



# WIREGUARD

FAST, MODERN, SECURE VPN TUNNEL

project by:



## **Índice**

Introdução .....	3
O que é o WireGuard / Apresentação da Ferramenta.....	3
Arquitetura e Funcionamento.....	3
Casos Uso e Vantagens .....	4
Comparação com alternativas .....	5
Laboratórios Propostos .....	5
Instrumentos de Avaliação.....	12
Conclusão .....	12
Webgrafia .....	12

## **Introdução**

Com o crescimento do trabalho remoto e havendo a necessidade que este seja seguro, o uso de VPNs (Virtual Private Networks) é essencial. No nosso trabalho vamos usar o Wireguard, que é uma das soluções para esta necessidade. É uma solução moderna de VPN que se destaca pela sua simplicidade, alto desempenho e forte segurança criptográfica.

## **O que é o WireGuard / Apresentação da Ferramenta**

WireGuard é um protocolo e software VPN moderno, focado em:

- Simplicidade de código.
- Alta performance (kernel-level em Linux).
- Criptografia de última geração.
- Baixa latência e baixo overhead.
- Configuração minimalista.

Usa pacotes UDP que encapsulam a informação que é encriptada pelo WireGuard para ser enviada na comunicação. Usa UDP por ser mais rápido e não influencia na segurança quando transmite dados pelo túnel.

O WireGuard é Open Source e é suportado nativamente por vários Sistemas Operativos, como Linux, Windows, macOS, iOS e Android.

## **Arquitetura e Funcionamento**

O WireGuard opera na layer 3 e utiliza o modelo peer-to-peer. Cada dispositivo possui um par de chaves criptográficas (pública e privada) para autenticação.

- Criptografia moderna: Curve25519, ChaCha20, Poly1305, BLAKE2s, SipHash24, HKDF
- Modelo peer-to-peer: cada nó é simultaneamente cliente e servidor.
- Interface de rede virtual: cria uma interface de rede virtual.
- Roaming integrado: muda de IP sem perder a sessão.
- Criptokey Routing: cada chave pública está associada a um conjunto de IPs permitidos.

Funcionamento:

1. Cada peer gera um par de chaves (pública/privada).
2. Os peers trocam apenas as chaves públicas.
3. Cada peer define os IPs permitidos para o outro (AllowedIPs).
4. O tráfego é encapsulado e encriptado através de UDP.
5. A ligação é estabelecida automaticamente quando há tráfego.

## Casos Uso e Vantagens

Uso:

- VPN corporativa simples, rápida e Segura.
- VPN pessoal para segurança e privacidade.
- Acesso remoto seguro a servidores.
- Túneis seguros entre datacenters.
- VPN para dispositivos móveis (roaming eficiente).
- Proteção e Privacidade de tráfego em Redes Públicas.

Vantagens:

- Desempenho superior a OpenVPN e IPsec.
- Configuração simples.
- Código reduzido, facilitando auditoria.
- Segurança moderna com criptografia atualizada.
- Roaming automático, ideal para telemóveis.
- Cross-platform.

## Comparação com alternativas

Comparado com OpenVPN e IPsec, o WireGuard tende a ser mais rápido, mais seguro, e muito mais simples de configurar.

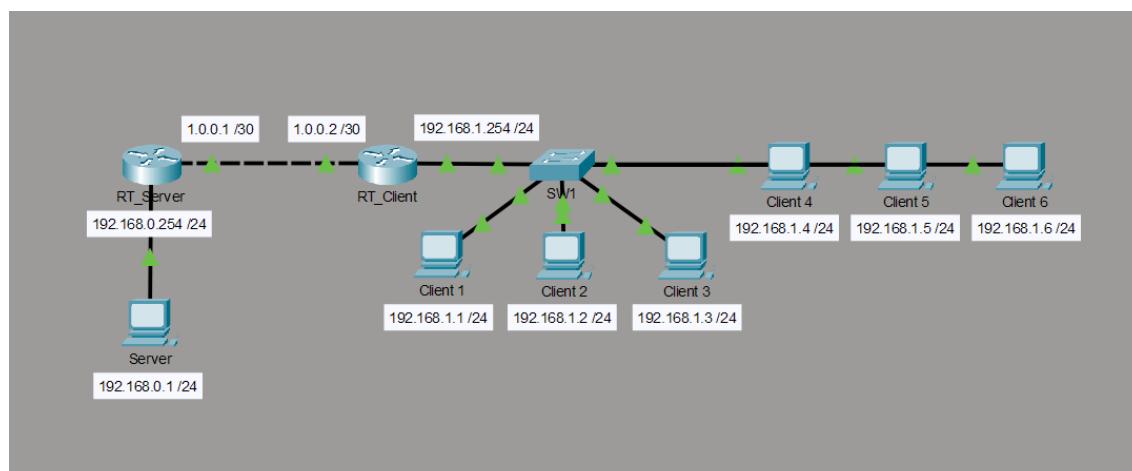
Critério	WireGuard	OpenVPN	IPsec
Desempenho	Muito alto	Médio	Alto
Complexidade	Baixa	Média	Alta
Criptografia	Moderna	Variável	Forte mas complexa
Código-fonte	Pequeno	Grande	Muito grande
Facilidade de configuração	Muito fácil	Média	Difícil
Suporte móvel	Excelente	Bom	Médio
Kernel Linux	Integrado	Não	Parcial

