UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Proyecto Final

Ingeniería Electrónica

2021

Monitoreo de alarmas para Datacenter

Tesina

Casas Daiana, Pregelj José, Riveros Ruben, Vieiro Alexis

Índice

1. Abstract

El presente proyecto se enfoca en la monitorización de Datacenters donde se necesita un sistema de monitoreo para prever eventos o atenderlos y generar acciones correctivas o un tratamiento adecuado, configurándose en un sistema central incorporando distintos parámetros del ambiente a través de un dispositivo electrónico.

El objetivo principal se centra en la realización de este dispositivo y el manejo de las notificaciones basados en una aplicación de celular para mayor portabilidad. El dispositivo es capaz de aceptar diferentes sensores y establecer conexión con el servicio de monitorización para volcar las distintas métricas y configurar las alertas pertinentes del datacenter. Dentro del mercado local se encuentra una oportunidad de desarrollo de este tipo de sistema de monitorización que independiza al usuario de grandes soluciones llave en mano y limitaciones en su expansión de métricas.

Para el desarrollo se realiza una etapa de investigación y análisis del mercado de la zona, incluyendo la participación y apoyo de una empresa local argentina. Se identifican las tecnologías actuales y limitaciones de las mismas aplicadas a distintos entornos para establecer el alcance de la solución.

Para llevar a cabo el proyecto de desarrollo, las pertinentes pruebas y finalmente conclusiones se utiliza un sistema de gestión de división de tareas, responsables, duplas de trabajo y definición de hitos importantes en fecha y tiempo para cumplir. Así también se identificaron los potenciales problemas para definir plan de acción y minimizar impacto en tiempos y alcance del proyecto.

El servicio de monitorización utilizado es Zabbix, muy conocido e instalado en grandes compañías de comunicaciones, es el servicio central y se complementa con su Zabbix-Daemon que permite disponibilizar la información en tiempo real. Esta información se basa en el uso del protocolo SNMP.
El dispositivo electrónico se encarga de suministrar la información al servidor con la siguiente capacidad:

- 1 sensor de temperatura
- 4 acciones de puerta

Por otra parte se dispone de una aplicación llamada 'Zabbix Mobile App', la cual permite acceder en tiempo real a los valores de las métricas, el historial de las mismas y la recepción de notificaciones de aquellos parámetros de interés. La aplicación admite las versiones de Android 4.4 en adelante.

Gracias al punto de vista ofrecido por Ericnet S.A. se pudo orientar el ambiente de aplicación del producto, acotar la solución y buscar un resultado para implementar en clientes.

Finalmente se obtiene un producto y servicio orientado a pequeñas empresas que tengan salas de Datacenters,como así también un punto inicial para expandirse en grandes empresas gracias a la robustez alcanzada que permite expandir las aplicaciones.

2. Introducción Teórica del tema a tratar y Marco Teórico

Para establecer los fundamentos conceptuales que sustentan este proyecto, se procede a definir cada concepto y proporcionar un marco teórico para su comprensión.

2.1. Datacenters

Según VMWare , se define [**Datacenter**] a aquel ambiente físico dedicado a alojar equipos físicos, ya sean servidores, conexiones y otros recursos necesarios para mantener una red o un sistema de computadoras de información .
Este ambiente físico son salas donde se disponen todos los equipos acomodados en estructuras de forma organizada, llamada racks (Fig. 1).

Figura 1: Instalaciones de datacenter

Algunos [**Datacenter**] pueden tener una sala de energía que mantiene energizado todo el equipamiento en todo momento. También pueden tener una sala de control (Fig. 2) de los equipos, generalmente son los administradores de los mismos, llamada *NOC* (Network Operations Center).

2.2. Monitoreo

Dentro de las soluciones de Cisco System encontramos la explicación y aplicación de [**Datacenter**]. Del cuál remarcamos que es importante conocer el estado de nuestros equipos, recursos y servicios para prever eventos o atenderlos y generar acciones correctivas o un tratamiento adecuado. Esta visualización constante de estos estados, mensajes o cualquier otro evento en tiempo real se llama monitorización.

Existen desarrollos de software que nos permiten realizar estas tareas, que se denominan *NMS* "Network Management Software". Este es el encargado de recopilar la información, procesarla y hace la disponible para que el usuario interactúe con aquello que esta monitorizando.

Figura 2: Monitorización

Algunos ejemplos de *NMS* actuales son Zabbix, Nagios, OpenNMS, PandoraFMS, The Dude, PRTG Network Monitor.

2.3. SNMP

SNMP (Simple Network Management Protocol) es un protocolo estandarizado para la capa de aplicación (Fig. 3). Tiene una arquitectura cliente - servidor donde los agentes/clientes son los equipos y el servidor es el *NMS*. ([**SNMP**])

Dentro de este intercambio de información hay distintas formas de realizarlo, las cuales se detallan a continuación.

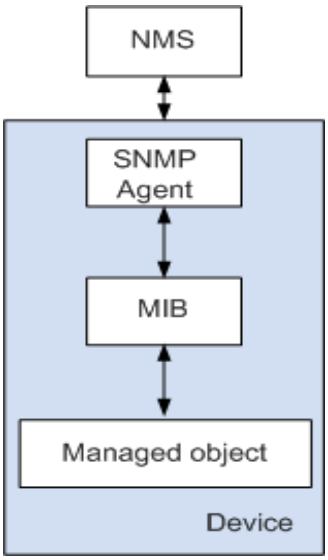


Figura 3: Protocolo SNMP

2.3.1. GET

GET es una solicitud enviada por el servidor *NMS* al equipo. Luego de recibir el *GET* el agente *SNMP* ejecuta la instrucción especificada en la solicitud y correspondiente en el *MIB* para luego responder con el resultado al *NMS* (Fig. 4). Las operaciones que pude realizarse en *SNMP GET* incluye Get, GetNext y GetBulk.

- Get: esta operación permite al *NMS* obtener una o más variables desde el *SNMP* agente.
- GetNext: esta operación permite al *NMS* obtener una o mas variables subsecuentes desde el *SNMP* agente.
- GetBulk: esta operación es igual que ejecutar consecutivos GetNext. Se puede configurar la cantidad de operaciones que implica en la operación GetBulk.

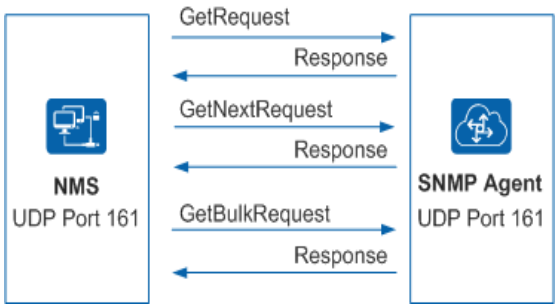


Figura 4: Proceso GET

En la figura 5 se observa el formato de los paquetes de la operación *GET SNMP*

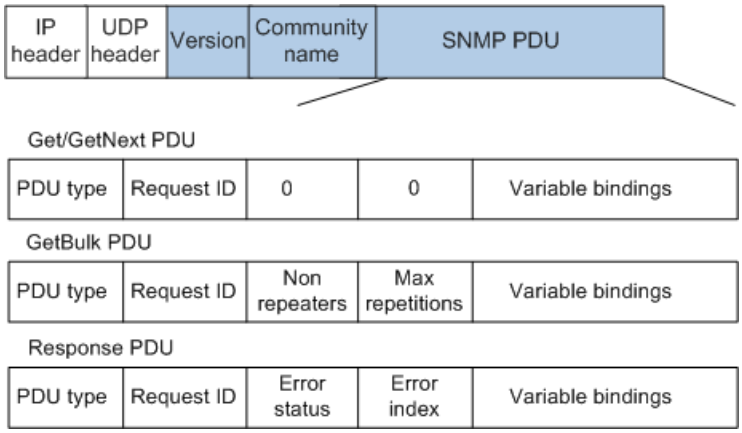


Figura 5: Contenido del paquete GET

A continuación se detallan los campos del paquete

- Versión: especifica la versión de *SNMP* a utilizar. El valor 0 es para la versión 1 y 1 para la versión 2c.
- Community name: especifica la autenticación entre el agente *SNMP* y el *NMS*.Este valor es de tipo texto.
- Request ID: especifica un único ID de cada solicitud realizada. Se utiliza para la correspondencia solicitud - respuesta.
- Non repeaters/Max repetitions: La operación GetBulk es igual a realizar operaciones consecutivas GetNext. Ambos campos permiten setear el

numero de operaciones GetNext.

- Error status: especifica el estado de un error que ocurre en el procesamiento de la solicitud.
- Error index: especifica el índice de un error. Si ocurre una excepción, se especifica información de la causa de la excepción.
- Variable Bindings: especifica la lista de nombres de variables y valores correspondientes a la operación.

2.3.2. SET

El sistema *NMS* puede enviar solicitudes al agente de *SNMP* para configurar se utiliza para modificaciones de valores de información para el control del equipo (Fig. 6).

El agente *SNMP* ejecuta la instrucción correspondiente y especificada en el *MIB*, para luego retornar el resultado al *NMS*.

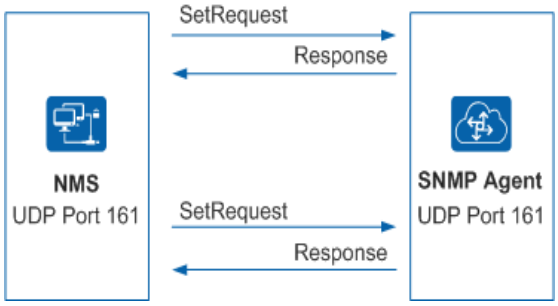


Figura 6: Proceso SET

En la figura 7 se observa el formato de los paquetes de la operación *SET SNMP*

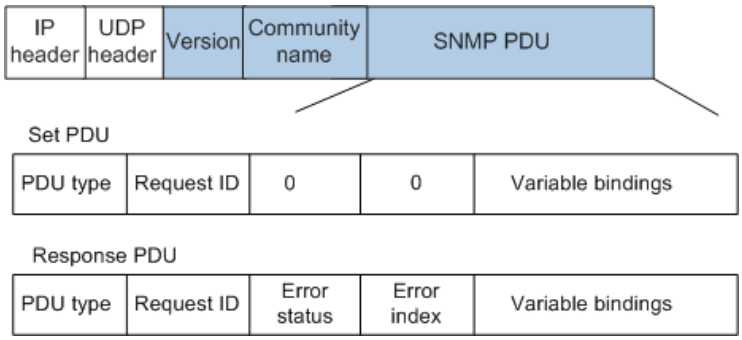


Figura 7: Contenido del paquete SET

A continuación se detalla el paquete en cuestión

- Versión: especifica la versión de *SNMP* a utilizar. El valor 0 es para la versión 1 y 1 para la versión 2c.

- Community name: especifica la autenticación entre el agente *SNMP* y el *NMS*. Este valor es de tipo texto.
- Request ID: especifica un único ID de cada solicitud realizada. Se utiliza para la correspondencia solicitud - respuesta.
- Error status: especifica el estado de un error que ocurre en el procesamiento de la solicitud.
- Error index: especifica el índice de un error. Si ocurre una excepción, se especifica información de la causa de la excepción.
- Variable Bindings: especifica la lista de nombres de variables y valores correspondientes a la operación.

2.3.3. TRAP

Los *TRAP* son mensajes de notificación que envía por el agente *SNMP* hacia el sistema *NMS* para informar alarmas o eventos generados por el equipo. Hay dos tipos de esta operación: *TRAP* e *INFORM* (Fig. 8). La diferencia entre ellos es la existencia de respuesta del sistema *NMS*.

