UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Monitorización y gestión del tráfico de VPNs mediante un orquestrador de conexiones

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Novella Nebot, Javier

Tutor/a: Tavares de Araujo Cesariny Calafate, Carlos Miguel

CURSO ACADÉMICO: 2022/2023

Resum

Aquest treball de fi de grau s'ha centrat en el desenvolupament d'una plataforma web que realitze les funcions d'un orquestador de VPN. Observant les solucions ja propostes, s'ha elaborat una nova amb l'objectiu de poder oferir funcionalitats tant de creació com d'administració d'aquestes. Per a poder complir amb la seua elaboració, s'han emprat tecnologies com Ruby on Rails, GitHub, entre altres. Els resultats obtinguts mostren una aplicació web amb funcionalitats diferents depenent del tipus d'usuari registrat, entre aquestes es permet la creació de VPNs mitjançant de OpenVPN, així com els clients d'aquestes. Pel costat de l'administració, s'inclou una gestió d'usuaris, dels elements creats i gràfiques a prop del trànsit de xarxa que discórrega per les xarxes creades. Com a conclusions es desitja destacar la importància d'oferir un entorn segur a l'hora de crear connexions entre VPN, permetre una àmplia varietat d'opcions de configuració, a més de mostrar uns gràfics clars i comprensibles per a qualsevol tipus d'usuari.

Paraules clau: VPN, tràfic, monitorització, orquestador

Resumen

Este trabajo de fin de grado se ha centrado en el desarrollo de una plataforma web que realice las funciones de un orquestador de VPN. Observando las soluciones ya propuestas, se ha elaborado una nueva con el objetivo de poder ofrecer funcionalidades tanto de creación como de administración de las mismas. Para poder cumplir con su elaboración, se han empleado tecnologías como Ruby on Rails, GitHub, entre otras. Los resultados obtenidos muestran una aplicación web con funcionalidades distintas dependiendo del tipo de usuario registrado, entre estas se permite la creación de VPNs mediante OpenVPN, así como los clientes de las mismas. Por el lado de la administración, se incluye una gestión de usuarios, de los elementos creados y gráficas a cerca del tráfico de red que discurra por las redes creadas. Como conclusiones se desea destacar la importancia de ofrecer un entorno seguro a la hora de crear conexiones entre VPN, permitir una amplia variedad de opciones de configuración, además de mostrar unos gráficos claros y entendibles para cualquier tipo de usuario.

Palabras clave: VPN, tráfico, monitorización, orquestador

Abstract

This final degree project has focused on the development of a web platform that performs the functions of a VPN orchestrator. Observing the solutions already proposed, a new one has been elaborated with the aim of being able to offer both creation and administration functionalities. In order to accomplish its elaboration, technologies such as Ruby on Rails, GitHub, among others, have been used. The results obtained show a web application with different functionalities depending on the type of registered user, including the creation of OpenVPN VPN, as well as VPN clients. On the administration side, it includes a management of users, of the created elements and graphs about the network traffic flowing through the created networks. As conclusions, we would like to highlight the importance of offering a secure environment when creating connections between VPNs, allowing a wide variety of configuration options, as well as displaying clear and understandable graphics for any type of user.

Key words: VPN, traffic, monitoring, hub

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

En numerosas instituciones se ha fomentado el uso de las redes virtuales privadas o VPN debido al incremento de los puestos de teletrabajo, generados por la necesidad de deslocalizar a los trabajadores de sus puestos de trabajo por la pandemia global. Además, hoy en día se requiere la interconexión de distintos entornos que se encuentran separados geográficamente. Sin embargo, esta interconexión debe producirse mediante un canal seguro ya que las comunicaciones entre éstas no deben ser accesibles para entornos ajenos a la misma corporación. Para ello se genera una VPN, la cual permitirá interconectar ambas sedes pese a estar en entornos geográficamente distantes.

La implantación de un cuantioso número de redes virtuales, ha generado un tráfico díficil de gestionar y monitorizar. Por lo tanto, podemos evidenciar una clara problemática a la hora de controlar todo este tráfico, y para las empresas esto supone un enorme reto.

Numerosas empresas requieren limitar la cantidad de tráfico para no saturar la red, poder limitar el ancho de banda a ciertos usuarios y poder monitorizar el tráfico que existe en la red, entre otros casos. Es decir, para poder ofrecer una correcta gestión de la VPN es necesario tener un control de la cantidad de tráfico que se maneja, limitarlo si es necesario, y además poder ofrecer una gestión sobre los usuarios que se interconectarán a la misma, entre otras muchas utilidades existentes para facilitar esta labor.

Para ello, en este trabajo se desarrollará una herramienta que ofreza la creación y la gestión de dichas VPNs; además, se proporcionará la respectiva monitorización del tráfico de la red mediante gráficas en tiempo real.

1.1 Motivación

Debido a la pandemia, las empresas se han visto en la necesidad de promover una deslocalización física de los empleados y, consecuentemente, se ha normalizado el acceso remoto a las redes corporativas desde las redes públicas. Además, debido a esta obligación, se ha impulsado la interconexión entre distintas sedes geográficas de una misma entidad.

En la actualidad, pese ya a haber podido superar la pandemia, sigue habiendo un gran auge en el uso y comercialización de este tipo de redes, por lo que hay una gran cantidad de tráfico generado por estas conexiones. Dichas conexiones pueden llegar a generar conflictos para dicha entidad, ya que no hay un control sobre el tráfico que generan las mismas, y puede llegar a haber una saturación en la red.

Como se ha mencionado con anterioridad, también es importante controlar las interconexiones con las demás sedes geográficas; por ello es importante ofrecer una interconexión segura y fiable. Para ello se suelen emplear VPNs debido a su flexibilidad y buenos niveles de seguridad, sin embargo, también cabe resaltar que se debe poder gestionarlas y monitorizarlas.

Es por ello que surge la idea de este proyecto: un orquestador de conexiones VPN que monitorice y controle el tráfico generado a través de las VPN.

1.2 Objetivos

El principal objetivo de este trabajo es elaborar un orquestador de VPN que permita monitorizar y gestionar el tráfico de las mismas.

Para ello se ha desglosado el objetivo principal en distintos subobjetivos para facilitar su comprensión:

- Gestionar la creación de VPN, estableciendo los parámetros requeridos, así como otros más específicos.
- Ofrecer herramientas para gestionar el tráfico que transcurra entre las VPN creadas.
- Monitorizar el tráfico de las mismas VPN, mediante herramientas como gráficos en tiempo real.

1.3 Impacto esperado

Este producto va a suponer una mejora a la hora de gestionar y monitorizar las VPN. Las mejoras se pueden analizar desde dos puntos de vista: desde el punto de vista del administrador, o desde el punto de vista del cliente.

En primer lugar, desde la perspectiva del administrador, se podrá observar los clientes conectados a la misma, así como utilizar herramientas para limitar el ancho de banda y el acceso a las mismas VPN. En resumen, se ofrecerán distintas herramientas para el administrador, con el fin de poder gestionar la red para evitar congestión del tráfico, observar qué clientes consumen más recursos mediante gráficas, o limitar la conexión de determinados clientes.

Desde la parte del cliente, se ofrecerá una pantalla en la que se identifique de forma clara y concisa la configuración de las mismas VPN. En ésta se mostrarán todos los aspectos relevantes en la configuración de la VPN; además, el cliente dispondrá de todas las VPN a las cuales tiene permitido conectarse.

1.4 Estructura de la memoria

En el presente trabajo, se realiza el desarrollo de un orquestador mediante una aplicación web que gestione y monitorice las conexiones VPN.

En primer lugar, en el apartado de introducción, se explicará la motivación de la elaboración del trabajo, los objetivos del mismo además del impacto que se espera lograr con él.

En el tercer capítulo, se trata el estado del arte, es decir, se mencionan distintas soluciones similares al trabajo desarrollado, comentando las diferencias y los motivos por los cuales éste es único.

A continuación, el capítulo referido a análisis del problema. En él se estudia el problema a resolver, en este caso la gestión y monitorización de las conexiones VPN. Se realiza un análisis de los requisitos de la misma aplicación, para identificarlos se elaboran distintos diagramas que los muestran explícitamente. Se hace un estudio de las posibles soluciones que existen para desarrollar el trabajo, presentando un abanico de pros y contras; por último, se explica la solución propuesta y los motivos por los cuales se ha escogido.

En el quinto capítulo se presenta el diseño de la solución; en él se muestra la arquitectura del sistema, gracias a los mockups se presentan los prototipos de la aplicación web y la justificación de los mismos. Posteriormente, se explica la estructura de la base de datos. Por úlltimo se explicará de forma individual las tecnologías que han sido escogidas para desarrollar esta aplicación.

Seguidamente, en el capítulo referido al desarrollo de la solución propuesta se comenta la creación del orquestador web y las distintas herramientas de monitorización que ofrece.

A lo largo del sexto capítulo se describe detalladamente como se ha podido desplegar la aplicación web de manera que sea accesible para todos los usuarios.

En el séptimo capítulo, se explican las pruebas que se han llevado a cabo para verificar el correcto funcionamiento de la aplicación.

Por último se muestran los capítulos referentes a las conclusiones obtenidas tras haber finalizado el trabajo, además de una sección donde se mencionan diferentes propuestas que podrían incluirse en el trabajo para ofrecer una herramienta más completa.

CAPÍTULO 2
Estado del arte

El mercado global de las VPN en 2019 se estima que llegó a estar valuado en torno a los 25,65 mil millones de dólares, y se ha realizado un estudio en el cual se espera que la tasa de crecimiento anual compuesto (CAGR) se encuentre en torno al 17.4 % entre 2020 y 2027.[2]

Este crecimiento se puede verificar ya que en el año 2022,donde el mercado de las VPN facturó en torno a 44.6 mil millones de dólares [3] en Estados Unidos, tal y como ilustra la figura 2.1.

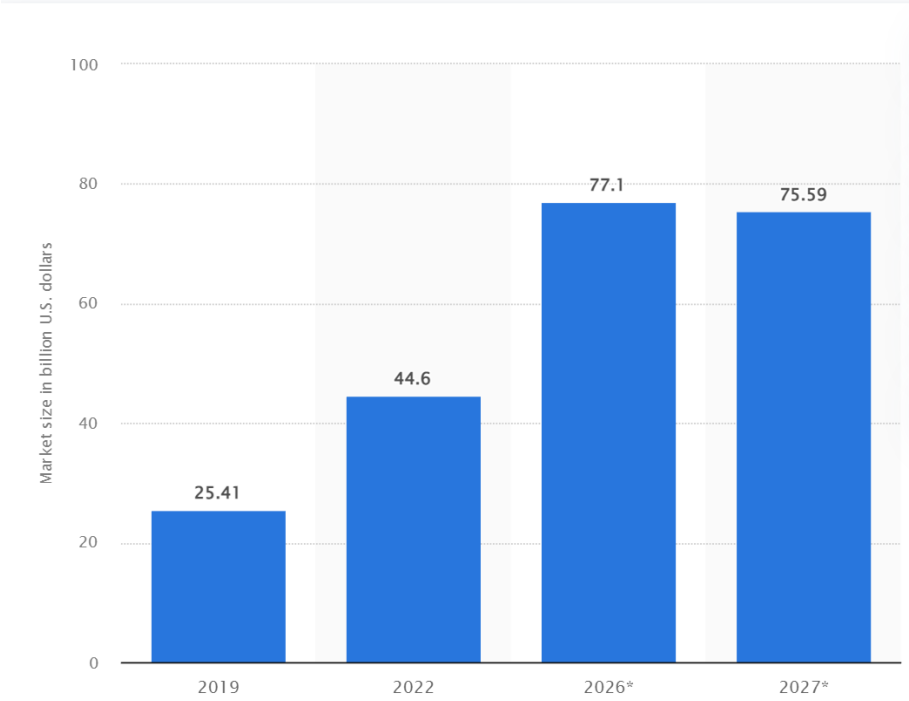


Figura 2.1: Mercado de las VPN entre 2019-2027. Fuente: (Statista, 2023)

Al ser un sector tan en auge, se han realizado muchos esfuerzos por elaborar herramientas que puedan gestionar las VPN.

2.1 Definición de una VPN

Sus siglas se refieren a una red virtual privada [4]. Emula una red pública, pero permite proteger la dirección IP del usuario, además de cifrar el tráfico de Internet haciendo que así sea más seguro.

Es decir, permite recrear una red local a través de Internet evitando que sus miembros se encuentren físicamente conectados entre sí. Los principales usos que se encuentran sobre las VPN enfocados a este proyecto podrían concentrarse en los siguientes puntos:

1. Teletrabajo: Permite la interconectividad entre redes. De esta manera los trabajadores en remoto pueden acceder al entorno privado de la empresa.

   La conexión estará previamente cifrada, y el trabajador tendrá las mismas facilidades a nivel de red, tal y como si estuviera trabajando de manera presencial.

2. Seguridad: Las VPN ofrecen una capa extra de seguridad en la mayoría de los casos, cifrando los paquetes que son transmitidas con ellas. Sin embargo, el servidor de VPN puede, como una red pública, capturar el tráfico u emplear distintas utilidades.

2.2 Crítica al estado del arte

Se han desarrollado diferentes propuestas con una funcionalidad similar. Como por ejemplo: SoftEtherVPN, LibreSwan o Strongswan. A continuación se analiza brevemente las caracterísitcas de cada una:

1. SoftEtherVPN [5]

   Es una herramienta open source [6], la cual contiene una función clonada de OpenVPN Server. Permite establecer VPN tanto remote-access como site-to-site. Además, contiene una función para la creación de VPN mediante L2TP/IPsec server.

   Permite configurar la encriptación de la red, el estado del servidor, y la lista de conexiones TCP/IP (entre otras cosas). Además, ofrece una alta resistencia frente a los firewalls. Dicha herramienta es compatible con Windows, Mac, IOS y Android.

2. LibreSwan [7]

   Libreswan es una implementación de IPSec [8] open source. Se puede instalar en hosts de red local o en una red de proveedor en la nube.

   Soporta una gran cantidad de tipos de algoritmos de cifrado

3. Strongswan [9]

   StrongSwan implementa el protocolo IKEv2 y también IKEv1. Proporciona soporte en IPv6 y los túneles IPSec.

   Emplea mecanismos de autenticación utilizando certificados de clave pública X.509[10].

Estas herramientas mencionadas se centran en la creación de VPN, algunas de ellos implementan protocolos propios como SoftEtherVPN, que usa SSL-VPN [11]. No obstante, no se proporciona una monitorización de las redes creadas.

Desde el punto de vista de la gestión, se ofrece en un nivel muy bajo. Es decir, si se quiere ofrecer un control sobre los clientes conectados, o limitar el ancho de banda para una determinada conexión, estas herramientas mencionadas no permiten las configuraciones mencionadas.

2.3 Propuesta

Este proyecto consiste en una herramienta que permita la creación de VPN, así como la gestión y monitorización de las mismas. Ofreciendo una herramienta que permita a los administradores tener un control completo de las redes creadas. Cubriendo las necesidades destacadas con anterioridad.

En la tabla a continuación se ofrece una comparativa de las herramientas descritas previamente y la que se desarrolla en este documento.

Tabla 2.1: Comparación de las herramientas existentes y el proyecto diseñado

	SoftEtherVPN	LibreSwan	StrongSwan	OrchestratorVPN
Open source	Sí	Sí	Sí	Sí
Uso de OpenVPN	Clonación de OpenVPN Server	No	No	Sí
Tipos de VPN creadas	Site-to-Site y Remote-Access	Site-to-Site	Site-to-Site y Remote-Access	Site-to-Site y Remote-Access
Protocolos empleados	VPN	IPsec, IKE	IPsec	TCP, UDP
Seguridad	Firewall y túnel VPN	PSK, RSA y certificados	Firewall y métodos como LibreSwan	Túnel VPN, RSA, certificados
Encriptación	AES, RSA	IKE	IKE	AES, RSA
Interoperabilidad	Windows, Linux, OS FreeBSD y Solaris	Windows	Windows, iOS, Linux	Por el momento Linux
Gestión de clientes	No	No	No	Sí
Herramientas de monitorización	No	No	No	Sí

CAPÍTULO 3
Análisis del problema

En este capítulo se realiza un análisis del problema el cual pretende resolver este proyecto. En primer lugar, se analizan los requisitos que debe cumplir este proyecto para así poder cumplir los objetivos que han sido propuestos. A continuación, se mostrarán las posibles soluciones que logran satisfacer estos requisitos.

3.1 Análisis de los requisitos

En esta sección se van a extraer los requisitos de la aplicación desarrollada. Para ello se van a realizar los diagramas de casos de uso para los actores que interactuarán con la aplicación, además del diagrama de clases, y más adelante se desarrollará en detalle cada caso de uso.

Para representar los diagramas se utilizará el estándar UML (*Unified Modeling Language*)[12]. Éste es un lenguaje que fue diseñado para crear un modelado visual común, además de incorporar un significado para la arquitectura, diseño e implementación de diferentes sistemas de software.

3.1.1. Diagrama de contexto

El diagrama de contexto es una herramienta que se utiliza para establecer los límites del sistema a desarrollar. Tras haber definido el sistema, se debe mostrar como interactúan los actores con él; cabe destacar que los actores pueden ser tanto seres humanos como cualquier dispositivo o módulo fuera del sistema que tenga cierta interacción con el mismo. En la siguiente figura se muestran los actores externos al sistema que han sido identificados.

Como se puede ver en la figura 3.1, se encuentran distintos actores, los cuales van a ser explicados a continuación:

1. Usuario no autenticado: dicho actor tendrá limitaciones a la hora de acceder al sistema.

2. Usuario autenticado: podrá conectarse a las VPN que tenga disponibles y crear unas nuevas, entre otras funciones.

3. Usuario administrador de VPN: poseerá las funcionalidades descritas para el usuario autenticado, además de poder gestionar y monitorizar el tráfico para las VPN de las cuales es administrador.

4. Usuario administrador: tendrá todas las funciones del usuario administrador de VPN, además de determinadas funciones para gestionar y monitorizar el tráfico.

5. Servidor OpenVPN: se comunica con la aplicación para la creación y eliminación de VPN, además de determinadas funciones para la gestión de las mismas. Se encarga de responder las peticiones que solicita el usuario a través de la aplicación.

6. Monitorizador de tráfico: dicha herramienta generará las gráficas que muestren los gráficos referentes al tráfico de red en tiempo real.

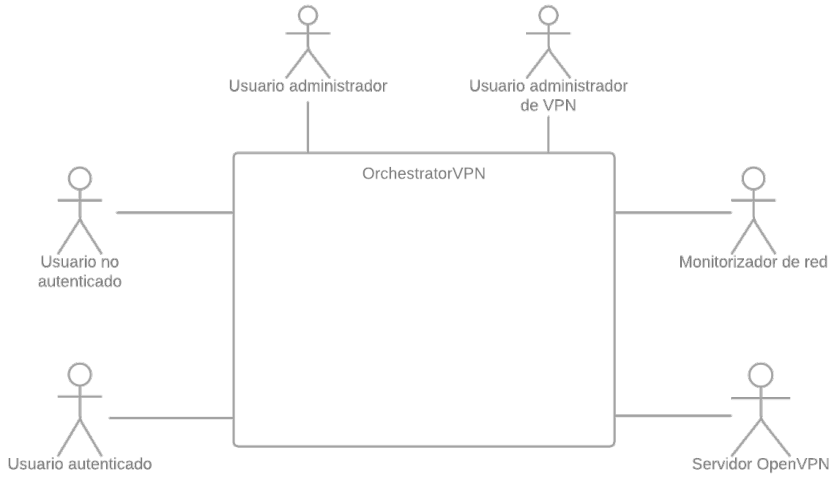


Figura 3.1: Diagrama de contexto.

3.1.2. Diagrama de casos de uso

Tras haber detectado el entorno del sistema y los actores externos, se deben determinar los casos de uso para cada actor. Los casos de uso pueden definirse como la descripción de una actividad para llevar a cabo un proceso.

En los diagramas siguientes se muestran los posibles casos de uso que puede tener cada actor determinado, y su interacción con ellos, mediante las lineas de comunicación.

En primer lugar se representa el usuario que no ha sido autenticado. Dicho usuario estará limitado a registrarse o a identificarse en la aplicación. Esto se ha determinado así ya que las funcionalidades han sido diseñadas teniendo en cuenta que el usuario haya sido autenticado con anterioridad. Por ende, ha sido descartado el uso del sistema sin que exista una autenticación previa por parte del usuario.

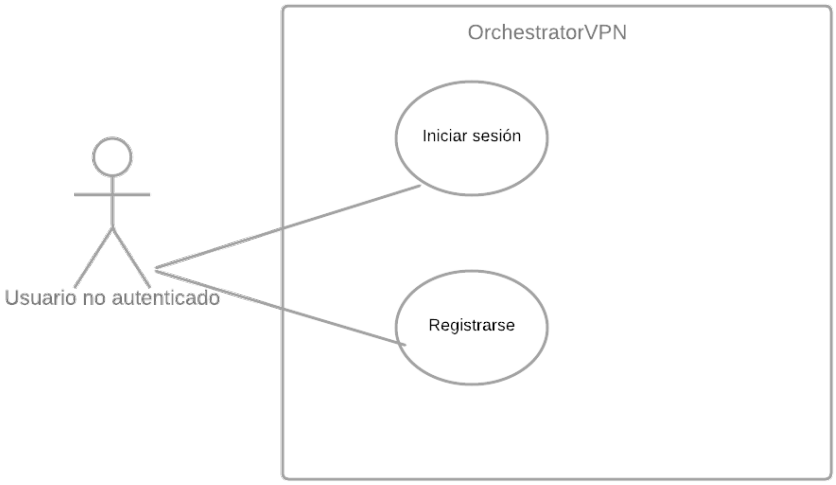


Figura 3.2: Diagrama de casos de uso para el usuario no autenticado.

A continuación, se va a representar el diagrama de casos de uso para el usuario que ha sido autenticado. Dicho usuario ya puede acceder a ciertas funcionalidades de la aplicación.

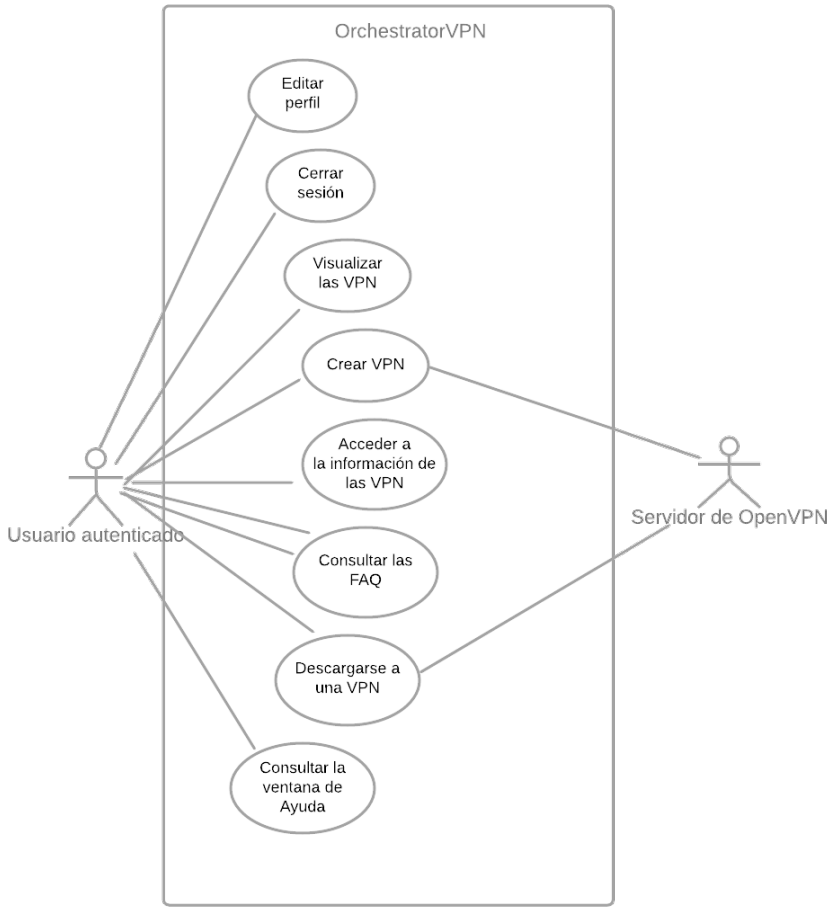


Figura 3.3: Diagrama de casos de uso para el usuario autenticado.

En el gráfico mostrado anteriormente, se puede observar como el servidor de OpenVPN actúa también en los casos de uso que consisten en la creación y conexión de VPN.

Tras mostrar el diagrama para el usuario autenticado, se va a representar el mismo para el administrador de VPN, el cual poseerá las mismas funcionalidades que el usuario anterior, además de otras que se verán a continuación.

