UNIVERSIDADE SÃO JUDAS TADEU
MODELOS, MÉTODOS E TÉCNICAS DA ENGENHARIA DE SOFTWARE

Gabriel Almeida Portela - 825233281
Daniel Almeida Portela

NETPROBE

São Paulo
2025

Gabriel Almeida Portela - 825233281
Daniel Almeida Portela

NETPROBE

Documentação do Produto de Software apresentado à Unidade Curricular de Modelos, métodos e técnicas da engenharia de software da Universidade São Judas Tadeu, como requisito parcial para concluir o mesmo.
Orientador: Prof. Dr. Carlos L. Noriega

São Paulo
2025

RESUMO

Este documento apresenta a documentação do produto de software NetProbe, desenvolvido como parte dos requisitos da disciplina Modelos, Métodos e Técnicas da Engenharia de Software na Universidade São Judas Tadeu. O NetProbe é uma ferramenta projetada para monitorar e analisar o desempenho de redes de computadores, oferecendo insights valiosos para administradores de rede. A documentação abrange desde a concepção inicial até a implementação final, incluindo diagramas UML, especificações técnicas e instruções de uso.

Palavras-chave: Documentação de Software, Engenharia de Software, Monitoramento de Redes, NetProbe.

Sumário

1. Introdução

1.1. Tema

O software **NetProbe** é uma solução inovadora focada em segurança, eficiência e confiabilidade em redes de computadores, que serve como base para toda solução digital moderna. Este projeto também alinha-se com o objetivo 9 da ONU, que busca construir infraestruturas resilientes, promover a industrialização inclusiva e sustentável, e fomentar a inovação.

1.2. Objetivos

O **NetProbe**, como projeto, tem como objetivo desenvolver um analisador de tráfego de rede que seja capaz de monitorar o tráfego de rede em tempo real, e tem como publico-alvo estudantes de ciência/engenharia da computação, entusiastas de redes, e educadores.

1.3. Escopo Principal

1.3.1. Atores do Sistema

- **Usuário**: Entusiasta de redes que irá se registrar, autenticar e utilizar o sistema para monitorar e analisar o tráfego da rede local.

1.3.2. Casos de Uso

1.3.2.1. Cadastrar-se no Sistema

- **Código**: UC01
- **Ator Principal**: Usuário
- **Pré-condições**: O usuário não possui uma conta de acesso ao sistema.
- **Gatilho**: O usuário acessa a página inicial e decide criar uma nova conta para acessar o dashboard.
- **Cenário de Sucesso Principal**: **1.** O usuário acessa a interface web do sistema. **2.** O sistema exibe a página de login e apresenta uma opção para “Cadastro”. **3.** O usuário seleciona a opção de cadastro. **4.** O sistema exibe um formulário solicitando informações como e-mail e senha. **5.** O usuário preenche o formulário com dados válidos e o submete. **6.** O sistema valida os dados, cria a conta de usuário e armazena as informações de forma segura. **7.** O sistema redireciona o usuário para a página de login com uma mensagem de sucesso, indicando que o cadastro foi concluído.
- **Exceções**: **6a.** Se o e-mail fornecido já estiver cadastrado, o sistema exibe uma mensagem de erro informando que o usuário já existe. **6b.** Se os dados fornecidos forem inválidos (ex: formato de e-mail incorreto, senha fora do padrão exigido), o sistema exibe uma mensagem de erro e solicita a correção.

1.3.2.2. Autenticar-se no Sistema (Login)

- **Código**: UC02
- **Ator Principal**: Usuário

- **Pré-condições**: O usuário deve ter uma conta previamente cadastrada no sistema.
- **Gatilho**: O usuário deseja acessar o dashboard de monitoramento de rede.
- **Cenário de Sucesso Principal**: **1.** O usuário acessa a página de login do sistema. **2.** O usuário insere suas credenciais (e-mail e senha) nos campos correspondentes. **3.** O usuário submete o formulário de login. **4.** O sistema valida as credenciais. **5.** Após a autenticação bem-sucedida, o sistema concede acesso e exibe o dashboard principal de monitoramento de rede.
- **Exceções**: **4a.** Se as credenciais estiverem incorretas, o sistema exibe uma mensagem de "Usuário ou senha inválidos" e permite que o usuário tente novamente.

