

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO  
AMAZONAS**

**TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

**Relatório Técnico - Logicus;**

**MANAUS, AM  
DEZEMBRO, 2025**

Anthony Jesus Ramos Rodrigues

Beatriz Barauna Bitar

Cauan da Silva Pinho

Ivo Antonio Ribeiro Rodrigues

Kinan Principe

## **Relatório Técnico - Logicus;**

Documentação apresentada como requisito para obtenção de nota na disciplina Algoritmos e Técnicas de Programação ministrada pelo Professor Emmerson Santa Rita da Silva.

**MANAUS, AM**  
**DEZEMBRO, 2025**

## 1. INTRODUÇÃO

*Logicus*; é um jogo que, por meio da ludicidade, incentiva o aprendizado de linguagem de programação e como aplicá-la em determinados contextos.

Quando os métodos tradicionais de ensino não cativam alunos ingressantes em cursos de tecnologia da informação o suficiente, *Logicus* visa – por meio de desafios de nível introdutório e uma narrativa descontraída – reforçar os fundamentos da programação na linguagem C e demonstrar aplicações práticas destes na resolução de problemas abstratos.

Na necessidade de fugir da metodologia bancária de ensino, Paulo Freire defende que a ludicidade não é mero passatempo, mas uma necessidade humana fundamental e ferramenta pedagógica vital para a autonomia, criatividade e consciência crítica, essencial na educação infantil e para adultos, permitindo a construção do conhecimento de forma engajada, dialógica e libertadora, não neutra. O que inspira a índole questionadora de *Logicus* de contribuir para a formação acadêmica de maneira não ortodoxa e de incentivar projetos que visem o mesmo objetivo.

## 2. DESCRIÇÃO DO SISTEMA

Através da interface gráfica é que o usuário navega. O formato escolhido para a implementação do jogo foi o de TBRPG (*Turn-Based Role-Play Game*), também conhecido como *Visual Novel* pela sua fácil implementação e alta intuitividade.

O usuário navega pelos textos que lhe são mostrados conforme se adentra no jogo, sendo introduzido à premissa e resolvendo problemas iniciais que são comuns na prática do programador; neste caso, o primeiro desafio é a declaração de bibliotecas.

O uso do jogo se assemelha bastante a um ciclo, que poderia ser entendido como sendo da seguinte forma: o jogador chega em um determinado lugar; o jogador resolve um problema de acordo com os elementos que estão presentes em tal lugar; o jogador prossegue para a seguinte sala e assim sucessivamente.

O local em que se passa o jogo é situado em uma torre, na que em seus diferentes andares possui desafios diferentes um dos outros, sendo estes:

**Input:** O jogador, tem que inserir uma resposta que atenda às exigências dadas no enunciado do desafio.

**Caixinhas:** O famigerado modelo de preenchimento de lacunas que aplicativos para celulares como o Duolingo popularizaram está presente neste jogo, cumprindo a mesma função, adequada para o contexto.

**Enigmas:** Com base em dicas sutis que são dadas ao jogador, este tem que se desdobrar para passar através de obstáculos e labirintos para chegar no seu objetivo final.

**Combate:** Como é próprio do gênero de RPG, temos um combate baseado em turnos em que o jogador pode realizar uma ação e seguidamente responder à ação do inimigo. Esta mecânica ainda está em estágio muito prematuro de desenvolvimento e garante uma pluralidade de usos que só um projeto de maior escala poderia garantir.

### 3. ESTRUTURA DOS ARQUIVOS

O projeto está dividido em três partes principais, as *prototipações*, as *implementações* e os *recursos gráficos* nas pastas *include*, *src* e *imagens/sprites* respectivamente.

#### 3.1 Prototipações

Os protótipos de funções e declaração de variáveis globais estarão aqui para serem acessados em todo o projeto, permitindo, desse modo, a modularização. Assim, a divisão do trabalho no desenvolvimento em equipe torna-se mais organizado e evita conflitos graves entre versões divergentes.