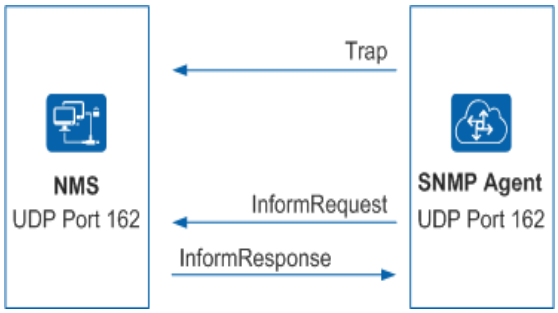


Figura 8: Proceso TRAP

En la figura 9 se observa el formato de los paquetes de la operación *GET SNMP*

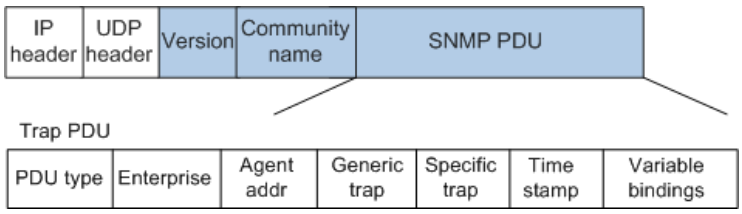


Figura 9: Contenido del paquete TRAP

A continuación, se detalla el paquete utilizado para esta operación

- Versión: especifica la versión de *SNMP* a utilizar. El valor 0 es para la versión 1 y 1 para la versión 2c.
- Community name: especifica la autenticación entre el agente *SNMP* y el *NMS*.Este valor es de tipo texto.

- Enterprise: especifica el tipo de *TRAP* que genera e equipo.
- Generic trap: especifica un tipo de *TRAP* común, incluyendo coldStart, warmStart, linkDown, linkUp, authenticationFailure, egpNeighborLoss y enterpriseSpecific.
- Specific trap: especifica el *TRAP* privado con su correspondiente información.
- Time stamp: especifica el tiempo transcurrido entre el tiempo cuando la entidad de red es reiniciada y el tiempo cuando el mensaje de *TRAP* es generado.
- Variable Bindings: especifica la lista de nombres de variables y valores correspondientes a la operación.

2.4. Puertos utilizados

El protocolo *SNMP* utiliza, comúnmente, el puerto de tipo UDP. El mismo puede ser por default:

- Puerto 161: este puerto es utilizado por el *NMS* para las operaciones de solicitud Get, GetNext, GetBulk, y Set. Sobre este mismo puerto responde el agente SNMP a esas operaciones.
- Puerto 162: este puerto es utilizado por el agente *SNMP* para enviar los *TRAPs* o mensajes con información hacia el *NMS*. Este puerto es configurable, este debe ser configurado en ambos extremos de la comunicación.

2.5. MIB

Management Information Base , contiene las variables que maneja y permite el equipo. El archivo MIB define los atributos manejados por el equipo, incluyendo el nombre y estado, entre otros. El archivo MIB contiene una estructura que mantiene el almacenamiento de la información conforme a un árbol, con nodos que determinan un único objeto e identificable atributo. Esto permite tener una interfaz entre la cola de mensajes del NMS y las variables del equipo.

2.5.1. IANA

Existen MIBs estándares y privadas controladas por la IANA, que juega un papel importante en la asignación de valores y registros específicos. La IANA (Internet Assigned Numbers Authority) administra el registro de números y parámetros asignados para protocolos de Internet.

Los registros MIB en la IANA se organizan de acuerdo a las normas y estándares del SNMP. Algunos de los registros más importantes y comunes que se encuentran en la IANA en relación al SNMP incluyen:

- Internet Standard MIBs: Estas son MIBs que están definidas como estándares en la serie de documentos STD del IETF. Por ejemplo, la MIB-2 (también conocida como la MIB de Internet) es una de las MIBs más conocidas y se encuentra registrada en la IANA.
- Private Enterprise MIBs: Las organizaciones o empresas pueden registrar sus propias MIBs privadas en la IANA. Cada empresa recibe un número de registro de empresa (OID - Object Identifier) que se utiliza para identificar sus propias MIBs.
- Experimental MIBs: A veces, se registran MIBs de manera experimental para propósitos de desarrollo y pruebas.
- Reservación de números: La IANA también maneja la reserva de números para nuevos MIBs o para extensiones de MIBs existentes.

2.5.2. PEM

Particularmente, en los PEMs (Private Enterprise Numbers) los números son parte de la jerarquía de números de objetos (OIDs) y se utilizan para garantizar que las MIBs desarrolladas por una organización específica sean únicas y no entren en conflicto con otras MIBs. Esto permite a las organizaciones definir y administrar información de gestión específica sin tener que compartirla con el público en general. Cada empresa u organización que desarrolla MIBs privadas recibe un número de PEM único. Esto garantiza que las MIBs privadas de diferentes organizaciones tengan identificadores únicos y no se superpongan.

El número de PEM se utiliza como parte de los OIDs que identifican los objetos en las MIB de una organización. Por lo tanto, un OID típico para un objeto en una MIB privada se vería así: 1.3.6.1.4.1.XXXX.1.2, donde "XXXX.[es] el número de PEM asignado a la organización.

3. Ingeniería de la solución

3.1. Análisis del mercado

Dentro del mercado nacional no se ha encontrado algún producto/servicio que solvente la necesidad que encontramos, como así dispositivos que permitan utilizar distintos sensores analógicos y digitales dentro de su red y monitorizarlos. Por otra parte, se realiza una búsqueda a nivel internacional y se encuentran distintas opciones las cuales se destacan las siguientes descritas a continuación.

- Uptrends

  Empresa que se enfoca en desarrollar herramientas de monitoreo de servidores y websites. Dentro de sus servicios se encuentra:

  Synthetics: monitorea distintas características de servicios en línea.

  Real User Monitoring: monitorea la experiencia del usuario, los accesos que realizan y toda la información asociada, por ejemplo el trafico.

  Internal Server Monitoring: monitorea servidores y equipos de networking de la red.

- Cloud Radar Permite monitorear toda la infraestructura IT e incorpora alertas. Incluye servidores Windows y Linux,intranets, websites y cualquier equipo de red.

3.2. Factibilidad

El proyecto se compone de 3 partes esenciales las cuales se complementan para otorgar un servicio de monitoreo que abarca a distintos tipos de dispositivos conectados a una red. Las 3 partes son el sistema NMS, la aplicación de celular y finalmente el dispositivo electrónico que nos permitirá incluir sensores de ambiente.

El valor agregado se centra en otorgar datos en tiempo real para explotarlos de forma visual e intuitiva para los usuarios finales. Se incluye la emisión de alarmas para cualquier tipo de evento que suceda.

Para la elección de las tecnologías a utilizar como así el marco contextual y alcance se realiza el estudio de distintas propuestas similares dentro del mercado y al mismo tiempo dando espacio al usuario final para comprender aún más las necesidades diarias buscando mejorar la calidad de sus actividades.

3.2.1. Sistema NMS

Para la elección del sistema de monitoreo se identifican puntos necesarios y críticos que impactan en el objetivo final. Estos puntos vienen determinados por las funcionalidades a cubrir por el proyecto en sí. Identificamos que son los siguientes:

- interfaz gráfica: define el valor entregado, es determinante para el usuario final.
- configuración: el acceso a la configuración del sistema permite definir la facilidad de uso de la solución, como así también agregar nuevas funcionalidades.
- plantillas: la inclusión de distintos equipos mediante plantillas define el alcance de los equipos a soportar por la solución.
- precio: un factor importante en cuanto a la limitación de versiones del sistemas, plug-ins considerados, y otras variantes del mismo estilo. Así mismo infiere en el precio final y costos extras que podrían concluir en no ser competitivos en el mercado local.

Dentro de la investigación llevada a cabo se encuentra que los sistemas de monitoreo dentro del ámbito de telecomunicaciones son Zabbix, Nagios, Pandora FMS, Open NMS, PRTG, Cacti, entre otros. Dentro ellos se realizó la primera selección en base al uso, casos de aplicación, y comunidad se determinó por estudiar las 3 grandes soluciones los cuales son Zabbix, Nagios y Pandora FMS.

Se encuentra que Zabbix otorga la posibilidad de monitorizar equipos mediante SNMP, JMX, IPMI y expande su capacidad mediante el uso de su agente disponible para varios Sistemas Operativos. Permite configurar alarmas, gráficos y scripts programables en función de los datos obtenidos. Considerando que los clientes solicitan tener en una pantalla una visión general del estado de todos sus equipos, descartamos el uso de PandoraFMS debido a que este posee la función de mapa de red en la versión de pago. Respecto a su principal competidor, Nagios, si bien la versión gratuita de este posee todas las funciones que se requieren, detalles como la interfaz gráfica, facilidad de configuración o la incorporación de plantillas, hacen que Zabbix sea simplemente la mejor opción.

A su vez, Zabbix ofrece una interfaz API , la cual nos permite acceder a los datos históricos. Lo cual nos permite crear nuevas aplicaciones para trabajar con Zabbix, automatizar tareas rutinarias y más. El API de Zabbix es un API basado en web y está integrado como parte de la interfaz de la web. Usa el protocolo JSON-RCP , conjunto de métodos separados y solicitudes/respuestas codificados usando el formato JSON.

Por otra parte, Zabbix también ofrece un proceso llamado Zabbix Proxy el cual puede recopilar datos de monitoreo de uno o más dispositivos monitorizados y enviar la información al servidor Zabbix central, esencialmente trabajando en nombre del servidor. Todos los datos recopilados son almacenado en el disco localmente y luego transferidos al servidor Zabbix al que el proxy pertenece. La implementación de un proxy es beneficioso ya que nos permite distribuir la carga del servidor central Zabbix. Se destaca que este tipo de proxy solo recopilan y envían datos.

Un proxy Zabbix es la solución ideal para el monitoreo centralizado de ubicaciones remotas, sucursales y redes sin administradores locales.

	Zabbix	PandoraFMS	Nagios
Precio	Gratis	Paga	Gratis
Interfaz Gráfica	Ya vienen por defecto	Ya vienen por defecto	Hay que instalar Plug-in
Configuración	Vía interfaz Web	Vía interfaz web	Principalmente con archivos de texto
Plantillas	Gran cantidad	Limitados	No

Cuadro 1: Comparación de soluciones

3.2.2. Aplicación

Para abordar el análisis de la viabilidad de nuestro producto, nos centramos en un aspecto distintivo fundamental: la interfaz de usuario. Reconocemos la importancia de un diseño que favorezca la claridad y la intuición, contribuyendo así a una experiencia del usuario mejorada. En contraste con ([**Tabbix**]), la aplicación móvil más difundida, nuestra propuesta se distingue por ofrecer una interfaz fluida y visualmente atractiva. Este enfoque facilita la navegación a través de la abundancia de información proporcionada por Zabbix.

Con respecto a la representación visual de datos, que es una faceta esencial para comprender de manera eficiente conjuntos complejos de información, definimos que nuestra aplicación no se limita a la presentación de gráficos; sino que transformará datos en visualizaciones significativas que son estéticamente agradables y altamente informativas. Al superar las limitaciones observadas en aplicaciones existentes, nuestra propuesta no solo busca satisfacer las expectativas del usuario, sino también mejorar sustancialmente la eficacia en la interpretación de información compleja.

3.2.3. Concentrador de sensores

Dentro del mercado local, no se ha encontrado productos similares que cubran la necesidad demandada por nuestra solución. Esta se centra en la integración y traducción de datos provenientes de sensores de ambiente hacia un sistema de monitorización.
Sin embargo, si existen soluciones dentro del mercado las cuales pueden por una parte estar enfocadas en parámetros específicos y por otra parte se encuentran soluciones con una gran extension de aplicaciones. Sin embargo, estas soluciones presentan ciertas desventajas, principalmente en el precio del producto, la dependencia del desarrollo del producto y el soporte técnico que puedan ofrecer.
A continuación, se exponen las soluciones más sobresalientes encontradas en el mercado:

- ([**Poseidon**]) de Insumos NET, Chile: Esta solución admite una cantidad determinada de sensores y se complementa con un sistema capaz de emitir alarmas, además tiene la capacidad de conectarse a redes IoT. Aunque ofrece una amplia funcionalidad, el sistema central es propio de la empresa por lo cual su enfoque específico pueden no satisfacer completamente las necesidades de nuestro proyecto.

- Temperatura de SensMsx, Europa: Esta solución se enfoca principalmente en la medición de temperaturas en distintos ambientes y el envío de datos a un sistema central en la nube propio de la empresa. Si bien ofrece una funcionalidad específica y confiable, su limitación a un único parámetro puede no ser adecuada para aplicaciones más amplias que requieren la monitorización de varios factores ambientales.