1.3.2.3. Monitorar Tráfego de Rede em Tempo Real

- **Código**: UC03
- **Ator Principal**: Usuário
- **Pré-condições**: O usuário está autenticado e na página do dashboard web. O backend em C++ está ativo e capturando pacotes da rede local.
- **Gatilho**: O usuário acessa o dashboard para visualizar a atividade da rede.
- **Cenário de Sucesso Principal**: **1.** Após o login, o usuário visualiza o dashboard principal. **2.** O backend em C++, integrado ao frontend via WebAssembly, captura os pacotes da rede. **3.** O dashboard exibe uma tabela ou lista que é atualizada em tempo real com os pacotes capturados. **4.** Para cada pacote, o sistema exibe informações essenciais como: endereço IP de origem, endereço IP de destino, porta de origem, porta de destino, protocolo (TCP, UDP, ICMP, etc.) e o volume de dados (tamanho do pacote).
- **Exceções**: **3a.** Caso o backend em C++ não consiga capturar o tráfego (ex: falta de permissões), o dashboard exibe uma mensagem de erro informando sobre a falha na captura de pacotes.

1.3.2.4. Filtrar Tráfego de Rede

- **Código**: UC04
- **Ator Principal**: Usuário
- **Pré-condições**: O usuário está autenticado e visualizando o tráfego em tempo real no dashboard.
- **Gatilho**: O usuário deseja isolar e visualizar pacotes específicos para uma análise mais focada.
- **Cenário de Sucesso Principal**: **1.** No dashboard, o usuário localiza os campos ou a seção de filtros. **2.** O usuário preenche um ou mais critérios de filtro, como: protocolo (ex: "HTTP"), endereço IP específico (origem ou destino) ou número de porta. **3.** O usuário aplica o filtro. **4.** O sistema processa a regra de filtro e atualiza a exibição do tráfego, mostrando apenas os pacotes que correspondem aos critérios definidos. **5.** O usuário tem a opção de modificar ou limpar os filtros para retornar à visualização completa.

- **Exceções**: **4a.** Se o usuário inserir um valor de filtro inválido (ex: um endereço IP mal formatado), o sistema exibe uma notificação de erro e não aplica o filtro até que seja corrigido.

1.3.2.5. Analisar Pacote de Dados

- **Código**: UC05
- **Ator Principal**: Usuário
- **Pré-condições**: O usuário está autenticado e visualizando a lista de pacotes no dashboard.
- **Gatilho**: O usuário identifica um pacote de interesse e deseja inspecionar seus detalhes técnicos.
- **Cenário de Sucesso Principal**: **1.** O usuário seleciona (clica em) um pacote específico na lista de tráfego. **2.** O sistema exibe uma visualização detalhada do pacote selecionado em uma janela modal ou em um painel lateral. **3.** Esta visualização apresenta o conteúdo decodificado do pacote, dividido por camadas, como o cabeçalho Ethernet, cabeçalho IP, cabeçalho do protocolo de transporte (TCP/UDP) e a carga útil (payload) dos dados. **4.** O usuário analisa as informações detalhadas para entender a natureza da comunicação. **5.** O usuário fecha a visualização de detalhes para retornar à lista principal de tráfego.
- **Exceções**: **3a.** Se o conteúdo do pacote estiver criptografado ou em um formato que não possa ser decodificado, o sistema exibirá as informações que conseguir interpretar (como os cabeçalhos) e indicará que a carga útil não é legível.

2. Modelo de Qualidade ISO/IEC 25010

A ISO/IEC 25010:2011 define o modelo de qualidade de produto de software com oito características e suas subcaracterísticas, servindo de base para especificação, avaliação e melhoria da qualidade.

2.1. Características e subcaracterísticas

- Adequação Funcional (Functional suitability)
  - Completude funcional; Correção funcional; Adequação/pertinência funcional.
- Eficiência de Desempenho (Performance efficiency)
  - Comportamento temporal; Utilização de recursos; Capacidade.
- Compatibilidade (Compatibility)
  - Coexistência; Interoperabilidade.
- Usabilidade (Usability)
  - Reconhecibilidade de adequação; Aprendibilidade; Operacionalidade; Proteção ao erro do usuário; Estética da interface; Acessibilidade.
- Confiabilidade (Reliability)
  - Maturidade; Disponibilidade; Tolerância a falhas; Recuperabilidade.
- Segurança (Security)
  - Confidencialidade; Integridade; Não repúdio; Responsabilização; Autenticidade.
- Manutenibilidade (Maintainability)

- Modularidade; Reusabilidade; Analisabilidade; Modificabilidade; Testabilidade.
- Portabilidade (Portability)
  - Adaptabilidade; Instalabilidade; Substituibilidade.

