Universidad Zaragoza

1542

Trabajo Fin de Grado

APLICACIÓN MÓVIL PARA LA GESTIÓN DE SERVIDORES LINUX/UNIX

Autor

ÁLVARO GÓMEZ MUÑOZ

Director

DARÍO SUÁREZ GRACIA

Escuela de Ingeniería y Arquitectura de la Universidad de Zaragoza

2019

RESUMEN EJECUTIVO

El mundo de los administradores de sistemas requiere de nuevas herramientas para la gestión de servidores, ya que la consolidación de servicios y los modelos 'cloud' hacen que el número de servidores administrados por persona haya aumentado considerablemente, y por ello se debe empezar a apoyar los sistemas de movilidad. El nombre de este proyecto es Aplicación Móvil para la gestión de servidores Linux, nombre escogido por lo que es la funcionalidad del proyecto. Este trabajo de fin de grado pretende llenar parcialmente este vacío con el desarrollo de una aplicación móvil, que es lo que suele faltar en el ámbito de los administradores de sistemas. Se ha implementado una aplicación cliente-servidor, la cual se ha denominado como ServCM (Server Control Master, también denominado en español Control maestro de servidores), en la que el servidor suministra los datos de su estado, como el uso de CPU, Memoria... al cliente y este tiene la posibilidad de mandar comandos y generar eventos en el servidor como manera de contrarrestar cualquier posible fallo.

Para la realización de este TFG se ha realizado una revisión exhaustiva de las tecnologías disponibles para resolver los retos que plantea el proyecto. Eligiendo así las más adecuadas y que a su vez, sean llamativas para una comunidad activa, de tal manera que tras la puesta en producción, éstas puedan ser mejoradas y usadas por distintos usuarios que quieran personalizarlas. Se han revisado tecnologías como Node.js, .NET Core, Spring para el servidor principalmente y Xamarin, Ionic y android nativo para la apliciación móvil.

La aplicación ha sido generada principalmente para administradores de sistemas de nivel medio a alto, encargados de sistemas operativos linux (pero no se descarta el paso a Windows), pues el 67.8 % de los servidor que funcionan en el mundo, utilizan esta plataforma según W3Techs.

Como administrador de sistemas, la idea de esta aplicación ha sido mía, puesto que necesitaba una aplicación que cubriera estas necesidades y que no estuvieran obsoleta. También todo el código generado en este proyecto ha sido liberado en la siguiente url: https://github.com/algomezmu/TFG para que toda la comunidad pueda aportar mejoras sobre el mismo y a la vez, para que se puedan cerciorar de como funciona por detrás la aplicación y de que no hay ningún truco o falsedad detrás del código. Actualmente la aplicación está en marcha y cumple los requisitos planteados por el proyecto.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

1 INTRODUCCIÓN

Este capítulo presenta los objetivos, motivación, alcance y la estructura de este trabajo fin de grado.

1.1 OBJETIVOS

El objetivo principal del TFG es proporcionar los mecanismos necesarios para monitorizar un conjunto de servidores de manera intuitiva en una aplicación móvil. Dicha aplicación será modular y expansible en un futuro.

El objetivo secundario es mejorar mis conocimientos en diferentes tipos de tecnologías que se están usando en el mercado actualmente, a su vez que probar el diseño de herramientas para el diseño en entornos móviles.

Los puntos fuertes de la aplicación que se va a desarrollar son, entre otros, el mínimo uso de los datos del dispositivo móvil y la posibilidad de disponer de varios perfiles de usuario, los cuales utilizen permisos de control y monitorización o solo de monitorización.

También es importante el desarrollo de la aplicación con las últimas tecnologías, pues son las más utilizadas y con más comunidad activa por lo general dentro del mercado laboral, y con este factor, que otros usuarios puedan editar y mejorar la aplicación si así lo desean. Estas tecnologías pueden ser Ionic para el cliente móvil y Node.js para el servidor pues ahora mismo tienen más de la mitad del mercado [17].

1.2 MOTIVACIÓN

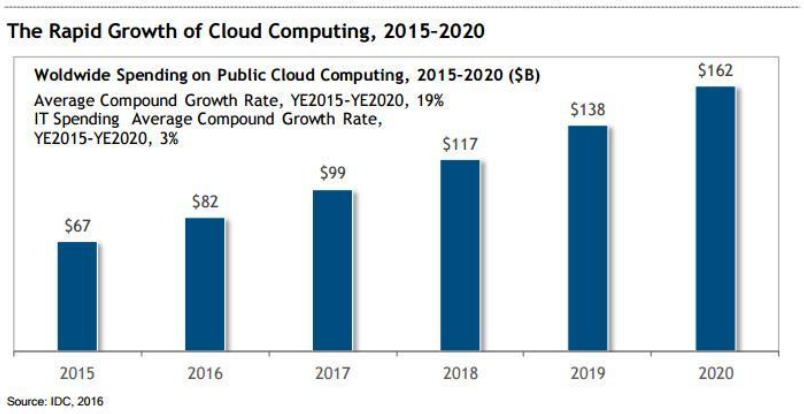


Figura 1: Crecimiento en Cloud computing

Estamos en una época de cambio tecnológico como se puede apreciar en la Figura 1, la cual representa el rápido incremento de las necesidades en la "nube" y su correspondiente económico en billones de dolares. Por lo tanto, es necesario cada vez más, nuevas herramientas de todos los niveles para el manejo de las infraestructuras de servidores actuales y futuras. Estás necesidades funcionalmente y económicamente hablando han ido en crecimiento y de momento no se vislumbra el momento en que pararán.
La idea de esta aplicación, es cubrir algunas de esas necesidades durante este cambio, como disponer de acceso rápido al conjunto de servidores en "cloud" desde el dispositivo móvil, a la vez que recibir notificaciones en éste en caso de problemas. Ahora mismo, las opciones que se encuentran en el mercado no son las que más se adaptan a las necesidades que un usuario medio puede requerir, puesto que alguna de las aplicaciones actuales o son opciones muy caras como Nagios o no se encuentran adaptadas a un entorno móvil como Zabbix.
Y aquellas que solo se centran en el entorno móvil, se encuentran limitadas a la hora de monitorizar el sistema; esto es debido a que solo se dispone de la aplicación móvil, lo que provoca que el móvil no reciba mucha información o que éste se encuentre con un uso excesivo uso de los datos de internet. Por ello este proyecto intenta poner a disposición del publico no solo la aplicación móvil, sino que también se proporciona una aplicación que se instala en el servidor del usuario y recopila los datos necesarios para monitorizar el sistema.

1.3 ALCANCE

Este proyecto intenta englobar varios puntos de interés:

1. La generación de un servidor dentro de un entorno Linux, que sea una pasarela que permita monitorizar el ordenador donde se ejecuta y a su vez, lanzar comandos y reaccionar a eventos.

2. La creación de una aplicación móvil que se conecte a este servidor y que permita mostrar de una forma intuitiva la información que se monitoriza y otras funcionalidades.

El primer punto corresponde a un servidor, que pueda realizar llamadas al sistema operativo y que permita recopilar la información del ordenador en todo su conjunto, como el estado de la CPU, el estado de la memoria RAM... Dicho servidor tendrá que permitir la recolección de la información desde el exterior desde cualquier plataforma. A su vez, el servidor permitirá lanzar comandos y crear eventos que se ejecuten en el momento que le ocurra cualquier cosa que le indiquemos a los datos monitorizados, como un alto uso de la CPU.
El segundo punto consiste en proporcionar una aplicación móvil que permita conectarse al servidor previamente mencionado para visua-

lizar la información monitorizada de una manera cómoda y generar desde la misma interfaz, los eventos y lanzar scripts.

1.4 DESCRIPCIÓN

En esta sección, se va a explicar como se ha organizado la memoria para un rápido desglose: Capitulo 1 corresponde a la descripción del proyecto y los puntos que se intentan introducir dentro de ésta.

El capítulo 2 analiza las aplicaciones existentes para ver sus virtudes y debilidades, de tal manera que se pueda ver en que aspectos se puede mejorar la aplicación que se va a generar, como la interfaz móvil.

El capítulo 3 describe cuales van a ser los requisitos para el desarrollo del proyecto, identificando los requisitos tanto funcionales como no funcionales e indicando qué tecnologías finales se han utilizado para el desarrollo.

El capitulo 4 explica la arquitectura propuesta para el sistema, mostrando como funcionan los aspectos tanto de comunicación como los internos del sistema.

El capitulo 5 explica la correspondencia con las metodologías usadas y los procesos que se han seguido hasta llegar al final del desarrollo.

El capitulo 6 explica las pruebas que se han realizado para comprobar que todo funciona como es debido, tanto que las aplicaciones se comunican correctamente entre si, como que no haya fallos de sistema.

Por ultimo, el capitulo 7 expresa las conclusiones finales del proyecto.

Se ha añadido el capitulo de Anexos para rellenar información que no es totalmente relevante para la memoria principal, pero es útil en caso de querer consultar más sobre ésta. Se ha agregado información como el prototipo en papel de todas las pantallas y también las definiciones de la API del servidor de una manera más extendida.

2

ANÁLISIS DE APLICACIONES EXISTENTES

Este capítulo describe algunas aplicaciones de monitorización relevantes centrándose en sus fortalezas y debilidades para realizar la captura de requisitos del TFG.

Se ha decidido analizar alguna de las aplicaciones más significativas del mercado, tanto Servidor como Clientes móviles, de tal manera que se pudiera hacer un análisis objetivo que a continuación se detallará. Se analizarán 2 aplicaciones para la sección del servidor, y 3 para la de dispositivos móviles, siendo éstas las más conocidas o puntuadas en el mercado.

2.1 APLICACIONES DE SERVIDOR

2.1.1 *Zabbix*

Figura 2: Zabbix

Zabbix [19], es un Sistema de Monitorización de Redes creado por Alexei Vladishev. Está diseñado para monitorizar y registrar el estado de varios servicios de red, servidores, y hardware

de red tal y como se muestra en la Figura 2. No dispone de una aplicación móvil oficial, sino versiones que utilizan una API.