Finalmente, utilizar sensores digitales de fácil adquisición para conectarlos al dispositivo electrónico y que el mismo dispositivo se encargue de procesar los valores y crear el paquete correspondiente para su envío al sistema NMS es factible y proporciona una dirección clara para el proyecto ya que permite una implementación más eficiente y económica, aprovechando la disponibilidad de sensores en el mercado y simplificando el proceso de adquisición y transmisión de datos al sistema de gestión de red (NMS). Además, al centrarse en sensores digitales, es posible obtener mediciones más precisas y confiables, lo que contribuirá a la efectividad y utilidad del sistema de monitorización.

3.3. Alcance

3.3.1. Sistema NMS

Sistema central de monitoreo
Implementar un Sistema Central de Monitoreo basado en Zabbix para supervisar y gestionar dispositivos de red, centrales telefónica, sensores de temperatura y sensores de estado de puertas en un entorno de datacenter. El sistema central soporta distintos tipos de equipos que abarcan aquellos que soporten el protocolo *SNMP* y/o que respondan a paquetes *PING*. Como objetivos específicos se tiene:

- Configurar y desplegar un servidor Zabbix para gestionar y monitorear distintos equipos de red mediante SNMP (get/set y traps) y ping.
- Desarrollar scripts personalizados en Zabbix para la generación de alarmas y notificaciones específicas para los usuarios con la aplicación Zabbix instalada.

Zabbix Proxy
Implementar un Proxy Zabbix en una Raspberry Pi 4 para recolectar datos de dispositivos remotos y enviarlos al Servidor Zabbix principal. El mismo establece un canal hacia el servidor *Zabbix* con un protocolo propio. Parte de sus funciones es realizar la traducción de los OID del cliente con los *MIB*s, aligerando el procesamiento en el servidor *Zabbix* como así también el ancho de banda de cada de la sede correspondiente en la que se encuentre este Proxy. El Proxy *Zabbix* es una versión simplificada servidor *Zabbix* sin interfaz gráfica pero soportando los mismos tipos de dispositivos que el servidor Zabbix: Switches, Firewalls, Centrales Telefónicas, Concentrador de Sensores, etc.

3.3.2. Aplicación

La aplicación contará con la capacidad de:

- Visualizar información detallada de hosts y eventos.
- Generar tableros personalizados que contengan gráficas interactivas para el ítem que se considere esencial.
- Servicio de notificaciones push en tiempo real en el caso de que se genere algún evento.

3.3.3. Concentrador de sensores

Diseñar e implementar una placa basada en ESP32 para la recolección de datos de temperatura y estados de puertas en un datacenter y enviarlos al Zabbix Proxy. Como objetivos específicos se tiene:

- Diseñar una placa utilizando como microcontrolador un ESP32 que permita la lectura de un sensor de temperatura, la lectura de sensores de apertura de puerta magnéticos como así también una conexión fiable entre la placa y el Zabbix Proxy mediante conexión Ethernet.
- Desarrollar el firmware necesario para la placa que permita la lectura de datos de sensores de temperatura y estados de puertas a través de SNMP y HTTPS.

3.4. Matriz FODA

La implementación de un proyecto de monitorización de datacenter emerge como una estrategia esencial para garantizar la operación óptima de los sistemas y la anticipación proactiva de posibles desafíos.

En este contexto, la Matriz FODA se presenta como una herramienta analítica integral que permite evaluar tanto los factores internos como externos que pueden influir en el éxito y la eficacia de nuestro proyecto. La Matriz FODA, que se basa en la identificación de Fortalezas, Oportunidades, Debilidades y Amenazas, proporciona una perspectiva estratégica valiosa para la toma de decisiones informada. A continuación se especifica cada una de las partes.

3.4.0.1. Fortalezas

Contacto directo como referente tecnológico
El proyecto permite aplicar la tecnología para el desarrollo de una solución dentro del mercado local. Lo que nos permite tener un contacto directo y cercano con los usuarios finales. Este acceso privilegiado nos permite entender a fondo sus necesidades y expectativas, permitiéndonos adaptar y mejorar continuamente nuestra oferta de acuerdo con las demandas del mercado.

Respaldo de tecnologías open source
La elección de basar nuestra solución en tecnologías open source constituye una fortaleza estratégica. Este enfoque no solo garantiza la transparencia y flexibilidad de nuestra solución, sino que también nos sitúa en un contexto colaborativo

y enriquecedor. Contamos con el respaldo activo de una comunidad de especialistas apasionados que, con su conocimiento y experiencia, potencian el desarrollo y crecimiento futuro de nuestro producto/servicio. Esta sinergia entre nuestra visión y la contribución de la comunidad crea una plataforma robusta para la innovación continua y la adaptabilidad a las cambiantes necesidades del mercado, como así tener la posibilidad de realizar alianzas por un propósito en común.

3.4.0.2. Oportunidades

Innovación local con baja inversión y soporte técnico
La incorporación de un producto innovador y una aplicación para el celular en el mercado local es una fortaleza distintiva de nuestro proyecto. Destacamos no solo por la calidad y eficacia de nuestra solución, sino también por la capacidad de introducir innovaciones relevantes con inversiones moderadas. Esta estrategia no solo diversifica el mercado local con propuestas tecnológicas avanzadas, sino que también demuestra nuestra capacidad para ofrecer soluciones accesibles y de alto valor. Además, al contar con un soporte técnico integral, garantizamos la continuidad y fiabilidad de nuestro producto, generando confianza en nuestros usuarios.

Despliegue ágil y eficiente
La rapidez con la que se despliega nuestra solución representa otra oportunidad estratégica. La agilidad en la implementación no solo satisface la urgencia de las organizaciones en busca de soluciones de monitorización de datacenter, sino que también nos confiere una ventaja competitiva. Este aspecto resalta nuestra capacidad para adaptarnos a las demandas del mercado y proporcionar resultados tangibles en un corto período. La eficiencia en el despliegue no solo optimiza la experiencia del usuario, sino que también refuerza nuestra posición como proveedores de soluciones ágiles y orientadas a resultados.

3.4.0.3. Debilidades

Limitaciones en el Equipo de Desarrollo
Una de las principales debilidades que enfrentamos se relaciona con la escasez de desarrolladores de aplicaciones en nuestro equipo. La limitada fuerza de trabajo en este aspecto puede afectar la velocidad y amplitud con la que podemos mejorar nuestra solución.

Dependencia de Componentes Electrónicos Importados
La dependencia de componentes electrónicos importados representa una vulnerabilidad significativa en nuestro proyecto. Los tiempos de entrega desfavorables pueden generar retrasos en la producción y entrega de nuestra solución, afectando la satisfacción del cliente y nuestras promesas de entrega oportuna. Para abordar esta debilidad, debemos explorar fuentes alternativas de suministro, evaluar posibles proveedores locales y considerar estrategias de almacenamiento

para mitigar los impactos negativos de los tiempos de entrega.

3.4.0.4. Amenazas

Amortización de Inversión a Largo Plazo
Una amenaza crítica que enfrentamos es la posibilidad de una amortización de inversión a largo plazo. Dada la naturaleza tecnológica y la evolución constante del mercado, existe el riesgo de que la vida útil de nuestra solución pueda ser más corta de lo anticipado. Esto podría resultar en desafíos significativos para recuperar la inversión inicial y limitar la capacidad de financiar futuras actualizaciones o desarrollos. Para mitigar esta amenaza, es esencial adoptar estrategias flexibles y estar preparados para adaptarnos rápidamente a cambios en el panorama tecnológico.

Contexto socio-económico Adverso
El entorno socio-económico adverso representa otra amenaza importante para nuestro proyecto. Fluctuaciones económicas, crisis financieras u otros eventos imprevistos pueden tener un impacto negativo en la demanda del mercado y en la capacidad de las organizaciones para invertir en soluciones de monitorización de datacenter. Para abordar esta amenaza, debemos ser proactivos en la diversificación de nuestro mercado objetivo, explorar modelos de precios flexibles y estar preparados para ajustar estrategias de marketing y ventas en respuesta a cambios en el contexto socio-económico.

3.5. Gestión de riesgo

3.5.1. Riesgos Técnicos

Evaluamos posibles desafíos en el desarrollo y ensamble de la solución, en particular, en relación con el concentrador de sensores. El diseño del PCB de la solución es una etapa crítica, especialmente por la integración del módulo de conectividad Ethernet, fundamental para la funcionalidad del dispositivo. Dado el tiempo de entrega prolongado de la empresa extranjera, que puede alcanzar hasta seis meses, esta etapa se considera de dificultad elevada.

La programación del sistema que administrará el concentrador de sensores también es un punto crítico, con posibles resultados no deseados como reinicios o cuelgues del sistema.

3.5.2. Riesgos Operativos

Consideramos aspectos relacionados con la conectividad de los equipos a monitorizar, destacando la necesidad de implementar y probar una maqueta con distintos equipos como una fase crucial en el desarrollo.

Además, la conexión a la base de datos del servidor Zabbix se presenta como un requisito esencial para el desarrollo, prueba y disponibilización de la aplicación Zabbix, un componente diferencial en nuestra solución.

3.5.3. Priorización y mitigación

Clasificamos los riesgos según su impacto potencial y probabilidad de ocurrencia, enfocándonos en mitigar aquellos más críticos.
Principal riesgo
El diseño del PCB se identifica como el riesgo principal, dada su importancia para el avance y finalización del proyecto. Para mitigar este riesgo, nos aseguramos de buscar información y diseños de referencia, disminuyendo la probabilidad de error. Ante posibles retrasos, establecemos acuerdos con proveedores alternativos, priorizando tiempos de entrega sobre costos.
Riesgos secundarios
Para abordar la programación del sistema, llevamos a cabo pruebas exhaustivas para detectar respuestas inesperadas, reduciendo la probabilidad de errores en la producción.
En relación a la conectividad con los equipos del cliente, contamos con un técnico interno en la empresa, capacitado y capaz de resolver cualquier eventualidad en el sitio.

3.6. Planificación

Se realiza un análisis del desarrollo que hay que llevar a cabo para finalizar el proyecto. De acuerdo a las etapas encontradas se realiza un análisis de riesgo del proyecto para identificar los hitos determinantes.

3.6.1. EDT

Se plantean 3 secciones importantes, los cuales se dividen en los siguientes puntos que serán explayados a continuación:

3.6.1.1. Zabbix

Server: Instalación, puesta en marcha y configuración de Firewall HTTP

- **Descripción**: Realizar la instalación inicial y configuración del Firewall HTTP en el servidor NMS.
- **Entregable Esperado**: Configuración documentada del Firewall HTTP.
- **Fecha de Inicio - Finalización**: 28/06/2021 - 16/07/2021.
- **Responsable/s**: Alexis.

Server: Investigación de MIBs

- **Descripción**: Investigar cómo cargarlos y usarlos en Linux.
- **Entregable Esperado**: Agregado de MIBs de equipos de prueba.
- **Fecha de Inicio - Finalización**: 16/07/2021 - 21/07/2021.
- **Responsable/s**: José.

Server: Agregado de ítems de SNMP

- **Descripción**: Agregar ítems de SNMP según los MIBs identificados.
- **Entregable Esperado**: Lista de ítems agregados.
- **Fecha de Inicio - Finalización**: 21/07/2021 - 26/07/2021.
- **Responsable/s**: José.

Server: Generación y testeo de alarmas

- **Descripción**: Generar y probar alarmas de prueba en el sistema NMS, con notificaciones por correo electrónico.
- **Entregable Esperado**: Informe de resultados.
- **Fecha de Inicio - Finalización**: 26/07/2021 - 29/07/2021.
- **Responsable/s**: José.

Server: Investigación gráficas de Zabbix

- **Descripción**: Investigar los tipos de gráficas disponibles en Zabbix para mejorar la visualización de datos.
- **Entregable Esperado**: Agregado de gráficas al servidor.
- **Fecha de Inicio - Finalización**: 29/07/2021 - 02/08/2021.
- **Responsable/s**: José.

