Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

NetManDS: Framework de Gestión para consola Nintendo 3DS

Autor: Andrés Martínez García
Tutor: Antonio Estepa Alonso

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

NetManDS: Framework de Gestión para consola Nintendo 3DS

Autor:
Andrés Martínez García

Tutor:
Antonio Estepa Alonso
Profesor titular

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Trabajo Fin de Grado: NetManDS: Framework de Gestión para consola Nintendo 3DS

Autor: Andrés Martínez García

Tutor: Antonio Estepa Alonso

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

A mi familia

A mis maestros,

y a mis amigos y compañeros

Agradecimientos

En primer lugar, quisiera agradecer a mi familia y amigos por estar ahí, en lo bueno y en lo malo, y observar mi progreso como ingeniero y como persona.

También al Departamento de Ingeniería Telemática de la Escuela Técnica Superior de Ingenieros de Sevilla, por proporcionarme una enseñanza sólida y la oportunidad de realizar proyectos bajo su tutela.

También a Alfonso, profesor del IES Virgen de Consolación de Utrera, por aconsejarme y animarme a ser ingeniero de Telecomunicación.

Y, por supuesto, a José, David Gila, César Rincón, David Murphy… por haberme influenciado de alguna u otra manera a llegar hasta donde estoy hoy.

Gracias de todo corazón.

Andrés Martínez García

Sevilla, 2020

Resumen

Las redes de telecomunicación son sistemas de información que, durante su vida operativa, pueden presentar problemas en su funcionamiento o pueden necesitar ajustes en su configuración debido a circunstancias que escapan de nuestro control. Para dar salida a esta problemática, el objetivo de este proyecto es desarrollar un conjunto de utilidades para facilitar las tareas de gestión de redes, encapsuladas en una aplicación para la videoconsola Nintendo 3DS®.

Se explorarán todos los estándares implementados, se realizará una valoración técnica del dispositivo a utilizar así como de las herramientas de desarrollo disponibles. Además, se elaborará un diseño de la aplicación a modo de proyecto software para, finalmente, comentar detalles de la implementación de la solución así como de las pruebas realizadas para comprobar su correcto funcionamiento.

Abstract

Telecommunication networks are information systems that, during their operational life, may present problems in their operation or may need adjustments in their configuration due to circumstances beyond our control. In order to solve these problems, the aim of this project is the development of a set of utilities to ease network management tasks, encapsulated in an application for the Nintendo 3DS® video game console.

All the implemented standards will be explored, a technical overview of the device to be used as well as its available development tools. Furthermore, an application design will be elaborated as a software project and, finally, details on the implementation will be discussed, as well as the tests carried out to verify its correct behaviour.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN Y OBJETIVOS

El objetivo de este proyecto es diseñar e implementar una aplicación para la videoconsola Nintendo 3DS que sea capaz de realizar tareas fundamentales en el marco de la gestión de redes de telecomunicación. Debe ser capaz, por tanto, de soportar los protocolos más usados en este ámbito, así como de poder cumplir las funciones más esenciales en la gestión de redes: acceder a los equipos que forman parte de la red de forma segura, modificar sus parámetros de configuración, ser capaz de programar dichos equipos para la recepción de notificaciones, e incluso realizar labores de mantenimiento cuando son necesarias como, por ejemplo, actualizar el firmware del dispositivo.

En este apartado daremos unas pinceladas en el ámbito de la gestión de redes y el creciente auge de usar dispositivos móviles para este fin. Exploraremos los principales problemas que podemos encontrarnos en esta situación y, por último, detallaremos una propuesta para dar solución a los problemas planteados. Se realizará un análisis, además, sobre las ventajas que nos aportaría esta solución y qué inconvenientes pueden surgir, así como maneras de paliarlos.

1.1 Descripción de la situación actual

En el contexto de la gestión de redes, hay una tendencia que se ha ido acrecentando durante los últimos años, y es la de poder gestionar los equipos de una red desde cualquier lugar, a precio reducido, optimizando recursos como la energía usada, y de manera sencilla y eficiente.

En los diseños iniciales de los estándares de gestión de redes no era una prioridad que los sistemas finales tuvieran estas características. Por esta razón, los equipos usados en un primer momento no tenían grandes limitaciones en términos de hardware. Según [19], nos referimos a los "non-constrained devices", es decir, equipos voluminosos y de propósito general que, aunque puedan llevar a cabo las tareas de gestión de redes sin problema, no siempre son la mejor solución en términos de eficiencia energética y comodidad de uso.

Aquí entran en juego los dispositivos móviles. Con su alta movilidad, su eficiencia en el uso de energía y su manejo de manera cómoda y sofisticada para las personas brindan una nueva posibilidad: usarlos como terminales para monitorizar el estado de una red y realizar configuraciones sobre los elementos de la misma.

1.2 Problema en el contexto

Como hemos venido comentando, los equipos que han venido usándose para las tareas de gestión de redes cuentan con una serie de problemas que dificultan poder realizar su labor de manera óptima.

La principal de ellas es que suelen ser equipos grandes con una capacidad tanto de memoria como de procesamiento muchísimo mayor de la que es necesaria para monitorizar una red. Esto, aunque puede significar una ventaja a primera vista, debido principalmente a que seremos capaces de desarrollar aplicaciones de gestión tan potentes como queramos, la realidad es que los recursos hardware necesarios para la gestión de redes no suele ser excesivamente elevado y, por ende, estamos desperdiciando parte de esos recursos.

Este desaprovechamiento radica en un gasto mayor, tanto de capital para adquirir tanto el hardware como el software necesarios para los equipos en cuestión, como el gasto energético adicional que supone tener capacidad de procesamiento y memoria que no está siendo utilizada.

Otro de los inconvenientes es la seguridad de estos equipos. Como hemos adelantado, los terminales usados en la gestión de redes han seguido la tendencia de ser de propósito general. Con esto nos referimos a que ejecutan sistemas operativos genéricos que, por ser ampliamente conocidos en su mayoría, pueden suponer un problema de seguridad debido a que pueden ser obtenidos por cualquier persona y, ciertamente, cualquier persona puede encontrar vulnerabilidades en ellos. Esto, a corto o largo plazo, puede suponer que nuestros equipos estén bajo la amenaza de hackers no éticos que quieran hacerse con el control de nuestra red, usando

como puerta de entrada los exploits encontrados tanto en los sistemas operativos usados en los equipos como en los paquetes software que estos contienen. Este riesgo se acrecienta si los equipos no están actualizándose constantemente o si no hay una manera de actualizarlos de manera eficiente.

Por último, tocaremos un poco el tema de la comodidad. Aunque es cierto que existen suites para la gestión de redes, como SolarWinds (ver [1]) o Splunk (ver [2]) que nos hacen la vida más fácil cuando queremos realizar tareas complejas o incluso cuando tenemos un gran volumen de información que gestionar, la realidad es que la comodidad que ofrece un dispositivo móvil es mucho mayor. Principalmente, al ser sistemas móviles, pueden acompañarnos en todo momento, significando que tendremos acceso a nuestra red en cualquier momento y lugar. Esto tiene ciertas ventajas, como la posibilidad de obtener en el acto los eventos que ocurren en la red con el fin de poder actuar de una forma más rápida para que, por ejemplo, la calidad de servicio (QoS [1]) percibida por los usuarios de la red se vea afectada lo menos posible. No solo eso, sino que la interfaz de usuario tiende a ser más amigable al poder ser manejada con la palma de nuestra mano, con una variedad mayor de elementos gráficos que nos ayudarán a discernir en qué es lo que necesitamos en cada momento.

Como podemos observar, el uso de dispositivos móviles nos proporciona una amplia variedad de ventajas y soluciona una buena cantidad de problemas o incomodidades que teníamos en los equipos tradicionales.

1.3 Solución propuesta

Con el objetivo de dar una solución a los problemas planteados, se ha decidido usar la videoconsola Nintendo 3DS para usarla como terminal de gestión de redes de telecomunicación. Se trata de un dispositivo móvil relativamente reciente que cuenta con unas capacidades bastante más limitadas que un sistema de sobremesa, por ejemplo, pero es más potente que los sistemas embebidos, lo que nos brinda la posibilidad de desarrollar aplicaciones potentes y con una mayor optimización de los recursos disponibles. Esta consola ha presentado algunas revisiones a lo largo del tiempo, aunque esta solución será funcional en cualquiera de ellas.

Figura 1 – Versiones del sistema Nintendo 3DS

A continuación iremos desarrollando las ventajas de este terminal con respecto a otros que están disponibles, e incluso qué problemas pueden existir por usar un dispositivo de este tipo.

1.3.1 Ventajas de la solución

La principal razón por las que se ha decidido usar este dispositivo es su resolución superior a la de los terminales móviles convencionales, como lo son los teléfonos Android e iOS. Su doble pantalla brinda un sinfín de posibilidades para hacer el trabajo de gestor de redes lo más ameno posible.

Otra de las grandes ventajas es su sistema operativo. Al ser propietario de Nintendo, es un sistema operativo cerrado de forma que muy pocas personas pueden saber cómo funciona internamente. Esto, como comentamos, significa que nuestro sistema será más resiliente en temas de seguridad y, más concretamente, de

explotación de vulnerabilidades, ya que el propietario se encarga de actualizar de forma regular el sistema operativo. Esto es algo que no suele ocurrir con los sistemas operativos como Android, donde los fabricantes tienden a dejar de actualizar el firmware de los dispositivos una vez pasa un cierto tiempo.

Este sistema operativo es bastante completo, sobre todo cuando hablamos de temas de red, ya que proporciona un conjunto de APIs que nos facilitan la labor de gestionar las conexiones hasta el nivel de transporte, e incluso de aplicación en algunos casos, como es el protocolo HTTP. Además, lleva implementados varios mecanismos de seguridad como son los certificados X.509, las capas de seguridad SSL y TLS, e incluso varios algoritmos de criptografía tanto simétrica como asimétrica.

Y no nos quedamos ahí, puesto que tenemos APIs para prácticamente cualquier cosa que queramos desarrollar con la consola. Las más destacables son la gestión de audio y efectos de sonido, los elementos gráficos predeterminados como un teclado interactivo en pantalla o cajas de texto, gestión de hilos y temporizadores, sensores… Las posibilidades son prácticamente infinitas, y para tareas sencillas el uso de estas API facilita en gran medida el desarrollo.

Además, al ser un sistema pensado para los videojuegos, cuenta con una batería con una duración bastante larga, en torno a unas 5-8 horas funcionando a pleno rendimiento. Si usamos el modo de ahorro de energía, podemos expandir la duración de la batería bastante más, haciéndola comparable e incluso superior a la de los teléfonos inteligentes más actuales.

En cuanto al precio, es bastante inferior al de un sistema de sobremesa y parecido al de un dispositivo móvil de gama media-baja. De esta forma tenemos un dispositivo de bajo coste y con unos recursos hardware que se ajustan a las necesidades de un gestor de redes.

Como colofón, debemos tener en cuenta el tema de la comodidad. Pese a ser un sistema con dimensiones mayores a las de un teléfono inteligente, su forma permite que se adapte a la palma de la mano de manera bastante cómoda. Además, su lápiz táctil incorporado facilita la navegación por parte del usuario en las interfaces gráficas, ofreciendo bastantes posibilidades en cuanto a la accesibilidad se refiere.

Como vemos, el sistema Nintendo 3DS posee las características necesarias para paliar los problemas que venimos arrastrando en el contexto de la gestión de redes y no solo eso, sino que además nos abre una ventana de nuevas posibilidades.

1.3.2 Retos de la solución

Pese a todas las ventajas mencionadas en apartados anteriores, existen ciertas cuestiones que este sistema no puede abordar.

El principal problema es la dificultad de encontrar herramientas para desarrollar aplicaciones en esta plataforma. Debido a que es un sistema propietario, siempre cabe la posibilidad de pagar para obtener el kit de desarrollo oficial de Nintendo (CTR-SDK), aunque lo deseable es tener un kit de desarrollo de código libre y accesible por todos. Han habido intentos de realizar esto con bastante éxito, como son las herramientas proporcionadas por el grupo devkitpro (ver [3]), que de hecho es el SDK usado en este proyecto.

Otra de las dificultades es la imposibilidad de modificar los elementos hardware, por lo que añadir, por ejemplo, una nueva tarjeta de red para que soporte comunicaciones en la banda de 5G resulta prácticamente imposible debido al carácter propietario del sistema. Además, implementar los recursos software necesarios como los drivers para el nuevo hardware instalado sería una labor titánica si no se tiene acceso al kernel del sistema operativo.

También resulta difícil en ocasiones encontrar librerías y recursos software que satisfagan nuestras necesidades, debido a la baja gama de lenguajes de programación disponibles para desarrollar aplicaciones para este sistema, que tienden a ser de bajo nivel como C, u otros como C++ con características de orientación a objetos. Incluso si encontramos librerías en estos lenguajes, en ocasiones deberemos adaptarlas y refactorizarlas para que sean funcionales en el sistema, principalmente debido a las diferencias entre las API estándar y las API que usa la consola.

Pese a todo esto, se sigue considerando a la Nintendo 3DS como un sistema bastante potente cuyas dificultades pueden solventarse en su gran medida, aunque muchas veces radica en implementar mucho

código por nuestra cuenta.

2 CONCEPTOS PREVIOS Y ESTÁNDARES IMPLEMENTADOS

Antes de comenzar a explicar el proceso de diseño y desarrollo de esta aplicación, es preciso dar unas pequeñas pinceladas en el tema de los sistemas de gestión de forma general, qué es lo que se espera de ellos, qué dispositivos pueden ser interesantes de gestionar y de qué manera podemos realizar esto de forma sencilla y eficiente.

Una vez introducido este marco, indagaremos de una forma un poco más profunda en los protocolos de gestión y los modelos de datos involucrados en este proyecto. Se explicarán algunos conceptos básicos sobre los mismos pero sin entrar en demasiada profundidad en los aspectos técnicos a fin de que la comprensión de estos sea lo más amena posible.

2.1 Protocolos y sistemas de gestión

Cuando hablamos de un sistema o modelo de gestión de redes, debemos tener en cuenta qué es lo que se quiere conseguir. De forma rápida y resumida, un sistema de gestión de redes es un conjunto de normativas y estándares relacionados entre sí con el fin de asegurar la autenticación, aprovisionamiento, configuración, monitorización, mantenimiento, conectividad y seguridad de una red de computadoras. Veamos cada uno de estos objetivos de manera separada.

Asegurar la autenticación en una red es conocer de primera mano y sin ninguna duda quién está accediendo a los recursos de la misma, de forma que el usuario de la red quede unívocamente identificado frente a cualquier otro. Además, debe conocerse qué recursos (o qué cantidad de estos) está accediendo ese usuario. Por supuesto, se debe poder controlar que los recursos accedidos cumplen con las políticas establecidas (por ejemplo en un acuerdo de niveles de servicio o SLA en el caso de una red de datos) y en caso contrario debe de ser posible penalizar a ese usuario, o incluso bloquear temporal o definitivamente su acceso a la red.

Aprovisionar una red se refiere a su proceso de preparación y a tener desplegado el equipamiento adecuado para poder ofrecer servicios a los clientes. Indudablemente, este equipamiento debe ser accesible por los usuarios para poder consumir los servicios ofrecidos.

El tema de la configuración cobra vital importancia en un sistema de gestión de redes. Es fundamental poder modificar los parámetros de funcionamiento de los nodos de una red (especialmente cuando están en funcionamiento) y saber qué parámetros son los que nos interesa modificar en cada momento, dependiendo de las circunstancias o de los problemas encontrados en el funcionamiento de la red. Por eso, también tiene su utilidad tener una clasificación de los parámetros de configuración disponibles en los equipos de la red, así como una manera estándar de acceder y modificar esos parámetros cuando es necesario.

El mantenimiento de las redes comprende un conjunto de actividades que se suelen realizar de manera periódica para asegurar el buen funcionamiento de la misma. Algunos ejemplos de estas actividades son asegurar las conexiones físicas entre los elementos de red, comprobar el buen funcionamiento de los equipos de forma individual (como los encaminadores y puentes), comprobar si hay servicios o protocolos innecesarios que pueden estar generando tráfico en la red y, por ende, colapsando las comunicaciones, así como verificar el buen funcionamiento de los recursos compartidos por los nodos de la red, como los servidores.

La conectividad nos ofrece una métrica que nos permite medir la facilidad de interconectar varias redes entre sí. Aquí entran en juego temas como las topologías de red, tecnologías y protocolos usados por cada una de las redes, los medios físicos utilizados (cableado o inalámbrico, por ejemplo), así como la disponibilidad de la red hacia sus usuarios.

Por último, con seguridad se quiere dar a entender que se cumple la confidencialidad de los datos (los datos que circulan por la red no podrán ser entendidos por cualquier otro que no sea la fuente y el sumidero de los mismos), la integridad de los datos (se debe asegurar que los datos, si son modificados durante su transcurso por la red, se detectarán en el lado del receptor), autenticidad (debe ser posible identificar el emisor y receptor de los mensajes que pasan por la red), la disponibilidad y el no repudio (se debe poder probar la participación de las partes en una comunicación). Existen varios mecanismos que facilitan el cumplimiento de estas dimensiones, como es el uso de redes privadas virtuales (VPN).

Antes de adentrarnos en los modelos es preciso mencionar el modelo FCAPS (definido en X.700), que es una clasificación en áreas funcionales de las tareas que es necesario realizar en la gestión de redes de telecomunicación. Trata temas como la gestión de fallos, administración de la configuración de los equipos, gestión de la contabilidad de los recursos de la red y de la tarificación, el rendimiento o aprovisionamiento mediante la medida de ciertos parámetros como el caudal y el retardo que nos permiten asegurar el buen funcionamiento de la red, y aspectos de seguridad.

Como modelos de gestión propiamente dichos, que incluyen un protocolo y modelos de datos propios, tenemos el modelo TMN, de la ITU-T, definido en sus normas M.3000, o el OSI-NMM propuesto por el IETF (Internet Engineering Task Force), entre otros.

La amplia variedad de estándares para la gestión de redes tiene como consecuencia una enorme heterogeneidad en los equipos de una red, debido a que cada uno de ellos puede cumplir un modelo distinto y, por tanto, soportar protocolos o simplemente formas de trabajar distintas. Esto ocasiona un enorme problema cuando queremos interconectar varias redes y gestionarlas de manera uniforme, debido a que no hay suficientes estándares universales en este ámbito. De forma deseable, los sistemas de gestión deberían soportar una gran variedad de tecnologías y poder operar de forma transparente sobre ellas (Sinche, Soraya et al. "A Survey of IoT Management Protocols and Frameworks" IEEE Communications Surveys & Tutorials (2019) [19]).

Por esa razón, en los últimos años se ha venido intentando usar protocolos y herramientas comunes para gestionar las redes. Algunos ejemplos pueden ser los protocolos RESTCONF o LwM2M para dispositivos IoT (Internet of Things) que intentan usar protocolos comunes como HTTP en el primer caso o CoAP en el segundo. Sin embargo, algunas alternativas siguen gozando más popularidad que otras, así que tener una cierta homogeneidad en los equipos es una ardua tarea que requerirá el diseño de nuevos estándares que se adapten a las necesidades de los gestores de red.

Además, estos nuevos protocolos de gestión tienen cabida en aplicaciones de gestión localizadas en la nube para administrar dispositivos de todo tipo, especialmente dispositivos con pocos recursos. Algunos ejemplos son Azure en Microsoft, Amazon Web Services (AWS) en Amazon…

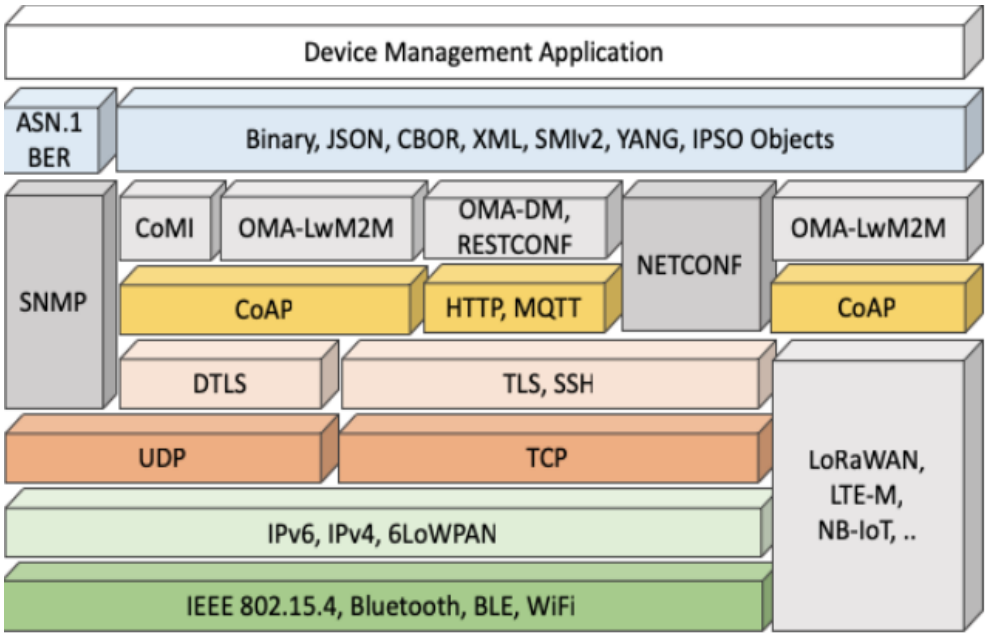

Figura 2- Torre de protocolos de gestión para dispositivos IoT [19]

Como se puede observar en la imagen, la cantidad de protocolos de gestión de redes es inmenso. Por eso, en este proyecto nos centraremos en implementar algunos de ellos que pueden tener más relevancia en la mayor parte de los escenarios. Cabe destacar que bastantes de estos protocolos están pensados para ser usados por dispositivos IoT (en concreto aquellos que funcionan sobre CoAP, o incluso RESTCONF y MQTT), aunque pueden ser usados por cualquier dispositivo en la práctica. Veamos a continuación con un poco más de detalle los protocolos y modelos de datos que se han usado en este proyecto.

2.2 ASN.1

En una aplicación distribuida, es necesario poseer una manera común de representar la información que se intercambia entre los equipos que ejecutan dicha aplicación. Esto es debido a que la representación nativa de los datos depende fuertemente del tipo de procesador que usa un sistema (su arquitectura, si es de 32 o 64 bits…), del lenguaje de programación utilizado para desarrollar las aplicaciones, incluso del sistema operativo que se esté usando. Por eso, usar un modelo de datos que se adapte fácilmente y de forma estándar a las distintas representaciones nativas es crucial para asegurar una comunicación sólida entre los equipos.

ASN.1 (Abstract Syntax Notation One) es un lenguaje de descripción de interfaces usado para definir modelos de datos de forma que estos puedan ser codificados y decodificados en un entorno multiplataforma. Está definido por la ITU-T en su norma X.680. Su uso principal es el de representar sintaxis (tipos de datos), valores que toman esos tipos de datos y, por supuesto, mecanismos para codificar y decodificar esos valores para poder ser transmitidos por la red y ser entendidos por cualquier receptor que implemente este estándar.

Para definir la sintaxis y los valores se usa una notación específica de ASN.1. Esta notación permite el uso de varios tipos de datos primitivos y datos estructurados con los que crear estructuras de datos más complejas. Algunos ejemplos de tipos de datos primitivos son las cadenas de texto, los números enteros, los números en coma flotante o de doble precisión, valores booleanos, enumeraciones, elecciones... En cuanto a los datos estructurados, tenemos las secuencias (SEQUENCE), que no es más que la agrupación de varios campos bajo un mismo contenedor de información. También tenemos maneras de definir arreglos (SEQUENCE OF).

```
FooProtocol DEFINITIONS ::= BEGIN

    FooQuestion ::= SEQUENCE {
        trackingNumber INTEGER,
        question       VisibleString
    }

    FooAnswer ::= SEQUENCE {
        questionNumber INTEGER,
        answer         BOOLEAN
    }

END
```

Figura 3- Ejemplo de documento ASN.1

En la anterior figura se observa un ejemplo de documento escrito en la notación ASN.1 que define un protocolo de comunicaciones. Se aprecia que en un mensaje (PDU) de este protocolo se incluye una pregunta, identificada por un número de secuencia y una cadena de texto con la pregunta a realizar, y una respuesta que contiene el identificador de la pregunta a la que hace referencia y la subsiguiente respuesta, y la respuesta que puede ser sí o no.

```
myQuestion FooQuestion ::= {
    trackingNumber     5,
    question           "Anybody there?"
}
```

Figura 4- Ejemplo de instancias en ASN.1

Una vez definida la sintaxis de este protocolo de prueba, podemos en el mismo documento definir valores concretos (instancias) de los tipos de datos que hemos definido anteriormente. En este caso, hemos definido una pregunta de acuerdo a nuestras definiciones anteriores. Se ha identificado una pregunta con el número 5, y la pregunta se trata de si hay alguien ahí.

