VOL. 7
2025

Revista HACKOONSPACE

Projeto Hackerspace
UFSCar Sorocaba

Apresentação dos artigos e projetos realizados na edição 2025 do HackoonSpace

Universidade Federal de São Carlos
Campus Sorocaba

PROEX
Pró-Reitoria de Extensão

Revista Hackoonspace

Vol. 7

Projeto Hackoonspace
2025

Apresentação

O Projeto Hackoonspace é um projeto de extensão promovido pela UFSCar Campus Sorocaba e realizado na forma de encontros que discutem a contracultura hacker, apresentando personagens, acontecimentos, aspectos socioculturais, artefatos e atividades com a finalidade de que os participantes sejam expostos à contracultura em questão e desenvolvam um projeto ou material acerca da mesma.

A principal intenção do projeto é desmistificar o conceito e figura do hacker, enquanto proporcionando um espaço de exposição, produção e compartilhamento de conteúdo que se configura como dentro da contracultura de forma que participantes de diferentes níveis de conhecimento técnico possam participar.

Esta revista tem como objetivo divulgar os trabalhos desenvolvidos pelos alunos participantes do Projeto Hackerspace no ano de ***2025****. Esperamos que esta revista possibilite a difusão dos conhecimentos adquiridos na atividade para a comunidade em geral. Gostaríamos de agradecer todos os alunos que contribuíram com artigos e projetos para esta edição da revista.*

Organização e edição:
Ana Luiza Salgado de Paula
Fernando Favareto Abromovick
Bianca Ribeiro Barros
Jean Rodrigues Rocha

Supervisão:
Gustavo M. D. Vieira

Conteúdo

Parte I

Artigos

Capítulo 1

Comparando LLMs e suas limitações

João Pedro Moro Bolognini
Renan Aparecido de Almeida

OBS: Para ver o artigo na íntegra, com imagens maiores, clique **aqui.**

1.1 Introdução

Modelos de linguagem são, em sua essência, preditores de texto que utilizam regras e/ou abordagens probabilísticas para adivinhar a próxima palavra dada uma frase. Os avanços recentes na área de aprendizado de máquina introduziram atenção ao contexto nos modelos neurais, o que iniciou uma nova era no problema de previsão de palavras. Essa nova arquitetura possibilitou o treinamento de modelos mais capazes e a sua escalabilidade [7]. Hoje vive-se o fruto desses avanços com modelos de larga escala, onde seus bilhões de parâmetros são utilizados para gerar textos extremamente semelhantes aos escritos por humanos. A utilidade pratica dos modelos de linguagem em larga escala (LLMs) é ampla e crescente. Elas são empregadas em assistentes virtuais, sistemas de recomendação, motores de busca, plataformas educacionais, ambientes corporativos e até em ambientes criativos e artísticos. Ao serem incorporadas em produtos como o *ChatGPT*, *Claude*, *Copilot* ou *Gemini*, essas ferramentas não apenas respondem perguntas, mas colaboram ativamente na resolução de problemas, ajudam a escrever e revisar textos, programam trechos de código e apoiam a tomada de decisão. Esse novo paradigma representa uma mudança de foco: de sistemas que apenas processam informação para sistemas que criam com base no conhecimento aprendido.

Contudo, esses sistemas continuam atuando como preditores de texto, sem realmente possuir um entendimento real do conhecimento que é reproduzido [1], [3]. Isso levanta o questionamento da confiabilidade das respostas e textos criados por esses modelos. Neste contexto, o presente artigo se dedica à análise comparativa dos principais modelos da atualidade. Avaliando a confiabilidade de suas respostas no idioma português brasileiro, e em tarefas lógicas, matemáticas e linguísticas. Dentre a gama de modelos disponíveis, esse artigo se propõe a comparar um conjunto de seis modelos: *GPT-4.0* (*OpenAI*), *Claude Sonnet 4* (*Anthropic*), *Gemini 2.5 Flash* (*Google*), *LLaMA 4* (*Meta*), *DeepSeek R1* (*DeepSeek*) e o *Copilot* (*Microsoft*). Cada um desses modelos foi desenvolvido por empresas com diferentes abordagens, objetivos e filosofias de projeto. O *GPT-4.0*, por exemplo, é reconhecido por sua alta consistência e profundidade de raciocínio em tarefas complexas, sendo amplamente usado em contextos profissionais e acadêmicos 4]. O *Claude Sonnet 4*, por sua vez, adota uma abordagem centrada em segurança, sensatez e equilíbrio ético, destacando-se em tarefas que exigem cuidado com ambiguidade ou sensibilidade de conteúdo. O *Gemini 2.5 Flash*, da *Google*, prioriza a velocidade de resposta e integração com dados atualizados da web, sendo ideal para aplicações em tempo real 5]. O *LLaMA 4*, com sua proposta

open source e foco em leveza e eficiência, visa democratizar o acesso a IA de ponta 6]. Já o *DeepSeek R1* é um modelo chinês que chamou a atenção por seu desempenho matemático e lógica formal 2]. Por fim, o *Copilot* é uma LLM especializada, que atua como assistente de codificação e produtividade, integrado ao ecossistema *Microsoft*.

Essas diferenças decorrem de escolhas fundamentais de projeto: os dados usados no treinamento, os filtros aplicados, o ajuste fino (*fine-tuning*), os mecanismos de segurança, o alinhamento com valores humanos e ate os *trade-offs* feitos entre qualidade e velocidade. Esse tipo de variação não é apenas técnico — é também estratégico, refletindo os diferentes públicos, mercados e finalidades para os quais cada modelo foi desenvolvido. Diante disso, a motivação deste trabalho é realizar um *survey* sistemático e comparativo, que avalie os pontos fortes e limitações de cada LLM quando confrontados com uma variedade de desafios. Estudos comparativos como este oferecem uma base valiosa para decisões informadas, seja na escolha da tecnologia mais adequada para um produto, na seleção de um assistente virtual mais ético, ou na adoção de um modelo mais eficiente para tarefas específicas. Ao revelar os pontos fortes e os limites de cada modelo, este artigo contribui para o avanço de uma inteligência artificial mais transparente, segura e eficaz.

1.2 Avaliando os modelos

Para avaliar o desempenho dos LLMs, utilizaremos ate 3 métricas em cada desafio, podendo variar dependendo do tipo de raciocínio esperado. Uma das métricas é a correção das respostas obtidas, sendo baseada em uma escala de porcentagem que considera a quantidade e magnitude de inconsistências observadas. Com base nisso, pode-se ter uma ideia de qual modelo chegou mais perto da resposta correta, ou qual, mesmo tendo uma resposta errada, teve alguma veracidade nela. Vale lembrar que são consideradas como corretas as respostas base conhecidas pelos autores deste trabalho, também podendo conhecer alguma nova a partir das LLMs.

Correção	Nível de resposta obtido
100%	resposta correta
75%	resposta correta com pequenas inconsistências
50%	resposta correta com muitas inconsistências
25%	resposta errada com alguns pontos corretos
0%	resposta errada

TABLE I: Níveis de Correção das respostas

Outra métrica utilizada para comparação é o tempo de geração da resposta. Esse tempo é contado a partir do momento em que é feito o *input* para o LLM até o momento em que ela termina de enviar a resposta; dessa forma, é padronizada da forma justa a todos os modelos.

Em relação à métrica de fluxo de resolução, isso pode depender de modelo para modelo, mas vai ser observado principalmente a construção da resposta, se existe uma lógica para pensar e trazer ao usuário ou apenas é a resposta sem resolução.

A. Desafios no campo lógico

Tarefas complexas costumam exigir encadeamento de raciocínios e informações relevantes do contexto na elaboração de soluções adequadas. Quando submetidos a tarefas que necessitem de lógica, é esperado que modelos de linguagem considerem as limitações e detalhes descritos para fornecer uma resposta à tarefa solicitada. Assim, dois problemas clássicos de lógica foram utilizados para avaliar a capacidade dos LLMs neste campo: Travessia da Alface, lobo e ovelha; Medições com Jarras de Água. No primeiro problema, temos 3 elementos que apresentam relação de exclusividade entre si, limitando as possibilidades de travessia. O objetivo é transportar todos os itens, tendo a capacidade de transportar apenas 1 por vez. Na resolução desse problema, é esperado que essas limitações sejam consideradas no desenvolvimento do raciocínio, a fim de encontrar a sequência de passos que permite a travessia dos 3 elementos sem violar as relações de exclusividade.

O segundo problema compreende a medição de um intervalo de volumes de água utilizando duas jarras de 4 e 9 litros. Tal desafio necessita de um certo nível de criatividade e raciocínio lógico para elaborar as sequências de passos corre-

tas para cada medição. No problema submetido aos modelos, foi pedido o cálculo das medições de 1 até 9 litros. Abaixo se encontram os resultados dos experimentos no campo lógico, contendo a correção e o tempo de resposta medido de cada modelo:

Alface, Ovelha e Lobo

Model	Tempo de Resposta (s)	Correção
GPT-4o	10.44	100%
Gemini 2.5 flash	8.72	100%
Claude Sonet 4	14.21	100%
Llama 4	11.65	100%
DeepSeek-R1	68.89	100%
Copilot	4.36	100%

Jarras de Água

Model	Tempo de Resposta (s)	Correção
GPT-4o	35.34	75%
Gemini 2.5 flash	12.67	100%
Claude Sonet 4	20.58	50%
Llama 4	26.12	75%
DeepSeek-R1	130.96	75%
Copilot	6.70	25%

TABLE II: Avaliação dos modelos no campo logico

Para o primeiro problema, o *Copilot* teve o melhor tempo médio, sendo quase a metade do tempo médio do modelo com o segundo menor tempo médio. Porém, em relação ao fluxo de raciocínio e explicação, ele deixou a desejar quando comparado com os demais fluxos, já que ele foi muito rápido na explicação, enquanto o resto dos modelos não poupou detalhes. Em relação a correção, todos se saíram iguais, com 100% de correção, o que implica que todos são capazes de responder a desafios lógicos simples, com cada um tendo seu jeito.

Observando o segundo problema, é perceptível a dificuldade enfrentada pelos modelos ao gerar uma resposta para um desafio lógico que necessite de um encadeamento de informações mais complexo. Apenas o *Gemini 2.5* respondeu de forma satisfatória com a sequência de passos correta para realizar a medição dos diferentes volumes de água. Os demais modelos apresentam algumas inconsistências com relação aos volumes que necessitam de uma sequência de passos mais elaboradas para serem medidos. Sobre essas inconsistências apresentadas, o modelo de linguagem *Claude* apresentou um numero considerável em suas respostas, enquanto o *Copilot* forneceu algumas respostas que continham falhas de lógicas fundamentais. Isso demonstra que, apesar de serem capazes de processar lógicas simples, em fluxos de raciocínio mais complexos, os modelos ainda possuem certa dificuldade e cometem pequenos erros.

B. Desafios no campo matemático

A explicabilidade dos LLMs ainda é uma questão estudada dentro da área, mas suas habilidades se aplicam a diferentes cenários. Apesar dos modelos de linguagem serem treinados para gerar texto a partir da predição de palavras, uma dessas habilidades observadas e a resolução de problemas matemáticos. Tais modelos são capazes de construir respostas para equações e problemas puramente matemáticos, contudo não existe uma garantia de que eles tenham um entendimento dos conceitos ali aplicados na resolução. Investigando essa limitação, os modelos em analise foram submetidos a 2 problemas. Diferente do campo lógico, esses testes visam apenas avaliar o emprego dos conceitos matemáticos na resolução de equações, sem a necessidade de um entendimento do problema ou contexto.

No primeiro cenário, o problema envolve o cálculo de uma derivada pelo conceito da regra da cadeia. Para o segundo cenário, espera-se que o modelo calcule as raízes de uma equação de segundo grau a partir da formula quadrática. Os problemas são perguntados de forma direta e simples, o que permite observar quais modelos incorporam os passos de resolução na resposta entregue ao usuário e se esses passos tem coerência com a solução esperada.

Derivadas - Regra da Cadeia

Model	Tempo de Resposta (s)	Correção
GPT-4o	17.50	100%
Gemini 2.5 flash	4.18	100%
Claude Sonet 4	9.29	100%
Llama 4	7.50	100%
DeepSeek-R1	24.35	100%
Copilot	6.67	75%

Polinômio Quadrático

Model	Tempo de Resposta (s)	Correção
GPT-4o	13.25	100%
Gemini 2.5 flash	4.77	25%
Claude Sonet 4	14.32	100%
Llama 4	7.56	100%
DeepSeek-R1	31.54	100%
Copilot	5.26	100%

TABLE III: Avaliação dos modelos no campo matemático

A capacidade dos LLMs de aplicar conceitos matemáticos e resolver equações é notável. Em quase todos os testes realizados, eles demonstraram o uso das técnicas esperadas e chegaram à solução correta. Um caso especial ocorreu com o *Copilot*, que, apesar de chegar à resposta esperada, forneceu um fluxo de resolução raso e com algumas inconsistências nas notações. Os demais modelos fizeram o uso correto de notações matemáticas para ilustrar os passos no fluxo de raciocínio para a resolução do problema.

Outro caso interessante de se observar foram as respostas do *Gemini 2.5* para as equações polinomiais, onde em algumas ocasiões o modelo recusou-se a fornecer uma resposta direta para a pergunta. Em sua resposta, o modelo alegou incapacidade para computar tal expressão e logo em seguida informou a forma correta para se chegar à resolução enquanto incentivava o usuário a encontrar a solução por si mesmo. Contudo, em alguns outros casos, o *Gemini 2.5* seguiu com a resposta esperada e computou o resultado dos polinômios corretamente.

C. Desafios no campo linguístico

Contexto e sentido de um texto são aspectos que influenciam fortemente o entendimento de tarefas e problemas. No campo linguístico da língua portuguesa, existem certas dificuldades com relação a construção e características da língua que podem confundir os modelos de linguagem. Além de recursos como ironias, quebras de expectativas ou subversão de lógica que agravam a dificuldade de compreensão da língua por uma maquina. Considerando isso, esse campo foi o mais explorado dentro desta analise comparativa, contendo 3 desafios que explorem características distintas.