Server: Generación de mapa para la red de prueba

- **Descripción**: Generar un mapa visual para representar la red de prueba.
- **Entregable Esperado**: Mapa de red generado.
- **Fecha de Inicio - Finalización**: 02/08/2021 - 12/08/2021.
- **Responsable/s**: José.

Server: Programación de script multiclientes

- **Descripción**: Programar un script en Python para notificaciones Push a través de varios clientes.
- **Entregable Esperado**: Código fuente del script Python.
- **Fecha de Inicio - Finalización**: 12/08/2021 - 19/08/2021.
- **Responsable/s**: José, Alexis.

Proxy: Instalación y configuración de proxy en la red de prueba

- **Descripción**: Realizar la instalación y configuración inicial del proxy en la red de prueba.
- **Entregable Esperado**: Habilitación del Firewall para recibir conexiones entrantes.
- **Fecha de Inicio - Finalización**: 19/08/2021 - 22/08/2021.
- **Responsable/s**: José, Alexis.

Proxy: Instalación de agentes en la red de prueba

- **Descripción**: Instalar agentes en la red de prueba para habilitar la monitorización en Zabbix, incluyendo alarmas y gráficos.
- **Entregable Esperado**: Agentes instalados y configurados con éxito.
- **Fecha de Inicio - Finalización**: 22/08/2021 - 01/09/2021.
- **Responsable/s**: José, Alexis.

Proxy: Testeo de funcionalidad de proxy a través del Firewall

- **Descripción**: Realizar pruebas de funcionalidad del proxy a través del Firewall, incluyendo pruebas de caída de enlace y ajustes de configuración.
- **Entregable Esperado**: Resultado de pruebas satisfactorio.
- **Fecha de Inicio - Finalización**: 01/09/2021 - 11/09/2021.
- **Responsable/s**: José, Alexis.

3.6.1.2. Adquisición de la información

Ethernet Chip: Investigación y adquisición

- **Descripción**: Realizar la investigación correspondiente. En base a lo investigado, adquirir el producto evaluando las diferentes opciones en caso de que no pueda conseguirse debido a las situaciones de público conocimiento.
- **Entregable Esperado**: Recepción del Ethernet Chip.

- **Fecha de Inicio - Finalización**: 12/06/2021 - 25/08/2021.
- **Responsable/s**: Alexis.

Ethernet Board: Adquisición

- **Descripción**: Realizar la compra de un Arduino con Shield de Ethernet para hacer pruebas con las librerías de SNMP
- **Entregable Esperado**: Recepción del Ethernet Board.
- **Fecha de Inicio - Finalización**: 15/06/2021 - 25/06/2021.
- **Responsable/s**: Daiana.

Adquisición de sensores y placa de pruebas

- **Descripción**: Realizar la compra de los sensores y placa de pruebas a utilizar.
- **Entregable Esperado**: Recepción de los sensores y placa de pruebas.
- **Fecha de Inicio - Finalización**: 19/06/2021 - 03/07/2021.
- **Responsable/s**: Daiana.

Sensores: Programación de drivers

- **Descripción**: Programar los drivers a ser utilizados por los sensores.
- **Entregable Esperado**: Código fuente de los drivers programados.
- **Fecha de Inicio - Finalización**: 03/07/2021 - 24/07/2021.
- **Responsable/s**: Daiana.

Sensores: Investigación de librería SNMP

- **Descripción**: Realizar la investigación correspondiente.
- **Entregable Esperado**: Definición de librería SNMP a utilizar.
- **Fecha de Inicio - Finalización**: 03/07/2021 - 18/07/2021.
- **Responsable/s**: Daiana.

Sensores: Conexión con la placa

- **Descripción**: Conectar los sensores a la placa de pruebas.
- **Entregable Esperado**: Conexión satisfactoria.
- **Fecha de Inicio - Finalización**: 24/07/2021 - 03/08/2021.
- **Responsable/s**: Daiana.

Sensores: Integración con SNMP

- **Descripción**: Integrar código con la librería SNMP investigada anteriormente.
- **Entregable Esperado**: Código fuente actualizado.
- **Fecha de Inicio - Finalización**: 24/07/2021 - 08/08/2021.
- **Responsable/s**: Daiana.

Sensores: Prueba de prototipo para PMV

- **Descripción**: Probar el código desarrollado en la placa de pruebas como para cumplir con el Producto Mínimo Viable. Aplicar solucion de bugs.
- **Entregable Esperado**: Prueba de prototipo satisfactoria.
- **Fecha de Inicio - Finalización**: 01/08/2021 - 31/08/2021.
- **Responsable/s**: Daiana.

Sensores: Prototipo de ESP32 e investigación de Wifi-Ethernet

- **Descripción**: Desarrollar prototipo compatible con ESP32. Realizar la investigación de Wifi-Ethernet para aplicarlo al prototipo.
- **Entregable Esperado**: Prototipo satisfactorio.
- **Fecha de Inicio - Finalización**: 12/06/2021 - 27/07/2021.
- **Responsable/s**: Alexis.

Sensores: Construcción de placa en KiCad

- **Descripción**: Realizar el ruteo correspondiente a la placa utilizando el software KiCad.
- **Entregable Esperado**: PCB del circuito completamente ruteado.
- **Fecha de Inicio - Finalización**: 27/07/2021 - 11/08/2021.
- **Responsable/s**: Alexis.

Agente Zabbix: Investigación, programación y configuración

- **Descripción**: Realizar la investigación del agente apuntando a ampliar las capacidades del mismo. Programar el agente y configurarlo.
- **Entregable Esperado**: Configuración satisfactoria del agente.
- **Fecha de Inicio - Finalización**: 11/08/2021 - 14/08/2021.
- **Responsable/s**: Alexis.

Agente Zabbix: Instalación de agente en PCs de prueba

- **Descripción**: Realizar la instalación del agente de Zabbix en equipos de prueba.
- **Entregable Esperado**: Instalación satisfactoria del agente.
- **Fecha de Inicio - Finalización**: 14/08/2021 - 24/08/2021.
- **Responsable/s**: Alexis.

Equipos de red: Búsqueda de MIBs para equipos de prueba

- **Descripción**: Realizar la búsqueda de MIBs y módulos compatibles para instalarlo en los equipos de prueba.
- **Entregable Esperado**: Instalación satisfactoria de los módulos.
- **Fecha de Inicio - Finalización**: 14/08/2021 - 24/08/2021.
- **Responsable/s**: Alexis.

Equipos de red: Habilitación de SNMP y configuración de comunidad

- **Descripción**: Habilitar el módulo SNMP. Habilitar la configuración de comunidad en los equipos de prueba.
- **Entregable Esperado**: Habilitación satisfactoria.
- **Fecha de Inicio - Finalización**: 24/08/2021 - 29/08/2021.
- **Responsable/s**: Alexis.

3.6.1.3. Aplicación Móvil para Android

Creación del esqueleto de la aplicación

- **Descripción**: Realizar el diseño de pantallas, orden de aparición de las mismas, botones y funcionalidades de la aplicación.
- **Entregable Esperado**: Esqueleto finalizado.
- **Fecha de Inicio - Finalización**: 27/06/2021 - 18/07/2021.
- **Responsable/s**: Rubén.

Generación de gráficas

- **Descripción**: Realizar la investigación y las pruebas correspondientes de la librería Any Chart para la generación de las gráficas a utilizar.
- **Entregable Esperado**: Pruebas satisfactorias.
- **Fecha de Inicio - Finalización**: 18/07/2021 - 02/08/2021.
- **Responsable/s**: Rubén.

Lectura de JSON desde URL

- **Descripción**: Realizar las pruebas correspondientes para la interpretación correcta de la información obtenida en formato JSON.
- **Entregable Esperado**: Pruebas satisfactorias.
- **Fecha de Inicio - Finalización**: 02/08/2021 - 17/08/2021.
- **Responsable/s**: Rubén.

Puesta a punto

- **Descripción**: Realizar la corrección de errores encontrados, ajustar el formato y terminar el ajuste fino de la aplicación
- **Entregable Esperado**: Aplicación finalizada.
- **Fecha de Inicio - Finalización**: 07/09/2021 - 15/10/2021.
- **Responsable/s**: Rubén.

3.6.1.4. Tareas entre módulos

Adquisición de la información y Zabbix: Conexión

- **Descripción**: Realizar la conexión entre los módulos de adquisición de información probados previamente junto con el servidor Zabbix.
- **Entregable Esperado**: Conexión satisfactoria.
- **Fecha de Inicio - Finalización**: 03/07/2021 - 18/07/2021.
- **Responsable/s**: Alexis, Daiana.

Zabbix y Aplicación móvil: Manejo de base de datos

- **Descripción**: Realizar la conexión entre la aplicación y el servidor para poder obtener la información de los componentes mediante request HTTP.
- **Entregable Esperado**: Conexión satisfactoria.
- **Fecha de Inicio - Finalización**: 13/08/2021 - 28/08/2021.
- **Responsable/s**: José, Rubén.

Zabbix y Aplicación móvil: Manejo de notificaciones push

- **Descripción**: Realizar la conexión entre la aplicación y el servidor para adaptar los eventos con el objetivo de ser mostrados al usuario como notificaciones push del sistema Firebase
- **Entregable Esperado**: Conexión satisfactoria.
- **Fecha de Inicio - Finalización**: 02/09/2021 - 17/09/2021.

- **Responsable/s**: José, Rubén.

3.6.2. Diagrama de Gantt

La descripción del diagrama que visualiza las tareas nombradas anteriormente se encuentra adjunto en la sección de apéndices de este informe.

3.7. Gestión de calidad

La gestión de calidad es un pilar esencial en el desarrollo y ejecución exitosa del proyecto. Dentro de la monitorización de datacenter, hemos implementado un enfoque integral para garantizar el correcto funcionamiento de cada fase. Cada componente que integra la solución cumple con su correspondiente estándar de calidad relevante para la aplicación acotada para el presente proyecto. Esos detalles buscan que nuestro proyecto cumpla con las expectativas y requisitos tanto internos como externos. A continuación se detallan y se exponen los estándares buscados a lo largo del proyecto.

3.7.1. Sistema NMS

El sistema de monitoreo desempeña un papel central en la efectividad y utilidad de la solución de monitorización de datacenter. Dentro del sistema Zabbix se define un esquema de permisos de accesos. El administrador es quien se encarga de dicho esquema y del servicio de monitorización en sí, el cual no solo será correspondiente al cliente sino incluirá a un técnico del proyecto para la solución de problemas.

Se configura y prueba las suscripciones de los usuarios de la aplicación.

La implementación de las gráficas para monitorización se deja a elección del cliente.

3.7.2. Aplicación

Teniendo en cuenta los principales objetivos de este componente se procede a efectuar pruebas de comunicación de la aplicación hacia la red del servicio de monitorización, establecer un esquema de errores e identificar procedimientos para la solución de dichos problemas.

Para el sistema de notificaciones se realiza una suscripción al servidor central. Esto permite poder ser administrado desde un único punto de entrada.

3.7.3. Concentrador de sensores

El sistema desarrollado define etapas que incluye pruebas unitarias con cada sensor, e integración de la capacidad total de sensores y del sistema completo

para garantizar que cada parte funcione según lo previsto y cumpla con los requisitos especificados.

Por otra parte, físicamente, se desarrolla la plataforma con componentes aptos para el manejo adecuado dentro de las instalaciones de datacenter. Conectores y cables pertinentes para el conexionado y se adiciona una carcasa la cual se modela e imprime en 3D.

