Desenvolvimento em Ambiente iOS de um Aplicativo para Conexões de Redes Privadas Utilizando Protocolo OpenVPN

Dener Araújo[1], André M. Silva[1]

[1]Centro Universitário Campo Limpo Paulista (UNIFACCAMP)
Jardim América – CEP 13231-230, Campo Limpo Paulista – SP – Brasil

deneraraujo@ibest.com.br, andre@faccamp.br

Abstract. *This article proposes the development of a mobile application prototype, for the iOS platform, to establish VPN connections, based on the OpenVPN protocol, between the user and the private network. The prototype must grant the connection, regardless of the server configuration, and allow the user to make customizations as needed.*

Resumo. *Este artigo propõe o desenvolvimento de um protótipo de aplicativo móvel, para plataforma iOS, com o intuito de estabelecer conexões VPN, baseadas no protocolo OpenVPN, entre o usuário e a rede privada. O protótipo deverá garantir a conexão, independentemente das configurações do servidor, e permitir ao usuário fazer customizações conforme necessário.*

1. Introdução

Devido ao recente crescimento da demanda de trabalho no formato *home office*, tornou-se popular a utilização de conexões VPN (Virtual Private Network) para garantir, de forma segura, a integração de usuários às redes corporativas fechadas. Dentre as conexões desse tipo, uma das mais populares é a baseada no protocolo OpenVPN. Entretanto, nem todas as plataformas possuem pleno suporte a conexão com os servidores que implementam este protocolo. É o caso da plataforma móvel iOS, que só dispõe em seu catálogo de uma única ferramenta para estabelecer tal conexão; e, dependendo do nível de customização das configurações do servidor, a conexão torna-se impossível com essa aplicação. Foi essa deficiência, somada à popularidade da plataforma iOS, que motivou o desenvolvimento de uma ferramenta com opções de customização inexistentes nas poucas soluções existentes hoje no mercado.

2. Objetivos e Metodologia

O objetivo deste trabalho é disponibilizar uma nova ferramenta para os usuários da plataforma iOS, que necessitam estabelecer conexões VPN entre seu aparelho e redes privadas.

Para a criação dessa aplicação, foi utilizada a IDE Xcode, da Apple. O *software* foi construído sobre uma variação da arquitetura MVVM (Model View ViewModel) e depurado em um iPhone 7. Para testes, foi criado um servidor virtual na nuvem (VPS), utilizando o ambiente Google Cloud Platform. O servidor possui a distribuição Debian 10 do sistema operacional Linux; e nele foi instalada e configurada a rede OpenVPN.

Para complementar os testes, foi criado um segundo servidor, conectado à mesma intranet do primeiro, porém sem acesso externo. Esta segunda VPS roda um *web server* que disponibiliza apenas uma página ilustrativa.

3. Desenvolvimento

Para codificação do protótipo, foi utilizada a linguagem de programação Swift (Apple, 2020), em sua versão 5.0. A arquitetura utilizada para organização do código foi a MVVM, com algumas adaptações. E para criação da interface gráfica foi utilizado o *framework* SwiftUI.

Para simular um caso de uso real, foi montada uma rede virtual na plataforma Google Cloud Platform, contendo dois servidores. Sendo um servidor *web* e um servidor OpenVPN. Ambos rodando sistema operacional Debian (Linux) e comunicáveis entre si.

As seguintes ferramentas de apoio foram utilizadas no desenvolvimento do aplicativo e criação do ambiente de testes:

- Xcode: IDE (Integrated Development Environment) para desenvolvimento de aplicações em ambiente Apple.
- OpenVPNAdapter: Framework em Objective-C, baseado na biblioteca OpenVPN3, originalmente escrita em C++, utilizada para configurar uma conexão de protocolo OpenVPN.
- CocoaPods: Gerenciador de dependências para projetos Swift e Objective-C.
- Visual Studio Code: Editor de texto utilizado na edição dos *markdowns*, arquivos *pod* e outros arquivos não gerenciados pelo Xcode.
- Google Cloud Console: Interface para gerenciamento dos serviços da plataforma Google Cloud.
- GitHub: Plataforma para hospedagem de código-fonte, baseada no sistema de controle de versão Git.
- LucidChart: Plataforma utilizada para criação do diagrama de caso de uso UML.