Por supuesto, esto es solo una introducción al modelo de datos ASN.1 para ver y comprender su funcionamiento básico. Este modelo posee muchas más posibilidades que permiten adecuarlo a prácticamente cualquier representación de la información, por rebuscada que sea, como la definición de macros, asignación de etiquetas, referenciación a otros documentos…

2.2.1 Codificación BER

Como venimos explicando en el apartado anterior, una vez que tenemos la sintaxis a usar y los valores concretos usando esa sintaxis hay que codificarlos de forma que sean aptos para enviarlos por la red y puedan ser entendidos por los receptores de esos datos. El modelo ASN.1 proporciona diversos mecanismos para codificar y decodificar la información.

Algunos de ellos son GSER (Generic String Encoding Rules) que permiten serializar la información en una cadena de texto legible para las personas, PER (Packed Encoding Rules) que permite una codificación compacta de la información, OER (Octet Encoding Rules) que permite reducir el ancho de banda de las comunicaciones debido a que usa menos memoria que cualquiera de los anteriores en la mayoría de los casos, o incluso JER (JSON Encoding Rules) que permiten codificar los datos en formato JSON. Como vemos, la variedad de formas de codificación es muy grande y adaptable a cada necesidad, pero en este proyecto nos centraremos en hablar de las BER (Basic Encoding Rules) que son las que usan protocolos como SNMP de los que se hablará más adelante en este documento.

Las BER son unas reglas de codificación que serializan la información en tuplas tipo-longitud-valor (TLV). Esto quiere decir que, para cada dato que se quiera codificar, se almacenará el tipo de dato de ese valor, la longitud en bytes del mismo y, por último, el valor que toma el mismo (que dependerá del tipo de dato que se

trate). Como vemos, es una manera de codificar muy sencilla de implementar y que permite una alta adaptabilidad.

Et. Universal	Tipo ASN.1	Et. Universal	Tipo ASN.1
1	BOOLEAN	18	NumericString
2	INTEGER	19	PrintableString
3	BIT STRING	20	TeletexString
4	OCTET STRING	21	VideotexString
5	NULL	22	IA5String
6	OBJECT IDENTIFIER	23	UTCTime
7	Object Descriptor	24	GeneralizadTime
8	EXTERNAL	25	GraphicsString
9	REAL	26	VisibleString
10	ENUMERATED	27	GeneralString
16	SEQUENCE, SEQUENCE OF	28	CharacterString
17	SET, SET OF		

Figura 5- Identificación de los tipos de datos ASN.1 en las BER

En la figura anterior podemos observar que a cada tipo de dato primitivo o estructurado de ASN.1 se le asocia una etiqueta que servirá para identificar cada uno de ellos en un mensaje codificado según las BER. Además, el valor de cada tipo de dato se codificará de forma distinta según la etiqueta que posea, que también los define la norma.

Una descripción más exhaustiva de cómo se codifica la información usando estas reglas puede encontrarse en la norma X.690 de la ITU-T donde se explican las BER, entre otros.

2.3 SNMP

SNMP (Simple Network Management Protocol) es un protocolo de gestión de redes definido por el IETF en las RFC 1065-1067, 1155-1157 en su primera versión. Con el paso del tiempo, este protocolo ha ido adquiriendo mejoras para solventar problemas como la seguridad que se han ido definiendo en RFC posteriores.

Antes de entrar de lleno en su funcionamiento, es necesario definir una serie de conceptos que se usan no sólo en el protocolo SNMP, sino en bastantes protocolos y sistemas de gestión en general.

Para poder gestionar los elementos de una red, primero debemos saber qué aspectos o características de un nodo de la red queremos acceder o modificar. Para eso, debemos abstraer esas características a lo que llamamos un objeto de gestión (managed object). A la hora de definir un objeto de gestión, es importante detallar la descripción de ese objeto (qué representa en la realidad o cuál es su finalidad), así como la sintaxis (tipo de datos) que se usará para representar ese objeto de gestión con el fin de poder ser transmitido por la red. Por otro lado, necesitamos una forma de identificar unívocamente a cada objeto de gestión para poder saber a qué objeto de gestión nos referimos en cada momento.

Una vez tenemos definidos nuestros objetos de gestión, es crucial introducir los conceptos de gestor y agente. Un agente es una entidad que tiene acceso directo al equipo o nodo gestionado y puede modificar su funcionamiento de forma directa. Normalmente, un agente suele ser un servidor que escucha peticiones en una máquina remota. Estas peticiones las envían los gestores, indicando a qué objeto de gestión quieren acceder, si quieren obtener su valor o escribir uno nuevo, entre otros parámetros. El agente, cuando recibe estas peticiones, si el gestor posee los permisos necesarios, realizará las operaciones que indiquen estas, modificando, posiblemente, el comportamiento del nodo de la red bajo gestión.

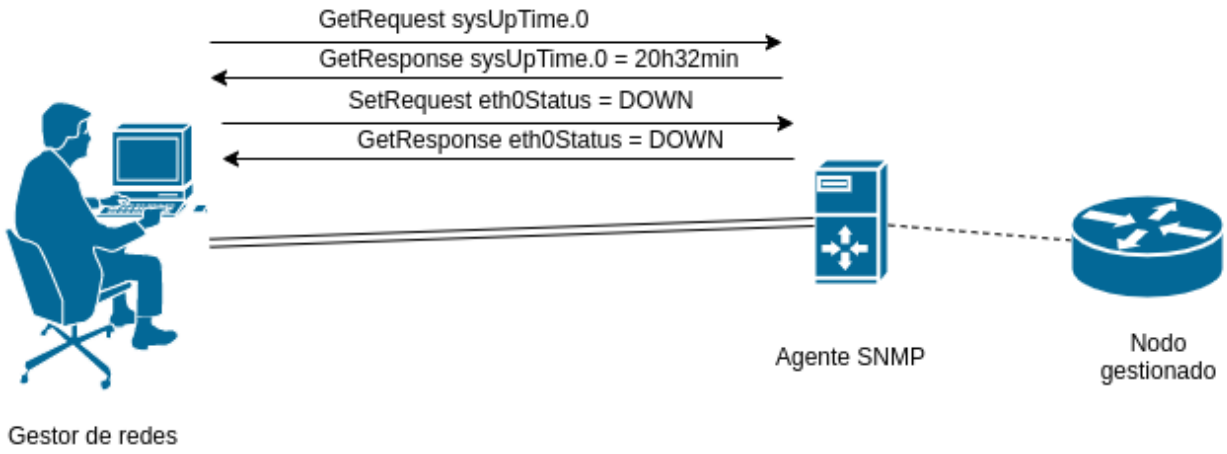

Figura 6- Agente y Gestor SNMP

Indudablemente, los objetos de gestión definidos anteriormente deben ser almacenados de forma uniforme y deben estar disponibles tanto para el agente como para el gestor. El agente necesita saber la definición de los objetos de gestión para entender qué es lo que se está recibiendo de las peticiones de los gestores. Por otro lado, los gestores necesitan saber a qué objetos de gestión pueden acceder, cuál es su utilidad y cuál es su identificación. Para lograr este doble propósito, la definición de los objetos de gestión se empaquetan en lo que se conoce como una MIB (Management Information Base), que es un documento o base de datos que contiene los objetos de gestión que son accesibles en un nodo de la red, siendo implementados por un agente.

En el caso del protocolo SNMP, estas MIB son documentos en notación ASN.1 que contienen la definición de los objetos de gestión que son encapsulados en un módulo, atendiendo a su temática. Con esto, por ejemplo, podemos tener un módulo de conexiones TCP, otro que defina las interfaces de un nodo… En cuanto a la identificación de los objetos de gestión, se utilizan los OID (Object Identifiers). Un OID es un tipo primitivo del modelo de datos ASN.1 que consiste en una ristra de números separados por puntos. Por ejemplo, el OID 1.3.6.1.2.1.1.3 hace referencia al objeto de gestión que indica cuánto tiempo un equipo ha estado en funcionamiento, como se aprecia en la siguiente figura.

```
sysUpTime OBJECT-TYPE
    SYNTAX      TimeTicks
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "The time (in hundredths of a second) since the
            network management portion of the system was last
            re-initialized."
    ::= { system 3 }
```

Figura 7- Extracto de una MIB

Como se observa, el OID de un objeto de gestión no se expresa tal cual con todos sus números, sino que se hace referencia a otro OID y a este se le añade un número más, en este caso el 3. Esto quiere decir que antes de este objeto de gestión había otro llamado “system” y de identificación 1.3.6.1.2.1.1, y así sucesivamente. Esta manera de organizar los objetos de gestión nos da como resultado una estructura en árbol que nos permite ver de forma jerárquica los objetos de gestión que componen una MIB. Además del OID, en un objeto de gestión se indica, como ya se comentó, la sintaxis o tipo de dato usado para este objeto (en este caso es un tipo de dato TimeTicks, definido en algún otro lugar de la MIB o en otra MIB referenciada por esta), el acceso que se puede dar para este objeto (solo lectura en este caso), el estado del objeto y una descripción que nos informa del propósito de este objeto.

En el caso de SNMP, existe una MIB especial que define los tipos de datos que pueden tener un objeto de gestión. Esta MIB se denomina SMI (Structure of Management Information), y está definida en la RFC 1155 para SNMPv1, aunque se ha ido actualizando en subsiguientes RFC cuando han aparecido nuevas versiones

del protocolo. También existe otra MIB, definida en la RFC 1212, que indica de qué forma deben definirse los objetos de gestión, a fin de que la definición de la figura anterior sea válida.

Ahora que hemos introducido los conceptos subyacentes al protocolo SNMP, procederemos a explorar cada una de sus versiones, observando las principales características y problemas de cada una de ellas.

2.3.1 SNMPv1

Esta es la primera versión del protocolo, definida en la RFC 1157, que nos proporciona un conjunto de operaciones (encapsuladas en A-PDUs) que se transmiten entre entidades de gestión (agentes y gestores). En esta versión, se soportan operaciones de lectura de objetos de gestión (GET), escritura de objetos de gestión (SET) y lectura del siguiente objeto de gestión en orden lexicográfico (GETNEXT). Estas operaciones se encapsulan en una PDU que el gestor envía y el agente recibe. Si el gestor posee los permisos y credenciales necesarias, el agente realiza la operación solicitada y envía un mensaje de respuesta (GET-RESPONSE) indicando los resultados de la operación, o detalles sobre los errores sucedidos, si hubieran.

Adicionalmente, un agente puede enviar mensajes a un gestor o conjunto de gestores sin que exista una petición previa. Estos son los denominados traps, que son PDUs cuya función es informar a los gestores cuando ha ocurrido un evento de relevancia en el nodo gestionado. El gestor, al recibir un mensaje de este tipo, puede tomar decisiones de forma rápida y eficaz. Estos mensajes son útiles cuando se quiere notificar de errores en la red o eventos de importancia para el gestor sin que tengan que ser críticos.

En la siguiente tabla se resumen las operaciones soportadas por el protocolo SNMPv1, donde se expresan los campos que se incluyen en estas PDU. Observemos que se incluye en las peticiones un identificador de petición que sirve de número de secuencia para detectar peticiones duplicadas. En la PDU de respuesta, se observan los errores ocurridos.

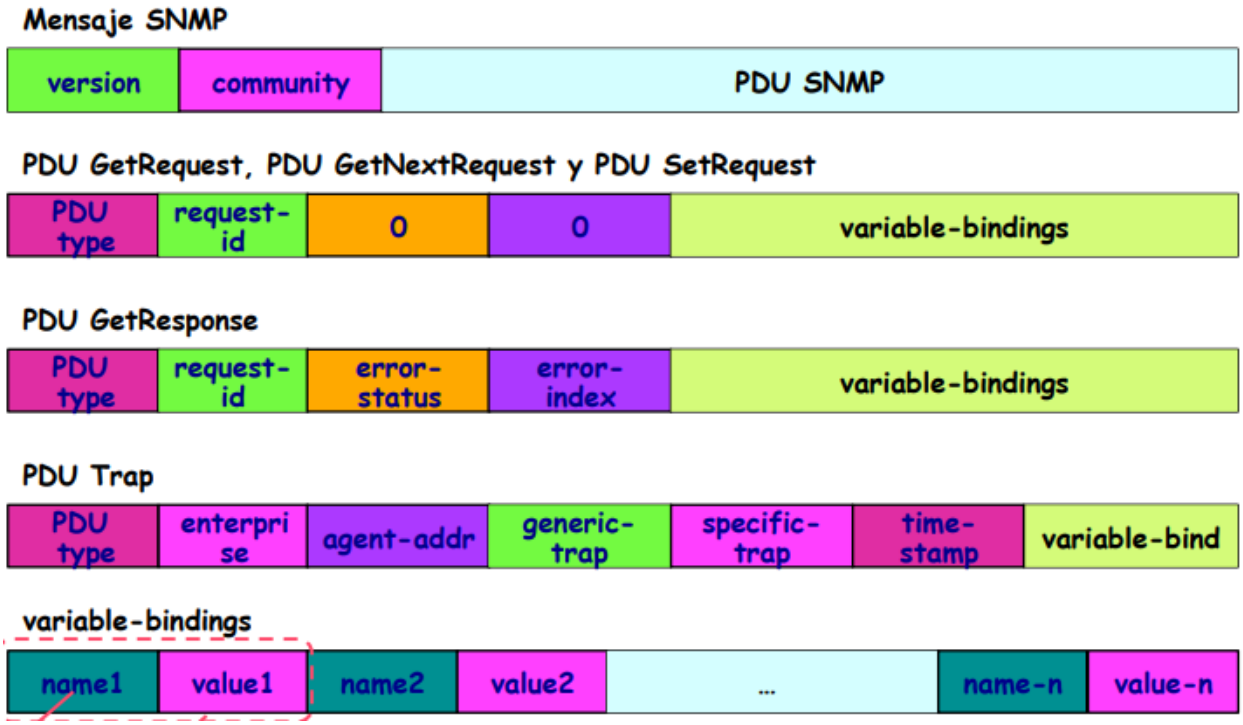

Figura 8- PDUs del protocolo SNMPv1

El campo Variablebindings hace referencia a los objetos de gestión que se quieren acceder, indicando el OID y el valor de dicho campo. En el caso de peticiones GET y GETNEXT, el valor será nulo (tipo de dato NULL). Por último, un trap incluye información como el identificador de la empresa, la dirección IP del agente, el tipo de notificación, cuándo ha ocurrido este evento y objetos de gestión adicionales para la descripción de la notificación en cuestión.

Este protocolo, al ser bastante simple tanto en su estructura como en su funcionamiento, posee una serie de

carencias que puede comprometer temas como la seguridad de los mensajes. El método de autenticación usado en este protocolo es mediante comunidades. Una comunidad es una cadena de texto que se incluye en todas las PDU (campo Community de la figura anterior), de forma que el agente proporciona unos permisos u otros dependiendo de la comunidad que se refiera la petición. El problema es que las peticiones SNMP (como sus respuestas y los traps) se mandan en claro por la red, de forma que un ataque como usar un sniffer de paquetes puede obtener el nombre de la comunidad y, por ende, los permisos de esa comunidad.

Otro de sus principales problemas es que los trap no son asentidos, ya que las PDU del protocolo SNMP se envían por puerto 161 usando transporte UDP (162 para los traps), de forma que un trap puede descartado en la red, no llegar a su destino y ni el agente ni mucho menos el gestor son capaces de saber si ha llegado o no. Estos traps pueden contener información vital para la detección precoz de fallos en la red y su descarte puede suponer pérdidas para una organización.

Por último, SNMPv1 no posee ningún mecanismo de cifrado que asegure ni la confidencialidad, ni la autenticidad ni la integridad de los datos. Por estas, esta versión del protocolo es bastante vulnerable y ha sido necesario actualizarlo en reiteradas ocasiones para que pueda seguir siendo usable.

2.3.2 SNMPv2

Esta nueva versión del protocolo, pese a las mejoras implementadas que solucionan varios problemas de la versión anterior, ha sido bastante controversial debido a la excesiva complejidad de los mecanismos de seguridad implementados. Se ha definido en las RFC 1441 y 1452 en su versión oficial. Nos referimos a versión oficial porque fue la primera que el IETF publicó, aunque han aparecido otras versiones como SNMPv2c (versión de comunidad) que ofrece las nuevas características de la versión oficial pero eliminando por completo el complejo sistema de seguridad sustituyéndolo por la autenticación basada en comunidades de SNMPv1. Actualmente, esta versión es la que se implementa en la mayoría de agentes SNMP (por compatibilidad), y es la versión que se ha dado soporte en este proyecto.

Aparte de las nefastas características de seguridad, esta versión del protocolo incluye otras mejoras. Una de ellas es la inclusión de contadores de 64 bits como nuevo tipo de dato para los objetos de gestión. Esto es útil, por ejemplo, para contar el número de octetos o paquetes recibidos por una interfaz donde 32 bits pueden ser insuficientes en algunos casos. Además, se han definido dos nuevas PDU: GetBulk e Inform.

La primera de ellas, GetBulk, supone un gran avance para leer objetos de gestión de forma más eficiente y que involucre un menor número de peticiones al agente. Su principal uso es la lectura de tablas donde con una sola petición podemos leer filas enteras, mientras que en la versión 1 del protocolo debíamos hacer una petición GetNext por cada fila que queramos leer, en ocasiones sobrecargando al agente.

La segunda de ellas, Inform, proporciona una manera de enviar notificaciones en la que el gestor deba dar un asentimiento al agente para confirmar que se ha recibido correctamente. Esto asegura que los eventos que ocurren en la red son notificados apropiadamente al gestor, aplicando retransmisión si es necesario.

Adicionalmente, se soportan nuevos protocolos de transporte aunque, con las mejoras implementadas, el protocolo UDP sigue siendo el más utilizado por su mayor rapidez.

2.3.3 SNMPv3

La versión 3 del protocolo SNMP, definida en las RFC 3411-3418, es la última versión del protocolo e incorpora, por fin, varios modelos de seguridad que se pueden usar para ofrecer confidencialidad, autenticidad e integridad a los mensajes. En cuanto al funcionamiento básico del protocolo, este apenas varía, manteniéndose las PDU soportadas en versiones anteriores.

Una mejora que es interesante destacar es el uso de VACM (control de acceso basado en vistas), donde en el agente se puede configurar a cada usuario registrado unos permisos para acceder a cada objeto de gestión, permitiéndole o no el acceso.

En cuanto a los modelos de seguridad implementados, tenemos varios disponibles. El primero de ellos es el modelo USM, donde al autenticación se realiza por medio de un usuario y sus correspondientes credenciales, y

otro es el modelo TSM, que permite encapsular los mensajes SNMP en una capa TLS o DTLS (si usamos UDP) para su transmisión segura en la red mediante el uso de certificados X.509. En este proyecto se ha hecho hincapié en el modelo USM debido a su mayor versatilidad, que se explica a continuación.

2.3.3.1 Modelo USM

El modelo USM (User Security Model) es el primer modelo de seguridad que se ha implementado en SNMPv3. Consiste en definir usuarios para autenticar y encriptar una petición SNMP, que el agente almacena en un formato definido en una MIB especial dedicada al modelo USM. En una petición SNMPv3, el gestor debe incluir parámetros como el modelo de seguridad usado, el nombre de usuario, el modo de autenticación usado (que puede ser sin credenciales, sólo autenticación o autenticación y cifrado), y por último incluir una serie de parámetros de seguridad como un hash aplicado sobre la PDU que sirve para asegurar la autenticidad e integridad del mensaje, y una sal usada en el proceso de cifrado, si este decidiera usarse. Una vez almacenados estos parámetros, los datos de la petición son encapsulados y, posiblemente, cifrados en una secuencia ASN.1 aparte.

Este modelo de seguridad soporta una serie de algoritmos de resumen y de cifrado definidos en la RFC 3414. Los más básicos son el algoritmo MD5 y SHA1 para la autenticación del mensaje y DES para la encriptación, aunque la norma indica que puede incluirse cualquier algoritmo que se desee. Por cuestiones de simplicidad, en este proyecto se han implementado estos algoritmos únicamente, aunque es posible añadir sin mayor esfuerzo algoritmos más sofisticados y, ciertamente, más seguros, como son SHA256 o AES.

2.4 IETF Syslog

Syslog es un estándar de facto para el envío de mensajes de registro en una red IP. Estos mensajes de registro son útiles para notificar a un destino información como los intentos de acceso al sistema, anomalías en el funcionamiento normal del sistema o en algún servicio concreto, información sobre las actividades del sistema operativo o incluso errores en el hardware y software del dispositivo en cuestión.

En este estándar suelen formar parte un servidor de syslog que suele ser el destinatario de los mensajes de registro, que los almacena para ser consultados por, típicamente, un gestor a fin de que pueda mejorar su toma de decisiones para solucionar errores en la red o incluso mejorar su funcionamiento. Estos mensajes son enviados por un servicio que se aloja en el dispositivo gestionado, que generalmente lee notificaciones de un fichero de registro (como puede ser /var/log/messages en sistemas Linux), los encapsula en un mensaje Syslog y, finalmente, los envía por la red a unos destinatarios que tiene previamente configurados.

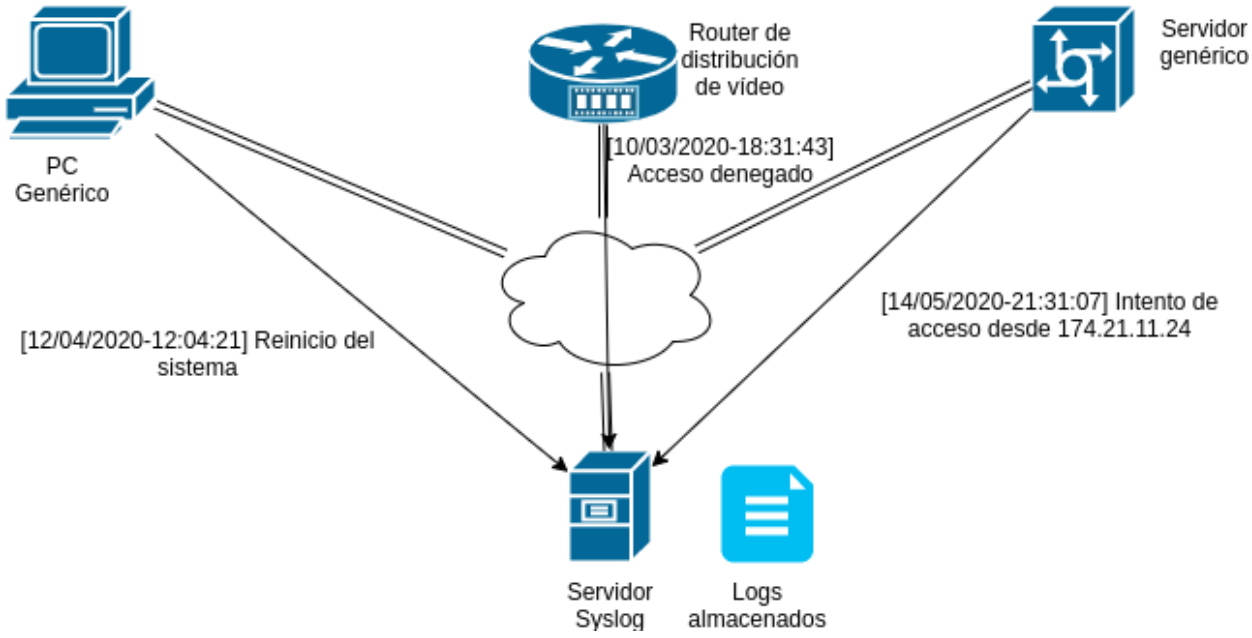

Figura 9- Escenario de uso del protocolo Syslog

En esta figura es posible observar una serie de equipos de diversa funcionalidad que tienen desplegado un servicio Syslog que se encarga de enviar registros a un servidor remoto ante la ocurrencia de algún evento. Este servidor simplemente recolecta los registros recibidos y los almacena en su fichero para su posterior revisión por parte del gestor de redes o del administrador de sistemas.

