

Controlador SDN

Licenciatura em Engenharia Informática

João Martins Tendeiro

Miguel Francisco Lopes

Leiria, Abril de 2025

Controlador SDN

Licenciatura em Engenharia Informática

João Martins Tendeiro

Miguel Francisco Lopes

Trabalho Laboratorial nº1 (TL1) da unidade curricular de Laboratório de Tecnologias de
Informação realizado sob a orientação do Professor Daniel Fuentes

Leiria, Abril de 2025

Resumo

Um controlador SDN (*Software Defined Networking*) é um software que gera uma rede definida por software, que permite o controlo centralizado do tráfego de rede.

Adquirimos um aparelho de *hardware* da marca MikroTik e realizámos a automação da rede via API (*Application Programming Interface*) REST.

Foi desenvolvido um controlador SDN utilizando C#, na plataforma Visual Studio 2022, o que proporcionou uma integração eficiente com outras tecnologias Microsoft, como o *.NET Framework* e *SQL (Structured Query Language) Server* (LocalDB) para a gestão de dados.

O nosso controlador SDN permite a gestão e automação de redes MikroTik. Oferece funcionalidades como listar, criar, editar e apagar interfaces de rede, incluindo interfaces wireless e bridge, além de permitir a gestão de perfis de segurança, ativar e configurar redes wireless, bem como a gestão de rotas estáticas, endereços IP (*Internet Protocol*) e servidores DHCP (*Dynamic Host Configuration Protocol*).

O controlador também permite ativar e desativar o servidor DNS (*Domain Name System*), e a gestão de DNS estáticos. Em termos de segurança, o controlador também oferece funcionalidades para a gestão de conexões VPN (*Virtual Private Network*) com suporte ao WireGuard, permitindo listar interfaces e *peers*, criar novos *peers* e gerar configurações de clientes com QR codes.

Em conclusão, o controlador SDN desenvolvido em C# oferece uma solução robusta e flexível para a gestão centralizada de redes MikroTik.

Palavras-chave: Controlador SDN, MikroTik, API REST, C#

Abstract

An SDN (Software Defined Networking) controller is software that manages a software-defined network, which allows centralized control of network traffic.

We acquired a MikroTik hardware device and performed network automation via API (Application Programming Interface) REST.

An SDN controller was developed using C#, on the Visual Studio 2022 platform, which provided efficient integration with other Microsoft technologies, such as the .NET Framework and SQL (Structured Query Language) Server (LocalDB) for data management.

Our SDN controller enables the management and automation of MikroTik networks. It provides features such as listing, creating, editing, and deleting network interfaces, including wireless and bridge interfaces. It also allows the management of security profiles, the activation and configuration of wireless networks, as well as the management of static routes, IP (Internet Protocol) addresses, and DHCP (Dynamic Host Configuration Protocol) servers.

The controller also allows the activation and deactivation of the DNS (Domain Name System) server, and the management of static DNS entries. In terms of security, it offers features for managing VPN (Virtual Private Network) connections with WireGuard support, allowing the listing of interfaces and peers, the creation of new peers, and the generation of client configurations with QR codes.

In conclusion, the SDN controller developed in C# offers a robust and flexible solution for the centralized management of MikroTik networks.

Keywords: SDN Controller, MikroTik, REST API, C#

Índice

Resumo	1
Abstract	2
Lista de Figuras	6
Lista de Tabelas	7
Lista de siglas e acrónimos	8
1 Introdução	1
2 Conceitos	2
2.1 Software Defined Networking (SDN)	2
2.2 Controlador SDN	3
2.3 MikroTik RouterOS	3
2.4 Application Programming Interface (API)	4
2.5 Representational State Transfer (REST) API	5
3 Trabalho Desenvolvido	6
3.1 Linguagem de Programação	7
3.2 Arquitetura da Solução	7
3.3 Base de Dados	8
3.4 Configuração de HTTPS no MikroTik	10
3.5 Lista de Endpoints	11
3.6 Login	14
3.7 Obtenção de dados da tabela Devices	15
3.8 Lista de todas as Interfaces	16
3.9 Interfaces Wireless	17

3.9.1	Listar	18
3.9.2	Ativar e Desativar	18
3.9.3	Editar	18
3.10	Interfaces Bridge e respectivas portas associadas	18
3.10.1	Listar	19
3.10.2	Criar	19
3.10.3	Editar	20
3.10.4	Apagar	20
3.11	Perfis de segurança para utilizar nas redes wireless	20
3.11.1	Criar	21
3.11.2	Editar	21
3.11.3	Apagar	22
3.12	Rotas estáticas	22
3.12.1	Listar	22
3.12.2	Criar	23
3.12.3	Editar	23
3.12.4	Apagar	23
3.13	Endereços IP	23
3.13.1	Listar	24
3.13.2	Criar	24
3.13.3	Editar	25
3.13.4	Apagar	25
3.14	Servidores de DHCP	25
3.14.1	Listar	26
3.14.2	Criar	27
3.14.3	Editar	27
3.14.4	Apagar	27

3.15 Servidor de DNS	27
3.15.1 Ativar	28
3.15.2 Desativar	28
3.15.3 Configurar	29
3.16 Wireguard	29
3.16.1 Listar	30
3.16.2 Create Peer	30
3.16.3 Delete Peer	31
4 Testes	32
4.1 Teste cliente Wireguard	32
4.2 Teste HTTPS	35
5 Análise crítica e proposta de melhorias	37
6 Conclusão	38
Bibliografia	39
Anexos	40

Lista de Figuras

1	Arquitetura geral SDN	3
2	Estrutura geral de pedidos API	4
3	Diagrama Lógico	8
4	Estrutura geral da Tabela Devices	9
5	Método EncryptPassword	10
6	Configuração do serviço HTTPS no MikroTik	11
7	LoginForm	15
8	Método para preencher a lista de Devices	16
9	<i>TabPage</i> Interfaces	17
10	<i>TabPage</i> Wireless	17
11	<i>TabPage</i> Bridge	19
12	<i>TabPage</i> Security Profile	21
13	<i>TabPage</i> Static Route	22
14	<i>TabPage</i> IP Address	24
15	<i>TabPage</i> DHCP	26
16	<i>TabPage</i> Address Pool	26
17	<i>TabPage</i> DNS	28
18	<i>TabPage</i> Wireguard	30
19	Interface Wireguard	33
20	Secção Create Peer	33
21	Peer criado	34
22	Ficheiro .conf criado	34
23	Teste QR code	35
24	Pedido HTTPS realizado no Postman	36

Lista de Tabelas

1	Lista de endpoints	14
---	------------------------------	----

Lista de siglas e acrónimos

AES Advanced Encryption Standard. 9, 14

API Application Programming Interface. 1–6, 14, 16, 18–25, 27–31, 35

DHCP Dynamic Host Configuration Protocol. 1, 2, 4, 6, 25–27, 38

DNS Domain Name System. 1, 2, 5, 6, 27–29, 36, 38

HTTP Hypertext Transfer Protocol. 5, 14, 36

HTTPS Hypertext Transfer Protocol Secure. 6, 10, 32, 35, 36

IP Internet Protocol. 1, 2, 4, 6, 8, 14, 15, 23–25

ISP Internet Service Provider. 8

IV Initialization Vector. 9

JSON JavaScript Object Notation. 36

MTU Maximum Transmission Unit. 32

QR Quick Response. 32, 34

REST Representational State Transfer. 2, 3, 5, 35

SDN Software Defined Networking. 1–3, 6, 38

SQL Structured Query Language. 1, 2, 7, 8

TLS Transport Layer Security. 36

URL Uniform Resource Locator. 5

VPN Virtual Private Network. 1, 2, 6, 8, 29, 38

1 Introdução

Atualmente, a configuração e a gestão de redes tradicionais são processos complexos e sujeitos a erros, devido à necessidade de configuração manual em cada equipamento. O *Software Defined Networking (SDN)* surge então como solução para este problema, permitindo um controlo centralizado e dinâmico da infraestrutura de rede.

O presente trabalho tem como objetivo o desenvolvimento de um controlador SDN simples para interagir com equipamentos de rede reais e virtuais. Visa a concepção e implementação de uma aplicação que atue como controlador SDN, facilitando a gestão de um ou mais equipamentos de rede, com ênfase nos routers da Mikrotik baseados no sistema operativo RouterOS.

A importância deste tema justifica-se pela crescente adoção de arquiteturas baseadas em SDN no cenário das redes modernas, proporcionando maior flexibilidade, escalabilidade e eficiência na gestão da infraestrutura de redes. A implementação de um controlador SDN simplificado permite explorar, na prática, os conceitos fundamentais desta tecnologia.

Como metodologia, foi realizada uma revisão bibliográfica sobre os conceitos de SDN, API REST e RouterOS, seguida da conceção e implementação do controlador SDN com interface gráfica. Por fim, foram realizados testes práticos para validação das funcionalidades desenvolvidas.

Este relatório está estruturado da seguinte forma: no capítulo 2 são apresentados os conceitos fundamentais para contextualizar o projeto. No capítulo 3 detalha-se o trabalho desenvolvido, com especial foco nas funcionalidades da aplicação. O capítulo 4 descreve os testes realizados. No capítulo 5 é feita uma análise crítica e proposta de melhorias, e no capítulo 6 são apresentadas as conclusões finais. O relatório inclui ainda a bibliografia e anexos relevantes.

2 Conceitos

Neste capítulo abordamos alguns conceitos relevantes para o projeto, como Software Defined Networking (SDN), Controlador SDN, MikroTik RouterOS, Application Programming Interface (API) e Representational State Transfer (REST) API.

2.1 Software Defined Networking (SDN)

Software Defined Networking utiliza controladores SDN para organizar o tráfego dos dados na rede, melhorando assim o seu desempenho, monitorização, flexibilidade e eficiência na administração da infraestrutura de rede.

Diferente das redes tradicionais, que administraram as suas funcionalidades a partir do *hardware* (modelo tradicional), uma rede SDN separa o plano de controlo do plano de dados, permitindo que a administração ocorra através de uma API (*Application Programming Interface*). Dessa forma o controlador SDN controla o fluxo de dados na rede, e os equipamentos de rede tratam apenas do encaminhamento dos dados. Com isso, o SDN permite a gestão centralizada dos dispositivos de redes e a implementação/controlo de redes virtuais com base numa infraestrutura física existente.

Igualmente é interoperável, ou seja, deve ser capaz de funcionar com qualquer dispositivo de rede independentemente do fabricante.

