Antônio Mariano, Davi Paulino, Lucas Neves, João Victor, Piero Tubino, Marcelo de Lima

Arcade Retrô Conectado:

Arcade Fight

Relatório Técnico elaborado conforme a ABNT NBR 10719:10, apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, como parte dos requisitos para a obtenção do grau de Técnico em Manutenção e Suporte em Informática.

Projeto Integrador

Orientador: Prof. Paulo Evaristo

São João da Boa Vista

2026

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO - CÂMPUS SÃO JOÃO DA BOA VISTA	Junho	2026

Técnico de Nível Médio em Manutenção e Suporte em Informática

Arcade Retrô Conectado:
Arcade Fight

Antônio Mariano, Davi Paulino, Lucas Neves, João Victor, Piero Tubino,
Marcelo de Lima e
Prof. Paulo Evaristo

Palavras-chave:
Arcade Fight. Linux Mint. Hardware.

28 páginas

SUMÁRIO

RESUMO

O projeto 'Arcade Retrô Conectado: Arcade Fight' consistiu na melhoria de um gabinete de jogos focado na integração escolar por meio de partidas online do jogo "Arcade Fight", game desenvolvido por estudantes do IFSP Campus São João da Boa Vista. A execução técnica estruturou-se em três pilares: hardware (processador Intel Core 2 Duo, 4 GB de memória RAM e SSD de 128 GB), software e redes (sistema operacional Linux Mint, jogo Arcade Fight conectado ao banco de dados na internet). No desenvolvimento do game utilizou-se as tecnologias HTML5, CSS3, JavaScript, PostgreSQL e Python. Como resultado, obteve-se um produto acessível e inovador que aplica de forma multidisciplinar os conhecimentos do curso Técnico em Manutenção e Suporte em Informática, demonstrando-se viável como ferramenta de lazer e aprendizado prático para a comunidade escolar.

Palavras-chave: Arcade Fight; Linux Mint; Hardware.

1 INTRODUÇÃO

A evolução da tecnologia e dos sistemas computacionais tem propiciado a modernização de plataformas clássicas de entretenimento. No contexto dos jogos eletrônicos, os fliperamas representam um marco histórico de interação social e lazer.

Os gabinetes de jogos clássicos eram historicamente limitados a conexões locais e hardwares analógicos obsoletos. Diante da necessidade de modernização e adaptação desses equipamentos para cenários tecnológicos atuais, surge a problemática de como transformar uma estrutura física preexistente em uma plataforma digital estável, conectada e capaz de suportar interações online dinâmicas.

Diante dessa perspectiva, unificar as áreas de hardware, software e redes para redefinir a experiência clássica do fliperama, aliando a nostalgia dos jogos retrô às demandas contemporâneas de conectividade global é um grande desafio.

Para que a junção dessas três partes seja possível na área do hardware é essencial a Arquitetura Von Neumann (clico de dados). Já na questão do software, é necessário o sistema operacional e os games do fliperama (front-end, banco de dados e back-end). Para que tudo isso seja conectado a internet, é fundamental as redes. Com essa conectividade de redes se torna possível a nostalgia dos jogos retrô vinculada as demandas da modernidade.

No âmbito pedagógico, os arcades funcionam como um laboratório prático multidisciplinar, no qual conceitos teóricos de configuração de sistemas, redes e manutenção são aplicados de forma concreta e desafiadora. Eles também estimulam a criatividade, trabalho em equipe e a resolução de problemas de desempenho e integração de componentes.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo geral desse trabalho é aprimorar o fliperama existente no IFSP Campus São João da Boa Vista, a fim de beneficiar os alunos da instituição.

1.1.2 Objetivos Específicos

- Realizar o aprimoramento dos componentes de hardware do arcade existente;
- Substituir o sistema operacional do fliperama;
- Desenvolver o jogo online Arcade Fight e implementar o game na máquina física.

2 REVISÃO DA LITERATURA

2.1 Arquitetura Von Neumann

2.1.1 Ciclo de Dados

De acordo com Tanenbaum e Austin (2013), a base de funcionamento dos computadores digitais modernos fundamenta-se no modelo proposto por John von Neumann. A essência dessa arquitetura consiste na divisão do sistema em uma Unidade Central de Processamento (CPU), uma memória principal capaz de armazenar tanto dados quanto instruções de programas, e mecanismos para a comunicação externa.

Conforme explica Stallings (2017), o funcionamento lógico desse modelo ocorre por meio de um ciclo contínuo e ordenado de processamento de dados. O processamento de instruções básico consiste em duas etapas: a CPU lê (busca) instruções da memória, uma de cada vez, e executa cada instrução. Esse fluxo compreende três fases fundamentais: entrada, processamento e saída de dados.

Segundo Stallings (2017), a entrada de dados é a captação de dados brutos vindos do ambiente externo. O processamento de dados se refere a execução de operações aritméticas e lógicas pela CPU sobre as informações armazenadas temporariamente na memória. E por fim a saída de dados faz o envio dos resultados processados e inteligíveis de volta ao usuário ou a outros sistemas.

2.1.2 Arquitetura da Placa-mãe

Segundo Torres (2001), a placa-mãe (ou motherboard) representa o elemento físico central de interconexão de um computador. Ela funciona como a "espinha dorsal" do hardware, sendo o componente responsável por garantir que todos os módulos internos e periféricos do sistema se comuniquem entre si de maneira coordenada e eficiente. A arquitetura interna da placa-mãe abriga e integra barramentos e componentes vitais para a operação do sistema, sendo eles: processador, memórias, vídeo, slots de expansão e fonte de alimentação.

De acordo com Stallings (2017), o processador é considerado o "cérebro" do computador, ele interpreta os dados e executa as instruções enviadas pelos softwares.

Conforme apontam Tanenbaum e Austin (2013), a memória RAM atua como o armazenamento primário, volátil e de alta velocidade para dados que estão sendo usados no exato momento. Em contrapartida o HDD e o SSD exercem o papel de memória secundária

não-volátil, retendo arquivos, programas e o sistema operacional permanentemente, mesmo quando a máquina é desligada.

De acordo com Torres (2001), o vídeo é um circuito encarregado do processamento gráfico e da geração de imagens na tela. Ele pode ser integrado nativamente aos chips da placa-mãe ou processador (onboard) ou conectado de forma independente através de uma placa de vídeo dedicada (offboard).

Com base em Torres (2001), slots de expansão são conectores padronizados (como as linhas PCI Express) que viabilizam o encaixe de novas placas e funcionalidades (como placas de rede, som ou vídeo), permitindo atualizações e customizações no hardware.

2.1.3 Dispositivos de Entrada e Saída

Segundo Stallings (2017), os dispositivos de entrada e saída (também conhecidos como periféricos de E/S ou I/O) constituem a interface de comunicação direta entre o usuário e o processador do computador.

