



Escola Profissional
BENTO DE JESUS CARAÇA
DELEGAÇÃO DO BARREIRO

PROVA DE APTIDÃO PROFISSIONAL

CURSO PROFISSIONAL DE GESTÃO E PROGRAMAÇÃO DE SISTEMAS INFORMÁTICOS

CICLO DE FORMAÇÃO 2022/2025

Prova de Aptidão
Profissional



APOIO E SUPORTE INFORMÁTICO

Julho de 2025

Ba2490 – Ander Bogalho

Declaro que este trabalho se encontra em condições de ser apresentado a provas públicas.

O Professor Orientador

Barreiro, Julho de 2025

Agradecimentos

O aluno gostaria de agradecer à Escola Profissional Bento de Jesus Caraça pelo acolhimento e pela oportunidade que lhe foi concedida de frequentar o curso de Gestão e Programação de Sistemas Informáticos ao longo dos três anos de formação. Esta experiência foi fundamental para o seu crescimento académico e profissional, permitindo-lhe adquirir conhecimentos e competências que moldaram o seu percurso. O aluno agradece igualmente ao Diretor de Turma, professor Marcelo Simão, pela ajuda prestada durante o tempo em que desempenhou funções como coordenador de turma neste último ano, sempre com dedicação e apoio, que fizeram toda a diferença no seu desenvolvimento.

Índice

1 Introdução	1
1.1 Fundamentação da escolha do Projeto	1
1.2 Finalidades do Projeto	2
1.3 Enquadramento do Projeto	3
1.4 Cronograma	3
2 Projeto PAP – (Monitorização de Redes)	5
2.1 Ferramentas e linguagens utilizadas	6
2.2 Etapas e funcionalidades	8
3 Conclusão - análise crítica global da execução do projeto	16
3.1 Dificuldades	16
3.2 Problemas e obstáculos	16
3.3 Soluções encontradas	17
Anexos	18

Índice de Ilustrações

Imagem 1 - Interface principal do "Monitor de Rede"	10
Imagem 2 - Saída no terminal	11
Imagem 3 - Captura de pacotes	12
Imagem 4 - Topologia de rede	13
Imagem 5 - Alerta de colisão detetada	14
Imagem 6 - Janela de colisões detetadas	15

1 Introdução

A Prova de Aptidão Profissional (PAP) constitui um elemento essencial no percurso formativo dos alunos do curso profissional de Gestão e Programação de Sistemas Informáticos (GPSI), uma vez que representa a consolidação dos conhecimentos e das competências técnicas adquiridos ao longo dos três anos de formação. Este projeto não só demonstra a capacidade de aplicar os saberes em contextos reais, como também evidencia a maturidade profissional e a autonomia na resolução de problemas na área das tecnologias de informação.

No contexto do GPSI, a PAP assume uma importância particular, pois permite ao aluno integrar conceitos de programação, bases de dados, redes e gestão de sistemas, desenvolvendo um trabalho prático que pode ir desde a criação de uma aplicação funcional até à implementação de uma solução tecnológica inovadora. Para além disso, este projeto funciona como uma ponte entre o meio académico e o mercado de trabalho, preparando o aluno para os desafios do setor e valorizando o seu perfil junto de potenciais empregadores.

1.1 Fundamentação da escolha do Projeto

Ao longo do curso de Gestão e Programação de Sistemas Informáticos (GPSI), o aluno demonstrou um interesse particular pela área das redes de computadores, destacando-se pela sua curiosidade em compreender a complexidade técnica e a relevância destas nos sistemas atuais. A interligação de dispositivos, a segurança dos dados e a otimização do tráfego de rede foram temas que o cativaram desde o início da formação, levando-o a aprofundar os seus conhecimentos nesta área.

Foi neste contexto que surgiu a ideia de desenvolver um projeto relacionado com a monitorização de redes, aproveitando a infraestrutura disponível na escola, nomeadamente os computadores ligados em rede no laboratório de informática. Esta estrutura permitiu-lhe testar e validar conceitos na prática, conferindo um carácter mais realista e aplicado ao trabalho. Além disso, o Diretor de Turma, ao lançar o desafio de criar uma aplicação do zero, motivou-o a ir além do básico e a desenvolver uma solução que fosse não só funcional, mas também útil num ambiente real.

1.2 Finalidades do Projeto

Este projeto teve como objetivos principais:

- Simplificar a monitorização de redes para os serviços de informática ou equipas de gestão de TI, proporcionando uma visão unificada dos equipamentos ligados e do tráfego da rede;
- Automatizar processos manuais de deteção de anomalias e identificação de dispositivos, reduzindo o tempo necessário para intervenções técnicas;
- Reforçar a segurança da rede através de sistemas de autenticação de utilizadores e registo de atividades, prevenindo acessos não autorizados;
- Armazenar e organizar dados técnicos de forma centralizada, tornando a análise histórica e a produção de relatórios mais simples.

1.3 Enquadramento do Projeto

O projeto insere-se no domínio da administração de redes, destacando-se pelas seguintes características fundamentais:

- Monitorização simplificada - Interface intuitiva que consolida informação sobre dispositivos e tráfego de rede num único painel de controlo;
- Registo automático de dados - Armazenamento centralizado de eventos e acessos, eliminando a necessidade de registos manuais;
- Alertas e deteção de anomalias - Sistema de notificações em tempo real para colisões ou atividades suspeitas na rede;
- Acesso seguro e hierarquizado - Mecanismo de autenticação de utilizadores com diferentes níveis de autorização, garantindo a proteção dos dados;
- Base de dados consultável - Histórico organizado que permite pesquisas rápidas e auditorias regulares.