Inicialmente, foi feito um levantamento dos possíveis *frameworks* e bibliotecas disponíveis para trabalhar com conexões OpenVPN no sistema operacional iOS. Após o levantamento, foi concluído que o mais seguro seria utilizar a biblioteca oficial do protocolo, escrita em C++, uma vez que esta possui uma documentação confiável. Porém devido à barreira da linguagem, foi necessária a ajuda do *framework* OpenVPNAdapter, que adapta a biblioteca oficial para a linguagem Objective-C. Tornando possível incorporá-la ao projeto, uma vez que a IDE Xcode possibilita o uso da linguagem Objective-C em conjunto com linguagem Swift para desenvolvimento de projetos iOS.

O passo seguinte foi entender como o sistema operacional lida com interfaces de rede virtuais. Ao contrário de outros sistemas, como o Windows e Android, o iOS não trabalha com interfaces de rede individuais para cada conexão VPN. Ao invés disso, são definidos perfis de configuração para a conexão principal. Sendo assim, o projeto foi

construído para criar um perfil de configuração de rede, com base nas informações fornecidas pela biblioteca.

Uma vez estabelecidos os elementos necessários para desenvolver o cliente, o próximo passo seria criar um ambiente que simulasse uma rede, contendo um servidor OpenVPN. Desta forma, foi montada então uma rede virtual na plataforma Google Cloud Platform, contendo dois servidores. Sendo um deles um servidor *web*, que retorna uma página HTML (HyperText Markup Language) para comprovar o acesso, e um servidor OpenVPN. O primeiro representa um servidor qualquer dentro de uma rede corporativa, que possui as portas fechadas para acesso externo. Já o segundo, é o servidor VPN ao qual o aplicativo se conectará para ter acesso ao primeiro, bem como outros possíveis servidores dentro da rede.

Após estabelecida com sucesso uma conexão entre o cliente e o servidor, foi desenvolvida uma interface gráfica utilizando o *framework* SwuiftUI, capaz de criar um *layout* que se adapta aos variados dispositivos Apple. A interface trabalha da mesma forma que as dos clientes OpenVPN já existentes em outros sistemas operacionais: o aplicativo carrega as configurações pré-definidas em um arquivo de extensão ".opvn" e então disponibiliza opões de customização para o usuário.

A nível de protótipo, o aplicativo está limitado a um único perfil de usuário, e possibilita a customização dos servidores DNS (Domain Name System) utilizados na conexão, bem como diferentes formas de autenticação. Na figura 1 é demonstrada a interação dos atores com o protótipo, através de um diagrama de caso de uso.