De acordo com Stallings (2017), em termos conceituais, os dispositivos de entrada (como teclados, mouses e escâneres) convertem as ações e dados do mundo real em sinais binários inteligíveis para o processador. Por outro lado, os dispositivos de saída (como monitores, impressoras e caixas de som) realizam a tarefa inversa: traduzem os dados binários já processados pela máquina de volta em formatos visuais, táteis ou sonoros compreensíveis para o ser humano.

2.2 Sistemas Operacionais

2.2.1 Função dos Sistemas Operacionais

Segundo Tanenbaum e Bos (2016), o sistema operacional atua essencialmente como um intermediário entre o usuário de um computador e o hardware da máquina. Sua principal função é gerenciar de forma eficiente todos os recursos do sistema como o processador, as memórias e os dispositivos de entrada e saída, fornecendo uma interface simplificada para que os programas possam ser executados sem que o desenvolvedor precise conhecer os detalhes físicos dos circuitos.

De forma complementar, Silberschatz, Galvin e Gagne (2015) apontam que esse software busca otimizar a utilização dos recursos de computação, garantindo ao mesmo tempo uma operação justa, organizada e segura entre os diferentes usuários e aplicativos que disputam a atenção do hardware simultaneamente.

2.2.2 Sistemas Operacionais: Microsoft Windows, macOS e Linux

Conforme explicam Silberschatz, Galvin e Gagne (2015), o Microsoft Windows consolidou-se historicamente como o sistema operacional mais popular para computadores de mesa e laptops de uso pessoal e corporativo. Seu sucesso e adoção em massa estão fortemente atrelados à sua ampla compatibilidade com diferentes componentes de hardware do mercado e a uma interface gráfica intuitiva que facilitou a popularização da informática.

Tanenbaum e Bos (2016) destacam que o macOS, desenvolvido pela Apple, adota um modelo de negócios de plataforma fechada, operando estritamente nos computadores fabricados pela própria empresa. O macOS possui uma base tecnológica fundamentada em sistemas Unix, o que combina uma estabilidade e segurança robustas a um forte apelo visual, sendo altamente valorizado em setores de design gráfico, edição de mídia e desenvolvimento de software.

Segundo Nemeth et al. (2015), diferenciando-se das opções anteriores por seu modelo de código aberto, o Linux não é de propriedade de uma única corporação. Ele é desenvolvido de forma colaborativa por uma comunidade global de desenvolvedores e serve de núcleo para uma infinidade de sistemas que operam desde servidores de internet e supercomputadores até dispositivos móveis e sistemas embarcados de automação.

2.2.3 Linux e a Distribuição Linux Mint

Com base em Nemeth et al. (2015), em termos estritamente técnicos, o termo "Linux" refere-se apenas ao núcleo (kernel) do sistema operacional, que é o componente encarregado da comunicação direta com os componentes físicos da máquina. Para que se torne um sistema operacional completo e utilizável pelo usuário comum, esse núcleo é combinado com ferramentas utilitárias, ambientes gráficos e aplicativos variados, formando o que se conceitua como uma distribuição Linux.

De acordo com a documentação oficial do projeto Linux Mint (2026), o propósito fundamental do sistema é fornecer um ambiente moderno, elegante e confortável, focado na facilidade de uso imediato. A distribuição já inclui nativamente o suporte a formatos multimídia e apresenta uma interface gráfica clássica que remete ao modelo tradicional de desktop, minimizando a curva de aprendizado para novos usuários.

2.3 Pilares de Desenvolvimento

Segundo Teruel (2014), a construção de sistemas digitais modernos baseia-se em uma arquitetura dividida em camadas funcionais interdependentes. O desenvolvimento divide-se tradicionalmente entre o front-end, focado na interface e na experiência visual com a qual o usuário interage diretamente, o back-end, encarregado do processamento

lógico e das regras de negócio nos servidores, e o banco de dados é responsável pelo armazenamento seguro e persistente de todas as informações do sistema.

Conforme definem Pressman e Maxim (2021), para otimizar a criação dessas camadas, a engenharia de software faz uso de estruturas auxiliares padronizadas. Um framework consiste em um conjunto de ferramentas e códigos preexistentes que fornece um esqueleto arquitetural reutilizável. Sua adoção permite que os desenvolvedores foquem nas particularidades da aplicação, em vez de reconstruírem funcionalidades estruturais básicas do zero.

2.3.1 Front-end

De acordo com Duckett (2014), no ecossistema do desenvolvimento front-end, a construção de interfaces digitais apoia-se na integração sinérgica de três tecnologias fundamentais: HTML5, CSS3 e JavaScript. O HTML5 atua como a fundação estrutural e o esqueleto semântico da página; a introdução de tags nativas estruturais (como <header>, <article> e <section>) permitiu não apenas a organização padronizada de elementos textuais e de mídia, mas também a otimização da acessibilidade e do indexamento por motores de busca.

Conforme Duckett (2014), o CSS3 opera na camada de apresentação estética e design. Essa tecnologia gerencia aspectos visuais como tipografia, cores e efeitos de transição, destacando-se pela utilização de media queries e sistemas de layout modernos (como Flexbox e Grid) para viabilizar o design responsivo, garantindo que a interface se adapte dinamicamente a diferentes tamanhos de tela e dispositivos.

Segundo Flanagan (2012), o JavaScript introduz o comportamento dinâmico e a lógica programática essencial para a experiência do usuário. A linguagem atua diretamente na manipulação do Modelo de Objeto do Documento (DOM), permitindo que o navegador processe eventos em tempo real, valide dados de formulários e execute requisições assíncronas sem a necessidade de recarregar a página, o que transforma documentos estáticos em aplicações web interativas e reativas no ambiente do cliente.

Com base na documentação do Projeto Electron (2026), o Electron é um framework que viabiliza a criação de aplicativos desktop nativos reaproveitando o conhecimento técnico de HTML5, CSS3 e JavaScript, alcançado por meio da união do motor de renderização do Chromium ao ambiente de execução Node.js.

2.3.2 Banco de Dados

O armazenamento e a recuperação de dados são críticos para a perenidade de qualquer software de gestão. Banco de dados é uma coleção organizada e logicamente

coerente de dados correlacionados, desenhada especificamente para refletir transações e informações do mundo real de maneira estruturada (ELMASRI; NAVATHE, 2011).

Dentre as soluções tecnológicas para gerenciar essas coleções, destaca-se o PostgreSQL. Ele é classificado como um sistema gerenciador de banco de dados relacional de código aberto que se destaca pela robustez computacional, conformidade estrita com os padrões da linguagem SQL e alta tolerância a falhas em ambientes corporativos de grande escala (SILVEIRA; SILVA, 2023).

2.3.3 Back-end

A engrenagem lógica do servidor depende de instruções formais e protocolos de comunicação bem definidos. Linguagem de programação é um sistema formal de escrita contendo regras de sintaxe e semântica que permite ao programador codificar algoritmos e controlar o comportamento físico e lógico de um computador (SEBESTA, 2018).

