UNIVERSIDADE FEDERAL DO PAMPA

DANIEL FELIPE TOMM

PAMPAI SECURITY: UM SISTEMA DE DETECÇÃO DE *PHISHING* EM TEMPO REAL BASEADO EM CASCATA DE CLASSIFICADORES BERT PARA ANÁLISE DE URLS

Bagé
2026

DANIEL FELIPE TOMM

PAMPAI SECURITY: UM SISTEMA DE DETECÇÃO DE *PHISHING* EM TEMPO REAL BASEADO EM CASCATA DE CLASSIFICADORES BERT PARA ANÁLISE DE URLS

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Engenharia de Computação como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Sandro da Silva Camargo

Bagé
2026

Ficha catalográfica elaborada automaticamente com os dados fornecidos pelo(a) autor(a) através do Módulo de Biblioteca do Sistema GURI (Gestão Unificada de Recursos Institucionais).

T661p Tomm, Daniel Felipe

PampAI Security: um sistema de detecção de *phishing* em tempo real baseado em cascata de classificadores BERT para análise de URLS / Daniel Felipe Tomm.

121 f.: il.

Orientador: Sandro da Silva Camargo

Trabalho de Conclusão de Curso (Graduação) – Universidade Federal do Pampa, Engenharia de Computação, 2026.

1. Engenharia social na web. 2. Aprendizado de máquina. 3. Extensão de navegador. 4. Processamento de linguagem natural. 5. Segurança cibernética. I. Título.

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
Universidade Federal do Pampa

DANIEL FELIPE TOMM

PAMPAI SECURITY: UM SISTEMA DE DETECÇÃO DE PHISHING EM TEMPO REAL BASEADO EM CASCATA DE CLASSIFICADORES BERT PARA ANÁLISE DE URLS

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Computação.

Trabalho de Conclusão de Curso defendido e aprovado em: 06 de Julho de 2026.

Banca examinadora:

Prof. Dr. Sandro da Silva Camargo
Orientador
(UNIPAMPA)

Prof. Dr. Érico Marcelo Hoff do Amaral
(UNIPAMPA)

Prof. Dr. Leonardo Bidese de Pinho
(UNIPAMPA)

Assinado eletronicamente por **LEONARDO BIDESE DE PINHO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 13/07/2026, às 09:14, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **ERICO MARCELO HOFF DO AMARAL**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 13/07/2026, às 10:51, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **SANDRO DA SILVA CAMARGO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 14/07/2026, às 21:32, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2085291** e o código CRC **2D2CE379**.

Referência: Processo nº 23100.010883/2026-81 SEI nº 2085291

AGRADECIMENTOS

Agradeço, em primeiro lugar, ao professor Sandro pela orientação, pelas discussões e pelas sugestões que foram fundamentais para a realização deste trabalho. Estendo o agradecimento aos colegas de curso, que ajudaram nas demais disciplinas com o compartilhamento de materiais, o esclarecimento de dúvidas e o apoio nos momentos de maior carga de estudos.

Agradeço também à minha família, pelo incentivo constante, pela compreensão e por acreditar em mim ao longo de toda esta caminhada acadêmica, e à minha namorada, Mileny Silva, pelo apoio, paciência e motivação contínua, que foram essenciais para que eu pudesse seguir em frente mesmo nos períodos mais desafiadores.

RESUMO

O *phishing* segue como um dos principais vetores de fraude na navegação *web* e em serviços de *e-mail*. Por meio de páginas que imitam serviços legítimos, induz usuários a entregar credenciais e dados financeiros. Este trabalho apresenta o PampAI Security, uma solução de detecção em tempo real para navegadores Chromium (*Manifest Version* 3) e para os *webmails* Gmail e Outlook, estes últimos em caráter de prova de conceito. A proposta inicial previa um classificador leve embarcado no cliente. Durante a execução, um modelo *Gradient Boosting* foi treinado e avaliado em mais de 600 mil Localizadores Uniformes de Recursos brasileiros, balanceados entre as classes, e descartado como decisor único, por apresentar taxa de falsos positivos próxima a 12 %, incompatível com a meta de proposta, fixada em valor não superior a 0,5 %. A arquitetura final deslocou o aprendizado de máquina para o servidor: no cliente, três camadas locais (*whitelist* de domínios confiáveis, *blacklist* de domínios maliciosos e *cache* de resultados recentes) resolvem a maioria dos endereços em tempo constante; na Interface de Programação de Aplicações (API), uma cascata em dois estágios combina o modelo principal DomURL-BERT, ajustado com o endereço e com dados de registro do domínio, com um classificador complementar acionado apenas na zona ambígua, em que o primeiro decide os casos de alta confiança e o segundo refina os ambíguos. Um *pipeline* dedicado ao *e-mail*, baseado em um modelo *transformer* destilado pré-treinado de terceiros com tradução automática do português para o inglês, foi integrado aos *webmails* e avaliado apenas em inglês. No conjunto de teste, o modelo principal alcançou F1-*score* próximo a 0,90 e área sob a curva ROC próxima a 0,96, com latência fim-a-fim adequada para uso interativo, abaixo do limiar de resposta perceptível pelo usuário. A meta de proposta, de taxa de falsos positivos não superior a 0,5 %, não foi atingida, e a redução desse indicador permanece como limitação reconhecida, registrada como trabalho futuro. O artefato é entregue como protótipo funcional, em caráter de prova de conceito, e não como produto comercial acabado. Os experimentos foram conduzidos com os recursos disponíveis ao autor; entre as contribuições estão a reformulação da arquitetura motivada por evidência empírica e a comparação de modelos sobre um conjunto de endereços brasileiros.

Palavras-chave: Engenharia social na web. aprendizado de máquina. extensão de navegador. processamento de linguagem natural. segurança cibernética.

ABSTRACT

Phishing remains one of the main fraud vectors in web browsing and email services, leading users to disclose credentials and financial data on pages that imitate legitimate services. This work presents PampAI Security, a real-time phishing detection solution for Chromium-based browsers (*Manifest Version* 3) and for the Gmail and Outlook webmail clients, the latter as a proof of concept. The initial proposal considered a lightweight classifier embedded in the client; during execution, a *Gradient Boosting* model was trained and evaluated on more than 600 thousand balanced Brazilian Uniform Resource Locators and discarded as a single decision-maker, since its false positive rate of about 12 % proved incompatible with the proposed target, set at no higher than 0.5 %. The final architecture moved machine learning to the server side: on the client, three local layers (a whitelist, a blacklist and a result cache) handle most addresses in constant time; on the Application Programming Interface (API), a cascade combines the primary DomURL-BERT model, fine-tuned on the address and on domain registration data, with a complementary classifier triggered only in the ambiguous region. A dedicated email analysis pipeline, based on a third-party pre-trained distilled *transformer* model and on Portuguese-to-English machine translation, was integrated into the webmail clients and evaluated in English only. On the test set, the primary model reached an F1-score close to 0.90 and an area under the ROC curve close to 0.96, with end-to-end latency suitable for interactive use, below the threshold of user-perceived response time. The proposed target was not reached and the reduction of false positives was registered as future work. Experiments were conducted with the resources available to the author; the contributions include the evidence-driven redesign of the architecture and the comparison of models over a set of Brazilian addresses.

Keywords: web-based social engineering; machine learning; browser extension; natural language processing; cybersecurity.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface* (Interface de Programação de Aplicações)
AUC	*Area Under the Curve* (Área sob a curva)
AUC-ROC	*Area Under the Receiver Operating Characteristic Curve* (Área sob a curva ROC)
BERT	*Bidirectional Encoder Representations from Transformers* (Representações de Codificador Bidirecional baseadas em *Transformers*)
CDN	*Content Delivery Network* (Rede de distribuição de conteúdo)
CLI	*Command Line Interface* (Interface de Linha de Comando)
CPU	*Central Processing Unit* (Unidade Central de Processamento)
DDR4	*Double Data Rate 4* (memória RAM padrão DDR4)
DL	*Deep Learning* (Aprendizado Profundo)
DNS	*Domain Name System* (Sistema de Nomes de Domínio)
DOM	*Document Object Model* (Modelo de Objeto de Documento)
FPR	*False Positive Rate* (Taxa de falsos positivos)
GBDT	*Gradient Boosting Decision Tree* (Árvore de decisão com *gradient boosting*)
GBM	*Histogram-based Gradient Boosting Machine* (Máquina de *gradient boosting* baseada em histograma)
GPO	*Group Policy Object* (Política de Grupo do Windows)
GPU	*Graphics Processing Unit* (Unidade de Processamento Gráfico)
HTML	*HyperText Markup Language* (Linguagem de Marcação de Hipertexto)
HTTP	*HyperText Transfer Protocol* (Protocolo de Transferência de Hipertexto)
HTTPS	*HyperText Transfer Protocol Secure* (Protocolo de Transferência de Hipertexto Seguro)
IA	Inteligência Artificial

IDN	*Internationalized Domain Name* (Nome de Domínio Internacionalizado)
IP	*Internet Protocol* (Protocolo de Internet)
LGPD	Lei Geral de Proteção de Dados Pessoais
LIME	*Local Interpretable Model-agnostic Explanations* (Explicações Locais Interpretáveis Independentes de Modelo)
LLM	*Large Language Model* (Modelo de Linguagem Grande)
LRU	*Least Recently Used* (política de substituição de *cache*: menos recentemente usado)
MCC	*Matthews Correlation Coefficient* (Coeficiente de Correlação de Matthews)
ML	*Machine Learning* (Aprendizado de Máquina)
MV3	*Manifest Version 3* (versão 3 do manifesto de extensões Chrome)
ONNX	*Open Neural Network Exchange* (Intercâmbio Aberto de Redes Neurais)
P2P	*Peer-to-Peer* (ponto a ponto, sem servidor central)
PR	*Precision–Recall* (Curva precisão–revocação)
RF	*Random Forest* (Floresta Aleatória)
RoBERTa	*Robustly Optimized BERT Pretraining Approach* (Abordagem Robustamente Otimizada para Pré-treinamento do BERT)
ROC	*Receiver Operating Characteristic* (Característica de Operação do Receptor)
SHAP	*SHapley Additive exPlanations* (Explicações Aditivas baseadas em valores de Shapley)
SLA	*Service Level Agreement* (Acordo de Nível de Serviço)
SMS	*Short Message Service* (Serviço de Mensagens de Texto Curtas)
SMTP	*Simple Mail Transfer Protocol* (Protocolo Simples de Transferência de Correio)
SPA	*Single Page Application* (Aplicação de Página Única)
SPF	*Sender Policy Framework* (Estrutura de Política do Remetente)

SVM	*Support Vector Machine* (Máquina de Vetores de Suporte)
SWG	*Secure Web Gateway* (*gateway* de segurança da *web* corporativa)
TCC	Trabalho de Conclusão de Curso
TLS	*Transport Layer Security* (Segurança da Camada de Transporte)
TPR	*True Positive Rate* (Taxa de verdadeiros positivos)
TTL	*Time To Live* (Tempo de vida do registro em *cache*)
UCI	*University of California, Irvine*
URL	*Uniform Resource Locator* (Localizador Uniforme de Recursos)
UX	*User Experience* (Experiência do Usuário)

SUMÁRIO

1 INTRODUÇÃO

A digitalização intensificou-se na última década e elevou a dependência de interações via *web*, com 5,5 bilhões de pessoas *online* em 2024 (International Telecommunication Union, 2024; United Nations Department of Economic and Social Affairs, 2022; OECD, 2024; ANSAR *et al.*, 2022). Em paralelo a essa expansão, o *Anti-Phishing Working Group* (APWG) registrou mais de 3,8 milhões de ataques únicos de *phishing* ao longo de 2025 (Anti-Phishing Working Group, 2026). Em 2024, o relatório anual do *Internet Crime Complaint Center* (IC3), unidade do *Federal Bureau of Investigation* (FBI), apontou *phishing* como a categoria mais reportada de crime cibernético, com mais de 193 mil reclamações (Federal Bureau of Investigation, 2025).

Esse movimento ampliou também a superfície de fraudes, e o *phishing* consolidou-se como um dos vetores mais prevalentes (DAS *et al.*, 2020; ZIENI; MASSARI; CALZAROSSA, 2023; Verizon Business, 2025). O termo designa tentativas de induzir o usuário, por engenharia social, a interagir com páginas ou Localizadores Uniformes de Recursos (*Uniform Resource Locator* – URL) que mimetizam serviços legítimos, a fim de coletar credenciais e dados financeiros. A detecção é dificultada pela semelhança visual com os sítios legítimos, pelo uso de domínios recém-registrados ou ofuscados e pela rápida adaptação das táticas (ZIENI; MASSARI; CALZAROSSA, 2023). Como resposta a esse cenário, o Aprendizado de Máquina (*Machine Learning* – ML) e o Aprendizado Profundo (*Deep Learning* – DL) têm sido investigados na classificação de URL em tempo real (ZIENI; MASSARI; CALZAROSSA, 2023; DO *et al.*, 2022). Modelos baseados em *transformers*, arquitetura de rede neural especializada em processar sequências de texto, passaram a tratar a URL como texto e a aprender padrões de ofuscação diretamente dos dados (MANERIKER *et al.*, 2021). Arquiteturas híbridas que distribuem a decisão entre cliente e servidor têm sido exploradas em extensões de navegador (ROSE; BASIR; HENG, 2022).

Essa necessidade de defesa em tempo real convive com uma preocupação crescente de privacidade e soberania sobre dados. Organizações e usuários que lidam com informações sensíveis podem preferir não delegar a terceiros a inspeção do seu tráfego de navegação, seja por exigências da Lei Geral de Proteção de Dados Pessoais (LGPD), seja por políticas internas de governança. Uma solução auto-hospedável permite que a análise de URL permaneça sob controle do próprio operador, reduzindo a dependência de serviços externos.

As seções seguintes formalizam o problema da pesquisa, os objetivos, o escopo e a organização do trabalho.

1.1 Problema da Pesquisa

O problema central que orientou este trabalho é: *como identificar tentativas de phishing diretamente no navegador e em webmails, com boa taxa de acerto e impacto mínimo na experiência de navegação do usuário?*

A exigência de atuar no navegador decorre das limitações das alternativas disponíveis. Listas de bloqueio mantidas por terceiros são reativas e levam horas a dias para incorporar campanhas recém-detectadas, janela em que o usuário permanece desprotegido. Serviços remotos de reputação, por sua vez, exigem o envio de cada URL visitada a um terceiro e adicionam latência à navegação. A detecção no navegador, em contraposição, atua no ponto de interação do usuário, antes ou durante o carregamento da página, e, nos *webmails*, antes mesmo do clique no *link*.

Para tornar a pergunta verificável, adotaram-se metas-alvo da fase de proposta, alinhadas ao regime de operação reportado na literatura de detecção de *phishing* por URL (MANERIKER *et al.*, 2021; LI *et al.*, 2024): taxa de verdadeiros positivos (*True Positive Rate* – TPR) em torno de 90 % e taxa de falsos positivos (*False Positive Rate* – FPR) próxima ou inferior a 0,5 %. Delimitou-se o escopo técnico ao navegador de *desktop* compatível com *Manifest Version* 3 e aos *webmails* Gmail e Outlook. O atendimento de cada meta é discutido no capítulo de Resultados e Discussão.

Da pergunta central derivam cinco questões de pesquisa, que tratam dos sinais de baixo custo extraíveis no cliente, da arquitetura que equilibra precisão e impacto na navegação, do tratamento de deriva de conceito e de estratégias de evasão, da garantia de privacidade, transparência das decisões e soberania sobre o tráfego de navegação, minimizando a dependência de serviços de terceiros, e do protocolo de avaliação em dados representativos. A solução foi ainda submetida a cinco critérios de sucesso declarados na proposta: desempenho, eficiência, experiência do usuário, privacidade e transparência, e reprodutibilidade. As questões de pesquisa e os critérios de sucesso, com suas definições operacionais, são detalhados no Capítulo 3 e no documento de requisitos, disponível no repositório público do projeto[1]. Nem todas as questões foram exaustivamente respondidas: deriva de conceito, evasão sistemática e explicações locais

[1]<https://github.com/TommDaniel/PampAI_Security/blob/main/docs/requisitos-do-projeto.md>

voltadas ao usuário foram tratadas em profundidade reduzida e registradas como trabalho futuro no Capítulo 6.

Dessa forma, o problema foi tratado como a concepção, a implementação e a validação inicial de uma solução integrada ao navegador, capaz de identificar *phishing* em URLs e em *webmails* com qualidade aceitável dentro das condições avaliadas, impacto limitado na navegação, respeito à privacidade e base reproduzível para evoluções futuras.

1.2 Objetivos

O objetivo geral deste trabalho foi construir e avaliar uma solução de detecção de *phishing* em tempo real[2] para navegadores compatíveis com *Manifest Version* 3 (MV3) e para os *webmails* Gmail e Outlook. A solução integra uma extensão de navegador a uma Interface de Programação de Aplicações (API) de análise apoiada por modelos de aprendizado de máquina.

Os objetivos específicos foram os seguintes:

1. Mapear e selecionar bases rotuladas adequadas ao contexto de navegação *web* e à análise de mensagens de *e-mail*.

2. Identificar sinais tratáveis no cliente: listas de permissão (*whitelist*), listas de bloqueio (*blacklist*) e *cache* local de resultados, junto a indicadores do Localizador Uniforme de Recursos (URL) usados pela API.

3. Investigar abordagens de aprendizado de máquina adequadas a um serviço de análise *server-side*, com a comparação de modelos clássicos de *gradient boosting* e de modelos *transformer* adaptados a URL, sob restrições de acurácia e tempo de resposta.

4. Projetar e implementar a arquitetura da extensão MV3 e do serviço de análise, de modo a integrar coleta, decisão local, consulta à API e alerta ao usuário para *web* e *e-mail*.

5. Definir e aplicar um protocolo de avaliação que comparasse os modelos candidatos por acurácia, taxa de falsos positivos e tempo de resposta, em conjuntos de teste e em validação empírica.

[2] Aqui, "tempo real" designa latência interativa no sentido de Nielsen (1993): o alvo é uma resposta abaixo do limiar de cerca de 1 s que preserva a sensação de fluxo de interação, e não garantias de sistemas de tempo real estritos. A definição operacional e as latências medidas constam das Seções 3.5 e 5.8.

6. Estabelecer diretrizes de minimização de dados, transparência das decisões, auto-hospedagem do servidor de inferência e segregação de eventos por organização contratante, de modo a sustentar o uso individual e corporativo.

7. Organizar artefatos, código e documentação em pacote reprodutível por meio de *containers*, suficiente para replicar a execução do serviço e os resultados obtidos.

Cabe delimitar o escopo da entrega. O trabalho produz uma prova de conceito: um protótipo funcional que demonstra a viabilidade técnica da arquitetura proposta e a submete a avaliação experimental, e não um produto comercial acabado. O amadurecimento para uso em produção fica delimitado como trabalho futuro (Capítulo 6) e abrange a redução da taxa de falsos positivos, a avaliação do módulo de *e-mail* sobre mensagens em português brasileiro, um canal de atualização das listas de permissão e de bloqueio independente do empacotamento e o endurecimento do provisionamento de organizações.

1.3 Organização do trabalho

O restante deste trabalho está organizado em cinco capítulos: o Capítulo 2, *Revisão de Literatura*, apresenta os conceitos fundamentais e situa o trabalho em relação à literatura; o Capítulo 3, *Metodologia*, descreve a classificação da pesquisa, os conjuntos de dados, as características extraídas, os modelos avaliados, as métricas, o procedimento experimental, o ferramental e os critérios de comparação adotados; o Capítulo 4, *Desenvolvimento da Solução*, descreve a arquitetura final adotada e as mudanças em relação à proposta inicial; o Capítulo 5, *Resultados e Discussão*, apresenta os experimentos realizados, a comparação entre as variantes avaliadas e a discussão em relação às metas declaradas; por fim, o Capítulo 6, *Conclusões*, sintetiza o trabalho desenvolvido, registra as principais contribuições, reconhece as limitações e indica trabalhos futuros.

2 REVISÃO DE LITERATURA

Este capítulo organiza os fundamentos teóricos e o estado da arte que sustentam o trabalho. A revisão começa pela definição operacional de *phishing* e por sua taxonomia (Seção 2.1), apresenta o aprendizado de máquina aplicado à detecção (Seção 2.2), discute modelos baseados em URL e em registros WHOIS (Seção 2.3), examina o papel das listas de bloqueio, das listas de permissão e do *cache* de reputação (Seção 2.4), descreve a arquitetura das extensões de navegador modernas (Seção 2.5) e fecha com os trabalhos relacionados, seção que posiciona explicitamente a contribuição deste trabalho de conclusão em relação à literatura recente (Seção 2.6).

2.1 Phishing

O termo *phishing* designa um ataque de engenharia social no qual o adversário induz a vítima a executar uma ação indesejada (revelar credenciais, fornecer dados financeiros ou instalar *malware*) a partir de mensagens e páginas que imitam fontes legítimas (DAS *et al.*, 2020; KUMAR; GUPTA *et al.*, 2022). O ataque combina canais de comunicação (correio eletrônico, *Short Message Service* (SMS), redes sociais, páginas *web*) com artefatos visuais e textuais cuidadosamente preparados, que exploram urgência, confiança ou curiosidade (ZIENI; MASSARI; CALZAROSSA, 2023). A natureza social do ataque torna a defesa puramente técnica insuficiente e motiva, ao mesmo tempo, soluções automatizadas que reduzam a janela de exposição do usuário.

A literatura classifica os ataques em campanhas em massa (*phishing* genérico), *spear phishing* direcionado a indivíduos, *whaling* focado em executivos e variantes como *clone phishing* e *pharming* (DAS *et al.*, 2020; ZIENI; MASSARI; CALZAROSSA, 2023). No *pharming*, o usuário acessa um endereço legítimo mas é redirecionado para um sítio malicioso por envenenamento do *Domain Name System* (DNS) ou por manipulação de arquivos de resolução de nomes (RAO; VAISHNAVI; PAIS, 2020). Os vetores complementares incluem *smishing* (mensagens SMS), *vishing* (chamadas de voz), anúncios maliciosos e páginas que exploram vulnerabilidades em navegadores ou *plugins* (ZIENI; MASSARI; CALZAROSSA, 2023).

Os indicadores específicos de URL maliciosa são bem documentados: uso de endereço *Internet Protocol* (IP) em vez de nome de domínio, domínios visualmente semelhantes (*Internationalized Domain Name* (IDN) *homograph*), domínios

recém-registrados, caminhos ofuscados, encurtadores e códigos QR que escondem o destino real do *link* (YAZDANI; TOORN; SPEROTTO, 2020; KUMAR; GUPTA *et al.*, 2022; DAS *et al.*, 2020; RAO; VAISHNAVI; PAIS, 2020). A combinação desses indicadores com sinais de infraestrutura, em particular a idade do domínio extraída de consultas WHOIS, forma a base empírica explorada pelos modelos discutidos nas seções seguintes. O WHOIS é o protocolo de consulta a bancos públicos de registro de domínios e retorna dados como registrante, data de criação e data de expiração.

Em cenários avançados, o atacante combina *phishing* com *ransomware*, *keyloggers* ou *botnets* para maximizar o impacto do comprometimento inicial (DO *et al.*, 2022; ZIENI; MASSARI; CALZAROSSA, 2023). Iniciativas de educação do usuário, como o aplicativo NoPhish (CANOVA *et al.*, 2014), complementam as defesas técnicas. Detecção e formação do usuário compõem, em conjunto, uma estratégia em camadas.

2.2 Aprendizado de máquina aplicado à detecção de phishing

O aprendizado de máquina (*Machine Learning* – ML) reúne métodos que ajustam parâmetros internos de um modelo a partir de exemplos, em vez de depender exclusivamente de regras programadas (MITCHELL, 1997; BISHOP, 2006). A literatura organiza ML em três grandes paradigmas: supervisionado, não supervisionado e por reforço (BISHOP, 2006; GOODFELLOW; BENGIO; COURVILLE, 2016). A detecção de *phishing* usa, predominantemente, o paradigma supervisionado, formulado como classificação binária (DAS *et al.*, 2020; ZIENI; MASSARI; CALZAROSSA, 2023; KUMAR; GUPTA *et al.*, 2022). Por essa razão, esta seção concentra a discussão nesse paradigma.

O fluxo de trabalho típico envolve sete passos: (i) coleta de URLs, páginas *HyperText Markup Language* (HTML) e rótulos; (ii) limpeza e normalização; (iii) extração e seleção de características; (iv) divisão em conjuntos de treinamento, validação e teste; (v) treinamento dos modelos; (vi) avaliação com métricas adequadas; e (vii) implantação e monitoramento (BISHOP, 2006; PEDREGOSA *et al.*, 2011; DAS *et al.*, 2020). Quando o conjunto de teste respeita ordem temporal, o procedimento simula a chegada de novas campanhas e expõe efeitos de *drift* de conceito. O *drift* de conceito é a mudança gradual dos padrões de ataque ao longo do tempo, que reduz a eficácia do modelo quando ele não é atualizado.

Entre os algoritmos supervisionados mais empregados destacam-se árvores de

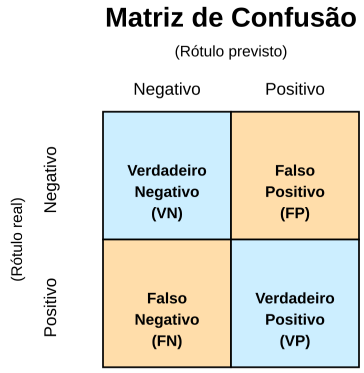
decisão, Floresta Aleatória (*Random Forest* – RF), métodos de *Gradient Boosting Decision Tree* (GBDT), que treinam uma sequência de árvores em que cada nova árvore corrige os erros das anteriores (XGBoost (CHEN; GUESTRIN, 2016), LightGBM (KE *et al.*, 2017) e CatBoost (PROKHORENKOVA *et al.*, 2018)), Máquina de Vetores de Suporte (*Support Vector Machine* – SVM), Regressão Logística e redes neurais profundas (DAS *et al.*, 2020; DO *et al.*, 2022; ZIENI; MASSARI; CALZAROSSA, 2023; GOODFELLOW; BENGIO; COURVILLE, 2016). Os *ensembles* de árvores, métodos que combinam as predições de vários modelos de árvore para reduzir variância e erro de generalização (BISHOP, 2006), costumam oferecer compromisso favorável entre acurácia, robustez e custo computacional. As redes neurais profundas, em especial as arquiteturas baseadas em *transformer* (MANERIKER *et al.*, 2021), oferecem representações mais ricas. Em troca, exigem treinamento mais custoso, apresentam latência maior e perdem explicabilidade. Esse compromisso é retomado nas Seções 2.3 e 2.6.

Dois desafios específicos tensionam o desempenho dos classificadores. O primeiro é o desbalanceamento de classes: muitos conjuntos públicos contêm muito mais URLs legítimas do que de *phishing* e, sem cuidado, o modelo passa a favorecer a classe majoritária (HE; GARCIA, 2009; DAS *et al.*, 2020). O segundo é o *drift* de conceito: novas campanhas modificam padrões de URL e de mensagem ao longo do tempo e degradam modelos estáticos (DAS *et al.*, 2020; ZIENI; MASSARI; CALZAROSSA, 2023). Estratégias de reamostragem, ponderação de classes, métricas robustas a desbalanceamento e adaptação contínua a partir de dados rotulados parcialmente são pesquisadas em paralelo na literatura de detecção de intrusões e segurança (SKOPIK; WURZENBERGER; LANDAUER, 2021; YANG *et al.*, 2025; DO *et al.*, 2022).

Quanto às métricas de avaliação, a análise do classificador supervisionado parte da matriz de confusão, que resume acertos e erros em cada classe (BISHOP, 2006). A Figura 1, adaptada de Cechinel e Camargo (CECHINEL; CAMARGO, 2020), apresenta a forma usual da matriz para problemas binários: Verdadeiros Positivos (VP), Falsos Positivos (FP), Falsos Negativos (FN) e Verdadeiros Negativos (VN).

A partir da matriz, definem-se a acurácia, a *precision*, o *recall* (ou *true positive rate* – TPR) e o F1-score, conforme as Equações 1 a 4 (BISHOP, 2006; FAWCETT, 2006). O F1-score combina *precision* e *recall* pela média harmônica e é particularmente útil em cenários desbalanceados, em que a acurácia isolada engana (HE; GARCIA, 2009; FAWCETT, 2006).

Figura 1 – Matriz de confusão binária. Os quadrantes da diagonal principal (VP e VN) correspondem aos acertos do classificador; os demais (FP e FN), aos erros.



Fonte: Autor (2026), adaptado de CECHINEL; CAMARGO (2020).

$$\text{Acurácia} = \frac{VP + VN}{VP + FP + FN + VN} \tag{1}$$

$$\text{Precision} = \frac{VP}{VP + FP} \tag{2}$$

$$\text{Recall} = \frac{VP}{VP + FN} \tag{3}$$

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

A curva *Receiver Operating Characteristic* (ROC) relaciona TPR e *false positive rate* (FPR) para diferentes limiares de decisão, e a Área sob a curva ROC (*Area Under the Curve* – AUC) sintetiza a capacidade global de separação entre classes; valores próximos de um indicam melhor separação (FAWCETT, 2006). Em problemas de segurança, o desempenho médio importa menos do que o TPR mantido em regimes de FPR muito baixos: detectar a maior parte dos ataques sem sobrecarregar o usuário com alarmes falsos é o critério prático (DAS *et al.*, 2020; DO *et al.*, 2022). Por essa razão, a avaliação deste trabalho considera TPR em pontos de FPR-alvo, área sob a curva precisão-revocação (*Precision–Recall* – PR) e Coeficiente de Correlação de Matthews (*Matthews Correlation*

Coefficient – MCC). A área sob a curva PR concentra o foco na classe positiva (*phishing*) e é mais sensível em conjuntos nos quais os ataques são minoria. O detalhamento dessas métricas aparece na Seção 3.5.

2.3 Modelos baseados em URL e WHOIS

Modelos que classificam URLs sem carregar o conteúdo da página formam uma família distinta de detectores de *phishing*, atrativa por agir antes da renderização e por independer do *Document Object Model* (DOM) ou de capturas de tela (MANERIKER *et al.*, 2021; RAO; VAISHNAVI; PAIS, 2020). As características usadas dividem-se em três blocos: (i) lexicais (comprimento da URL, número de subdomínios, presença de IP em vez de nome, caracteres especiais, n-gramas); (ii) sintáticas e estruturais (uso de portas não padrão, complexidade do caminho, parâmetros de *query*); e (iii) baseadas em registros externos, especialmente WHOIS (idade do domínio, registrante, data de expiração, presença de privacidade WHOIS) (DUA; GRAFF, 2017; RAO; VAISHNAVI; PAIS, 2020; MOSA *et al.*, 2023; YAZDANI; TOORN; SPEROTTO, 2020). A combinação de URL e WHOIS dá ao modelo acesso a sinais que dependem do tempo de vida do domínio, indicador empiricamente forte para *phishing* (YAZDANI; TOORN; SPEROTTO, 2020; RAO; VAISHNAVI; PAIS, 2020). Quanto ao desempenho, a métrica decisiva para essa família não é a acurácia isolada, mas o TPR sustentado em regime de FPR muito baixo, conforme apresentado na Seção 2.2. O URLTran, avaliado em ambiente industrial e em larga escala, reporta Revocação de 86,80 % a uma FPR de 0,01 % (MANERIKER *et al.*, 2021). Os valores da literatura, contudo, não são diretamente comparáveis entre si, porque cada trabalho adota conjunto de dados, divisão de treino e teste e protocolo de avaliação próprios.

Modelos clássicos (Floresta Aleatória, GBDT e SVM) continuam competitivos sobre essas características quando o conjunto de treino é volumoso, e são empregados tanto isoladamente quanto em *ensembles* (CALZAROSSA; GIUDICI; ZIENI, 2024; FAJAR; YAZID; BUDI, 2024; DO *et al.*, 2022). A explicabilidade nativa dessas famílias (por importância de *features* ou por análise SHAP, *SHapley Additive exPlanations*) justifica seu uso em cenários nos quais o operador precisa auditar a decisão.

A partir da publicação de URLTran, Maneriker *et al.* (MANERIKER *et al.*, 2021) demonstraram que tratar a URL como sequência de subpalavras e processá-la com um *transformer* pré-treinado (BERT, *Bidirectional Encoder*

Representations from Transformers) supera abordagens com características manuais em conjuntos suficientemente grandes. Essa estratégia consolida uma família de modelos especializados em URL, da qual o DomURL-BERT é um representante recente (MAHDAOUY *et al.*, 2024); é justamente esse modelo, em sua variante URL + WHOIS, que este trabalho adota como classificador principal no servidor (ver Seção 2.6). Como linha de comparação, são também relevantes modelos de *transformer* pré-treinados em *corpora* de cibersegurança (*corpus*, no singular, é uma coleção de textos do domínio usada na etapa de pré-treinamento, em que o modelo aprende o vocabulário e os padrões da área antes do ajuste fino em uma tarefa específica), como o SecureBERT (AGHAEI *et al.*, 2022), que reaproveita o vocabulário e os pesos para tarefas de segurança.

A profundidade dos modelos baseados em *transformer* traz, contudo, três compromissos. Primeiro, o custo computacional de treinamento e inferência cresce em ordens de grandeza em comparação com GBDT (KUMAR; GUPTA *et al.*, 2022): modelos com centenas de milhões de parâmetros exigem unidade de processamento gráfico (GPU) para o treinamento e tornam a inferência em unidade central de processamento (CPU) cara, como ilustra o DomURL-BERT adotado neste trabalho, com aproximadamente 110 milhões de parâmetros. Segundo, a explicabilidade fica menos imediata do que em modelos de árvore: estes oferecem importância de *features* e análise SHAP de forma direta, enquanto *transformers* exigem análise pós-hoc por mapas de atenção, duas vias empregadas no capítulo de Resultados e Discussão. Terceiro, a generalização para domínios fora do conjunto de treino depende fortemente da diversidade dos dados, e o ganho marginal de incluir mais modalidades (DOM, capturas de tela, redirecionamentos) nem sempre compensa o ruído introduzido, comportamento observado empiricamente neste trabalho. Esse ponto é retomado na Seção 2.6.

