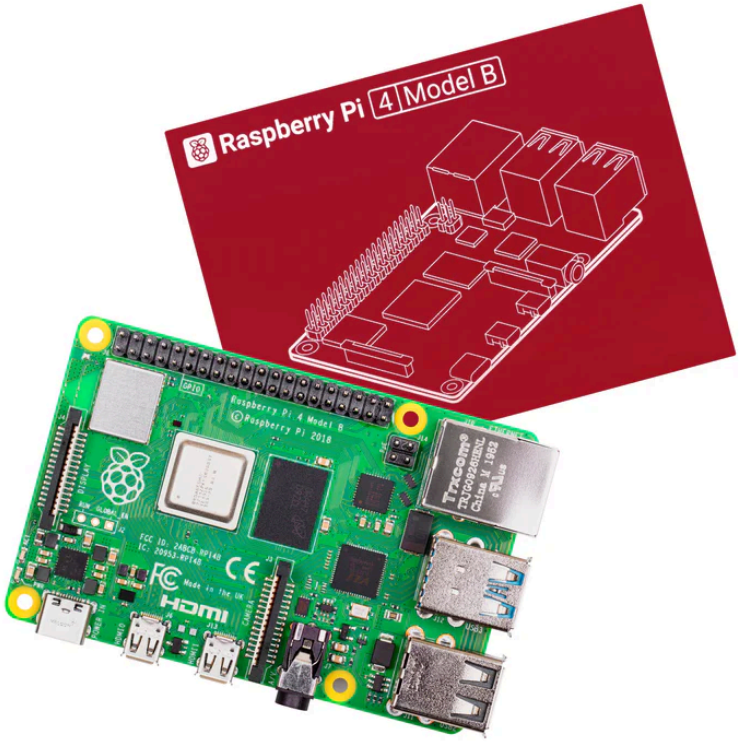
# PROYECTO INTEGRADO
# **VPN Autogestionada con Django y Raspberry Pi.**





| |
|---|
| Autor: Nicolás Quintero Gil |
| Tutor: Blanca Vargas Machuca |
| |

| |
|---|
| I.E.S. Francisco Romero Vargas (Jerez de la Frontera) |
| Administración de Sistemas Informáticos en Red |
| Curso: 2024/2025 |

# Tabla de contenido

# Introducción.

## Introducción.

Este proyecto tiene como meta diseñar e implementar una red VPN segura y autogestionada en una Raspberry Pi. Para lograrlo, se combinarán tecnologías como OpenVPN, ZeroTier One y Django, creando así un sistema de administración centralizado a través de una interfaz web. La idea es ofrecer una opción accesible y segura para que los clientes se conecten de forma remota a una red privada desde cualquier lugar, todo gestionado completamente a través de una plataforma web.

## Finalidad.

El propósito de este proyecto es proporcionar un entorno de red privada virtual (VPN) que sea funcional y seguro, permitiendo a los usuarios conectarse desde cualquier lugar con sus credenciales. Al mismo tiempo, el administrador podrá gestionar de manera centralizada los accesos, certificados y privilegios de los usuarios. Todo esto se hará sin depender de soluciones comerciales costosas, utilizando software libre y hardware asequible.

## Objetivos.

Este proyecto busca implementar un sistema de red privada virtual (VPN) autogestionada, utilizando una Raspberry Pi como servidor principal. Para ello, se desarrollará una plataforma web con Django, que facilitará la administración centralizada de los usuarios del sistema.

Uno de los objetivos técnicos más importantes es instalar y configurar OpenVPN como software de red privada, automatizando tanto la generación como la revocación de certificados digitales.

Para facilitar el acceso remoto sin necesidad de complicadas configuraciones de red, se integrará ZeroTier One. Gracias a esto, no será necesario lidiar con NAT ni depender de direcciones IP públicas.

A través de la interfaz web, cada usuario podrá iniciar sesión de manera sencilla y descargar sus credenciales en un archivo comprimido. Al mismo tiempo, el administrador tendrá a su disposición un panel de control claro e intuitivo, que le permitirá gestionar los accesos de manera intuitiva: activar o desactivar usuarios y revocar certificados cuando sea necesario.