El cuadro 1 muestra los principales datos monitorizados por Zabbix.

Cuadro 1: Datos Monitorizados Zabbix

Monitorización
Uso de CPU
Uso de disco
Uptime, Downtime
Dispone de eventos y trigger (respuestas a los eventos)
Trafico de Red

La aplicación es completa pero con una interfaz Web poco amigable y compleja a la vista, diseñadas en distintos lenguajes. A su vez, el código es Open Source (código libre para ser editado) y gratuito. En este TFG, se podría haber utilizado la API de Zabbix pero se ha preferido moldear el problema a uno que permitirá disponer de algo simplificado para su posterior desarrollo.

2.1.2 *Nagios*

2.1.3 *Server & Website Monitor*

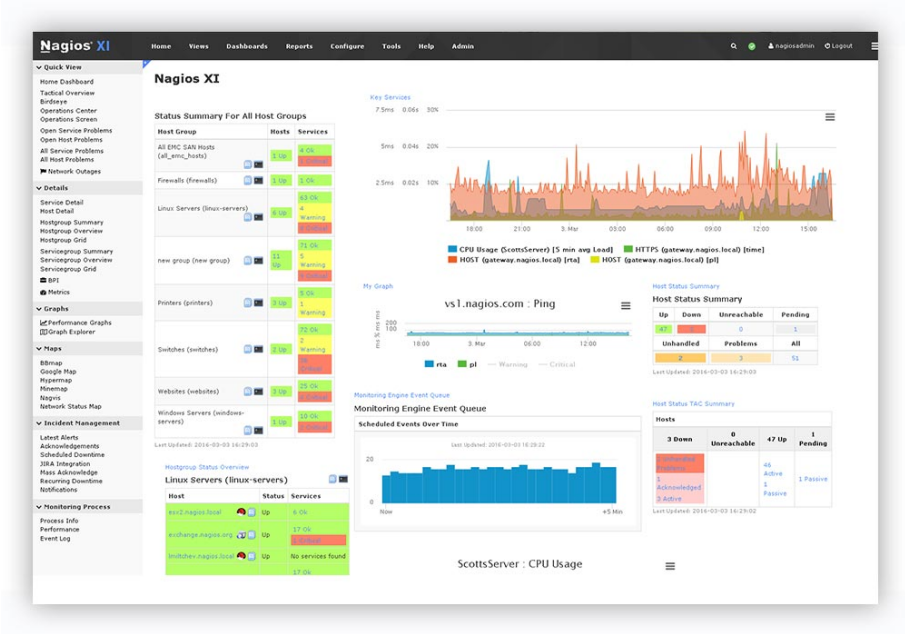


Figura 3: Nagios

Nagios [21], es un sistema de monitorización de redes ampliamente utilizado, de código abierto, que vigila los equipos (hardware) y servicios (software) que se especifiquen tal y como se muestra en la Figura 3 y alertando cuando el comportamiento de los mismos no sea el deseado.

El cuadro 2 muestra los principales datos monitorizados por Nagios.

Cuadro 2: Datos Monitorizados Nagios

Monitorización
Uso de CPU
Uso de disco
Uptime, Downtime
Dispone de eventos y trigger (respuestas a los eventos)
Trafico de Red
Usuarios Loggeados

La aplicación es una de las más completas del mercado pero tiene un precio elevado, en torno a unos 2000 €, que para un usuario medio es un precio inviable. Ésta funciona a través de un proceso de linux y proporciona una API. Su página web corre en otro proceso.

2.2 APLICACIONES MÓVILES

Ambas aplicaciones mencionadas en el apartado anterior disponen de aplicaciones móviles oficiales o no oficiales. Estas se encuentran limitadas, pues han sido pensadas para el uso exclusivo de un entorno web y se han ido adaptando. Estas limitaciones entran dentro de la visualización y de la funcionalidad en general.

2.2.1 *Server & Website Monitor*

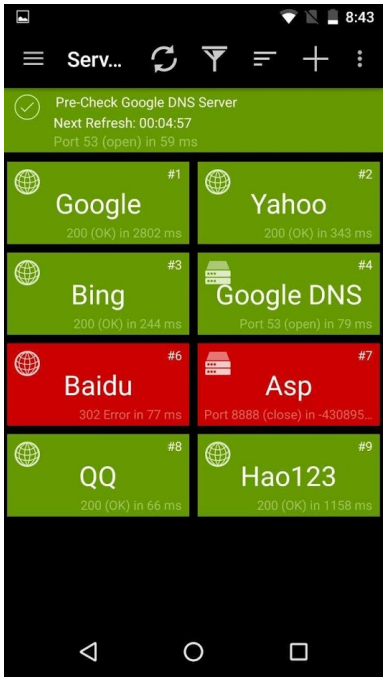


Figura 4: Server & Website Monitor

Server & Website Monitor es una aplicación orientada únicamente a saber si el servidor o servidores registrados dentro de la aplicación se encuentra en funcionamiento, mandando un paquete del tipo ping, tal y como se muestra en la Figura 4, la cual corresponde a la pantalla principal de la aplicación. Los colores verde y rojo

indican que el servidor está en funcionamiento y apagado, respectivamente.

Es una aplicación muy limitada con respecto a sus capacidades, pues le falta funcionalidad como la necesidad de permitir gestionar el servidor.

2.2.2 *Cura SysAdmin*

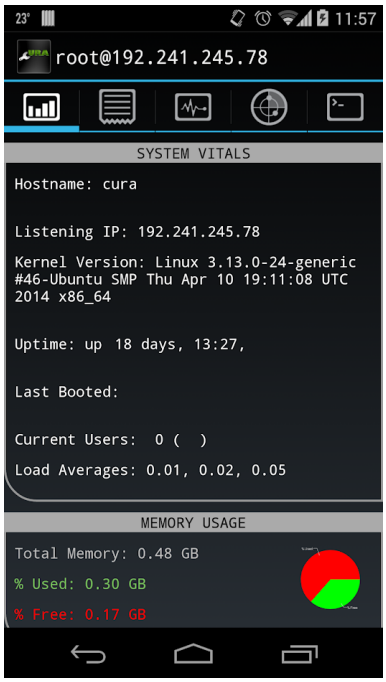


Figura 5: Cura SysAdmin

Cura es una aplicación orientada a ver un mínimo de información del sistema y lanzar comandos vía SSH como se muestra en la Figura 5, por lo tanto está limitada a sistemas que usen SSH, lo que implica que el sistema gestor de usuarios se encuentra en el sistema operativo y es más difícil de gestionar, a la vez que hay que proporcionar permisos a estos. Al no disponer de API, tiene que realizar una extracción completa de los datos del servidor sin un historial de lo que pueda haber ocurrido. En la actualidad, esta aplicación ha dejado de actualizarse y está dejando de funcionar en ciertos dispositivos móviles, a la vez que le faltan algunas funcionalidades como puede ser un sistema de eventos.

Aun disponiendo solo de SSH como principal capa de comunicación entre el servidor y el cliente, tiene la ventaja de dar al

usuario la libertad que este sistema de comunicación proporciona, pero sin muchas funcionalidades extras.

2.2.3 *Server and Service Monitor*

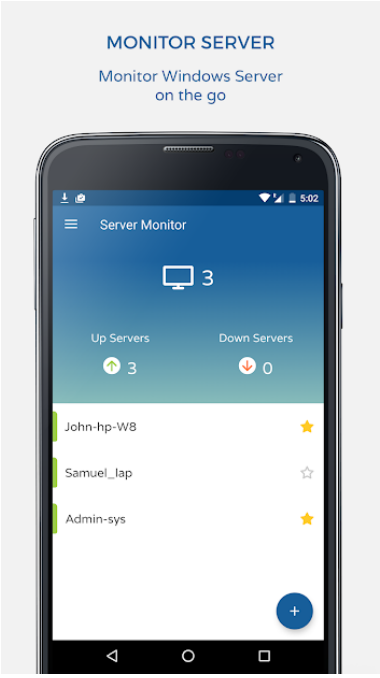


Figura 6: Server and Service Monitor

Server and Service Monitor [1] es una aplicación orientada a ver información del sistema y detener servicios que se ejecuten en el sistema si el usuario lo indica, todo realizado vía SSH. Está centrada en que no se usen comandos ssh, controlándolo todo desde una interfaz amigable. Tiene las limitaciones de no controlar eventos dentro del sistema y de no permitir que se usen comandos ssh en caso de querer realizar alguna comprobación.

3

REQUISITOS DEL SISTEMA

En este capítulo se van a observar los requisitos de los componentes del sistema. En este caso el servidor y el cliente. Definiendo en cada punto los requisitos tanto funcionales como no funcionales e indicando los puntos principales que se van a tener en cuenta.

3.1 SERVIDOR

Como primer paso a la hora de definir el servidor, se han recogido los requisitos para desarrollarlo, los cuales han sido los siguientes, dividiendo entre funcionales y no funcionales.

Tras comprobar los requisitos, se ha decidido investigar varios de los lenguajes que hoy en día predominan en el mundo web y las necesidades que el proyecto puede tener, tanto ahora como a posteriori.

3.2 CLIENTE

Para esta tarea, se ha decidido investigar cuál es la plataforma móvil que más predomina en el mercado, y según cual sea la plataforma, por qué versión de ésta empezar y siempre intentando abarcar la mayor cantidad de usuarios posibles.

Para la puesta en marcha, se han recogido los requisitos funcionales y no funcionales como en la anterior sección.