2.4 Listas de bloqueio, listas de permissão e *cache* de reputação

Listas de bloqueio (*blacklists*) reúnem URLs ou domínios previamente classificados como maliciosos e são amplamente empregadas pela simplicidade e pelo baixo custo de consulta (DAS *et al.*, 2020). Fontes consolidadas incluem PhishTank (OpenDNS, 2025), OpenPhish (OpenPhish, 2026), URLhaus (abuse.ch, 2026), PhishStats (PhishStats, 2026) e o repositório Phishing.Database (Phishing-Database community, 2026); a combinação dessas fontes amplia cobertura, à custa de duplicidade e de variabilidade de qualidade entre feeds. Apesar da utilidade, listas de bloqueio são

reativas e pouco eficazes contra ataques de primeira hora; os domínios descartáveis foram apontados como ponto fraco recorrente em campanhas de curta duração (ZIENI; MASSARI; CALZAROSSA, 2023).

Listas de permissão (*whitelists*) operam pelo princípio oposto: catalogam domínios reconhecidamente legítimos e dispensam análise adicional. Rankings de popularidade como o Tranco (POCHAT *et al.*, 2019) e o Majestic Million (Majestic-12 Ltd., 2026) são fontes recorrentes; em conjunto com listas curadas regionalmente, formam o núcleo de uma camada local que evita falsos positivos sobre domínios de alto perfil. O Tranco, em particular, é um *ranking* projetado para pesquisa, com mecanismos contra manipulação (POCHAT *et al.*, 2019).

O *cache* de reputação no cliente é uma terceira camada de aceleração. Resultados recentes de classificações são memorizados localmente com tempo de vida (*Time To Live* – TTL) controlado, o que evita consultas repetidas a serviços externos e reduz a latência percebida pelo usuário. A escolha do TTL precisa refletir o equilíbrio entre frescor da decisão e custo da consulta: TTLs curtos protegem contra mudanças rápidas de reputação, mas reduzem o ganho de *cache*; TTLs longos reduzem chamadas, mas atrasam a propagação de listas atualizadas. Esse compromisso e o uso combinado de *whitelist*, *blacklist* e *cache* são frequentemente discutidos como primeira linha de defesa, na qual o classificador supervisionado atua apenas sobre URLs cuja reputação local não é decisiva (KUMAR; GUPTA *et al.*, 2022; LI *et al.*, 2024).

A limitação central das três camadas é a latência de atualização frente a novos ataques. Listas levam horas a dias para incorporar campanhas recém-detectadas; rankings de popularidade são atualizados em escala mensal; *caches* locais herdam essa janela. O trabalho de adaptação contínua e detecção de *drift* discutido na Seção 2.2 justifica acoplar essas listas a um classificador supervisionado capaz de lidar com URLs ainda não observadas (SKOPIK; WURZENBERGER; LANDAUER, 2021; YANG *et al.*, 2025).

2.5 Extensões de navegador

A extensão de navegador (*browser extension*) é um módulo de *software* que estende as funcionalidades padrão de Chrome, Firefox e similares e se integra à interface do navegador e ao conteúdo das páginas visitadas (Google Chrome Developers, 2025). Tipicamente, a extensão é composta por um manifesto (*manifest.json*), um *service worker* (ou *background script*), *content scripts* injetados nas páginas, páginas de opções e

recursos auxiliares (ícones, folhas de estilo) (Google Chrome Developers, 2025; ROSE; BASIR; HENG, 2022).

Os navegadores expõem interfaces de programação (*Application Programming Interface* – API) específicas para extensões, agrupadas no modelo WebExtensions. Essas APIs permitem interceptar requisições do *HyperText Transfer Protocol* (HTTP), acessar informações de abas, ler e alterar o DOM por meio de *content scripts*, armazenar dados localmente e estabelecer canais entre os componentes internos da extensão (PERROTTA; HAO, 2018). Esse ecossistema viabiliza tanto aplicações de produtividade quanto extensões de segurança, como bloqueadores de anúncios, filtros de rastreamento e mecanismos de detecção de *phishing* (ROSE; BASIR; HENG, 2022; ZIENI; MASSARI; CALZAROSSA, 2023).

A extensão de segurança herda riscos do próprio modelo: permissões excessivas podem registrar credenciais, injetar conteúdo malicioso ou recrutar o navegador para *botnets* controladas remotamente (PERROTTA; HAO, 2018; PICAZO-SANCHEZ; TAPIADOR; SCHNEIDER, 2020). O reconhecimento desse risco motiva o princípio de *least privilege* no projeto da extensão e justifica a escolha de manter o conjunto de permissões mínimo (SKOPIK; WURZENBERGER; LANDAUER, 2021; ROSE; BASIR; HENG, 2022).

A migração para o *Manifest Version* 3 (MV3) em navegadores baseados em Chromium reorganiza a forma como extensões interceptam tráfego. O mecanismo *webRequest* bloqueante, antes usado para examinar requisições de forma síncrona em tempo real, foi substituído por regras de bloqueio declaradas previamente pela extensão. O componente que rodava continuamente em segundo plano (*background page*) deu lugar ao *service worker*, processo de curta duração que o navegador encerra quando ocioso (PANTELAIOS; KAPRAVELOS, 2024; Google Chrome Developers, 2025). Essas mudanças afetam diretamente as extensões que inspecionam conteúdo durante a navegação. Em particular, restringem a filtragem síncrona pesada no cliente e tornam mais atrativa a combinação de listas declarativas locais com serviços remotos.

A inferência de modelos de aprendizado de máquina dentro da extensão impõe restrições adicionais. Como a extensão compartilha o ambiente de execução com o navegador, modelos embarcados precisam ser compactos e rápidos para não degradar a experiência (DO *et al.*, 2022; MANERIKER *et al.*, 2021). Os trabalhos que adotam essa via discutem explicitamente o compromisso entre acurácia, tempo de resposta e uso de memória e recorrem com frequência a esquemas de *cache* local e a chamadas remotas em

casos ambíguos (ROSE; BASIR; HENG, 2022; MALIK *et al.*, 2024; RAO; VAISHNAVI; PAIS, 2020).

A quantificação desse compromisso organiza-se, na prática, em três eixos de medição. A latência de inferência é reportada por percentis (P50, P95) do tempo entre a submissão da URL e a emissão do rótulo. A memória é avaliada pelo tamanho do *bundle* da extensão e pelo consumo de *heap* em execução. A acurácia e o F1 são calculados sobre um conjunto de teste fixo. A literatura não padroniza esse protocolo, o que dificulta a comparação direta entre trabalhos. O protocolo deste trabalho é definido no Capítulo 3, no qual a latência fim-a-fim é monitorada pelos percentis P50 e P95, e o impacto de disco, memória e tempo da instrumentação local da extensão é quantificado no capítulo de Desenvolvimento (Tabela 1), que distingue valores medidos de estimados. Em síntese, as restrições técnicas impostas pelo MV3 e os limites da inferência na extensão tornam a divisão de tarefas entre cliente e servidor uma decisão central no projeto de extensões de segurança.

2.6 Trabalhos relacionados

Esta seção apresenta soluções da literatura relacionadas à detecção de *phishing* por URL, à execução em extensão de navegador e à identificação de homógrafos, sem antecipar as escolhas de projeto deste trabalho. Quatro revisões recentes sintetizam a literatura de detecção de *phishing*: Das *et al.* (2020), Do *et al.* (2022), Kumar, Gupta *et al.* (2022) e Li *et al.* (2024). As três primeiras consolidam a literatura anterior; Li *et al.* atualizam o panorama até 2024 e cobrem desde modelos clássicos até arquiteturas com *transformers* e grafos. As revisões convergem em que os melhores resultados atuais combinam representações vetoriais densas de URL e conteúdo, com ênfase crescente em arquiteturas *transformer* e em GBDT treinados sobre *features* ricas (GOODFELLOW; BENGIO; COURVILLE, 2016; CHEN; GUESTRIN, 2016; KE *et al.*, 2017; MOSA *et al.*, 2023).

No eixo das abordagens baseadas exclusivamente em URL, Maneriker *et al.* (2021) introduzem o URLTran, que aplica BERT a sequências de URL e supera *baselines* com características manuais em ambientes de larga escala. CatchPhish, de Rao, Vaishnavi e Pais (2020), ilustra o ganho que pode ser obtido com um classificador clássico que opera apenas sobre características lexicais e estruturais da URL ao mesmo tempo em que evidencia o limite dessa estratégia diante de campanhas com URLs muito curtas ou

ofuscadas. No eixo de homógrafos e infraestrutura, Yazdani, Toorn e Sperotto (2020) propõem detecção ativa de IDN homográficos por DNS.

No eixo de execução em extensão de navegador, Rose, Basir e Heng (2022) descrevem mecanismo de detecção implementado como extensão Chrome, que intercepta a navegação e classifica a URL antes da interação do usuário. Malik *et al.* (2024) avançam essa direção e integram GBDT a um *plugin* voltado à detecção de URLs falsas, com ênfase em manter boa acurácia sem comprometer a experiência. Ambos os trabalhos discutem a tensão entre acurácia do modelo e fadiga de alerta. A migração para o MV3 reorganiza essas escolhas: restringe filtragem síncrona pesada e torna declarativas as regras locais (PANTELAIOS; KAPRAVELOS, 2024; Google Chrome Developers, 2025).

Modelos compactos baseados em *transformer* foram propostos para suportar inferência local. A proposta PhishLang adapta o MobileBERT à classificação de URLs e é citada como alternativa para execução no cliente (ROY; NILIZADEH, 2024). A proposta PhishGuard adota um conjunto multicamada de classificadores GBDT (Floresta Aleatória, Gradient Boosting, CatBoost e XGBoost) treinados sobre características da URL para a tarefa de detecção (OVI; RAHMAN; HOSSAIN, 2024). TinyBERT (JIAO *et al.*, 2020) e DistilBERT (SANH *et al.*, 2019) aparecem como exemplares da família de *transformers* destilados, com *footprint* reduzido e latência menor; DistilBERT, em particular, é usado neste trabalho como classificador de e-mail. No eixo visual, a proposta VisualPhishNet adota uma rede convolucional *triplet* para identificar páginas falsas pela aparência, com acurácia elevada ao custo de modelos maiores e latência variável (ABDELNABI; KROMBHOLZ; FRITZ, 2020). A proposta MultiPhishGuard, por sua vez, descreve um sistema multiagente baseado em modelo de linguagem grande (*Large Language Model* – LLM) para detecção de *phishing* em mensagens de *e-mail*, que combina sinais de texto, URL e metadados da mensagem (XUE *et al.*, 2025).

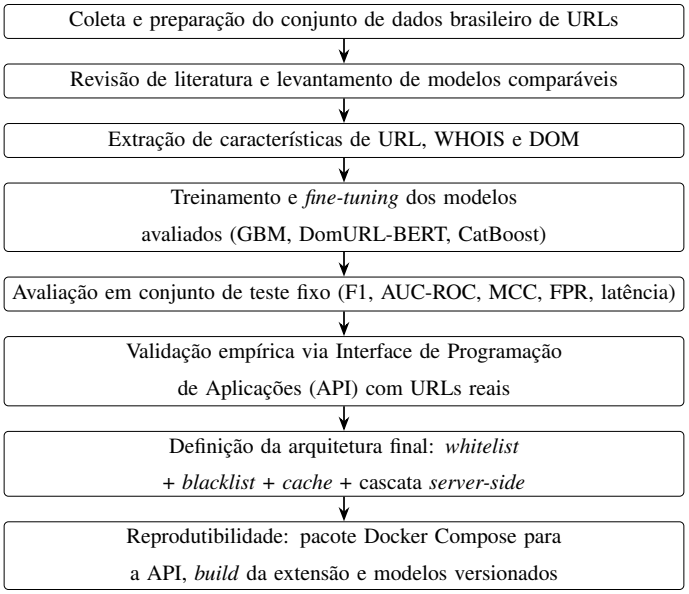
A revisão acima sustenta o posicionamento deste trabalho de conclusão. O PampAI Security adota uma arquitetura na qual o cliente decide localmente em $O(1)$ por listas de permissão, listas de bloqueio e *cache* de reputação, e delega a uma API o classificador de aprendizado de máquina; no servidor, o classificador principal é uma cascata composta por DomURL-BERT URL + WHOIS e por um GBDT CatBoost que decide na zona ambígua, a faixa de probabilidade em que o modelo principal não tem confiança suficiente para decidir sozinho. A escolha aproxima o trabalho de Rose, Basir e Heng (2022) e de Malik *et al.* (2024) no princípio de levar a detecção para a extensão. Diferencia-se, contudo, ao retirar o modelo de aprendizado do navegador,

decisão de projeto cuja justificativa empírica é apresentada no capítulo de Resultados e Discussão. Em relação a URLTran (MANERIKER *et al.*, 2021) e a outros usos de *transformer* sobre URL, o presente trabalho contribui com a comparação direta entre uma variante apenas com URL + WHOIS e uma variante adicional com sinais de DOM e outras características da página, cujos resultados são discutidos no capítulo de Resultados e Discussão. Frente a propostas que mantêm a inferência no cliente, como o PhishLang baseado em MobileBERT (ROY; NILIZADEH, 2024), a opção feita aqui é deliberadamente diferente: a inferência fica em servidor e o cliente fica responsável apenas pelas camadas $O(1)$, em troca de simplicidade de manutenção e menor consumo de recursos no navegador. A discussão detalhada dos resultados, inclusive os *trade-offs* e as limitações reconhecidas, é apresentada no capítulo de Resultados e Discussão.

3 METODOLOGIA

Este capítulo descreve o procedimento metodológico adotado na construção e na avaliação da solução de detecção de *phishing* apresentada neste trabalho. O capítulo está organizado em dez seções principais. A Seção 3.1 apresenta a classificação da pesquisa. A Seção 3.2 descreve os dados utilizados. A Seção 3.3 detalha as características extraídas. A Seção 3.4 lista os modelos avaliados. A Seção 3.5 enumera as métricas de avaliação adotadas. A Seção 3.6 descreve o procedimento experimental. A Seção 3.7 apresenta o ferramental utilizado no trabalho. A Seção 3.8 declara os critérios de comparação entre os modelos. A Seção 3.9 trata dos aspectos éticos e da conformidade com a Lei Geral de Proteção de Dados Pessoais (LGPD), e a Seção 3.10 discute os riscos identificados durante a execução do trabalho. As escolhas de engenharia que materializam essa metodologia, bem como a arquitetura final da solução, são apresentadas no Capítulo 4. A Figura 2 resume as etapas metodológicas descritas neste capítulo.

Figura 2 – Visão geral das etapas metodológicas do projeto.



Fonte: Autor (2026).

O desenvolvimento e a avaliação foram orientados por cinco questões de pesquisa derivadas do problema central (Seção 1.1):

1. quais sinais de baixo custo podem ser extraídos no cliente sem degradar a experiência do usuário (atributos léxicos da URL, características básicas da página e metadados leves do domínio);

2. qual arquitetura equilibra precisão e impacto na navegação, com verificação leve no cliente e verificação robusta no servidor acionada apenas em casos ambíguos;

3. como lidar com deriva de conceito e estratégias de evasão (homógrafos, domínios recém-registrados, ofuscação) preservando TPR alto e FPR baixo ao longo do tempo;

4. como garantir privacidade e transparência na decisão do modelo, minimizando a coleta de dados sensíveis e oferecendo justificativas compreensíveis das classificações;

5. como avaliar a solução em dados representativos de tráfego real, com protocolo reprodutível e o conjunto de métricas adotado.

Nem todas foram exaustivamente respondidas: deriva de conceito, evasão sistemática e explicações locais voltadas ao usuário foram tratadas em profundidade reduzida e registradas como trabalho futuro no Capítulo 6.

A solução foi avaliada ainda contra cinco critérios de sucesso declarados na fase de proposta: (i) **desempenho**, entendido como a aproximação das metas-alvo de TPR e FPR, com a latência fim-a-fim introduzida pela solução monitorada pelos percentis P50 e P95 (NIELSEN, 1993); (ii) **eficiência**, com uso de memória e de processamento compatível com a navegação típica, sem travamentos perceptíveis nem aumento expressivo de consumo de recursos; (iii) **experiência do usuário**, com mensagens claras, poucas interrupções e ausência de falsos alarmes recorrentes que levem o usuário a ignorar os avisos; (iv) **privacidade e transparência**, com processamento local por padrão quando viável, envio mínimo de dados quando o apoio remoto for necessário e justificativas sucintas das decisões; e (v) **reprodutibilidade**, com registro e disponibilização do protocolo de avaliação, da divisão de dados e dos parâmetros relevantes, de modo a permitir a replicação dos resultados. O atendimento desses critérios é discutido no capítulo de Resultados e Discussão.

3.1 Classificação da pesquisa

Este trabalho enquadra-se, simultaneamente, nas seguintes categorias clássicas de pesquisa, conforme tipologias consagradas em metodologia científica:

- **Pesquisa científica *aplicada*:** o objetivo central é gerar conhecimento com

aplicação prática para detecção e mitigação de *phishing* no contexto do navegador e entregar artefatos concretos (modelos, extensão *Web*, Interface de Programação de Aplicações (API) e *scripts* de reprodução). A natureza é resolutiva e dirigida a um problema específico de segurança do usuário final, distinta de investigação puramente teórica.

- **Abordagem *quantitativa*:** todo o ciclo de avaliação é numérico e estatístico. Os modelos foram comparados por métricas de classificação binária (acurácia, *precision*, *recall*, F1-*score*, área sob a curva ROC, área sob a curva PR e coeficiente de correlação de Matthews) e por medidas de latência (percentis P50, P95 e P99), com particionamento estratificado em treino e teste, validação cruzada estratificada e estimação de probabilidades calibradas. Estudos comparativos foram conduzidos para mensurar a contribuição de grupos de características (URL apenas *vs.* URL combinada com WHOIS *vs.* URL combinada com WHOIS e *Document Object Model* (DOM, a representação hierárquica em árvore dos elementos de uma página HTML, construída pelo navegador ao carregá-la e usada por *scripts* para acessar e manipular o documento)).

- ***Experimental*:** variáveis independentes foram manipuladas sob condições controladas para observar efeitos sobre o desempenho. Entre essas variáveis estão a seleção e a combinação de características, as famílias de modelos avaliadas (*Histogram-based Gradient Boosting* (CHEN; GUESTRIN, 2016; KE *et al.*, 2017), *transformers* especializados (MANERIKER *et al.*, 2021) e *boosting* categórico (PROKHORENKOVA *et al.*, 2018)) e os limiares de decisão da arquitetura em cascata. O delineamento incluiu controle de ambiente (*notebooks* versionados em Google Colab e contêineres Docker em máquina pessoal), repetições controladas e mensuração padronizada.

- ***Exploratória*:** as fases iniciais tiveram caráter exploratório, com revisão focalizada do estado da arte de detecção de *phishing* baseada em *transformers*, mapeamento de taxonomias de sinais (URL, WHOIS, DOM, *e-mail*), *profiling* do conjunto de dados (limpeza, deduplicação, normalização) e prospecção de indicadores promissores. Essa etapa permitiu delimitar o problema, escolher modelos candidatos e formular hipóteses testáveis antes do delineamento final.

Observação. Este estudo adota, simultaneamente, natureza *aplicada*, abordagem *quantitativa* e procedimentos *experimentais*, com fase inicial *exploratória*. A combinação

é coerente porque as tipologias metodológicas não são mutuamente excludentes: o trabalho partiu de um mapeamento exploratório e avançou para experimentos controlados orientados a um artefato prático.

3.2 Dados utilizados

A maior parte dos experimentos foi conduzida sobre um conjunto de dados de URLs brasileiras, a seguir designado conjunto BR. O conjunto BR reuniu **660 838 URLs**, balanceadas em 50 % *phishing* (330 419 amostras com rótulo positivo) e 50 % legítimas (330 419 amostras com rótulo negativo). Cada registro contém quatro colunas: `url`, `label`, `source` (fonte de origem) e `region` (recorte regional, quando aplicável).

A composição por fonte é dominada por uma única origem. A fonte `kmack_phishing_urls` (kmack, 2024) contribuiu com 656 122 amostras (99,3 % do total). As demais fontes correspondem a *rankings* de domínios populares (Tranco, com 1 977 amostras; Majestic, com 1 267 amostras), repositórios públicos de *phishing* (PhishTank, com 609 amostras; OpenPhish, com 3 amostras) e curadorias regionais de URLs legítimas brasileiras que abrangem portais de governo, universidades e empresas dos estados do Paraná, Rio Grande do Sul e Santa Catarina, além de curadorias de saúde, financeiro e nacional, que somam aproximadamente 859 amostras (cerca de 0,4 % do total). Esta dominância de uma única fonte é uma condição experimental relevante: o desempenho reportado nas métricas globais é determinado, em larga medida, pelo comportamento do modelo nas URLs de `kmack_phishing_urls`, e não nas curadorias regionais brasileiras. Essa ressalva é retomada na discussão de limitações no capítulo de Resultados e Discussão.

A partir do conjunto BR, foram derivados dois subconjuntos para a validação empírica conduzida via API durante o desenvolvimento. O primeiro, `validacao_modelo.json`, reuniu 62 URLs (36 legítimas brasileiras de alto perfil e 26 *phishing* simuladas) e foi usado para avaliar o comportamento do DomURL-BERT URL+WHOIS **sozinho**, sem proteção de listas. O segundo, `validacao_cascata.json`, reuniu 101 URLs (51 legítimas e 50 *phishing*) e foi usado para comparar três modos de inferência: BERT puro, CatBoost puro e cascata BERT+CatBoost. Esses dois subconjuntos não substituem a rodada formal de avaliação em campo, conduzida e reportada na Tabela 10, mas sustentam a discussão qualitativa apresentada no capítulo de Resultados e Discussão.

Para o módulo de análise de *e-mail*, foi usado o *benchmark* associado ao modelo DistilBERT *fine-tuned* para detecção de *phishing* em *e-mail* (SANH *et al.*, 2019), avaliado em *notebook* dedicado. Os detalhes do conjunto e do procedimento de inferência aparecem no Capítulo 4.

O pré-processamento dos dados envolveu limpeza (remoção de duplicatas exatas), normalização do domínio e dos parâmetros, separação consistente de *path* e *query*, e divisão em partições estratificadas de treino e teste, sem conjunto de validação independente. Para o *Histogram-based Gradient Boosting Machine* (GBM) brasileiro, a divisão final foi uma partição *holdout* 80/20 com 528 670 amostras de treino e 132 168 amostras de teste, com a validação realizada por validação cruzada estratificada, conforme detalhado na Seção 3.6. Para o *fine-tuning* do DomURL-BERT (ajuste de um modelo já treinado em larga escala a uma tarefa específica, com um conjunto de dados menor), foram usadas 200 000 amostras de treino e 50 000 amostras de avaliação extraídas do conjunto BR, sem conjunto de teste separado. Para o CatBoost da cascata, foram usadas 16 000 amostras de treino e 4 000 amostras de teste, derivadas de um subconjunto de 20 000 URLs com características enriquecidas por consultas WHOIS, DNS, redirecionamentos e *Transport Layer Security* (TLS), denominado `dataset_cascata_20k.csv`.

3.3 Extração de características

As características extraídas foram organizadas em três grupos, descritos a seguir.

- **Características baseadas em URL.** Foram derivadas características lexicais e estruturais a partir da URL bruta, entre elas: comprimento total, comprimento do domínio, contagens de caracteres especiais (pontos, hifens, barras, arroba, pontos de interrogação), uso de encurtadores conhecidos, presença de IP no domínio, uso ou não de *Hypertext Transfer Protocol Secure* (HTTPS) e proporção de vogais no domínio. Esses sinais alimentam tanto o GBM avaliado quanto o classificador CatBoost da cascata.

- **Características baseadas em WHOIS, DNS e TLS.** Para um subconjunto de URLs, foram coletadas informações de registro de domínio (idade do domínio em dias, dias até a expiração, *registrar*, código de país, indicador de *privacy*), informações de *Domain Name System* (DNS) (número de servidores *Mail Exchange* e *Name Server*, presença de registro *Sender Policy Framework* – SPF, indicador de

domínio que resolve para IP) e informações de TLS (emissor do certificado, dias de validade, contagem de *Subject Alternative Names*). O total de 24 características (21 numéricas e 3 categóricas: `registrar`, `country_code` e `tls_issuer`) compõe a entrada do CatBoost da cascata.

- **Características baseadas em DOM e na página renderizada.** Foram avaliadas características adicionais extraídas do DOM (existência de formulário de *login*, ação de envio para outro domínio, contagem de *iframes*, consistência entre o texto do *link* e o destino) na variante experimental do DomURL-BERT denominada multimodal. Essa variante combina, na mesma sequência tokenizada, a URL, os campos de WHOIS e marcadores derivados do DOM.

Para as variantes baseadas em *transformer*, a entrada foi tokenizada com o vocabulário do modelo base (MAHDAOUY *et al.*, 2024). Os *tokens* constituíram as pequenas unidades em que o modelo fatiou o texto, do tamanho de pedaços de palavras ou caracteres; o vocabulário foi estendido com *tokens* especiais que delimitam regiões da entrada (por exemplo, `[URL]`, `[WHOIS]`, `[AGE]`, `[REG]`, `[EXPIRE]` na variante com WHOIS+DOM; `[DOMAIN]`, `[PATH]`, `[IP]`, `[IPv6]` na variante adotada). Esses *tokens* permitiram que um mesmo modelo *transformer* processasse simultaneamente o endereço e os sinais auxiliares como uma sequência única, sem necessidade de uma cabeça multimodal explícita. Detalhes adicionais sobre essas representações e sobre as famílias de modelos comparáveis são apresentados no Capítulo 2, Seção 2.3.

3.4 Modelos avaliados

Foram avaliadas múltiplas famílias de modelos. Os critérios de escolha entre eles são apresentados na Seção 3.8; aqui apresenta-se cada um com a sua função no trabalho.

- **GBM brasileiro (avaliado e descartado como modelo embarcado).** Foi treinado um *Histogram-based Gradient Boosting Classifier* sobre o conjunto BR completo de 660 838 URLs, com objetivo inicial de funcionar como modelo único embarcado no cliente. A busca de hiperparâmetros explorou 30 combinações (*learning rate*, profundidade máxima, número de folhas, mínimo de amostras por folha e regularização L2). A configuração vencedora teve `min_samples_leaf` = 30, `max_leaf_nodes` = 255,

`learning_rate` = 0,1 e `l2_regularization` = 0,1. O modelo foi avaliado e descartado como solução final por especificidade insuficiente para o uso embarcado pretendido, conforme detalhado no capítulo de Resultados e Discussão.

- **DomURL-BERT URL + WHOIS (modelo principal adotado).** Foi adotado o *transformer* DomURL-BERT *fine-tuned* sobre o conjunto BR como classificador primário no lado servidor (MAHDAOUY *et al.*, 2024). A variante adotada considera, na entrada, a URL e os campos extraídos por consulta WHOIS. O *fine-tuning* usou três épocas, comprimento máximo de sequência de 192 *tokens*, *batch* efetivo de 64 e taxa de aprendizado de 2×10^{-5}.

- **DomURL-BERT URL + WHOIS + DOM (variante avaliada e não adotada).** Foi avaliada uma segunda variante do DomURL-BERT que estende a entrada com características derivadas do DOM e de outros sinais da página. Apesar de considerar mais informação, esta variante apresentou desempenho inferior na validação empírica, conforme discutido na Seção 3.8 e detalhado no capítulo de Resultados e Discussão.

- **CatBoost da cascata (modelo de refinamento).** Foi treinado o algoritmo de *boosting* categórico CatBoost sobre o subconjunto enriquecido de 20 000 URLs descrito na Seção 3.2, com 1 000 iterações máximas (*early stopping* acionado em 601), profundidade 8 e *learning rate* 0,05 (PROKHORENKOVA *et al.*, 2018). Esse modelo foi integrado à cascata *server-side* e atua exclusivamente sobre amostras em zona ambígua de probabilidade do classificador primário.

- **SecureBERT (linha de comparação).** O *transformer* SecureBERT (AGHAEI *et al.*, 2022), especializado em texto de cibersegurança e baseado em *Robustly Optimized BERT Pretraining Approach* (RoBERTa), foi avaliado em modo URL apenas e em modo multimodal (URL+WHOIS+DOM), como linha de comparação de *backbone*. A presença do SecureBERT permite responder à pergunta "o ganho observado vem do BERT específico de URL ou poderia vir de outro *transformer* de cibersegurança?".

- **DistilBERT para análise de *e-mail*.** Para o módulo de análise de *e-mail* em *webmails*, foi adotado um DistilBERT já *fine-tuned* para detecção de *phishing* em *e-mail* (`<cybersectony/phishing-email-detection-distilbert_v2.4.1>`) (SANH *et al.*, 2019), com tradução prévia do português para o

inglês via MarianMT (`<Helsinki-NLP/opus-mt-tc-big-pt-en>`) e detecção de idioma com `langdetect`. A escolha do DistilBERT como modelo de referência seguiu o critério de maturidade do *checkpoint* público disponível e do desempenho reportado pelos próprios autores no *benchmark* associado. Diferentemente dos modelos de URL, este *checkpoint* não foi treinado nem avaliado quantitativamente pelo autor sobre dados próprios. O módulo de *e-mail* é, portanto, uma integração de modelo pré-treinado de terceiros em caráter de prova de conceito, e o erro introduzido pela tradução do português para o inglês não foi medido (ver Seção 5.7). A integração com a extensão e os *content scripts* para Gmail e Outlook são descritos no Capítulo 4.

3.5 Métricas de avaliação

A avaliação dos modelos combinou métricas de classificação e de eficiência computacional. As definições formais e as equações de acurácia, *precision*, *recall*, F1-*score*, ROC e AUC seguem a apresentação adotada na revisão de literatura, em particular o material de Cechinel e Camargo (2020), e estão descritas no Capítulo 2, Seção 2.2; aqui são apenas listadas as métricas efetivamente reportadas.

As métricas adotadas são:

- **F1-*score*** – métrica principal de classificação, que equilibra *precision* e *recall*.
- **Acurácia** – usada como referência complementar, com a ressalva clássica do paradoxo da acurácia em conjuntos com classes desbalanceadas; o conjunto BR está balanceado, o que limita a relevância dessa ressalva neste trabalho específico.
- ***Precision* e *Recall* (TPR)** – reportadas em separado para tornar visível o compromisso entre falsos positivos e falsos negativos. O *Recall* corresponde à *True Positive Rate* (TPR), métrica-alvo declarada na proposta. O patamar-alvo em torno de 90 % é compatível com o regime reportado na literatura de detecção de *phishing* por URL baseada em *transformers*, em que classificadores como o URLTran operam em torno de 87 % de revocação (MANERIKER *et al.*, 2021; LI *et al.*, 2024).
- **Especificidade e *False Positive Rate* (FPR)** – A Especificidade (*True Negative Rate*) mede a capacidade do modelo de reconhecer corretamente as URLs legítimas.

Como a extensão atua sobre tráfego real predominantemente legítimo, manter especificidade elevada é tão relevante quanto maximizar a revocação, pois falsos positivos frequentes comprometeriam a adesão do usuário. A FPR é calculada como o complemento da especificidade (FPR $= 1 -$ Specificity) e corresponde à meta-alvo declarada na fase de proposta, com FPR $\leq 0{,}5\,\%$, descrita na Seção 1.1. O teto de meio ponto percentual representa um compromisso exequível dentro dos recursos de dados e de treino desta prova de conceito, compatível com a Revocação almejada. Soluções industriais relatam patamares mais agressivos (o URLTran, por exemplo, opera a uma FPR de $0{,}01\,\%$ (MANERIKER *et al.*, 2021)), mas esse ponto de operação é alcançado com volume de dados de escala industrial e ao custo de uma Revocação inferior à buscada neste trabalho. Assim, a meta de $0{,}5\,\%$ reflete o regime de operação exigido em larga escala, em que cada falso positivo recai sobre tráfego legítimo, mas foi fixada como limiar factível para o escopo acadêmico do estudo.

- ***Area Under the Curve* (AUC) – ROC e PR** – reportadas como métricas globais de qualidade do classificador, robustas a variações de limiar.

- ***Matthews Correlation Coefficient* (MCC)** – métrica robusta a desbalanceamento; varia de -1 a $+1$, com 0 equivalente a chute aleatório e $+1$ a acerto perfeito; reportada por consistência com a literatura comparada.

- **Latência fim-a-fim** – reportada nos percentis P50, P95 e P99 (P50 é a mediana, ou seja, metade das requisições fica abaixo desse valor; P95 e P99 indicam que 95 % e 99 % das requisições ficam abaixo do respectivo valor), para inferência local (em *benchmark*) e para o caminho fim-a-fim na API, com entrada e saída em rede. A expressão "tempo real" empregada neste trabalho designa tempo de resposta interativo, no sentido de Nielsen (NIELSEN, 1993): o objetivo de projeto é manter a resposta abaixo do limiar de cerca de 1 s que preserva a sensação de fluxo contínuo de interação, e não atender garantias de sistemas de tempo real estritos.

A escolha por reportar simultaneamente F1-*score*, MCC e FPR/Especificidade segue a recomendação prática de não confiar em uma única métrica para problemas em que erros do tipo I (falsos positivos) e do tipo II (falsos negativos) têm impactos qualitativamente distintos, como em uma extensão de navegador, na qual o falso positivo degrada a experiência cotidiana do usuário, conforme retomado na Seção 3.8.

3.6 Procedimento experimental

- **Ambiente computacional.** Os experimentos foram conduzidos com os recursos disponíveis ao autor, sem auxílio monetário específico para infraestrutura. O *fine-tuning* dos modelos *transformer* (DomURL-BERT em ambas as variantes e SecureBERT) foi realizado em *notebooks* executados no Google Colab com GPU NVIDIA T4 da versão gratuita do serviço. O treinamento do GBM brasileiro e do CatBoost da cascata também foi realizado no Google Colab. Os testes locais e a validação empírica via API foram conduzidos em computador pessoal do autor, com processador AMD Ryzen 5 5600X (seis núcleos), 24 GB de memória *Double Data Rate 4* (DDR4) e GPU AMD Radeon RX 6600. Os serviços do *back-end* foram executados em contêineres Docker, com a inferência em CPU (uma vez que a GPU AMD em uso não é compatível com o caminho padrão CUDA do PyTorch para os modelos BERT adotados). Esta limitação é registrada formalmente na discussão do capítulo de Resultados e Discussão. A inferência em CPU no ambiente local, sem aceleração CUDA, representa um cenário de pior caso em relação a servidores com GPU dedicada; as latências reportadas nesse cenário devem ser lidas como limite superior, de modo que tempos aceitáveis ali reforçam a viabilidade prática, ainda que limitem a generalização direta dos valores para outras infraestruturas.