2.2. Aplicação ao NetProbe

- Adequação funcional: cobertura dos RF005–RF012 (captura, filtro, análise de pacotes).
- Desempenho: latência baixa no pipeline C++ → WebAssembly → UI (RNF001–RNF003).
- Compatibilidade: interoperar via formatos abertos (CSV/JSON) e APIs.
- Usabilidade: UI clara para entusiastas de redes; proteção a erros de entrada.
- Confiabilidade: execução contínua e recuperação de falhas de captura.
- Segurança: autenticação, criptografia de senhas e controle de acesso (RNF004–RNF006).
- Manutenibilidade: modularização do backend e componentes de UI testáveis.
- Portabilidade: build C++ multiplataforma; suporte a navegadores modernos.

2.3. Medição e avaliação

- Use ISO/IEC 25023 para indicadores (ex.: tempo de resposta, uso de CPU/RAM, taxa de falhas, cobertura de requisitos).
- Planeje a avaliação conforme ISO/IEC 25040 (processo de avaliação), definindo métricas, critérios e evidências (testes, logs, inspeções).

3. Requisitos do Sistema de Software

3.1. Requisitos Funcionais

3.1.1. Autenticação e Gerenciamento de Usuários

- **RF001**: O sistema deve permitir que novos usuários se cadastrem fornecendo um e-mail e uma senha.
- **RF002**: O sistema deve permitir que usuários cadastrados sejam autenticados utilizando seu e-mail e senha.
- **RF003**: O sistema deve permitir que um usuário autenticado encerre sua sessão de forma segura.
- **RF004**: O sistema deve garantir que apenas usuários autenticados possam acessar as páginas do dashboard de visualização de dados.

3.1.2. Monitoramento de Tráfego

- **RF005**: O backend em C++ deve capturar pacotes da rede local em tempo real.
- **RF006**: O sistema deve processar os pacotes capturados para extrair informações relevantes.
- **RF007**: O backend deve enviar os dados processados para o frontend para visualização.
- **RF008**: O dashboard web deve exibir os dados de tráfego de rede recebidos do backend de forma dinâmica.
- **RF009**: Os dados na interface devem ser apresentados de maneira clara e organizada.

3.1.3. Filtragem de Tráfego

- **RF010**: O sistema deve fornecer ao usuário a capacidade de filtrar os dados de tráfego exibidos no dashboard.

3.1.4. Análise de Pacotes

- **RF011**: O sistema deve permitir que o usuário inspecione os detalhes de um pacote específico.
- **RF012**: O sistema deve detectar e informar ao usuário sobre erros que impeçam a captura de dados.

3.2. Requisitos Não-Funcionais

3.2.1. Desempenho

- **RNF001**: O backend em C++ deve capturar e processar pacotes com baixa latência com o intuito de garantir a visualização dos dados em tempo real.
- **RNF002**: A *UI* (Interface do Usuário) do dashboard web deve ser responsiva a atualizar os dados de tráfego de forma fluida, ou seja, sem travamentos.
- **RNF003**: A filtragem de dados exibidos pelo dashboard web deve produzir resultados de forma ideal—rápido, e suave ou harmonizada.

3.2.2. Segurança

- **RNF004**: As senhas dos usuários cadastrados devem ser guardadas de forma segura no banco de dados, utilizando técnicas e algoritmos avançados de criptografia.
- **RNF005**: A comunicação entre o frontend e o backend deve ser segura, especialmente durante o processo de autenticação.
- **RNF006**: O acesso ao dashboard web para visualização de dados deve ser estritamente controlado por sessão de usuário autenticada.

3.2.3. Usabilidade

- **RNF007**: A *UI* do dashboard web deve ser intuitiva e de fácil compreensão para usuários com conhecimento em redes.
- **RNF008**: Os dados de tráfego da rede devem ser apresentados de forma clara e organizada para fácil visualização.
- **RNF009**: A filtragem de dados e a inspeção de pacotes devem ser de fácil acesso e utilização.

3.2.4. Compatibilidade

- **RNF010**: O dashboard web deve ser compatível com as versões mais recentes dos navegadores mais utilizados no mercado (e.g. Google Chrome, Mozilla Firefox, Brave).
- **RNF011**: O backend em C++ deve ser multiplataforma, ou seja, compilável para o Kernel dos sistemas operacionais mais utilizados (e.g. Unix-based, Windows).

3.2.5. Confiabilidade

- **RNF012**: O sistema de captura de dados (backend) deve operar de forma estável e contínua, sem interrupções inesperadas.

