INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Visconde da Graça

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE – CÂMPUS PELOTAS - VISCONDE DA GRAÇA
CURSO TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS

LOG.IK – FACILITANDO A APRENDIZAGEM DE CONCEITOS DE LÓGICA DE PROGRAMAÇÃO PARA INICIANTES

Emerson Brahm da Silva

Pelotas, 2024.

Emerson Brahm da Silva

LOG.IK: Facilitando a aprendizagem de conceitos de lógica de programação para iniciantes

Trabalho de Conclusão de Curso apresentado como requisito na disciplina de Metodologia da Pesquisa II do curso Técnico em Desenvolvimento de Sistemas, do Instituto Federal Sul-rio-grandense – Campus Pelotas - Visconde da Graça.

Orientadora: Profa. Dra. Maria Isabel Giusti Moreira

Pelotas, 2024.

SUMÁRIO

1. INTRODUÇÃO

O ensino de lógica desde a juventude pode trazer inúmeros benefícios para o indivíduo, tanto na esfera pessoal quanto na acadêmica. Entre as vantagens de aprender cedo sobre lógica e lógica computacional está o desenvolvimento de habilidades que vão muito além da simples programação. O pensamento crítico, a resolução de problemas, o aprimoramento cognitivo e a tomada de decisões informadas são competências fundamentais para o sucesso em diversas áreas da vida. Além disso, compreender os fundamentos da lógica é essencial para lidar com a crescente digitalização do mundo, onde a lógica computacional está presente em tecnologias, sistemas e processos que usamos diariamente.

Apesar da sua relevância, o ensino e a aprendizagem desses conceitos enfrentam diversos obstáculos que precisam ser superados. Entre os principais desafios, destacam-se a capacidade de abstração e a compreensão de conceitos abstratos, que muitas vezes exigem um raciocínio mais elaborado por parte dos alunos. Soma-se a isso a escassez de ferramentas pedagógicas adequadas e metodologias eficazes que possam tornar o aprendizado acessível, especialmente para iniciantes. Outro ponto crucial é a necessidade de estratégias atrativas e envolventes que despertem o interesse dos jovens, especialmente em um mundo repleto de estímulos tecnológicos e imediatistas. Sem essas abordagens, o ensino de lógica pode parecer desinteressante e distante, o que dificulta a aprendizagem e a retenção de conceitos fundamentais.

Diante desse cenário, este trabalho apresenta o **LOG.IK**, um jogo interativo e lúdico desenvolvido para auxiliar no ensino de conceitos lógicos fundamentais, com ênfase na aplicação prática da tabela-verdade. A tabela-verdade é uma ferramenta central no ensino da lógica, pois permite a visualização clara de combinações lógicas, como os operadores AND, OR e NOT, entre outros. No entanto, sua abordagem tradicional, muitas vezes teórica, pode ser desafiadora e pouco estimulante para iniciantes.

O LOG.IK busca tentar preencher essa lacuna por meio de mecânicas de gamificação, criando uma experiência dinâmica, intuitiva e divertida para os alunos. O jogo transforma o aprendizado em desafios interativos, como o desarmamento de bombas virtuais, que exigem o uso da tabela-verdade e de conceitos lógicos para serem solucionados. Essa abordagem promove o envolvimento e oferece um

aprendizado prático, onde os usuários aplicam os conceitos em situações simuladas, fortalecendo sua compreensão e habilidade de raciocínio lógico.

Com essa proposta, o LOG.IK visa não apenas facilitar o aprendizado, mas também criar um ambiente que desperte a curiosidade e o interesse dos jogadores. Ao transformar conceitos abstratos em experiências tangíveis, o jogo oferece uma oportunidade de aprendizado acessível e inclusiva, ajudando iniciantes a superar barreiras comuns no ensino de lógica e fomentando o desenvolvimento de competências essenciais para o século XXI.

2. TEMA

Desenvolvimento de um jogo interativo denominado **LOG.IK**, um jogo interativo e lúdico desenvolvido para auxiliar no ensino de conceitos lógicos fundamentais, com ênfase na aplicação prática da tabela-verdade. Por meio de desafios interativos que requerem a resolução de problemas lógicos a partir de comandos escritos, o **LOG.IK** busca proporcionar uma experiência prática e lúdica, tornando o processo de ensino mais dinâmico, intuitivo e acessível.

3. MOTIVAÇÕES

A lógica de programação é uma competência essencial no mundo contemporâneo, com aplicações que vão desde a Ciência da Computação até a resolução de problemas em diversas áreas. Dessa forma, compreender e aplicar esses conceitos é fundamental para a formação acadêmica e profissional de estudantes.

Ensinar lógica computacional a jovens tem várias razões importantes. Uma delas é o desenvolvimento do pensamento crítico e analítico, que permite abordar problemas de forma estruturada e lógica. Outra razão é a habilidade de resolver problemas, já que a lógica computacional ajuda a quebrar problemas complexos em partes menores e mais gerenciáveis, tornando-os mais fáceis de resolver.

Além disso, a lógica computacional serve como fundamentação para a programação. Ela é a base da programação e, ao compreender seus conceitos, os estudantes tornam a aprendizagem de linguagens de programação mais fácil e acessível.

Há também a preparação para carreiras futuras. Em um mundo cada vez mais digital, habilidades em lógica computacional são altamente valorizadas em

diversas áreas. Isso inclui não apenas o setor tecnológico, mas também campos como ciência, engenharia, finanças e muitos outros. Por fim, a lógica computacional estimula a criatividade e a inovação. Resolver problemas lógicos e criar algoritmos promove o pensamento criativo, encorajando os jovens a pensar fora da caixa e encontrar soluções únicas para os desafios.