3.8. Componentes de la solución

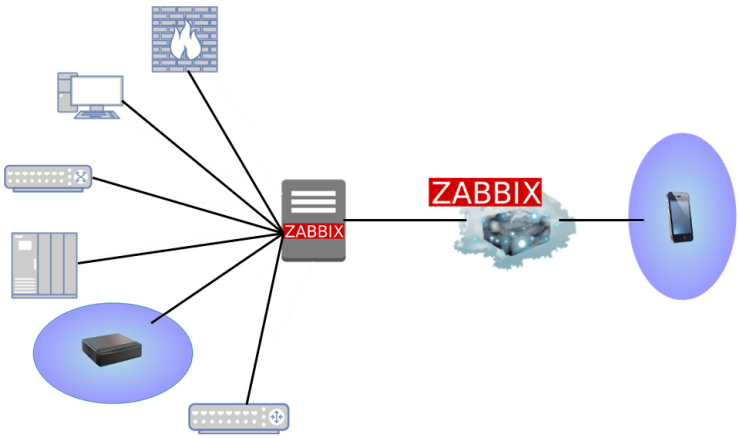


Figura 10: Mapa topológico de solución

Las partes que conforman la solución son:

- Servidor de Zabbix y Proxy de Zabbix
- Concentrador de Sensores
- Aplicación móvil
- Otros equipos de red conectados por protocolo SNMP

La solución se compone de un servidor *Zabbix* en la nube y múltiples servidores Proxy *Zabbix*,cada uno un sitio particular. Esta topología es adecuada para aquellos clientes que no cuentan con despliegues de red interconectados y centralizados, por lo cual ante la caída de uno de los sitios no se pierde conexión con el servidor central como así tampoco con el resto de los sitios.

3.8.1. Sistema NMS

Servidor Zabbix

Acceso a sistema
La interfaz GUI del servidor *Zabbix* presenta una pantalla de Login a través de

la cual accederemos a la plataforma para su monitoreo y gestión.

Monitoreo de equipos y eventos
La sección de monitoreo se realiza a través de gráficas presentadas en un Dashboard estándar (Fig. 11).
La solución otorgada expone los siguientes gráficos inicialmente:

- visualización general de la cantidad de eventos sin resolver
- una cronología de los eventos
- mapas de red global o de cada sitio

Cabe destacar que el Dashboard principal puede modificarse, eliminando e incluso crear nuevas gráficas, de acuerdo a las necesidades puntuales del cliente.

Figura 11: Pantalla de Dashboard

A su vez se disponibiliza una sección de problemas presentada (Fig. 12), en la cual se permite aplicar un filtrado específico y avanzado para buscar eventos determinados de interés del usuario. La solución contempla la identificación de los equipos conectados por *SNMP* al servidor central a través de la sección hosts (Fig. 13). Expone la IP, estado y conteo de eventos de cada equipo.

Para la visualización del mapa de la red se disponibiliza la seccion Mapas (Fig. 14). Esta sección permite visualizar los equipos lógicamente conectados y el estado de los mismos, donde se identifican los eventos que puedan poseer cada equipo a través de colores en el mapa.

Configuración del sistema
Dentro de la sección de configuración (Fig. 15) se realizan las configuraciones que permitirán la monitorización como así sus correspondientes eventos. A continuación se destacan los puntos relevantes para la solución propuesta.

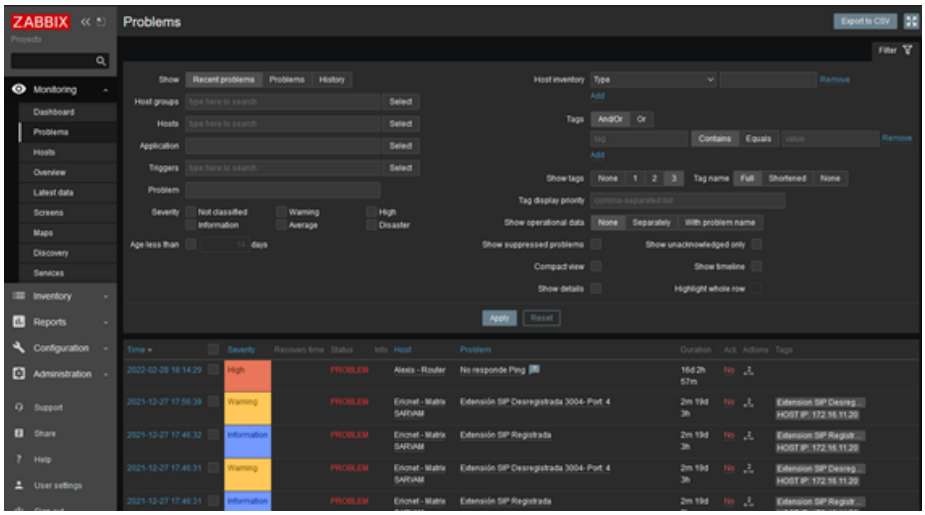


Figura 12: Pantalla de eventos

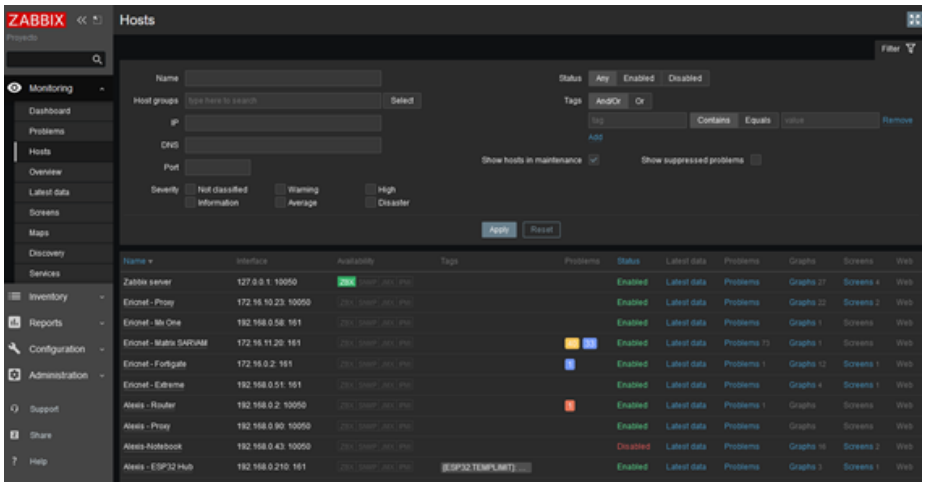


Figura 13: Pantalla de equipos

Posibilidad de agregar los equipos asignados a cada cada Zabbix Proxy.
Agregar plantillas que permitan crear las funciones para los eventos como así también distintos comportamientos del sistema *NMS*.
Acciones que permitirán hacer cambios (paquete tipo *SET* de *SNMP*) en nuestros equipos de acuerdo con las posibilidades de configuración que ofrezca el equipo remoto.

Si bien estás son las opciones de personalización general, existen otras que se podrán utilizar de acuerdo con las exigencias y requerimientos de nuestro cliente.

Dentro de las dificultades que superamos, se destacan las siguientes:

1. Desconocimiento de puesta en funcionamiento del servicio Zabbix:

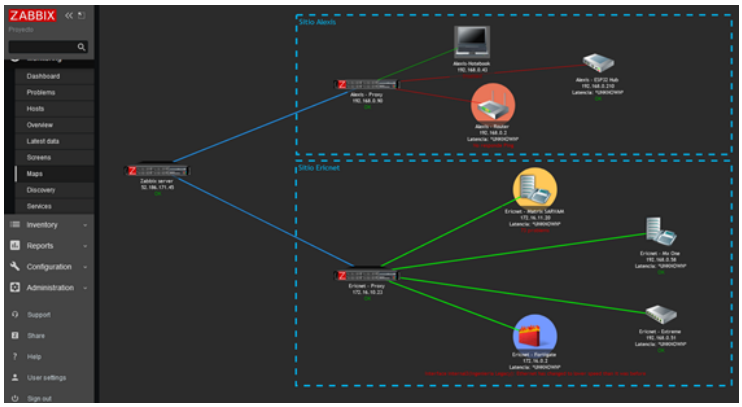


Figura 14: Pantalla de mapas

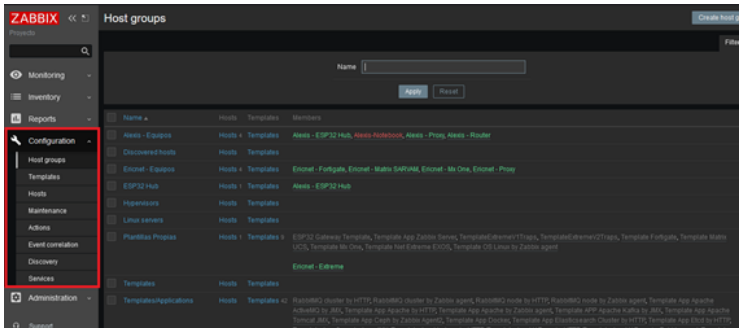


Figura 15: Sección de Configuración

- Investigación del Protocolo SNMP.
- Investigación de la forma de instalación de Servidor Zabbix: Procedimiento recomendado por el autor del software.
- Investigación de la forma de instalación de instalación de Proxy Zabbix: Procedimiento recomendado por el autor del software.

2. Necesidad de una aplicación para la comunicación de Zabbix con las notificaciones de la aplicación Android: Script escrito en Python utilizando la librería FCMNotification para conectarse con el servicio de Firebase Cloud Messaging de Google.

3.8.2. Concentrador de sensores

El concentrador de sensores es un dispositivo desarrollado por los integrantes del presente proyecto con la finalidad de completar la solución con la adición de un componente de valor. Como se ha mencionado anteriormente, el ambiente de [**Datacenter**] representa un espacio con equipos de gran interés que permiten

a la instalación funcionar. El objetivo es tener control de todos ellos pero no se extiende al control del espacio en que se disponen. Un ejemplo claro es una falla de refrigeración o un acceso no autorizado a este espacio.
Por lo tanto, la finalidad del concentrador de sensores es proporcionar la posibilidad de medir temperatura dentro de un rack, de la sala y controlar el estado de las puertas de los racks y la de acceso a la sala misma.
El núcleo de dispositivo está formado por un microcontrolador ESP32 de la marca Espressif, el cual es una solución de conectividad todo en uno, integrada y certificada que proporciona interfaces de conectividad Bluetooth y Wi-Fi, un coprocesador de ultra bajo consumo, un stack TCP/IP integrado y otros periféricos comunes en la industria de microcontroladores. Su elección se debe a que el equipo se encuentra familiarizado con su uso y sus múltiples tipos de conectividad permiten mejorar el proyecto a futuro.
La comunicación de este dispositivo con el servidor o proxy *Zabbix* lo realiza a través del protocolo *SNMP*, con el uso de mensajes *GET*, *TRAP* y *SET*. La comunicación de este dispositivo con el servidor o proxy *Zabbix* lo realiza a través del protocolo *SNMP*, con el uso de mensajes *GET*, *TRAP* y *SET*.
Se desarrolló una interfaz web (Fig. 17) para la configuración inicial de la dirección IP, máscara de subred y la puerta de enlace predeterminada además de la visualización de los estados de los sensores.
Para lograr la comunicación de red se adquirió el circuito integrado LAN8710 a través de Aliexpress (única tienda con stock) el cual es un transceptor de Ethernet que proporciona conectividad física a una red cableada, pero no incluye un stack TCP/IP. Otra opción es el uso del chip W5500 que incluye un TCP/IP completo, pero su uso es innecesario ya que el ESP32 lo tiene integrado.
En la placa se encuentra un conector RJ45 (Fig. 19) para conectarse por red y enlazarse con el servidor o Proxy *Zabbix*. Se observa en el centro de la placa (Fig. 16) el chip de Ethernet LAN8710 con todos los elementos pasivos asociados y en la esquina superior izquierda el microcontrolador ESP32 utilizado.
La placa (Fig. 18) además cuenta con 4 conectores conectados a entradas digitales del ESP32 para sensores de puerta magnéticos y un conector para el sensor de temperatura DS18B20 (Fig. 20). Este sensor cuenta con una resolución de 12 bits, calibración interna sin necesidad de ajustes externos, rango de temperatura de -55 a +125 grados centígrados, bajo consumo y económico en relación con otras soluciones. La variante de este sensor utilizada tiene el chip encapsulado y con un cable el cual se puede extender y ubicar en cualquier punto del rack o la habitación del Datacenter. Para lograr la correcta aislación de la placa con el posible ruido externo, los sensores de puerta están conectados cada uno directamente a un opto-acoplador ACPL217. Además de aislamiento galvánico para ruido, sobretensiones y corrientes parásitas ofrece bajo consumo de energía, robustez y fiabilidad, lo que lo hace adecuado para este proyecto.

En uno de los laterales del dispositivo se encuentra un conector Micro USB (Fig. 21) para alimentar a la fuente switching conformada por un TLV62569 que se encarga de convertir los 5V de entrada a los 3.3V necesarios para los diferentes circuitos integrados.