O WireGuard destaca-se pela simplicidade e eficiência, enquanto o OpenVPN e IPsec oferecem mais flexibilidade mas com maior complexidade.

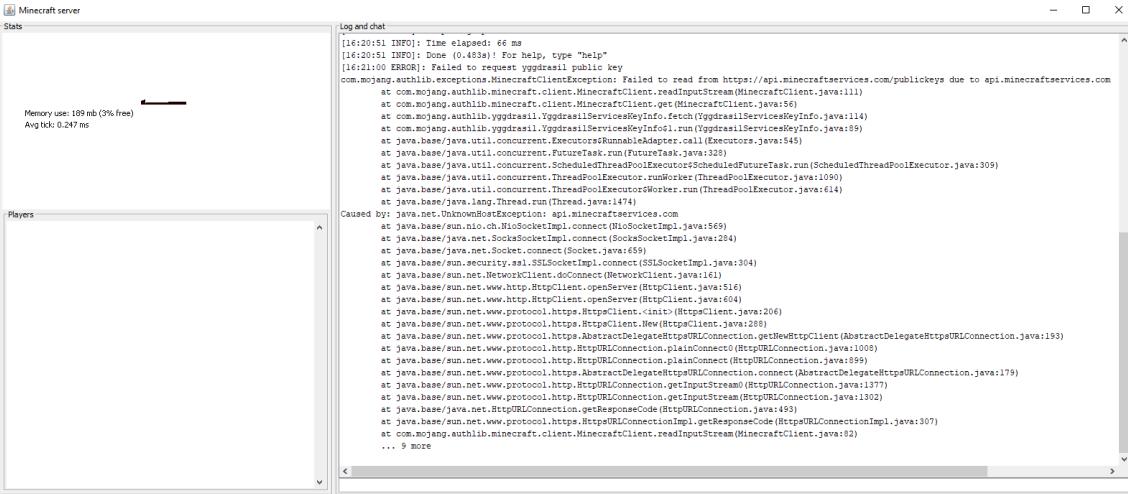
## Laboratórios Propostos

Para o nossa demonstração vamos estabelecer uma ligação através de um túnel com o Wireguard entre um Cliente e um Servidor e mostrar que só conseguimos entrar no servidor se tivermos com o WireGuard configurado entre o Cliente e o Servidor. Como Servidor vamos hospedar uma sessão de jogo, neste caso Minecraft.

Utilizámos a seguinte topologia de rede:



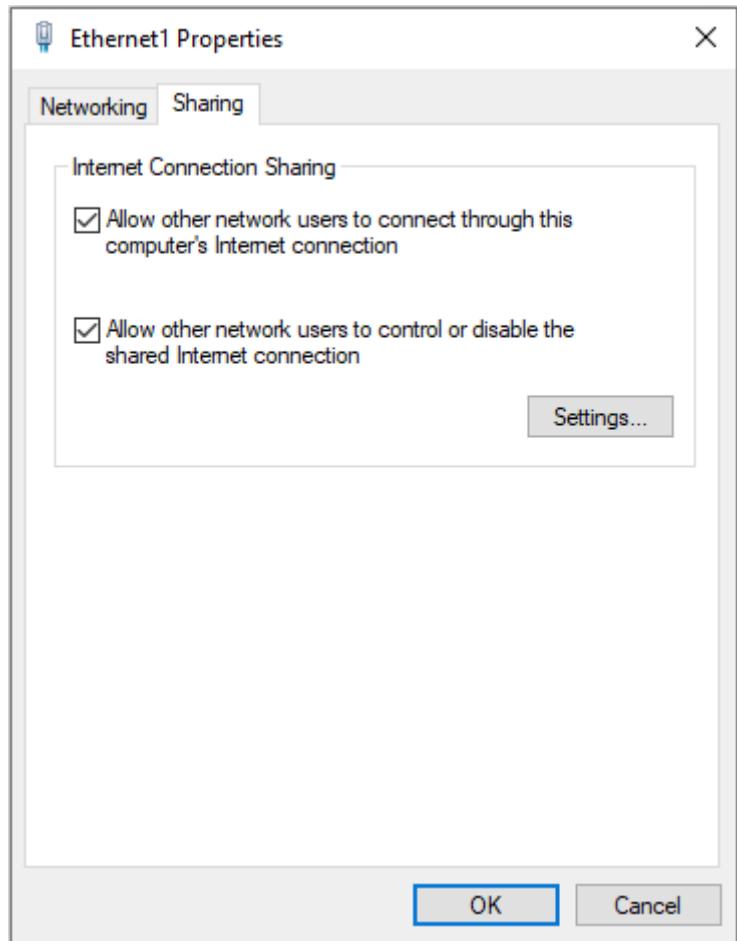
Primeiro, no Servidor tivemos que iniciar o servidor de Jogo (Minecraft):



The screenshot shows the 'Log and chat' window of the Minecraft server interface. The 'Stats' tab is selected, displaying memory usage (199 mb, 3% free) and average tick time (0.247 ms). The 'Players' tab is also visible. The main area contains a detailed stack trace of an exception:

```
[16:20:51 INFO]: Time elapsed: 66 ms
[16:20:51 INFO]: Done (0.439s)! For help, type "help"
[16:21:00 ERROR]: Failed to request yggdrasil public key
com.mojang.authlib.exceptions.MinecraftClientException: Failed to read from https://api.minecraftservices.com/publickeys due to api.minecraftservices.com
    at com.mojang.authlib.minecraft.client.MinecraftClient.readInputStream(MinecraftClient.java:111)
    at com.mojang.authlib.minecraft.client.MinecraftClient.get(MinecraftClient.java:56)
    at com.mojang.authlib.yggdrasil.YggdrasilClient.fetch(YggdrasilServicesKeyInfo.java:114)
    at com.mojang.authlib.yggdrasil.YggdrasilServicesKeyInfo.fetch(YggdrasilServicesKeyInfo.java:99)
    at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:545)
    at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:328)
    at java.base/java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThreadPoolExecutor.java:309)
    at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1090)
    at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:614)
    at java.base/java.lang.Thread.run(Thread.java:1474)
Caused by: javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target
    at java.base/sun.net.www.protocol.https.HttpsURLConnectionImpl.connect(HttpsURLConnectionImpl.java:569)
    at java.base/java.net.Socket.connect(Socket.java:1659)
    at java.base/java.net.SSLSocketImpl.connect(SSLSocketImpl.java:304)
    at java.base/sun.security.ssl.SSLSocketImpl.connect(SocketClient.java:161)
    at java.base/sun.net.NetworkClient.doConnect(NetworkClient.java:161)
    at java.base/sun.net.www.http.HttpClient.openServer(HttpClient.java:516)
    at java.base/sun.net.www.http.HttpClient.openServer(HttpClient.java:604)
    at java.base/sun.net.www.protocol.https.HttpsClient.openServer(HttpsClient.java:206)
    at java.base/sun.net.www.protocol.https.HttpsClient.NewHttpsClient.java:209)
    at java.base/sun.net.www.protocol.https.AbstractDelegateHttpsURLConnection.getNewHttpsClient(AbstractDelegateHttpsURLConnection.java:193)
    at java.base/sun.net.www.protocol.https.HttpsURLConnection_plainConnect0(HttpsURLConnection.java:1008)
    at java.base/sun.net.www.protocol.http.HttpURLConnection_plainConnect(HttpURLConnection.java:895)
    at java.base/sun.net.www.protocol.https.AbstractDelegateHttpsURLConnection.connect(AbstractDelegateHttpsURLConnection.java:179)
    at java.base/sun.net.www.protocol.http.HttpURLConnection_getInputStream0(HttpURLConnection.java:1377)
    at java.base/sun.net.www.protocol.http.HttpURLConnection.getInputStream(HttpURLConnection.java:1302)
    at java.base/sun.net.www.protocol.https.HttpsURLConnection.get.GetResponseCode(HttpsURLConnection.java:493)
    at java.base/sun.net.www.protocol.https.HttpsURLConnectionImpl.getgetResponseCode(HttpsURLConnectionImpl.java:307)
    at com.mojang.authlib.minecraft.client.MinecraftClient.readInputStream(MinecraftClient.java:92)
    ... 9 more
```