O primeiro desafio linguístico consiste em uma série de 6 perguntas, as quais estão interligadas com as anteriores, assim, sendo visto se o modelo é capaz de identificar os fatos subsequentes que estão interligados. é perceptível um tom cômico nas perguntas, onde elas não possuem um sentido lógico ou coerência direta. É exatamente essa ausência de lógica e racionalidade que pode confundir alguns modelos que não prestam tanta atenção no contexto linguístico incorporado ao fluxo de raciocínio para computar as respostas. As 6 perguntas e as respostas esperadas que foram utilizadas para analisar os LLMs foram:

- 1) Um avião está transportando 500 tijolos, porém 1 tijolo cai dele, quantos tijolos sobram? 499;
- 2) Como que faz para colocar um elefante na geladeira? Primeiro você abre a geladeira, depois coloca o elefante e depois fecha a geladeira;
- 3) Como que faz para colocar uma girafa na geladeira? Primeiro você abre a geladeira, aí você tira o elefante de dentro, coloca a girafa e fecha a geladeira;
- 4) O leão deu uma festa para todos os animais da selva, porém um animal não compareceu, qual animal e por quê? A girafa, pois ela ainda está dentro da geladeira;
- 5) Uma moça precisa passar pelo rio onde os jacarés moram e sempre estão famintos, porém, ela não vai morrer para eles, você sabe por quê? Pois eles estão na festa do leão;
- 6) Porém, durante a travessia, a moça morre, por qual motivo? Ela foi atingida por um tijolo que caiu do céu.

Sobre o segundo problema linguístico oferecido aos modelos, foi utilizado um *script* pedindo que ensinassem 2 receitas. As receitas solicitadas foram brigadeiro e *strogonoff* de frango, como se fossem chefes de cozinha e estivessem ensinando seus alunos no início do curso de culinária. O intuito desse desafio é verificar a lógica de construção para o aprendizado do usuário, já que muitas das vezes os modelos dão as mesmas respostas, porém o usuário não absorve da mesma maneira. Esse desafio também visa observar o entendimento de medidas e quantidades dos ingredientes, e como os modelos incorporam esses elementos na construção das informações presentes na resposta.

O terceiro desafio linguístico a qual os modelos foram submetidos consiste de uma simples pergunta de percepção de contexto. A pergunta fornecida para todos os modelos foi: "O pai de João tem 5 filhos: Zazá, Zezé, Zizí, Zozó e?". Este teste serve para ver se o modelo responde de maneira extremamente rápida seguindo o padrão observado, ou se ele faz uso das informações do contexto na construção de uma resposta mais elaborada.

Tijolo e Animais

Model	Tempo de Resposta (s)	Correção
GPT-4o	5.85	75%
Gemini 2.5 flash	3.73	50%
Claude Sonet 4	8.67	100%
Llama 4	5.98	100%
DeepSeek-R1	30.88	100%
Copilot	2.69	75%

Receitas

Model	Correção
GPT-4o	100%
Gemini 2.5 flash	100%
Claude Sonet 4	100%
Llama 4	100%
DeepSeek-R1	100%
Copilot	100%

Os 5 Filhos

Model	Tempo de Resposta (s)	Correção
GPT-4o	3.26	100%
Gemini 2.5 flash	3.72	100%
Claude Sonet 4	5.35	100%
Llama 4	1.96	100%
DeepSeek-R1	10.79	100%
Copilot	2.09	100%

TABLE IV: Avaliação dos modelos no campo linguístico

Analisando as respostas para os diferentes problemas linguísticos, é perceptível a capacidade dos modelos de lidar com vários contextos e informações. No primeiro desafio, os LLMs demonstraram entendimento do contexto e incorporaram no fluxo de informações para gerar as respostas esperadas. Em alguns casos, foram observadas algumas inconsistências, principalmente no final da sequência de eventos, onde os modelos *GPT-4o* e *Gemini 2.5* desviavam a atenção do contexto construído e respondiam baseadas na lógica comum.

Observando as respostas para as receitas solicitadas, todos os modelos forneceram respostas satisfatórias, incluindo medidas e unidades para os ingredientes. Alguns ainda adicionaram dicas e pontos importantes do modo de preparo que necessitam de certa atenção, mesmo as receitas sendo de complexidade relativamente simples. Seguindo para a análise do terceiro desafio proposto, todos os modelos rapidamente entenderam o contexto e responderam corretamente a questão apresentada. Apesar do tempo de resposta relativamente veloz, nenhum deles seguiu com a lógica simples de terminação fonética dos nomes e se ativeram as informações que tinham disponíveis. Isso fica mais evidente observando o processo de *Reasoning* que o *DeepSeek* realiza na construção das respostas, onde ele inclui no processo uma geração intermediaria para identificação e organização das informações antes de gerar a resposta final para o usuário. Neste processo é possível observar a consideração da sequência fonética, sendo descartada ao modelo analisar a informação presente no contexto.

1.3 Sobre os resultados e conclusão

Comparando os resultados obtidos nos 3 campos analisados, é notável o avanço e a capacidade demonstrada dos modelos de linguagens quando apresentados a tarefas complexas. Os modelos conseguem construir soluções para os casos mais simples, considerando informações do contexto no processo de raciocínio para fornecer respostas satisfatórias.

Entretanto, em uma parcela dos casos observados, os resultados produzidos pelos modelos apresentam certas inconsistências ou pontos incoerentes com a solução esperada. Isso pode ser observado principalmente no campo lógico, onde, quando submetidos a um problema que exige um nível de complexidade maior, os modelos falham em manter o fluxo de raciocínio esperado. Tal falha demonstra que os modelos modernos, ainda que pareçam muito inteligentes, atuam simplesmente como preditores de palavras com base em probabilidades. Não sendo capazes de realmente processar de maneira lógica as perguntas, mas sim de construir respostas satisfatórias para problemas bem evidenciados e discutidos no material de aprendizado.

A resolução de desafios puramente matemáticos, que consistem apenas na aplicação de conceitos e resolução de equações, é bem resolvida pelos modelos. Esse não é um desafio novo para a área da computação, que já conta com outras soluções e modelos focados nesse tópico específico. Contudo, a aplicação de modelos de linguagem para esta tarefa é interessante, obtendo desempenho satisfatório nos testes que foram executados. Uma observação interessante é com relação às respostas fornecidas pelo *Gemini 2.5*, que alegou incapacidade de resolver as equações em alguns casos. Essas respostas podem ser fruto de algum filtro aplicado ao modelo, visto que ele providencia as ferramentas necessárias para a resolução e, em outros casos, ele prossegue com a resolução completa da equação solicitada.

No campo linguístico, todos os modelos obtiveram desempenho satisfatório, com apenas algumas pequenas falhas na resolução em casos isolados. No geral, eles demonstraram capacidade de atenção ao contexto e incorporaram as informações fornecidas para gerar as respostas. O teste das receitas avaliou o uso de medidas e unidades para notação dos ingredientes e passos necessários, onde novamente os LLMs produziram respostas coerentes correspondendo ao esperado do teste. Os pontos de falha se concentram apenas no primeiro teste, mais especificamente em sua parte final, onde alguns dos modelos analisados ignoram a linha de raciocínio construída com o contexto completo e seguem uma linha lógica simples para fornecer a resposta.

Em relação ao tempo para geração das respostas, é notável a diferença entre a velocidade de processamento dos modelos e a abordagem utilizada na geração de texto. O *DeepSeek R1* demonstra o maior tempo tomado na construção das suas soluções, grande parte devido a sua etapa de *reasoning* incluída no processo de resolução. Sobre o modelo mais veloz, o *Copilot* consegue retornar uma solução no menor tempo, principalmente pelo tamanho do modelo ser inferior aos demais analisados, mas ao mesmo tempo produzindo respostas inexatas em alguns casos. Essas comparações relacionadas ao tempo podem ser observadas no gráfico abaixo:

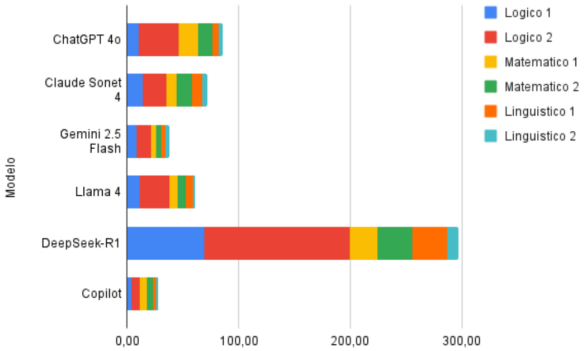


Fig. 1: Tempo de Geração de resposta por problema

Concluindo o experimento, pode-se notar a surpreendente capacidade de aplicação dos LLMs para diferentes contextos e campos além dos previstos inicialmente em sua concepção. Suas habilidades são melhor aproveitadas dentro do campo linguístico, mas para alguns problemas simples e bem documentados, eles são capazes de prover uma solução adequada.

Contudo, em tarefas complexas e mais elaboradas, os resultados obtidos não são muito confiáveis, visto que os modelos não compreendem de fato as tarefas ou as respostas que eles fornecem. Em cenários similares durante os testes, foram observadas diversas inconsistências menores na solução obtida e, em alguns casos, erros fundamentais que afetam a confiabilidade de tal solução.

1.4 Ameaças à validade

Devido ao estudo proposto analisar as respostas de maneira qualitativa, a questão da correção medida nas soluções fornecidas pelos modelos é subjetiva aos entendimentos dos autores. Em alguns testes, a resposta esperada é fácil de se analisar, dado que os problemas utilizados são relativamente bem explorados ou possuem métodos para verificar equivalência na resolução (caso do campo matemático). Contudo, essa característica subjetiva da analise dos resultados ainda é uma questão a ser considerada quanto à validade e relevância das informações avaliadas. Outro ponto a ser considerado sobre o estudo é sua amplitude. Os testes foram realizados com uma variedade de modelos; contudo, a gama de testes e campos de conhecimentos testados não é extremamente ampla. Neste estudo, os desafios submetidos aos modelos se restringiram apenas ao campo linguístico, lógico e matemático. Tal amplitude pode ser expandida por estudos futuros para incluir outros campos de conhecimento ou ainda ampliar a gama de testes realizados nesses campos.

1.5 Bibliografia

[1] Nicholas asher, swarnadeep bhar, akshay chaturvedi, julie hunter, and soumya paul. limits for learning with language models. 6 2023.

[2] Deepseek-ai, daya guo, dejian yang, haowei zhang, junxiao song, ruoyu zhang, et al. deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. 1 2025.

[3] James huckle and sean williams. easy problems that llms get wrong. lecture notes in networks and systems, 1283 lnns:313–332, 2025.

[4] Openai, josh achiam, steven adler, sandhini agarwal, lama ahmad, ilge akkaya, et al. gpt-4 technical report. 3 2023.

[5] Gemini team, rohan anil, sebastian borgeaud, jean-baptiste alayrac, jiahui yu, radu soricut, johan schalkwyk, et al. gemini: A family of highly capable multimodal models. 12 2023.

[6] Hugo touvron, thibaut lavril, gautier izacard, xavier martinet, marie-anne lachaux, and timothee lacroix others. llama: Open and efficient foundation language models. 2 2023.

[7] Ashish vaswani, noam shazeer, niki parmar, jakob uszkoreit, llion jones, aidan n. gomez, Łukasz kaiser, and illia polosukhin. attention is all you need. advances in neural information processing systems, 30, 2017.

Capítulo 2

Padrões de Cibercriminalidade no Brasil: Uma Análise da Vulnerabilidade e do Comportamento do Usuário Individual

Ayrton Santos Bernardes

OBS: Para ver o artigo na íntegra, com imagens maiores, clique **aqui.**

Resumo — O presente artigo analisa os padrões de cibercriminalidade no Brasil, destacando a transição dos ataques puramente técnicos para a exploração da vulnerabilidade humana através da Engenharia Social. Em um cenário onde o Brasil figura como um dos líderes globais em fraudes digitais e uso de internet *banking*, o estudo investiga como a discrepância entre a alta conectividade e o baixo letramento digital da população cria um ambiente propício para o crime. Através de uma revisão bibliográfica que integra dados estatísticos de segurança (Axur, Febraban, Cetic.br), teorias da psicologia comportamental (Kahneman, Mitnick) e a legislação vigente (LGPD, Leis nº 12.965 e nº 14.155), demonstra-se que os cibercriminosos exploram gatilhos cognitivos — como urgência, medo e confiança — para induzir o usuário ao erro, operando no sistema de pensamento rápido e emocional. A análise aponta que, embora o arcabouço legal brasileiro tenha evoluído significativamente na tipificação penal e proteção de dados, ele atua de forma reativa. Conclui-se que a mitigação eficaz dos riscos cibernéticos no país depende imperativamente de estratégias de educação digital preventiva, visando transformar o usuário, atualmente o "elo mais fraco", em uma barreira consciente de segurança.

Palavras-chave: Cibercriminalidade. Engenharia Social. Segurança da Informação. Comportamento do Usuário. Legislação Digital.

Abstract — This paper analyzes cybercrime patterns in Brazil, highlighting the transition from purely technical attacks to the exploitation of human vulnerability through Social Engineering. In a scenario where Brazil stands as a global leader in digital fraud and internet banking usage, this study investigates how the gap between high connectivity and the population's low digital literacy creates a conducive environment for crime. Through a bibliographic review integrating security statistical data (Axur, Febraban, Cetic.br), behavioral psychology theories (Kahneman, Mitnick), and current legislation (LGPD, Laws No. 12.965 and No. 14.155), it is demonstrated that cybercriminals exploit cognitive triggers—such as urgency, fear, and trust—to induce user error by engaging the fast, emotional thinking system. The analysis indicates that while the Brazilian legal framework has evolved significantly regarding criminal typology and data protection, it remains largely reactive. It is concluded that effective mitigation of cyber risks in the country imperatively depends on preventive digital education strate-

gies aimed at transforming the user, currently the "weakest link", into a conscious security barrier.