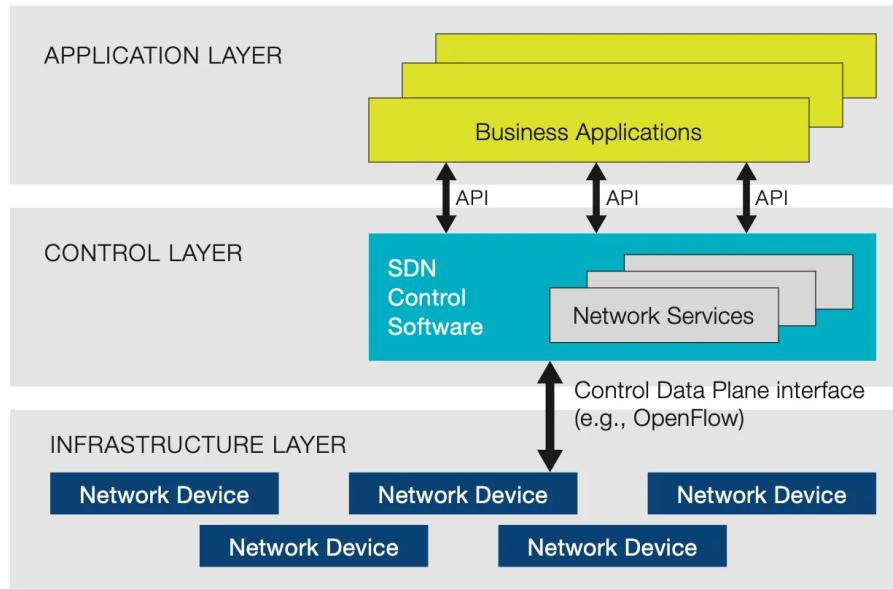


Figura 1: Arquitetura geral SDN

2.2 Controlador SDN

Software responsável por administrar uma rede de computadores de forma centralizada e programável. Recebe informações da rede, como o tráfego de dados e estatísticas de desempenho, e utiliza esses dados para controlar o fluxo de informações na rede. É o componente responsável por separar logicamente a camada de controlo da camada de encaminhamento de dados, sendo por isso uma peça fundamental na arquitetura SDN.

2.3 MikroTik RouterOS

MikroTik RouterOS é um sistema operativo de código aberto que fornece serviços de encaminhamento, rede sem fios e *firewall* para redes domésticas e de pequenos escritórios. O *software* foi desenvolvido pela empresa MikroTik, um fabricante de routers e outros *hardwares* de rede.

O conjunto de *hardware* e *software* MikroTik é projetado para atender a diversas necessi-

dades, garantindo uma rede confiável e eficiente para todos. Destaca-se pela sua flexibilidade, pelo vasto conjunto de funcionalidades que oferece e baixo custos, diferenciando-se de outras soluções disponíveis no mercado.

2.4 Application Programming Interface (API)

Uma API (Application Programming Interface) é um conjunto de protocolos e definições que permitem a interação e a troca de dados entre diferentes componentes de software. Os programadores utilizam APIs para integrar diferentes blocos de código, permitindo a criação de aplicações poderosas, resilientes e seguras que atendem às exigências dos utilizadores. Embora invisíveis, as APIs estão omnipresentes, operando continuamente em segundo plano para suportar as experiências digitais que são fundamentais no nosso dia a dia.

O funcionamento das APIs, ocorre através da troca de dados entre aplicações, sistemas e equipamentos, por meio de um ciclo de solicitação e resposta. A solicitação é enviada à API, acede aos dados necessários e envia-os de volta ao solicitante. Como podemos observar na **figura 2**.

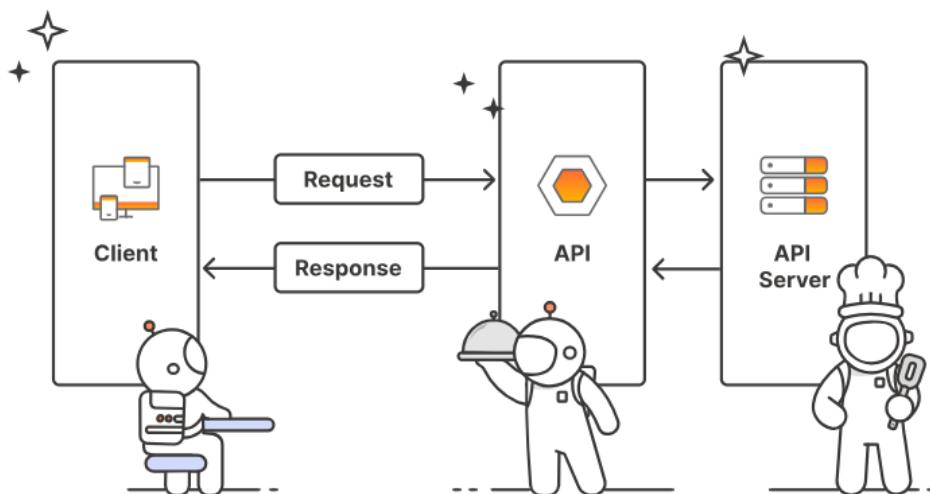


Figura 2: Estrutura geral de pedidos API

2.5 Representational State Transfer (REST) API

REST API, também conhecida como RESTful API, é a arquitetura de API mais popular para transferência de dados pela internet. Consiste em uma interface simples e uniforme, usada para disponibilizar dados, conteúdo, algoritmos e outros recursos digitais por meio de Uniform Resource Locator (URL) na web (*endpoints*).

Os recursos são acessíveis por meio de *endpoints*, e as operações são executadas com métodos Hypertext Transfer Protocol (HTTP) padrão, como GET, POST, PATCH, PUT e DELETE.

3 Trabalho Desenvolvido

Neste capítulo iremos abordar e explicar todas as funcionalidades implementadas no projeto. Começamos por referir as funcionalidades e de seguida mostramos, com recurso a imagens, a forma como foram implementadas e o seu funcionamento.

Funcionalidades implementadas

- Controlar mais do que um dispositivo com a mesma aplicação.
- Listar todas as interfaces do dispositivo.
- Listar apenas as interfaces wireless.
- Listar, criar, editar e apagar interfaces bridge e respetivas portas associadas.
- Criar, editar e apagar perfis de segurança para utilizar nas redes wireless.
- Ativar, desativar e configurar redes wireless.
- Listar, criar, editar e apagar rotas estáticas.
- Listar, criar, editar e apagar endereços IP.
- Listar, criar, editar e apagar servidores de DHCP.
- Ativar, desativar e configurar o servidor de DNS.
- Implementação de um servidor/cliente VPN cuja configuração e gestão é feita exclusivamente através do controlador criado.

Funcionalidades adicionais implementadas

- Integração com uma base de dados para armazenamento persistente dos dados provenientes dos dispositivos MikroTik.
- Funcionalidade para criar, editar e apagar pools de endereços IP (address pools) via API.
- Comunicação com a API efetuada de forma segura através do protocolo HTTPS.

3.1 Linguagem de Programação

C# foi a linguagem de programação escolhida, por ser uma linguagem orientada a objetos que permite criar uma variedade de aplicações seguras e robustas, sendo fortemente utilizada para o desenvolvimento de aplicações tradicionais em Windows, *Web* e também para equipamentos móveis. Desenvolvida pela Microsoft faz parte da sua plataforma *.NET Framework*.

A plataforma **Visual Studio 2022**, foi a escolhida para desenvolver o nosso projeto, devido ao conjunto de ferramentas, interface amigável, suporte nativo à linguagem C# e integração com o *.NET Framework*, o que permite o desenvolvimento, análise e teste dos projetos. Ainda oferece uma excelente integração com base de dados, principalmente através da tecnologia *Entity Framework*, o que permite a comunicação eficiente com bases de dados relacionais, como *Structured Query Language (SQL) Server*. A criação da base de dados e as suas operações podem ser realizadas diretamente no Visual Studio 2022, tornando o processo mais prático e centralizado.

3.2 Arquitetura da Solução

A nossa solução foi construída diretamente no *Form Designer* da plataforma Visual Studio 2022, que permite testar e melhorar constantemente o visual da interface.

Foram criados 2 *forms*, sendo um específico somente para a autenticação (*LoginForm*), abstraindo das restantes funcionalidades, que estão implementadas no 2 *form* (*MainForm*). Este é aberto posteriormente, caso a autenticação seja realizada com sucesso. A interface inclui um *TabControl* com várias *TabPage's*, sendo que cada separador representa uma funcionalidade distinta na aplicação.

O contraste visual da nossa interface é composto por uma imagem de fundo em tom castanho com marcas d'água contendo a palavra "MikroTik". As caixas de texto possuem uma cor cinza, enquanto os botões apresentam cores mais vivas, de forma a se destacarem na interface.

De forma a assegurar uma comunicação segura e eficiente entre os utilizadores e a infra-

estrutura da rede, foi desenvolvida a seguinte arquitetura lógica. Esta configuração apresenta, de modo geral, a organização da nossa rede, integrando o router principal do ISP (Internet Service Provider) ligado à internet, um router MikroTik com funcionalidades avançadas de gestão de tráfego e VPN, bem como os equipamentos dos clientes, que se ligam diretamente ao router MikroTik para gestão através da nossa aplicação ou através de túneis seguros estabelecidos pelo protocolo Wireguard (**figura 3**).

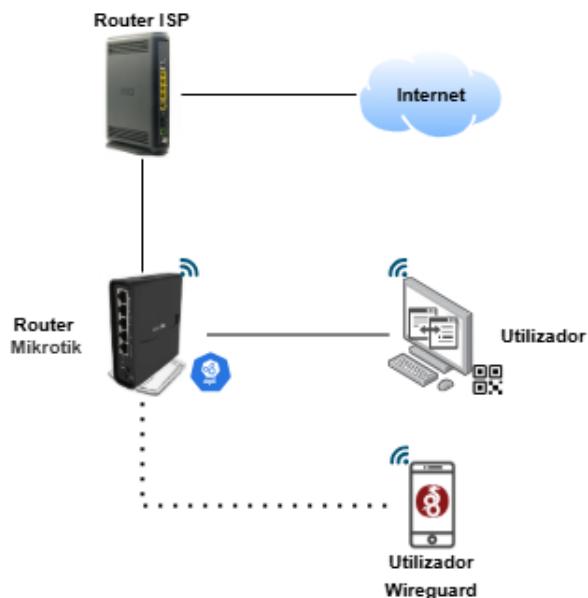


Figura 3: Diagrama Lógico

3.3 Base de Dados

A base de dados foi criada a partir da tecnologia *Entity Framework* do Visual Studio 2022, sendo uma base de dados *SQL Server LocalDB*. Foi criada com o propósito de guardar somente os dados de autenticação de vários MikroTiks numa tabela nomeada como "Devices". Esta tabela (**figura 4**) contém 5 campos:

- **Id:** chave primária, preenchida automaticamente de forma sequencial;
- **Name:** nome do dispositivo;
- **IpAddress:** endereço IP do MikroTik;

- **Username:** nome de utilizador para autenticação;
- **Password:** palavra-passe correspondente.