- **RNF013**: O sistema deve apresentar *Exception handling*—i.e. lidar de forma adequada com possíveis erros de captura de dados da rede (e.g. falta de permissão na interface de rede), informando o usuário sobre o problema.

3.2.6. Tecnologia

- **RNF014**: O backend do sistema deve ser desenvolvido em C++ para garantir alto desempenho na manipulação de pacotes.
- **RNF015**: A integração e comunicação entre o frontend e o backend em C++ deve ser realizada utilizando WebAssembly.

4. Protótipo de Interface

4.1. Visão geral

- Frontend: Next.js + Supabase (autenticação).
- URL de demonstração: netprobe.vercel.app
- Objetivo do protótipo: demonstrar autenticação, visualização em tempo real e inspeção básica de pacotes.

4.2. Fluxos demonstrados

- Login e cadastro de usuário.
- Monitoramento em tempo real: listagem/tabela de pacotes.
  - Observe que o protótipo usa dados simulados (mock) para demonstração.
- Filtragem por protocolo, IP, porta.
- Inspeção de pacote (detalhes de cabeçalhos).

4.3. Evidências visuais

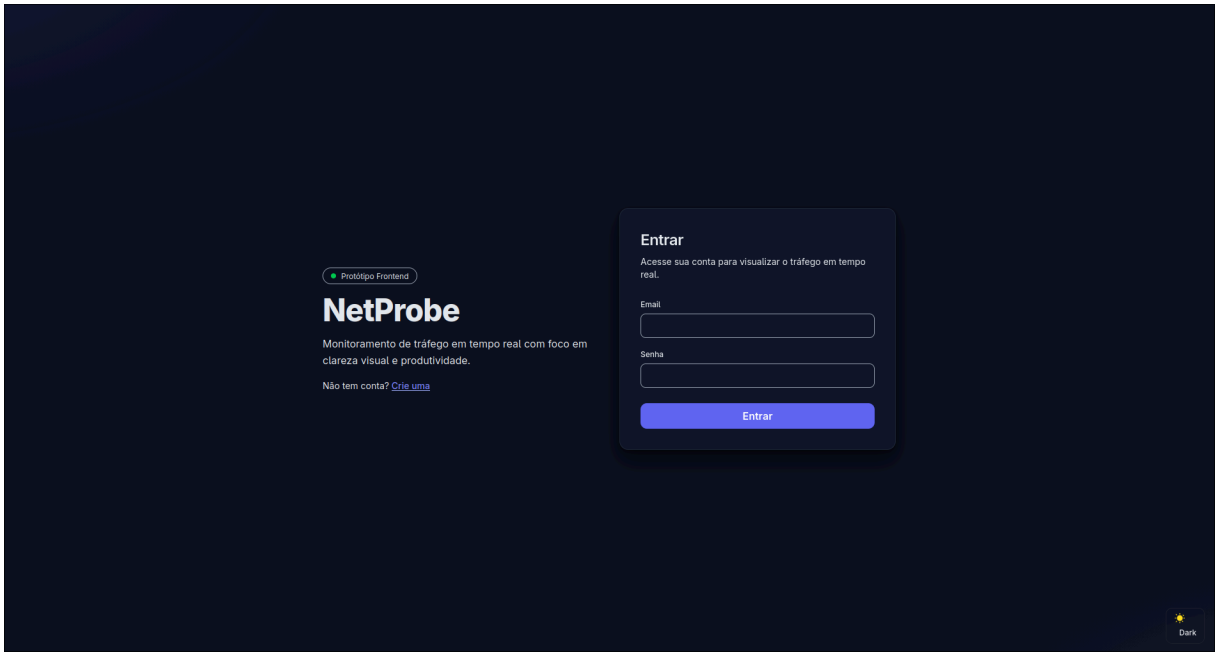

Figura 1: Tela de login. Fonte: Autor.