- **Partição dos dados e validação.** Nenhum modelo empregou conjunto de validação independente; as partições foram estratificadas e os tamanhos exatos de cada divisão estão na Seção 3.2. Para o GBM brasileiro, adotou-se partição *holdout* 80/20 em treino e teste; a seleção de hiperparâmetros usou validação cruzada estratificada em três *folds* aplicada apenas sobre a partição de treino, e a estabilidade da configuração vencedora foi avaliada por validação cruzada estratificada em cinco *folds* sobre o conjunto BR completo, com média e desvio-padrão reportados em F1 e AUC. A parada antecipada interna do algoritmo utilizou 10 % da partição de treino, sem constituir conjunto de validação definido neste trabalho. Para o DomURL-BERT, a divisão foi em treino e avaliação, sem conjunto de teste separado; para o CatBoost da cascata, em treino e teste. Nos modelos com conjunto de teste, este foi mantido isolado até a etapa final de avaliação.

- **Hiperparâmetros e busca.** Os hiperparâmetros vencedores de cada modelo são detalhados nas seções correspondentes do Capítulo 4 e do capítulo de Resultados

e Discussão. O GBM teve busca explícita em 30 combinações; o CatBoost foi treinado com uma única configuração orientada por valores de referência da literatura; o DomURL-BERT e o SecureBERT foram treinados com configuração única de hiperparâmetros, sem busca formal, em função das limitações de tempo de GPU e de ciclos de *fine-tuning* no Colab gratuito.

- **Validação empírica via API.** Em complemento à avaliação em conjunto de teste fixo, foi conduzida uma validação empírica via API com URLs reais. O procedimento consistiu em executar requisições controladas contra a API local (`http://localhost:8000`) e registrar a resposta do modelo, a confiança e a latência por amostra. Foram executadas duas rodadas: uma com 62 URLs (`validacao_modelo.json`), que avaliou o DomURL-BERT URL+WHOIS isoladamente, sem proteção de listas; e outra com 101 URLs (`validacao_cascata.json`), que comparou BERT puro, CatBoost puro e cascata BERT+CatBoost. Esta validação foi conduzida durante o desenvolvimento, com caráter qualitativo, e foi posteriormente complementada pela rodada formal de avaliação em campo sobre lista pré-registrada de 1 946 URLs, reportada no capítulo de Resultados e Discussão (Tabela 10). A escolha da validação via API local permitiu controlar os artefatos de entrada, isolar o comportamento do modelo das latências de rede externa e inspecionar individualmente cada predição; a alternativa seria uma implantação remota, que comprometeria a reprodutibilidade dos tempos de resposta. Esperava-se observar coerência entre as predições da API e as do conjunto de teste, latência inferior ao limiar de interatividade de cerca de 1 s e diferenças qualitativas entre os modos BERT puro, CatBoost puro e cascata.

- **Reprodutibilidade.** A reprodutibilidade foi tratada como requisito do procedimento experimental. A API de análise é distribuída como pacote Docker Compose (`docker compose up -d` no diretório `phishing-api`); a extensão é construída via `npm install` seguido de `npm run build` no diretório `extensao-phishing`; os *notebooks* de treinamento, com *seeds* fixadas onde tecnicamente viável, foram preservados; e os pesos dos modelos foram versionados como artefatos de *release* no repositório Git do projeto. Esses elementos permitem que terceiros reproduzam, com mínima reconfiguração, o ambiente experimental descrito.

- **Implementação da extensão e da API.** A descrição detalhada da extensão de

navegador, da API de análise e do *pipeline* de *e-mail* é apresentada no Capítulo 4.

3.7 Ferramental

As ferramentas computacionais empregadas neste trabalho compreendem as bibliotecas e os ambientes de execução descritos na Seção 3.6. Em complemento a elas, ferramentas de Inteligência Artificial (IA) generativa, baseadas em modelos de linguagem, foram utilizadas como apoio à revisão de escrita e à consistência terminológica do texto da monografia. Essas ferramentas foram configuradas como agentes especializados executados em interface de linha de comando (CLI), cada um com papel definido (redação, revisão e verificação bibliográfica), e foram instruídas por meio de uma memória de projeto contendo regras de estilo, decisões técnicas e glossário de termos. A atuação dessas ferramentas restringiu-se à forma textual: nenhum dado, resultado, métrica ou referência bibliográfica foi gerado por elas. A autoria de todo o conteúdo técnico permanece com o autor, que revisou e validou cada intervenção sugerida.

3.8 Critérios de comparação

A escolha entre as variantes avaliadas seguiu critérios objetivos, declarados explicitamente para tornar a decisão auditável.

Critério 1 – Comportamento em validação empírica quanto a falsos positivos (decisivo para o modelo principal). Entre as variantes URL+WHOIS e URL+WHOIS+DOM do DomURL-BERT, o critério decisivo foi o comportamento observado em validação empírica quanto a falsos positivos sobre URLs legítimas reais. A variante URL+WHOIS foi adotada por apresentar menor incidência de falsos positivos em uso prático, mesmo quando algumas métricas isoladas em conjunto de teste fixo apresentavam vantagem marginal para a variante com DOM. Em uma extensão de navegador, falsos positivos degradam diretamente a experiência cotidiana do usuário e geram fadiga de alerta; este é, portanto, um critério de **projeto**, não puramente um critério de métrica.

Critério 2 – Adequação a uso embarcado no cliente (decisivo para o GBM). O GBM brasileiro foi avaliado como modelo único embarcado no cliente. A combinação de tamanho de *bundle*, latência local e especificidade observada não atingiu o patamar

requerido para esse uso. Em particular, a especificidade observada no conjunto de teste implicou taxa de falsos positivos muito acima da meta de proposta de 0,5 %, o que tornaria a maior parte das navegações cotidianas alvo de alarmes falsos. O GBM permaneceu na narrativa do trabalho como referência comparativa, e não como solução adotada.

Critério 3 – Ganho da cascata em *Recall* a custo aceitável de latência (decisivo para a cascata). A cascata BERT+CatBoost foi adotada quando se observou que ela preservava o *Recall* obtido pelo BERT direto e reduzia o número de falsos positivos em validação empírica, ao custo de aumento moderado de latência fim-a-fim no caminho que passa pelo CatBoost. Esse compromisso foi considerado aceitável para a fração minoritária do tráfego que cai na zona ambígua de probabilidade do BERT.

A consequência dessas três decisões é a configuração de modelos descrita no Capítulo 4 e os resultados reportados no capítulo de Resultados e Discussão.

3.9 Aspectos éticos e conformidade

O presente trabalho envolveu o uso de dados oriundos de navegação na *web* e a construção de uma extensão de navegador voltada à detecção de *phishing*. Por esse motivo, são explicitados a seguir os cuidados adotados em relação à privacidade dos usuários, à proteção de dados e à conformidade com normas institucionais e legais, em especial com a Lei Geral de Proteção de Dados Pessoais (LGPD, Lei n. 13.709/2018).

Do ponto de vista dos dados utilizados, o foco recaiu sobre URLs e sinais derivados do conteúdo das páginas (estrutura de DOM e metadados técnicos), sem coleta de dados sensíveis ou diretamente identificáveis de indivíduos. Foram utilizados, sempre que possível, conjuntos de dados públicos ou previamente anonimizados, em que as URLs já se encontravam rotuladas como legítimas ou *phishing*. A coleta complementar para enriquecimento de WHOIS, DNS e TLS foi restrita a domínios públicos. Não foram armazenados credenciais, conteúdos de formulários, textos inseridos pelo usuário ou qualquer dado pessoal.

A arquitetura final adotada para a extensão é orientada pelo princípio de privacidade por padrão, alinhado à minimização de dados e à transparência das decisões ao usuário. O processamento na maior parte das navegações cotidianas ocorre no próprio navegador, por meio das três camadas locais (*whitelist*, *blacklist* e *cache*), descritas no Capítulo 4. Apenas quando essas camadas não resolvem a decisão localmente, a URL é enviada à API do *back-end* para análise por modelo. Mesmo nesse caminho, a

extensão envia somente a URL e indicadores derivados, sem conteúdo bruto da página, sem credenciais e sem campos de formulário. As permissões da extensão foram limitadas ao mínimo necessário para a função de detecção, em conformidade com o princípio de privilégio mínimo.

A telemetria adotada segue política de *opt-in*: a coleta de eventos destinada à avaliação do protótipo (contagem de alertas, taxa de falsos positivos percebida, medições de latência) só ocorre se explicitamente habilitada por um cabeçalho de identidade na requisição. A telemetria é limitada a dados estatísticos agregados por organização, sem identificadores individuais persistentes nem conteúdos de páginas. Em modo anônimo (sem cabeçalho de identidade), a API processa as requisições sem associá-las a uma conta.

No âmbito institucional, o trabalho enquadra-se como pesquisa de **baixo risco**, uma vez que não envolveu intervenção direta em seres humanos nem coleta de dados pessoais sensíveis. Não foi necessário, portanto, submeter o projeto a comitê de ética nesta etapa. Qualquer ampliação de escopo que venha a incluir testes com usuários reais identificados, ou integração em ambientes de produção com coleta de dados de navegação, deverá ser precedida de análise adicional de conformidade, com eventual submissão a comitê de ética e revisão da política de privacidade da solução.

3.10 Riscos

A execução deste trabalho envolveu riscos de natureza técnica, organizacional e ética. Esta seção descreve os principais riscos identificados, seus impactos potenciais e as estratégias de mitigação adotadas, em alinhamento com o planejamento descrito no Capítulo 4.

Qualidade, disponibilidade e representatividade dos dados. Conjuntos de dados públicos podem apresentar rótulos ruidosos, ausência de informações estruturais ou dominância de uma única fonte. No conjunto BR, a base `kmack_phishing_urls` responde por 99,3 % das URLs de *phishing*. Para mitigar esses problemas, adotou-se a combinação de bases públicas consolidadas com curadorias regionais e nacionais, o versionamento dos arquivos de origem e o balanceamento explícito do conjunto BR durante o treinamento. O conjunto BR foi balanceado com 50 % de URLs de *phishing* e 50 % de URLs legítimas. Esse balanceamento artificial, porém, não reflete a distribuição típica de campo, em que URLs legítimas são majoritárias e a classe *phishing* ocorre como evento raro. Além disso, não foram conduzidos experimentos

sistemáticos com proporções desbalanceadas (por exemplo, 1:10, 1:100 ou configurações mais extremas), nem foram comparadas estratégias de reamostragem ou ponderação de classes. Como atenuantes parciais, adotaram-se métricas robustas a desbalanceamento, como F1-*score* e coeficiente de correlação de Matthews (MCC). A validação da robustez do modelo sob desbalanceamentos controlados e a comparação com técnicas de reamostragem/ponderação permanecem como trabalhos futuros. Os limites da estratégia de curadoria são discutidos no capítulo de Resultados e Discussão.

***Drift* de conceito em *phishing*.** Padrões de ataque, estruturas de URL e estratégias de evasão evoluem ao longo do tempo, o que tende a reduzir a eficácia de modelos estáticos. O risco foi mitigado pelo registro explícito da janela temporal dos dados, pelo versionamento de modelos e por um *pipeline* reprodutível que viabiliza re-treinamentos. Estratégias formais de atualização contínua permanecem como trabalho futuro.

Restrições de plataforma e desempenho. A necessidade de manter latência baixa no cliente impõe limites ao custo computacional. Mudanças no *Manifest Version* 3 (MV3) podem alterar a disponibilidade de APIs ou o comportamento dos *service workers*. A mitigação consistiu em concentrar a inferência por modelo no *back-end*, conforme evidência empírica registrada no capítulo de Resultados e Discussão. Manteve-se no cliente apenas decisão local por listas e *cache*, operação em $O(1)$. Adotou-se também arquitetura modular, que permite substituir componentes sem reescrever toda a extensão.

Limitação de recursos computacionais e financeiros. A condução do trabalho ocorreu sem auxílio monetário específico para infraestrutura. A combinação de Google Colab gratuito com GPU NVIDIA T4 e máquina pessoal Ryzen 5 5600X com GPU AMD Radeon RX 6600 estabeleceu um teto operacional. Esse teto limitou a quantidade de experimentos, o tempo de treinamento, o tamanho dos modelos e os ajustes de hiperparâmetros. Também restringiu as repetições com múltiplas *seeds* e a exploração de arquiteturas mais custosas. Essa limitação é discutida no capítulo de Resultados e Discussão como restrição experimental e no capítulo de Conclusões como limitação reconhecida.

Infraestrutura de desenvolvimento. Incompatibilidades de versões, descontinuidade de bibliotecas e perda de histórico foram mitigadas por controle de versão sistemático, uso de arquivos de configuração de ambiente (Docker Compose, `requirements.txt`, `package.json`) e *scripts* automatizados de preparação e execução dos experimentos.

Conformidade ética e privacidade. O projeto foi conduzido com minimização

de dados e processamento prioritariamente local. Mudanças de escopo, como inclusão de testes com usuários reais em ambientes de produção, exigiriam conformidade adicional com a LGPD e com normas institucionais. A mitigação adotada consistiu em manter a coleta restrita a ambientes de teste, trabalhar apenas com URLs e metadados não identificáveis e documentar a política de privacidade da solução.

Riscos organizacionais. A conclusão do trabalho dependeu da coordenação de múltiplas frentes de trabalho: revisão bibliográfica, engenharia de características, treinamento e avaliação de modelos, implementação da extensão e redação da monografia. Cada frente apresentou duração difícil de prever com precisão. A complexidade subestimada de integrações entre cliente e API e a curva de aprendizado de bibliotecas e APIs do navegador criaram risco de atraso. A indisponibilidade temporária de recursos compartilhados, como GPU em ambiente gratuito e tempo de máquina pessoal, também contribuiu para esse risco. A necessidade de conciliar o cronograma acadêmico com obrigações pessoais e institucionais ampliou ainda mais a incerteza. A mitigação consistiu em priorizar um caminho mínimo viável, com extensão funcional baseada em listas e *cache* e API com modelo principal. Realizaram-se revisões periódicas de progresso, com replanejamento local quando necessário. Tarefas que dependiam de recursos externos ou de aprendizado de novas ferramentas foram antecipadas. Reduziu-se o escopo de experimentos secundários quando a margem de tempo se mostrou insuficiente. A redução foi decisão de mitigação, e não a única alternativa. Os critérios adotados foram a dependência de recursos externos, como a GPU gratuita do Colab, a contribuição marginal de cada experimento para os objetivos específicos e a prioridade do caminho mínimo viável, composto pela extensão funcional e pela API com o modelo principal. Os experimentos adiados foram registrados como trabalhos futuros.

4 DESENVOLVIMENTO DA SOLUÇÃO

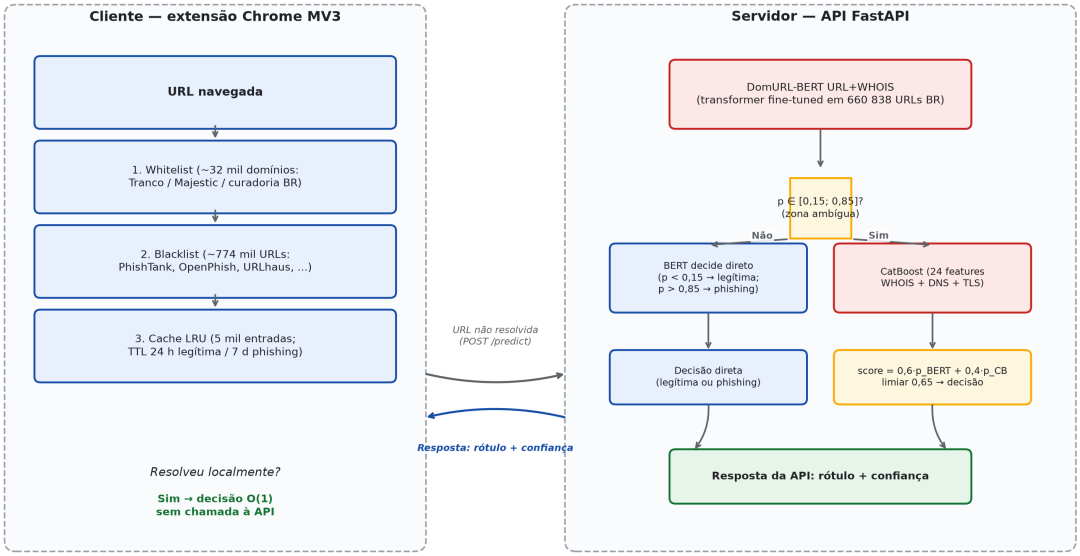
Este capítulo descreve a solução efetivamente entregue neste trabalho, denominada PampAI Security. A apresentação está organizada em doze seções. A Seção 4.1 apresenta a visão geral da arquitetura final. A Seção 4.2 explicita as mudanças em relação à proposta inicial. A Seção 4.3 detalha a extensão de navegador. As Seções 4.4, 4.5 e 4.6 descrevem, respectivamente, a *whitelist*, a *blacklist* e o *cache* local de resultados que compõem as três camadas de decisão local no cliente. A Seção 4.7 descreve a API de análise. A Seção 4.8 descreve o modelo DomURL-BERT adotado como classificador primário. A Seção 4.9 apresenta o fluxo de decisão completo. A Seção 4.10 sintetiza as principais decisões de projeto. A Seção 4.11 descreve a análise de *e-mail* em *webmails* (Gmail e Outlook). A Seção 4.12 descreve a plataforma para uso *enterprise* (*dashboard*, *multi-tenancy*, alertas). Os experimentos comparativos entre modelos avaliados e a discussão dos resultados obtidos com a solução entregue são apresentados no capítulo de Resultados e Discussão.

4.1 Visão geral da arquitetura final

A solução entregue é uma plataforma de detecção de *phishing* composta por uma extensão MV3 para Chromium e uma API em *containers* Docker. A extensão observa URLs visitadas e mensagens abertas em *webmails*; a API hospeda os modelos de aprendizado de máquina e a camada de persistência. A arquitetura segue o modelo cliente-servidor: o cliente decide localmente em três camadas, e o servidor executa a cascata *server-side*.

No cliente, o fluxo segue a ordem: a extensão consulta primeiro a *whitelist*, depois a *blacklist* e, por fim, o *cache* local de resultados. A URL só é enviada à API quando nenhuma das três camadas locais decide a classificação. Na API, o classificador primário DomURL-BERT URL+WHOIS de Mahdaouy *et al.* (2024) decide diretamente quando $p \leq 0{,}15$ ou $p \geq 0{,}85$, ou seja, fora da zona ambígua. A zona ambígua corresponde ao intervalo $0{,}15 < p < 0{,}85$, em que o modelo não tem confiança suficiente sozinho. Nesse caso, a API executa o CatBoost de Prokhorenkova *et al.* (2018) sobre 24 características extraídas de WHOIS, DNS e TLS. A decisão final combina as duas probabilidades pela média ponderada score $= 0{,}6 \times p_{\text{BERT}} + 0{,}4 \times p_{\text{CatBoost}}$ e aplica o limiar $0{,}65$. A Figura 3 sintetiza esse fluxo.

Figura 3 – Arquitetura final do PampAI Security: fluxo de decisão no cliente (*whitelist*, *blacklist* e *cache*, nesta ordem) e escalonamento para a API. No servidor, DomURL-BERT URL+WHOIS decide fora da zona ambígua e CatBoost atua dentro dela, com média ponderada ($0{,}6\,p_{\text{BERT}} + 0{,}4\,p_{\text{CatBoost}}$) e limiar $0{,}65$.



Fonte: Autor (2026).

A decisão local consulta *sets* de *strings* gerados a partir dos arquivos JSON e um objeto indexado por URL no *cache*. Essas estruturas são implementadas internamente como tabelas *hash*. No melhor caso e no caso médio, a consulta custa $O(1)$ (CORMEN *et al.*, 2022): o tempo de busca não depende do número de elementos armazenados. Nesse texto, n denota o número de elementos da estrutura consultada, seja cerca de 32 000 domínios na *whitelist*, 774 000 URLs na *blacklist* ou até 5 000 entradas no *cache*.

O comportamento $O(1)$ no caso médio é uma propriedade padrão das tabelas *hash*, e não uma suposição sobre o tamanho pequeno das listas. Quando o número de elementos inseridos ultrapassa o fator de carga da tabela, o motor JavaScript (V8, no caso do Chromium) realoca a tabela com mais *buckets* e reinsere os elementos (operação conhecida como *rehash*). Dessa forma, o número médio de elementos por *bucket* permanece constante, mesmo que a lista cresça entre atualizações periódicas. Isso difere de estruturas baseadas em árvores balanceadas, cujas buscas têm complexidade $O(\log n)$, e de listas encadeadas, cujas buscas são $O(n)$.

As listas crescem entre versões porque os *scripts* de atualização periódica acrescentam novos domínios e URLs. A cada nova versão da extensão, os *sets* são reconstruídos a partir dos JSONs atualizados, gerando uma nova tabela *hash* com

redimensionamento adequado. Por isso, o *lookup* permanece $O(1)$ no caso médio mesmo quando as listas aumentam. No pior caso teórico, colisões sucessivas de *hash* podem degradar a consulta para $O(n)$. Esse cenário exigiria que um conjunto grande de chaves distintas fosse mapeado para o mesmo *bucket*, o que não ocorre com dados naturais de domínios e URLs, mas pode ser induzido deliberadamente em ataques de *hash flooding*. Navegadores modernos implementam contramedidas contra esse tipo de ataque, e as listas do PampAI Security provêm de fontes públicas curadas, de modo que o pior caso não se materializa na operação rotineira da extensão. A inferência no servidor aplica o modelo treinado a uma URL nova e devolve uma probabilidade. Essa separação evita chamadas à API na maior parte das navegações. O ciclo completo de processamento, com tempos típicos de cada caminho e a fração de tráfego que atinge a cascata, é discutido no capítulo de Resultados e Discussão.

4.2 Mudanças em relação à proposta inicial

A proposta apresentada no TCC I previa uma extensão híbrida com modelo leve embarcado no cliente e *fallback* robusto no servidor, acionado apenas em casos de baixa confiança. Durante a execução do TCC II, a arquitetura foi reorganizada por três motivos principais.

Reversão do modelo leve embarcado. O modelo leve baseado em *Histogram-based Gradient Boosting Machine* (GBM), *fine-tuned* sobre o conjunto BR de 660 838 URLs (Capítulo 3), foi exportado para *Open Neural Network Exchange* (ONNX) e instalado no cliente em uma versão preliminar. A taxa de falsos positivos observada (FPR aproximada de 12 % no conjunto de teste) ficou muito acima da meta de proposta (FPR $\leq$ 0,5 %) e disparou chamadas à API de *fallback* na maioria das navegações. A combinação de tamanho de *bundle* (cerca de 95 MB com o modelo embutido), latência local em CPU e especificidade insuficiente levou à decisão de remover o modelo do cliente. Após a remoção, o *bundle* caiu para aproximadamente 26 MB e o cliente passou a decidir localmente apenas por listas e *cache*.

Promoção do modelo robusto a classificador primário. Como consequência direta da reversão acima, o DomURL-BERT, originalmente previsto como *fallback*, passou a ser o classificador primário no servidor. Para preservar a complementaridade entre métodos prevista na proposta original, foi introduzida uma cascata *server-side* na qual o CatBoost atua exclusivamente sobre amostras na zona ambígua de probabilidade

do BERT. A discussão completa dos modelos avaliados e dos critérios de comparação está na Seção 3.8 do Capítulo 3 e no capítulo de Resultados e Discussão.

Ampliação de escopo. Após a estabilização da arquitetura de URL, o trabalho foi ampliado em duas direções. A primeira foi a inclusão de um *pipeline* de detecção de *phishing* em *e-mail* (Gmail e Outlook), descrito na Seção 4.11. A segunda foi a introdução de uma plataforma para uso *enterprise*, com persistência *multi-tenant*, *dashboard* e alertas, descrita na Seção 4.12. Ambas as ampliações eram consequência natural da arquitetura cliente-servidor adotada e foram conduzidas com os mesmos princípios de privacidade e minimização de dados declarados na Seção 3.9.

4.3 Extensão de navegador

A extensão foi construída com a ferramenta Plasmo, que organiza o ciclo de *build* de extensões MV3 sobre o ecossistema Node.js, e implementada em TypeScript. A interface de *popup* usa React. O empacotamento final reside em `extensao-phishing/` no repositório do projeto e produz um *bundle* de aproximadamente 26 MB.

A adoção do MV3 é um requisito de plataforma: os navegadores baseados em Chromium descontinuaram o *Manifest V2* e passaram a aceitar apenas o novo padrão (PANTELAIOS; KAPRAVELOS, 2024; Google Chrome Developers, 2025). Sobre essa base, as demais escolhas seguiram critérios de engenharia compatíveis com o escopo de prova de conceito. O Plasmo (Plasmo, 2024) automatiza o empacotamento MV3 e o recarregamento durante o desenvolvimento, o que reduz o código repetitivo de configuração do `manifest.json` e do *bundling*. O TypeScript (Microsoft, 2023) adiciona verificação estática de tipos, útil em um cliente que manipula dados de navegação. O React (Meta Open Source, 2022) permite compor a interface do *popup* de forma declarativa. A configuração manual de uma extensão MV3, sem ferramenta dedicada, foi descartada porque não oferece o recarregamento automático durante o desenvolvimento e exige manter à mão o `manifest.json` e o *bundling* que o Plasmo gera de forma assistida. As três ferramentas são gratuitas e de código aberto, sem custo de licenciamento.

A arquitetura interna divide-se em quatro grupos de arquivos:

- ***Service worker.*** O arquivo `background.ts` mantém o estado de configuração e

atende mensagens dos *content scripts* e do *popup*.

- ***Content scripts*.** O `contents/detector.ts` observa a navegação em abas ativas e dispara a verificação por URL; o `contents/webmail-detector.ts` detecta a abertura de mensagens em Gmail e Outlook e dispara a análise de *e-mail* (Seção 4.11).

- **Interface.** O `popup.tsx` apresenta o cartão de resultado da última URL analisada e o cartão de *e-mail* quando aplicável, com indicador de estado da API e configurações.

- **Utilitários.** O diretório `utils/` agrupa `api.ts` (cliente HTTP), `blacklist.ts` (consulta à *blacklist* embutida), `cache.ts` (*cache* LRU), `identity.ts` (identidade gerenciada), `logger.ts` e `clientFeatures.ts` (extração de características leves no cliente).

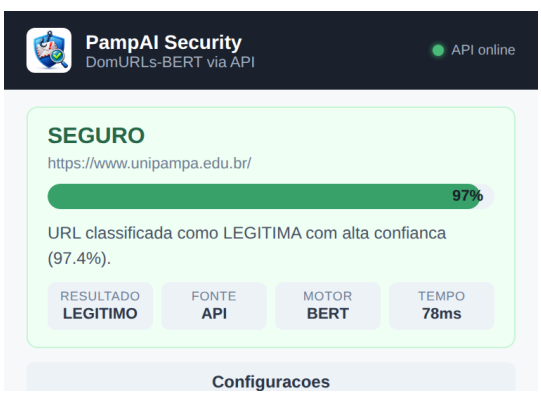
A extensão segue o princípio de *least privilege*: as permissões declaradas em `manifest.json` restringem-se ao mínimo necessário para a função de detecção, sem acesso a histórico de navegação completo, conteúdo de formulários ou credenciais. As duas listas (*whitelist* e *blacklist*) são distribuídas embutidas no *bundle*. Cada lista é convertida de arquivo JSON em *set* de *strings*, implementado como tabela *hash* pelo motor JavaScript. A consulta local custa $O(1)$ no caso médio, independentemente do tamanho atual das listas, e não exige chamada externa.

O *popup* reúne, no cartão de resultado da última URL analisada, o rótulo, a barra de confiança, a fonte da decisão (lista local, *cache* ou API) e o motor acionado no servidor. A Figura 4 mostra os dois desfechos de uma análise de URL. A cor verde indica veredicto legítimo e a vermelha, *phishing*; o cartão de *phishing* também sinaliza quando a cascata BERT+CatBoost participou da decisão.

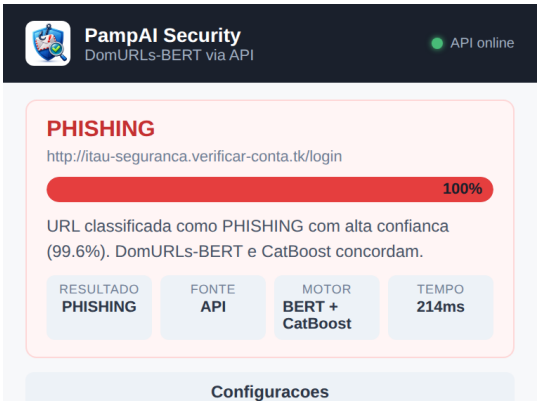
Além do cartão no *popup*, a extensão sinaliza o veredicto por um *badge* sobreposto ao seu ícone na barra do navegador, atualizado por aba. A Figura 5 mostra os três estados: marca verde para legítimo, âmbar com `?` e vermelha com `!` para *phishing*. O estado âmbar corresponde a um veredicto não *phishing* de baixa confiança, definido por probabilidade de classificação inferior a 70 %, e não a um rótulo distinto do classificador. O *badge* oferece sinalização imediata sem exigir a abertura do *popup*.

O *popup* inclui ainda um painel de configurações, mostrado na Figura 6, que permite definir o *endpoint* da API e limpar o *cache* local de resultados.

Figura 4 – *Popup* da extensão após a análise de uma URL: (a) resultado legítimo, com barra de confiança verde, fonte e motor de decisão; (b) resultado de *phishing*, em vermelho, decidido pela cascata BERT+CatBoost.



(a) URL legítima



(b) URL de *phishing*

Fonte: Autor (2026).

Figura 5 – Estados do *badge* no ícone da extensão, da esquerda para a direita: legítimo (verde), suspeito (âmbar, `?`) e *phishing* (vermelho, `!`).

Fonte: Autor (2026).

4.4 Whitelist

A primeira camada local é uma *whitelist* composta por aproximadamente 32 000 domínios consolidados a partir de três fontes: o *ranking* Tranco, o *Majestic Million* e curadorias regionais brasileiras (governo, universidades e empresas dos estados do Paraná, Rio Grande do Sul e Santa Catarina, além de curadoria nacional de portais de saúde e financeiros). A consolidação e a atualização periódica são feitas pelo *script* `scripts/atualizar_listas.py`, que produz o arquivo `whitelist.json` embarcado na extensão.

A presença da *whitelist* é uma decisão arquitetural sustentada por evidência empírica: o classificador primário, sem proteção de listas, classifica como *phishing* domínios brasileiros legítimos de alto perfil em proporção elevada, conforme detalhado no capítulo de Resultados e Discussão. A *whitelist* elimina essas chamadas à API antes que cheguem ao modelo.

Figura 6 – Painel de configurações do *popup*: definição do *endpoint* da API e limpeza do *cache* local.

Fonte: Autor (2026).

4.5 Blacklist

A segunda camada é uma *blacklist* de aproximadamente 774 000 URLs consolidada a partir de cinco fontes públicas: OpenPhish, PhishTank (OpenDNS, 2025), URLhaus, PhishStats e Phishing.Database. O *pipeline* de coleta é implementado em `blacklist/coletar_blacklist.py` e produz o arquivo `blacklist.json` embarcado na extensão. O processo inclui deduplicação exata, normalização do domínio e consolidação dos rótulos.

A atualização exige novo *build* e nova versão da extensão (decorrência da decisão de embarcar as listas no *bundle*), e não há, na entrega atual, um canal de atualização das listas independente desse ciclo de versão. A regeneração periódica das listas é feita pelos *scripts* de coleta (`scripts/atualizar_listas.py` para a *whitelist* e `blacklist/coletar_blacklist.py` para a *blacklist*), mas o resultado só chega ao usuário quando uma nova versão da extensão é publicada. Esse *trade-off* é compensado pela consulta local em *sets* gerados a partir dos arquivos JSON. Como esses *sets* são tabelas *hash*, a latência de consulta é $O(1)$ no caso médio, independentemente do tamanho atual das listas e das atualizações periódicas. A ausência de dependência de serviço externo no caminho rotineiro também reduz a variabilidade de latência. Uma evolução natural da arquitetura seria um canal de entrega de listas pelo servidor, nos moldes de serviços de reputação como o Google Safe Browsing (Google, 2024), que dispensaria novo empacotamento a cada atualização e reduziria a defasagem entre a coleta e a chegada

ao cliente.

4.6 Cache local de resultados

A terceira camada local é um *cache* de resultados implementado sobre `chrome.storage.local`, com política de substituição *Least Recently Used* (LRU) e capacidade de 5 000 entradas. A política LRU descarta primeiro as entradas usadas há mais tempo, o que preserva no *cache* os domínios visitados com maior frequência recente. O *Time To Live* (TTL) é diferenciado por rótulo: 24 horas para domínios classificados como legítimos e 7 dias para domínios classificados como *phishing*.

A escolha por TTL maior para *phishing* reflete uma decisão de projeto orientada pela hipótese de que domínios maliciosos tendem a permanecer maliciosos por janelas mais longas, enquanto a classificação de legítimos pode mudar com mais frequência por revogação de certificado, mudança de propriedade ou comprometimento. O *cache* é o último anel de proteção local antes da chamada à API e atende, em conjunto com a *whitelist* e a *blacklist*, à maior parte das navegações cotidianas.

Quanto ao custo das três camadas locais em conjunto, a extensão embarca no *bundle* duas listas embutidas e um *cache* dinâmico de resultados. A Tabela 1 sintetiza o impacto dessa instrumentação nas dimensões de disco, memória e tempo. Os valores obtidos diretamente dos artefatos do projeto são classificados como medição; os demais, como estimativa.

A memória residente das listas não foi medida diretamente no navegador. A estimativa parte do tamanho do JSON em disco e aplica um fator de 2 a 4 para contemplar a representação UTF-16 das *strings* em JavaScript e o *overhead* da tabela *hash* subjacente ao `Set` (CORMEN *et al.*, 2022). O tempo de consulta local também não foi cronometrado em bancada, mas a operação `Set.has()` custa $O(1)$ no caso médio, mesmo com 774 mil entradas na *blacklist*, e permanece muito abaixo do limiar de 0,1 s de resposta instantânea para o usuário (NIELSEN, 1993). A complexidade não se degrada para $O(\log n)$ ou $O(n)$ simplesmente porque a lista cresceu: a tabela *hash* é redimensionada pelo motor JavaScript para manter o fator de carga constante.