En cuanto a servicios Syslog, existe una amplia variedad en el mercado, y especialmente de software libre que se suelen incluir por defecto en las distribuciones Linux. Algunos ejemplos son syslog-ng (el usado en este proyecto para realizar pruebas de funcionamiento), SolarWinds Kiwi Syslog Server, Splunk Light, rsyslogd, entre otros.

Existe una inmensa variedad de formatos de mensajes para enviar mensajes syslog por una red IP, todos definidos por su propio estándar. El primero que apareció fue el definido en la RFC 3164, que habla del protocolo Syslog BSD, el más tradicional de todos. Aunque sigue siendo usado como un protocolo por defecto en la mayoría de servicios Syslog, es más conveniente usar protocolos más modernos que son capaces de incluir más información útil para el gestor. Por ejemplo, tenemos el estándar oficial del IETF, que es el que se ha seleccionado para este proyecto. Se define en la RFC 5424, y es el protocolo preferido para dispositivos que necesitan ser gestionados, como routers y switches.

En cuanto a la información incluida en este protocolo, se incluyen datos como el nombre del equipo, el nombre de la aplicación y el identificador de proceso que ha generado el registro. Además, incluye un formato de marcas de tiempo bastante preciso, indicando fecha y hora hasta una precisión de centésimas de segundo, además de la zona horaria. Tras estos parámetros iniciales, se adjunta información específica de la aplicación que genera el registro, encapsulada en tuplas clave-valor. Finalmente, se puede incluir de forma opcional un mensaje descriptivo del mensaje de registro que resume qué es lo que contiene el mismo, o las causas de su generación.

Finalmente, estos mensajes de registro pueden transmitirse sobre cualquier capa de transporte. Las más populares suelen ser TCP, UDP para mensajes no críticos, o DTLS y TLS para agregar confidencialidad, autenticidad e integridad a los mensajes, usando opcionalmente certificados X.509.

2.5 SSH

SSH (Secure Shell o intérprete de órdenes seguro) es el nombre del protocolo y del programa que lo implementa cuya principal función es el acceso remoto a un servidor o equipo cualquiera por medio de un

canal seguro en el que toda la información está cifrada. No sólo se limita a la ejecución de comandos de forma remota, sino que además es posible transferir archivos de cliente a servidor y viceversa, e incluso redirigir ventanas gráficas en X para poder visualizarlas desde el cliente, aunque sea el servidor el que las esté gestionando. El puerto estándar usado de este protocolo es el 22 de TCP.

La razón principal de la aparición de este protocolo seguro es la sustitución de protocolos de funcionalidad simular pero que enviaban los comandos en claro por la red, como son el caso de Telnet, definido en la RFC 854, que los deja completamente obsoletos, exceptuando casos donde la comunicación entre cliente y servidor sea en un entorno controlado o en un laboratorio de pruebas.

SSH, definida en las RFC 4250-4256, entre otras, tiene 2 versiones principales. La primera versión hace uso de algunos algoritmos de cifrado de uso comercial (patentados) y contiene una vulnerabilidad que permite a un usuario insertar datos en el flujo TCP de la comunicación. Para solucionar estos errores y hacer que el protocolo sea más open-source se implementó una segunda versión en la suite OpenSSH de Red Hat Enterprise Linux, conteniendo esta vez un mejor algoritmo de intercambio de claves que soluciona la vulnerabilidad de su primera versión. Esta suite sigue siendo, a día de hoy, el estándar para realizar conexiones remotas a equipos para su configuración tanto en el entorno profesional como de usuario.

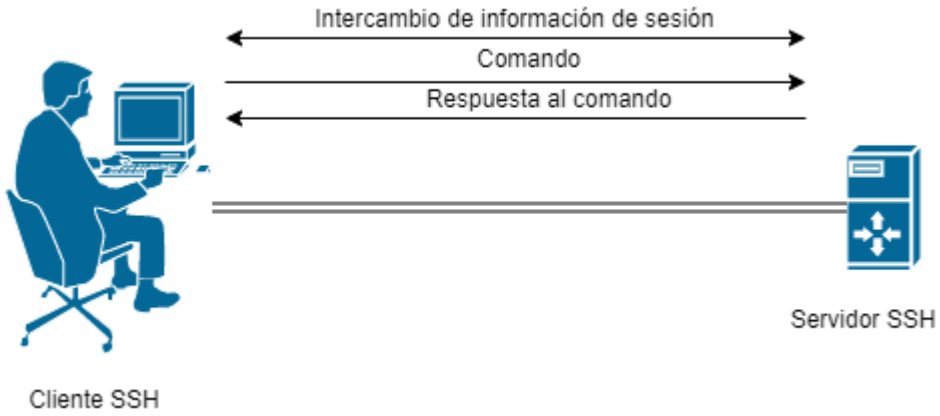

Figura 10- Funcionamiento de SSH

En la figura anterior se puede observar el funcionamiento del protocolo SSH. Primero se debe autenticar al usuario con alguno de los mecanismos que explicaremos a continuación. Si la autenticación es correcta, se procederá a realizar un intercambio de información de la sesión SSH. Una vez establecida la sesión, el cliente podrá enviar comandos de terminal al servidor, que este ejecutará y enviará de vuelta al cliente la respuesta.

Por último, este protocolo permite diversas formas de autenticación, aunque claramente deben estar soportadas por el servidor SSH utilizado. Al establecer la sesión SSH, este servidor envía al cliente un listado de los métodos de autenticación disponibles, de los que el cliente debe escoger uno. Los más utilizados suelen ser el clásico usuario-contraseña, aunque existen otros como son mediante clave pública o mediante certificados.

2.6 REST CONF

REST CONF es un protocolo basado en HTTP que es la evolución natural del protocolo de gestión de la configuración NetConf. Está definido en la RFC 8040. Su principal utilidad es proporcionar mecanismos a las aplicaciones web de acceder a los datos de configuración de un dispositivo gestionado, información sobre su estado o incluso las llamadas de procedimiento remoto (RPC) soportadas de una manera modular y extensible.

De forma similar a como trabaja NetConf, este protocolo usa un modelo de datos YANG, tanto en su versión 1.0 definida en la RFC 6020 como en su versión 1.1 definida en la RFC 7950 para almacenar los recursos que son accesibles a un cliente, de qué manera lo son (solo lectura, lectura y escritura…), una descripción de ese recurso y una identificación unívoca del mismo mediante una URI. Si hacemos una comparación, esta manera de proceder es bastante similar a las MIB del protocolo SNMP.

En cuanto a los documentos YANG soportados, sirven exactamente los mismos que se han estado usando para

el protocolo NetConf, simplemente se modifica la manera de acceder, que en este caso es mediante peticiones HTTP a una URI que identifica al recurso accedido. Como era de esperar, REST CONF permite usar todas las características implícitas en el protocolo HTTP, como la autenticación (básica o digest), encapsulación de la petición en una capa SSL o TLS usando HTTPS, inclusión de cabeceras para indicar el formato de la respuesta requerido (los más comunes suelen ser documentos JSON o XML), así como indicar opciones adicionales al servidor mediante parámetros del método HTTP GET.

Mientras que NetConf definía en su protocolo métodos para realizar operaciones de creación, lectura, actualización y borrado (CRUD), REST CONF enlaza estas operaciones a métodos HTTP para asegurar la compatibilidad con NetConf. Por ejemplo, la lectura de recursos se realiza mediante peticiones GET, la escritura o la llamada a procedimientos remotos mediante peticiones POST, el borrado de elementos (en un recurso tipo lista) mediante una petición DELETE, entre otros.

2.6.1 Modelo de datos YIN

El modelo de datos YIN es una versión XML del modelo de datos YANG, pensada para facilitar su procesamiento por parte de sistemas embebidos. Al no existir librerías ligeras para la manipulación de los datos en un módulo YANG, la opción más inteligente (y que se ha usado en este proyecto), es convertir los módulos YANG a utilizar a un formato XML debido a que, por su estructura en árbol, ambos formatos son compatibles entre sí. Una vez convertidos los módulos, es bastante sencillo leerlos mediante una librería estándar de ficheros XML, donde podemos obtener la información que necesitemos de los recursos del módulo de una manera sencilla y eficiente.

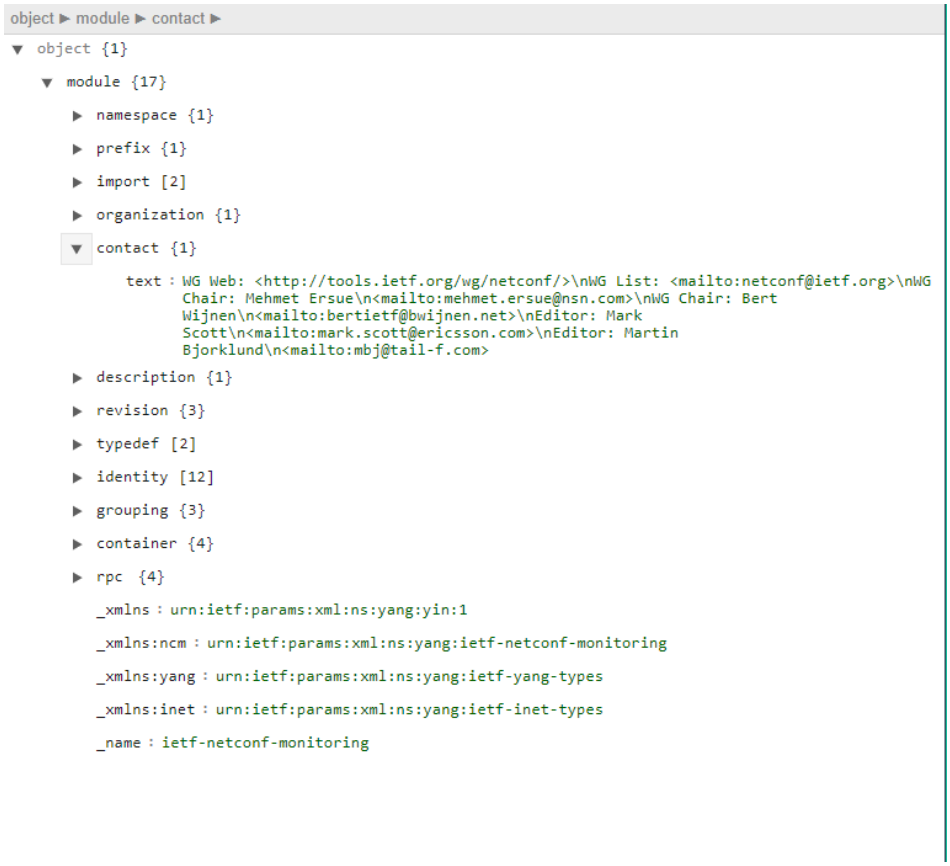

Figura 11- Extracto de un módulo YIN

Figura 12- Extracto de un módulo YANG

Como podemos observar, en la primera figura corresponde a la representación XML de un fragmento de un módulo YANG correspondiente a la monitorización de un equipo usando NetConf. Se puede observar la traducción de las hojas del árbol del módulo YANG en sus correspondientes etiquetas XML que dan lugar al formato YIN.

Como último apunte, es preciso decir que en un módulo YANG los recursos se agrupan mediante contenedores, y cada contenedor puede tener agrupados recursos escalares (que pueden ser accedidos mediante un URI), listas de datos (en las que, además del URI, es necesario indicar qué elemento de la lista se quiere acceder), e incluso otra información como es una descripción del contenedor o llamadas a procedimiento remoto (RPC) soportadas en ese contenedor. Generalmente, un contenedor response a una temática concreta dentro del módulo, y permite una mayor jerarquización de los recursos con el fin de almacenarlos de forma más organizada.

3 TERMINAL Y TECNOLOGÍAS EMPLEADAS

En este capítulo hablaremos más en profundidad sobre la solución propuesta en este proyecto a la problemática introducida en el punto 1. Principalmente daremos una introducción al sistema que se va a usar para ejecutar la aplicación, así como el conjunto de herramientas auxiliares que se han usado para facilitar el diseño y el desarrollo de la misma.

3.1 El sistema Nintendo 3DS

El sistema Nintendo 3DS es una videoconsola de octava generación desarrollada por Nintendo y lanzada en 2011 para el propósito de ofrecer un modo de entretenimiento para todas las edades a un precio equilibrado para el público general. En este caso, no usaremos el sistema con ese fin, sino para convertirlo en una potente herramienta de gestión de redes de telecomunicación.

Figura 13- El sistema Nintendo 3DS

En cuanto a sus características técnicas, este sistema posee un procesador de arquitectura ARM11 de 32 bits con dos núcleos funcionando a 266MHz (aunque algunas revisiones de la consola permiten frecuencias superiores). Como tarjeta gráfica, posee un modelo DMP Pica200 funcionando a 133MHz con una memoria de vídeo asociada de unos 4MB. Permite dibujar en pantalla simultáneamente hasta 15,3 millones de polígonos, destacando por su velocidad, potencia y bajo consumo.

La memoria principal es de unos 128MB en total, de bajo consumo y alta velocidad al tratarse de ser una memoria de tipo Fast Cycle RAM (FCRAM), pudiendo alcanzar velocidades de lectura de hasta 4.2GB/s. Sin embargo, parte de esta memoria está reservada para el sistema operativo, de forma que en este proyecto usaremos la mitad, 64MB, cantidad que parece más que razonable para las tareas que vamos a realizar.

En tema de conectividad, se soporta la tecnología Wi-Fi en la banda de 2.4GHz según los estándares IEEE 802.11b y IEEE 802.11g, con modos de autenticación WPA2-PSK y WPS, aunque puede conectarse a puntos de acceso (AP) abiertos (sin autenticación). Adicionalmente, posee un transmisor y receptor en la banda de infrarrojos para comunicaciones de corto alcance.

También posee una ranura para tarjetas SD para aumentar su capacidad de almacenamiento (que es de unos pocos GB usando la memoria interna del dispositivo). El tamaño máximo es de unos 32GB.

En cuanto a las pantallas, posee 2 pantallas de tipo LCD para mostrar elementos gráficos. La pantalla superior posee una resolución de 400x240 y permite, de forma opcional, usar un segundo framebuffer (zona de

memoria donde se almacena la imagen a mostrar en cada fotograma) de 400x320 para ofrecer gráficos 3D estereoscópicos. En este proyecto esta funcionalidad no nos interesa, luego usaremos la pantalla superior en modo 2D, es decir, mostrando un framebuffer a la vez. En cuanto a la pantalla inferior, tiene una resolución inferior, de 320x240, pero a diferencia de la superior, esta es táctil y no permite mostrar gráficos 3D estereoscópicos.

Finalmente, es un sistema bastante cómodo que pesa aproximadamente 250g en su versión original y de tamaño 13,462cm de ancho, 7,366cm de largo y 1,545cm de grosor, lo que hace posible sostenerlo con las dos manos o incluso con una si lo sostenemos como si fuera un libro. El uso del lápiz táctil facilita mucho la labor de interactuar con las interfaces de usuario asegurando una selección sencilla y precisa.

Tal y como hemos visto, estamos usando un sistema con una potencia bastante considerable para tratarse de un sistema portátil, aunque es bastante eficiente en el tema de gestión de la energía, con una batería que dura hasta 8 horas (incluso más si usamos el ahorro de energía o cerramos la tapa para entrar en el modo sleep).

Especialmente, se trata de un sistema que no suele ser el objetivo para los hackers, aunque algunos estudios han logrado vulnerar su seguridad. En este estudio (McClintic, Matthew, et al. "Keyshuffling Attack for Persistent Early Code Execution in the Nintendo 3DS Secure Bootchain" [20]) se ha logrado, aprovechando una vulnerabilidad en el modo ECB del algoritmo AES, lograr un descifrado erróneo del firmware de la consola, consiguiendo codificar una instrucción de salto a una zona de memoria que contiene un programa desarrollado por los propios hackers que permite modificar el almacén de claves de la consola. Otro ejemplo de ataque es el encontrado en (Seire, Michael, et al. "Attacking the Nintendo 3DS Boot ROMs [21]) que aprovecha un bug en el decodificador ASN.1 implementado en la consola usado para leer la firma digital del firmware de la consola.

A pesar de estos ataques, no se ha encontrado ninguno hasta la fecha que tenga que ver con los protocolos de red implementados en la consola, lo que significa que debemos tener acceso físico a la misma para poder realizar un exploit. Además, si se diera el caso en el futuro, Nintendo lanza actualizaciones periódicas del sistema operativo de la consola que corrigen estas vulnerabilidades y brindan mayor estabilidad al sistema.

3.2 SDK para Nintendo 3DS

Para desarrollar aplicaciones para este sistema, se ha hecho uso de un kit de desarrollo (SDK) de código abierto que permite la programación de bastantes videoconsolas. Su nombre es devkitPro y puede obtenerse de forma gratuita (aunque se aceptan donaciones a los creadores) mediante el enlace en [3].

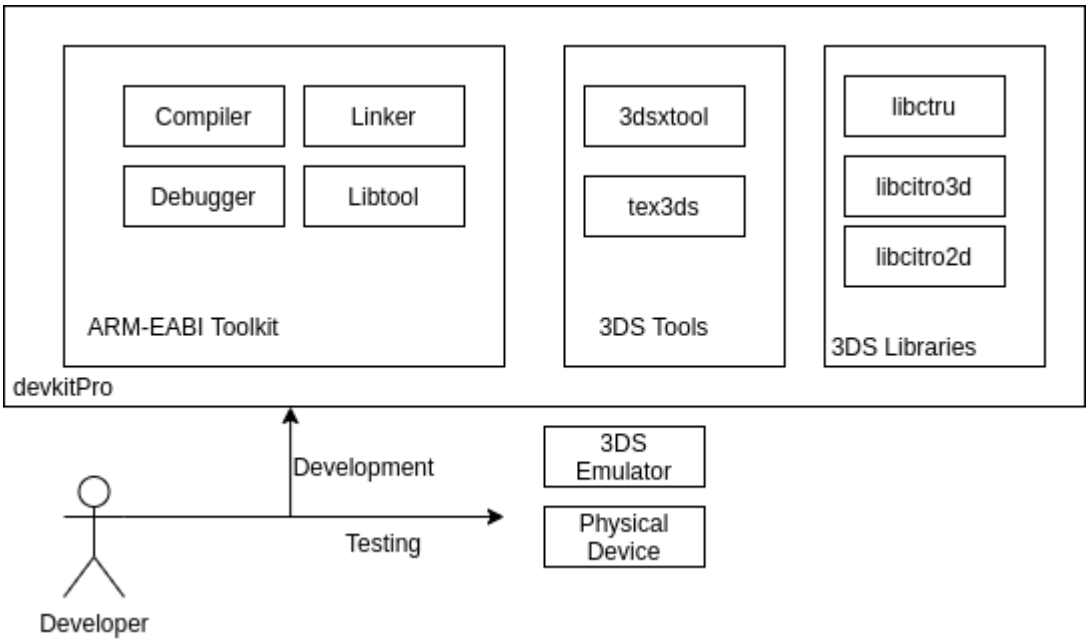

Figura 14- Entorno de desarrollo para Nintendo 3DS

En este kit de desarrollo se adjuntan diversos compiladores basados en GCC para una amplia variedad de arquitecturas, como ARM, PowerPC (PPC), MIPS, entre otros. También posee enlazadores (linkers) y herramientas para generar librerías para cada arquitectura, así como las librerías fundamentales para interactuar con el sistema operativo de cada plataforma. En nuestro caso, usaremos el compilador de ARM11 (aunque tiene incorporadas otras versiones como ARM7 y ARM9) y un conjunto de librerías para facilitar la programación y acceso al sistema operativo del sistema Nintendo 3DS. Además, tenemos a nuestra disposición otro tipo de utilidades para generar el ejecutable final a partir de uno en formato ELF (3dsxtool) o incluso para convertir nuestras imágenes a un formato optimizado para la consola (tex3ds).

Su instalación y actualización es bastante sencilla, debido a que incluye instaladores para Windows y, para Linux, es posible registrar su propio repositorio de pacman para descargar las últimas versiones de las herramientas de desarrollo y librerías. Además, posee un conjunto de librerías de uso común (multiplataforma) que han sido precompiladas para ciertas plataformas. Algunos ejemplos son libpng para la carga de imágenes PNG, libSDL para la gestión de gráficos, etc.

No sólo eso, sino que además contiene una exhaustiva documentación y un gran número de ejemplos para poder familiarizarnos con este SDK de manera rápida. Además, tenemos a nuestra disposición un foro en la página oficial (en [3]) donde podemos preguntar nuestras dudas o incluso publicar nuestras propias creaciones. DevkitPro posee una comunidad bastante sólida en este aspecto.

3.3 Emulador Citra3DS

Una vez que tenemos un kit de desarrollo listo, necesitamos un emulador para poder depurar nuestro código o, simplemente, realizar pruebas sin tener que estar usando constantemente el sistema físico. Para solucionar este problema, se ha decidido usar el emulador Citra3DS que es el pionero en la emulación de este sistema. Es posible obtenerlo para los sistemas operativos más conocidos desde su página oficial en [4].

Figura 15- El emulador Citra3DS en funcionamiento

Este emulador es capaz de simular la inmensa mayoría de características de la consola y, ciertamente, de su sistema operativo. Es capaz de dibujar los elementos gráficos que necesita nuestra aplicación en pantalla, reproducir audio y vídeo, simular los botones de la consola y la pantalla táctil, proveer conectividad a Internet y acceder a servicios del sistema operativo como el navegador web, gestor de notificaciones, tarjeta SD y memoria interna, y mucho más. De forma muy interesante, permite depurar nuestra aplicación desplegando un pequeño servidor de comandos que podemos conectar a gdb.

Sin embargo, el emulador no es perfecto y hay aspectos en los que flaquea, como es la implementación de todas las llamadas al sistema. Existen algunas que no han sido implementados y que, por tanto, tendremos que recurrir al sistema físico si queremos utilizarlas. Algunos ejemplos son el módulo SSL, el teclado software y las notificaciones.

Pese a sus limitaciones, sigue siendo una opción muy recomendable y sofisticada para desarrollar aplicaciones para Nintendo 3DS, que sin duda nos ayudará a optimizar nuestro tiempo de desarrollo en gran medida.

3.4 Librerías adicionales

Como hemos indicado anteriormente, se han usado una serie de librerías para facilitar el desarrollo de la aplicación. Algunas están preinstaladas en devkitPro, mientras que otras se ha tenido que compilar mediante código fuente para la arquitectura ARM11. A continuación se expondrán las librerías usadas y para qué han resultado útiles en el desarrollo de la aplicación.

3.4.1 Librería ctru

La librería ctru es básica para el desarrollo de aplicaciones para Nintendo 3DS. Está incluida en devkitPro y nos proporciona una serie de wrappers (adaptadores) para usar las llamadas al sistema del sistema operativo de la consola en modo usuario. El nombre de la librería viene de CTR que es el nombre técnico del sistema operativo del dispositivo Nintendo 3DS, y la letra "u" de modo usuario.

En este proyecto, no se ha requerido usar llamadas al sistema en modo kernel, de forma que usar una librería en modo usuario proporciona más seguridad al no estar manipulando aspectos que puedan afectar al sistema de forma crítica.

Los servicios principales utilizados de esta librería son el servicio RomFS para almacenar archivos en el propio ejecutable, el servicio SDMC para acceder a la tarjeta SD del dispositivo, el servicio HTTP para realizar peticiones HTTP con los parámetros que deseemos, el servicio de sockets para acceder a Internet, el servicio GX usado para dibujar elementos en pantalla, el servicio NDSP para la reproducción de audio y, por último, servicios adicionales para mostrar un teclado por pantalla o notificar mensajes de error al usuario.