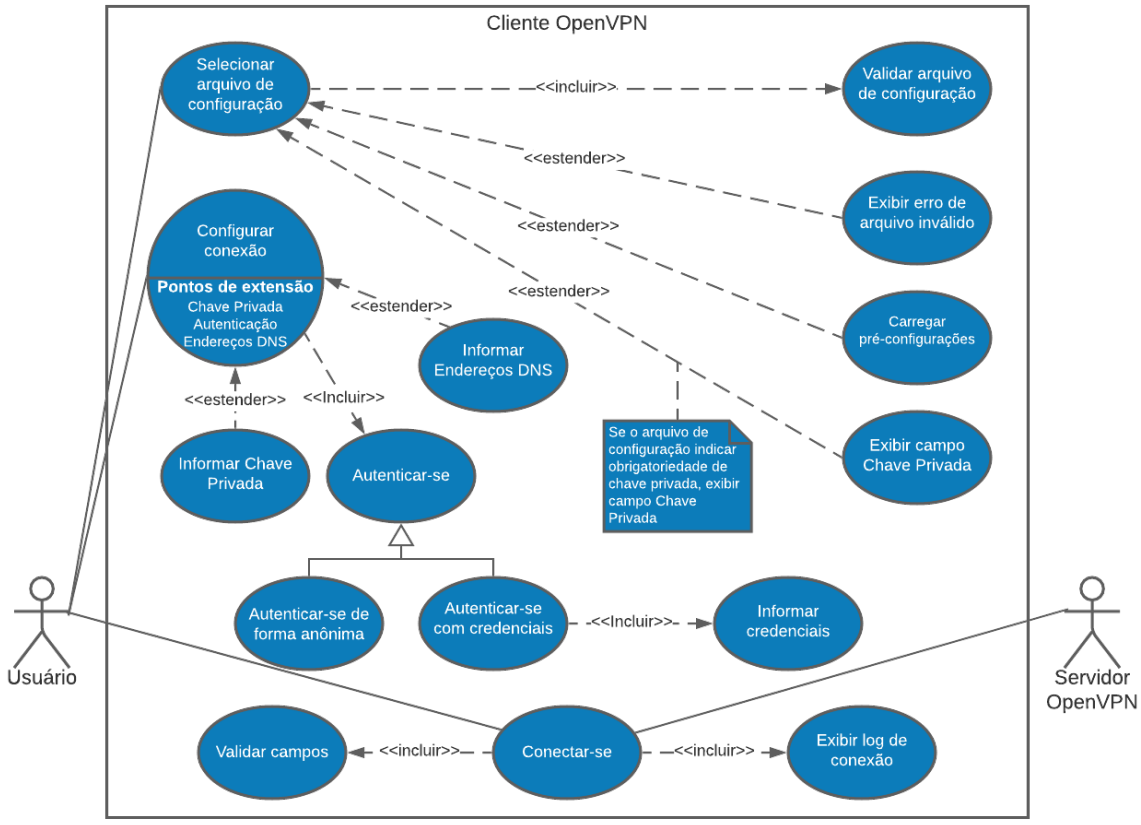

Figura 1. Diagrama de caso de uso

4. Resultados

Ao executar o aplicativo, o usuário deverá, antes de mais nada, carregar o arquivo de configuração referente à rede VPN que ele pretende se conectar. Este arquivo é e gerado no servidor e possui a extensão “.opvn”. Normalmente é fornecido pelo administrador da rede. Para este trabalho, o arquivo foi gerado no servidor de testes da nuvem.

A figura 2 representa todo o processo de conexão e exibição do *log*.

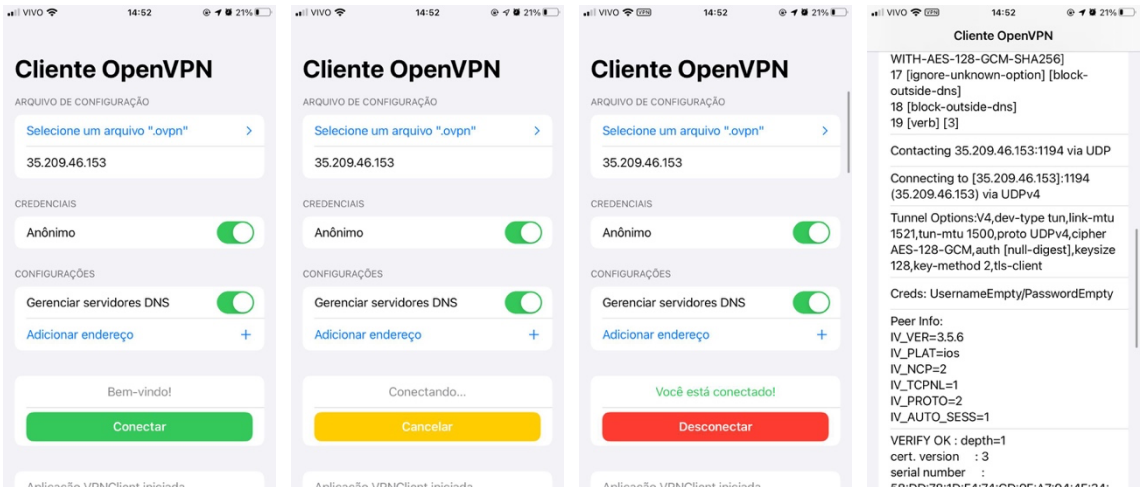

Figura 2. Estados da conexão e log

Uma vez carregado o arquivo, é feita a validação do mesmo. Caso o arquivo esteja íntegro, é exibido o endereço do servidor OpenVPN. Caso não, o usuário é alertado.

O servidor OpenVPN permite que os arquivos de configuração que serão distribuídos para os usuários sejam protegidos com uma chave privada de segurança. Caso o arquivo que o usuário esteja carregando enquadre-se nesse caso, será exibido um campo obrigatório chamado “Chave privada”.

Em seguida, o usuário deve escolher como se autenticará no servidor: de forma anônima ou através de usuário e senha. A opção de autenticação anônima se faz necessária, pois é possível que o administrador da rede opte apenas pela chave privada do arquivo como forma de segurança, e dispense autenticação por usuário e senha.

Com o arquivo carregado e dados de autenticação preenchidos, o próximo passo é o usuário escolher se informará endereços extras de servidores DNS para uso com a conexão. É comum que redes internas possuam servidores DNS dedicados, principalmente em ambiente corporativo. Nesses casos, os eventuais endereços serão fornecidos ao usuário, para que ele preencha no aplicativo.