A memória residente, o tempo de inicialização e o tempo médio de consulta local serão medidos em trabalhos futuros. A coleta pode usar o perfil de memória do Chrome e marcações `performance.now()` no *service worker* da extensão.

Tabela 1 – Impacto da instrumentação local da extensão.

Métrica	**Valor**	**Origem**	**Observação**
Tamanho do `whitelist.json`	650 582 bytes (aprox. 635 KB)	Medição no arquivo JSON	32 497 domínios únicos
Tamanho do `blacklist.json`	22 714 155 bytes (aprox. 22 MB)	Medição no arquivo JSON	774 909 entradas únicas
Tamanho total do *bundle*	26 MB (aprox.)	Medição no *build*	Inclui listas, código e recursos
Capacidade do *cache* local	5 000 entradas	Definição de projeto	Política LRU; TTL de 24 h (legítimo) e 7 dias (*phishing*)
Memória residente estimada (*whitelist*)	1,3 a 2,6 MB	Estimativa	Conversão para `Set` com *strings* UTF-16; fator 2 a 4 sobre o JSON em disco
Memória residente estimada (*blacklist*)	44 a 88 MB	Estimativa	Conversão para `Set` com *strings* UTF-16; fator 2 a 4 sobre o JSON em disco; *overhead* de tabela *hash* (CORMEN *et al.*, 2022)
Tempo de inicialização estimado	centenas de ms	Estimativa	*Parse* do JSON de 22 MB e construção dos `Sets`
Tempo médio de consulta local	sub-milissegundo	Estimativa	Operação `Set.has()` em $O(1)$ no caso médio, independentemente do tamanho da lista (CORMEN *et al.*, 2022)

Fonte: Autor (2026).

4.7 API de análise

A API de análise foi implementada em Python sobre o *framework* FastAPI e empacotada em contêineres Docker. A persistência é feita em PostgreSQL 16 com SQLAlchemy 2.0 em modo assíncrono e *driver* `asyncpg`. As migrações de esquema estão em `phishing-api/migrations/` (`001_init.sql`, `002_user_email.sql`).

Os principais *endpoints* expostos são:

- `POST /predict` e `POST /predict-batch`: análise de URL única ou em lote, que retorna rótulo, probabilidade, caminho da cascata acionado e latência.
- `POST /analyze-email`: análise de *e-mail* (Seção 4.11).

- `POST /events`: registro *fire-and-forget* de evento de análise para fins de telemetria *opt-in*.
- `POST /admin/orgs`: criação de organização, que retorna a chave de API gerada (Seção 4.12).
- `GET /reports/{org_id}/summary`: sumário estatístico por organização.
- `GET /dashboard/{org_id}/events` e `GET /dashboard/{org_id}/timeline`: dados consumidos pelo *dashboard* (Seção 4.12).
- `GET`, `POST`, `PUT` e `DELETE` em `/alerts/{org_id}/configs`: gestão dos canais de alerta por *webhook* e *Simple Mail Transfer Protocol* (SMTP).

A autenticação é por chave de API, enviada no cabeçalho HTTP `X-API-Key` e armazenada como *hash* SHA-256. A API foi concebida como *multi-tenant* desde o início: cada análise é persistida com referência a uma organização (`org_id`), e o esquema relacional compreende as tabelas `organizations`, `phishing_events` e `alert_configs`. A atribuição por usuário usa a coluna `user_email` da tabela `phishing_events`, alimentada pelo cabeçalho `X-User-Email`. A operação em modo anônimo (sem cabeçalho de identidade) é preservada e não cria registros associados a uma conta.

A criação de organizações é feita pelo *endpoint* `POST /admin/orgs`, que gera e retorna a chave de API. Esse *endpoint* não exige autenticação por destinar-se ao provisionamento inicial, o que constitui uma fragilidade reconhecida: em um sistema cujo tema é segurança, criar organização e emitir chave de API sem autenticação é um ponto de abuso. Na entrega atual, a mitigação é a restrição por rede; uma versão de produção exigiria um *token* de *bootstrap* ou lista de endereços autorizados, registrado como trabalho futuro (Capítulo 6).

O ambiente de execução é distribuído como pacote Docker Compose: `docker compose up -d` no diretório `phishing-api/` sobe a API, o banco PostgreSQL e os volumes necessários, conforme descrito na Seção 3.6.

A pilha do servidor foi escolhida pela proximidade com o ecossistema de aprendizado de máquina. O FastAPI (FastAPI, 2024) é um *framework web* assíncrono em Python, a mesma linguagem em que os modelos são treinados e servidos, o que mantém treinamento e inferência sob um único ecossistema e ainda gera documentação OpenAPI

de forma automática. O PostgreSQL (PostgreSQL Global Development Group, 2023) atende à integridade relacional exigida pelo isolamento *multi-tenant* por `org_id`, e o SQLAlchemy (SQLAlchemy, 2023) opera em modo assíncrono para não bloquear o laço de eventos durante o acesso ao banco. Outras opções foram consideradas e preteridas por aderência ao escopo. O Flask exigiria camadas adicionais para o suporte assíncrono que o FastAPI já oferece de forma nativa, e o Django traria um mapeamento objeto-relacional e um acoplamento maiores do que uma API de inferência enxuta demanda. O SQLite, por seu modelo de arquivo único, não oferece o controle de concorrência de escrita nem o isolamento entre organizações exigidos pelo uso *multi-tenant*. O Docker Compose padroniza o ambiente e sustenta a reprodutibilidade descrita na metodologia. Todos esses componentes são de código aberto e sem custo de licenciamento, coerentes com o escopo de prova de conceito.

4.8 Modelo DomURL-BERT

O classificador primário da API é o DomURL-BERT (MAHDAOUY *et al.*, 2024), um *transformer* baseado na arquitetura BERT-base com aproximadamente 110 milhões de parâmetros, pré-treinado por MAHDAOUY *et al.* em tarefas de classificação de domínios e URLs, incluindo geração algorítmica de domínios (DGA), *DNS tunneling* e *phishing*. O ponto de partida dos pesos é o *checkpoint* público `amahdaouy/DomURLs_BERT`, cujo vocabulário de cerca de 32 000 *tokens* é especializado em URLs. Sobre esse *backbone* foi aplicado *fine-tuning* supervisionado no conjunto BR descrito na Seção 3.2. Entre as três variantes de *fine-tuning* avaliadas no Capítulo 3 (URL apenas; URL + WHOIS; URL + WHOIS + características de DOM e da página), foi adotada a variante URL + WHOIS como modelo principal. A justificativa da escolha está nos critérios da Seção 3.8 e a discussão dos números comparativos está no capítulo de Resultados e Discussão.

O *fine-tuning* foi conduzido em *notebook* executado no Google Colab com *Graphics Processing Unit* (GPU) NVIDIA T4 da versão gratuita do serviço. A Tabela 2 sintetiza a configuração de treinamento. O otimizador é o AdamW, com taxa de aprendizado de 2×10^{-5}, *warmup ratio* de 6 %, *weight decay* de 0,01 e função de perda *cross-entropy*. O *batch* por dispositivo é 32, com acumulação de gradiente em duas etapas, resultando em *batch* efetivo de 64. O modelo é avaliado ao final de cada época e o mecanismo `load_best_model_at_end` seleciona o *checkpoint* de maior F1 no conjunto de avaliação, funcionando como critério de parada e seleção do melhor modelo.

Tabela 2 – Hiperparâmetros do *fine-tuning* do DomURL-BERT URL+WHOIS.

Hiperparâmetro	**Valor**
Backbone	`amahdaouy/DomURLs_BERT`
Variante adotada	URL + WHOIS
Número de épocas	3
Comprimento máximo de sequência	192 *tokens*
Batch por dispositivo	32
Acumulação de gradiente	2
Batch efetivo	64
Taxa de aprendizado	2×10^{-5}
Warmup ratio	6 %
Weight decay	0,01
Otimizador	AdamW
Função de perda	*Cross-entropy*
Métrica de seleção do melhor modelo	F1-score
Seed	42

Fonte: Autor (2026).

O número de épocas foi definido empiricamente, considerando as limitações práticas do ambiente de execução: a versão gratuita do Google Colab impõe tetos de tempo de execução e de memória na GPU NVIDIA T4, o que restringe a duração admissível do treinamento. Adicionalmente, o mecanismo `load_best_model_at_end` preserva o *checkpoint* de maior F1, garantindo que o melhor modelo seja mantido independentemente da época em que ocorre.

A divisão dos dados foi realizada com `train_test_split` em proporção 80/20, estratificada por rótulo e com *seed* 42, resultando em 200 000 amostras de treino e 50 000 amostras de avaliação. Não houve conjunto de teste separado do treinamento; as métricas reportadas no capítulo de Resultados e Discussão referem-se ao conjunto de avaliação fixo do próprio *fine-tuning*. Os pesos *fine-tuned* são versionados como artefato de *release* no repositório Git do projeto.

O *tokenizer* foi estendido com *special tokens* `[URL]`, `[WHOIS]`, `[EXTRA]`, `[AGE]`, `[REG]` e `[EXPIRE]`, que delimitam regiões da entrada e permitem que um único *transformer* processe simultaneamente o endereço e os campos de WHOIS como uma sequência única. O vocabulário foi expandido de 32 000 para 32 006 *tokens* e os *embeddings* da rede foram redimensionados com `resize_token_embeddings` para acomodar os novos *tokens*.

4.9 Fluxo de decisão da solução

O fluxo completo de decisão executa, em sequência, cinco passos. A Figura 7 representa o fluxo.

1. ***Whitelist* local.** O domínio é consultado na *whitelist* embutida (Seção 4.4). Em caso de acerto, a URL é classificada como legítima e o fluxo termina sem chamada à API.

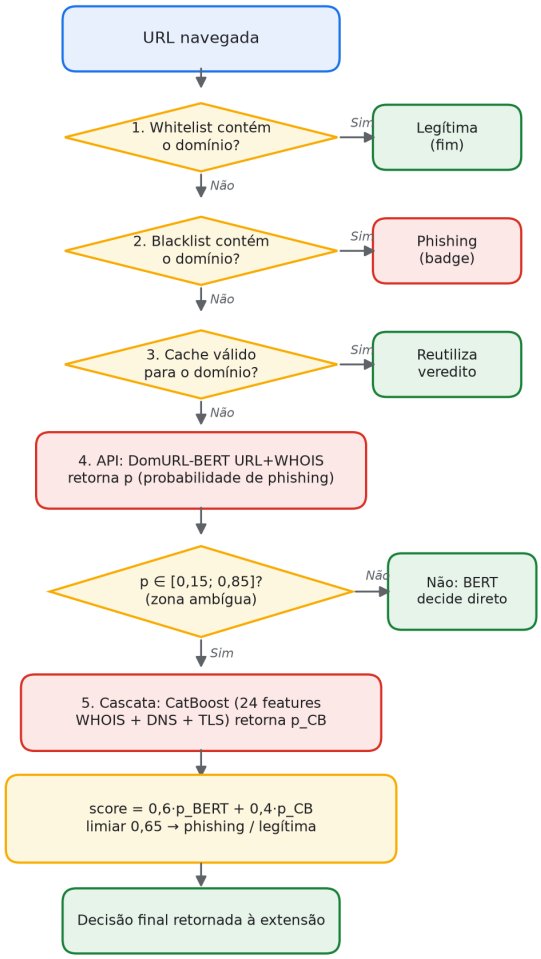
2. ***Blacklist* local.** Se a *whitelist* não responder, o domínio é consultado na *blacklist* (Seção 4.5). Em caso de acerto, a URL é classificada como *phishing*, o ícone da extensão recebe um *badge* vermelho e o *popup* exibe o cartão de resultado em vermelho (Seção 4.3).

3. ***Cache* local.** Se nenhuma das listas responder, a URL é consultada no *cache* de resultados (Seção 4.6). Em caso de acerto e dentro do TTL, o veredicto em *cache* é reutilizado.

4. **API: classificador primário.** Em caso de *cache miss*, isto é, quando a URL não está no *cache* local, a extensão envia a URL e características leves do cliente à API. O DomURL-BERT URL+WHOIS retorna a probabilidade p de *phishing*. Se $p \geq 0{,}85$ ou $p \leq 0{,}15$, o BERT decide diretamente.

5. **API: cascata.** Se p cair na zona ambígua $0{,}15 < p < 0{,}85$, a API enriquece a amostra com consultas WHOIS, DNS, redirecionamentos e TLS, executa o CatBoost da cascata (24 características) e compõe a decisão final pela média ponderada score $= 0{,}6 \times p_{\text{BERT}} + 0{,}4 \times p_{\text{CatBoost}}$, com limiar de classificação 0,65.

Figura 7 – Fluxo de decisão da solução: três camadas locais no cliente seguidas, em caso de *cache miss*, pela cascata BERT+CatBoost na API.



Fonte: Autor (2026).

Em caso de indisponibilidade da API, a extensão adota política de *fail-open*: a navegação não é bloqueada e o *badge* de *phishing* não é exibido. A política oposta seria *fail-closed*, que bloquearia toda navegação durante a indisponibilidade do *back-end*. Esta escolha prioriza a redução de falso positivo perceptível ao usuário sobre a redução de falso negativo silencioso e preserva a confiabilidade percebida da extensão durante janelas de indisponibilidade do *back-end*. Durante essas janelas, a proteção depende exclusivamente das três camadas locais. A *blacklist* embarcada continua a bloquear o *phishing* já conhecido, pois a consulta usa um *set* gerado a partir do arquivo JSON, com complexidade $O(1)$ no caso médio, mesmo quando as listas crescem. Essa proteção persiste mesmo com a API indisponível. O *fail-open* expõe, portanto, apenas a cauda de URLs desconhecidas, e não as ameaças já catalogadas. O PampAI Security opera como complemento, e não

como filtro único do navegador, o que mantém o *fail-open* coerente com a coexistência de outras proteções ativas, como o Google Safe Browsing (Google, 2024). Uma política *fail-closed* com *bypass* explícito permanece viável como configuração *enterprise* distribuída por `chrome.storage.managed` (Seção 4.12), para organizações que prefiram bloquear a navegação durante a indisponibilidade do *back-end*.

4.10 Decisões de projeto

Esta seção sintetiza as decisões de projeto que conformaram a solução entregue. As decisões aparecem dispersas no texto deste capítulo e são reunidas aqui por conveniência de leitura.

- **Sem aprendizado de máquina embarcado no cliente.** A inferência por modelo ocorre exclusivamente no servidor; o cliente decide localmente apenas por listas e *cache*. Esta decisão é consequência empírica do desempenho insuficiente do modelo leve avaliado, conforme detalhado no capítulo de Resultados e Discussão.

- **Cascata *server-side* em vez de divisão cliente-servidor.** A complementaridade entre métodos prevista na proposta foi preservada como cascata *server-side* BERT+CatBoost (Seção 4.9).

- **Listas embutidas na extensão.** *Whitelist* e *blacklist* são distribuídas no *bundle*, com atualização via *rebuild* e nova versão. A consulta local usa *sets* gerados a partir dos arquivos JSON e opera em $O(1)$ no caso médio mesmo após o crescimento das listas, o que compensa o custo de atualização menos frequente.

- ***Fail-open* offline.** Durante indisponibilidade da API, a extensão não bloqueia a navegação. Prefere falso negativo silencioso a falso positivo perceptível ao usuário.

- **TTL diferenciado no *cache*.** 24 horas para legítimos, 7 dias para *phishing*, o que reflete a hipótese de que domínios maliciosos tendem a permanecer maliciosos por janelas mais longas.

- **Identidade gerenciada e telemetria *opt-in*.** A identidade do usuário é configurável via `chrome.storage.managed` (configuração distribuída por uma organização) com *fallback* para `chrome.storage.sync` (configuração

definida pelo próprio usuário); telemetria de evento é *fire-and-forget* e só ocorre quando o cabeçalho de identidade está presente.

- ***Multi-tenancy* desde o início.** O esquema relacional inclui as tabelas `organizations`, `phishing_events` e `alert_configs`, com autenticação por chave de API por organização.

- **Inclusão de *e-mail* no escopo do TCC II.** O *pipeline* Gmail/Outlook foi incorporado após validação positiva da arquitetura de URL (Seção 4.11).

- **Direção de produto *enterprise*.** O *dashboard*, a *multi-tenancy* e os alertas posicionam a solução como produto comercializável para organizações (Seção 4.12).

4.11 Análise de *e-mail* (Gmail e Outlook)

A análise de *e-mail* foi incorporada ao escopo do TCC II após a estabilização do *pipeline* de URL. O módulo cobre dois *webmails*: Gmail (`mail.google.com`) e Outlook em suas variantes públicas (`outlook.live.com`, `outlook.office365.com`, `outlook.office.com`).

Coleta no cliente. O *content script* `webmail-detector.ts` detecta a abertura de uma mensagem na aba ativa e envia ao *back-end* o assunto, o remetente, o corpo em texto e a lista de URLs presentes no corpo. Não são enviados anexos, identidades adicionais ou outros conteúdos do *webmail*.

Inferência no servidor. O módulo de *e-mail* constitui a prova de conceito anunciada no resumo para os *webmails*. Como o classificador opera apenas em inglês, a etapa inicial de detecção de idioma existe para viabilizar a tradução automática das mensagens em português antes da classificação. O *endpoint* `POST /analyze-email` aplica o seguinte *pipeline*:

1. **Detecção de idioma** do corpo com `langdetect`.

2. **Tradução do português para o inglês** via MarianMT (`Helsinki-NLP/opus-mt-tc-big-pt-en`) quando o idioma detectado é português.

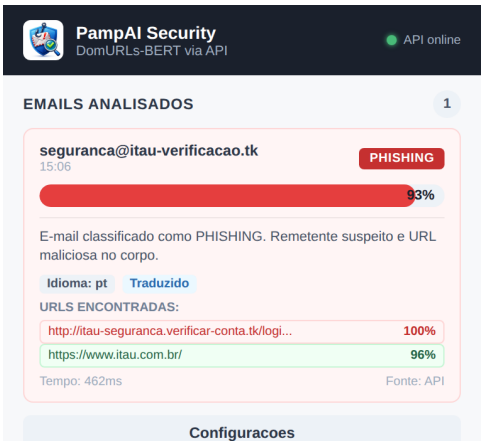
3. **Classificação do corpo** com DistilBERT *fine-tuned* para detecção de *phishing* em *e-mail* (`<cybersectony/phishing-email-detection-distilbert_v2.4.1>`) (SANH *et al.*, 2019).

4. **Análise das URLs do corpo** pelo classificador primário DomURL-BERT URL+WHOIS, sem cascata (Seção 4.8).

5. **Combinação ponderada**: o *score* combina a probabilidade do texto ($p_{\text{e-mail}}$) com um sinal das URLs, definido como a razão de URLs maliciosas multiplicada pela confiança média delas. Quando todas as URLs do corpo corroboram com confiança média de pelo menos 0,85, aplica-se um piso de 0,85; quando a maioria das URLs é maliciosa e o texto é suspeito, combina-se score $= 0{,}6 \times p_{\text{e-mail}} + 0{,}4 \times \text{sinal}_{\text{URL}}$; na ausência de corroboração, o texto isolado é limitado a 0,60.

A decisão final é categorizada em três rótulos: `PHISHING` se o *score* é maior que 0,7; `SUSPICIOUS` se está entre 0,4 e 0,7; e `LEGITIMO` se é menor que 0,4. A interface da extensão exibe um cartão dedicado no *popup* com o rótulo, a lista de URLs do corpo, o idioma detectado e a indicação de tradução. Esse cartão permite ao usuário inspecionar a base da decisão. A Figura 8 mostra esse cartão para uma mensagem classificada como *phishing*.

Figura 8 – Cartão de análise de *e-mail* no *popup*: rótulo `PHISHING`, idioma detectado, indicação de tradução e as URLs do corpo avaliadas individualmente (maliciosa em vermelho, legítima em verde).



Fonte: Autor (2026).

A escolha do DistilBERT como modelo de *e-mail* seguiu dois critérios: a maturidade do *checkpoint* público disponível e o desempenho reportado pelos próprios autores no *benchmark* associado, descrito no Capítulo 3. A separação por tradução do

português para o inglês é uma adaptação prática à ausência de *checkpoints* comparáveis treinados em português brasileiro.

Cabe registrar com clareza o estatuto deste módulo. Ao contrário do *pipeline* de URL, treinado e avaliado neste trabalho, o módulo de *e-mail* reúsa um modelo pré-treinado de terceiros, sem *fine-tuning* próprio e sem avaliação quantitativa sobre um conjunto de *e-mails* reais em português. Além disso, o erro introduzido pela etapa de tradução automática do português para o inglês não foi medido. Por essas razões, o módulo de *e-mail* é apresentado como prova de conceito de integração, e não como componente avaliado com o mesmo rigor experimental do *pipeline* de URL. A avaliação quantitativa própria e o treinamento de um classificador de *e-mail* em português brasileiro ficam registrados como trabalho futuro (Capítulo 6).

4.12 Plataforma para uso *enterprise*

A solução foi concebida com perspectiva de uso por organizações, mas permanece um protótipo acadêmico e prova de conceito. O que foi demonstrado até aqui é a viabilidade técnica da arquitetura e das funcionalidades implementadas, não a viabilidade comercial de um produto. A comprovação desta última exigiria análises operacionais, financeiras, de segurança, governança, conformidade, acordos de nível de serviço (SLAs) e outros indicadores corporativos que fogem ao escopo acadêmico deste trabalho. Esta seção descreve, portanto, os componentes que materializam essa direção técnica, sem inflar o escopo da pesquisa.

Dashboard web. A API serve uma *Single Page Application* (SPA) em `phishing-api/dashboard/index.html`, acessível em `/dashboard-ui/`. A SPA é construída com HTML e folhas de estilo próprias, com a biblioteca de gráficos Chart.js (Chart.js Contributors, 2023) carregada por *Content Delivery Network* (CDN). O *login* é feito por chave de API (Figura 9); a tela principal apresenta cartões com métricas agregadas, série temporal de eventos e tabela paginada com filtros por usuário, rótulo e janela temporal (Figura 10).

A pilha do *dashboard* foi mantida deliberadamente simples, sem etapa de *build*. A escolha prioriza a entrega rápida e dispensa infraestrutura de compilação e publicação, em linha com o escopo de prova de conceito e com a ausência de orçamento dedicado de infraestrutura, assumida na metodologia. Para o *dashboard*, *frameworks* de SPA com etapa de compilação, como React ou Vue, foram descartados por contrariarem esse

objetivo de *zero-build* e por adicionarem complexidade desproporcional a um painel de escopo restrito. O uso do React no *popup* da extensão (Seção 4.3) não contradiz essa escolha: na extensão, o React já se integra ao ciclo de *build* do Plasmo, sem custo marginal de compilação. O *dashboard*, por sua vez, não possui etapa de *build* que justifique adicionar tal dependência. Uma eventual evolução técnica para ambiente de produção substituiria o carregamento por CDN por ativos versionados e servidos localmente, por questões de disponibilidade e privacidade, mas essa melhoria não configura, por si só, uma proposta comercial completa.

Figura 9 – Tela de *login* do *dashboard*, com autenticação por identificador da organização e chave de API.

Fonte: Autor (2026).

A Figura 9 mostra a tela de acesso ao painel administrativo. O operador informa o identificador da organização (`org_id`) e a chave de API correspondente; ao validar as credenciais, a aplicação carrega as métricas e eventos vinculados exclusivamente àquela organização. Essa autenticação simples é suficiente para o escopo de prova de conceito, embora uma implantação corporativa exigiria mecanismos adicionais de identidade federada, controle de sessão e auditoria. Os valores exibidos na interface pertencem a um ambiente de demonstração local e não a dados reais de produção.

A Figura 10 apresenta a tela principal do *dashboard* após a autenticação. Na parte superior, cartões resumem o total de eventos processados, a quantidade de eventos classificados como *phishing* e o número de eventos legítimos. Abaixo, um gráfico

de série temporal permite acompanhar a evolução dos eventos ao longo dos dias, facilitando a identificação de picos de atividade. A tabela paginada lista os eventos individuais e oferece filtros por veredicto, tipo e usuário, possibilitando ao operador localizar rapidamente incidentes relevantes para a organização. Todos os dados exibidos correspondem a uma organização fictícia inserida no ambiente de demonstração, servindo apenas para ilustrar a disposição e o comportamento da interface.

A tabela de eventos pode ser filtrada por veredicto, tipo e usuário. A Figura 11 mostra a listagem com o filtro de *phishing* aplicado.

***Multi-tenancy*.** A persistência segue o esquema relacional descrito na Seção 4.7: cada análise é gravada em `phishing_events` com referência à organização (`org_id`) e, opcionalmente, ao usuário (`user_email`). A separação por `org_id` permite que múltiplas organizações compartilhem a mesma instância da API sem cruzamento de dados.

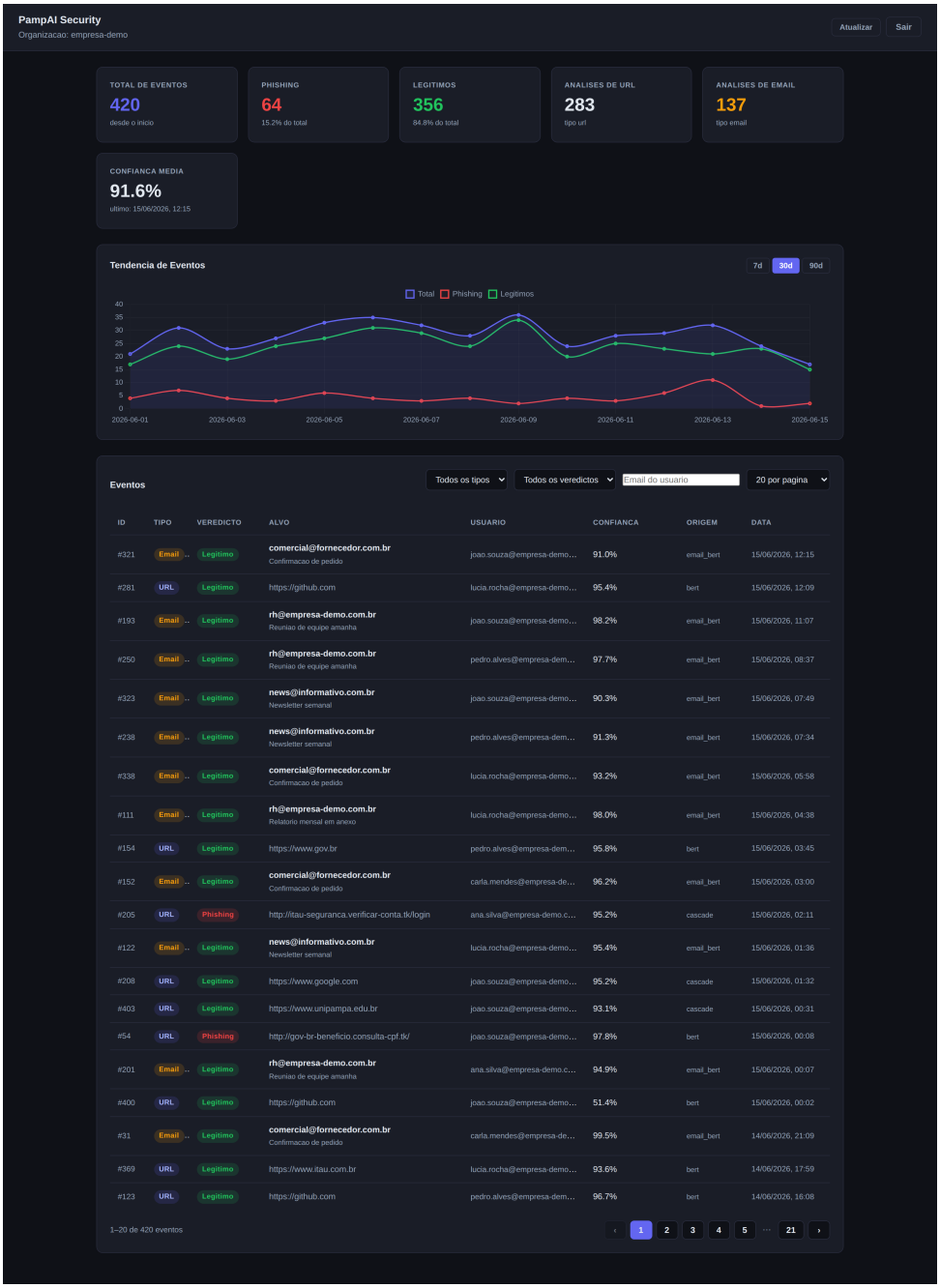
Alertas. O módulo `alerts.py` implementa dois canais: *webhook* HTTP (*timeout* de 5 segundos, modo *fire-and-forget*) e *e-mail* via SMTP. A configuração é feita pelos *endpoints* de *Create*, *Read*, *Update* e *Delete* (CRUD) em `/alerts/{org_id}/configs`.

***Deploy enterprise*.** A extensão consulta `chrome.storage.managed` para receber configuração distribuída via Política de Grupo (*Group Policy Object* – GPO) no Windows ou *managed preferences* no macOS. Esse caminho permite a uma organização provisionar a chave de API, o *endpoint* da API e a identidade do usuário sem intervenção manual em cada estação.

***Onboarding* de organizações.** O cadastro de uma organização é uma operação administrativa em dois passos. Primeiro, o administrador cria a organização pelo *endpoint* `POST /admin/orgs`, que registra o `org_id` e devolve a chave de API gerada. Em seguida, a chave, o *endpoint* da API e a identidade do usuário são distribuídos às estações pela configuração gerenciada (`chrome.storage.managed`), de modo que cada extensão se autoconfigura sem ação do usuário final. Não há, na entrega atual, uma tela de autocadastro: o *onboarding* é conduzido por quem administra a infraestrutura, opção condizente com o estágio de protótipo. Um fluxo de autocadastro com registro, cobrança e emissão autônoma de chaves fica registrado como trabalho futuro (Capítulo 6).

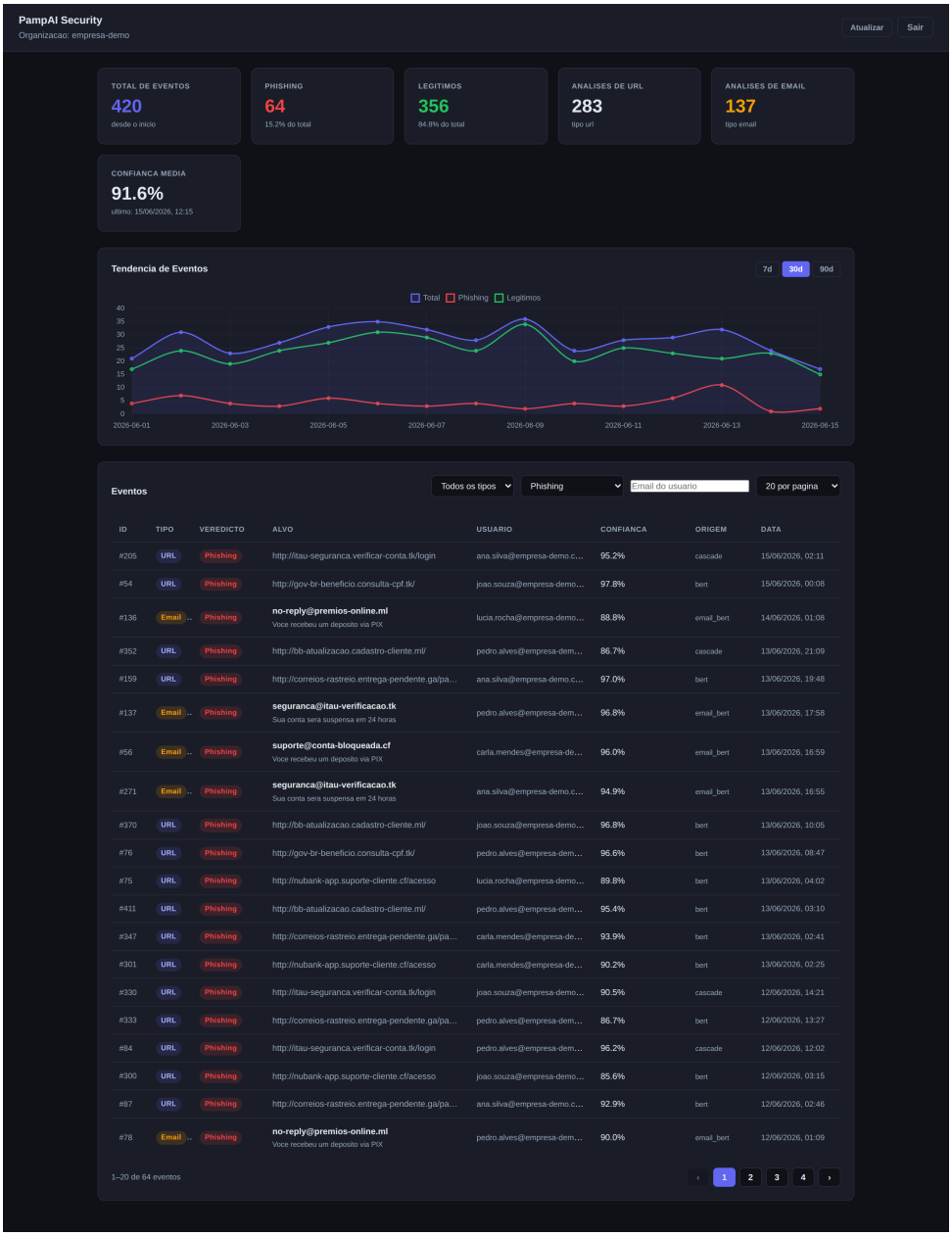
A discussão sobre o custo de manutenção e o valor agregado dessa direção de produto, bem como o posicionamento em relação a protótipos puramente acadêmicos da literatura, é apresentada no capítulo de Resultados e Discussão.

Figura 10 – Visão geral do *dashboard*: cartões de métricas agregadas, série temporal de eventos (total, *phishing* e legítimos) e tabela paginada de eventos com filtros.



Fonte: Autor (2026).

Figura 11 – Tabela de eventos do *dashboard* com o filtro por veredicto *phishing* aplicado.



Fonte: Autor (2026).