1.4 Cronograma

2º Feira	4º Feira	5º Feira	2º Feira	4º Feira	5º Feira	2º Feira	4º Feira	5º Feira	2º Feira	4º Feira	5º Feira
	16/10/ 2024	17/10/ 2024	21/10/ 2024	23/10/ 2024	24/10/ 2024	28/10/ 2024	30/10/ 2024	31/10/ 2024	4/11/20 24	6/11/20 24	7/11/20 24
	Pesquisa e implementação de funcionalidades das bibliotecas na aplicação	Relatório das pesquisas efetuadas e mudanças no código	Esclarecimento de dúvidas e implementação de código juntamente com a interface da aplicação	Pesquisa e implementação de funcionalidades das bibliotecas na aplicação	REL	Esclarecimento de dúvidas e implementação de código juntamente com a interface da aplicação	Pesquisa e implementação de funcionalidades das bibliotecas na aplicação	REL	Esclarecimento de dúvidas e implementação de código juntamente com a interface da aplicação	Implementação de funcionalidades no código e testes na rede	REL
11/11/2 024	13/11/2 024	14/11/2 024	18/11/2 024	20/11/2 024	21/11/2 024	25/11/2 024	27/11/2 024	28/11/2 024	2/12/2 024	4/12/2 024	5/12/2 024
Esclarecimento de dúvidas e implementação de código juntamente com a interface da aplicação	Implementação de funcionalidades no código e testes na rede	REL	Esclarecimento de dúvidas e implementação de código juntamente com a interface da aplicação	Implementação de funcionalidades no código e testes na rede	REL	Esclarecimento de dúvidas e implementação de código juntamente com a interface da aplicação	Implementação de funcionalidades no código e testes na rede	REL	Implementar a aplicação em outros sistemas operativos ou em outras redes	Implementação de funcionalidades no código e testes na rede	REL

9/12/2024	11/12/2024	12/12/2024	16/12/2024	18/12/2024	19/12/2024	23/12/2024	25/12/2024	26/12/2024	30/12/2024	1/1/2025	2/1/2025
Implementar a aplicação em outros sistemas operativos ou em outras redes	Implementação de funcionalidades no código e testes na rede	REL	Implementar a aplicação em outros sistemas operativos ou em outras redes	Implementação de funcionalidades no código e testes na rede	REL	Implementar a aplicação em outros sistemas operativos ou em outras redes	Implementação de funcionalidades no código e testes na rede	REL	Implementar a aplicação em outros sistemas operativos ou em outras redes	Implementação de funcionalidades no código e testes na rede	REL
6/1/2025	8/1/2025	9/1/2025	13/1/2025	15/1/2025	16/1/2025	20/1/2025	22/1/2025	23/1/2025	27/1/2025	29/1/2025	30/1/2025
Preparação do ambiente onde mostrar a aplicação na apresentação da PAP	Implementação de funcionalidades no código e testes na rede	REL	Preparação do ambiente onde mostrar a aplicação na apresentação da PAP	Implementação de funcionalidades no código e testes na rede	REL	Preparação do ambiente onde mostrar a aplicação na apresentação da PAP	Implementação de funcionalidades no código e testes na rede	REL	Preparação do ambiente onde mostrar a aplicação na apresentação da PAP	Implementação de funcionalidades no código e testes na rede	REL
3/2/2025	5/2/2025	6/2/2025	10/2/2025	12/2/2025	13/2/2025	17/2/2025	19/2/2025	20/2/2025	24/2/2025	26/2/2025	27/2/2025
Ajustes no código, organização e otimizar aplicação	Implementação de funcionalidades no código e testes na rede	REL	Ajustes no código, organização e otimizar aplicação	Implementação de funcionalidades no código e testes na rede	REL	Ajustes no código, organização e otimizar aplicação	Implementação de funcionalidades no código e testes na rede	REL	Ajustes no código, organização e otimizar aplicação	Implementação de funcionalidades no código e testes na rede	REL
3/3/2025	5/3/2025	6/3/2025	10/3/2025	12/3/2025	13/3/2025						
Ajustes no código, organização e otimizar aplicação	Implementação de funcionalidades no código e testes na rede	REL	Ajustes no código, organização e otimizar aplicação	Implementação de funcionalidades no código e testes na rede	REL						

2 Projeto PAP – (Monitorização de Redes)

O projeto desenvolvido consiste numa aplicação integrada de monitorização e análise de redes, concebida para oferecer uma solução completa e intuitiva para a gestão de infraestruturas de rede em ambientes educativos ou empresariais de pequena dimensão.

A aplicação possui uma interface gráfica desenvolvida com *Tkinter*, que permite ao utilizador interagir de forma simples e eficiente com todas as funcionalidades disponíveis. Através desta interface, é possível iniciar e parar a monitorização da rede, visualizar os dispositivos conectados em tempo real, analisar o tráfego de dados e consultar históricos de eventos registados. A organização por separadores facilita a navegação entre as diferentes opções, tornando a experiência do utilizador mais fluída e acessível.

No núcleo da aplicação, o módulo de monitorização, implementado com a biblioteca *Scapy*, é responsável por realizar rastreamentos automáticos da rede, identificando dispositivos ativos e recolhendo informações como endereços *IP*, *MAC* e *hostnames*. Este módulo inclui ainda funcionalidades avançadas para a deteção da topologia da rede e identificação de anomalias ou colisões, contribuindo para uma gestão mais proativa da infraestrutura.

