Universidad Politécnica de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

TRABAJO FIN DE CARRERA

Arquitectura escalable y segura para plataformas IIoT

Autor: Carlota García Varés

Director: Fernando Pérez Costoya

Matrícula: c950184

MADRID, JULIO 2018

Tabla de contenidos

Capítulo 1. Introducción

Necesidades de la solución

Actualmente se habla mucho del Internet de las cosas (**IoT**) y del Internet Industrial de las cosas (**IIoT**), de cómo aplicarlo a los distintos tipos negocios, casas domotizadas, Smart metering, tracking de mercancías, etc. pero pocas veces se habla de los problemas que se pueden presentar en las comunicaciones, en el ahorro del envío de datos, reducir los riesgos de ataques a los dispositivos conectados... Y dentro de esto, las **plataformas IoT** son clave para el desarrollo de aplicaciones, software y servicios para la interconexión de personas y “cosas”.

Empezaremos con la definición de 2 conceptos tecnológicos que influyen en el proyecto: **IoT** e **IIoT**.

Internet of Things(IoT)

Internet of Things o el Internet de las Cosas, se define habitualmente como una red de objetos físicos o "cosas" que llevan incorporados circuitos electrónicos, software, sensores y sistemas de comunicaciones que les permiten el intercambio de datos con el fabricante, con un operador o con otros objetos. Esto permitiría que dichos objetos sean monitorizados y controlados remotamente, permitiendo una integración directa entre los sistemas computacionales y el mundo físico. Estos objetos disponen, gracias a su conexión con la red, de funciones y capacidades más allá de lo que podrían ofrecer por sí mismos. El Internet de las cosas puede ser aplicado a diversos campos y aplicaciones, teniendo repercusiones en campos como la domótica, los sistemas médicos, el transporte, etc.

Figura 1.1 : IoT.- Imagen que lo representa

Industrial Internet of Things(IIoT)

El concepto de Industrial Internet of Things nace como una evolución sobre el IoT, centrándose como su nombre indica en el uso de aplicaciones industriales y empresariales. Es un concepto reciente en el mundo del IoT para diferenciar entre las aplicaciones destinadas al mercado doméstico y aquellas diseñadas pensando en su uso comercial en empresas y complejos industriales. Esta diferencia en el objetivo de esta tecnología marca todas sus diferencias, como el alto número de dispositivos simultáneos a monitorizar o el énfasis en servicios críticos.

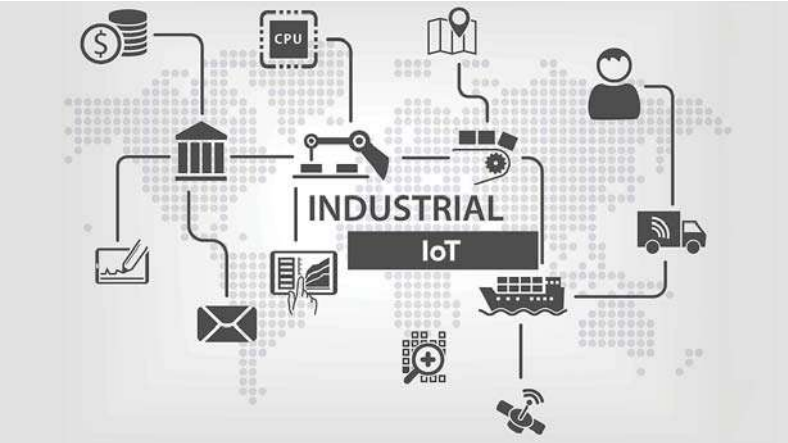

Figura 1.2 : IIoT.- Imagen que lo representa

Una plataforma IoT o IIoT consta de los siguientes bloques funcionales:

1. **Conectividad y normalización:** garantiza la transmisión de datos y la interacción con los dispositivos a través de diferentes protocolos y diferentes formatos de datos en una interfaz *de software* así como los modelos de información intercambiados.
2. **La gestión de dispositivos:** asegura que todo los elementos conectados funcionan correctamente. Así como la capacidad de operar remotamente sobre ellos y la actualización del firmware y configuración de los mismos.
3. **Base de datos:** almacenamiento escalable de información del dispositivo.
4. **Procesamiento y gestión de la acción:** aporta información basada en reglas que permitan la ejecución de operaciones basadas en datos recibidos del sensor.
5. **Analítica:** lleva a cabo una serie de análisis complejos de la agrupación de datos básicos y de aprendizaje automático.

6. **Visualización:** permite a los usuarios observar las tendencias de cuadros de mando de visualización de datos, que se retratan a través de gráficos.
7. **Interfaces externas:** se integran con los sistemas de terceros a través de una función de interfaces de programación de aplicaciones (API).

En la empresa amplía))) (Amplía Soluciones S.L), con una gran experiencia en M2M e IoT, se ha desarrollado una plataforma IIoT denominada OpenGate® que implementa las características clave para administrar, gestionar y proteger los dispositivos.

OpenGate® permite:

- Automatizar la recolección del inventario de los dispositivos.
- Aplicar configuración automática sobre los dispositivos.
- Autenticación de los dispositivos y cifrado de comunicaciones entre ellos y la plataforma.
- Comprobar automáticamente el estado de los dispositivos y de su conexión.
- Crear reglas para supervisar parámetros específicos de los dispositivos y de su comportamiento.

La arquitectura de OpenGate® consta de los bloques anteriormente comentados, como puede verse en la Figura 1.3.

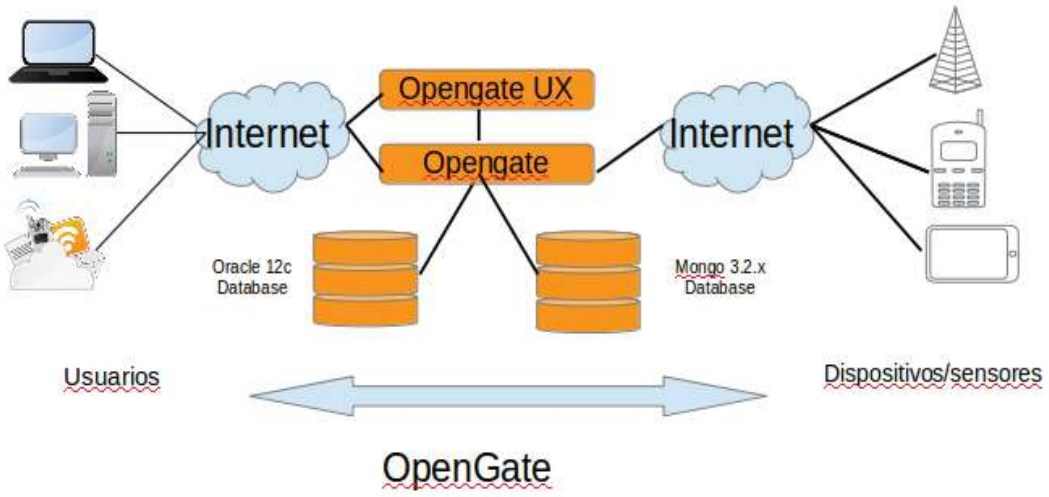

Figura 1.3 : Arquitectura de OpenGate actual

En el mundo del IoT cada vez parece más necesario una estandarización en los protocolos IoT ya que actualmente es casi inexistente. Contrasta bastante con otro tipo de soluciones, como por ejemplo las de entornos Web, en donde el *stack* de protocolos de trabajo está claramente definido. La siguiente imagen representa el *stack* IoT frente al de Web.

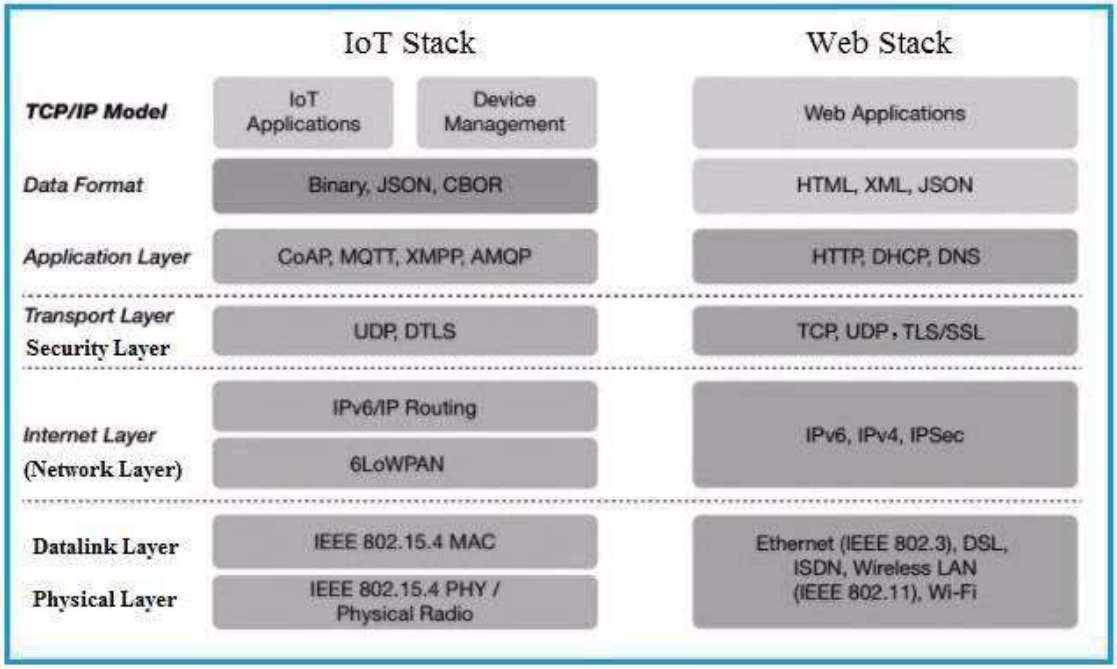

Figura 1.4 : IoT stack frente a Web stack

Las causas de esta falta de acuerdos para lograr una estandarización entre los actores del mundo IIoT (fabricantes de hardware, programadores, clientes finales, etc.) son varias pero destacan algunas: es un mundo en el que existen restricciones de memoria, capacidad de proceso, batería, almacenamiento, etc. que dificulta el uso de protocolos más estandarizados. Paradójicamente, en las soluciones IIoT (y pese a las limitaciones de los dispositivos) se requieren propiedades de otras soluciones como las de entornos web: conectividad, garantía de que los datos recolectados se entregan, seguridad en las comunicaciones, etc.

En los últimos años se ha avanzado mucho en las comunicaciones inalámbricas , en el campo de la sensorización y en los microcontroladores que permiten incorporar capacidad de proceso local y de interconexión en casi cualquier ubicación. Por ello, los fabricantes de dispositivos están incorporando capacidad de almacenamiento y procesamiento de aplicaciones, de forma que se puedan ejecutar programas que realicen operaciones locales tanto para la gestión de las comunicaciones como de

operaciones. También incorporan diferentes interfaces de entrada/salida para la interconexión con las máquinas. Por otro lado, deben de presentar un reducido tamaño y consumo así como precios competitivos que permitan ser desplegados en el mayor número de máquinas posible con un coste también reducido.

En la actualidad se está incrementando el número de dispositivos interconectados y hay que considerar la necesidad de establecer cierta seguridad en los accesos entre los dispositivos/máquinas/plataformas IoT así como reducir el riesgo de ataques a los dispositivos/sensores.

Por ello, en amplía))) surge la necesidad de ofrecer la plataforma OpenGate® en una infraestructura pública, con nuevas capacidades según vayan cambiando las necesidades de la IIoT, de cara a que pueda ser usada por clientes pequeños y medianos, sin necesidad de usar una infraestructura propia.

Objetivos y alcance del proyecto

El objetivo de este proyecto consiste en diseñar un gran sistema de monitorización, gestión y localización de dispositivos a través de la plataforma de IIoT OpenGate®, en el que las interconexiones dispositivo-plataforma IIoT están basadas en tecnologías de VPN que se apoye en una arquitectura altamente escalable.

El sistema diseñado ofrecerá, en una infraestructura de Cloud, la posibilidad de poder acceder a los dispositivos y a la plataforma de una forma segura y fiable desde “redes privadas” a través de las VPN.

Muchos dispositivos de hoy en día, tienen, para su gestión, sus propias primitivas para que desde las empresas que los ofrecen puedan realizar determinadas tareas. Para esto, resulta interesante que los dispositivos se conecten a redes VPN desde donde sólo se tenga a acceso a ellos desde la plataforma OpenGate® y desde servidores de las empresas que los gobiernan, para que nadie más pueda acceder a ellos.

Gracias a este tipo de conexiones se puede ofrecer:

- Seguridad en el acceso y las comunicaciones e información intercambiada entre los distintos elementos de la solución IoT.
- Reducir al máximo posible el *overhead* de comunicaciones generado por el cifrado del canal de datos.
- Facilitar la escalabilidad haciendo uso de protocolos no orientados a conexión UDP que limiten la cantidad de recursos necesarios en la infraestructura cloud para soportar cientos de miles/millones de dispositivos.
- *Multitenant*: Confidencialidad entre usuarios y aplicaciones de distintas organizaciones hacia las máquinas.

La arquitectura del sistema debe facilitar la escalabilidad, es decir, el sistema debe ser capaz de adaptarse al crecimiento continuo de usuarios sin que se pierda calidad de servicio.

Actualmente este servicio se está ofreciendo en el cloud de Amazon (*AWS*), por lo que habrá que tener en cuenta las ventajas que este entorno ofrece así como sus limitaciones.

Rol del alumno en el proyecto

El alumno que presenta este proyecto ha participado en él como miembro de la empresa amplía))). El trabajo se ha centrado en el diseño de la arquitectura software, la implantación de la solución en la infraestructura definida y la validación de las pruebas de funcionamiento para la interconexión de los dispositivos y la plataforma, concretamente en la instalación de un servicio de VPN para ofrecer una conectividad segura.

Aunque el alumno tenía su parte asignada dentro del proyecto, el trabajo en equipo ha sido fundamental y por lo tanto el alumno ha colaborado, bien sea con ideas, sugerencias o consejos, en más partes del mismo que las que son responsabilidad directa suya.

Contenido del documento

Este documento pretende ilustrar todo el proceso de diseño y desarrollo de la solución para cumplir los objetivos anteriormente indicados. Para ello se han definido partes diferenciadas que ayuden no solo a visualizar el proyecto en sí sino a entender las decisiones de diseño tomadas y la razón de las mismas.

Las partes de las que se compone el documento son:

1. ***Estudio de las tecnologías disponibles***: Este apartado lo compone el capítulo 2 "Estado del arte" y consiste en el análisis de las tecnologías disponibles de cara a la selección de las que vayan a usarse en el proyecto. Este apartado es importante ya que las elecciones tomada condicionará el diseño del sistema.
2. ***Análisis***: Este apartado lo compone el capítulo "Análisis de requisitos". El análisis es parte fundamental de cualquier proyecto, siendo un correcto análisis fundamental para el éxito del proyecto.
3. ***Diseño***: Es la parte principal de esta memoria. Lo compone el capítulo "Diseño del sistema". En este apartado se definirá el diseño tanto hardware como software del sistema.
4. ***Conclusiones***: Este apartado se explica en el capítulo "Conclusiones" y tiene como objetivo analizar el resultado del proyecto.