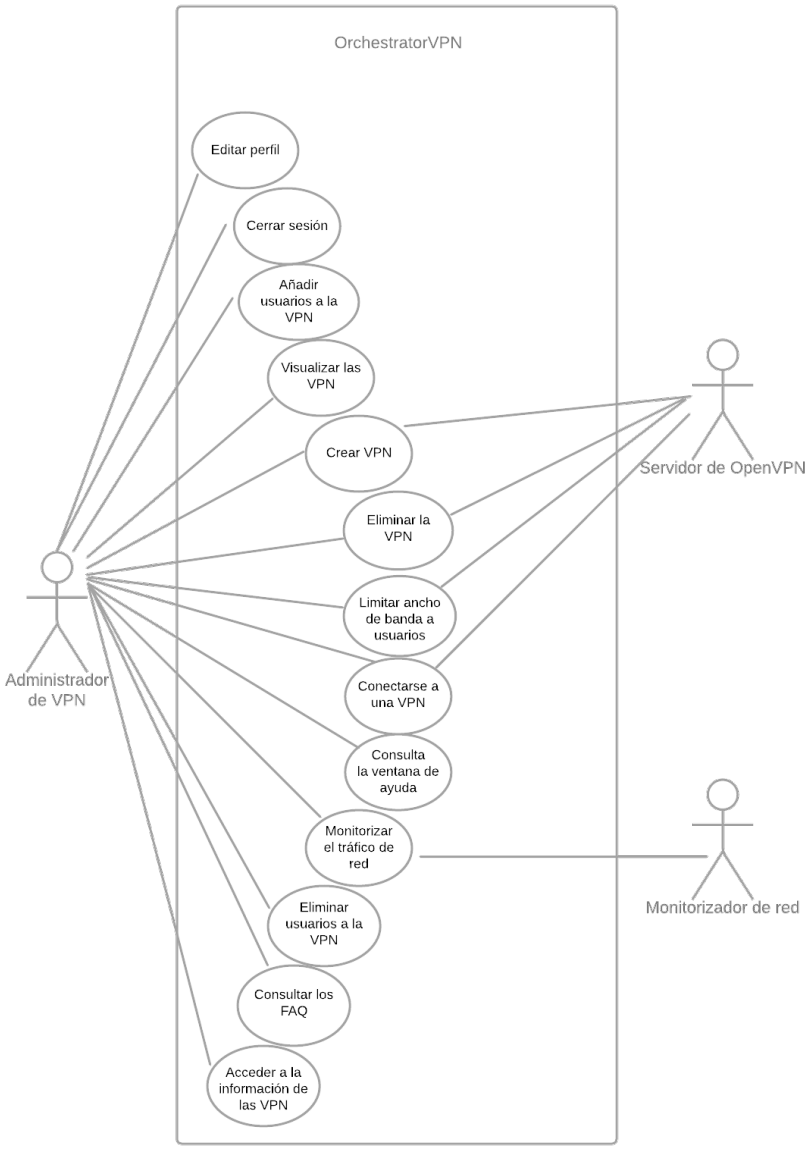


Figura 3.4: Diagrama de casos de uso para el administrador de VPN.

En el caso del administrador de VPN, se añaden ciertos casos de uso en el cual interviene el servidor de OpenVPN, mantiene los mencionados anteriormente y añade los casos de uso siguientes: limitar ancho de banda a usuarios, y eliminiar la VPN. Cabe destacar que se incluye el actor referente al monitorizador de red, y este interactúa en el caso de uso en el que se monitoriza el tráfico de red.

Por último, se va a representar el diagrama de casos de uso para el administrador. Este usuario dispondrá de todas las funcionalidades que ofrece la aplicación desarrollada.

Figura 3.5: Diagrama de casos de uso para el administrador.

En este último diagrama, se mantienen las relaciones presentadas anteriormente y se añade un nuevo caso de uso, el cual afecta al servidor de OpenVPN; este se refiere a la creación o eliminación del servidor.

3.1.3. Casos de uso

Tras haber generado los diagramas de casos de uso, se procede a definir los diferentes casos de una forma más detallada.

Tabla 3.1: Caso de uso 1. Registrarse

Referencia	CU.01
Nombre	Registrarse
Descripción	El usuario debe introducir los datos necesarios para darse de alta. Tras haberlo hecho, se redirige a la pantalla principal de la aplicación
Actor	Usuario no autenticado
Precondición	El usuario no ha iniciado sesión con anterioridad y accede a la misma con unas credenciales que no han sido utilizadas por otra cuenta.
Postcondición	Se completa el registro del usuario en el sistema.

Tabla 3.2: Caso de uso 2. Iniciar sesión

Referencia	CU.02
Nombre	Iniciar sesión
Descripción	El usuario debe introducir los datos con los que se ha registrado. Una vez hecho se inicia sesión y se redirige a la pantalla principal.
Actor	Usuario no autenticado
Precondición	El usuario no ha iniciado sesión con anterioridad y accede a la misma con unas credenciales que han sido registradas en el sistema.
Postcondición	

Tabla 3.3: Caso de uso 3. Editar perfil

Referencia	CU.03
Nombre	Editar perfil
Descripción	El usuario puede cambiar ciertos datos introducidos durante su registro.
Actor	Usuario autenticado
Precondición	El usuario ha iniciado sesión previamente.
Postcondición	Se almacenan los nuevos datos en el sistema

Tabla 3.4: Caso de uso 4. Cerrar sesión

Referencia	CU.04
Nombre	Cerrar sesión
Descripción	El usuario cierra su sesión actual y se le redirige a la pantalla de inicio de sesión.
Actor	Usuario autenticado
Precondición	El usuario ha iniciado sesión previamente.
Postcondición	

Tabla 3.5: Caso de uso 5. Visualizar las VPN disponibles.

Referencia	CU.05
Nombre	Visualizar las VPN disponibles
Descripción	El usuario puede observar las VPN a las cuales tiene acceso.
Actor	Usuario autenticado, administrador de VPN y administrador
Precondición	El usuario ha iniciado sesión previamente.
Postcondición	

Tabla 3.6: Caso de uso 6. Creación de una VPN.

Referencia	CU.06
Nombre	Crear una VPN
Descripción	El usuario solicita la creación de una VPN, para ello ha de introducir los datos requeridos para la generación de la misma.
Actor	Usuario autenticado, administrador de VPN, administrador y servidor de OpenVPN
Precondición	El usuario ha iniciado sesión previamente.
Postcondición	Se introduce en el sistema los datos de la VPN y se crea.

Tabla 3.7: Caso de uso 7. Creación de una VPN.

Referencia	CU.07
Nombre	Acceder a la información de la VPN
Descripción	El usuario demanda acceder a la información que hay registrada en el sistema de la VPN solicitada.
Actor	Usuario autenticado, administrador de VPN, administrador.
Precondición	El usuario ha iniciado sesión previamente.
Postcondición	

Tabla 3.8: Caso de uso 8. Consultar los FAQ.

Referencia	CU.08
Nombre	Consultar las FAQ
Descripción	El usuario accede a las preguntas más recurrentes con respecto al uso de la aplicación.
Actor	Usuario autenticado, administrador de VPN, administrador.
Precondición	El usuario ha iniciado sesión previamente.
Postcondición	

Tabla 3.9: Caso de uso 9. Conectarse a una VPN.

Referencia	CU.09
Nombre	Descargarse el archivo cliente
Descripción	El usuario se descarga el archivo cliente que posibilita la conexión a la VPN mediante OpenVPN.
Actor	Usuario autenticado, administrador de VPN, administrador.
Precondición	El usuario ha iniciado sesión previamente.
Postcondición	El usuario se conecta a la VPN y se muestra un pop-up donde se verifica si la conexión ha sido aceptada o no.

Tabla 3.10: Caso de uso 10. Consultar la ventana de ayuda.

Referencia	CU.10
Nombre	Consultar la ventana de ayuda
Descripción	El usuario accede a la ventana de ayuda donde se muestra información para ayudar al usuario en el uso de la aplicación.
Actor	Usuario autenticado, administrador de VPN, administrador.
Precondición	El usuario ha iniciado sesión previamente.
Postcondición	

Tabla 3.11: Caso de uso 11. Eliminar una VPN.

Referencia	CU.11
Nombre	Eliminar una VPN
Descripción	El usuario desea eliminar una VPN de la cual es administrador.
Actor	Administrador de VPN, administrador.
Precondición	El usuario ha iniciado sesión previamente y selecciona la VPN a eliminar.
Postcondición	Se elimina la VPN seleccionada y se muestra un pop-up donde se verifica la eliminación de la VPN.

Tabla 3.12: Caso de uso 12. Monitorizar el tráfico de red.

Referencia	CU.12
Nombre	Monitorizar el tráfico de red
Descripción	El usuario solicita monitorizar el tráfico que transcurre por la VPN que ha sido escogida.
Actor	Administrador de VPN, administrador.
Precondición	El usuario ha iniciado sesión previamente, selecciona la VPN y selecciona el usuario a limitar.
Postcondición	Se observan gráficos que muestra el tráfico de red que transcurre por la VPN observada

Tabla 3.13: Caso de uso 13. Añadir usuarios a la VPN.

Referencia	CU.13
Nombre	Añadir usuarios a la VPN
Descripción	El administrador puede gestionar a los usuarios que pueden acceder a la VPN .
Actor	Administrador de VPN, administrador.
Precondición	El usuario ha iniciado sesión previamente y selecciona la VPN la cual debe gestionar, posteriormente selecciona los usuarios a añadir de la misma.
Postcondición	Se guardan en el sistema los usuarios que han sido incluidos en la VPN.

Tabla 3.14: Caso de uso 14. Eliminar usuarios a la VPN.

Referencia	CU.14
Nombre	Eliminar usuarios a la VPN
Descripción	El administrador puede gestionar a los usuarios que pueden acceder a la VPN .
Actor	Administrador de VPN, administrador.
Precondición	El usuario ha iniciado sesión previamente y selecciona la VPN la cual debe gestionar, posteriormente selecciona los usuarios a añadir de la misma.
Postcondición	Se guardan en el sistema los usuarios que han sido eliminados de la VPN.

Tabla 3.15: Caso de uso 15. Limitar ancho de banda a usuarios.

Referencia	CU.15
Nombre	Limitar ancho de banda a determinados usuarios
Descripción	El administrador puede gestionar el ancho de banda que tendrá un determinado usuario para una VPN seleccionada .
Actor	Administrador de VPN, administrador.
Precondición	El usuario ha iniciado sesión previamente y selecciona la VPN la cual debe gestionar, posteriormente selecciona el usuario al cual debe limitar el ancho de banda.
Postcondición	Se guardan en el sistema la cantidad de ancho de banda que posee el usuario determinado y solo podrá tener éste en la VPN que haya sido seleccionada.

Tabla 3.16: Caso de uso 16. Crear servidor.

Referencia	CU.16
Nombre	Crear servidor
Descripción	El administrador puede crear un nuevo servidor para que interaccione con la aplicación.
Actor	Administrador.
Precondición	El usuario ha iniciado sesión previamente e introduce la información necesaria para cumplimentar el registro del nuevo servidor.
Postcondición	Se guardan en el sistema la configuración del servidor y se instala el nuevo servidor.

Tabla 3.17: Caso de uso 17. Eliminar servidor.

Referencia	CU.17
Nombre	Eliminar servidor
Descripción	El administrador puede eliminar un servidor que haya sido creado con anterioridad.
Actor	Administrador.
Precondición	El usuario ha iniciado sesión previamente y selecciona el servidor que desea eliminar.
Postcondición	Se elimina del sistema el servidor.

Tabla 3.18: Caso de uso 18. Crear cliente.

Referencia	CU.18
Nombre	Crear cliente
Descripción	El administrador de una VPN o el mismo administrador, pueden crear clientes que se conecten a la VPN deseada.
Actor	Administrador y administrador de VPN.
Precondición	El usuario ha iniciado sesión previamente , selecciona la VPN sobre la cual crear el cliente y rellena la información necesaria para el cliente.
Postcondición	Se crea el cliente en el sistema.

Tabla 3.19: Caso de uso 19. Eliminar cliente.

Referencia	CU.19
Nombre	Eliminar cliente
Descripción	El administrador de una VPN o el mismo administrador, pueden eliminar clientes.
Actor	Administrador y administrador de VPN.
Precondición	El usuario ha iniciado sesión previamente , selecciona el cliente que se quiere eliminar.
Postcondición	Se elimina el cliente del sistema.

3.1.4. Diagrama de clases

El diagrama de clases define la estructura del sistema incluyendo las clases, sus atributos y las relaciones entre estas mismas. Para representar este diagrama se utiliza la notación UML.

- **User:** esta clase representa a todos los usuarios del sistema y cada instancia de la misma contiene un conjunto de atributos los cuales son: correo electrónico, contraseña, administrador, administrador de VPN , la lista de VPN y la lista de VPN de las que es administrador. En la clase usuario se encuentra una relación con la tabla users-vpn; esta relación, así como la tabla se describirá a continuación. Esta relación será del tipo "0..*". Es decir muchos usuarios pueden existir en la tabla mencionada, por lo que muchos usuarios podrán tener muchas redes virtuales privadas.

- **user-vpn:** esta clase se ha realizado para facilitar la referencia entre las tablas *User* y *VPN*. Esta tabla contendrá tres atributos: *server-id* , *user-id*, *admin-vpn*. Se indexará por los identificadores tanto del servidor como del usuario, esto quiere decir que se establecerá una instancia dentro de esta tabla; de tal forma que cada usuario tendrá una relación con cada VPN a la cual puede acceder. Además sobre cada una de estas relaciones, existirá un atributo que identifique si ese usuario será administrador de esa VPN. Como se ha comentado anteriormente, esta tabla actuará como tabla auxiliar para poder completar las referencias y poder establecer correctamente los atributos sobre la misma.

- **VPN:** en esta se describen las VPN creadas. Cada uno contendrá atributos como el nombre, la descripción, ciertas opciones como los ajustes para la creación de la misma, el certificado, el servidor en el que se hospeda,el CIDR, el identificador del servidor en el cual se hospedará dicha VPN, entre otras cosas. Contiene una relación con la clase Servidor, y se podría definir como que varias VPN se hospedan en un único servidor, por lo tanto esta relación es del tipo "*..1".

- **Client:** define el cliente creado para que pueda conectarse a la VPN creada. En esta clase se definen atributos como la VPN, el nombre, la descripción de la misma, el certificado de la misma, distintas opciones y el identificador de la vpn a la cual se conecta. También existe una relación desde esta clase hacia la clase OpenVPN, la cual establece que cada cliente puede tener una cantidad indefinida de instancias de VPN. Ésta es del tipo "*..1", la cual establece que una VPN puede tener muchos clientes, pero un cliente únicamente puede tener una VPN.

- **Server:** en esta clase se detalla el servidor que interactúa con la aplicación; cuyos atributos serán el nombre, la clave de la autoridad certificadora y el certificado de la CA[13]. El servidor será el encargado de establecer las configuraciones previas

en el sistema, poseyendo las credenciales necesarias para la creación y gestión de VPNs. Y, como se ha mencionado anteriormente, tendrá una relación con la clase VPN, mediante la cual un servidor puede tener muchas VPN hospedadas dentro de él.

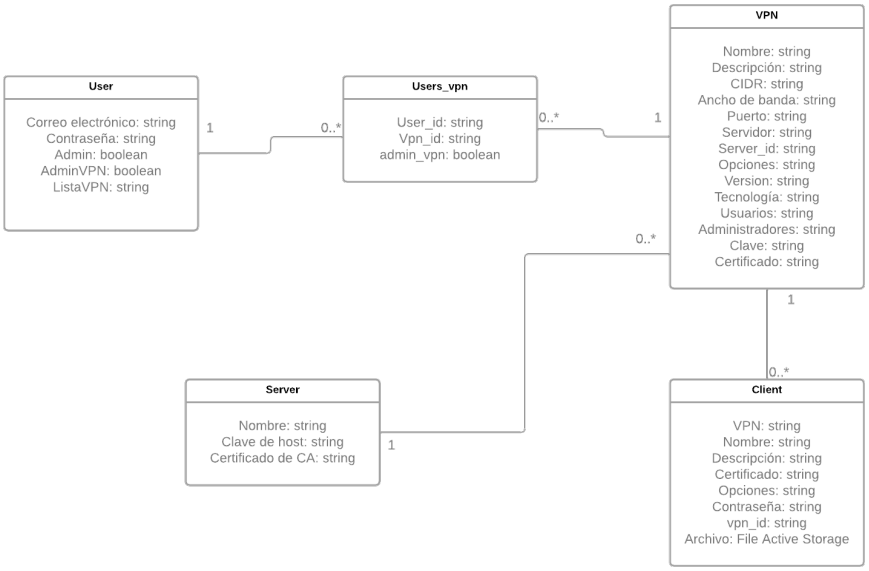


Figura 3.6: Diagrama de clases.

3.2 Identificación de las soluciones posibles

Tras haber realizado un análisis sobre los requisitos que demanda la aplicación, teniendo en cuenta los objetivos propuestos, se procede a discutir las posibles soluciones que se pueden desarrollar para cumplir con los requisitos especificados y ofrecer un producto atractivo para el público de este mismo proyecto.

En primer lugar, se va a ofrecer una ventana de inicio de sesión fácil e intuitiva. En ella se incluirán enlaces para redirigir al usuario a la ventana de registro para poder ofrecer una solución a los casos de uso pertinentes al registro(CU.01) e inicio de sesión(CU.02). En caso de que el usuario no recuerde su contraseña, se le ofrecerá la posibilidad de generar una nueva contraseña.

En todo momento el usuario autenticado tendrá disponible un botón con el cual podrá cerrar sesión(CU.04) y otro con el cual podrá modificar los datos con los cuales ha sido registrado(CU.03).

A continuación, se presentará en una ventana los clientes que tiene disponibles el usuario para poder descargarse y, mediante este archivo, establecer la conexión (CU.05), desplegadas en forma de lista. En primera instancia se mostrará el nombre identificador del cliente y estos se mostrarán de manera a que se ofrezca una paginación de las mismas. Es decir, existirá una posibilidad de navegar entre todas las VPN disponibles para poder seleccionar la que se desea. En caso de ser un usuario autenticado corriente, junto al nombre de la VPN, se habilitará un botón que permita al usuario descargarse el archivo para pdoer conectarse a la misma (CU.09), además se mostrará la información más básica del cliente en esta vista.

Si este usuario, es un administrador de dicha VPN, también se mostrará un botón que redirija al usuario a una ventana donde se visualizarán las gráficas que muestren en

tiempo real el tráfico de red existente para dicha VPN seleccionada (CU.12). Cabe destacar que el administrador de la VPN podrá eliminar la red de la cual es administrador, lo cual eliminará todos los clientes de la misma, y el administrador general podrá eliminar cualquier VPN, además de sus respectivos clientes (CU.14). Por otra parte, el administrador de la VPN podrá limitar el ancho de banda a determinados usuarios, y de esta manera se puede evitar la congestión del tráfico de red (CU.15). Y, siguiendo el caso de que el usuario sea administrador de la red, podrá eliminar o añadir clientes de una VPN, teniendo un botón el cual facilite su administración. (CU.18 y CU.19)

En caso de ser un administrador de una determinada VPN o el administrador, existirá una ventana que facilite la gestión de usuarios, permitiendo añadir o eliminar usuarios a una VPN (CU.13 y CU.14), se puede modificar la propiedad de administrador sobre un usuario, entre otras funcionalidades.

En la ventana se dispondrá de un apartado en el cual el usuario podrá encontrar información para poder emplear las funcionalidades del proyecto al completo (CU.10). Por otra parte, también existirá un apartado de preguntas frecuentes para poder resolver dudas comunes a los usuarios (CU.08).

Existirá otra ventana donde cualquier usuario podrá crear VPNs tras cumplimentar la información necesaria (CU.06); esta información se trata del nombre, contraseña, usuarios que pueden acceder a la misma, una lista de administradores, además de otros datos necesarios. En esta ventana se mostrarán todas las VPN y los administradores podrán gestionarla de la manera que consideren.

Por último, el administrador general podrá crear servidores donde se almacenarán las VPN creadas (CU.16); además podrá eliminarlas y editarlos como considere (CU.17).

CAPÍTULO 4
Diseño de la solución

A lo largo de este capítulo se va a detallar el proceso de desarrollo que se ha llevado a cabo para poder realizar el proyecto.

4.1 Arquitectura del Sistema

La estructura empleada para el desarrollo de este proyecto será una arquitectura cliente/servidor[14]. En esta arquitectura, los clientes realizan peticiones al servidor y este último las contesta. En este caso el sistema actuará de la siguiente forma:

- Cliente: los usuarios serán los clientes, los cuales solicitarán al servidor realizar las operaciones requeridas, que han sido definidas en los casos de uso.
- Servidor: se encargará de realizar las peticiones solicitadas por los clientes y proporcionar las soluciones necesarias.

En esta arquitectura se puede remarcar la separación entre el cliente y el servidor. El cliente visualizará únicamente los datos ofrecidos por el servidor, y será el encargado de solicitar los procesos a ejecutar en el servidor. Por otra parte, en el servidor se concentran las gestiones y el tráfico de datos más importantes. De esta manera se pueden evitar ciertos riesgos relacionados con la vulnerabilidad de los mismos, ya que éste solo mostrará los datos que desee al usuario.

Los clientes del sistema serán los usuarios del mismo. Éstos serán almacenados en la base de datos y, como se ha mencionado anteriormente, tendrán la posibilidad de realizar ciertas acciones dependiendo del rol que tengan. Los clientes accederán a la plataforma web y podrán solicitar al servidor que ejecute las operaciones que requiera.

El servidor, en este caso, será la plataforma web desarrollada, la cual se utilizará para solicitar al servidor donde se hospeda la plataforma, la creación y eliminación de las mismas VPNs, al igual que la creación y destrucción de los clientes, por otra parte también manejará la monitorización de las redes y, por último, manejará la gestión de todos los usuarios.

Por otro lado, los datos serán almacenados en la base de datos para tener un acceso más rápido y eficiente. Sin embargo, se limitarán las acciones que puedan realizar los usuarios para tener un control sobre los mismos, y también para poder evitar ciertos riesgos y vulnerabilidades que puedan ocurrir.

Cabe destacar que, pese a que en la posterior figura 4.1 se muestra al servidor como una computadora ajena a los clientes. Se pueden tener en una misma máquina tanto

cliente como servidor. Aunque en este caso, se ha diseñado principlamente para que los clientes y servidores se encuentren en máquinas distintas, pese a que funcionaría correctamente si no fuese así.

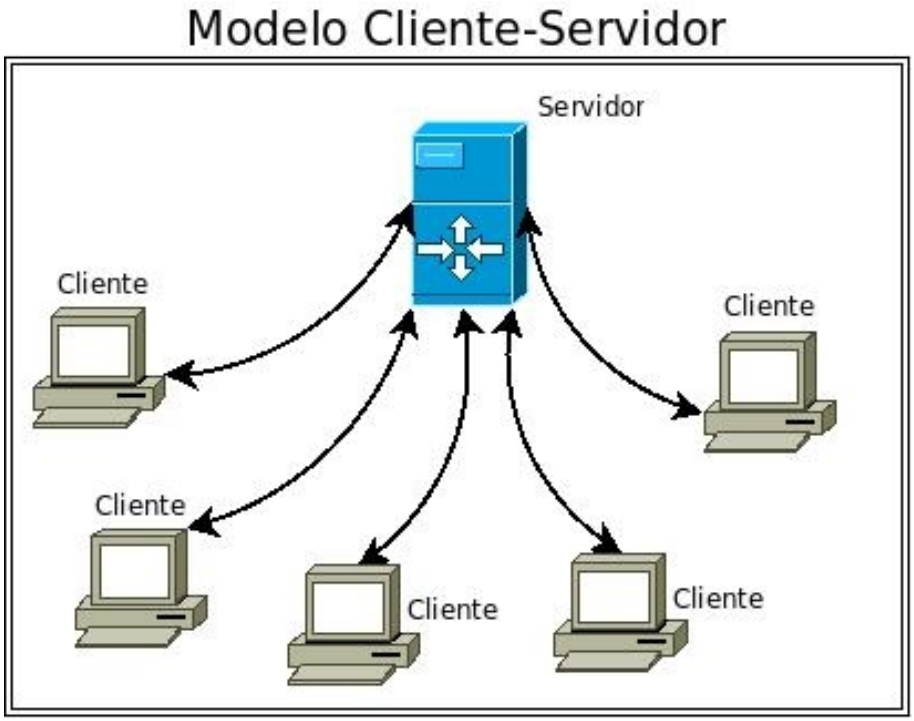


Figura 4.1: Modelo cliente-servidor. Fuente: (redespomactividad, 2019)

4.2 Diseño detallado

En este apartado se va a describir el aspecto gráfico del proyecto. En primer lugar se va a realizar el prototipado donde se determine el aspecto de la plataforma, a continuación se justificará el diseño elegido, el cual se describirá en profundidad.

4.2.1. Prototipado

Un prototipado describe la versión inicial del diseño de un producto. Mediante el prototipo se permite evaluar si se cumplen los objetivos propuestos. También permite identificar como interacciona un usuario con el producto desarrollado.