Para que esses algoritmos se comuniquem com outros sistemas, utilizam-se interfaces padronizadas. Uma API (Application Programming Interface) é como um conjunto de contratos, rotinas e ferramentas que estabelece uma ponte tecnológica, permitindo que softwares distintos troquem dados e serviços entre si de forma automatizada e segura (MASSE, 2012).

No cenário do desenvolvimento back-end, a linguagem Python é frequentemente adotada devido ao seu design focado na legibilidade. Python é uma linguagem de alto nível e multiparadigma, cujo propósito principal é acelerar a produtividade do desenvolvedor por meio de uma sintaxe limpa e clara (LUTZ, 2014).

De acordo com a documentação do Projeto FastAPI (2026), para maximizar o ecossistema Python no desenvolvimento de serviços web, adota-se o micro-framework FastAPI. FastAPI é um framework moderno e de alta performance projetado para construir APIs estáveis. Ele se diferencia no mercado por sua velocidade de execução comparável a tecnologias baseadas em Node.js e por automatizar a criação de documentações interativas utilizando a tipagem nativa da linguagem.

2.4 Bash e Podman

2.4.1 Bash (Bourne Again SHell)

De acordo com Shotts (2020), o Bash consiste em um interpretador de comandos e linguagem de script padrão para a grande maioria das distribuições Linux e sistemas baseados em Unix. Ele funciona como uma interface de texto fundamental que viabiliza a interação direta do usuário com o sistema operacional. Por meio dele, torna-se possível

não apenas executar programas isolados, mas também construir scripts complexos capazes de automatizar tarefas rotineiras e operacionais de forma ágil.

Adicionalmente, Nemeth et al. (2015) salientam que a relevância do Bash no ecossistema de software livre se deve à sua capacidade de gerenciar processos, manipular fluxos de arquivos e encadear comandos distintos. Essa flexibilidade transforma a linha de comando em um ambiente altamente eficiente para a administração de servidores e configuração de ambientes de desenvolvimento.

2.4.2 Podman (Pod Manager)

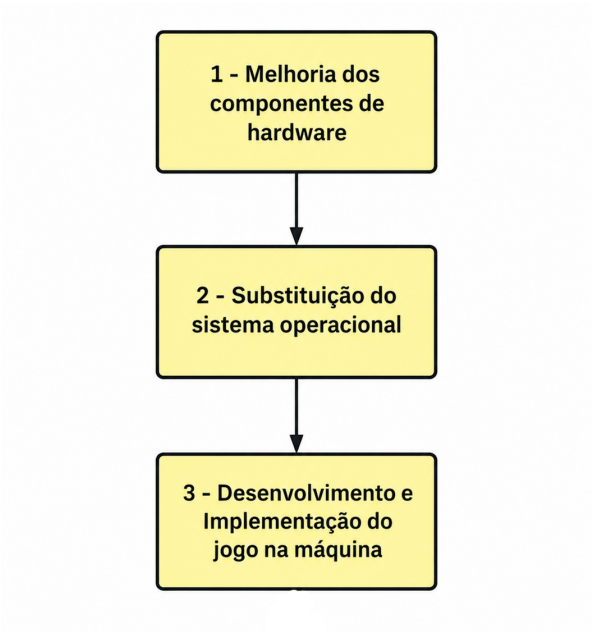
Conforme explica Walsh (2022), o Podman é uma ferramenta de código aberto voltada para a criação, gerenciamento e execução de contêineres e pods em conformidade com as diretrizes da Open Container Initiative (OCI). O principal diferencial do Podman em comparação com os motores de contêiner mais tradicionais, como o Docker, está em sua arquitetura sem daemon (daemonless). Isso significa que ele opera sem a necessidade de um processo centralizado rodando continuamente em segundo plano com privilégios de administrador.

Segundo o Projeto Podman (2026), essa independência de um daemon central eleva substancialmente o nível de segurança do ecossistema computacional. A ferramenta permite o isolamento completo de aplicações em modo rootless, ou seja, usuários comuns conseguem implantar e administrar seus próprios contêineres sem colocar em risco a integridade do sistema operacional hospedeiro.

3 METODOLOGIA

Nesse presente trabalho foi proposto o aprimoramento do fliperama existente no IFSP Campus São João da Boa Vista, a fim de beneficiar os alunos da instituição. Para tanto será aplicada a metodologia apresentada na Figura 1 que segue logo abaixo.

Figura 1 – Metodologia do trabalho



Fonte: Elaborada pelos autores

Inicialmente foi realizado o aprimoramento dos componentes de hardware da placa-mãe do arcade. Primeiramente foi efetuado o diagnóstico do estado dos equipamentos, identificou-se quais eram as configurações do computador desktop e por fim ocorreu a substituição de alguns componentes da placa-mãe para que o sistema operacional e o jogo que seriam escolhidos futuramente tivessem um melhor desempenho.

Em seguida executou-se a substituição do sistema operacional Windows 10 que já estava na máquina. Para tanto efetuou-se o backup dos arquivos e programas do sistema. Então foi instalado o novo sistema operacional: Linux Mint.

Por fim, foi realizado o desenvolvimento e a implementação do jogo online Arcade Fight na máquina. Para tanto no desenvolvimento do game foram seguidas as seguintes etapas: criação do front-end, banco de dados e back-end. No desenvolvimento do front-end utilizadas as seguintes tecnologias: HTML5 Canvas, CSS3, JavaScript e o framework front-end Electron. Para o banco de dados: PostgreSQL. E por fim no back-end: Python 3.12 e a API FastAPI. Logo em seguida para implementar o jogo na máquina do Linux Mint aplicaram-se as seguintes ferramentas: Bash e Podman.

4 ANÁLISE DOS RESULTADOS

Nessa seção serão apresentados os resultados obtidos nesse trabalho com a aplicação da Metodologia abordada na seção anterior.

O desenvolvimento deste projeto integrador fundamenta-se na construção de uma plataforma autônoma de entretenimento, materializada em um jogo de luta 2D multiplayer no estilo arcade clássico. A execução prática e o planejamento metodológico foram segmentados estrategicamente para contemplar de forma sistêmica a descrição clara do funcionamento do jogo, a modelagem estrutural do gabinete, o cronograma das atividades e os recursos financeiros utilizados. Essa abordagem integrada garantiu que a infraestrutura física de hardware e a configuração lógica do sistema operacional operassem em perfeita harmonia, otimizando o desempenho do sistema para a execução nativa dos de jogos.

4.1 Hardware

A engenharia de hardware do arcade foi projetada com foco em três diretrizes fundamentais: estabilidade estrutural, ergonomia e eliminação de latência de entrada nos comandos. O processo seguiu uma sequência lógica que compreendeu desde a seleção dos componentes computacionais internos até o acabamento estético e funcional do gabinete. Diferente de sistemas comerciais fechados, a escolha dos componentes internos priorizou a eficiência no processamento de aplicações de jogos e a sustentabilidade técnica:

CPU (Unidade Central de Processamento): foi integrado um computador configurado para garantir a execução fluida dos gráficos e gerenciamento dos periféricos. Os componentes (processador, memória RAM e armazenamento de massa via disco rígido SATA) foram selecionados especificamente para suprir as demandas do motor do jogo e de ferramentas de contêineres locais.

