INSTITUTO POLITÉCNICO DE BEJA
Escola Superior de Tecnologia e Gestão
Licenciatura em Engenharia Informática
Projeto Final de Curso

Desenvolvimento de um sistema de comunicações seguras

Martinho José Novo Caeiro - 23917
Paulo António Tavares Abade - 23919
Rafael Conceição Narciso - 24473

Beja, julho de 2025

INSTITUTO POLITÉCNICO DE BEJA
Escola Superior de Tecnologia e Gestão
Licenciatura em Engenharia Informática
Projeto Final de Curso

Desenvolvimento de um sistema de comunicações seguras

Martinho José Novo Caeiro - 23917
Paulo António Tavares Abade - 23919
Rafael Conceição Narciso - 24473

Orientador: Professor Rui Silva

Beja, julho de 2025

Resumo

Neste relatório será abordado o processo de criação de uma solução de comunicações seguras, que permita a troca de mensagens de texto entre os seus utilizadores. Este relatório foi realizado no âmbito da Unidade Curricular de Estágio ou Projeto (IPBeja, 2025).

Keywords: aplicações, cibersegurança, comunicações, criptografia, c#, python, bash, alma-linux, wireguard, sqlite, kotlin

Abstract

In this report, we will address the creation process of a secure communication solution that allows text message exchange between its users. This report was carried out within the scope of the Curricular Unit of Internship or Project (IPBeja, 2025).

Keywords: applications, cybersecurity, communications, cryptography, c#, python, bash, almalinux, wireguard, sqlite, kotlin

Índice

1 Introdução **1**

2 Analise de Requisitos **2**

3 Tecnologias Utilizadas **3**

4 Arquitetura do Sistema **4**

5 Modulos do Sistema **5**

5.1 Interface para Computador 5

5.2 Interface para Android 5

5.3 Módulo da VPN 6

5.4 Base de Dados 6

6 Desenvolvimento **8**

6.1 Interface para Computador 8

6.1.1 Ecrã de Login 8

6.1.2 Ecrã de Lista de Chats 9

6.1.3 Ecrã de Chat 10

6.1.4 Ecrã de Configuração WireGuard 11

6.2 Interface para Android 11

6.3 AlmaOS - Serviço de VPN 11

7 Problemas Encontrados **15**

7.1 Problemas de Conexão 15

7.2 Problemas na aplicação Windows/Android 15

8 Testes **16**

8.1 Servidor 16

8.2 Comunicações 17

9 Conclusão **18**

Índice de Figuras

1 Cronograma do Projeto 1

2 Processo de Encriptação das Mensagens 4

3 Processo de Decriptação das Mensagens 5

4 Estrutura da Base de Dados 6

5 Ecrã de Login - Computador 8

6 Ecrã de Lista de Chats - Computador 9

7 Ecrã de Chat - Computador 10

8 Ecrã de Configuração WireGuard - Computador 11

9 Teste com Servidor 16

10 Teste com Wireshark 17

1 Introdução

Para a realização do projeto é necessário desenvolver um sistema de comunicações seguras, que permita a troca de mensagens de texto entre os seus utilizadores. Para tal, é necessário implementar um sistema de autenticação de utilizadores, que permita a criação de contas de utilizador e a autenticação dos mesmos. O sistema deve ser capaz de garantir a confidencialidade, integridade e autenticidade das mensagens trocadas entre os utilizadores. As tecnologias a utilizadas no desenvolvimento deste projeto são: WireGuard (WireGuard, 2025), SQLite (SQLite, 2025), C# (Microsoft, 2025), Kotlin (JetBrains, 2025), Python (Foundation, 2025), Bash (Bash, 2025), AlmaOS (AlmaOS, 2025). Em relação a criptografia, serão utilizados os algoritmos Rijndael (vencedor do AES) e Serpent (segundo lugar do AES), que são algoritmos de criptografia simétrica.

Para este projeto, foi decidido utilizar uma variação da metodologia do *SCRUM*, onde eram realizadas reuniões semanais ao ínicio, e depois foi sendo feito para ocorrer mais tempo entre reuniões, de forma a que fosse possível ter mais tempo para o desenvolvimento das tarefas. Os três membros da equipa foram divididos entre as três áreas do projeto, nomeadamente: desenvolvimento da aplicação Windows, desenvolvimento da aplicação Android e por último, a configuração da infraestrutura da rede.

O cronograma do projeto foi dividido em duas fases principais: Análise de Requisitos, Desenvolvimento, como é possível ver na figura 1. O relatório foi modificado a cada semana, acrescentando novas informações e atualizações sobre o progresso do projeto.

