UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO
FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA
ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA Y DE SISTEMAS

TESIS

DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA IoT PARA LA GESTIÓN DE LOS CONTROLADORES SEMAFÓRICOS EN LA CIUDAD DEL CUSCO

Para optar al título profesional de:

INGENIERO INFORMÁTICO Y DE SISTEMAS

Presentado por:

Br. BENGIE NICK SERRANO QUISPE

Br. JOSE CARLOS HUALLPAMAITA QUISPE

Asesor:

LIC. JOSE MAURO PILLCO QUISPE

Co-asesor:

ING. JUAN PABLO VIZCARDO ZUÑIGA

CUSCO-PERÚ

2018

DEDICATORIA

Este trabajo está dedicado a las personas, que, a lo largo de mi vida han sido un aliciente para ser una mejor persona. Se la dedico a mis padres y a mis hermanos, personas que me aconsejan, me apoyan y me motivan a seguir adelante.

Bengie Nick Serrano Quispe.

A mis padres Bonifacio y Ricarda por su esfuerzo y sacrificio que los caracteriza siempre y por el valor mostrado en todo momento para salir adelante.

A mis Hermanos Washington, Edgar, Mary Luz y Rómulo por su apoyo constante en todo momento.

A mis sobrinas Fiorella, Andrea, Valentina y Michelle que son siempre el motivo de alegría para la familia.

José Carlos Huallpamaita Quispe.

AGRADECIMIENTOS

Los agradecimientos están dirigidos a todas las personas y a la entidad que hicieron posible la realización exitosa de esta investigación, principalmente agradecemos a nuestros padres y hermanas por el apoyo y motivación entregada.

A nuestro asesor Lic. José Mauro Pillco Quispe, por su permanente soporte y dedicación manifestados en este proyecto de investigación, así como a lo largo de todo el desarrollo de nuestra preparación para este proyecto.

A la plana docente de la Universidad Nacional de San Antonio Abad del Cusco que tuvimos el privilegio de tener como maestros, que nos dio la oportunidad de compartir experiencias y contar con herramientas irreemplazables que nos acompañarán a lo largo de nuestra vida profesional.

A nuestro co-asesor Ing. Juan Pablo Vizcardo Zúñiga, por habernos permitido colaborar en la realización de un proyecto real en el cual pudimos poner a prueba toda nuestra investigación.

RESUMEN

En la actualidad, la ciudad del Cusco se enfrenta a la rápida expansión demográfica de la zona urbana y una creciente demanda turística, lo cual, contribuye al aumento de vehículos del parque automotor, agravando los problemas en el ordenamiento vehicular y la falta de sincronización inmediata de los semáforos, genera malestar entre los peatones y usuarios vehiculares por el impedimento de llegar a sus destinos de manera oportuna. Dicha dificultad, implica hacer necesaria la implementación y optimización de los mecanismos que permitan realizar una mejor regulación de la circulación de vehículos, bicicletas y peatones.

La presente tesis propone el diseño e implementación de una plataforma de software en base a los conceptos de IoT (*Internet of Things* o Internet de las cosas), capaz de gestionar en tiempo real la configuración las fases, horarios, ubicación, envío de configuración especial, sincronización de fecha e información de cada intersección; además de permitir la visualización del estado de cada controlador semafórico conectado por parte del personal operario. Los controladores semafóricos quedarían bajo administración de la Municipalidad Provincial del Cusco logrando interactuar, determinar, predecir y actuar en función del contexto de los datos recolectados; para ello se planteó diseñar e implementar una arquitectura que permita el flujo de datos bidireccional en dos fases; la primera con mecanismos que permita a cualquier controlador semafórico ya sea propio o de terceros que cumplan con las especificaciones del protocolo de comunicación MQTT**,** conectarse y enviar datos, cifrados mediante el protocolo TLS, hacia un servidor de red, mientras que la segunda fase empezará en la recepción de datos por parte servidor de red o *bróker* (al que llamaremos *MQTT-Server* implementado con *Mosca-MQTT* en base a Node.js*),* a partir de este punto se realizará dos procesos en paralelo que completan el flujo de datos de la arquitectura planteada.

Palabras Clave*: Plataforma IoT, Gestión, Semaforización, Controlador Semafórico*.