3.4.2 Librería citro2d

Esta librería es bastante útil para el dibujado de primitivas como rectángulos, círculos, triángulos y polígonos en general en dos dimensiones, haciendo uso de la proyección ortográfica para lograrlo. A estas primitivas es posible asignarles colores por vértice e incluso aplicarles texturas en diferentes formatos de píxeles (con transparencia o sin ella). También aporta utilidades para aplicar texturas animadas (con múltiples fotogramas), también conocidas como "texture atlas". Además, permite aplicar transformaciones elementales a las primitivas dibujadas en pantalla, como trasladarlas, rotarlas, escalarlas, deformarlas…

En cuanto al formato de las texturas, es necesario realizar una conversión antes de utilizarlas, usando un programa conversor que viene incluido con devkitPro. Nos permite introducir varias imágenes como entrada, elegir el formato que deseemos, incluso la compresión a usar, soportando algunas como LZ77, Huffman, RLE…

A pesar de que es una librería muy útil, las posibilidades que admite son bastante limitadas y para realizar cosas complejas como animaciones o interfaces gráficas deberemos realizarlo nosotros. No obstante, es una

gran librería que nos ayuda como base para desarrollar aplicaciones gráficas más complicadas. Está incluida en los paquetes esenciales de devkitPro. Su documentación está disponible en [5].

3.4.3 Librería tinyxml2

Esta librería es usada simplemente para leer y escribir ficheros en formato XML. Está desarrollada en C++ y es bastante sencilla de usar y de extraer información específica de un fichero XML. En el proyecto se usa específicamente para la creación de ficheros de definición de vistas en la interfaz gráfica y para la lectura de módulos YIN. Además, posee una excelente documentación lo que hace trivial adaptarse a su uso bajo cualquier circunstancia, que pude consultarse en [6].

Figura 16- Logotipo de tinyxml

3.4.4 Librería jansson

Es una librería cuyo funcionamiento es similar a tinyxml solo que esta permite la lectura y escritura de ficheros en formato JSON. A diferencia de la anterior, esta librería está escrita en C puro, lo que hace más engorroso su manejo, aunque cuenta con utilidades como el conteo de referencias que ayuda a la gestión de la memoria usada para los ficheros JSON.

Esta librería se usa en el proyecto para gestionar las notificaciones y los traps recibidos por la aplicación, almacenándolos en una pequeña base de datos. Además, es primordial su uso para usar el protocolo RESTCONF debido a que los datos intercambiados suelen estar en formato JSON.

Además, está incluida en los paquetes extra de devkitPro, lo que hace su instalación bastante amena, aunque se trata de una librería muy ligera que no suele dar problemas al compilarla en una plataforma distinta a x86. Contiene además una documentación con numerosos ejemplos al grano, que se pueden consultar en [7].

3.4.5 Librería mbedtls

Esta librería (antes denominada PolarSSL) proporciona una suite de algoritmos de cifrado y hashing que resultan bastante útiles para programar modelos de seguridad, como es el caso del modelo USM de SNMPv3 en este proyecto. En concreto, se han usado los algoritmos MD5, SHA1 y DES, aunque proporciona otros más novedosos como AES y versiones superiores de SHA.

También nos permite realizar una gestión y carga de certificados X.509 a fin de que podamos crear sesiones SSL, TLS y DTLS de forma sencilla y rápida. No solo eso, sino que además es una librería optimizada para los sistemas embebidos, y en especial a la arquitectura ARM debido a que esta marca la ha adquirido y es la que le da soporte en la actualidad, optimizando el uso de las instrucciones para los procesadores de esta categoría. Por estas razones, se ha escogido esta librería sobre otras más pesadas y menos optimizadas como openssl para el desarrollo de esta aplicación.

Como las anteriores, es una librería que se incluye en los paquetes extra de devkitPro. Posee una detallada documentación que puede consultarse en [8].

Figura 17- Logotipo de mbedtls

3.4.6 Librería ssh2

Libssh2 es una excelente librería para gestionar conexiones usando el protocolo SSH. Soporta cualquiera de sus versiones y permite múltiples métodos de autenticación. Además, es posible combinarla con diferentes librerías de criptografía, siendo fácilmente adaptable a nuestras necesidades. En nuestro caso, estamos usando libssh2 combinado con mbedtls, aunque puede usarse cualquier otra librería que sea medianamente conocida.

Su uso es bastante intuitivo y nos permite una amplia variedad de opciones como seleccionar el tipo de terminal que el servidor nos debe proporcionar, establecer el timeout, proporcionar un canal de lectura y escritura síncrono o asíncrono, entre muchas otras.

Esta librería no está incluida en los paquetes esenciales de devkitPro, aunque su compilación en la arquitectura ARM es muy sencilla. Su documentación está disponible en [9].

Figura 18- Logotipo de libssh2

3.4.7 Librería tmt

Esta librería es bastante simple y es usada para emular un terminal de comandos, es decir, recibe cadenas de texto con secuencias de escape usadas en los terminales, como posicionar el cursor en una determinada posición, limpiar la pantalla, almacenar el color de cada carácter del terminal, si está en negrita, cursiva, su color de fondo… Además nos permite configurar el tamaño en celdas del terminal, haciéndolo adaptable a cualquier tipo de pantalla.

Cabe destacar que no es una librería gráfica, de forma que la matriz de caracteres que nos ofrece esta librería como salida deberemos usarla nosotros para dibujar, mediante el uso de una fuente de texto del tamaño apropiado, el terminal de forma que el usuario pueda interactuar con él.

Es una librería tan sencilla que solo viendo un ejemplo se puede observar prácticamente todo su funcionamiento. El código de esta librería está disponible en [10].

3.5 Git y GitHub

Git es una herramienta usada principalmente por desarrolladores que permite gestionar las versiones de un proyecto. Nos proporciona una serie de comandos para crear repositorios locales e ir almacenando cada versión de nuestro proyecto en una pequeña base de datos, que luego podemos subir a servidores de repositorios remotos como es GitHub.

En cuanto a las operaciones más usuales, tenemos la posibilidad de crear nuevas versiones del proyecto, ir hacia atrás en el caso de que hayamos cometido algún error, asignar etiquetas a cada versión, crear ramas (branches) que son especialmente útiles para fomentar el trabajo en equipo, donde cada miembro del equipo puede desarrollar la aplicación en su propia rama e ir fusionando ramas cuando van terminando las características que les toca desarrollar, dando como resultado la aplicación final.

Es una herramienta disponible en prácticamente cualquier sistema operativo y, una vez entendido su funcionamiento, resulta bastante cómoda de usar y permite una mejor y más eficiente gestión de nuestro

proyecto.

El código de esta aplicación se ha ido almacenando en un repositorio de Git que, posteriormente, se ha ido subiendo a GitHub. Es posible consultarlo en [11].

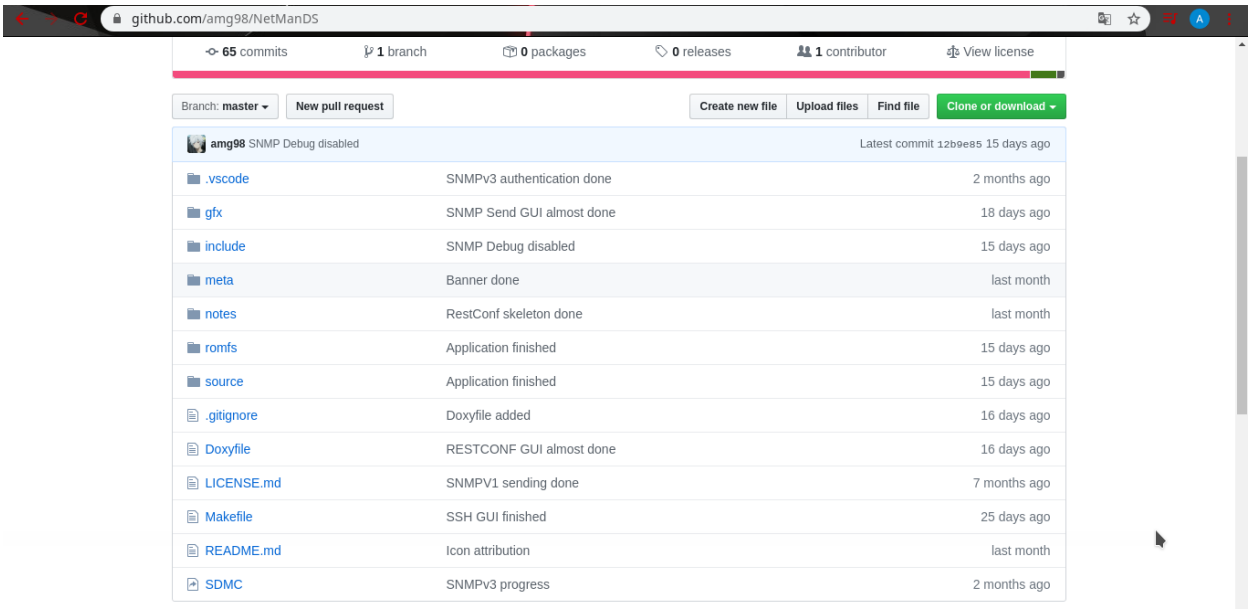

Figura 19- Vista previa del repositorio del proyecto

Como podemos observar en la figura, GitHub nos permite ver, para para archivo o directorio del proyecto, su fecha de última modificación, la descripción del "commit" en el cual ha sido modificado por última vez, así como la posibilidad de navegar por todas sus versiones, pudiendo ver los cambios realizados de una versión a otra.

3.6 Visual Studio Code

Como última herramienta, se ha decidido usar Visual Studio Code (VSCode), un IDE (entorno de desarrollo integrado) desarrollado por Microsoft que nos permite gestionar la compilación y ejecución de cualquier proyecto software, sin importar las tecnologías que se estén usando.

Es un IDE que esta gozando de una gran popularidad entre los usuarios. Principalmente, es una herramienta bastante pulida que funciona a gran velocidad, con unos tiempos de carga prácticamente imperceptibles. Además, nos permite personalizarlo a nuestro gusto, pudiendo hacer uso de una amplia biblioteca de plugins y addons que se nos proporciona de manera completamente gratuita. Aquí tenemos una gran cantidad de aditivos que nos pueden servir para casi cualquier circunstancia. Contiene plugins para los lenguajes de programación más conocidos, como C, C++, Java, Python… e incluso para las tecnologías web más demandadas, como plugins para Angular y React. Además, podemos integrar Git en este IDE con pocos clicks.

Por si no fuera poco, tiene un sistema de Intellisense y autocompletado de código bastante elegante y eficiente, que podemos configurar hasta el más mínimo detalle. La creación de plantillas, generación de tests para la aplicación y la integración con kits de desarrollo (SDK) es bastante sencillo de realizar en este IDE, donde podemos tener todas las herramientas que necesitemos para nuestra aplicación en la palma de nuestra mano.

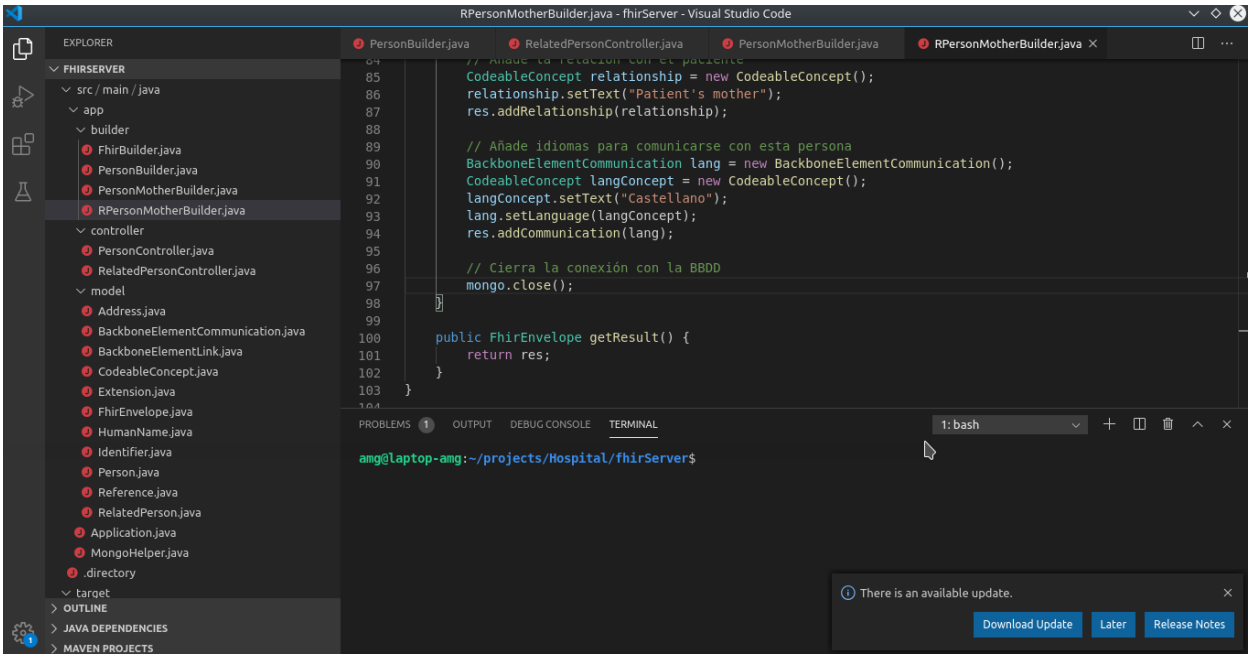

Figura 20- El IDE Visual Studio Code

Como se observa en la figura, este IDE posee una interfaz básica con 3 elementos principales: un editor de código con realzado de sintaxis (syntax highlighting) y autocompletado de código (IntelliSense), un navegador donde podemos explorar el proyecto, así como un terminal donde ejecutar comandos. Además, posee numerosos atajos de teclado que podemos personalizar a nuestro gusto para realizar tareas concretas o ejecutar scripts que facilitan el desarrollo de la aplicación.

4 DISEÑO DE LA SOLUCIÓN

En este apartado se pretende ofrecer el proceso de diseño de esta aplicación dedicada a la gestión de redes de telecomunicación previo a su implementación. Para ello, exploraremos los diferentes aspectos que se han tenido en cuenta a fin de que el desarrollo de la aplicación sea más sencillo y obtenga una mayor modularidad, mayor organización y, ciertamente, tener una mejor visión sobre los objetivos de la aplicación en sí.

Por esa razón, primero iremos explorando los diagramas de casos de uso para comprender la interacción de los actores con las diferentes funcionalidades de la aplicación. Seguidamente, abarcaremos la especificación de requisitos del proyecto, donde trataremos todas las características que deben estar o son deseables que estén en la aplicación según los casos de uso especificados. Una vez concretados los casos de uso y los requisitos, abordaremos una serie de diagramas que facilitarán la comprensión y la implementación del proyecto software, como los diagramas de clases para tener una visión general de la estructura del código y de los patrones de diseño utilizados. Por último, veremos algunos diagramas de actividad que aclaran algunos aspectos sobre el comportamiento dinámico de la aplicación.

Cabe destacar que, para la especificación de requisitos, se ha seguido en parte el estándar propuesto por MADEJA (Marco de Desarrollo de la Junta de Andalucía), que es accesible desde [12]. No se ha realizado de manera completa para no extender esta memoria más de lo que se debería.

4.1 Diagramas de casos de uso

Los diagramas de casos de uso tienen por objetivo mostrar las funcionalidades del sistema software que representan, y qué actores interactúan con estas funcionalidades. Además, se tiende a aumentar la expresividad del diagrama mostrando las relaciones entre los casos de uso, con aspectos como las inclusiones, extensiones e incluso herencias.

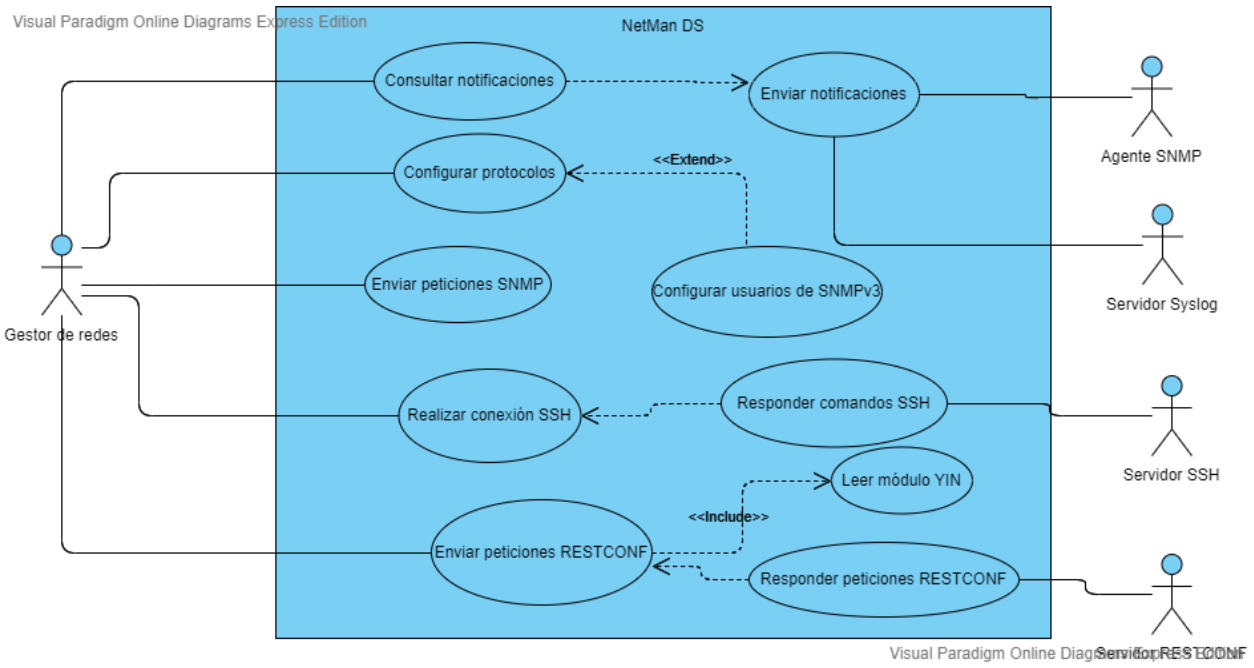

Figura 21- Primer diagrama de casos de uso

En este primer diagrama de casos de uso se exponen las funcionalidades básicas de la aplicación junto con los actores principales que interactúan con la misma.

Si nos centramos en los actores, podemos destacar la figura del gestor de redes, que es el usuario que interactúa con la interfaz gráfica de la aplicación. Entre sus acciones más habituales, tenemos la de consultar las notificaciones que se reciben tanto de agentes SNMP como de servidores Syslog, configurar los parámetros más significativos de los protocolos que se han implementado (como la comunidad de SNMPv1 y SNMPv2, el timeout de los sockets, los usuarios de SNMPv3, la MIB del SMI a usar…), enviar peticiones SNMP que previamente se han debido de configurar (se explicará con más detalle en el segundo diagrama), establecer conexiones SSH a un servidor para ofrecer servicios de configuración remota, e incluso enviar peticiones a dispositivos que soporten el protocolo RESTCONF como medio de gestión de la configuración. Para realizar esto último es preciso leer algún módulo YIN (YANG en formato XML) para seleccionar algún recurso del servidor que sea necesario gestionar.

El resto de actores indicados son simplemente los servidores que escuchan peticiones provenientes de la aplicación (de nombre NetMan DS) como es el caso del agente SNMP, el servidor SSH y el servidor de RESTCONF. Por otra parte, tanto el agente SNMP como el servidor Syslog pueden ser proactivos y enviar notificaciones, traps y registros a la aplicación, funcionando esta ahora como servidor. Toda esta información será almacenada para su posterior revisión por el gestor de redes, al que se le avisará con un pequeño sonido de alarma.

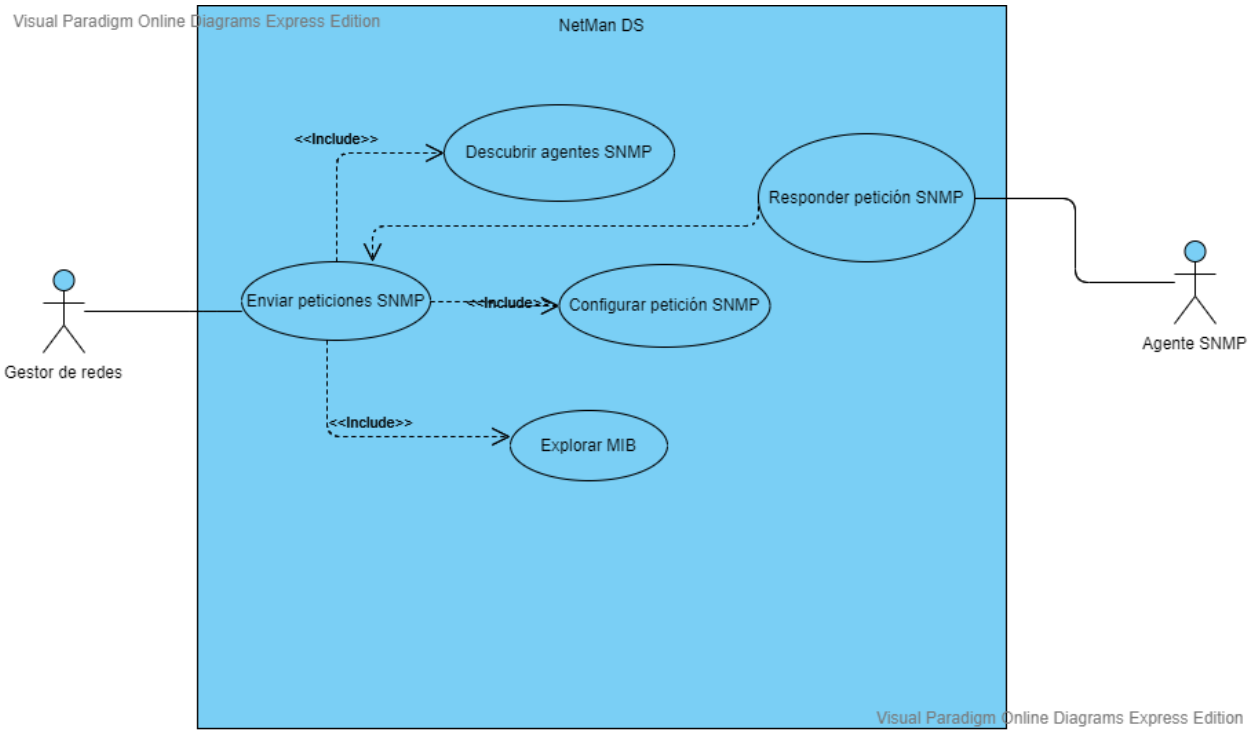

Figura 22- Segundo diagrama de casos de uso

En este segundo diagrama de casos de uso se muestran las operaciones que es posible realizar con el protocolo SNMP, a excepción de la recepción de notificaciones y traps que se trató en el diagrama anterior. En resumidas cuentas, todo se reduce a enviar peticiones SNMP a un servidor remoto para poder gestionar el nodo de una red. Para poder conseguirlo, es necesario realizar varias acciones antes. La aplicación se encarga de guiarnos en este proceso mediante el uso de la interfaz gráfica.

En primer lugar, debemos seleccionar qué MIB (Management Information Base) queremos usar, para saber a qué objeto de gestión queremos referirnos en la petición que vamos a mandar. Debemos tener, ya que estamos cumpliendo el rol de gestor de redes, las MIB que necesitemos en formato ASN.1 almacenada la tarjeta SD de nuestra consola. Una vez cargada la MIB, tendremos la posibilidad de seleccionar qué objetos de gestión queremos incluir en nuestra petición. De esto trata el caso de uso “Configurar petición SNMP”, donde indicamos a qué objetos queremos acceder, en qué modo (lectura, escritura…) y qué valores deben toman en

el caso de que le demos un nuevo valor.