O sistema de captura de pacotes, também baseado na *Scapy*, permite a captura e análise em tempo real do tráfego de rede. Com capacidade de filtragem por protocolo e inspeção de conteúdo, esta funcionalidade é essencial para diagnósticos técnicos e otimização do desempenho da rede.

Todos os dados recolhidos são armazenados numa base de dados *SQLite*, que centraliza informações sobre dispositivos, eventos e registos de segurança. Esta base de dados não só facilita consultas rápidas e a geração de relatórios, como também suporta auditorias periódicas, garantindo a rastreabilidade de todas as atividades realizadas.

A segurança foi uma prioridade no desenvolvimento do projeto, pelo que a aplicação integra um módulo de segurança robusto. Este módulo inclui autenticação de utilizadores com palavras-passe protegidas por hash, encriptação de dados sensíveis através da biblioteca *Cryptography* e registo detalhado de todas as ações realizadas. Estas medidas garantem a confidencialidade e integridade dos dados, prevenindo acessos não autorizados.

2.1 Ferramentas e linguagens utilizadas

O desenvolvimento deste projeto recorreu a diversas ferramentas e uma linguagem de programação, cada uma desempenhando um papel específico na implementação das diferentes funcionalidades do sistema:

Python:

Como principal linguagem de desenvolvimento, *Python* foi escolhida pela sua versatilidade, vasta coleção de bibliotecas e sintaxe clara. Serviu como base para integrar todos os componentes do sistema.

Scapy:

Biblioteca essencial para operações de rede, utilizada para:

- Realizar rastreamentos de rede e deteção de dispositivos;
- Capturar e analisar pacotes de dados;
- Implementar funcionalidades de monitorização em tempo real.

Tkinter:

Biblioteca utilizada para o desenvolvimento da interface gráfica, permitindo:

- Criar uma interface intuitiva e acessível;
- Apresentar dados de forma organizada;
- Facilitar a interação do utilizador com todas as funcionalidades.

SQLite:

Sistema de gestão de bases de dados relacional, empregue para:

- Armazenar informações sobre dispositivos e eventos;
- Manter registos históricos para consulta;
- Garantir persistência dos dados recolhidos.

Cryptography:

Biblioteca de criptografia utilizada para:

- Proteger dados sensíveis;
- Implementar mecanismos de segurança;
- Garantir a confidencialidade das informações.

Git:

Sistema de controlo de versões utilizado para:

- Gerir diferentes versões do projeto;
- Facilitar o trabalho colaborativo;
- Manter um histórico de alterações.

2.2 Etapas e funcionalidades

O desenvolvimento do projeto foi planeado seguindo uma sequência lógica que permitiu a construção progressiva de todas as funcionalidades propostas. Abaixo, descrevem-se detalhadamente as etapas do projeto, acompanhadas de uma explicação dos respetivos objetivos e das razões pelas quais certas decisões foram tomadas, sempre procurando justificar as opções escolhidas com base nas necessidades do sistema e nas melhores práticas de desenvolvimento. Quando aplicável, são fornecidos exemplos que ilustram as escolhas, como o uso de um menu lateral para melhorar a legibilidade e a interação do utilizador. Além disso, as imagens fornecidas documentam o funcionamento da aplicação, destacando as suas funcionalidades principais.

Configurações e Segurança: A primeira etapa envolveu a implementação de um sistema robusto de segurança para proteger os dados sensíveis manipulados pelo projeto, como endereços *IP* e endereços *MAC*. Para isso, foi utilizada a biblioteca *cryptography.Fernet*, que permite a geração e a gestão de chaves de criptografia simétrica. A abordagem incluiu a geração automática de uma chave única em caso de ausência (4.1), seguida pelo seu carregamento (4.2), garantindo que o sistema pudesse aceder aos dados criptografados. A escolha do *Fernet* justificou-se pela sua simplicidade de uso combinada com um elevado nível de segurança, ideal para um projeto que lida com informações de rede sensíveis. Esta etapa foi essencial para estabelecer uma base segura antes de qualquer manipulação de dados.

Configuração da Base de Dados de Utilizadores: Na segunda etapa, foi configurado um base de dados *SQLite* para gerir informações de utilizadores e os seus registos de acesso. A estrutura incluiu a criação de duas tabelas (4.3), uma para armazenar os utilizadores e outra para registar os registos de acesso. A autenticação foi implementada utilizando a biblioteca *hashlib* para criptografar as palavras-passe, garantindo que fossem armazenadas de forma segura e não reversível. Esta decisão foi tomada para atender aos requisitos de segurança e conformidade, permitindo rastrear atividades dos utilizadores, como logins bem-sucedidos ou falhas, o que foi considerado crucial para auditoria e depuração.

Configuração da Base de Dados de Dispositivos: Em seguida, foi estruturado um segundo base de dados, *network_devices.db*, para gerir informações sobre dispositivos de rede e colisões detetadas. A estrutura incluiu a criação de tabelas (4.4) para dispositivos e colisões, com dados criptografados (4.5) e descriptografados (4.6), escolhendo esta abordagem para proteger informações de rede sensíveis contra acessos não autorizados. Esta etapa foi justificada pela necessidade de organizar os dados recolhidos durante o rastreio e monitorização, tornando as consultas e atualizações em tempo real mais simples.

Funções de Rede: A quarta etapa concentrou-se no desenvolvimento de funcionalidades para interagir com a rede. A abordagem incluiu varreduras *ARP* (4.7) para identificar dispositivos ativos, verificação da conectividade e latência utilizando o comando ping (4.8), e resolução de nomes de host (4.9). Estas escolhas foram motivadas pela necessidade de ferramentas eficientes e amplamente testadas para análise de rede, permitindo um diagnóstico preciso e em tempo real, essencial para a monitorização contínua proposta pelo projeto.