Sistema de Exibição de Imagem: foi reaproveitado o monitor de vídeo já implementado no arcade pois estava em bom estado e adequado para o uso novo. Houve melhorias como a pintura dos seus moldes mostrando melhor aparência e transmitindo ergonomia ideal para visualização confortável de seus jogadores.

Subsistema de Áudio: para proporcionar imersão sonora retrô acústica, caixas de som (alto-falantes combinado com um amplificador de som) foram embutidas na parte superior da estrutura física, garantindo a reprodução fiel de trilhas e efeitos sonoros.

4.2 Controles e Mapeamento

A fidelidade da experiência de um fliperama clássico reside na precisão dos comandos. Para o desenvolvimento do painel de controle, foram reaproveitados e adotados os seguintes procedimentos:

- **Montagem dos Comandos Físicos:** o painel foi equipado com joysticks analógico-digitais.
- **Placa Controladora Zero Delay:** para realizar a interface entre os botões físicos e o computador, utilizou-se uma placa controladora Zero Delay USB. Este componente eletrônico converte os acionamentos mecânicos dos switches em sinais digitais limpos via barramento USB.
- **Mapeamento Nativo de Baixo Nível:** visando eliminar gargalos de processamento causados por softwares intermediários de terceiros (como emuladores de controle), os comandos USB foram mapeados nativamente de forma direta dentro do código-fonte do próprio jogo. Isso garantiu um tempo de resposta imediato a cada ciclo de processamento da física.
- **Estrutura:** visualmente, a estrutura foi personalizada com reforçamento e realce da pintura em spray para demonstrar uma identidade moderna e chamativa ao projeto.

4.3 Sistema Operacional: Configuração e Otimização

Agora falando do Sistema Operacional do arcade, inicialmente no trabalho anterior estava com o Windows 10 como uma primeira versão do sistema. Mas ao decorrer do desenvolvimento do projeto, houve instabilidades nos desempenhos e na performance do jogo escolhido. Então houve um replanejamento para garantir o SO com a maior fluidez para garantir a boa qualidade de experiência de gameplay. Houve uma formatação e limpeza da máquina para garantir que o sistema funcione de forma adequada e sem complicações para o novo sistema operacional que a seguir irá ser descrito.

Então para garantir a maior performance, a distribuição de software escolhida foi o Linux Mint pois, os principais motivos técnicos e os passos para a sua configuração foram:

- **Desempenho e Leveza:** Linux Mint gerencia os recursos do computador de forma eficiente, garantindo que o processador e a memória RAM fiquem livres para focar na execução e nos gráficos do jogo.
- **Inicialização direta (Autostart):** Como era mais complicado fazer um script de inicialização personalizada no Windows 10, o Linux veio de bom grado para desenvolver esse script. O sistema operacional foi configurado para ignorar o carregamento da

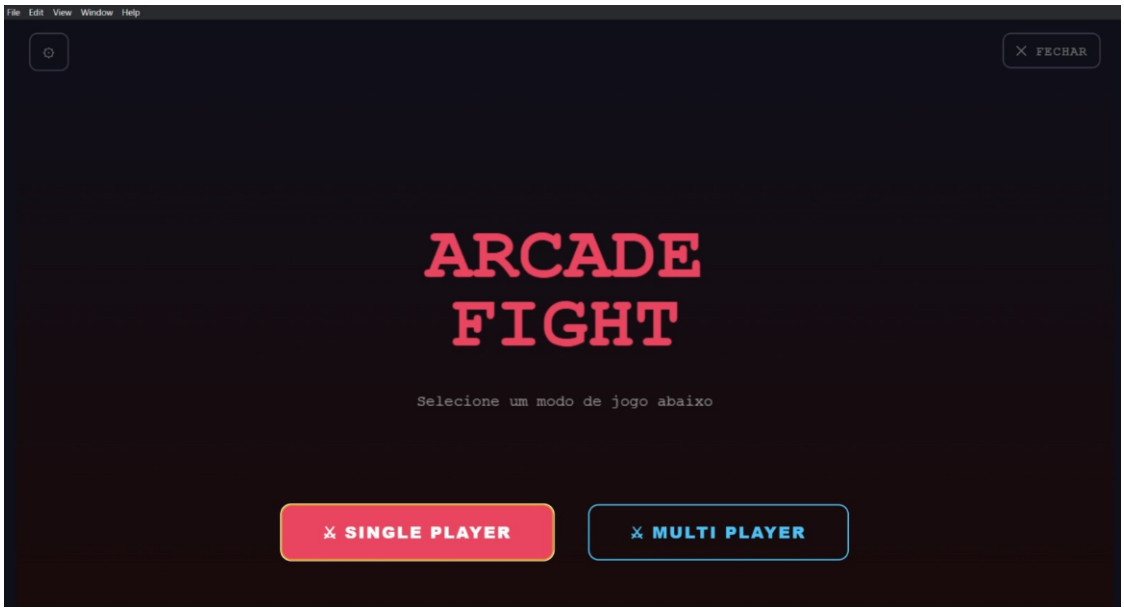
área de trabalho comum. Assim que a máquina é ligada, uma rotina de inicialização automática executa os scripts do jogo garantindo que ao iniciar o computador, já entrasse direto nos carregamentos iniciais do jogo.

A seguir é detalhado o jogo e as sua funcionalidades.

4.4 Desenvolvimento e Implementação

Arcade Fight é um jogo de luta 2D inspirado nos clássicos dos fliperamas dos anos 90. Dois jogadores escolhem entre cinco personagens únicos e se enfrentam em combates no melhor de dois rounds. O projeto foi desenvolvido para rodar em um gabinete arcade físico da instituição, mas funciona normalmente em qualquer computador com Windows ou distribuições Linux.