Funcionales	**No Funcionales**
El sistema debe permitir crear y modificar los usuarios del servidor	El sistema debe permitir conexión a múltiples clientes
El sistema debe devolver la información actual e histórica de los datos monitorizados (cpu, memoria...)	El sistema debe tener acceso a la información relevante de los datos monitorizados
El sistema debe permitir que se filtre la información histórica por rango de tiempo	El sistema debe permitir un rápido desarrollo del mismo
El sistema debe permitir leer y modificar la configuración del servidor respecto a los tiempos de monitorización	El sistema debe usar un lenguaje moderno de programación
El sistema debe permitir la creación de eventos y lanzamiento de scripts	El lenguaje de programación debe tener una comunidad fuerte con materiales y frameworks
El sistema debe permitir subscribirse a un evento y notificar si hay algún problema	El sistema debe permitir enviar notificaciones
El sistema debe permitir que los usuarios se conecten	El sistema debe proteger la información del usuario y ser seguro
	El sistema debe permitir monitorizar la información del entorno donde se ejecuta

Cuadro 4: Requisitos Funcionales y No Funcionales del servidor

Funcionales	**No Funcionales**
El sistema debe permitir crear y modificar los usuarios que se encuentran en el servidor	El sistema debe usar un lenguaje moderno de programación
El sistema debe permitir registrar varios servidores	El sistema debe permitir un rápido desarrollo del mismo
El sistema debe mostrar la información actual e histórica de los datos monitorizados en el servidor	El lenguaje de programación debe tener una comunidad fuerte con materiales y frameworks
El sistema debe permitir que se filtre la información histórica por rango de tiempo	El sistema debe proteger la información del usuario
El sistema debe permitir mostrar y modificar la configuración del servidor respecto a los tiempos de monitorización	El sistema debe permitir procesar la información recibida
El sistema debe permitir la creación de eventos y lanzamiento de scripts	
El sistema debe permitir subscribirse a las notificaciones de eventos	
El sistema debe permitir la recepción y visualización de notificaciones	
El sistema debe permitir registrar, modificar y borrar varios servidores	

Cuadro 5: Requisitos Funcionales y No Funcionales cliente

Con estos requisitos, también hay que tener en cuenta, que se pretende aprender a utilizar los últimos lenguajes y frameworks que se están usando en el mercado laboral. Por ello se ha investigado y probado la diferencia entre dos frameworks dentro del mercado (Ionic y NativeScript) que más adelante se explicará 5.4.2.3.

4

ARQUITECTURA DEL SISTEMA

Este capítulo describe la arquitectura del cliente y del servidor, incluyendo la comunicación entre ambas aplicaciones.

4.1 VISIÓN GENERAL

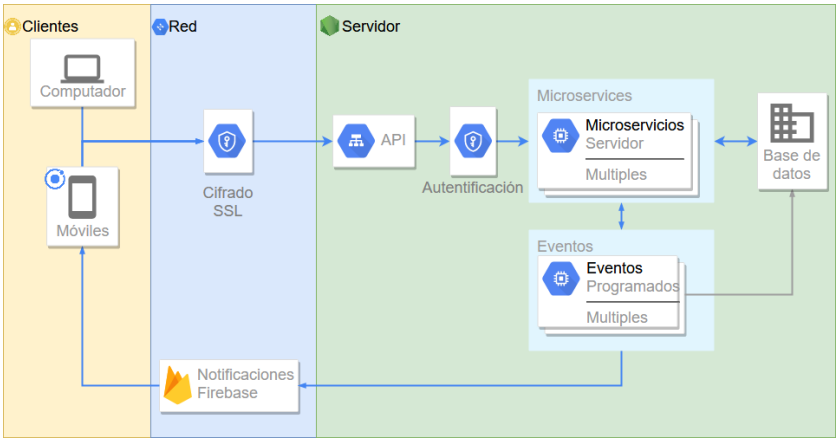


Figura 7: Arquitectura del sistema

La Figura 7 muestra la arquitectura general del sistema. Se ha empleado una arquitectura cliente/servidor porque es una manera cómoda de escalar el sistema en el futuro y poder tener un mejor control de la información. De tal manera que varios servidores puedan enviar información a un cliente, que los tenga configurados en su interfaz. Así se consigue cohesionar la información en un dispositivo amigable. Y los servidores no están limitados a un solo cliente, sino que pueden comunicar la información a todos aquellos clientes que se hayan configurado dentro de ese sistema.

A continuación se detalla cada elemento del sistema siguiendo el criterio de servidor-cliente [7].

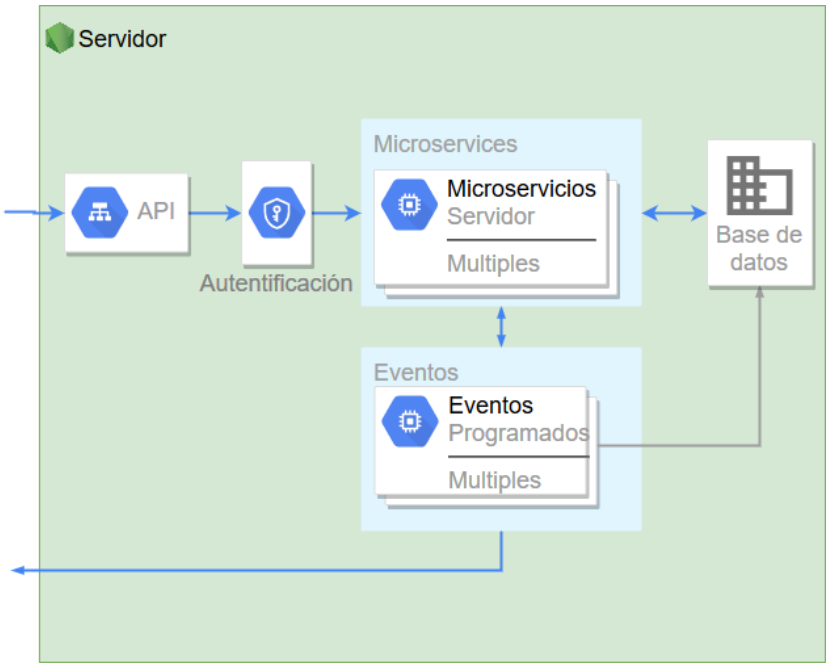


Figura 8: Arquitectura Backend

La Figura 8 representa el funcionamiento del servidor. El API / API REST [8] corresponde al acceso mediante el protocolo HTTPS al servidor, dicho acceso requiere de la dirección URL concreta al microservicio que se quiere usar. Este TFG, define como microservicio a cualquier funcionalidad del servidor que es independiente de otras partes o servicios del servidor.

Entre la API y los microservicios, se encuentra una capa de seguridad que impide el acceso a éstos excepto que el cliente se encuentre autentificado, de tal manera que cualquier petición entrante que no esté autorizada, se expulsará del sistema tras comprobar sus credenciales.

Hay ciertos microservicios, que se encuentran fuera de la zona de autentificación, que aunque no se muestre en la Figura 8, se clarifica en el apartado de microservicios 4.2.1.

El apartado de Eventos corresponde a todos los eventos que el usuario ha programado y la recopilación del estado del sistema, en este caso son los datos de CPU, Memoria... Los eventos programados por el usuarios, son avisos o acciones que se van a ejecutar cuando determinado momento ocurra.

Los microservicios y los eventos, están conectados a una base de datos que gestiona toda la información necesaria para que la aplicación funcione, guardando un histórico de los datos recopilados y de acciones del usuario.

Dicha base de datos mantiene los datos almacenados durante un periodo de tiempo determinado, que se define en los ficheros de configuración iniciales del sistema. Por otra parte, la información que se almacena, no requiere un alto porcentaje de disponibilidad ni de seguridad, por ello no se han propuesto sistemas de replicación o almacenamientos alternativos, ni sistemas de encriptación para dichos datos.

Como puntos a tener en cuenta para la elección de una base de datos, se ha tenido en cuenta que los datos que la aplicación va a generar no son del tipo que requieran relaciones y que pueden ser datos con formato variable o sin formato a la vez que se requerirán guardar datos históricos. Por lo tanto, se cree que una bases de datos NoSQL es la mejor para el almacenamiento. Más adelante en este documento se detallarán este tipo de bases de datos.

4.2.1 *Microservicios*

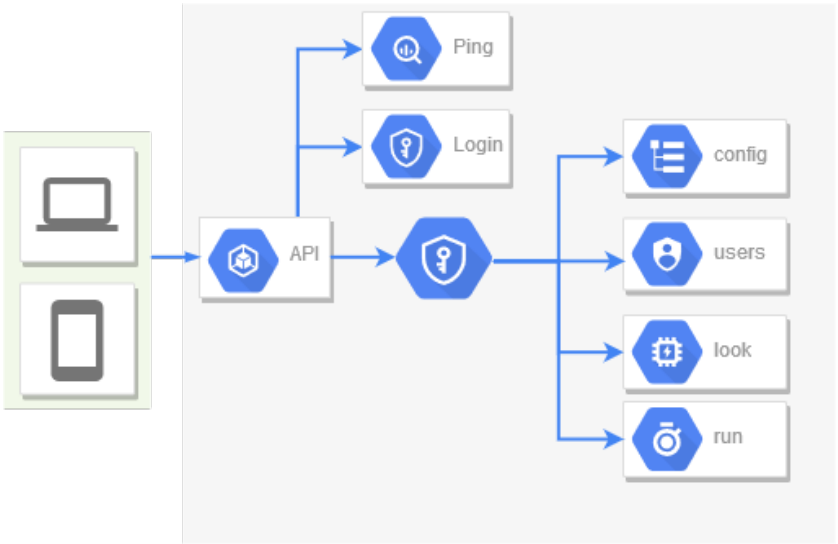


Figura 9: API Servidor

Para precisar mejor con la capa de microservicios, con respecto a la API, se va a mostrar el diseño global tal y como se muestra en la Figura 9.

En la Figura 9 se observan el punto principal de entrada "API", que es indicativo de que estamos entrando en el servidor y nos redirigirá al microservicio que le indiquemos. Todo microservicios es indicado mediante la URL, concretamente la parte de ruta [20].