En el otro lateral se encuentra los pines y el botón para la programación y debug de la misma (Fig. 22).

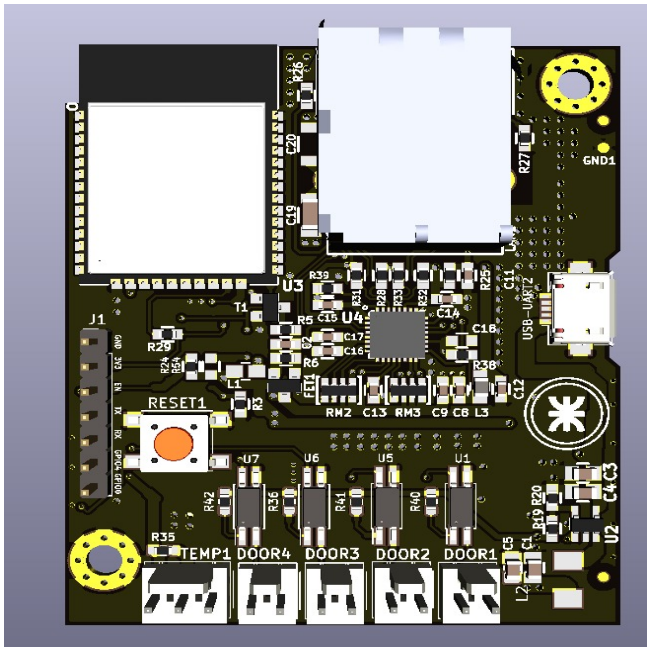


Figura 16: Placa completa.

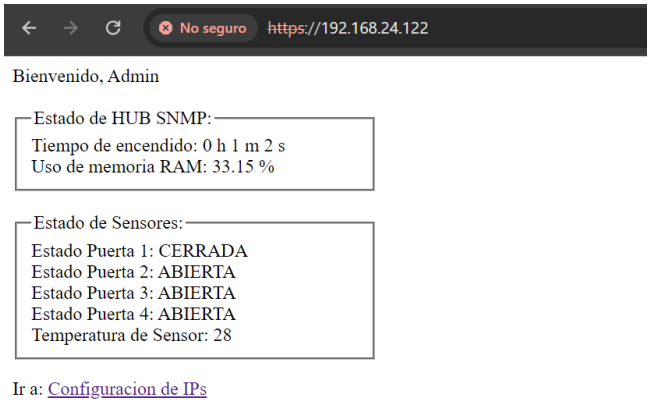


Figura 17: Interfaz web del Concentrador de Sensores.

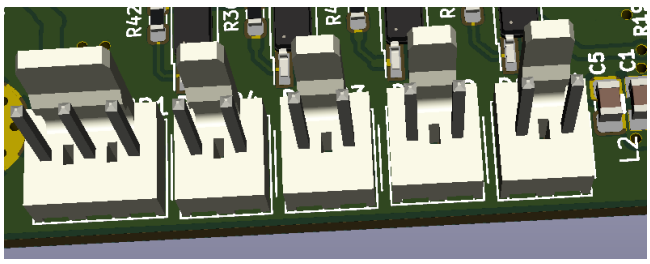


Figura 18: Conectores del Concentrador de Sensores.

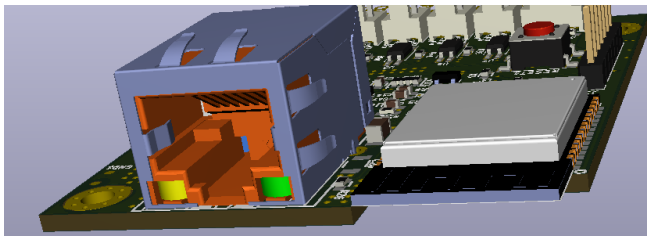


Figura 19: Conector RJ45 para la conexión Ethernet.

Figura 20: Sensor DS18B20.

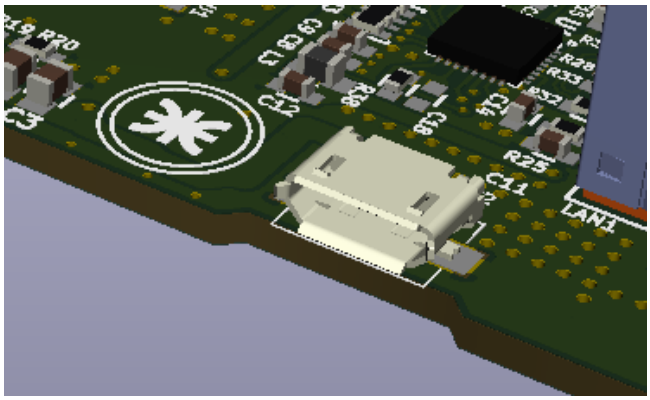


Figura 21: Conector Micro USB para alimentar el dispositivo.

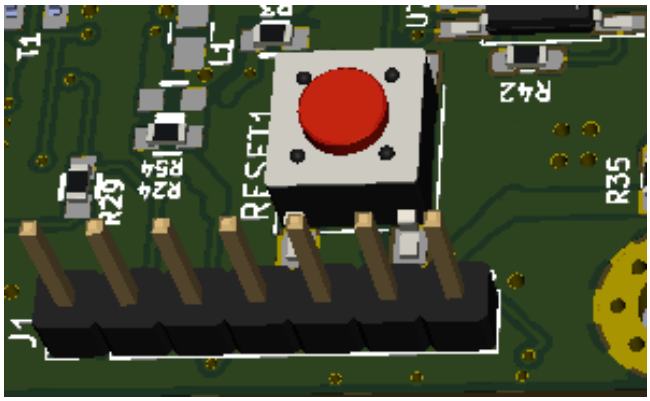


Figura 22: Conectores para la programación y debug de la placa.

En cuanto a la programación del ESP32, el código se encuentra organizado en dos secciones (Fig. 23): *include* que contiene archivos de encabezado .h) que se utilizan para declarar las interfaces de funciones y estructuras de datos utilizadas en el proyecto y la sección *src* contiene los archivos de código fuente (.cpp) que implementan las funciones y estructuras declaradas en los archivos de encabezado (.h) ubicados en la carpeta *include*. Estos archivos de código fuente son los que contienen la implementación real de las funciones de Ethernet, SNMP, lectura de sensores y el servidor HTTPS.

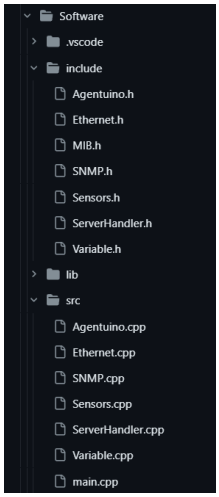


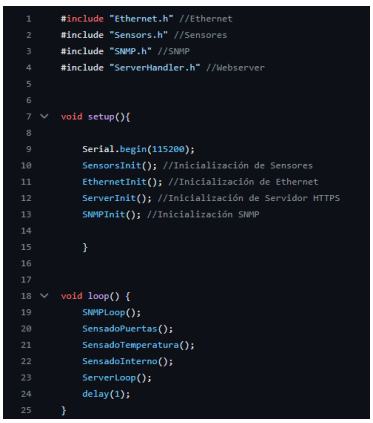
Figura 23: Estructura de código ESP32.

El código fuente consta de varias secciones:

- **Agentuino.cpp**: implementación del agente SNMP para el microcontrolador. Se encarga de administrar la comunicación con el exterior a través del protocolo SNMP, así como de manejar las solicitudes y respuestas SNMP
- **Ethernet.cpp**: configuración y la comunicación a través de la interfaz

Ethernet. Se encarga de establecer la conexión de red y de manejar la transferencia de datos entre el microcontrolador y otros dispositivos a través de Ethernet.

- **SNMP.cpp**: inicialización y recepción de paquetes SNMP.
- **Sensors.cpp**: se encarga de leer los datos de los sensores y el estado de los recursos internos del microcontrolador para convertirlos en formatos compatibles para su posterior transmisión a través del protocolo SNMP.
- **ServerHandler.cpp**: crea una interfaz web y maneja las solicitudes HTTP para proporcionar la información del estado de los sensores.
- **variable.cpp**: contiene las variables globales necesarias para el funcionamiento del código.
- **main.cpp**: contiene la llamada a las funciones de inicialización y bucle principal del programa (Fig. 24).



```
#include "Ethernet.h" //Ethernet
#include "Sensors.h" //Sensores
#include "SNMP.h" //SNMP
#include "ServerHandler.h" //Webserver


void setup(){

    Serial.begin(115200);
    SensorsInit(); //Inicialización de Sensores
    EthernetInit(); //Inicialización de Ethernet
    ServerInit(); //Inicialización de Servidor HTTPS
    SNMPInit(); //Inicialización SNMP

    }


void loop() {
    SNMPLoop();
    SensadoPuertas();
    SensadoTemperatura();
    SensadoInterno();
    ServerLoop();
    delay(1);
}
```

Figura 24: Código principal de ESP32.

Las dificultades que se encontraron y fueron superadas en esta etapa son las siguientes:

1. Desconocimiento del hardware necesario para la interfaz Ethernet del Concentrador de Sensores.
2. Dificultad de importación de la electrónica necesaria debida a la pandemia.
3. Limitado e inexistente stock de los diferentes posibles circuitos integrados para la interfaz Ethernet.
4. Librería de SNMP muy antigua y con errores.

3.8.3. Aplicación Móvil

Como previamente se definió en la sección de alcance de la aplicación, la misma contará con la capacidad de visualizar información de host/eventos, generar tableros con gráficas personalizadas y activar, de ser necesario, un servicio de notificaciones de alarmas/eventos que sucedan.

Para una mejor visualización de las diversas pantallas de la aplicación se elaboró un diagrama de navegación el cual podrá observarse en detalle en la sección de apéndices de la tesina.

Como detalle de la aplicación, se la diseñó de manera tal de que se encuentre dividida en modulos mediante el uso de *Fragments*, usando solo tres *Activities* durante toda la aplicación y otorgando de esta manera una mayor flexibilidad.

A continuación se detalla cada sección.

3.8.3.1. Splash Screen y Login

Para empezar se tiene una Splash Screen (Fig. 25 izquierda) , que muestra el logo del *NMS* utilizado por tres segundos mientras se carga la aplicación.

Luego de esto, se avanza a la pantalla de login (Fig. 25 derecha), en la cual se carga la dirección IP, el nombre de usuario y la contraseña para poder ingresar. Cabe destacar que la aplicación recuerda el último inicio de sesión y autocompleta los campos la próxima vez que se inicie la misma.

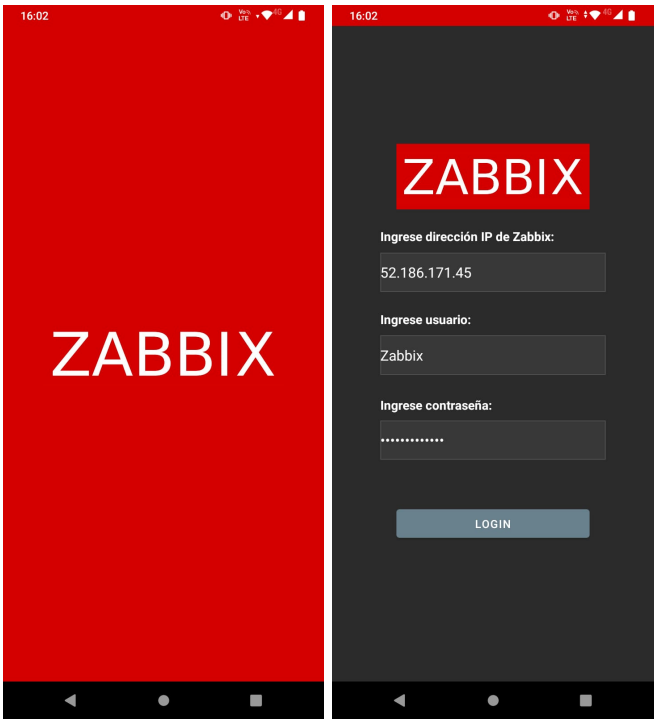


Figura 25: Izquierda: Splash Screen. Derecha: pantalla de login.