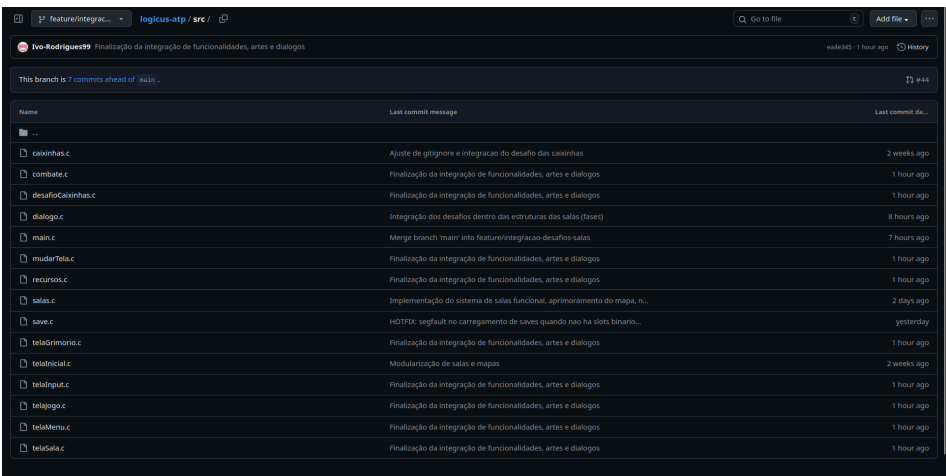
A screenshot of a GitHub commit history page for the 'include' directory. The page shows a list of commits with columns for Name, Last commit message, and Last commit date. The commits are listed in descending order of time.

Name	Last commit message	Last commit date
..		
caixinhas.h	Ajuste de gitignore e integracao do desafio das caixinhas	2 weeks ago
combate.h	Implementação de desafios e estruturação de salas	yesterday
dialogo.h	Merge branch 'main' of https://github.com/luaraunab/logicus-atp	yesterday
grimorio.h	Desafios do grimorio	yesterday
recursos.h	Finalização da integração de funcionalidades, artes e diálogos	1 hour ago
salas.h	Implementação do sistema de salas funcional, aprimoramento do mapa, n...	2 days ago
save.h	HOTFIX: segfault no carregamento de saves quando nao ha slots binario...	yesterday
telaSala.h	Correcao de bug no estado do mapa e salas	2 weeks ago
telas.h	Finalização da integração de funcionalidades, artes e diálogos	1 hour ago

Pasta “include”

#### 3.2 Implementações

Nas implementações estarão as funcionalidades propriamente ditas, que utilizam de informações declaradas nos arquivos de prototipação (*header files*).



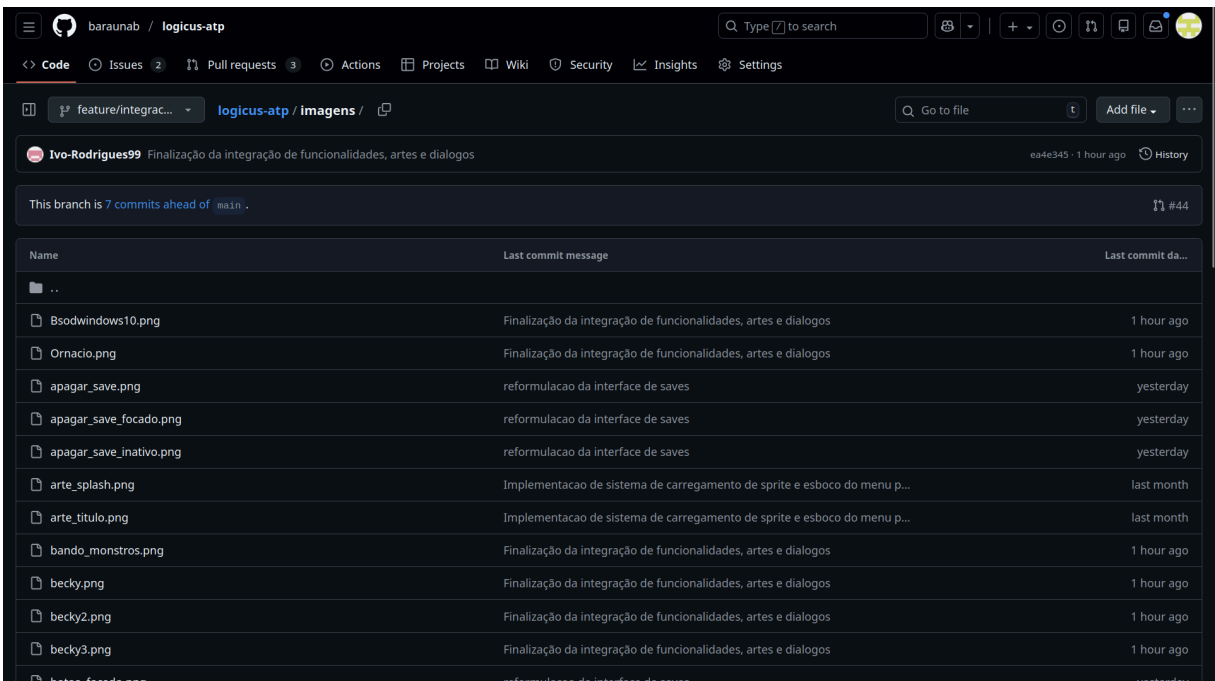
A screenshot of a GitHub commit history page for the 'src' directory. The page shows a list of commits with columns for Name, Last commit message, and Last commit date. The commits are listed in descending order of time.