5 RESULTADOS E DISCUSSÃO

Este capítulo apresenta os resultados experimentais obtidos durante a execução do trabalho e discute o que esses números indicam, com atenção aos compromissos (*trade-offs*) entre métricas e às limitações de cada resultado. O capítulo está organizado em dez seções. A Seção 5.1 retoma os experimentos com modelos locais conduzidos ao longo do projeto, do experimento histórico do TCC I sobre o conjunto da *University of California, Irvine* (UCI) ao *Histogram-based Gradient Boosting Machine* (GBM) treinado sobre o conjunto brasileiro consolidado neste TCC II. O experimento do TCC I é retomado neste capítulo como *baseline* histórico, de modo a evidenciar a evolução entre as duas fases do trabalho e a maior dificuldade imposta pelo conjunto brasileiro. A Seção 5.2 narra a mudança de arquitetura motivada pela evidência empírica acumulada nesses experimentos. As Seções 5.3 e 5.4 apresentam, respectivamente, o DomURL-BERT URL+WHOIS adotado como modelo principal e o DomURL-BERT multimodal avaliado e não adotado. A Seção 5.5 reúne a comparação direta entre os modelos avaliados, inclui os resultados da validação empírica via API e justifica a decisão de projeto. A Seção 5.6 apresenta a explicabilidade por decisão dos três modelos da solução, via SHAP e mapas de atenção. A Seção 5.7 reconhece as limitações experimentais, entre elas a moldura de recursos computacionais e financeiros sob a qual o trabalho foi conduzido. A Seção 5.8 discute o significado dos resultados em conjunto, reposiciona a meta de Taxa de Falsos Positivos (*False Positive Rate* – FPR) declarada na proposta e posiciona a contribuição. A Seção 5.9 posiciona a solução frente ao estado da prática e a compara com serviços de detecção de *phishing* consolidados no mercado. A Seção 5.10 fecha com trabalhos futuros derivados diretamente dos resultados.

5.1 Experimentos com modelos locais

A construção da solução iniciou-se na fase do TCC I com a investigação de modelos leves capazes de operar embarcados na extensão, ou seja, executados localmente dentro do próprio navegador, sem chamada a um servidor remoto. Nessa primeira rodada, o classificador adotado para os experimentos preliminares foi o *Random Forest* (RF) treinado sobre o *Phishing Websites Data Set* mantido no UCI *Machine Learning Repository* (DUA; GRAFF, 2017), com 11 055 amostras e 30 *features* extraídas de URL e da página. As métricas obtidas naquela rodada, F1 próximo a 0,976 e Área sob a

Curva ROC (*Area Under the Receiver Operating Characteristic Curve* – AUC-ROC) próxima a 0,996, indicaram que classificadores baseados em conjuntos de árvores podiam obter desempenho elevado em conjuntos consolidados internacionais. Esse experimento, contudo, era insuficiente para o objetivo deste trabalho: a amostra era pequena (11 055 registros), o conjunto era de origem internacional, sem foco em URLs brasileiras, e tratava-se de um conjunto consolidado antigo. Por isso, ele foi tratado neste TCC II como ponto de partida histórico, não como evidência de adequação ao escopo final.

Para superar essas limitações, o trabalho passou, na fase do TCC II, a usar um conjunto consolidado a partir de fontes brasileiras e internacionais (descrito em detalhe na Metodologia, Seção 3.2), com recorte nacional e total de 660 838 URLs, cerca de 60 vezes maior que o conjunto UCI, balanceadas em 50 % *phishing* e 50 % legítimas. Sobre esse conjunto foi treinado o GBM final, na implementação `HistGradientBoostingClassifier` da biblioteca *scikit-learn* (PEDREGOSA *et al.*, 2011), com a busca de hiperparâmetros descrita em Metodologia, que produziu a configuração vencedora indicada na Tabela 3. O modelo foi avaliado em uma partição estratificada do tipo *holdout* 80/20, com 528 670 amostras de treino e 132 168 amostras de teste, sem conjunto de validação independente; o resultado final sobre o conjunto de teste está na mesma tabela. A seleção de hiperparâmetros empregou validação cruzada estratificada em três *folds* aplicada apenas sobre a partição de treino, e a estabilidade da configuração vencedora foi verificada por validação cruzada estratificada em cinco *folds* sobre o conjunto BR completo, com os resultados apresentados adiante nesta seção.

Tabela 3 – Métricas e configuração do GBM treinado sobre o conjunto brasileiro de 660 838 URLs.

Item	**Valor**
Modelo	`HistGradientBoostingClassifier`
Dataset	Conjunto BR (660 838 URLs, 50/50)
Treino / Teste	528 670 / 132 168
Acurácia	0,8857
Precisão	0,8798
Revocação (Recall)	0,8935
F1	0,8866
AUC-ROC	0,9544
AUC *Precision–Recall* (PR-AUC)	0,9546
Coeficiente de Correlação de Matthews (MCC)	0,7715
Log Loss	0,2712
Especificidade (FPR $= 1-$Esp.)	0,8779 (FPR $\approx 12{,}21\,\%$)
Latência P50 / P95 / P99	5,69 / 6,80 / 10,10 ms
Tempo de treino	88,51 s
Hiperparâmetros vencedores	`lr`=0,1; `max_leaf_nodes`=255; `min_samples_leaf`=30; `l2`=0,1

Fonte: Autor (2026).

A validação cruzada estratificada em cinco *folds* produziu F1 igual a 0,8859 $\pm$ 0,0009 e AUC-ROC igual a 0,9541 $\pm$ 0,0004, o que indica estabilidade da configuração vencedora sob partições distintas do conjunto completo. A latência P50 abaixo de 6 ms foi medida no ambiente de *benchmark* do *notebook* no Google Colab, com inferência em *Central Processing Unit* (CPU) via *scikit-learn* sobre as 500 amostras de teste, e deve ser lida como referência de ordem de grandeza, não como garantia de desempenho em dispositivos de usuário final. Em uso real embarcado, a latência dependeria da CPU do usuário, do *runtime WebAssembly* do navegador e das restrições de *sandbox* da extensão, de modo que *hardware* mais modesto tenderia a apresentar valores superiores. Dispositivos móveis, ademais, estão fora do escopo da extensão entregue, que é destinada a navegadores *desktop*: o Chrome para Android não oferece suporte a extensões. Essa

dependência de *hardware* e de navegador, somada ao custo de *bundle* e à FPR alta do modelo, está entre os fatores que motivaram a remoção do modelo embarcado, discutida na Seção 5.2.

Na comparação entre o experimento histórico do TCC I sobre o conjunto UCI (DUA; GRAFF, 2017) (Tabela 4, com métricas do RF reproduzidas do *notebook* preliminar do TCC I e métricas do GBM extraídas em ambiente próprio) e o GBM treinado neste TCC II sobre o conjunto BR, observa-se uma queda das métricas, que pode ser explicada item a item. Primeiro, o conjunto BR é cerca de 60 vezes maior (660 838 URLs contra 11 055) e mais heterogêneo que o conjunto UCI, com forte dominância de uma das fontes de *phishing* (ver Seção 3.2 e Seção 5.7). Segundo, a queda de F1 (de $\approx$ 0,976 para 0,8866) e de AUC-ROC (de $\approx$ 0,996 para 0,9544) reflete uma tarefa intrinsecamente mais difícil, dado o recorte brasileiro e a dominância de uma única fonte de *phishing*, e não uma regressão do algoritmo. Terceiro, a FPR, ausente no relato do TCC I, passou a ser medida neste TCC II e revelou-se o gargalo, com valor próximo a 12 %. Em conjunto, a comparação indica que o desempenho elevado do TCC I era específico de um conjunto pequeno e internacional, e que a dificuldade do cenário brasileiro motivou a evolução para o modelo principal (DomURL-BERT) e para a cascata, discutida na Seção 5.2. A inviabilidade prática dessa FPR para um decisor único embarcado é detalhada a seguir.

Tabela 4 – Comparação histórica entre RF (TCC I, conjunto UCI) e GBM (TCC II, conjunto BR).

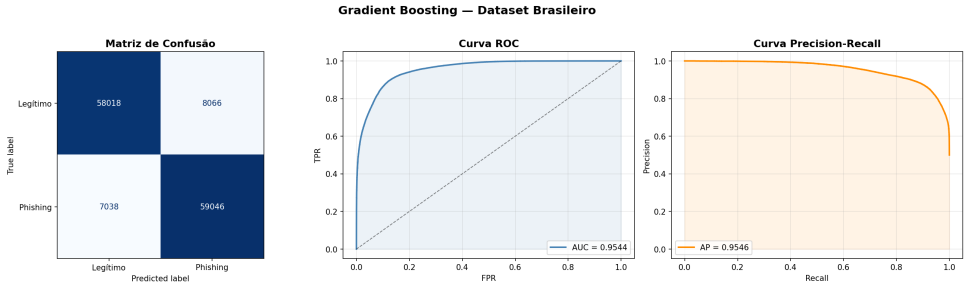
Item	**RF (TCC I, UCI)**	**GBM (TCC II, BR)**
Dataset	UCI Phishing Websites	Conjunto BR
Tamanho	11 055	660 838
F1	$\approx$ 0,976	0,8866
AUC-ROC	$\approx$ 0,996	0,9544
FPR	não reportado	$\approx 12{,}21\,\%$

Fonte: Autor (2026).

Os números reportados para o GBM no conjunto BR são pertinentes em si, mas o ponto crítico para a arquitetura da extensão é a Especificidade igual a 0,8779, que corresponde a FPR aproximada de 12 %. Esse valor está muito acima da meta inicial declarada na proposta (FPR $\leq 0{,}5\,\%$, ver Seção 1.1) e tornou inviável usar

o GBM como decisor único embarcado: a maior parte das URLs cotidianas seria rotulada incorretamente como suspeita e dispararia chamadas à *Application Programming Interface* (API) "de *fallback*" prevista na proposta original. Pelo critério da meta de proposta, o GBM foi **avaliado e descartado** como decisor único embarcado, e não abandonado sem teste. A consequência arquitetural dessa decisão é discutida na Seção 5.2. As curvas ROC e *Precision–Recall* obtidas para o GBM brasileiro estão reproduzidas na Figura 12.

Figura 12 – Curvas ROC e PR do GBM treinado sobre o conjunto BR de 660 838 URLs.



Fonte: Autor (2026).

5.2 Mudança de arquitetura

À luz da evidência empírica acumulada nos experimentos com modelos locais (Seção 5.1) e das observações em uso prático com a extensão instalada localmente, a arquitetura inicialmente prevista (extensão híbrida com modelo leve embarcado no cliente e *fallback* robusto no servidor) foi reformulada. Esta seção narra essa decisão como uma mudança de engenharia justificada por dado, e não como um recuo ou uma simplificação por desistência.

A versão preliminar do GBM foi exportada para *Open Neural Network Exchange* (ONNX) e instalada no cliente. A taxa de FPR observada em conjunto de teste fixo (cerca de 12 %) e o comportamento em rotina prática (URLs cotidianas como `google.com`, `amazon.com`, `netflix.com`, `mercadolivre.com` e `guri.unipampa.edu.br` foram rotuladas como *phishing* pelo modelo embarcado em testes do autor) indicaram que o decisor local não atendia ao papel pretendido. Adicionalmente, o *bundle* da extensão (o pacote final que o usuário baixa e instala) alcançava aproximadamente 95 MB com o modelo embutido, o tempo de carregamento em CPU era percebido pelo usuário e a manutenção exigia inferência paralela em duas

linguagens (TypeScript no cliente, Python no servidor). A combinação de FPR alta no caminho rotineiro, custo de *bundle*, latência local em CPU e custo de manutenção sustentou empiricamente a remoção do modelo do cliente.

A arquitetura adotada como solução final está descrita em detalhe no Capítulo 4 (Seções 4.1 a 4.9) e é representada no fluxo da Figura 7 daquele capítulo. Em síntese, o cliente passou a decidir localmente em tempo $O(1)$ por meio de três camadas (*whitelist*, *blacklist* e *cache* de resultados) e a inferência por modelo de aprendizado de máquina passou a residir exclusivamente na API. A complementaridade entre modelos prevista na proposta original foi preservada como cascata *server-side*. O DomURL-BERT URL+WHOIS decide diretamente quando opera com alta confiança, ou seja, quando a probabilidade estimada cai fora da zona ambígua. Na faixa intermediária ($0{,}15 < p_{\text{BERT}} < 0{,}85$), o CatBoost com 24 *features* enriquecidas é acionado para refinar a decisão. A decisão final combina os dois modelos pela média ponderada $0{,}6 \times p_{\text{BERT}} + 0{,}4 \times p_{\text{CatBoost}}$ e a URL é classificada como *phishing* quando o resultado supera o limiar de 0,65.

Duas consequências mensuráveis dessa reorganização merecem registro neste capítulo, por contextualizarem os resultados que seguem. A primeira é a redução do tamanho do *bundle* da extensão de aproximadamente 95 MB para aproximadamente 26 MB após a remoção do modelo embarcado. A segunda é que, no caminho rotineiro resolvido pelas três camadas locais, a FPR percebida pelo usuário tende a zero por construção: somente URLs que não constam na *whitelist*, na *blacklist* nem no *cache* chegam ao modelo. A FPR reportada nas seções seguintes refere-se, portanto, à fração de tráfego que entra na zona em que o modelo precisa decidir, e não ao tráfego total da extensão. A medição quantitativa dessa fração em produção está pendente (ver Seção 5.7).

5.3 DomURL-BERT URL + WHOIS

O modelo principal adotado na API é o DomURL-BERT *fine-tuned* sobre URL e campos de *Whois Information* (WHOIS) (MAHDAOUY *et al.*, 2024). A descrição do modelo, dos *special tokens* e dos hiperparâmetros do *fine-tuning* está em Desenvolvimento (Seção 4.8); esta seção concentra-se nos números obtidos em conjunto de teste fixo.

Em conjunto de teste do *corpus* BR, o DomURL-BERT URL+WHOIS apresentou as métricas reportadas na Tabela 5. As latências citadas correspondem ao mesmo ambiente de *benchmark* unificado (Google Colab com *Graphics Processing Unit* – GPU NVIDIA T4) usado para os demais modelos *transformer* da Seção 5.5, o que garante a

comparabilidade.

Tabela 5 – Métricas do DomURL-BERT URL+WHOIS (modelo principal adotado).

Métrica	**Valor**
Acurácia	0,8969
Precisão	0,8596
Revocação	0,9498
F1	0,9024
AUC-ROC	0,9632
PR-AUC	0,9630
MCC	0,7982
Log Loss	0,2627
Especificidade (FPR $\approx$ 15,64 %)	0,8436
Latência P50 / P95 / P99	9,01 / 11,82 / 15,79 ms
Tempo de treino	3 799,26 s ($\approx$ 63 min)

Fonte: Autor (2026).

O F1 igual a 0,9024 e a Revocação igual a 0,9498 são compatíveis com a literatura comparável de detecção de *phishing* baseada em URL com *transformers* (MAHDAOUY *et al.*, 2024; MANERIKER *et al.*, 2021; LI *et al.*, 2024). A Revocação superou a meta inicial declarada na proposta (Taxa de Verdadeiros Positivos, *True Positive Rate* – TPR, próxima de 90 %; ver Seção 1.1). A Especificidade igual a 0,8436, equivalente a FPR aproximada de 15,64 % no conjunto de teste fixo, ficou substancialmente acima da meta inicial de FPR $\leq 0{,}5\,\%$. A discussão dessa lacuna está nas Seções 5.5 e 5.8.

Para isolar o ganho atribuível ao DomURL-BERT como *backbone* específico de URL, isto é, o modelo pré-treinado de base sobre o qual se aplica o *fine-tuning*, a mesma rotina de ajuste foi aplicada ao SecureBERT (AGHAEI *et al.*, 2022), modelo *transformer* especializado em texto de cibersegurança. O resultado em modo URL apenas (F1 igual a 0,8992; AUC-ROC igual a 0,9617; Especificidade igual a 0,8452) ficou ligeiramente abaixo do DomURL-BERT URL+WHOIS, com diferença consistente porém pequena (cerca de 0,003 em F1). A Tabela completa com os quatro modelos *transformer* está na Seção 5.5.

5.4 DomURL-BERT multimodal

A variante multimodal do DomURL-BERT (*fine-tuned* sobre URL, campos de WHOIS e *features* adicionais derivadas do *Document Object Model* (DOM) e de outras características da página) foi avaliada como linha de comparação e como teste empírico da hipótese de que mais informação de entrada melhora o desempenho do modelo. Esta seção apresenta as métricas dessa variante; a discussão da decisão de não adotá-la como modelo principal está na Seção 5.5.

Em conjunto de teste fixo, o DomURL-BERT multimodal apresentou as métricas reportadas na Tabela 6.

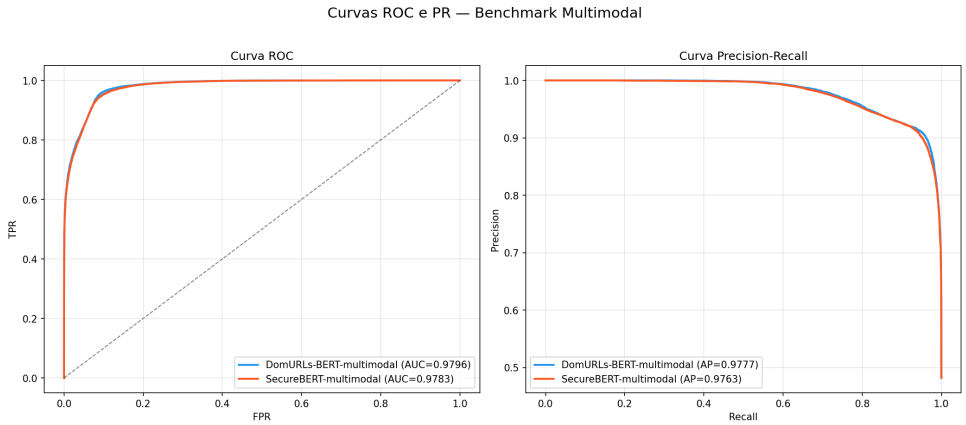
Tabela 6 – Métricas do DomURL-BERT multimodal (avaliado e não adotado como modelo principal).

Métrica	**Valor**
Acurácia	0,9308
Precisão	0,9081
Revocação	0,9528
F1	0,9299
AUC-ROC	0,9796
PR-AUC	0,9777
MCC	0,8626
Log Loss	0,1980
Especificidade (FPR $\approx$ 8,97 %)	0,9103
Latência P50 / P95 / P99	11,67 / 17,20 / 20,45 ms
Tempo de treino	5 168,94 s ($\approx$ 86 min)

Fonte: Autor (2026).

As curvas ROC e *Precision–Recall* dessa variante estão reproduzidas na Figura 13. Em todas as métricas agregadas reportadas pelo *benchmark* unificado, o multimodal apresentou-se marginalmente superior ao DomURL-BERT URL+WHOIS adotado como modelo principal. Esse fato é registrado abertamente e não é compatível com a hipótese ingênua de que o multimodal seria simplesmente "pior". A divergência entre métricas em conjunto de teste fixo e comportamento operacional observado em validação empírica é discutida em detalhe na Seção 5.5; ela é um dos resultados reportáveis deste trabalho.

Figura 13 – Curvas ROC e PR do DomURL-BERT multimodal em conjunto de teste fixo.



Fonte: Autor (2026).

Como linha de comparação adicional, o SecureBERT foi também treinado em modo multimodal (AGHAEI *et al.*, 2022), com F1 igual a 0,9252; AUC-ROC igual a 0,9783; Especificidade igual a 0,9117 e Latência P50 igual a 12,34 ms. Os números, novamente, ficam ligeiramente abaixo do DomURL-BERT multimodal, com diferença pequena e consistente. A tabela completa com os quatro modelos está na próxima seção.

5.5 Comparação e decisão de projeto

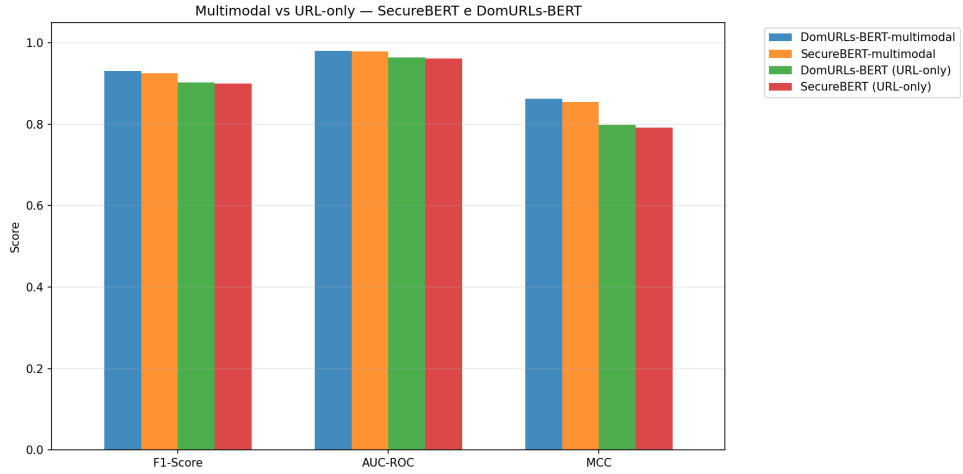
Esta seção reúne a comparação direta entre os modelos avaliados e justifica a decisão de adotar o DomURL-BERT URL+WHOIS como modelo principal, mesmo com métricas agregadas de teste fixo marginalmente piores do que as do multimodal. A Tabela 7 consolida as métricas dos quatro modelos *transformer* avaliados no mesmo ambiente de *benchmark*; a ressalva sobre a comparabilidade entre os modos URL apenas e multimodal é discutida na Seção 5.8.

Tabela 7 – Comparação dos quatro modelos *transformer* avaliados no mesmo ambiente de *benchmark*.

Modelo	**F1**	**AUC-ROC**	**MCC**	**Recall**	**Esp.**	**FPR**	**P50 (ms)**	**P95 (ms)**
DomURL-BERT URL+WHOIS (adotado)	0,9024	0,9632	0,7982	0,9498	0,8436	15,64 %	9,01	11,82
DomURL-BERT multimodal (não adotado)	0,9299	0,9796	0,8626	0,9528	0,9103	8,97 %	11,67	17,20
SecureBERT URL apenas	0,8992	0,9617	0,7916	0,9424	0,8452	15,48 %	12,51	15,37
SecureBERT multimodal	0,9252	0,9783	0,8536	0,9425	0,9117	8,83 %	12,34	19,86

Fonte: Autor (2026). Esp. = Especificidade; FPR = 1 − Esp.

Figura 14 – Comparação visual entre as variantes URL+WHOIS e multimodal do DomURL-BERT.



Fonte: Autor (2026).

A leitura objetiva da Tabela 7 e da Figura 14 pode ser sumarizada em três pontos. Primeiro, o DomURL-BERT é consistentemente superior ao SecureBERT em ambos os modos (URL apenas e multimodal), com diferença pequena (cerca de 0,003 em F1 em ambos os modos) mas consistente, o que sugere que parte do desempenho observado vem do *backbone* especializado em URL e não exclusivamente do *backbone* de cibersegurança, ainda que a diferença não seja decisiva. Segundo, em ambos os *backbones*, o modo multimodal tem melhor F1, AUC-ROC, MCC e Especificidade do que o modo URL

apenas; o ganho em Especificidade é de cerca de 7 pontos percentuais, com queda da FPR de 15,5 % para 9,0 %. Terceiro, o custo computacional acompanha: o multimodal tem latência P50 cerca de 30 % maior e P95 cerca de 45 % maior do que o modo URL apenas. Pelo critério estrito da Tabela 7, o DomURL-BERT multimodal seria o modelo de melhor compromisso métrica/latência.

A decisão de adotar a versão URL+WHOIS, e não a multimodal, foi baseada em validação empírica complementar. Durante a fase de integração da API com a extensão, o autor conduziu rodadas qualitativas sobre URLs reais, sustentadas pelos artefatos `validacao_modelo.json` (62 URLs) e `validacao_cascata.json` (101 URLs: 51 legítimas e 50 *phishing*), produzidos pelo *script* `whitelist/validacao_modelo.py` e executados em 2026-03-29. Esses arquivos correspondem, em modo qualitativo, à avaliação em campo prevista para a Semana 15 do cronograma original. Os números relevantes para esta seção estão nas Tabelas 8 e 9.

Tabela 8 – Validação empírica 1: DomURL-BERT URL+WHOIS sem listas, sem *cache* e sem cascata, sobre 62 URLs.

Item	**Valor**
URLs testadas	62 (36 legítimas + 26 *phishing*)
Acurácia global	41,9 % (26 acertos)
Falsos positivos sobre legítimas	36
Falsos negativos sobre *phishing*	0
Confiança típica dos falsos positivos	> 95 %

Fonte: Autor (2026).

Tabela 9 – Validação empírica 2: comparação entre DomURL-BERT puro, CatBoost puro e cascata BERT+CatBoost sobre 101 URLs.

Métrica	**BERT puro**	**CatBoost puro**	**Cascata**
Acurácia	94,1 %	99,0 %	96,0 %
Precisão	89,3 %	100,0 %	92,6 %
Revocação	100,0 %	98,0 %	100,0 %
F1	94,3 %	99,0 %	96,2 %
Falsos positivos	6	0	4
Falsos negativos	0	1	0
Latência P50 (ms)	47,4	2 002,6	44,8
Latência P95 (ms)	55,4	4 007,5	83,8

Fonte: Autor (2026).

Tabela 10 – Validação empírica 3: rodada formal sobre lista pré-registrada de 1 946 URLs (1 013 legítimas e 933 *phishing*).

Métrica	**URL+WHOIS (adotado)**	**Multimodal**
Verdadeiros Positivos (VP)	843	832
Falsos Positivos (FP)	66	68
Verdadeiros Negativos (VN)	947	945
Falsos Negativos (FN)	90	101
Acurácia	0,9198	0,9132
Precisão	0,9274	0,9244
Revocação	0,9035	0,8917
F1	0,9153	0,9078
FPR	6,52 %	6,71 %
MCC	0,8395	0,8263
Latência P50 (ms)	34,97	38,38
Latência P95 (ms)	45,73	49,83
Latência P99 (ms)	65,28	73,35

Fonte: Autor (2026).

A rodada formal endereça a pendência metodológica reconhecida em trabalhos

anteriores deste mesmo capítulo: comparar o DomURL-BERT URL+WHOIS adotado com a variante multimodal não adotada por meio de uma lista pré-registrada e medição automática de Verdadeiros Positivos, Falsos Positivos e latência fim-a-fim. A lista, gerada de modo determinístico (*seed* fixa) a partir das fontes consolidadas do projeto (uma *whitelist* com cerca de 32 000 domínios e uma *blacklist* com cerca de 774 000 URLs reais, ambas montadas a partir de *feeds* públicos como Tranco, Majestic, PhishTank e OpenPhish, entre outros), balanceia 1 013 URLs legítimas e 933 URLs de *phishing*. Entre as URLs legítimas, foram priorizadas marcas globais (*big-tech*, serviços financeiros) e domínios brasileiros (bancos, governo, universidades, mídia, viagem, saúde). Entre as URLs de *phishing*, foram priorizadas instâncias reais que imitam essas mesmas marcas, condição que aproxima a avaliação do cenário operacional da extensão no Brasil.

A leitura da Tabela 10 confirma empiricamente a decisão de projeto registrada nas seções anteriores. O URL+WHOIS apresentou Acurácia, Precisão, Revocação, F1 e MCC superiores ao multimodal e foi mais rápido em todos os percentis de latência (P50 cerca de 9 % inferior, P95 cerca de 9 % inferior, P99 cerca de 12 % inferior). A Especificidade dos dois modelos foi praticamente indistinguível na prática (FPR de 6,52 % para o URL+WHOIS e 6,71 % para o multimodal, com diferença de apenas 0,19 ponto percentual). O multimodal não trouxe ganho operacional que justificasse o custo adicional de latência, ao contrário do que sugeriam suas métricas ligeiramente superiores no conjunto de teste fixo (Tabela 7). Esse contraste entre a métrica isolada de *benchmark* e o comportamento operacional sobre URLs reais brasileiras é precisamente o argumento técnico que sustenta a escolha do URL+WHOIS como modelo principal adotado.

A rodada formal também documenta achado independente sobre a qualidade dos *feeds* públicos consolidados na *blacklist*. Da amostra original de 946 URLs marcadas como *phishing* na geração da lista, a inspeção manual identificou 13 que correspondem a órgãos públicos brasileiros legítimos (câmaras municipais como `crecise.gov.br`, sistemas de licitação como `licitacao.rc.sp.gov.br`, prefeituras como `brejogrande.se.gov.br`), inseridas em algum dos *feeds upstream* por engano. Essas 13 URLs (cerca de 1,4 % do conjunto inicial de *phishing*) foram reclassificadas como legítimas antes do cálculo das métricas reportadas na Tabela 10; o critério adotado foi manter como *phishing* apenas os domínios que usam `gov.br` como subdomínio enganoso (por exemplo, `mda.gov.br.tripod.com`) e reclassificar como legítimos os domínios que terminam exatamente em `.gov.br`, `.leg.br` ou `.jus.br`. A taxa de contaminação observada é compatível com relatos

da literatura sobre *ground truth* de *feeds* públicos de *phishing* e fica aqui reportada como observação metodológica.

Dois fatos da Tabela 8 sustentaram a presença obrigatória das três camadas locais na arquitetura final. Primeiro, sem proteção das listas, o DomURL-BERT URL+WHOIS rotulou como *phishing* 36 das 36 URLs legítimas brasileiras de alto perfil testadas, entre elas `bb.com.br`, `itau.com.br`, `caixa.gov.br`, `nubank.com.br`, `gov.br`, `globo.com`, `mercadolivre.com.br` e `wikipedia.org`, com confiança frequentemente acima de 95 %. Segundo, todas as 26 URLs simuladas de *phishing* foram corretamente rotuladas. Esse comportamento (Revocação muito alta combinada com FPR muito alta sobre legítimas brasileiras) é incompatível com uso direto em uma extensão de navegador. A presença da *whitelist* de aproximadamente 32 000 domínios na arquitetura final (Seção 4.4) deixa de ser apenas otimização e passa a ser camada de proteção contra esse comportamento.

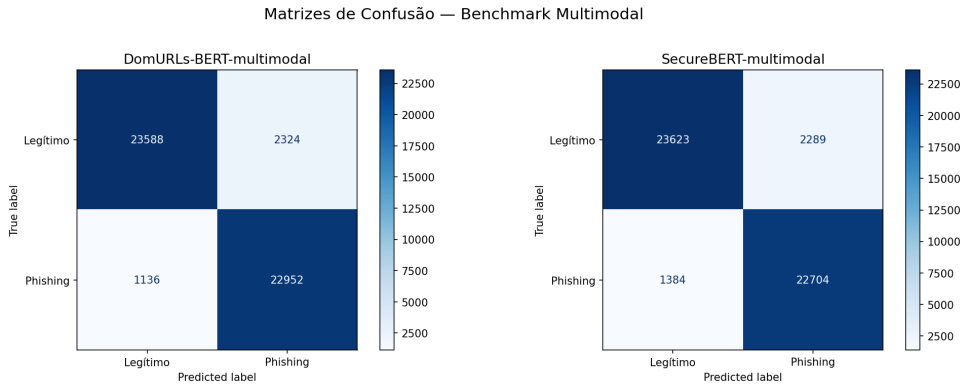
A Tabela 9 reforça o valor da cascata: ela reduziu de 6 para 4 o número de falsos positivos do BERT puro e manteve Revocação igual a 100 %, com latência P50 da ordem da do BERT puro (44,8 ms contra 47,4 ms) e P95 da ordem do dobro (83,8 ms contra 55,4 ms), mas muito menor do que a do CatBoost puro com consultas WHOIS, *Domain Name System* (DNS) e *Transport Layer Security* (TLS) sequenciais (P50 acima de 2 000 ms). Os falsos positivos residuais da cascata foram `banco.bradesco`, `outlook.live.com`, `mail.google.com` e `chatgpt.com`, domínios com estrutura particular (subdomínios curtos, ausência de *path*, *Top-Level Domain* fora do `.com.br` tradicional) que são naturalmente atendidos pela *whitelist* no fluxo completo da extensão.

A justificativa para adotar o DomURL-BERT URL+WHOIS, e não o multimodal, foi observada empiricamente durante a integração: o multimodal apresentou maior incidência de falsos positivos em URLs reais cotidianas brasileiras, mesmo com métricas marginalmente superiores em conjunto de teste fixo. Essa observação está sustentada pelo comportamento documentado na Tabela 8 para o URL+WHOIS, pela observação direta do autor para o multimodal e, complementarmente, pela rodada formal sobre 1 946 URLs reportada na Tabela 10, na qual o URL+WHOIS manteve-se superior em F1, Precisão, Revocação e MCC, e mais rápido em todos os percentis de latência, o que confirma o critério de adoção sob condição operacional. Em uma extensão de navegador, FPR alto degrada a experiência do usuário e gera fadiga de alerta, com efeito direto sobre a aceitação da ferramenta. Por isso, o critério decisivo entre URL+WHOIS e multimodal

foi de *User Experience* (UX), e não a métrica isolada.

Esta divergência (métricas marginalmente superiores no conjunto de teste fixo, comportamento operacional inferior em validação empírica) é uma observação reportável do trabalho. Mais informação na entrada do modelo nem sempre se traduz em melhor desempenho operacional. A causa subjacente da diferença permanece em hipótese (heterogeneidade dos DOMs em sites legítimos, diluição do espaço de *tokens* por sinais menos informativos, sobreajuste à distribuição do conjunto de treino) e não foi confirmada experimentalmente; uma análise sistemática fica como trabalho futuro (Seção 5.10). As matrizes de confusão das duas variantes estão reproduzidas na Figura 15.

Figura 15 – Matrizes de confusão das variantes URL+WHOIS e multimodal do DomURL-BERT no conjunto de teste fixo.



Fonte: Autor (2026).

5.6 Explicabilidade das decisões

Por explicabilidade, este trabalho entende a capacidade de, para uma decisão específica do classificador (esta URL é *phishing*, este *e-mail* é *phishing*), indicar quais sinais da entrada pesaram nessa decisão. O conceito não se confunde com a descrição da arquitetura do modelo, que é documentação de projeto; trata-se de responder, decisão a decisão, à pergunta "por que o modelo decidiu assim". Em segurança, essa capacidade importa para auditoria, depuração de erros e construção de confiança por parte do operador.