	Nome	Tipo de Dados	Permitir Nulos	Padrão
1	Id	int	<input type="checkbox"/>	
2	Name	nvarchar(100)	<input type="checkbox"/>	
3	IpAddress	nvarchar(100)	<input type="checkbox"/>	
4	Username	nvarchar(100)	<input type="checkbox"/>	
5	Password	nvarchar(100)	<input type="checkbox"/>	
			<input type="checkbox"/>	

Figura 4: Estrutura geral da Tabela Devices

Os dados do mesmo MikroTik podem ser guardados mais de uma vez, desde que seja com um nome diferente.

Antes de ser armazenada na tabela, a palavra-passe (plainText) é encriptada através do método **”Método EncryptPassword”**, que utiliza o algoritmo *Advanced Encryption Standard (AES)* para encriptar a palavra-passe com uma chave simétrica (key). O método ajusta a chave para garantir que tem 16 bytes e gera um vetor de inicialização (IV) aleatório para cada operação de encriptação, aumentando assim a segurança. O IV é armazenado no início do fluxo de dados encriptados, para poder ser reutilizado na desencriptação. A palavra-passe é escrita num **”CryptoStream”**, que aplica a encriptação utilizando o método **”CreateEncryptor”** do AES. Por fim, o conteúdo encriptado, incluindo o IV, é convertido para Base64 e devolvido como uma cadeia de caracteres (*string*).

```

1 referência
private string EncryptPassword(string plainText, string key)
{
    using (var aes = Aes.Create())
    {
        aes.Key = Encoding.UTF8.GetBytes(key.PadRight(16).Substring(0, 16));
        aes.GenerateIV();

        using (var encryptor = aes.CreateEncryptor(aes.Key, aes.IV))
        using (var ms = new MemoryStream())
        {
            ms.Write(aes.IV, 0, aes.IV.Length);
            using (var cs = new CryptoStream(ms, encryptor, CryptoStreamMode.Write))
            using (var writer = new StreamWriter(cs))
            {
                writer.WriteLine(plainText);
            }
            return Convert.ToBase64String(ms.ToArray());
        }
    }
}

```

Figura 5: Método EncryptPassword

3.4 Configuração de HTTPS no MikroTik

De forma a garantir uma comunicação segura entre o utilizador e o dispositivo MikroTik, foi implementado o protocolo HTTPS. Para isso, foram criados e configurados certificados digitais diretamente no RouterOS.

Começou-se por criar um certificado de autoridade (CA), que foi posteriormente utilizado para assinar um certificado válido para o serviço HTTPS. Esse certificado foi, então, associado ao serviço www-ssl (porta 443), permitindo o acesso à interface web do dispositivo através de HTTPS.

Na interface gráfica do MikroTik, é possível verificar que o serviço HTTPS está ativo, com o certificado correto atribuído, assegurando que as ligações ao dispositivo são encriptadas. Esta configuração é essencial para proteger as credenciais de acesso e os dados trocados durante a administração remota do dispositivo.

A imagem seguinte ilustra o processo de criação dos certificados e a sua associação ao serviço HTTPS.

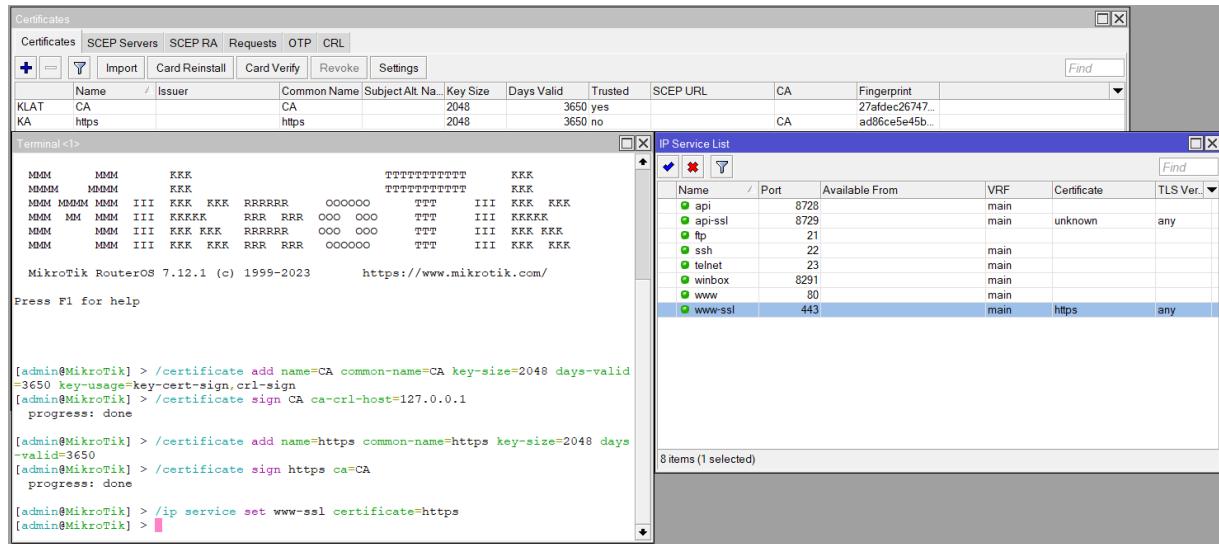


Figura 6: Configuração do serviço HTTPS no MikroTik

3.5 Lista de Endpoints

Pedido	Endpoint + Body
Listar todas as interfaces do dispositivo	GET - /rest/interface
Listar apenas as interfaces wireless	GET - /rest/interface/wireless
Listar interfaces bridge	GET - /rest/interface/bridge
Criar interface bridge	PUT - /rest/interface/bridge {"name": "bridgeTeste", "arp": "proxy-arp"}
Editar interface bridge	PATCH - /rest/interface/bridge/*ID {"id": "*9", "name": "bridgeTeste", "arp": "replay-only"}
Apagar interface bridge	DELETE - /rest/interface/bridge/*ID
Listar portas de bridge	GET - /rest/interface/bridge/port
Adicionar porta à bridge	PUT - /rest/interface/bridge/port {"interface": "interfaceName", "bridge": "bridgeId"}
Editar porta da bridge	PATCH - /rest/interface/port/*ID {"name": "bridge1"}
Remover porta da bridge	DELETE - /rest/interface/bridge/port/*ID

Pedido	Endpoint + Body
Listar perfis de segurança wireless	GET - /rest/interface/wireless/security-profiles
Criar perfil de segurança wireless	PUT - /rest/interface/wireless/security-profiles <pre>{"name": "profileTeste", "authentication-types": "wpa-psk", "mode": "dynamic-keys", "wpa2-pre-shared-key": "teste"}</pre>
Editar perfil de segurança wireless	PATCH - /rest/interface/wireless/security-profiles/*ID <pre>{"name": "profileTeste", "authentication-types": "wpa-psk", "mode": "dynamic-keys", "wpa2-pre-shared-key": "teste"}</pre>
Apagar perfil de segurança wireless	DELETE - /rest/interface/wireless/security-profiles/*ID
Ativar rede wireless	POST - /rest/interface/wireless/enable <pre>{"id": "*123"}</pre>
Desativar rede wireless	POST - /rest/interface/wireless/disable <pre>{"id": "*123"}</pre>
Editar rede wireless	PATCH - /rest/interface/wireless/*ID <pre>{"mode": "ap-bridge", "band": "2ghz-b/g/n", "channel-width": "20/40mhz-Ce", "frequency": "2412", "ssid": "Vassoura-Wi-Fi", "security-profile": "profile1"}</pre>
Listar rotas estáticas	GET - /rest/ip/route
Criar rota estática	PUT - /rest/ip/route <pre>{"dst-address": "0.0.0.0/0", "gateway": "10.20.139.254"}</pre>
Editar rota estática	PATCH - /rest/ip/route/*ID <pre>{"dst-address": "0.0.0.0/0", "gateway": "10.20.139.254"}</pre>
Apagar rota estática	DELETE - /rest/ip/route/*ID
Listar endereços IP	GET - /rest/ip/address
Criar endereço IP	PUT - /rest/ip/address

Pedido	Endpoint + Body
	{"address": "10.10.0.1/24", "interface": "bridge1"}
Editar endereço IP	PATCH - /rest/ip/address/*ID {"address": "10.10.0.2/24"}
Apagar endereço IP	DELETE - /rest/ip/address/*ID
Listar servidores DHCP	GET - /rest/ip/dhcp-server
Criar servidor DHCP	PUT - /rest/ip/dhcp-server {"address-pool": "dhcp_pool0", "interface": "bridge1", "lease-time": "30m", "name": "dhcp2"}
Editar servidor DHCP	PATCH - /rest/ip/dhcp-server/*ID {"address-pool": "dhcp_pool0", "interface": "bridge1", "lease-time": "30m", "name": "dhcp2"}
Apagar servidor DHCP	DELETE - /rest/ip/dhcp-server/*ID
Listar pools de endereços	GET - /rest/ip/pool
Criar pool de endereços	PUT - /rest/ip/pool {"name": "dhcp_pool1", "ranges": "192.168.1.100-192.168.1.200"}
Editar pool de endereços	PATCH - /rest/ip/pool/*ID {"name": "dhcp_pool2", "ranges": "192.168.2.100-192.168.2.200"}
Apagar pool de endereços	DELETE - /rest/ip/pool/*ID
Listar configuração DNS	GET - /rest/ip/dns
Editar configuração DNS	POST - /rest/ip/dns/set {"servers": "8.8.8.8, 1.1.1.1"}
Ativar/Desativar DNS remoto	POST - /rest/ip/dns/set {"allow-remote-requests": "true", "servers": "172.22.1.101,172.22.1.102"}
Listar entradas DNS estáticas	GET - /rest/ip/dns/static
Criar entrada DNS estática	PUT - /rest/ip/dns/static

Pedido	Endpoint + Body
	{"address": "10.20.139.40", "name": "test.com"}
Editar entrada DNS estática	PATCH - /rest/ip/dns/static/*ID {"address": "10.20.139.41", "name": "test2.com", "disabled": "true"}
Listar interfaces WireGuard	GET - /rest/interface/wireguard
Listar peers WireGuard	GET - /rest/interface/wireguard/peers
Criar peer WireGuard	PUT - /rest/interface/wireguard/peers {"interface": "wireguard1", "private-key": "auto", "allowed-address": "::/0"}
Gerar configuração cliente WireGuard	POST - /rest/interface/wireguard/peers/show-client-config {".id": "*9"}
Apagar peer WireGuard	DELETE - /rest/interface/wireguard/peers/*ID

Tabela 1: Lista de endpoints

3.6 Login

O processo de *login* foi implementado no *form* ”LoginForm”, que contém quatro campos obrigatórios: nome do dispositivo, endereço IP, nome de utilizador e palavra-passe. O utilizador deve preencher todos os campos; caso contrário, será apresentada uma mensagem a informar que é necessário preencher todos os campos.