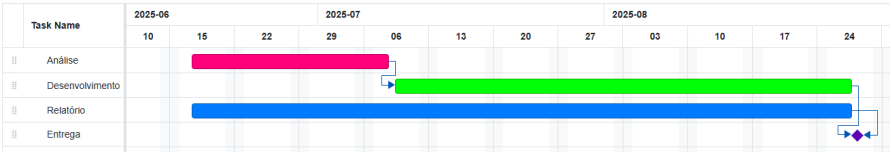


Figura 1: Cronograma do Projeto

Este projeto ficou guardado no repositório do GitHub da equipa, onde pode ser consultado por outros alunos e professores, a partir do seguinte link:
https://github.com/MartinhoCaeiro/Projeto-Cripto.

2 Analise de Requisitos

Para a realização deste projeto, foram necessários os seguintes requisitos:

- **Requisitos Funcionais:**
  - O sistema deve permitir a criação de contas de utilizador.
  - O sistema deve permitir a autenticação de utilizadores.
  - O sistema deve permitir o envio e receção de mensagens entre utilizadores.
  - O sistema deve garantir a confidencialidade, integridade e autenticidade das mensagens trocadas.
- **Requisitos Não Funcionais:**
  - O sistema deve ser seguro e resistente a ataques.
  - O sistema deve ser fácil de usar e intuitivo.
  - O sistema deve ser escalável e capaz de suportar um grande número de utilizadores.

3 Tecnologias Utilizadas

Para o desenvolvimento deste projeto, foram utilizadas as seguintes tecnologias:

- **Linguagens de Programação:** C#, Kotlin
- **Frameworks:** .NET, WireGuard
- **Base de Dados:** SQLite
- **Criptografia:** Rijndael (Vencedor AES), Serpent (Segundo lugar AES)
- **Ferramentas de Desenvolvimento:** Git, Visual Studio Code, Android Studio
- **Serviços de Hospedagem:** GitHub

4 Arquitetura do Sistema

O sistema é composto por uma aplicação que funcionará como um chat, onde os utilizadores que estiverem registado na VPN poderão comunicar entre si. Dentro da aplicação, os utilizadores podem saber quem está associado a um endereço IP da VPN, e assim enviar mensagens para esse utilizador. Ao enviar uma mensagem, a aplicação irá contatar o servidor da VPN, para obter o caminho até ao destinatário, e assim enviar a mensagem. Essa mensagem será encriptada antes de ser enviada, para garantir a confidencialidade e integridade da mensagem. Existe a variação da encriptação utilizada na mensagem, de acordo com uma lógica pré-definida na aplicação, sendo que os métodos de encriptação utilizados são o AES e o Serpent.

Sendo assim, a parte de encriptação das mensagens será feita como está demonstrado na figura 2. A mensagem será encriptada consoante um número pseudo-aleatório gerado pela aplicação, sendo assim escolhido o algoritmo de encriptação a utilizar naquele momento, o AES256 ou Serpent. Para encriptar, é necessário fornecer também uma chave de encriptação. Nesta fase, a chave de encriptação é "***Spartacus***".

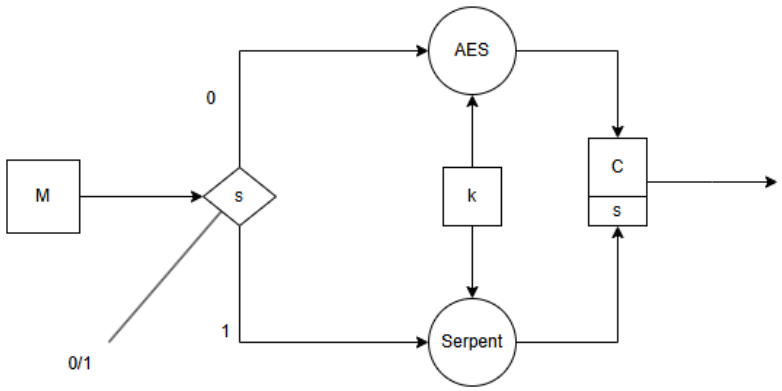


Figura 2: Processo de Encriptação das Mensagens

Para desencriptar, é necessário fazer o processo inverso. Desta forma, o número pseudo-aleatório está anexado à mensagem encriptada, permitindo assim descobrir qual o algoritmo de encriptação utilizado. É possível ver este processo na figura 3.

