3-Qual é um exemplo de laço (loop)?

a) Contar de 1 a 10 várias vezes
b) Escrever uma carta
c) Desenhar um quadrado
d) Somar dois números

4-O que acontece se um laço (loop) não tiver um fim?

a) Ele continua repetindo para sempre
b) Ele para automaticamente
c) Ele desenha na tela
d) Ele toca uma música

5-Quando devemos usar um laço (loop)?

a) Quando queremos que algo se repita várias vezes
b) Quando queremos que algo aconteça uma única vez
c) Quando queremos desenhar na tela
d) Quando queremos desligar o computador

6-Qual das opções a seguir representa um laço (loop)?

a) Repetir a mesma tarefa até ela ser concluída
b) Fazer uma tarefa apenas uma vez
c) Desenhar um círculo
d) Somar dois números

7-O que devemos fazer para parar um laço (loop)?

a) Definir uma condição para ele terminar
b) Desligar o computador
c) Desenhar algo na tela
d) Tocar uma música

8-O que é um laço infinito?

a) Um laço que nunca termina
b) Um laço que termina imediatamente
c) Um laço que desenha círculos infinitos
d) Um laço que toca música infinitamente

9-Como podemos garantir que um laço (loop) funcione corretamente?

a) Definindo claramente quando ele deve parar
b) Desenhando na tela antes de começar
c) Tocando uma música antes do laço
d) Somando números antes de começar

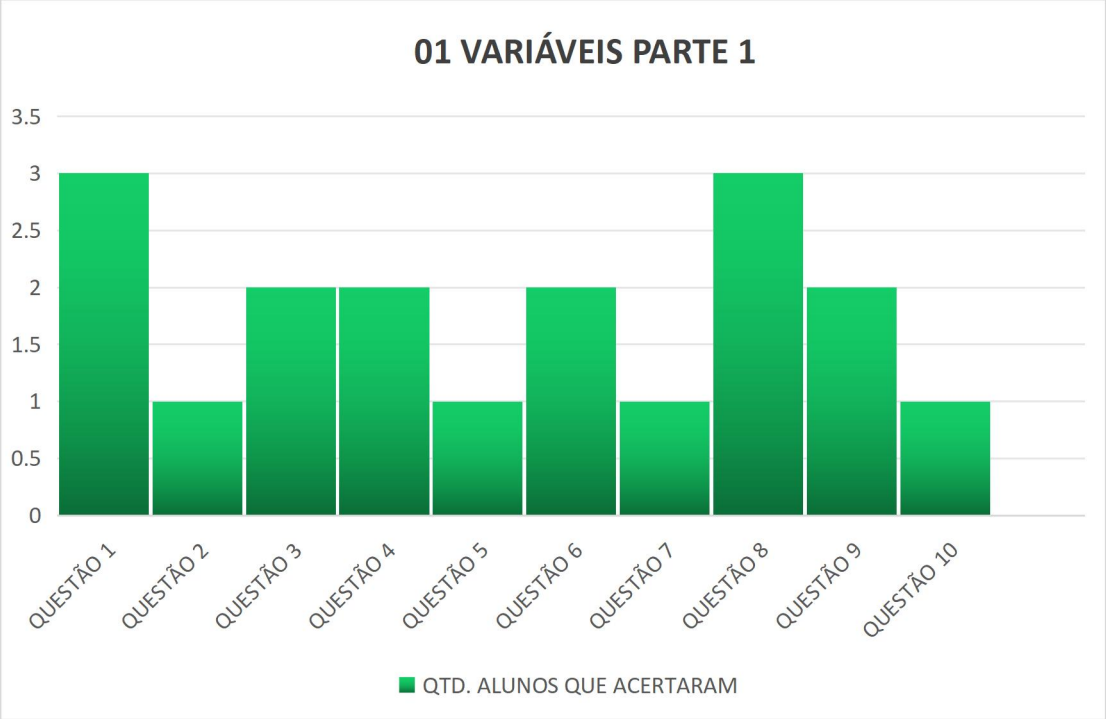
10-Por que é importante definir um fim para um laço (loop)?