Capítulo 2. Estado del arte

El objetivo de este apartado es enumerar las distintas tecnologías que existen actualmente y decidir cuales de ellas, son las que más pueden adaptarse a las necesidades del proyecto.

A continuación analizaremos las distintas tecnologías existentes en la actualidad desde el punto de vista de necesidades del proyecto dependiendo a qué áreas se verán enfocadas:

- Protocolos *stack* IoT
- Acceso VPN
- Interfaces de provisión
- Tecnología de control de acceso
- Resolución de nombres

Tecnologías disponibles

Protocolos stack IoT

A continuación se van a enumerar y explicar algunos de los protocolos más extendidos en el mundo IoT en función de las distintas capas del *IoT stack*.

Capa de transporte

La mayoría de los escenarios de IoT son adecuados para **UDP**, ya que es un protocolo más rápido y mucho más ligero, al ser no orientado a conexión, que TCP. El tamaño del encabezado es mucho menor que en TCP, lo que hace que sea adecuado para un entorno restringido de dispositivos y sensores.

Capa de aplicación

Algunos de los protocolos más extendidos son: CoAP (Constrained Application Protocol), MQTT (Message Queue Telemetry Transport), XMPP (Extensible Messaging and Presence Protocol) y AMQP (Advanced Message Queuing Protocol).

COAP: protocolo especializado para el uso de nodos inalámbricos restringidos y limitados de baja potencia que pueden comunicarse de forma interactiva a través de Internet. Es un protocolo UDP y los paquetes son más

pequeños, pero mantiene la arquitectura cliente/servidor como HTTP pero realiza el intercambio de mensajes de forma asíncrona y soporta operaciones GET,PUT,POST y DELETE. Incluye otros requisitos como multicast, bajo *overhead* y simplicidad, que es importante para el mundo IoT y M2M.

MQTT: es un protocolo de mensajería de "peso ligero" basado en publicación-suscripción muy ligero y pensado para dispositivos alimentados con baterías ya que tiene un reducido consumo de energía. Utiliza una arquitectura pub-sub basada en intermediarios en el entorno IoT restringido. Entonces existen los siguientes agentes:

- **Publisher** : un sensor o dispositivo que publica información.
- **Suscriptor** : cualquier dispositivo (teléfono inteligente, dispositivo portátil, etc.) que esté interesada en suscribirse y recibir información que le interese.
- **Broker MQTT** : intermediario que recibe información del editor y la envía a los suscriptores.

Tiene 3 modos de funcionamiento:

- QoS0 (*At most once*): El modo menos fiable pero también el más rápido. La publicación se envía pero no se recibe confirmación.
- QoS1 (*At least once*): Se asegura que el mensaje es entregado al menos una vez, pero pueden recibirse duplicados.
- QoS2 (*Exactly once*): El modo más fiable y que más ancho de banda consume. Se controlan los duplicados para garantizar que el mensaje es entregado una única vez.

A pesar de sus características, puede suponer un problema para algunos dispositivos muy restrictivos, por el hecho de ir sobre TCP y de manejar nombres de *topics* largos. Esto se soluciona con la variante MQTT-SN que se puede definir como un MQTT para redes de sensores, que permite integrar dispositivos embebidos con cualquier red MQTT, sin perder las ventajas de una red de sensores como el bajo consumo y superando sus limitaciones como la pequeña capacidad de cómputo en los microcontroladores.

XMPP: protocolo de comunicación para middleware orientado a mensajes basado en XML . Permite el intercambio de datos estructurados pero extensibles entre dos o más entidades de red casi en tiempo real . El protocolo se ha utilizado también para sistemas de suscripción de publicación, señalización para VoIP, video, transferencia de archivos y juegos.

A diferencia de la mayoría de los protocolos de mensajería instantánea, XMPP se define en un estándar abierto y utiliza un enfoque de desarrollo y aplicación de sistemas abiertos , mediante el cual cualquiera puede implementar un servicio XMPP e interoperar con las implementaciones de otras organizaciones.

AMQP: protocolo estándar abierto en la capa de aplicaciones de un sistema de comunicación. Las características que definen al protocolo AMQP son la orientación a mensajes, encolamiento , enrutamiento (tanto punto-a-punto como publicación-subscripción), exactitud y seguridad. AMQP estipula el comportamiento tanto del servidor que provee los mensajes como del cliente de la mensajería hasta el punto de que las implementaciones de diferentes proveedores son verdaderamente interoperables. A diferencia de JMS, que solamente define una API, AMQP es un protocolo a nivel de cable. Un protocolo a nivel de cable es una descripción del formato de los datos que son enviados a través de la red como un flujo de octetos. En consecuencia, cualquier programa que pueda crear e interpretar mensajes conforme a este formato de datos puede interoperar con cualquier otra herramienta que cumpla con este protocolo, independientemente del lenguaje de implementación.

Acceso VPN

Antes se van a aclarar algunos conceptos que se consideran necesarios:

Red privada virtual (VPN)

Es una tecnología de red que se usa para conectar equipos, sensores, máquinas… a una red privada sobre una red pública o Internet. Permite el envío y recepción de datos sobre redes públicas como si fuera una red privada. A través de la red pública, la comunicación entre los dos extremos de la red privada se hace creando túneles virtuales entre esos dos puntos y usando sistemas de encriptación y autenticación que aseguren la confidencialidad e integridad de los datos transmitidos a través de esa red pública.

Para hacer esto posible de forma segura, es necesario garantizar:

- Autenticación y autorización. Deben verificar la identidad de los usuarios y restringir su acceso.
- Integridad de los datos. Garantizar que los datos enviados no han sido alterados.

- Confidencialidad. Cifrar los datos antes de ser enviados por la red pública.
- Autoría. El mensaje debe ir firmado, y quien lo firma no puede negar que lo envió.
- Control de acceso. Garantizar que los usuarios autenticados sólo tienen acceso a los datos a los que están autorizados.

La justificación para usar el acceso VPN en lugar de una red privada básicamente se reduce al coste y la viabilidad.

Los tipos más comunes de VPN son las *VPN de acceso remoto* y las *VPN de sitio a sitio*.

VPN de acceso remoto

En esta, los usuarios se conectan a la red interna desde sitios remotos utilizando Internet como vínculo de acceso. Una vez autenticados tienen nivel de acceso similar a la LAN de la empresa.

VPN punto a punto

El servidor VPN, conectado a Internet, acepta conexiones provenientes de Internet y establece el túnel VPN. Este esquema es el que se suele usar para conectar sucursales con la sede central de una organización.

Para ambos tipos de VPN hay 2 posibilidades de implantación, una hardware y otra software, pero, aunque las primera ofrece mayor rendimiento y facilidad de configuración, es menos flexible que la segunda. Esto unido al coste económico que conlleva nos lleva a centrarnos en una solución *software*.

Entre las tecnologías que hay disponibles están:

Point-To-Point Tunneling Protocol (PPTP)

Es un protocolo VPN y se basa en varios métodos de autenticación para proporcionar seguridad. El protocolo de cifrado estándar usado en Microsoft Point-to-Point Encryption (MPPE). Fue desarrollado para crear VPNs través de redes de acceso telefónico

Está disponible como estándar en casi todos los dispositivos y plataformas con capacidad para conectarse a VPN. Es fácil de configurar.

Por contra, no proporciona integridad de los datos que se envían, es decir, no se garantiza que los datos no se modificaron por el camino, ni

autenticación de origen de los datos, por lo que es extremadamente inseguro, y se ha visto comprometida por la Agencia de Seguridad Nacional de Estados Unidos (NSA).

Existen aplicaciones de donde se puede obtener fácilmente las claves de las sesiones y descifrar el tráfico de la VPN.

No recomendable.

Layer 2 Tunneling Protocol (L2TP)

Es un protocolo utilizado para soportar la VPN. Aunque define su propio protocolo de establecimiento de túneles, no provee de ningún servicio de encriptación por sí mismo. Pero presenta problemas a la hora de la autenticación:

- Sólo se autentican los puntos finales del túnel, por lo que podría dar lugar a suplantaciones de identidad.
- No se comprueba la integridad de los paquetes, por lo que podría realizarse ataques de denegación de servicio.
- No cifra el tráfico entre usuarios.
- No dispone de mecanismos para la generación de claves o refresco de las mismas.

Debido a esto, suele implementarse con el protocolo IPSec para cifrar los datos antes de la transmisión, proporcionando a los usuarios privacidad y seguridad**.**

Los sistemas operativos y los dispositivos modernos con capacidad de conectarse a VPN, tienen L2TP/IPSec integrado.

Las desventajas son que es más lento que otros protocolos debido a que se encapsula los datos dos veces y que utiliza un número limitado de puertos por lo que es fácilmente bloqueable por firewalls NAT.

OpenVPN

Es una solución de conectividad multiplataforma que ofrece conexiones punto-a-punto con validación jerárquica de usuarios y host conectados remotamente. Ofrece una combinación de seguridad a nivel empresarial, seguridad, facilidad de uso y riqueza de características.

Es una tecnología de código abierto que utiliza protocolos TLS y OpenSSL para proporcionar una solución confiable y sólida. Es altamente configurable, y aunque no es soportado nativamente por las plataformas, existe software de terceros.

El cifrado de OpenVPN comprende dos partes: encriptación de canal de datos y cifrado de canal de control. El primero es para proteger la información y el segundo protege la conexión entre el servidor VPN y el cliente. Al utilizar OpenSSL para proporcionar encriptación tanto de los canales de datos como de los de control, se pueden usar todos los cifrados disponibles en la librería.

No ha sido comprometido por la NSA. Su configuración es algo más complicada en comparación con L2TP/IPSec y PPTP.

Secure Socket Tunneling Protocol (SSTP)

SSTProtocol es un protocolo principalmente para Windows, aunque ahora está disponible para Linux y MAC OS X. Utiliza SSL 3.0, por lo que ofrece ventajas similares a OpenVPN.

Pero SSTP sólo admite la autenticación del usuario, es decir, no admite autenticación ni de dispositivo ni de equipo. Otra desventaja es que SSL v3.0 es vulnerable a ”el ataque de POODLE”, que es un problema en el esquema de cifrado CBC. El ser un estándar propiedad de Microsoft, es decir, que el código no está abierto, no inspira confianza, ya que se habla de puertas traseras en el S.O. Windows.

Internet Key Exchange version 2 (IKEv2)

IKEv2 es únicamente un protocolo de tunelización fue desarrollado por Microsoft y Cisco. A pesar de esto, hay versiones compatibles para Linux y otros S.O. y muchas de estas iteraciones son de código abierto.

Como pasa con L2TP, sólo se convierte en un protocolo VPN cuando se empareja con IPSec. Es óptima para restablecer automáticamente una conexión VPN al perder temporalmente las conexiones a Internet.

Se considera tan buena, o más, a L2TP/IPSec en seguridad, rendimiento, estabilidad y capacidad para establecer una conexión.

Interfaces de provisión

De cara a dar servicio a los clientes es necesario que el equipo que lleva el mantenimiento y soporte del mismo tenga una API para gestionar estas redes. El punto de vista en el que nos centraremos será la facilidad de uso y de gestión.

SOAP (Simple Object Access Protocol)

Es un protocolo utilizado en interacciones de servicios web por medio de intercambio de datos XML. Los mensajes SOAP habitualmente se envían

sobre HTTP, aunque también puede usar los protocolos de envío FTP, POP3, TCP, Colas de mensajería (JMS, MQ, etc).

Este protocolo es uno de los mejores, por no decir el mejor, para la comunicación servidor-servidor ya que es muy robusto, permite añadir metadatos a través de atributos, espacios de nombres evitando ambigüedad... Pero este formato es muy pesado tanto en procesamiento como en parseado de los XML a un árbol DOM. A diferencia de los JSON, los XML poseen métodos de validación muy potentes y muy utilizados.

REST

Es un protocolo cliente/servidor sin estado muy flexible que transporta datos por medio del protocolo HTTP para obtener datos o generar operaciones sobre los datos. Cada petición contiene la información necesaria para ejecutarla por lo que no es necesario, que ni el cliente ni el servidor recuerden ningún estado anterior. Permite transmitir casi cualquier tipo de datos, ya que estos están definidos en el Header Content-Type, lo que nos permite mandar, XML, JSON, Binarios, Text, etc.

Permite utilizar los diversos métodos que proporciona HTTP para comunicarse (GET, POST, PUT, DELETE) y utiliza los códigos de respuesta nativos de HTTP (404, 200, 204, 409).

Es la propia URI el identificador único de cada recurso de ese sistema REST. La URI nos facilita acceder a la información para su modificación o borrado.

Aunque se puede enviar gran variedad de tipos de datos, la mayoría manda JSON ya que es interpretado de forma natural por JavaScripts, lo que ha hecho que frameworks como Angular y React se aprovechen al máximo. Los formularios de HTML pueden ser apuntados a los servicios REST sin ningún problema.

Mediante el uso de este protocolo sin estado y operaciones estándar, los sistemas REST buscan fiabilidad, rendimiento rápido y capacidad de crecer reutilizando componentes que se pueden gestionar y actualizar sin afectar al sistema en su conjunto.

Tecnología de control de acceso

En este punto se analizarán las tecnologías dirigidas a conseguir AAA (Authentication, Authorization and Accounting). Antes de seguir se analizará cada uno de los 3 conceptos:

- **Autenticación**: proceso por el que una entidad demuestra ser quien es ante otra. Normalmente la primera entidad es un cliente y la segunda un

servidor. Esto se consigue mediante un dato que identifica al cliente, por ejemplo nombre de usuario, y la demostración de estar en poseer las credenciales que permitan comprobarla, como por ejemplo certificados, contraseñas, etc.

- **Autorización**: proceso en el que se conceden privilegios específicos a un cliente basándose en su identidad (autenticación), el estado actual del sistema y los privilegios que solicita. La mayor parte de estos privilegios consisten en el uso de un determinado servicio. Algunos ejemplos son: filtrado de direcciones IP, asignación de direcciones, de rutas, asignación de ancho de banda.
- **Contabilización**: proceso que realiza un registro de los eventos de forma secuencial que permite determinar las acciones realizadas por una entidad activa en una red. La información recolectada puede usarse para la facturación, administración, planificación u otros propósitos.

De la familia de protocolos que ofrece AAA, RADIUS (Remote Authentication Dial-In User Server) es el más conocido y extendido, funciona como cliente-servidor. Destaca sobre todo por ofrecer un mecanismo de seguridad, flexibilidad, capacidad de expansión y una administración simplificada de las credenciales de acceso a un recurso de red.

FreeRadius

FreeRADIUS es el servidor que implementa el protocolo RADIUS. Es de código abierto y es el más ampliamente usado en el mundo.