Com os campos preenchidos, basta que o usuário acione o botão “Conectar”. A obrigatoriedade dos campos é validada e a caso haja alguma irregularidade, o usuário é alertado antes de continuar. É feito então um *ping* para verificar a disponibilidade do servidor. Uma vez acessível, as credenciais são validadas, e caso estejam irregulares, o usuário é alertado e a tentativa de conexão é interrompida. Caso contrário, inicia-se o contrato de conexão.

Todo o processo de negociação do cliente com o servidor é exibido em um *log*, que é preenchido em tempo de execução e exibido na tela. Quaisquer falhas de conexão

serão destacadas em vermelho no *log* e resultarão em uma mensagem de falha para o usuário.

Com a conexão bem-sucedida, o usuário estará apto a acessar os endereços tanto da Internet, quanto da intranet agora disponível, como demonstrado na figura 3.

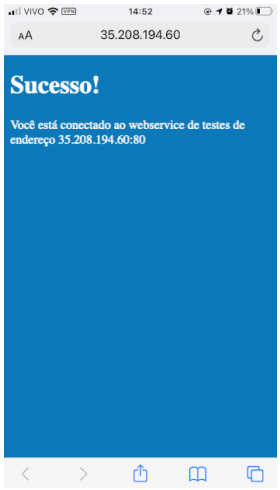

Figura 3. Navegador acessando endereço da rede interna

5. Conclusão

Com os estudos e informações levantadas para o desenvolvimento do projeto, foi possível compreender de forma mais ampla o funcionamento de redes privadas virtuais nos mais variados sistemas operacionais. Conhecimentos adquiridos em curso, como engenharia de *software* e levantamento de requisitos, foram indispensáveis no desenvolvimento desse trabalho, assim como algumas noções de funcionamento de redes.

A aplicação de arquitetura de *software* também foi significativamente aprimorada, bem como o uso de linguagens e tecnologias modernas voltadas para o desenvolvimento de aplicações móveis. Também foram adquiridas melhores noções de controle de versão e gerenciamento de tarefas, uma vez que o GitHub foi fundamental no gerenciamento do desenvolvimento da aplicação.

Além da gama de conhecimentos adquirida, o intuito é que esse projeto sirva de contribuição para a comunidade de desenvolvedores *mobile*, principalmente voltada para o sistema operacional iOS. Uma vez que essa plataforma dispõe de um menor conteúdo, quando comparada a outras.

O conhecimento e resultado obtidos até aqui foram satisfatórios e abriram precedente para o aprimoramento do protótipo, a fim de desenvolver uma aplicação completa e robusta, que em um futuro próximo faça-se útil para usuários reais, que encontram dificuldades ao conectar-se a redes privadas através de seus dispositivos móveis.

Referência Bibliográfica

Apple (2020); The Swift Programming Language. Documentação oficial da linguagem Swift [online] https://swift.org/documentation/#the-swift-programming-language, acessado em dezembro de 2020.

Hoffman, J. (2019); "Mastering Swift 5", 5th Edition, Packt Publishing. 2019.

Apple (2020); Documentação oficial do *framework* SwiftUI [online] https://developer.apple.com/documentation/swiftui/, acessado em dezembro de 2020.

OpenVPNAdapter (2020); Framework em Objective-C, baseado na biblioteca OpenVPN3, para configuração de conexões de protocolo OpenVPN [online] https://github.com/ss-abramchuk/OpenVPNAdapter, acessado em setembro de 2020.

OpenVPN3 (2020); Biblioteca em C++, utilizada para configuração de conexões de protocolo OpenVPN [online] https://github.com/OpenVPN/openvpn3, acessado em setembro de 2020.

Ankit Sinhal (2020); "MVC, MVP and MVVM Design Pattern". Artigo sobre as características e diferenças entre as 3 arquiteturas de *software* mais populares. [online] https://medium.com/@ankit.sinhal/mvc-mvp-and-mvvm-design-pattern-6e169567bbad, acessado em setembro de 2020.

Kouraklis, J. (2016); "MVVM in Delphi", Apress. 2016.

Jan Just Keijser, J. J. (2017); "OpenVPN Cookbook", Second Edition, Packt Publishing. 2017.

Google (2021); Documentação oficial do Google Cloud [online] https://cloud.google.com/docs, acessado em fevereiro de 2021.