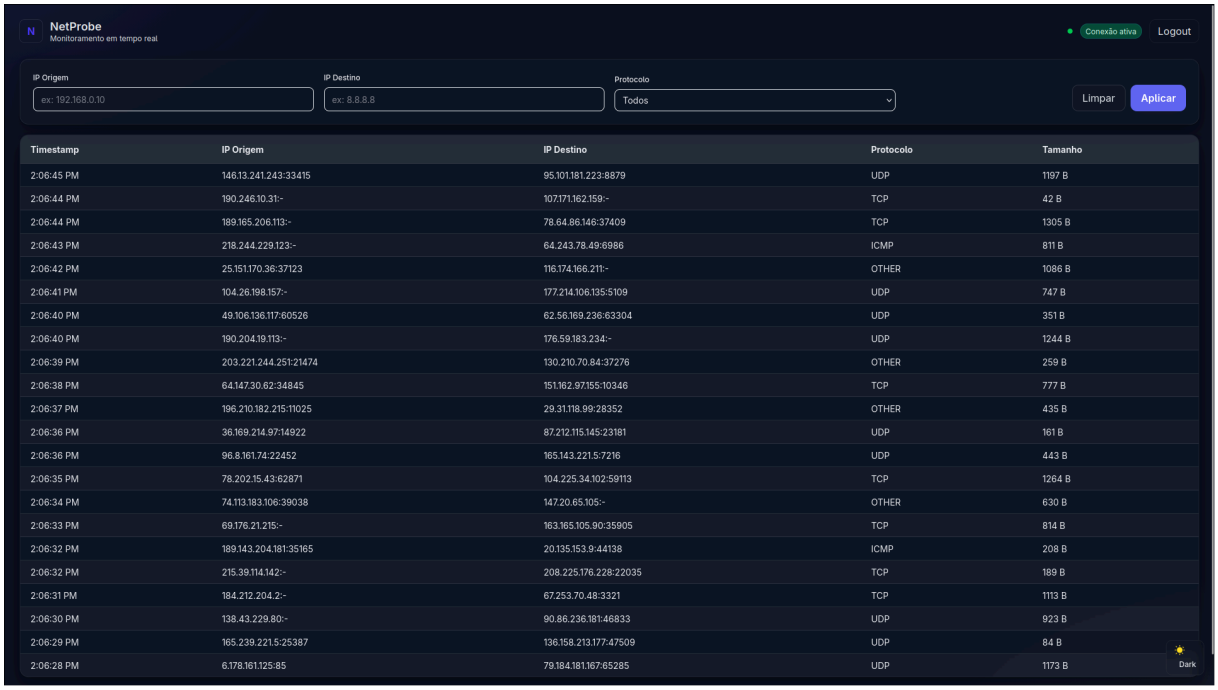

Figura 2: Dashboard em tempo real. Fonte: Autor.

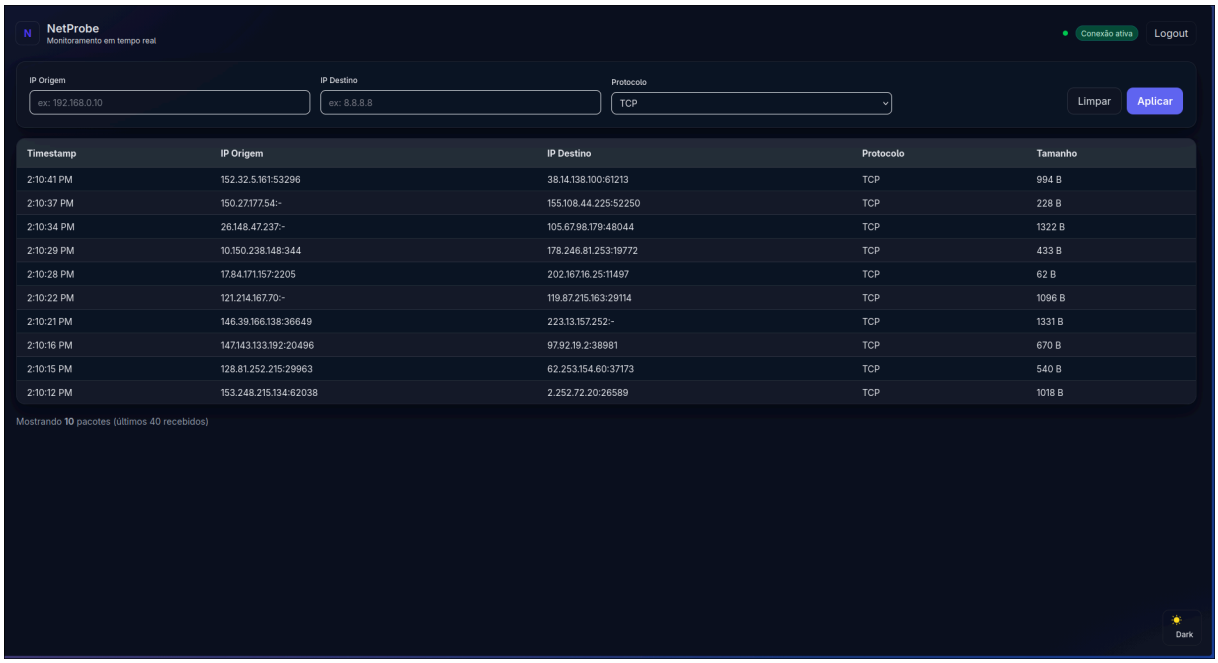

Figura 3: Filtro aplicado. Fonte: Autor.