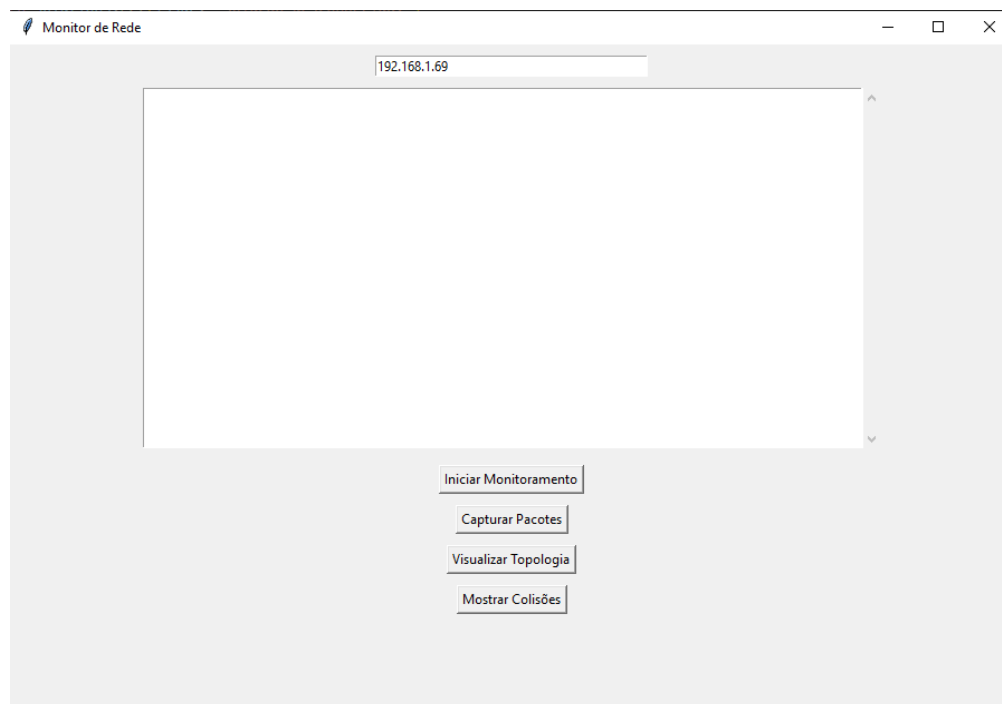
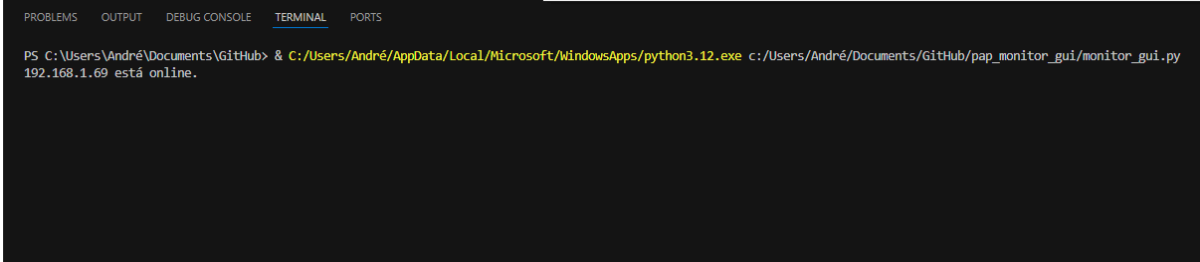


Imagem 1: Interface principal do "Monitor de Rede"

A imagem 1 mostra a interface principal do "Monitor de Rede", onde o utilizador pode inserir um intervalo de *IP* (neste caso, "192.168.1.69") e iniciar as funcionalidades disponíveis, como a monitorização, captura de pacotes, visualização da topologia e exibição de colisões, demonstrando a integração destas funções.