Soporta los protocolos de autenticación más comunes y garantiza la compatibilidad con una amplia gama de dispositivos NAS.

La versión principal de v2.2.x ha entrado en la fase final de su ciclo de vida y ahora solo se desarrollan correcciones de seguridad ya que está muy extendido su uso.

Su desarrollo se inició en agosto de 1999 por Alan DeKok y Miquel van Smoorenburg. Se inició para crear un nuevo servidor RADIUS basado en un diseño modular para fomentar la participación activa de la comunidad de software libre.

OpenRadius

Se trata también de un servidor que implementa el protocolo Radius. Entre sus características más destacables están:

- Libre de usar, modificar y redistribuir bajo los términos de la Licencia Pública General de GNU.

- Ofrece la posibilidad de obtener secretos compartidos, información de autenticación, políticas y perfiles de usuario de cualquier fuente de datos externa disponible.
- Compatibilidad con bases de datos de contraseñas de Unix, incluidos NIS / NIS +, archivos ASCII estilo Livingston, directorios LDAP y bases de datos SQL listas para usar.
- Esquemas de autenticación personalizable y políticas de seguridad. Lo que permite especificar cómo el servidor toma sus decisiones, en función de cualquier combinación de información disponible interna y externamente.
- Interfaz de módulo simple, escalable y totalmente documentada. Los módulos pueden suministrar datos como la información del usuario, y también pueden almacenar datos tales como el registro y la contabilidad. Los módulos se vuelven a utilizar en lugar de ejecutarse para cada solicitud, lo que preserva la capacidad de escalado al tiempo que permite el uso de scripts externos simples. Todos los subprocesos externos son supervisados por el proceso del servidor principal y se reinician automáticamente si fallan.
- Diccionario extremadamente flexible que se puede hacer para admitir cualquier tipo de atributo no estándar específico del proveedor.
- Se une a una o varias direcciones IP / tarjetas de red y escucha en múltiples puertos.

Resolución de nombres

Uno de los objetivos del proyecto es el de ofrecer un sistema de acceso sencillo y claro a los dispositivos de las distintas organizaciones. Para ello es vital contar en la arquitectura con un sistema que permita hacer la traducción de los nombres de los dispositivos.

DNS (Domain Name System)

Se trata del protocolo más usado en Internet para traducir nombres de servicios en identificadores binarios asociados a los equipos conectados a la red. El objetivo final es el de localizar y direccionar los equipos nivel mundial.

El sistema usa una base de datos distribuida y jerárquica que almacena información asociada a nombres de dominio en redes como Internet.

El protocolo permite asociar diferentes tipos de información a cada nombre, pero el uso más común es el de la traducción de nombres a direcciones IP y la localización de los servidores de correo de cada dominio.

Se utilizan tres componentes principales:

- Los clientes fase 1: Se trata de los clientes DNS que corren en los equipos finales y son los que hacen las peticiones de traducción
- Los servidores DNS: son los que contestan a las peticiones anteriores. Si no tienen la información, son capaces de reenviar la petición a otro servidor.
- Las zonas de autoridad. Son las partes del espacio de nombres de dominio sobre la que es responsable un servidor DNS, que puede tener autoridad sobre varias zonas.

GNUNet – GNS

Se trata de un conjunto de tecnologías que permite a los usuarios el uso de Internet en la modalidad P2P sin la necesidad de cualquier tipo organización o sistema que tenga un rol determinado. Hay un componente de GNUnet llamado GNS que constituye un sistema alternativo de nombrado a DNS. Es de carácter seguro y descentralizado.

Su diseño proporciona:

- Resistencia a la censura
- Privacidad a la hora de hacer las consultas
- Resolución segura de nombres
- Compatibilidad con DNS

A diferencia de DNS no depende de zonas o autoridades centrales. En su lugar, cualquier usuario administra su propia raíz y puede crear asignaciones de valores de nombres arbitrarios. Además, los usuarios pueden delegar la resolución a las zonas de otros usuarios al igual que los registros DNS NS. Las zonas se identifican de manera única a través de claves públicas y los registros de recursos se firman usando la clave pública correspondiente.

Namecoin

Permite registrar, sin censura alguna, dominios .bit independiente de la ICAAN (organismo que controla los nombres de dominio de nivel superior). *Namecoin* es un sistema de registro y transferencia de pares clave-valor basada en la tecnología de *bitcoin*. Los registros *namecoin* son

una tupla de clave-valor . Las claves son la rutas con el nombre de dominio precediendo al nombre del registro.

Bitcoin libera dinero: *Namecoin* libera DNS, identidades y otras tecnologías.

Usar un registro *namecoin* tiene un coste y además los registros expiran (200 días) salvo que se renueven. Los *namecoins* usados para cada registros quedan marcados como usados y no pueden usarse para pagos y es un *namecoin* puede usarse también como cripto-moneda.

Los nombres de *namecoin* son difíciles de censurar y las búsquedas no generan tráfico de red.

Tor / Onion

Tor es un software libre y una red abierta que permite defenderse contra el análisis del tráfico ofreciendo anonimato. Esto se logra gracias al encaminamiento del tráfico a través de varios nodos sucesivos de la red Tor. Permite también el anonimato y la censura ya que permite navegar sin tener que proporcionar al ISP el nombre del dominio y la dirección IP del servidor al que se está intentando acceder.

Proporciona un mecanismo, los servicios ocultos, que se pueden usar para ocultar el destino. Un servicio oculto utiliza un identificador Tor, que habilita el enrutamiento (seguro) en la red Tor. Para permitir su uso por aplicaciones tradicionales, el identificador puede colocarse en el dominio de nivel superior (no delegado) .onion. como por ejemplo https://kgquuvig3tvxmzna.onion/

Los identificadores bajo el .onion se eligen al azar (cada uno es el condensado de una clave criptográfica), pero hay software disponible que se puede usar para probar claves sistemáticamente hasta que obtenga un nombre que se asemeje a lo que desea.

El acceso a este servicio, que oculta el sitio real al que se accede, requiere un software especial para el cliente. La técnica más simple (y, por lo tanto, la más segura) es descargar navegadores web que tengan embebido el software de Tor.

Tecnologías seleccionadas

Una vez analizadas las tecnologías disponibles, llega el momento de decidir cuál de ellas se va a utilizar durante el proyecto. Para tomar esta decisión se tendrán en cuenta los siguientes factores:

- **Capacidades técnicas**: Es necesario tener en cuenta lo que nos ofrece cada una de las tecnologías y sus características.
- **Condiciones de funcionamiento**: La tecnología debe de adaptarse al entorno en el que va a aplicarse.
- **Rendimiento económico**: la tecnología utilizada debe tener un coste adecuado a su funcionalidad

Protocolos stack IoT

Ante el escenario que nos encontramos vamos a seleccionar los protocolos enumerados en el punto anterior.

Capa de transporte: udp, ya que su ligereza y rapidez es ideal a la hora de seleccionar las capacidades necesarias de los dispositivos.

Capa de aplicación: Debido a la compatibilidad con la plataforma OpenGate se han seleccionado CoAP y MQTT. OpenGate ya que tiene conectores que procesan este tipo de mensajes.

Acceso VPN

Por todo ello se ha seleccionado finalmente la tecnología **OpenVPN**. Los motivos han sido:

- Gran variedad de tipos de cifrados. Al usar la biblioteca OpenSSL para que haga el trabajo de encriptación y autenticación, permite utilizar todos los cifrados disponibles en él.
- Distintas maneras para autenticar. Autenticación basada en usuario/contraseña, ofrecer claves previamente compartidas, y basada en certificados.
- Puede ejecutarse a través de UDP o TCP.
- Soporte IPv6 como protocolo de red

- Protección de los usuarios remotos. Una vez que OpenVPN ha establecido un túnel el firewall de la organización protegerá al usuario aún cuando no es un equipo de la red local. Además, sólo un puerto de red será abierto hacia la red local por el remoto asegurando protección en ambos sentidos.
- Conexiones OpenVPN pueden ser realizadas a través de casi cualquier firewall.
- Las interfaces virtuales permiten la implementación de reglas de firewall muy específicas.
- Todos los conceptos de reglas, restricciones, reenvío y NAT10 pueden ser usados en túneles OpenVPN.
- Alta flexibilidad y posibilidades de extensión mediante scripting, ya que ofrece numerosos puntos para poder ejecutar scripts durante su arranque.
- Soporte transparente para IPs dinámicas.
- NO presenta ningún problema con NAT.
- Instalación sencilla en cualquier plataforma. Incluso ofrece la posibilidad de “enviar” opciones de configuración de red a los clientes conectados, como dirección IP, comandos de enrutamiento..

Interfaces de provisión

Para la implementación del servicio de provisión de VPNs y de los usuarios que se conectarán, teniendo en cuenta los factores indicados anteriormente, se ha seleccionado el uso de tecnología **REST**.
Los motivos han sido:

- El servicio que se va ofrecer será sencillo y no requiere estado.
- Las acciones que se van a realizar y los objetos de provisión encajan con los métodos HTTP.
- Soap es más lento y complejo, y no se usarán las funcionalidades adicionales que ofrece.

Tecnología de control de acceso

Se ha seleccionado FreeRadius por la estabilidad del software, que ofrece características de escalabilidad, rendimiento, facilidad de uso, etc. Un factor también importante es el coste, al ser código abierto no es necesario gestionar licencias. También ha sido determinante los conocimientos que hay en la empresa amplía))) así como la compatibilidad que ofrece con otros proyectos de la casa.

Resolución de nombres

Se ha seleccionado como tecnología de resolución de nombres, la tecnología DNS, fundamentalmente por su uso extendido por Internet y, por tanto, la familiaridad que sentirán los clientes del sistema cuando la usen. Hay que recordar que la arquitectura se basará en redes vpn, por tanto ya se gozará de una seguridad y una privacidad. Además, los dispositivos y los que intenten acceder a ellos serán de la misma organización, es decir, serán intranets.

Capítulo 3. Descripción del problema

Como se ha explicado en la introducción, el objetivo del proyecto es que la plataforma OpenGate® ofrezca acceso a los dispositivos y a la plataforma de una forma segura y fiable a través de VPNs ofreciendo el servicio en una infraestructura pública para que pueda ser usada por varios clientes simultáneamente.

Las empresas del mundo IIoT tienen miles de dispositivos que deben ser gestionados y monitorizados para lograr los objetivos finales del negocio particular de cada empresa. Ejemplos hay varios, veamos algunos para contextualizar el entorno:

- Las empresas eléctricas lo que que necesitan de sus dispositivos IOT (contadores industriales y contadores domésticos) es llevar a cabo las mediciones de consumo o generación de energía para poder llevar a cabo la facturación y la planificación de sus recursos.
- Las compañías de agua necesitan algo parecido al caso anterior, controlar el consumo de agua de las diferentes redes para poder llevar a cabo tareas de planificación, gestión del consumo, facturación, etc.
- El sector alimentario y las máquinas de *vending* son también otro ejemplo de sector IIoT en el que es necesario conocer el estado de cada punto de venta y poder enviar personal a reponer las máquinas de los productos agotados o que están a punto de hacerlo.

Como vemos, dentro del mundo IIoT, en el que se manejan diferentes dominios, se tienen problemas comunes.

OpenGate®, como plataforma IIoT ofrece un solución integral a los problemas anteriores pero tiene algunas limitaciones:

- Para cada dominio anterior, se requiere un entorno aislado si es que no se quieren enviar los datos a través de Internet.
- Si se desea seguridad en las comunicaciones, es necesario hacer uso de protocolos de comunicación con un coste añadido en el cifrado de las comunicaciones.
- El direccionamiento de los dispositivos IoT de cada empresa debe ser diferente para evitar problemas de solape de red.

Las limitaciones anteriores, si bien son salvables técnicamente, generan una barrera de entrada en algunos clientes potenciales de OpenGate.

El presente proyecto busca ofrecer una solución al problema anterior:

- Permitirá a las empresas usar el servicio público de OpenGate sin necesidad de contratar una instancia privada.
- Logrará que las comunicaciones entre los dispositivos IoT de cada compañía y OpenGate se lleven a cabo sin necesidad de utilizar protocolos de comunicación cifrado.
- Abrirá la posibilidad de tener direccionamiento de red que podrá solaparse entre diferentes organizaciones/empresas.

Para ello se apoyará en las soluciones de VPN.

Como se verá más adelante, parte de la funcionalidad requerida quedará soportada por la especialización realizada sobre OpenVPN. Una de sus propiedades, que afecta directamente al problema que se requiere resolver, es la capacidad de alta disponibilidad (HA). Dada la tipología de clientes: dispositivos del mundo IIoT y organizaciones del mundo IIoT, es esencial que las soluciones aportadas sean robustas y soporten escenarios críticos como puede ser la caída de los servidores que prestan el servicio. Un cliente como una empresa del sector eléctrico no puede permitirse quedarse sin servicio durante unas horas por un problema en algún servidor. En capítulos posteriores se entrará en mayor detalle: posibles configuraciones, pros y contras, impacto en el coste del servicio, etc.

Capítulo 4. Análisis de requisitos

Introducción

Una vez que se han establecido las necesidades básicas del proyecto y la tecnología de la que se hará uso, hay que definir de forma detallada que debe cumplir la solución que se va a diseñar. Para ello se definirá una serie de requisitos que debe cumplir el proyecto para considerar que cubre las necesidades del problema. Los requisitos los dividiremos en tres apartados:

- **Requisitos funcionales**: Son aquellos requisitos que indican cómo debe comportarse el sistema. Indican qué tareas o funciones debe cumplir la solución para que se considere apropiada.
- **Requisitos no funcionales**: Son los requisitos que hacen referencia al diseño o la implementación del sistema.
- **Requisitos de proyecto**: Son los requisitos que afectan al proyecto en sí en vez de a la solución software obtenida.

Cada uno de los requisitos se describe con la siguiente plantilla:

REQ_TIPO_NUM

Título	
Descripción	
Prioridad	
Notas	

Tabla 1. Plantilla de definición de requisitos

La descripción de cada uno de los campos es:

- ***REQ_TIPO_NUM***: Identificador del requisito. TIPO, tipo de requisito y NUM, id de requisito.
- **Título**: Título del requisito, este debe ser una solo sentencia corta que describa de forma general el requisito.
- **Descripción**: Una descripción más detallada del requisito.
- **Prioridad**: Nivel de importancia que se da a este requisito.
- **Notas**: Cualquier información importante que no se corresponda con otros apartados.

Requisitos funcionales

Los requisitos funcionales de este proyecto especifican las necesidades que debe cumplir la interconexión entre los dispositivos y la plataforma para que sea considerada segura y escalable.