Tras entrar en el punto principal, se identifican 2 grupos: uno de los grupos dentro de un sistema protegido que requiere una previa autenticación, tal y como se explica en el apartado anterior y el otro grupo que no lo requiere.

El primer grupo, el cual no requiere de autenticación, tiene 2 microservicios, uno llamado "Login", que permite a un usuario autentificarse con respecto al servidor y acceder al segundo grupo de microservicios y otro que sirve como el comando Ping de cualquier sistema informático "Ping", de tal manera que se puede hacer un calculo de la latencia y el tiempo que tarda el servidor en contestar.

Tras habernos autentificado sin fallos en el sistema, se visualizan 4 grupos principales como se muestra en el cuadro 6, los cuales se han desglosado de esa manera para simplificar el esquema, pues cada uno, tienen más microservicios.

Cuadro 6: Servicios API Autentificados

Servicios	Descripción
config	Encargado de las opciones de configuración como los tiempos de recogida de datos.
users	Encargado de las altas, bajas... de los usuarios dentro de la aplicación.
look	Encargado de devolver los datos tanto históricos como actuales de los distintos sistemas monitorizados.
run	Encargado de la creación, edición, borrado de los eventos del sistema, incluyendo el lanzamiento de comandos.

Figura 10: Comunicación SSL

La comunicación existente entre Cliente y Servidor, se realiza mediante SSL [18] (Secure Sockets Layer), el cual es un protocolo de seguridad que usa una moderna encriptación para enviar y recibir información sensible en todo internet.

El servidor crea un canal seguro con el cliente (móvil o web o cualquier otra plataforma) al que el usuario quiere acceder o conectarse. Cualquier información que pase entre ese canal está encriptada y es descifrada cuando llega a su destino. De esta forma, aunque alguien consiga la información, es prácticamente imposible sacarle un uso, debido a que esa información está encriptada.

4.3.1 *Firebase*

Figura 11: Firebase

Este servicio es una plataforma diseñada por Google, que permite entre muchas cosas, la comunicación segura por ssl de notificaciones entre clientes móviles y el servidor de Backend. Se ha decidido su uso para facilitar dicha tarea, siendo un servicio altamente usado y gratuito para las necesidades de esta aplicación. Entre sus ventajas se encuentra que permite su uso entre los distintos dispositivos del mercado, incluyendo el sistema operativo Android e iOS. También la facilidad de notificar al dispositivo de nuevas notificaciones [10]. a la

vez que la unión de más dispositivos a dichas notificaciones, de tal manera que si se dispone de múltiples clientes, todos reciban el aviso. Como desventaja para este TFG, se ha observado que el servicio para mandar notificaciones, requiere que los servidores dispongan de las claves privadas de este servicio, lo que puede ser un riesgo en la seguridad. Por ello, para solucionarlo, se ha prohibido el acceso de lectura a dicho servicio, y solo se le dan permisos de escritura a los clientes. De tal manera que los servidores que se generen, tendrán que mantener una lista de los clientes a los que proporcionan el servicio y no dispondrán de toda la lista de todos los clientes dentro de la plataforma. Como punto positivo a esto, centralizando todos los clientes en la plataforma, permite que si se viera necesario, se podría mandar mensajes a todos los clientes como informes de mejoras del servicio.

4.3.2 *Usuarios*

Dentro de la propia aplicación, se gestionan dos tipos de usuario:

Cuadro 7: Usuarios Aplicación

Tipo	Rol
Administradores	Tienen permisos para monitorizar el sistema y crear eventos o lanzar comandos. También disponen de permisos para crear usuarios dentro de la aplicación.
Monitores	Solo disponen de permisos de monitorización del servidor, sin ningún tipo de opción de lanzar comandos.

Se han querido generar estos 2 usuarios de una manera separada, para permitir que otras personas que no deberían tener permisos sobre el sistema puedan monitorizarlo igualmente. De tal manera que si ocurriera algo, se pudiera avisar al administrador.

Figura 12: Dispositivos

En la Figura 13, se muestran los dispositivos a los que se orienta la aplicación de cliente, los cuales se van a comunicar con el servidor.

En el caso de este proyecto, el cliente es una aplicación móvil, pero no se descarta que en el futuro, se prepare una interfaz Web. Pues la aplicación del servidor ofrece sus datos vía una API que permite dar soporte a múltiples clientes, de tal manera que se puedan conectar todos a la vez.

Los clientes mantienen el mínimo número de datos dentro de su sistema, de tal forma que se reduce considerablemente el espacio utilizado en el dispositivo móvil. Se ha decidido diseñar la aplicación de esta manera, puesto que aún existen bastantes limitaciones en lo que respecta al espacio en los dispositivos móviles; en cambio la potencia y la red, siguen siendo bastante altas por lo general.

A continuación, se ha realizado un prototipo en papel, el cual sirva de punto inicial a la hora de generar la aplicación cliente, no teniendo que ser el productor final de igual manera.

Como ejemplo, se muestran algunos bocetos realizados a mano sobre las pantallas, siguiendo la metodología de prototipo en papel, tal y como se ha enseñado durante el grado en asignaturas como Interacción persona ordenador o Diseño persona ordenador. Las demás pantallas se pueden observar en el apartado de Anexos.

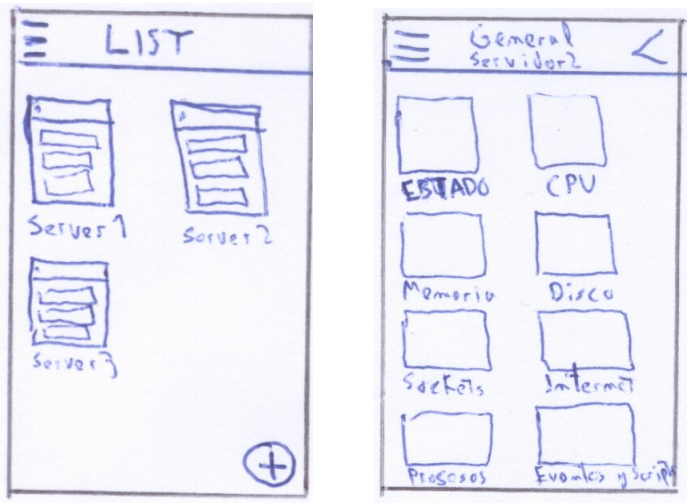


Figura 13: Dispositivo en papel

La imagen izquierda de la Figura 13a representa la pantalla desde donde se permitirá ver todos los servidores agregados a la aplicación, permitir acceder a cada uno de ellos y agregar nuevos.

La imagen derecha de la Figura 13b representa el panel desde donde se mostrarán todas las opciones dentro del servidor, entre ellas las siguientes:

Figura 14: Estado del servidor

CPU	Memoria
Disco	Sockets
Consumo de Internet	Procesos
Eventos	Scripts
Usuarios	Configuraciones

Las opciones corresponden a las acciones vistas en la arquitectura del servidor principalmente.

5

METODOLOGÍA E IMPLEMENTACIÓN

Este capítulo describe aquellas metodologías de trabajo junto a sus herramientas que se han utilizado, cómo se ha gestionado el tiempo y cómo se ha llevado a cabo la implementación de del proyecto.

5.1 METODOLOGÍA KANBAN

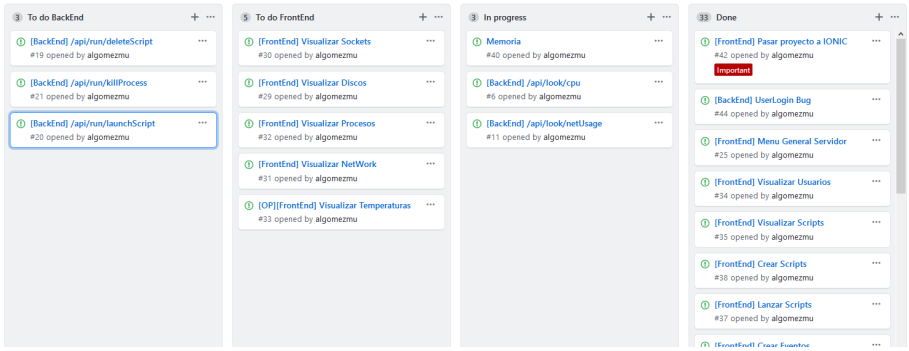


Figura 15: Kanban

La metodología Kanban es una de muchas metodologías ágiles que se usan en el mercado y tiene como objetivo establecer un flujo de trabajo constante y organizado a través del uso de tarjetas donde se escribe y evalúa la tareas de un proyecto. Dicha tarea va pasando por distintas fases hasta ser terminada.

Este proyecto se ha dividido en subtareas, las cuales son añadidas a distintas tarjetas, tal y como se muestra en la Figura 15.

En este proyecto se han generado primero 2 columnas, las cuales corresponden al entorno del cliente (Frontend) y al entorno del servidor (BackEnd).

Tras empezar una nueva tarea, dicha tarea se pasa a la columna de In process para indicar que se está trabajando en ella y tras su finalización se mueve a la columna de Done, para indicar que se encuentra finalizada.

Al final del proyecto, todas las tareas se encuentran en el apartado de Done.

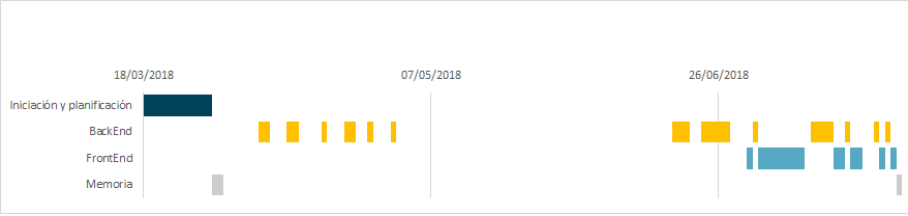
5.2 CONTROL DE VERSIONES GITHUB Y LICENCIA

Este es un punto importante a la hora de generar cualquier proyecto. Un sistema de control de versiones te permite gestionar el código de la aplicación de tal manera que se pueda volver a una versión anterior en caso de fallo de la versión que se está generando.