La herramienta empleada para realizar el prototipado será **Figma**, y consiste en una plataforma para generar prototipos, normalmente basasdos en la web. A partir de ésta, se indicarán tanto los diseños propuestos como el flujo que se dé entre las distintas páginas de la plataforma web desarrollada.

Es importante resaltar que no se muestran todas las ventanas de la plataforma, se han escogido las más representativas para poder ofrecer así una visión lo más cercana a la vesión que será desarrollada.En las siguientes imágenes se muestra el diseño especificado y planteado en los prototipos. Más adelante se describen y se justifican dichos aspectos.

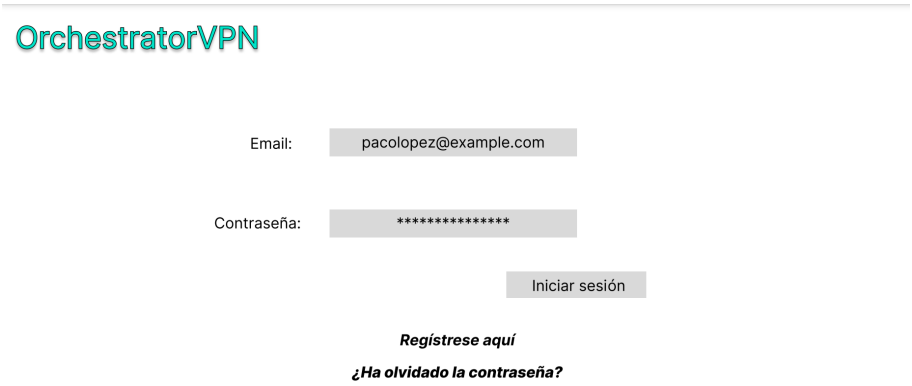


Figura 4.2: Prototipo iniciar sesión en Figma.

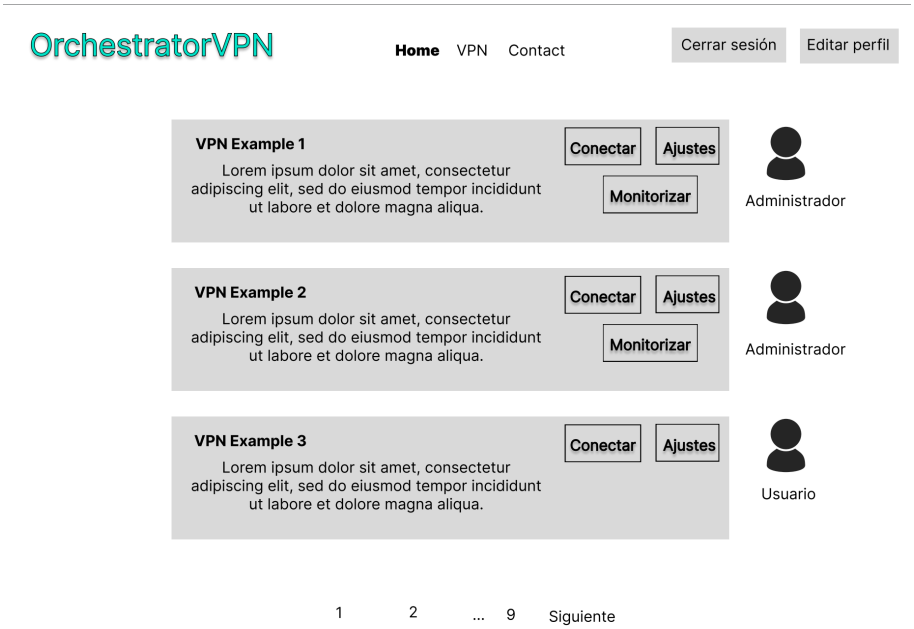


Figura 4.3: Prototipo de la ventana principal en Figma.

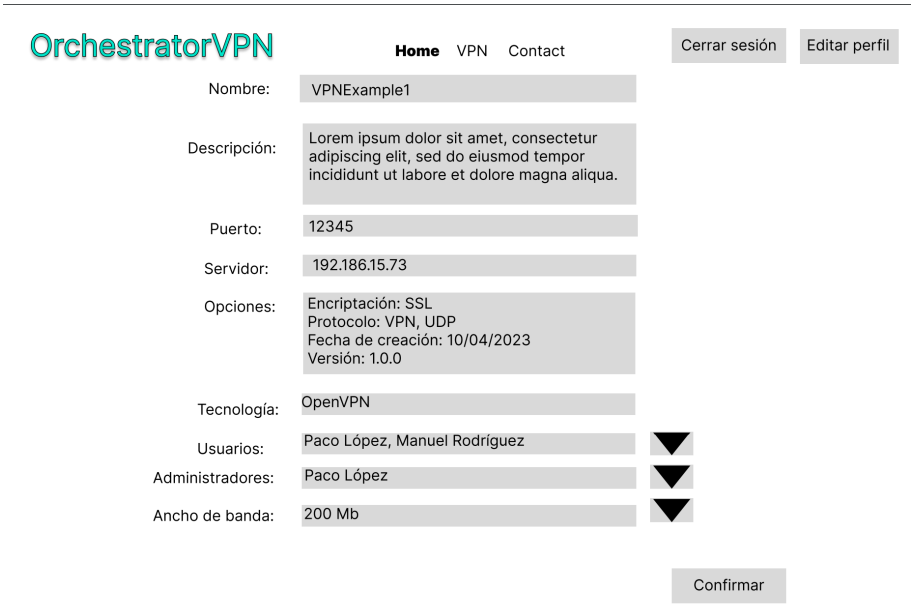


Figura 4.4: Prototipo de la ventana de ajustes de una VPN en Figma.

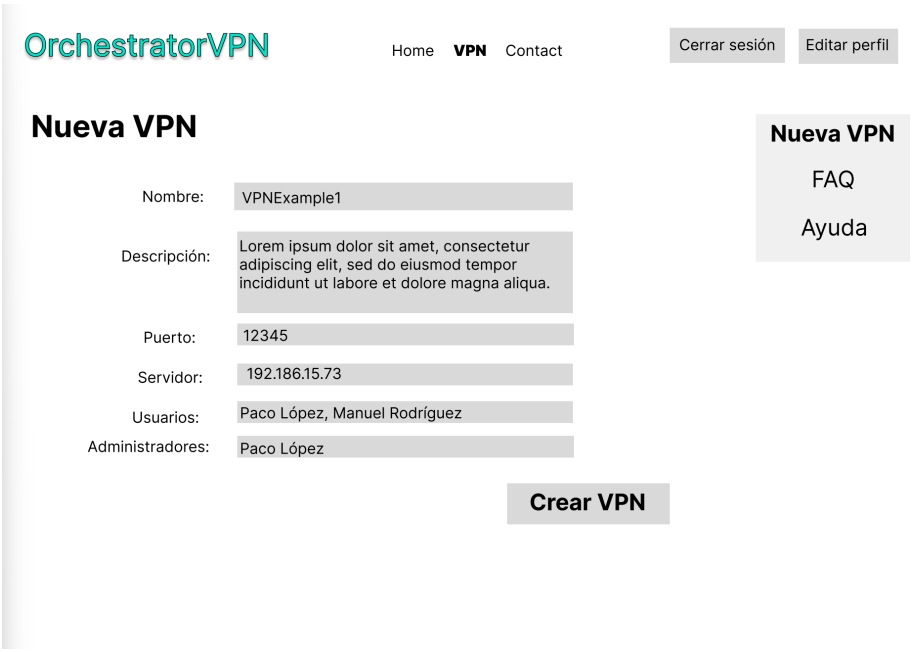


Figura 4.5: Prototipo de la creación de una VPN en Figma.

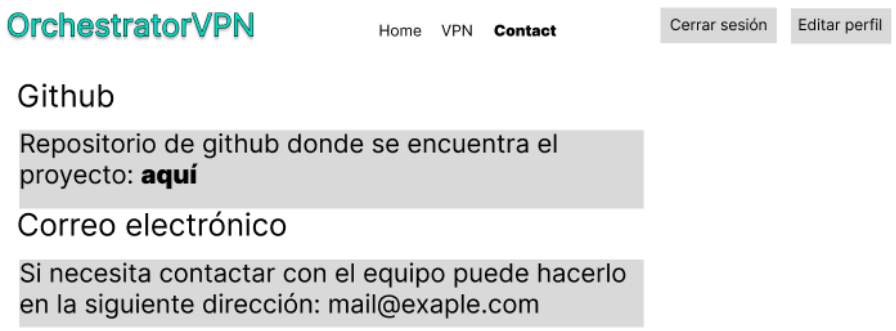


Figura 4.6: Prototipo de la ventana de contacto en Figma.

4.2.2. Justificación del diseño

En primer lugar se ha escogido una paleta de colores básicos, pero que contrasten entre sí como lo son el blanco y el negro. De esta forma será más sencillo para el usuario poder distinguir las diferentes funcionalidades que se ofrecen en la plataforma web. Además se han incluido detalles en azul para enfatizar los títulos y ciertas acciones. Por otra parte cabe destacar que, en caso de que el usuario tenga el navegador web en modo oscuro, los colores se adaptarán para ofrecer una experiencia que se acomode a los deseos del usuario.

Se va a comenzar describiendo el diseño de la ventana de inicio de sesión, mostrada en la figura 4.2. En todas las páginas de la plataforma se encuentra una cabecera en la que se muestra el título, así como el logo de la misma plataforma web que se describirá más adelante. Además, incluye los campos que debe introducir el usuario para iniciar sesión, los cuales son: el correo electrónico y la contraseña. En caso de que el usuario no se haya registrado o no recuerde su contraseña, tendrá disponibles unos enlaces que redirijan al usuario hacia estas páginas. El aspecto de estas será muy similar al planteado en la ventana de inicio de sesión, lo único en lo que diferirán será en los campos que debe cumplimentar el usuario. Una vez haya completado el registro o el inicio de sesión, se redirigirá a la ventana principal.

A continuación, la ventana principal,ver figura 4.3, incluye la cabecera que ha sido descrita con anterioridad; cabe destacar que se han añadido tres botones para que el usuario pueda navegar entre la ventana principal, la ventana de creación de VPN, y la ventana de contacto. Por último se han incluido dos botones en la esquina superior derecha para que el cliente pueda cerrar sesión o editar la información de su perfil. En caso de que el cliente seleccione la opción de cerrar sesión, se redirigirá a la ventana de inicio de sesión, y en el caso de que seleccione la opción de editar perfil, se redireccionará a su ventana pertinente. El cuerpo principal de esta ventana incluirá los distintos clientes a las que el usuario puede acceder. Se dispondrán como bloques en los que se incluirán el nombre, la descripción, un botón para descargar el archivo que servirá para conectarse, otro que muestre los ajustes de la VPN seleccionada, y en caso de ser administrador, se incluirá otro botón para poder monitorizar el tráfico de la red. Para poder navegar entre las distintas VPN se incluirá una paginación para facilitar su acceso al usuario. Si el usuario selecciona el botón de ajustes será redireccionado a la ventana correspondiente. En caso de seleccionar el botón de monitorizar se redirigirá a la ventana pertinente de monitorización de la VPN seleccionada.

La ventana de ajustes, figura 4.4, cuenta con la cabecera que se ha descrito en la ventana anterior. En ella se muestran todas las opciones de la VPN que se utilizaron para crear dicha red. Sin embargo, en caso de ser administrador de la VPN, se incluirán las opciones para poder añadir usuarios a la misma VPN, de tal manera que existirá un desplegable con todos los usuarios para facilitar la selección. Lo mismo ocurrirá para incluir administradores de la red a la misma. Por otra parte, también existe un campo mediante el cual se puede seleccionar un ancho de banda máximo para un determinado usuario que será seleccionado también en el mismo campo. De esta manera se puede ofrecer una gestión en el tráfico, ya que se puede limitar la cantidad de ancho de banda de la que puede disponer un usuario.

También se encuentra la ventana de creación de una VPN, mostrado en la figura 4.5; para acceder a ella basta con seleccionar en la cabecera el botón VPN, y dentro de esta sección encontrará distintas opciones, pero por defecto accederá a la creación de éstas. La función principal, como describe el nombre, se basa en la creación de las redes virtuales privadas. Para ello se proporcionan distintos campos que debe rellenar el usuario para completar la generación de la red. Entre estos aspectos se encuentra: el nombre, la descripción, el puerto, el servidor donde se hospedará, los usuarios que podrán acceder a ella, y los administradores de la misma. Para confirmar los datos y proceder a la creación de la red se usará el botón con el texto "Crear VPN".

Por último se muestra la ventana de contacto descrita en la figura 4.6, la cual muestra las posibles formas con las que el usuario podrá contactar con el equipo para solicitar soporte. Además, se incluye la dirección de Github donde se encuentra todo el código empleado para el desarrollo del proyecto.

Cabe destacar que estos prototipos de vistas no incluyen la totalidad de las mismas, ya que durante el desarrollo del trabajo se han añadido otras vistas complementarias a éstas para ofrecer al usuario una experiencia más gratificante y sencilla.

4.3 Diseño de la base de datos

Para el desarrollo de este proyecto se va a emplear una base de datos relacional [15]. Este tipo de base de datos, como dice el nombre, es aquel que establece relaciones entre sus datos. El principal motivo de su elección es su facilidad a la hora de desarrollarlas incluso en un funcionamiento a gran escala.

En este proyecto se van a realizar constantemente operaciones de escritura y de lectura, que implicarán la creación o actualización de datos, por tanto, el modelo relacional será más efectivo.

Una ventaja que facilitará la creación de esta base de datos, serán las restricciones que ofrece el modelo relacional, se definen como los estados que son permitidos dentro de la base de datos.

Se pueden distinguir cinco tipos de restricciones:

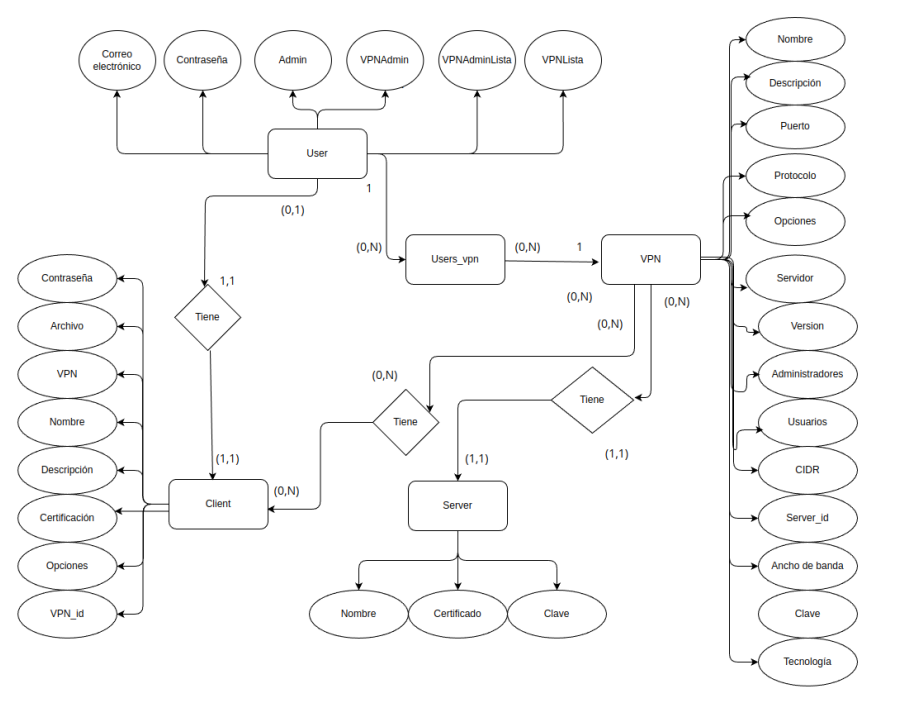
- Restricción no nulo: se imposibilita que se introduzcan valores nulos en una o más columnas.
- Restricción de unicidad: se prohíbe los valores duplicados en una o más columnas.
- Restricción de clave primaria: tiene las mismas propiedades que una restricción de unicidad.
- Restricción de clave ajena: se refiere a una columna o conjunto de ellas que apuntan a la clave primaria de una tabla.

- Restricción de comprobación: mediante esta restricción se especifican los valores que se permiten en una o más columnas de una tabla.

De esta manera, se podrá almacenar todos los datos necesarios para el correcto funcionamiento de la aplicación y se incluirá la menor información posible, de tal manera que se eliminará información redundante y se empleará el menor número de tablas.

4.3.1. Diseño conceptual de la base de datos

El principal objetivo del diseño conceptual es elaborar un esquema de alto nivel de la base de datos, para ello se emplea el diagrama de clases . Este diseño es independiente de la tecnología a partir del análisis de requisitos. En el diagrama de clases mostrado en la Figura 3.6, se muestran todas las clases, cada una de ellas será una tabla nueva en la base de datos, para cada una de ellas se crearán las restricciones pertinentes. En la Figura 4.7, se muestra el diseño conceptual de la base de datos, incluyendo las relaciones entre tablas y sus atributos.



,

Figura 4.7: Diagrama entidad-relación.

4.3.2. Diseño lógico de la base de datos

En este proceso se va a obtener un esquema lógico a partir del diagrama mostrado en la figura 4.7. Para realizar dicho esquema se creará una tabla dentro del mismo esquema por cada entidad del diagrama mostrado en el diagrama entidad-relación.

Dicho esto, el esquema será el siguiente:

Users(correo electrónico, contraseña, admin, adminVPN, VPNLista, VPNAdminLista, identificador(users-vpn),))
CP: ID
Caj: identificador-user(users-vpn) ->identificador-users-vpn(user)

Users-vpn(user-id, vpn-id, identificador-users-vpn(users), identificador-users-vpn(vpn)))
CP: user-id | vpn-id
Caj: identificador-user(users-vpn) ->identificador-users-vpn(user)

VPN(Nombre, descripción, protocolo, CIDR, puerto, opciones, servidor, versión, administradores, usuarios, identificador(Servidor), identificador(users-vpn), clave de host, certificado)
CP: ID
Caj: identificador(Servidor) ->Servidor(Nombre)
Caj: nombre(Cliente) ->Cliente(Nombre)

Client(VPN, nombre, descripción, certificado, opciones, contraseña, vpn-id, archivo)
CP: ID

Server(Nombre, certificado de la CA, clave de la CA)
CP: ID

4.4 Tecnologías utilizadas

A lo largo de esta sección se van a analizar las principales tecnologías usadas para el desarrollo de este proyecto. Para ello se describen brevemente las herramientas empleadas.

4.4.1. Ruby on Rails [1]

Es un `framework` de Ruby empleado para el desarrollo de aplicaciones web. Sigue el paradigma de modelo, vista y controlador o MVC[16], el cual separa las definiciones de las clases que serán necesarias en el proyecto, la parte visual de la misma y las clases en las que se incluyen los métodos que darán funcionalidad a la plataforma. Ruby permite la metaprogramación y Ruby on Rails se aprovecha de ello, es por eso que se puede usar mucho menos código y conseguir una mayor eficiencia que empleando otros lenguajes. Se han empleado distintas librerías, en el caso de Ruby se denominan gemas, como `Devise`. Ésta se ha usado para desarrollar la autenticación de los usuarios de una manera más sencilla y efectiva.

4.4.2. Tailwind CSS

Tailwind es un `framework` de CSS empleado para el diseño de páginas web. Crea listas de clases de utilidad de CSS que pueden ser utilizadas para otorgar estilos de manera individual.

Las clases de utilidad se refieren a que se crean clases en torno a las distintas características que se describen. Tailwind permite que determinadas clases de utilidad sean usadas específicamente en determinados momentos, como cuando la plataforma ha sido accedida desde un navegador en un dispositivo móvil, o cuando el cursor de un ratón está encima de un botón. En el archivo "tailwind.config.js"se pueden configurar los valores de las clases de utilidad.

4.4.3. OpenVPN

Es una herramienta que ofrece conectividad punto a punto y con los usuarios conectados de manera remota. Proporciona funciones para la creación de VPN y distintos aspectos relevantes para la gestión de las mismas, estas redes estarán cifradas mediante SSL de esta manera se puede ofreer una comunicación segura.

Dentro de OpenVPN se encuentra el servidor y los clientes. El primero será el que contenga la información y está configurado de tal manera que acepte las conexiones desde el cliente y viceversa. Los clientes serán las distintas VPN que se crean y a las que los usuarios podrán conectarse.

4.4.4. SQL

SQL es un lenguaje creado para trabajar con bases de datos, se utiliza para administralas y poder recuperar información de sistemas de gestión de bases de datos relacionales. Este lenguaje declarativo permite una alta productividad a la hora de codificar ya que mediante una sola línea se pueden realizar operaciones muy potentes computacionalmente.

Mediante este lenguaje se ha desarrollado una base de datos en la que se van a almacenar los datos y se empleará para solicitar las consultas que demandarán los datos que deberán mostrarse en la plataforma.

4.4.5. Github

Github es una plataforma de desarrollo colaborativo empleado para alojar distintos proyectos usando el sistema de control de versiones Git. Permite gestionar y controlar las versiones de los distintos proyectos alojados en el repositorio. Con ofrecer control de versiones se hace referencia a permitir a los desarrolladores organizar los cambios en el software a medida que se avanza en el proyecto.

Durante la elaboración del orquestador, conforme se vaya avanzando con la elaboración del código, se irán subiendo actualizaciones de las versiones a esta herramienta online donde se alojará dicho software.

4.4.6. RRDTool

Es una herramienta de código abierto ampliamente utilizada para la gestión y almacenamiento de datos de series temporales. Su principal utilidad radica en la recopilación, almacenamiento y representación gráfica de datos que varían en función del tiempo.

En este proyecto, esta herramienta se utiliza para recopilar datos sobre el tráfico de red, para poder ofrecer gráficos en tiempo real. Así el usuario puede monitorizar en todo momento las redes que tenga disponible.

4.4.7. RSpec

RSpec es una biblioteca de pruebas diseñada para facilitar el enfoque *BDD* en aplicaciones Rails. Con su sintaxis legible, amplia cobretura de tipos de pruebas y configuración flexible, RSpec permite a los desarrolladores escribir pruebas claras, concisas y bien estructuradas para verificar el comportamiento de sus aplicaciones web.

CAPÍTULO 5
Desarrollo de la solución propuesta

A lo largo de este capítulo se va a describir todo el proceso de desarrollo que se ha realizado para poder implementar la aplicación.

5.1 Desarrollo del proyecto

Este trabajo ha sido desarrollado, entre otras tecnologías, mediante Ruby on Rails; un framework destinado a la creación de páginas web escrito en el lenguaje de programación **Ruby**.

Rails ofrece muchas facilidades ya que posibilita la metaprogramación, lo cual ha permitido muchas facilidades a la hora de desarrollar código. Estas herramientas facilitarán la creación de los modelos, controladores y vistas, ofreciendo potentes configuraciones y empleando unas pocas líneas de código. Por otra parte, también se encargará de la creación y la gestión de la base de datos. Que, con la metaprogramación, resulta mucho más sencilla de implementar. En este caso, la base de datos será creada mediante el lenguaje SQL.

Por otra parte, otra peculiaridad que posee este entorno es el uso de *gemas*, que son las librerías del lenguaje de programación de Ruby. El uso de éstas permite incluir una gran variedad de funcionalidades al proyecto sin la necesidad de incluir muchas líneas de código. Cabe destacar que se usará código nativo de la consola de Linux para poder ejecutar los comandos de OpenVPN. Sin embargo, Rails permite utilizar código de este tipo. Por lo tanto, se integrarán dichas porciones en el mismo de Rails y podrá ser ejecutado sin ningún problema.

Las VPN y clientes que se vayan creando, se instalarán en una máquina servidor que se utilizará a su vez para ejercer todo el despliegue, este apartado se desarrolla en su totalidad en el capítulo 6 de este proyecto. Los servidores se encargarán de ofrecer toda la gestión para las VPN que se deseen crear.

Para el uso de los clientes de OpenVPN, el usuario solicita su descarga y se instalan en la máquina del cliente. Una vez descargado el cliente, el usuario tendrá que importar dicho archivo a su aplicación OpenVPN y, a continuación, podrá conectarse a dicha red.

Para ofrecer una monitorización sobre las VPN se ha instalado en la máquina servidor una herramienta que pueda controlar el tráfico que pasa por una interfaz de red. Mediante los datos que se obtengan, se generan archivos que servirán para crear gráficas en tiempo real. Gracias a éstas, el usuario podrá observar como discurre el tráfico por dichas redes.

5.2 Estructura de la solución

A lo largo de este apartado se va a detallar la estructura de todo el proyecto y ,además, se va a explicar cada una de sus partes con detalle.

5.2.1. Modelos

Para saber que es un modelo primero hay que conocer que es *ActiveRecord*. *ActiveRecord* es una capa que permite al usuario acceder y manipular la información almacenada en la base de datos sin necesidad de escribir código en SQL. El modelo en sí, es una clase de Ruby que se representa en una tabla de la base de datos, en el paradigma MVC implementa las reglas y la lógica de negocio. Por convención de Ruby, el nombre del modelo coincide con el de la tabla, sin embargo, este último se escribe en plural y en minúsculas. En éste no se describen las columnas de la tabla, *ActiveRecord* las toma directamente de la tabla.