Name	Last commit message	Last commit date
..		
caixinhas.c	Ajuste de gitignore e integracao do desafio das caixinhas	2 weeks ago
combate.c	Finalização da integração de funcionalidades, artes e diálogos	1 hour ago
desafioCaixinhas.c	Finalização da integração de funcionalidades, artes e diálogos	1 hour ago
dialogo.c	Integração dos desafios dentro das estruturas das salas (fases)	8 hours ago
main.c	Merge branch 'main' into feature/integracao desafios salas	7 hours ago
mudarTela.c	Finalização da integração de funcionalidades, artes e diálogos	1 hour ago
recursos.c	Finalização da integração de funcionalidades, artes e diálogos	1 hour ago
salas.c	Implementação do sistema de salas funcional, aprimoramento do mapa, n...	2 days ago
save.c	HOTFIX: segfault no carregamento de saves quando nao ha slots binario...	yesterday
telaGrimorio.c	Finalização da integração de funcionalidades, artes e diálogos	1 hour ago
telaInicial.c	Modularização de salas e mapas	2 weeks ago
telaInput.c	Finalização da integração de funcionalidades, artes e diálogos	1 hour ago
telaJogo.c	Finalização da integração de funcionalidades, artes e diálogos	1 hour ago
telaMenu.c	Finalização da integração de funcionalidades, artes e diálogos	1 hour ago
telaSala.c	Finalização da integração de funcionalidades, artes e diálogos	1 hour ago