REQ_FUNC_001

Título	Acceso seguro de dispositivos
Descripción	La plataforma deberá disponer de un mecanismo seguro para que todos los dispositivos, de todos los clientes, puedan identificarse de manera fiable e inequívoca y usar los recursos del sistemas y poder ser gestionables por el sistema. Este acceso se deberá basar en VPN.
Prioridad	Alta
Notas	

REQ_FUNC_002

Título	Conectividad sólo con los elementos de su propia red
Descripción	La plataforma deberá disponer de un mecanismo para asegurar que sólo los elementos de la misma red podrán conectarse a los dispositivos de esa red.
Prioridad	Alta
Notas	Se deben cumplir varias propiedades: • Dispositivos del mismo cliente, podrán tener conectividad entre ellos una vez esté conectados por VPN. • Dispositivos, de diferentes clientes, no tendrá conectividad entre ellos cuando estén conectados por VPN.

REQ_FUNC_003

Título	Comunicaciones UDP
Descripción	La plataforma deberá facilitar la comunicación entre los elementos de una misma red VPN a través del protocolo UDP.
Prioridad	Media
Notas	Una vez exista el canal seguro de comunicaciones que proporciona la VPN, se debe priorizar/facilitar el tráfico UDP

	para reducir el consumo de datos y mejorar la velocidad de las comunicaciones.

REQ_FUNC_004

Título	DNS por VPN
Descripción	La plataforma deberá disponer de mecanismos que hagan posible la identificación de los dispositivos a través de su hostname o identificador. Además: • Se deberán soportar direcciones IPv4. • No se tendrá una jerarquía, es decir, peticiones que no puedan resolverse en local no se elevarán a servidores DNS públicos
Prioridad	Media
Notas	El soporte a direcciones IPv6 no es necesario en una primera implantación del sistema pero se valoraría su soporte en el futuro. El diseño de la solución debería tener en cuenta esto.

REQ_FUNC_005

Título	Interfaz para gestionar las provisiones de VPN
Descripción	Se deberá ofrecer una interfaz/API para poder gestionar las VPN en el sistema. Esta interfaz deberá permitir: • El alta de nuevas VPN asociadas a un cliente determinado. • La baja de una VPN existente. • La lectura de una VPN existente. No se requiere tener capacidad de modificar una VPN y en caso necesario, se procederá al borrado de la VPN existente y a la creación con los nuevos datos. Cualquier actividad en la API debe quedar registrada en archivos de logs para poder consultarse a posteriori. Las entradas en estos archivos deben registrar la actividad realizada, la hora a la que se realizó y el usuario que generó

	la actividad.
Prioridad	Media
Notas	La interfaz debe tener en cuenta situaciones de error como el alta de una VPN existente, el borrado de una VPN que no existe, etc. y en general todos los posibles errores que puedan darse al hacer uso de la interfaz. El diseño debería tener en cuenta que en el futuro se pueden querer nuevos servicios, como por ejemplo, el alta masiva, y por tanto debe ser una API modular y flexible.

REQ_FUNC_006

Título	Interfaz para gestionar las provisiones de usuarios en las VPN del sistema
Descripción	Se deberá ofrecer una interfaz/API para poder gestionar los usuarios en las VPN del sistema. Esta interfaz deberá permitir: • Crear un nuevo usuario en una VPN existente. • La baja de un usuario. • La lectura de un usuario • La modificación de un usuario. Cualquier actividad en la API debe quedar registrada en archivos de logs para poder consultarse a posteriori. Las entradas en estos archivos deben registrar la actividad realizada, la hora a la que se realizó y el usuario que generó la actividad.
Prioridad	Media
Notas	La interfaz debe tener en cuenta situaciones de error como el alta de un usuario existente, el borrado de un usuario que no existe o está conectado, etc. y en general todos los posibles errores que puedan darse al hacer uso de la interfaz. No se podrá borrar un usuario conectado sin antes llevar a cabo la desconexión. El borrado de un usuario no debe provocar el borrado del histórico de conexiones.

	El diseño debería tener en cuenta que en el futuro se pueden querer nuevos servicios, como por ejemplo, el alta masiva, y por tanto debe ser una API modular y flexible.

REQ_FUNC_007

Título	Notificación de conexiones y desconexiones de dispositivos
Descripción	El sistema deberá notificar a OpenGate mediante un paquete de Radius Accounting de tipo Start de la conexión de un dispositivos a la VPN. También, cuando un dispositivo se desconecte de su conexión VPN, se deberá notificar a OpenGate mediante un paquete de Radius Accounting de tipo Stop de la desconexión.
Prioridad	Alta
Notas	De este modo, se podrá visualizar en la consola web de OpenGate los dispositivos conectados y desconectados y se podrán plantear las diferentes operaciones sobre los dispositivos.

REQ_FUNC_008

Título	Contabilización del tráfico generado por los dispositivos
Descripción	Aprovechando la capacidad que ofrece el protocolo Radius, se usarán los atributos destinados a la información de facturación (paquetes transferidos y tiempo de sesión) para notificar a OpenGate del gasto de datos hecho por un dispositivo.
Prioridad	Alta
Notas	El tiempo de duración de la sesión se enviará en segundos y el el número de paquetes transferidos se desglosará en paquetes enviados y recibidos.

Requisitos no funcionales

Estos requisitos especifican necesidades no funcionales del sistema, esto es, aquellos puntos importantes del diseño que no están cubiertos en punto anterior. Esto incluiría características como las prestaciones, la fiabilidad o la mantenibilidad.

REQ_NO_FUNC_001

Título	El sistema de comunicaciones de la solución no debe interferir en la solución actual de comunicaciones
Descripción	No deben de producirse interferencias entre ambos sistemas de comunicación y deben funcionar ambos correctamente.
Prioridad	Alta
Notas	De cara al sistema ya existente, el sistema a desarrollarse debe ser transparente en términos de comunicaciones. Las nuevas funcionalidades y el apoyo que tienen en la redes, serán añadidos a la arquitectura actual.

REQ_NO_FUNC_002

Título	El sistema de debe seguir recibiendo y procesando la información como hace actualmente
Descripción	No deben de perderse información para poder ser procesada correctamente.
Prioridad	Alta
Notas	El incremento de nuevas rutas, conexiones OpenVPN, clientes conectados y accesibles no afectará al tráfico existente.

REQ_NO_FUNC_003

Título	El sistema debe ser escalable
Descripción	El sistema permitirá aumentar el número de VPN según las necesidades
Prioridad	Alta

Notas	

REQ_NO_FUNC_004

Título	Rendimiento
Descripción	El sistema debe soportar el volumen de los dispositivos provisionados en producción.
Prioridad	Media
Notas	El número de dispositivos de producción es de XXX y el perfil de dispositivo medio genera XXX mensajes por segundo.

REQ_NO_FUNC_005

Título	Las interfaces humanas del sistema serán sencillas de usar
Descripción	Todas las interfaces que ofrezca el sistema: • Para provisiones de vpn • Para provisiones de clientes • De auditoría • De operaciones deben estar diseñadas y pensadas para operadores con poca experiencia por lo que se requiere que sean sencillas y protejan a los usuarios de fallos.
Prioridad	Media
Notas	

REQ_NO_FUNC_006

Título	Seguridad en las VPN
Descripción	Dada la criticidad de los datos a gestionarse y a la inclusión de un punto de acceso a la redes de los diferentes clientes que supondrá el sistema a desarrollar, se requiere que la configuración de las VPN sea tal que ofrezca una alta seguridad y que se evite en la medida de lo posible cualquier tipo de ataques.

Prioridad	Media
Notas	La configuración de cada VPN y de cada cliente, será como mínimo de: • Certificados RSA de 2048 • Las sesiones no durarán más de 10 minutos • Protección ante intentos fallidos de acceso

Requisitos de proyecto/sistema

Estos requisitos especifican las necesidades del proyecto en sí, como la documentación o las pruebas a realizar.

REQ_PROY_001

Título	Documentación completa del proyecto
Descripción	Se debe generar documentación del proyecto.
Prioridad	Alta
Notas	Al menos se deberán generar documentos que permitan: • Conocer el diseño de la arquitectura. • Administrar el sistema. • Operar en el sistema. Cualquier documentación adicional deberá ir en la línea de los puntos anteriores.

REQ_PROY_002

Título	Confidencialidad de los datos
Descripción	Todos los datos de las diferentes organizaciones y clientes del sistema serán confidenciales
Prioridad	Alta
Notas	Cualquier tipo de actividad registrada será de carácter confidencial y no podrá consumirse para ninguna actividad no relacionada con el sistema. Además, los datos de las diferentes organizaciones se almacenarán de modo independiente. Clientes del sistema sólo podrán consultar los datos asociados a ellos o sus dispositivos.

REQ_SIST_001

Título	El software de la plataforma correrá sobre un sistema GNU/Linux
Descripción	El software para el proyecto debe correr al menos en una máquina con el sistema operativo GNU/Linux
Prioridad	Alta
Notas	El sistema diseñado podrá funcionar en modo *single-instance*, es decir, no será necesaria una arquitectura en clúster para disfrutar de la funcionalidad. Además, el sistema debe poder ejecutarse en servidores con sistema operativo Linux, como mínimo en sistemas operativos de la familia *RedHat*.

REQ_SIST_002

Título	Resistencia a fallos de entorno
Descripción	El software del proyecto debe reiniciarse en caso de caída
Prioridad	Alta
Notas	Cuando el software del proyecto sufra una caída no planeada o la máquina en la que se ejecuta se reinicie, el software volverá a arrancar automáticamente.

REQ_SIST_003

Título	Auditoría
Descripción	El sistema debe ser auditable
Prioridad	Alta
Notas	Dada la criticidad del sistema y su naturaleza, en la que se producen accesos y se dan comunicaciones securizadas entre diferentes localizaciones, es necesario: • Poder acceder a los registros de conexiones, desconexiones e intentos de los mismos. • Se debe también almacenar el tiempo que duran las

	sesiones de los diferentes dispositivos. • En el caso de asignación de direcciones IP dinámicas, se almacenará un histórico de todas las direcciones asignadas a cada uno de los clientes.

REQ_SIST_004

Título	Rotado de logs
Descripción	Todos los archivos de logs generados en la actividad del sistema deberán rotar de modo automático.
Prioridad	Alta
Notas	Existirá un período de retención de logs no inferior a seis meses. Los archivos se deberán guardar en formato comprimido y se tendrá acceso a los mismos en base a fecha.

REQ_SIST_005

Título	El software del proyecto debe ofrecer HA
Descripción	El sistema debe estar diseñado e implementado de modo que se garantice el servicio que ofrecerá, independientemente de los problemas que puedan existir en la infraestructura como caídas de servidores, pérdidas de conectividad, etc. Cuando una de esas situaciones se ponga de manifiesto, tan pronto se restaure la situación de normalidad el servicio se prestará con normalidad. El módulo crítico y sobre el que se requiere este comportamiento de alta disponibilidad es el de OpenVPN.
Prioridad	Alta
Notas	A través de OpenVPN se ofrecerá la conectividad a los dispositivos y la comunicación con OpenGate, de ahí que sea prioritario.

REQ_SIST_006

Título	Copias de seguridad
Descripción	Se dispondrá de un sistema de copia de seguridad que garantice que no se perderán datos en caso de fallo catastrófico que obligue a la reinstalación del software. Se deberá hacer una copia de seguridad diaria, de tipo incremental y una copia de seguridad quincenal que deberá ser completa. Las copias de seguridad deberán incluir todo lo necesario: configuración de los distintos módulos, *backups* de las bases de datos, etc. Las copias de seguridad deberán estar encriptadas.
Prioridad	Alta
Notas	Las copias deberían utilizar algún sistema como Bacula o simular.

Capítulo 5. Diseño del sistema

Introducción al diseño

En este capítulo describiremos el diseño propuesto del sistema de VPNs del proyecto. La arquitectura de este sistema se ha diseñado teniendo en cuenta los objetivos que se han marcado así como los requisitos expuestos anteriormente. Para facilitar la comprensión del diseño y permitir la consulta de partes específicas del mismo, se ha dividido este capítulo en tres partes principales:

- **Arquitectura de sistemas y de comunicaciones**: En este apartado se explicarán las diversas partes que componen el sistema y cómo se realiza la comunicación entre ellas.
- **Arquitectura funcional**: En este apartado se explica la funcionalidad del sistema.

Arquitectura de sistemas y de comunicaciones

Para poder ofrecer una funcionalidad que abarque las necesidades del proyecto es necesario diseñar una arquitectura de sistemas que permita la conectividad de los dispositivos con la plataforma así como el funcionamiento “habitual” de OpenGate ®, es decir, la recopilación de información, gestión de dispositivos, supervisión de parámetros de dispositivos, comprobación de su estado, etc.

El diseño propuesto para esta arquitectura es el siguiente que puede dividirse en 6 partes principales:

- Plataforma OpenGate®.
- Sistema de almacenamiento (Oracle, MongoDB y MySQL)
- OpenVPN
- MaraDNS
- FreeRadius
- OpenGate UX

En la siguiente imagen puede apreciarse la arquitectura de la solución propuesta así como las relaciones de los distintos componentes.

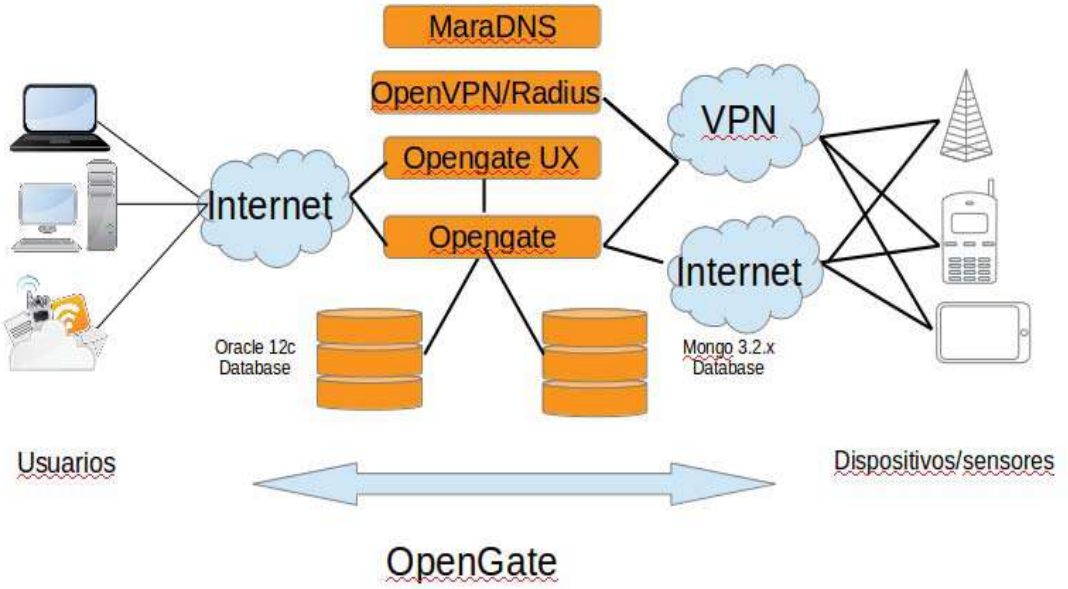

Figura 5.1 : Arquitectura propuesta

Plataforma OpenGate ®

Este componente es uno de los principales del sistema, ya que gracias a él se podrá gestionar y administrar los dispositivos, ya envíen la información a través de la VPN o a través de Internet.