Keywords: Cybercrime. Social Engineering. Information Security. User Behavior. Digital Legislation.

2.1 Introdução e motivação

A transformação digital acelerada nas últimas décadas redefiniu profundamente as relações sociais, comerciais e financeiras no Brasil. A onipresença de dispositivos móveis e a democratização do acesso à internet inseriram milhões de brasileiros no ambiente virtual, tornando o país um dos mais conectados do mundo. Segundo dados recentes da pesquisa TIC Domicílios (CETIC.BR, 2024), a internet já está presente na vasta maioria dos lares brasileiros, com o *smartphone* consolidado como o principal dispositivo de acesso. No entanto, essa hiperconectividade, embora benéfica para a inclusão digital e econômica, expandiu proporcionalmente a superfície de ataque para atividades ilícitas.

O Brasil figura consistentemente entre as nações com maiores índices de cibercriminalidade global. Relatórios de segurança digital (AXUR, 2024; SYMANTEC, 2024) apontam que o país não é apenas um alvo frequente de *malware* e *ransomware*, mas também um "laboratório"para fraudes financeiras sofisticadas. O avanço dos sistemas de pagamento instantâneo e a digitalização bancária, embora eficientes, tornaram-se vetores preferenciais para criminosos. Dados da Federação Brasileira de Bancos (FEBRABAN, 2024) indicam prejuízos bilionários anuais decorrentes de fraudes eletrônicas, evidenciando que a infraestrutura tecnológica, por mais robusta que seja, não é capaz de blindar totalmente o ecossistema digital.

Neste cenário, observa-se uma mudança de paradigma no *modus operandi* do cibercrime. Se outrora os ataques focaram na exploração de vulnerabilidades técnicas (falhas de *software* ou *hardware*), hoje o foco recai predominantemente sobre a vulnerabilidade humana. A Engenharia Social — técnica que manipula a psicologia do usuário para obter informações confidenciais — explora o que Mitnick e Simon (2003) definem como o "elo mais fraco"da segurança da informação: o indivíduo. O usuário brasileiro, muitas vezes dotado de baixo letramento digital em segurança, torna-se suscetível a táticas de persuasão, urgência e falsa autoridade.

Motivação e Justificativa

A motivação para este estudo reside na discrepância crítica entre a sofisticação das ameaças digitais e o comportamento de risco do usuário final. Embora o Brasil tenha avançado no arcabouço legal com a aprovação da Lei Geral de Proteção de Dados (Lei nº 13.709/2018) e o endurecimento de penas para crimes cibernéticos (Lei nº 14.155/2021), a lei, por si só, age de forma reativa e punitiva, não preventiva.

Justifica-se, portanto, a necessidade de analisar não apenas a tecnologia do crime, mas a "criminologia do clique". Compreender os gatilhos comportamentais que levam um indivíduo a compartilhar senhas, clicar em *links* de *phishing* ou realizar transferências para golpistas é essencial para o desenvolvimento de políticas públicas e estratégias de conscientização mais eficazes.

O presente artigo tem, assim, o objetivo de analisar os padrões de cibercriminalidade no Brasil sob a ótica da vulnerabilidade do usuário. Busca-se investigar como a intersecção entre a falta de cultura de segurança e as táticas de manipulação psicológica contribui para o êxito das fraudes digitais, propondo uma reflexão sobre a eficácia das atuais medidas de proteção e letramento digital.

2.2 Objetivo

Objetivo Geral

Analisar os principais padrões de cibercriminalidade vigentes no cenário brasileiro contemporâneo, estabelecendo uma correlação direta entre as táticas de ataque — predominantemente baseadas em Engenharia Social — e os comportamentos de risco e vulnerabilidades cognitivas do usuário individual.

Objetivos Específicos

Para alcançar o objetivo geral proposto, definem-se os seguintes objetivos específicos:

- **Mapear** as modalidades de crimes cibernéticos mais incidentes no Brasil (como *phishing*, *ransomware* e fraudes bancárias), utilizando dados estatísticos recentes de entidades públicas e privadas.
- **Identificar** os gatilhos psicológicos e as técnicas de manipulação exploradas pelos cibercriminosos para contornar mecanismos de segurança tecnológica.
- **Examinar** o perfil do usuário brasileiro sob a ótica do letramento digital, investigando se a rápida inclusão digital foi acompanhada de uma conscientização de segurança proporcional.
- **Discutir** a eficácia das atuais legislações brasileiras (LGPD e Lei nº 14.155/2021) frente à dinâmica volátil dos crimes digitais e o papel da educação digital como ferramenta preventiva.

2.3 Panorama da criminalidade do Brasil

A inserção do Brasil no cenário digital global ocorre de maneira ambígua: ao mesmo tempo em que o país se destaca pela rápida adoção de tecnologias e pelo alto volume de usuários conectados, ele se consolida como um dos ambientes mais hostis e ativos para a cibercriminalidade mundial. Para compreender a vulnerabilidade do usuário individual, é necessário primeiramente mapear o ecossistema de ameaças que opera no território nacional, caracterizado por alto volume de ataques e sofisticação crescente.

O Brasil como Alvo Global

O Brasil apresenta características demográficas e tecnológicas que o tornam extremamente atraente para agentes maliciosos. Com uma das maiores populações online do mundo e uma cultura de uso intensivo de redes sociais e comunicadores instantâneos, o país oferece uma vasta "superfície de ataque". Relatórios de segurança, como o *Internet Security Threat Report da Symantec* (2024), frequentemente posicionam o Brasil entre os líderes globais em origem e destino de ataques cibernéticos, evidenciando que o país não é apenas um alvo passivo, mas um polo ativo de cibercrime.

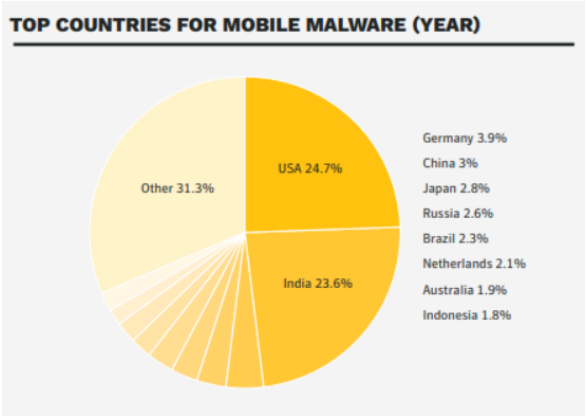


Figura 2.1: *Ranking* de países por incidência de *malware* móvel (2024)

Uma das principais razões para essa proeminência é a massificação do acesso via *smartphones*. Diferente de países desenvolvidos, onde a inclusão digital ocorreu primeiramente via computadores de mesa (que possuem *firewalls* e antivírus tradicionais), no Brasil, milhões de usuários tiveram sua primeira experiência digital via dispositivos móveis. Segundo dados da Axur (2024), o Brasil lidera rankings de detecção de *phishing* — páginas falsas criadas para roubar credenciais. A pesquisa aponta que a onipresença do celular facilita a disseminação desses golpes, uma vez que telas menores dificultam a verificação de URLs maliciosas e a usabilidade dos aplicativos prioriza a velocidade em detrimento de verificações de segurança detalhadas.

Portanto, a infraestrutura de conectividade brasileira, embora robusta em alcance, apresenta fragilidades críticas na ponta do usuário final, criando um ambiente propício para ataques de volume massivo e baixo custo de execução para os criminosos.

A "Indústria" da Fraude Financeira

Historicamente, o cibercrime passou por uma evolução motivacional: se nas décadas passadas o foco era o vandalismo digital (*defacement*) ou a demonstração de habilidade técnica, o cenário atual é dominado pela motivação puramente econômica. O cibercrime no Brasil industrializou-se, operando com estruturas hierárquicas, divisão de tarefas e foco na monetização rápida.

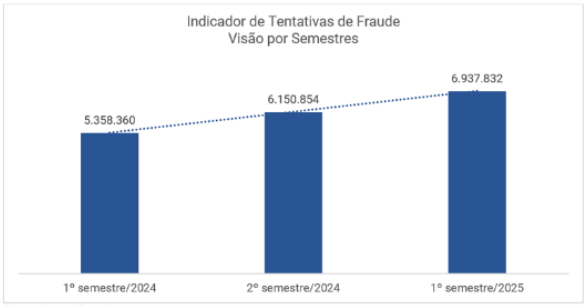


Figura 2.2: Indicador de Tentativas de Fraude no Brasil

O setor financeiro é o alvo primário dessa indústria. A digitalização bancária e a implementação do sistema de pagamentos instantâneos (Pix) revolucionaram a economia, mas também agilizam a consumação de fraudes. Antes, mecanismos de compensação bancária (como DOC ou TED) ofereciam janelas de tempo que permitiam, em alguns casos, a reversão de operações suspeitas. Com o Pix, a transferência de valores ocorre em tempo real, 24 horas por dia, o que é explorado pelos criminosos para pulverizar o dinheiro roubado em diversas contas "laranjas" em questão de minutos, dificultando o rastreamento e a recuperação de ativos.

Dados da Federação Brasileira de Bancos (FEBRABAN, 2024) indicam que as tentativas de fraude contra bancos e seus clientes resultam em prejuízos bilionários anualmente. O relatório destaca, contudo, um ponto crucial para a tese deste trabalho: a maioria esmagadora das fraudes bem-sucedidas não ocorre por violação dos sistemas de segurança dos bancos (ataques à infraestrutura), mas sim através da manipulação do cliente. A engenharia social tornou-se a ferramenta padrão para contornar a robustez tecnológica, provando que, na "indústria" da fraude financeira brasileira, investir em enganar o usuário é mais rentável do que tentar quebrar a criptografia bancária.

2.4 Engenharia social: A técnica de ataque

À medida que os sistemas de segurança cibernética evoluem, incorporando criptografia avançada, autenticação multifator e inteligência artificial para detecção de ameaças, observa-se um deslocamento estratégico no foco dos ataques. A barreira técnica tornou-se dispendiosa e complexa para ser transposta por força bruta. Em resposta, o cibercrime voltou-se para um alvo mais acessível e desprovido de atualizações de *software*: o operador humano. Esta seção analisa a Engenharia Social não apenas como um golpe, mas como uma técnica estruturada de exploração de falhas processuais e cognitivas.

Definição e Mecanismos: O "Human Hacking"

A Engenharia Social pode ser definida como a arte de manipular pessoas para que executem ações ou divulguem informações confidenciais. Diferente de um *exploit* técnico, que se aproveita de um erro no código de um programa, a engenharia social explora a programação comportamental e cultural do indivíduo. Mitnick e Simon (2003), em sua obra seminal sobre o tema, argumentam que a segurança da informação é uma corrente, e a segurança total é definida pela resistência do seu "elo mais fraco". Invariavelmente, o ser humano — propenso ao erro, à fadiga e, principalmente, à confiança — representa esse elo frágil.

O mecanismo central dessa técnica é o que se convencionou chamar de *human hacking*. O atacante não precisa descobrir uma falha *zero-day* no servidor de um banco se ele conseguir convencer o correntista a entregar sua senha voluntariamente. A eficácia da engenharia social reside na exploração da confiança inerente às relações sociais e na aversão ao conflito. O criminoso frequentemente assume uma posição

de autoridade (um gerente de banco, um policial, um suporte técnico) ou de familiaridade, criando um cenário onde a vítima sente que obedecer ou colaborar é a atitude lógica e segura a ser tomada, quando, na verdade, está comprometendo sua própria segurança (MITNICK; SIMON, 2003).

Portanto, o ataque técnico torna-se secundário; ele é apenas o meio para coletar o fruto da manipulação psicológica. A barreira de entrada para o cibercrime diminui drasticamente, pois não exige conhecimentos avançados de programação, mas sim habilidades de persuasão e manipulação social.

Vetores de Ataque Comuns e a Mobilidade

Para que a manipulação psicológica atinja a vítima, é necessário um canal de comunicação, tecnicamente denominado vetor de ataque. No cenário brasileiro, a predominância do uso de dispositivos móveis moldou a preferência dos criminosos por vetores que atinjam o usuário em seu *smartphone*.

Segundo dados da pesquisa TIC Domicílios (CETIC.BR, 2024), o celular é o dispositivo exclusivo de acesso à internet para a maioria da população das classes C, D e E. Essa dependência do *mobile* cria um ambiente propício para ataques rápidos, onde a interface reduzida dos aparelhos dificulta a verificação detalhada de remetentes e URLs.

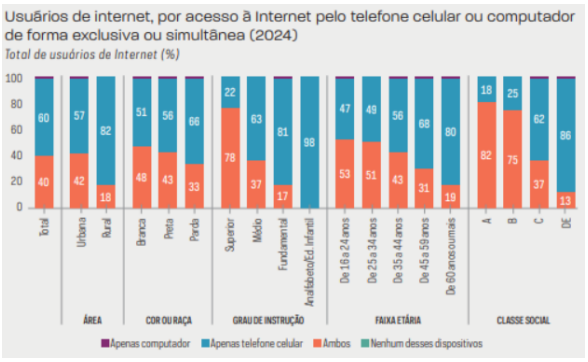


Figura 2.3: Usuários de internet no Brasil, por acesso à Internet pelo telefone celular ou computador de forma exclusiva ou simultânea (2024)

Os vetores mais incidentes incluem:

- **Phishing (E-mail)**: A forma mais tradicional, porém ainda altamente eficaz. Relatórios da Axur (2024) indicam que o *phishing* permanece como o vetor número um de fraudes no Brasil. A tática evoluiu de e-mails mal escritos para comunicações visualmente idênticas às de instituições reais, induzindo o usuário a clicar em links maliciosos que capturam credenciais ou instalam *malware*.

- **Smishing (SMS Phishing)**: Uma adaptação do *phishing* para mensagens de texto. Dada a taxa de abertura quase imediata de SMS, os criminosos utilizam este canal para mensagens de alta urgência, como "Sua compra de R$2.000,00 foi aprovada. Para cancelar, clique aqui". A brevidade do SMS impede análises mais profundas por parte da vítima.