Para este caso se ha utilizado Github [11], un servicio que proporciona este control de forma gratuita y que con la versión de estudiantes, permite disponer de un repositorio privado para almacenar el código hasta el momento en que se pretenda liberar.

Otro punto a tener en cuenta a la hora de generar un proyecto, es comprobar qué licencia es la que está más acorde con el desarrollador. En este caso y como se ha hablado en otros puntos del proyecto, se pretende liberar el código para su posterior uso o modificación. Por ello, se ha decidido utilizar la licencia GPL [2] ya que permite que cualquiera pueda modificar el código pero le obliga a poner esas modificaciones con licencia GPL a su vez. Y esta licencia, permite también la venta de la aplicación.

5.3 DIAGRAMA DE GANTT



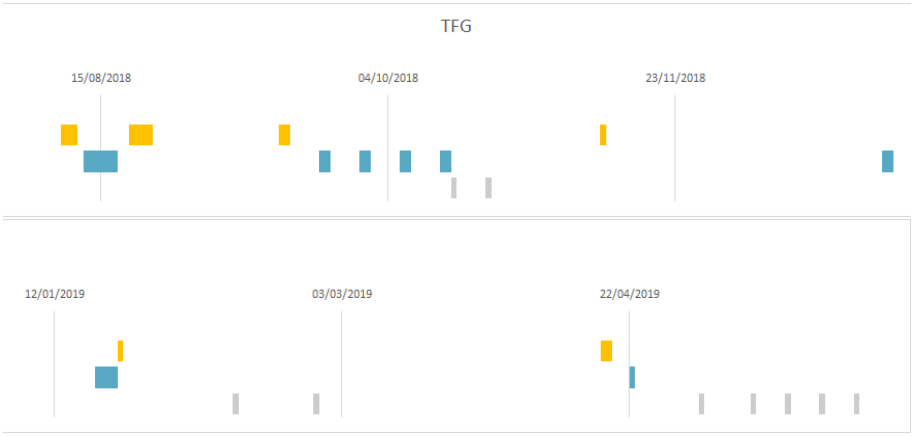


Figura 16: Diagrama de Gantt

Durante el proceso del proyecto, se han ido apuntando las tareas que se han ido realizando con su correspondiente tiempo, Por ello se proporciona un diagrama de Gantt en la Figura 16 que muestra dicho transcurso subdividido en las 4 tareas que de describen a continuación. En el sección de Apéndice, se proporciona otra versión del diagrama de Gantt, que muestra el desglose de tareas realizadas más en detalle.

El proyecto se ha dividido en 4 aparados principales:

- Iniciación y planificación
- BackEnd
- FrontEnd
- Memoria

Las tareas realizadas al principio del proyecto, han sido Iniciación y planificación, y Memoria. Con estas tareas, se han definido los criterios del proyecto, tanto las tecnologías como los puntos clave con respecto a la competencia. Recolectando la información necesaria para analizar y preparar el desarrollo con respecto a otras aplicaciones y los objetivos que se tienen para este proyecto.

A continuación, se han desarrollado el BackEnd y el FrontEnd, pasando por realizar la memoria hasta la finalización de este trabajo.

El porcentaje de tiempo que se ha dedicado a cada tarea, se ha definido en la siguiente Figura 17:

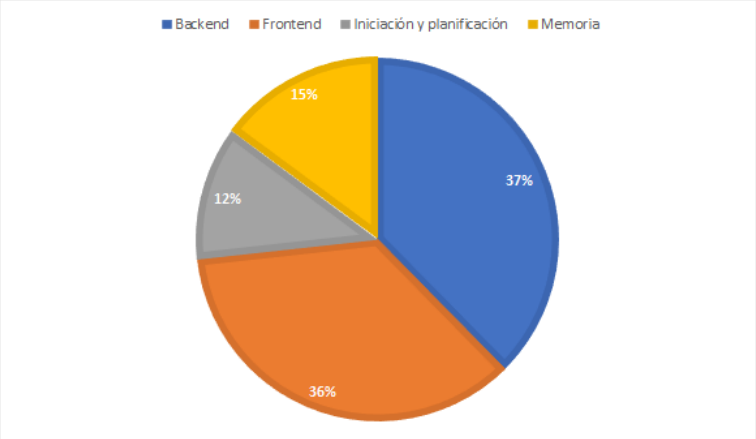


Figura 17: Diagrama de dedicación a tareas

El tiempo total dedicado a este proyecto ha sido de 287 horas aproximadamente.

Con respecto al diagrama de Gantt de la Figura 16, se pueden apreciar los puntos de inactividad del proyecto y los días de realización de trabajo. Estos puntos de inactividad se van a explicar a continuación, pues se creen relevantes a la hora de realizar un trabajo de este tipo. Como dato a tener en cuenta, me he encontrado en el mundo laboral durante la realización del proyecto. Por lo tanto, no se ha podido realizar el proyecto seguidamente.

- Del 1 de mayo al 18 de Junio, se realizó un parón por motivos de estudio para los exámenes de ese cuatrimestre.

- Del 23 de Agosto al 15 de Septiembre por motivo de exámenes del final de curso, se tuvo que realizar otro parón.

- Del 11 de Noviembre al 19 de Enero aproximadamente, no se pudo realizar la continuación del proyecto, por disponer de una alta carga de trabajo y por las vacaciones de Navidad.

- Por último, del 25 de Febrero al 16 de Abril, no se ha continuado por la realización de un viaje, tanto personal como otro a continuación por parte de la empresa a la que pertenezco.

5.4 IMPLEMENTACIÓN

Siguiendo el modelo de arquitectura del capitulo 4, se ha empezado la creación del proyecto. Lo primero que se ha realizado ha sido el

servidor, de tal manera que el cliente tuviera información de la que nutrirse para mostrarla de una manera correcta y precisa.

A continuación se muestra el desarrollo de ambos sistemas.

5.4.1 *Desarrollo del servidor*

Tal y como se ha visto en de Microservicios 4.2.1, se ha implementado cada uno de los Endpoint de la API, identificando qué método o petición [14] utilizan y cual es su identificador de llamada exacto en el servidor.

A continuación se va mostrar un ejemplo de Endpoint de la API, en concreto el microservicio de 'login'.

Cuadro 8: Acceso a Administrador y Monitor

Método	Servicios	Descripción
POST	/api/login	Permitirá al usuario loguearse dentro de la API y poder realizar operaciones dentro de ésta.

Para más información sobre los demás Endpoints que se han generado, visualizar el Anexo.

También hay que tener en cuenta que la aplicación se encontrara limitada dentro de los permisos del usuario que ejecute esta API. Por lo tanto se recomienda que dicho usuario disponga como mínimo de permisos de lectura sobre los ficheros más importante a la hora de la recopilación de información del sistema y los comandos que se vayan a ejecutar a través de los botones personalizados. También se proporciona una forma de ejecutar comandos como el usuarios Root, pero no se recomienda su utilización a excepción de su necesidad.

5.4.1.1 *Lenguaje de programación*

El lenguaje Javascript junto al entorno en tiempo de ejecución llamado Nodejs [12] ha sido elegido para este propósito porque es actualmente uno de los más usados por su alta escalabilidad y facilidad de desarrollo. La versión de Nodejs que se va a usar es la v8 o Superior,

por el hecho de que es la versión más estable actualmente y la que va a recibir el soporte a largo plazo (LTS). Lo que implica que mejorará en seguridad y en funcionalidad. A su vez es una tecnología con una gran comunidad.

Se han desechado tecnologías como Java o PHP porque aun siendo una tecnologías con muchos años de experiencia y desarrollo, no entran dentro del modelo de tecnología para el desarrollo ágil aunque en sus ultimas versiones están entrando dentro de este modelo y su comunidad aún siendo estable, no es una que esté en alto crecimiento.

5.4.1.2 *Base de datos*

Para el uso de la base de datos, se ha decidido utilizar una base de datos NoSQL. La razón de su uso, es que los datos que la aplicación va a generar no son del tipo que requieran relaciones y esta base de datos, a parte de no usar relaciones, es más flexible a la hora de almacenar datos con formato variable o sin formato y es una tecnología que es muy útil para guardar datos históricos. A su vez, es recomendable que la base de datos sepa gestionar las informaciones con el modelo JSON, pues es un modelo de intercambio más usado por internet por su simplicidad y gestión.

Por lo tanto, se va a usar MongoDB [15] como base de datos NoSQL pues trabaja con objetos JSON y es adaptativa a la hora de introducir nueva información y escalarla. También siendo una aplicación con pocos datos relacionados, es una buena opción para este punto. También es una base de datos altamente usada en estos momentos para guardar multitud de información histórica, cumpliendo así los requisitos.

Otras bases de datos se han desechado, pues esta es la más ampliamente usada a su vez que es la que más se adapta a las necesidades de la aplicación. Uno de los puntos más importantes de esta base de datos, que la mayor parte no comparte, es el tratamiento de objetos que esta base de datos gestiona, pues permite entre otras cosas, el uso de objetos anidados.

5.4.1.3 *Framework*

Durante el desarrollo del servidor, se ha utilizado el siguiente framework principal para trabajar:

Express [5], el cual se definen a continuación alguno de sus beneficios:

- Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL (rutas).
- Establecer ajustes de aplicaciones web, como qué puerto usar para conectar.
- Añadir procesamiento de peticiones "middleware" adicional en cualquier punto dentro de la tubería de manejo de la petición.

Esto, lo que permite, es facilitar al desarrollador la manera de agilizar alguno de los aspectos de la aplicación.

5.4.1.4 *Bibliotecas*

En este caso se va a destacar una de las bibliotecas más importantes que se han utilizado dentro de la aplicación.