El sistema en su conjunto estará bien protegido por algunas medidas de seguridad, como el uso de TLS para cifrar las comunicaciones y un esquema de control de acceso basado en roles, asegurando que tanto los datos como el acceso estén siempre bien resguardados.

## Medios necesarios.

Hardware:
- Raspberry Pi 4
- Router con acceso a internet (uso desde el instituto con ZeroTier)
- PC cliente con Windows

Software:
- Raspberry Pi OS
- OpenVPN 2.6.3
- ZeroTier One
- Django + Python 3

# Planificación.

- Instalación de Raspberry Pi OS y configuración básica: 3 horas
- Instalación y configuración de OpenVPN: 6 horas
- Instalación y configuración de ZeroTier One: 2 horas
- Desarrollo del portal web en Django: 25 horas
- Funciones de gestión de usuarios (alta, baja, activación, desactivación): 8 horas
- Automatización de generación y revocación de certificados: 8 horas
- Implementación del sistema de descarga automática de archivos .zip: 3 horas
- Seguridad: configuración de TLS, control por roles: 4 horas
- Pruebas de conexión, revocación, regeneración y acceso remoto: 5 horas
- Preparación de documentación y defensa: 4 horas

Total estimado: 68 horas

# Realización del Proyecto.

## Trabajos realizados.

Durante el desarrollo del proyecto, se llevó a cabo la configuración completa del servidor OpenVPN en una Raspberry Pi 4, configurando la infraestructura de la red privada segura. Esta configuración incluyó la generación de certificados seguros, la definición de políticas de red, y la implementación de reglas de firewall y NAT necesarias para enrutar correctamente el tráfico de los clientes conectados.

Luego, se integró ZeroTier One, una solución de red virtual que permite conectar dispositivos a través de Internet sin necesidad de configurar puertos ni contar con IPs públicas. Esta herramienta ha permitido acceder remotamente a la Raspberry Pi desde ubicaciones como el instituto, facilitando tanto el desarrollo como la demostración del sistema en entornos externos.

Uno de los pilares del proyecto ha sido el desarrollo de una aplicación web utilizando Django, que actúa como plataforma de gestión centralizada. Esta interfaz permite al administrador realizar de manera intuitiva tareas como la creación de usuarios, activación, desactivación, eliminación o asignación de privilegios. Además, se añadió un sistema de autenticación web con protección mediante credenciales y control de acceso por roles, separando claramente las funciones del administrador de las de los usuarios finales.

Para reducir la intervención manual y mejorar la seguridad, se han desarrollado varios scripts que automatizan procesos, como la generación, revocación y regeneración de certificados. Estos scripts se integran con la lógica del portal web para ejecutar las acciones según las órdenes del administrador, asegurando que cada cambio en la interfaz tenga un efecto directo en el backend del sistema.

Una vez generados los certificados, el sistema los empaqueta junto con los archivos de configuración necesarios (.ovpn, claves CA, etc.) en un archivo comprimido .zip, el cual se almacena temporalmente y quedaría disponible para el usuario desde su panel personal. Este mecanismo garantiza que cada usuario reciba exactamente los archivos que necesita.

En materia de seguridad, se han aplicado buenas prácticas como el uso de cifrado TLS en todas las comunicaciones del portal y un control de permisos para restringir funciones avanzadas solo a usuarios con privilegios administrativos, además el propio acceso al portal web está cifrado por Zero Tier. Todo ello ha sido probado con distintos escenarios para asegurar la estabilidad, compatibilidad (especialmente con clientes Windows).

# Problemas encontrados.

Durante el desarrollo del proyecto se presentaron diversos problemas técnicos que requirieron soluciones específicas. Uno de los primeros obstáculos fue a la hora de que un sistema operativo Windows se conectara correctamente a la VPN. Esto se debía a diferencias en la configuración del adaptador de red, incompatibilidades con ciertos drivers, y la necesidad de ejecutar el cliente OpenVPN con privilegios de administrador para aplicar correctamente las rutas.