3.8.3.2. Servicio de notificaciones y Menú Principal

Luego de ingresar sesión se aprecia el menú principal, desde el cuál se puede cerrar sesión desde la *Toolbar* (Fig. 26 izquierda), y solicitar el alta o la baja del servicio de notificaciones desde un menú contextual en la Toolbar (Fig. 26 centro). Si se elige cerrar sesión, se vuelve a la pantalla anterior. Si se elige solicitar alguna acción de las notificaciones se abre una ventana para confirmar la decisión (Fig. 26 derecha), y se utiliza el servicio de correo registrado en el dispositivo del usuario para poder mandar un correo haciendo la solicitud. Una vez terminado esto, se vuelve al menú principal. De esta manera un administrador agrega el token generado por el dispositivo y habilita el servicio de notificaciones mediante el uso de *Firebase Cloud Messaging* (Fig. 27).

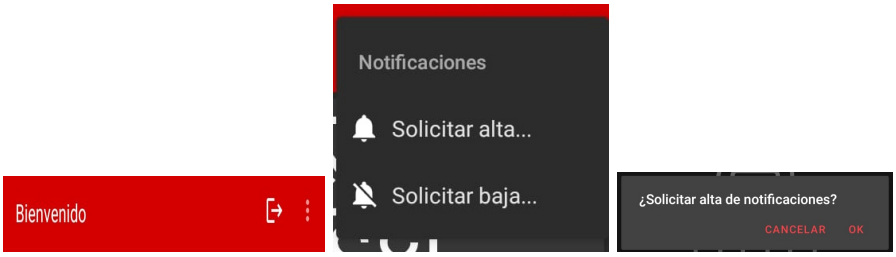


Figura 26: Izquierda: Barra del menú principal. Medio: Acciones de notificación. Derecha: Confirmación de notificación.

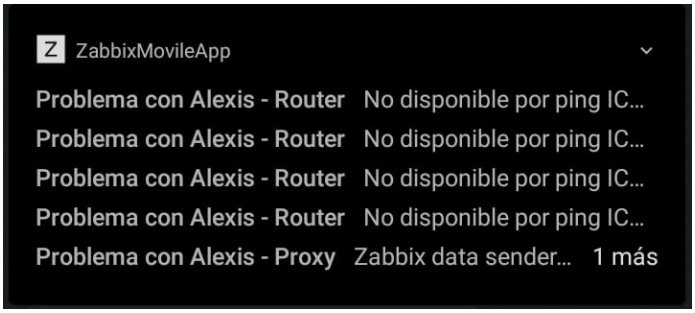


Figura 27: Notificaciones vistas desde el dispositivo del usuario.

Como última funcionalidad del menú se puede ingresar a las tres secciones de la aplicación (Fig. 28 izquierda): un tablero personalizable en el cual se ven los ítems que el usuario considere importante monitorizar, la lista completa de hosts y sus respectivos ítems junto con la última medición de los mismos, y un listado de eventos histórico.

3.8.3.3. Sección de eventos

En la sección de eventos (Fig. 28 centro) aparece el listado de los mismos de los últimos 30 días, ordenados con su respectivo código de color de acuerdo a su severidad: sin categoría, información, de alerta, de riesgo promedio, de alto riesgo y desastrosos. Si uno lo desea puede filtrar los eventos por el host en el cuál se producen (Fig. 28 derecha).

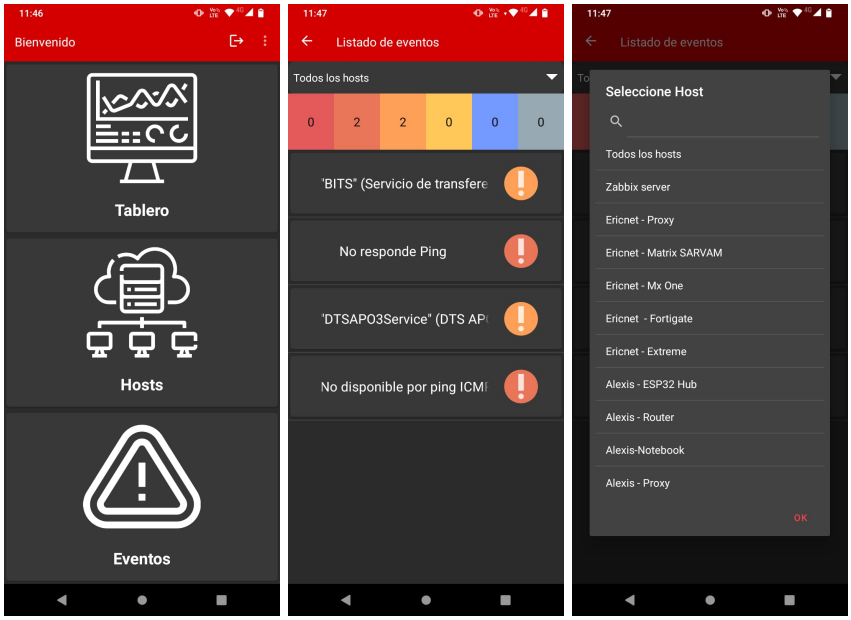


Figura 28: Izquierda: Menú principal. Centro: Menú de eventos. Derecha: Filtro de eventos por host

Al seleccionar alguno de los eventos, aparece su nivel de severidad, el momento en que se detectó y si está atendido o no (Fig. 29).

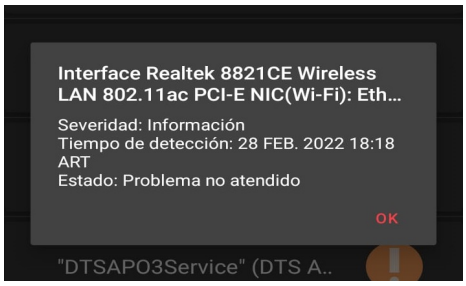


Figura 29: Selección de evento.

3.8.3.4. Sección de hosts

En la sección de hosts (Fig. 30 izquierda) aparece primero un listado de hosts que tiene disponible para visualizar el usuario, si se selecciona uno aparece un listado de grupos de ítems del host seleccionado (Fig. 30 centro) y si a continuación se selecciona alguno de estos grupos aparece un listado de todos los ítems pertenecientes a ese grupo con su último valor y la fecha y hora en el cual se midió el mismo(Fig. 30 derecha). Si se selecciona alguno de estos ítems aparecerá una descripción si es que tiene alguna (Fig. 31).

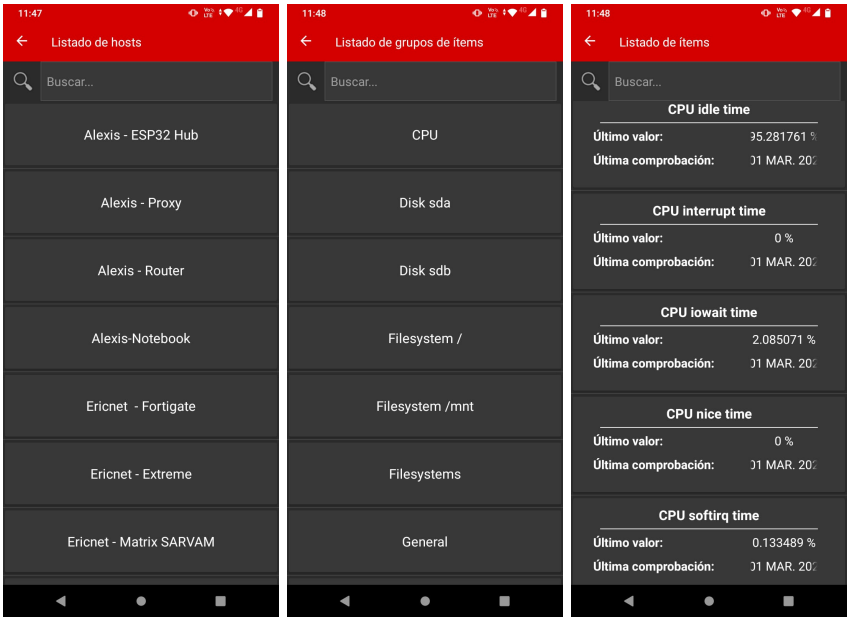


Figura 30: Izquierda: Menú de hosts. Medio: Menú de grupos de ítems. Derecha: Menú de ítems.

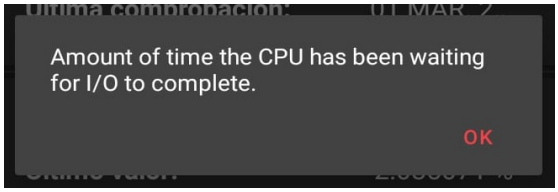


Figura 31: Selección de ítem.

3.8.3.5. Sección de tablero

Finalmente en la sección de tablero (Fig. 32 izquierda) aparecerá un listado de ítems guardados de manera local en el dispositivo del usuario de manera tal que no tenga que agregar un nuevo ítem cada vez que quiera monitorizarlo. Si se quiere agregar alguno primero se deberá seleccionar un host en el menú desplegable correspondiente, y un ítem en el otro menú desplegable. Si se selecciona algún ítem en el tablero, se mostrará una gráfica (Fig. 32 centro) con una ventana de tiempo modificable (Fig. 32 derecha). De la lista de la sección tablero se pueden agregar y quitar ítems (Fig. 33).

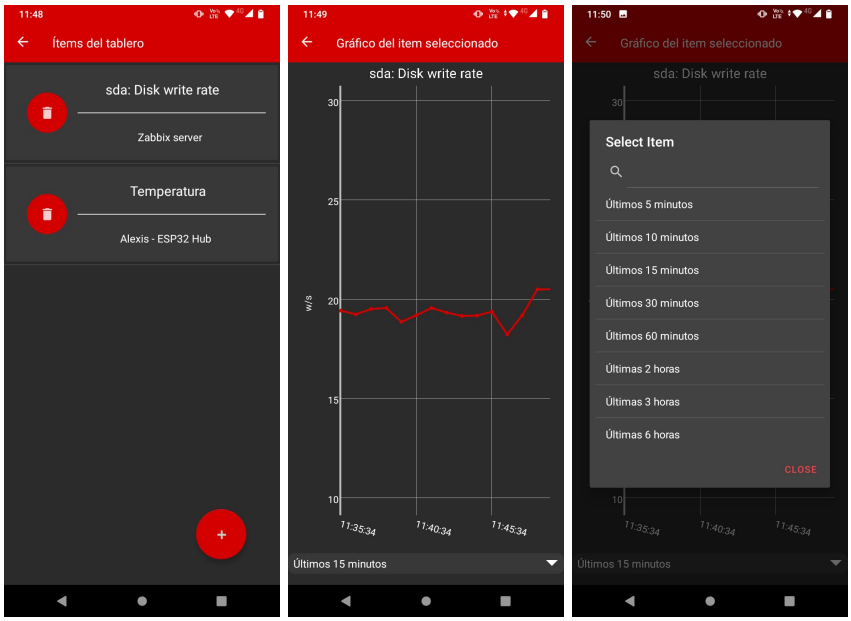


Figura 32: Izquierda: Menú del tablero. Medio: Gráfica de ítem. Derecha: Elección de ventana de tiempo.

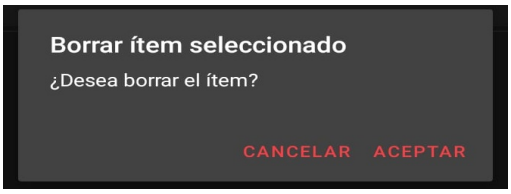


Figura 33: Confirmación de borrado de ítem de tablero.

3.8.3.6. Estructura del código, funciones principales y dificultades encontradas El código de la aplicación está estructurado de dos maneras: una sección que contiene los archivos de backend (la lógica de funcionamiento) y una sección que contiene los archivos de frontend (la apariencia estética). Dicha estructura se puede ver en la Fig. 34.