Para superar os desafios do ensino de lógica, métodos lúdicos e interativos têm se mostrado alternativas eficazes. Tais abordagens podem ser uma forma mais prática de captar a atenção dos jovens e mantê-los motivados a aprender e desenvolver suas habilidades lógicas. Ao integrar elementos de gamificação e cenários interativos, o aprendizado se torna mais dinâmico e envolvente.

Atualmente, uma das formas mais promissoras de tornar o ensino mais lúdico é por meio da utilização de jogos. Com base nessa perspectiva, foi desenvolvido um software educativo que combina aprendizado e diversão, funcionando como um jogo interativo. Inicialmente, o foco está na tabela-verdade, mas a proposta prevê futuramente a ampliação para outros cenários, nos quais o usuário deverá resolver desafios utilizando uma linguagem descritiva.

Esse jogo se chama **LOG.IK**, um trocadilho entre a palavra "lógica" e o comando **log**, utilizado em contextos relacionados à tabela-verdade e à resolução de desafios, como o desarmamento de bombas no jogo. O intuito foi oferecer um ambiente interativo que permitisse aos usuários compreender e aplicar conceitos fundamentais de lógica, utilizando mecânicas de gamificação para estimular o envolvimento e facilitar o aprendizado.

4. OBJETIVOS

4.1. OBJETIVO GERAL

Desenvolver um jogo interativo denominado **LOG.IK**, com o objetivo de auxiliar iniciantes na aprendizagem de lógica de programação, utilizando a tabela-verdade como base para a resolução de desafios.

4.2. OBJETIVOS ESPECÍFICOS

a) Utilizar a linguagem de programação Python e a biblioteca Pygame para o desenvolvimento do jogo, garantindo flexibilidade e eficiência na criação da plataforma;
b) construir uma plataforma interativa que facilite a compreensão dos conceitos da tabela-verdade, promovendo o aprendizado de forma prática e intuitiva;
c) criar desafios que atribuam pontuações a cada vitória, oferecendo ao jogador um retorno imediato e estimulando o envolvimento durante o aprendizado;
d) incorporar elementos visuais e sonoros atraentes, com o objetivo de aumentar o envolvimento e a motivação dos usuários ao longo do processo de aprendizado.

5. ESPECIFICAÇÃO DE REQUISITOS

Especificação de Requisitos de Software é uma etapa fundamental no desenvolvimento de sistemas, em que são detalhadas as funcionalidades, características e restrições do software a ser construído. Esse processo visa documentar de forma clara e compreensível tudo o que o sistema deve realizar, garantindo que as necessidades dos usuários sejam atendidas e que a equipe de desenvolvimento tenha uma base sólida para o trabalho.

Os requisitos podem ser classificados como funcionais, que descrevem o que o sistema deve fazer, e não funcionais, que especificam qualidades como desempenho, segurança e usabilidade. Neste capítulo, serão apresentados, primeiramente, os métodos utilizados para o levantamento de requisitos, seguidos pelos requisitos identificados para o desenvolvimento do **LOG.IK**, abordando tanto as funcionalidades principais do jogo quanto os aspectos técnicos e de interação que garantirão sua eficiência e atratividade.

5.1. MÉTODOS DE ESPECIFICAÇÃO DE REQUISITOS

A primeira técnica de levantamento de requisitos utilizada foi a **prototipagem**, que consiste em um processo iterativo no qual se cria um modelo preliminar do sistema para visualizar e experimentar suas funcionalidades antes do

desenvolvimento completo. O uso de protótipos permite antecipar a visualização do software, proporcionando ao usuário uma ideia clara do que esperar como resultado final. Para desenvolver as telas do protótipo do sistema (conforme mostram as Figuras 1, 2 e 3) foi utilizada a ferramenta **InVision**.

A Figura 1 apresenta a prototipação da tela de login do sistema **LOG.IK**. Nessa tela, o usuário deve inserir seu nome de usuário e senha para acessar o sistema. Além disso, se for o primeiro acesso do usuário, ele poderá clicar no “Registre-se.” A interface foi projetada com o objetivo de ser intuitiva e amigável, garantindo uma experiência de usuário eficiente e segura durante o processo de autenticação.

Figura 1: Tela de login

Fonte: Autoria Própria

A Figura 2 apresenta a prototipação da tela de registro do sistema LOG.IK. Nessa tela, o usuário deve preencher um formulário com seu nome de usuário, senha e endereço de e-mail.