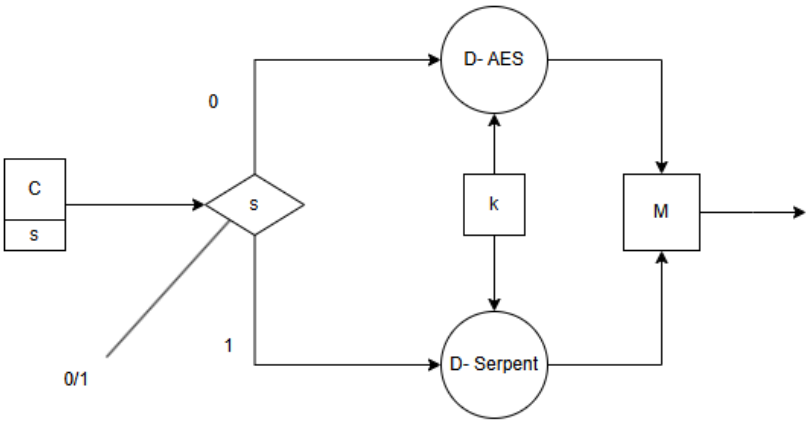


Figura 3: Processo de Decriptação das Mensagens

5 Modulos do Sistema

5.1 Interface para Computador

A interface para computador será desenvolvida em C#, cuja qual é uma linguagem rápida e fácil de aprender, e que permite o desenvolvimento de aplicações desktop de forma rápida e eficiente. A aplicação será desenvolvida utilizando o framework .NET. Esta interface é composta por 4 ecrãs principais:

- **Ecrã de Login:** onde o utilizador pode autenticar-se na aplicação utilizando as suas credenciais, tambem é possivel ver o status da ligação WireGuard.
- **Ecrã de Lista de Chats:** onde o utilizador pode ver a lista de chats existentes, bem como criar novos chats.
- **Ecrã de Chat:** onde o utilizador pode enviar e receber mensagens de outros utilizadores.
- **Ecrã de Configuração WireGuard:** onde o utilizador pode criar um ficheiro para configuração do WireGuard.

5.2 Interface para Android

A interface para Android será desenvolvida em Kotlin, que é uma linguagem de programação moderna e concisa, que permite o desenvolvimento de aplicações Android de forma rápida e

eficiente.

5.3 Módulo da VPN

A VPN estará alojada num servidor com o sistema operativo AlmaOS 8.10 (AlmaOS, 2025), e será responsável por gerir as ligações dos utilizadores à VPN, bem como a autenticação dos mesmos. O serviço escolhido foi o WireGuard, que é um serviço de VPN de código aberto, leve e de alto desempenho, que utiliza criptografia moderna para garantir a segurança das comunicações. É necessário configurar previamente o WireGuard no dispositivo do utilizador, para que este possa estabelecer uma ligação à VPN.

TODO Tentar fazer com que seja feito da maneira mais automatizada possível.

5.4 Base de Dados

A base de dados será utilizada para armazenar as informações dos utilizadores, como as suas credenciais, bem como as mensagens trocadas entre os utilizadores, esta será implementada utilizando o SQLite. A base de dados irá ter a seguinte estrutura:

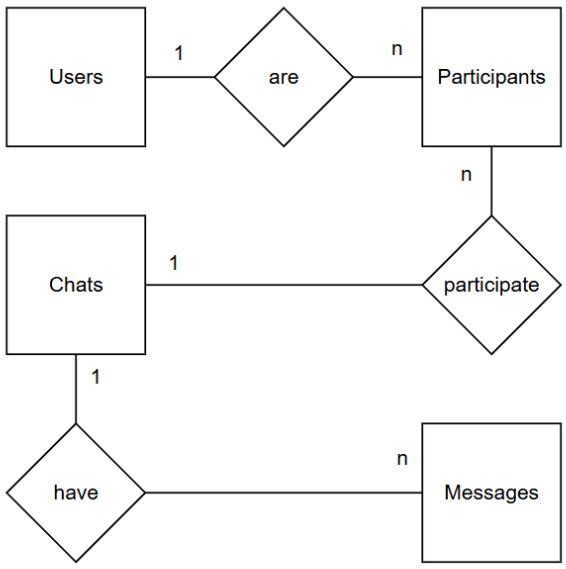


Figura 4: Estrutura da Base de Dados

Sendo que a base de dados é composta por 4 tabelas, cada uma com as suas respetivas colunas, sendo que as tabelas são:

- **User:**
  - UserID (chave primária)
  - Username (string)
  - Password (string)
- **Chat:**
  - ChatID (chave primária)
  - Name (string)
  - AdminID (chave estrangeira, referência à tabela User)
- **Participant:**
  - ParticipantID (chave primária)
  - ChatID (chave estrangeira, referência à tabela Chat)
  - UserID (chave estrangeira, referência à tabela User)
- **Message:**
  - MessageID (chave primária)
  - ParticipantID (chave estrangeira, referência à tabela Participant)
  - Content (string)
  - Date (data e hora da mensagem)
  - SenderUserID (chave estrangeira, referência à tabela Participant)