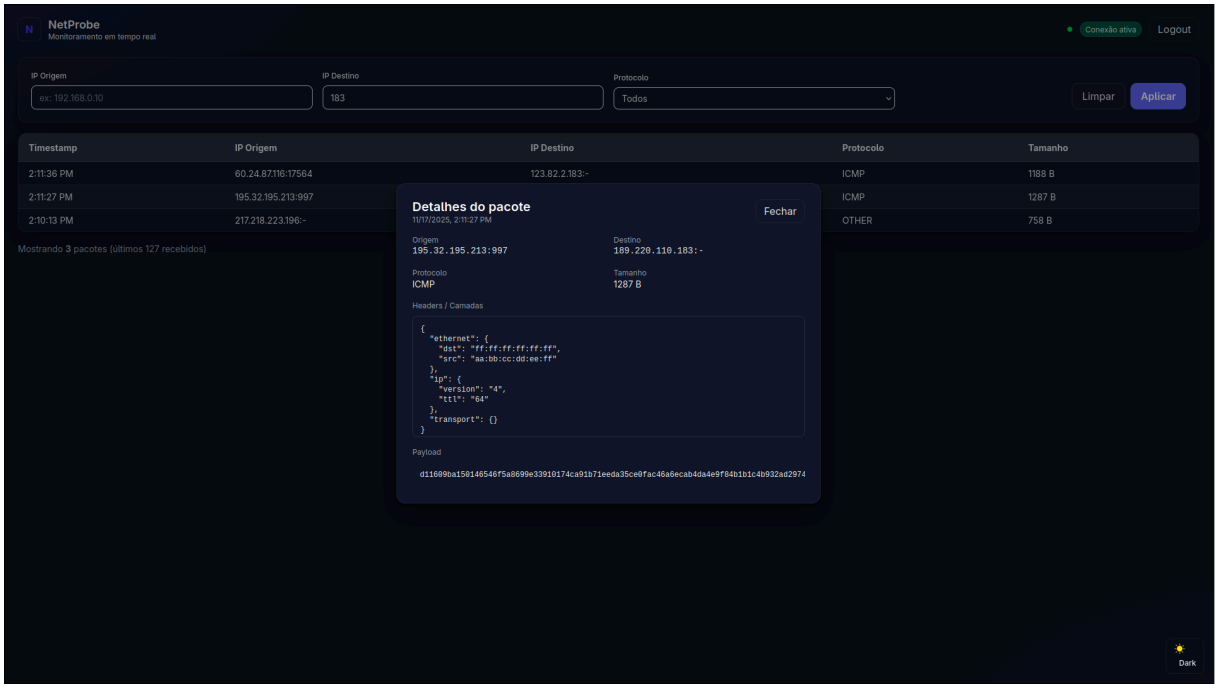

Figura 4: Detalhes do pacote. Fonte: Autor.

4.4. Limitações atuais

- Protótipo sem paginação/otimizações de UI para grandes volumes.
- Cobertura parcial de protocolos na visualização de detalhes.

4.5. Protótipo de Backend (Prova de Conceito)

- Escopo: captura de pacotes em ambiente local, fornecendo metadados (IP origem/destino, portas, protocolo, tamanho, e payload).
- Ambiente: Arch Linux x86_64, interface `enp3s0`, permissões via `sudo`.
- Resultado: captura de pacotes bem-sucedida.
- Evidências: amostras de registros de captura e contadores (pacotes/s, latência média).
- Limitações: suporte parcial a protocolos; ausência de persistência/armazenamento histórico.
- Próximos passos: métricas RNF (latência, CPU/RAM), robustez (reconexão e tratamento de erros), testes com `.pcap`.

4.6. Relação com requisitos de qualidade

- Desempenho (RNF001–RNF003): registrar tempos de atualização e uso de recursos.
- Usabilidade (RNF007–RNF009): validar clareza da UI nos fluxos acima.
- Compatibilidade (RNF010–RNF011): testes em navegadores e SOs alvo.
- Confiabilidade (RNF012–RNF013): execução contínua e tratamento de falhas.