En este caso se ha utilizado "systeminformation" [22], una biblioteca que se encarga de recopilar la información del sistema, incluyendo temperaturas, usuarios, discos... Dicha librería esta desarrollada por "plusinnovations" con licencia MIT, y es muy completa.

A pesar de ser una biblioteca muy completa, se encuentra en desarrollo, lo que ha causado que varias de las funcionalidades deseadas no se encuentren o en su defecto, no estuvieran terminadas. Por ello se ha apoyado al proyecto, lanzando una petición al usuario con modificaciones de código, las cuales el usuario ha aceptado y ha subido al repositorio.

Dichas modificaciones iban relacionadas con temas de devolver información relevante a usuarios con pocos permisos dentro del sistema. De tal manera que dichos usuarios no podrían actuar de monitores del sistema, al no poder recibir información de la librería.

Otra biblioteca que se ha utilizado, ha sido la que gestiona el sistema de eventos, una biblioteca que simula el sistema de crontab de linux. Crontab es un administrador que ejecuta procesos a intervalos regulare según los tiempos definidos.

Realizándolo de esta manera y no usando el propio sistema de eventos de linux, se pretende que en el futuro, se porten las funcionalidades de este servidor a windows.

5.4.1.5 *Seguridad*

Aun pudiendo ser suficiente con el uso del protocolo SSL mencionado en el apartado de Arquitectura 4, la seguridad para mantener la información del usuario es importante, por lo tanto, se cifran todas las posibles contraseñas que se comunican con el cifrado SHA512 que genera un HASH que no puede ser descifrado. De tal manera que se agrega una segunda capa de seguridad al sistema en caso de que en un futuro, el protocolo SSL, se vuelva inseguro.

Otro punto de seguridad que se ha agregado, ha sido la imposibilidad de un ataque a fuerza bruta, ataque por el cual, un usuario prueba a obtener una clave de acceso probando todas las combinaciones posibles hasta encontrar aquella que le permite entrar. Esto se ha solventado limitando a aquellos usuarios que intenten hacer demasiadas peticiones sobre el servicio de acceso o login, de tal manera que si alguien intenta demasiados accesos a ésta, se le limitan las peticiones y el tiempo de acceso a las mismas aumenta.

5.4.1.6 *Logs*

Un sistema generador de logs, que van registrando toda actividad tanto de eventos que se han lanzado o también toda actividad que ha generado el usuario.

De tal manera, que en caso de error o ataque al sistema, este quede registrado para una posterior depuración.

5.4.2 *Desarrollo del cliente*

A continuación, se explica qué decisiones se han tomado y cómo ha evolucionado el desarrollo de la aplicación cliente. Qué técnicas, bibliotecas y frameworks se han usado para su desarrollo.

5.4.2.1 *Plataforma*

Durante el desarrollo del Cliente, se ha utilizado la plataforma Android como el sistema operativo al cual proporcionar soporte de momento, por la cantidad de clientes que hay disponibles en la plataforma.

La versión y API elegidas corresponden a la Versión Oreo de Android 8.0 lanzada el 8 de junio de 2017 con la API 26, siendo esta API y versión, la mínima obligada por Google para los nuevos desarrollos. El número 26 de la API, corresponde al nivel o las veces que esta ha sido actualizada para interactuar con el móvil.

La razón de usar esta versión (8.0), es que aun no siendo la más usada, ni tampoco más nueva (pues la versión más usada es justo la anterior), esta irá tomando terreno en el mercado y a su vez, permite dar soporte a las versiones posteriores que también llenarán los dispositivos móviles. Aparte, es la recomendada por Google pues las versiones anteriores están perdiendo el soporte.

5.4.2.2 *Lenguaje de programación*

Para el lenguaje de programación, se ha escogido Typescript junto al framework Angular [13] y el framework Ionic [9] como punto de partida para desarrollar en dispositivos móviles.

Typescript es un lenguaje de programación que es un superconjunto de JavaScript. En este caso Typescript agrega tipos estáticos y objetos basados en clases, además de lo que tiene Javascript; lo que hace que sea mucho más fácil de programar.

Angular es un framework que actualmente se está usando en profundidad dentro del mundo web, al integrar el sistema de Modelo, Vista, Controlador [3], proporcionando un esquema para los proyectos y a la vez utiliza HTML y CSS para dar el entorno visual.

Para la portabilidad a móvil, se usa el framework Ionic, gracias a su gran versatilidad para portar las aplicaciones a este entorno; estas pueden ser entre plataformas móviles en general o web.

De tal manera que más adelante se puedan realizar algunas modificaciones y hacerla perfectamente compatible para un entorno como Ios de Apple o desarrollar con muy pocas modificaciones, la misma aplicación en la web.

5.4.2.3 *Frameworks*

Angular, como se ha comentado, es un lenguaje que está muy orientado a los entornos web, pero existen Frameworks que dan soporte al desarrollo de aplicaciones móviles con esta tecnología.

Existen varios Frameworks en el mercado que pueden llevar a este resultado.

Uno de ellos es NativeScript, el cual se testeó durante el principio de este desarrollo y se comprobó que es un Framework que aun sabiendo que está en desarrollo, se encuentra falto de un montón de funcionalidades como el soporte multiplataforma o diseño adaptativo a todos los dispositivos, a la vez que documentación para su desarrollo.

Tras comprobar que no era viable el desarrollo con dicho Framework, se pasó a usar Ionic, actualmente uno de los más utilizados dentro del mercado y que más características y funcionalidades abarca.

Trabajando con él se ha comprobado que es bastante versátil y permite su desarrollo no solo desde entornos móvil, sino que también permite el desarrollo dentro de cualquier navegador web. Lo que ofrece una alta capacidad para desarrollar dentro de cualquier entorno.

5.4.2.4 *Aplicación Final*

Este apartado describe la aplicación ya finalizada, mostrando alguna captura de como ha quedado el apartado visual.

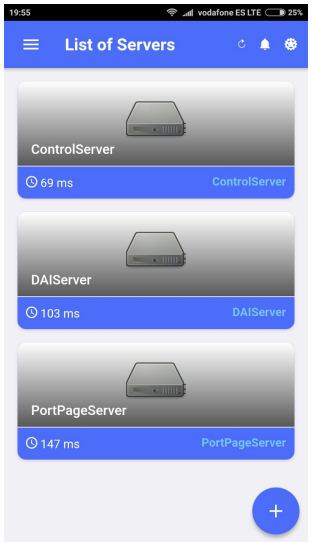


Figura 18: Lista de Servidores

La Figura 18 representa la pantalla final desde donde se permitirá ver todos los servidores agregados a la aplicación, permitir acceder a cada uno de ellos y agregar nuevos.

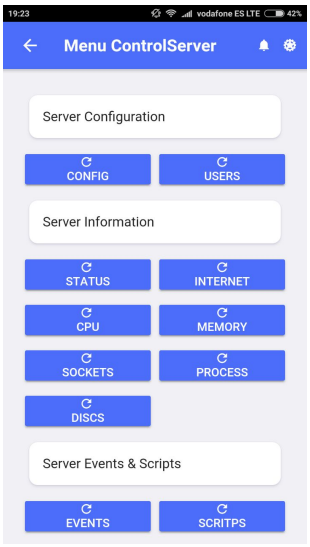


Figura 19: Menu Servidor

La Figura 19 representa el panel final desde donde se mostraran todas las opciones dentro del servidor.

6

PRUEBAS DE VALIDACIÓN

6.1 TEST API

Para realizar una correcta comprobación de que la API que se ha generado en el entorno del servidor,funciona correctamente, se ha realizado un programa de testeo [16] realizado con las siguientes tecnologías de NodeJS, Chai [4] y Mocha [6].

Dichas tecnologías permiten hacer un chequeo de cada servicio o Endpoint de la API. Cada endpoint se ha testeado tanto transmitiendo datos erróneos como datos correctos, de tal manera que se compruebe que la aplicación devuelve los datos correspondientes a su funcionamiento.

Ejemplo de Test de Endpoint (los datos internos son de ejemplo):

Cuadro 9: Endpoint testeado /api/login

Prueba	Datos de entrada	Resultado
Login Correcto	{username: 'username', login: 'password'}	Correct Login + token login
Usuario incorrecto	{username: 'user', login: 'password'}	Incorrect Login
Contraseña incorrecta	{username: 'username', login: 'pass'}	Incorrect Login

Se han hecho diversas pruebas con los distintos Endpoints, tal y como se muestra en el ejemplo de arriba, proporcionando así, una seguridad de que el servicio funciona correctamente.

Para testear la aplicación móvil, se debe comentar que la tecnología usada (Ionic), permite el despliegue o ejecución de la aplicación tanto en un entorno Web como en los entornos móviles.

Esta aplicación ha sido testeada primero en un entorno Web, teniendo en cuenta que el entorno Web esta limitado en ciertas funcionalidades como por ejemplo el uso de notificaciones. De tal manera que a la hora del desarrollo de la aplicación, se pudiera realizar un desarrollo rápido sin necesidad de compilaciones en un entorno móvil.

Dicho lo anterior, se han realizado pruebas en dicho entorno siguiendo el mismo modelo de la tabla 9 pero en este caso de forma manual. Comprobando en cada pantalla que se ha generado, que la respuesta recibida es la correcta y que si se proporcionan datos erróneos u ocurre algún evento extraño, se proporcione un error visual al usuario.

Algunos de los errores encontrados a la hora de este testeo, han sido la visualización errónea de mensajes mostrados por pantalla para advertir al usuario de la existencia de errores en el punto donde se encuentra, como por ejemplo que el servidor al que se intenta conectar, está desconectado y no funciona.

Tras el testeo en el entorno web, se ha pasado a un desarrollo centrado en móvil para terminar todas aquellas funcionalidades que están limitadas en Web.