Como último paso, debemos saber a qué agente SNMP queremos mandar la petición. Para ello, la aplicación cuenta con un sistema de descubrimiento de agentes basado en el escaneo de un rango de IP seleccionable por el gestor. Se nos deberá proporcionar una lista con los agentes encontrados para seleccionar al que queremos mandarle la petición.

Una vez completados los pasos, la petición SNMP se enviará al agente y este, si todo ha ido bien, responderá con un mensaje GetResponse, que la aplicación esperará para informar al usuario del estado de la petición que hemos mandado. Este mensaje, entre otra información, contendrá los campos modificados u obtenidos, incluso errores que hayan ocurrido, como la denegación de acceso por falta de permisos, intentar modificar un objeto que es de solo lectura, modificar un objeto inexistente…

4.2 Requisitos funcionales

Los requisitos funcionales tratan las funcionalidades básicas que debe soportar la aplicación, generalmente, según las especificaciones de un cliente. Podemos dividirlas en 3 tipos: requisitos de información, requisitos de reglas de negocio y requisitos de conducta.

4.2.1 Requisitos de información

Según MADEJA (ver [12]), los requisitos de información deben especificar qué información debe almacenar el sistema para poder ofrecer la funcionalidad descrita en los casos de uso del sistema o en otros requisitos.

INF-001	*Almacenamiento de notificaciones y registros*
Descripción	El sistema deberá almacenar tanto las notificaciones recibidas por el protocolo SNMP en cualquiera de sus versiones y los registros recibidos por el protocolo Syslog.
Datos específicos	• *El límite de elementos a almacenar será de 30 por defecto y por tipo de notificación, ajustable hasta 100.* • *El formato usado será legible a través de un editor de texto convencional.*

Tabla 1- INF-001

INF-002	*Almacenamiento de la configuración*
Descripción	El sistema deberá almacenar los parámetros de configuración más característicos de los protocolos SNMP, Syslog y REST Conf, además del timeout de los sockets.
Datos específicos	• *El formato será binario y sin seguir ningún estándar específico.* • *El formato deberá ser fácilmente ampliable para nuevas versiones de la aplicación.*

Tabla 2- INF-002

4.2.2 Requisitos de reglas de negocio

Estos requisitos deben contener las reglas de negocio que debe cumplir el sistema a desarrollar, evitando que se incumplan durante su funcionamiento.

NEG-001	*Exploración de MIBS*
Descripción	El usuario de la aplicación podrá explorar MIBs (Management Information Base), seleccionar objetos de gestión y ver información acerca de cada uno de ellos.
Comentarios	*Las MIBs deberán localizarse en un directorio específico.*

Tabla 3- NEG-001

NEG-002	*Recepción de traps y logs*
Descripción	El usuario de la aplicación podrá recibir traps del protocolo SNMP y registros recibidos por el protocolo IETF-Syslog.
Comentarios	*El usuario deberá ser notificado en el momento en el que se reciben y deberá poder observar su contenido en cualquier momento.*

Tabla 4- NEG-002

NEG-003	*Acceso por SSH a un servidor remoto*
Descripción	El usuario de la aplicación podrá acceder usando el protocolo SSH a una máquina remota para realizar labores de configuración y mantenimiento.
Comentarios	*Deberá soportarse la versión 2 del protocolo SSH.*

Tabla 5- NEG-003

NEG-004	*Exploración de módulos YIN*
Descripción	El usuario de la aplicación podrá explorar módulos en formato YIN (YANG XML), así como observar su descripción y usarlos para enviar peticiones RESTCONF.
Comentarios	*Los módulos YIN deberán localizarse en un directorio específico.*

Tabla 6- NEG-004

NEG-005	*Envío de peticiones SNMP*
Descripción	El usuario de la aplicación podrá enviar peticiones SNMP a cualquier agente que haya sido descubierto, usando cualquier formato de PDU soportado por el protocolo. Deberá poder visualizarse la respuesta obtenida.
Comentarios	*Deberá soportarse las 3 versiones del protocolo SNMP. En el caso de una petición a una tabla, debe poderse visualizar adecuadamente su contenido.*

Tabla 7- NEG-005

NEG-006	*Descubrimiento de agentes*
Descripción	El usuario de la aplicación podrá escanear un rango de IPs con el fin de descubrir agentes SNMP localizados en dicha subred.
Comentarios	*Deberán obtenerse los parámetros más característicos de cada agente.*

Tabla 8- NEG-006

NEG-007	*Envío de peticiones RESTCONF*
Descripción	El usuario de la aplicación podrá enviar peticiones RESTCONF a un servidor remoto, así como visualizar la respuesta obtenida.
Comentarios	*Se debe poder seleccionar el servidor a contactar y el método HTTP a usar. La autenticación HTTP Basic debe ser obligatoria, así como el uso de HTTPS cuando el usuario lo indique.*

Tabla 9- NEG-007

4.2.3 Requisitos de conducta

Estos requisitos deben especificar cualquier otro comportamiento deseado del sistema que no se haya especificado mediante los casos de uso del sistema, como generación de informes, funcionalidades transversales a varios casos de uso del sistema, etc.

CON-001	*Acceso a los parámetros de configuración*
Descripción	Los parámetros de configuración deben ser accesibles en cualquier momento sin que esto suponga la carga del mismo desde un fichero.
Interfaz de Servicio	*Sí*
Comentarios	*La carga de la configuración deberá realizarse únicamente cuando se necesite por primera vez.*

Tabla 10- CON-001

4.3 Requisitos no funcionales

A continuación exploraremos los requisitos que no tiene que ver con alguna funcionalidad concreta de la aplicación, sino con características transversales de la misma que se deben cumplir en todo momento.

4.3.1 Requisitos de fiabilidad

Estos requisitos deberán establecer, de la manera más objetiva y medible posible, los niveles que debe cumplir el sistema a desarrollar en aspectos como recuperabilidad y tolerancia a fallos.

FIA-001	*Notificación de errores*
Descripción	*La aplicación deberá informar al usuario de cualquier error que ocurra en tiempo de ejecución, especialmente de errores en el uso de la red.*
Comentarios	*Una vez notificado un error, deberá abortarse la operación actual.*

Tabla 11- FIA-001

FIA-002	*Generación de volcados de pila*
Descripción	*En el caso de ocurrencia de errores críticos, deberá generarse un volcado de pila para facilitar la depuración al desarrollador.*
Comentarios	*Este volcado deberá incluir el valor de los registros de la CPU, así como el punto de ejecución donde ocurrió el error.*

Tabla 12- FIA-002

4.3.2 Requisitos de usabilidad

Estos requisitos deberán establecer, de la manera más objetiva y medible posible, los niveles que debe cumplir el sistema a desarrollar en aspectos como facilidad de aprendizaje, comprensión, operatividad y atractividad.

USA-001	*Número de elementos en pantalla*
Descripción	*En una pantalla de la aplicación no deberá haber más de 6 elementos gráficos con los que el usuario pueda interacturar en un momento dado.*
Comentarios	*Esto incluye botones, listas desplegables, barras de desplazamiento, pero no cajas de texto estáticas o imágenes de fondo, por ejemplo.*

Tabla 13- USA-001

USA-002	*Lenguaje de la aplicación*
Descripción	*Para facilitar el aprendizaje de cualquier usuario, el lenguaje de la aplicación debe ser exclusivamente en inglés.*
Comentarios	*Todos los elementos en inglés deberán estar contenidos exclusivamente en los recursos de la aplicación y no en el código, a excepción de los mensajes de error.*

Tabla 14- USA-002

USA-003	*Animaciones de la aplicación*
Descripción	*Los botones de la interfaz gráfica deben cambiar su aspecto cuando son pulsados. Además, debe haber transiciones de una pantalla a otra.*
Comentarios	*Las animaciones usadas no durarán más de 3 segundos y no supondrán más del 5% del uso de la CPU.*

Tabla 15- USA-003

4.3.3 Requisitos de mantenibilidad

Estos requisitos deberán establecer, de la manera más objetiva y medible posible, los niveles que debe cumplir el sistema a desarrollar en aspectos como estabilidad, facilidad de análisis, facilidad de cambio, facilidad de pruebas.

MAN-001	*Generación de documentación*
Descripción	*La aplicación deberá incluir una documentación del código en formato Doxygen.*
Comentarios	*Debe incluirse al menos un diagrama de clases en la misma.*

Tabla 16- MAN-001

MAN-002	*Modularidad de los protocolos de red*
Descripción	*Cada protocolo de red implementado debe estar separado en módulos con acoplamiento débil.*
Comentarios	*Un módulo que represente un protocolo de red solo debería depender de la librería de sockets y de librerías de criptografía.*

Tabla 17- MAN-002

MAN-003	*Extensibilidad de las funcionalidades*
Descripción	*Debe poder añadirse nuevos elementos gráficos, protocolos de red, algoritmos de seguridad e interfaces gráficas sin modificar más del 1% del código existente.*
Comentarios	*Los elementos y funcionalidades existentes deben poder ampliarse bajo las mismas restricciones.*

Tabla 18- MAN-003

4.3.4 Requisitos de eficiencia

Estos requisitos deberán establecer, de la manera más objetiva y medible posible, los niveles que debe cumplir el sistema a desarrollar en aspectos como tiempo de respuesta.

EFI-001	*Tiempo de respuesta estimado*
Descripción	*La aplicación deberá ofrecer un tiempo de respuesta no superior a 4 segundos para tareas que no usen la red y no superior al timeout especificado por el usuario en tareas que usen la red.*
Comentarios	*Se debe abortar toda operación de red que exceda estos límites.*

Tabla 19- EFI-001

4.3.5 Requisitos de portabilidad

Estos requisitos deberán establecer, de la manera más objetiva y medible posible, los niveles que debe cumplir el sistema a desarrollar en aspectos relacionados con la escalabilidad: capacidad de instalación, capacidad de sustitución, adaptabilidad, coexistencia, compatibilidad con hardware o software, etc.

POR-001	*Soporte para la plataforma Nintendo 3DS*
Descripción	*La aplicación deberá ser compatible con el sistema Nintendo 3DS.*
Comentarios	*Debe ser portable a otras plataformas sin requerir la refactorización de más del 20% del código desarrollado.*

Tabla 20- POR-001

4.3.6 Requisitos de seguridad

Estos requisitos deberán establecer, de la manera más objetiva y medible posible, los niveles que debe cumplir el sistema a desarrollar en aspectos como accesos al sistema, identificación y autenticación, protección de datos y privacidad.

SEG-001	*Encriptación y autenticación de la información*
Descripción	*La información enviada o recibida de la red deberá estar encriptada y debe ser autenticable en al menos la mitad de los protocolos implementados.*
Comentarios	*Es recomendable el uso de certificados X.509 para este fin.*

Tabla 21- SEG-001

4.4 Diagramas de clases

Como siguiente punto, echaremos un vistazo a los diagramas de clases correspondientes a la aplicación para poder observar el funcionamiento estático o estructural de la misma. Las diferentes clases se han dividido en paquetes o módulos agrupándolas por su rol en la ejecución del software. Podrá determinar cómo esta designación de paquetes es imprescindible para cumplir con requisitos como MAN-002, donde cada protocolo implementado está agrupado en un paquete y débilmente acoplado (o incluso no) con el resto. Además, se han indicado las relaciones más significativas (aunque no todas para no ensuciar demasiado el diagrama) entre las clases como las herencias, asociaciones, composiciones y dependencia, entre otras.

4.4.1 Diagrama de clases global

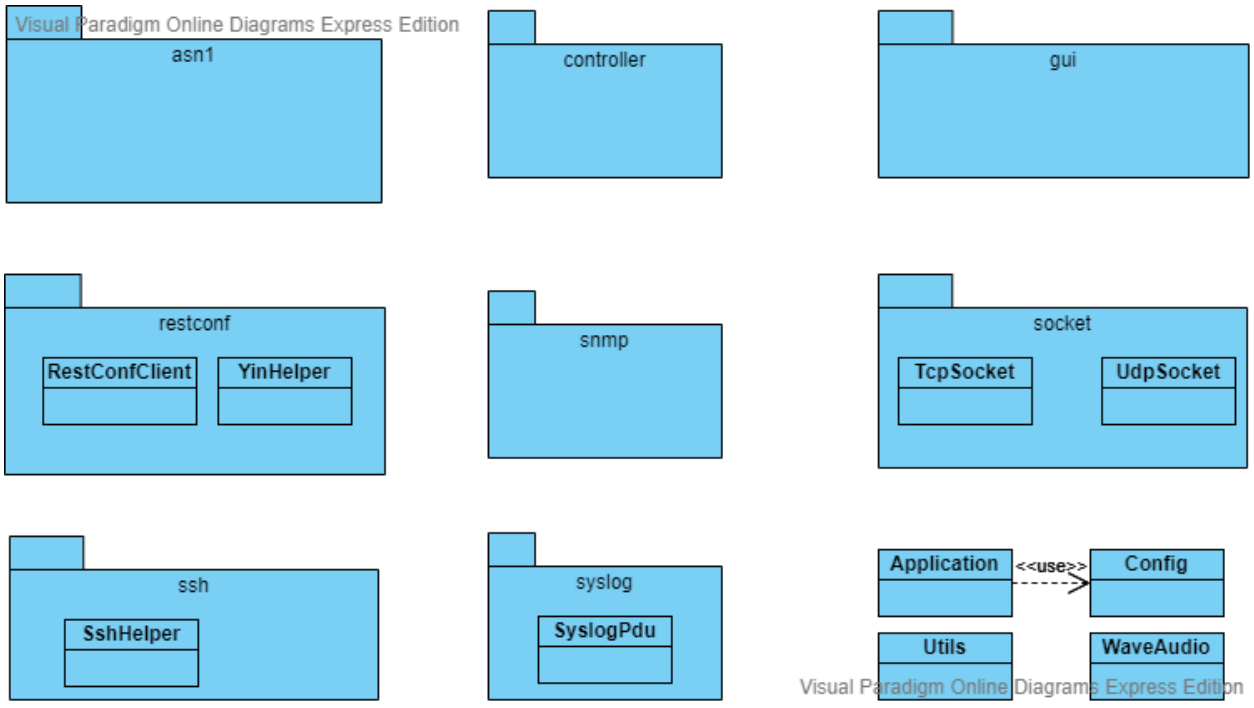

Figura 23- Diagrama de clases global

En este primer diagrama de clases de hace una agrupación de las clases de la aplicación en paquetes dependiendo a qué funcionalidad pertenecen. Sin embargo, se puede observar 4 clases que no pertenecen a ningún paquete. La primera de ellas es la clase Application que es un Singleton que se encarga de la gestión global de la aplicación: la carga de recursos, la gestión de la carga y descarga de pantallas, inicialización de librerías, mostrar mensajes de error… Es decir, un conjunto de funciones transversales a toda la aplicación que se ha encapsulado en una única clase.

Similar es la clase Utils, una clase estática que contiene métodos que no son críticos para el funcionamiento global del software, pero sí para algunas funcionalidades que poco o nada tienen que ver. Por ejemplo, aquí se trata la carga de ficheros JSON, gestión de campos en formularios, lectura de directorios, envío de peticiones

SNMP y RESTCONF que pueden ser usados por diferentes controladores, entre otros.+

La clase Config es otro Singleton que se encarga de la gestión de la configuración de la aplicación. Se encarga de cargarla de un archivo y guardarla cuando es necesario, así como la edición de sus parámetros.

Por último, la clase WaveAudio es un simple cargador de ficheros en formato WAV. Es usado para cargar un pequeño efecto de sonido que se reproduce al recibir una notificación o trap.

Aunque iremos tratando los paquetes más grandes en otros subapartados (los que no tienen clases en este diagrama), podemos ir comentando los paquetes más simples. Uno de ellos es el paquete "socket" que comprende el uso de sockets TCP y UDP, usados por todos los protocolos implementados. El paquete "syslog" solo contiene una clase SyslogPdu que se encarga de la recepción y desencapsulación (unmashalling) de los paquetes Syslog que llegan por la red. También se encarga de su serialización en formato JSON para ser almacenados en un fichero.

El paquete "ssh" también contiene una única clase llamada SshHelper que es un ayudante que gestiona la conexión SSH con un servidor remoto, así como proporcionar métodos para enviar comandos de terminal y recibir sus respectivas respuestas.

Por último, el paquete "restconf" contiene las clases referentes a este protocolo. La clase YinHelper es un ayudante que carga módulos en formato YIN y proporciona métodos para navegarlo y para generar la URL que se usará en la petición REST. Esta URL generada será usada por la clase RestConfClient, que abrirá una sesión HTTP o HTTPS con el servidor (que se especifica en la configuración de la aplicación), enviará la petición y recibirá la respuesta.

4.4.2 Paquete ASN.1

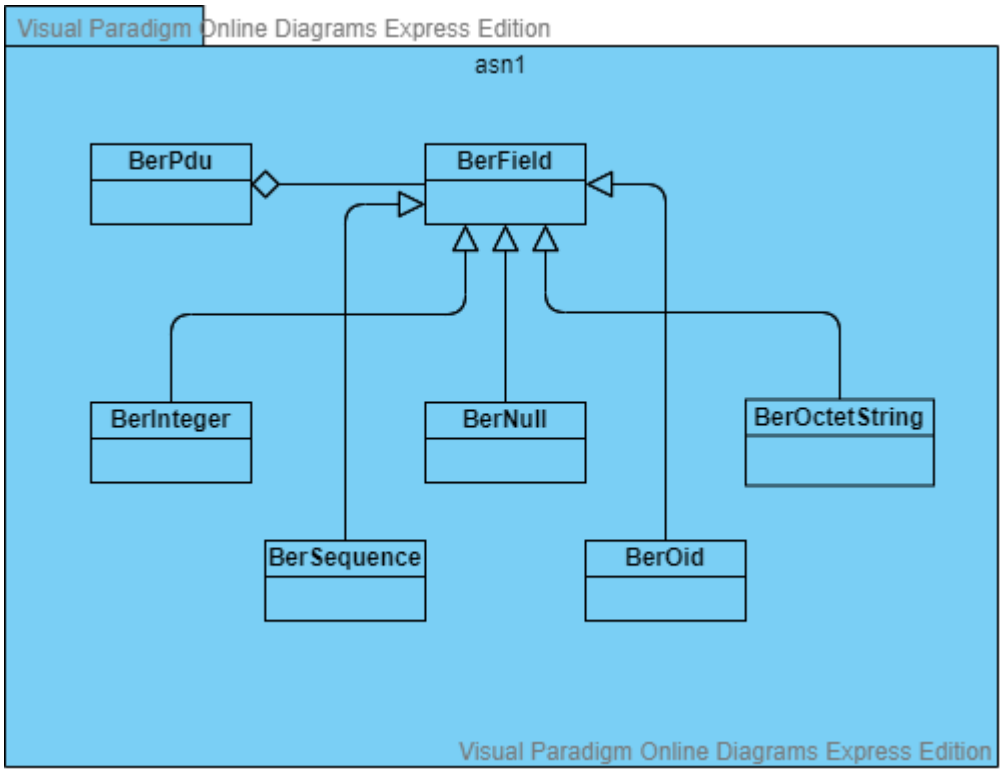

Figura 24- Diagrama de clases del paquete ASN.1

Ahora que hemos observado una visión global de la estructura de la aplicación, veamos los paquetes más densos por separado. El primero con el que vamos a tratar es el paquete "asn1", que comprende clases que usan el patrón "modelo" para almacenar datos que después serán serializados (o deserializados) según las BER (Basic Encoding Rules). Se parte de una clase abstracta llamada BerField de la que heredan cada tipo de dato ASN.1 soportado. Podemos almacenar los tipos ASN.1 INTEGER, NULL, OCTET STRING, OBJECT

IDENTIFIER y SEQUENCE, que son los necesarios para poder implementar el protocolo SNMP.

Además, tenemos una clase BerPdu que se encarga de almacenar varios BerField, de serializarlos en un paquete y de transmitirlo (usando un socket UDP) a un equipo remoto.

4.4.3 Paquete de vistas

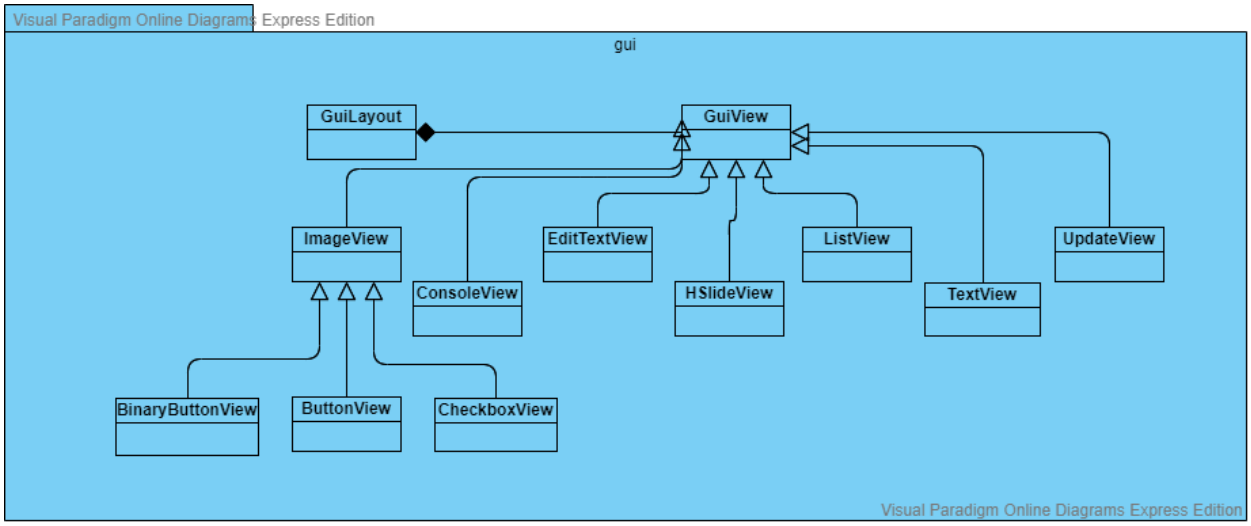

Figura 25- Diagrama de clases del paquete vistas

En este paquete se incluyen las vistas de la interfaz de usuario (GUI) de la aplicación. Para realizar esta GUI, se ha usado el patrón de diseño modelo-vista-controlador (MVC). En este paquete concreto, todas las clases (a excepción de GuiLayout) son clases que usan el patrón vista. Se parte de una interfaz GuiView de la que heredan el resto de elementos que pueden aparecer en pantalla.

Si nos proponemos una clasificación de estos elementos gráficos, tenemos los siguientes: cuadros de texto, imágenes, botones, checkboxes (botones que almacenan un estado binario), botones binarios (que almacenan un estado binario como el anterior pero usando 2 botones en vez de uno, en los que si uno está seleccionado estamos en un estado y si es el otro pues en el otro estado).

También tenemos cajas donde podemos editar el texto que contiene (aparecerá un teclado en pantalla para editar su contenido), elementos para visualizar listas de datos (ListView), diversas pantallas en las que podemos navegar usando un scroll horizontal (elemento HSlideView).