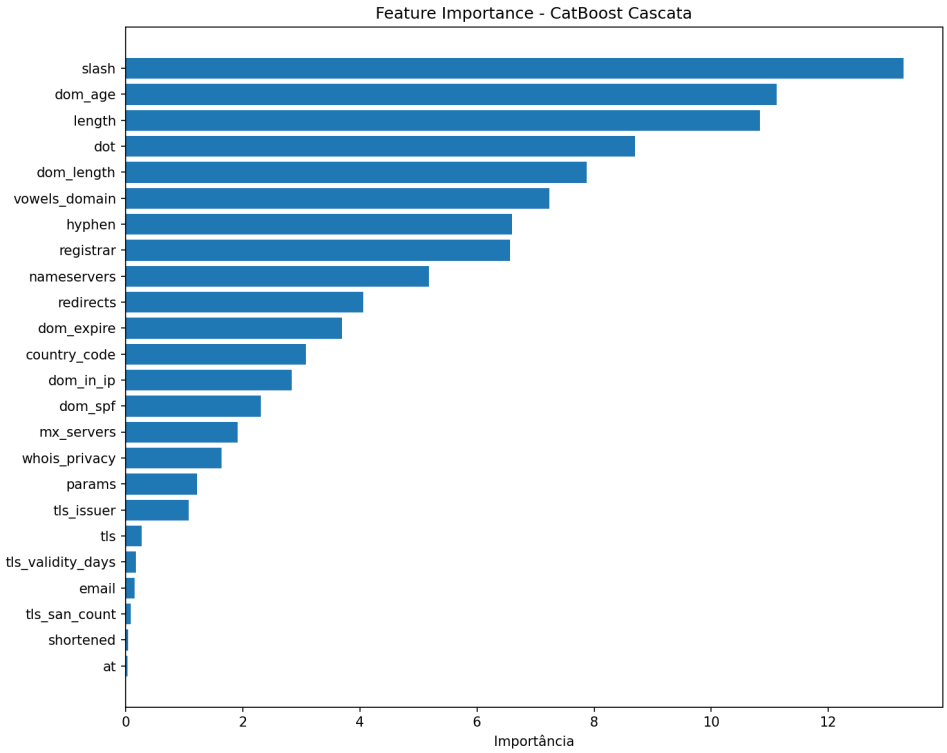
Os três modelos da solução receberam atribuição por decisão. O CatBoost da cascata foi explicado por *SHapley Additive exPlanations* (SHAP) (LUNDBERG; LEE, 2017), técnica que mede quanto cada sinal de entrada contribuiu para a decisão: ela

reparte o resultado do modelo entre esses sinais e atribui a cada um um valor com sinal positivo ou negativo. Esses sinais de entrada são chamados de *features*, e cada *feature* é uma característica medível da URL, como o seu comprimento ou a idade do domínio. O DomURL-BERT e o DistilBERT, por trabalharem diretamente sobre texto, foram explicados por duas técnicas complementares. A primeira é o SHAP por *token*, em que um *token* é cada pedaço (uma palavra, um fragmento de palavra ou um símbolo) em que o modelo divide o texto ou a URL antes de processá-los, e o SHAP mede a contribuição de cada pedaço. A segunda é o mapa de atenção, recurso interno do modelo que revela em quais *tokens* ele mais concentra o processamento ao decidir. Em todas as figuras desta seção vale a mesma convenção de sinal: um valor positivo empurra a decisão em direção ao rótulo *phishing* e um valor negativo, em direção ao rótulo legítimo. As contribuições do SHAP aparecem na escala interna de pontuação do modelo, chamada de log-chances (*logits*); para a leitura das figuras, importam o sinal (positivo ou negativo) e o tamanho relativo de cada barra, e não o valor absoluto.

5.6.1 CatBoost da cascata

O CatBoost é o caso mais legível dos três modelos, e sua explicação é apresentada em três níveis crescentes de detalhe. O primeiro é a importância nativa de *features*, métrica interna do modelo reproduzida na Figura 16, que ordena os 24 atributos pela influência média sobre o conjunto das decisões refinadas pelo CatBoost. A leitura é apenas de magnitude: o atributo `slash` (número de barras na URL) aparece no topo, seguido de `dom_age` (idade do domínio), `length` (comprimento da URL), `dot` (número de pontos) e `dom_length` (comprimento do domínio). A métrica revela quais atributos o modelo mais usa, mas não informa se cada um empurra a decisão para *phishing* ou para legítimo.

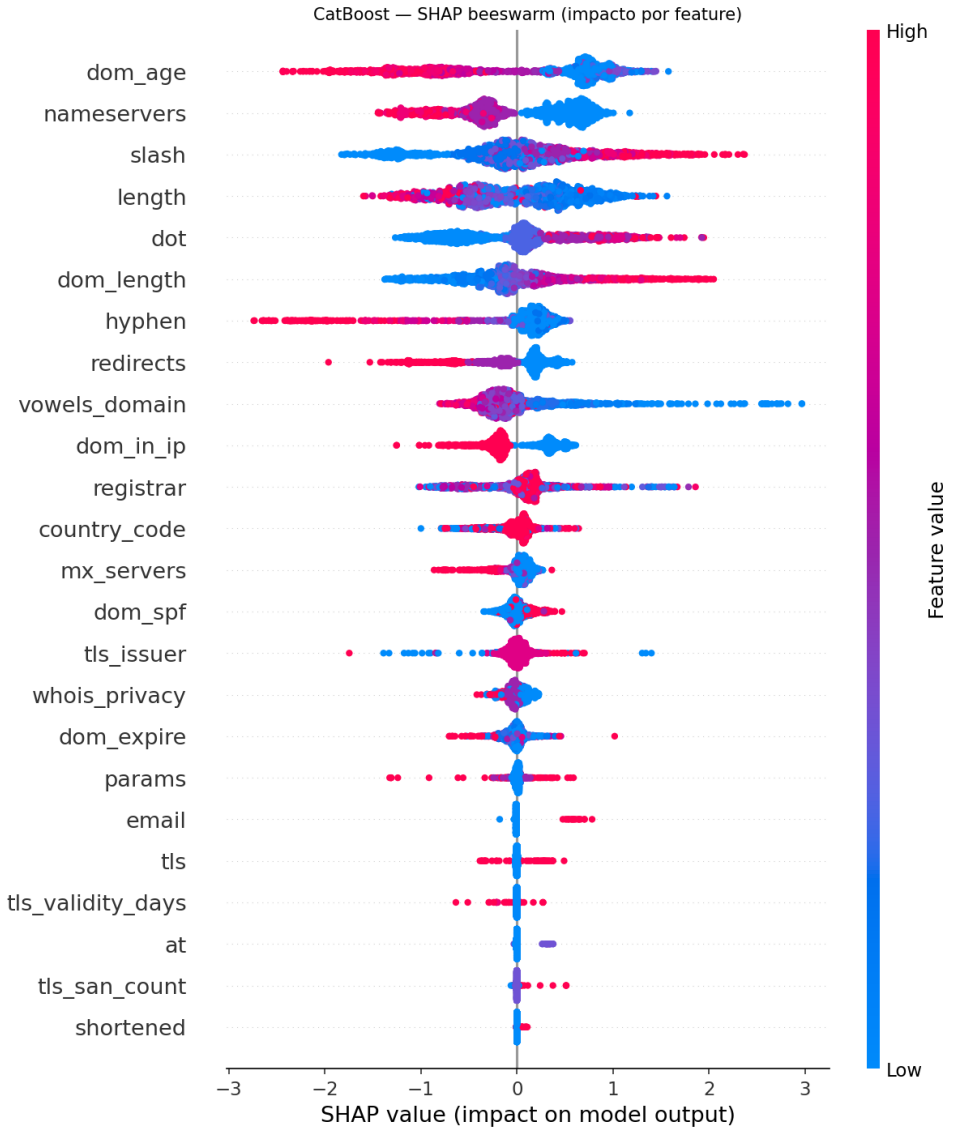
Figura 16 – Importância nativa de *features* do CatBoost da cascata.



Fonte: Autor (2026).

O segundo nível é o gráfico *beeswarm* da Figura 17, gerado pela variante TreeSHAP, específica para modelos de árvore. Cada linha corresponde a uma *feature*, ordenada de cima para baixo pela importância média; cada ponto é uma URL do conjunto avaliado; a posição horizontal é o valor SHAP daquela URL, com o zero ao centro e o eixo variando de cerca de -3 a $+3$ na escala de saída do modelo; a cor indica o valor da *feature*, do azul (valores baixos) ao vermelho (valores altos). As *features* de WHOIS `dom_age` e `nameservers` lideram a ordenação, à frente de atributos lexicais como `slash`, `length` e `dot`. As três *features* categóricas (`registrar`, `country_code` e `tls_issuer`) aparecem na parte intermediária da lista. A largura da dispersão dos pontos em torno do zero mostra que o efeito de cada atributo varia conforme o valor que ele assume em cada URL, em vez de ser fixo.

Figura 17 – Valores SHAP globais (*beeswarm*) das 24 *features* do CatBoost da cascata.

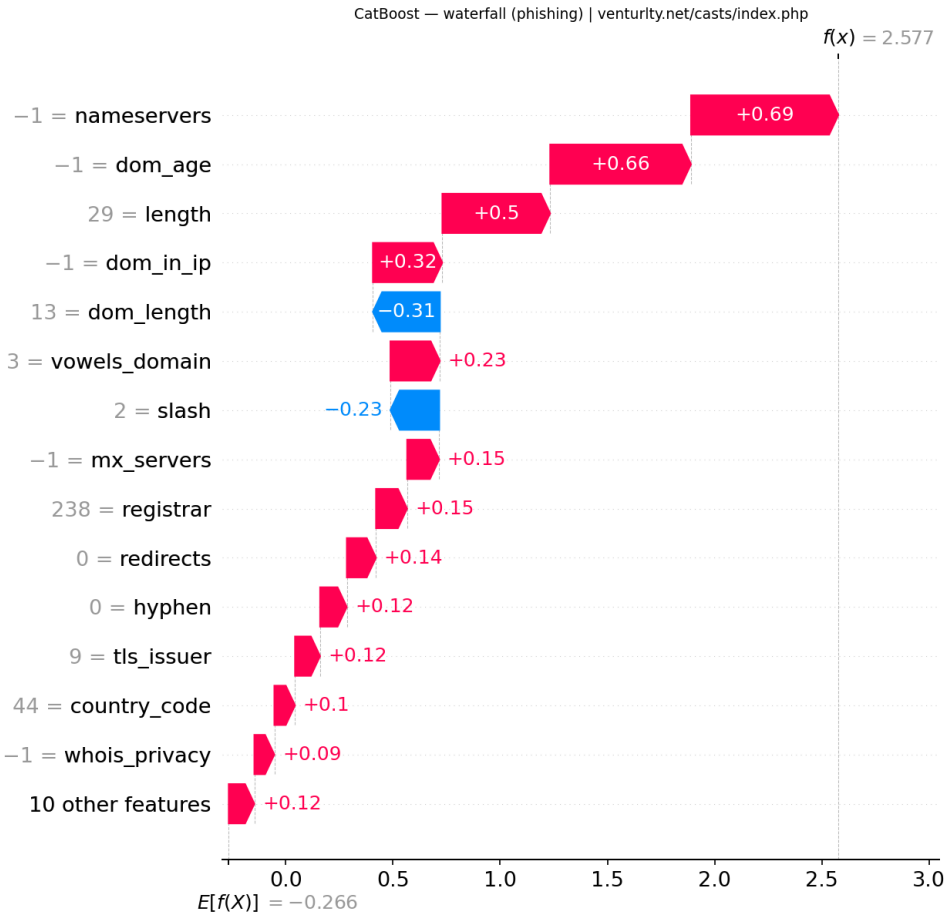


Fonte: Autor (2026).

O terceiro nível é o gráfico *waterfall* da Figura 18, que desce a uma decisão individual. Para a URL de *phishing* `venturlty.net/casts/index.php`, o gráfico parte do valor base do modelo, $E[f(X)] = -0{,}266$ (a saída média sobre o conjunto), e soma a contribuição de cada *feature* até a saída final $f(x) = 2{,}577$, valor alto que corresponde à classe *phishing*. As maiores contribuições partem de atributos de WHOIS que o sistema não conseguiu resolver, marcados com o valor -1: a ausência de `nameservers` ($+0{,}69$), a idade de domínio desconhecida em `dom_age` ($+0{,}66$) e a falta de registro de IP do domínio em `dom_in_ip` ($+0{,}32$). Em seguida vêm atributos lexicais, como o comprimento da URL igual a 29 em `length` ($+0{,}5$) e as três vogais

do domínio em `vowels_domain` (+0,23). Poucos atributos empurram em sentido contrário, como o comprimento do domínio igual a 13 em `dom_length` (−0,31) e as duas barras da URL em `slash` (−0,23). A leitura é direta: para esta URL, a falta de dados de registro confiáveis foi o principal sinal de fraude, comportamento coerente com domínios recém-criados ou opacos.

Figura 18 – Decomposição SHAP (*waterfall*) de uma decisão individual de *phishing* do CatBoost.



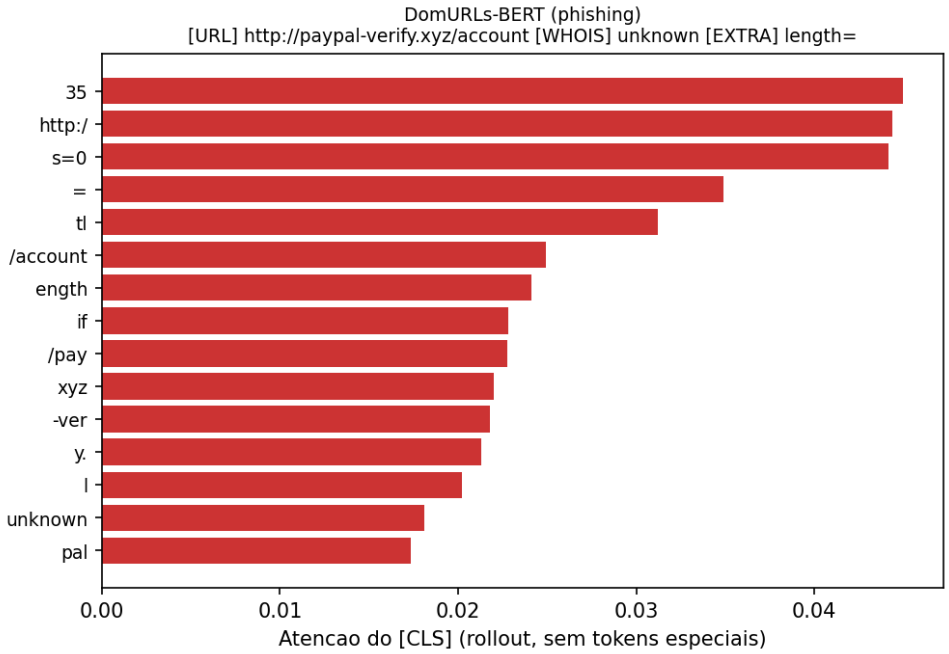
Fonte: Autor (2026).

5.6.2 DomURL-BERT do classificador de URL

Para o DomURL-BERT URL+WHOIS, classificador principal de URL, a explicação por decisão combina o mapa de atenção e o SHAP por *token*, sobre a entrada no formato em que o modelo foi treinado, com os marcadores `[URL]`, `[WHOIS]` e `[EXTRA]`. O mapa de atenção da Figura 19 usa *attention rollout* do *token* `[CLS]`,

técnica que acompanha como a atenção se propaga pelas camadas do *transformer* até o *token* que resume a sequência (ABNAR; ZUIDEMA, 2020); o eixo horizontal é a atenção recebida por cada *token*, e os *tokens* especiais de controle foram excluídos do gráfico, de modo que aparecem apenas os *tokens* de conteúdo. Para a URL de *phishing* `http://paypal-verify.xyz/account`, as maiores atenções recaem sobre o valor de comprimento da URL (*token* `35`), o esquema `http:` sem criptografia e o marcador `s=0`, que registra a ausência de TLS. Logo abaixo aparecem os trechos de caminho `/account` e `/pay`, o domínio de topo barato `xyz` e os pedaços `-ver` e `pal` do domínio falso "paypal-verify". A atenção, portanto, distribui-se justamente sobre os elementos que distinguem uma imitação de marca de um endereço legítimo.

Figura 19 – Atenção (*attention rollout* do `[CLS]`, sem *tokens* especiais) do DomURL-BERT sobre a URL de *phishing* `paypal-verify.xyz`.

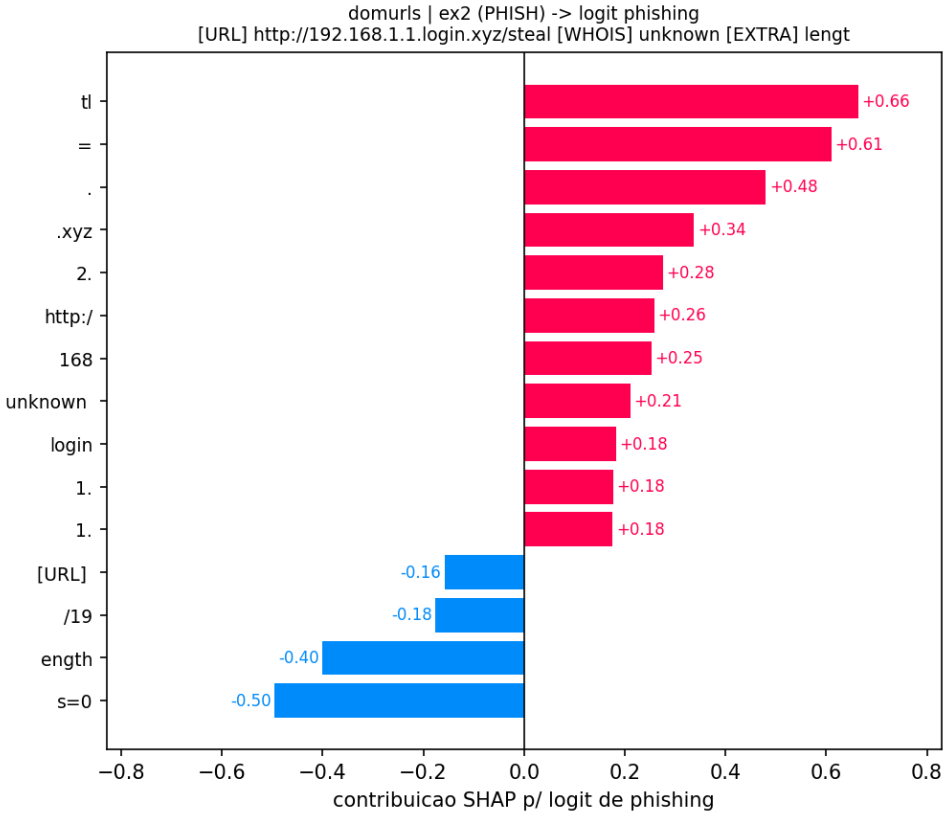


Fonte: Autor (2026).

O SHAP por *token* da Figura 20 quantifica, para a URL de *phishing* `http://192.168.1.1.login.xyz/steal`, a contribuição de cada *token* para a decisão, na escala de log-chances. As barras vermelhas, à direita do zero, empurram para *phishing*, e as azuis, à esquerda, para legítimo. Os sinais mais fortes em direção à fraude são o domínio de topo `.xyz` (+0,34), o esquema `http:` (+0,26), os fragmentos do endereço IP `192.168.1.1` (*tokens* `2.` com +0,28, `168` com +0,25 e `1.` com +0,18), o WHOIS `unknown` (+0,21) e a palavra `login` (+0,18). Alguns *tokens* estruturais puxam em sentido oposto, como o marcador de comprimento `ength` (−0,40) e `s=0`

($-0{,}50$). A tokenização em subpalavras quebra a URL em pedaços pequenos, como `tl`, = e o ponto isolado, o que torna parte das barras menos interpretável; ainda assim, os *tokens* de maior peso correspondem aos sinais clássicos de URL maliciosa: endereço IP no lugar de domínio, domínio de topo barato e ausência de HTTPS.

Figura 20 – SHAP por *token* do DomURL-BERT sobre a URL de *phishing* `192.168.1.1.login.xyz`.



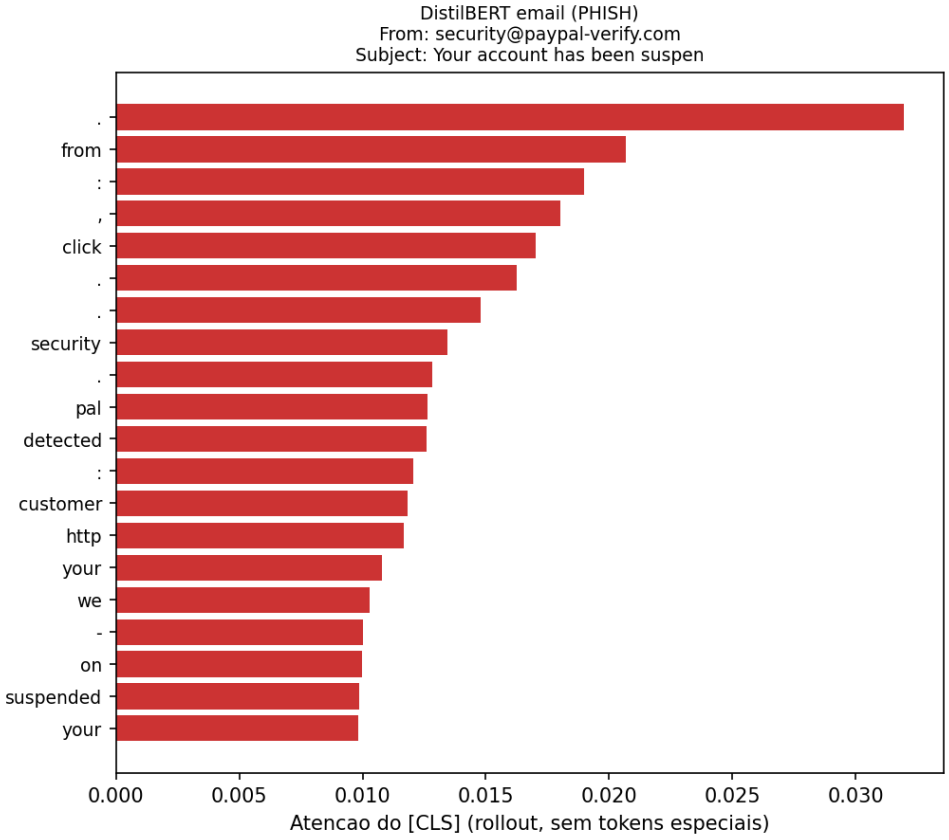
Fonte: Autor (2026).

5.6.3 DistilBERT do *pipeline* de *e-mail*

Assim como no classificador de URL, também para o DistilBERT do *pipeline* de *e-mail* empregam-se o *attention rollout* do *token* `[CLS]` (ABNAR; ZUIDEMA, 2020) e o SHAP por *token*, agora sobre a concatenação de remetente, assunto e corpo da mensagem. A Figura 21 apresenta esse mapa de atenção, também sem os *tokens* especiais, para um *e-mail* de *phishing* com remetente `security@paypal-verify.com` e assunto "Your account has been suspended". Entre os *tokens* de maior atenção estão o verbo de ação `click`, o termo `security`, o fragmento `pal` de "paypal", as palavras `detected` e

`suspended`, que constroem o senso de urgência, o `http` do *link* e a saudação genérica `customer`. Os sinais de pontuação que estruturam os campos `From:` e `Subject:` também recebem atenção. O modelo concentra o processamento no vocabulário típico de mensagens fraudulentas: chamada para ação, urgência e imitação de marca.

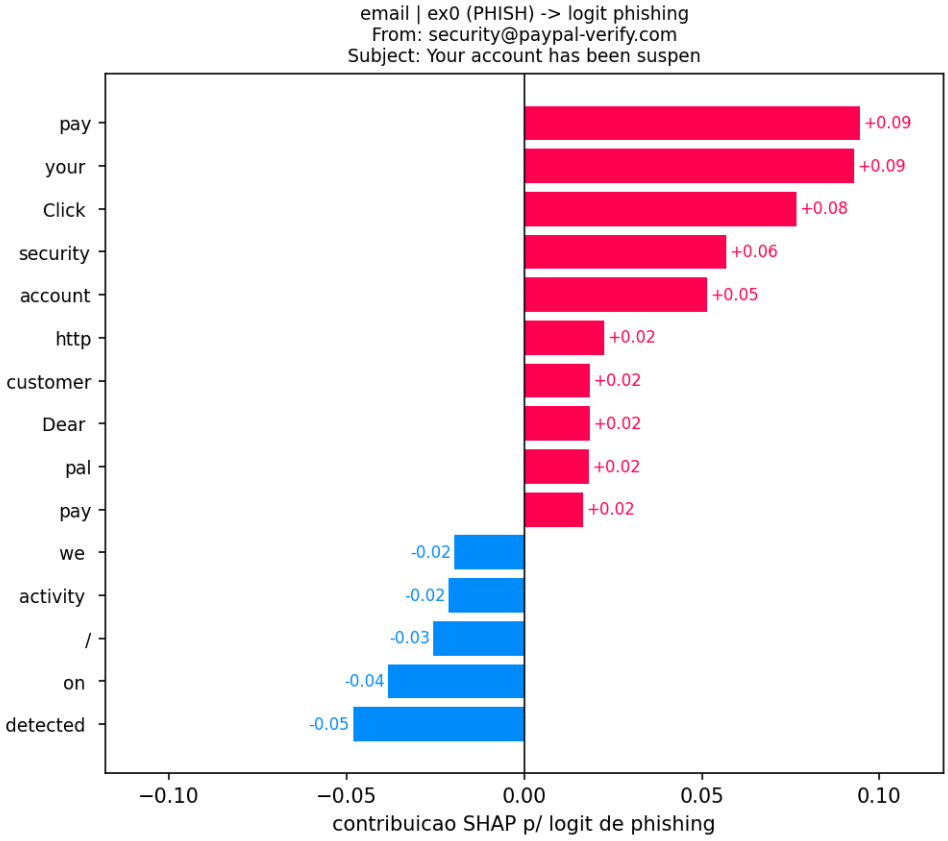
Figura 21 – Atenção (*attention rollout* do `[CLS]`, sem *tokens* especiais) do DistilBERT sobre um *e-mail* de *phishing*.



Fonte: Autor (2026).

O SHAP por *token* da Figura 22, sobre o mesmo *e-mail* e na escala de log-chances, confirma esses sinais e os ordena por contribuição. Empurram para *phishing* os fragmentos `pay` e `pal` da marca imitada "paypal" ($+0{,}09$ e $+0{,}02$), o verbo `Click` ($+0{,}08$), os termos `security` ($+0{,}06$) e `account` ($+0{,}05$), a saudação `Dear` ($+0{,}02$), `customer` ($+0{,}02$) e o `http` do *link* ($+0{,}02$). Em sentido contrário pesam, com magnitude menor, palavras neutras como `detected` ($-0{,}05$), `on` ($-0{,}04$) e a barra do endereço ($-0{,}03$). As duas técnicas convergem para o mesmo conjunto de sinais, isto é, marca imitada, chamada para ação, urgência e *link*, o que reforça a leitura da decisão.

Figura 22 – SHAP por *token* do DistilBERT sobre o mesmo *e-mail* de *phishing*.



Fonte: Autor (2026).

O DistilBERT exige uma ressalva adicional. A decisão opera sobre o texto traduzido para o inglês pela etapa de tradução automática do português para o inglês executada por MarianMT (Seção 4.11), e não sobre o texto original em português. Os *tokens* destacados pela atenção e pelo SHAP pertencem, portanto, à versão traduzida, o que insere uma camada adicional a explicar: o sinal atribuído à decisão depende também da fidelidade da tradução, cuja contribuição de erro não foi quantificada neste trabalho.

Os três modelos da solução passam, assim, a ter atribuição por decisão. O CatBoost recebe explicação em três níveis, da importância global ao *waterfall* de uma decisão; o DomURL-BERT e o DistilBERT, antes tratados como caixa-preta, passam a ter SHAP por *token* e mapas de atenção que apontam quais trechos da entrada pesaram em cada veredicto, com convergência entre as duas técnicas no caso do *e-mail*. Permanecem como trabalho futuro a aplicação de *Local Interpretable Model-agnostic Explanations* (LIME) e de gradiente integrado, a calibração das explicações e a análise sistemática de viés e justiça, conforme registrado na Seção 5.10.

5.7 Limitações experimentais

Os experimentos relatados foram conduzidos com os recursos disponíveis ao autor, sem auxílio monetário específico para infraestrutura de treinamento. O treinamento dos modelos foi feito no Google Colab com GPU NVIDIA T4 da versão gratuita do serviço. Os testes locais e a validação da API foram executados em CPU no computador pessoal do autor (processador AMD Ryzen 5 5600X de seis núcleos, 24 GB de memória DDR4 e GPU AMD Radeon RX 6600), com os serviços empacotados em contêineres Docker, uma vez que a GPU AMD não é diretamente utilizável para inferência PyTorch CUDA padrão dos modelos BERT. Essa moldura impõe seis dimensões de restrição:

- a quantidade de experimentos realizados;
- o tempo de treinamento dedicado a cada modelo;
- o tamanho dos modelos que puderam ser explorados, sem investigação de DeBERTa-v3-large, modelos de linguagem grandes ou variantes BERT maiores;
- a profundidade da busca de hiperparâmetros;
- a impossibilidade de repetir treinamentos com múltiplas *seeds* para análise de variância;
- a inviabilidade de explorar arquiteturas mais custosas.

Os resultados reportados nas seções anteriores devem ser lidos como o melhor que pôde ser obtido dentro dessas condições.

Além da moldura de recursos, o trabalho reconhece sete limitações experimentais adicionais.

A avaliação em campo prevista para a Semana 15 do cronograma original foi conduzida em duas etapas. A primeira foi qualitativa, com os artefatos consolidados em `validacao_modelo.json` (62 URLs) e `validacao_cascata.json` (101 URLs), discutidos na Seção 5.5. A segunda foi formal, sobre lista pré-registrada de 1 946 URLs, com medição automática de Verdadeiros Positivos, Falsos Positivos, Verdadeiros Negativos, Falsos Negativos e latência fim-a-fim por modelo, reportada na Tabela 10. A rodada formal foi executada no computador pessoal do autor, com os mesmos modelos carregados via biblioteca `transformers` (sem passar pela API da extensão, para isolar a latência de inferência da latência de rede).

A calibração formal de probabilidades do modelo principal e da cascata não foi conduzida. Os limiares 0,15, 0,85 e 0,65 que governam a cascata foram escolhidos empiricamente durante o desenvolvimento; um trabalho de calibração com método como `CalibratedClassifierCV` ou *Platt scaling* fica como trabalho futuro.

A explicabilidade das decisões é tratada como resultado próprio na Seção 5.6, com atribuição por decisão para os três modelos. Resta, porém, uma limitação residual quanto à profundidade dessa explicação nos *transformers*. Para o DomURL-BERT, a tokenização em subpalavras quebra a URL em fragmentos pequenos, o que torna parte da atribuição por *token* menos interpretável do que a do CatBoost. Técnicas adicionais como LIME, gradiente integrado e a análise sistemática de viés e justiça permanecem pendentes e ficam como trabalho futuro, conforme a Seção 5.10.

A análise qualitativa sistemática de erros (categorização manual de exemplos representativos de falsos positivos e de falsos negativos para entender padrões) não foi feita. Apenas observações pontuais sobre os falsos positivos residuais da cascata estão registradas na Seção 5.5.

A comparação formal entre variantes do DomURL-BERT cobriu apenas duas configurações (URL+WHOIS e multimodal completo). Variantes adicionais, por exemplo, URL apenas sem WHOIS ou subconjuntos parciais das *features* de DOM, não couberam no escopo experimental.

O conjunto BR de 660 838 URLs apresenta forte dominância da fonte `kmack_phishing_urls`, que responde por aproximadamente 99,3 % das amostras. As fontes de curadoria nacional e regional brasileira somam menos de 0,5 % do volume. Essa dominância delimita a generalização: o desempenho reportado é dominado pelo comportamento do modelo em URLs do `kmack`, e parte da discrepância entre a FPR no conjunto de teste fixo (15,64 % para o URL+WHOIS) e o comportamento sobre URLs brasileiras de alto perfil em `validacao_modelo.json` pode estar associada a esse fator. Este trabalho não reivindica que o conjunto `kmack_phishing_urls` represente o *phishing* brasileiro; o recorte nacional da solução é carregado pela curadoria da *whitelist* e da *blacklist* e pelos conjuntos de validação empírica, e não pelo *corpus* de treino do modelo principal. A redução dessa dominância de fonte fica registrada como trabalho futuro na Seção 5.10.

A rodada formal reportada na Tabela 10 apresenta uma contaminação parcial entre o conjunto de teste e o conjunto de treino do modelo principal, que precisa ser registrada como limitação porque afeta a leitura de uma das métricas. Por contaminação

treino-teste entende-se aqui a situação em que a mesma URL aparece tanto na lista usada para avaliar o modelo quanto entre as URLs com que ele aprendeu durante o ajuste; quando isso ocorre, parte do acerto observado não corresponde a generalização, e sim a reconhecimento de um endereço já visto. Uma medição reprodutível, conduzida pelo *script* `recalcular_fpr_disjunta.py` versionado no repositório, comparou as URLs legítimas da rodada formal com as partições de treino e validação do conjunto `kmack_phishing_urls`, a fonte que responde por aproximadamente 99 % das amostras de treino do modelo principal (parágrafo anterior). A comparação normaliza os endereços (caixa baixa, sem `www.` e sem barra final) antes do confronto. Das 1 013 URLs legítimas avaliadas, 306 (cerca de 30 %) constam dessas partições e foram, portanto, efetivamente vistas pelo modelo durante o ajuste; as 707 restantes são inéditas.