Como se ha mencionado, se crean tantos modelos como tablas hayan en la base de datos. En la figura 4.7 se muestra el diagrama entidad-relación, el cual se corresponde con el diseño de la base de datos. Se observan cuatro clases: usuario, VPN, cliente y servidor. Por lo que existirá un modelo por cada una de estas clases. Cada uno de ellos se describirá más detalladamente a continuación. Todos ellos se encuentran en la ruta *app/models*.

Usuario

Este modelo se corresponde con cada usuario registrado en el sistema. Para poder acceder a la página es necesario darse de alta, si el usuario no lo hace no tendrá la posibilidad de entrar. Se encuentran tres tipos de usuario: usuario corriente, administrador de VPN y administrador general.

El usuario corriente podrá visualizar las VPN a las cuales puede conectarse, además de poder visualizar sus ajustes y poder crear otras VPN. Por otro lado está el administrador de VPN, el cual, además de tener las funcionalidades de un usuario común, podrá monitorizar las redes de las cuales es administrador, además de poder tener otras herramientas de gestión sobre la red. Por último se encuentra el administrador general, éste podrá utilizar todas las herramientas previamente descritas, pero sobre todas las redes existentes en el sistema, además de poder designar otros administradores y crear o eliminar servidores.

En este modelo se definen los métodos básicos sobre el modelo, tales como si el usuario está autenticado, si es administrador o administrador de la VPN, entre otras. Por otra parte, se detalla que los datos del registro son recuperables, validados y que se puedan recordar. También se define la referencia del modelo **Usuario** con el de **VPN**, que en este caso es una relación de muchos a muchos; ya que un usuario puede tener la cantidad de VPNs que desee.

Los atributos serán los siguientes:

- Correo electrónico: el correo con el cual se ha registrado el usuario en el sistema.
- Contraseña: a la hora de registrarse, el usuario ha empleado una contraseña para poder proteger su información. Esta contraseña aparece cifrada en la base de datos para evitar posibles vulnerabilidades.
- Admin: indica si el usuario es un administrador global.

- VPNAdmin: indica si el usuario es administrador de una VPN en concreto.
- VPNAdminLista: indica la lista de VPN en la que el usuario actúa como administrador.
- VPNLista: muestra la lista de VPN a las cuales puede acceder el usuario.

Usuario-vpn

Este modelo como se ha comentado es auxiliar para poder establecer las referencia correctamente entre usuario y VPN. De tal manera que existirá una única instancia en la cual se una el identificador del usuario y el identificador de la red virtual privada. Sobre esta indexación existirá un atributo que determine si ese usuario actúa como administrador sobre esa VPN. En la descripción de este modelo únicamente se han establecido las referencias a las otras tablas.

En este modelo la lista de atributos es la que se ve a continuación:

- user-id: el identificador del usuario sobre el cual se establece la relación con la VPN pertinente.
- vpn-id: es el identificador de la VPN mediante el cual se crea la conexión con el usuario.
- vpnadmin: determina si el usuario en la VPN mencionada, es administrador o no.

VPN

El modelo asociado a las VPN, describe cada una de las redes virtuales privadas que han sido creadas por los usuarios. En este modelo se define también la relación especificada anteriormente. A la hora de crearla será necesario recibir el servidor en el cual se hospedará la red. Estableciendo así la relación pertinente con el servidor en el cual se hospedará. Mediante esta relación, la VPN podrá generar sus certificados a través del servidor y adaptar su configuración según los requisitos que se le hayan establecido en su respectivo formulario de creación.

Este modelo también estará relacionado con el modelo cliente, de tal manera que una VPN podrá tener múltiples clientes; los cuales se conectarán a la red asociada. La creación del cliente, empleará los ajustes configurados en la creación de la VPN y podrá añadir una configuración que incluya el nombre del cliente, una descripción y ,en caso de solicitar el uso de una contraseña, una contraseña.

La tabla referente a las VPNs constará de los siguientes campos:

- Nombre: el nombre descriptivo de la VPN
- Descripción: una breve descripción de la finalidad de la red, así como la información que el administrador quiera proporcionar.
- CIDR: es un sistema de asignación de direcciones IP que permite una asignación más flexible y eficiente de las direcciones IP en las redes de las computadoras. En resumen, es el rango de direcciones IP que se enrutará en los clientes conectados a esta VPN.
- Puerto: el puerto sobre el cual se conectarán a la red.

- Opciones: todos los ajustes descritos para la configuración de la VPN en su formulario de creación.
- Servidor: el nombre del servidor en el cual se alojará la red.
- Version: el número de versión de la VPN.
- Tecnología: la tecnología empleada para crear esta VPN, en nuestro caso siempre será **OpenVPN**.
- Usuarios: la lista de usuarios que podrán conectarse a esta VPN.
- Administradores: la lista de usuarios que serán los administradores de esta red y podrán ejercer las funciones correspondientes a los mismos.
- server-id: el identificador del servidor, se incluye meramente para poder establecer la referencia correctamente.
- Certificado: certificado generado que da validez a la red creada.
- Clave privada: clave para poder lograr la conexión correctamente

En la descripción a este modelo se han descrito además de las referencias a las otras tablas, una condición que expresa que es necesario que en el formulario se le establezca el parámetro del identificador del servidor, el cual se recoge al seleccionar un servidor.

Servidor

El modelo servidor representa a un servidor de OpenVPN, el cual tendrá una asociación uno a muchos con el modelo VPN, lo que significa que éste podrá tener varias VPN. Este servidor se corresponde con la máquina que hospeda todo el sistema, se encargará de realizar todas las peticiones de los clientes.

Los atributos son estos:

- Nombre: una cadena de texto que almacena el nombre del servidor.
- Certificado de la CA: almacena el certificado de la autoridad de certificación que es utilizada por el servidor.
- Clave de la CA: almacena la clave de la autoridad de certificación que es utilizada por el servido.

Cliente

Se corresponde con las instancias de clientes asociados a una VPN, los cuales se configurarán a partir de los ajustes preestablecidos en su respectiva VPN.

Este modelo contiene los siguientes campos:

- VPN: es el nombre de la VPN sobre la cual se está creando el cliente.
- Nombre: es el nombre del cliente.
- Descripción: una breve explicación del uso del cliente.
- Contraseña: en caso de utilizarla, se almacenará la clave.

- Opciones: la configuración creada para este cliente.
- Archivo: contendrá el archivo que se genere al crear un cliente.

Este modelo mantiene una asociación de pertenencia con el modelo VPN, lo que significa que un cliente pertenece a una VPN específica.

5.2.2. Vistas

Las vistas almacenan el código que visualizará el usuario, es decir, las interfaces de la misma. Este código permitirá renderizar los posibles estados del sistema. En el proyecto en el cual se está trabajando dichas vistas definen como se presenta la información de las VPN, en los prototipos se muestra el aspecto deseado de las mismas.

En el framework de Ruby on Rails, todas las vistas utilizan la disposición ubicada en la ruta *app/views/layouts/application.html.erb*, por lo que cualquier cambio que se haga en este archivo modificará cualquiera de las vistas que se hayan creado. Para este trabajo, este archivo contendrá el código de la cabecera de la aplicación, en el cual se muestra el título de la aplicación, los botones que permiten al usuario redirigirse a cualquiera de las ventanas principales en caso de estar dado de alta y ,por último, en la esquina superior derecha existirán dos botones que permitan al usuario cerrar la sesión o editar su perfil, en caso de haber iniciado sesión, en caso contrario existirán dos botones que redirijan al usuario a la ventana indicada para iniciar sesión o registrarse.

En el archivo descrito anteriormente, se encuentra en el código la palabra *yield*, indica que a partir de ahí se reemplaza el código por el contenido indicado de cada vista empleada. Por ejemplo, en el caso de que el usuario desee crear una VPN, se mostrará el código de la cabecera; ya que este se carga siempre y, a continuación, se incluirá el código de la vista referida a esta acción.

Por convención en este framework se crea una vista por cada controlador generado, sin embargo, en este proyecto hay distintos controladores que utilizan diferentes vistas. El controlador de VPN utiliza hasta 6 vistas, entre ellas se incluyen la visualización de las mismas o la vista donde existe un formulario a rellenar para poder crear correctamente una VPN. En estas vistas también se incluye código auxiliar para resaltar cierta información y dotar a la aplicación con ciertos detalles, de esta manera se facilita su uso al usuario.

Para los modelos los cuales necesiten un formulario de creación, como es el caso de los servidores, las VPN y los clientes, se han generado otras vistas auxiliares al formulario en si, tales como, una donde se desplieguen los objetos creados, dependiendo del tipo de usuario se observarán de una forma u otra, una vista que permita la edición de los objetos, así como su eliminación del sistema. Esta creación de vistas se ha generado mediante una funcionalidad del framework de *Ruby on Rails* llamada **scaffold**, permite generar automáticamente código tanto para un modelo, controlador y vistas. Al ejecutar esta función se crea todo el código necesario para la gestión del recurso especificado en la aplicación, incluyendo la migración de la base de datos, las rutas en el archivo de configuración y las vistas en el HTML que serán las encargadas de gestionar las operaciones CRUD [17].

5.2.3. Controladores

Los controladores se utilizan para modificar el modelo y las vistas, dependiendo de como el usuario interactúe con la aplicación, es decir, gestionan y procesan las peticiones HTTP que se realicen. En el proyecto, todos los controladores se encuentran en la

ruta *app/controllers*. Existirá un controlador para cada vista que necesite un control de su lógica.

En el caso de la vista de autenticación o de registro de usuario se encuentran los controladores dedicados a la creación de sesiones o destrucción de las mismas al iniciar o cerrar la sesión por parte de un usuario previamente registrado. Lo mismo ocurre para la gestión de registros de nuevos usuarios, editar la información de los usuarios creados, entre otras funcionalidades.

En la vista principal se tiene un controlador para poder mostrar todos los clientes que pueda visualizar el usuario, a continuación se encuentran otros controladores que sirven para renderizar las vistas referidas a enseñar la configuración y la monitorización. También se encuentra un método utilizado para la función conectar, la cual sirve para lanzar a ejecución el código necesario para que el usuario se descargue el archivo cliente que permita realizar la conexión con la VPN.

En la vista referida a las VPN, se definen distintos métodos los cuales muestras todas las VPN creadas, facilitan la creación de una nueva red virtual privada; éste almacenará los datos del formulario rellenados por el usuario en distintas variables que serán pasadas como argumento a la hora de ejecutar el código elaborado para crear una nueva red. Además existe otro método con una operativa similar que sirve para eliminar la VPN seleccionada.

Por otra parte se tiene el controlador de los servidores, el cual contiene los métodos tanto de destrucción, creación, actualización. La función de creación recibe desde el formulario todos los parámetros necesarios, además con ellos se genera la cadena de texto que será almacenada en el campo de opciones. Mediante los parámetros recogidos desde el formulario, se lanzará a ejecución un script que permitirá la creación de los servidores en el sistema.

También se tiene el controlador de la vista de usuarios, que puede observar el administrador. Desde ésta, se puede designar a los usuarios como administradores globales o usuarios normales. Por otra parte también se puede hacer que un determinado usuario pase a ser adminsitrador de una VPN o viceversa.

5.2.4. Código funcional

Para poder dotar de funcionalidad a la aplicación, se han creado distintos códigos. Entre ellos cabe destacar aquel que crea un cliente de VPN. Para poder desarrollarlo, se ha hecho mediante códido en la consola de Linux. Se van a destacar las partes más características. Todos estos scripts se encuentran en la carpeta *vendor/sh/*.

En el siguiente segmento de código se verifica si existe un cliente en el archivo donde se registran los mismos. En el caso de existir, sale de la función, si no existe continuará con la ejecución.

```
#!/bin/bash

# Verificar si el cliente ya existe
CLIENTEXISTS=$(tail -n +2 /etc/openvpn/easy-rsa/pki/index.txt |
grep -c -E "/CN=$CLIENT\$")
if [[ $CLIENTEXISTS == '1' ]]; then
  echo ""
  echo "El cliente especificado ya existe en easy-rsa, por favor elige otro nombre."
  exit
fi
```

En este caso se ejecuta un comando para conseguir la generación del cliente dependiendo de si se ha establecido la contraseña en el formulario de la creación. El comando easyrsa es una herramienta para crear y gestionar una autoridad certificadora dentro de una infraestructura de clave pública, cuyo conjunto de tecnologías, políticas y procedimientos que facilitan la gestión de claves públicas y certificados digitales.

```
# Generar el cliente utilizando easyrsa
cd /etc/openvpn/easy-rsa/ || exit
case $CONTRASENA in
  "")
    ./easyrsa --batch build-client-full "$CLIENT" nopass
    ;;
  *)
    echo " Se te solicitará la contraseña del cliente a continuación "
    ./easyrsa --batch build-client-full "$CLIENT"
    ;;
esac
echo "Cliente $CLIENT añadido."
```

Por otra parte se va a explicar con brevedad el script dedicado a la creación de servidores. En la siguiente porción de código se observa como se crea el servidor. La primera línea ejecuta un comando mediante la herramienta *easy-rsa*, la ejecución del mismo será en modo no interactivo y se generará un certificado de servidor completo para el nombre almacenado en la variable correspondiente al nombre del servidor. Además el servidor carecerá de contraseña. La segunda línea define que el período de validez del certificado de revocación, el cual será de 3650 días. Posteriormente, se ejecuta el comando 'gen-crl' el cual genera un archivo correspondiente a la lista de recovación de certificados, que contiene información sobre los certificados que han sido revocados previamente a su fecha de vencimiento.

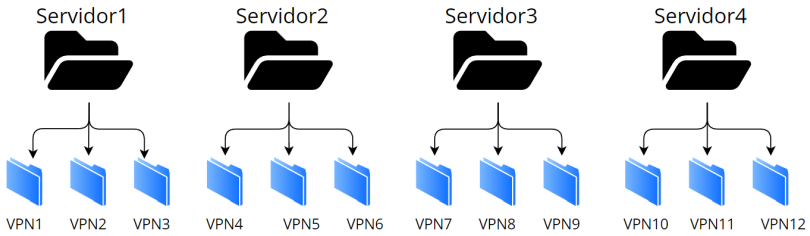
```
#Generación del servidor
./easyrsa --batch build-server-full"$SERVER_NAME" nopass
EASYRSA_CRL_DAYS=3650 ./easyrsa gen-crl
```

En el caso de eliminación de clientes se ejecuta otro comando *easy-rsa*, el cual se ejecuta en modo no interactivo. El comando *revoke* revoca el certificado asociado al cliente especificado en la variable cliente. A continuación se genera una nueva lista de revocación de certificados, dándole un periodo de validación de 3650 días. Por otra parte se elimina la carpeta asociada a dicho cliente y ,por último, se utiliza el comando *sed* para eliminar la línea asociada al cliente en el archivo de clientes miembros en ese servidor, de esta forma ese cliente ya no aparecerá como disponible dentro de dicho servidor.

```
cd /etc/openvpn/easy-rsa/ || exit
./easyrsa --batch revoke "$CLIENT"
EASYRSA_CRL_DAYS=3650 ./easyrsa gen-crl
rm -f /etc/openvpn/crl.pem
cp /etc/openvpn/easy-rsa/pki/crl.pem /etc/openvpn/crl.pem
chmod 644 /etc/openvpn/crl.pem
rm -r "/$ruta/$CLIENT"
sed -i "/^$CLIENT,.*/d" /etc/openvpn/ipp.txt
cp /etc/openvpn/easy-rsa/pki/index.txt{,.bk}
```

Los scripts dedicados a la generación de VPN, clientes y servidores se muestran con un mayor detalle en los anexos correspondientes. Por lo que se recomienda su lectura previa para identificar cuales son los objetivos de éstos y los resultados que se obtendrán con su ejecución.

Por último cabe destacar que para facilitar la gestión y el almacenamiento de las VPN, será de la siguiente forma: se creará una carpeta por cada uno de los servidores y dentro de cada uno de ellos se creará una carpeta por cada una de las VPN creadas en dichos servidores. La estructura mencionada se ve en la figura 5.1.



,

Figura 5.1: Estructura de los archivos creados.

Dentro de cada carpeta dedicada a las VPN, se encontrarán sus archivos de configuración, sus credenciales y claves. Además se incluye un directorio denominado *Clients* el cual recogerá cada uno de los clientes generados sobre esa red.

5.2.5. Archivos generados

En este apartado se van a explicar los archivos que se generen al crear tanto las VPN como los clientes. Se va a describir su estructura y se va a justificar la misma

VPN

Al generar una VPN, se crean tres archivos distintos, el archivo de configuración, el certificado y la clave de la misma.

El archivo de configuración de una VPN de OpenVPN es un archivo de texto que contiene los parámetros y opciones necesarios para establecer y controlar la conexión de la VPN. Este archivo se utiliza para configurar el comportamiento de OpenVPN y define cómo se establecerán los túneles VPN y cómo se cifrarán y autenticarán los datos.

Para este archivo de configuración de ejemplo:

```
port 1194
proto udp
dev tun
user nobody
group nogroup
persist-key
dh dh.pem
persist-tun
```

```
keepalive 10 120
topology subnet
server vpn.thenotes.co 255.255.255.0
management 8694
ifconfig-pool-persist ipp.txt
push "dhcp-option DNS 1.1.1.1"
push "dhcp-option DNS 8.8.8.8"
push "redirect-gateway def1 bypass-dhcp"
ecdh-curve prime256v1
tls-crypt tls-crypt.key
crl-verify crl.pem
ca ca.crt
cert Example.crt
key Example.key
auth SHA256
cipher AES-128-GCM
ncp-ciphers AES-128-GCM
tls-server
tls-version-min 1.2
tls-cipher TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256
client-config-dir /etc/openvpn/ccd
status /var/log/openvpn/status.log
verb 3
```

Se tiene la siguiente configuración:

- port 1194: Especifica el puerto en el que el servidor OpenVPN escuchará las conexiones entrantes.
- proto udp: Define el protocolo de transporte que se utilizará para establecer la conexión. En este caso, se utilizará UDP.
- dev tun: Indica que se utilizará un dispositivo de red virtual tipo "tun"para el túnel VPN.
- user nobody y group nogroup: Especifican el usuario y el grupo de sistema bajo los cuales se ejecutará el proceso de OpenVPN, lo que brinda una capa adicional de seguridad.
- persist-key y persist-tun: Permiten que el proceso de OpenVPN persista en la reutilización de claves y persista en la persistencia del túnel VPN, respectivamente.
- keepalive 10 120: Configura el intervalo de tiempo en segundos en el que OpenVPN enviará paquetes de keepalive al cliente para mantener viva la conexión.
- topology subnet: Define la topología de red para la VPN como una subred.
- server vpn.thenotes.co 255.255.255.0: Especifica el dominio y la máscara de red del servidor VPN.
- management 8694: Establece la interfaz de administración de OpenVPN en el puerto 8694.
- ifconfig-pool-persist ipp.txt: Guarda el estado persistente de la asignación de direcciones IP a clientes en el archivo ïpp.txt".

- push dhcp-option DNS 1.1.1.1 y push dhcp-option DNS 8.8.8.8: Empuja opciones DHCP al cliente para configurar los servidores DNS que se utilizarán.
- push redirect-gateway def1 bypass-dhcp: Redirige todo el tráfico de Internet del cliente a través de la conexión VPN.
- ecdh-curve prime256v1: Especifica la curva elíptica que se utilizará para el intercambio de claves ECDH.
- tls-crypt tls-crypt.key: Habilita el cifrado adicional utilizando una clave compartida definida en el archivo tls-crypt.key.
- crl-verify crl.pem: Verifica la lista de revocación de certificados (CRL) utilizando el archivo crl.pem.
- ca ca.crt: Especifica el archivo de certificado de autoridad (CA) raíz.
- cert Example.crt y key Example.key: Indican los archivos de certificado y clave que se utilizarán para la autenticación del servidor.
- auth SHA256: Configura el algoritmo de hash que se utilizará para la autenticación de mensajes.
- cipher AES-128-GCM y ncp-ciphers AES-128-GCM: Establecen el cifrado simétrico que se utilizará para la comunicación en el túnel VPN.
- tls-server: Indica que el servidor OpenVPN actuará como servidor TLS

Por otra parte se genera el certificado. El certificado de una VPN es esencial para garantizar la autenticación mutua entre el cliente y el servidor, establecer conexiones seguras y cifradas, y proteger la integridad y la confidencialidad de los datos transmitidos a través de la red privada virtual.

Por último, se crea la clave privada. Es necesaria para descifrar, firmar y autenticar los datos en una VPN. Trabaja en conjunto con el certificado y la clave pública para proporcionar seguridad en la autenticación, el cifrado y la protección de las comunicaciones en la red privada virtual.

Cliente

A la hora de crear un cliente, se genera su archivo .ovpn. Este es n archivo de configuración utilizado por el software OpenVPN para establecer una conexión VPN. Contiene todos los parámetros necesarios para configurar tanto el cliente como el servidor de la VPN. El archivo .ovpn es generado por el administrador de la VPN y luego se distribuye a los clientes para que puedan configurar fácilmente su conexión.

A continuación se adjunta un archivo de ejemplo, obviando los certificados y claves necesarios para la autenticación y el cifrado en la conexión de la VPN.

```
    client
proto udp
explicit-exit-notify
remote ::1 1194
dev tun
resolv-retry infinite
nobind
```

```
persist-key
persist-tun
remote-cert-tls server
verify-x509-name Example name
auth SHA256
auth-nocache
cipher AES-128-GCM
tls-client
tls-version-min 1.2
tls-cipher TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256
ignore-unknown-option block-outside-dns
setenv opt block-outside-dns # Prevent Windows 10 DNS leak
verb 3
ifconfig 10.8.0.3 255.255.255.0
ifconfig 10.8.0.4 255.255.255.0
```

Se procede a explicar los campos adjuntados:

- client: Indica que este archivo de configuración es para un cliente OpenVPN.
- proto udp: Especifica que se utiliza el protocolo UDP para la comunicación de la VPN.
- explicit-exit-notify: Notifica explícitamente al cliente cuando la conexión VPN se cierra.
- remote ::1 1194: Indica la dirección IP o el nombre de dominio del servidor VPN y el puerto utilizado (en este caso, ::1 representa la dirección IPv6 loopback y 1194 es el puerto).
- dev tun: Especifica el tipo de dispositivo de red utilizado por la conexión VPN (en este caso, "tun"para túnel).
- resolv-retry infinite: Configura el cliente para volver a intentar resolver las consultas DNS infinitamente si no se puede resolver inicialmente.
- nobind: Permite al cliente conectarse a cualquier puerto disponible en el servidor.
- persist-key y persist-tun: Indican que el cliente debe mantener persistencia en la clave y en el túnel respectivamente.
- remote-cert-tls server: Configura el cliente para verificar el certificado del servidor.
- verify-x509-name Example name: Verifica el nombre del certificado del servidor para asegurarse de que coincida con el valor proporcionado (.Example.en este caso).
- auth SHA256: Utiliza el algoritmo de hash SHA256 para la autenticación.
- auth-nocache: No almacena en caché las credenciales de autenticación después de la autenticación inicial.
- cipher AES-128-GCM: Especifica el algoritmo de cifrado AES-128-GCM para la conexión VPN.
- tls-client y tls-version-min 1.2: Configuran el cliente para utilizar el protocolo TLS y especifican que se debe utilizar como mínimo la versión 1.2.

- tls-cipher TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256: Indica el conjunto de cifrado TLS que se utilizará para la comunicación.
- ignore-unknown-option block-outside-dns y setenv opt block-outside-dns: Evitan que el cliente utilice servidores DNS externos.
- verb 3: Establece el nivel de verbosidad de los registros de OpenVPN a 3 (detallado).
- ifconfig 10.8.0.3 255.255.255.0: Asigna la dirección IP y la máscara de subred al cliente cuando se establece la conexión VPN.
- ifconfig 10.8.0.4 255.255.255.0: Asigna la dirección IP y la máscara de subred al cliente cuando se establece la conexión VPN.

Los parámetros pueden variar según las necesidades y configuraciones aplicadas a la VPN. Cabe destacar que el cliente posee dos direcciones IP ya que para el túnel necesita una dirección para cada extremo del mismo.

5.3 Análisis del funcionamiento de la aplicación

A lo largo de esta sección va a explicar en detalle cada parte del proyecto desarrollado. En cada uno de los siguientes apartados se expone cual ha sido la metodología empleada y su adecuada justificación. Es por ello que se ha dividido esta sección en cuatro apartados, para poder diferenciar las distintas utilidades que ofrece esta herramienta.