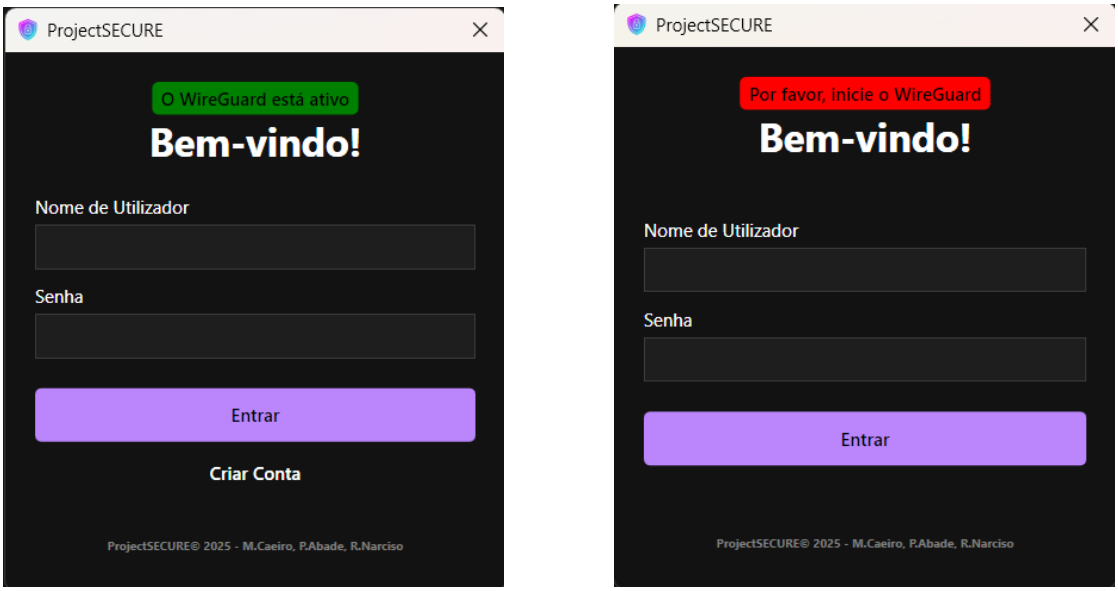
6 Desenvolvimento

6.1 Interface para Computador

Para o desenvolvimento da interface para computador, foi utilizado o Visual Studio Code, que é um ambiente de desenvolvimento integrado (IDE) da Microsoft.

6.1.1 Ecrã de Login

O ecrã de login é composto por um formulário onde o utilizador pode inserir as suas credenciais, e um botão para autenticar-se na aplicação. Caso não tenha uma conta criada apenas tem que preencher o formulario e clicar no botão de criar conta, esta ação só é possivel com o Wireguard ativado. Também é possivel ver o estado da ligação WireGuard, caso esteja ligado ou desligado.