Las tareas que debe realizar este componente, centrándonos en el proyecto, son las siguientes:

- Controlará el envío de mensajes desde los sensores hacia el sistema OpenGate ® central, encargándose de que los mensajes se reciban correctamente.
- Comunicación entre sistema central y componentes: El componente también controla el envío de mensajes desde el sistema central hacia los sensores.
- Comprobación de estado: Adicionalmente el sistema proporcionará al operador un mecanismo para comprobar el estado de conexión de los distintos componentes, sirviendo de ayuda para diagnosticar incidencias en el sistema.

Este componente se instalará en un sistema operativo GNU/Linux con el software OpenGate® de amplía))) instalado. Este software arranca a través

de procesos wrapper que aseguran que los componentes, a través de sus procesos Java 8 están en ejecución. Además de este servidor, será necesario una base de datos para almacenar la información, que actualmente es Oracle (este punto se detalla más adelante).

Para el correcto funcionamiento, es necesario que todos los “componentes” que vayan a enviar y/o recibir mensajes estén dados de alta en OpenGate. Para que la comunicación se pueda establecer se asocian todos los componentes a un canal de comunicación, que es una entidad interna de la plataforma usada para diferenciar entre diversos flujos independientes de mensajes.

OpenGate proporciona un conjunto de interfaces que permiten el uso y la integración con el mundo real. Destacaremos la Interfaz Norte y el Sur.

- *Interfaz norte*. Centrada en proporcionar a los usuarios y aplicaciones de Back-office el acceso a todas las características e información expuestas por la plataforma.
- *Interfaz sur.* Enfocada en la integración con todos los dispositivos u otros sistemas externos, para recopilar los datos IoT/IIoT e interactuar permitiendo que la plataforma realice la ejecución remota de operaciones y diagnósticos.

A través de la interfaz sur se ofrecen distintas capas de transporte para poder conectarse a la plataforma y poder enviar información. Estos tipos son :

- Conector radius: usa el protocolo estándar radius.
- Conector webservice: usa el protocolo estándar HTTP/HTTPS.
- Conector websocket: usa el protocolo estándar WS/WSS.
- Conector mqtt: usa el protocolo MQTT/MQTTS.
- Conector coap: usa el protocolo CoAP.
- Conector sigfox: usa el protocolo estándar HTTP.

Sistema de almacenamiento

En la solución se van a usar distintos gestores de bases de datos: Oracle 12c, Mongo 3.2 y MySQL 5.

1. **Oracle12c**

   OpenGate almacena en este gestor de base de datos la información relacional de la plataforma, es decir, las organizaciones, dispositivos, usuarios que han sido dados de alta en la plataforma así como sus relaciones. También se guardan las reglas de alarmas que se configuran para cada organización, así como las alarmas que se han generado hasta el momento.

2. **MongoDB 3.2**

   OpenGate® almacena en este gestor de base de datos los históricos de la información recolectada, el último dato de cada tipo de información recolectada de cada dispositivo y los perfiles asociados a la información que se quiere guardar de cada organización.

3. **MySQL 5**

   Este es el gestor de base de datos que se ha decido usar para los elementos ajenos a OpenGate®. En él habrá 2 bases de datos

   1. Una donde se guardará la información básica relativa a las VPN creadas así como a los “clientes” dados de alta en cada una de ellas.

   2. Otra propia del componente Freeradius. No se han hecho cambios respecto al propio esquema que provee Freeradius.

MaraDNS

MaraDNS es un Sistema de Nombres de Dominios (DNS) pequeño, ligero y fácil de configurar que se usará en el proyecto para poder acceder a través del nombre de los distintos “clientes” de la VPN. La versión que se usará será 2.0.13, y se instalará en un servidor con sistema operativo GNU/Linux.

Su configuración es sencilla, y se creará un fichero con información para cada VPN creada, ya que se diferenciarán por el nombre de la red. Es decir, para acceder a los clientes, los nombres se resolverán por:

<nombre_cliente>.<nombre_red_vpn>

El nombre del fichero que “configurará” cada VPN se deberá llamar “db.<*nombrered_vpn*>” y la información que contendrá será del tipo:

```
dns.% a       172.19.18.185    ~
cliente1.%   a      10.1.1.10    ~
```

Cada vez que llegue accounting start de un cliente se añadirá su IP así como su “*username*” en el fichero y así el resto de usuarios de su red, podrán conocer su IP, o hacer referencia a él según su hostname.

Por ejemplo la IP de cliente1.red será 10.1.1.10

FreeRadius

Este componente es otro de los que se suelen usar con OpenGate ya que permite conocer determinada información de los dispositivos si estos envían accountings, y a través de esta información se puede gestionar o deducir otra en la plataforma. La versión que se usará será 2.2.6, y se instalará en un servidor con sistema operativo GNU/Linux compatible.

De este componente en el proyecto, sólo se usará el envío de accounting, para que llegue a la plataforma información sobre los dispositivos conectados a las distintas VPNs. En FreeRadius deben estar dados de alta como clientes NAS los distintos servidores OpenVPN que darán servicio en las organizaciones clientes. Así, cuando se produzca un evento de conexión de un dispositivo en una VPN, se podrá informar a FreeRadius y se aceptaran esos mensajes. FreeRadius a su vez notificara a OpenGate, mediante un paquete radius de accounting start de la conexión realizada.

Para que los mensajes puedan ser procesados correctamente por la plataforma OpenGate, FreeRadius deberá añadir información adicional a los paquetes accountings recibidos de OpenVPN.

OpenVPN

Es la tecnología VPN que se quiere instalar para ofrecer redes privadas a las distintas organizaciones para cubrir ciertas necesidades. Se instalará en un servidor con sistema operativo GNU/Linux la versión OpenVPN 2.3.11 x86_64.

Se tendrá un fichero de configuración para cada red VPN que se añada al sistema. Este fichero será el que se use en el arranque de cada OpenVPN. Todos los ficheros usarán como base una plantilla definida que se completará con los datos de cada organización en el momento del alta. Se ha establecido una plantilla estándar que se copiará cada vez que se cree una VPN a través del API REST **provapi**.

Como se indicaba uno de los requisitos, es necesario que los clientes conectados a una misma VPN se vean entre ellos, para ello es necesario activar:

```
# Uncomment this directive to allow different
# clients to be able to "see" each other.
# By default, clients will only see the server.
# To force clients to only see the server, you
# will also need to appropriately firewall the
# server's TUN/TAP interface.
client-to-client
```

Para poder tener acceso entre los distintos *usuarios* de una misma VPN, a través de sus identificadores, se tienen que configurar los clientes que se conectan para que informe al servidor DNS. Esto se hace a través de la configuración de la VPN con

```
push "dhcp-option DNS 35.157.148.125"
```

Para cada organización que solicite la necesidad de tener una red privada, se generará,mínimo, una red OpenVPN. Si en la misma red es necesario

tener clientes Linux y Windows será necesario crear 2 redes VPN, cada una con una topología distinta.

topology subnet (windows)

---

topology p2p (Linux)

El servidor OpenVPN necesita de un “firewall” para proteger accesos no deseados. Para gestionar eso se ha creado un fichero vpn_firewall.sh que se ejecutará cuando se inicie una VPN y cuyo contenido será:

```
[root@openvpn openvpn]# more vpn_firewall.sh

# OpenVPN

# VPN's
${VPN_NAME}=IP_VPN_LOCAL/MASK

# Loopback address
LOOP=127.0.0.1

# Delete old iptables rules
# and temporarily block all traffic.
iptables -P OUTPUT DROP
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -F

# Set default policies
iptables -P OUTPUT ACCEPT
iptables -P INPUT DROP
iptables -P FORWARD DROP

# Allow local loopback
iptables -A INPUT -s $LOOP -j ACCEPT
iptables -A INPUT -d $LOOP -j ACCEPT

# Allow incoming pings (can be disabled)
#iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT

# Allow services such as ssh (can be disabled)
iptables -A INPUT -p tcp --dport ssh -j ACCEPT

# Allow incoming OpenVPN packets
# Duplicate the line below for each
# OpenVPN tunnel, changing --dport n
# to match the OpenVPN UDP port.

iptables -A INPUT -p tcp --dport ${VPN_PORT} -j ACCEPT
```

```
# OpenGate Ports
iptables -A INPUT -i tun+ -p tcp --dport 25281 -j ACCEPT
iptables -A FORWARD -i tun+ -p tcp --dport 25281 -j ACCEPT
iptables -A INPUT -i tun+ -p tcp --dport 9955 -j ACCEPT
iptables -A FORWARD -i tun+ -p tcp --dport 9955 -j ACCEPT
iptables -A INPUT -i tun+ -p tcp --dport 1883 -j ACCEPT
iptables -A FORWARD -i tun+ -p tcp --dport 1883 -j ACCEPT
# Keep state of connections from local machine and private subnets
iptables -A OUTPUT -m state --state NEW -o eth1 -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -m state --state NEW -o eth1 -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

# NAT the VPN client traffic to the internet
iptables -t nat -A POSTROUTING -s ${VPN_NAME} -o eth0 -j MASQUERADE
```

La última línea de este fichero es necesaria para encapsular la IP del servidor VPN en el tráfico que se propaga hacia los servidores de la red privada.

Plugin radius

Se ha tenido que añadir un plugin para poder interconectar OpenVPN con FreeRadius, que a la vez mandará los mensajes a la plataforma OpenGate. Para su correcta instalación habrá que compilarlo en el servidor para que no haya problemas de librerías.

Además, se deberá configurar un fichero de configuración del plugin por cada VPN. Para ello hay una plantilla que se clonará y configurará.

Interfaz de provisión

Uno de los requisitos no funcionales era ofrecer una interfaz/API amigable para poder dar de alta/baja las distintas VPN así como los clientes que las usan. Como se ha comentado, este módulo implementa una interfaz REST, lo que implica que las peticiones se harán a través del protocolo HTTP con los métodos estándar: POST, DELETE y GET.

Se ofrece una interfaz REST que ha sido desarrollada con python (***provapi***) gracias al framework Flask. Este *framework* es muy ligero y facilita el desarrollo de aplicaciones API REST. Dispone de un número reducido de características para implementar peticiones del protocolo HTTP y su manipulación, pero se pueden añadir paquetes según las necesidades del desarrollo. Gracias a esto ayuda a reducir el tiempo de desarrollo así como

elegir las extensiones que mejor se adapten a nuestras necesidades y poder tener un código fácilmente mantenible, limpio y ligero.

Los casos de uso que debe cubrir este módulo serían:

- Alta VPN
- Baja VPN
- Consulta VPN concreta
- Consulta lista de VPNs
- Alta cliente
- Baja cliente
- Consulta de un cliente

Cada caso de uso actuará sobre las estructura de OpenVPN y su base de datos realizando los cambios esperados.

Habrá casos de uso asociados a errores de provisión que el sistema tiene que tener en cuenta y tratar de manera correcta, y devolviendo errores catalogados. Estos casos de uso serían:

- Alta de VPN que ya existe en el sistema
- Alta de VPN con falta de información en el JSON
- Alta de VPN con fallo en la configuración de la red
- Lectura de VPN que no existe
- Baja de VPN que no existe
- Baja de VPN con fallo en el sistema OpenVPN
- Alta de un cliente VPN con falta de información en el JSON
- Alta de un cliente VPN con fallo durante el proceso
- Alta de un cliente VPN que ya existe
- Baja de un cliente VPN que no existente

Como este modulo implementa una interfaz REST, las peticiones se harán a través del protocolo HTTP con los métodos estándar: POST, DELETE y GET.

El contenido de cada petición POST HTTP ser un JSON que representa al objeto que se quiere aprovisionar. Un objeto representará una VPN o cliente VPN. El contenido de cada objeto requiere de una información concreta.

Una VPN vendrá definida por los atributos:

- name: Nombre de la VPN que coincidirá con el identificador de la organización en OpenGate
- port: Puerto para conectarse a la VPN
- subnet: Tipo de red que se va a crear. Idirección Ip de la red (3.1.1.0/24)
- topology: Topología que usará (subnet, p2p)

Todos los atributos son de tipo string.

Un ejemplo en formato JSON de lo anterior sería:

```
{
 "name" : "demotest1",
 "port" : "2294",
 "subnet" : "18.1.1.0/24",
 "topology" : "subnet"
}
```

Un cliente de VPN sólo será identificado por el atributo *client* que será el identificador de este cliente en la red.

JSON de ejemplo para un cliente:

```
{
 "client" : "00001"
}
```

Para poder usar esta interfaz de provisión se han tenido que definir las URLs que darán acceso. Estas son:

- Para VPN: http://localhost/provapi/VPNdomain
- Para clientes de VPN: http://localhost/provapi/VPNclient/<vpn_name>

Para acceder a un recurso creado previamente (para hacer un GET o un DELETE) habrá que añadir a cada URL el identificador del recurso, el *name* de la VPN o el *client* de los clientes VPN.

Como todo API REST hay que definir el formato de las respuestas, que como usan el protocolo HTTP cada respuesta tendrá una cabecera y un cuerpo. En la cabecera se utilizará el elemento Status-Code para informar

del resultado de la petición. El resultado puede ser erróneo o correcto y seguiremos el estándar RFC 2616 de los códigos HTTP. Por lo que, como respuesta a peticiones válidas habrá que devolver:

- Un 201 para POST
- Un 204 para DELETE
- Un 200 para GET

En el cuerpo viajará en formato texto el JSON formado cuando la petición sea de tipo GET.

Si hay algún error en el proceso de la petición, el servicio deberá informar del error. Los escenarios posibles son:

- El JSON es correcto pero no tiene la información necesaria para la petición.
- Se intenta leer/eliminar una VPN o cliente de VPN que no existe.
- Se produce un error interno en el servicio.
- Se produce un problema de acceso a base de datos.

Cuando se responde con un error habrá que aportar información que permita al usuario que realizó la petición ver que fué mal. Por esto, las respuestas de error tendrán un cuerpo en formato JSON con la estructura:

- *code*: un entero con el código de error
- *message*: un string con la descripción del error

Un ejemplo de este tipo sería :

```
{
   "code": 4003,
   "message": "Error parsing JSON. VPN port information is missing"
}
```

Se puede ver el listado completo de errores en el apartado Catálogo de errores del apéndice de la interfaz de provisión.

Hay que tener en cuenta que cada vez que a través del API se crea una red VPN, se “creará” el fichero del DNS que “configurará” la nueva red, es decir, se genera el fichero “db.<*nombrered_vpn*>”.

OpenGate UX

Es la web desarrollada por amplía))) para poder mostrar toda la información recolectada a través de la plataforma OpenGate ®. A través de ella se podrá visualizar la información guardada en la plataforma que los dispositivos conectados han ido enviando, que ha inferido a partir de la información recolectada, y se podrán realizar operaciones sobre los dispositivos, como comprobar su estado, actualizar el software..