- **Fraudes via WhatsApp e Redes Sociais**: Este vetor é particularmente devastador no Brasil devido à popularidade do aplicativo. Os ataques variam desde a clonagem de contas (solicitando dinheiro aos contatos da agenda) até a engenharia social direta, onde o criminoso cria um perfil falso (perfil *fake*) utilizando a foto da vítima e alega ter mudado de número. A natureza informal e imediata desses aplicativos desarma as defesas cognitivas do usuário, que tende a transferir valores acreditando estar ajudando um amigo ou familiar em emergência.

Esses vetores demonstram que a tecnologia serve apenas como o canal de entrega para a mensagem fraudulenta, enquanto o sucesso do ataque depende inteiramente da reação comportamental do usuário ao estímulo recebido.

2.5 Comportamento e vulnerabilidade do usuário

Se a Engenharia Social é a arma, a psique humana é o campo de batalha. A persistência e o sucesso dos crimes cibernéticos no Brasil não podem ser explicados apenas pela audácia dos criminosos, mas devem ser analisados sob a ótica de quem recebe o ataque. Esta seção investiga as raízes

cognitivas e educacionais que tornam o usuário brasileiro vulnerável, transformando decisões cotidianas em riscos de segurança.

Gatilhos Psicológicos: O Sistema 1 em Ação

A eficácia dos golpes digitais reside na capacidade do atacante de sequestrar o processo de tomada de decisão da vítima. Para compreender esse fenômeno, recorre-se à teoria do duplo processamento cognitivo apresentada pelo psicólogo e Nobel de Economia, Daniel Kahneman. Em sua obra "Rápido e Devagar" (2012), Kahneman descreve o cérebro humano operando em dois sistemas distintos:

- **Sistema 1**: Rápido, intuitivo, emocional e automático. É o modo que usamos para reações imediatas.
- **Sistema 2**: Lento, deliberativo, lógico e analítico. É o modo que usamos para cálculos complexos ou verificações detalhadas.

A estratégia central do cibercrime é forçar a vítima a operar exclusivamente no **Sistema 1**, impedindo que o **Sistema 2** seja ativado para verificar a veracidade da informação. Para isso, os criminosos utilizam gatilhos mentais específicos que exigem resposta imediata:

- **Medo e Urgência**: Mensagens como "Sua conta foi bloqueada preventivamente" ou "Seu nome será incluído no Serasa em 2 horas" geram um pico de ansiedade. O medo desabilita o pensamento crítico; a vítima age para estancar o risco percebido, clicando no *link* fraudulento sem checar o remetente.
- **Autoridade**: O uso de logotipos de instituições federais, tribunais ou grandes bancos explora o viés de obediência à autoridade. O usuário tende a cumprir ordens de supostos "auditores" ou "gerentes de segurança" sem questionar.
- **Curiosidade e Oportunidade**: Golpes que prometem benefícios exclusivos ("Você ganhou um Pix de R$500,00") ou apelam para a curiosidade social ("Veja as fotos da festa de ontem") exploram a impulsividade gratificante do cérebro.

Portanto, o clique no *link* malicioso não é necessariamente um ato de ignorância técnica, mas uma falha cognitiva induzida, onde a emoção do momento supera a lógica da segurança (KAHNEMAN, 2012).

Letramento Digital e "Fadiga de Segurança"

Além dos fatores psicológicos inatos, a vulnerabilidade do usuário brasileiro é exacerbada por uma lacuna educacional. Embora o Brasil tenha vivido uma rápida inclusão digital, esta não foi acompanhada de um letramento digital proporcional.

Dados da pesquisa TIC Domicílios (CETIC.BR, 2024) revelam que, para uma parcela significativa da população — especialmente nas classes C, D e E —, o acesso à rede é exclusivamente via dispositivo móvel. Esse perfil de uso, muitas vezes restrito a aplicativos de redes sociais ("Jardins Murados"), limita a compreensão do funcionamento da internet aberta e dos riscos associados a ela. O usuário sabe operar a interface do aplicativo, mas desconhece os princípios básicos de verificação de identidade, criptografia ou privacidade de dados. Existe acesso à ferramenta, mas não competência plena para o uso seguro.

Paralelamente, observa-se o fenômeno da "Fadiga de Segurança" (*Security Fatigue*). O usuário moderno é bombardeado diariamente por solicitações de atualização de senhas, autenticação em dois fatores, avisos de *cookies* e alertas de antivírus. O excesso de interrupções e a complexidade das medidas de proteção geram uma exaustão cognitiva. Como resultado, o usuário tende a desenvolver um comportamento de resignação ou indiferença, clicando em "Aceitar" ou "Ignorar" automaticamente apenas para prosseguir com sua tarefa, desativando alertas cruciais que poderiam prevenir um ataque.

Essa combinação de impulsividade emocional (Sistema 1), baixo letramento digital estrutural e exaustão mental cria o cenário perfeito para que o cibercrime prospere, independentemente das defesas tecnológicas implementadas pelas instituições.

2.6 O arcabouço legal e a proteção dos dados

Diante da escalada da cibercriminalidade e da vulnerabilidade do usuário, a resposta estatal brasileira manifestou-se através de uma rápida e progressiva evolução legislativa. O Direito Digital no Brasil, outrora incipiente, consolidou-se na última década na tentativa de criar um ambiente virtual mais seguro e juridicamente estável. Esta seção analisa como as leis brasileiras buscam cercar o problema, desde a tipificação criminal até a proteção preventiva de dados.

A Evolução Legislativa: Da Reação à Tipificação Específica

Historicamente, a legislação brasileira enfrentava dificuldades para enquadrar delitos virtuais nos tipos penais clássicos do Código Penal de 1940. Essa lacuna começou a ser preenchida de maneira reativa com a **Lei nº 12.737/2012**, conhecida popularmente como "Lei Carolina Dieckmann". Embora tenha sido um marco ao tipificar o crime de invasão de dispositivo informático, a lei possuía um escopo limitado e penas brandas, focando mais na intrusão técnica do que na fraude financeira massiva (BRASIL, 2012).

Em 2014, o **Marco Civil da Internet (Lei nº 12.965/2014)** estabelece os princípios, garantias, direitos e deveres para o uso da internet no Brasil, funcionando como uma "Constituição da Internet". Ele foi fundamental para definir a responsabilidade civil de provedores e a guarda de registros de conexão (logs), essenciais para a investigação criminal e identificação de autoria (BRASIL, 2014).

Contudo, foi com a **Lei nº 14.155/2021** que o Estado brasileiro endureceu severamente o tratamento ao cibercrime financeiro. Esta legislação alterou o Código Penal para agravar as penas do crime de furto qualificado e estelionato quando cometidos por meio eletrônico ou informático. A lei reflete a compreensão de que o crime digital não é de menor potencial ofensivo; pelo contrário, sua escala e impacto econômico exigem uma resposta punitiva mais rigorosa (BRASIL, 2021).

LGPD e a Responsabilidade das Empresas

Paralelamente à esfera penal, o Brasil avançou na proteção administrativa e civil dos dados pessoais com a **Lei Geral de Proteção de Dados (LGPD - Lei nº 13.709/2018)**. A relevância da LGPD para o estudo da cibercriminalidade é direta: a maioria dos ataques de engenharia social é alimentada por dados vazados. Criminosos utilizam nomes, CPFs, endereços e históricos de compras obtidos em vazamentos corporativos para dar verossimilhança aos seus golpes (*spear phishing*).

A LGPD impõe às empresas a responsabilidade objetiva sobre a segurança dos dados que custodiam. Ao exigir a implementação de medidas técnicas e administrativas de segurança, a lei visa "secar a fonte" de informações que nutre a indústria da fraude. O vazamento de dados deixou de ser apenas um incidente de TI para se tornar um passivo jurídico e reputacional, forçando o setor privado a investir em cibersegurança (BRASIL, 2018).

Limites da Lei: O Fator Humano como Barreira Final

Apesar do robusto aparato legislativo construído, observa-se um limite intrínseco à eficácia da norma jurídica: a lei é, por natureza, repressiva e *ex post facto* (atua após o fato). O endurecimento das penas pela Lei 14.155/2021 pode dissuadir alguns criminosos, mas não impede tecnicamente o envio de um *link* malicioso. Da mesma forma, a LGPD pune empresas que vazam dados, mas não conseguem recolher informações que já estão circulando na *Dark Web*.

O arcabouço legal não tem poder sobre o "clique" do usuário. A vulnerabilidade explorada pela engenharia social ocorre na esfera cognitiva, onde a lei não alcança. Se o usuário, movido por urgência ou falta de conhecimento, entrega voluntariamente suas credenciais bancárias, a barreira tecnológica e a barreira legal foram transpostas simultaneamente.

Conclui-se, portanto, que a legislação é uma ferramenta indispensável para a punição e regulação, mas insuficiente para a prevenção primária.

A proteção efetiva contra os padrões de cibercriminalidade atuais exige que a evolução legislativa seja acompanhada, com a mesma intensidade, por políticas de educação digital preventiva.

2.7 Conclusão

A análise dos padrões de cibercriminalidade no Brasil permite concluir que o país vive um paradoxo digital: possui uma infraestrutura de conectividade e digitalização bancária de vanguarda, mas sustentada por uma base de usuários com baixa maturidade em segurança da informação. O presente estudo demonstrou que, embora o volume de ataques técnicos (*malware*, invasões de servidor) continue relevante, a tônica da fraude contemporânea no Brasil deslocou-se para a Engenharia Social, onde a manipulação psicológica substitui a exploração de falhas de *software*.

Evidenciou-se que a onipresença dos dispositivos móveis e a facilidade das transações instantâneas, exemplificadas pelo Pix, criaram um ecossistema lucrativo para o cibercrime. A "indústria" da fraude, motivada puramente por ganhos econômicos, capitaliza sobre a vulnerabilidade cognitiva do indivíduo. Ao explorar gatilhos mentais como urgência, medo e confiança — operando no que a psicologia define como Sistema 1 —, os criminosos conseguem contornar as mais robustas barreiras tecnológicas simplesmente solicitando que a própria vítima abra a porta.

Verificou-se, ainda, que a resposta estatal, materializada na Lei Geral de Proteção de Dados (LGPD) e no endurecimento penal da Lei nº 14.155/2021, representa um avanço institucional indispensável. No entanto, o arcabouço legal atua predominantemente nas esferas punitiva e regulatória, possuindo eficácia limitada na prevenção do ato impulsivo do usuário no momento do ataque. A lei pode punir o fraudador e multar a empresa negligente, mas é incapaz de intervir no "clique" precipitado do cidadão.

Portanto, conclui-se que a tecnologia de segurança, por si só, atingiu um ponto de retornos decrescentes. A solução para a mitigação dos riscos cibernéticos no Brasil não reside apenas na aquisição de novos *firewalls* ou na criação de novas leis, mas imperativamente no investimento em capital humano. A transição de uma postura de "usuário passivo" para a de um "*firewall* humano" exige políticas públicas e privadas massivas de letramento digital. Somente através da educação preventiva, que ensina o usuário a identificar e neutralizar os gatilhos da engenharia social, será possível reduzir a eficácia das fraudes e construir um ambiente digital sustentável e seguro para a sociedade brasileira.

2.8 Bibliografia

[1] Relatório de atividade criminosa online: Panorama de fraudes e ameaças digitais. URL: `https://www.axur.com`.

[2] Lei nº 12.965, de 23 de abril de 2014. estabelece princípios, garantias, direitos e deveres para o uso da internet no brasil. diário oficial da união: Seção 1, brasília, df, p. 1, 24 abr. 2014.

[3] Lei nº 13.709, de 14 de agosto de 2018. lei geral de proteção de dados pessoais (lgpd). diário oficial da união: Seção 1, brasília, df, ano 155, n. 157, p. 59-64, 15 ago. 2018.

[4] Lei nº 14.155, de 27 de maio de 2021. altera o decreto-lei nº 2.848... para tornar mais graves os crimes de violação de dispositivo informático, furto e estelionato... diário oficial da união: Seção 1, brasília, df, 28 maio 2021.

[5] Pesquisa sobre o uso das tecnologias de informação e comunicação nos domicílios brasileiros: Tic domicílios 2023. URL: `https://cetic.br`.

[6] Pesquisa febraban de tecnologia bancária 2024. URL: `https://febraban.org.br`.

[7] Rápido e devagar: duas formas de pensar. tradução de cássia zanon. rio de janeiro: Objetiva, 2012.

[8] A arte de enganar: ataques de engenharia social e como se defender. tradução de arlete simille marques. são paulo: Pearson prentice hall, 2003.

[9] Internet security threat report (istr). URL: https://www.broadcom.com.

Parte II

Projetos

Capítulo 3

Dungeon Master's Tools

Cauã Marques da Silva

Acesse o repositório do projeto clicando aqui

Read this paper in english

3.1 Conceito do Projeto

Esse projeto foi desenvolvido com o intuito de prover uma aplicação *desktop*, para uso em *D&D 4e e outros RPGs* por *Dungeon Masters*, que fosse de fácil manutenibilidade e expansível. Para tal, o programa dispõe de uma funcionalidade básica para ler arquivos *SQL* do antigo serviço *D&D Insiders* e mostrar esses dados em tabelas permitindo filtragem, busca e ordenação. Além, de dispor da capacidade de carregar *plugins* feitos por usuários.

3.2 Pré-requisitos e recursos

O projeto utiliza *Python* 3.10.12 (versões mais novas podem não dar suporte a biblioteca *Yapsy*), além de duas bibliotecas externas:

- - *PySide6* (versão 6.8.3), disponível em (*PyPi*)
- - *yapsy*, disponível em (*PyPi*)

Essa aplicação foi desenvolvida para ler os arquivos de dados (em SQL) do antigo serviço *Dungeons and Dragons Insiders* (DDI), esse serviço **não** está mais disponível. Para utilizar a parte base da aplicação é necessário uma cópia desses arquivos (por serem propriedade da *Wizards of the Coast*, não podem ser compartilhados aqui), entretanto a aplicação ainda vai iniciar e carregar *plugins* normalmente sem os arquivos (já há planos para adicionar a capacidade de detectar se o usuário possuí esses arquivos e desativar qualquer funcionamento que necessite deles).