Figura 2 – Tela inicial do game com opções



Fonte: Elaborada pelos autores

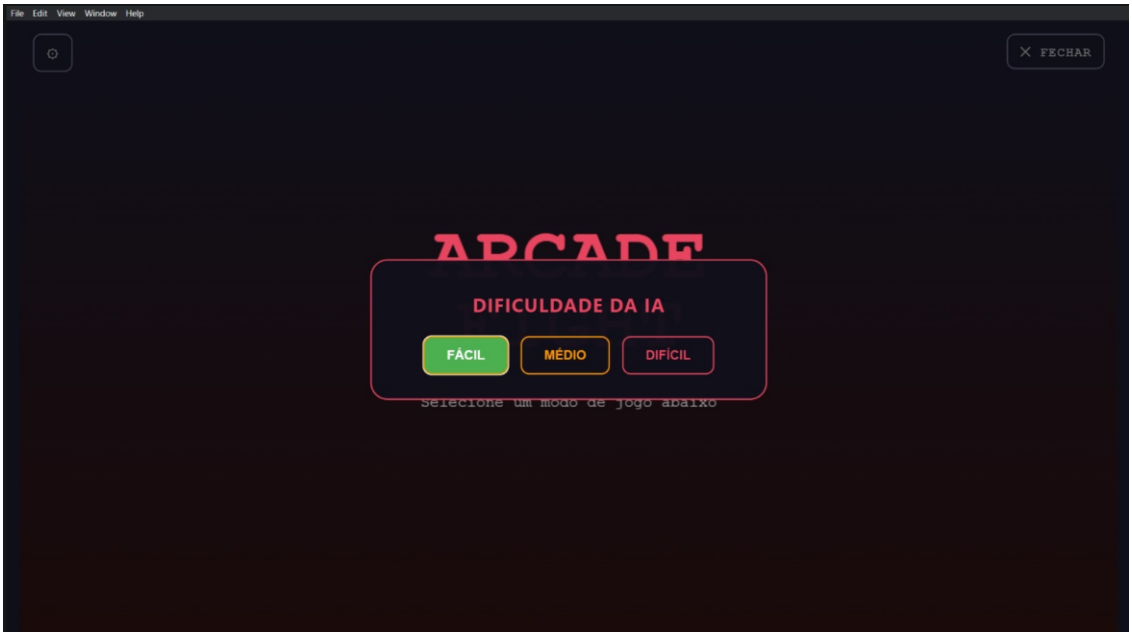
Descrição: Tela inicial do game com as opções de jogo contra I.A. ou contra um segundo player.

4.4.1 Sistemas e Mecânicas de Jogo

- **Barra de Especial:** Uma barra dedicada de super-habilidade que acumula energia conforme o dano sofrido e mitigado (pela postura de defesa). Habilidade especial estará disponível quando a barra estiver cheia, dando chance para uma virada na partida.

- **Mecânica de Defesa Eficiente:** A ativação do escudo reduz em, 70 por cento, o dano recebido e habilita janelas curtas de frame para a execução de contra-ataques. A janela para contra-ataque é de 300ms e revida o golpe com, 50 por cento, a mais de dano para o contra-atacante.
- **Animações:** Movimentos e efeitos visuais/sonoros que ocorrem em quadros (ou o popular Frame Per Second, "Quadros Por Segundo") previstos no código e comandados pelo jogador, sendo que os personagens terão animações para as seguintes circunstâncias: ocioso(parado), andando, correndo, pulando, ataque 1, ataque 2, ataque 3, dano, morte e defesa.
- **IA adaptativa:** três níveis (fácil, médio e difícil) com reações e agressividade diferentes.
- **Sistema de rounds:** melhor de 3 rounds (Best of 3) decide o vencedor (primeiro a vencer 2 rounds).

Figura 3 – Seção de dificuldade para a I.A.



Fonte: Elaborada pelos autores

4.4.2 Personagens e suas Características

O balanceamento foi estruturado matematicamente através de atributos exclusivos para 5 classes distintas:

Figura 4 – Personagens e suas Características

Personagem	Estilo de Combate	Vantagem de Atributo	Dinâmica Principal
Espadachim	Melee Baseado em Armas	Alcance Médio Ampliado	Controle de espaço e punição a média distância.
Lutador	Grappler / Brawler	Dano Alto e Vida Elevada	Pressão constante em combate corpo a corpo estrito.
Mago	Zoner / Projetar	Habilidades de Longo Alcance	Cadência de magias mantendo o oponente afastado.
Vampira	Agile Striker	Velocidade de Movimentação Elevada	Esquivas rápidas e counters dinâmicos.
Vampiro	Sustained Fighter	Mecânica de Regeneração/ Vampirismo	Trocas prolongadas de dano com recuperação sutil.

Fonte: Elaborada pelos autores

Figura 5 – Seleção dos Personagens

Fonte: Elaborada pelos autores

Descrição: Cena com a seleção dos personagens em que é possível ver os atributos do personagem.

Figura 6 – Partida em andamento

Fonte: Elaborada pelos autores

Descrição: Partida em andamento com os elementos principais (barra de vida (verde), barra de especial (amarela), cenário e notação do round - no centro superior.

4.4.3 Redes: Conexão automática Wi-Fi

Quando o PC for inicializado, automaticamente irá se conectar ao Wi-Fi do IF. Pedimos autorização para a CTI para que liberasse o MAC (identificador da placa de rede) para que se conecte automaticamente sem fazer o procedimento de login na rede do IF.

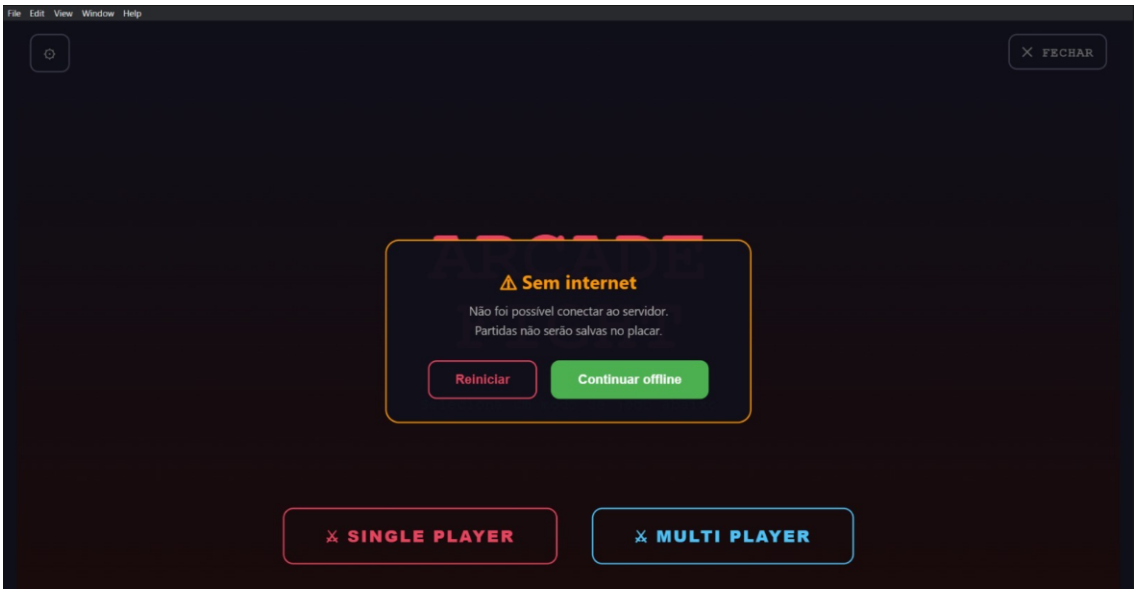
Para a catalogação do placar no site do Arcade Fight, o banco de dados PostgreSQL atua como repositório central e definitivo do sistema. Quando o jogador realiza o login via QR Code, sua sessão móvel é vinculada à partida do gabinete. É o PostgreSQL que armazena o histórico de partida e acumula a pontuação diretamente na conta de cada usuário de forma persistente. Se não houver conexão, o jogo abre no modo offline com jogadores identificados como convidados, sem salvar resultados no site oficial.