Do lado do Cliente podemos adicionar várias máquinas, mas isso depois faz com que a máscara da rede do WireGuard tenha que ser alterada. Para o nosso laboratório, como vamos fazer uma demonstração, vamos utilizar uma máscara fixa /28 (255.255.255.240) - 14 IP's utilizáveis. E vamos fazer a conexão de 6 Clientes diferentes ao nosso Servidor.

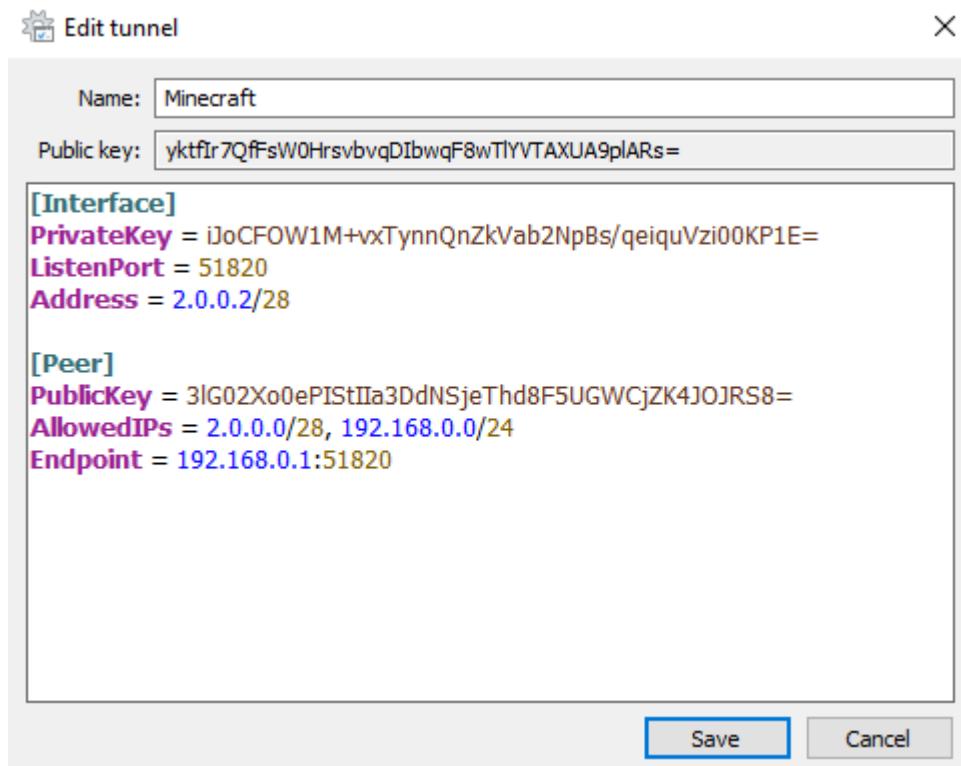


No servidor ativámos o ICS (Internet Connection Sharing) para que o tráfego dos peers WireGuard seja encaminhado pelo servidor para a LAN, permitindo que os dispositivos locais os vejam como máquinas da própria rede graças ao NAT e ao encaminhamento feitos pelo ICS.