Figura 2: Tela de Registro

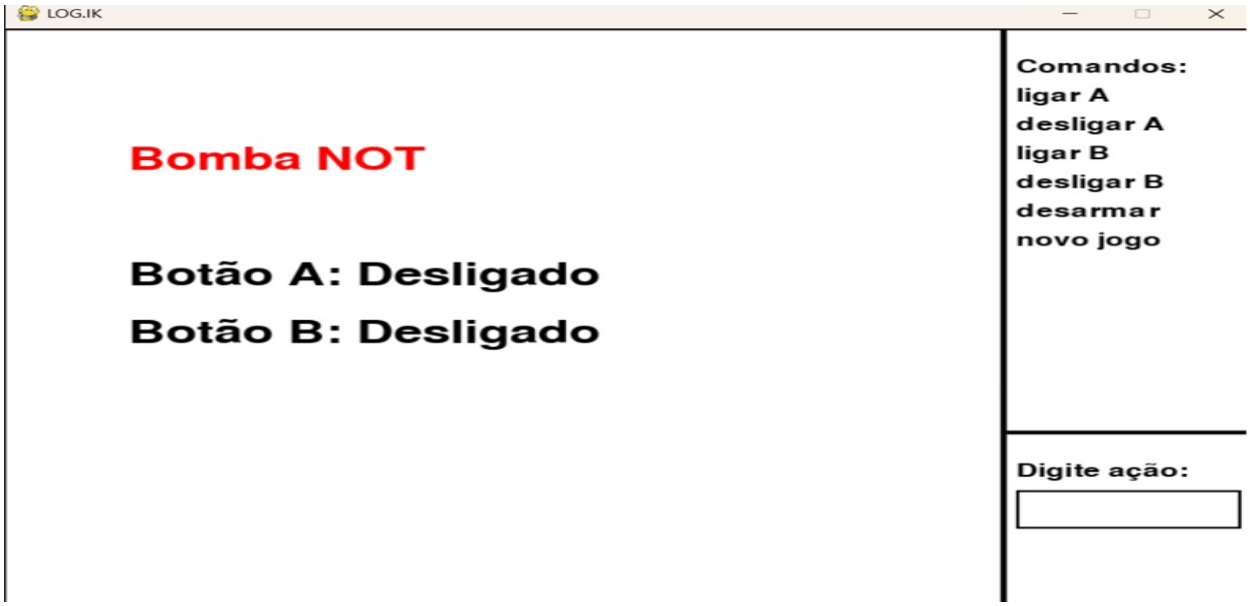
Fonte: Autoria Própria

A Figura 3 apresenta a prototipação da tela do jogo no sistema LOG.IK, organizada em três quadrantes distintos:

- **Quadrante Esquerdo:** dedicado à interface principal do jogo, onde o usuário interage diretamente com o ambiente e acompanha o progresso.
- **Quadrante Superior Direito:** projetado para fornecer dicas e orientações sobre as ações necessárias para avançar nas fases, ajudando o usuário a compreender melhor as tarefas e desafios.
- **Quadrante Inferior Direito:** reservado para a entrada de comandos e exibição do histórico das ações realizadas, facilitando o acompanhamento das atividades e a execução de novas tarefas.

Essa divisão em quadrantes foi cuidadosamente pensada para otimizar a experiência do usuário, proporcionando uma navegação clara, funcional e intuitiva dentro do jogo.

Figura 3: Tela de Jogo



Fonte: Autoria Própria

Outra técnica utilizada para o levantamento de requisitos foi a análise de sistemas semelhantes. Essa abordagem permitiu identificar funcionalidades, interfaces e boas práticas já aplicadas em soluções existentes, servindo como referência para o desenvolvimento do LOG.IK. Entre os sistemas analisados, destacam-se:

- **Code.org:** uma plataforma educacional que ensina conceitos básicos de programação e lógica por meio de atividades interativas e jogos.
- **Lightbot:** um jogo que utiliza desafios para ensinar lógica de programação, incentivando o raciocínio computacional.
- **Scratch:** uma ferramenta desenvolvida pelo MIT que permite criar histórias, jogos e animações, introduzindo conceitos de lógica de forma visual e intuitiva.
- **Tynker:** um sistema que oferece jogos e atividades para aprender programação, com desafios que incorporam lógica e resolução de problemas.

5.2. REQUISITOS FUNCIONAIS

Os requisitos funcionais descrevem as funcionalidades e os comportamentos necessários para que um sistema atenda às expectativas das partes interessadas. De acordo com Sommerville (2015), esses requisitos abrangem os serviços que o sistema deve fornecer, especificando como ele deve processar determinadas entradas e responder a situações específicas, detalhando seu comportamento em cenários particulares. Os requisitos funcionais do sistema estão listados no Quadro 1.

Quadro 1 – Requisitos Funcionais

REF	Caso de Uso	Descrição
REF 01	Criar conta	O sistema deve permitir que os usuários criem sua conta a partir de login e senha
REF 02	Fazer login	O sistema deve permitir que o usuário faça login a partir de login e senha previamente cadastrado
REF 03	Recuperar progresso	O sistema deve carregar automaticamente o progresso do usuário após o login
REF 04	Iniciar novo jogo	O sistema deve permitir que o usuário inicie um novo jogo
REF 05	Jogar fase	O sistema deverá permitir que o usuário interaja com o cenário a partir de uma área da tela exclusiva para input de texto onde os comandos serão executados
REF 06	Instruir ações	O sistema deve oferecer ao usuário instruções para interação com os elementos do cenário.
RE 07	Dar feedback	O sistema deve dar feedback em forma de texto sobre ações realizadas
REF 08	Passar de fase	Uma vez que os requisitos para concluir o desafio sejam atingidos o jogador poderá concluir a fase

Fonte: Autoria Própria

5.3. REQUISITOS NÃO FUNCIONAIS

Os requisitos não funcionais são especificações que detalham os critérios operacionais de um sistema, diferenciando-se das suas funcionalidades específicas. Eles descrevem o comportamento esperado do sistema e as restrições a serem atendidas, com foco em atributos de qualidade como desempenho, usabilidade, confiabilidade e segurança. Segundo Sommerville (2015), os requisitos não funcionais podem ser ainda mais críticos que os funcionais; se não forem atendidos, podem tornar o sistema inutilizável.