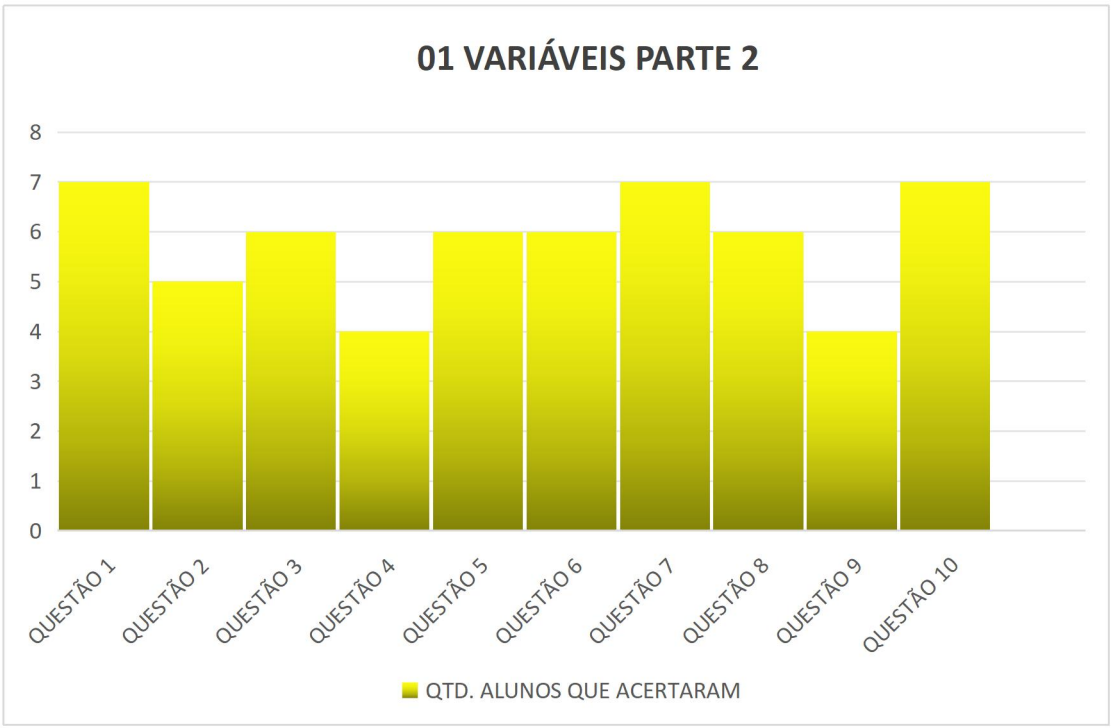
a) Para evitar que ele continue repetindo para sempre
b) Para tocar música no final
c) Para desenhar na tela
d) Para somar números no final