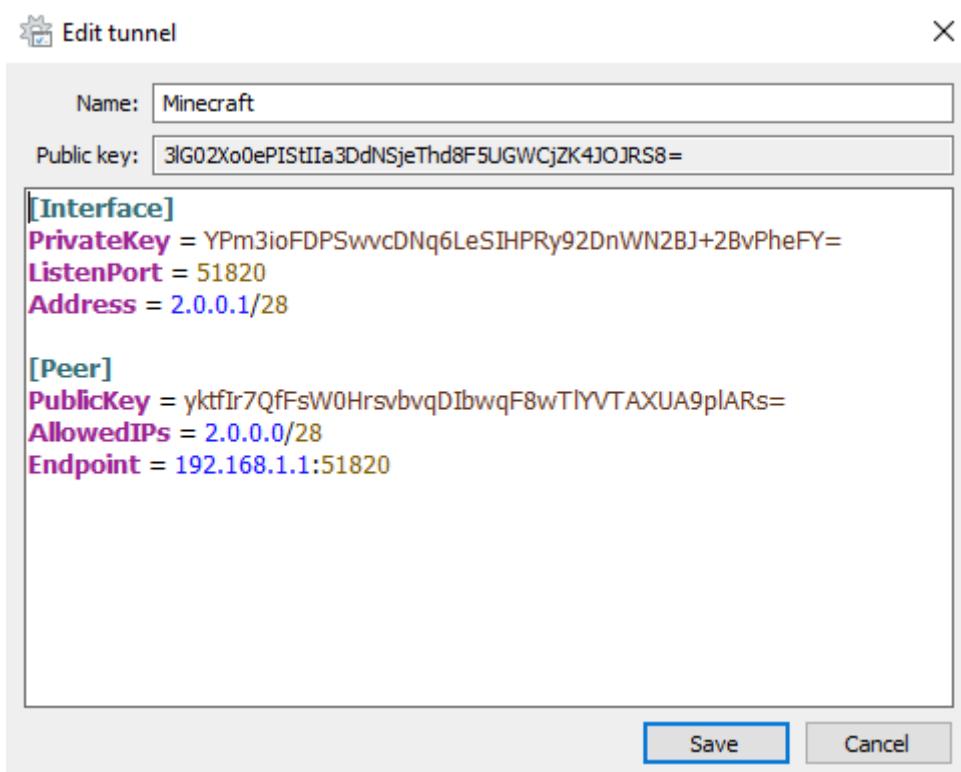
Agora temos que fazer as configurações do WireGuard:

Temos que partilhar a chave Pública do Servidor com o Cliente, a chave Pública do Cliente com o Servidor.

Cliente:

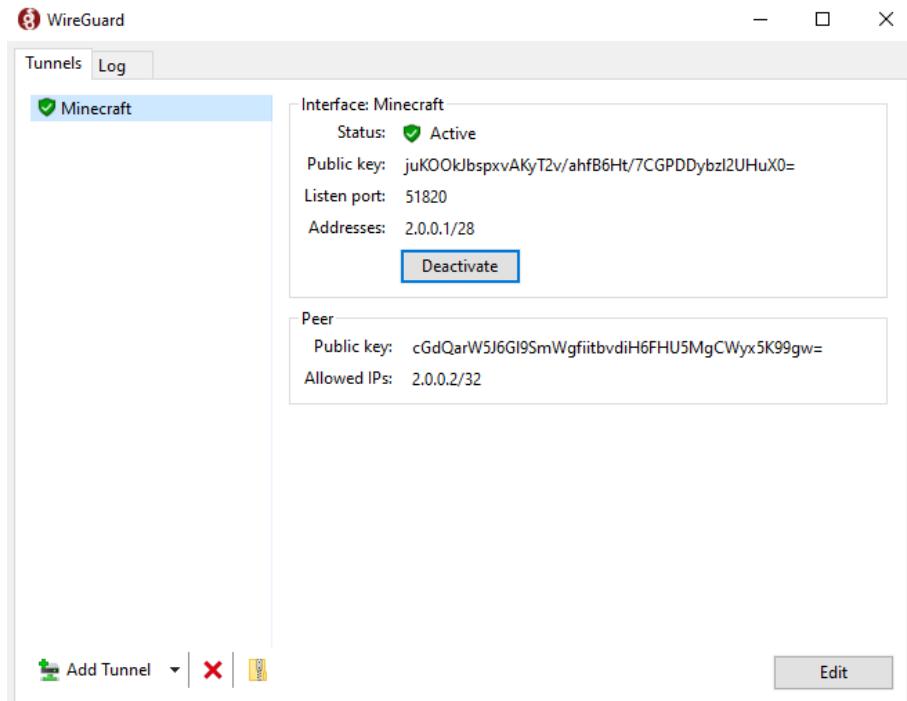


Server:



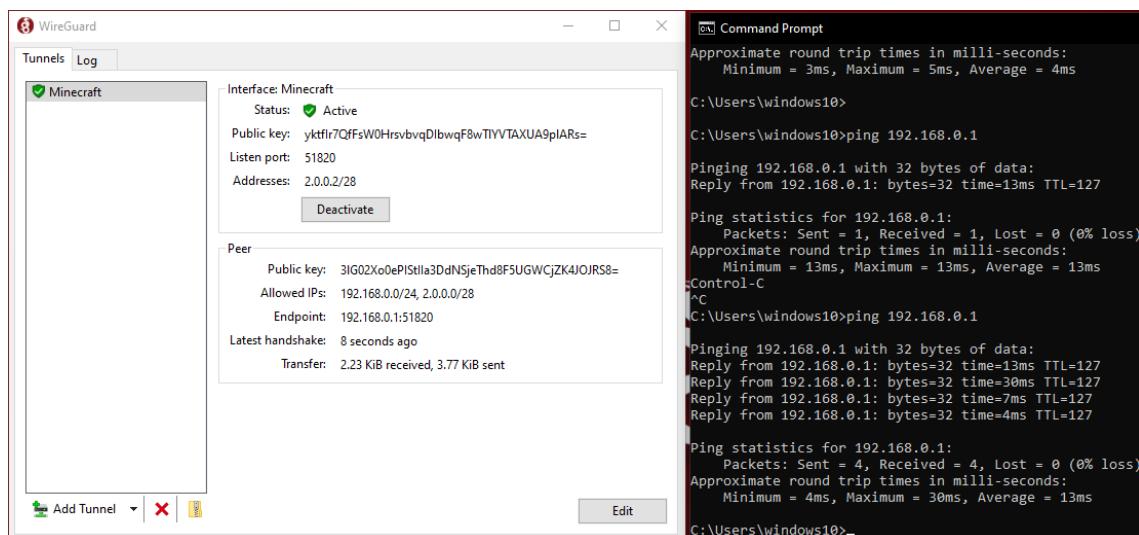
Com a configuração do WireGuard da parte do servidor precedemos à ligação do túnel.

Servidor:



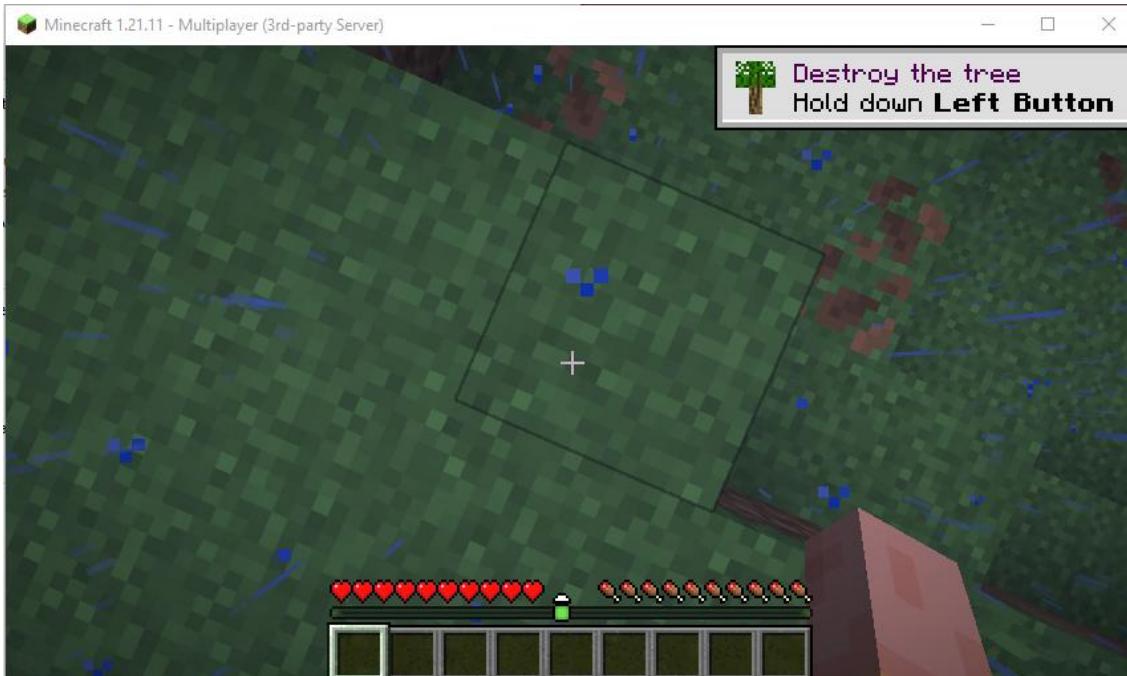
Com toda a configuração concluída, procedemos ao teste de ligação para confirmar a conectividade com o túnel ativo entre o Cliente e o Servidor.