Por último, destacamos ConsoleView que nos muestra una terminal de comandos de las dimensiones que queramos en la que podemos introducir texto. Usa internamente la librería tmt que se comentó en el punto 3 para gestionar el terminal y el cursor del mismo. La vista UpdateView no muestra ningún elemento en pantalla, solo gestiona un callback que se llamará periódicamente en el dibujado de cada fotograma, útil para realizar animaciones, monitorización de sockets o de hilos…

La clase GuiLayout se encarga de almacenar varios elementos de tipo vista con el fin de mostrarlos en pantalla. Como diseñar interfaces gráficas es un proceso engorroso y poco eficiente si se realiza programáticamente, se ha diseñado un pequeño esquema XML (sin especificación XSD ni nada parecido) donde vamos introduciendo los elementos gráficos que queremos en una pantalla, así como especificar el controlador de la misma. La clase GuiLayout, por tanto, crea los elementos gráficos a partir de un fichero XML. A continuación se muestra un ejemplo de layout que pertenece a una pantalla de la aplicación:

```
<root controller="AddUserController">
    <ImageView name="bottomScreen" x="160" y="120"/>
    <TextView text="Add SNMPv3 user" x="20" y="10" size="1.0"/>

    <TextView text="Username" x="20" y="50" size="0.75"/>
    <EditTextView x="160" y="50" width="140" height="20" length="12" onEdit="editUsername"/>

    <TextView text="Privacy" x="20" y="75" size="0.75"/>
    <EditTextView x="160" y="75" width="140" height="20" length="12" onEdit="editPriv"/>

    <TextView text="Password" x="20" y="100" size="0.75"/>
    <EditTextView x="160" y="100" width="140" height="20" length="12" password="true" onEdit="editPrivPass"/>

    <TextView text="Authentication" x="20" y="125" size="0.75"/>
    <EditTextView x="160" y="125" width="140" height="20" length="12" onEdit="editAuth"/>

    <TextView text="Password" x="20" y="150" size="0.75"/>
    <EditTextView x="160" y="150" width="140" height="20" length="12" password="true" onEdit="editAuthPass"/>

    <ButtonView name="menuButton" x="150" y="216" onClick="addUser" sx="0.5" sy="0.4"/>
    <TextView text="OK" x="135" y="205" size="0.75"/>

    <ButtonView name="backArrow" x="24" y="216" onClick="gotoOptions" sx="-0.75" sy="0.75"/>
</root>
```

Figura 26- Ejemplo de layout XML

Este ejemplo representa una pantalla de la aplicación que consiste en un formulario donde se añade un usuario SNMPv3 a la base de datos de usuarios (según el modelo USM). Primero indicamos en el nodo raíz la clase que se va a usar de controlador de esta pantalla. Después, como nodos hijos tenemos cada uno de los elementos de la pantalla. Podemos observar imágenes, textos, cajas de texto editables y botones con parámetros distintos para cada uno de ellos.

Las propiedades que comienzan con “on” (onClick, onEdit) especifican el nombre del método del controlador que se va a llamar cuando ocurra un cierto evento. Así, cuando el usuario pulse un botón, se llamará al callback especificado en la propiedad “onClick”, cuando el usuario desee introducir un texto se llamará al callback especificado en “onEdit”, y así con el resto.

Como vemos, podemos configurar muchos parámetros para cada vista, como la posición y escala, indicar la longitud máxima de una caja de texto editable, si se almacenará una contraseña, un número o una cadena de texto.

Sin embargo, para que esta manera de proceder sea más óptima es necesario un editor gráfico (que no se ha realizado en este proyecto) con el que realizar estas pantallas de manera interactiva. De otra forma, habría que introducir, por ejemplo, la posición de los elementos a mano, necesitando un buen número de comprobaciones para ver si el elemento gráfico está bien posicionado. Tampoco se han implementado aspectos más complejos como puntos de referencia (anchor points), márgenes… Sin embargo, esta es una buena manera de diseñar interfaces gráficas de usuario (GUI) sin complicar excesivamente el código, realizándolo todo de forma genérica.

4.4.4 Paquete SNMP

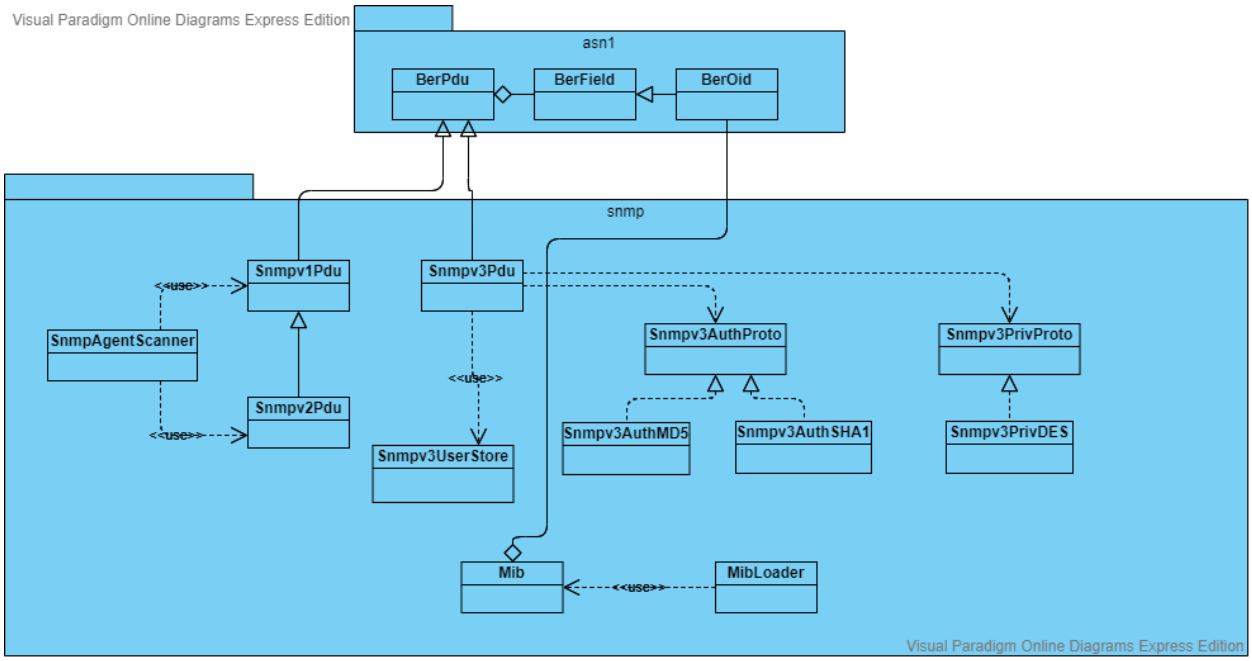

Figura 27- Diagrama de clases del paquete SNMP

Como último diagrama de clases, tenemos los que involucran el paquete "snmp", que implementa el protocolo SNMP casi en su totalidad (faltaría algún modelo de seguridad más como el TSM o más algoritmos de encriptación y resumen). Se soportan las 3 versiones de SNMP de manera prácticamente completa, además de una serie de utilidades como un cargador de MIBs y un escáner de agentes SNMP.

Como base para implementar el protocolo, se ha tomado la clase BerPdu del paquete "asn1" debido a que las PDU del protocolo SNMP son datos ASN.1 codificados según las BER. El caso de SNMPv3 hereda de BerPdu y no de Snmpv2Pdu debido a que tiene unas importantes diferencias que impiden la reutilización del código ya realizado para las anteriores versiones.

En cuanto a SNMPv3, la clase Snmpv3Pdu se apoya en las interfaces Snmpv3AuthProto y Snmpv3PrivProto para cuestiones de seguridad como la autenticación y el cifrado de las PDU según el modelo de seguridad USM. Se observa que estas interfaces usan el patrón estrategia (Strategy) para implementar los diferentes algoritmos de seguridad disponibles. En los algoritmos de resumen tenemos MD5 y SHA1, y de encriptación tenemos DES. Se puede ver que, teniendo el sistema diseñado de esta manera, resulta trivial añadir nuevos algoritmos de seguridad sin modificar apenas código existente.

Por otro lado, la clase Snmpv3Pdu se apoya en la clase Snmpv3UserStore, que es un Singleton que proporciona los usuarios registrados, así como sus credenciales y mecanismos de autenticación y cifrado usados.

Además, observamos la clase SnmpAgentScanner que nos proporciona un mecanismo de descubrimiento de agentes SNMP basado en el escaneo de un rango de IPs o en una subred. Este escáner envía una petición SNMP (en su versión 1 o 2, de manera configurable) a cada IP del rango especificado. Si se obtiene una respuesta, se registra en una lista la IP del agente que ha respondido, así como sus parámetros más característicos (el nombre de la máquina, los servicios disponibles, su localización, información de contacto…) que posteriormente se podrá consultar para enviar peticiones SNMP a estos agentes. La clase proporciona parámetros para ajustar el escaneo, como el número máximo de peticiones a enviar simultáneamente y el tiempo de espera para obtener las respuestas de los agentes.

Para finalizar, tenemos la clase MibLoader que se encarga de cargar una MIB y almacenarla en un objeto de tipo Mib, que no es más que una clase que usa el patrón de diseño "modelo" para almacenar un árbol con los OID que conforman esa MIB. En cada entrada del árbol se tiene el OID mismamente, además de información adicional sobre ese objeto de gestión, como la sintaxis que debe usarse, el modo de acceso (lectura,

escritura…), una descripción… es decir, todo lo que contiene la MIB (en su mayoría) es accesible a través de esta clase.

4.5 Diagramas de actividad

Para finalizar el apartado de diseño de la aplicación, se tratarán un par de diagramas de actividad que tienen como objetivo especificar el comportamiento dinámico (en tiempo de ejecución) de la aplicación. En cada diagrama, se han organizado en "swimlanes" los diferentes actores y/o objetos que toman relevancia en cada acción de la actividad especificada. También se indican los eventos más relevantes de la actividad, mostrando quién los envía y quién los recibe, así como posibles excepciones que pueden ocurrir en las acciones realizadas.

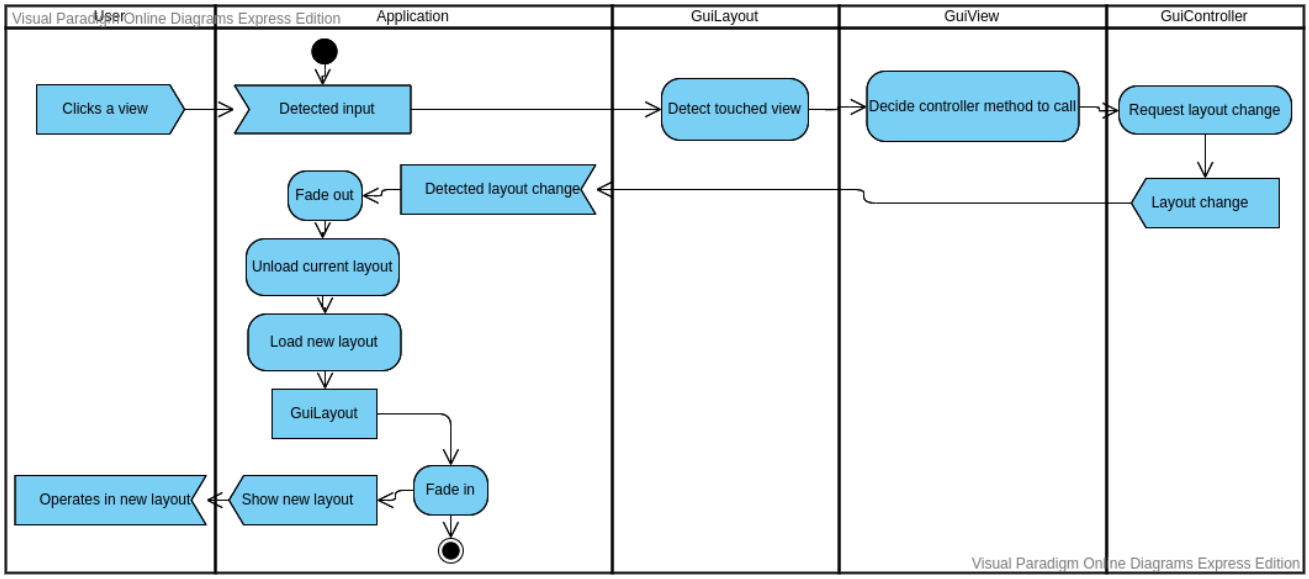

Figura 28- Diagrama de actividad 1

En este primer diagrama de actividad se muestra el proceso por el que, cuando un usuario pulsa en cierto elemento gráfico que requiere un cambio de pantalla, la aplicación realiza efectivamente este cambio, descargando la pantalla actual y cargando la nueva. Esta actividad es un buen ejemplo, además, para poder observar el comportamiento del patrón modelo-vista-controlador (MVC) explicado anteriormente en los diagramas de clases.

Para comenzar, el usuario de la aplicación (el gestor de redes) ocasiona un evento que es hacer click en una vista. Este click es detectado, en primer lugar, por la clase Singleton llamada Application, que detecta la pulsación del usuario y la envía a un objeto GuiLayout que representa la pantalla que está siendo mostrada actualmente. Este objeto, a su vez, determina qué vista se ha tocado con la información de entrada recibida, avisando a la misma. La vista que ha sido pulsada, a su vez, decide qué método del controlador que tiene asignado hay que llamar para gestionar la pulsación del usuario. Una vez decidido el método, se llama, provocando un cambio de pantalla en el cuerpo del mismo. Este fenómeno lo tratamos como una señal que parte del controlador de la vista actual al singleton de tipo Application.

La clase Application, al detectar la petición de cambio de pantalla, realiza primero unm efecto de "fade out" que oscurece la pantalla gradualmente. Cuando la pantalla queda en negro, se elimina la pantalla actual y se carga la nueva. Una vez finalizado este proceso, se realiza un efecto de "fade in" que aclara la pantalla poco a poco. Cuando este efecto finaliza, el usuario podrá visualizar e interactuar con la nueva pantalla que ha sido cargada.

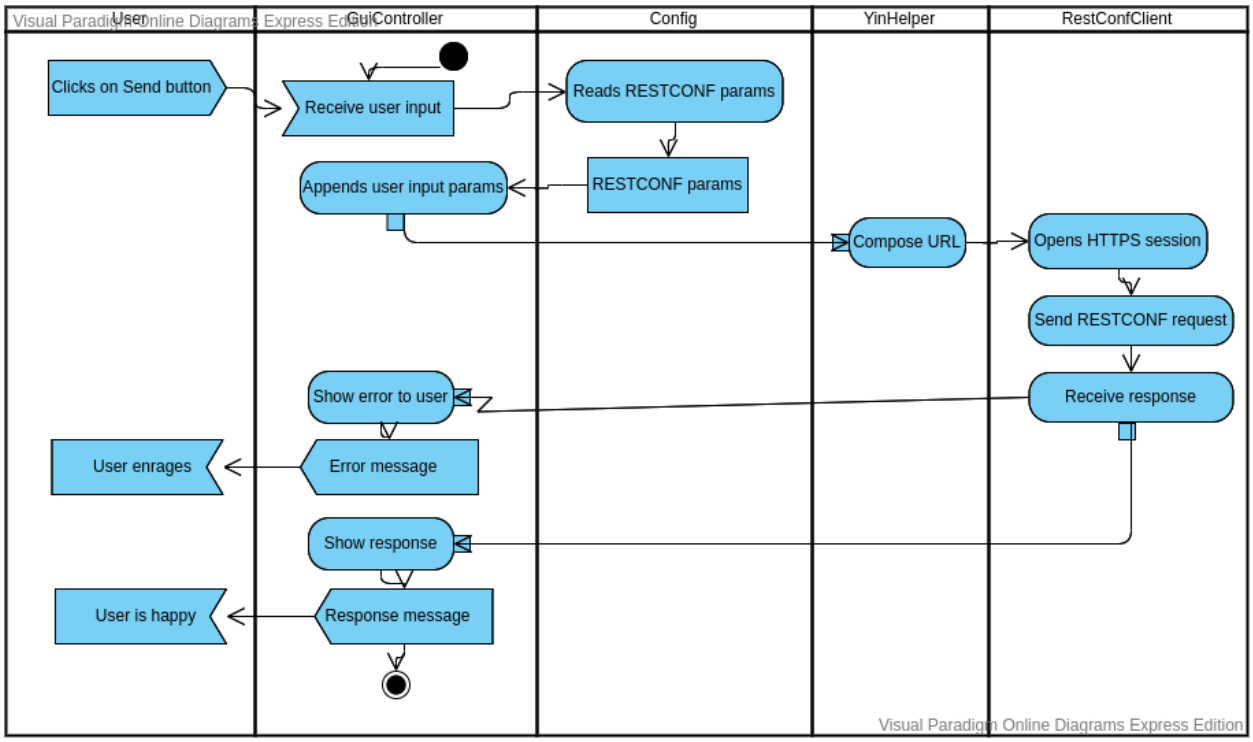

Figura 29- Diagrama de actividad 2

En este segundo diagrama de actividad se trata el proceso por el que un usuario envía una petición RESTCONF a un servidor remoto, indicando los objetos que juegan un papel más importante en esta funcionalidad.

En primer lugar, el usuario de la aplicación manda una señal de que ha pulsado el botón de "Enviar" de alguna pantalla de la interfaz gráfica. Como ya explicamos en el diagrama anterior como se gestiona la entrada del usuario, nos saltamos en este toda esta parte hasta llegar al controlador, que recibe los datos de entrada del usuario.

Después, para realizar la petición REST, es necesario obtener de la configuración parámetros como la URL del servidor REST a usar, el puerto, el usuario con el que queremos autenticarnos y la contraseña (si el servidor usa autenticación HTTP Basic). Una vez el controlador tiene esos parámetros, añade los que tiene almacenados que ha introducido el usuario en la pantalla que gobierna, como el nodo del módulo YIN que se va a consultar o modificar, incluso la operación HTTP que se quiere realizar (se soportan GET, POST y DELETE en este proyecto).

Con todos esos parámetros, un objeto de tipo YinHelper compone la URL completa y abre una sesión HTTP o HTTPS usando la clase RestConfClient, que recibe la URL formada y el método HTTP, enviando la petición al servidor y esperando a la recepción de una respuesta. Si existe algún error al recibirla (como un timeout o un error que notifica el servidor), se lanzará una señal de excepción, mostrando al usuario el error que ha sido ocasionado. En caso contrario, se mostrará al usuario la respuesta de la petición REST.

5 IMPLEMENTACIÓN DE LA SOLUCIÓN

Ahora que se ha indagado de forma profunda en el diseño de la aplicación, veamos ahora algunos detalles sobre la implementación de la misma. Se ofrecerá un resumen de todas las funcionalidades que la aplicación soporta, mostrando capturas de su funcionamiento real. Hay que decir que las capturas han sido obtenidas a partir del hardware real, es decir, no se ha usado ningún emulador en el proceso. También es preciso comentar que el emulador Citra3DS no puede ejecutar de forma completa la aplicación, debido a que algunos servicios del sistema operativo que esta usa no son soportados por el emulador. Sin embargo, como se observará en este apartado, todas las funcionalidades implementadas tienen un funcionamiento correcto en el dispositivo físico, ofreciendo una serie de valiosas herramientas para cualquier gestor de redes de telecomunicación.

5.1 Interfaz gráfica

Al iniciar la aplicación, se nos mostrará una pantalla de bienvenida en la que, al tocar la pantalla inferior (recordemos que el dispositivo Nintendo 3DS posee 2 pantallas, como se comentó en el punto 3), nos redirige al menú principal de la aplicación, que se puede observar en la siguiente figura:

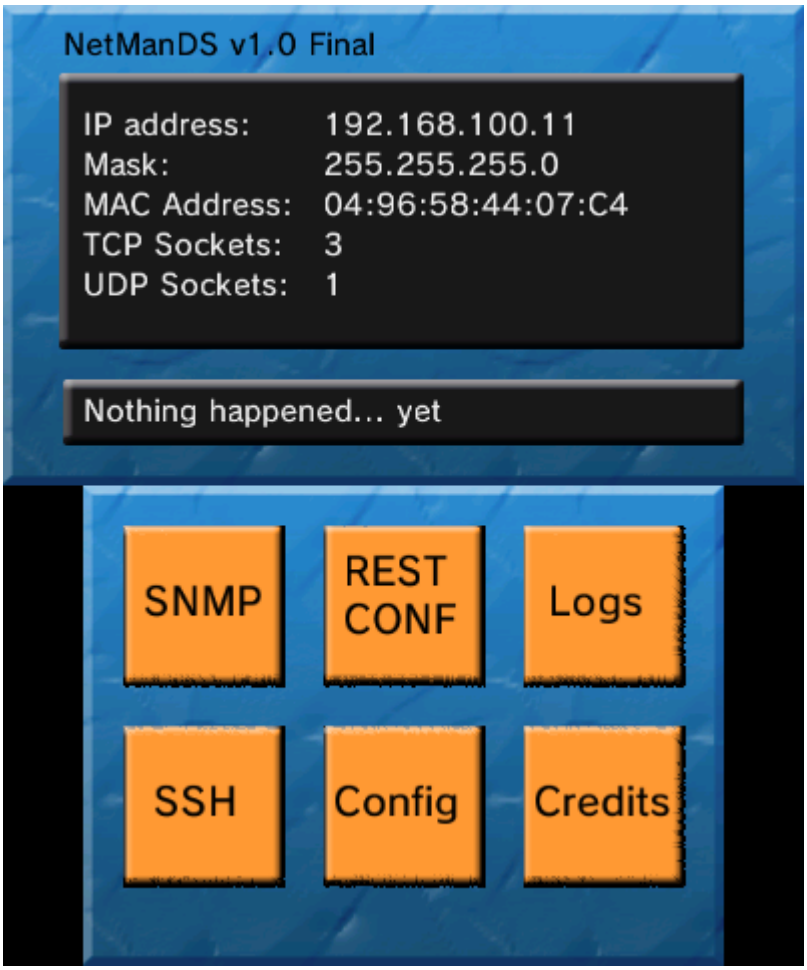

Figura 30- Menú principal de la aplicación

Se puede ver que en la pantalla superior se muestra información general de red del dispositivo Nintendo 3DS. Se nos muestra la dirección IP, la máscara, la dirección MAC de la consola, e incluso el número de sockets

TCP y UDP que estamos usando en este momento. Cabe destacar que el mensaje que aparece debajo se usa para mostrar notificaciones, es decir, si tenemos activada en las opciones la recepción de traps SNMP o de registros Syslog, cuando nos llegue alguno de estos mensajes aparecerá en ese cuadro de texto información acerca de cuándo llegó el mensaje y el tipo de mensaje recibido. Más adelante veremos en mayor profundidad cómo funciona la recepción de notificaciones y cómo mostrarlas en una lista para su cómoda inspección.

En la pantalla inferior, se nos proporciona una serie de botones que, al tocarlos, nos irán redirigiendo a cada una de las pantallas de la aplicación, donde podremos realizar diversas funciones de gestión de redes. Además, hay un botón para poder editar la configuración de la aplicación así como de los protocolos usados y, por último, una pantalla de créditos donde se identifica al creador, las librerías y SDK usados, así como agradecimientos a otras personas.

5.2 Explorador de MIBS

Si en el menú principal tocamos, por ejemplo, el botón SNMP, se nos ofrecerán 3 opciones: descubrimiento de agentes, explorador de MIBs y envío de peticiones SNMP. Por ahora, nos centraremos en el explorador de MIBs, una parte esencial cuando gestionamos nodos de una red que soportan protocolos SNMP, debido a que, como gestor de redes, se necesita conocer qué objetos de gestión están disponibles en el agente. Este explorador proporciona una limpia presentación de los objetos de gestión de una MIB concreta que seleccionaremos previamente, donde podremos ver información sobre cada objeto, añadirlo a una lista para posteriormente enviarlo y, por supuesto, si el objeto de gestión es una tabla, visualizarla y tener la posibilidad de editar sus campos.

Figura 31- Explorador de MIBS I

En esta primera captura podemos visualizar la pantalla de selección de la MIB que queremos explorar. Las MIBs deben almacenarse en el directorio “mibs/” que debe situarse donde está el ejecutable de la aplicación.

Entonces, nos aparecerán las MIBs colocadas y podremos seleccionarlas para proceder a su exploración.

Figura 32- Explorador de MIBS II

En esta segunda captura hemos seleccionado, por ejemplo, la MIB de SNMPv2, donde podemos observar objetos de gestión relacionados con los parámetros generales de un nodo de red. Con las flechas a la derecha podemos navegar por los elementos de un nodo concreto de la MIB. La flechita en la esquina superior izquierda nos permite retroceder un nodo. Para salir al menú principal, podemos tocar el botón en la esquina inferior izquierda.

Para cada objeto de gestión, tenemos varios iconos a la derecha que nos permite realizar distintas acciones sobre el mismo. El icono con un símbolo de suma (+) se usa para añadir este objeto de gestión a una lista de objetos que almacena la aplicación que, posteriormente, podremos usar para mandar una petición SNMP a un agente remoto. El icono con un "tick", al tocarlo, nos mostrará información almacenada en la MIB acerca del objeto de gestión seleccionado. Es simplemente los contenidos de la macro OBJECT-TYPE (o alguna otra que se use). Es preciso decir que no todas las macros se soportan en esta aplicación, por simplicidad, mostrando un texto vacío cuando no es posible leer la macro de un objeto de gestión.