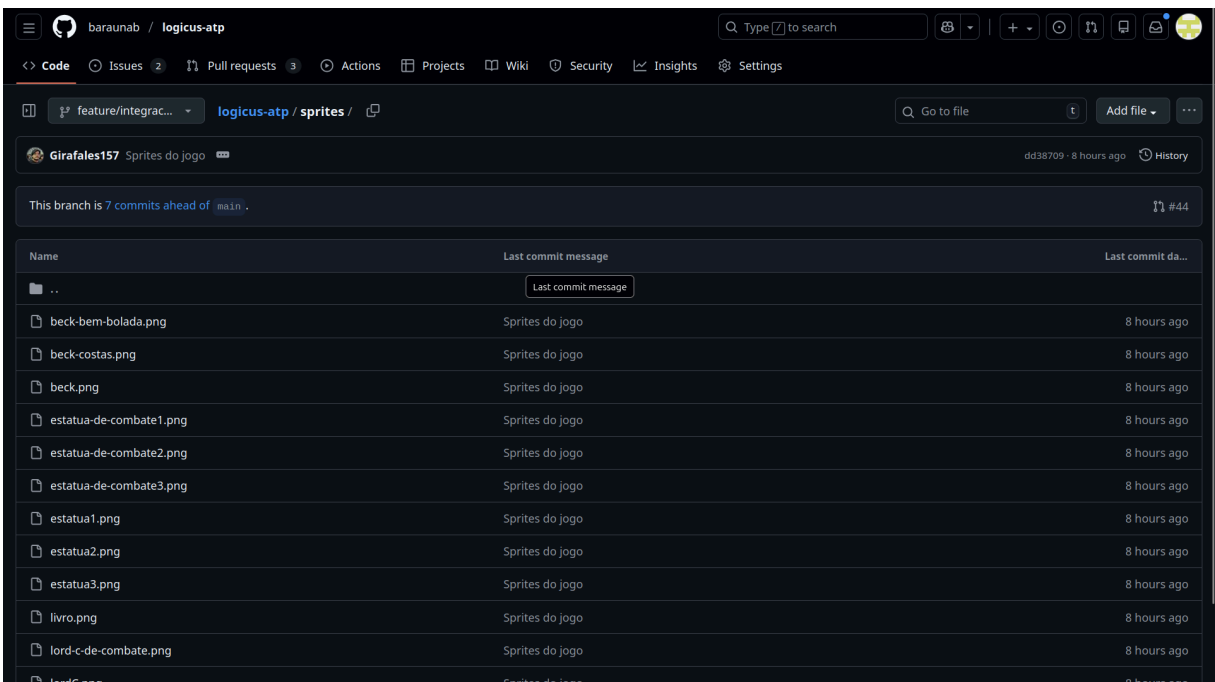
Pasta “src”

### 3.3 Recursos Gráficos

A interface do jogo usa imagens de fundo e de personagens para ilustrar a história de autoria da equipe de arte do projeto. Todas elas foram colocadas nas pastas “Imagens” e “Sprites”.



Pasta “Imagens”



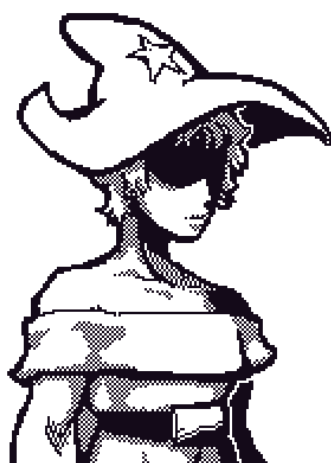
Pasta “Sprites”

### 3.3.1 Imagens usadas

Algumas das imagens utilizadas no projeto feitas pela equipe de artes visuais:



Cenário “Sala das estátuas” feito por Kinan Principe



Sprite da protagonista “Becky” feita por Anthony Rodrigues

## 4. CONCLUSÃO E MELHORIAS

Tendo em vista as informações contidas neste relatório e as demonstrações práticas, pode-se concluir que o "Logicus;" alcançou seu objetivo de criar uma ferramenta lúdica que complementa o aprendizado de iniciantes em programação e também em linguagem C.

A arquitetura do código e sua modularização demonstram grande capacidade de escalabilidade de maneira organizada e funcional. O jogo implementou mecânicas clássicas de RPG como o combate por turnos e o estilo Dungeon Crawler de forma eficiente, adaptando-as ao contexto da lógica de programação.

Para futuras expansões e melhorias, tem-se em vista os seguintes pontos:

1. **Analizador sintático integrado:** Substituir a validação de inputs através de strings comparativas por um interpretador integrado ao jogo, permitindo uma maior flexibilidade na resolução de enigmas e desafios.
2. **Expansão de conteúdos:** Adicionar mais salas (fases), inimigos, mecânicas e conteúdos referentes à linguagem C como estruturas, funções e ponteiros.
3. **Sistema de áudio:** Utilizar o módulo de áudio da RayLib para implementação de trilhas e efeitos sonoros.
4. **Aprimoramento visual:** Adição de animações, movimentação de personagens, feedbacks visuais para o usos de ataques e habilidades, reação visual para situações de tomar ou causar dano e aprimoramentos similares.