O Quadro 2 apresenta os requisitos não funcionais do sistema

Quadro 2 – Requisitos não funcionais

RNF	Classificação	Descrição
RNF01	Desempenho	O sistema deve ser capaz de processar comandos de linguagem natural e realizar ações correspondentes em tempo real, com um tempo de resposta inferior a 100ms.
RNF02	Desempenho	O sistema deve ser capaz de carregar e salvar o estado do jogo no banco de dados em menos de 1 segundo
RNF03	Usabilidade	A interface de usuário deve ser intuitiva e fácil de usar, permitindo que jogadores de todas as idades e níveis de habilidade possam jogar.
RNF04	Usabilidade	O sistema deve fornecer mensagens de erro claras e úteis quando comandos não reconhecidos são inseridos.
RNF05	Segurança	Os dados dos usuários e do jogo devem ser armazenados de forma segura e protegidos contra acesso não autorizado.
RNF06	Segurança	O sistema deve garantir que os dados do jogo não sejam corrompidos ou perdidos durante o salvamento e carregamento do estado do jogo.
RNF07	Desenvolvimento	O jogo deve ser compatível com as versões

		mais recentes do Python e Pygame.
RNF08	Desenvolvimento	O jogo deve ser compatível com o SGBD MySQL
RNF09	Manutenibilidade	O código do sistema deve ser bem documentado e organizado, seguindo as melhores práticas de programação para facilitar a manutenção e a expansão futura.
RNF10	Manutenibilidade	O sistema deve ser modular, permitindo que novos recursos e melhorias sejam adicionados sem afetar negativamente as funcionalidades existentes.

Fonte: Autoria Própria

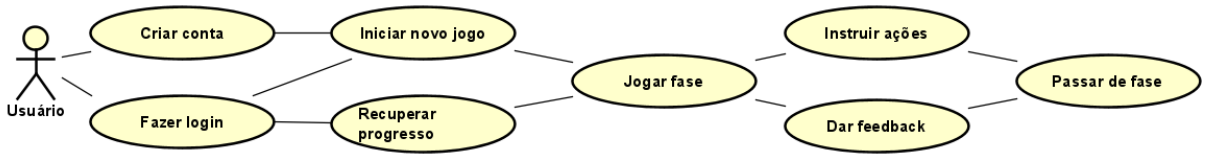
6. MODELAGEM

6.1. MODELO DE CASOS DE USO

De acordo com Sommerville (2015), os modelos de casos de uso são uma ferramenta essencial para capturar e especificar os requisitos funcionais de um sistema, proporcionando uma visão clara das interações entre os atores e o sistema.

Dessa forma, o modelo de casos de uso do sistema a ser desenvolvido, que explicita as principais ações e seus atores, é apresentado na Figura 4.

Figura 4. Modelo de Caso de Uso

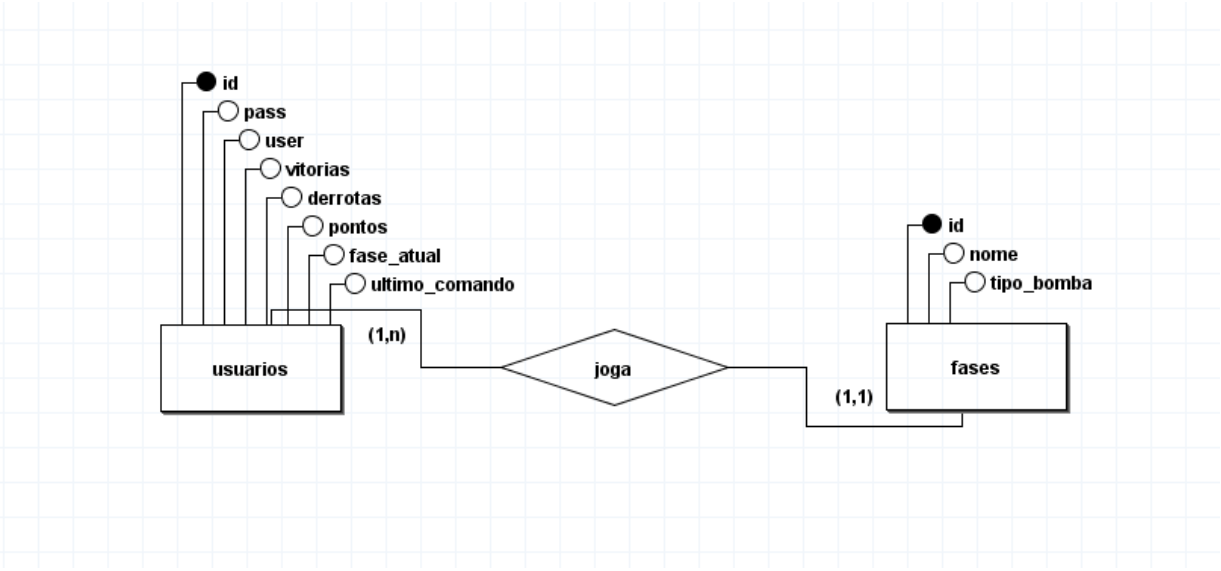


Fonte: Autoria Própria

6.2. MODELAGEM CONCEITUAL DO BANCO DE DADOS

De acordo com Sommerville (2015), a modelagem conceitual é essencial para garantir que os requisitos de dados sejam compreendidos e documentados de forma clara e precisa antes da implementação do banco de dados. O modelo conceitual de banco de dados, que explicita as entidades e seus relacionamentos no sistema a ser desenvolvido, está representado na Figura 5.