Para el entorno móvil, se han generado pruebas con respecto a las funcionalidades faltantes, y para las ya testeadas, se han realizado alguna prueba puntual para comprobar el correcto funcionamiento en la plataforma.

7

CONCLUSIONES

Este proyecto ha consistido en el desarrollo de un sistema cliente-servidor compuesto por una aplicación móvil y una aplicación servidor que en su conjunto permiten capturar datos del sistema como el uso de CPU y mostrarla en el cliente; a su vez permite la creación de eventos y lanzamiento de comandos. Dichos servidores se encuentran conectados a través de una API que los gestiona y entre otras, se encarga de monitorizar el servidor donde esté instalado guardando la información para crear un histórico.

Este proyecto ha sido de interés propio y ha sido interesante realizarlo y ponerlo en marcha. La falta de herramientas de este carácter en el mercado móvil es notable y están bastante limitadas. Por ello se ha querido proporcionar otra alternativa dentro de este mercado con las ultimas tecnologías de éste.

Ha habido algunas dificultades técnicas que se han ido sobrepasando e incluso se han realizado aportes en el desarrollo de librerías abiertas al publico para la tecnología NodeJs.

Alguna funcionalidad se ha tenido que posponer para una próxima versión de la herramienta, pues las herramientas y frameworks que se han utilizado, no soportan la funcionalidad que se querían proporcionar (SSH).

Para posteriores mejoras, se ha abierto la posibilidad de dar soporte a nuevas opciones de monitorización e incluso el dar soporte a las nuevas tecnologías cloud como puede ser Amazon y sus servicios de monitorización x-Ray.

BIBLIOGRAFÍA

[1] Anónimo. "Google Commerce Ltd". En: (2018).

[2] Anónimo. "Licencias – licencia MIT vs GPL". En: (2018).

[3] Anónimo. "Modelo–vista–controlador". En: (2018).

[4] Anónimo. "Chai". En: (2019).

[5] Anónimo. "Expressjs". En: (2019).

[6] Anónimo. "Mocha". En: (2019).

[7] Anónimo. "Modelo–cliente-servidor". En: (2019).

[8] BBVAOpen4U. "Api Rest". En: (2016).

[9] "Build amazing apps in one codebase, for any platform, with the web." En: (2018). https://ionicframework.com/.

[10] Jeff Delaney. "Ionic Native With Firebase FCM Push Notifications". En: (2018).

[11] "Discover interesting projects and people to populate your personal news feed." En: (2018).

[12] Node.js Foundation. "NodeJS". En: (2009).

[13] Google. "Angular". En: (2016).

[14] Mangel. "Peticiones HTTP". En: (2018).

[15] Inc. MongoDB. "MongoDB". En: (2009).

[16] Rafael Márquez. "Testeando JavaScript con Mocha y Chai". En: Version 1 (2017).

[17] Steven Máñez. "Backend Server". En: (2018).

[18] Symantec. "SSL". En: (2019).

[19] Alexei Vladishev. "Zabix". En: (2018).

[20] ibm. "The components of a URL". En: (2019).

[21] Daniel Luna y otros. "Nagios". En: (2018).

[22] plusinnovations. "Systeminformation". En: (2018).

Parte I

ANEXO

A

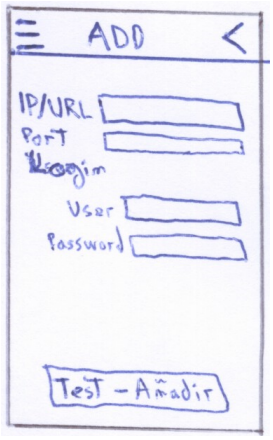
ANEXOS

En los siguientes apartados, se van a añadir todas aquellas informaciones que aun aumentando el conocimiento sobre el TFG, no son totalmente requeridas para una mejor comprensión de este:

A.1 PROTOTIPADO EN PAPEL

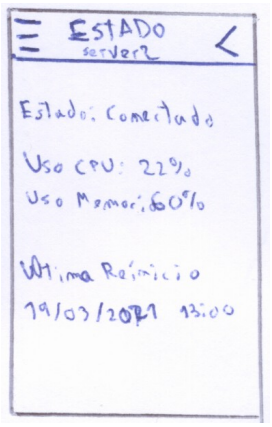
Aquí se muestran todas las demás capturas sobre el prototipo en papel y su finalidad:

A.1.1 *Añadir Nuevo Servidor*



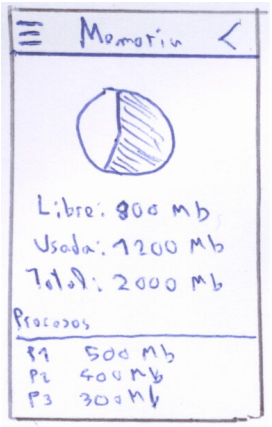
Desde esta pantalla se podrá agregar un nuevo servidor, con las correspondientes credenciales de este.

A.1.2 *Estado del servidor*



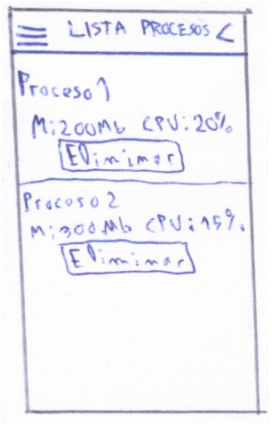
Este panel mostrara información genérica del estado del servidor y alguna información extra que puede ser de utilidad.

A.1.3 *Visualización de datos*



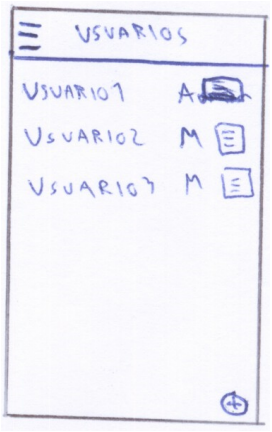
Desde esta pantalla se podrá visualizar la memoria del sistema con los procesos que más consumen.
Esta pantalla será similar a las pantallas de visualización de CPU, Internet... pero con distintas gráficas o distinta manera de visualizar los datos. De tal manera que cada uno disponga de la mejor manera de visualización.

A.1.4 *Procesos del sistema*



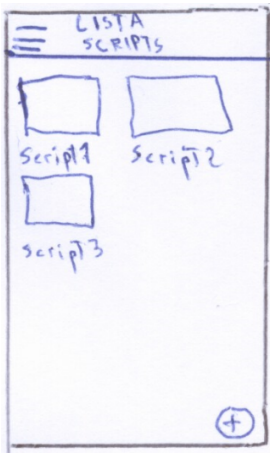
Esta pantalla mostrará los datos de cada proceso en el sistema y permitirá detener cada uno si se solicita.

A.1.5 *Usuarios*



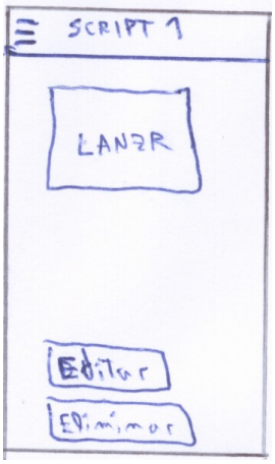
Esta pantalla permitirá visualizar cada usuario dentro de la API del servidor y su creación y modificación.

A.1.6 *Lista de Scripts*



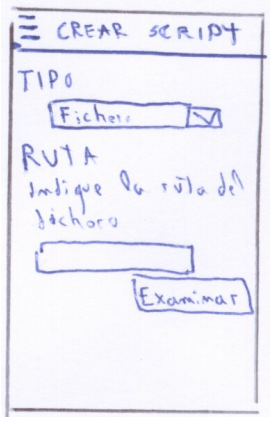
También definidos como los botones personalizados. Este panel permite ver todos los botones personalizados, permitiendo añadir nuevos botones.

A.1.7 *Lanzamiento de Scripts*



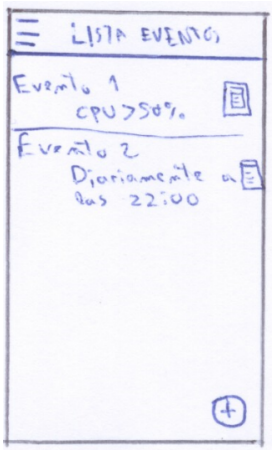
Panel que permite ejecutar el script seleccionado.
Si el script ha sido creado con la opción de ejecución con permisos de Root, se solicitará una contraseña que previamente debe haber sido configurada en el servidor de Backend.
Dicha opción aun siendo una opción segura, se recomienda que no se use. Y para ello se recomienda que se le proporcione al usuario los permisos necesarios para la ejecución de los comandos que esté vaya a necesitar usar con la aplicación.

A.1.8 *Crear Script*



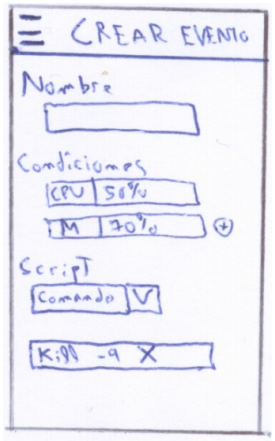
Permite crear un nuevo script.
Dicho script se escribirá en el campo de texto proporcionada.
Dentro se encuentra también la opción de ejecutar dicho comando como Root dentro del sistema y posteriormente en el lanzamiento de dicho script, se solicitará una contraseña poder lanzar el comando de Root.

A.1.9 *Lista de eventos*



Panel que permite visualizar los eventos que se han preparado con sus respectivas condiciones de ejecución.

A.1.10 *Crear Evento*



Permite crear nuevos eventos para que se ejecuten en caso de que se cumpla una condición. Las opciones de las condiciones, serán por ejemplo, CPU, Memoria...