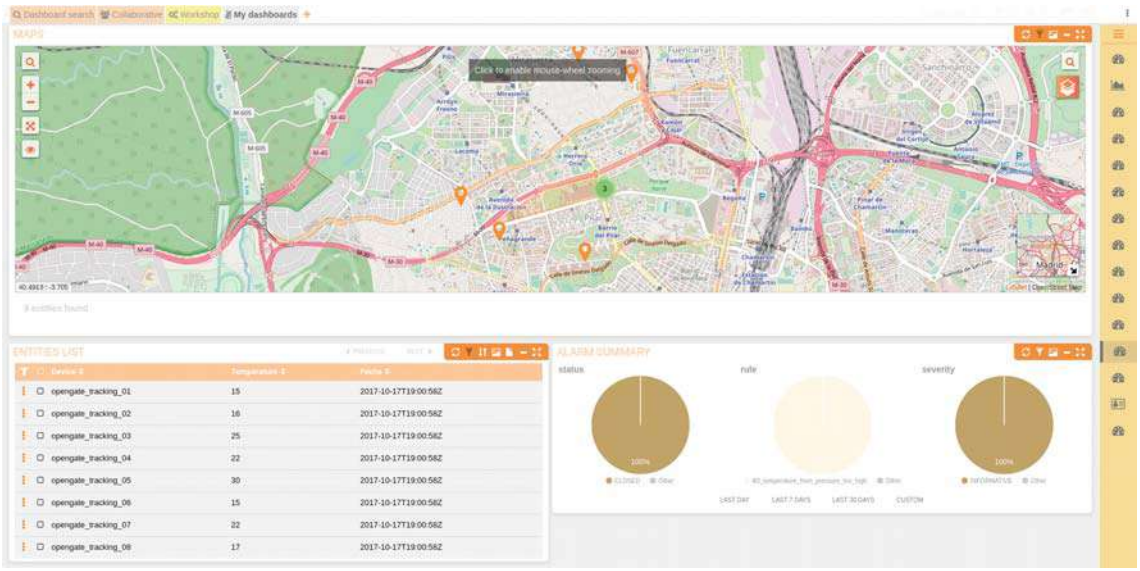

Figura 5.2 – Visión global de los dispositivos

Un cliente puede, con un simple vistazo ver el estado de sus dispositivos, es decir, ubicación, estado en general de cada dispositivo (que dependerá del color de los dispositivos posicionados), ver si se han producido alarmas, etc. También se puede ver toda la información tanto aprovisionada como recolectada de un dispositivo, así como si ha generado alarmas o no, el estado de las operaciones que se han realizado sobre él, etc.

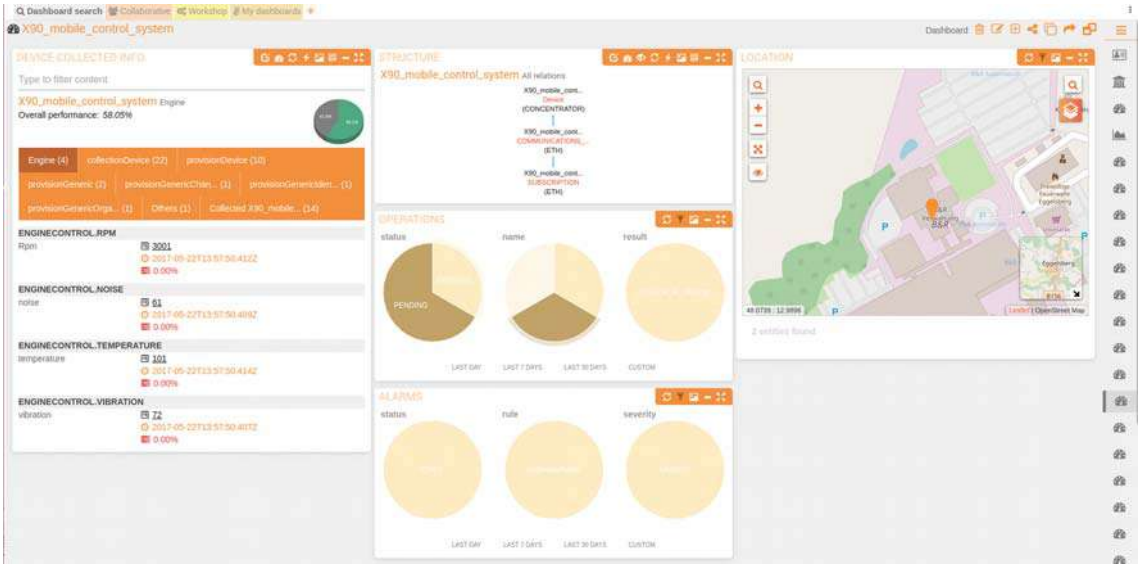

Figura 5.3 – Visión global de un dispositivo

Arquitectura funcional

Una vez explicada la arquitectura del sistema pasamos a detallar el funcionamiento del mismo. Para que quede más claro se va a simular el diagrama de flujo cuando se quiere dar de alta una organización en el sistema con la característica de que necesita hacer uso de una VPN ya que necesita conectarse a sus clientes directamente.

Paso 1.- **Alta de la organización**

Cuando se quiere dar de alta una organización habrá que:

I. Dar de alta el dominio y usuarios en OpenGate para que se puedan gestionar los dispositivos así como acceder acceder a OpenGate UX.

   Se dará de alta, a través de OpenGate UX, en la plataforma primero el dominio (demotest1) y luego los usuarios (cgvares@fi.upm.es) correspondientes para que puedan acceder a la web.

II. Crear la VPN. Para ello es necesario definir un nombre, y un rango de direcciones IP que serán las que se asignen a los dispositivos cuando se conecten.

   Para esto se tiene una interfaz de provisión de cara a facilitar la gestión de las VPN. A través de cualquier cliente REST se puede llevar a cabo la tarea.

   El servidor donde está la plataforma OpenGate se dará de alta como cliente de la VPN para que se conecte y así podrá recibir la información de los distintos clientes conectados a dicha red a través de la VPN.

Paso 2.- **Alta de los dispositivos**

Para cada uno de los dispositivos que se van a poder conectar a la VPN es necesario que tengan un certificado para así poder identificarlos de manera unívoca. En los dispositivos es necesario que esté su certificado y el certificado de la Autoridad Certificadora (CA) que firma estos certificados.

Cuando se da de alta un cliente a través del API, se generan automáticamente sus certificados de acceso. Estos habrá que facilitárselos a los clientes que se van a conectar.

El alta de un usuario puede hacerse a través de cualquier cliente REST.

También habrá que dar de alta los dispositivos en OpenGate para que puedan ser gestionados y monitorizados, se podrán ver tanto las conexiones que hace contra la VPN como la información que recolecta y envía. Se darán

de alta a través de OpenGate UX con uno de los usuarios ya creados de acceso a la web. Se dan de alta 2 clientes (00001 y 00002). En el apéndice podemos ver los dos dispositivos provisionados y su información.

Paso 3.- **Acceso cliente a la VPN**

Una vez el dispositivo tiene los certificados, puede conectarse a la VPN. El OpenVPN mandará el mensaje de conexión al FreeRadius, que lo procesará y mandará un paquete de accounting que “comprenda” OpenGate, el cual procesará la información correspondiente. A partir de ese momento el cliente podrá enviar información a la plataforma OpenGate por cualquiera de sus conectores.

Trazas de conexión del cliente 00001:

```
[root@radiusm openvpn]# openvpn --config client_demotest1.conf
Tue Jun  5 17:05:05 2018 OpenVPN 2.4.6 x86_64-redhat-linux-gnu [Fedora EPEL patched] [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] built on Apr 26 2018
Tue Jun  5 17:05:05 2018 library versions: OpenSSL 1.0.2k-fips  26 Jan 2017, LZO 2.06
Tue Jun  5 17:05:05 2018 NOTE: the current --script-security setting may allow this configuration to call user-defined scripts
Tue Jun  5 17:05:05 2018 TCP/UDP: Preserving recently used remote address: [AF_INET]172.19.18.103:2294
Tue Jun  5 17:05:05 2018 Socket Buffers: R=[87380->87380] S=[16384->16384]
Tue Jun  5 17:05:05 2018 Attempting to establish TCP connection with [AF_INET]172.19.18.103:2294 [nonblock]
Tue Jun  5 17:05:06 2018 TCP connection established with [AF_INET]172.19.18.103:2294
Tue Jun  5 17:05:06 2018 TCP_CLIENT link local: (not bound)
Tue Jun  5 17:05:06 2018 TCP_CLIENT link remote: [AF_INET]172.19.18.103:2294
Tue Jun  5 17:05:06 2018 TLS: Initial packet from [AF_INET]172.19.18.103:2294, sid=9208e682 b132400d
Tue Jun  5 17:05:06 2018 VERIFY OK: depth=1, C=ES, ST=Madrid, L=Madrid, O=Amplia, OU=IoT Security, CN=Amplia CA, name=Opengate, emailAddress=iot-security@amplia.es
Tue Jun  5 17:05:06 2018 VERIFY KU OK
Tue Jun  5 17:05:06 2018 Validating certificate extended key usage
Tue Jun  5 17:05:06 2018 ++ Certificate has EKU (str) TLS Web Server Authentication, expects TLS Web Server Authentication
Tue Jun  5 17:05:06 2018 VERIFY EKU OK
Tue Jun  5 17:05:06 2018 VERIFY OK: depth=0, C=ES, ST=Madrid, L=Madrid, O=Amplia, OU=IoT Security, CN=demotest1, name=Opengate, emailAddress=iot-security@amplia.es
Tue Jun  5 17:05:06 2018 Control Channel: TLSv1.2, cipher TLSv1/SSLv3 DHE-RSA-AES256-GCM-SHA384, 2048 bit RSA
Tue Jun  5 17:05:06 2018 [demotest1] Peer Connection Initiated with [AF_INET]172.19.18.103:2294
Tue Jun  5 17:05:07 2018 SENT CONTROL [demotest1]: 'PUSH_REQUEST' (status=1)
Tue Jun  5 17:05:07 2018 PUSH: Received control message: 'PUSH_REPLY,dhcp-option DNS 172.19.18.103,route-gateway 18.1.1.1,topology subnet,ping 10,ping-restart 120,ifconfig 18.1.1.3 255.255.255.0'
Tue Jun  5 17:05:07 2018 OPTIONS IMPORT: timers and/or timeouts modified
Tue Jun  5 17:05:07 2018 OPTIONS IMPORT: --ifconfig/up options modified
Tue Jun  5 17:05:07 2018 OPTIONS IMPORT: route-related options modified
```

```
Tue Jun  5 17:05:07 2018 OPTIONS IMPORT: --ip-win32 and/or --dhcp-option options
modified
Tue Jun  5 17:05:07 2018 Outgoing Data Channel: Cipher 'BF-CBC' initialized with 128 bit key
Tue Jun  5 17:05:07 2018 WARNING: INSECURE cipher with block size less than 128 bit (64
bit).  This allows attacks like SWEET32.  Mitigate by using a --cipher with a larger block size
(e.g. AES-256-CBC).
Tue Jun  5 17:05:07 2018 Outgoing Data Channel: Using 160 bit message hash 'SHA1' for
HMAC authentication
Tue Jun  5 17:05:07 2018 Incoming Data Channel: Cipher 'BF-CBC' initialized with 128 bit key
Tue Jun  5 17:05:07 2018 WARNING: INSECURE cipher with block size less than 128 bit (64
bit).  This allows attacks like SWEET32.  Mitigate by using a --cipher with a larger block size
(e.g. AES-256-CBC).
Tue Jun  5 17:05:07 2018 Incoming Data Channel: Using 160 bit message hash 'SHA1' for
HMAC authentication
Tue Jun  5 17:05:07 2018 WARNING: cipher with small block size in use, reducing reneg-bytes
to 64MB to mitigate SWEET32 attacks.
Tue Jun  5 17:05:07 2018 TUN/TAP device tun0 opened
Tue Jun  5 17:05:07 2018 TUN/TAP TX queue length set to 100
Tue Jun  5 17:05:07 2018 do_ifconfig, tt->did_ifconfig_ipv6_setup=0
Tue Jun  5 17:05:07 2018 /sbin/ip link set dev tun0 up mtu 1500
Tue Jun  5 17:05:07 2018 /sbin/ip addr add dev tun0 18.1.1.3/24 broadcast 18.1.1.255
Tue Jun  5 17:05:07 2018 /root/openvpn/update-resolv-conf tun0 1500 1544 18.1.1.3
255.255.255.0 init
dhcp-option DNS 172.19.18.103
tun0.inet
Tue Jun  5 17:05:07 2018 WARNING: this configuration may cache passwords in memory --
use the auth-nocache option to prevent this
Tue Jun  5 17:05:07 2018 Initialization Sequence Completed
```

Paso 4.- **Visualización acceso cliente VPN en UX**

El usuario a través de UX podrá ver la información del cliente VPN, que ha sido dado de alta con anterioridad en la plataforma OpenGate.

En él se podrá ver que está conectado y toda la información que ha enviado en esta sesión y en sesiones previas gracias al histórico de OpenGate.

Paso 5.- **Otro cliente accede a VPN**

Se da de alta otro cliente tanto en la plataforma OpenGate como en la VPN. Este se conectará a la VPN y podrá también enviar información.

Paso 6.- **Conectividad entre los usuarios**

Ahora estos 2 clientes podrán comunicarse entre ellos de forma directa a través de la VPN.

Tendrán conectividad en base a su dirección IP y también en base a su nombre simbólico gracias al DNS que provee el sistema y donde quedaron registrados al establecer la conexión VPN. Para ello podrán acceder a través de su DNS interno por lo nombres, es decir:

```
[root@clouddruida001 certs_pfc]# nslookup 00001.demotest1
Server:         172.19.18.103
Address:        172.19.18.103#53
Non-authoritative answer:
Name:  00001.demotest1
Address: 18.1.1.3

[root@radiusm opengate-api-python-examples]# nslookup demotest1-
99999.demotest1
Server:         172.19.18.103
Address:        172.19.18.103#53
Non-authoritative answer:
Name:  demotest1-99999.demotest1
Address: 18.1.1.2
```

Capítulo 6. Plan de proyecto para implementación

División en tareas del proyecto

El primer paso a realizar para la planificación del proyecto es dividir el mismo en las tareas y subtareas en que se compone. Esta división debe tener en cuenta el tipo de trabajo a realizar en cada tarea, la importancia de la misma y su interrelación con el resto de puntos del proyecto. Para este proyecto en particular se han definido las siguiente tareas:

- PT0. Gestión del Proyecto: Incluye todas las subtareas de gestión de los distintos equipos encargados del proyecto, tanto individualmente como en la relación entre ellos. Es una tarea que afecta a todas las otras tareas del proyecto pero sin estar directamente relacionada con ninguna.
- PT1. Requisitos y arquitecturas: Esta tarea implica el estudio de los requisitos del proyecto y de la arquitectura hardware, software y de redes del mismo.
- PT2. Instalación plataforma OpenGate: Esta tarea implica la instalación y configuración de la plataforma OpenGate y OpenGate UX. Tanto la plataforma en sí como las bases de datos de las que hace uso, Oracle y Mongo.
- PT3. Instalación de OpenVPN: Esta tarea implica la instalación y la configuración correspondiente del OpenVPN como la instalación de los software adicionales para el funcionamiento completo del sistema (MaraDNS, FreeRadius, MySQL,etc.)
- PT4. API de provisión. Esta tarea implica el diseño, desarrollo, instalación y configuración del nuevo API provisión (provapi).
- PT5. Pruebas de concepto: Esta tarea implica realizar las pruebas para validar que el nuevo sistema cumple con todos los requisitos del sistema.