3.3 Passo a Passo

O projeto se deu da vontade de implementar uma alternativa ao software, baseado em *java/javaFx*, *Portable D&D Compendium* (infelizmente indisponível para compartilhamento aqui). O processo de implementação se deu através de:

1. Baixar o software *Portable D&D Compendium*;
2. Extrair o conteúdo do seu pacote *javac*;
3. Utilizar de ferramentas (*vscode* e JD-GUI) para descompilar o conteúdo dos arquivos *class*;
4. Reproduzir parcialmente o funcionamento do seu *Parser* para os arquivos *SQL* do antigo *D&D Insider* (o resultado disso foi a classe *DDIParser*);
5. Utilizando a documentação do *Pyside/QT* e experimentação a recriação da tabela do software original foi alcançada com sucesso;
6. A partir daqui o desenvolvimento focou-se em apresentar esses dados em um formato humano (uma tabela, parcialmente igual ao *Portable D&D Compendium*, o resultado é a classe *CompendiumScreen*;

7. Fora a recriação do software original em *Python/Pyside*, buscou-se introduzir novas funcionalidades como a capacidade de carregar *plugins* e a criação de uma filtragem dinâmica baseada na categoria dos itens dispostos na tabela; Exemplo disponível aqui
8. Como efeito colateral do uso de *Python* a aplicação agora é nativamente compatível com *Linux* e *Windows* e, possivelmente, qualquer sistema operacional que dê suporte ao *Python* e a biblioteca *Pyside6*;

3.4 Instalação

- Clone o projeto pelo *git*, ou faça o *download* pelo *Github*:

```
git clone https://github.com/CMS999/
    Dungeon-Master-s-Tools.git
```

3.5 Execução

- Na pasta raiz do projeto (Dungeon-Master-s-Tools) execute o arquivo *main.py*:

```
python main.py
```

3.6 Bugs

Alguns *bugs* já são conhecidos, mas devido a natureza do projeto espera-se encontrar mais:

- O número no rodapé da página indicando a quantidade de itens sendo visualizados em um dado momento, ainda não está sendo atualizado sempre;
- A funcionalidade das *QToolBar* de cada aba, não funciona apropriadamente, abas sem um *QToolBar* associado geram uma *QToolbar* "fantasma";
- Mudar para aba Filtros enquanto uma entrada da tabela está selecionada faz com que cada filtragem altere o *display* para o *HTMLRender*;
- Alguns textos e ícones ainda são temporários/*placeholders* (Ex.: o ícone da aplicação no canto superior esquerdo é o padrão do sistema);
- A versão mais nova do *PySide6* (pelo menos até 03/08/2025), possuí um *bug* em relação a *QtWebEngine* em *Linux*, que gera *bugs* visuais;
- O programa não foi testado em nenhum momento em *Windows* e, embora, ele execute, é possível que diferenças de sistemas operacionais apareçam (Ex.: foi necessário adicionar um estilo '*Fusion*' no *app* em *main.py*, pois o estilo padrão do *Windows* não estava sendo aplicado, como é no *linux*).

3.7 Autor

- Cauã Marques da Silva (*GitHub*)

3.8 Plugins

Ainda que rudimentares, a aplicação é capaz de carregar e exibir outras telas criadas com o *Pyside6*, desde que respeitem a interface especificada em PluginTypes. Exemplo disponível aqui

Um *plugin* é composto de no mínimo dois arquivos:

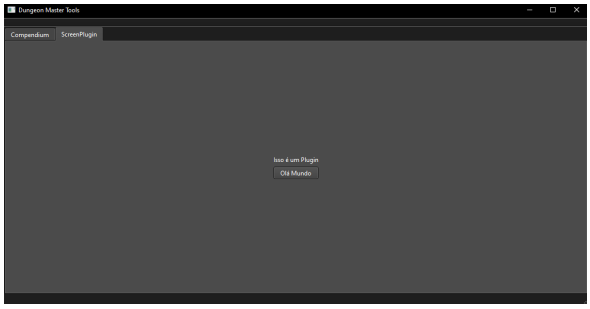
- Arquivo *python* com uma classe que herde da classe *MainPlugin*, a classe herdeira é a que será exibida na tela;
- Um arquivo de especificação do tipo ***.yapsyplugin*** com o seguinte conteúdo:

```
[Core]
Name = Nome\_do\_seu\_plugin
Module = Nome\_do\_arquivo\_python\_com\_o
\_plugin
```

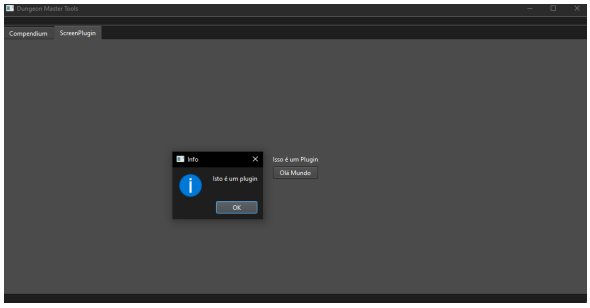
3.9 Imagens

Exemplo de Plugin

Tela do Plugin



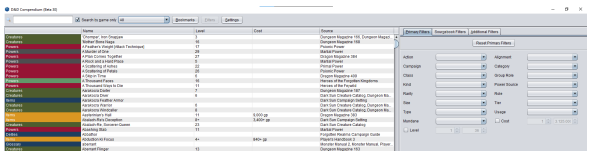
Após clicar no botão



Filtro Dinâmico/*Primary Filters*

Filtro na categoria ALL

- *Portable D&D Compendium*

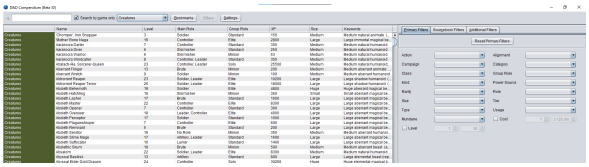


- *Dungeon Master's Tools*

Aqui a categoria ALL só possui colunas que já tem seus respectivos filtros, por isso a aba está vazia.

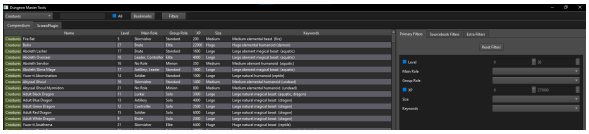
Filtro na categoria *Creature*

- *Portable D&D Compendium*



- *Dungeon Master's Tools*

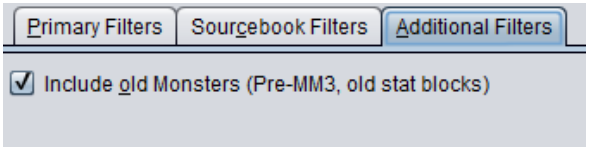
Aqui somente as colunas pertinentes são exibidas.



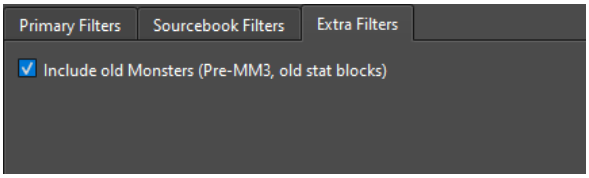
Monster Manual 3 (MM3) Filter

Esse filtro remove da tabela todos os monstros que foram escritos antes do terceiro manual de monstros, no qual houve uma mudança no design das fichas de monstro.

- *Portable D&D Compendium*



- *Dungeon Master's Tools*



Pré-MM3

Goblin Blackblade **Level 1 Lurker**
Small natural humanoid , goblin XP 100

Initiative +7 **Senses** Perception +1; low-light vision
HP 25; **Bloodied** 12
AC 16; **Fortitude** 12, **Reflex** 14, **Will** 11
Speed 6

Short Sword (standard, at-will) **+ Weapon**
+5 vs AC; 1d6+2 damage.

Combat Advantage
The goblin blackblade deals an extra 1d6 damage against any target it has combat advantage against.

Goblin Tactics (immediate reaction, when missed by a melee attack; at-will)
The goblin shifts 1 square.

Sneaky
When shifting, a goblin blackblade can move into a space occupied by an ally of its level or lower. The ally shifts into the blackblade's previous space as a free action.

Alignment Evil **Languages** Common, Goblin
Skills Stealth +10, Thievery +10
Str 14 (+2) **Dex** 17 (+3) **Wis** 12 (+1)
Con 13 (+1) **Int** 8 (-1) **Cha** 8 (-1)
Equipment: leather armor , short sword .

Published in Monster Manual, page(s) 136, Dungeon Delve, page(s) 22, Forgotten Realms Campaign Guide, page(s) 25, Dungeon Magazine 156, page(s) 12, Dungeon Magazine 177, page(s) 34, Dungeon Magazine 172,

Pós-MM3

Shuffling Zombie
Medium natural animate (undead)

HP 0; **Bloodied** 0 ; your bloodied value **Initiative** as conjurer
AC 0, **Fortitude** 0, **Reflex** 0, **Will** 0 **Perception**
Speed 4
Immune disease, poison

Standard Actions

Bash (weapon) **+ At-Will**
Attack: Melee 1 (one creature); +Intelligence +2 vs. AC
Hit: 6 damage.

Grab (weapon) **+ At-Will**
Attack: Melee 1 (one creature); +Intelligence vs. Reflex
Hit: 3 damage, and the target is grabbed (escape DC equals your Will).

Move Actions

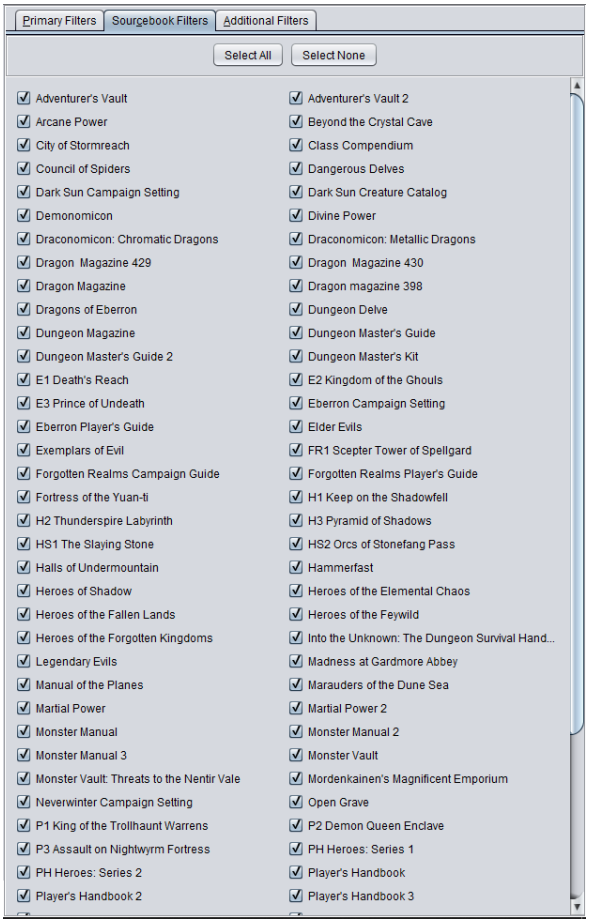
Move + At-Will
Effect: The zombie stands up, crawls or shifts 1, or climbs or walks its speed. It cannot run.

Triggered Actions

Follow + At-Will
Trigger: The shuffling zombie is more than 10 squares from you when you end your turn.
Effect: As a free action, the zombie walks by the most direct route to the nearest unoccupied square that is within 10 squares of you. If the zombie must walk more than 8 squares to reach this square, it instead drops to 0 hit points.

Str (0) **Dex** (0) **Wis** (0)
Con (0) **Int** (0) **Cha** (0)
Alignment Unaligned **Languages** -

Published in Dungeon Magazine 182.

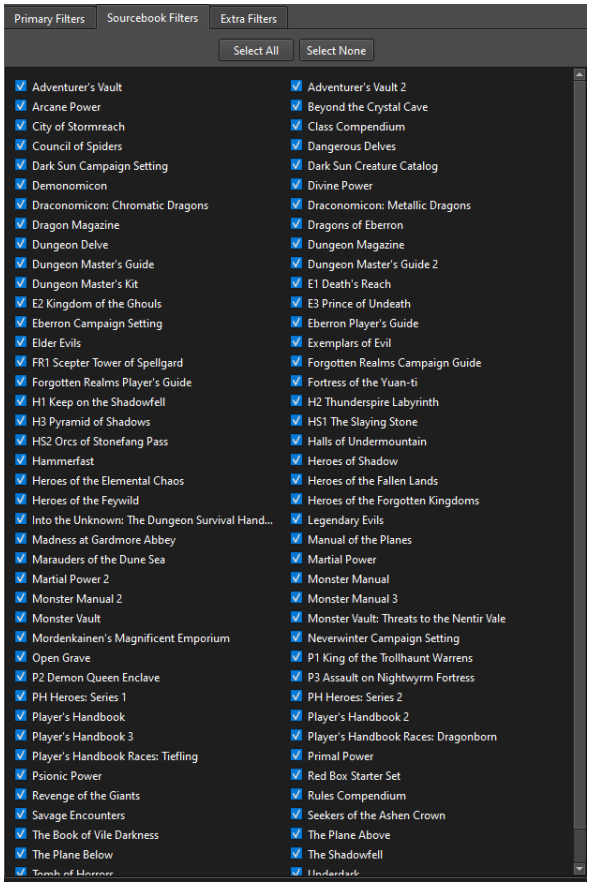


Source Filter

Esse filtro restringe a exibição das entradas que fazem parte das fontes exibidas.

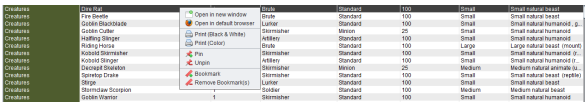
- *Portable D&D Compendium*

- *Dungeon Master's Tools*



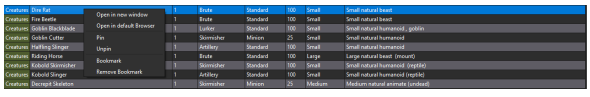
Context Menu

- *Portable D&D Compendium*



- *Dungeon Master's Tools*

Ainda sem ícones.