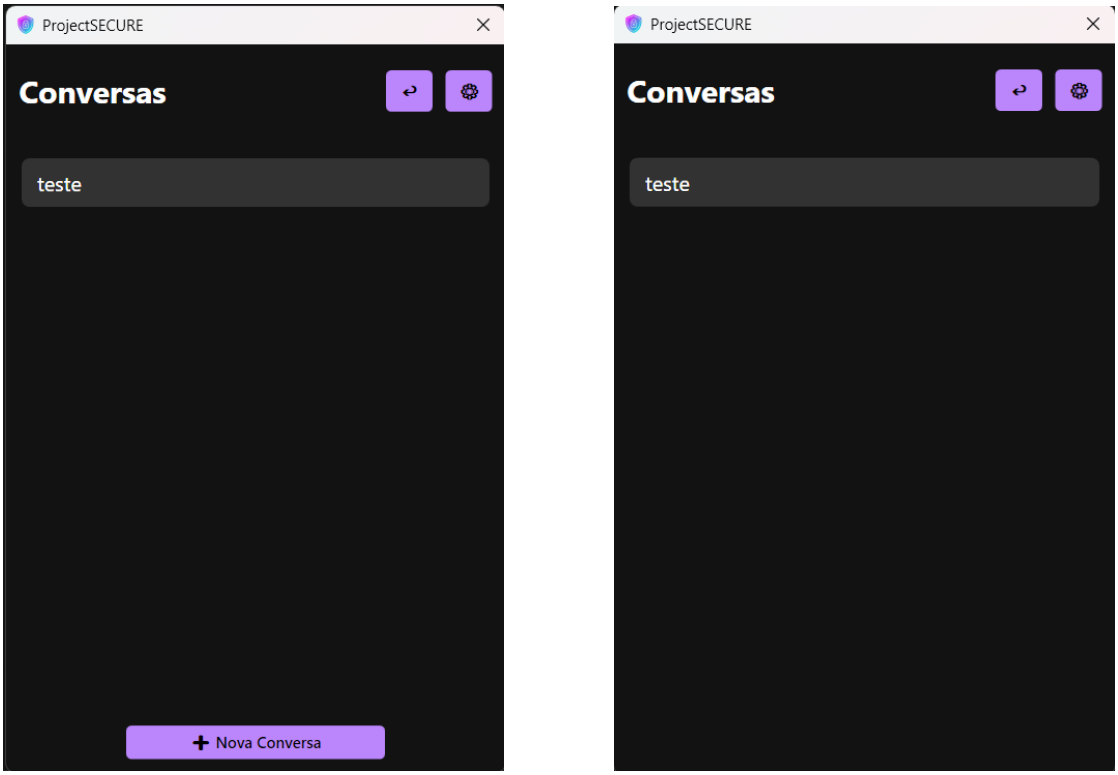


(a) WireGuard Ativado (b) Wireguard Desativado

Figura 5: Ecrã de Login - Computador

6.1.2 Ecrã de Lista de Chats

O ecrã de lista de chats é composto por uma lista de chats existentes e um botão para criar novos chats, este só pode ser criado com o WireGuard ativado. No seu canto superior direito, existe um botão para aceder às definições da aplicação, onde o utilizador pode configurar se quer a aplicação em modo escuro ou claro, e no seu canto inferior esquerdo existe um botão para terminar sessão caso deseja mudar de utilizador.

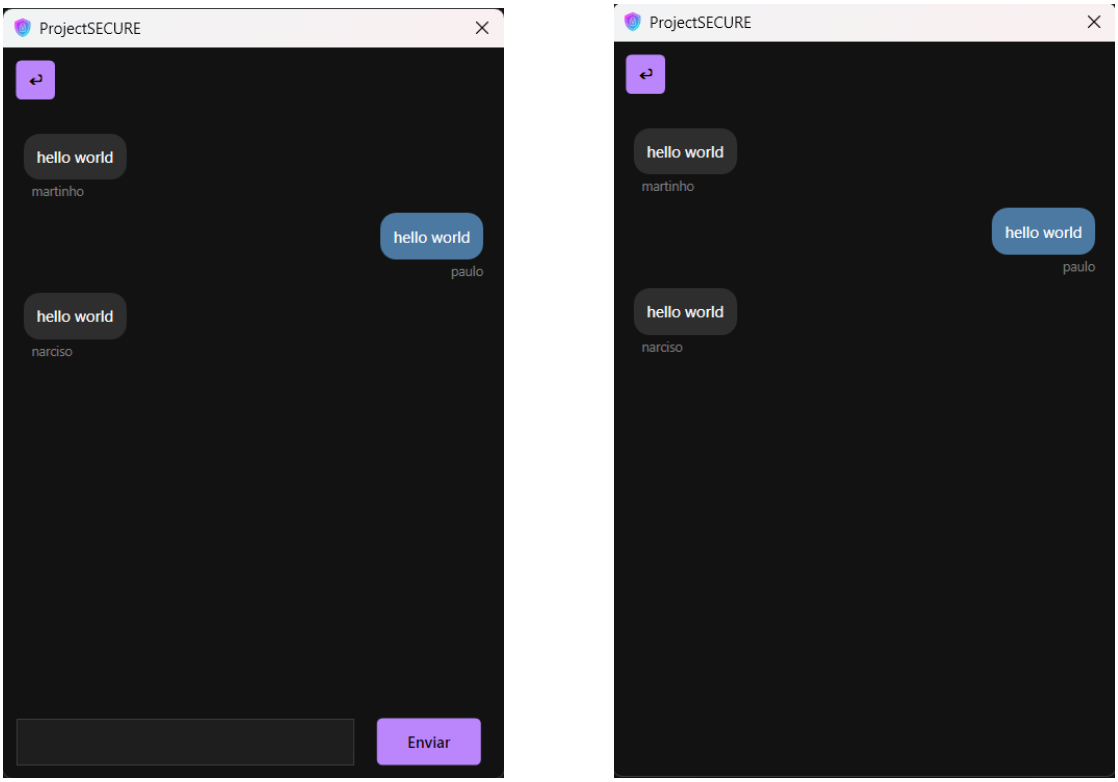


(a) WireGuard Ativado (b) Wireguard Desativado

Figura 6: Ecrã de Lista de Chats - Computador

6.1.3 Ecrã de Chat

O ecrã de chat é composto por uma lista de mensagens trocadas entre os utilizadores, mensagens recebidas ficam do lado esquerdo e mensagens enviadas no lado direito. Apenas é possivel enviar mensagens com o WireGuard ativado, para enviar uma mensagem basta escrever no campo de texto e clicar no botão de enviar.