Por último, si el objeto de gestión es una tabla, el explorador lo detecta y activa un icono en forma de "T" que, al pulsar sobre él, nos permitirá visualizar la tabla. Más adelante visualizaremos una captura mostrando esta funcionalidad más en detalle.

5.3 Gestor SNMP

El módulo de gestión SNMP, además de proporcionar un cómodo explorador de MIBs, nos permite identificar los agentes SNMP que hay en una subred dada, y enviar peticiones SNMP a esos agentes con los objetos de gestión que previamente hemos seleccionado de una o varias MIB. Veamos cada funcionalidad en detalle.

5.3.1 Descubrimiento de agentes

La aplicación incluye, como se comentó anteriormente, un pequeño escáner de agentes SNMP que nos permite descubrir qué agentes están disponibles en un momento dado. En primer lugar, debemos rellenar este formulario para configurar los parámetros del escáner:

Figura 33- Formulario de descubrimiento de agentes SNMP

En este formulario podemos observar que debemos especificar el rango de IPs que se ha de escanear, el puerto que se va a consultar para cada IP de ese rango, la versión del protocolo SNMP que se va a usar (que puede ser 1 o 2), así como el número máximo de peticiones SNMP que se pueden mandar simultáneamente y el tiempo de espera para la respuesta del agente.

Una vez configurados los parámetros del escáner, si pulsamos en el botón “Scan” se procederá a descubrir los agentes SNMP en segundo plano. Se nos mostrará en tiempo real en la esquina inferior derecha el progreso del escaneo en tanto por ciento (%). Al llegar al 100%, se nos redirigirá a una nueva pantalla donde podremos observar los agentes SNMP que han sido descubiertos (debido a que han respondido un GetRequest). Podemos observar esta lista en la siguiente captura:

Figura 34- Lista de agentes SNMP descubiertos

En este ejemplo, se ha descubierto un agente SNMP en la IP 192.168.100.26. Si hubieran más, se mostrarían a continuación en esta lista. Por supuesto, si pulsamos en cualquier agente descubierto se nos mostrará un mensaje con los parámetros más básicos del mismo (los pertenecientes al grupo "system" de la MIB SNMPv2). Estos parámetros pueden ser los servicios soportados, la localización física del agente, la información de contacto, su OID característico, así como el nombre del nodo gestionado. En la siguiente captura podemos observar un extracto de estos parámetros para el agente mostrado en la lista anterior:

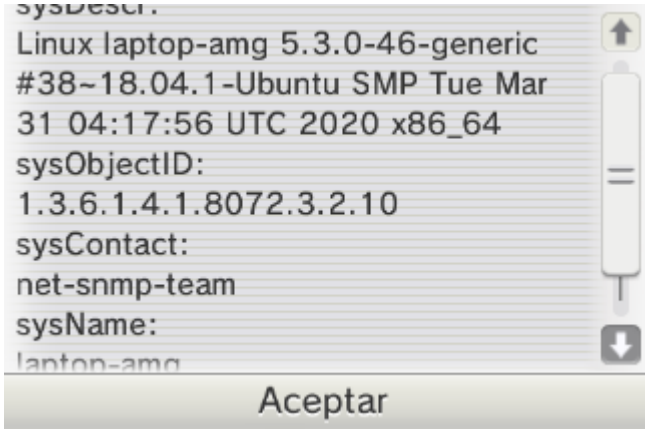

Figura 35- Información sobre un agente SNMP

5.3.2 Envío de peticiones

Por último, veamos el menú de envío de peticiones SNMP. Una vez hemos seleccionado los objetos de gestión

que queremos acceder usando las MIBs pertinentes, y hemos escaneado los agentes SNMP para tener un destino para la petición, podemos observar en la siguiente captura los objetos de gestión seleccionados:

Figura 36- Campos a enviar en una petición SNMP

En este caso concreto, hemos seleccionado los objetos correspondientes al ya mencionado grupo "system" de la MIB de SNMPv2. Como podemos observar, podemos visualizar cada objeto de gestión seleccionado, eliminarlo si lo consideramos necesario pinchando en la cruz (x) a la derecha de cada elemento, e incluso darle un valor en el caso de que vayamos a hacer un SetRequest.

Una vez hemos configurados los objetos de gestión, podemos enviarlos usando cualquier mensaje SNMP. Se incluye la opción de enviar un GetRequest, un SetRequest, un GetNextRequest o un GetBulkRequest. Al pinchar en cada una de las opciones, nos redirigirá a una pantalla para seleccionar el agente al que queremos enviarle la petición (que debe haber sido escaneado antes, si no, la aplicación nos redirigirá al escáner de agentes SNMP).

Una vez seleccionado el agente, antes de mandar la petición se nos muestra una última pantalla donde debemos especificar la versión del protocolo SNMP a usar y los parámetros de autenticación necesarios. Si seleccionamos la opción "v1/2", la petición se mandará usando SNMPv1 en el caso de las GetRequest, SetRequest y GetNextRequest, y usando SNMPv2c en el caso de GetBulkRequest. En este caso deberemos especificar, por tanto, el nombre de la comunidad con la que nos autenticaremos.

Si, por el contrario, seleccionamos la opción USM, la petición se transmitirá usando SNMPv3. Deberemos indicar obligatoriamente el usuario con el que nos autenticaremos, que deberá estar previamente registrado en la base de datos de usuarios (en el menú de opciones).

Ahora que están todos los parámetros de la petición SNMP configurados, podemos pulsar en el botón "Send" para enviar la petición. Nos aparecerá una ruedecita en la esquina superior derecha donde nos informará de que la petición se está enviando o que se está esperando una respuesta del agente. Si el agente no responde a la

petición o la respuesta no tiene el formato correcto, se informará al usuario de la aplicación con un mensaje de error, abortando la operación.

Figura 37- Parámetros de la petición SNMP

Sin embargo, si todo ha ido bien, la aplicación nos redirigirá a una nueva pantalla donde podremos observar la respuesta del agente. Se nos mostrará en una lista los campos con lo que el agente ha respondido, así como el valor de dichos campos. Si el valor de un campo es extenso y no cabe en la lista, podemos tocar un campo para ver su valor completo. A continuación se observa un ejemplo de respuesta a la petición que configuramos anteriormente:

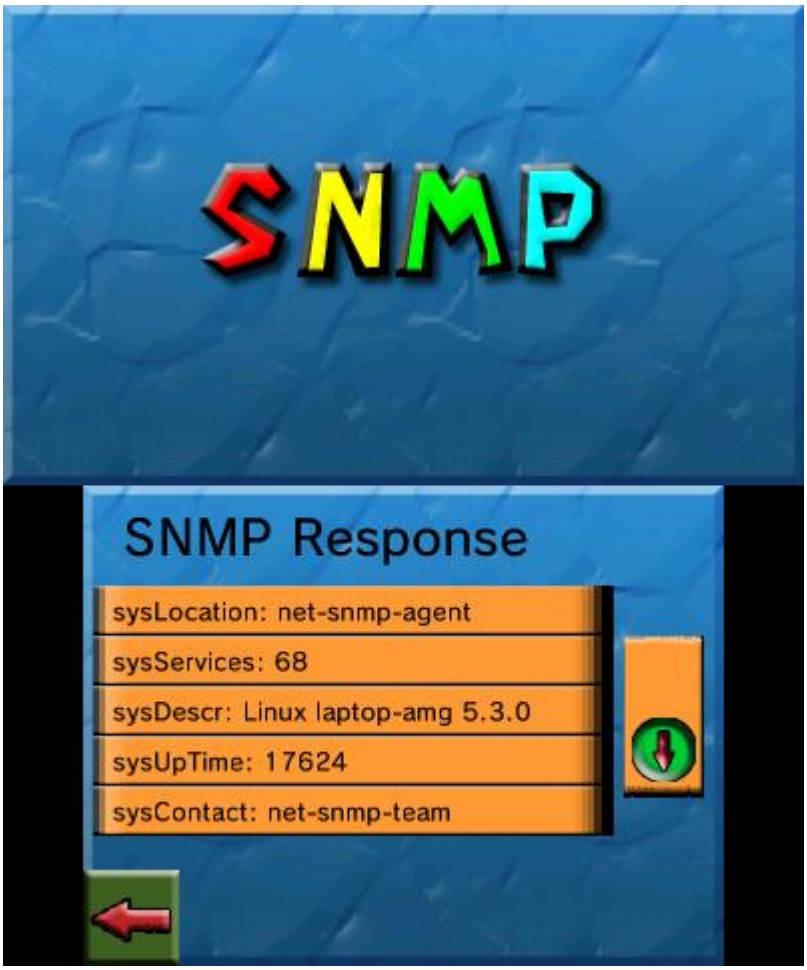

Figura 38- Ejemplo de respuesta SNMP

Por supuesto, esta opción es bastante útil si queremos acceder a objetos de gestión que sean escalares, pero si queremos acceder a los contenidos de una tabla debemos realizarlo de otra manera. Para este caso, deberemos seleccionar la tabla a visualizar en el explorador de MIBs, que nos redirigirá a una nueva pantalla con los valores de las columnas de una fila de la tabla.

Dicha fila podremos cambiarla introduciendo el número de fila que queramos. Si la fila especificada no existe en el agente, simplemente aparecerán las columnas de la tabla vacías.

Como podemos observar en la siguiente captura, tenemos una pequeña cajita en la que podemos especificar el tipo de una columna de la tabla y, posteriormente, el valor que debe tomar dicha columna para la fila seleccionada. Nada más especificar el valor, se enviará un SetRequest al agente con la modificación deseada. Si hacemos de nuevo un GetRequest a la fila de la tabla deseada, podremos ver los cambios reflejados si no ha ocurrido ningún error en la transmisión de la petición.

Aunque este visor de tablas es bastante útil, para que fuera más completo faltaría la opción de añadir y borrar filas de la tabla, que no se ha realizado por cuestión de simplicidad en las funciones implementadas. Sin embargo, se ve que cumple con su función de manera correcta.

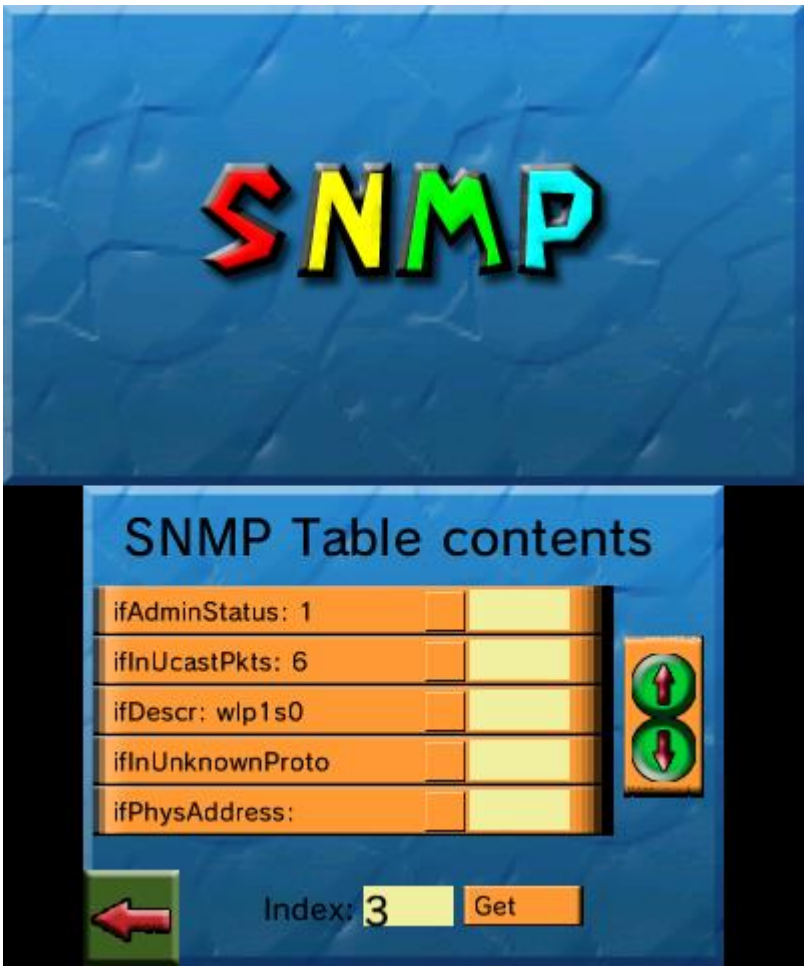

Figura 39- Visor de tablas SNMP

5.4 Receptor de alertas y notificaciones

En este apartado se tratará la recepción de alertas y notificaciones tanto de agentes SNMP en cualquiera de las 3 versiones del protocolo como de servidores Syslog que nos envíen registros usando el transporte TCP o UDP.

5.4.1 Recepción de traps SNMP

Para la recepción de traps SNMP, aunque se soportan las 3 versiones del protocolo y los 2 tipos de mensajes soportados (Trap y Inform, este último necesitando ser asentido por el gestor), la opción más recomendable a la hora de enviar notificaciones SNMP por la red es usar la última versión disponible (la 3) con algún modelo de seguridad (en este caso el USM).

Para poder lograr esto, primero debemos dirigirnos a las opciones y registrar el usuario SNMPv3 con el que vamos a autenticarnos para recibir las notificaciones y alertas. En la siguiente captura se muestra el formulario a rellenar para introducir un usuario en la base de datos de usuarios SNMPv3, donde especificamos el nombre de usuario, los algoritmos de autenticación y cifrado (puede no usar alguno o ninguno de ellos), así como la contraseña para cada algoritmo.

Figura 40- Añadir un usuario SNMPv3

Una vez definido el usuario con el que vamos a recibir las notificaciones, debemos indicar en las opciones de SNMPv3, en el campo "Trap User" el usuario que vamos a usar, puesto que en la base de datos de usuarios puede haber varios de ellos. Sin embargo, si usamos los protocolos SNMPv1 o SNMPv2, con activar la recepción de traps e indicar el puerto de escucha, así como el nombre de la comunidad para autenticarse, sería suficiente.

Cabe destacar que activar o desactivar las recepción de notificaciones tanto del protocolo SNMP como del protocolo IETF-Syslog tomará efecto una vez se reinicie la aplicación. Esto es debido a una limitación de la API de sockets que impide reutilizar el puerto de escucha de un socket por otro socket. De esta forma, la única opción es crear los sockets de escucha necesarios al comienzo de la aplicación y destruirlos al salir de ella. Esta limitación es comprensible debido a que es bastante raro que una videoconsola como es la Nintendo 3DS se use como servidor, cuando normalmente suele cumplir el rol de cliente.

Como último parámetro, podemos especificar cuántos traps se almacenarán como máximo. Si se supera el límite de traps almacenados, se borrará el más antiguo y se añadirá el nuevo. Se ha establecido un límite de 100 notificaciones como tope que no se puede superar.

En la siguiente captura se puede observar esta pantalla de configuración para la recepción de notificaciones y alertas del protocolo SNMP:

Figura 41- Configuración de la recepción de traps

Una vez hemos configurado y activado la recepción de traps del protocolo SNMP, cuando reiniciemos la aplicación se creará uno o varios sockets de escucha en los puerto que hemos especificado para recibir las notificaciones. En el menú principal, además, cuando llegue una notificación aparecerá en la barra de notificaciones de la pantalla superior, así como la reproducción de un pequeño sonido para alertar al usuario de la aplicación (el gestor de redes).

Este comportamiento puede observarse en la siguiente captura, donde se recibe un trap de SNMPv2 a las 13:49 horas.

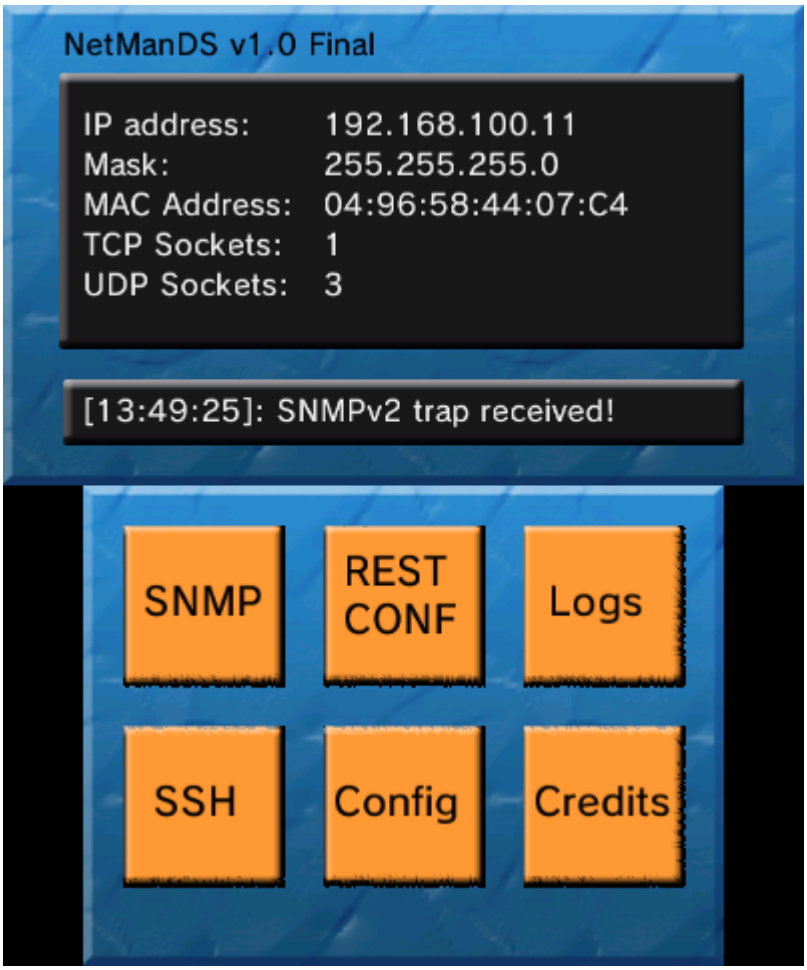

Figura 42- Ejemplo de recepción de trap SNMPv2

Una vez hemos recibido unas cuantas notificaciones, es hora de mostrarlas para poder tomar decisiones dependiendo de la información que estas contengan. Para ello, se ha preparado el botón de Logs en el menú principal que nos permitirá navegar de forma sencilla y cómoda por las notificaciones que han sido recibidas, tanto por el protocolo SNMP como el protocolo IETF-Syslog.

En la siguiente captura se puede observar una lista con los traps recibidos, indicando la hora de recepción así como la versión del protocolo SNMP usada en la recepción de la notificación. Nótese que se ha incluido un botón para visualizar, además, los registros Syslog recibidos, que se mostrarán en una lista similar a la ya comentada.

Figura 43- Visualización de traps SNMP

Podemos observar que se han recibido 4 traps, dos mediante el protocolo SNMPv3, otro con el protocolo SNMPv2c y otro con el protocolo SNMPv1, para dar seguridad de que se soportan las 3 versiones del mismo. Además, aparecen ordenados por orden de recepción, visualizando los traps más recientes primero.

Si pinchamos en cualquiera de ellos, podemos observar la información que contienen, y que nos servirá como gestores de redes para tomar decisiones o sacar conclusiones acerca del funcionamiento del nodo gestionado. En la siguiente captura, hemos pinchado en el trap recibido por SNMPv1, observando que se indica en una lista los elementos que el trap incluye, como el OID de la empresa, la dirección IP del agente que envía ese trap, así como el tipo de trap recibido. Aparecen más campos que se pueden mostrar navegando por la lista, como los objetos de gestión que se incluyen en la notificación o el tipo específico de trap, que no se muestra en la captura por falta de espacio en la pantalla.

Figura 44- Visualización de un trap SNMPv1

5.4.2 Recepción de notificaciones Syslog

De forma similar a como se ha comentado en el protocolo SNMP, la recepción de registros usando el protocolo IETF-Syslog funciona de manera similar. Esta vez, debemos configurar el transporte a utilizar y número de puerto en las opciones de la aplicación, en su sección dedicada a Syslog.

Cuando hayamos configurado y activado la recepción de registros Syslog, debemos reiniciar la aplicación para que los cambios sean aplicados. A partir de este momento, en el menú principal de la aplicación, en la barra de notificaciones, se nos avisará cuando un nuevo registro Syslog haya sido recibido, notificando al usuario de la aplicación nuevamente con un pequeño sonido.

Si nos vamos a la pantalla de visualización de notificaciones que cubrimos con el protocolo SNMP y seleccionamos la opción “Syslog”, podremos visualizar los registros recibidos, indicando el transporte usado (TCP o UDP) así como la hora de su recepción. En la siguiente captura podemos observar este comportamiento:

Figura 45- Visualización de registros Syslog recibidos

Como en el caso de las notificaciones SNMP, podemos pulsar cada uno de los registros para obtener más detalles acerca del mismo. En las siguientes capturas se muestra esa información, donde se incluyen aspectos como la prioridad de ese registro, la versión de Syslog utilizada, la fecha y hora de generación de ese registro, el nombre de la máquina donde se ha generado el registro, el proceso de esa máquina, así como un mensaje descriptivo de ese registro. Como siempre, podemos tocar cada elemento de la lista para mostrar la información completa de cualquiera de los campos, si esta aparece cortada debido a la falta de espacio en la pantalla.

Figura 46- Visualización de un registro Syslog I

Figura 47- Visualización de un registro Syslog II

5.5 Cliente SSH

Otra de las funcionalidades que se ha añadido a la aplicación es la de poner realizar conexiones SSH con un servidor remoto a fin de poder realizar tareas de configuración y mantenimiento. Para poder realizar una conexión, es necesario rellenar el siguiente formulario:

Figura 48- Formulario de sesión SSH

En este formulario debemos indicar la IP o nombre de host (que se resolverá por DNS) que identifica al servidor SSH al que queremos conectarnos. Además, debemos introducir el usuario y la contraseña para identificarnos. Una vez rellenado el formulario, si pulsamos en el botón "Connect" nos redirigirá a una pantalla donde podremos observar el terminal, y una caja de texto editable que, si la pulsamos, podremos enviar un comando al servidor para que se ejecute, obteniendo la respuesta del mismo.

En la siguiente captura se puede observar la conexión satisfactoria con un servidor SSH, así como la ejecución de algunos comandos para poder comprobar su funcionamiento correcto. Podemos cerrar la conexión pinchando en la flechita en la esquina inferior izquierda.