A contaminação concentra seu efeito nas URLs legítimas, e esse efeito sobre a FPR foi quantificado. Recalculada apenas sobre as 707 URLs legítimas disjuntas do treino, a FPR do modelo principal é de 8,35 %, ante os 6,52 % medidos sobre o conjunto completo de 1 013 legítimas; sobre as 306 URLs já vistas no ajuste, a FPR cai para 2,29 %, o que evidencia o efeito de reconhecimento. As URLs legítimas afetadas pertencem a domínios populares globais e brasileiros (entre outros, `playstation.com`, `roblox.com`, `unb.br` e `bancopan.com.br`), exatamente o tipo de endereço que aparece com frequência em listas de tráfego como Tranco e Majestic e que portanto tende a estar presente em qualquer conjunto público de URLs legítimas. O valor de 8,35 % é, assim, a estimativa mais fiel da FPR do modelo principal sobre URLs legítimas inteiramente novas, acima da Especificidade reportada na Tabela 10, que reflete em parte o efeito de reconhecimento. O recálculo recai sobre o lado legítimo porque é nele que a contaminação infla a Especificidade; o efeito sobre a Revocação é tratado na discussão do *trade-off*. O DomURL-BERT multimodal foi treinado sobre o *dataset* `GregaVrbancic`, que disponibiliza apenas *features* extraídas e não as URLs originais, o que impediu medição equivalente para esse modelo e adiciona uma limitação correlata. A seleção da lista da rodada formal não fez deduplicação URL a URL contra o conjunto de treino antes de fixar as 1 946 amostras; uma rodada com conjunto inteiramente disjunto e deduplicação explícita fica registrada como trabalho futuro na Seção 5.10.

A fração de tráfego que efetivamente atinge o modelo na zona ambígua não foi medida em produção. A inferência indireta a partir de `validacao_cascata.json` sugere que cerca de 2 das 101 URLs daquele conjunto pequeno tiveram veredicto alterado pelo CatBoost na cascata, o que indica fração baixa, mas o número exato em tráfego real

depende do comportamento de uso da extensão e fica como pendência metodológica.

A capacidade de carga da API foi medida sob concorrência por meio de um teste local. Cada *cache miss* no servidor aciona uma inferência da cascata em CPU, e a latência fim-a-fim reportada na Tabela 10 foi obtida em execução sequencial. Para caracterizar o comportamento sob requisições simultâneas, a API foi submetida a um teste de carga local, em contêiner Docker e com inferência em CPU, com 200 requisições por nível de concorrência ao *endpoint* `POST /predict`. A Tabela 11 reporta a vazão e os percentis de latência por nível de concorrência.

Tabela 11 – Carga da API por nível de concorrência (vazão e percentis de latência), com inferência da cascata em CPU.

Concorrência	**Vazão (req/s)**	**P50 (ms)**	**P95 (ms)**	**P99 (ms)**
1	20,1	44,6	67,3	147,3
2	24,1	79,6	97,5	280,7
4	24,4	154,8	259,3	374,8
8	23,2	308,2	545,9	778,1
16	25,1	611,0	986,5	1 299,0
32	23,7	1 258,5	1 875,7	2 124,0

Fonte: Autor (2026). Teste sobre a API em execução local (*Docker*, CPU), 200 requisições por nível.

A vazão satura em cerca de 24 requisições por segundo e não cresce com o aumento da concorrência, comportamento esperado de um serviço limitado por CPU: acima de cerca de duas requisições simultâneas, as novas requisições apenas se enfileiram e a latência cresce de forma aproximadamente linear. O P95 dobra em relação ao valor base (67,3 ms na concorrência 1) já a partir de quatro requisições simultâneas e alcança 1 875,7 ms na concorrência 32, sem que ocorram erros ou *timeouts* até esse nível. O escalonamento da solução depende, portanto, de réplicas horizontais (cerca de 24 requisições por segundo por réplica) ou de inferência em GPU com *batching*, e não do aumento de concorrência por instância. Um teste de carga mais extenso, com escalonamento horizontal e medição em produção, permanece como trabalho futuro nas Seções 5.10 e 6.5.

Por fim, o módulo de análise de *e-mail* recebeu apenas uma avaliação quantitativa parcial. O módulo reúsa um *checkpoint* DistilBERT pré-treinado de terceiros (Seção 4.11), sem *fine-tuning* próprio. Sobre uma amostra estratificada de 4 000 *e-mails* do conjunto público `zefang-liu/phishing-email-dataset` (2 000 de *phishing*

e 2 000 legítimos, em inglês), a estratégia de decisão usada pela API obteve F1 de 0,9881, Revocação de 0,9965 e FPR de 0,0205. Dois fatores restringem a leitura desse resultado. Primeiro, a avaliação cobre apenas texto em inglês; não há medição sobre *e-mails* em português brasileiro, e o erro introduzido pela etapa de tradução automática do português para o inglês permanece não quantificado. Segundo, o *checkpoint* de terceiros pode ter sido treinado sobre fontes que incluem esse conjunto, o que configura possível contaminação treino-teste e recomenda tratar as métricas como teto otimista, e não como desempenho de campo. Por esses motivos, o módulo permanece tratado como prova de conceito de integração, e a avaliação sobre *e-mails* reais em português brasileiro fica como trabalho futuro.

5.8 Discussão dos resultados

Os números reportados nas seções anteriores permitem três leituras conjuntas, apresentadas a seguir.

Antes das três leituras, convém fixar o sentido operacional de "tempo real" adotado neste trabalho. O termo designa latência interativa, no sentido de Nielsen (NIELSEN, 1993). O alvo é manter a resposta abaixo do limiar de cerca de 1 s, que preserva a sensação de fluxo contínuo de interação, e não oferecer garantias de sistemas de tempo real estritos. As latências fim-a-fim medidas na rodada formal (P50 de 34,97 ms e P95 de 45,73 ms para o modelo principal, Tabela 10) situam-se cerca de uma ordem de grandeza abaixo desse limiar, e o caminho rotineiro resolvido pelas três camadas locais responde em tempo $O(1)$ sem chamada à rede. Nesse sentido operacional, a meta de tempo real foi atendida.

A primeira leitura diz respeito às metas declaradas na fase de proposta. A Seção 1.1 declarou TPR próxima de 90 % e FPR menor ou igual a 0,5 % como referência de projeto. A Revocação do modelo principal adotado (0,9498 no conjunto de teste fixo) superou a meta inicial de TPR; a Revocação observada no DomURL-BERT puro em validação empírica (100 % sobre 62 e sobre 101 URLs testadas, conforme Tabelas 8 e 9) reforçou esse comportamento. A Especificidade do modelo principal, por outro lado, ficou em 0,8436, equivalente a FPR aproximada de 15,64 % em conjunto de teste fixo, valor cerca de trinta vezes maior do que a meta inicial. A meta de FPR não foi atingida por nenhum dos modelos avaliados isoladamente: o melhor valor observado no conjunto de teste fixo foi de cerca de 9 % pelo DomURL-BERT multimodal. A meta original de FPR é

tratada neste trabalho como meta-alvo declarada na fase de proposta, e não como resultado atingido. O patamar de meio ponto percentual não é, porém, arbitrário: ele corresponde ao regime de operação que a literatura de detecção de *phishing* de URL considera necessário, dado que cada falso positivo recai sobre tráfego legítimo em larga escala. O URLTran, classificador baseado em *transformer* desenvolvido em ambiente industrial, opera nesse regime e reporta Revocação de 86,80 % a uma FPR de 0,01 % (MANERIKER *et al.*, 2021). Esse ponto de operação é alcançado, contudo, com volume de dados de treino de escala industrial e ao custo de uma Revocação inferior à obtida neste trabalho. A meta de 0,5 % refletia, portanto, o regime correto da literatura, mas a sua obtenção dependia de recursos de dados e de calibração fora do alcance desta prova de conceito, o que remete à moldura de recursos da Seção 5.7. A redução adicional da FPR para aproximar-se da meta de proposta fica registrada como trabalho futuro na Seção 5.10.

A segunda leitura diz respeito ao *trade-off* entre Revocação e Especificidade no modelo principal. Limiares mais conservadores reduziriam a FPR, ao custo de reduzir a Revocação; o oposto também é verdadeiro. A escolha de limiares fixos (0,15, 0,85 e 0,65) usada na cascata foi feita empiricamente durante o desenvolvimento e permanece pendente de calibração formal; a abordagem de banda de confiança que difere a decisão a um segundo estágio segue a estratégia clássica de *reject option* de Chow (1970), em tradução livre "opção de não decidir": o classificador abstém-se na faixa em que tem menos confiança e remete a decisão a outro estágio, sem que este trabalho proponha justificativa específica para os valores adotados. Adicionalmente, a arquitetura final mitiga arquiteturalmente o impacto da FPR sobre o usuário: o número de 15,64 % aplica-se à fração de tráfego que entra na zona em que o modelo precisa decidir, e não ao tráfego total da extensão. URLs cobertas pela *whitelist*, pela *blacklist* ou pelo *cache* são resolvidas localmente e não passam pelo modelo. Aproximação qualitativa indica fração baixa de tráfego efetivamente roteado para o modelo (cerca de 2 das 101 URLs do `validacao_cascata.json` tiveram veredicto refinado pelo CatBoost), mas o número exato em produção não foi medido.

Uma análise de sensibilidade do limiar de decisão do modelo principal quantifica esse *trade-off*. A partir das probabilidades por URL registradas na rodada formal (1 946 URLs), recalculou-se a Revocação, a FPR e o F1 para diferentes valores de limiar, conforme a Tabela 12. O comportamento de FPR e Revocação é monótono: elevar o limiar reduz ambas simultaneamente. O F1 permanece estável (entre 0,910 e 0,917) ao longo de toda a faixa, o que indica que o ponto de operação de 0,65 é estável frente a

variações do limiar. A análise também reforça a leitura da meta de FPR discutida nesta seção: o patamar de 0,5 % é inatingível por ajuste de limiar neste modelo, pois mesmo a uma FPR de 0,89 % (limiar 0,998) a Revocação cai para 0,7042. A sensibilidade aos demais parâmetros da cascata, os pesos $0{,}6$ e $0{,}4$ e os limites $0{,}15$ e $0{,}85$ da zona ambígua, foi avaliada à parte, com a captura dos escores do modelo principal e do CatBoost por URL, e está reportada na Tabela 13.

Tabela 12 – Sensibilidade da Revocação, da FPR e do F1 ao limiar de decisão do modelo principal (DomURL-BERT URL+WHOIS) na rodada formal de 1 946 URLs.

Limiar	**Revocação**	**FPR**	**F1**	**Observação**
0,50	0,9132	0,0731	0,9166	
0,65	0,9035	0,0652	0,9153	ponto de operação
0,75	0,8939	0,0563	0,9145	
0,85	0,8864	0,0454	0,9158	
0,90	0,8757	0,0424	0,9113	
0,95	0,8671	0,0355	0,9100	

Fonte: Autor (2026). Métricas recalculadas sobre as probabilidades por URL da rodada formal (`validacao_formal.py`).

A captura dos escores do modelo principal e do CatBoost para as 1 946 URLs permitiu variar, de forma reprodutível, os pesos da combinação e a largura da zona ambígua. A Tabela 13 reúne os cenários, e dela emergem três leituras. Primeiro, a cascata melhora sobre o modelo principal isolado: o CatBoost, acionado na zona ambígua, reduz a FPR de 13,33 % para 11,55 % na configuração de produção e eleva o MCC. Segundo, a escolha do peso é robusta: variar o peso do modelo principal entre 0,5 e 0,7 quase não altera as métricas (F1 entre 0,9170 e 0,9182), o que mostra que o par 0,6 e 0,4 adotado não é um ponto frágil. Terceiro, a largura da zona pesa mais do que os pesos: estreitar a zona para a faixa de 0,4 a 0,6 praticamente anula o efeito da cascata, pois as métricas colapsam para as do modelo isolado, ao passo que a zona ampla de 0,15 a 0,85 da configuração de produção é a melhor entre as testadas. As taxas absolutas desta tabela diferem das da rodada formal (Tabela 10) porque a captura usou condições distintas, com valores *default* para as *features* de WHOIS e TLS; a leitura válida é a comparação relativa entre os cenários, e não o valor absoluto da FPR.

Tabela 13 – Sensibilidade da cascata aos pesos da combinação e à largura da zona ambígua, sobre as 1 946 URLs da rodada formal.

Cenário	**F1**	**FPR**	**MCC**
Modelo principal isolado	0,9119	0,1333	0,8267
Cascata, peso 0,5 (zona 0,15–0,85)	0,9174	0,1145	0,8381
Cascata, peso 0,6 (produção)	0,9170	0,1155	0,8371
Cascata, peso 0,7 (zona 0,15–0,85)	0,9182	0,1165	0,8395
Cascata, peso 0,6, zona 0,3–0,7	0,9150	0,1254	0,8331
Cascata, peso 0,6, zona 0,4–0,6	0,9124	0,1333	0,8278

Fonte: Autor (2026). Escores por URL capturados por `q4_cascata_sweep.py`; *features* de WHOIS e TLS em valor *default*. As taxas não são comparáveis às da Tabela 10.

O outro lado do mesmo *trade-off* merece tratamento explícito, pois o capítulo até aqui privilegiou a Especificidade. Em detecção de *phishing*, o falso negativo é o erro de custo mais grave: corresponde a um ataque que passa sem alerta, ao passo que o falso positivo apenas degrada a experiência do usuário. O ponto de operação adotado favoreceu FPR baixa e UX, escolha que cobra preço em Revocação. Na rodada formal da Tabela 10, o modelo principal registrou 90 falsos negativos em 933 URLs de *phishing*, o que corresponde à Revocação de 0,9035; cerca de 9,6 % do *phishing* que chega ao modelo não foi sinalizado. O valor representa queda em relação à Revocação de 0,9498 obtida no conjunto de teste fixo. A diferença decorre das condições de medição: o conjunto de teste fixo é uma partição do mesmo *dataset* usado no treino, ao passo que a rodada formal emprega uma lista pré-registrada de 1 946 URLs reais, distribuição mais difícil e mais próxima do uso em produção. A Revocação de 0,9035 é, por isso, a estimativa mais conservadora e mais realista do comportamento operacional do modelo, e expõe uma limitação de segurança concreta. A *blacklist* local de aproximadamente 774 000 URLs mitiga apenas em parte esse risco, pois resolve em $O(1)$ o *phishing* já conhecido antes que a requisição alcance o modelo; o ônus de falso negativo do classificador recai, então, sobre a cauda de URLs novas ou desconhecidas. Para campanhas de primeira hora, fora das listas, o modelo é a última linha de defesa, e um falso negativo nesse ponto deixa o usuário desprotegido; a política de *fail-open*, que não bloqueia a navegação quando a API está indisponível, soma-se a essa exposição. Elevar a Revocação sem inflar a FPR depende da calibração formal de limiares, registrada como trabalho futuro na Seção 5.10.

A terceira leitura diz respeito à evidência de que mais informação na entrada nem sempre melhora o desempenho operacional. As Tabelas 6 e 7 mostram que o

DomURL-BERT multimodal foi marginalmente superior ao URL+WHOIS em todas as métricas agregadas do conjunto de teste fixo. Essa comparação de teste fixo deve ser lida com cautela: os modos multimodais incorporam no treino *features* do conjunto `GregaVrbancic`, de modo que a diferença entre os modos URL apenas e multimodal mistura o efeito da modalidade com o do *corpus* de treino, e os dois modos não partilham exatamente a mesma divisão de teste. A comparação sob conjunto de teste idêntico para o URL+WHOIS e o multimodal é a da rodada formal, sobre as mesmas 1 946 URLs reais. A validação empírica complementar (Tabela 8 e a observação qualitativa registrada na Seção 5.5 para o multimodal) indicou comportamento operacional inferior do multimodal sobre URLs reais brasileiras. A escolha do URL+WHOIS como modelo principal foi, portanto, uma decisão de projeto que privilegiou comportamento operacional observado em UX, e não uma decisão por métrica isolada de *benchmark*. A rodada formal da Tabela 10 confirma essa leitura com números diretos: sobre as 1 946 URLs reais pré-registradas, o multimodal não superou o URL+WHOIS em nenhuma das métricas agregadas, com Revocação de 0,8917 contra 0,9035, F1 de 0,9078 contra 0,9153 e FPR de 0,0671 contra 0,0652. A vantagem marginal do multimodal observada no conjunto de teste fixo não se traduziu, portanto, em vantagem sobre URLs reais, o que sustenta a escolha do URL+WHOIS como modelo principal. A investigação da causa subjacente dessa divergência (análise de *loss* de treino, importância das *features* de DOM e exemplos contrastantes) fica como trabalho futuro.

Em relação ao estado da arte, o desenho da solução tem cinco contribuições reportáveis. A primeira é a arquitetura cliente-servidor que separa explicitamente decisão local em $O(1)$ por listas e *cache* de inferência por modelo no servidor; o desenho não é inédito na literatura, mas a evidência empírica que o sustenta é própria deste trabalho. Um classificador treinado sobre um *corpus* dominado por uma fonte única (Seção 5.7) rotula como *phishing* domínios brasileiros legítimos de alto perfil em proporção elevada; essa lacuna de generalização para o nicho brasileiro é justamente o que torna a *whitelist* local uma camada de proteção necessária, e não mera otimização. A segunda é a comparação empírica entre GBM, DomURL-BERT URL+WHOIS, DomURL-BERT multimodal e SecureBERT (URL apenas e multimodal) sobre o conjunto BR de 660 838 URLs, no mesmo ambiente de *benchmark* (Tabela 7); a comparação sob conjunto de teste idêntico entre o URL+WHOIS e o multimodal, livre do efeito do *corpus* de treino, é a da rodada formal (Tabela 10). A terceira é a evidência empírica de que mais *features* de entrada não se traduziu em melhor comportamento operacional, no sentido descrito na

terceira leitura desta seção. A quarta é a entrega de um pacote reprodutível: Docker Compose para a API e `npm install && npm run build` para a extensão, com especificação completa do ambiente. A reprodutibilidade declarada estende-se a três eixos complementares: (i) o pacote operacional já descrito no parágrafo anterior, que cobre a instalação automatizada da API via Docker Compose e a construção da extensão via `npm`; (ii) o ajuste fino (*fine-tuning*) dos modelos, com *notebooks* e configurações de treino publicados no repositório; e (iii) os testes que sustentam a Tabela 10, com lista pré-registrada de 1 946 URLs, *seed* fixa igual a 42 (número inicial do gerador aleatório, fixado para que cada execução produza o mesmo resultado) e *scripts* de execução versionados. A quinta é a direção de produto *enterprise*: *dashboard web*, *multi-tenancy* com persistência por organização, alertas via *webhook* e *Simple Mail Transfer Protocol* (SMTP) e *deploy* gerenciado via `chrome.storage.managed`, descritos no Capítulo 4. A direção *enterprise* não corrige as limitações de modelo aqui reportadas, mas oferece os mecanismos operacionais (auditoria, governança, roteamento por organização) que produtos comerciais demandam, o que diferencia o trabalho de protótipos puramente acadêmicos da literatura comparada (LI *et al.*, 2024; KUMAR; GUPTA *et al.*, 2022; DO *et al.*, 2022).

O resultado conjunto é classificado neste trabalho como aceitável dentro das condições avaliadas. Não corresponde a "supera o estado da arte" nem a "solução definitiva"; corresponde a um protótipo funcional com resultado aceitável em métricas operacionais relevantes, com limitações reconhecidas em FPR, em recursos de avaliação e em escopo experimental. O capítulo seguinte (Conclusões) sintetiza essas leituras e recapitula trabalhos futuros derivados delas.

5.9 Posicionamento frente ao estado da prática

As seções anteriores compararam o PampAI Security com modelos da literatura acadêmica. Esta seção responde a uma pergunta complementar e legítima de avaliação: por que uma organização ou um usuário adotaria a solução proposta em vez das ferramentas de detecção de *phishing* já consolidadas e de uso disseminado? Este trabalho é um protótipo de prova de conceito acadêmico, e não um produto final; portanto, o objetivo desta comparação não é demonstrar superioridade sobre serviços maduros de mercado, mas delimitar o nicho em que a arquitetura proposta oferece diferencial defensável.

O estado da prática em detecção de *phishing* na navegação é dominado por serviços de reputação consolidados há quase duas décadas. O Google Safe Browsing (Google, 2024), ativo desde 2007 e embutido gratuitamente em Chrome, Firefox e Safari, e o Microsoft Defender SmartScreen (Microsoft, 2025), integrado ao Windows e ao Edge, protegem a maior parte dos usuários da *web* por meio de listas de reputação e heurísticas mantidas em escala industrial. No segmento corporativo, *Secure Web Gateways* como o Cloudflare Gateway (Cloudflare, 2026) e o Cisco Umbrella (Cisco, 2026) aplicam filtragem em nuvem nas camadas de DNS e HTTP. A Tabela 14 resume o posicionamento da solução proposta diante desses representantes do estado da prática, em eixos relevantes para o recorte deste trabalho. A comparação é qualitativa e apoia-se na documentação pública de cada serviço (Google, 2024; Microsoft, 2025; Cloudflare, 2026; Cisco, 2026): as posições nos eixos de cobertura e maturidade refletem a escala de operação documentada desses serviços, e não uma medição direta conduzida neste trabalho. Este trabalho não mede, em particular, o percentual de *phishing* brasileiro coberto por qualquer um desses serviços; tal medição exigiria telemetria e acesso a dados fora do seu alcance e fica registrada como direção futura. As posições atribuídas devem ser lidas, portanto, como classificação relativa fundamentada na documentação dos fornecedores, e não como resultado de *benchmark* comparativo.

Tabela 14 – Posicionamento do PampAI Security frente a representantes do estado da prática.

Eixo	**Google Safe Browsing**	**Defender SmartScreen**	**SWG corporativo**	**PampAI Security (este trabalho)**
Método principal	Listas de reputação e heurística, com ML	Reputação de URL em nuvem	ML com filtragem de DNS e HTTP	Listas locais em $O(1)$ e cascata BERT+CatBoost
Tráfego não resolvido localmente	Consultado no servidor sob suspeita	URL enviada ao servidor	Inspeção na nuvem do fornecedor	Inferência em servidor auto-hospedável
Soberania de dados (auto-hospedagem)	Não	Não	Não	Sim
Recorte para *phishing* brasileiro	Não específico	Não específico	Não específico	Sim
Explicabilidade ao cliente	Limitada	Limitada	Limitada	Por decisão (SHAP nos três modelos; atenção nos *transformers*)
Custo de adoção	Gratuito (embutido)	Gratuito (embutido)	Licença por usuário	Auto-hospedável
Cobertura global e tempo de proteção	Muito alta	Muito alta	Alta	Limitada (protótipo)
Maturidade	Consolidado (desde 2007)	Consolidado	Consolidado	Prova de conceito

Fonte: Autor (2026), com base em (Google, 2024; Microsoft, 2025; Cloudflare, 2026; Cisco, 2026). SWG: *Secure Web Gateway*. O estado de explicabilidade do PampAI Security está detalhado na Seção 5.6.

A leitura da Tabela 14 separa dois grupos de eixos. Nos eixos de cobertura e maturidade, o PampAI Security não compete. O Google Safe Browsing e o SmartScreen têm telemetria de bilhões de eventos, equipes dedicadas e cobertura global, com tempo de proteção a campanhas de primeira hora que um protótipo acadêmico não alcança. A manutenção contínua das listas (cerca de 32 000 domínios na *whitelist* e 774 000 na *blacklist*) é, na solução proposta, um custo operacional ainda não industrializado. O argumento de processamento local também não é inédito. O Google Safe Browsing, em

seu modo padrão, mantém prefixos de *hash* no cliente e só consulta o servidor diante de suspeita; desde 2024, emprega *Oblivious HTTP* para desacoplar o endereço do cliente do conteúdo consultado (Google, 2024).

Nos eixos de soberania de dados, recorte regional e governança, contudo, a arquitetura proposta apresenta diferencial concreto. Soberania de dados designa, neste trabalho, a capacidade de a organização decidir onde e por quem seus dados são processados. Primeiro, a soberania: o servidor de inferência é auto-hospedável, o que permite a uma instituição processar o tráfego ambíguo em infraestrutura sob seu próprio controle, por exemplo um órgão do setor público brasileiro sujeito à Lei Geral de Proteção de Dados Pessoais (LGPD), em vez de enviá-lo a um terceiro como Google ou Microsoft. O SmartScreen, em particular, envia a URL completa ao serviço de reputação quando o endereço não consta nas listas locais (Microsoft, 2025). A diferença não está no volume de dados que trafega: tanto o PampAI Security quanto o Google Safe Browsing e o SmartScreen entregam a URL a um servidor de inferência quando as camadas locais não resolvem. O diferencial é de governança: na arquitetura proposta, esse servidor pode ser hospedado pela própria instituição que opera a extensão, sob o seu marco legal e a sua política de retenção, em vez de pertencer a um terceiro fora da jurisdição do operador. Segundo, o recorte brasileiro: os dados e as listas foram consolidados com foco em URLs e marcas nacionais (bancos, varejo, governo), nicho que os *feeds* globais não otimizam. Terceiro, o custo e a transparência: a solução é auto-hospedável, com composição de cascata transparente (pesos visíveis $0{,}6$ e $0{,}4$) e explicabilidade por decisão dos três modelos via SHAP e mapas de atenção (Seção 5.6), ao passo que os *Secure Web Gateways* corporativos são licenciados por usuário (Cloudflare, 2026; Cisco, 2026) e operam como caixas-pretas para o cliente.

Conforme introduzido na motivação do trabalho e refletido nos objetivos de minimização de dados e auto-hospedagem do servidor de inferência, o posicionamento defensável do PampAI Security não é o de substituto do Google Safe Browsing, mas o de complemento com soberania de dados: uma arquitetura *privacy-first* auto-hospedável, com recorte brasileiro e explicabilidade por decisão dos três modelos (SHAP para todos e mapas de atenção para os *transformers*), voltada a organizações e usuários que preferem não delegar a um terceiro a inspeção do seu tráfego de navegação. Como prova de conceito, o diferencial demonstrado é arquitetural; a evolução para competir em cobertura e maturidade depende das direções registradas como trabalho futuro, em particular o treinamento dos modelos com infraestrutura mais robusta (Seção 5.10) e o

amadurecimento da camada *enterprise* (Capítulo 6).

5.10 Trabalhos futuros relacionados aos resultados

Esta seção lista os trabalhos futuros que derivam diretamente dos resultados apresentados neste capítulo. A lista de trabalhos futuros do projeto como um todo, que inclui a direção de produto, está no Capítulo 6.

Em termos de *tuning* e calibração do modelo principal, três frentes ficam abertas. A primeira é a redução da FPR para aproximar-se da meta inicial declarada na proposta, por meio de calibração formal de probabilidades, ajuste sistemático de limiares e exploração de *ensembles* com sinais adicionais, isto é, combinações de vários modelos cujas decisões são fundidas em uma decisão única. A segunda é uma rodada formal sobre conjunto inteiramente disjunto do treino, com deduplicação explícita, que isolaria o efeito de reconhecimento quantificado na contaminação treino-teste (Seção 5.7) e fixaria a FPR sobre URLs inteiramente novas. A terceira é a medição da fração de tráfego que efetivamente atinge a zona ambígua em produção, por meio de instrumentação do *endpoint* `POST /predict` para registrar o `cascade_path` por amostra; sem essa medição, o argumento de que a FPR efetiva da extensão é menor do que a FPR do modelo permanece qualitativo.

Em termos de avaliação, três frentes ficam abertas. A primeira é uma sessão controlada de navegação com participantes, complementar à rodada formal automática já conduzida (Tabela 10), que acrescentaria avaliação de uso real à medição de TPR, FPR e latência fim-a-fim. A segunda é a análise qualitativa sistemática de erros, com categorização manual de 50 a 100 amostras representativas de falsos positivos e falsos negativos, para identificar padrões reaproveitáveis em iterações de modelo. A terceira é a investigação da causa subjacente do comportamento operacional inferior do multimodal, com análise de *loss* de treino contra validação, importância das *features* de DOM e exemplos contrastantes, uma forma de avançar de "observação empírica" para "explicação técnica".

Em termos de explicabilidade, a atribuição por decisão já entregue (SHAP para os três modelos e mapas de atenção para os *transformers*, Seção 5.6) abre três frentes de aprofundamento. A primeira é a adoção de técnicas complementares, em particular LIME e gradiente integrado, que poderiam atenuar o ruído observado no SHAP por *token* dos *transformers*. A segunda é a calibração das explicações, de modo a tornar comparáveis

as magnitudes de atribuição entre modelos com níveis distintos de confiança. A terceira é a análise sistemática de viés e justiça apoiada nessas atribuições, ainda pendente neste TCC por cruzamento direto com a moldura de recursos descrita na Seção 5.7.

Em termos de infraestrutura experimental, a repetição dos experimentos com infraestrutura mais robusta (por exemplo, FabLab da Unipampa, GPUs dedicadas) permitiria múltiplas *seeds* para análise de variância, busca de hiperparâmetros mais ampla, modelos maiores (DeBERTa-v3-large, modelos de linguagem grandes) e *datasets* ampliados. A ampliação da diversidade do conjunto BR (em particular a redução da dominância de `kmack_phishing_urls`) é uma frente complementar que reduziria os vieses de fonte registrados na Seção 5.7.

Em termos de produto, o Capítulo 6 retoma a direção *enterprise*: ampliação do *dashboard* com novas métricas; *onboarding self-service* para novas organizações; a ampliação do teste de carga já conduzido para perfil realista e escalonamento horizontal; ampliação de idiomas no *pipeline* de *e-mail*; integração com *webmails* além de Gmail e Outlook; políticas de retenção de dados; e *roadmap* comercial.

6 CONCLUSÕES

Este capítulo encerra a monografia e consolida o trabalho desenvolvido. A Seção 6.1 apresenta uma síntese da execução, da arquitetura final adotada e das métricas-chave do modelo principal. A Seção 6.2 retoma os objetivos específicos declarados na Seção 1.2 e comenta o atingimento de cada um. A Seção 6.3 destaca as contribuições reportáveis do trabalho. A Seção 6.4 consolida as limitações reconhecidas, em diálogo com a discussão experimental do Capítulo 5. A Seção 6.5 fecha com os trabalhos futuros derivados dos resultados e com a direção de produto que o trabalho aponta.

6.1 Síntese do trabalho desenvolvido

Este trabalho desenvolveu o PampAI Security, uma solução para detecção de *phishing* em tempo real no navegador, com arquitetura cliente-servidor em que a extensão decide localmente em três camadas (*whitelist*, *blacklist* e *cache* de resultados) e a inferência por modelo de aprendizado de máquina reside na *Application Programming Interface* (API). O modelo principal adotado foi o DomURL-BERT *fine-tuned* sobre URL e campos de *Whois Information* (WHOIS) (MAHDAOUY *et al.*, 2024); quando a probabilidade do DomURL-BERT cai na zona ambígua entre 0,15 e 0,85, é acionada uma cascata *server-side* que combina o DomURL-BERT com um classificador CatBoost sobre 24 *features* enriquecidas com WHOIS, *Domain Name System* (DNS), *Transport Layer Security* (TLS) e dados de redirecionamento (PROKHORENKOVA *et al.*, 2018). A solução inclui ainda um *pipeline* de análise de *e-mail* para Gmail e Outlook, baseado em DistilBERT e em tradução automática do português para o inglês com MarianMT (SANH *et al.*, 2019), e uma plataforma com *dashboard web*, *multi-tenancy*, alertas via *webhook* e SMTP e *deploy* gerenciado via `chrome.storage.managed`.

A arquitetura final é distinta da prevista na proposta original. A proposta inicial previa modelo leve embarcado no cliente com *fallback* robusto no servidor; durante a execução, evidência empírica acumulada nos experimentos com modelos locais (em particular o *Histogram-based Gradient Boosting Machine* (GBM) treinado sobre o conjunto brasileiro consolidado de mais de 600 mil URLs balanceadas 50 %/50 % entre legítimas e *phishing*) indicou que o modelo embarcado no cliente disparava chamadas à API "de *fallback*" em rotina cotidiana e degradava a experiência do

usuário. A arquitetura foi reformulada como decisão de engenharia justificada por dado (MAHDAOUY *et al.*, 2024; PROKHORENKOVA *et al.*, 2018; LI *et al.*, 2024), não como recuo. O detalhamento desta decisão está em Desenvolvimento (Capítulo 4) e em Resultados (Capítulo 5).

O modelo principal apresentou, em conjunto de teste fixo, F1 próximo a 0,90, *Area Under the Receiver Operating Characteristic Curve* (AUC-ROC) próxima a 0,96 e Revocação próxima a 0,95, com Taxa de Falsos Positivos (*False Positive Rate* – FPR) próxima a 15,6 %. A meta inicial de FPR declarada na fase de proposta, FPR menor ou igual a 0,5 %, não foi atingida por nenhum dos modelos avaliados isoladamente. O melhor valor observado em conjunto de teste fixo foi de aproximadamente 9 % pelo DomURL-BERT multimodal, que não foi adotado como modelo principal por motivos discutidos no Capítulo 5. O resultado conjunto é classificado neste trabalho como aceitável dentro das condições avaliadas, com limitações reconhecidas e trabalhos futuros derivados diretamente delas.

6.2 Objetivos específicos atingidos

Esta seção retoma, de forma direta, os sete objetivos específicos declarados na Seção 1.2 e comenta o atingimento de cada um, com remissão aos capítulos e seções que sustentam empiricamente cada afirmação. Cada item recebe um status entre **atingido** e **atingido parcialmente**; este último é usado quando o objetivo foi executado, mas alguma cláusula declarada na proposta não foi cumprida na íntegra.

O1 – Mapear bases rotuladas: atingido. Foi consolidado o conjunto brasileiro com mais de 660 mil URLs balanceadas a partir de fontes públicas, descrito na Seção 3.2 do Capítulo 3. O mapeamento incluiu bases rotuladas para URL e o conjunto utilizado pelo *pipeline* de análise de *e-mail*.

O2 – Identificar sinais tratáveis no cliente: atingido. Foram definidas e implementadas as três camadas locais (*whitelist*, *blacklist* e *cache* de resultados), descritas nas Seções 4.4, 4.5 e 4.6 do Capítulo 4. Os indicadores de URL utilizados pela API estão descritos na Seção 3.3.

O3 – Investigar abordagens de aprendizado de máquina: atingido. A comparação empírica entre o GBM, o DomURL-BERT URL+WHOIS, o DomURL-BERT

multimodal e o SecureBERT em duas configurações, com metodologia descrita na Seção 3.4, está consolidada na Seção 5.5 do Capítulo 5 e na Tabela 7, sobre o mesmo conjunto de teste e o mesmo ambiente de *benchmark*.

O4 – Projetar e implementar a arquitetura: atingido. A arquitetura cliente-servidor da extensão MV3 e do serviço de análise está descrita no Capítulo 4, com a visão geral na Seção 4.1 e diagrama na Figura 3. A integração entre coleta, decisão local em três camadas, consulta à API e alerta ao usuário cobre tanto navegação *web* (Seção 4.3) quanto *e-mail* (Seção 4.11).