Cliente:

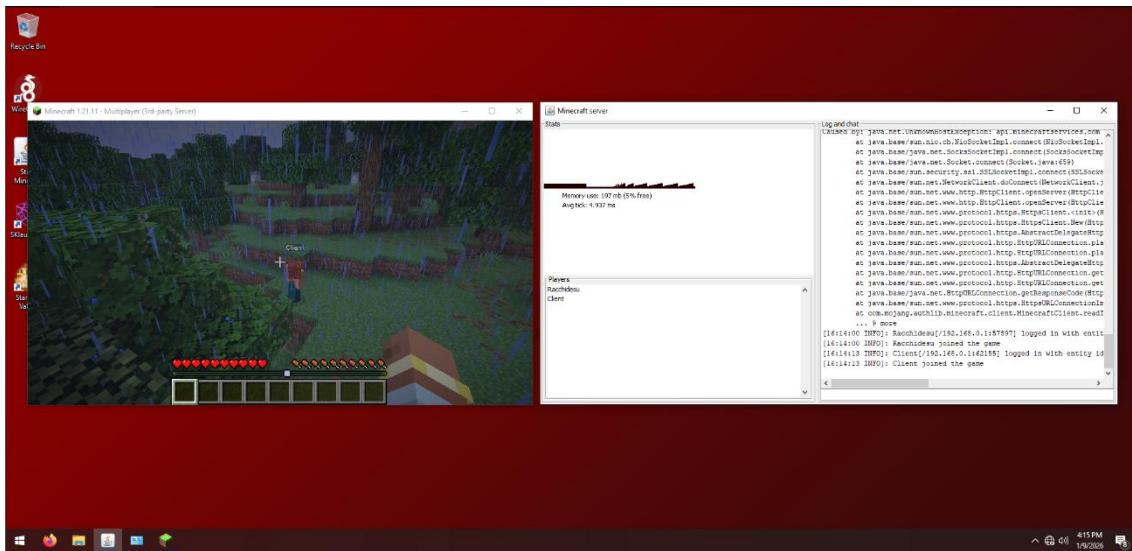


A comunicação entre as partes está estabelecida. Procedemos então à ligação ao Servidor para demonstrar que o funcionamento está garantido.

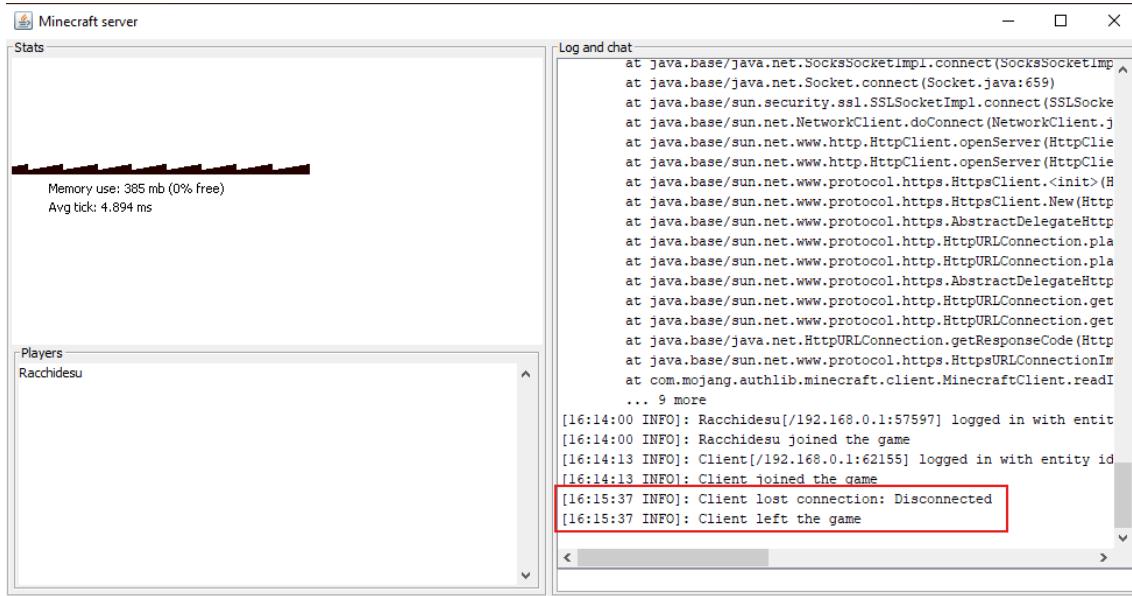
Cliente:



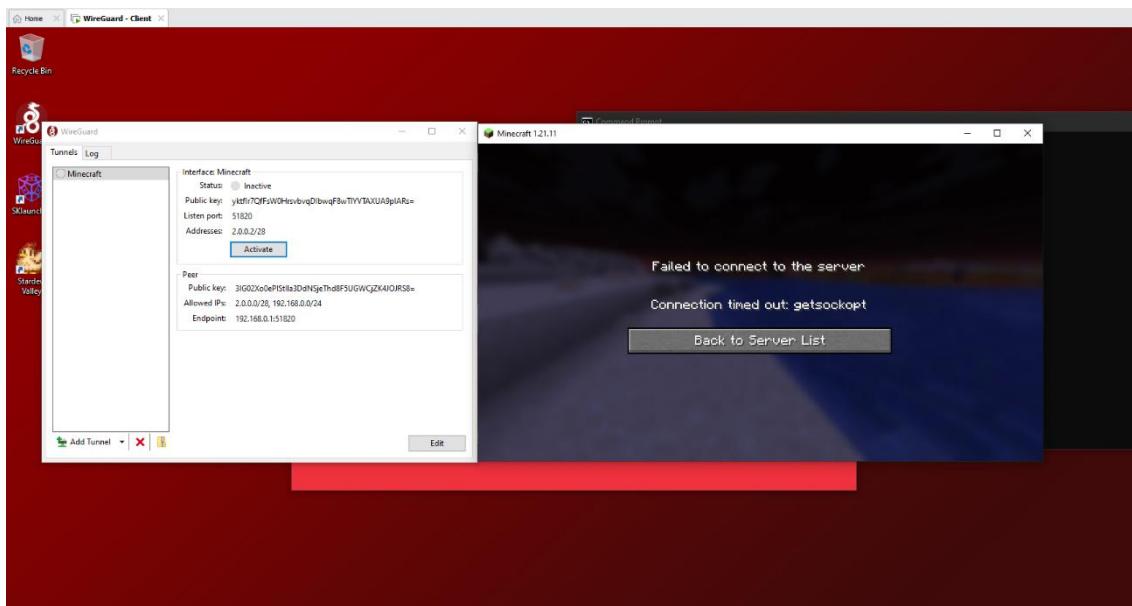
O servidor está a registar corretamente a entrada do Cliente, confirmando que a ligação foi estabelecida com sucesso.



Ao desativar o WireGuard no cliente, a consola do servidor indica de imediato a perda da ligação por parte desse cliente, confirmando a interrupção da sessão.



Procedemos à tentativa de ligação ao Servidor, porém a conexão não foi estabelecida.



### Proposta prática:

Instalar WireGuard numa OVA fornecida;

Configurar o WireGuard do lado do cliente;

Testar Conexão ao Servidor Minecraft;

## Instrumentos de Avaliação

KAHOOT - [Kahoot!](#)

<https://create.kahoot.it/share/redes-wireguard-quiz/121af377-fb80-43ef-9c36-ba4fdd38ec26>

## Conclusão

O WireGuard mostrou ser uma solução VPN moderna, rápida e fácil de configurar. No trabalho foi possível perceber como funciona, para que se usa, quais as suas vantagens e como se destaca face a outras alternativas. Através do laboratório prático confirmámos que o túnel criado garante segurança e controlo de acesso, permitindo a ligação ao servidor apenas quando o WireGuard está ativo. No geral, é uma ferramenta eficiente e adequada para ambientes que exigem segurança e bom desempenho.

## Webgrafia

[WireGuard: fast, modern, secure VPN tunnel](#)

[PiVPN + WireGuard Complete Setup - Build Your Own VPN Server!](#)

[What is Wireguard? A "New" VPN Protocol + How it Compares to OpenVPN](#)

[ChatGPT](#)

[Microsoft Copilot: Your AI companion](#)