Definiremos también una serie de hitos del proyecto que permiten definir objetivos y facilitan la planificación del mismo. Los principales hitos del proyecto son:

- Arranque del proyecto: Este hito marca el inicio de las primeras tareas del proyecto, y por tanto no hay ninguna tarea que se inicie antes de este hito.

- Final del análisis: Este hito marca el final de la fase de análisis de requisitos del proyecto.
- Final del diseño: Este hito marca el final de la fase de diseño tanto de la arquitectura del sistema como de los componentes individuales.
- Final de la implementación: Este hito marca el final de la implementación de los componentes del sistema.
- Final del proyecto. Este hito marca el final del proyecto. Todas las tareas deben de haberse completado antes de alcanzar este hito.

Planificación de tareas

Una vez se han definido las distintas tareas que conforman el proyecto hay que asignar a cada una el tiempo que se considere necesario para su finalización y establecer las relaciones entre las distintas tareas. Tras un estudio de las tareas se ha llegado a la siguiente planificación:

- PT0. Gestión del Proyecto. Esta tarea es especial ya que no consiste en una serie de puntos u objetivos que cumplir, sino que engloba los trabajos y actividades que es necesario llevar a cabo durante todo el proyecto. Por ello la duración de esta tarea equivale a la duración del proyecto completo.
- PT1. Requisitos y arquitecturas. Esta tarea se divide en tres partes diferenciadas. La primera es el análisis de requisitos, le sigue el diseño de la arquitectura y finalmente el diseño de los componentes. A cada una de estas fases se le asignarán 5 días de trabajo.
- PT2. Instalación plataforma OpenGate: Esta tarea se puede dividir en 2 partes, instalación del software propio de OpenGate (OpenGate y OpenGate-UX) e instalación de los sistemas de almacenamiento (Oracle y MongoDB). A esta tarea completa se le asignarán 10 días, ya que habrá que realizar las pruebas correspondientes para validar el entorno.
- PT3. Instalación OpenVPN: Esta tarea puede dividirse en varias. Una será la instalación de servidor MaraDNS, el FreeRadius y la base de datos MySQL y otra la del software del OpenVPN. A esta fase se le asignará 5 días.
- PT4. API de provisión. Esta tarea se divide básicamente en el análisis/desarrollo del software y su posterior instalación. A esta fase se le dedicará 6 días.

- PT5. Pruebas de concepto. Esta tarea consistirá en hacer pruebas completas del sistema final. Tanto de conexión a la VPN como de conectividad con la plataforma OpenGate y entre los distintos dispositivos de la misma red. A esta fase se le dedicarán 4 días.

Diagrama de Gantt

Con las hitos ya definidos asignamos las tareas teniendo en cuenta los hitos y la duración de las mismas.

La tarea *PT0 Gestión del proyecto* dura todo el proyecto, por lo que se le asignará desde el arranque del proyecto hasta el final del mismo. La tarea *PT1 Requisitos y Arquitecturas* se inicia nada más arrancar el proyecto, su primera subtarea acaba con el hito de final del análisis y la tarea completa finaliza con el hito de final de diseño. La tarea *PT2 Instalación plataforma OpenGate* inicia el hito de implantación, y se realiza después de terminar el diseño. Inmediatamente después se empezará la tarea *PT3 Instalación OpenVPN* y después la tarea *PT4 API provisión,* y se podrá dar por finalizado el hito de implantación. Finalmente se realizará la tarea PT4 Pruebas de concepto, que se dará por finalizado con el hito de final del proyecto.

El diagrama de Gantt resultante es el siguiente:

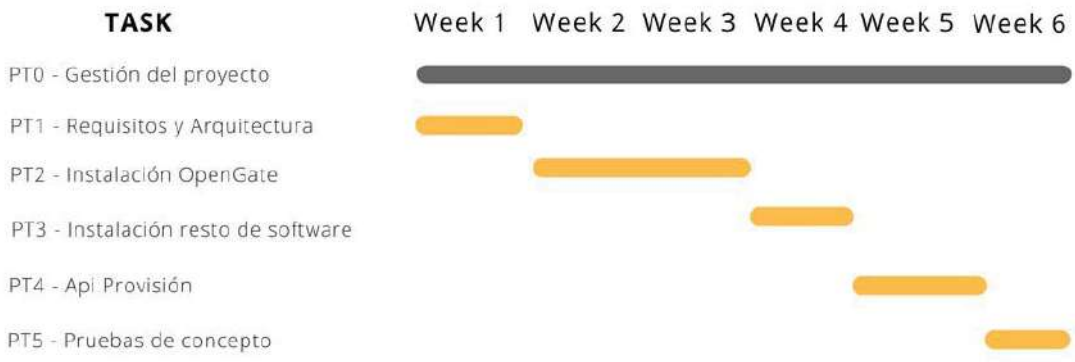

Figura 6.1 – Diagrama de Gantt

Capítulo 7. Conclusiones

Análisis del cumplimiento de objetivos

Una vez completado el proyecto llega el momento de analizar el resultado del mismo y sacar conclusiones útiles para futuros desarrollos. Primero estudiaremos los objetivos del proyecto según se plantearon en el principio de esta documentación y analizaremos si se han cumplido y en qué grado. Tomemos por tanto la lista de objetivos del primer capítulo para analizarlos uno a uno.

El sistema diseñado cubre la necesidad de dar seguridad tanto en el acceso de las conexiones con la plataforma, como con la información intercambiada entre los distintos elementos de la solución IIoT. Los elementos se conectan a la red VPN a través de certificados generados por la organización, y permite, que una vez conectados a la VPN enviar mensajes a todos los elementos de la misma red.

Esto a su vez, puede reducir el *overhead* ya que se pueden enviar mensajes no seguros, protocolo UDP o http y mqtt sin ser https ni mqtts al confiar ya en los elementos que se encuentran en nuestra red.

Al facilitarse también el envío de mensajes a través del protocolo UDP se facilita la escalabilidad de la solución ya que se podrán limitar los recursos en la solución.

Valoraciones personales

Este proyecto ha resultado, desde el punto de vista personal, bastante satisfactorio. El proyecto me ha servido para aprender muchas cosas , entre las que puedo destacar:

- **Trabajo en equipo**: Durante el proyecto he trabajado en un equipo con personas dedicadas a distintas tareas (programadores, sistemas, gestores, etc.) lo que ha requerido aprender a coordinarse y colaborar y con el resto de los compañeros.
- **Colaboración con otras empresas**: Además de trabajar con los compañeros de mi empresa, ha sido necesario coordinarse con otras empresas de las que somos *partners* para poder realizar pruebas con sus dispositivos.
- **Experiencia en M2M**: Este proyecto ha servido además para aumentar mis conocimientos y experiencia en las tecnologías M2M,

IoT e IIoT, campos que están en auge y que representan el futuro para muchos sectores.

Futuras evoluciones del sistema

Por último vamos a analizar cómo podría evolucionar este proyecto en el futuro. Como las opciones son muchas vamos a limitarnos a mencionar brevemente algunas de las posibles mejoras y nuevas características que se podrían añadir al sistema. Son las siguientes:

- Generar certificados propios, y no a través de terceros para así ser autónomos.
- Dockerización del proyecto,para según las necesidades puntuales se puedan levantar en paralelo los procesos correspondientes.
- Generar informe de las sesiones completas de los dispositivos en función de los datos enviados por cada una de las organizaciones.
- Soporte a operaciones masivas en las API de provisión.
- Soporte a direcciones IPv6. En este sentido comentar que MaraDNS ya trae soporte y se han hecho pruebas de concepto con resultado exitoso.
- Aprovechar las posibilidades que ofrece el sistema de reglas de automatización de OpenGate e implementar reglas para la solución diseñada e implementada. Sobre el papel parece que sería interesante tener monitorizados a los dispositivos cuando superen un cierto consumo de datos (ya sea diario, semanal, etc.)
- Ampliar la arquitectura para contar con *routers* en las sedes de las organizaciones que serían los que establecerían las conexiones VPN con el sistema. De este modo se podrían usar dispositivos con menos capacidades lo que abarataría las soluciones basadas en este sistema.
- Detección de intentos de acceso fallidos e incorporación a una lista negra a las direcciones IP que origen dicho tráfico.

Capítulo 8. Bibliografía

Amplia Blog. http://www.amplia-iiot.com/blogamplia/

Amplia Doc.- configuración_de_radius_v10.odt

VV. AA.. Wikipedia.. http://www.wikipedia.org.

https://es.wikipedia.org/wiki/Protocolo_AAA

https://technet.microsoft.com ???

https://www.oscarblancarteblog.com

https://www.w3.org/Protocols/rfc2616

https://www.afnic.fr/en/resources/publications/issue-papers/alternative-naming-systems-to-the-dns-2.html#GNUnet

https://en.wikipedia.org/wiki/Comparison_of_DNS_server_software

http://seguridadinformacion.blogspot.com/2015/11/servicios-aaa-ventajas-que-ofrece-el.html

Capítulo 9. Apéndices

OpenVPN

Plantilla de arranque de VPN

```
[root@openvpn openvpn]# more VPN_NAME.cnf

##############################################
# Sample OpenVPN 2.0 config file for  multi-client server.                  #
##############################################

local SERVER_IP
port VPN_PORT
proto tcp
dev tun

ca /etc/openvpn/certs/VPN_NAME_ca.crt
cert /etc/openvpn/certs/VPN_NAME.crt
dh /etc/openvpn/certs/dh2048.pem

topology TOPOLOGY
server IP_VPN_LOCAL MASK
ifconfig-pool-persist ipp.txt
push "route openvpn vpn_gateway"
push "dhcp-option DNS DNS_SERVER"
client-to-client
keepalive 10 120
comp-lzo

user nobody
group nobody

persist-key
persist-tun

status openvpn-status.log
log-append /var/log/VPN_NAME.log
verb 3

# Plugin FreeRadius
# Ubicación de la configuración del plugin para radius de openVPN
plugin /etc/openvpn/plugins/radiusplugin.so /etc/openvpn/VPN_NAME_radiusplugin.cnf
```

Plantilla de pluging radius

```
[root@openvpn openvpn]# more template_radiusplugin.cnf
# The NAS identifier which is sent to the RADIUS server
NAS-Identifier=VPN_NAME
# The service type which is sent to the RADIUS server
Service-Type=5
```

```
# The framed protocol which is sent to the RADIUS server
Framed-Protocol=1
# The NAS port type which is sent to the RADIUS server
NAS-Port-Type=5
# The NAS IP address which is sent to the RADIUS server
NAS-IP-Address=openvpn_server_ip
# Path to the OpenVPN configfile. The plugin searches there for
# client-config-dir PATH   (searches for the path)
# status FILE                    (searches for the file, version must be 1)
# client-cert-not-required (if the option is used or not)
# username-as-common-name  (if the option is used or not)
OpenVPNConfig=/etc/openvpn/vpntest1.conf
# Support for topology option in OpenVPN 2.1
# If you don't specify anything, option "net30" (default in OpenVPN) is used.
# You can only use one of the options at the same time.
# If you use topology option "subnet", fill in the right netmask, e.g. from OpenVPN option "–
# server NETWORK NETMASK"
subnet=255.255.0.0
# p2p=10.8.0.1
# Allows the plugin to overwrite the client config in client config file directory,
# default is true
overwriteccfiles=true
# Allows the plugin to use auth control files if OpenVPN (>= 2.1 rc8) provides them.
# default is false
# useauthcontrolfile=false
# Only the accouting functionality is used, if no user name to forwarded to the plugin, the
# common name of certificate is used as user name for radius accounting.
# default is false
accountingonly=true
# If the accounting is non essential, nonfatalaccounting can be set to true.
# If set to true all errors during the accounting procedure are ignored, which can be
# - radius accounting can fail
# - FramedRouted (if configured) maybe not configured correctly
# - errors during vendor specific attributes script execution are ignored
# But if set to true the performance is increased because OpenVPN does not block during the
accounting procedure.
# default is false
nonfatalaccounting=false
# A radius server definition, there could be more than one.
# The priority of the server depends on the order in this file. The first one has the highest
priority.
server
{
        # The UDP port for radius accounting.
        acctport=1813
        # The UDP port for radius authentication.
        authport=1812
        # The name or ip address of the radius server.
        name=IP_freeradius
        # How many times should the plugin send the if there is no response?
        retry=1
        # How long should the plugin wait for a response?
        wait=1
```

```
        # The shared secret.
        sharedsecret=freeradiuspwd
}
```