(a) WireGuard Ativado (b) Wireguard Desativado

Figura 7: Ecrã de Chat - Computador

6.1.4 Ecrã de Configuração WireGuard

O ecrã de configuração WireGuard é composto por um formulário onde o utilizador pode inserir as suas credenciais, e um botão para criar o ficheiro de configuração do WireGuard.

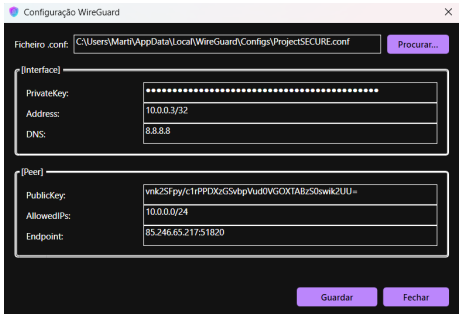


Figura 8: Ecrã de Configuração WireGuard - Computador

6.2 Interface para Android

Para o desenvolvimento da interface para Android, foi utilizado o Android Studio, que é um ambiente de desenvolvimento integrado (IDE) oficial para o sistema operativo Android.

6.3 AlmaOS - Serviço de VPN

Para configurar corretamente o serviço de VPN, é necessário instalar o WireGuard no servidor AlmaOS 8.10, para isso ser feito, foi necessário seguir os seguintes passos:

1. Adicionar o repositório EPEL - sudo dnf install epel-release
2. Adicionar o repositório ELREPO - sudo dnf install https://www.elrepo.org/elrepo-release-8.el8.elrepo.noarch.rpm
3. Ativar o CodeReady Builder - sudo /usr/bin/crb enable
4. Atualizar os metadados - sudo dnf makecache
5. sudo dnf –enablerepo=elrepo install kmod-wireguard -y
6. Instalar o WireGuardTools - sudo dnf install wireguard-tools -y

Para verificar se o WireGuard está instalado corretamente, pode-se utilizar o comando **"sudo modprobe wireguard"**, se este não devolver nenhum output, significa que o WireGuard está instalado corretamente.

Agora, para configurar o WireGuard, é necessário criar as chaves de criptografia, para isso é necessário ir para a pasta **/etc/wireguard** e executar o seguintes comando:

```
wg genkey | tee server_private.key | wg pubkey > server_public.key
chmod 600 server_private.key
```

Isto irá gerar duas chaves, uma privada e uma pública, que serão utilizadas para autenticar os utilizadores na VPN.

Após isso, é necessário criar o ficheiro de configuração do WireGuard, onde estará definida a configuração da VPN em si, como o endereço IP da VPN, a porta de escuta, as chaves de criptografia, entre outros. Para isso, é necessário criar o ficheiro **wg0.conf** na pasta /**etc**/**wireguard**, e adicionar o seguinte conteúdo:

```
[Interface]
PrivateKey = <Chave Privada do Servidor>
Address = 10.0.0.1/24
ListenPort = 51820
SaveConfig = true
[Peer]
PublicKey = <Chave Pública do Cliente>
AllowedIPs = 10.0.0.2/32
```

Para garantir o bom funcionamento da VPN, é necessário ativar o encaminhamento de endereços IP, para assim permitir o tráfego de rede. Para isso, é necessário fazer o seguinte comando:

```
# Para permitir o tráfego de rede
echo "net.ipv4.ip_forward = 1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p

# Para permitir o tráfego pela porta 51820
sudo firewall-cmd --add-masquerade --permanent
sudo firewall-cmd --add-port=51820/udp --permanent
sudo iptables -t nat -A POSTROUTING -s 10.0.0.0/24 -o enp0s3 -j MASQUERADE
sudo firewall-cmd --reload
```

Para ser possível encaminhar o tráfego de rede, é necessário adicionar uma regra ao Router para que, todos os pacotes que cheguem à porta 51820 sejam encaminhados para o servidor WireGuard. Por fim, basta iniciar o serviço do WireGuard, para isso é necessário executar o seguinte comando:

```
sudo systemctl enable --now wg-quick@wg0
```

Do lado do cada cliente, vai ser necessário configurar o WireGuard, de maneira a que apenas a informação proveniente da aplicação desenvolvida seja enviada através da VPN, onde assim serão poupados recursos do dispositivo que está a fazer o papel de servidor. Para isso, foi criado um *script* que cria o ficheiro de configuração do WireGuard para o cliente, sendo que este só deve adicionar o ficheiro de configuração na aplicação do WireGuard. O *script* ainda automatiza a adição do cliente nas configurações do servidor.