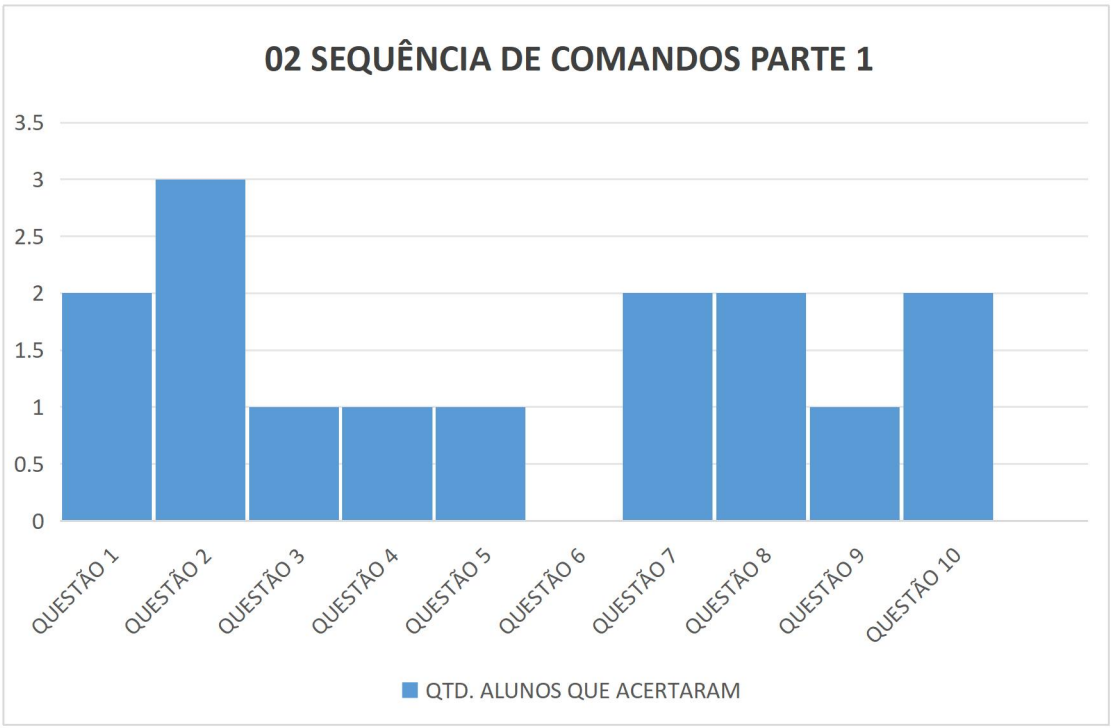
GABARITO

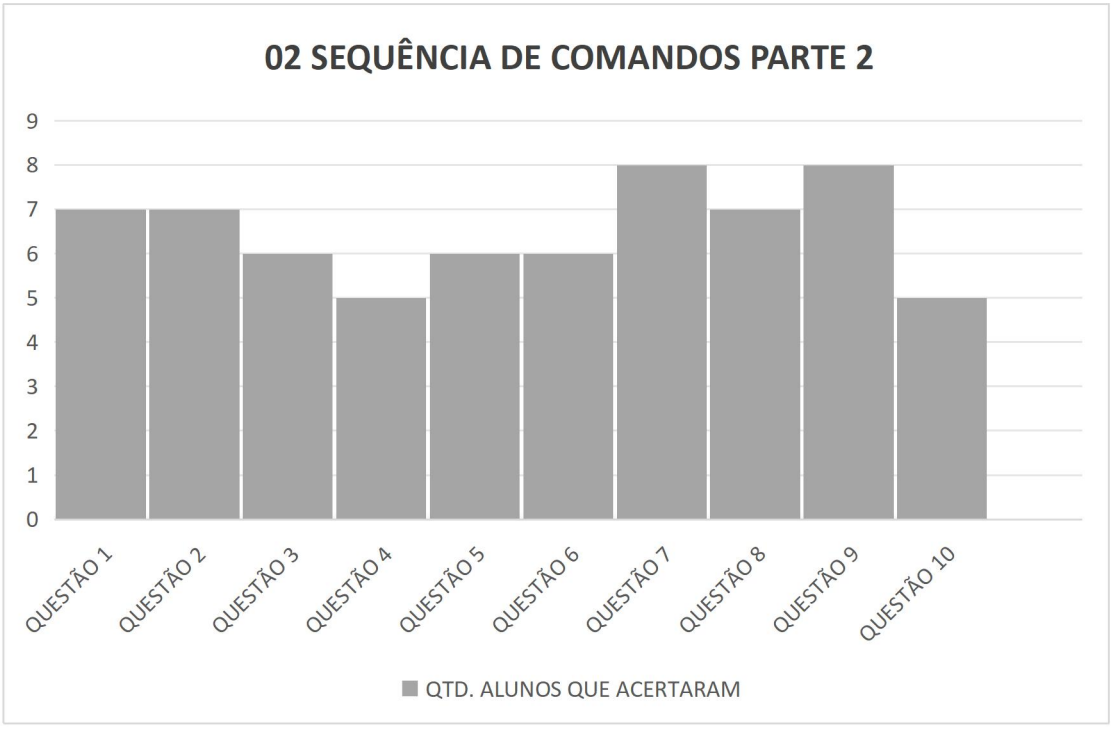
1-Resposta correta: a) Um comando que repete uma ação várias vezes
2-Resposta correta: b) Para repetir uma ação várias vezes
3-Resposta correta: a) Contar de 1 a 10 várias vezes
4-Resposta correta: a) Ele continua repetindo para sempre
5-Resposta correta: a) Quando queremos que algo se repita várias vezes
6-Resposta correta: a) Repetir a mesma tarefa até ela ser concluída
7-Resposta correta: a) Definir uma condição para ele terminar
8-Resposta correta: a) Um laço que nunca termina
9-Resposta correta: a) Definindo claramente quando ele deve parar
10-Resposta correta: a) Para evitar que ele continue repetindo para sempre