Monitorização e Detecção: Nesta etapa, foram implementadas as funcionalidades de monitorização e deteção de colisões. A análise incluiu a identificação de problemas como alta latência ou inacessibilidade (4.10), com sugestões de soluções (4.11). A captura de pacotes foi realizada de forma assíncrona (4.12), utilizando threading para evitar bloqueios na interface gráfica. Esta abordagem foi escolhida para garantir que o sistema pudesse operar de forma assíncrona, mantendo a responsividade da aplicação enquanto processava dados da rede, um requisito crítico para a usabilidade.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\André\Documents\GitHub> & C:/Users/André/AppData/Local/Microsoft/WindowsApps/python3.12.exe c:/Users/André/Documents/GitHub/pap_monitor_gui/monitor_gui.py
192.168.1.69 está online.
```

Imagem 2: Saída no terminal

A imagem 2 apresenta uma captura de saída no terminal, confirmando que o dispositivo "192.168.1.69" está online, validando a funcionalidade de verificação de conectividade. A autoria dos módulos de captura e análise de pacotes foi desenvolvida pelo aluno, e seria útil solicitar uma demonstração da interface para verificar o desempenho em tempo real. Além disso, levanta-se a questão sobre limitações em ambientes com firewalls, que podem bloquear certas operações de rede.

Gestão de Dados: A gestão de dados foi abordada com funcionalidades para rastrear a rede e guardar dispositivos (4.13), além de exibir colisões registadas numa janela gráfica com *scrolledtext* (4.14). A decisão de usar *scrolledtext* foi tomada para oferecer uma melhor leitura das informações, permitindo que o utilizador visualizasse registos extensos sem sobrecarregar a interface, justificando-se pela necessidade de uma apresentação clara e acessível dos dados recolhidos.

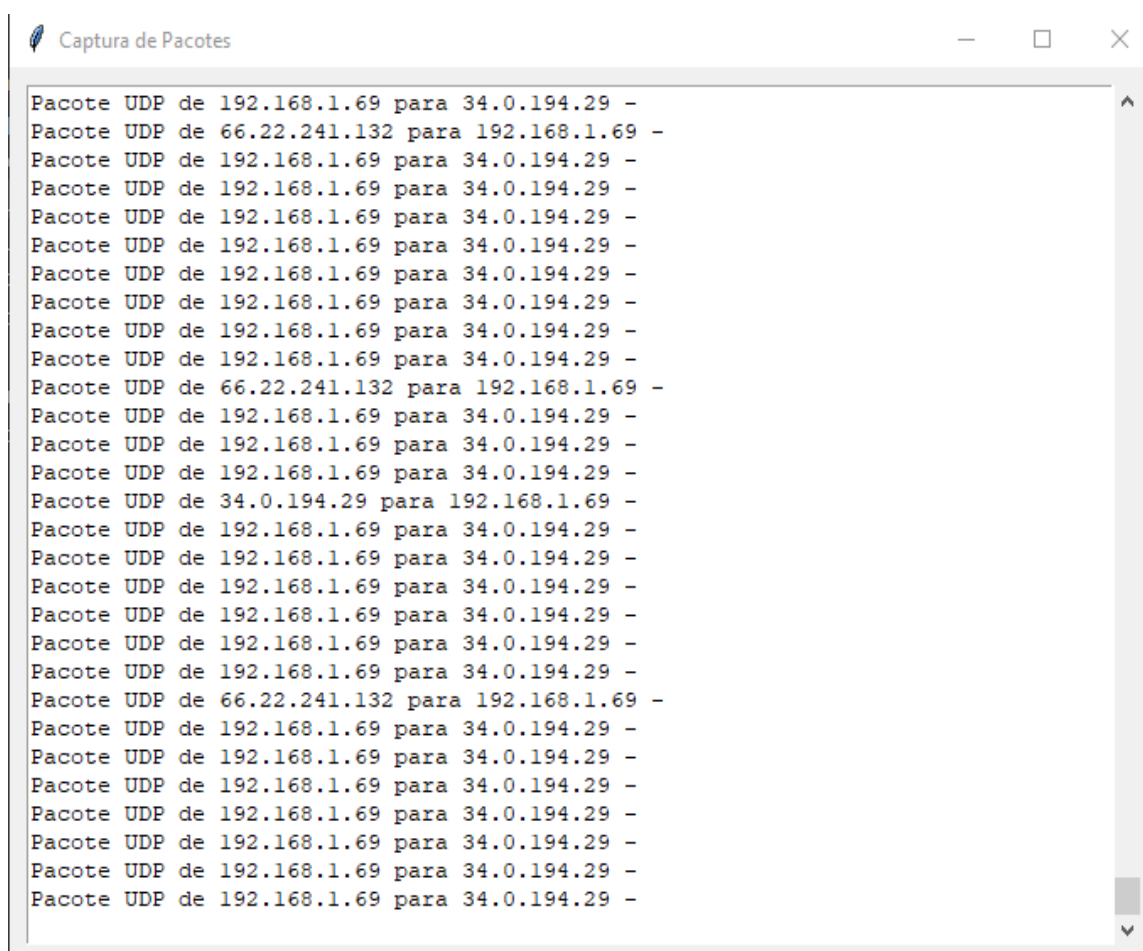


Imagem 3: Captura de pacotes

A imagem 3 mostra a janela de captura de pacotes, exibindo uma lista de pacotes *UDP* detetados, o que destaca a funcionalidade de captura em ação.

Topologia de Rede: A visualização da topologia foi desenvolvida para oferecer uma representação visual intuitiva, utilizando um grafo (4.15) que incluía o router como nó

central e dispositivos como nós conectados, com cores distintas para diferenciação. Esta escolha foi motivada pela vontade de ajudar o utilizador a compreender a estrutura da rede de forma mais eficaz, um diferencial importante para análise avançada.

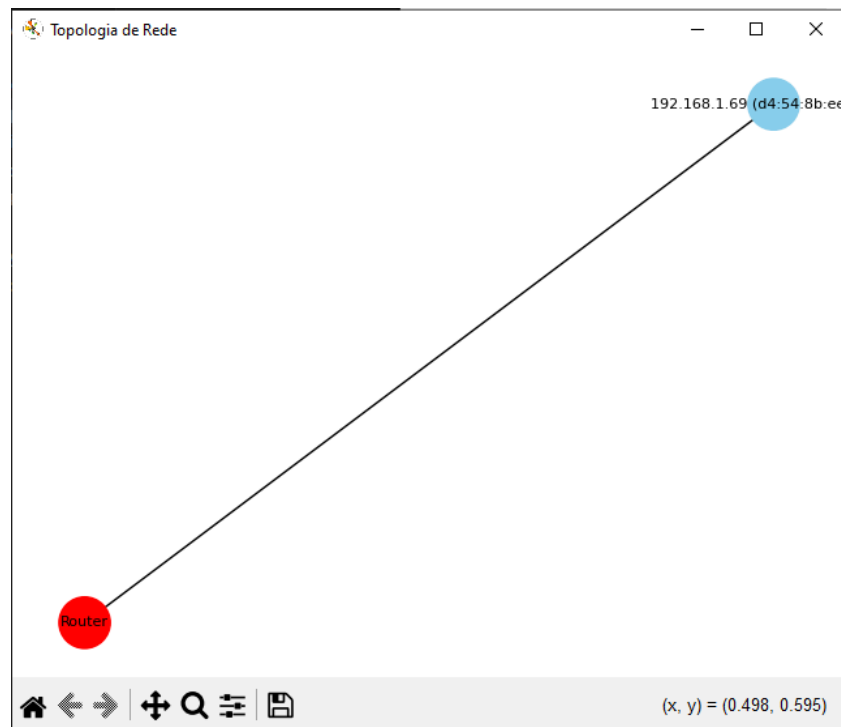


Imagem 4: Topologia de rede

A imagem 4 apresenta a topologia de rede, mostrando uma ligação entre o router e o dispositivo "192.168.1.69", ilustrando a funcionalidade de visualização gráfica.

Interface de Autenticação e Principal: Por fim, foi criada a interface gráfica principal com *tkinter*, incluindo ecrãs de login e registo, e um painel de controlo com botões para monitorização, captura de pacotes, visualização de topologia e colisões (4.16). O uso de um menu lateral, implementado via disposição dos botões, permitiu