A.2 DESCRIPCIÓN DE APIS

A continuación se muestran todos los demás servicios API que se gestionan dentro de la aplicación. Teniendo en cuenta todos los puntos de entrada y de qué métodos se utilizan. También se indica qué usuarios tienen acceso a dicho apartado, teniendo en cuenta que la aplicación dispone de 2 usuarios, Monitores y Administradores.

A.2.1 *Servicios de configuración*

El cuadro 10 muestra los servicios de monitorización correspondientes a la gestión de la configuración del sistema, incluyendo el mostrar como se encuentra configurado y su modificación.

Cuadro 10: Endpoints config

Método	Servicios	Descripción
GET	/api/config	Permitirá recopilar la configuración del sistema.
POST	/api/config	Permitirá configurar la aplicación con algunos parámetros de interés: Cada cuanto tiempo recogerá información la aplicación y su tiempo de expiración.

A.2.2 *Servicios de gestión de usuarios*

El cuadro 11 muestra los servicios de monitorización correspondientes a la gestión de usuarios dentro de la plataforma, incluyendo el listado de estos, su creación y borrado.

Cuadro 11: Endpoints users

Método	Servicios /api/users/	Descripción
GET	/api/users	Lista los usuarios de la aplicación y sus roles
POST	/api/users	Creación de usuarios.
DELETE	/api/users/:id	Borrar un usuario creado por el admin

A.2.3 *Servicio de comprobación de conexión*

El cuadro 12 muestra el servicio de monitorización correspondientes a lo que sería un ping realizado desde cualquier ordenador a otro.

Cuadro 12: Endpoints ping

Método	Servicios	Descripción
GET	/api/ping	Indica si hay respuesta del servidor

A.2.4 *Servicios de visualización de la información*

El cuadro 13 muestra los servicios utilizados para devolver información relacionado al sistema que se ha generado, tanto devolver el uso de CPU, Memoria...

Cuadro 13: Endpoints look

Método	Servicios /api/-look/	Descripción
POST	/api/look/cpu	Muestra el uso CPU Total y que 5 procesos la cargan más.
POST	/api/look/mem	Muestra el uso de Memoria Total y que 5 procesos la cargan más.
POST	/api/look/network	Muestra el uso de la red, cuántos datos se consumen y qué procesos lo hacen.
GET	/api/look/disk	Muestra el uso del disco, espacio libre y espacio ocupado.
GET	/api/look/status	Muestra los datos de los últimos reinicios, apagado del sistema, uso de CPU, Uso de Memoria....
GET	/api/look/ processList/ :order/:nProcess	Muestra la lista de procesos ordenados por el parámetro order y el número que se desee recibir.

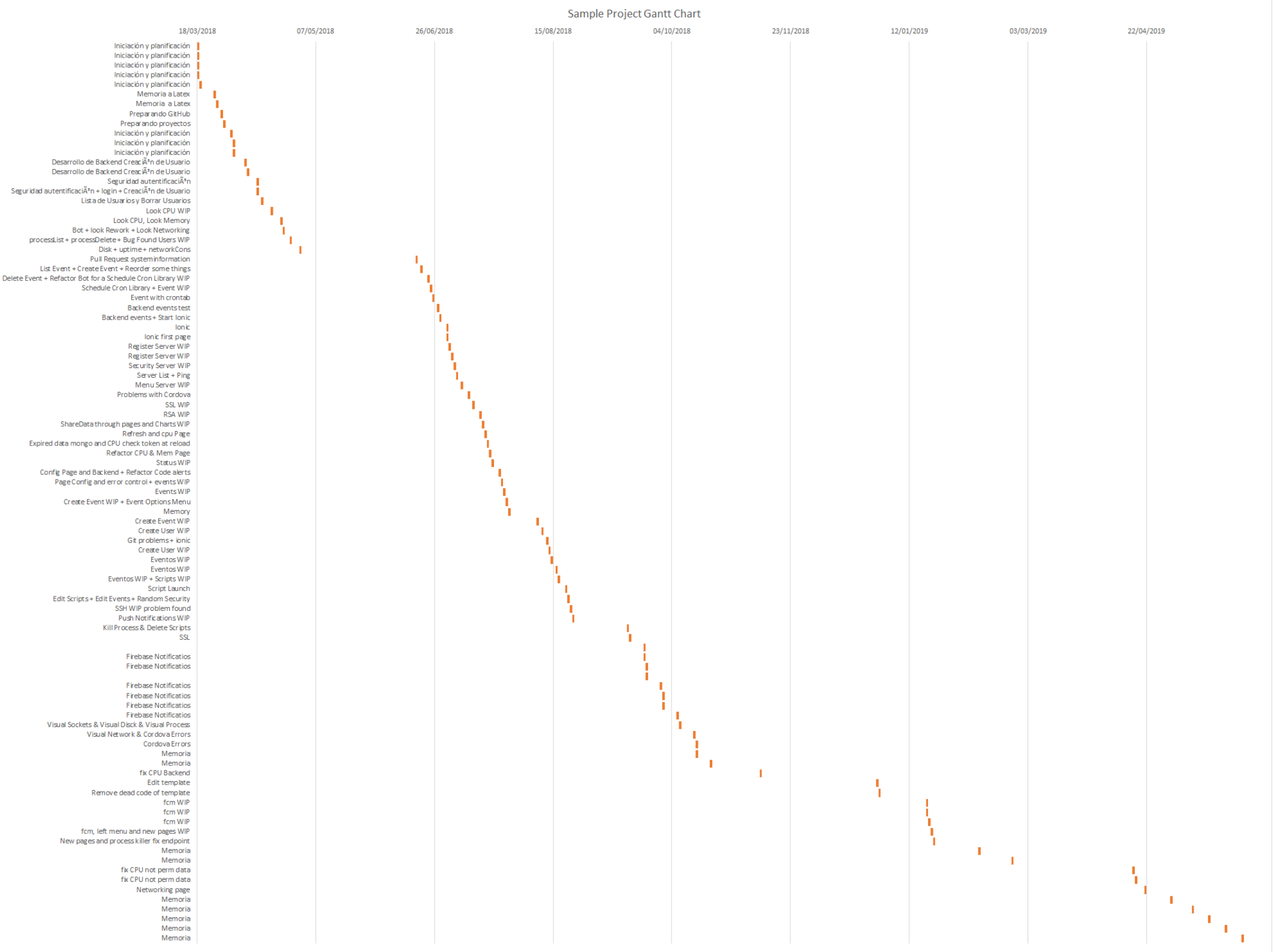
A.2.5 *Servicios de Comandos*

El cuadro 14 muestra los servicios utilizados para la creación, listado y ejecución de eventos y scripts-

Cuadro 14: Endpoints run

Método	Servicios /api/event/	Descripción
GET	/api/run/events	Muestra los eventos programados.
POST	/api/run/events	Creación de Eventos configurables para que se ejecuten en determinados momentos deseados. Dicho evento será un script a ejecutar. Ejemplos: Uso de CPU superior a X Hora determinada
DELETE	/api/run/events/:id	Borrado de un evento programado.
GET	/api/run/scripts	Lista los scripts dentro del sistema.
POST	/api/run/scripts	Creación de script, subiendo un comando.
DELETE	/api/run/scripts/:id	Borrar un script que se haya creado.
POST	/api/run/launch	Ejecución de script, guardado en el sistema.
POST	/api/look/processKiller	Permite cerrar un proceso ejecutándose en el sistema.

A.3 DIAGRAMA DE GANTT


Sample Project Gantt Chart
18/03/2018
07/05/2018
26/06/2018
15/08/2018
04/10/2018
23/11/2018
12/01/2019
03/03/2019
22/04/2019
Iniciación y planificación
Iniciación y planificación
Iniciación y planificación
Iniciación y planificación
Iniciación y planificación
Memoria a Latex
Memoria a Latex
Preparando GitHub
Preparando proyectos
Iniciación y planificación
Iniciación y planificación
Iniciación y planificación
Desarrollo de Backend CreaciÃ³n de Usuario
Desarrollo de Backend CreaciÃ³n de Usuario
Seguridad autentificaciÃ³n
Seguridad autentificaciÃ³n + login + CreaciÃ³n de Usuario
Lista de Usuarios y Borrar Usuarios
Look CPU WIP
Look CPU, Look Memory
Bot + look Rework + Look Networking
processList + processDelete + Bug Found Users WIP
Disk + uptime + networkCons
Pull Request systeminformation
List Event + Create Event + Reorder some things
Delete Event + Refactor Bot for a Schedule Cron Library WIP
Schedule Cron Library + Event WIP
Event with crontab
Backend events test
Backend events + Start Ionic
Ionic
Ionic first page
Register Server WIP
Register Server WIP
Security Server WIP
Server List + Ping
Menu Server WIP
Problems with Cordova
SSL WIP
RSA WIP
ShareData through pages and Charts WIP
Refresh and cpu Page
Expired data mongo and CPU check token at reload
Refactor CPU & Mem Page
Status WIP
Config Page and Backend + Refactor Code alerts
Page Config and error control + events WIP
Events WIP
Create Event WIP + Event Options Menu
Memory
Create Event WIP
Create User WIP
Git problems + ionic
Create User WIP
Eventos WIP
Eventos WIP
Eventos WIP + Scripts WIP
Script Launch
Edit Scripts + Edit Events + Random Security
SSH WIP problem found
Push Notifications WIP
Kill Process & Delete Scripts
SSL
Firebase Notificatios
Firebase Notificatios
Firebase Notificatios
Firebase Notificatios
Firebase Notificatios
Firebase Notificatios
Visual Sockets & Visual Disck & Visual Process
Visual Network & Cordova Errors
Cordova Errors
Memoria
Memoria
fix CPU Backend
Edit template
Remove dead code of template
fcm WIP
fcm WIP
fcm WIP
fcm, left menu and new pages WIP
New pages and process killer fix endpoint
Memoria
Memoria
fix CPU not perm data
fix CPU not perm data
Networking page
Memoria
Memoria
Memoria
Memoria
Memoria