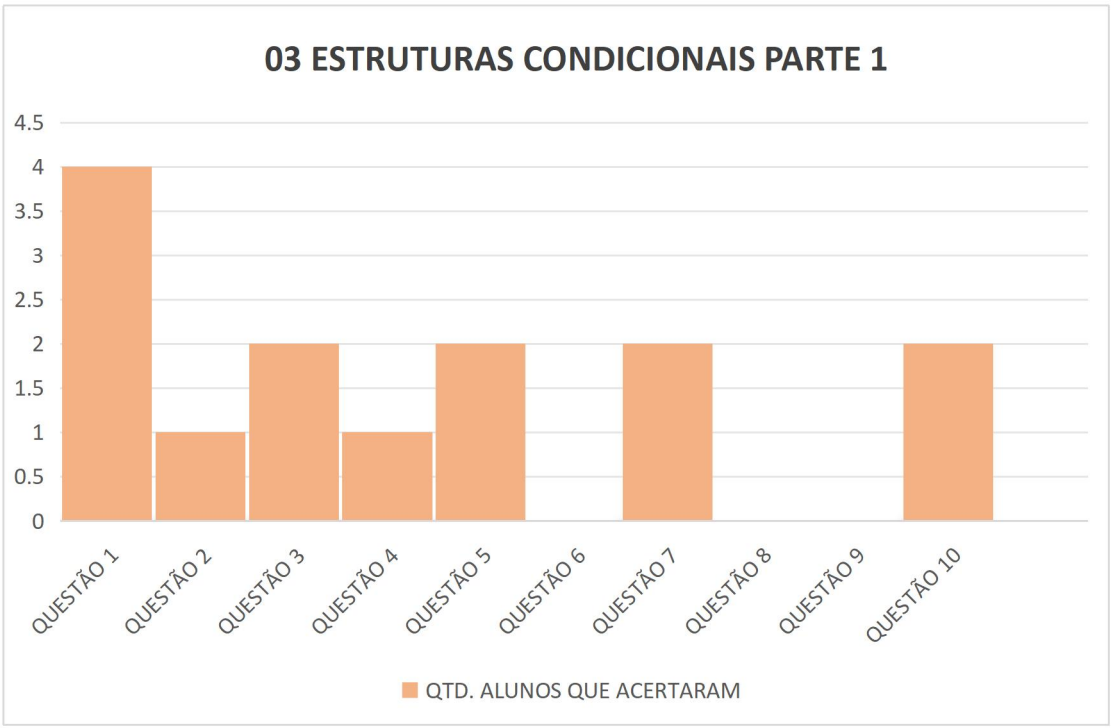
Anexo B - Resultados dos questionários

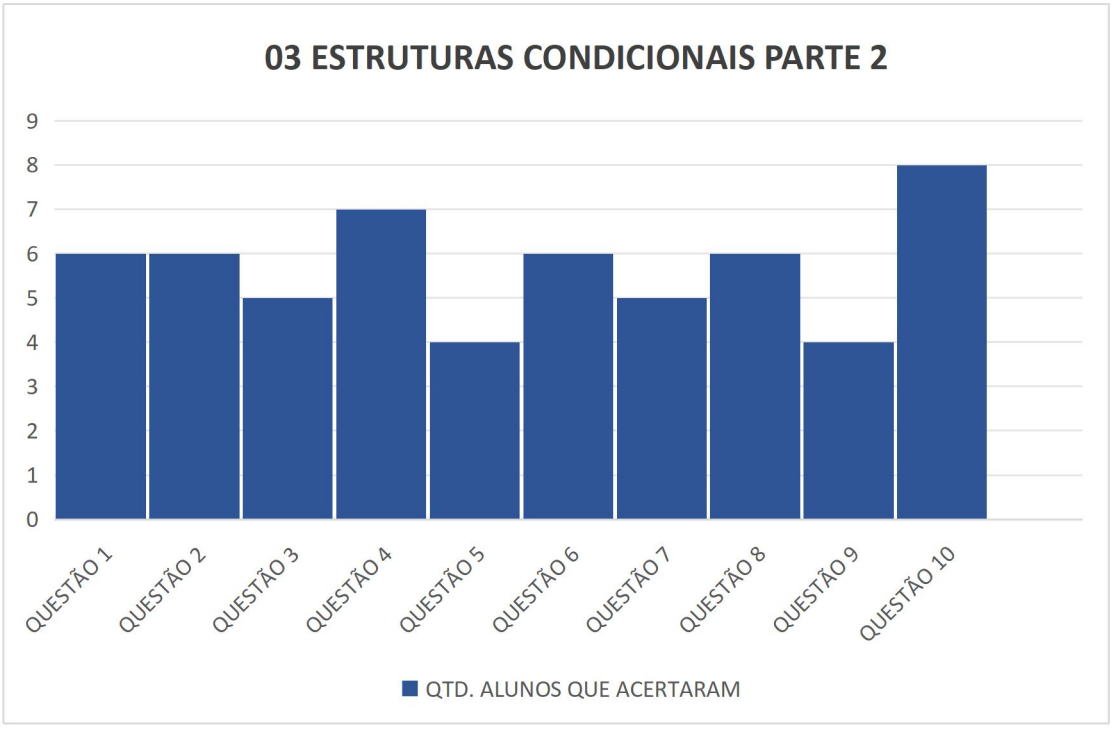


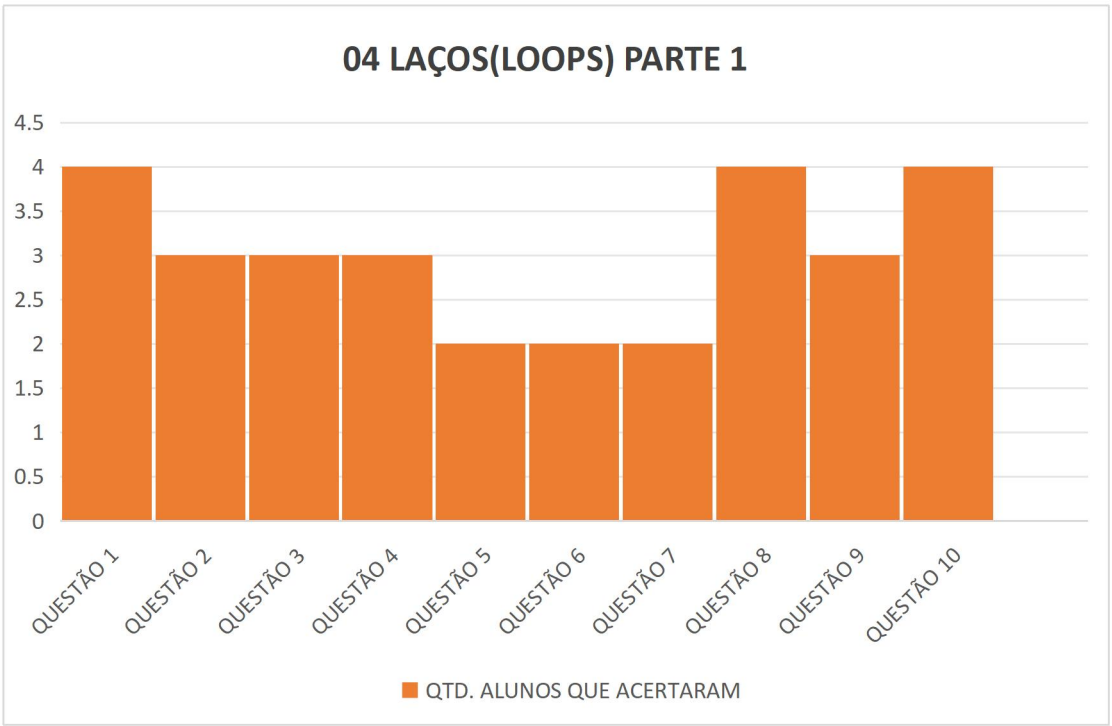

01 VARIÁVEIS PARTE 2
8
7
6
5
4
3
2
1
0
QUESTÃO 1
QUESTÃO 2
QUESTÃO 3
QUESTÃO 4
QUESTÃO 5
QUESTÃO 6
QUESTÃO 7
QUESTÃO 8
QUESTÃO 9
QUESTÃO 10
QTD. ALUNOS QUE ACERTARAM