Pinned items

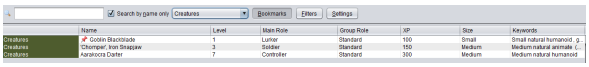
- *Portable D&D Compendium*

- *Dungeon Master's Tools*

O ícone de coração é um *placeholder*.

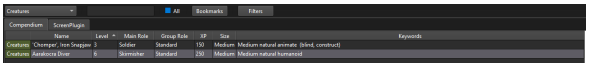
Bookmarks

- *Portable D&D Compendium*



- *Dungeon Master's Tools*

Itens *Pinned* não aparecem no *bookmarks* (isso é intencional)



3.10 Repositório

O repositório com os arquivos e demais informações sobre o projeto se encontra em:

`https://github.com/CMS999/Dungeon-Master-s-Tools`

Capítulo 4

GoFramework

Lucas Alves Zito

Rafael de Campos Villa da Silveira

Acesse o repositório do projeto clicando aqui

4.1 Visão Geral

GoFramework é um framework REST simples em Go que permite criar APIs através de uma estrutura de diretórios. O framework converte automaticamente a estrutura de pastas em rotas HTTP, onde cada arquivo representa um método HTTP e cada pasta representa um segmento da URL.

4.2 Estrutura de Arquivos

```
pkg/goframework/
# Aplicação principal e gerenciamento de
rotas
 app.go

# Sistema de roteamento e matching
 router.go

# Contexto da requisição HTTP
 context.go

# Registro de handlers durante init()
 registry.go

# Tipos e erros HTTP
 types.go
```

4.3 Pré-requisitos

Para o desenvolvimento do GoFramework, utilizamos a linguagem Go, acompanhada de suas bibliotecas padrão para criação de servidores HTTP, manipulação de JSON e controle de concorrência. Além disso, utilizamos um ambiente de desenvolvimento compatível com Go 1.21+ e um sistema de arquivos local para organização da estrutura de rotas.

4.4 Instalação

Para executar o projeto, é necessário ter instalado o **Go 1.25.3**. Abaixo segue um passo a passo rápido para realizar a instalação:

1. **Acesse o site oficial do Go**
   Entre em: `https://go.dev/dl/` e baixe o instalador correspondente ao seu sistema operacional (Windows, Linux ou macOS).

2. **Execute o instalador**
   No Windows ou macOS, basta seguir o assistente padrão de instalação. No Linux, extraia o arquivo `.tar.gz` para `/usr/local` utilizando:

   ```
   sudo tar -C /usr/local -xzf
   go1.25.3.linux-amd64.tar.gz
   ```

3. **Configure a variável de ambiente `PATH`**
   Adicione o diretório do Go ao PATH

   ```
   export PATH=$PATH:/usr/local/go/bin
   ```

4. **Verifique a instalação**
   Execute o comando:

   ```
   go version
   ```

   Ele deve exibir **`go version go1.25.3 ...`**

Com isso, o ambiente está pronto para rodar o projeto.

4.5 Componentes Principais

1. **types.go - Tipos Base**
   Define os tipos fundamentais do framework.

   HandlerFunc

   - Tipo: `func(*Context) error`
   - Assinatura padrão para todas as funções de handler
   - Recebe um Context e retorna um error (nil se sucesso)

   HTTPError

   - Estrutura para erros HTTP customizados
   - Campos: `Status int`, `Message string`
   - Implementa a interface `error`

   NewHTTPError(status int, message string) error

   - Cria um erro HTTP com status e mensagem específicos
   - Usado pelos handlers para retornar erros HTTP

2. **registry.go - Sistema de Registro**
   Gerencia o registro de handlers durante a inicialização do programa.

   Variáveis globais:

   - `routesReg`: mapa que armazena todas as rotas registradas
   - `registryMu`:mutex para sincronização thread-safe

   Register(handler HandlerFunc)

   - Função chamada pelos handlers em suas funções `init()`
   - Usa `runtime.Caller(1)` para obter o caminho do arquivo que chamou Extrai o método HTTP do nome do arquivo (get.go = GET, post.go = POST, etc)
   - Armazena o handler no mapa global `routesReg` usando o caminho absoluto do arquivo como chave
   - Thread-safe através de mutex

   listRegistrations() []registration

   - Retorna lista de todas as rotas registradas
   - Usado por `LoadRoutes()` para carregar as rotas no router
   - thread-safe

3. **router.go - Sistema de Roteamento**
   Gerencia o roteamento de requisições HTTP para os handlers corretos.
   Estruturas:
   `route`

   - Representa uma rota registrada
   - Campos: `method string`, `pattern string`, `segments []routeSegment`, `handler HandlerFunc`

   `Router`

   - Gerencia todas as rotas
   - Campo: `routes []route`

   NewRouter() Router

   - Cria uma nova instância do Router
   - Inicializa slice vazio de rotas

   AddRoute(method, pattern string, handler HandlerFunc) error

   - Adiciona uma rota ao router
   - Valida método, pattern e handler
   - Normaliza o método para maiúsculas
   - Normaliza o pattern através de `normalizePattern()`
   - Verifica duplicatas (mesmo método + pattern)

- Converte o pattern em segmentos através de `buildSegments()`
- Adiciona a rota ao slice

Match(method, path string) (HandlerFunc, map[string]string, bool)

- Busca uma rota que corresponda ao método e caminho
- Normaliza método para maiúsculas
- Converte o path em segmentos através de `splitPath()`
- Itera sobre todas as rotas procurando correspondência
- Para cada rota, chama `matchSegments()` para verificar correspondência
- Retorna handler, parâmetros extraídos e true se encontrou
- Retorna nil, nil, false se não encontrou

matchSegments(patternSegs, pathSegs []routeSegment) (map[string]string, bool)

- Compara segmentos do pattern com segmentos do path
- Verifica se têm o mesmo tamanho
- Para cada segmento do pattern:
  - Se `isParam` é true, extrai o valor do path e adiciona ao mapa de parâmetros
  - Se não é parâmetro, compara valores exatamente
- Retorna mapa de parâmetros e true se todos correspondem
- Retorna nil, false se não corresponde

buildSegments(pattern string) ([]routeSegment, error)

- Converte um pattern string em slice de routeSegment
- Primeiro chama `splitPath()` para obter segmentos básicos
- Para cada segmento:
  - Se começa com `:`, cria um routeSegment com `isParam = true`
  - Caso contrário, cria segmento normal
- Valida que parâmetros têm nome válido (não vazio após `:`)

splitPath(path string) []routeSegment

- Divide um caminho em segmentos
- Normaliza o path primeiro
- Se path é `/`, retorna slice vazio
- Remove `/` inicial e divide por `/`
- Cria routeSegment para cada parte (ainda sem identificar parâmetros)

normalizePattern(pattern string) string

- Normaliza um pattern de rota
- Se vazio, retorna `/`
- Remove espaços
- Garante que começa com `/`
- Remove `/` final (exceto se for apenas `/`)
- Retorna pattern normalizado

4. context.go - Contexto da Requisição
Fornece acesso aos dados da requisição HTTP e métodos para resposta.
Estruturas:
`Context`

- Contexto passado para cada handler
- Campos:
  - `Req *http.Request`: requisição HTTP original
  - `Res http.ResponseWriter`: response writer original
  - `Params map[string]string`: parâmetros extraídos da URL
  - `writer *responseWriter`: wrapper do ResponseWriter
  - `bodyOnce sync.Once`: garante leitura única do body
  - `body []byte`: corpo da requisição em cache
  - `bodyErr error`: erro na leitura do body

`responseWriter`

- Wrapper do `http.ResponseWriter` padrão
- Campos: `ResponseWriter http.ResponseWriter, wrote bool`
- `wrote` indica se headers já foram escritos

newContext(w *responseWriter, r http.Request, params map[string]string) Context

- Construtor privado do Context
- Cria cópia do mapa de parâmetros
- Inicializa Context com requisição, response writer e parâmetros

Context.JSON(status int, data any) error

- Escreve resposta JSON
- Verifica se já escreveu headers (retorna nil se sim)
- Define Content-Type como application/json
- Escreve status code
- Codifica data como JSON e escreve no response writer

Context.Param(name string) string

- Retorna valor de parâmetro da URL por nome
- Exemplo: para rota `/users/:id`, `Param("id")` retorna o valor

Context.Query() url.Values

- Retorna query parameters da URL
- Wrapper direto para `Req.URL.Query()`

Context.QueryParam(name string) string

- Retorna valor de um query parameter específico
- Wrapper para `Query().Get(name)`

Context.SetHeader(name, value string)

- Define um header HTTP na resposta
- Só funciona se headers ainda não foram escritos
- Wrapper para `Header().Set()`

Context.Body() ([]byte, error)

- Lê o corpo da requisição HTTP
- Usa `sync.Once` para garantir leitura única
- Fecha o body após leitura
- Cacheia o resultado em `body` e `bodyErr`
- Retorna o mesmo resultado em chamadas subsequentes

responseWriter.WriteHeader(statusCode int)

- Escreve status code HTTP
- Previne escrita múltipla através da flag `wrote`
- Chama `ResponseWriter.WriteHeader()` do Go padrão

responseWriter.Write(b []byte) (int, error)

- Escreve dados na resposta
- Se headers não foram escritos, escreve StatusOK automaticamente
- Delega para `ResponseWriter.Write()` do Go padrão

5. **app.go - Aplicação Principal**
Gerencia a aplicação, carregamento de rotas e tratamento de requisições HTTP.
Estrutura:
`App`

- Aplicação principal do framework
- Campos:
  - `router *Router`: instância do router
  - `routesDir string`: diretório raiz das rotas

New() App

- Construtor da aplicação
- Cria novo Router através de `NewRouter()`
- Retorna instância de App inicializada

App.LoadRoutes(root string) error

- Carrega rotas do sistema de arquivos

- Valida que root não está vazio
- Converte root para caminho absoluto
- Chama `listRegistrations()` para obter rotas registradas
- Valida que existem rotas registradas
- Para cada rota registrada:
  - Verifica se o arquivo está dentro do diretório root
  - Converte caminho do arquivo em pattern HTTP através de `buildPatternFromFile()`
  - Adiciona rota ao router através de `router.AddRoute()`
- Valida que pelo menos uma rota foi carregada
- Armazena routesDir para referência futura

App.Listen(addr string) error

- Inicia servidor HTTP
- Valida que router está inicializado
- Chama `http.ListenAndServe()` passando addr e a própria App (que implementa http.Handler)
- Bloqueia até erro ou interrupção

App.ServeHTTP(w http.ResponseWriter, r http.Request)

- Implementa interface `http.Handler`
- Chamado automaticamente pelo servidor HTTP do Go para cada requisição
- Fluxo:
  i. Chama `router.Match()` para encontrar handler correspondente
  ii. Se não encontrou, retorna erro 404 JSON
  iii. Cria `responseWriter` wrapper
  iv. Cria `Context` através de `newContext()`
  v. Chama o handler passando o Context
  vi. Se handler retornou erro e resposta ainda não foi escrita, chama `handleError()`

App.handleError(w http.ResponseWriter, err error)

- Trata erros retornados pelos handlers
- Se erro é `*HTTPError`, retorna status e mensagem do erro
- Caso contrário, retorna erro 500 genérico
- Usa `writeJSONError()` para escrever resposta JSON

writeJSONError(w http.ResponseWriter, status int, message string)

- Função auxiliar para escrever erros em formato JSON
- Cria objeto `"error": message`
- Define Content-Type como application/json
- Escreve status code
- Codifica e escreve JSON

buildPatternFromFile(root, file string) (string, error)

- Converte caminho de arquivo em pattern HTTP
- Calcula caminho relativo do arquivo em relação ao root
- Valida que arquivo está dentro do root (não permite `..`)
- Obtém diretório do arquivo relativo
- Se diretório é `.`, trata como vazio
- Divide diretório em segmentos
- Para cada segmento:
  - Ignora segmentos vazios ou `index`
  - Se segmento começa com `_`, trata como parâmetro (converte `_id` para `:id`)
  - Caso contrário, adiciona como segmento normal
- Junta segmentos com `/` e adiciona `/` inicial
- Retorna pattern (ex: `/users/:id`)

6. **FRONTEND.sh - Interface**
   O frontend do projeto consiste em um script Bash interativo responsável por consumir a API e oferecer ao usuário uma interface simples via terminal. Ele utiliza curl para enviar requisições HTTP e, quando disponível, o utilitário jq para interpretar respostas em JSON. Por meio deste script, é possível listar usuários, buscar por ID, editar e excluir registros, tudo de forma rápida e intuitiva, sem necessidade de abrir um navegador ou ferramentas externas.