Para conseguir estabelecer a troca de ficheiros entre o servidor e o cliente, foi necessário implementar um *script* que gerencia as trocas de ficheiros da base de dados entre os clientes e o servidor. Esse script foi feito em Python, utiliza a biblioteca Flask para criar uma API REST que permite a comunicação entre o cliente e o servidor. Foi necessário permitir a porta 8000 no servidor, para que o cliente possa comunicar com o servidor.

```
sudo firewall-cmd --add-port=8000/tcp --permanent
sudo firewall-cmd --reload
```

7 Problemas Encontrados

Durante o desenvolvimento do projeto, foram encontrados alguns problemas, que foram resolvidos da seguinte forma:

7.1 Problemas de Conexão

Logo no inicio da implementação da VPN foi necessário achar uma solução para a configuração de encaminhamento de pacotes, pois o servidor Linux não possui um endereço IP público próprio. Para contornar este problema, foi necessário configurar o encaminhamento de pacotes no router, de forma a que todos os pacotes que chegassem à porta 51820 fossem encaminhados para o servidor Linux, e este resolveria a situação encaminhando o resto dos pacotes pela VPN. Outro problema encontrado foi a dificuldade em encontrar uma maneira para estabelecer a conexão pela primeira vez, ou seja, enviar o ficheiro de configuração da VPN para o dispositivo do utilizador, de forma a que este pudesse estabelecer a conexão com o servidor. Para resolver este problema, foi necessário criar um *script* que gerasse o ficheiro de configuração do WireGuard, e que fosse possível enviar esse ficheiro para o dispositivo do utilizador (#TODO).

7.2 Problemas na aplicação Windows/Android

Durante o desenvolvimento da aplicação, foram encontrados diversos problemas, como por exemplo:

- Problemas de compatibilidade entre tipos de base de dados (SQLite e Schemas)
  - Incompatibilidade de nomes entre tabelas
  - Configurações por defeito diferentes entre as plataformas.
- Sincronização de dados entre dispositivos.
- Visualização incorreta das informações
- Permissões do Android
- Impossibilidade de modificar a base de dados enquanto a aplicação está a correr

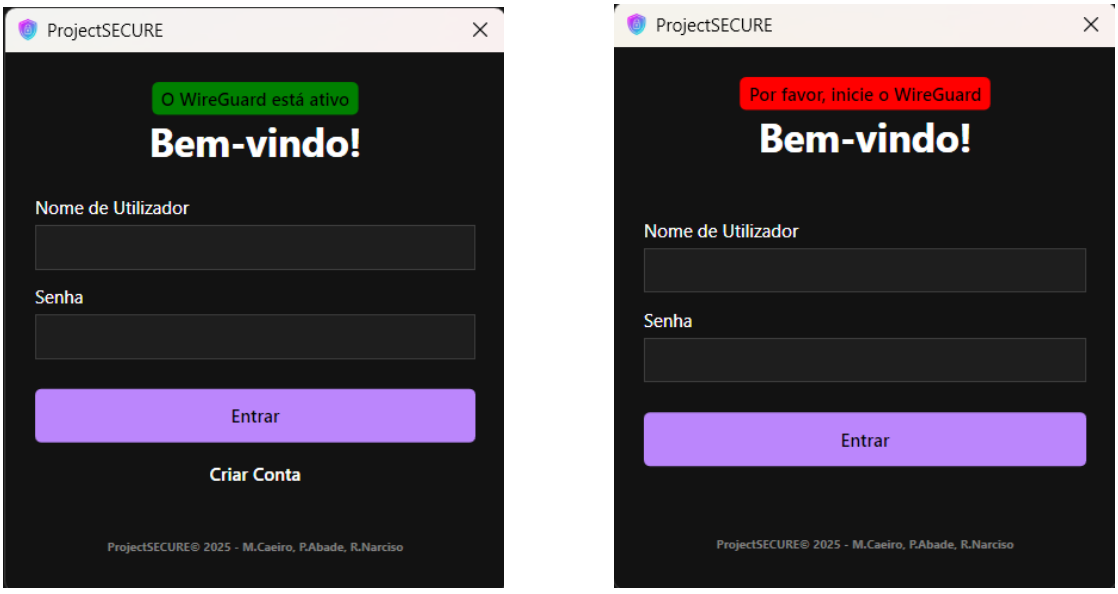
8 Testes

Os testes realizados tem como base 2 metodos:

- Verificação da base de dados através do servidor
- Sniffing das comunicações com o uso do Wireshark

8.1 Servidor

Os testes realizados no servidor foram focados na verificação da base de dados e na monitorização das comunicações. Para a verificação da base de dados, é feita a tentativa de ler o ficheiro enviado, e como é possivel verificar nas figuras abaixo, ao ler o ficheiro sem criptografia, é possível visualizar o seu conteúdo perfeitamente, ao ler o ficheiro com criptografia, não é possível visualizar o seu conteúdo pois o servidor guarda a base de dados sem desencriptar, isto leva a que o ficheiro nem seja reconhecido como uma base de dados.