Figura 5: Modelo conceitual do banco de dados

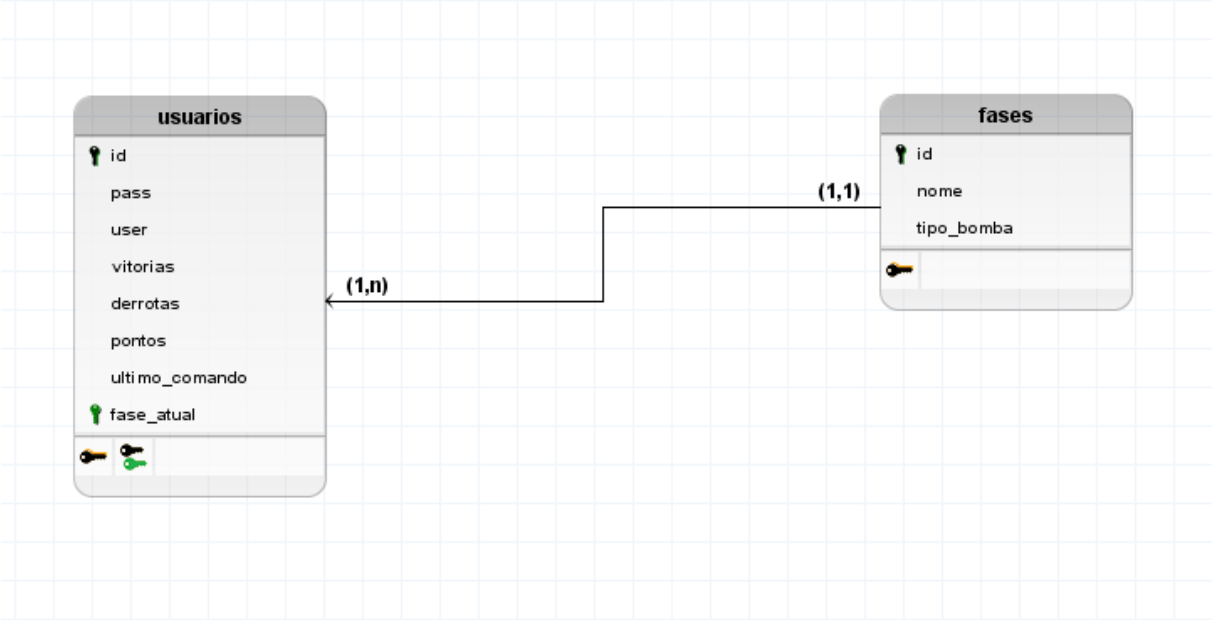


Fonte: Autoria Própria

6.3. MODELAGEM LÓGICA DO BANCO DE DADOS

A modelagem lógica, segundo Sommerville (2015), é o processo de traduzir os requisitos e o modelo conceitual de um sistema para representações mais detalhadas, geralmente destinadas à implementação. Esse tipo de modelagem busca capturar como os dados serão organizados em um banco de dados, usando conceitos como tabelas, chaves primárias, e relações entre tabelas, enquanto ainda é independente de um banco de dados específico. Esse nível é essencial para assegurar que o projeto atenda aos requisitos de dados do sistema antes da implementação técnica. Está representado na Figura 6.

Figura 6: Modelo lógico do banco de dados



Fonte: Autoria Própria

7. TECNOLOGIAS UTILIZADAS

O desenvolvimento do sistema LOG.IK foi realizado com base em tecnologias amplamente reconhecidas por sua eficiência, versatilidade e simplicidade, tanto no desenvolvimento de jogos quanto na criação de interfaces e no gerenciamento de dados. Este capítulo apresenta as principais tecnologias empregadas no projeto, destacando suas características e contribuições para a implementação do sistema.

- **Python:** é uma linguagem de programação de alto nível e propósito geral, criada por Guido van Rossum em 1991. Ela se destaca pela facilidade de aprendizado e por sua sintaxe clara e legível, o que a torna uma escolha popular entre iniciantes e desenvolvedores experientes. Sua versatilidade permite o desenvolvimento de aplicações em diversas áreas, incluindo jogos, análise de dados e inteligência artificial.
- **Pygame:** é uma biblioteca desenvolvida em Python para a criação de jogos e aplicativos multimídia. Focada em simplicidade e eficiência, ela é construída sobre a biblioteca SDL (*Simple DirectMedia Layer*), o que facilita o uso de gráficos, sons e controles em jogos 2D. Sua estrutura oferece recursos fundamentais para o desenvolvimento de jogos interativos, tornando-a ideal para projetos educacionais como o LOG.IK.
- **QTDesign:** é um framework que possibilita o desenvolvimento de interfaces gráficas de usuário (GUIs) utilizando Python e o toolkit Qt, uma biblioteca amplamente usada para criar aplicações visuais multiplataforma. Desenvolvido pela Riverbank Computing, PyQt combina a flexibilidade e a simplicidade do Python com o poder do Qt, permitindo criar interfaces robustas e intuitivas, adequadas para sistemas complexos.
- **Tkinter:** é a biblioteca padrão do Python para a criação de interfaces gráficas de usuário (GUIs). Ela oferece ferramentas simples e rápidas para o desenvolvimento de aplicativos visuais, utilizando widgets pré-configurados, como botões, caixas de texto, menus e rótulos. Sua integração nativa com Python a torna uma escolha prática para projetos que exigem interfaces gráficas básicas.
- **MySQL:** é um sistema de gerenciamento de banco de dados relacional (SGBDR) de código aberto, amplamente utilizado para armazenar e gerenciar dados de forma estruturada. Inicialmente desenvolvido por Michael "Monty" Widenius e atualmente mantido pela Oracle Corporation, o MySQL é uma das tecnologias de banco de dados mais populares, reconhecida por sua confiabilidade, desempenho e ampla adoção em aplicações web e sistemas corporativos.