O servidor da aplicação (aqui "aplicação" significa os elementos do jogo, e "servidor" o gerenciador da comunicação entre o jogo e o PostgreSQL) gera dinamicamente um QR Code com base no IP ou URL dentre os endereços utilizáveis. O jogador pode realizar o escaneamento nativo com a câmera do celular, sendo redirecionado para a interface web móvel da sessão para iniciar a partida e, posteriormente, para a interface de exibição dos resultados ao fim do combate. Após o login, a sessão do jogador é vinculada à partida do gabinete, e seu histórico e pontuação somam diretamente o Ranking Global (Top 20) mantido no banco de dados PostgreSQL.

Agora, se não houver conexão, o jogo poderá ser jogado em modo offline, sem

sessão, com o jogador entrando como convidado e sem atualizar o seu placar online ao fim da partida.

Figura 7 – Aviso de jogo off-line



Fonte: Elaborada pelos autores

Descrição: Aviso de jogo off-line. A pontuação não será somada ao fim do jogo na tabela do site do PostgreSQL.

Durante qualquer partida, pressionar os botões Single Player ou Multi Player exibe uma confirmação de saída. No modo online, um aviso informa que os dados não serão salvos caso o jogador opte por retornar ao menu.

4.4.4 Adaptação de controles

Figura 8 – Controles arcade

Controle Arcade (mapeamento padrão)

Ação	P1	P2
Pular	Verde	Verde
Ataque rápido	Amarelo	Amarelo
Ataque forte	Preto	Preto
Bloquear	Vermelho	Vermelho
Especial	Azul	Azul
Singleplayer	L2	—
Multiplayer	—	R2

Fonte: Elaborada pelos autores

Figura 9 – Modos Singleplayer e Multiplayer.

Fonte: Elaborada pelos autores

Descrição: Os botões que alternam o modo do jogo entre singleplayer e multiplayer.

4.4.5 Modo de execução dos comandos

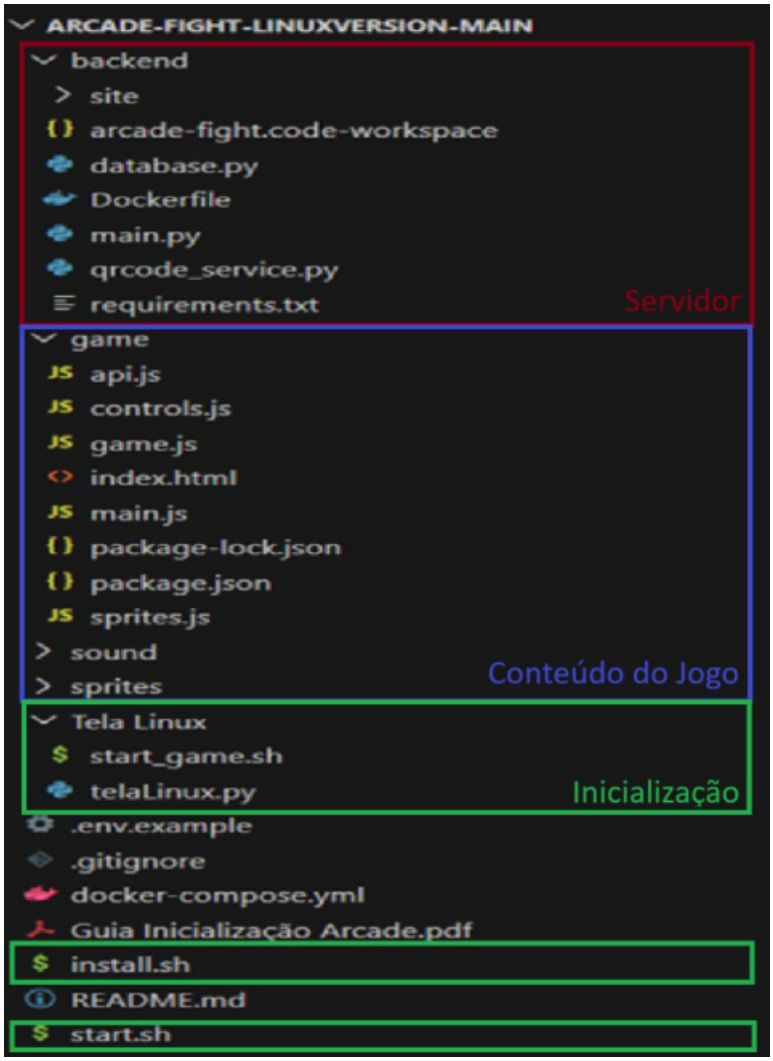
O mapeamento dos joysticks foi feito para que fossem processados nativamente pelo código do jogo os comandos dados pelo jogador (sem intermediários como emuladores ou camadas densas de tradução): Comunicação de baixo nível direta com as controladoras USB do gabinete arcade dentro da arquitetura de software (ecossistema), eliminando camadas extras de tradução (como emuladores de terceiros x360ce), minimizando o input lag (atraso de resposta do comando de entrada).

Nota: intermediários entre o comando dado e o código que vai executar tal comando poderiam ser máquinas virtuais (Java Virtual Machine, por exemplo) ou emuladores como o próprio x360ce já mencionado.

4.4.6 Infraestrutura e Estrutura do jogo

Considerando os elementos que o projeto carrega, sendo elas combate, placar online, sistema operacional, execução de comandos de joysticks, cenários e músicas, ele foi estruturado em seções com funções específicas:

Figura 10 – Estrutura do Código



Fonte: Elaborada pelos autores

A primeira seção é voltada para o servidor, banco de dados e container. Todo o processo servidor-cliente entre o gabinete, o site do placar e o banco de dados se encontra ali. A princípio, todas as informações online serão alocadas no postgreSQL. Para o site em que o placar está, foram usadas as ferramentas JavaScript, HTML e CSS para regras de funcionamento, estruturação de página e design visual, respectivamente. Para padronização simplificada e mais leve do sistema (em comparação a uma máquina virtual, que precisa de RAM e processamento adicionais isolados do S.O. rodando em paralelo), o Docker serve para gerar um ambiente para rodar todo o projeto como se estivesse em uma "caixa" com as configurações ideais pré-determinadas na imagem que ele vai executar. O Docker, basicamente, vai executar uma imagem, feita de antemão, onde todos os arquivos necessários rodarão para que o jogo, banco de dados e S.O. interajam entre si dentro do gabinete que estiver rodando o Docker. É um "estruturador de ambientes", em suma.