ABSTRACT

Currently, the city of Cusco faces the rapid demographic expansion of the urban area and a growing tourist demand, which contributes to the increase of vehicle fleet, aggravating the problems in the ordering of vehicles and the lack of synchronization of the traffic lights, which generates discomfort among pedestrians and vehicular users due to the impediment of reaching their destinations in a timely manner. This difficulty implies the need to implement and optimize the mechanisms that allow a better regulation of the circulation of vehicles, bicycles and pedestrians.

This thesis proposes the design and implementation of a software platform based on the concepts of IoT (Internet of Things or Internet of Things), capable of managing in real time the configuration phases, timetables, location, sending special configuration, synchronization of date and information of each intersection; in addition to allowing the visualization of the status of each connected traffic light controller by the operating personnel. The traffic controllers would be under the administration of the Provincial Municipality of Cusco, interacting, determining, predicting and acting according to the context of the data collected; For this purpose, it was proposed to design and implement an architecture that allows bidirectional data flow in two phases; the first with mechanisms that allow any traffic light controller either own or of third parties that comply with the specifications of the MQTT communication protocol, connect and send data, encrypted using the TLS protocol, to a network server, while the second phase will begin in the reception of data by a network server or broker (which we will call MQTT-Server implemented with Fly-MQTT based on Node.js), from this point two parallel processes will be performed that complete the data flow of the architecture raised.

Keywords: *IoT Platform, Management, Signaling, Traffic Light Controller.*

ÍNDICE GENERAL

INDICE DE TABLAS

INDICE DE FIGURAS

DICCIONARIO DE ACRÓNIMOS

API*: Application Programming Interface*.

BIND*: Berkeley Internet Name Domain*.

BSON*: Binary JSON.*

CSP*: Content Security Policy*.

DOM*: Document Object Model*.

GUI*: Graphics User Interface*.

HTTP*: Hypertext Transfer Protocol*.

HTTPS*: Hypertext Transfer Protocol Secure*.

IoT: *Internet of the Things*

JSON*: JavaScript Object Notation.*

LPWAN*: Low Power WAN*.

MAC*: Media Access Control*..

MVC*: Model View Controler*.

MPC: *Municipalidad Provincial del Cusco.*

MQTT*: Message Queue Telemetry Transport*.

NPM*: Node Package Module*.

M2M:*Machine to Machine*

NCS: *Networked Control System*

PLC: *Programmable Logic Controller.*

OASIS: *Organization for the Advancement of Structured Information Standards.*

QoS*: Quality of Service.*

RAS:*Remote Access Server***.**

RFID: *Radio Frequency Identification.*

SNTP*: Simple Network Time Protocol.*

SOA: *Service Oriented Architecture.*

SSL: *Secury Sockets Layer.*

TI*:* Tecnologias de Informacion.

TLS*:Transport Layer Security.*

WAN*: Wide Area Network.*

WSN*: Wireles Sensor Network.*

TERMINOS Y ABREVIATURAS

API: Conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Berkeley Internet Name Domain (BIND): Es el servidor de DNS más comúnmente usado en Internet, especialmente en sistemas Unix.

Binary JSON (BSON): Formato de Intercambio de datos usado para almacenamiento y transferencia en la base de datos MongoDB.

Content Security Policy (CSP): Es un estándar de seguridad informática introducido para evitar secuencias de comandos entre sitios, ataques de clickhacking y otros ataques de inyección de código.

Databinding: Sincronización automática de datos entre el modelo y la vista.

Document Object Model (DOM): Modelo de objecto para la representación de documentos.

Front End - Back End: Términos que se refieren a la separación de intereses entre una capa de presentación y una capa de acceso a datos.

Git: Programa para el control de versiones de un proyecto de software.

GUI: Interfaz Grafica de Usuario.

HTTP: Protocolo de Transferencia de hipertexto.

JavaScript Object Notation (JSON): Formato ligero de intercambio de datos.

Low Power WAN (LPWAN): Red WAN de baja potencia.

Media Access Control (MAC): Es el identificador único asignado por el fabricante a una pieza de hardware de red (como una tarjeta inalámbrica o una tarjeta Ethernet).

Model View Controler (MVC): Estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Message Queue Telemetry Transport (MQTT): protocolo de mensajeria basado en la publicacion y subscripcion.

Node Package Module (NPM): Es el manejador de paquetes por defecto para Node.js, un entorno de ejecución para JavaScript.