5.3.1. Autenticación y registro

La autenticación consiste en verificar que un usuario existe en la base de datos del sistema y, por lo tanto, se ha registrado en él. Para poder acceder a la aplicación, es necesario que el usuario se haya registrado correctamente en el sistema, para ello es necesario cumplimentar el formulario de creación de usuario, en el cual se pide un correo electrónico, una contraseña y la confirmación de la contraseña escogida por el usuario. Como se puede ver en la figura 5.2

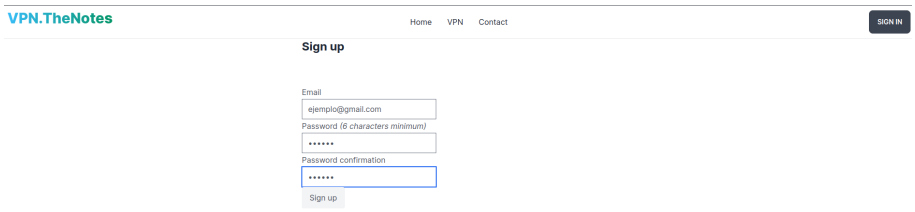


Figura 5.2: Ventana de registro de usuario.

Una vez el usuario ha sido registrado, se redirige directamente a la pantalla principal. Sin embargo, en el caso de que se cierre su sesión u otro usuario desee iniciarla, deberá adjuntar la dirección de correo electrónico con la que se ha registrado, así como la contraseña para poder iniciar sesión. Así como se muestra en la figura 5.3

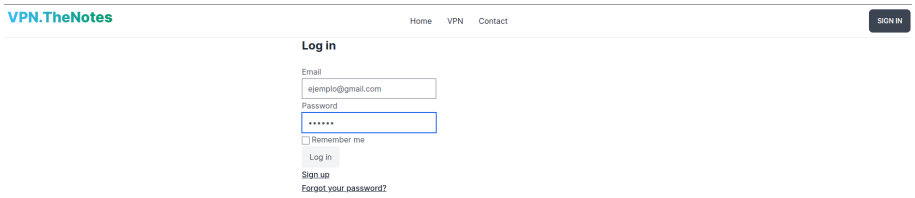


Figura 5.3: Ventana de inicio de sesión.

A la hora de gestionar la autenticación y el registro de usuarios, se ha usado una gema de Rails denominada **Devise**. **Devise** es una gema que implementa estas funcionalidades de manera muy sencilla. Para poder incluirla en la aplicación, se debe añadir la gema en el archivo *Gemfile*, a continuación se debe instalar la gema mediante el siguiente comando en la terminal:

```
bundle install
```

Tras haber instalado la gema, se va a ejecutar el siguiente comando para generarlo en el proyecto y, con el código posterior, se crea el modelo usuario. Tras haber creado el modelo correctamente se debe lanzar las migraciones para generarlas en la base de datos.

```
rails generate devise:install
rails generate devise User
```

Esta gema crea distintas rutas para gestionar los registros, el inicio y cierre de sesión, entre otras. También se incluyen otras para editar la contraseña del usuario registrado o recuperarla. Por otra parte las vistas y controladores son generadas de manera automática, únicamente es necesario modificar los estilos si es necesario al igual que para las vistas. Por otra parte también establece otros métodos auxiliares, así como filtros, para facilitar la gestión de la aplicación y dotarla de más funcionalidades de una manera más sencilla y efectiva.

El modelo usuario ya ha sido descrito anteriormente, sin embargo, hay que destacar que **devise** incluye en el modelo la siguiente línea:

```
devise :database_authenticatable, :registerable,
:recoverable, :rememberable, :trackable, :validatable
```

Mediante esta sentencia de código se logra añadir ciertas especificaciones al modelo, de esta manera se dota al mismo de ciertas funciones con la finalidad de ofrecer una experiencia más sencilla y efectiva al usuario. A continuación se explica cada uno de los módulos añadidos.

1. **database-authenticatable**: los usuarios se autentican mediante un usuario y contraseña que se almacena en la base de datos.

2. **registerable**: cada uno de los usuarios tendrá la posibilidad de registrarse, eliminar y/o actualizar su perfil.

3. **recoverable**: un usuario puede solicitar recuperar su contraseña.

4. **rememberable**: habilitando el checkbox indicado, se recordará la información del usuario a la hora de iniciar sesión.

5. **trackable**: se habilita la función de seguimiento al usuario, es decir, desde donde se ha autenticado, la misma dirección IP, entre otras cosas.

6. **validatable**: valida el formato del correo y de la contraseña.

5.3.2. Descarga de clientes

Para gestionar la descarga los archivos que se generan al crear un cliente, se utiliza la gema *ActiveStorage*. Es una utilidad que simplifica el archivo de adjuntos y almacenamiento en aplicaciones web.

Las principales características y conceptos relacionados con Active Storage son:

- Adjuntos y modelos: Active Storage permite asociar archivos adjuntos a modelos en tu aplicación. Puedes declarar una asociación en el modelo para indicar que tiene uno o varios archivos adjuntos. Los archivos adjuntos se almacenan en la nube o en el sistema de archivos local.

- Variaciones y transformaciones: Active Storage te permite definir variaciones de un archivo adjunto, lo que te permite generar automáticamente versiones en diferentes tamaños o formatos. Por ejemplo, puedes tener una variación de imagen en miniatura y una variación de imagen de alta resolución. Active Storage utiliza la biblioteca ImageMagick para realizar transformaciones en imágenes.

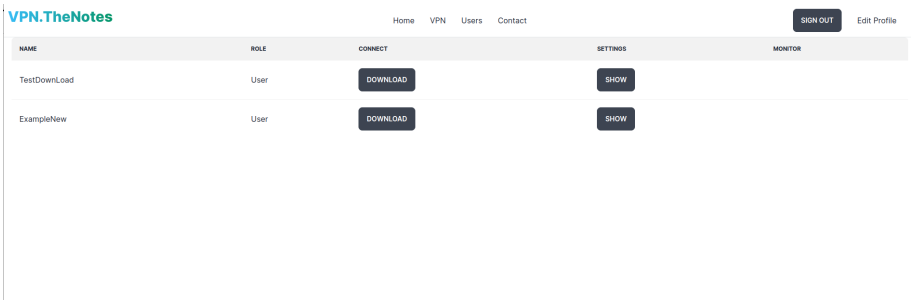
- Integración con Active Record: Active Storage se integra estrechamente con Active Record, el ORM (mapeo objeto-relacional) de Rails. Puedes utilizar métodos y consultas familiares de Active Record para trabajar con archivos adjuntos. Active Storage también se encarga de almacenar metadatos de archivos en la base de datos, como el nombre del archivo y el tipo MIME.

- Cargas directas: Active Storage admite la carga directa de archivos desde el navegador del usuario, lo que evita que los archivos pasen a través de tu servidor de aplicaciones. Esto mejora el rendimiento y la escalabilidad de la aplicación al reducir la carga en el servidor. Interfaz de programación sencilla: Active Storage proporciona una interfaz de programación sencilla y consistente para adjuntar, eliminar y recuperar archivos adjuntos.

En este proyecto se ha adjuntado al modelo *Client* el archivo que se genera tras la creación del mismo. De esta forma, en caso de eliminar un cliente, elimina también el archivo adjunto del sistema.

Con esto se logra gestionar los archivos localmente, integrándose con los modelos del sistema. Así se logra hacer cargas directas de archivos desde el navegador del usuario, evitando que los archivos pasen a través del servidor de aplicaciones, logrando mejorar el rendimiento y la escalabilidad de la aplicación.

Tras pulsar el botón de *Download*, se descarga el archivo correspondiente al cliente seleccionado. Como se muestra en la Figura 5.4.



,

Figura 5.4: Descarga de cliente.

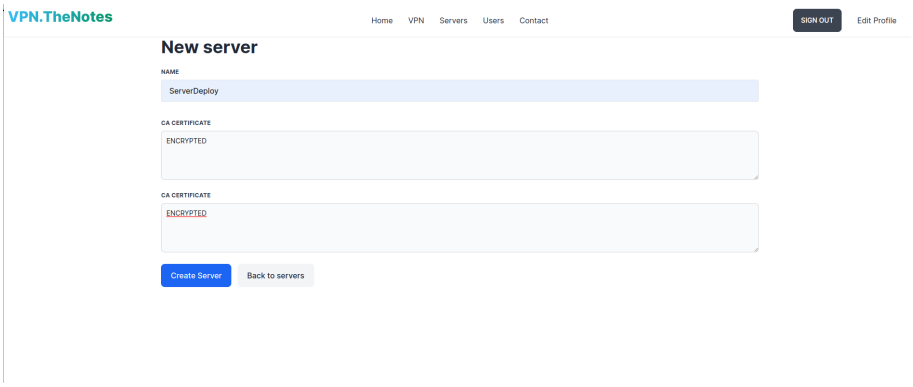
5.3.3. Gestión de VPN, servidores y clientes

A lo largo de este apartado se va a tratar como este proyecto realiza todas las operaciones referidas tanto a las VPN como a los mismos servidores. Para ello hay que explicar como trata OpenVPN a cada uno de los mismos. En OpenVPN un servidor actúa como un punto de conexión segura para los clientes que deseen conectarse a la red VPN. Para ello los clientes se autentican y el servidor les permite acceder a los recursos de la red. A su vez el servidor puede ofrecer otras herramientas, tales como limitar ancho de banda, controlar las políticas de acceso, asignación de direcciones IP, entre muchas otras. Por otra parte se encuentra el cliente, éste es un software utilizado para realizar la conexión segura a través de la VPN, se establece como el punto final de la conexión VPN y permite acceder a la red de manera segura y remota.

Tras haber descrito los componentes que se utilizan en este proyecto, a continuación se explica como se tratan en la misma. Cada usuario dispondrá de ciertas VPNs, éstas habrán sido creadas de tal manera que habrán incluido a los usuarios como miembros de la misma o habrán sido añadidos posteriormente por uno de los administradores de la red o por el administrador general. En la ventana principal aparecerá un listado de todos los clientes a los cuales puede acceder, podrá descargárselos, ver los ajustes y, en caso de que sea administrador, podrá monitorizar la VPN a la que esté vinculada el cliente. Por otro lado, en la ventana VPN existirá un listado de VPN de las cuales es administrador y, en caso de que el usuario sea administrador general, aparecerán todas las existentes en el sistema. Se podrá mostrar la VPN, editarla o incluso eliminarla. Además se podrán añadir clientes a cada VPN de las cuales el usuario sea administrador y, en caso de que sea administrador global, podrá añadir clientes a cualquier red.

En primer lugar se va a describir el proceso de creación de servidores. Se debe mencionar que para esta primera versión de la aplicación, únicamente se crea un servidor en el sistema. Éste hospedará todo el conjunto de VPN, esto es así ya que el hecho de añadir diferentes CA y diferenciar entre las mismas suponía un reto para la evolución del proyecto. Sin embargo se han creado las vistas pertinentes, así como todas sus funcionalidades, para que, en las próximas versiones, sea más sencillo implementar esta funcionalidad. Los servidores únicamente podrán ser creados por un usuario administrador, se crean rellenando un formulario que, como se ve en la Figura X.X, contiene los siguientes campos:

- Nombre: Un nombre descriptivo para el servidor.
- Certificado de la CA: almacena el certificado de la autoridad ceritficadora.
- Clave de la CA: almacena la clave privada empleada por la CA.



,

Figura 5.5: Formulario para la creación de un servidor.

A continuación se va a describir en detalle el proceso de creación de las VPN. Para poder crear una VPN, es necesario que el usuario haya sido registrado correctamente; en caso de que lo esté, se solicita al mismo que rellene el siguiente formulario, que se observa en la figura 5.5. Mediante este formulario el usuario podrá configurar los siguientes campos:

- Nombre: por conveniencia para el usuario, se establece un nombre para la VPN.
- Dirección IP: Se solicita la dirección IP pública, en la cual existirá el servidor, por defecto se rellena automáticamente este cmapo.
- Uso de IPv6: el usuario podrá elegir si quiere emplear IPv6[18].
- Puerto: se solicita el puerto que se usará para establecer las conexiones.
- Protocolo: se puede elegir el protocolo que va a usarse, pese a que en OpenVPN es recomendado usar el protocolo UDP, porque permite un transporte ligero de datagramas sin sobrecargar el tráfico con paquetes relacionados con la verificación de entrega y la retransmisión, además de que tiene una latencia menor en comparación con TCP, además que IDP en un entorno en el que pueden perderse algunos paquetes, UDP puede seguir enviándolos sin espera.
- DNS: las DNS pueden influir en el funcionamiento del servidor, como por ejemplo en la autenticación del servidor, ya que al configurar OpenVPN, los clientes deben conectarse a un servidor específico mediante la IP o nombre de dominio, si se usa éste último, las DNS se emplean para resolver ese nombre en la IP del servidor, permitiendo así a los clientes poder autenticarse correctamente en el servidor. El usuario también podrá elegir una DNS propia, en vez de emplear las opciones preestablecidas.
- Compresión: el objetivo a la hora de habilitar esta opción es reducir el tamaño de los datos antes de enviarlos a través de la VPN, pero esta opción no es recomendable ya que la compresión puede ocasionar un incremento en la carga de procesamiento del servidor, lo cual puede afectar al rendimiento del mismo. En el caso de que se habilite esta opción, podrá elegir entre lz4, lz4-v2 y lzo. LZ4 y LZO son algoritmos de compresión rápidos a la par que eficientes, mientras que LZ4-v2 propone una versión mejorada del algoritmo original y ofrece una posible mejor tasa de compresión.

- Encriptación: juega un papel crucial en lo que es la protección de la confidencialidad y autenticidad de los datos transmitidos a través de la conexión VPN. En el caso de que se habilite esta opción el usuario podrá elegir entre distintas opciones:

  - Tipo de cifrado: los algoritmos de cifrado permiten ofrecer una capa adicional de seguridad a la encriptación de la VPN, los algoritmos que se van a emplear serán los conocidos como **AES**, son un conjunto de algoritmos de cifrado simétrico que utilizan distintas longitudes de clave tanto para cifrar como para descifrar datos en bloques de 128 bits.
  - Certificado: la elección del certificado en la encriptación es importante para garantizar la autenticación de las entidades, el uso de algoritmos criptográficos adecuados o el intercambio seguro de claves. La elección entre ECDSA[19] o RSA[20] radicará entre la fortaleza criptográfica deseada, el rendimiento requerido y la compatibilidad con los dispositivos del sistema. En caso de elegirse ECDSA, el usuario habrá de elegir el tipo de curva que deberá aplicarse a la hora de generar la clave. Por otra parte, si se selecciona RSA se deberá seleccionar el tamaño con el cual se quiere generar la clave.

- Cifrado en el canal de control: el canal de control es un canal de comunicación separado y seguro que se establece entre cliente y servidor para facilitar el intercambio de información de control relacionada con la conexión VPN; el cifrado proporciona seguridad, integridad y protección contra ataques en la comunicación y en el intercambio de información crítica entre cliente y servidor.

- Clave Diffie-Hellman: Diffie-Hellman [21] es un algoritmo criptográfico que permite el intercambio seguro de claves entre dos partes sin revelarlas en un canal inseguro. En caso de elegir ECDH, deberá elegir el tipo de curva mediante la cual se cifrará. Por otra parte, si se eligiese el algoritmo DH, únicamente será necesario elegir el tamaño de la clave.

- Algoritmo de resumen criptográfico para HMAC: se emplea para generar códigos de autenticación y verificar la integridad de los datos que están siendo transmitidos. En caso de no tener requisitos específicos de seguridad, SHA-256 es la elección más recomendada, si se debe cumplir con ciertos estándares de seguridad, el usuario debería considerar emplear SHA-384 o SHA-512.

- Encriptación adicional: se puede proporcionar una protección extra para las comunicaciones dentro de un túnel VPN. La opción *tls-crypt* permite cifrar el tráfico de control. En vez de utilizar el intercambio de claves del protocolo TLS/SSL, dicho tráfico se cifra y se autentica por medio de una clave previamente compartida. Por otra parte, si se escoge la opción *tls-auth* también añade autenticación adicional a la conexión VPN utilizando una clave HMAC, ésta se calcula a partir de los paquetes de control intercambiados entre cliente y servidor. Esta opción ayuda a prevenir ataques de inundación o denegación de servicio, ya que únicamente los paquetes que han sido correctamente autenticados se procesan.

- Servidor: es necesario que la VPN se encuentre en un determinado servidor, por lo que en este campo se especifica en cual de todos los servidores existentes se quiere alojar.

- Tecnología: se especifica la tecnología empleada para crear la red, por el momento siempre será OpenVPN, por lo que este valor no será modificable, al menor por el momento.

- Usuarios: en este campo se seleccionan los usuarios que se quiere que tengan acceso a la red creada.

- Administradores: aquí se eligen los usuarios que se quiere que tengan permisos de administrador sobre la VPN, éstos serán los encargados de gestionar la red.

- Ancho de banda: en este atributo se puede limitar el ancho de banda que se empleará en la VPN.

En la figura 5.6 y 5.7 se tiene un ejemplo de formulario completado. Tras completar la creación, se generará el archivo de configuración de la misma red; éste será empleado por los clientes para generar sus archivos correspondientes para poder establecer la conexión pertinente a la red.

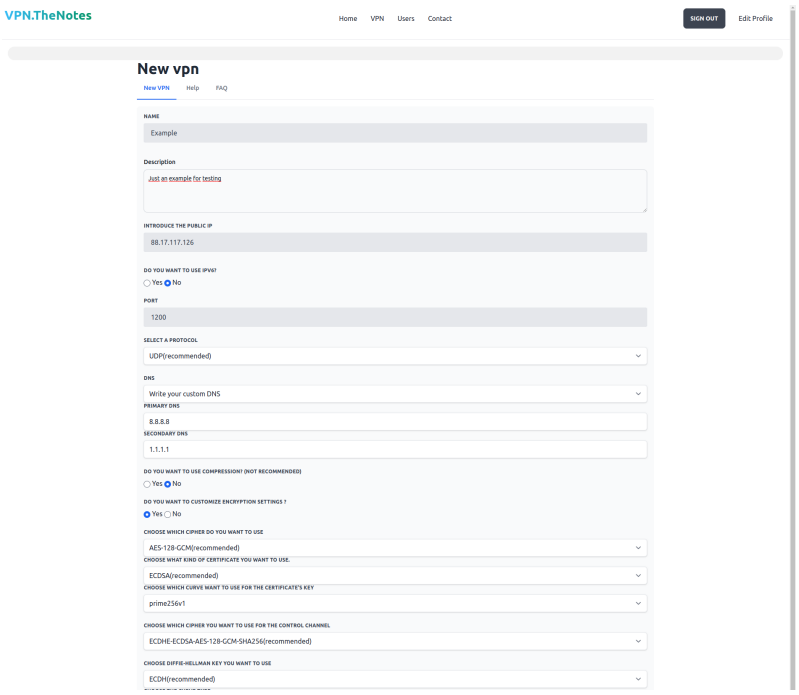


Figura 5.6: Formulario de creación de VPN(I).

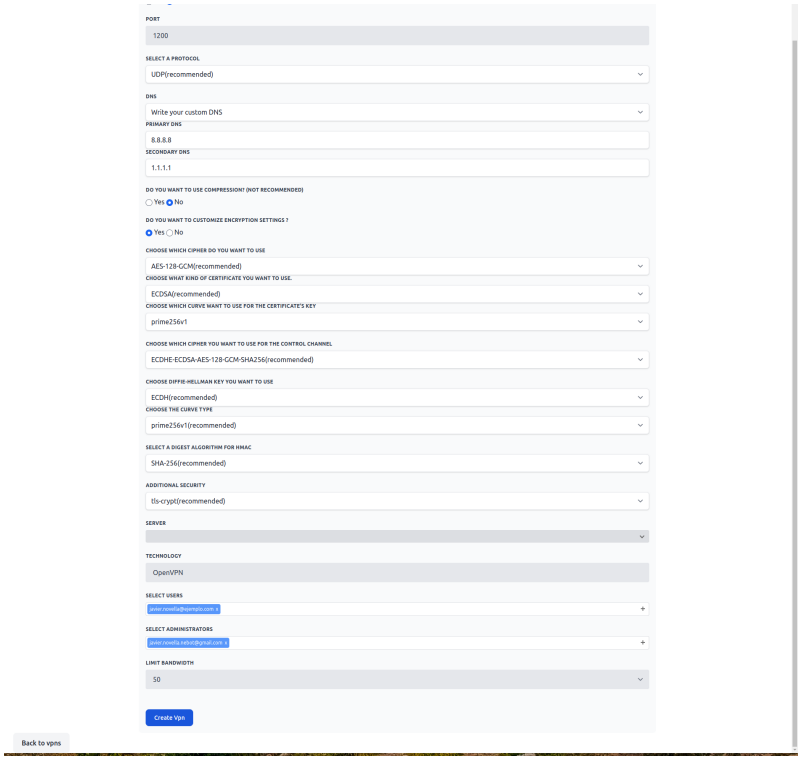


Figura 5.7: Formulario de creación de VPN(II).

Tras haber cumplimentado el formulario, se creará en la base de datos la red virtual privada y en el sistema se creará también con los parámetros previamente establecidos. Tras la creación de la misma, se ejecutará el código necesario para la creación de la VPN de OpenVPN en el sistema. Este código recogerá los parámetros introducidos en el formulario para su posterior creación, por otra parte se observa si el sistema tiene instalado OpenVPN, en caso de no tenerlo se instalará dependiendo del sistema operativo que se tenga. Además también se instala easy-rsa en caso de no haber sido instalado previamente.

Easy-RSA[22] es una herramienta de administración de certificados y claves utilizado en conjunto con OpenVPN. Por otra parte, también es un conjunto de scripts y utilidades que facilitan la generación y gestión de certificados y claves necesarios para asegurar las conexiones VPN establecidas con OpenVPN, todos estos son esenciales para autenticar y cifrar la comunicación entre los dispositivos que participan en la conexión VPN. Mediante Easy-RSA se puede generar un par de claves criptográficas, crear una autoridad de certificación y emitir certificados de cliente y servidor que se utilizarán en las conexiones VPN.

Prosiguiendo con la explicación del código de la creación de las VPN, tras la generación de claves con Easy-RSA, se establecerá el PKI y la CA y otros parámetros referentes al cifrado, en caso de que en el sistema no esté instalado OpenVPN, ya que si ya se ha instalado e inicializado estos parámetros, no es necesario volverlos a crear. Tras esto se generará el archivo de configuración de la VPN y se irán escribiendo las configuraciones para los clientes, así como el certificado o la clave del mismo servidor, entre otras cosas, en dicho archivo mediante los parámetros que han sido establecidos en el formulario. Por otra parte se reinicia y se activa el servidor de OpenVPN, se crean unos scripts para la adición o eliminación de reglas para el cortafuegos. Tras completar la configuración de la red, se puede instalar, en el caso de que se escoja una determinada opción para las DNS, *Unbound* [23] que es un servidor de nombres de dominio que responde a las consultas DNS realizadas por los clientes para traducir los nombres de dominio en direcciones IP;

este software es altamente configurable y flexible, lo que permite a los administradores de red personalizar su funcionamiento según sus necesidades específicas.

Cabe destacar que cada VPN se hospedará en un puerto único y diferente. Por otra parte, y como se mencionó anteriormente, se estructura por carpetas y dentro de cada carpeta que se creará por servidor generado, dentro de cada una existirán una carpeta por cada VPN creada, dentro estarán todos los archivos necesarios para la configuración del mismo y sus respectivos clientes.

Al verificar que la creación de la VPN se haya dado correctamente, se redirigirá al usuario a la ventana donde se muestran todas las VPNs creadas con anterioridad, como se muestra en la figura 5.8. A partir de aquí, la VPN está preparada para recibir sus respectivos clientes.

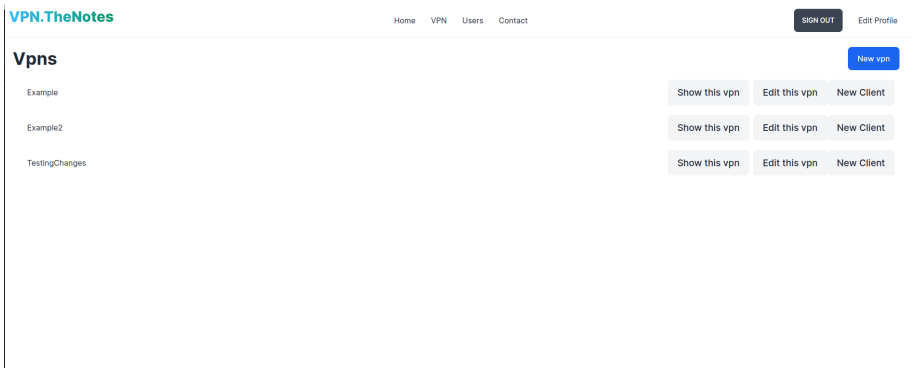


Figura 5.8: Vista de muestra de las VPN creadas.

Una vez tratada la generación de VPNs, se va a proceder con la creación de los clientes. Así como en el proceso anterior, para la creación de estas redes también será necesario rellenar un formulario, para ello se solicitarán los siguientes datos:

- Nombre: un nombre descriptivo de la red.
- Descripción: una breve descripción que explique el uso y la función que se espera dar a la red que se va a generar.
- IP privada: se asignará un rango de direcciones IP privadas a la red, estas direcciones se asignarán a los dispositivos que se conecten a la misma mediante los clientes generados.
- Puerto: la VPN se establecerá en un puerto determinado del sistema.