A segunda seção é o conteúdo do jogo propriamente - os personagens, os cenários, as animações, a postura de defesa, os ataques, os atributos do personagem e regras da

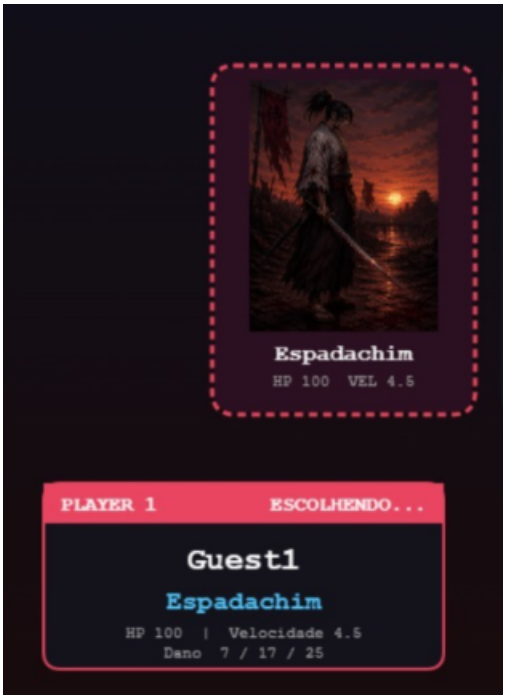
partida. As regras de funcionamento do jogo e os sprites estão em JavaScript.

A terceira seção está posta para inicializar o jogo automaticamente. O segredo para o programa rodar sozinho está em um recurso do Linux chamado Autostart (inicialização automática de sessão). Basicamente, nós criamos um pequeno arquivo de atalho com a extensão .desktop dentro de uma pasta oculta do sistema (/.config/autostart). Esse arquivo funciona como uma instrução direta para o sistema operacional, dizendo: "Assim que o usuário fizer o login, ignore a Área de Trabalho comum e execute imediatamente este script em Python". Para dar certo, o arquivo aponta o caminho exato de onde o código está salvo no computador e força a interface a abrir diretamente em modo tela cheia, ocultando até o ponteiro do mouse.

Uma vez acionada, essa interface em Python (feita com a biblioteca Tkinter) assume o papel de gerenciador de ciclo de vida do fliperama. Ela fica ativada em segundo plano o tempo todo. Quando o jogador aperta um botão do gabinete, essa tela em Python faz uma chamada interna e invoca o jogo principal (que roda em Electron, o motor principal) por cima dela. O grande truque está no ciclo de vida: se o jogador sair do jogo ou a partida fechar, a tela em Python detecta isso instantaneamente e volta para o foco principal (tela inicial do jogo) em tela cheia, garantindo que o desktop do Linux nunca fique exposto e mantendo a experiência de um fliperama de verdade.

Toda a comunicação, interna e externa entre banco de dados, servidor, gabinete, site é feita com arquivos JSON, que é um formato de dados muito comum no mercado por sua simplicidade: funciona por correspondência entre uma chave e um ou mais valores. Então, para o personagem “Espadachim”, a exemplo, temos o valor de vida “100”, a velocidade de “4,5” e os danos de ataque “7”,”17”,”25” para os três ataques, respectivamente.

Figura 11 – Personagens e suas Habilidades



Fonte: Elaborada pelos autores

Nota: todos os códigos mencionados nesta seção estão postos em maior detalhe no repositório: <https://github.com/marcelod6427/Arcade-Fight-LinuxVersion>.

4.5 Orçamento

Os recursos financeiros utilizados ao longo do projeto estão logo abaixo na tabela:

Ordem	Item	Quantidade	Valor Estimado Total (R$)
1	Kit 4 rodas em silicone industrial	1	47
2	MDF 2m x 0,63m	3 placas	452,85
3	Suporte em L - 10mm	10	22,5
4	Sprays	4	93,92
5	Fita em T para acabamento	10 metros	34,41
6	Parafuso rosca soberba	60/90	36
7	KIT Parafuso e Porca MK8	20	28,98
8	Caixa de som KNUP	1	58,9
9	Kit Zero Delay Usb Sanwa	1	196,33
10	Computador (cérebro Arcade)	1	609
11	Monitor	1	268,7
12	Solda para eletrônica	1 rolo pequeno	38,7
13	Cabo de rede par trançado	1 metro	5
Total			1.892,29

Descrição Detalhada:

1. Conjunto de rodízios para instalar na base do gabinete do arcade, permitindo movimentá-lo facilmente sem danificar o piso, garantindo mobilidade e estabilidade;

2. Material base para construção do gabinete do arcade. É resistente, fácil de cortar e montar, permitindo criar a estrutura principal e laterais personalizadas;

3. Suportes metálicos utilizados internamente para reforçar e fixar as partes do gabinete, garantindo maior firmeza e durabilidade da estrutura;

4. 4 latas spray cores: azul, amarelo, preto e vermelho para dar cor e vida ao arcade;

5. Perfil em "T" utilizado nas bordas do MDF para dar acabamento, cobrindo cortes aparentes e deixando o gabinete com visual profissional;

6. Parafusos utilizados para montar e fixar o MDF da estrutura e suportes internos, sem necessidade de buchas. Essenciais para a montagem do gabinete;

7. Conjunto usado especialmente para fixar joystick, botões e componentes mecânicos do painel de controle, garantindo firmeza nas peças que serão pressionadas constantemente;

8. Sistema de áudio que ficará embutido no gabinete, proporcionando som ambiente durante os jogos, criando imersão e experiência retrô;

9. Controladora USB + botões e joystick padrão arcade. Permite a conexão dos comandos diretamente ao computador, Raspberry ou placa, sem latência (resposta imediata ao jogador);

10. O computador será o cérebro do arcade, responsável por rodar os jogos, gerenciar os controles e exibir os gráficos na tela com desempenho estável e rápido;

11. Tela principal do arcade, onde os jogos serão exibidos. Pode ser um monitor comum reaproveitado ou um modelo específico dependendo do tamanho da arte e do gabinete;

12. Material utilizado para fixar e reforçar conexões elétricas (ex: fios dos botões, LEDs ou pequenos reparos na montagem), garantindo bom contato elétrico;

13. Cabo de rede para fazer a ligação do botão power do arcade para o computador.

4.6 Cronograma do Trabalho

Segue abaixo a tabela com o cronograma de trabalho das atividades realizadas ao longo desse trabalho:

Atividades	Meses						
	Jan	Fev	Mar	Abr	Mai	Jun	Jul
1				X			
2					X		
3					X		
4						X	
5						X	
6						X	

1. Descrição da atividade 1: Iniciamos o reparo do hardware do arcade;
2. Descrição da atividade 2: Estávamos enfrentados o problema do jogo, analisando possibilidades para a apresentação;
3. Descrição da atividade 3: Iniciamos a construção do jogo Arcade Fight;
4. Descrição da atividade 4: Fizemos o reparo da estrutura física do arcade;
5. Descrição da atividade 5: Fizemos os testes do jogo e da estrutura, onde ficamos jogando para achar bugs e testar os botões;
6. Descrição da atividade 6: Início e conclusão da documentação e apresentação.