Quando a autenticação é bem-sucedida, a palavra-passe é previamente encriptada com o algoritmo AES através do método ”EncryptPassword”, utilizando uma chave simétrica. De seguida, é criada uma instância da classe Device com os dados introduzidos. Se a combinação de nome e IP ainda não existir na base de dados (Devices), essa informação será armazenada.

A autenticação é feita através de um pedido HTTP usando autenticação básica (*Basic Auth*) ao Mikrotik via REST API. Caso o *login* seja aceito (código 200 OK), o *form* principal da aplicação (MainForm) é aberto, recebendo como parâmetro o equipamento autenticado, e o *form* de *login* é ocultado. Caso contrário, é apresentada uma mensagem a indicar que o *login* falhou.

devido a credenciais inválidas ou erro de ligação.

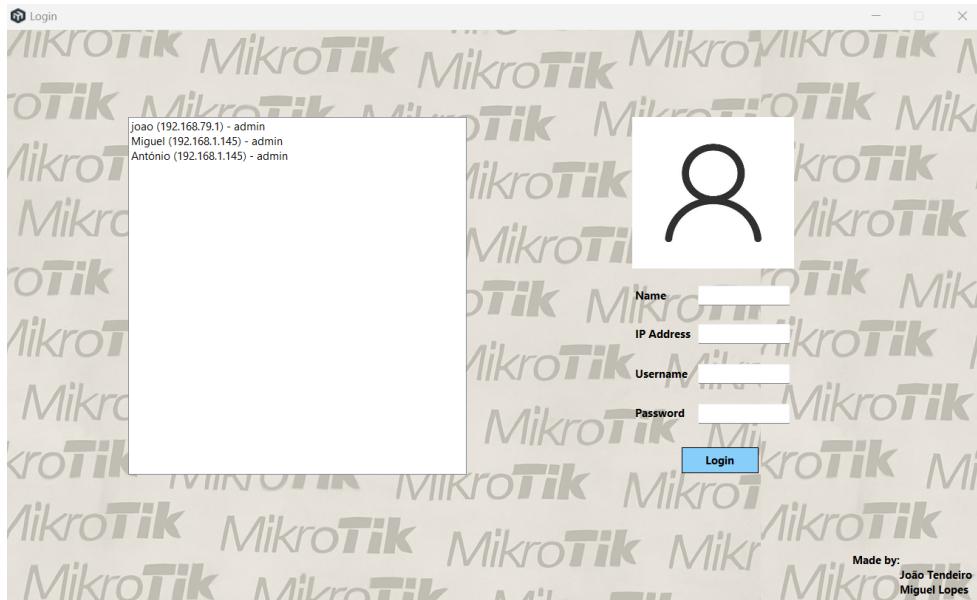


Figura 7: LoginForm

3.7 Obtenção de dados da tabela Devices

A tabela Devices, contém os dados de autenticações anteriores de routers Mikrotik. Estes dados são apresentados no *form* "LoginForm" através de uma lista, podendo ser usados para efetuar a autenticação.

Todos os regtos da tabela são carregados automaticamente quando se inicializa o "LoginForm". A lista apresentada ao utilizador inclui o nome do dispositivo, o endereço IP e o nome de utilizador. A palavra-passe, por questões de segurança, não é exibida, mas é carregada em memória e descodificada de forma transparente, ficando pronta para ser utilizada.

Quando o utilizador seleciona um dos dispositivos da lista, os campos do formulário de autenticação são automaticamente preenchidos com os dados correspondentes, facilitando o processo de *login*.

```
1 referência
private void listBox1_SelectedIndexChanged_1(object sender, EventArgs e)
{
    if (listBox1.SelectedIndex >= 0 && listBox1.SelectedIndex < devices.Count)
    {
        var selecionado = devices[listBox1.SelectedIndex];
        name.Text = selecionado.name;
        ipAddress.Text = selecionado.ipAddress;
        username.Text = selecionado.username;

        string encryptionKey = "ChaveSimetrical23";
        try
        {
            password.Text = DecryptPassword(selecionado.password, encryptionKey);
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Erro ao descriptografar a senha: {ex.Message}");
            password.Text = string.Empty;
        }
    }
}
```

Figura 8: Método para preencher a lista de Devices

3.8 Lista de todas as Interfaces

A *TabPage* inicial do ”MainForm” lista todas as interfaces do router. Estas são carregadas no momento em que o *form* é inicializado e são atualizadas sempre que se edita, apaga ou cria uma interface no nosso controlador. Para listar as interfaces, é feito um pedido GET à API, com o *endpoint* /rest/interface.

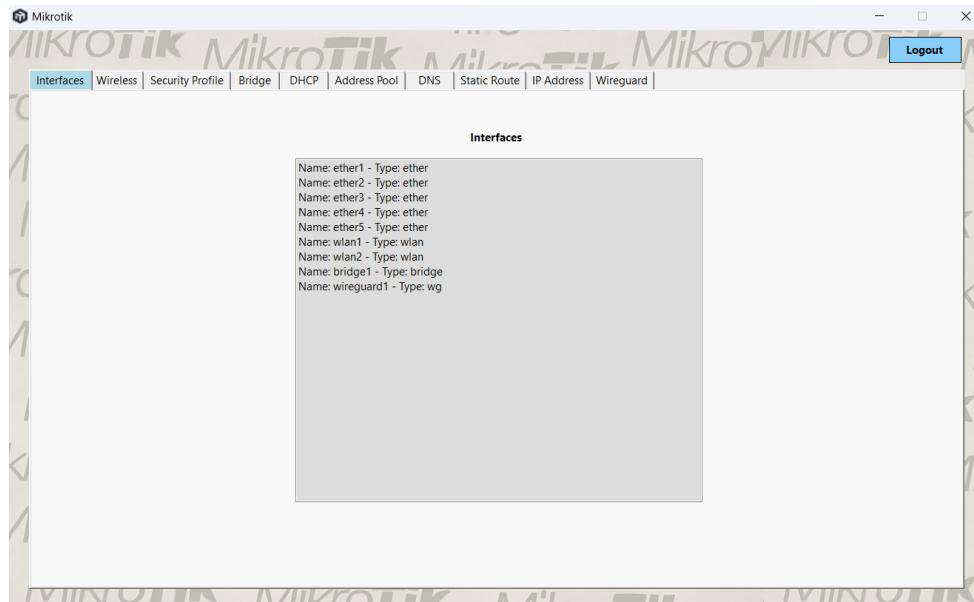


Figura 9: TabPage Interfaces

3.9 Interfaces Wireless

Na TabPage "Wireless", o utilizador tem a possibilidade de **Editar**, **Apagar**, **Ativar** e **Desativar** interfaces wireless.

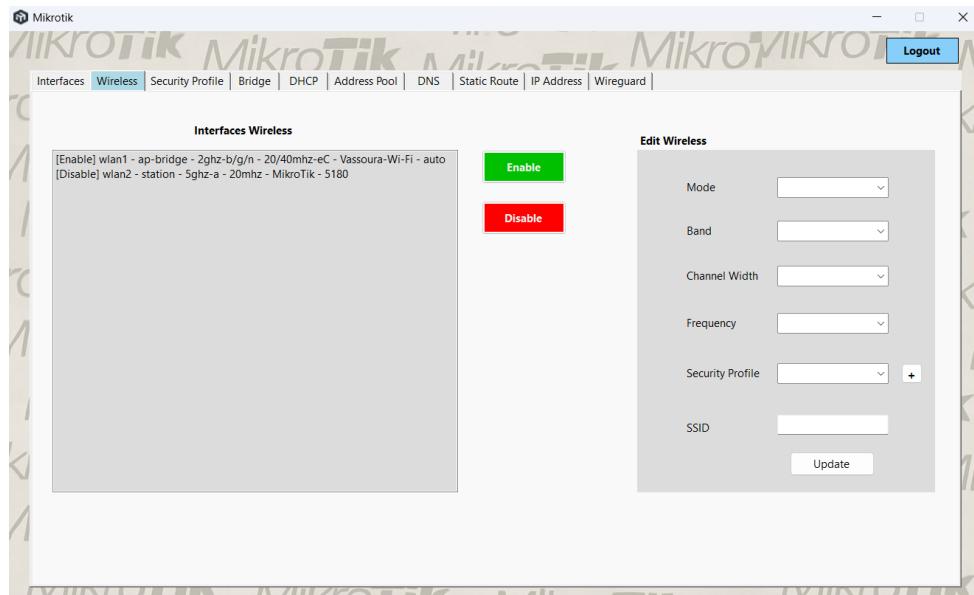


Figura 10: TabPage Wireless

3.9.1 Listar

Na *TabPage* "Wireless" a aplicação apresenta uma lista de interfaces wireless disponíveis no equipamento. Cada interface é listada com informações sobre o seu estado (se está ativada ou desativada), o tipo de conexão (como "ap-bridge" ou "station"), as especificações de banda (como 2GHz ou 5GHz), largura de canal, SSID (nome da rede) e a frequência em MHz. Para listar as interfaces é efetuado um pedido GET à API, com o *endpoint* `/rest/interface/wireless`.

3.9.2 Ativar e Desativar

O botão "Enable" permite ativar a interface selecionada na lista e o botão "Disable" para desativar. O utilizador ao clicar num dos botões é feito um pedido POST à API partir dos *endpoint's* `/rest/interface/wireless/enable` ou `/rest/interface/wireless/disable`.

3.9.3 Editar

No *GroupBox* "Edit Wireless", o utilizador pode editar as configurações de uma interface wireless selecionada. Ao clicar na interface, as configurações existentes são preenchidas automaticamente nas *Combobox* e na *TextBox*. Após ajustar as configurações da interface wireless, o utilizador clica no botão "Update" e é feito um pedido PATCH à API com o *endpoint* `/rest/interface/wireless/*id`, onde *id representa o identificador único da interface a ser atualizada. Nesta *TabPage*, caso o utilizador pretenda adicionar um Security Profile, também é possível ao clicar no botão "+", sendo redirecionado para a *TabPage* "Security Profile".

3.10 Interfaces Bridge e respetivas portas associadas

Na *TabPage* "Bridge", o utilizador tem a possibilidade de **Criar**, **Editar** e **Eliminar** interfaces bridge e as respetivas portas.