Al haber finalizado este proceso, se redirige al usuario a la ventana donde se muestran todas las VPN que tiene disponibles el usuario, como se ve en la figura 5.8 y, en caso de ser administrador, el usuario tendrá la posibilidad de generar nuevos clientes.

En la figura 5.8 se muestra la ventana con las VPN disponibles, en caso de que el usuario que haya iniciado sesión sea administrador de la VPN, podrá añadir clientes a la VPN seleccionada. Para poder crear un cliente, así como en los casos de servidor y VPN, el usuario deberá rellenar un breve formulario en el que se incluirá una mínima configuración. En el formulario se solicitan los siguientes campos:

- Nombre: se pide detallar un nombre descriptivo para el cliente.
- Descripción: una breve descripción del uso y funcionamiento esperado de esta red.

- Contraseña: si el usuario así lo desea podrá añadir una contraseña a este cliente, de esta manera se ofrecerá una seguridad adicional.

- VPN: en el formulario se incluirá el nombre de la VPN a la cual se conectará el cliente, el nombre se pondrá por defecto y el usuario no tendrá la posibilidad de modificarlo.

En la figura 5.9 se muestra un ejemplo de creación de un cliente. Para este ejemplo, se ha creado sobre la VPN *VPNExample*, un cliente con el nombre *VPNExampleClient* al cual se le ha asignado una contraseña y, por último, una descripción de la misma.

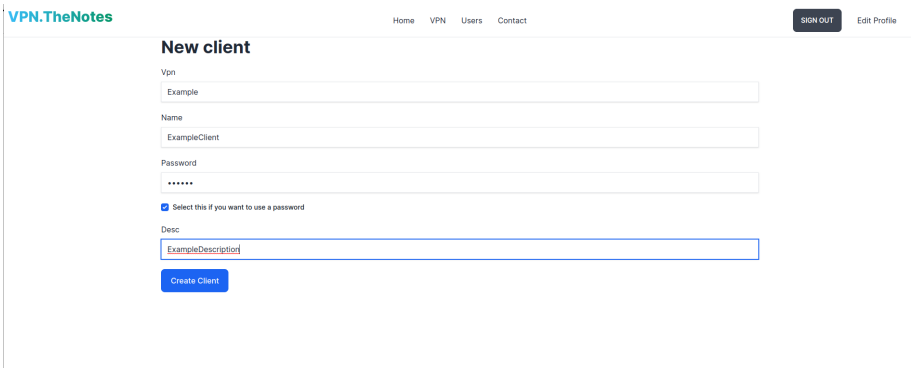


Figura 5.9: Formulario de creación de un cliente.

Al completar el formulario se creará el archivo ovpn del cliente, este archivo podrá descargarlo el usuario y, mediante éste, se podrá conectar el usuario a la red. Por otra parte, el usuario no será redirigido a la ventana con todos los clientes, en este caso se redirigirá a la ventana de las VPN para que el usuario pueda añadir más clientes directamente. Sin embargo el usuario también podrá acceder a la ventana de clientes y eliminar clientes desde ahí.

Es importante destacar la eliminación de elementos. En el caso de que se elimine un cliente, se revoca dicho cliente, se elimina de la base de datos y su respectivo archivo de configuración. En caso de eliminar una VPN, se elimina la carpeta correspondiente a la VPN, así como sus archivo pertinentes, se revocan y se eliminan los archivos asociados al cliente. Y , por último, en caso de eliminar un servidor, se revocan las credenciales del servidor y se eliminan tanto el servidor, como las VPN asociadas al mismo y los clientes vinculados a cada una de las VPN del servidor. Los scripts destinados a la eliminación de éstos se detallan en el glosario.

5.3.4. Gestión de usuarios

En esta sección se trata como la aplicación puede gestionar a los usuarios. La gestión de usuarios puede variar en difenres aspectos, los cuales son:

- Modificación del atributo de administrador

- Añadir o eliminar usuarios a una VPN

- Añadir o eliminar administradores a una VPN

- Eliminar usuarios del sistema

Modificación del atributo de administrador

Esta funcionalidad únicamente se dispondrá si el usuario dado de alta es administrador. El usuario dispondrá de dos tablas en las que se mostrará por una parte los usuarios administradores y por otro lado los que no lo son, como se muestra en la figura 5.10. Cada usuario será un botón y si se pulsa sobre él se cambiará su atributo administrador, en caso de que esté en la lista de administradores, si se pulsa sobre él, pasará a la otra lista y se modificará su campo administrador, lo mismo pasará en caso contrario.De esta manera los administradores pueden designar a otros usuarios como administradores. A su vez también pueden retirar permisos de administrador a otros usuarios.

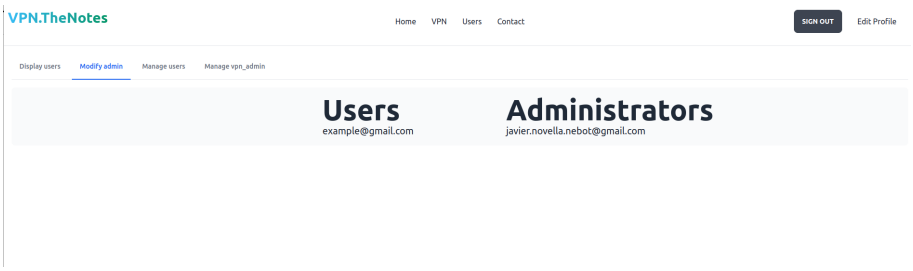


Figura 5.10: Modificación de administradores.

Añadir o eliminar usuarios a una VPN

Esta funcionalidad puede darse de dos formas, en caso de que el usuario sea administrador podrá añadir o eliminar usuarios a cualquiera de las VPN creadas en el sistema. Sin embargo, si el usuario no es administrador, únicamente podrá añadir o eliminar usuarios a las VPN de las cuales es administrador.

Esta herramienta funciona de la siguiente manera, se muestra una lista con todas las VPN disponibles que tenga el usuario. Por otra parte tendrá un selector donde podrá elegir a múltiples usuarios para añadirlos. En caso de querer eliminarlos, se muestran todos los usuarios asignados a esa red y se han de seleccionar los que se desean retirar de la misma. En la figura 5.11 se puede observar dicha funcionalidad.

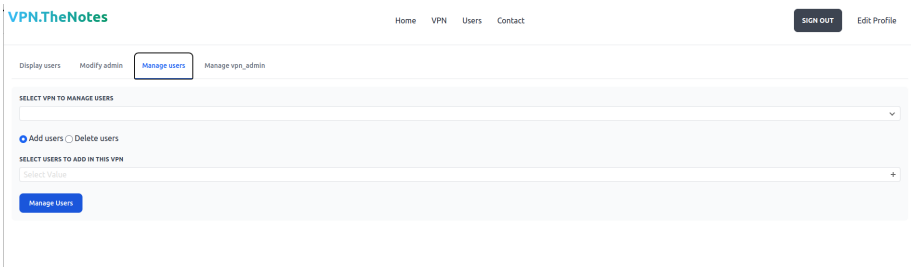


Figura 5.11: Adición o eliminación de usuarios a una VPN.

Añadir o eliminar administradores a una VPN

Esta herramienta trabaja de la misma manera que en el caso anterior, únicamente en vez de mostrar los usuarios de la VPN se mostrarán los administradores de la misma en el caso de querer eliminar. Por lo demás funciona exactamente igual que en el caso anterior y posee las mismas casuísticas.

Eliminar usuarios del sistema

Esta funcionalidad únicamente la pueden usar los usuarios administradores. Consiste en poder eliminar a usuarios registrados en el sistema, de esta manera si se desea retirar a un usuario del sistema se puede hacer de una manera más sencilla. Para ello se muestra una lista con todos los usuarios dentro del sistema, en esta lista el usuario tendrá las opciones de mostrar el usuario o editarlo. En caso de que el usuario presione el botón de mostrar la información, se permitirá al usuario mostrar la información o eliminar el usuario, como se muestra en la figura 5.12. Eliminar al usuario del sistema implica borrar las referencias que tenga este usuario en la tabla **users-vpns**.

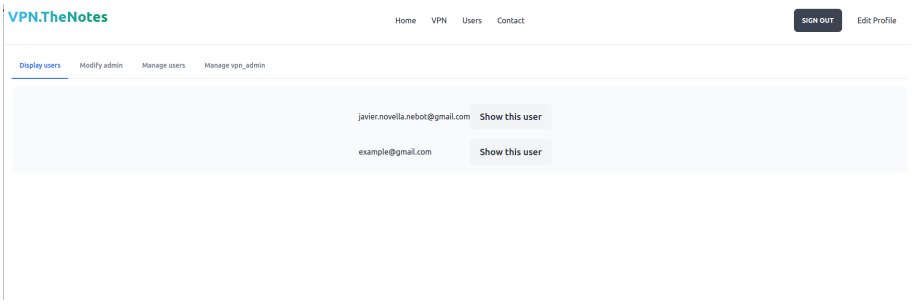


Figura 5.12: Eliminar usuarios del sistema

5.3.5. Ventana de contacto

En esta vista se proporciona al usuario la información necesaria para que pueda contactar con el equipo desarrollador, por otra parte se facilita un enlace al repositorio de GitHub donde se encontrará todo el código del proyecto. De esta manera cualquier usuario podrá consultar todo el código referente al desarrollo de este trabajo.

En caso de que el usuario tenga alguna duda o consulta, se facilita un correo de soporte, mediante el cual se ofrecerá ayuda al usuario para resolver cualquier problema que le haya podido surgir.

En la figura 5.13 se muestra la ventana de contacto, tal y como se ha descrito en este apartado.

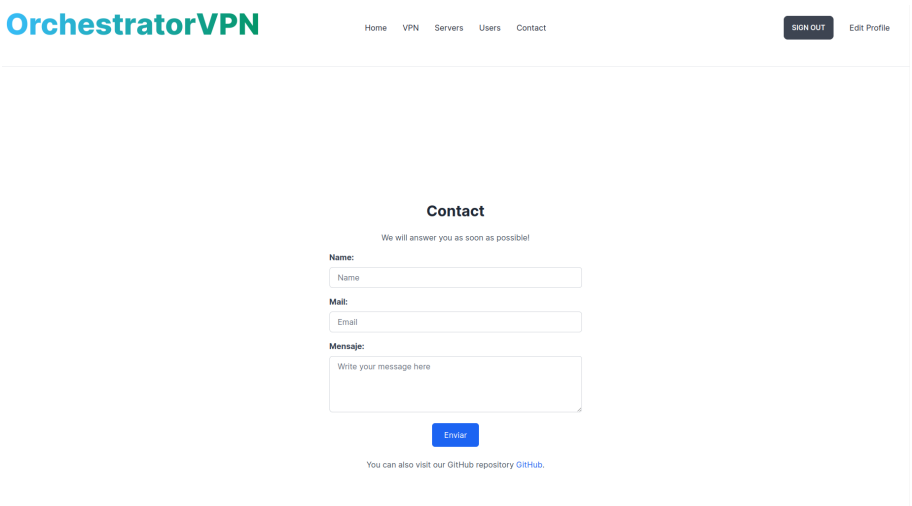


Figura 5.13: Ventana de contacto

5.3.6. Monitorización

Para poder ofrecer al usuario una monitorización eficiente y que pueda ser comprendida sin necesitar un nivel elevado en la comprensión del tráfico de redes, se emplean gráficas que muestren en tiempo real el ancho de banda que se esté empleando, así como la cantidad de tráfico que discurre por la red.

Obtención de los datos

Para poder recopilar los datos relacionados con el tráfico de red de la VPN, se va a establecer una sesión mediante Telnet al puerto en el que se encuentra la interfaz de gestión de la VPN. De esta forma se podrán obtener los datos relacionados con el tráfico de red. En un principio se van a obtener la cantidad de bytes de entrada y bytes de salida por cliente.

Telnet[24]es un protocolo de red utilizado para establecer una conexión remota a través de la línea de comandos. Es una herramienta de acceso remoto que permite interactuar con dispositivos y servidores utilizando comandos específicos.

Telnet puede utilizarse para recoger datos del tráfico de red para interactuar con dispositivos de red y recopilar información disponible a través de la interfaz de línea de comandos.

A continuación se muestra una lista de las múltiples características que ofrece este protocolo:

1. Acceso remoto: Telnet es una herramienta de acceso remoto que permite establecer una conexión a un servidor o dispositivo remoto a través de la red. Puedes acceder a sistemas operativos, servidores, dispositivos de red y otros equipos que admitan el protocolo Telnet.

2. Protocolo estándar: Telnet utiliza el protocolo Telnet estándar (definido en el RFC 854) para establecer y controlar sesiones remotas. Este protocolo especifica cómo se deben intercambiar los datos entre el cliente Telnet y el servidor Telnet.

3. Interfaz de línea de comandos: Telnet proporciona una interfaz basada en texto donde puedes ingresar comandos y recibir respuestas del servidor remoto. La comunicación se realiza a través de una sesión de texto sin formato, lo que significa que los datos enviados y recibidos son texto plano.

4. Puerto de conexión: Telnet utiliza el puerto 23 de manera predeterminada para establecer la conexión con el servidor Telnet remoto. Sin embargo, es posible que el puerto Telnet se haya modificado en el servidor para usar un número de puerto diferente.

5. Soporte de autenticación: Telnet admite diferentes métodos de autenticación para verificar la identidad del cliente y proporcionar acceso seguro al servidor remoto. Esto incluye la autenticación básica mediante nombre de usuario y contraseña, así como otros mecanismos de autenticación más seguros, como SSH (Secure Shell).

6. Comandos remotos: Telnet permite ejecutar comandos y enviar instrucciones al servidor remoto. Puedes controlar y administrar el servidor, realizar configuraciones, acceder a servicios y realizar operaciones específicas según las capacidades del servidor al que te conectes.

En resumen, Telnet es una herramienta de acceso remoto que permite establecer conexiones a servidores y dispositivos remotos a través de una interfaz de línea de comandos. Proporciona un mecanismo para enviar y recibir comandos e interactuar con el servidor remoto, aunque carece de características de seguridad avanzadas como la encriptación de datos.

En OpenVPN, la interfaz de gestión se refiere a un canal de comunicación establecido entre el servidor de OpenVPN y una interfaz de control remoto para administrar y supervisar la instancia de OpenVPN en tiempo real. A través de esta interfaz, es posible enviar comandos y recibir información sobre el estado de la VPN, las estadísticas de la conexión y realizar configuraciones dinámicas sin necesidad de reiniciar el servidor.

La interfaz de gestión utiliza el protocolo de comunicación de administración de OpenVPN (OpenVPN Management Interface Protocol) para la interacción entre el servidor y la interfaz de control remoto. A continuación se incluye una lista de características y funcionalidades que se pueden lograr mediante esta interfaz:

- Control y administración remota: La interfaz de gestión permite administrar y controlar el servidor OpenVPN desde un sistema remoto. Puedes enviar comandos para iniciar, detener o reiniciar la instancia de OpenVPN, así como realizar configuraciones o cambios en tiempo real.

- Monitoreo y estadísticas: Mediante la interfaz de gestión, es posible obtener información en tiempo real sobre el estado de la VPN y las estadísticas de conexión. Puedes supervisar el número de clientes conectados, el uso del ancho de banda, la carga del servidor, los registros de eventos y otros datos relevantes para el rendimiento y la operación de la VPN.

- Configuración dinámica: La interfaz de gestión permite realizar configuraciones dinámicas en el servidor OpenVPN sin necesidad de reiniciar el servicio. Puedes ajustar parámetros de configuración, agregar o eliminar rutas, habilitar o deshabilitar clientes, entre otras acciones, sin interrumpir la conectividad de los clientes existentes.

- Integración con herramientas de monitoreo: La interfaz de "management"puede ser utilizada para integrar OpenVPN con herramientas de monitoreo y administración de red existentes. Esto permite una gestión centralizada y automatizada de la VPN junto con otras infraestructuras y servicios de red.

Esta interfaz permite proporcionar un monitoreo de estadísticas a cerca del tráfico a nivel de red, es decir, se logra proporcionar los datos necesarios para generar los gráficos para controlar la congestión de red.

Generación de archivos .rrd y los gráficos pertinentes

Los datos obtenidos mediante la sesión de Telnet serán almacenados en archivos .rrd(Round Robin Database), estos archivos son una forma de almacenamiento de datos utilizada comúnmente para generar gráficos en tiempo real. Estos archivos esán diseñados específicamente para almacenar datos de series temporales, como es el tráfico de red en este caso, y se utilizan en conjunto con herramientas de generación de gráficos, que en este trabajo será *RRDtool*.

Se van a detallar las características principales de los archivos .rrd:

- Estructura de los archivos.rrd: Los archivos .rrd, se basan en una estructura de base de datos circular, es decir, los datos se almacenan en forma de rondas. Una ronda simboliza un intervalo de tiempo fijo y contiene una cantidad predefinida de puntos de datos. Los puntos de datos tienen un valor númerico que simboliza una métrica específica en ese intervalo.
- Almacenamiento de datos: A medida que se van recopilando los datos de tráfico de red, se agregan al archivo .rrd definido en función del intervalo que se haya establecido previamente. Conforme el tiempo vaya avanzando, los datos más antiguos se descartan y automáticamente para que se inserten los nuevos datos. De esta forma se consigue que el tamaño del archivo se mantenga constante a lo largo del tiempo.
- Consolidación de datos: Los archivos .rrd también realizan una consolidadción de datos para reducir la cantidad de puntos almacenados. Según se acumulan los datos en cada ronda, se pueden generar puntos de consolidación que resuman una cantidad de puntos de datos en uno solo. Así se logra un almacenamiento más eficiente y a la hora de representar la gráfica se observará de una manera más clara.

Para resumir lo explicado, los archivos .rrd proporcionan un almacenamiento eficiente y una estructura idónea para monitorizar el tráfico de red a lo largo del tiempo. Se muestra en la figura 5.14 el funcionamiento de los archivos RRD.

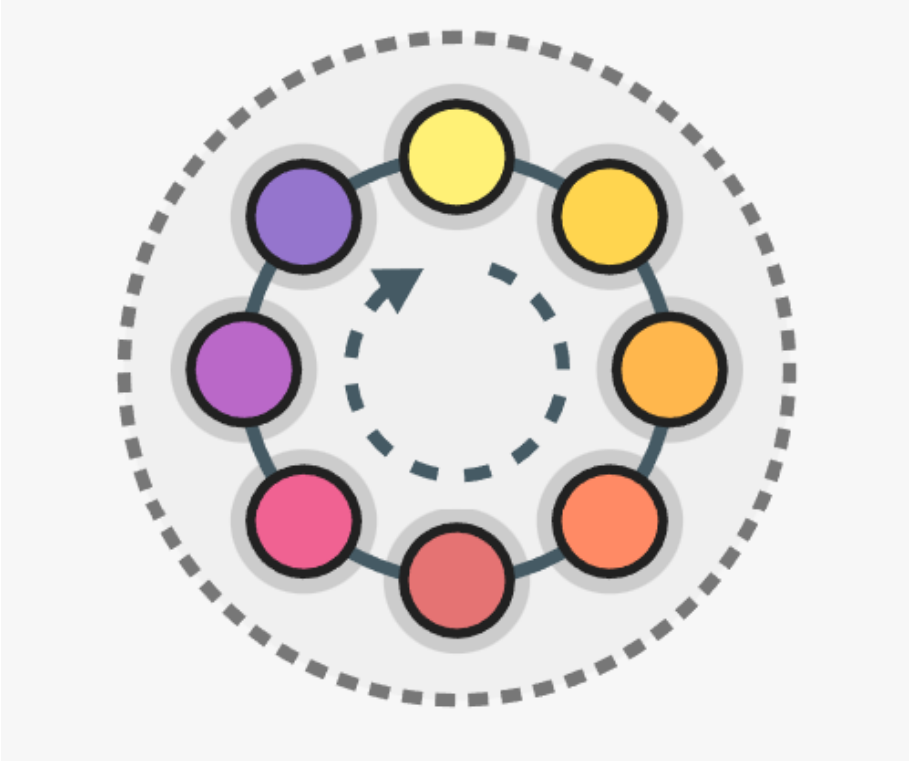


Figura 5.14: Funcionamiento de los archivos RRD. Fuente(Aggregate-digital, 2023)

Para extraer los datos de la interfaz de gestión de OpenVPN y, posteriormente, insertarlos en archivos .rrd, se hace mediante la línea de comandos y utilizando la herramienta *RRDtool*. Para poder integrar esta funcionalidad en el poryecto, se ha incluido la gema *RRD-ffi*.

Esta gena proporciona una interfaz para trabajar con archivos RRD utilizando la biblioteca librrd de C. Esta gema se basa en otra que permite la interoperabilidad entre Ruby y bibliotecas escritas en C. Utiliza enlaces FFI (Foreign Function Interface), para

llamar a las funciones de la biblioteca librrd desde el proyecto Rails. La funcionalidad principal de la misma es permitir la creación, actualización y consulta de bases de datos RRD. Adicionalmente, proporciona métodos para crear bases de datos RRD, agregar datos a ellas y recuperar los datos almacenados. Estos métodos permiten especificar la estructura de datos, la frecuencia de muestreo, el tipo de datos, entre muchas otras opciones. Y también existen otros métodos destinados a la generación de gráficos a partir de los datos almacenados.

Se ha definido la interfaz de rrd, la cual tendrá los siguientes métodos:

- Inicializador: Sirve para guardar como variables locales la vpn sobre la cual se quiere realizar la monitorización, el archivo RRD de la misma y el contenido de la base de datos RRD.

- Actualización: En este método se indica que se cree el archivo RRD de la VPN en caso de que no exista, si ya existe debe actualizar los argumentos de la misma periódicamente.

- Gráficos: Este método está enfocado en la creación de los gráficos. Se crean dos gráficos uno para indicar el tráfico que se da dentro de la VPN y, por otro lado, existe otro gráfico que muestra los clientes existentes.

- Creación: Este método realiza la función de crear la base de datos RRD e ir periódicamente actualizando el valor de los datos.

En el anexo correspondiente se incluye el código en su totalidad.

Para la obtención de los datos a tiempo real, como se ha comentado anteriormente, se hará a través de la interfaz de gestión de las VPN en OpenVPN, para ello se establecerá una conexión por Telnet con el servidor donde se tienen ubicados los archivos, una vez allí se irán ejecutando los comandos que se han descrito en la clase *openvpnmanager*, logrando así extraer correctamente los datos requeridos e incluso pudiendo parar los procesos de las VPN en el servidor.

Los gráficos se almacenan en el proyecto de Rails y estos se adjuntan a la ventana dedicada al despliegue de monitorización. En las figuras 5.14 y 5.15, se muestra un ejemplo de los gráficos generados, al monitorizar el tráfico de una red.

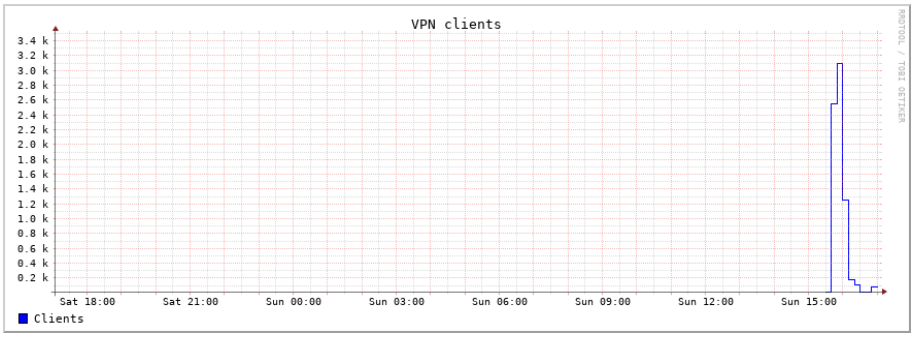


Figura 5.15: Gráfico de clientes

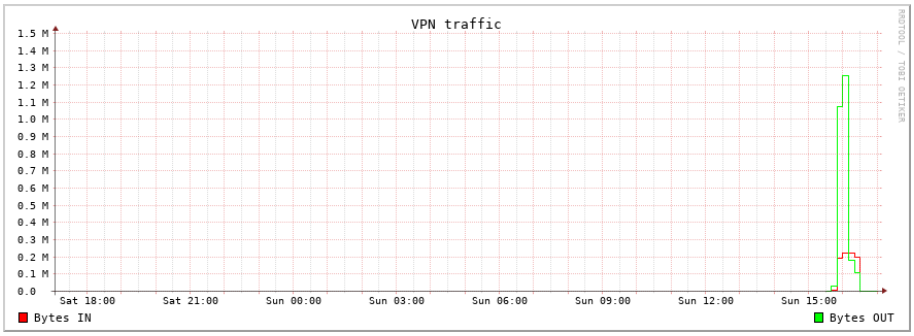


Figura 5.16: Gráfico de tráfico

CAPÍTULO 6
Despliegue

Tras haber logrado desarrollar todo el proyecto con éxito, se ha decidido desplegar la aplicación en un dominio web para facilitar el uso al usuario. Cabe destacar que, como se ha comentado anteriormente, todo el código estará alojado en un repositorio remoto en GitHub.