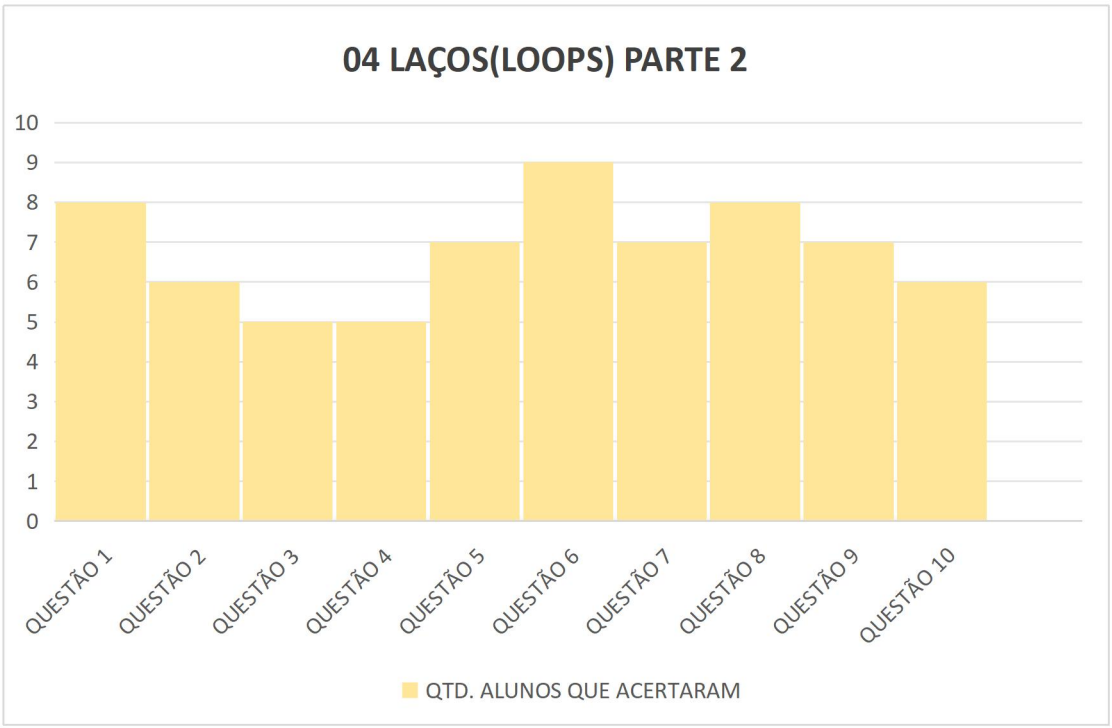

02 SEQUÊNCIA DE COMANDOS PARTE 1
3.5
3
2.5
2
1.5
1
0.5
0
QUESTÃO 1
QUESTÃO 2
QUESTÃO 3
QUESTÃO 4
QUESTÃO 5
QUESTÃO 6
QUESTÃO 7
QUESTÃO 8
QUESTÃO 9
QUESTÃO 10
QTD. ALUNOS QUE ACERTARAM


02 SEQUÊNCIA DE COMANDOS PARTE 2
9
8
7
6
5
4
3
2
1
0
QUESTÃO 1
QUESTÃO 2
QUESTÃO 3
QUESTÃO 4
QUESTÃO 5
QUESTÃO 6
QUESTÃO 7
QUESTÃO 8
QUESTÃO 9
QUESTÃO 10
QTD. ALUNOS QUE ACERTARAM


03 ESTRUTURAS CONDICIONAIS PARTE 1
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
QUESTÃO 1
QUESTÃO 2
QUESTÃO 3
QUESTÃO 4
QUESTÃO 5
QUESTÃO 6
QUESTÃO 7
QUESTÃO 8
QUESTÃO 9
QUESTÃO 10
QTD. ALUNOS QUE ACERTARAM


03 ESTRUTURAS CONDICIONAIS PARTE 2
9
8
7
6
5
4
3
2
1
0
QUESTÃO 1
QUESTÃO 2
QUESTÃO 3
QUESTÃO 4
QUESTÃO 5
QUESTÃO 6
QUESTÃO 7
QUESTÃO 8
QUESTÃO 9
QUESTÃO 10
QTD. ALUNOS QUE ACERTARAM


04 LAÇOS(LOOPS) PARTE 1
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
QUESTÃO 1
QUESTÃO 2
QUESTÃO 3
QUESTÃO 4
QUESTÃO 5
QUESTÃO 6
QUESTÃO 7
QUESTÃO 8
QUESTÃO 9
QUESTÃO 10
QTD. ALUNOS QUE ACERTARAM


04 LAÇOS(LOOPS) PARTE 2
10
9
8
7
6
5
4
3
2
1
0
QUESTÃO 1
QUESTÃO 2
QUESTÃO 3
QUESTÃO 4
QUESTÃO 5
QUESTÃO 6
QUESTÃO 7
QUESTÃO 8
QUESTÃO 9
QUESTÃO 10
QTD. ALUNOS QUE ACERTARAM