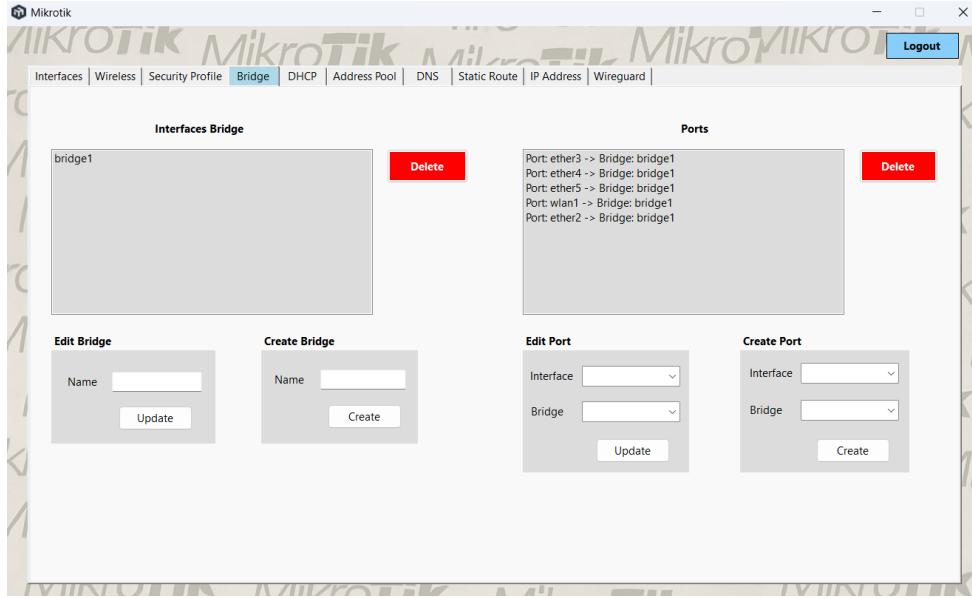


Figura 11: TabPage Bridge

3.10.1 Listar

Para a TabPage "Bridge", a aplicação apresenta a lista de interfaces bridge configuradas no dispositivo, como também as portas existentes. Para listar as interfaces bridge e as respectivas portas é feito um pedido GET à API, com os endpoint's **/rest/interface/bridge** e **/interface/bridge/port**.

3.10.2 Criar

Na GroupBox "Create Bridge", o utilizador pode criar uma nova bridge ao inserir um nome no campo "Name" e ao clicar no botão "Create", aplicação envia um pedido PUT à API com o endpoint **/rest/interface/bridge** para criar a nova bridge.

Na GroupBox "Create Port", o utilizador deve preencher os campos "Interface" e "Bridge". Após isso, o utilizador clica no botão "Create", que envia um pedido PUT à API com o endpoint **/rest/interface/port**.

3.10.3 Editar

Na *GroupBox* "Edit Bridge", o utilizador pode editar uma bridge existente. Ao selecionar uma bridge da lista de "Interfaces Bridge", o nome da bridge é carregado automaticamente. Após ajustar o nome, o utilizador clica no botão "Update", e a aplicação envia um pedido PATCH à API com o *endpoint* **/rest/interface/bridge/*id**, onde *id é o identificador único da bridge a ser atualizada.

Na *GroupBox* "Edit Port", o utilizador pode editar uma porta existente. Para isso, o utilizador seleciona uma interface e uma bridge. Após ajustar as configurações pretendidas, o utilizador clica no botão "Update", que envia um pedido PATCH à API com o *endpoint* **/rest/interface/port/*id**, onde *id é o identificador único da porta a ser atualizada.

3.10.4 Apagar

Na secção "Interfaces Bridge", o utilizador também pode apagar uma bridge existente. Para isso, basta selecionar a bridge pretendida e clicar no botão "Delete". A aplicação envia um pedido DELETE à API com o *endpoint* **/rest/interface/bridge/*id**, removendo a bridge selecionada.

Na secção "Ports", o utilizador pode apagar uma porta existente. Para isso, basta selecionar a porta que deseja remover e clicar no botão "Delete". A aplicação envia um pedido DELETE à API com o *endpoint* **/rest/interface/port/*id**, removendo a porta selecionada da bridge correspondente.

3.11 Perfis de segurança para utilizar nas redes wireless

Na *TabPage* "Security Profile", o utilizador tem a possibilidade de **Criar, Editar e Eliminar** perfis de segurança.

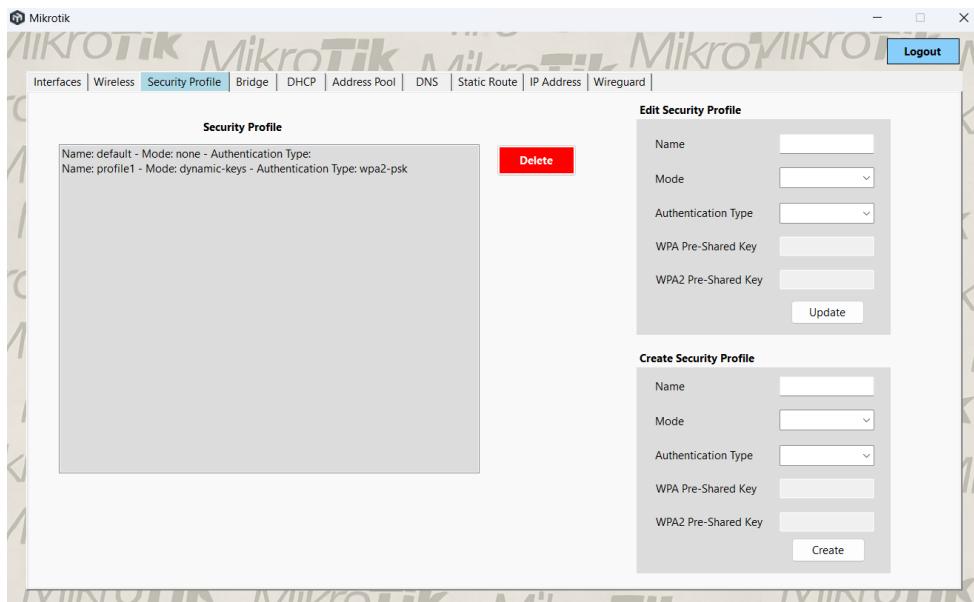


Figura 12: TabPage Security Profile

3.11.1 Criar

Na *GroupBox* "Create Security Profile", o utilizador pode criar um novo perfil de segurança preenchendo os campos "Name", "Mode", "Authentication Type", "WPA Pre-Shared Key" e "WPA2 Pre-Shared Key". Os campos "WPA Pre-Shared Key" e "WPA2 Pre-Shared Key" ficam visíveis/invisíveis dependendo do tipo de autenticação selecionado. Após preencher os campos, o utilizador clica no botão "Create", que envia um pedido PUT à API com o *endpoint* /rest/interface/wireless/security-profiles. A aplicação cria o novo perfil de segurança e o adiciona à lista de perfis existentes.

3.11.2 Editar

Na *GroupBox* "Edit Security Profile", o utilizador pode editar as configurações de um perfil de segurança. Para isso, o utilizador seleciona o perfil de segurança que deseja editar. Os campos do perfil selecionado são preenchidos automaticamente. O utilizador pode alterar as configurações, como "Mode", "Authentication Type", "WPA Pre-Shared Key" e "WPA2 Pre-Shared Key". Após fazer as alterações, o utilizador clica no botão "Update", que envia um pedido PATCH à

API com o *endpoint* /rest/interface/wireless/security-profiles/*id, onde *id é o identificador único do perfil de segurança a ser atualizado.

3.11.3 Apagar

O utilizador pode apagar um perfil de segurança existente. Para isso, basta selecionar o perfil que deseja remover e clicar no botão "Delete", que envia um pedido DELETE à API com o *endpoint* /rest/interface/wireless/security-profiles/*id, removendo o perfil selecionado.

3.12 Rotas estáticas

Na *TabPage* "Static Route", o utilizador tem a possibilidade de **Criar, Editar e Apagar** rotas estáticas.

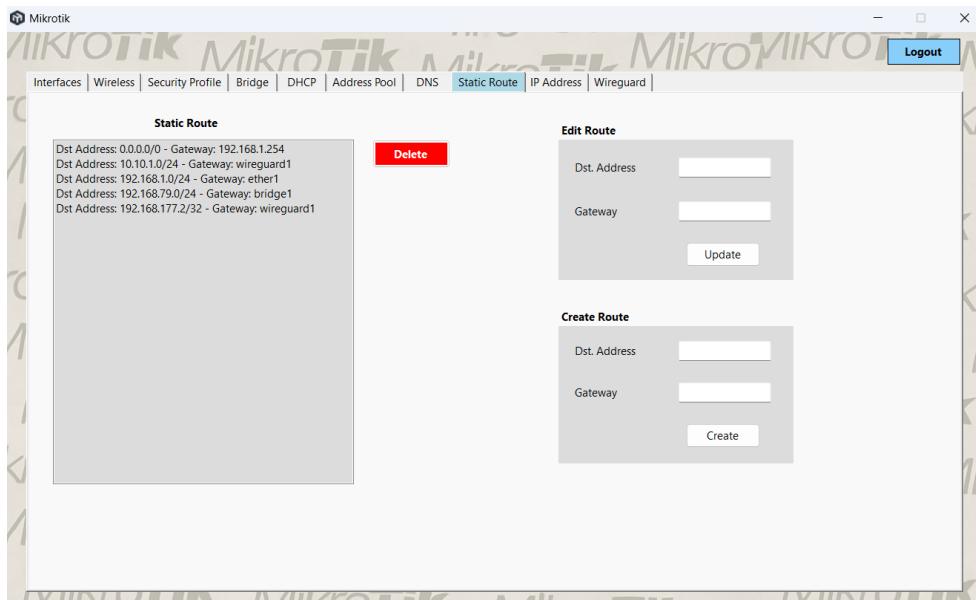


Figura 13: *TabPage* Static Route

3.12.1 Listar

Na secção "Static Route", é apresentada uma lista de rotas estáticas configuradas no dispositivo. Cada rota é exibida com o "Dst.Address"(endereço de destino) e o "Gateway"(endereço

do gateway para essa rota). Para listar as rotas estáticas, é feito um pedido GET à API, com o *endpoint /rest/ip/route*.

3.12.2 Criar

Na *GroupBox* "Create Route", o utilizador pode adicionar uma nova rota estática preenchendo os campos "Dst.Address"(endereço de destino) e "Gateway"(endereço do gateway). Após preencher os campos, o utilizador clica no botão "Create", que envia um pedido PUT à API com o *endpoint /rest/ip/route* para criar a nova rota.

3.12.3 Editar

Na *GroupBox* "Edit route", o utilizador pode editar uma rota estática existente. Para editar, o utilizador seleciona a rota que pretende modificar, e os campos "Dst.Address" e "Gateway" são automaticamente preenchidos com as informações da rota selecionada. Após fazer as alterações pretendidas, o utilizador clica no botão Update, o que envia um pedido PATCH à API com o *endpoint /rest/ip/route/*id*, onde *id representa o identificador do endereço IP.