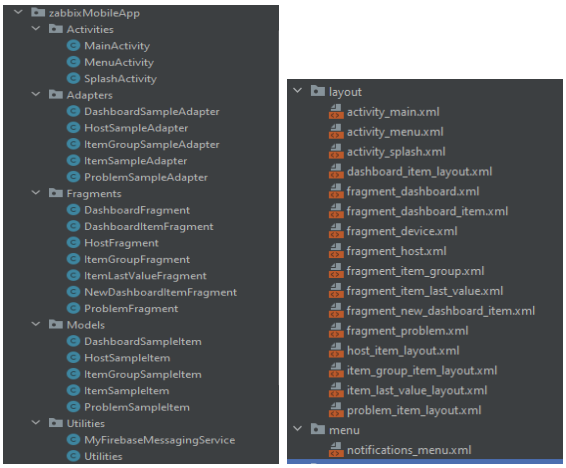


Figura 34: Organización de archivos

En cuanto al backend, el código está organizado en:

- **Activities**: tienen un layout asociado y organizan el código correspondiente a una pantalla de la aplicación.
- **Fragments**: tienen un layout asociado y organizan el código correspondiente a una pantalla de la aplicación. Son más dinámicas que las activities.
- **Adapters**: se encargan de organizar el funcionamiento lógico de los models dentro de una activity o fragment.
- **Models**: tienen un layout asociado y organizan el código correspondiente a una sección de una pantalla, por ejemplo un ítem de un menú.
- **Utilities**: contienen diversas funcionalidades auxiliares que se utilizan en el código de la aplicación.

Del funcionamiento general del código explicado anteriormente cabe destacar tres funciones principales:

- **client.newCall(request).enqueue()**: se encarga de conectarse con la API de Zabbix, para obtener los datos utilizados a lo largo de toda la aplicación.
- **graphView**: se encarga de representar gráficamente los datos adquiridos por la función anterior, es utilizada en la sección de tablero.
- **MyFirebaseMessagingService.onMessageReceived()**: se encarga de recibir las notificaciones si esta función está activada por el usuario.

Hubo dos dificultades principales encontradas a la hora de programar la aplicación.

La primer dificultad fue encontrar una librería que pudiera conectar con la API de Zabbix y mandar requests. Se optó por usar *OkHttp*, y aunque cumplió con su función se tuvieron bastantes problemas debido a que la API de Zabbix es bastante limitada.
La segunda dificultad fue encontrar una librería acorde que pudiera procesar los datos recibidos y pudiera levantar gráficas dinámicas, ya que el tamaño de un dispositivo móvil es limitado y se tiene que poder tener una buena definición punto a punto. Se optó por usar *GraphView*, ya que esta librería permite hacer zoom en la gráfica como para compensar las limitaciones de tamaño.

4. Resultado, Mediciones y Verificación

4.1. Resultado

Para la implementación de la solución propuesta se establece el siguiente ambiente de trabajo:
Un Servidor *Zabbix* instalado y configurado en una maquina virtual de Azure.
Un Proxy *Zabbix* instalado y configurado en una Raspberry Pi 4 conectado en la red local.
Una red local para conectar el Proxy *Zabbix* y el concentrador de sensores. No hace falta abrir ningún puerto en el Router.
Una PC con un navegador web para visualizar los gráficos provistos por *Zabbix* y un emulador de Android corriendo la aplicación móvil.

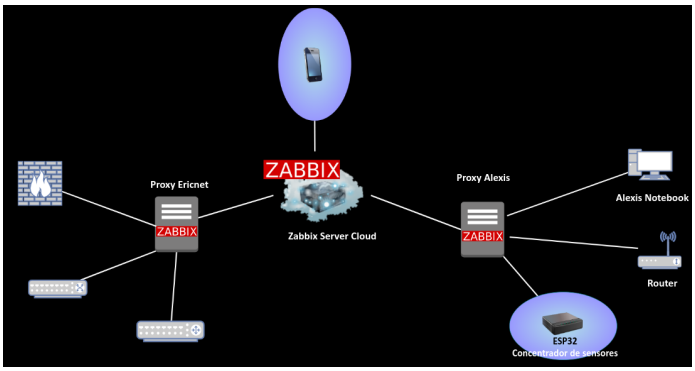


Figura 35: Ambiente de prueba

4.2. Mediciones

Se manipularon los sensores con el fin de observar en el sistema de monitoreo y su aplicación asociada los resultados. Esta serie de pruebas consta de:

- Sensor de puerta: manipular la puerta del gabinete de PC donde se encuentra el sensor de apertura de puerta.
- Sensor de Temperatura: cambios bruscos de temperatura.
- Se interactúa con la aplicación del sistema para ver los resultados de la creación de gráficos para monitorizar y configuración de notificaciones según los eventos de interés.

4.3. Verificación

La descripción del procedimiento llevado a cabo se encuentra mejor detallada en el informe del Test de Aceptación, el cual forma parte de los anexos de este informe.
Realizando los procedimientos detallados anteriormente se obtuvieron los siguientes resultados:

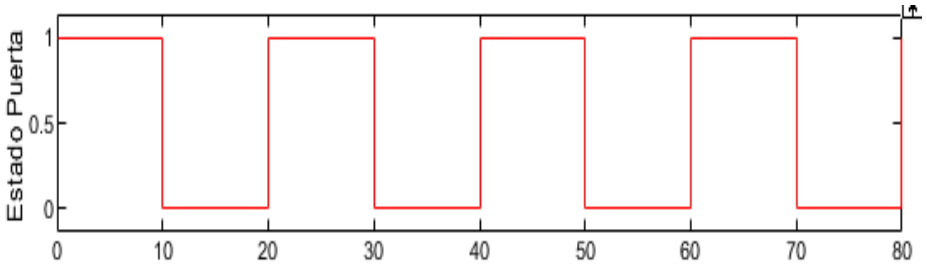


Figura 36: Excitación aproximada de la puerta, se abre y se cierra cada 20 segundos

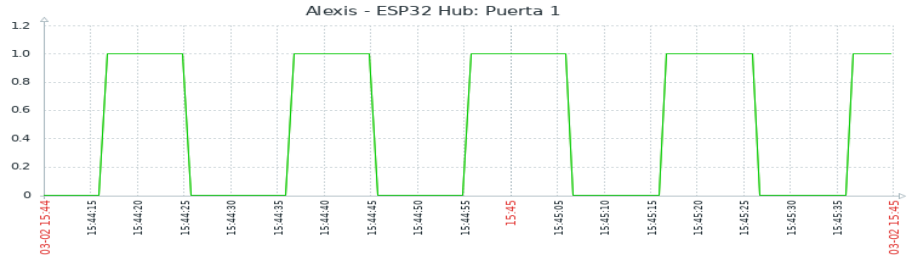


Figura 37: Respuesta de la puerta a la excitación

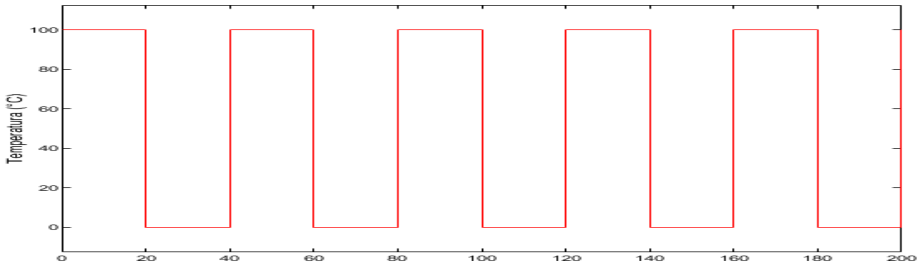


Figura 38: Excitación aproximada del sensor de temperatura, se lo coloca y se lo saca de un vaso con agua caliente

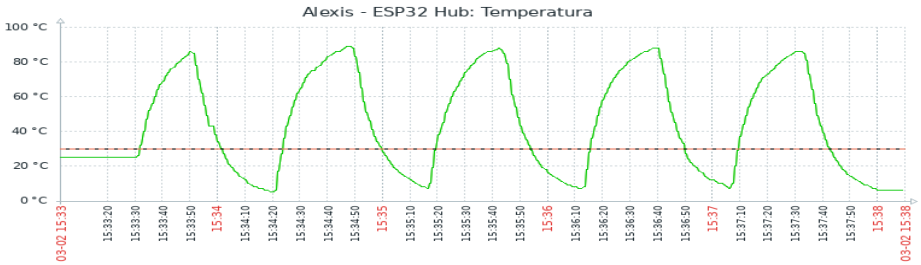


Figura 39: Respuesta del sensor de temperatura a la excitación

5. Conclusiones

El proyecto de diseño e implementación de una placa compuesta por el chip ESP32 utilizando el protocolo SNMP, chip y puerto Ethernet, fuente switching a 3.3V y optoacopladores ha culminado con éxito, logrando varios hitos importantes en el campo de la monitorización remota y la gestión de redes. A lo largo de este proyecto, se llevaron a cabo múltiples tareas, incluyendo la programación de la placa, la integración de sensores para la recopilación de datos y el desarrollo de una aplicación Android para visualizar gráficas provenientes de un servidor Zabbix conectado a la placa ESP32 diseñada.

La placa ESP32 diseñada y programada demostró tener un funcionamiento fiable para la monitorización de las condiciones de un datacenter propuestas en el alcance. La utilización del protocolo SNMP permitió una comunicación eficiente entre la placa y el servidor Zabbix, garantizando la transmisión confiable de datos desde los sensores conectados a la placa hacia el servidor.

La utilización de sensores para la recopilación de datos, junto con el protocolo SNMP completó la funcionalidad de la placa ESP32 al permitir la monitorización de las condiciones de un Datacenter.

El desarrollo de la aplicación Android fue un componente esencial del proyecto, ya que proporciona a los usuarios una interfaz intuitiva y accesible para visualizar las gráficas generadas por el servidor Zabbix en tiempo real. La aplicación facilitó la interpretación de los datos y mejoró la experiencia del usuario final al ofrecer una representación de fácil acceso del estado de la red monitorizada.

En términos de beneficios y aplicaciones prácticas, este proyecto se podría utilizar como complemento en diversos campos, incluyendo la monitorización de infraestructuras críticas dentro de la gestión de sistemas de energía o aspectos ambientales. La combinación de la placa ESP32, SNMP y la aplicación Android ofrece una solución integral y escalable para las necesidades de monitorización en tiempo real.

En resumen, este proyecto ha demostrado contribuir en el campo de la monitorización remota y la gestión de redes, destacando la eficacia y versatilidad de la placa ESP32 como plataforma para soluciones de IoT. Los logros alcanzados destacan el potencial de la integración de tecnologías y protocolos estándar para crear soluciones efectivas y prácticas en el monitoreo y control de sistemas.

6. Referencias

Referencias

[1] VMware, ¿Que es un Centro de Datos?, https://www.vmware.com/es/topics/glossary/content/data-center.html

[2] Cisco, ¿Que es el monitoreo de red?, https://www.cisco.com/c/es_mx/solutions/automation/what-is-network-monitoring.html

[3] Huawei Technologies (2019), What is SNMP?, https://support.huawei.com/enterprise/en/doc/EDOC1100086963#EN-US_TOPIC_0172881540

[4] Zabbix, (2021), Proxy, https://www.zabbix.com/documentation/5.4/en/manual/concepts/proxy

[5] Zabbix, (2021), API Manual , https://www.zabbix.com/documentation/current/manual/api

[6] JSON RPC Organization, (2021), Especification, https://www.jsonrpc.org/specification

[7] Unixstimestamp, (2021), https://www.unixtimestamp.com/

[8] Uptrends, (2021), https://www.uptrends.com/

[9] Tabbix Application, (2021), play.google.com/store/apps/details?id=com.tirgil.tabbix

[10] Poseidon, (2021), hhttps://www.hw-group.com/devices/poseidon#p

7. Índice de Figuras y Tablas

Índice de figuras

8. Apéndices

8.1. Circuitos eléctricos

Esquemático Concentrador

8.2. Hojas de Datos

Sensor de Temperatura: DS18B20
Sensor de Apertura de Puerta
Microcontrolador: ESP32 Wroom 32D

8.3. Programas Fuente

Repositorio de la aplicación Android
Repositorio del concentrador de sensores

8.4. Layout PCB

Repositorio con los Layouts

8.5. Otros Informes

Test de Aceptación
Póster
Fotos y Videos
Diagrama de Gantt