OpenGate UX

Creación de dominio

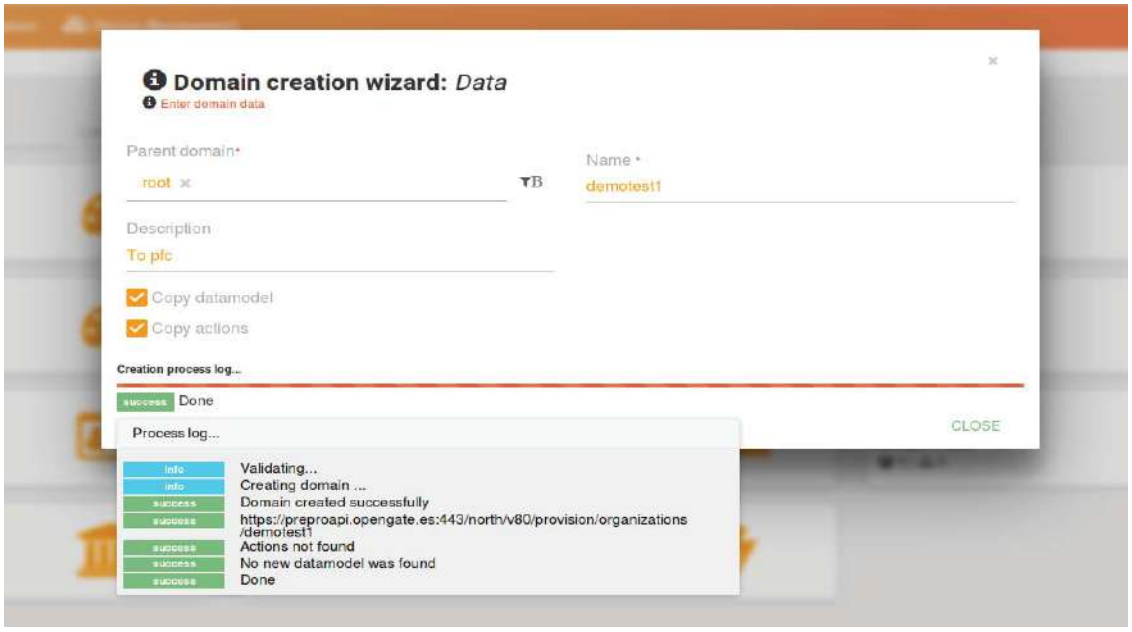

Figura 1 - Creación de dominios

Creación de usuario

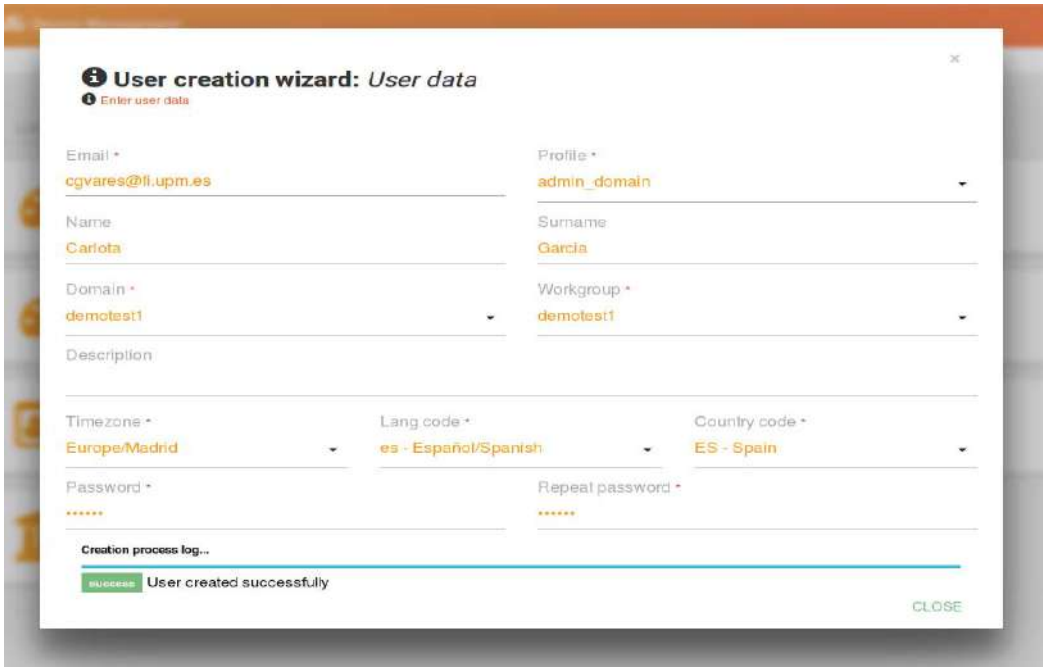

Figura 2 - Creación de usuario

Creación de dispositivos

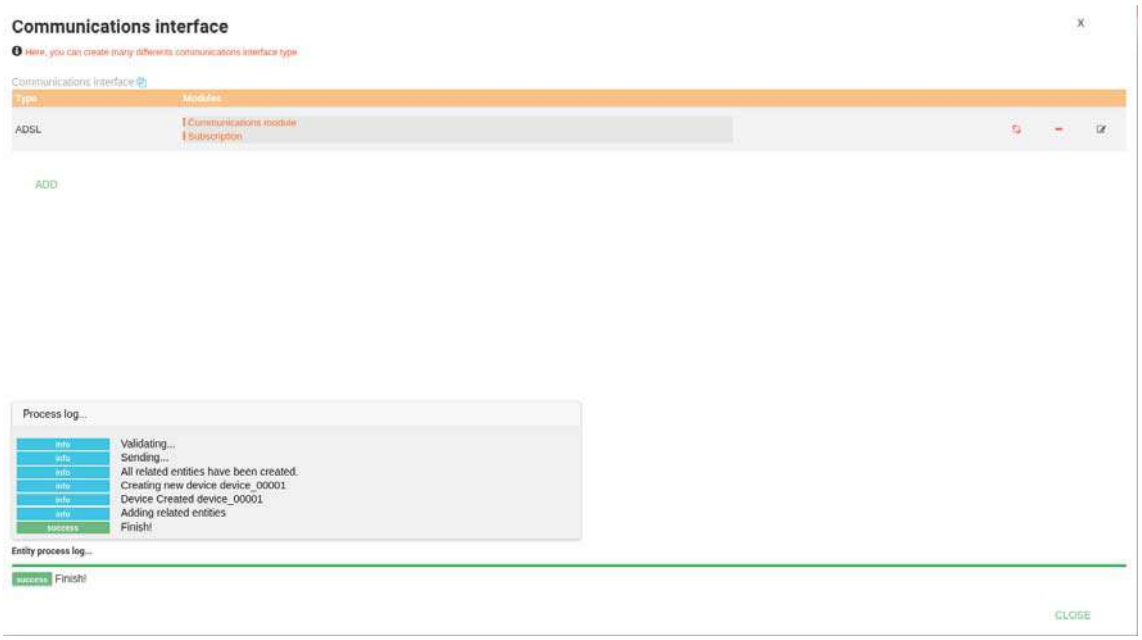

Figura 3 - Creación de dispositivo

Dispositivos provisionados

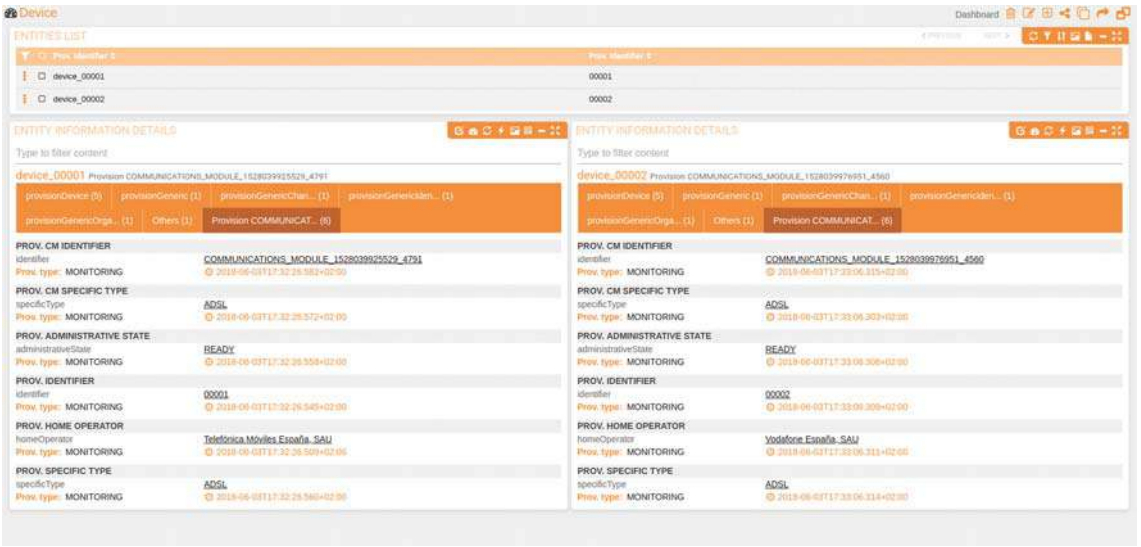

Figura 4 – Visualización dispositivos provisionados

Interfaz de provisión

Creación de VPN

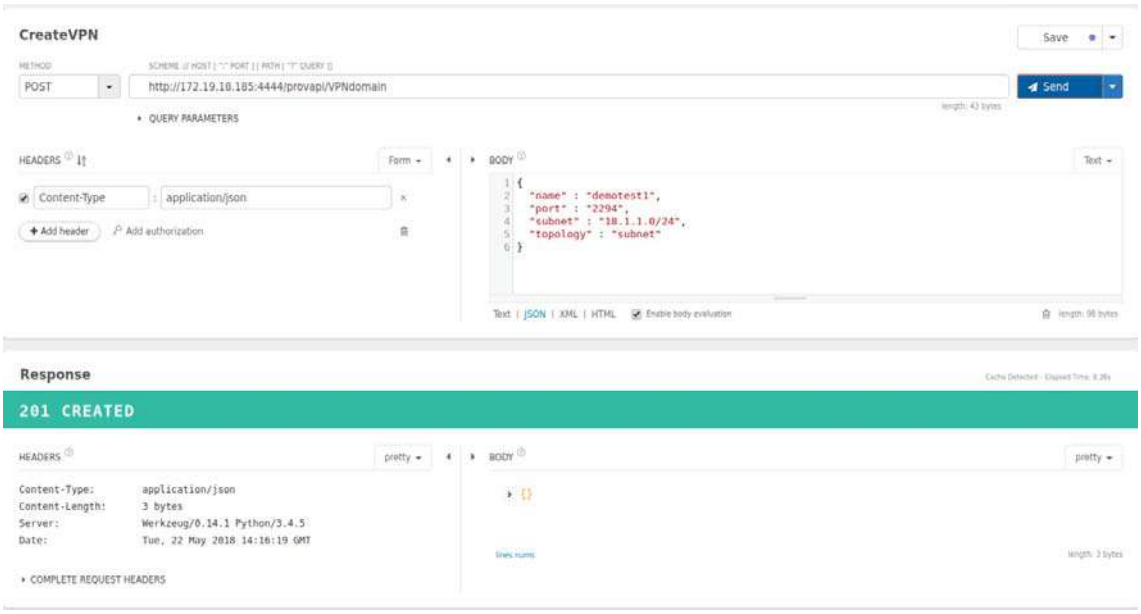

Figura 1 - Creación de VPN

Alta cliente de VPN

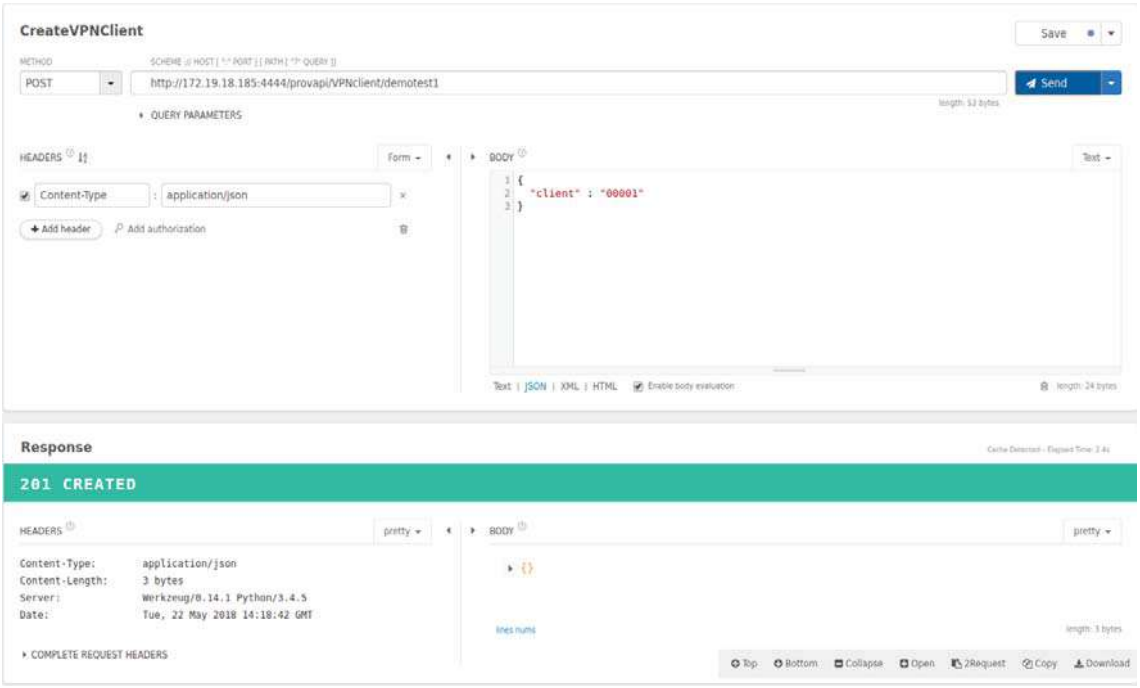

Figura 2 - Creación de cliente VPN

Catálogo de errores

Status-Code	Provapi-Code	Provapi-message
409	4001	VPN already exist
400	4002	Error creating VPN
400	4003	Error parsing JSON
404	4004	VPN not exist
400	4005	Error while removing VPN
400	4006	Error creating client
409	4007	VPN client already exist
404	4008	VPN client nost exist
400	4009	Error deleting VPN client
500	5001	Database error

Definiciones y acrónimos

Definiciones

- *AAA*: protocolo de autenticación, autorización y contabilidad.
- *6LoWPAN* : estandar que posibilita IPv6 sobre redes inalámbricas de área personal de baja potencia.
- *Multitenant*: principio de arquitectura software que permite a una sóla instancia de software servir a muchos clientes.
- *TLS*: protocolo criptográfico que garantiza las comunicaciones en Internet,permite y garantiza el intercambio de datos en un entorno securizado y privado entre dos entes, el usuario y el servidor.
- *SSL*: tecnología estándar para mantener segura una conexión a Internet, así como para proteger cualquier información confidencial que se envía entre dos sistemas e impedir que los delincuentes lean y modifiquen cualquier dato que se transfiera, incluida información que pudiera considerarse personal.
- *OpenSSL*: biblioteca de software para aplicaciones que aseguran las comunicaciones a través de redes informáticas contra el espionaje o la necesidad de identificar a la parte del otro extremo.

Acronimos

- WPAN: wireless personal area networks.
- 6LoWPAN : IPv6 over Low power Wireless Personal Area Networks
- UDP: User Datagram Protocol
- TCP: Transmision Control Protocol
- CoAP: Constrained Application Protocol
- XMPP: Extensible Messaging and Presence Protocol
- XML: Extensible Markup Language
- AMQP: Advanced Message Queuing Protocol
- MQTT: Message Queue Telemetry Transport
- MQTT-SN: Message Queue Telemetry Transport Sensors Network

- PPTP: Point-To-Point Tunneling Protocol
- MPPE: Microsoft Point-to-Point Excription
- L2TP: Layer 2 Tunneling Protocol
- TLS: Transport Layer Security
- SSL: Secure Sockets Layer
- SSTP: Secure Socket Tunneling Protocol
- IKEv2: Internet Key Exchange version 2
- CBC: Cipher-block chaining
- SOAP: Simple Object Access Protocol
- NAS: Network Access Server
- AAA: Authentication, Authorization and Accounting
- JSON: JavaScript Object Notation
- LDAP: Lightweight Directory Access Protocol
- DNS: Domain Name System
- ICAAN: Corporación de Internet para la Asignación de Nombres y Números
- HA: High Availability