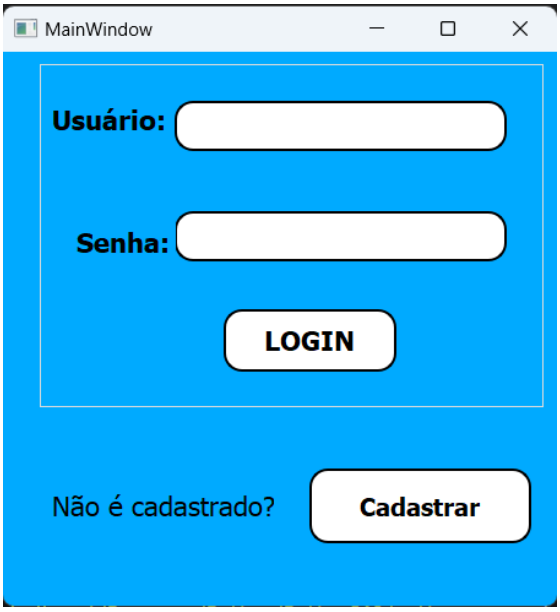
8. DESCRIÇÃO DO SISTEMA

Este capítulo apresenta uma visão detalhada do sistema LOG.IK, explicando sua estrutura, funcionalidades e o fluxo de interação com o usuário. O objetivo é descrever como o sistema foi projetado para alcançar seus propósitos educacionais, focando na aprendizagem de lógica de programação por meio de um ambiente interativo e lúdico. Serão abordados os principais componentes do sistema, as telas desenvolvidas, os cenários de uso e a dinâmica de funcionamento, destacando como cada elemento contribui para a experiência do usuário e para o cumprimento dos objetivos do projeto.

8.1 Tela de login

Ao abrir o sistema, o usuário será direcionado para a tela de login (Figura 7). Nessa tela, além de poder inserir suas credenciais para acessar o sistema, também há a opção de criar uma nova conta, caso o usuário ainda não esteja cadastrado.

Figura 7: Tela de login

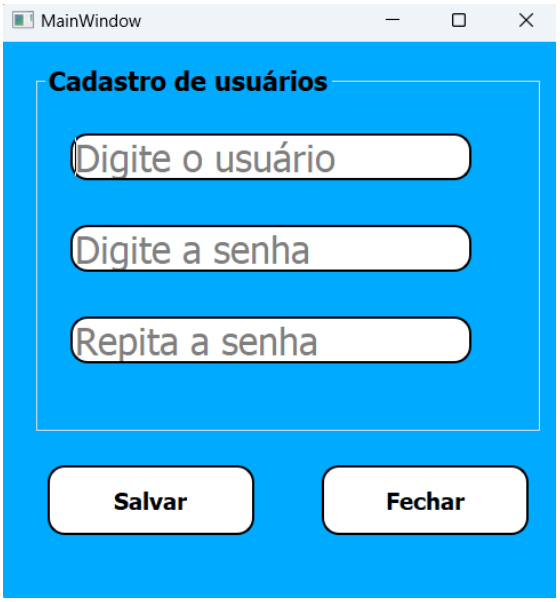


Fonte: Autoria própria

8.2 Tela de registro

Ao clicar na opção **Cadastrar**, o usuário será redirecionado para a página de registro (Figura 8). Nessa página, será necessário informar um nome de usuário, criar uma senha e confirmá-la repetindo no campo correspondente.

Figura 8: Tela de registro

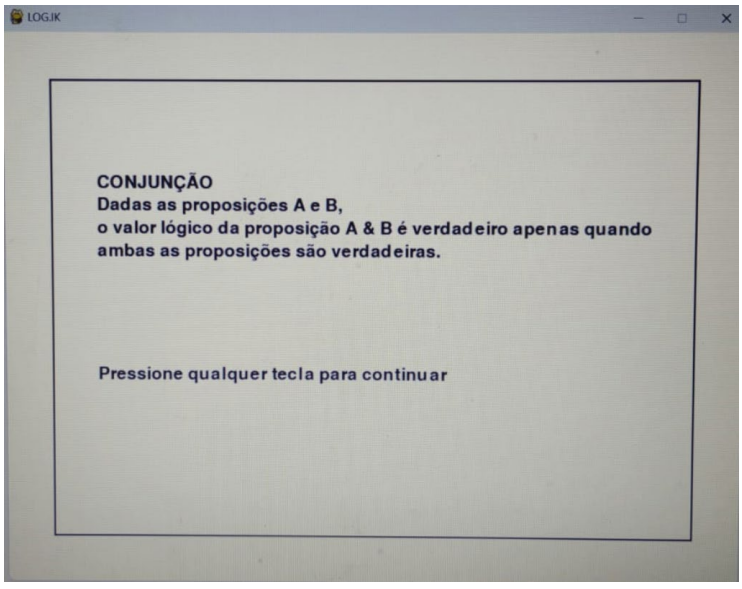


Fonte: Autoria própria

8.3 Tela de instruções

Após fazer o login, o usuário será direcionado para uma tela introdutória que apresenta a fase em que está jogando (Figura 9). Nessa tela, serão exibidas informações detalhadas sobre o contexto do desafio e as condições necessárias para vencer.