3.12.4 Apagar

Na secção "Static Route", o utilizador pode remover uma rota estática existente. Ao selecionar a rota que pretende apagar e clicar no botão Delete, a aplicação envia um pedido DELETE à API com o *endpoint /rest/ip/route/*id*, removendo a rota selecionada da configuração do dispositivo.

3.13 Endereços IP

Na *TabPage* "IP Address", o utilizador tem a possibilidade de **Criar**, **Editar** e **Apagar** endereços IP atribuídos a interfaces da rede.

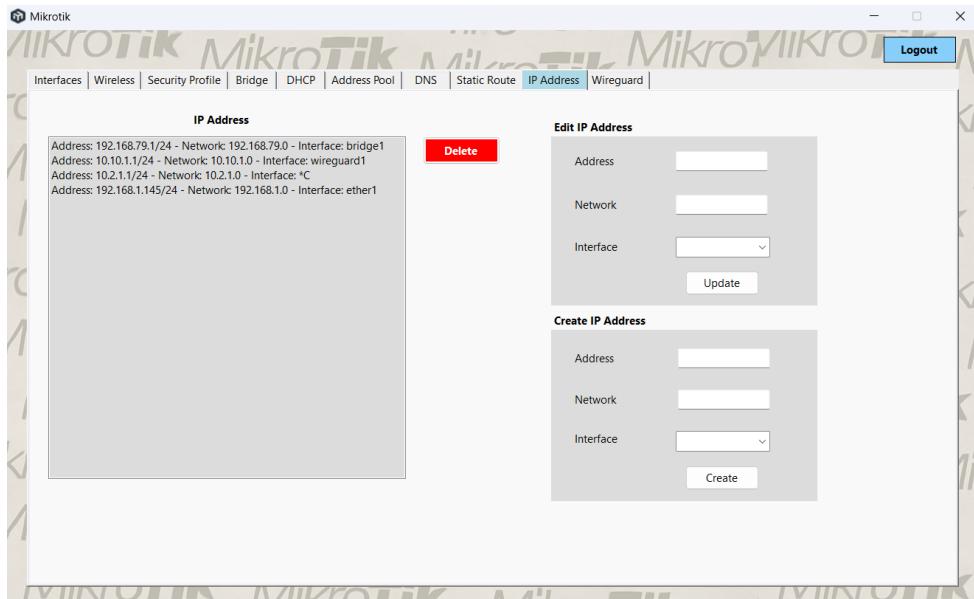


Figura 14: TabPage IP Address

3.13.1 Listar

Na secção ”IP Address”, é apresentada uma lista com os endereços IP já configurados. Para listar os endereços IP é feito um pedido GET à API, com o endpoint `/rest/ip/address`. Cada linha apresenta a seguinte informação:

- **Address:** endereço IP e máscara de sub-rede;
- **Network:** rede associada;
- **Interface:** interface à qual o endereço está atribuído.

3.13.2 Criar

Na *GroupBox* ”Create IP Address”, o utilizador pode criar um novo endereço IP preenchendo os campos ”Address”, ”Network” e ”Interface”. Após preencher os campos, o utilizador clica no botão ”Create”, que envia um pedido PUT à API com o endpoint `/rest/ip/address`. A aplicação cria o novo endereço IP e adiciona-o à lista de endereços configurados.

3.13.3 Editar

Na *GroupBox* "Edit IP Address", o utilizador pode editar as informações de um endereço IP existente. Para tal, deve selecionar o endereço pretendido da lista, sendo os campos "Address", "Network" e "Interface" automaticamente preenchidos. O utilizador pode alterar os valores pretendidos e clicar no botão "Update", que envia um pedido PATCH à API com o *endpoint* /rest/ip/address/*id, onde *id representa o identificador do endereço IP.

3.13.4 Apagar

Para eliminar um endereço IP, o utilizador deve selecionar o endereço pretendido da lista e clicar no botão "Delete". Esta ação envia um pedido DELETE à API com o *endpoint* /rest/ip/address/*id, removendo o endereço IP selecionado da configuração do router.

3.14 Servidores de DHCP

No *TabPage* "DHCP", o utilizador tem a possibilidade de **Criar**, **Editar** e **Apagar** servidores de DHCP. Nas *GroupBoxes* desta *TabPage* existe um botão "+", que ao ser clicado e redireciona o utilizador para a *TabPage* "Address Pool", onde é possível **Criar** (PUT - /rest/ip/pool), **Editar** (PATCH - /rest/ip/pool/*id) e **Apagar** um Address Pool (DELETE - /rest/ip/pool/*id).

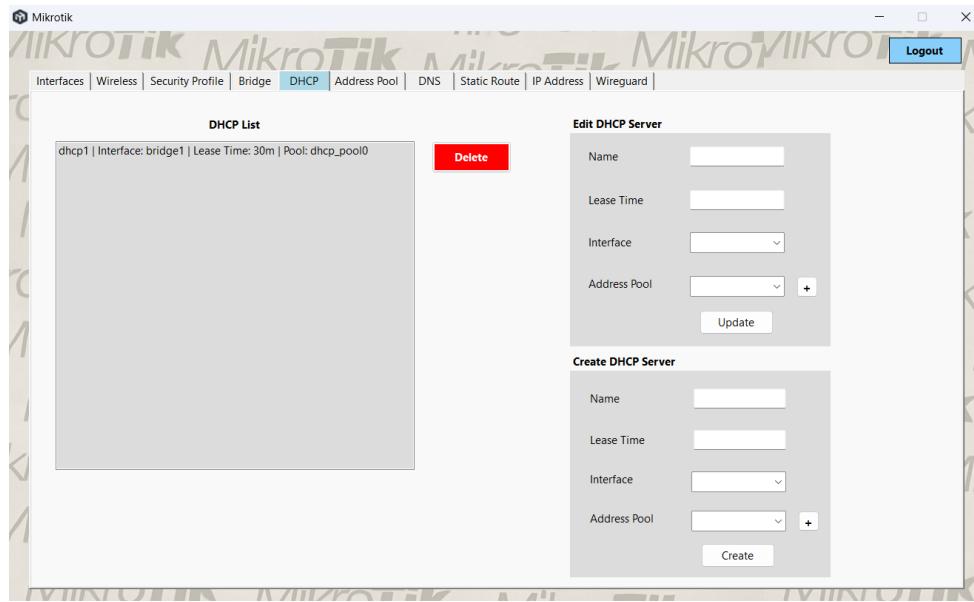


Figura 15: TabPage DHCP

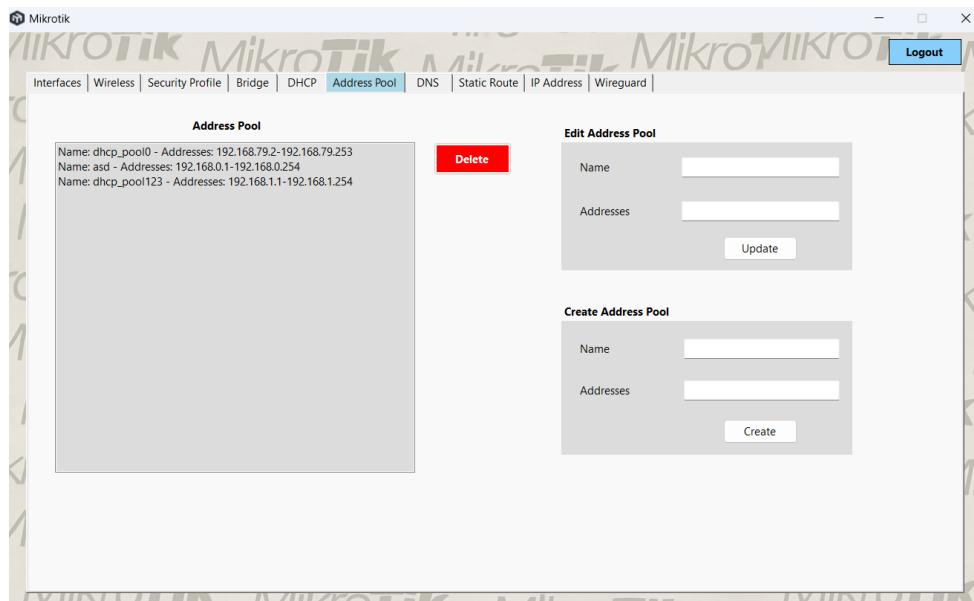


Figura 16: TabPage Address Pool

3.14.1 Listar

Na secção ”DHCP”, a aplicação apresenta uma lista de servidores DHCP configurados no dispositivo. Cada servidor é exibido com o seu ”Nome”, ”Interface”, ”Lease Time” e ”Address Pool”(pool de endereços atribuídos). Para listar os servidores DHCP, é feito um pedido GET à

API, com o *endpoint* **/rest/ip/dhcp-server**.

3.14.2 Criar

Na *GroupBox* ”Create DHCP Server”, o utilizador pode adicionar um novo servidor DHCP. Para isso, deve preencher os campos ”Name”(nome do servidor), ”Lease Time”(tempo de concessão), ”Interface”(interface de rede) e ”Address Pool”(pool de endereços). Após preencher os campos, o utilizador clica no botão Create, que envia um pedido PUT à API com o *endpoint* **/rest/ip/dhcp-server** para criar o novo servidor DHCP.

3.14.3 Editar

Na *GroupBox* ”Edit DHCP Server”, o utilizador pode editar as configurações de um servidor DHCP existente. Após selecionar o servidor da lista, os campos ”Name”, ”Lease Time”, ”Interface” e ”Address Pool” são preenchidos automaticamente com os dados da configuração atual. O utilizador modifica as informações e clica no botão ”Update” para enviar um pedido PATCH à API com o *endpoint* **/rest/ip/dhcp-server/*id**, onde *id representa o identificador único do servidor DHCP a ser atualizado.

3.14.4 Apagar

Na secção ”DHCP List”, o utilizador pode apagar um servidor DHCP. Após selecionar o servidor que pretende apagar e clicar no botão ”Delete”, é enviado um pedido DELETE à API com o *endpoint* **/rest/ip/dhcp-server/*id**, removendo o servidor selecionado da configuração do dispositivo.

3.15 Servidor de DNS

Na *TabPage* ”DNS”, o utilizador tem a possibilidade de **Ativar**, **Desativar** e **Configurar** o servidor de DNS, assim como adicionar ou editar registos DNS estáticos.