4.6 Fluxo de Execução

1. **Inicialização do Programa**
   i. main() executa
   ii. Imports executam funções init() dos pacotes de rotas
   iii. Cada init() chama goframework.Register(handler)
   iv. Register() armazena handler no mapa global routesReg
   v. main() chama app.New()
   vi. New() cria Router vazio
   vii. main() chama app.LoadRoutes("example/app/routes")
   viii. LoadRoutes() chama listRegistrations()
   ix. Para cada rota registrada:
      - buildPatternFromFile() converte caminho em pattern
      - router.AddRoute() adiciona rota ao router
   x. main() chama app.Listen(":8080")
   xi. Listen() inicia servidor HTTP
2. **Processamento de Requisição HTTP**
   i. Cliente envia requisição HTTP
   ii. Go chama App.ServeHTTP(w, r)
   iii. ServeHTTP() chama router.Match(method, path)
   iv. Match() itera sobre rotas:
      - Para cada rota, chama matchSegments()
      - Se corresponde, retorna handler e parâmetros
   v. Se não encontrou, ServeHTTP() retorna 404 JSON
   vi. Se encontrou:
      - Cria responseWriter wrapper
      - Cria Context com requisição e parâmetros
      - Chama handler(ctx)
   vii. Handler executa lógica:
      - Pode ler ctx.Body() para obter dados
      - Pode ler ctx.Param() para obter parâmetros
      - Pode ler ctx.QueryParam() para query params
      - Chama ctx.JSON() para retornar resposta
   viii. Se handler retornou erro:
      - ServeHTTP() chama handleError()
      - handleError() escreve erro JSON
3. **Exemplo: Requisição GET /users/1**
   i. Requisição: GET /users/1
   ii. ServeHTTP() recebe w, r
   iii. router.Match("GET", "/users/1")
   iv. Match() itera rotas:
      - Rota 1: GET /users
        – matchSegments() compara ["users"] com ["users", "1"]
        – Não corresponde (tamanhos diferentes)
      - Rota 2: GET /users/:id
        – matchSegments() compara [users, :id] com [users, 1]
        – Segmento 1: "users"== "users"
        – Segmento 2: :id é parâmetro, extrai "1"
        – Retorna handler, params={"id": "1"}, true
   v. ServeHTTP() cria Context com params
   vi. Handler Get(ctx) executa:
      - ctx.Param("id") retorna "1"
      - Converte "1"para int
      - Busca usuário no serviço
      - ctx.JSON(200, user) escreve resposta
   vii. Resposta enviada ao cliente

4.7 Exemplo de Execução

Em primeiro lugar a API desenvolvida com o Go-Framework é iniciada. O servidor é carregado, as rotas são registradas e a aplicação fica pronta para receber requisições HTTP.

```
my-go-framework master ● ? × go run ./example/
2025/12/05 19:12:14 servidor rodando em http://localhost:8080
```

Em seguida, é inicializado o frontend.

E por fim, a execução da aplicação já em funcionamento, exibindo a listagem de usuários obtida a partir da API.

4.8 Convenções de Estrutura de Arquivos

Mapeamento de Arquivos para Rotas

- `routes/index/get.go` → `GET /`
- `routes/users/get.go` → `GET /users`
- `routes/users/post.go` → `POST /users`
- `routes/users_id/get.go` → `GET /users/:id`
- `routes/users/_id/put.go` → `PUT /users/:id`
- `routes/users/_id/delete.go` → `DELETE /users/:id`

Regras

- Nome do arquivo determina método HTTP: `get.go` = GET, `post.go` = POST, etc.
- Estrutura de pastas determina o caminho da URL
- Pasta `index` é ignorada (vira `/`)
- Pastas começando com `_` viram parâmetros (ex: `_id` vira `:id`)
- Cada arquivo deve ter função `init()` que chama `goframework.Register(HandlerFunc)`
- Handler deve ter assinatura `func(*Context) error`

4.9 Exemplo de Handler

```go
package users

import (
    "my-go-framework/pkg/goframework"
)

func init() {
    goframework.Register(Get)
}

func Get(ctx *goframework.Context)
 ↪  error {
    // Ler parâmetros da URL
    id := ctx.Param("id")

    // Ler query parameters
    page := ctx.QueryParam("page")

    // Ler corpo da requisição (se
     ↪  necessário)
    body, err := ctx.Body()

    // Retornar resposta JSON
    return ctx.JSON(200,
     ↪  map[string]string{"id": id})
}
```

4.10 Tratamento de Erros

Handlers podem retornar erros de duas formas:

1. **HTTPError customizado:**

   ```
   return goframework.NewHTTPError(404,
   ↪  "usuário não encontrado")
   ```

   - Retorna status HTTP específico com mensagem

2. **Erro genérico:**

   ```
   return fmt.Errorf("erro interno")
   ```

   - Retorna status 500 com mensagem "internal_error"

   O framework automaticamente:

   - Detecta se resposta já foi escrita (não sobrescreve)
   - Converte erros em respostas JSON
   - Trata erros HTTPError com status correto

4.11 Thread Safety

- `registry.go` usa mutex para proteger o mapa de rotas durante registro
- `context.go` usa `sync.Once` para garantir leitura única do body
- `responseWriter` previne escrita múltipla de headers
- Router é thread-safe para leitura (rotas não mudam após LoadRoutes)

4.12 Limitações e Considerações

1. Rotas são carregadas uma vez na inicialização (não suporta hot-reload)
2. Matching de rotas é linear (O(n) onde n é número de rotas)
3. Parâmetros de rota são sempre strings (conversão manual necessária)
4. Body da requisição é lido completamente na memória
5. Não há middleware system nativo
6. Não há validação automática de dados
7. Estrutura de diretórios deve seguir convenções estritas

4.13 Repositório

O repositório com os arquivos e demais informações sobre o projeto se encontra em:

`https://github.com/rafacvs/go-rest-framework`

Capítulo 5

Strive APP

Rafael Mori Pinheiro

Joao Vitor Averaldo Antunes

Pedro Enrico Barchi Nogueira

Acesse o repositório do projeto clicando aqui

5.1 Conceito do Projeto

O **Strive** é uma aplicação móvel multiplataforma desenvolvida para centralizar a gestão da saúde de atletas e entusiastas do bem-estar. A principal finalidade do projeto é resolver a fragmentação de dados de saúde, integrando em um único ambiente o controle de treinos de musculação, dietas nutricionais, monitoramento de hidratação e, mais recentemente, a gestão medicamentosa.

O diferencial do projeto reside em seu sistema robusto de **Gamificação**. Inspirado em aplicativos de aprendizado de idiomas, o Strive implementa um sistema de ligas semanais (com promoção e rebaixamento), ganho de XP (pontos de experiência) por tarefas concluídas e um Leaderboard dinâmico. Isso visa aumentar o engajamento do usuário e transformar a manutenção da saúde em um hábito consistente e divertido.

5.2 Pré-requisitos e recursos utilizados

O grupo utilizou a linguagem **Dart** e o framework Flutter para o desenvolvimento front-end e mobile, aplicando conceitos avançados de Engenharia de Software.

Tecnologias e Bibliotecas Principais:

- **Flutter & Dart:** Base do desenvolvimento.
- **Firebase (Google):**
  - *Firestore Database:* Para persistência de dados em nuvem (NoSQL).
  - *Firebase Authentication:* Para gestão de identidade e login.
- **Riverpod:** Para gerenciamento de estado reativo e injeção de dependência (utilizando `AsyncNotifier` e `FutureProvider`).
- **GoRouter:** Para navegação declarativa e deep linking.
- **Slang:** Para internacionalização (i18n) e suporte a múltiplos idiomas.
- **Clean Architecture:** Padrão arquitetural utilizado para separar as camadas de *Domain, Data e Presentation.*

APIs e Recursos Externos:

- **OpenFoodFacts API:** Utilizada para busca de informações nutricionais de alimentos.
- **Wger Workout API:** Utilizada como base de dados para exercícios e grupos musculares.

5.3 Passo a passo

O desenvolvimento do projeto seguiu uma metodologia iterativa, focada na implementação de funcionalidades por módulos (Feature-first):

1. **Estruturação e Arquitetura:** Inicialmente, definimos a estrutura de pastas seguindo a Clean Architecture e configuramos o ambiente com Flutter e Firebase.
2. **Módulo de Autenticação e Perfil:** Implementamos o login e a criação de perfil do usuário, salvando dados vitais como peso, altura e nível de atividade no Firestore.
3. **Core Features (Treino e Dieta):**
    - Desenvolvemos o CRUD de treinos, permitindo criar planos personalizados.
    - Integramos a API de alimentos para o cálculo automático de macros na dieta.
4. **Refatoração para Riverpod:** Migramos o gerenciamento de estado para Riverpod para lidar melhor com a assincronicidade e evitar *rebuilds* desnecessários.
5. **Sistema de Gamificação (Engine):**
    - Criamos a lógica de cálculo de XP (Total vs. Semanal).
    - Implementamos o algoritmo de "Virada de Semana", que processa promoções e rebaixamentos de liga com base no desempenho do usuário.
    - Desenvolvemos a UI do Leaderboard com animações e filtros por liga.

5.4 Instalação

Para recriar o ambiente de desenvolvimento do Strive, siga os passos abaixo:

1. **Clone o repositório:** Execute o comando no terminal:

    ```
    git clone https://github.com/Rafael-Mori-2022/strive-app.git

    cd strive-app
    ```

2. **Instale as dependências:** Na pasta raiz do projeto, execute:

    ```
    flutter pub get
    ```

3. **Configuração do Firebase (Importante):**
    - Como os arquivos de configuração (`google-services.json` e `GoogleService-Info.plist`) são sensíveis e geralmente ignorados no git, você precisará criar um projeto no Console do Firebase.
    - Adicione os arquivos gerados pelo Firebase nas pastas `android/app` e `ios/Runner`, respectivamente.
4. **Geração de Código:** O projeto utiliza geradores de código para tradução e serialização. Execute:

    ```
    dart run build_runner build -delete-conflicting-outputs

    dart run slang
    ```

5.5 Execução

Para rodar o projeto em um emulador ou dispositivo físico:

1. Abra o emulador (Android Studio ou Simulator do iOS) ou conecte seu dispositivo via USB.
2. No terminal, dentro da pasta do projeto, execute:

    ```
    flutter run
    ```

    Dica: Caso encontre erros de cache, utilize flutter clean antes de rodar novamente.

5.6 Bugs/problemas conhecidos

Durante o desenvolvimento, identificamos os seguintes comportamentos que podem requerer atenção:

- **Escalabilidade do Leaderboard:** Atualmente, o Leaderboard busca todos os usuários para realizar a ordenação em memória. Em um cenário com milhares de usuários, isso pode gerar lentidão ou alto consumo de leitura no Firestore. A solução futura seria implementar Cloud Functions para processar o ranking no servidor.

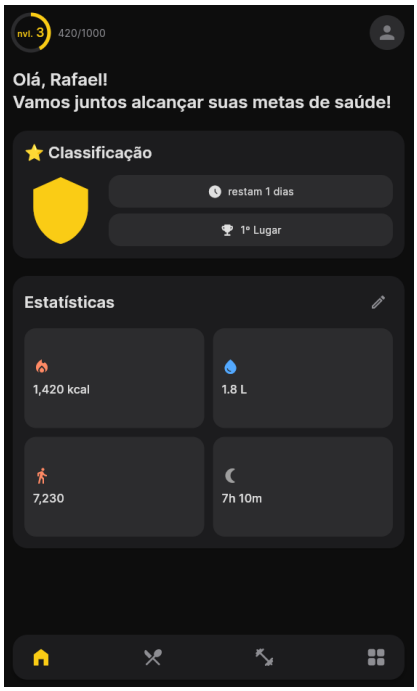
- **Delay na atualização de XP:** Em conexões lentas, o SnackBar de "Ganho de XP"pode aparecer antes da confirmação do banco de dados, devido à natureza otimista da UI.

5.7 Autores

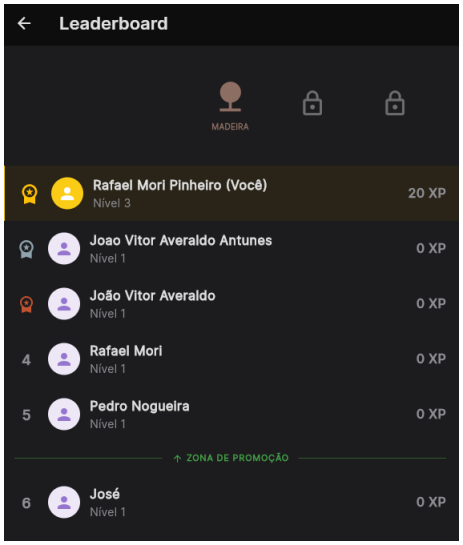
- **Rafael Mori Pinheiro** (Rafael-Mori-2022) - *Desenvolvimento Full-Stack & Gamification Logic*
- **Joao Vitor Averaldo Antunes** (Mun1nm) - *Desenvolvimento Mobile & Integrações*
- **Pedro Enrico Barchi Nogueira** (Pedro-bnogueira) - *Desenvolvimento Mobile & UI/UX*

5.8 Imagens/screenshots

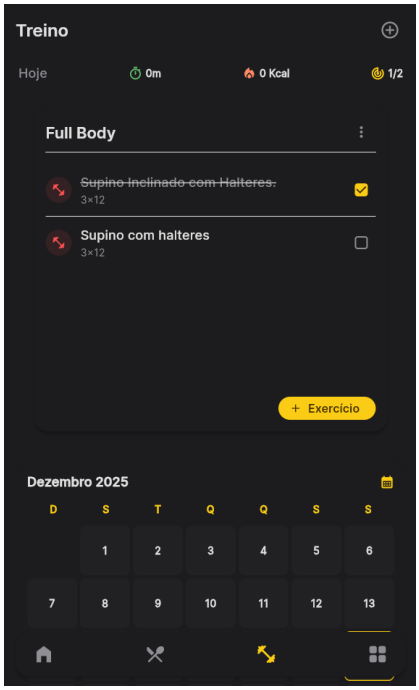
Abaixo estão as capturas de tela das principais funcionalidades do Strive:



1. **Dashboard**
   Progresso diário, nível e atalhos.



2. **Leaderboard**
   Ranking semanal e sistema de ligas.



3. **Treinos**
   Gestão e execução de séries.

5.9 Repositório

O repositório com os arquivos e demais informações sobre o projeto se encontra em:

`https://github.com/Rafael-Mori-2022/strive-app`

Capítulo 6

Execução de um jogo Linux no Windows através de WSL 2: Uma Tool-Assisted Speedrun (TAS)

Rick Robert Tomaz Rezende

Acesse o repositório do projeto clicando aqui

6.1 Conceito do Projeto

Este projeto tem como objetivo viabilizar a execução de um jogo nativo de Linux em ambiente *Windows*, utilizando o *Windows Subsystem for Linux 2* (WSL2), com foco na criação de *Tool-Assisted Speedruns* (TAS). Para isso, foi explorado o uso da ferramenta ***LibTAS***, amplamente empregada na comunidade de *speedrunning* em sistemas *Linux*, permitindo controle preciso de entradas, execução determinística e análise quadro a quadro do jogo.