uma melhor leitura das opções, tornando a navegação e a interação do utilizador mais simples. Esta decisão foi justificada pela melhoria na experiência do utilizador, tornando o sistema mais organizado e acessível, especialmente para operações frequentes como iniciar a monitorização ou visualizar registos.

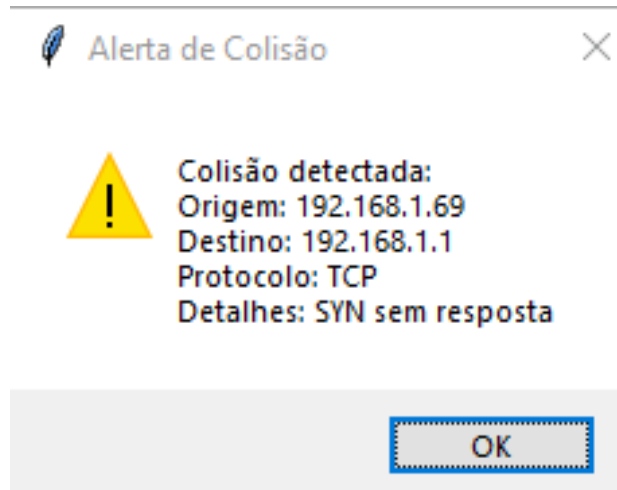
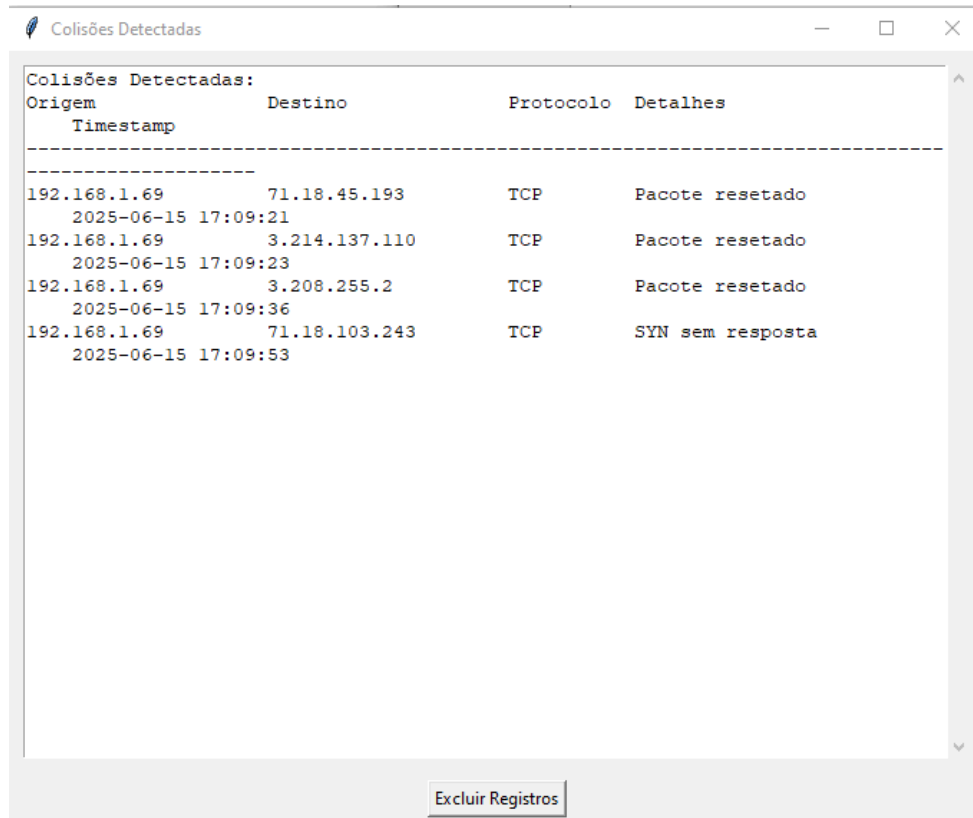


Imagem 5: Alerta de colisão detetada

A imagem 5 exibe um alerta de colisão detetada, indicando origem, destino, protocolo e detalhes, o que valida a funcionalidade de deteção de colisões.



Origem	Timestamp	Destino	Protocolo	Detalhes
192.168.1.69	2025-06-15 17:09:21	71.18.45.193	TCP	Pacote resetado
192.168.1.69	2025-06-15 17:09:23	3.214.137.110	TCP	Pacote resetado
192.168.1.69	2025-06-15 17:09:36	3.208.255.2	TCP	Pacote resetado
192.168.1.69	2025-06-15 17:09:53	71.18.103.243	TCP	SYN sem resposta

Excluir Registros

Imagem 6: Janela de colisões detetadas

A imagem 6 mostra a janela de colisões detetadas, listando eventos com timestamp, origem, destino, protocolo e detalhes, reforçando a capacidade de registo e análise.

3 Conclusão - análise crítica global da execução do projeto

O desenvolvimento do projeto "Monitor de Rede" foi o ponto alto dos três anos do aluno no curso de Gestão e Programação de Sistemas Informáticos na Escola Profissional Bento de Jesus Caraça. Esta última parte, a PAP, deu ao aluno a oportunidade de aplicar os conhecimentos adquiridos, explorar as suas capacidades e concretizar tudo num ambiente que lhe deu grande apoio. O aluno trabalhou arduamente e conseguiu transformar ideias em algo funcional, mas reconhece que há aspetos a melhorar, como a adaptação a diferentes contextos e a realização de mais testes. Este momento final levou o aluno a refletir bastante, evidenciando a importância da persistência e da capacidade de adaptação. O projeto é visto pelo aluno como um ponto de partida para evoluir e contribuir para a área da programação e da gestão de sistemas informáticos.

3.1 Dificuldades

No início, o aluno deparou-se com muitas dificuldades devido à escassez de informação disponível para investigação sobre este projeto. A falta de recursos tornou o arranque mais desafiante, exigindo que o aluno procurasse outras soluções e aprendesse de forma autónoma desde o início.

3.2 Problemas e obstáculos

Durante o desenvolvimento, o aluno enfrentou diversos obstáculos. Surgiram vários erros e complicações ao tentar otimizar a aplicação e executá-la noutros sistemas operativos, o que obrigou a ajustes constantes. Para além disso, detetar colisões ou dispositivos revelou-se difícil devido à complexidade da rede da escola, o que dificultou a execução da aplicação. O aluno teve de reestruturar várias partes do código para resolver os erros encontrados, mas superou essas dificuldades com grande esforço e com o apoio e sugestões do Diretor de Turma.

3.3 Soluções encontradas

Com dedicação e persistência, o aluno conseguiu encontrar soluções para os problemas enfrentados. As sugestões do professor Marcelo foram essenciais para corrigir erros e aperfeiçoar o código, enquanto o esforço contínuo permitiu ao aluno otimizar a aplicação e adaptá-la, apesar dos desafios colocados pela rede extensa da escola.

Anexos

- 4.1:

```
def generate_and_save_key():  
    with open("encryption_key.key", "wb") as key_file:  
        key = Fernet.generate_key()  
        key_file.write(key)
```

- 4.2:

```
def load_key():  
    with open("encryption_key.key", "rb") as key_file:  
        return key_file.read()
```

- 4.3:

```
def initialize_authentication_db():  
    connection = sqlite3.connect("auth.db")  
    cursor = connection.cursor()  
    cursor.execute("""  
        CREATE TABLE IF NOT EXISTS users (  
            id INTEGER PRIMARY KEY AUTOINCREMENT,  
            username TEXT UNIQUE NOT NULL,  
            password TEXT NOT NULL  
        )  
    """)  
    cursor.execute("""  
        CREATE TABLE IF NOT EXISTS access_logs (  
            id INTEGER PRIMARY KEY AUTOINCREMENT,  
            username TEXT,  
            action TEXT,  
            timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
        )  
    """)  
    connection.commit()
```

```
connection.close()
```

- **4.4:**

```
def initialize_database():  
    connection = sqlite3.connect("network_devices.db")  
    cursor = connection.cursor()  
    cursor.execute("""  
        CREATE TABLE IF NOT EXISTS devices (  
            id INTEGER PRIMARY KEY AUTOINCREMENT,  
            ip TEXT NOT NULL UNIQUE,  
            mac TEXT NOT NULL,  
            hostname TEXT NOT NULL,  
            last_seen TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
        )  
    """)  
    cursor.execute("""  
        CREATE TABLE IF NOT EXISTS collisions (  
            id INTEGER PRIMARY KEY AUTOINCREMENT,  
            source_ip TEXT NOT NULL,  
            destination_ip TEXT NOT NULL,  
            protocol TEXT NOT NULL,  
            details TEXT,  
            timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
        )  
    """)  
    connection.commit()  
    connection.close()
```

- **4.5:**

```
def encrypt_data(data):  
    return cipher_suite.encrypt(data.encode())
```


- **4.6:**

```
def decrypt_data(encrypted_data):  
    try:  
        return cipher_suite.decrypt(encrypted_data).decode()  
    except (cryptography.fernet.InvalidToken, AttributeError):  
        return encrypted_data
```

- **4.7:**

```
def scan_network(ip_range):  
    arp_request = scapy.ARP(pdst=ip_range)  
    broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")  
    arp_request_broadcast = broadcast / arp_request  
    answered_list = scapy.srp(arp_request_broadcast, timeout=1,  
verbose=False)[0]  
    devices = []  
    for element in answered_list:  
        device_info = {'ip': element[1].psrc, 'mac': element[1].hwsrc,  
'hostname': get_hostname(element[1].psrc)}  
        devices.append(device_info)  
    return devices
```

- **4.8:**