6.1 Provisión del servidor

En primer lugar, se comenta la provisión del servidor sobre el cual se ejecutará la aplicación web. El servidor es una máquina física HP Z6 G4, las características de la misma son descritas en el anexo, esta computadora tiene instalada la distribución *Linux Ubuntu Server 22.04* y se mantiene parcheada aplicando un decalaje semanal, de esta manera se logra evitar regresiones inducidas por actualizaciones no suficientemente verificadas.

El servidor con IP privada 10.110.0.11/21 se conecta a Internet tras una protección de doble NAT[25]. La protección mediante doble NAT o doble enrutamiento es un proceso en el cual se aplica la técnica NAT; consiste en que múltiples dispositivos en una red privada pueden compartir una única dirección IP, a ésta se le agrega una capa adicional de traducción de direcciones; es decir, los paquetes de datos se traducen dos veces a medida que atraviesan las puertas de enlace de red. Gracias al uso de este procedimiento, se proporciona una capa adicional de seguridad. La protección se realiza mediante un router HGU; combina las funciones de un router y una puerta de enlace de un único dispositivo, en una LAN con dirección 192.168.1.1/24 y en una WAN con la dirección IP pública dinámica.

En la IP 192.168.1.1, se corresponde con la dirección del router de fibra, se abren los puertos 80 (HTTP), 443 (HTTPS). Por otra parte se ha empleado el puerto 2222. Por último, se han abierto, por el momento, los puertos desde 1194 hasta el 1999 para poder crear las VPN, se abren desde el puerto 1194 porque es el que escoge OpenVPN por defecto para la creación de VPN. Todos estos puertos se redirigen a la dirección 192.168.1.30 en el que se encuentra un cortafuegos. En la dirección del cortafuegos se redirigen a la dirección 10.110.0.11 los puertos 80, 443 y desde el 1194 al 1999. Además el puerto 2222 se cambia al 22, el cual maneja nativamente el protocolo SSH del servidor.

En este servidor se mantendrá el código de la plataforma web en constante ejecución, también es donde se subirán los cambios para ofrecer mejoras y, de esta manera, ofrecer una mejor experiencia al usuario. Esta máquina será la encargada de almacenar los archivos que se generen al crear los servidores, tales como los archivos de configuración o los archivos **.ovpn**. Además será encargado de ejecutar todos los procesos que solicite el cliente, como la creación de servidores, VPN o clientes, entre otras muchas cosas.

6.2 DNS dinámico

Para poder tener el nombre de host *vpn.thenotes.co* siempre disponible sobre una conexión pública de Internet con dirección IP dinámica se escoge la opción de utilizar el servicio DNS del registrador de dominio; ya que es la opción más sencilla, frente a la creación y mantenimiento de toda una infraestructura propia de DNS. Para poder completar este proceso se ha tenido que crear un registro A+ de host en la configuración DNS del proveedor de dominios y, por otra parte, se debe instalar y configurar en el servidor un programa que se encargue de actualizar la dirección IP y la actualice en el DNS del proveedor en caso de cambio.

6.2.1. Creación de registro de host

Un registro DNS es una entrada en la configuración del servidor DNS que asocia un nombre de dominio con una dirección IP específica. El registro A+ [26] de host es una extensión del registro A estándar de DNS, éste se utiliza para asociar un nombre de dominio con una dirección IP; mientras que el registro A+ permite especificar de forma adicional una puntuación o calificación para el dominio.

Seguidamente se presenta el proceso desglosado para crear un registro A+ de host en la configuración DNS del proveedor de dominios:

1. Acceso al proveedor de dominios: Se debe acceder al panel de control del proveedor de dominios.

2. Ubicación de la sección de configuración de DNS: se debe buscar el apartado relacionado con la configuración del DNS.

3. Selección del dominio adecuado: si ya se tienen varios dominios registrados, se debe seleccionar el dominio deseado para el cual se quiere crear el registro A+ de host.

4. Creación de registro A+: dentro de la sección de configuración de DNS, se debe buscar la opción referente a la agregación de un nuevo registro.

5. Selección del tipo de registro: tras iniciar el proceso de creación del registro, se solicita escoger el tipo del mismo. Debe escoger el tipo A+ en este caso.

6. : Introducir la información necesaria: se solicita que el usuario inserte la información necesaria para el registro A+, como el nombre del host o subdominio al que se desea asignar el registro, además de la IP asociado, además de cualquier otra información relevante.

7. Guardar los cambios: tras completar el registro A+, debe guardar los cambios.

8. Propagación del DNS: después de guardar los cambios, el proveedor de dominios propaga la configuración DNS actualizada.

Una vez completada la propagación, el registro A+ de host se encontrará activo y se podrá utilizar para asociar el nombre de dominio, en este caso *vpn.thenotes.co*, con la dirección IP especificada.

6.2.2. Configuración de un cliente dinámico de DNS

Un cliente dinámico de DNS consiste en un software empleado para mantener actualizada la dirección IP asignada al nombre de dominio. En este caso es útil porque como la dirección IP puede cambiar con cierta frecuencia, se necesita verificar que el dominio esté continuamente vinculado con la dirección IP correcta. Para este proyecto se ha escogido el programa **Ddclient** [27]. A continuación se explica el proceso de instalación y configuración paso a paso:

1. Descarga e instalación de *ddclient*: se instala en el servidor empleado la última versión del software escogido.

2. Configuración del programa: se debe configurar *ddclient* para que pueda actualizar automáticamente la dirección IP en el DNS del proveedor. En la figura 6.1 se muestra la configuración establecida.

```
vpn@svvlclx02:/etc$ sudo cat ddclient.conf
[sudo] password for vpn:
# Configuration file for ddclient generated by debconf
#
# /etc/ddclient.conf

use=web, web=dynamicdns.park-your-domain.com/getip
protocol=namecheap
server=dynamicdns.park-your-domain.com
login=thenotes.co
password=[redacted]
vpn
```

Figura 6.1: Archivo de configuración del cliente dinámico de DNS

3. Reiniciar el servicio *ddclient*: tras guardar la configuración, se debe reiniciar el servicio para aplicar los cambios.

Tras completar los pasos, el cliente dinámico de DNS comenzará a comprobar la dirección IP pública cada cinco minutos y actualiza automáticamente el DNS del proveedor en caso de que haya un cambio. De esta manera se verifica que el dominio siempre estará vinculado con la dirección IP correcta.

6.3 Ecosistema de ejecución de Rails

Previamente a lanzar la plataforma, es necesario tener configurado todo el entorno de ejecución de *Ruby on Rails*. Para ello se debe instalar el gestor de versiones de ruby(RVM) en modo multiusuario. RVM es una herramienta que permite instalar y controlar múltiples versiones de Ruby en el sistema y el modo multiusuario permite que varios usuarios en el servidor compartan las mismas instalaciones de Ruby. Una vez instalado el gestor de versiones será necesario instalar una versión de Ruby, se ha escogido la versión 3.2.2, que es la última liberada hasta la fecha.

Una vez verificada toda la instalación previa, se debe comprobar que las utilidades *gem* y *bundle*, que controlan las gemas de la aplicación que se ha desarrollado, están instaladas en el sistema y actualizadas a su última version. Tras completar estos pasos, se tiene el ecosistema de Ruby listo para la ejecución.

6.4 Servidor web

Esta aplicación correrá en un servidor web que se va a definir en los siguientes subapartados.

6.4.1. Instalación de NGINX

Como servidor web se ha elegido NGINX[28], es un servidor web de alto rendimiento. Fue diseñado para manejar cargas elevadas de tráfico de manera eficiente. Cabe destacar que funciona como servidor proxy inverso, en el contexto de NGINX, quiere decir que es una configuración donde este servidor web actúa como intermediario entre los clientes y los servidores de destino. Cuando un cliente envía una solicitud a NGINX, este actúa como un proxy y reenvía la solicitud al servidor de destino correspondiente. A continuación, recibe la respuesta del servidor de destino y la reenvía al cliente. La principal ventaja de usar un proxy inverso como NGINX que beneficie al proyecto es que puede mejorar el rendimiento y la escabilidad de los servidores de destino. Al actuar como un proxy, el servidor web puede distribuir la carga de manera equitativa entre los servidores disponibles, evitando asía la sobrecarga en un único servidor. Además, NGINX puede realizar funciones de caché, compresión y balanceo de carga.

6.4.2. Configuración de los *server blocks*

En NGINX, los *server blocks* son configuraciones que permiten detallar múltiples sitios web dentro de un mismo servidor NGINX. Se ha de tener en cuenta que cada uno contiene la configuración específica para un dominio o una aplicación en particular.

Dentro de un *server block* se puede especificar el nombre de dominio a los que debe responder, la ubicación de los archivos del sitio web, las reglas de enrutamiento de las solicitudes, entre otras.

Para poder configurar el propio *server block* se debe acceder al directorio de configuración de NGINX:

```
cd /etc/nginx/sites-available
```

A continuación se debe crear un archivo de configuración para el dominio deseado, que en este caso será *vpn.thenotes.co*. Dentro del mismo se debe proporcionar la configuración necesaria, como la ubicación del servidor, el puerto y cualquier otra configuración específica.

6.4.3. Obtención de un certificado SSL válido

Un certificado SSL es un archivo digital utilizado para establecer una conexión segura y cifrada entre un servidor web y un navegador o cliente. El propósito principal de un certificado SSL es garantizar la privacidad, la integridad y la autenticidad de la información transmitida entre el servidor y el cliente. El certificado SSL permite la encriptación de los datos para que no puedan ser leídos o manipulados por personas no autorizadas. Cuando se establece una conexión segura mediante SSL, se realiza un proceso de *handshake* entre el servidor y el cliente para verificar la autenticidad del servidor y negociar una clave de cifrado compartida.

Para la obtención de un certificado SSL[29], se ha empleado Certbot. Certbot es una herramienta que facilita la obtención y renovación automatizada de certificados SSL gra-

tuitos emitidos por la autoridad de certificación *Let's Encrypt*. Esta herramienta automatiza gran parte del proceso de adquisición y gestión de certificados, además ofrece funciones de renovación automática; programando tareas de renovación y encargándose de otros detalles más tediosos. Adicionalmente configura el servidor web para utilizar los nuevos certificados, actualizando los archivos de configuración necesarios. De esta manera se asegura que el servidor esté configurado correctamente para utilizar el cifrado SSL y proporcionar conexiones seguras a los visitantes del sitio web.

En primer lugar se debe instalar en el servidor la herramienta **certbot**. A continuación se debe introducir el siguiente comando para poder obtener y configurar el certificado SSL:

```
sudo certbot certonly --nginx -d vpn.thenotes.co
```

Tras la ejecución del comando anterior, se debe seguir las instrucciones proporcionadas por *certbot* para poder completar el proceso de obtención del certificado.

6.4.4. Redirección del tráfico HTTP a HTTPS

Se recomienda redirigir el tráfico HTTP a HTTPS porque así se logra proporcionar seguridad al cifrar la comunicación, se protege la integridad de los datos transmitidos, autentica el servidor, genera confianza en los usuarios y mejora la posición en los motores de búsqueda. En resumen, garantiza una experiencia segura y confiable en la web.

Para ello se debe abrir el *server block* específico del dominio y añadir la siguiente configuración para redirigir todo el tráfico HTTP a HTTPS:

```
server {
    if ($host = vpn.thenotes.co) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    listen [::]:80;

    server_name vpn.thenotes.co;
    return 404; # managed by Certbot
}
```

6.4.5. Definir la ubicación del servidor

Se ha creado un *upstream* para el tráfico hacia el servidor Puma en el puerto 5000. Un *upstream* en NGINX es utilizado para definir la ubicación del servidor Puma de producción al cual se debe redirigir el tráfico. Para ello se debe agregar el siguiente esquema en el archivo de configuración del dominio, así como se ha hecho en el paso anterior:

```
upstream puma {
  server 0.0.0.0:5000;
}
```

6.4.6. Configuración de las excepciones

Para mejorar el rendimiento se ha configurado de tal forma que NGINX sirva los activos estáticos(imágenes, hojas de estilo, ...) directamente, es decir, sin tener que pasar por el servidor Puma. Para ello, como en los dos pasos anteriores, se debe abrir el archivo de configuración específico de nuestro dominio y añadir las siguientes líneas:

```
location  ~* ^/assets {
    root /var/www/VPN-Orchestrator/public;
    expires 1y;
    add_header Cache-Control public;
    add_header Last-Modified "";
    add_header ETag "";
    break;
}
```

Una vez finalizados estos pasos, se ha configurado NGINX para que funcione como proxy inverso del servidor Puma, que es el que utiliza el framework de Rails, se ha creado el *server block* para el dominio, se ha obtenido un certificado SSL válido el cual se revalidará automáticamente, se ha redirigido el tráfico HTTP a HTTPS, se ha configurado un *upstream* hacia Puma y, por último, se ha establecido una excepción para hacer servir activos directamente desde NGINX.

6.5 CI/CD

CI/CD[30] significa Integración Continua/Implementación Continua, es una metodología en el desarrollo de software que se centra en la automatización y mejora continua de la eficiencia en la entrega de aplicaciones.

La Integración Continua se refiere al proceso de combinar el código de diferentes usuarios en un repositorio centralizado de forma frecuente y automatiza. En este caso, solo ha habido un único desarrollador, sin embargo se ha hecho de esta manera por si en un futuro el proyecto escala a envergaduras mayores y, en ese caso, sea necesario disponer de más desarrolladores. El objetivo principal de la integración continua es la detección y resolución de problemas con la mayor rapidez posible. En vez de esperar hasta el final del ciclo de desarrollo para combinar el código, los cambios se integran regularmente y se prueban de manera automática.

La Implementación Continua se enfoca en automatizar el despliegue de aplicaciones a través de diferentes etapas de desarrollo, pruebas y producción. En vez de tener un proceso manual que puede llevar a errores, la implementación continua utiliza scripts y herramientas para desplegar la aplicación de manera consistente y repetible. Esto permite a los equipos de desarrollo entregar cambios de manera rápida y segura a los usuarios finales.

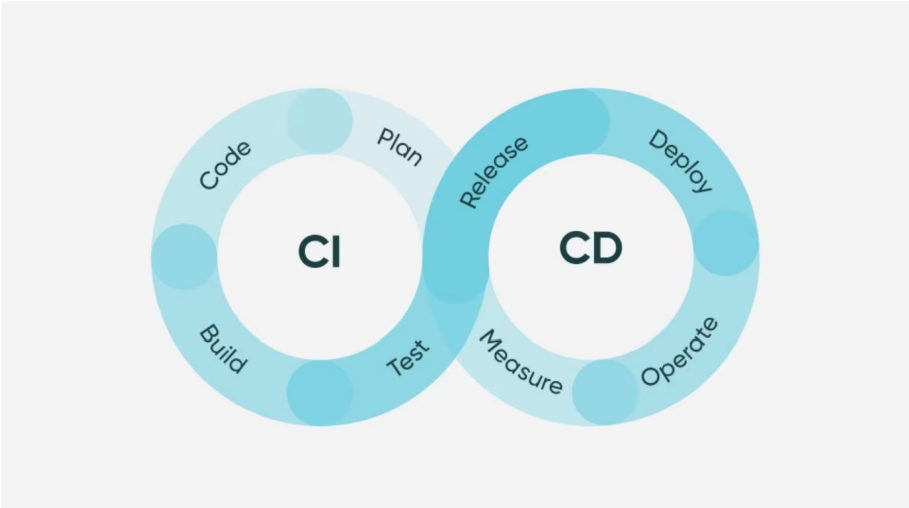


Figura 6.2: Ciclo de desarrollo de software con CI/CD. Fuente:(ServiceNow, 2021)

Para ejecutarlo sobre este proyecto, en primer lugar se ha creado un *GitHub Actions* que se ejecuta automáticamente cada vez que se suba el código al repositorio.

A continuación, la acción ejecuta las pruebas de **RSpec**. RSpec es un framework de pruebas para Ruby, dichas pruebas permiten verificar que el código funcione correctamente y cumple con los requisitos que se han propuesto. Al ejecutar las pruebas de forma automática y regular, puede ayudar a identificar rápidamente cualquier error o problema que se haya introducido en el código.

Seguidamente se realiza el *linting* mediante la herramienta Rubocop. El *linting* es un proceso utilizado en la programación para detectar y corregir errores, incumplimientos de estilo y posibles problemas en el código fuente. Consiste en utilizar una herramienta de análisis, en este caso Rubocop, para examinar el código y buscar patrones que puedan ser problemáticos o que no cumplan con las convenciones preestablecidas.

Además de las pruebas y el proceso de análisis anterior, la acción también realiza otras tareas de seguridad. Se utiliza *bundler-audit* para realizar una auditoría de las dependencias utilizadas en el proyecto. De esta manera se pueden identificar posibles vulnerabilidades conocidas en las dependencias y poder mitigar los riesgos detectados.

También se usa *brakeman* para identificar vulnerabilidades de seguridad específicas en el código Ruby. Brakeman es una herramienta de análisis estático que trata de detectar vulnerabilidades, tales como la inyección de código. Al ejecutar esta herramienta continuamente, se pueden identificar vulnerabilidades.

En cuanto al despliegue continuo, se va a utilizar la herramienta *Capistrano* para desplegar en el servidor **vpn.thenotes.co**. Capistrano es una herramienta de implementación en Ruby que permite automatizar y simplicar el proceso de despliegue de aplicaciones en servidores remotos. Con Capistrano se puede definir tareas específicas de implementación y configurar el flujo que es necesario para desplegar la aplicación en el servidor objetivo.

CAPÍTULO 7
Pruebas

Para abordar la problemática de las pruebas, se va a emplear un DSL interno de Ruby. Un DSL es un lenguaje de programación diseñado para abordar problemas en un dominio o contexto particular, es decir, se enfocan en resolver un conjunto limitado de retos relacionados con un dominio en concreto.

Las pruebas de este código se ejecutan en el entorno de *RSpec*, un DSL (Domain Specific Language) diseñado en Ruby para las pruebas de código, y basado en el BDD (Desarrollo Guiado por el Comportamiento) y con soporte para el TDD (Desarrollo Guiado por las Pruebas). Destaca por su expresividad y su habilidad para el *mocking* y el *stubbing* [31]. El término *mocking* hace referencia a una técnica empleada en el desarrollo de software para simular el comportamiento de objetos o componentes de software en pruebas unitarias. El uso de *mocks* en estas pruebas permite aislar el fragmento de código que se está probando y de esta forma se permite identificar los problemas más fácilmente. Por otra parte, el concepto *stubbing* describe una técnica aplicada en las pruebas de software para simular el comportamiento de métodos o funciones específicas en obketos o componentes. Los *stubs* son objetos falsos que implementan la misma interfaz que el objeto real y proporcionan resultados simulados predefinidos. De esta forma, permite probar unidades de código de forma aislada y controlada, evitando la dependencia de recursos externos.

También se puede resaltar su capacidad para poder integrar bajo el mismo paraguas los tests unitarios, funcionales y los de integración. Esto es así porque RSpec organiza las pruebas en una estructura jerárquica basada en bloques de código anidados. Esto permite agrupar y organizar las pruebas de manera lógica, lo que facilita la navegación y compresión del conjunto completo de pruebas. Además, esta estructura jerárquica permite escribir pruebas a diferentes niveles, como pruebas unitarias, funcionales y de integración, dentro del mismo archivo.

En RSpec, los tests en su forma más simple se escriben usando los métodos *describe* e *it*. El primero se utiliza para agrupar pruebas relacionadas entre sí y dotar de contexto a lo que se está probando. El siguiente método se emplea para definir las pruebas individuales, que son llamadas ejemplos en este entorno. Típicamente se anidan estos métodos con un bloque externo *describe* que visualmente dota de contexto a uno o varios bloques *it* internos a él.

Por ejemplo, para describir una clase y sus métodos, se puede usar la sintaxis siguiente:

```
describe Vpn do
    describe "conecta" do
        it "realiza el handshake" do
```

```
            # pruebas de código
        end
        it "comprueba certificados" do
            # otras pruebas de código
        end
    end
end
```

El método *describe* puede tomar un segundo argumento, una clase o método que es el objeto de las pruebas. Esto puede resultar útil para especificar el espacio de nombres si se hace *mixins*, que consiste en la reutilización flexible de código al incluir métodos y funcionalidades de uno o varios *mixins* en una clase, desde un módulo externo:

```
describe MiModulo, :type => :module do
    it "valida el usuario con MFA" do
        # pruebas de código
    end
end
```

El método *it*, como se puede apreciar en el código anterior, puede tomar también un argumento opcional que describe lo que el test intenta comprobar. Esto es útil para proporcionar una información adicional acerca del test y otorga un resultado de salida más descriptivo.

```
describe VPN do
    it "tiene un rango IP válido" do
        # pruebas para comprobar la validez del rango IP
    end
end
```

También se puede usar el método *before*, el cual ejecuta un bloque de código antes de la ejecución de cada ejemplo en un grupo. Se utiliza regularmente para hacer tareas de preparación para el test.

```
describe Vpn do
    before do
        @objeto_vpn = Vpn.new
    end
    it "la VPN tiene nombre" do
        expect(@objecto_vpn.name).to eq("VPN super secreta")
    end
end
```

El método *let* se utiliza para definir un valor que será memorizado y estará disponible para todos los ejemplos de un grupo.

```
describe Vpn do
    let(:mi_vpn) { Vpn.new }

    it "la VPN tiene un nombre" do
        expect(mi_vpn.name).to eq("VPN super secreta")
    end
end
```

Una vez explicada la sintaxis más común de RSpec, se procede a describir las pruebas desarrolladas para este proyecto. Para poder generar las pruebas mediante RSpec, se ha importado la gema, a continuación en la carpeta spec, se ha incluido dentro de cada subdirectorio una clase que cubre las pruebas de cada uno de los modelos generados, además de los códigos de creación de VPN y clientes y otros aspectos no tan relevantes.

Se va a mostrar la clase creada para las peticiones referentes al modelo clientes. Para ello se han creado diferentes métodos para validar los métodos de este controlador, tales como el siguiente:

```
    describe "GET /index" do
  it "renders a successful response" do
    Client.create! valid_attributes
    get clients_url
    expect(response).to be_successful
  end
end
```

Este método comprueba si el método *index* del controlador de clientes, funciona correctamente. Para ello crea una instancia de cliente, accede a la dirección especificada de los clientes. Por último comprueba si la respuesta ha sido tal y como se esperaba.

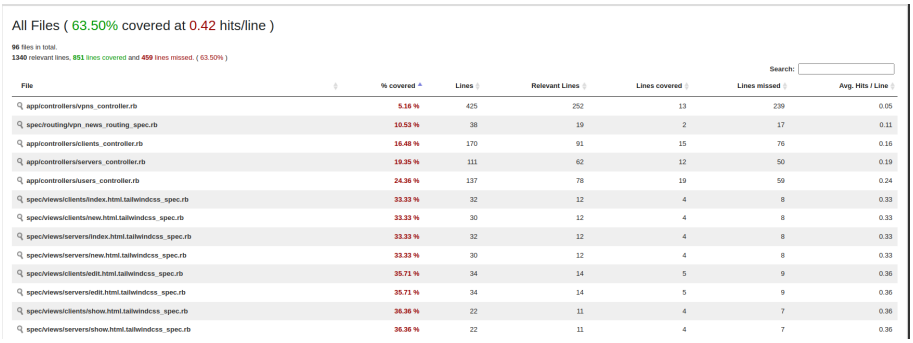
A continuación se muestra otro que consiste en la comprobación de si el método de creación de clientes funciona correctamente.

```
describe "POST /create" do
    context "with valid parameters" do
      it "creates a new Client" do
        expect {
          post clients_url, params: { client: valid_attributes }
        }.to change(Client, :count).by(1)
      end

      it "redirects to the created client" do
        post clients_url, params: { client: valid_attributes }
        expect(response).to redirect_to(client_url(Client.last))
      end
    end

    context "with invalid parameters" do
      it "does not create a new Client" do
        expect {
          post clients_url, params: { client: invalid_attributes }
        }.to change(Client, :count).by(0)
      end


      it "renders a response with 422 status (i.e. to display the 'new' template)" do
        post clients_url, params: { client: invalid_attributes }
        expect(response).to have_http_status(:unprocessable_entity)
      end

    end
  end
```

En este ejemplo se espera crear correctamente un cliente, para ello se le pasan los atributos necesarios para su creación y se redirecciona a la dirección específica de los clientes. En caso de que no reciba los parámetros específicos se muestra un mensaje explicativo. Además lanza un código de estado HTTP para invalidar la operación y mostrar por pantalla que claramente esa operación no ha sido realizada correctamente.

Para poder ejecutar todas las pruebas automáticamente, desde la terminal se debe ejecutar el comando *rspec*. Éste ejecutará todas las pruebas descritas, además generará un archivo html que muestra de manera descriptiva el porcentaje de pruebas superadas, las líneas cubiertas y las que no han podido ser evaluadas porque se han encontrado errores. Dentro de cada prueba se mostrará todo el porcentaje cubierto de esa prueba, además de otra información complementaria que facilita su análisis.