5 CONCLUSÕES E RECOMENDAÇÕES

Os gabinetes de jogos clássicos eram historicamente limitados a conexões locais e hardwares analógicos obsoletos. Diante dessa perspectiva, unificar as áreas de hardware, software e redes para redefinir a experiência clássica do fliperama, aliando a nostalgia dos jogos retrô às demandas contemporâneas de conectividade global é um grande desafio.

Esse trabalho teve o objetivo geral de aprimorar o fliperama existente no IFSP Campus São João da Boa Vista, a fim de beneficiar os alunos da instituição. A fim de alcançar o objetivo principal foi determinado três objetivos específicos.

O primeiro foi aprimorar os componentes de hardware do arcade existente. Para tanto foi feito o diagnóstico e identificou-se as configurações do hardware, para então substituir alguns dos componentes da placa-mãe.

No objetivo específico dois ocorreu a substituição do sistema operacional do fliperama que anteriormente era o Windows 10 e passou a ser o Linux Mint.

No terceiro objetivo específico desenvolveu-se o jogo online Arcade Fight (criaram-se o front-end, banco de dados e back-end) e então implementou-se o game na máquina física.

Através da realização dos objetivos específicos houve a demonstração da maneira como foi realizado o aprimoramento do fliperama existente no IFSP Campus São João da Boa Vista, a fim de beneficiar os alunos da instituição. Desse modo os alunos da instituição citada foram beneficiados. Com isso o objetivo geral do trabalho foi atendido.

Ao longo do trabalho encontraram-se pontos negativos e positivos. Os fatores positivos foram: ótima organização dos professores em relação ao trabalho e a colaboração entre os membros do grupo do projeto Arcade. Já os pontos negativos foram: infraestrutura da sala do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo Câmpus São João da Boa Vista onde foi realizado o trabalho; e dificuldades para desenvolver e implementar o jogo Arcade Fight na máquina.

Para realização de trabalhos futuros se recomenda que a infraestrutura do local onde será feito o trabalho seja adequada para atender as necessidades das atividades desenvolvidas ao longo dele. Além disso é aconselhável que os alunos recebam mais ajuda no desenvolvimento game, além do auxilio que já foi oferecido.

REFERÊNCIAS

DUCKETT, J. **HTML e CSS: projeta e constrói websites**. 1. ed. Rio de Janeiro: Alta Books, 2014. Disponível em: <https://www.altabooks.com.br/>. Acesso em: 24 de jun. de 2026. 10

ELECTRON. **Electron Documentation**. 2026. Disponível em: <https://www.electronjs.org/>. Acesso em: 24 de jun. de 2026. 10

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**. 6. ed. São Paulo: Pearson Addison Wesley, 2011. Disponível em: <https://br.pearson.com/>. Acesso em: 24 de jun. de 2026. 11

FASTAPI. **FastAPI Documentation**. 2026. Disponível em: <https://fastapi.tiangolo.com/>. Acesso em: 24 de jun. de 2026. 11

FLANAGAN, D. **JavaScript: O Guia Definitivo**. 6. ed. Porto Alegre: Bookman, 2012. Disponível em: <https://books.google.com.br/>. Acesso em: 24 de jun. de 2026. 10

LUTZ, M. **Aprendendo Python**. 4. ed. Porto Alegre: Bookman, 2014. Disponível em: <https://books.google.com.br/>. Acesso em: 24 de jun. de 2026. 11

MASSE, M. **REST API Design Cookbook**. 1. ed. Sebastopol: O'Reilly Media, 2012. Disponível em: <https://www.oreilly.com/>. Acesso em: 25 de jun. de 2026. 11

MINT, L. **Linux Mint User Guide**. 2026. Disponível em: <https://linuxmint.com/>. Acesso em: 21 de jun. de 2026. 9

NEMETH, E. et al. **Manual de Administração de Sistemas Linux**. 4. ed. Porto Alegre: Bookman, 2015. Disponível em: <https://books.google.com.br/>. Acesso em: 21 de jun. de 2026. 9, 12

PODMAN. **Podman Documentation**. 2026. Disponível em: <https://podman.io/>. Acesso em: 23 de jun. de 2026. 12

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software: uma abordagem profissional**. 9. ed. Porto Alegre: AMGH, 2021. Disponível em: <https://www.grupogen.com.br/>. Acesso em: 25 de jun. de 2026. 10

SEBESTA, R. W. **Conceitos de Linguagens de Programação**. 11. ed. Porto Alegre: Bookman, 2018. Disponível em: <https://books.google.com.br/>. Acesso em: 25 de jun. de 2026. 11

SHOTTS, W. **A Linha de Comando Linux: Uma Introdução Completa**. 1. ed. São Paulo: Novatec, 2020. Disponível em: <https://novatec.com.br/>. Acesso em: 23 de jun. de 2026. 11

SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. **Fundamentos de Sistemas Operacionais**. 9. ed. Rio de Janeiro: LTC, 2015. Disponível em: <https://www.kufunda.net/>. Acesso em: 21 de jun. de 2026. 8, 9

SILVEIRA, F.; SILVA, M. **Bancos de Dados Relacionais com PostgreSQL**. 1. ed. São Paulo: Novatec, 2023. Disponível em: <https://novatec.com.br/>. Acesso em: 25 de jun. de 2026. 11

STALLINGS, W. **Arquitetura e Organização de Computadores**. 10. ed. São Paulo: Pearson Education do Brasil, 2017. Disponível em: <https://br.pearson.com/>. Acesso em: 20 de jun. de 2026. 7, 8

TANENBAUM, A. S.; AUSTIN, T. **Organização Estruturada de Computadores**. 6. ed. São Paulo: Pearson Education do Brasil, 2013. Disponível em: <https://br.pearson.com/>. Acesso em: 20 de jun. de 2026. 7

TANENBAUM, A. S.; BOS, H. **Sistemas Operacionais Modernos**. 4. ed. São Paulo: Pearson Education do Brasil, 2016. Disponível em: <https://br.pearson.com/>. Acesso em: 21 de jun. de 2026. 8, 9

TERUEL, E. C. **Desenvolvimento Web com HTML5, CSS3 e JavaScript**. 1. ed. São Paulo: Érica, 2014. Disponível em: <https://www.saraiva.com.br/>. Acesso em: 25 de jun. de 2026. 9

TORRES, G. **Hardware Curso Completo**. 4. ed. Rio de Janeiro: Axcel Books, 2001. Disponível em: <https://biblioteca.ifpe.edu.br/bib/34157>. Acesso em: 20 de jun. de 2026. 7, 8

WALSH, D. **Podman in Action: Secure, daemonless container management**. 1. ed. Shelter Island: Manning Publications, 2022. Disponível em: <https://www.manning.com/>. Acesso em: 23 de jun. de 2026. 12