Figura 9: Tela de instrução



Fonte: Autoria própria

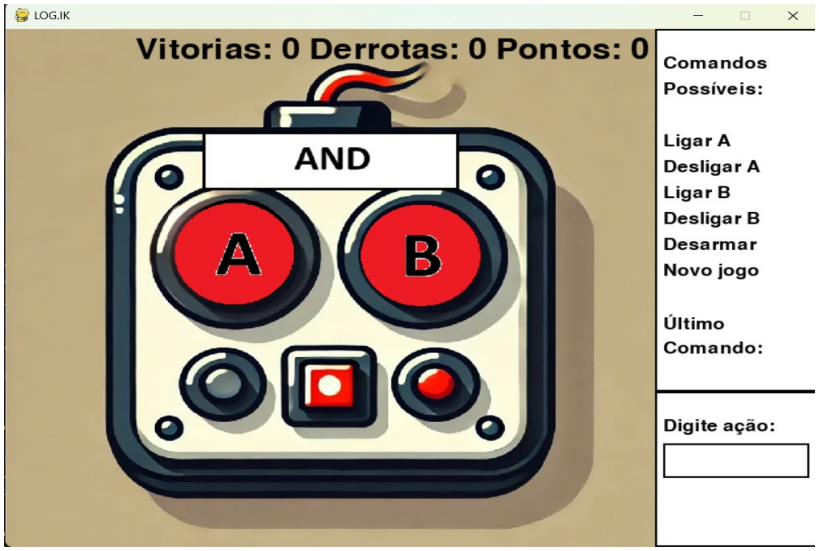
8.4 Tela dos Desafios

Na tela de desafio do jogo (Figura 10), o usuário encontrará três áreas principais:

- **Área Principal (à esquerda):** exibe a imagem da bomba e seu estado atual, permitindo ao jogador acompanhar visualmente o progresso do desafio. Além disso, apresenta um placar com o número de vitórias, derrotas e a pontuação total acumulada, proporcionando retorno imediato sobre o desempenho do jogador.
- **Área de Comandos (canto superior direito):** apresenta uma lista dos comandos possíveis reconhecidos pelo sistema, além de exibir o último comando inserido pelo jogador. Essa seção serve como guia para auxiliar na interação com o sistema.
- **Área de Ação (canto inferior direito):** é onde o jogador digita os comandos necessários para interagir com o sistema e resolver o desafio.

Essa divisão foi projetada para organizar as informações de forma clara e funcional, facilitando a navegação e garantindo uma experiência intuitiva para o usuário durante o jogo.

Figura 10: Tela de jogo

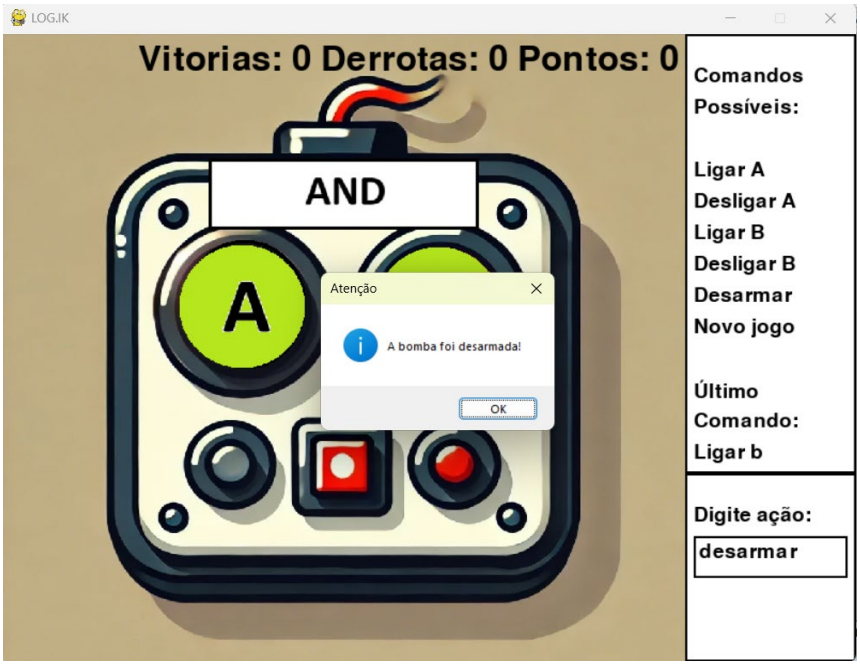


Fonte: Autoria própria

8.5 Aviso de passagem de fase

Quando o jogador inserir o comando desarmar, a fase será concluída, e o sistema exibirá uma mensagem (Figura 11) indicando sucesso ou falha, de acordo com o desempenho do jogador. Caso o desafio seja completado com sucesso, o contador de Vitórias será incrementado no banco de dados. Por outro lado, se o jogador falhar, o contador de Derrotas será atualizado. A pontuação total é calculada pela subtração entre o número de vitórias e derrotas, refletindo o desempenho acumulado do jogador ao longo do jogo.

Figura 11: Aviso de sucesso ou falha



Fonte: Autoria própria

8.6 Alerta de comando inválido

Caso o jogador insira um comando que não esteja na lista de comandos possíveis, o sistema exibirá automaticamente uma mensagem de erro (Figura 12). Essa mensagem alertará o jogador sobre o comando inválido, orientando-o a utilizar apenas os comandos reconhecidos pelo sistema.