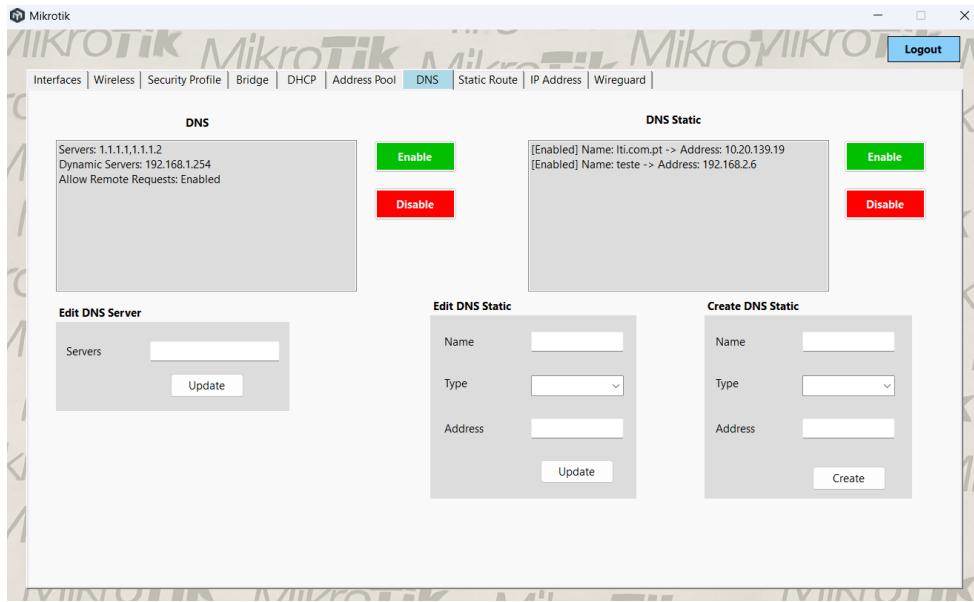


Figura 17: TabPage DNS

3.15.1 Ativar

A ativação do servidor de DNS, o utilizador deve clicar no botão "Enable", presente na secção "DNS". Ao efetuá-lo, a aplicação envia um pedido POST à API com o *endpoint /rest/ip/dns/set*, ativando a funcionalidade "Allow Remote Requests", o que permite que o router responda a pedidos DNS feitos remotamente.

Também é possível ativar DNS estáticos (como nomes de domínio específicos), selecionando o DNS estático pretendido da lista e clicando no botão "Enable" correspondente, na secção "DNS Static". Envia um pedido PATCH à API com o *endpoint /rest/ip/dns/static/*id*. Esta ação atualiza o estado do registo para ativo.

3.15.2 Desativar

Para desativar o servidor de DNS, o utilizador deve clicar no botão "Disable", que envia um pedido POST à API com o *endpoint /rest/ip/dns/set*, desativando a funcionalidade "Allow Remote Requests", não permitindo pedidos remotos.

De igual forma, os registos DNS estáticos podem ser desativados, seleccionando o DNS está-

tico pretendido da lista e clicando no botão "Disable" na secção "DNS Static". Envia um pedido PATCH à API com o *endpoint /rest/ip/dns/static/*id*. Esta ação atualiza o estado do registo para desativado.

3.15.3 Configurar

A configuração do servidor de DNS é feita na *GroupBox* "Edit DNS Server", onde o utilizador pode inserir manualmente os endereços dos servidores DNS pretendidos no campo "Servers" e clicar em "Update". Este procedimento envia um pedido POST à API com o *endpoint /rest/ip/dns/set*, atualizando os servidores DNS utilizados pelo router.

Para configurar registos estáticos de DNS, o utilizador pode:

Criar um novo registo na *GroupBox* "Create DNS Static", preenchendo os campos "Name", "Type" e "Address", e clicando em "Create", o que envia um pedido PUT para o *endpoint /rest/ip/dns/static*.

Para editar um registo existente tem que ser através da *GroupBox* "Edit DNS Static", onde os dados do registo são preenchidos automaticamente após seleção do pretendido na lista de "DNS Static". Após fazer as alterações desejadas, o utilizador clica em "Update", que envia um pedido PATCH ao *endpoint /rest/ip/dns/static/*id*, onde *id representa o identificador único do registo.

3.16 Wireguard

Na *TabPage* "Wireguard", o utilizador pode gerir interfaces e peers da VPN Wireguard. Esta secção permite **Criar** e **Remover** peers associados às interfaces configuradas.

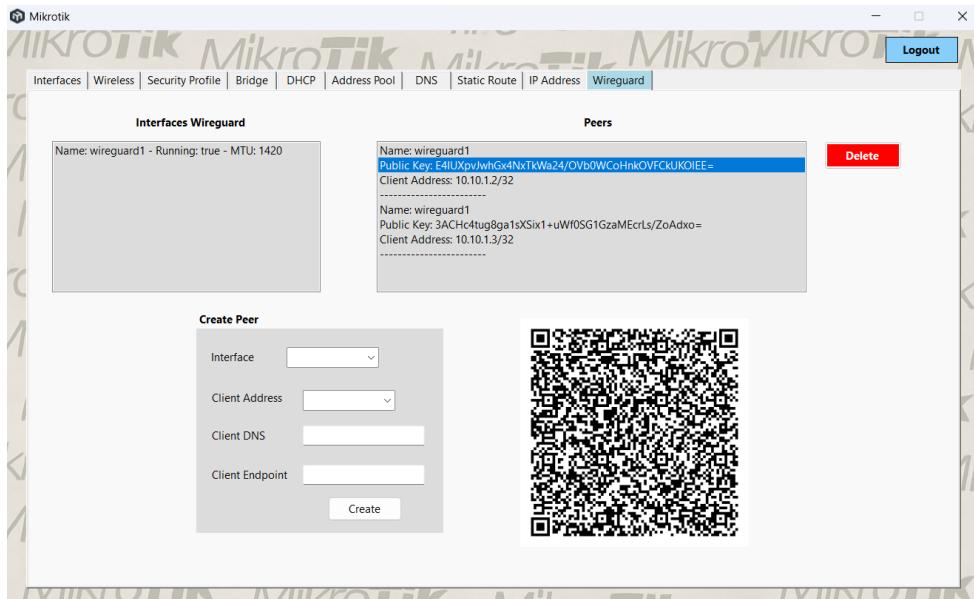


Figura 18: TabPage Wireguard

3.16.1 Listar

Na secção ”Interfaces Wireguard”, são apresentadas as interfaces Wireguard configuradas no equipamento. Cada interface mostra informações como o nome, o estado de funcionamento (running) e o valor do MTU(Unidade Máxima de Transmissão). Estas informações são obtidas através de um pedido GET à API, utilizando o *endpoint /rest/interface/wireguard*.

Na secção ”Peers”, são listados todos os peers associados às interfaces Wireguard. Para cada peer, são apresentados: a interface à qual está associado, chave pública do peer (*Public Key*) e o endereço atribuído ao cliente (*Client Address*). Estes dados são carregados a partir de um pedido GET à API com o *endpoint /rest/interface/wireguard/peers*.

3.16.2 Create Peer

Para criar um novo peer, o utilizador deve utilizar a *GroupBox* ”Create Peer”, preenchendo os seguintes campos:

- **Address:** Interface Wireguard à qual o peer será associado;

- **Client Address:** Endereço IP a atribuir ao peer;
- **Client DNS:** Servidor DNS a ser utilizado pelo cliente;
- **Client Endpoint:** *Endpoint* remoto do cliente.

Após preencher os campos, o utilizador clica no botão "Create", que envia um pedido PUT à API com o *endpoint* `/rest/interface/wireguard/peers`. O novo peer é então criado e adicionado à lista.

Além disso, após a criação, é gerado automaticamente um código QR (Quick Response), apresentado na interface, e um ficheiro `".conf"` é descarregado automaticamente para o dispositivo do utilizador. Estas funcionalidades facilitam a configuração do peer num equipamento do cliente.

3.16.3 Delete Peer

Para remover um peer existente, o utilizador deve selecioná-lo na lista dos "Peers" e clicar no botão "Delete". Esta ação envia um pedido DELETE à API com o *endpoint* `/rest/interface/wireguard/peers/*id`, onde *id é o identificador único do *peer* a eliminar.

4 Testes

Neste capítulo são descritos os testes realizados para validar o correto funcionamento das funcionalidades implementadas ao longo do projeto, com foco nas funcionalidades do **WireGuard** e **HTTPS**. Para cada uma dessas funcionalidades, são apresentados os objetivos dos testes, os passos seguidos e os resultados obtidos, garantindo que os requisitos definidos foram cumpridos de forma adequada.

Segue em anexo um vídeo com todos os testes realizados das restantes funcionalidades implementadas.

4.1 Teste cliente Wireguard

Objetivos do teste

- Verificar a correta visualização das interfaces WireGuard configuradas no equipamento.
- Validar a listagem e os detalhes dos peers associados às interfaces.
- Confirmar a atualização dinâmica do código QR ao selecionar um *peer*.
- Testar a criação de um novo *peer*, incluindo a criação do ficheiro *.conf* e QR *code*.
- Garantir o correto funcionamento da funcionalidade de remoção de *peers*.

Durante os testes do WireGuard, acedemos a *TabPage* "Wireguard" e verificamos a lista das interfaces. Confirmamos que as interfaces configuradas foram exibidas corretamente, com informações como nome, estado (*running*) e valor do MTU. Como podemos observar na **fig.19**.

Em seguida, verificámos a secção *Peers* para garantir que os peers associados às interfaces WireGuard estavam listados corretamente. Para cada *peer*, confirmámos que as informações — como a interface, chave pública (*Public Key*) e endereço do cliente (*Client Address*) — estavam completas. Também foi possível observar que, ao selecionar um *peer*, o código QR é atualizado dinamicamente, permitindo ainda a sua remoção através da opção "Delete".

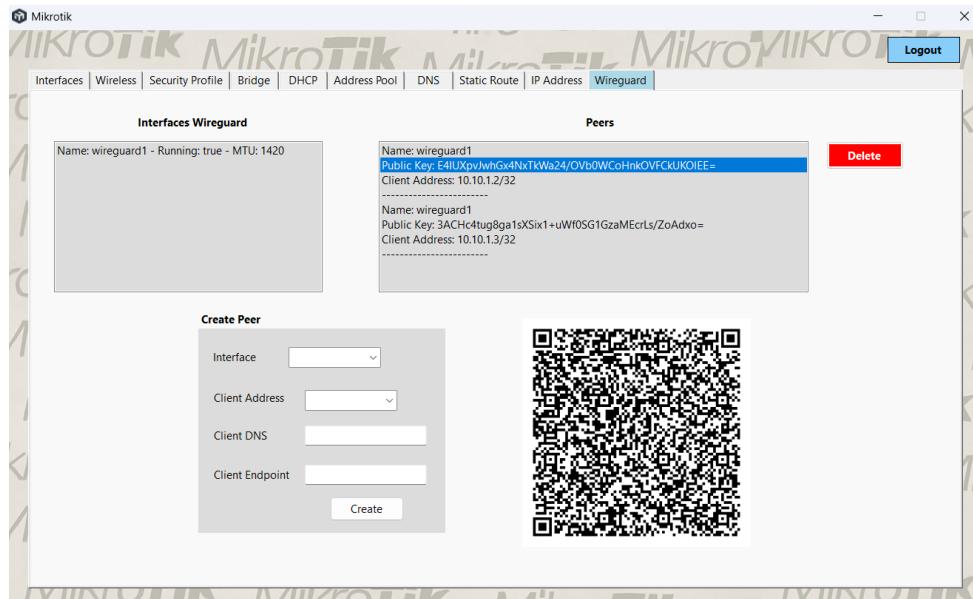


Figura 19: Interface Wireguard

Na criação de um *peer*, foram preenchidos todos os campos da secção "Create Peer" com os dados pretendidos. Após a criação, o *peer* foi automaticamente inserido na lista de *peers*, foi descarregado para o equipamento um ficheiro *.conf*, e apresentado na interface um código qr com os dados do *peer* criado, facilitando a configuração do cliente no WireGuard.