En la figura 7.1 se adjunta el resultado obtenido. En este momento se obtiene un porcentaje del 63.50 %. Con las siguientes versiones este porcentaje aumentará considerablemente, esperando llegar a un 95 %, que es el porcentaje esperado para un programa con un funcionamiento correcto.



All Files (63.50% covered at 0.42 hits/line)

96 files in total.
1340 relevant lines, **851** lines covered and **459** lines missed. (63.50%)

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
app/controllers/vpns_controller.rb	5.16 %	425	252	13	239	0.05
spec/routing/vpn_news_routing_spec.rb	10.53 %	38	19	2	17	0.11
app/controllers/clients_controller.rb	16.48 %	170	91	15	76	0.16
app/controllers/servers_controller.rb	19.35 %	111	62	12	50	0.19
app/controllers/users_controller.rb	24.36 %	137	78	19	59	0.24
spec/views/clients/index.html.tailwindcss_spec.rb	33.33 %	32	12	4	8	0.33
spec/views/clients/new.html.tailwindcss_spec.rb	33.33 %	30	12	4	8	0.33
spec/views/servers/index.html.tailwindcss_spec.rb	33.33 %	32	12	4	8	0.33
spec/views/servers/new.html.tailwindcss_spec.rb	33.33 %	30	12	4	8	0.33
spec/views/clients/edit.html.tailwindcss_spec.rb	35.71 %	34	14	5	9	0.36
spec/views/servers/edit.html.tailwindcss_spec.rb	35.71 %	34	14	5	9	0.36
spec/views/clients/show.html.tailwindcss_spec.rb	36.36 %	22	11	4	7	0.36
spec/views/servers/show.html.tailwindcss_spec.rb	36.36 %	22	11	4	7	0.36

Figura 7.1: Pruebas con RSpec

CAPÍTULO 8
Conclusiones y trabajos futuros

Para finalizar, se van a presentar las conclusiones que se han obtenido tras realizar este proyecto y la repercusión que ha tenido.

Se han podido lograr la gran mayoría de los objetivos propuestos al inicio del proyecto. La prueba de ello es que se ha logrado desarrollar una plataforma web completamente funcional, protegida con ciertos métodos de seguridad, con herramientas diseñadas para que, en el caso de que la aplicación se despliegue a gran escala, soporte múltiples conexiones. A pesar de no tener experiencia previa en el desarrollo de páginas web de este estilo, se han podido cumplir los requisitos con éxito.

Uno de los mayores desafíos que se dio a la hora de plantear el proyecto, fue afrontar la lógica del mismo, ya que en un principio desconocía como poder crear la base de datos correctamente y, por otra parte, no sabía como poder gestionar múltiples creaciones de servidores en una misma máquina. También se encontraron diversos problemas a la hora de la creación de gráficos en tiempo real sobre el tráfico que fluye a través de esas redes creadas, se encontró este problema porque no conocía como poder almacenar los datos de manera que no ocupasen mucho espacio, y que se pudieran ir actualizando cada ciertos intervalos de tiempo. En este sentido, los archivos ddr resultaron ser la opción más adecuada para poder resolver este dilema.

A la hora de hacer el despliegue, se tuvo que dedicar una gran parte del tiempo ya que era la primera vez que lo hacía de esta forma. Además, se quiso proporcionar funciones adicionales como el CI/CD, o incluso desplegarlo en un dominio real. De esta forma, para desarrollos futuos, se logrará detectar fallos de manera más eficiente, y se hará más sencillo implementar los cambios.

Se quiere destacar que el framework de Rails ha facilitado mucho el desarrollo de la plataforma, ya que gracias a la metaprogramación que incluye este entorno de desarrollo, agiliza mucho la creación de modelos, vistas y controladores. Además existen una gran variedad de gemas que dotan de funcionalidad añadida, como es el caso de *Devise* y *Chartkick* en este trabajo, la primera facilita toda la autenticación y registro de usuarios, y la segunda se encarga de los gráficos. También es un valor añadido que este framework posea una comunidad activa muy expandida que, continuamente, proporciona soluciones a las dudas de los usuarios.

A pesar de haber elaborado el proyecto cumpliendo los requisitos especificados, se pueden añadir extensiones a la plataforma para darle más funcionalidades. En primer lugar se quiere completar la función de limitar el ancho de banda para un determinado usuario. También se querría incluir diferentes tecnologías de creación de VPN distintas a OpenVPN. Por último, se quiere añadir la posibilidad de incluir varios servidores en una misma máquina, ofreciendo la posibilidad de poder elegir distintas CA. De esta ma-

nera se lograría añadir más funciones para administrar y gestionar las redes, además de ofrecer más configuraciones posibles para el usuario.

En conclusión, este proyecto ha supuesto una experiencia para consolidar todos los conocimientos adquiridos durante el grado y, además, adquirir unos nuevos. Ha supuesto un desafío desarrollar una aplicación en su totalidad por mi cuenta y con un tiempo reducido, sin embargo estoy satisfecho con el resultado final y con todos los conocimientos que he podido obtener a lo largo de todo este proceso.

8.1 Relación del trabajo desarrollado con los estudios cursados

Las asignaturas BDA, ISW, DEW y DCU han permitido realizar el análisis de la plataforma, permitiendo realizar el análisis y el diseño previo a la aplicación. Así ha permitido organizar la base de datos, los casos de uso y las casuísticas posibles que se puedan dar. Por otro lado, las asignaturas de DCLAN, RED han sido de gran ayuda para poder comprender como funcionan las VPN y como poder gestionarlas correctamente, además de entender como funcionan ciertos elementos para desplegar la aplicación. Por último, las asignaturas de SSR y SRE han permitido dotar de seguridad a toda la aplicación para ptrotegerla ante posibles ataques. Gran parte de estas asignaturas me permitieron trabajar en proyectos reales que me proporcionaron experiencia, la cual se ha podido reflejar a la hora de elaborar este trabajo.

Bibliografía

[1] David Heinemeier Hansson and Rails Core Team. Ruby on rails. *Development*, 4:0, 2009. Disponible en http://kelas-karyawan-bali.kurikulum.org/IT/en/2420-2301/Rails-web-framework_3884_kelas-karyawan-bali-kurikulumngetesumum.html. Fecha de acceso: 03/05/2023.

[2] Virtual private network market size, share trends analysis report by component, by type (site-to-site, remote access, extranet), by deployment mode, by end use, by region, and segment forecast, 2020 - 2027). Technical report, 2018. Disponible en https://www.grandviewresearch.com/industry-analysis/virtual-private-network-market. Fecha de acceso: 25/03/2023.

[3] Justina Alexandra Sava. Global vpn market size 2019-2027. Technical report, 2023. Disponible en https://www.statista.com/statistics/542817/worldwide-virtual-private-network-market/. Fecha de acceso: 25/03/2023.

[4] Paul Ferguson and Geoff Huston. What is a vpn? 1998. Disponible en https://cpham.perso.univ-pau.fr/ENSEIGNEMENT/COMMUN/vpn_ferguson.pdf. Fecha de acceso: 17/04/2023.

[5] Softether. Technical report. Disponible en https://www.softether.org/. Fecha de acceso: 03/04/2023.

[6] Opensource. Technical report. Disponible en https://opensource.com/resources/what-open-source. Fecha de acceso: 03/04/2023.

[7] Libreswan. Technical report. Disponible en https://libreswan.org/. Fecha de acceso 10/04/2023.

[8] Ipsec. Technical report. Disponible en https://www.cloudflare.com/es-es/learning/network-layer/what-is-ipsec/. Fecha de acceso: 08/04/2023.

[9] Strongswan. Technical report. Disponible en https://www.strongswan.org/. Fecha de acceso: 10/04/2023.

[10] Certificado x.509. Technical report. Disponible en https://protecciondatos-lopd.com/empresas/certificado-x509/. Fecha de acceso: 08/04/2023.

[11] Ssl-vpn. Technical report. Disponible en https://www.f5.com/es_es/glossary/ssl-vpn. Fecha de acceso: 15/04/2023.

[12] Bernhard Bauer and James Odell. Uml 2.0 and agents: how to build agent-based systems with the new uml standard. *Engineering applications of artificial intelligence*, 18(2):141–157, 2005. Disponible en https://www.sciencedirect.com/science/article/abs/pii/S0952197604001903. Fecha de acceso: 20/04/2023.

[13] Quirin Scheitle, Taejoong Chung, Jens Hiller, Oliver Gasser, Johannes Naab, Roland van Rijswijk-Deij, Oliver Hohlfeld, Ralph Holz, Dave Choffnes, Alan Mislove, et al. A first look at certification authority authorization (caa). *ACM SIGCOMM Computer Communication Review*, 48(2):10–23, 2018. Disponible en https://dl.acm.org/doi/abs/10.1145/3213232.3213235. Fecha de acceso: 1/05/2023.

[14] Oscar Lizama, Geordy Kindley, JI Jeria Morales, and Agustın Gonzales. Redes de computadores: Arquitectura cliente-servidor. *Universidad Tecnica Federico Santa Maria*, pages 1–8, 2016. Disponible enhttps://shorturl.at/gkmx1. Fecha de acceso: 1/05/2023.

[15] Enrique M Suárez. ¿ que es una base de datos relacional? *Universidad de murcia, Murcia, España*, 2008. Disponible en https://shorturl.at/xACH0. Fecha de acceso: 03/05/2023.

[16] Jesús Gamaliel Camarena Sagredo, Adrian Trueba Espinosa, Magally Martínez Reyes, and María de Lourdes López García. Automatización de la codificación del patrón modelo vista controlador (mvc) en proyectos orientados a la web. *CIENCIA ergo-sum, Revista Científica Multidisciplinaria de Prospectiva*, 19(3):239–250, 2012. Disponible en https://www.redalyc.org/pdf/104/10423895005.pdf. Fecha de acceso: 03/05/2023.

[17] freecodecamp. Technical report. Disponible en https://www.freecodecamp.org/news/crud-operations-explained/ Fecha de acceso: 14/05/2023.

[18] Pete Loshin. Ipv6: Theory, protocol, and practice. 2004. Disponible en https://books.google.es/books?hl=es&lr=&id=6JDuPUzMU4AC&oi=fnd&pg=PP1&dq=IPv6&ots=Q4vnOreODl&sig=KtFyu5bhr_C0iVZoZEMLfiav0IY#v=onepage&q=IPv6&f=false. Fecha de acceso: 11/05/2023.

[19] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1:36–63, 2001. Disponible en https://link.springer.com/article/10.1007/s102070100002 Fecha de acceso: 11/05/2023.

[20] Evgeny Milanov. The rsa algorithm. *RSA laboratories*, pages 1–11, 2009. Disponible en https://pdfdirectory.com/pdf/0702-the-rsa-algorithm.pdf Fecha de acceso: 11/05/2023.

[21] Ueli M Maurer and Stefan Wolf. The diffie–hellman protocol. *Designs, Codes and Cryptography*, 19(2-3):147–171, 2000. Disponible en https://link.springer.com/article/10.1023/A:1008302122286 Fecha de acceso: 11/05/2023.

[22] Easyrsa. Technical report. Disponible en https://easy-rsa.readthedocs.io/en/latest/ Fecha de acceso: 14/05/2023.

[23] Unbound. Technical report. Disponible en https://www.nlnetlabs.nl/projects/unbound/about/.

[24] Jon Postel and JK Reynolds. Rfc0854: Telnet protocol specification, 1983. Disponible en https://dl.acm.org/doi/pdf/10.17487/RFC0854.

[25] Kjeld Egevang and Paul Francis. The ip network address translator (nat). Technical report, 1994. Disponible en https://www.rfc-editor.org/rfc/rfc1631 Fecha de acceso: 26/05/2023.

[26] registro-a. Technical report. Disponible en https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/registro-a/ Fecha de acceso: 26/05/2023.

[27] Christos Karayiannis and Christos Karayiannis. Obtaining a domain name with ddns. *Web-Based Projects that Rock the Class: Build Fully-Functional Web Apps and Learn Through Doing*, pages 125–163, 2019. Disponible en https://link.springer.com/chapter/10.1007/978-1-4842-4463-0_4 Fecha de acceso: 26/05/2023.

[28] Will Reese. Nginx: the high-performance web server and reverse proxy. *Linux Journal*, 2008(173):2, 2008. Disponible en https://dl.acm.org/doi/fullHtml/10.5555/1412202.1412204 Fecha de acceso: 26/05/2023.

[29] Sandra Ortega Martorell and Liusbetty Canino Gutiérrez. Protocolo de seguridad ssl. *Ingeniería Industrial*, 27(2-3):57–62, 2006. Disponible en https://www.redalyc.org/pdf/3604/360433561012.pdf Fecha de acceso: 26/05/2023.

[30] Mohammad Rizky Pratama and Dana Sulistiyo Kusumo. Implementation of continuous integration and continuous delivery (ci/cd) on automatic performance testing. In *2021 9th International Conference on Information and Communication Technology (ICoICT)*, pages 230–235. IEEE, 2021. Disponible en https://ieeexplore.ieee.org/abstract/document/9527496 Fecha de acceso: 29/05/2023.

[31] Martin Fowler. mocking. Technical report. Disponible en https://martinfowler.com/articles/mocksArentStubs.html Fecha de acceso: 09/06/2023.

[32] JNovNeb. Vpninst. Technical report. Disponible en https://gist.github.com/jnovneb/3171e15f1d0e5e7230c35219258d9f4a Fecha de acceso: 17/06/2023.

[33] JNovNeb. Clientinst. Technical report. Disponible en https://gist.github.com/jnovneb/b73810a484916f20fa06bd3ce1ea5fc5 Fecha de acceso: 17/06/2023.

APÉNDICE A
Codigo de creación

A.1 Creación de una VPN

Se adjunta la parte más relevante en el script de creación, si quiere consultar todo el código puede hacerlo en la siguiente referencia [32]

```
# Generate a random identifier for CN and one for server name
SERVER_CN="cn_$(head /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w 16 | head
 -n 1)"
echo "$SERVER_CN" >SERVER_CN_GENERATED
SERVER_NAME="$NAME"
echo "$SERVER_NAME" >SERVER_NAME_GENERATED
if test "$DH_TYPE" = "2"; then
# ECDH keys are generated on-the-fly so we don't need to
  generate them beforehand
openssl dhparam -out dh.pem $DH_KEY_SIZE
fi
./easyrsa --batch build-server-full "$SERVER_NAME" nopass
EASYRSA_CRL_DAYS=3650 ./easyrsa gen-crl
```

A.2 Creación de un cliente

Como en el caso anterior, únicamente se adjunta el segmento de código que se considera más relevante. En caso de querer consultar la totalidad del mismo, se puede hacer a través de la siguiente referencia [33]

```
# Generate client using easyrsa
cd /etc/openvpn/easy-rsa/ || exit
case $CONTRASENA in
  "")
      ./easyrsa --batch build-client-full "$CLIENT" nopass
    ;;
  *)
    echo "You will have to enter your password"
    ./easyrsa --batch build-client-full "$CLIENT"
    ;;
esac
```

APÉNDICE B
Código de eliminación

B.1 Eliminación de VPN

```
#!/bin/bash

# Verify script is being executed with superusers privileges
if [ "$EUID" -ne 0 ]; then
  echo "Should be executes as a superuser."
  exit 1
fi

# Obtain server name and path
SERVIDOR="$1"
RUTA="$2"

#Stop and disable OpenVPN service
systemctl stop openvpn@${SERVIDOR}
systemctl disable openvpn@${SERVIDOR}

#Delete configuration files and each certificate
rm -rf /etc/openvpn/${SERVIDOR}
rm -rf ${RUTA}/

echo "VPN element: \"$SERVIDOR\" has been deleted correctly."
```

B.2 Eliminación de cliente

```
#!/bin/bash

# Verify script is being executed with superusers privileges
if [ "$EUID" -ne 0 ]; then
  echo "Este script debe ejecutarse con privilegios de superusuario."
  exit 1
fi

#Assign client name to revoke into a variable
CLIENT="$1"
ruta="$2"
```

```
# Verify if the client exists
CLIENTEXISTS=$(tail -n +2 /etc/openvpn/easy-rsa/pki/index.txt | grep
-c -E "/CN=$CLIENT\$")
if [[ $CLIENTEXISTS == '0' ]]; then
  echo ""
  echo "The client does not exist."
  exit
fi

cd /etc/openvpn/easy-rsa/ || exit
./easyrsa --batch revoke "$CLIENT"
EASYRSA_CRL_DAYS=3650 ./easyrsa gen-crl
rm -f /etc/openvpn/crl.pem
cp /etc/openvpn/easy-rsa/pki/crl.pem /etc/openvpn/crl.pem
chmod 644 /etc/openvpn/crl.pem
rm -r "/$ruta/$CLIENT.ovpn"
sed -i "/^$CLIENT,.*/d" /etc/openvpn/ipp.txt
cp /etc/openvpn/easy-rsa/pki/index.txt{,.bk}

echo ""
echo "The cert for the client: $CLIENT has been revoked."
```

APÉNDICE C
Descarga de cliente

```
document.querySelectorAll('.btnc').forEach((button) => {
  button.addEventListener('click', (event) => {
    event.preventDefault();
    const client_id = event.target.dataset.clientId;
    const client_name = event.currentTarget.dataset.clientName;

    // AJAX Request
    fetch('/home', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'X-CSRF-Token': '<%= form_authenticity_token %>'
      },
      body: JSON.stringify({ client_id: client_id })
    })
    .then(response => {
      if (response.ok) {
        return response.blob();
      } else {
        throw new Error('Error en la solicitud AJAX');
      }
    })
    .then(blob => {
      // Creat a temporary link and download the file
      const url = URL.createObjectURL(blob);
      const a = document.createElement('a');
      a.href = url;
      a.download = `${client_name}.ovpn`;
      a.click();

      // Delete the temporary URL
      URL.revokeObjectURL(url);
    })
    .catch(error => {
      console.error('Error:', error);
    });
  });
});
```

APÉNDICE D
Monitorización

D.1 Interfaz de RRD

```
    require 'rrd'

# Class to act as a interface between the VPN and the RRD files
class VPNtoRRD
  def initialize(vpn)
    @vpn = vpn.to_s
    @vpn_file = "#{Rails.root.join('vpn_files','rrd')}#{vpn}.rrd"
    @vpn_rrd  = RRD::Base.new(@vpn_file)
  end

  def update(bytes_in, bytes_out, clients)
    create unless File.exist? @vpn_file
    @vpn_rrd.update Time.now, bytes_in, bytes_out, clients
  end

  def graph
    # Local variable to avoid problems with the instance variable in module RRD
    rrd_file = @vpn_file
    RRD.graph "#{@vpn}-trf.png",
              title: 'VPN traffic',
              width: 800, height: 250,
              color: ['FONT#000000', 'BACK#FFFFFF'] do
      line rrd_file,
           bytes_in: :average,
           color: '#FF0000',
           label: 'Bytes IN', legend: 'Bytes IN',
           width: 1
      line rrd_file,
           bytes_out: :average,
           color: '#00FF00',
           label: 'Bytes OUT', legend: 'Bytes OUT',
           width: 1
    end
    RRD.graph "#{@vpn}-cli.png",
           title: 'VPN clients',
           width: 800, height: 250,
```

```
            color: ['FONT#000000', 'BACK#FFFFFF'] do
      line rrd_file,
            clients: :average,
            color: '#0000FF',
            label: 'Clients', legend: 'Clients',
            width: 1
    end
  end

  private

  def create
    @vpn_rrd.create start: Time.now - 1.minute, step: 5.minutes do
      datasource 'bytes_in',  type: :gauge, heartbeat: 1.minutes, min: 0,
      max: :unlimited
      datasource 'bytes_out', type: :gauge, heartbeat: 1.minutes, min: 0,
      max: :unlimited
      datasource 'clients',   type: :gauge, heartbeat: 1.minutes, min: 0,
      max: :unlimited
      archive :average, every: 10.minutes, during: 1.year
    end
  end
end
```

D.2 Métodos para openvpnmanager

```
private

  # Send a command to the OpenVPN management interface
  def send_command(command)
    return if @client.nil?

    c = @client.cmd('String' => command, 'Match' => /(SUCCESS:.*\n|ERROR:.*\n
    |END.*\n)/)
    if c =~ /\AERROR\: (.+)\n\Z/
      raise Regexp.last_match 1
    elsif c =~ /\ASUCCESS\: (.+)\n\Z/
      Regexp.last_match 1
    else
      c
    end
  end


  # Get information about clients connected list and routing table.
    Return two arrays of arrays with lists inside.
  # For each client in client_list array there is: Common Name,
    Addressing Infos, Bytes in/out, Uptime.
  # Instead for each route entry there is: IP/Eth Address
    (depend on tun/tap mode), Addressing, Uptime.
  # @return [Hash{Symbol->Unknown}]
  def status
```

```
  client_list_flag = 0, routing_list_flag = 0
  clients = {}
  routes = {}

  unless @client.nil?
    c = send_command 'status'
    c.split("\n").each do |line|
      # End Information Markers
      client_list_flag  = 0 if line == 'ROUTING TABLE'
      routing_list_flag = 0 if line == 'GLOBAL STATS'
      # Update Clients Connected List
      # Common Name,Real Address,Bytes Received,Bytes Sent,Connected Since
      if client_list_flag == 1
        client = line.split(',')
        clients[client[0]] ||= []
        clients[client[0]] << {
          real_address: client[1],
          bytes_received: client[2],
          bytes_sent: client[3],
          connected_since: client[4].chop
        }
      end
      # Update Routing Info List
      # Virtual Address,Common Name,Real Address,Last Ref
      if routing_list_flag == 1
        route = line.split(',')
        routes[route[0]] = { common_name: route[1], real_address: route[2],
        last_ref: route[3] }
      end
      # Start Information Markers
      client_list_flag  = 1 if line == 'Common Name,Real Address,Bytes,
      Received,Bytes Sent,Connected Since'
      routing_list_flag = 1 if line == 'Virtual Address,Common Name,
      Real Address,Last Ref'
    end
  end
  { clients:, routes: }
end


    # Get information about number of clients connected and
    traffic statistic (bytes in & bytes out)
def stats
  stats_arr = send_command('load-stats').split(',')
  {
    clients: stats_arr[0].gsub('nclients=', '').to_i,
    bytes_download: stats_arr[1].gsub('bytesin=', '').to_i,
    bytes_upload: stats_arr[2].chop!.gsub('bytesout=', '').to_i
  }
end
```

APÉNDICE E
Objetivos de desarrollo sostenible

Tabla E.1: Objetivos de Desarrollo Sostenible

Objetivos de Desarrollo Sostenible	Alto	Medio	Bajo	No procede
ODS 1. Fin de la pobreza.				X
ODS 2. Hambre cero.				X
ODS 3. Salud y bienestar.				X
ODS 4. Educación de calidad.			X	
ODS 5. Igualdad de género.				X
ODS 6. Agua limpia y saneamiento.				X
ODS 7. Energía asequible y no contaminante.				X
ODS 8. Trabajo decente y crecimiento económico.		X		
ODS 9. Industria, innovación e infraestructuras.		X		
ODS 10. Reducción de las desigualdades.				X
ODS 11. Ciudades y comunidades sostenibles.				X
ODS 12. Producción y consumo responsables.				X
ODS 13. Acción por el clima.				X
ODS 14. Vida submarina.				X
ODS 15. Vida de ecosistemas terrestres.				X
ODS 16. Paz, justicia e instituciones sólidas.				X
ODS 17. Alianzas para lograr objetivos.		X		

Tras el desarrollo de este proyecto basada en la creación de un orquestador de VPN, que es capaz de monitorizar el tráfico que discurre por las redes creadas, se ha tenido en cuenta cómo este trabajo puede contribuir a la consecución de los Obketivos de Desarrollo Sostenible de la UE. Más concretamente con los objetivos siguientes: Educación de calidad, Trabajo decente y crecimiento económico, Industria, innovación e infraestructuras y Alianzas para lograr objetivos.

Este trabajo puede ayudar a lograr una educación de calidad porque puede servir para crear redes que permitan al estudiante conectarse a la infraestructura del centro educativo para poder seguir con sus estudios de manera remota y segura. Lo cual se ha visto muy necesario, tras la pandemia global que hubo en el 2020, de esta manera se podría mantener el sistema de educación permitiendo a los estudiantes y al profesorado conectarse al centro con estas redes.

Por otro lado, también cumpliría con el objetivo relativo al trabajo decente y al crecimiento económico, ya que mediante este proyecto se pueden crear VPN que permitan a un trabajador ejercer su empleo en remoto, facilitando el teletrabajo a los mismos. Por

otra parte puede facilitar el crecimiento económico porque permite la conexión entre sedes de una misma compañía sin que estén físicamente conectadas y mediante un canal seguro. Con estas mismas características también se puede cumplir el objetivo relacionado con la industria, innovación e infraestructuras, ya que permite conectar distintas infraestructuas entre sí.

Por último, cumple con el objetivo referido a alianzas para lograr objetivos ya que fomenta la cooperación entre los administradores de red para lograr los objetivos que se planteen para crear las redes deseadas.