Prefetching: Es la captación previa de caché es una técnica utilizada por los procesadores de computadora para aumentar el rendimiento de ejecución al obtener instrucciones o datos de su almacenamiento.

Quality of Service (QoS): Rendimiento promedio de una red de telefonía o computadoras.

Sharding: Técnica que consiste en particionar los datos de una base de datos horizontalmente agrupándolos de algún modo que tenga sentido y que permita un direccionamiento más rápido.

Simple Network Time Protocol (SNTP): Protocolo simple de hora de red, es una aplicación de mantenimiento de hora que permite sincronizar el hardware en una red.

TCP/IP: Conjunto de protocolos de red en los que se basa internet y que permiten la transmisión de datos entre computadoras.

TI: Tecnologias de Informacion.

TypeScript: Es un superconjunto de JavaScript, que esencialmente añade tipado estático y objetos basados en clases.

WebSockets: Es una tecnología avanzada que hace posible abrir una sesión de comunicación interactiva entre el navegador del usuario y un servidor.

Wide Area Network (WAN): Red de computadoras que une varias redes locales.

Wireles Sensor Network (WSN): Red de Sensores Inalámbricos.

Wizard: Es un tipo de interfaz de usuario con una secuencia de cuadros de dialogo, que lleva al usuario a través de una serie de paso bien definido

CAPÍTULO I: GENERALIDADES

1.1 ASPECTOS GENERALES

1.1.1 Problema de investigación

1.1.1.1 Descripción del problema

Dentro de las funciones exclusivas de las municipalidades provinciales, que establece la Ley Orgánica de Municipalidades, específicamente en el numeral 1.1 del artículo 81° de la Ley 27972; está el de "Normar, regular y planificar el transporte terrestre, fluvial y lacustre a nivel provincial" (Ley-N°27972, 2003); igualmente, el numeral 1.3 del mismo artículo de la citada norma, establece como funciones específicas exclusivas de las municipalidades provinciales: "Normar, regular, organizar y mantener los sistemas de señalización y semáforos, y regular el tránsito urbano de peatones y vehículos" (Ley-N°27972, 2003). En tal sentido la municipalidad Provincial del Cusco mediante la Gerencia de Transito, Vialidad y Transporte, haciendo uso de esas facultades planifica e implementa mecanismos que ayuden en la mejor administración del tránsito vehicular y peatonal en la ciudad del Cusco y más aún cuando esta se enfrenta a numerosos desafíos en la mejora de la calidad de vida de sus ciudadanos; aspectos como el transporte, el ordenamiento vehicular, la cantidad de vehículos, el mal diseño de las vías, obras de mantenimiento realizadas en horas pico y la falta de sincronización de los semáforos es causa de malestar general entre los habitantes que no pueden movilizarse de forma ágil y segura de un lugar a otro, esto siempre ha sido un problema latente contribuyendo así de forma negativa en el desarrollo de la ciudad.

Por otro lado, los semáforos tradicionales son dispositivos del control del tránsito eléctricos y electrónicos cuya finalidad es controlar adecuadamente el tráfico vehicular motorizado, no motorizado y peatonal en intersecciones de calles y vías vehiculares en nuestra ciudad, a través de indicaciones luminosas de color rojo, verde y ámbar, programados mediante una configuración

horaria y divididas en diferentes fases tanto vehiculares, peatonales, de giro u otros que se puedan definir de acuerdo al contexto de la intersección. La regulación y control de la operación de los semáforos se realiza a través de unidades de control, llamados controladores semafóricos

La ciudad del Cusco carece de algún tipo de sistemas coordinados para la gestión de las intersecciones semafóricas distribuidas dentro de la ciudad, haciendo que muchas veces estos dispositivos no cumplan óptimamente con su función. La no implementación de este tipo de sistemas puede darse por diferentes factores políticos, administrativos, de recursos económicos u otros. Debido a este contexto se producen una serie de problemas para una mejor administración del tránsito, problemas que se describirán a continuación:

- Actualmente en cada intersección semafórica existen diferentes tipos de controladores semafóricos; más allá de que esta variedad produce un problema de compatibilidad para una solución de sistemas coordinados, estos controladores carecen de interfaces que permitan una interconexión hacia un sistema central y hacia otras intersecciones.
- La no conexión de las intersecciones hacia una estación central, hace que cada una de ellas actué de manera independiente y aislada, esto genera que, al momento de registrarse un inconveniente en los semáforos, la oficina encargada no esté enterada del caso hasta que sea reportada por algún agente policial, personal verificador o incluso el mismo ciudadano, generando así gran inconformidad en la población y el riesgo de accidentes de tránsito mientras el problema es reportado y resuelto.
- A la hora de realizar el mantenimiento específicamente en la reprogramación y configuración; se tiene que enviar personal que tiene que desplegarse a hasta cada intersección para realizar el trabajo directamente en el controlador, ocasionando gastos en el transporte y que además demanda un tiempo adicional desde que se reporta el problema hasta que este sea solucionado.
- La falta de implementación de mecanismos de interconexión imposibilita el despliegue de plataformas que ayuden a la gestión del comportamiento y en tiempo real de los ciclos en horas

de alto tráfico, bajo tráfico, la ola verde y en casos de emergencias o accidentes de tránsito, de cada uno de los controladores semafóricos.

- El desconocimiento del estado en tiempo real de cada fase, impide el monitoreo del comportamiento de cada intersección y así poder realizar una retroalimentación hacia los otros dispositivos, ocasionando largos tiempos de espera tanto para los peatones y vehículos, además de no contar con la capacidad de reprogramar la configuración establecida desde una estación base y así poder tener una mejor administración del tránsito.

En este contexto los semáforos actuales tienen una gran deficiencia en la resolución del problema de la congestión vehicular, ya que son como islas incomunicadas que actúan por sí mismas de acuerdo a su programación propia estática y en muchos casos la falta de funcionamiento debido a diferentes factores climáticos, o propios de su ciclo de vida.

1.1.1.2 Planteamiento del problema

La ciudad del Cusco no cuenta con un sistema de gestión centralizada para administrar los controladores semafóricos.

1.1.2 Antecedentes

A. "***Estudio de la Integración entre WSN y Redes TCP/IP***" (Iacono & Godoy, 2012)

Articulo presentado por la Facultad de Ingeniería en conjunto con el Instituto de Electrónica de la ciudad de Mendoza, Argentina, se obtuvo las siguientes conclusiones:

- El estudio de los enfoques a nivel de arquitectura sugiere que cada abordaje tiene características que lo hacen más apto para ser aplicado en ciertos tipos de aplicaciones y tecnologías de hardware de WSN.
- El enfoque de redes *overlay*, permite una integración más directa a redes TCP/IP ya que sus nodos se comunican utilizando protocolos de la pila TCP/IP. Esta posibilidad, puede ocasionar

retardos y requiere mayores recursos de hardware, carga de software y consumo de energía en los nodos de las WSN.

- El enfoque de *gateway* es más apto para integrar plataformas de WSN compuestas por nodos con recursos acotados de hardware y consumo de energía. Además, evita retrasos en la entrega de datos. El problema que presenta este enfoque es la centralización del tráfico de datos y el punto de falla en la estación base.
- Respecto al potencial de los enfoques en el nivel de protocolo de interconexión, parece más adecuado utilizar protocolos ad-hoc cuando se necesite optimizar tiempo de entrega de datos, recursos y consumo de energía. Sin embargo, estos protocolos de interconexión, presentan dificultades para integrar redes de sensores heterogéneas en hardware y protocolos de comunicación.
- El uso de protocolos basados en estándares, será adecuado cuando se busque integrar WSN heterogéneas y a la vez utilizar servicios web. En este caso se generan mayores cargas de software, retardos en la entrega de información y consumo de energía en las WSN. Es importante tener en cuenta al implementar la integración WSN – TCP/IP, con el propósito de brindar soporte a una aplicación, revisar los requerimientos específicos de la aplicación (tolerancia a retardos, volúmenes de datos, tipo de hardware WSN, etc.), y en base a estas necesidades decidir cuál es el enfoque que mejor se adapte a cada uno de los niveles que implican la integración a TCP/IP.