```
def check_ping(host):  
    response = subprocess.run(['ping', '-n', '4', host],  
stdout=subprocess.PIPE, stderr=subprocess.PIPE)  
    if response.returncode == 0:  
        output = response.stdout.decode()  
        return True, output  
    else:  
        return False, response.stderr.decode()
```

- **4.9:**

```
def get_hostname(ip):  
    try:  
        hostname = socket.gethostbyaddr(ip)[0]  
    except socket.herror:  
        hostname = "Desconhecido"  
    return hostname
```

- **4.10:**

```
def detect_collisions(devices, output_text):  
    output_text.insert(tk.END, "Simulando colisões na rede...\n")  
    collision_detected = False  
    for device in devices:  
        status_text, issue = analyze_device_status(device)  
        if issue == 'high_latency' or issue == 'device_unreachable':  
            collision_detected = True  
            output_text.insert(tk.END, f"Colisão detectada: {device['ip']}  
- {status_text}\n")  
            output_text.insert(tk.END,  
suggest_solution('collision_detected') + "\n")  
        if not collision_detected:  
            output_text.insert(tk.END, "Nenhuma colisão detectada.\n")  
    output_text.yview(tk.END)
```

- **4.11:**

```
def suggest_solution(issue):  
    solutions = {  
        'high_latency': "Sugestões: \n1. Verifique o router. \n2. Verifique a largura de banda.",  
        'packet_loss': "Sugestões: \n1. Verifique o cabo de rede. \n2. Tente reiniciar o dispositivo.",  
        'device_unreachable': "Sugestões: \n1. Verifique as conexões do dispositivo. \n2. Reinicie o dispositivo.",  
        'collision_detected': "Sugestões: \n1. Reduza dispositivos a competir pelo mesmo canal. \n2. Otimize a largura de banda.",  
        'normal': "Nenhum problema detectado.",  
    }  
    return solutions.get(issue, "Solução desconhecida.")
```

- **4.12:**

```
def capture_packets():  
    capture_window = tk.Toplevel()  
    capture_window.title("Captura de Pacotes")  
    global capture_text  
    capture_text = scrolledtext.ScrolledText(capture_window, width=80, height=30)  
    capture_text.pack(padx=10, pady=10)  
    capture_thread = threading.Thread(target=lambda: scapy.sniff(prn=packet_callback, store=False))  
    capture_thread.start()
```

- **4.13:**

```
def scan_and_save_network(ip_range):  
    devices = scan_network(ip_range)  
    for device in devices:  
        save_device_to_db(device)  
    return devices
```

- **4.14:**

```
def show_stored_collisions():
    connection = sqlite3.connect("network_devices.db")
    cursor = connection.cursor()
    cursor.execute("SELECT source_ip, destination_ip, protocol, details,
timestamp FROM collisions")
    collisions = cursor.fetchall()
    connection.close()
    collision_window = tk.Toplevel()
    collision_window.title("Colisões Detetadas")
    collision_text = scrolledtext.ScrolledText(collision_window, width=80,
height=30)
    collision_text.pack(padx=10, pady=10)
    collision_text.insert(tk.END, "Colisões Detetadas:\n")
    collision_text.insert(tk.END, f"{'Origem':<20} {'Destino':<20}
{'Protocolo':<10} {'Detalhes':<30} {'Timestamp'}\n")
    collision_text.insert(tk.END, "-" * 100 + "\n")
    for collision in collisions:
        source_ip, destination_ip, protocol, details, timestamp = collision
        collision_text.insert(tk.END, f"{source_ip:<20}
{destination_ip:<20} {protocol:<10} {details:<30} {timestamp}\n")
    collision_text.yview(tk.END)
    def delete_collisions():
        clear_collision_records()
        collision_text.delete(1.0, tk.END)
        collision_text.insert(tk.END, "Todos os registos de colisões foram
excluídos.\n")
        collision_text.yview(tk.END)
        delete_button = tk.Button(collision_window, text="Excluir Registos",
command=delete_collisions)
        delete_button.pack(pady=5)
```

- **4.15:**

```
def draw_network_topology():  
    devices = scan_network("10.5.50.254/24")  
    G = create_network_topology(devices)  
    pos = nx.spring_layout(G)  
    node_colors = [G.nodes[node].get('color', 'skyblue') for node in  
G.nodes()]  
    labels = nx.get_node_attributes(G, 'label')  
    plt.figure("Topologia de Rede")  
    nx.draw(G, pos, labels=labels, with_labels=True,  
node_color=node_colors, font_size=8, node_size=800)  
    plt.show()
```

- **4.16:**

```
def create_main_app(username=None):  
    window = tk.Tk()  
    window.title("Monitor de Rede")  
    frame = tk.Frame(window)  
    frame.pack(padx=10, pady=10)  
    ip_entry = tk.Entry(frame, width=40)  
    ip_entry.insert(0, "10.5.50.254/24")  
    ip_entry.pack()  
    output_text = scrolledtext.ScrolledText(frame, width=80, height=20)  
    output_text.pack(pady=10)  
    tk.Button(frame, text="Iniciar Monitorização",  
              command=lambda: threading.Thread(target=detect_collisions,  
              args=(scan_multiple_networks(ip_entry.get()),  
              output_text)).start()).pack(pady=5)  
    tk.Button(frame, text="Capturar Pacotes",  
              command=capture_packets).pack(pady=5)  
    tk.Button(frame, text="Visualizar Topologia",  
              command=draw_network_topology).pack(pady=5)  
    tk.Button(frame, text="Mostrar Colisões",  
              command=show_stored_collisions).pack(pady=5)  
    window.geometry("900x600")  
    window.mainloop()
```