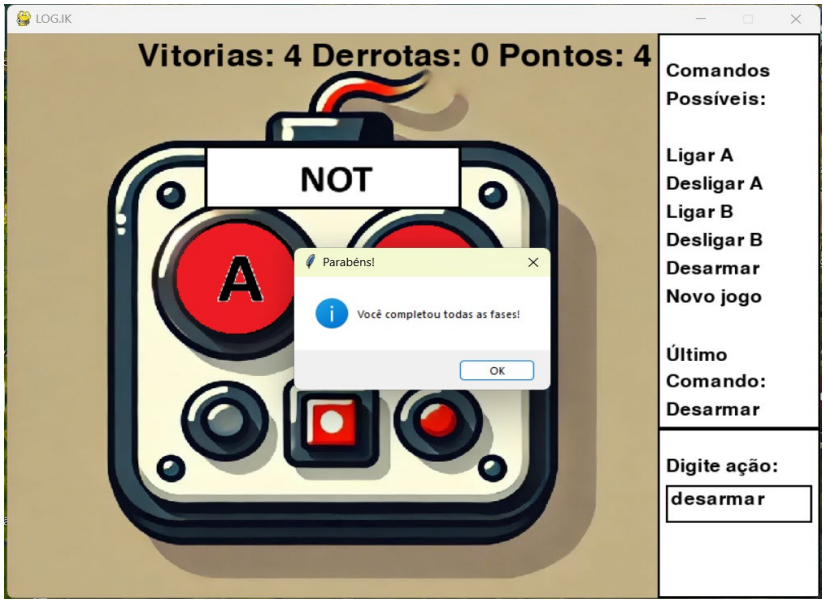
Figura 12: Aviso de comando inválido

Fonte: Autoria própria

8.7 Aviso de fim de jogo

Quando o jogador vencer todos os desafios propostos uma mensagem de conclusão aparecerá na tela e o jogo retornará à primeira fase (Figura 13).

Figura 13: Aviso de conclusão de jogo



Fonte: Autoria própria

9. CONSIDERAÇÕES FINAIS

Durante o ano de 2024, foi desenvolvido o jogo LOG.IK, com o objetivo de oferecer uma maneira lúdica de visualizar conceitos práticos de lógica de computação. Embora o sistema, em seu estado atual, ainda possua muitos pontos a serem aprimorados — como a ampliação do escopo e o aprofundamento dos tópicos abordados — ele cumpre a função para a qual foi projetado e se mostra funcional.

No entanto, não foi possível adicionar uma variedade maior de exemplos e desafios, que tornariam o aprendizado mais abrangente e dinâmico. Esse ponto representa oportunidade para melhoria futura no sistema, garantindo uma experiência ainda mais completa para os usuários.

A escolha de desenvolver um sistema utilizando uma biblioteca com a qual o autor nunca havia tido contato se revelou um grande desafio. No entanto, o processo de aprendizado adquirido ao longo do desenvolvimento foi extremamente enriquecedor e contribuiu significativamente para o crescimento profissional do autor, fazendo com que todos os esforços valessem a pena.

10. REFERÊNCIAS

SOMMERVILLE, Ian. Engenharia de Software. 10ª Edição. Pearson, 2015

APÊNDICE I - Instruções SQL para Criação da Base de Dados

```
-- phpMyAdmin SQL Dump
-- version 5.2.1
-- https://www.phpmyadmin.net/
--
-- Host: 127.0.0.1
-- Tempo de geração: 25/11/2024 às 18:05
-- Versão do servidor: 10.4.32-MariaDB
-- Versão do PHP: 8.2.12

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Banco de dados: `sistema`
--

-- --------------------------------------------------------

--
-- Estrutura para tabela `fases`
--

CREATE TABLE `fases` (
  `id` int(11) NOT NULL,
  `nome` varchar(255) NOT NULL,
  `tipo_bomba` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Despejando dados para a tabela `fases`
--

INSERT INTO `fases` (`id`, `nome`, `tipo_bomba`) VALUES
(0, 'Fase 1', 'AND'),
(1, 'Fase 2', 'OR'),
(2, 'Fase 3', 'IF/ELSE'),
```

```
(3, 'Fase 4', 'IF AND ONLY IF'),
(4, 'Fase 5', 'NOT');

-- --------------------------------------------------------

--
-- Estrutura para tabela `usuarios`
--

CREATE TABLE `usuarios` (
  `id` int(11) NOT NULL,
  `pass` varchar(255) NOT NULL,
  `user` varchar(255) NOT NULL,
  `vitorias` int(11) NOT NULL DEFAULT 0,
  `derrotas` int(11) NOT NULL DEFAULT 0,
  `pontos` int(11) GENERATED ALWAYS AS (`vitorias` - `derrotas`) STORED,
  `fase_atual` int(11) DEFAULT NULL,
  `ultimo_comando` varchar(255) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Despejando dados para a tabela `usuarios`
--

INSERT INTO `usuarios` (`id`, `pass`, `user`, `vitorias`, `derrotas`, `fase_atual`, `ultimo_comando`)
VALUES
(1, '1', 'a', 9, 1, 1, NULL);

--
-- Índices para tabelas despejadas
--

--
-- Índices de tabela `fases`
--
ALTER TABLE `fases`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `unique_nome` (`nome`);

--
-- Índices de tabela `usuarios`
--
ALTER TABLE `usuarios`
  ADD PRIMARY KEY (`id`),
  ADD KEY `fk_fase_atual` (`fase_atual`);
```

```
--
-- AUTO_INCREMENT para tabelas despejadas
--

--
-- AUTO_INCREMENT de tabela `fases`
--
ALTER TABLE `fases`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=219;

--
-- AUTO_INCREMENT de tabela `usuarios`
--
ALTER TABLE `usuarios`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

--
-- Restrições para tabelas despejadas
--

--
-- Restrições para tabelas `usuarios`
--
ALTER TABLE `usuarios`
  ADD CONSTRAINT `fk_fase_atual` FOREIGN KEY (`fase_atual`) REFERENCES `fases` (`id`) ON DELETE
SET NULL;
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```