Interface	wireguard1
Client Address	10.10.1.4
Client DNS	8.8.8.8
Client Endpoint	10.20.139.17

Figura 20: Secção Create Peer

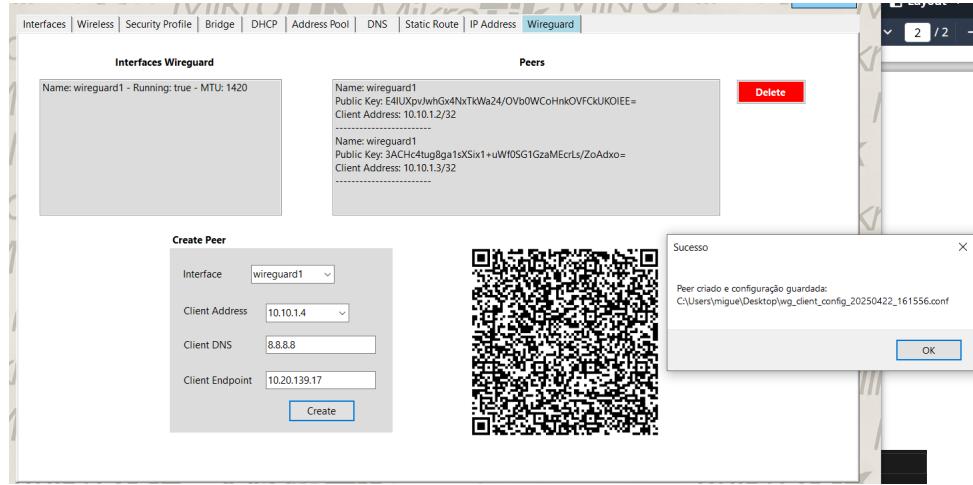


Figura 21: Peer criado

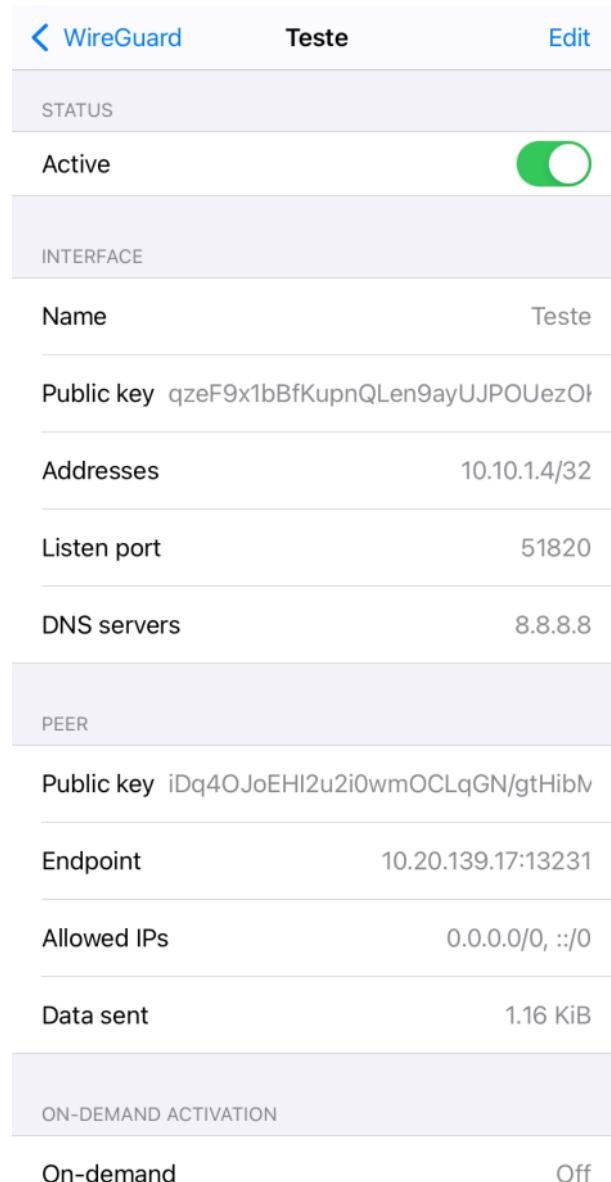
```

wg_client_config_20250422_161556.conf x
C: > Users > migue > Desktop > wg_client_config_20250422_161556.conf
1
2 [Interface]
3 ListenPort = 51820
4 PrivateKey = 8MVKeICWUi4KwXWowLeDm1djS3t5U1J0qgd0W6ty5HU=
5 Address = 10.10.1.4/32
6 DNS = 8.8.8.8
7
8 [Peer]
9 PublicKey = iDq4OJoEHII2u2i0wmOCLqGN/gtHibMR7JKzvSQcCwc=
10 AllowedIPs = 0.0.0.0/0, ::/0
11 Endpoint = 10.20.139.17:13231
12
13

```

Figura 22: Ficheiro .conf criado

Após ser gerado o QR *code* foi realizado um teste utilizando a aplicação oficial do Wireguard para equipamentos móveis. A configuração foi feita automaticamente através da leitura do código QR, como podemos ver na **fig.23**.

**Figura 23:** Teste QR code

4.2 Teste HTTPS

Para validar a comunicação segura com o dispositivo MikroTik via API REST, foi utilizado o Postman para realizar um pedido HTTPS ao endpoint `/rest/ip/dns`.

Na imagem seguinte, é possível observar o pedido GET efetuado ao endereço <https://10.20.139.17/rest/ip/dns>. A resposta retornada pelo servidor inclui os parâmetros de

configuração DNS em formato JSON, com um código de estado HTTP 200 OK, o que confirma o sucesso do pedido.

O objetivo do teste foi garantir que:

- A ligação utiliza o protocolo HTTPS com encriptação TLS;
- O certificado gerado anteriormente é reconhecido e utilizado corretamente;
- A API responde com sucesso ao pedido efetuado.

The screenshot shows a POST request to `https://10.20.139.17/rest/ip/dns`. The 'Params' tab is selected, showing a single query parameter 'Key'. The 'Body' tab shows a JSON payload with various configuration settings. The 'Headers' tab shows the response headers: 'HTTP Version' (1.1), 'Local Address' (192.168.79.253), 'Remote Address' (10.20.139.17), 'TLS Protocol' (TLSv1.2), 'Cipher Name' (ECDHE-RSA-AES128-GCM-SHA...), 'Certificate CN' (https), 'Issuer CN' (CA), and 'Valid Until' (Apr 13 09:49:12 2035 GMT). The status bar at the bottom indicates a 200 OK response with 341 ms latency and 706 B size.

Key	Value
Key	Value

```

1
2   "address-list-extra-time": "0s",
3   "allow-remote-requests": "true",
4   "cache-max-ttl": "1w",
5   "cache-size": "2048",
6   "cache-used": "69",
7   "doh-max-concurrent-queries": "50",
8   "doh-max-server-connections": "5",
9   "doh-timeout": "5s",
10  "dynamic-servers": "172.22.1.101,172.22.1.102",
11  "max-concurrent-queries": "100",
12  "max-concurrent-tcp-sessions": "20",

```

Figura 24: Pedido HTTPS realizado no Postman

5 Análise crítica e proposta de melhorias

De forma geral, o desenvolvimento deste trabalho decorreu de forma satisfatória, tendo sido possível alcançar os objetivos inicialmente propostos. Todas as funcionalidades planeadas foram implementadas com sucesso, bem como funcionalidades extras mencionadas anteriormente. Este processo permitiu não só consolidar conhecimentos previamente adquiridos, como também explorar novas abordagens e ferramentas.

Apesar dos resultados positivos, é importante reconhecer que existem sempre oportunidades para melhoria. Nesse sentido, alguns dos próximos passos que se poderiam considerar incluem a implementação de novas funcionalidades, o aperfeiçoamento do tratamento de erros e uma eventual adaptação da aplicação para funcionar noutras sistemas operativos, já que atualmente está limitada ao sistema operativo Windows.

6 Conclusão

Concluindo, o desenvolvimento do controlador SDN foi bem-sucedido, atingindo os objetivos propostos. A aplicação desenvolvida permite a gestão eficiente de equipamentos de rede MikroTik, com funcionalidades que abrangem a configuração e monitorização de interfaces, redes wireless, perfis de segurança, rotas estáticas, servidores DHCP e DNS, além da integração com o WireGuard para gestão de VPNs.

Todos os testes realizados confirmaram o funcionamento adequado do controlador, proporcionando uma solução robusta e prática para a automação de redes MikroTik. O trabalho cumpriu a proposta inicial, oferecendo uma ferramenta eficaz para a gestão centralizada de dispositivos de rede.

Bibliografia

- [1] IBM. What is software-defined networking (SDN)?, 2022, <https://www.ibm.com/think/topics/sdn>.
- [2] TechTarget. SDN controller (software-defined networking controller), 2023, <https://www.techtarget.com/searchnetworking/definition/SDN-controller-software-defined-networking-controller>.
- [3] Netmaker. What is the MikroTik RouterOS? Features & Capabilities, 2024, <https://www.netmaker.io/resources/mikrotik-routeros>
- [4] POSTMAN. What is an API?, 2025, <https://www.postman.com/what-is-an-api/>.
- [5] POSTMAN. What Is a REST API? Examples, Uses, and Challenges, 2020, <https://blog.postman.com/rest-api-examples/>
- [6] PrimaveraAcademy. Programação em C#, 2024, <https://campaigns.primaverabss.com/acton/media/17591/academy-programacao-c>
- [7] Tikoci. RouterOS API Schema Tools, 2025, <https://tikoci.github.io/restram/>
- [8] ChatGPT. 2025, <https://chatgpt.com/>

Anexos

Em anexo, encontra-se um vídeo com a demonstração dos testes realizados para a validação das funcionalidades implementadas, bem como um ficheiro de texto "Readme" da aplicação, devendo ser lido atentamente antes da sua utilização.

Vídeo:<https://youtu.be/WUTUS3tEQqY>