O jogo escolhido para os experimentos foi ***Hollow Knight*** (versão *Linux*), por se tratar de um jogo 2D relativamente leve, popular na comunidade de *speedruns* e amplamente documentado. O projeto investiga a compatibilidade gráfica e o desempenho da execução do jogo dentro do WSL2 no *Windows* 11, avaliando limitações, erros gráficos e estabilidade, bem como as possibilidades futuras para estudos envolvendo virtualização e execução cruzada de aplicações gráficas.

6.2 Pré-requisitos e recursos utilizados

O projeto utilizou os seguintes recursos e tecnologias:

- Sistema Operacional: *Windows* 11
- *Windows Subsystem for Linux 2* (WSL2)
- Distribuição *Linux*: *Ubuntu* (executada dentro do WSL2)
- Jogo: *Hollow Knight* (versão nativa para *Linux*)
- Ferramenta de TAS: *LibTAS*

Como material de apoio, foi utilizado o tutorial oficial do *LibTAS* disponível em: `https://clementgallet.github.io/libTAS/guides/wsl`

O *LibTAS* foi obtido a partir do repositório oficial de *releases*: `https://github.com/clementgallet/libTAS/releases/tag/v1.4.7`

6.3 Passo a passo

1. Instalação do WSL2 no *Windows* 11 por meio do *PowerShell* utilizando o comando:

   ```
   wsl --install
   ```

2. Instalação de uma distribuição *Linux* (*Ubuntu*) dentro do WSL2.

3. Download da versão *Linux* do jogo *Hollow Knight* e cópia dos arquivos para o sistema de arquivos do WSL.
4. Download e instalação do *LibTAS* utilizando o pacote '.deb' correspondente à arquitetura do sistema:

   ```
   sudo dpkg -i ./libtas_*_amd64.deb
   ```

5. Execução do *LibTAS* diretamente pelo terminal *Ubuntu* com o comando:

   ```
   libTAS
   ```

6. Tentativa inicial de execução do jogo via *script* 'start.sh', resultando em erro relacionado ao OpenGL.
7. Identificação do problema na variável de ambiente '$DISPLAY'.
8. Correção do erro gráfico ajustando a variável:

   ```
   export DISPLAY=:0
   ```

   (ou adicionando essa linha ao arquivo '.bashrc' para tornar a configuração permanente).
9. Execução do jogo forçando o uso do *OpenGL* colocando a *flag -force-opengl* no arquivo 'start.sh'. Ou se for uma execução direta usar o comando:

   ```
   ./HollowKnight -force-opengl
   ```

   na pasta onde está o executável (transformado em executável geralmente através do comando no *Ubuntu* 'chmod +x HollowKnight').
10. Testes de execução e observação de falhas gráficas e encerramento prematuro do jogo após aproximadamente 10 minutos.

6.4 Instalação

1. Abra o *PowerShell* no *Windows* 11.
2. Execute o comando:

   ```
   wsl --install
   ```

3. Reinicie o sistema, se necessário.
4. Abra o terminal *Ubuntu* (WSL).
5. Instale o *LibTAS* com:

   ```
   sudo dpkg -i ./libtas_*_amd64.deb
   ```

6. Copie os arquivos do jogo *Hollow Knight* (versão *Linux*) para o sistema de arquivos do WSL (por praticidade).

6.5 Execução

1. No terminal *Ubuntu* (WSL), configure a variável de ambiente:

   ```
   export DISPLAY=:0
   ```

2. Execute o *LibTAS* com:

   ```
   libTAS
   ```

3. Execução do jogo forçando o uso do *OpenGL* colocando a *flag -force-opengl* no arquivo 'start.sh'. Ou se for uma execução direta usar o comando:

   ```
   ./HollowKnight -force-opengl
   ```

   na pasta onde está o executável (transformado em executável geralmente através do comando no *Ubuntu* 'chmod +x HollowKnight').
4. Observe o comportamento do jogo durante a execução.

6.6 Bugs/problemas conhecidos

- O jogo encerra de forma inesperada após cerca de 10 minutos de execução.
- Ocorrência de distorções visuais nos *shaders*, afetando a estética do jogo.
- Possíveis incompatibilidades na conversão gráfica entre *Linux* e *Windows* via WSL2.
- Problemas iniciais de exibição gráfica causados por configuração incorreta da variável '$DISPLAY'.

6.7 Autor

- Rick Robert Tomaz Rezende (*GitHub*)

6.8 Demais anotações e referências

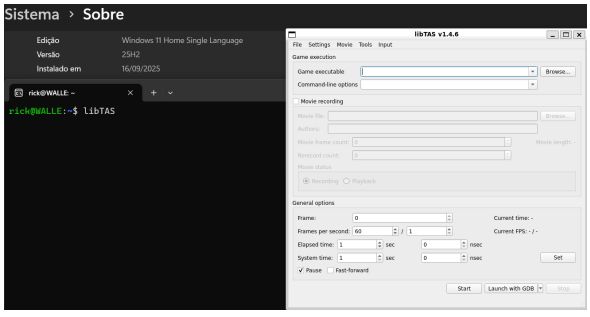
Este trabalho abre possibilidades para estudos futuros relacionados à execução de jogos nativos de outros sistemas operacionais, uso de máquinas virtuais e subsistemas, bem como aplicações que demandam maior processamento gráfico em ambientes virtualizados.

Adicionado posteriormente há um exemplo de sequência de inputs, contida no arquivo "Hollow Knight.ltm", que fazem uma execução não otimizada para derrotar o chefe Mãe-Mosca partindo do banco de *Dirtmouth*. Tal arquivo reproduz todos os inputs a partir de um *save* (ponto de salvamento) no segundo *slot* (espaço) onde o personagem está sentado no banco de *Dirtmouth*, para executar deve-se acessar através do *libTAS* na aba *movie* e em *open*, então selecionando o arquivo "Hollow Knight.ltm".

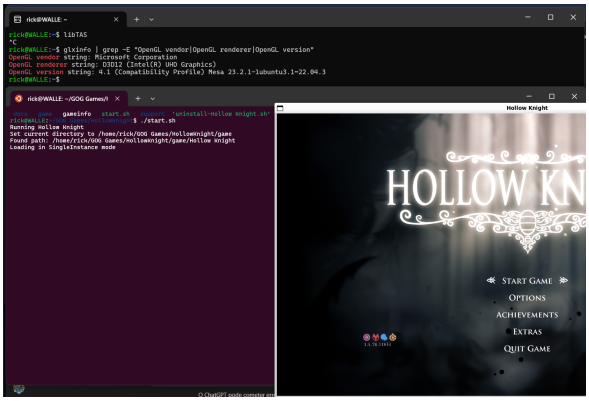
Para melhor visualização do todo, há um vídeo "Gravacao" que mostra iniciando o TAS e o jogo como também andando um pouco com o personagem. Para acessá-lo basta clicar *aqui*.

6.9 Imagens/screenshots

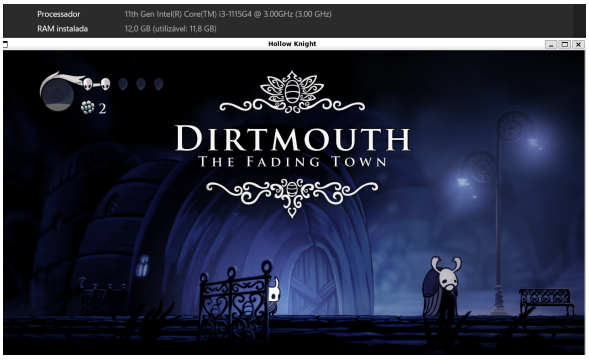
O primeiro teste e que depende de menos recursos gráficos é a execução da interface do *libTAS*, já que esta é simples e não requer um grande esforço dos recursos físicos. O resultado foi um sucesso conforme a seguinte imagem, que também mostra qual o Sistema Operacional que está sendo mostrado a interface:



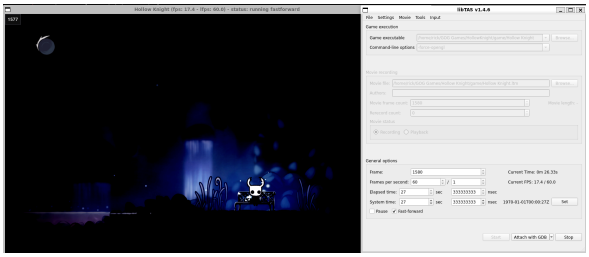
A outra peça importante para a execução da *Tool Assisted Speedrun* (TAS) é o jogo, e conforme a seguinte imagem é possível ver que foi viável abrir o jogo. Também como as informações que são mostradas durante a execução do jogo e alguns dados sobre a parte gráfica por parte do *openGL*.



Além disso, foi possível chegar em *Dirtmouth* (área inicial do jogo) sem com que o jogo parasse. Imagem acompanhada de informação da RAM e processado do dispositivo em que foi executado o jogo através do WSL2.



E por fim, foi possível executar o jogo utilizando a interface do *libTAS* que coloca informações como inputs recebidos e FPS (Frames Por Segundo) que o jogo está rodando. É possível ver também um dos *bugs* de *shader* comentados anteriormente.



6.10 Conclusão

Por mas que não seja uma execução ótima, ainda é possível, com paciência, dedicação e esforço realizar uma TAS com ferramentas e jogo de Linux no *Windows* através do *Windows Subsystem Linux 2* (WSL2) ao invés de uma abordagem *dual-boot*. Mesmo com suas limitações, conseguimos extrair o máximo das ferramentas usando a criatividade.

6.11 Repositório

O repositório com os arquivos e demais informações sobre o projeto se encontra em:

`https://github.com/Rick-Robert/JogoLinuxEmWindows`

Capítulo 7

Análise Inteligente de Interfaces

Felipe Bonadia de Oliveira Bravo

Luis Felipi Cruz de Souza

Acesse o repositório do projeto clicando aqui

7.1 Conceito do projeto

Com a rápida expansão das ferramentas de IA no desenvolvimento de software, a segurança muitas vezes é deixada em segundo plano. Este projeto visa auxiliar usuários, especialmente os menos técnicos, a identificar vulnerabilidades em suas aplicações. A partir de um print da tela, a ferramenta analisa possíveis brechas, oferece explicações detalhadas e gera testes específicos para validar a existência dessas falhas.

7.2 Pré-requisitos e recursos utilizados

Backend

O backend foi desenvolvido com o framework NestJS utilizando a linguagem TypeScript. A integração com inteligência artificial foi realizada através do SDK oficial do Google para o Gemini `(@google/genai)`

Frontend

O frontend foi desenvolvido com React e Vite, utilizando também a linguagem TypeScript. Para a renderização das respostas da IA, utilizou-se a biblioteca `marked`, permitindo a exibição de conteúdo em formato Markdown.

7.3 Passo a Passo

1. **Criação do backend**
   a) Construção da rota de análise
      i. Criação do endpoint de análise
      ii. Criação do prompt de descrição da imagem
      iii. Criação do prompt de análise de possíveis vulnerabilidades
      iv. Integração com o Gemini
2. **Criação do front end**
   a) Construção da tela principal
      i. Upload de arquivo
      ii. Mostrar análise em Markdown
   b) Integração com o backend

7.4 Instalação e Execução

> **Pré-requisito:** Certifique-se de ter o Node.js instalado.

Backend

1. Clone o repositório:

```
git clone https://github.com/Felipe
   Bonadia/An-lise-Inteligente-
   de-Interfaces.git
```

2. Entre na pasta do backend:

```
cd backend
```

3. Instale as dependências:

```
npm install
```

4. Crie um arquivo `.env` na raiz da pasta backend com base no arquivo `.env-example`:

```
GEMINI_API_KEY=sua_chave_api_do_gemini
FRONTEND_URL=url_do_frontend
```

5. Execute o backend:

```
nest start
```

O backend estará rodando em `http://localhost:3000`

Frontend

6. Abra um novo terminal na raiz do projeto, e entre na pasta do frontend:

```
cd frontend
```

7. Instale as dependências:

```
npm install
```

8. Crie um arquivo `.env` na raiz da pasta `frontend`, com base no arquivo `.env-example`:

```
VITE_BACKEND_URL=url_do_backend
```

9. Execute o frontend:

```
npm run dev
```

O frontend estará disponível em `http://localhost:5173`

7.5 Bugs/problemas conhecidos

Como utilizamos integração com modelos de linguagem (LLM), ressaltamos que podem ocorrer erros ou inconsistências nas respostas. Esta é uma ferramenta de auxílio e suporte, não devendo ser considerada uma fonte de verdade única ou absoluta.

7.6 Imagens/screenshots

Tela Inicial

Clique na imagem para visualizá-la em tamanho ampliado.

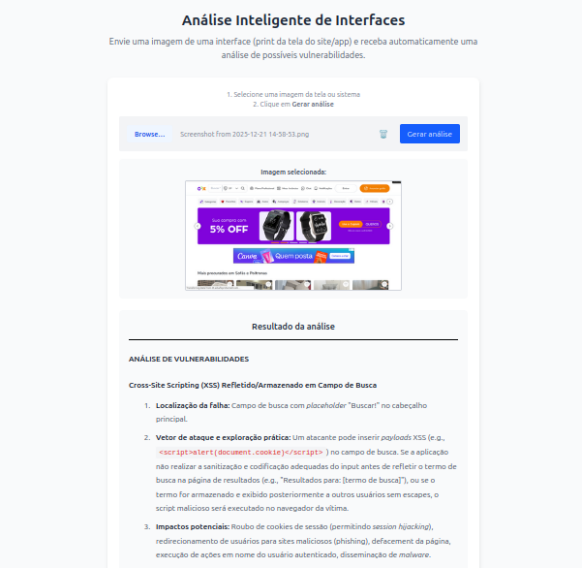
Escolha da interface

Clique na imagem para visualizá-la em tamanho ampliado.

Backend em execução

Clique na imagem para visualizá-la em tamanho ampliado.

Resultado



Clique na imagem para visualizá-la em tamanho ampliado.