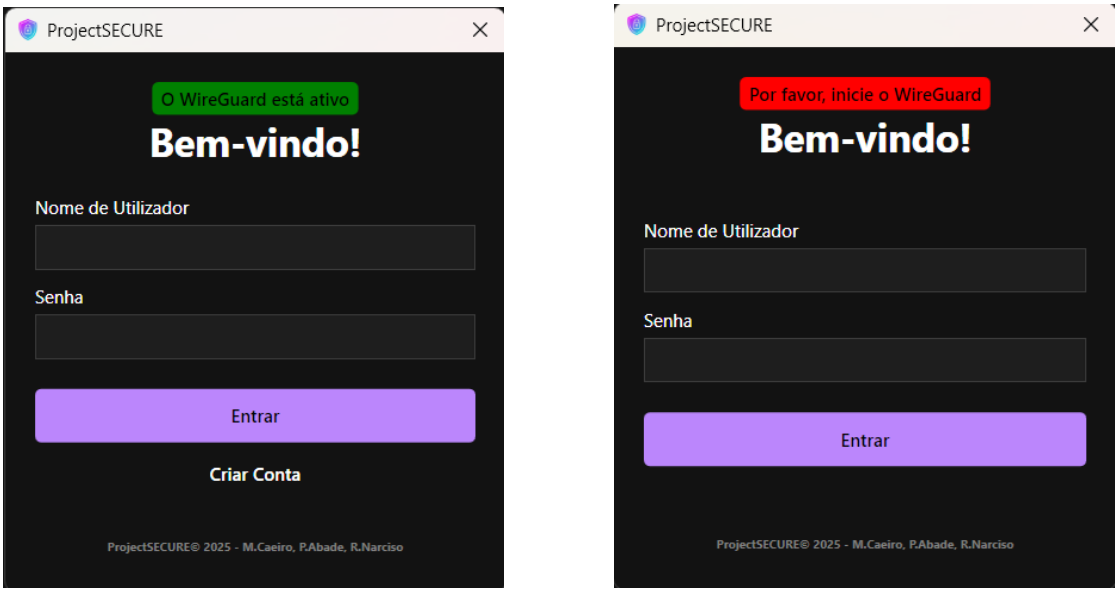


(a) Sem Criptografia (b) Com Criptografia

Figura 9: Teste com Servidor

8.2 Comunicações

Para a monitorização das comunicações, foi utilizado o Wireshark para fazer sniffing dos pacotes que estavam a ser enviados e recebidos pelo servidor. Como é possivel ver nas figuras abaixo, estes são os resultados:



(a) Sem Criptografia (b) Com Criptografia

Figura 10: Teste com Wireshark

9 Conclusão

Concluimos assim que o projeto foi um sucesso, pois conseguimos desenvolver um sistema de comunicações seguras que permite a troca de mensagens de texto entre os seus utilizadores, sem que haja a possibilidade de terceiros acederem às mensagens trocadas. Em geral, foi um projeto que nos permitiu aprender bastante sobre a área de cibersegurança e comunicações seguras, e que nos possibilitou aplicar os conhecimentos adquiridos ao longo do curso de Engenharia Informática.

Bibliografia

AlmaOS. (2025). *AlmaOS: A Secure Operating System* [Página oficial do AlmaOS]. Obtido agosto 16, 2025, de https://almalinux.org

Bash. (2025). *Bash: The GNU Project's Command-Line Interpreter* [Página oficial do Bash]. Obtido agosto 16, 2025, de https://www.gnu.org/software/bash/

Foundation, P. S. (2025). *Python: A Programming Language* [Página oficial do Python]. Obtido agosto 16, 2025, de https://www.python.org/

IPBeja. (2025). *Disciplina: Estágio ou Projeto | IPBeja* [Página EP]. Obtido junho 16, 2025, de https://cms.ipbeja.pt/course/view.php?id=238

JetBrains. (2025). *Kotlin: A Modern Programming Language* [Página oficial do Kotlin]. Obtido agosto 16, 2025, de https://kotlinlang.org/

Microsoft. (2025). *C#: A Modern Programming Language for .NET* [Página oficial do C#]. Obtido agosto 16, 2025, de https://learn.microsoft.com/en-us/dotnet/csharp/

SQLite. (2025). *SQLite: A Self-Contained, High-Performance, Embedded SQL Database Engine* [Página oficial do SQLite]. Obtido agosto 16, 2025, de https://www.sqlite.org/

WireGuard. (2025). *WireGuard: Next Generation Kernel Network Tunnel* [Página oficial do WireGuard]. Obtido agosto 16, 2025, de https://www.wireguard.com/