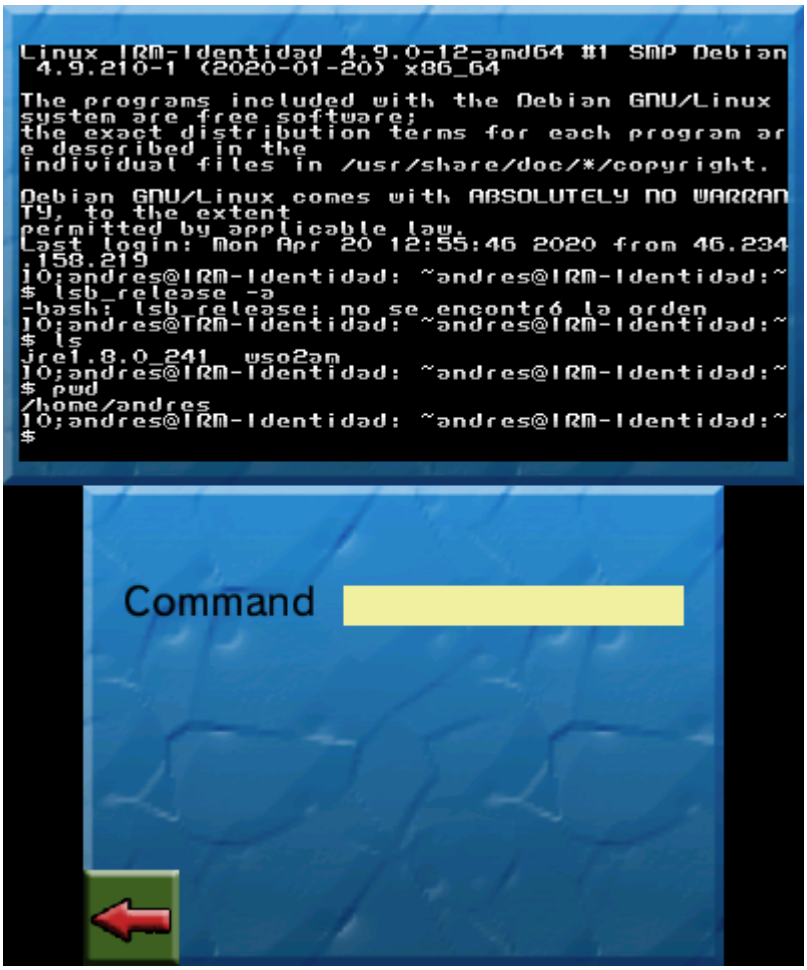

Figura 49- Prueba de conexión SSH

5.6 Cliente RESTCONF

Como última funcionalidad incluida en este proyecto, se ha implementado de forma parcial el protocolo RESTCONF para la configuración remota de nodos en una red IP. Como principales características, permite la exploración de módulos YIN, que no es más que un módulo YANG pero en formato XML para facilitar su lectura, al no haber muchas librerías ligeras para leer ficheros en YANG. En esos módulos se encuentran una especie de objetos de gestión con los que podemos realizar diversas operaciones. Las operaciones a las que se ha dado soporte es la lectura y escritura de objetos escalares y listas de objetos, y la eliminación de objetos de una lista de objetos. Por supuesto, el protocolo RESTCONF permite muchos más, como son las llamadas a procedimiento remoto (RPC), actualización de firmware… pero no se han realizado por cuestiones de tiempo, prefiriendo implementar solo las funcionalidades más básicas.

Para poder usar este protocolo, lo primero es entrar en las opciones de la aplicación y detallar en la sección dedicada a RESTCONF el servidor que se va a acceder (especificando la URL, usando HTTP o HTTPS y, posiblemente, el número de puerto). Además, debemos especificar las credenciales que vamos a usar para autenticarnos. En este caso, basta con un usuario y una contraseña puesto que se va a usar el modelo de autenticación HTTP Basic. En la siguiente captura se pueden observar la configuración para poder acceder al servidor RESTCONF de pruebas de Cisco, que es accesible de forma pública:

Ilustración 1- Configuración del protocolo RESTCONF

Una vez configurado el protocolo, debemos acceder a la opción "RestConf" del menú principal, que directamente nos direccionará al explorador de módulos YIN. De manera similar al explorador de MIBs, debemos seleccionar qué módulo leer. Los módulos deben situarse en una carpeta llamada "/yin" situada en el mismo directorio que el ejecutable de la aplicación. En la siguiente captura podemos observar el cargador de módulos YIN:

Figura 50- Selección de un módulo YIN

Una vez seleccionado el módulo, se mostrará un pequeño navegador que mostrará cada nodo del documento XML, donde podemos ir recorriendo el árbol.

Como se puede observar en la siguiente captura, en algunos nodos aparece un icono de suma (+) que significa que el explorador de módulos YIN ha determinado que este nodo referencia a un objeto gestionable. Existen 3 elementos a los que se les ha dado soporte para poder enviar peticiones RESTCONF: los "container", que no son más que un análogo a los grupos de una MIB, los "leaf" que referencian a objetos escalares, y los "list", que son una especie de tabla si lo comparamos con el protocolo SNMP.

Si pulsamos en cualquiera de los elementos, obtendremos más información incluida en el módulo sobre el mismo, o navegaremos más profundo en el árbol si tiene más nodos hijo. Al pulsar en el icono "+" que indicamos anteriormente, nos iremos a una nueva pantalla donde podremos detallar qué operación queremos realizar con el elemento seleccionado. Existen 3 posibles: Get, Post, o Delete, según el método HTTP que se quiera usar.

En el caso de las listas, al seleccionar un método HTTP se permite introducir más parámetros a la petición (que se introducen en la URL como parámetros GET). Algunos ejemplos son indicar qué elemento de una lista se quiere acceder, filtrar la información que el servidor debe devolver (distinguiendo entre información de configuración e información que no está relacionada con la configuración, o simplemente devolver ambos tipos de información), e incluso especificar qué campos de la lista se quieren acceder. Por ejemplo, podríamos obtener solamente las direcciones MAC de las interfaces de red del nodo gestionado, sin tener que recibir el resto de campos que conforma una interfaz de red.

Figura 51- Exploración de un módulo YIN

Una vez especificados todos estos parámetros, la petición RESTCONF se mandará al servidor. Si todo ha ido bien, el servidor mandará una respuesta a la aplicación. En caso contrario, se indicará al usuario de la aplicación qué error ha ocurrido. En la siguiente captura se puede ver un ejemplo de respuesta a una petición de obtener todas las interfaces de red, que se imprime en formato JSON.

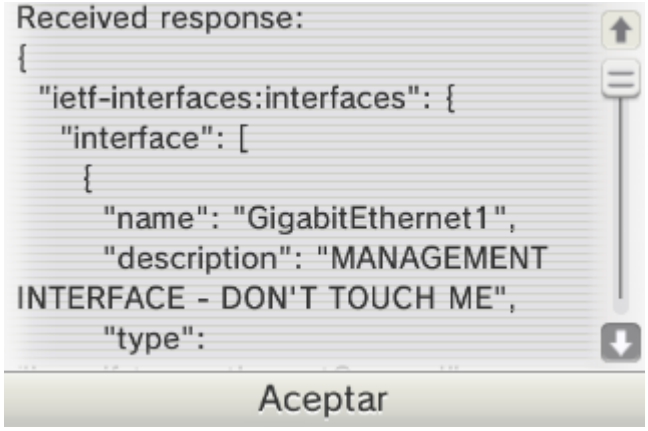

Figura 52- Ejemplo de respuesta a una petición RESTCONF

6 ESCENARIO DE PRUEBA

Una vez visto completamente el diseño de la aplicación y detalles sobre su implementación, así como aspectos de su funcionamiento general, es hora de detallar el software adicional que se ha usado para la realización de las pruebas de funcionamiento de la aplicación, a fin de verificar y validar los requisitos anteriormente desarrollados. Se detallará la topología del escenario de prueba que se ha usado, así como las pruebas que han sido realizadas, incluyendo los resultados de las mismas.

6.1 Descripción del escenario

A continuación se muestra un diagrama de red con todos los elementos que componen el escenario de prueba. Más adelante introduciremos cada uno de los servidores individualmente y qué implementación de los mismos se ha usado. El diagrama se resume en que un actor que es el gestor de redes de telecomunicación tiene acceso a un sistema Nintendo 3DS para poder realizar labores de gestión sobre diferentes servidores y nodos que soportan tecnologías y protocolos diferentes de gestión. En cuanto a la conexión, la Nintendo 3DS solo permite un medio WiFi según la norma IEEE 802.11b/g para transferir información, por lo que necesitamos de un AP (Access Point) para poder conectarnos a Internet, que nos proporcionará la infraestructura necesaria para poder alcanzar cada uno de los servidores.

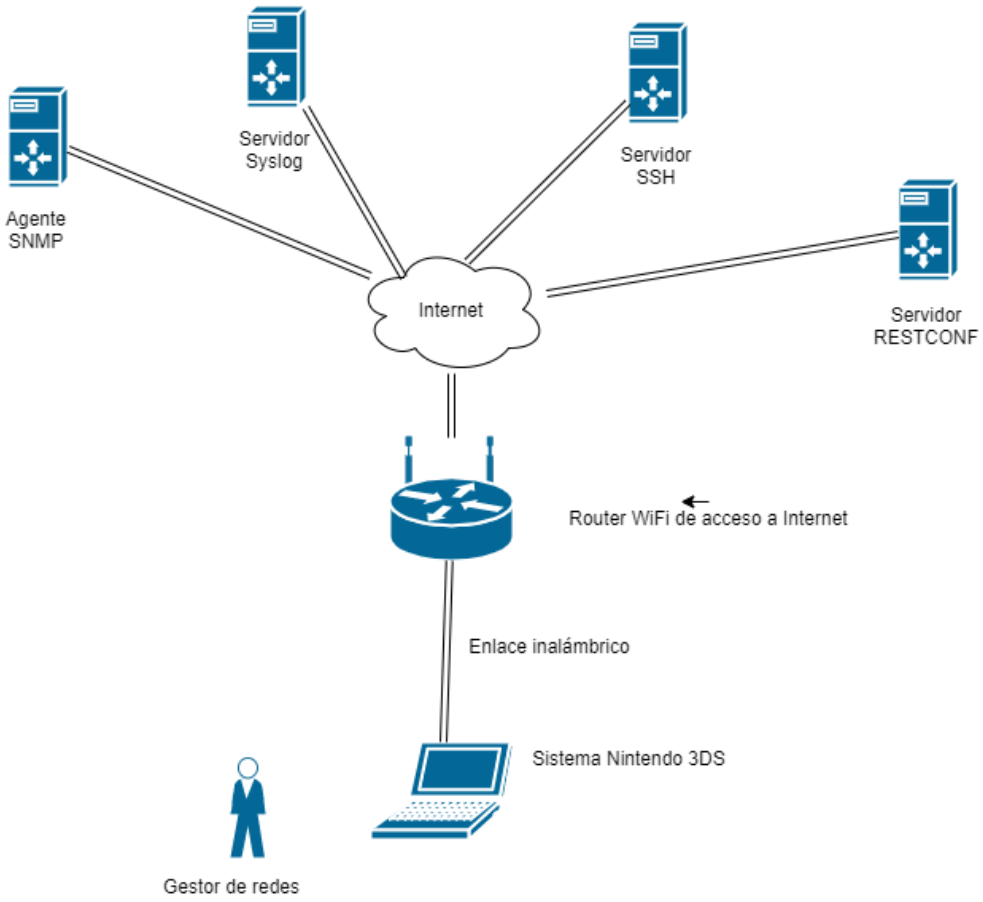

Figura 53- Escenario de pruebas

6.1.1 Agente SNMP

En cuanto al agente SNMP usado, se ha preferido usar la implementación del grupo Net-SNMP [13] debido a, principalmente, su facilidad de uso, su disponibilidad en todos los sistemas operativos de escritorio más conocidos y su gran estabilidad. Además, cuenta con una documentación excelente para familiarizarse con la configuración del servidor, que puede consultarse en [14], por no hablar de una comunidad bastante activa aún en la actualidad que está dispuesta a ayudarnos en cualquier problema que tengamos.

Figura 54- Logotipo de Net-SNMP

En cuanto a sus características, implementa prácticamente al 100% el protocolo SNMP tanto del lado del agente como del lado del gestor, puesto que incluye herramientas para realizar peticiones SNMP. Soporta SNMPv1, SNMPv2c y SNMPv3 con los modelos de seguridad USM y TSM, este último usando certificados X.509 para asegurar la confidencialidad, integridad y autenticidad de las comunicaciones.

Del lado del agente, además, soporta una manera sencilla de configurar su funcionamiento mediante la edición de un fichero de texto de configuración. Siguiendo los tutoriales en [14] podemos configurar este agente para cumplir cualquier especificación que tenga que ver con el protocolo SNMP. Soporta VACM (Control de Acceso basado en Vistas) que permite una mayor granularidad en cuanto a la asignación de permisos de acceso a los usuarios a determinados objetos de gestión.

Por si fuera poco, incluye un gestor SNMP gráfico, llamado TkMib, que nos permite realizar pruebas en el agente sin mayores complicaciones, así como explorar las MIBs que estemos usando en el agente. Para probar la recepción de traps, tenemos a nuestra disposición un receptor de traps cuya interfaz de configuración es similar a la del agente.

6.1.2 Servidor Syslog

La implementación del servidor Syslog escogida, a pesar de que existe una gran variedad disponibles en la red, se ha optado por usar “syslog-ng” [15], que es uno de los pocos que permiten configurar el envío de registros syslog según el protocolo IETF-Syslog, definido en la RFC 5424.

Figura 55- Logotipo de syslog-ng

En cuanto a las funcionalidades que ofrece, principalmente permite enviar registros syslog que hayan sido almacenados en nuestro sistema, e incluso recibir registros de otros servidores syslog, soportando una amplia variedad de protocolos syslog. Además, permite filtrar los registros que deben mandarse y a qué destino deben mandaste, soportando una amplia variedad de opciones en este aspecto. Podemos especificar el transporte usado, que puede ser TCP, UDP, TLS o DTLS, ofreciendo unos niveles de seguridad a los registros enviados en la red.

Por supuesto, podemos especificar qué fuentes usar para obtener los registros syslog que deben enviarse a los destinos previamente configurados. Pueden ser ficheros generales del sistema operativo (como en Linux tenemos el fichero /var/log/messages) o ficheros específicos que genera alguna aplicación concreta. Estos ficheros que se han especificado serán monitorizados por el servidor observando cambios. Si los detecta, enviará los registros nuevos a los destinos configurados.

En general, es una herramienta muy sencilla de utilizar y que apenas consume recursos de nuestra máquina, y que nos permite monitorizar el funcionamiento de nuestro servidor o de cualquier sistema informático en general.

6.1.3 Servidor SSH

Si nos centramos en el servidor SSH usado, es simplemente el servidor proporcionado por OpenSSH [16]. Se soportan las dos versiones del protocolo SSH y es fácilmente configurable, como los dos servidores anteriores. Permite la configuración de varios métodos de autenticación con los clientes, como puede ser el típico usuario-contraseña, o algunos más avanzados mediante el uso de la clave pública del cliente o incluso certificados.

Figura 56- Logotipo de OpenSSH

6.1.4 Servidor RESTCONF

Por último, el servidor RESTCONF utilizado para las pruebas es el servidor RESTCONF de pruebas que se usa en los Labs gratuitos de Cisco. Se puede acceder en [17], teniendo en cuenta que es necesario poseer el usuario y la contraseña para acceder a los módulos de configuración. Se pueden conseguir viendo el Lab correspondiente a RESTCONF disponible en la página oficial de Cisco [18].

Figura 57- Logotipo de Cisco

Soporta los módulos YANG más comunes en NetConf, como son los diseñados por el IETF. Para probar la aplicación, se ha usado el módulo ietf-interfaces para obtener información de las interfaces, crear interfaces nuevas e incluso borrarlas, para validar las operaciones del protocolo RESTCONF que han sido implementadas. Permite la llamada a algunos procedimientos remotos y otras funcionalidades no cubiertas por el proyecto. En resumen, es una gran herramienta para probar clientes RESTCONF o incluso experimentar con el protocolo.

6.2 Pruebas realizadas y resultados

Para finalizar la sección, vamos a resumir las pruebas que se han realizado a la aplicación usando el escenario anterior para verificar su correcto funcionamiento. Se irán incluyendo en pequeñas tablas donde se indica qué funcionalidad se ha probado y cuáles han sido los resultados obtenidos.

Prueba 1	*Recepción de notificaciones SNMP y Syslog*
Fecha y hora	29-04-2020@12:23
Descripción	Activar la recepción de notificaciones para las 3 versiones del protocolo SNMP y el protocolo Syslog en sus modos de transporte TCP y UDP. Comprobar que se reciben las notificaciones en el menú principal y que se almacenan en sus respectivos ficheros.
Resultados	*La aplicación es capaz de recibir todos los tipos de notificaciones, avisando al usuario de la aplicación en el menú principal y permitiendo su visualización en el menú de Logs.*

Tabla 22- Test de prueba 1

Prueba 2	*Conexiones SSH*
Fecha y hora	29-04-2020@12:35
Descripción	Conectarse a un servidor SSH remoto e introducir comandos. Observar que se recibe una respuesta coherente y comprobar, mediante el uso de otro cliente, que las acciones realizadas en la aplicación han surtido efecto en el servidor.
Resultados	*La aplicación es capaz de iniciar una conexión SSH con un servidor remoto, introducir comandos, enviarlos y recibir la pertinente respuesta. Usando un cliente SSH de escritorio, se ha podido observar la existencia de un fichero nuevo en el servidor, creado por la aplicación.*

Tabla 23- Test de prueba 2

Prueba 3	*Exploración de módulos YIN*
Fecha y hora	29-04-2020@12:49
Descripción	Explorar 5 módulos YIN y ser capaz de navegar entre todos sus nodos, mostrando la información pertinente en cada uno de ellos.
Resultados	*La aplicación es capaz de listar los ficheros YIN instalados con la aplicación, es capaz de abrir 5 de ellos y navegar por todos sus nodos, sin obtener ningún tipo de error o anomalía. Puede mostrar la información descriptiva de cada nodo.*

Tabla 24- Test de prueba 3

Prueba 4	*Peticiones RESTCONF*
Fecha y hora	29-04-2020@13:10
Descripción	Enviar una petición GET, otra POST y otra DELETE a un servidor RESTCONF y observar que se realizan correctamente dichas operaciones.
Resultados	*La aplicación es capaz de realizar las operaciones de lectura, escritura y borrado de un elemento definido en un módulo YIN.*

Tabla 25- Test de prueba 4

Prueba 5	*Edición de parámetros de configuración*
Fecha y hora	29-04-2020@13:33
Descripción	Configurar varios parámetros de las opciones y comprobar que surten efecto. Al reiniciar la aplicación, los cambios quedan guardados.
Resultados	*La aplicación es capaz de configurar los parámetros referentes a los protocolos que implementa, guardarlos y recuperarlos para otra sesión.*

Tabla 26- Test de prueba 5

Prueba 6	*Añadir usuarios SNMPv3*
Fecha y hora	29-04-2020@13:40
Descripción	Añadir varios usuarios SNMPv3, con distintos parámetros de autenticación y cifrado. Posteriormente eliminarlos. Comprobar la persistencia de los datos.
Resultados	*La aplicación es capaz de añadir, modificar y eliminar usuarios SNMPv3 con características diferentes. Los datos de los usuarios se almacenan de forma correcta en un fichero y, posteriormente, se recuperan de forma correcta.*

Tabla 27- Test de prueba 6

Prueba 7	*Descubrimiento de agentes*
Fecha y hora	29-04-2020@13:51
Descripción	Realizar un escaneo de agentes SNMP teniendo 2 agentes encendidos en una misma subred. Comprobar que los agentes se detectan y se muestra la información general de cada uno de ellos.
Resultados	*La aplicación es capaz realizar un escaneo de agentes SNMP en una subred dada, detectar todos los agentes existentes y mostrar la información general de cada uno de ellos.*

Tabla 28- Test de prueba 7

Prueba 8	*Exploración de MIBS*
Fecha y hora	29-04-2020@14:02
Descripción	Explorar 5 MIBS y ser capaz de navegar entre todos sus objetos de gestión, mostrando la información pertinente en cada uno de ellos.
Resultados	*La aplicación es capaz de listar las MIBS instaladas con la aplicación, es capaz de abrir 5 de ellas y navegar por todos sus objetos de gestión, sin obtener ningún tipo de error o anomalía. Puede mostrar la información descriptiva de cada objeto de gestión.*

Tabla 29- Test de prueba 8

Prueba 9	*Envío de peticiones SNMP*
Fecha y hora	29-04-2020@14:15
Descripción	Mandar una petición SNMP de cada tipo: Get, GetNext, Set y GetBulk y comprobar su correcto funcionamiento.
Resultados	*La aplicación es capaz de enviar peticiones SNMP de los tipos especificados, recibir una respuesta del agente y listar los objetos de gestión contenidos en ella.*

Tabla 30- Test de prueba 9

Prueba 10	*Visualización y edición de tablas SNMP*
Fecha y hora	29-04-2020@14:21
Descripción	Explorar una tabla SNMP y ser capaz de mostrar todas sus filas. Editar 3 campos y comprobar los efectos de la edición de estos campos.
Resultados	*La aplicación es capaz de visualizar una tabla SNMP (en este caso las interfaces del nodo de una red), es capaz de acceder a todas sus filas. Se ha editado el estado administrativo de 3 interfaces de red, comprobando los efectos en el hardware del agente.*

Tabla 31- Test de prueba 10

7 CONCLUSIONES Y LÍNEAS DE AVANCE

Por último, en este último apartado se van a sacar una serie de conclusiones sobre la aplicación desarrollada y un conjunto de posibles mejoras o líneas de avance que podrían tenerse en consideración para ampliar las funcionalidades del proyecto en algún momento dado o, incluso, en otro proyecto derivado de este.

En cuanto a la utilidad del mismo, es una aplicación que resulta bastante versátil puesto que puede usarse en todo tipo de circunstancias, aún cuando se usan tecnologías y protocolos de gestión no homogéneos en la red a gestionar. Además, el hecho de que se haya desarrollado para un dispositivo portátil para el que no se suelen desarrollar herramientas de este tipo le da cierta comodidad al usuario final y, claramente, cierto atractivo. Por no hablar de las ventajas a nivel de seguridad que proporciona usar sistemas poco reconocidos en el ámbito de la gestión de redes donde no se están constantemente intentando explotar agujeros de seguridad (o "exploits").

Sin embargo, la plataforma en la que se ha desarrollado la aplicación no es del todo sencilla de usar y requiere un aprendizaje previo, especialmente de las llamadas al sistema y los servicios que nos proporciona el sistema operativo para agilizar el desarrollo. Además, la cantidad de librerías disponibles es mínima (solamente las proporcionadas por devkitPro), siendo necesario que adaptemos librerías que necesitemos por nuestra cuenta para poder usarlas en el sistema Nintendo 3DS.

Además, hay que destacar que la aplicación desarrollada no es perfecta y tiene mucho margen de mejora. A continuación se exponen los principales incentivos para mejorar el funcionamiento de la aplicación:

- Añadir más algoritmos de seguridad al protocolo SNMPv3, como el algoritmo AES para cifrado o versiones más modernas del algoritmo de resumen SHA.
- Implementar el modelo de seguridad TSM en SNMPv3 para ofrecer una alternativa a USM.
- Añadir la recepción segura de registros Syslog mediante el uso de certificados X.509
- Añadir un editor de ficheros JSON más sofisticado para enviar peticiones POST en RESTCONF.
- Implementar más funcionalidades del protocolo RESTCONF, como las llamadas a procedimiento remoto (RPC).
- Habilitar otros métodos de autenticación en el módulo SSH.
- Incorporar una utilidad de gestión de certificados X.509 para aquellos protocolos que los usen.
- Implementar protocolos de gestión ampliamente usados en IoT como LwM2M, protocolos basados en CoAP o incluso MQtt para la recepción de notificaciones provenientes de motas.

REFERENCIAS

[1] SolarWinds: https://www.solarwinds.com/es/

[2] Splunk: https://www.splunk.com/es_es

[3] devkitpro: https://www.devkitpro.org

[4] Citra3DS: https://citra-emu.org

[5] Documentación de Citro2D: https://citro2d.devkitpro.org/

[6] Documentación de tinyxml2: http://leethomason.github.io/tinyxml2/

[7] Documentación de jansson: https://jansson.readthedocs.io/en/2.12/

[8] Documentación de mbedtls: https://tls.mbed.org/api/

[9] Documentación de libssh2: https://www.libssh2.org/docs.html

[10] Repositorio de libtmt: https://github.com/deadpixi/libtmt

[11] Repositorio de la aplicación: https://github.com/amg98/NetManDS

[12] Plantilla para la especificación de requisitos, por MADEJA: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/407

[13] Net-SNMP: http://www.net-snmp.org/

[14] Tutoriales de Net-SNMP: http://www.net-snmp.org/wiki/index.php/Tutorials

[15] Sitio oficial de syslog-ng: https://www.syslog-ng.com/

[16] Sitio oficial de OpenSSH: https://www.openssh.com/

[17] Servidor RESTCONF de pruebas: https://ios-xe-mgmt.cisco.com:9443

[18] Sitio oficial de Cisco: https://www.cisco.com/

[19] Sinche, Soraya et al. "A Survey of IoT Management Protocols and Frameworks" IEEE Communications Surveys & Tutorials (2019)

[20] McClintic, Matthew, et al. "Keyshuffling Attack for Persistent Early Code Execution in the Nintendo 3DS Secure Bootchain"

[21] Seire, Michael, et al. "Attacking the Nintendo 3DS Boot ROMs"