O5 – Aplicar protocolo de avaliação: atingido parcialmente. O protocolo de avaliação foi definido na Seção 3.5 e aplicado nos resultados experimentais consolidados no Capítulo 5, com métricas de acurácia, F1, Revocação, AUC-ROC e FPR sobre conjunto de teste fixo. A validação empírica foi conduzida em modo qualitativo, com os artefatos `validacao_modelo.json` e `validacao_cascata.json` (Tabelas 8 e 9). O atingimento é classificado como parcial porque a meta de FPR menor ou igual a 0,5 % declarada na fase de proposta não foi atingida por nenhum dos modelos avaliados isoladamente, conforme registrado nas Seções 6.1 e 6.4.

O6 – Diretrizes de privacidade e segregação por organização: atingido parcialmente. A segregação de eventos por organização contratante foi implementada por meio de *multi-tenancy* em PostgreSQL, autenticação por *API Key* por organização, *dashboard web* e *deploy* gerenciado via `chrome.storage.managed`, descritos na Seção 4.12. O atingimento é classificado como parcial porque as diretrizes formais de minimização de dados e a política explícita de retenção em PostgreSQL alinhada à Lei Geral de Proteção de Dados Pessoais (LGPD), assim como a comunicação ao usuário final do motivo da classificação, ficam apontadas como trabalhos futuros na Seção 6.5.

O7 – Pacote reprodutível por meio de *containers*: atingido. O pacote reprodutível é descrito na Seção 3.6 (procedimento metodológico) e sustentado pela orquestração em *Docker Compose* da API, pela construção da extensão por `npm` e pelos modelos versionados, detalhados no Capítulo 4. Os artefatos são suficientes para replicar a execução do serviço e os resultados obtidos.

Em síntese, dos sete objetivos específicos, cinco foram atingidos integralmente

(O1, O2, O3, O4 e O7) e dois foram atingidos parcialmente (O5 e O6). Em O5, o protocolo de avaliação foi aplicado, mas a meta numérica de FPR menor ou igual a 0,5 % declarada na proposta não foi cumprida. Em O6, a moldura operacional de *enterprise* foi entregue, mas as diretrizes formais de minimização e retenção de dados, e a explicabilidade ao usuário final, ficam como continuidade explicitada na Seção 6.5.

6.3 Principais contribuições

Este trabalho oferece seis contribuições reportáveis, listadas a seguir sem inflar mérito além do sustentável pelos experimentos relatados.

A primeira contribuição é a **arquitetura cliente-servidor** com decisão local em $O(1)$ por três camadas (*whitelist*, *blacklist* e *cache*) e inferência por modelo de aprendizado de máquina exclusivamente *server-side* em cascata DomURL-BERT seguida de CatBoost. O desenho não é inédito na literatura (LI *et al.*, 2024; KUMAR; GUPTA *et al.*, 2022); a contribuição está na **evidência empírica de que um classificador treinado sobre um *corpus* dominado por uma fonte única não generaliza para domínios brasileiros legítimos de alto perfil**, comportamento de FPR alta que documenta o papel arquitetural das listas locais como camada de proteção, e não apenas como otimização de latência.

O modelo de comunicação dessa arquitetura merece esclarecimento explícito. O desenho adotado é centralizado, do tipo cliente-servidor, e não *peer-to-peer* (P2P): a inferência por modelo ocorre exclusivamente no servidor, e as três camadas locais (*whitelist*, *blacklist* e *cache*) operam *offline* no cliente, sem comunicação direta entre extensões de usuários distintos. A opção pelo modelo centralizado decorre da simplicidade de governança, da latência controlada por um único ponto de decisão e da possibilidade de auditoria centralizada dos eventos, requisitos alinhados à direção *enterprise* descrita na quarta contribuição.

A segunda contribuição é a **comparação empírica** entre o GBM, o DomURL-BERT URL+WHOIS adotado, o DomURL-BERT multimodal não adotado e o SecureBERT em modos URL apenas e multimodal, sobre o conjunto BR de 660 838 URLs balanceadas (AGHAEI *et al.*, 2022), com tabela consolidada que respeita o mesmo conjunto de teste e o mesmo ambiente de *benchmark* (Tabela 7 do Capítulo 5).

A terceira contribuição é a **demonstração empírica** de que mais *features* de entrada nem sempre melhoram o desempenho operacional de modelos *transformer*

especializados em URL. O DomURL-BERT multimodal apresentou métricas marginalmente superiores em conjunto de teste fixo, mas comportamento operacional inferior em validação empírica sobre URLs reais brasileiras. A divergência entre os dois sinais é reportável em si e está consolidada em Resultados (Capítulo 5).

A quarta contribuição é a **direção de produto *enterprise***. Além do protótipo acadêmico, o trabalho entrega *dashboard web* servido pela própria API, *multi-tenancy* com persistência por organização em PostgreSQL, autenticação por chave de API (*API Key*) por organização, alertas via *webhook* e SMTP, e *deploy* gerenciado via `chrome.storage.managed` para distribuição corporativa. Esses mecanismos operacionais (auditoria, governança, roteamento por organização) posicionam a solução em direção a um produto comercializável para empresas e a diferenciam de protótipos puramente acadêmicos da literatura comparada (LI *et al.*, 2024; DO *et al.*, 2022; ZIENI; MASSARI; CALZAROSSA, 2023). A direção *enterprise* não corrige as limitações de modelo reportadas, mas oferece a moldura operacional que produtos comerciais demandam.

A quinta contribuição é a **validação empírica consolidada** em dois artefatos do projeto: `validacao_modelo.json` (62 URLs, DomURL-BERT URL+WHOIS sem proteção das listas) e `validacao_cascata.json` (101 URLs, comparação entre DomURL-BERT puro, CatBoost puro e cascata). Esses artefatos sustentam empiricamente, com observação direta sobre URLs reais brasileiras de alto perfil, a presença obrigatória das três camadas locais na arquitetura final (com 36 falsos positivos em 36 URLs legítimas testadas pelo modelo sem proteção das listas) e o ganho marginal da cascata em relação ao DomURL-BERT puro (redução de 6 para 4 falsos positivos com Revocação preservada em 100 %).

A sexta contribuição é o **pacote reprodutível**: *Docker Compose* para a API (`docker compose up -d` em `phishing-api/`), `npm install && npm run build` para a extensão (em `extensao-phishing/`) e modelos versionados como *release* Git. A reprodutibilidade do ambiente é parte do entregue, não apenas do código.

Sobre essas contribuições assenta-se o posicionamento defensável do trabalho frente ao estado da prática (Seção 5.9): o PampAI Security não se apresenta como substituto do Google Safe Browsing ou do Microsoft Defender SmartScreen, mas como complemento com **soberania de dados**. Por ser auto-hospedável, o servidor de inferência permite a uma organização processar o tráfego ambíguo sob o seu próprio marco legal,

como a Lei Geral de Proteção de Dados Pessoais (LGPD), em vez de delegá-lo a um terceiro fora da sua jurisdição. É nesse nicho, o de uma alternativa *privacy-first* com recorte brasileiro e explicabilidade por decisão, que reside o diferencial do trabalho, e não na cobertura ou na maturidade, em que os serviços consolidados permanecem superiores.

6.4 Limitações

As limitações reconhecidas neste trabalho consolidam-se em torno de três eixos: condições de execução, escopo experimental e abertura metodológica. A discussão completa, com números e referências cruzadas, está na Seção 5.7 do Capítulo 5; esta seção retoma os pontos principais como moldura de leitura.

A limitação mais ampla diz respeito à **moldura de recursos computacionais e financeiros**. Todo o desenvolvimento e treinamento foi conduzido sem auxílio monetário específico para infraestrutura. O treinamento dos modelos ocorreu em Google Colab com *Graphics Processing Unit* (GPU) NVIDIA T4 da versão gratuita do serviço; os testes locais e a validação da API foram executados em computador pessoal do autor (processador AMD Ryzen 5 5600X de seis núcleos, 24 GB de memória DDR4 e GPU AMD Radeon RX 6600), com os serviços empacotados em contêineres Docker. A GPU AMD Radeon RX 6600 não é diretamente utilizável para inferência PyTorch CUDA padrão dos modelos BERT e restringe os testes locais à *Central Processing Unit* (CPU). Essa moldura delimita a quantidade de experimentos realizados, o tempo dedicado ao treinamento de cada modelo, o tamanho dos modelos que puderam ser explorados, a profundidade da busca de hiperparâmetros, a impossibilidade de repetir treinamentos com múltiplas *seeds* para análise de variância e a inviabilidade de explorar arquiteturas de maior porte (por exemplo, DeBERTa-v3-large ou modelos de linguagem grandes).

A **meta inicial de FPR menor ou igual a 0,5 %**, declarada na fase de proposta, **não foi atingida**. O modelo principal apresentou FPR aproximada de 15,6 % no conjunto de teste fixo; o melhor valor observado entre os modelos avaliados foi de aproximadamente 9 % pelo DomURL-BERT multimodal, que não foi adotado por motivos de comportamento operacional em validação empírica. A redução adicional da FPR para aproximar-se da meta de proposta fica registrada como trabalho futuro (Seção 6.5).

Outras limitações, derivadas das condições de execução e do escopo experimental adotado, são listadas a seguir e detalhadas no Capítulo 5.

- A **avaliação em campo** prevista para a Semana 15 do cronograma original foi conduzida em duas etapas: uma qualitativa, com os artefatos `validacao_modelo.json` (62 URLs) e `validacao_cascata.json` (101 URLs); e uma formal, sobre lista pré-registrada de 1 946 URLs com medição automática (Tabela 10). Ficam como trabalho futuro uma rodada com conjunto inteiramente disjunto do treino e deduplicação explícita, e uma sessão controlada de navegação com participantes.

- A **calibração formal de probabilidades** do modelo principal e da cascata não foi conduzida; os limiares 0,15, 0,85 e 0,65 que governam a cascata foram escolhidos empiricamente.

- A **explicabilidade** das decisões foi entregue para os três modelos por meio de SHAP e de mapas de atenção (Seção 5.6); permanecem como trabalho futuro técnicas complementares como LIME, gradiente integrado e a análise sistemática de viés e justiça, e a profundidade da explicação nos *transformers* ainda é limitada pela tokenização em subpalavras.

- A **análise qualitativa sistemática de erros** (categorização manual de exemplos representativos de falsos positivos e falsos negativos) não foi realizada.

- A **comparação formal entre variantes do DomURL-BERT** cobriu apenas duas configurações (URL+WHOIS e multimodal completo); subconjuntos parciais não couberam no escopo experimental.

- O **conjunto BR** apresenta forte dominância da fonte `kmack_phishing_urls` (cerca de 99,3 % das amostras), o que delimita a generalização do desempenho reportado.

- A **fração de tráfego** que efetivamente atinge o modelo na zona ambígua não foi medida em produção; a aproximação inferida a partir de `validacao_cascata.json` sugere fração baixa, mas o valor exato em tráfego real depende do comportamento de uso da extensão.

- A **avaliação de carga e concorrência da API** restringiu-se a um teste local em contêiner Docker, com inferência em CPU no computador do autor e até 32 requisições simultâneas (Tabela 11 do Capítulo 5). A avaliação formal não foi conduzida com múltiplos clientes reais em condições de produção,

com escalonamento horizontal ou com perfil de tráfego realista; essa avaliação permanece registrada como trabalho futuro (Seção 6.5).

- O **módulo de análise de *e-mail*** está entregue e integrado ponta a ponta na extensão: faz parte do escopo funcional realizado, opera sobre Gmail e Outlook *web* (Seção 4.11) e devolve veredicto ao usuário no *webmail*. A avaliação quantitativa do módulo é parcial. O classificador reúsa um *checkpoint* DistilBERT pré-treinado de terceiros, sem *fine-tuning* próprio sobre *e-mails* em português brasileiro. A medição conduzida restringiu-se a um conjunto público de *e-mails* em inglês (F1 de 0,9881, tratado como teto otimista por possível contaminação treino-teste, Seção 5.7); não houve medição sobre *e-mails* reais em português brasileiro, nem quantificação do erro introduzido pela tradução automática do português para o inglês. O caráter de prova de conceito refere-se, portanto, à avaliação, não à existência ou ao funcionamento do módulo.

- O **escopo de plataforma** concentra-se em navegadores baseados em Chromium e no padrão *Manifest Version* 3 (MV3), sem cobertura formal de outros navegadores ou de plataformas móveis.

Em conjunto, essas limitações não invalidam os resultados obtidos, mas delineiam o contexto em que devem ser interpretados e apontam diretamente os trabalhos futuros descritos a seguir.

6.5 Trabalhos futuros

Os trabalhos futuros consolidados nesta seção combinam os derivados diretamente dos resultados experimentais (discutidos na Seção 5.10 do Capítulo 5) com a direção de produto *enterprise* apontada pelo trabalho. Estão organizados em três frentes.

A **frente técnica** abrange itens diretamente derivados das limitações listadas na seção anterior. Em primeiro lugar, a redução da FPR para aproximar-se da meta inicial declarada na proposta, por meio de calibração formal de probabilidades, ajuste sistemático de limiares e exploração de *ensembles* com sinais adicionais. Em segundo lugar, uma rodada de validação com conjunto inteiramente disjunto do conjunto de treino e deduplicação explícita, que isolaria o efeito de reconhecimento quantificado na Seção 5.7, complementada por uma sessão controlada de navegação com participantes.

Em terceiro lugar, a medição da fração de tráfego que efetivamente atinge a zona ambígua em produção, por meio de instrumentação do *endpoint* `POST /predict` para registrar o `cascade_path` por amostra, a fim de obter o valor que falta para reportar a FPR efetiva da extensão. Em quarto lugar, a análise qualitativa sistemática de erros, com categorização manual de 50 a 100 amostras representativas de falsos positivos e falsos negativos, e a investigação da causa subjacente do comportamento operacional inferior do multimodal: análise de *loss* de treino contra validação, importância das *features* de *Document Object Model* (DOM) e exemplos contrastantes. Em quinto lugar, a ampliação da explicabilidade já entregue (SHAP e mapas de atenção para os três modelos, Seção 5.6) com técnicas complementares como LIME e gradiente integrado, e a análise sistemática de viés, justiça e qualidade apoiada nessas atribuições. Em sexto lugar, a repetição dos experimentos com infraestrutura mais robusta (por exemplo, FabLab da Unipampa ou GPUs dedicadas), de modo a permitir múltiplas *seeds* para análise de variância, busca de hiperparâmetros mais ampla, modelos maiores (DeBERTa-v3-large, modelos de linguagem grandes) e ampliação da diversidade do conjunto BR para reduzir a dominância de `kmack_phishing_urls`.

A **frente de produto *enterprise*** estende a direção de produto descrita no Capítulo 4. Inclui os seguintes itens:

- ampliação do *dashboard* com novas métricas, por exemplo, *cohort* de organizações, séries temporais por categoria de URL e indicadores de qualidade do modelo em produção;
- relatórios de risco por usuário e por organização, com identificação das contas mais expostas a tentativas de *phishing*;
- política de bloqueio configurável pelo gestor, com níveis de severidade que bloqueiam ou apenas alertam;
- integração com provedores de identidade corporativos para autenticação única (*Single Sign-On* – SSO) e diretórios como o *Active Directory*;
- alertas direcionados ao gestor por usuário e por categoria de evento;
- *onboarding self-service* para novas organizações, sem dependência de operação manual via interface de linha de comando;

- a ampliação do teste de carga da API já conduzido (Seção 5.7, com vazão de cerca de 24 requisições por segundo por instância em CPU e P95 crescente sob concorrência) para um perfil de chamadas realista, com escalonamento horizontal e medição em produção;
- ampliação de idiomas no *pipeline* de *e-mail* além do par português-inglês, com alemão e espanhol como próximas adições;
- integração com *webmails* além de Gmail e Outlook, em particular Yahoo Mail, ProtonMail e Zoho;
- políticas explícitas de retenção de dados em PostgreSQL para alinhamento com a LGPD;
- *roadmap* comercial com modelos de licenciamento, suporte e nível de serviço (*Service Level Agreement* – SLA).

A **frente de pesquisa de longo prazo** aponta direções que ultrapassam o escopo deste trabalho, mas que decorrem naturalmente da arquitetura adotada. Inclui a **detecção de *drift* de conceito** e a **adaptação contínua** do modelo a novas distribuições de *phishing*, em linha com avanços recentes da literatura (YANG *et al.*, 2025; SKOPIK; WURZENBERGER; LANDAUER, 2021); a **explicabilidade interpretável ao usuário final**, não apenas ao pesquisador, com mensagens de alerta que comuniquem o motivo da classificação de forma compreensível em texto natural; e o **desenho de listas adaptativas** (*whitelist* auto-curada por reputação) que combinem a velocidade da decisão local em $O(1)$ com a capacidade de incorporar sinais novos sem redistribuição da extensão.

Essas frentes não esgotam o que pode ser feito a partir desta solução, mas representam os passos mais imediatos para reduzir as limitações reconhecidas e para evoluir o trabalho em direção a um produto operacional mais maduro.

REFERÊNCIAS

ABDELNABI, S.; KROMBHOLZ, K.; FRITZ, M. VisualPhishNet: Zero-day phishing website detection by visual similarity. In: **Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20)**. Nova York, NY: ACM, 2020. p. 1681–1698.

ABNAR, S.; ZUIDEMA, W. Quantifying attention flow in transformers. In: **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)**. Online: Association for Computational Linguistics, 2020. p. 4190–4197. Disponível em: <https://aclanthology.org/2020.acl-main.385>.

abuse.ch. **URLhaus: Malware URL Exchange**. 2026. Projeto do Institute for Cybersecurity and Engineering (ICE) da Bern University of Applied Sciences (BFH); API e feeds comunitários gratuitos. Disponível em: <https://urlhaus.abuse.ch/>. Acesso em: 4 maio 2026.

AGHAEI, E. *et al.* SecureBERT: A domain-specific language model for cybersecurity. **arXiv preprint arXiv:2204.02685**, 2022. Disponível em: <https://arxiv.org/abs/2204.02685>.

ANSAR, S. *et al.* **The Global Findex Database 2021: Financial Inclusion, Digital Payments, and Resilience in the Age of COVID-19**. Washington, D.C.: World Bank Group, 2022. Disponível em: <http://documents.worldbank.org/curated/en/099818107072234182>. Acesso em: 13 nov. 2025.

Anti-Phishing Working Group. **Phishing Activity Trends Report: 4th Quarter 2025**. Cambridge, MA, 2026. Relatório trimestral do APWG referente a 2025; consolidado anual obtido pela soma dos quatro trimestres do ano. Disponível em: <https://docs.apwg.org/reports/apwg_trends_report_q4_2025.pdf>. Acesso em: 17 maio 2026.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. New York, NY: Springer, 2006.

CALZAROSSA, M. C.; GIUDICI, P.; ZIENI, R. Explainable machine learning for phishing feature detection. **Quality and Reliability Engineering International**, v. 40, n. 1, p. 362–373, 2024.

CANOVA, G. *et al.* Learn to spot phishing URLs with the Android NoPhish app. In: **Financial Cryptography and Data Security (FC 2014), Workshop on Usable Security**. Berlim, Heidelberg: Springer, 2014. p. 54–61.

CECHINEL, C.; CAMARGO, S. da S. Mineração de dados educacionais: avaliação e interpretação de modelos de classificação. In: JAQUES, P. A. *et al.* (Ed.). **Metodologia de pesquisa científica em informática na educação: abordagem quantitativa**. Porto Alegre: Sociedade Brasileira de Computação, 2020, (Série Metodologia de Pesquisa em Informática na Educação, v. 2). ISBN 978-85-7669-494-6. Disponível em: <https://ceie.sbc.org.br/metodologia/livro-2>. Acesso em: 13 nov. 2025.

Chart.js Contributors. **Chart.js Documentation**. 2023. Biblioteca de gráficos para a *web*. Versão 4.4.0 utilizada neste trabalho. Disponível em: <https://www.chartjs.org/docs/latest/>. Acesso em: 15 jun. 2026.

CHEN, T.; GUESTRIN, C. XGBoost: A scalable tree boosting system. In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. Nova York, NY: ACM, 2016. p. 785–794.

CHOW, C. K. On optimum recognition error and reject tradeoff. **IEEE Transactions on Information Theory**, v. 16, n. 1, p. 41–46, 1970. Disponível em: <https://ieeexplore.ieee.org/document/1054406>.

Cisco. **Cisco Umbrella: DNS-layer security**. 2026. Serviço corporativo de segurança na camada de DNS, sucessor do OpenDNS (2006); bloqueia domínios maliciosos antes do estabelecimento da conexão, com filtragem centralizada na nuvem da Cisco. Disponível em: <https://umbrella.cisco.com/>. Acesso em: 29 maio 2026.

Cloudflare. **Cloudflare Gateway: Secure Web Gateway**. 2026. *Secure Web Gateway* corporativo com filtragem de DNS e HTTP e detecção de *phishing* baseada em aprendizado de máquina; processamento na nuvem do fornecedor, com plano gratuito até 50 usuários. Disponível em: <https://www.cloudflare.com/zero-trust/products/gateway/>. Acesso em: 29 maio 2026.

CORMEN, T. H. *et al.* **Introduction to Algorithms**. 4th. ed. Cambridge, MA: MIT Press, 2022.

DAS, A. *et al.* SoK: A comprehensive reexamination of phishing research from the security perspective. **IEEE Communications Surveys & Tutorials**, IEEE, v. 22, n. 1, p. 671–708, 2020.

DO, N. Q. *et al.* Deep learning for phishing detection: Taxonomy, current challenges and future directions. **IEEE Access**, IEEE, v. 10, p. 36429–36463, 2022.

DUA, D.; GRAFF, C. **Phishing Websites Data Set**. 2017. UCI Machine Learning Repository. Disponível em: <https://archive.ics.uci.edu/dataset/327/phishing+websites>. Acesso em: 20 nov. 2025.

FAJAR, A.; YAZID, S.; BUDI, I. Enhancing phishing detection through feature importance analysis and explainable AI: A comparative study of CatBoost, XGBoost, and EBM models. **arXiv preprint arXiv:2411.06860**, 2024. Disponível em: <https://arxiv.org/abs/2411.06860>.

FastAPI. **FastAPI Documentation**. 2024. *Framework* web assíncrono em Python. Versão 0.109 utilizada neste trabalho. Disponível em: <https://fastapi.tiangolo.com/>. Acesso em: 15 jun. 2026.

FAWCETT, T. An introduction to ROC analysis. **Pattern Recognition Letters**, Elsevier, v. 27, n. 8, p. 861–874, 2006.

Federal Bureau of Investigation. **Internet Crime Report 2024**. Washington, D.C., 2025. Relatório anual do IC3 referente ao ano-calendário 2024, publicado em 2025; *phishing* é a categoria mais reportada. Disponível em: <https://www.ic3.gov/AnnualReport/Reports/2024_IC3Report.pdf>. Acesso em: 17 maio 2026.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge, MA: MIT Press, 2016.

Google. **Safe Browsing**. 2024. Serviço de proteção contra *phishing* e *malware* integrado a Chrome, Firefox e Safari desde 2007. O modo padrão mantém prefixos de *hash* localmente e consulta o servidor apenas diante de suspeita; desde março de 2024 emprega *Oblivious HTTP* (OHTTP) para desacoplar o endereço IP do cliente do conteúdo consultado. Disponível em: <https://safebrowsing.google.com/>. Acesso em: 29 maio 2026.

Google Chrome Developers. **Chrome Extensions: Manifest V3 Documentation**. 2025. Disponível em: <https://developer.chrome.com/docs/extensions/mv3/>. Acesso em: 20 nov. 2025.

HE, H.; GARCIA, E. A. Learning from imbalanced data. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 21, n. 9, p. 1263–1284, 2009.

International Telecommunication Union. **Facts and Figures 2024: Internet use**. Genebra, Suíça, 2024. Disponível em: <https://www.itu.int/itu-d/reports/statistics/2024/11/10/ff24-internet-use/>. Acesso em: 13 nov. 2025.

JIAO, X. *et al.* TinyBERT: Distilling BERT for natural language understanding. In: **Findings of the Association for Computational Linguistics: EMNLP 2020**. Online: Association for Computational Linguistics, 2020. p. 4163–4174. Disponível em: <https://aclanthology.org/2020.findings-emnlp.372/>.

KE, G. *et al.* LightGBM: A highly efficient gradient boosting decision tree. In: **Advances in Neural Information Processing Systems 30 (NeurIPS 2017)**. Red Hook, NY: Curran Associates, Inc., 2017. p. 3146–3154.

kmack. **Phishing_urls: *dataset* de URLs rotuladas para detecção de *phishing***. 2024. Hugging Face Datasets. *Dataset* comunitário de 708 820 URLs (567 020 treino, 70 900 teste, 70 900 validação) em formato Parquet. Primeira publicação em 14 de abril de 2024. Sem *paper* associado e sem revisão por pares; usado neste trabalho como fonte primária de URLs (cerca de 99,3% do conjunto BR), com a limitação de confiabilidade discutida no Cap. 5 e no Cap. 6. Disponível em: <https://huggingface.co/datasets/kmack/Phishing_urls>. Acesso em: 17 maio 2026.

KUMAR, R.; GUPTA, B. B. *et al.* Detecting phishing websites using deep learning: A survey. **Computers & Security**, Elsevier, v. 108, p. 102367, 2022.

LI, W. *et al.* A state-of-the-art review on phishing website detection techniques. **IEEE Access**, IEEE, v. 12, p. 187976–188012, 2024.

LUNDBERG, S. M.; LEE, S.-I. A unified approach to interpreting model predictions. In: **Advances in Neural Information Processing Systems 30 (NeurIPS 2017)**. Long Beach, CA: Curran Associates, 2017. p. 4765–4774. Disponível em: <https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html>.

MAHDAOUY, A. E. *et al.* DomURLs_BERT: Pre-trained BERT-based model for malicious domains and URLs detection and classification. **arXiv preprint arXiv:2409.09143**, 2024. Disponível em: <https://arxiv.org/abs/2409.09143>.

Majestic-12 Ltd. **Majestic Million: lista diária dos um milhão de domínios mais referenciados**. 2026. Atualização diária; licença Creative Commons. Disponível em: <https://majestic.com/reports/majestic-million>. Acesso em: 4 maio 2026.

MALIK, L. G. *et al.* Browser extension for fake URL detection. **arXiv preprint arXiv:2411.13581**, 2024. Disponível em: <https://arxiv.org/abs/2411.13581>.

MANERIKER, P. *et al.* URLTran: Improving phishing URL detection using transformers. In: **MILCOM 2021 – IEEE Military Communications Conference**. San Diego, CA: IEEE, 2021. p. 197–204.

Meta Open Source. **React Documentation**. 2022. Biblioteca para construção de interfaces. Versão 18.2.0 utilizada neste trabalho. Disponível em: <https://react.dev/>. Acesso em: 15 jun. 2026.

Microsoft. **TypeScript Documentation**. 2023. Superconjunto tipado de JavaScript. Versão 5.3.3 utilizada neste trabalho. Disponível em: <https://www.typescriptlang.org/docs/>. Acesso em: 15 jun. 2026.

Microsoft. **Microsoft Defender SmartScreen**. 2025. Filtro de reputação de URL e de *download* integrado ao Windows e ao navegador Edge; envia a URL ao serviço de reputação da Microsoft quando o endereço não consta nas listas locais. Disponível em: <https://learn.microsoft.com/windows/security/operating-system-security/virus-and-threat-protection/microsoft-defender-smartscreen/>. Acesso em: 29 maio 2026.

MITCHELL, T. M. **Machine Learning**. New York, NY: McGraw-Hill, 1997.

MOSA, D. T. *et al.* Machine learning techniques for detecting phishing URL attacks. **Computers, Materials & Continua**, v. 75, n. 1, p. 1271–1290, 2023.

NIELSEN, J. **Usability Engineering**. Boston, MA: Academic Press, 1993. Capítulo 5 (*Response Times*) estabelece os limiares de 0,1 s, 1,0 s e 10 s para responsividade interativa percebida pelo usuário. ISBN 978-0-12-518405-2.

OECD. **OECD Digital Economy Outlook 2024 (Volume 1): Embracing the Technology Frontier**. Paris: OECD Publishing, 2024.

OpenDNS. **PhishTank: Join the Fight Against Phishing**. 2025. Repositório colaborativo de URLs de phishing e não phishing. Disponível em: <https://phishtank.org>. Acesso em: 20 nov. 2025.

OpenPhish. **OpenPhish: Phishing Intelligence Feeds**. 2026. Community feed gratuito atualizado a cada 12 h; feeds premium para CERTs e instituições acadêmicas. Disponível em: <https://www.openphish.com/>. Acesso em: 4 maio 2026.

OVI, M. S. I.; RAHMAN, M. H.; HOSSAIN, M. A. PhishGuard: A multi-layered ensemble model for optimal phishing website detection. **arXiv preprint arXiv:2409.19825**, 2024. Disponível em: <https://arxiv.org/abs/2409.19825>.

PANTELAIOS, N.; KAPRAVELOS, A. Work-in-progress: Manifest V3 unveiled: Navigating the new era of browser extensions. **arXiv preprint arXiv:2404.08310**, 2024. Disponível em: <https://arxiv.org/abs/2404.08310>.

PEDREGOSA, F. *et al.* Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, JMLR.org, v. 12, p. 2825–2830, 2011.

PERROTTA, R.; HAO, F. Botnet in the browser: Understanding threats caused by malicious browser extensions. **IEEE Security & Privacy**, IEEE, v. 16, n. 4, p. 66–73, 2018.

Phishing-Database community. **Phishing.Database: Phishing Domains, URLs and Threats Database**. 2026. Repositório aberto mantido pela comunidade Safekeepers; vendor oficial de dados do VirusTotal; validação periódica via PyFunceble. Disponível em: <https://github.com/mitchellkrogza/Phishing.Database>. Acesso em: 4 maio 2026.

PhishStats. **PhishStats: Phishing Intelligence & Cybercrime Data**. 2026. Compartilhamento de inteligência sobre phishing desde 2014; API e feed CSV atualizados em tempo quase real. Disponível em: <https://phishstats.info/>. Acesso em: 4 maio 2026.

PICAZO-SANCHEZ, P.; TAPIADOR, J.; SCHNEIDER, G. After you, please: Browser extensions order attacks and countermeasures. **International Journal of Information Security**, v. 19, n. 6, p. 701–721, 2020.

Plasmo. **Plasmo Framework Documentation**. 2024. Documentação oficial do *framework* de extensões. Versão 0.89.4 utilizada neste trabalho. Disponível em: <https://docs.plasmo.com/>. Acesso em: 15 jun. 2026.

POCHAT, V. L. *et al.* Tranco: A research-oriented top sites ranking hardened against manipulation. In: **Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS 2019)**. San Diego, CA: Internet Society, 2019. Disponível em: <https://tranco-list.eu/>.

PostgreSQL Global Development Group. **PostgreSQL 16 Documentation**. 2023. Sistema gerenciador de banco de dados relacional. Versão 16 utilizada neste trabalho. Disponível em: <https://www.postgresql.org/docs/16/>. Acesso em: 15 jun. 2026.

PROKHORENKOVA, L. *et al.* CatBoost: Unbiased boosting with categorical features. In: BENGIO, S. *et al.* (Ed.). **Advances in Neural Information Processing Systems 31 (NeurIPS 2018)**. Red Hook, NY: Curran Associates, Inc., 2018. p. 6639–6649. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2018/hash/14491b756b3a51daac41c24863285549-Abstract.html>.

RAO, R. S.; VAISHNAVI, T.; PAIS, A. R. CatchPhish: Detection of phishing websites by inspecting URLs. **Journal of Ambient Intelligence and Humanized Computing**, Springer, v. 11, p. 813–825, 2020.

ROSE, M. A. S. R.; BASIR, N.; HENG, N. F. N. R. Phishing detection and prevention using Chrome extension. In: **2022 10th International Symposium on Digital Forensics and Security (ISDFS)**. Kuala Lumpur, Malaysia: IEEE, 2022. p. 1–6.

ROY, S. S.; NILIZADEH, S. PhishLang: A real-time, fully client-side phishing detection framework using MobileBERT. **arXiv preprint arXiv:2408.05667**, 2024. Disponível em: <https://arxiv.org/abs/2408.05667>.

SANH, V. *et al.* DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. **arXiv preprint arXiv:1910.01108**, 2019. Disponível em: <https://arxiv.org/abs/1910.01108>.

SKOPIK, F.; WURZENBERGER, M.; LANDAUER, M. The seven golden principles of effective anomaly-based intrusion detection. **IEEE Security & Privacy**, IEEE, v. 19, n. 5, p. 36–45, 2021.

SQLAlchemy. **SQLAlchemy Documentation**. 2023. *Toolkit* SQL e ORM para Python. Versão 2.0 em modo assíncrono utilizada neste trabalho. Disponível em: <https://www.sqlalchemy.org/>. Acesso em: 15 jun. 2026.

United Nations Department of Economic and Social Affairs. **United Nations E-Government Survey 2022: The Future of Digital Government**. Nova York, NY: United Nations, 2022.

Verizon Business. **2025 Data Breach Investigations Report**. Basking Ridge, NJ, 2025. 18ª edição do DBIR; relatório anual sobre vetores iniciais de violação de dados, com *phishing* entre os principais vetores iniciais. Disponível em: <https://www.verizon.com/business/resources/T16f/reports/2025-dbir-data-breach-investigations-report.pdf>. Acesso em: 17 maio 2026.

XUE, Y. *et al.* MultiPhishGuard: An LLM-based multi-agent system for phishing email detection. **arXiv preprint arXiv:2505.23803**, 2025. Disponível em: <https://arxiv.org/abs/2505.23803>.

YANG, S. *et al.* Self-supervised adaptation method to concept drift for network intrusion detection. **IEEE Transactions on Dependable and Secure Computing**, IEEE, v. 22, n. 6, p. 7632–7646, 2025.

YAZDANI, R.; TOORN, O. van der; SPEROTTO, A. A case of identity: Detection of suspicious IDN homograph domains using active DNS measurements. In: **2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)**. Gênova, Itália: IEEE, 2020. p. 559–564.

ZIENI, R.; MASSARI, L.; CALZAROSSA, M. C. Phishing or not phishing? a survey on the detection of phishing websites. **IEEE Access**, IEEE, v. 11, p. 18499–18519, 2023.