Otro desafío fue la integración entre los scripts bash encargados de la gestión de certificados y la lógica interna del portal web desarrollado con Django. Fue necesario crear un sistema de comunicación fiable entre ambos entornos, garantizando que cada acción realizada desde la interfaz web se reflejara correctamente en los archivos del servidor OpenVPN, sin provocar inconsistencias o duplicidades. Esto implicó gestionar permisos de ejecución, validar rutas absolutas, manejar errores silenciosos y evitar bloqueos por concurrencia al acceder al sistema de archivos.

También surgieron complicaciones al hacer que OpenVPN y ZeroTier coexistieran en la misma Raspberry Pi. El principal problema residía en los conflictos de enrutamiento que se producían al levantar ambas interfaces virtuales, especialmente cuando el cliente conectado a través de ZeroTier intentaba acceder a recursos ofrecidos por OpenVPN. Fue necesario definir manualmente reglas de enrutamiento para evitar bucles o pérdidas de conectividad, ajustando prioridades de red y asegurando que cada servicio usara su propia interfaz de forma exclusiva.

Además, hubo ciertos inconvenientes menores, como errores de permisos al intentar generar archivos desde scripts lanzados por el servidor web, dificultades con los certificados Let’s Encrypt en redes sin acceso directo a internet, y problemas con certificados ssl a la hora de usar Zero Tier. También hubo que lidiar con reinicios del servidor que interrumpían servicios mal configurados para iniciar al arranque, lo que obligó a definir servicios en systemd correctamente.

# Modificaciones sobre el proyecto planteado inicialmente.

Durante el avance del proyecto, fue necesario realizar algunos cambios estratégicos con respecto al planteamiento inicial. En un principio se había previsto integrar herramientas de monitoreo como Zabbix o Nagios para supervisar el estado del servidor y las conexiones activas. Sin embargo, debido a la carga adicional que suponía para el rendimiento de la Raspberry Pi y al enfoque del proyecto en la gestión de usuarios, se decidió prescindir de estas herramientas para mantener la ligereza del sistema.

También se rediseñó la arquitectura del portal web. Inicialmente se concibió como una interfaz más básica, pero con el tiempo se reorganizó su estructura para facilitar futuras ampliaciones, mejorando la modularidad del código y dividiendo claramente los componentes según su responsabilidad (vistas, modelos, scripts, etc.). Este rediseño permitió incorporar nuevas funcionalidades como la descarga automática de paquetes .zip personalizados o el control por roles.

Además, se incorporaron funcionalidades no previstas como la regeneración automática de certificados al reactivar un usuario desactivado, y la revocación inmediata del certificado al desactivarlo o eliminarlo. Estas acciones fueron añadidas tras detectar la necesidad de mantener siempre sincronizado el acceso lógico (desde el panel) y físico (nivel de certificado).

# Posibles mejoras al proyecto.

Aunque el sistema creado ya hace lo que debe, aún se puede mejorar su seguridad, facilidad de uso y capacidad de gestión.

Por ejemplo, se podría poner un sistema de auditoría que guarde todo en una base de datos segura, como los cambios en la configuración,acciones realizadas por administradores o intentos de modificación de archivos. Esta información sería valiosa en caso de incidentes de seguridad o para cumplir con normativas de protección de datos.

Otra mejora interesante sería usar un sistema de identificación, como LDAP, o la integración de Google Workspace o Microsoft Azure AD. Así ,permitiría el uso de credenciales corporativas ya existentes y reduciría la carga de gestión de usuarios. También, estaría bien meter todo el sistema (servidor OpenVPN, servidor web, Django y los programas para manejarlo) en contenedores Docker. Esta mejora facilitaría enormemente el despliegue en otros entornos, simplificaría las actualizaciones y ayudaría a mantener una mayor coherencia en las versiones del software, minimizando posibles incompatibilidades y errores que puedan surgir de entornos mal configurados.

# Bibliografía.

Documentación oficial de OpenVPN: https://openvpn.net/

Documentación de Django: https://mentecatodev.github.io/django/

Documentación de ZeroTier: https://docs.zerotier.com/

Apuntes del módulo de Servicios en Red y Administración de Sistemas