Comentario:

Un comentario sobre la importancia de este trabajo, es que muestra el resultado de un análisis de diversas propuestas disponibles para implementar la problemática de la integración WSN - TCP/IP, lo cual significa que si se puede realizar esta integración WSN-TCP/IP y que la plataforma a diseñar e implementar trabaja bajo este último protocolo de comunicación , entonces es posible que se pueda utilizar no solo para la gestión de controladores semafóricos

en el ámbito del ordenamiento vehicular y peatonal, sino que también cualquier otro dispositivo en diferentes contextos como agricultura, domótica, industria, etc. Todo esto sin importar el tipo de red (LPWAN en el caso del IoT) y las especificaciones (LoRA, SIGFOX, etc.) de comunicación entre cada dispositivo, demostrando claramente ser una plataforma IoT.

B. "***Traffic Lights Control Via Web Application***" (Aleksovska, Taneva, & Ganchey, 2015)

Articulo presentado por "*Department of Control System Technical University, Bulgaria*", se obtuvo las siguientes conclusiones:

- En este trabajo basado en una aplicación web para el monitoreo y el control de luces de tráfico usando PLCs, el principal problema es que muchas *Web-Services* industriales son diseñados para comunicarse solo con dispositivos importantes. Este documento presenta una documentación basada en código abierto mediante un servidor XAMMP. De esta forma es controlado todo un sistema de red de varios dispositivos industriales. El problema con el control del tráfico sigue siendo un desafío. Por lo tanto, para un buen sistema de gestión del tráfico es necesario para maximizar la eficiencia del flujo. En este trabajo, el servidor web es una emergente tecnología para diseñar un tráfico basado en un sistema web de gestión. Las ventajas y desventajas de tales sistemas se resumen de la siguiente manera:
  - **Ventajas:** La solución presentada en este documento permite acceder a los PLC de la serie FP (o cualquiera) a través de Ethernet, implementado sobre un servidor web XAMPP.

    Algunas de las ventajas de Web-Server FP son:

    - ✓ Usar un navegador estándar, guardando el software SCADA.
    - ✓ Usar la Intranet existente, guardando el cableado.
    - ✓ Representación de datos PLC en páginas HTML Valor de entrada y cambio en el PLC (conjunto valor, salidas, relés internos) a través de páginas HTML.
    - ✓ Protección de contraseña para el acceso.

- ✓ Función de envío de correo electrónico Transferencia de datos: proceso sistema de control, programación del PLC, telemetría, función de estado supervisado.

Por lo tanto, los proyectos se pueden desarrollar y agregar más controladores que tienen tareas de control separadas en NCS. Eso es posible desarrollar y mejorar el control efectivo del tráfico.

- o **Desventajas:** el trabajo presentado puede ser visto como un paso hacia adelante para NCS con servidores desarrollando. No hay una solución estándar para comunicación entre cualquier servidor y cualquier PLC (como al caso con *Omron* CP1H PLC) El servidor XAMP que se usa no tiene un módulo integrado (opción) para la comunicación *Modbus* TCP, como en el caso de Servidor *FPWeb*.

  Se puede suponer que las ventajas valen el esfuerzo para desarrollar e implementar tal sistema web basado sistemas para monitoreo y control.

  El trabajo futuro puede enfocarse en archivar la información y la seguridad de la red para los semáforos desarrollados NCS.

Comentario:

El aporte de esta publicación está primero en demostrar que si es posible hacer una implementación para la gestión de semáforos mediante una aplicación web, a diferencia de usar un controlador semafórico en esta publicación se utiliza un PLC (Controlador Lógico Programable) esto significa que es posible realizar la gestión de los semáforos mediante plataformas de software sin importar si estos son controlados mediante un PLC o un controlador semafórico.

Otro de los aportes que se debe rescatar de esta publicación es la desventaja de utilizar cierta arquitectura o herramientas para la implementación de la plataforma, haciendo que se tenga que investigar más en otras formas de implementación, así como herramientas que se adapten perfectamente a los resultados a los que se pretende llegar.

C. "***Implementación de un Algoritmo para el Control de Tráfico Vehicular y Peatonal Mediante el Procesamiento de Imágenes***" (Roldan & Ccapatinta, 2015)

Tesis presentada por la escuela profesional de Ingeniería Informática y de Sistemas de la Universidad Nacional de San Antonio Abad del Cusco, Perú, se obtuvo las siguientes conclusiones:

- La alternativa propuesta en el presente proyecto, es un prototipo de prueba de operatividad de tecnología de reconocimiento de densidad, para el control de tráfico, el cual soluciona este problema ya que asigna tiempo de cruce en función a la densidad vehicular y peatonal, esto se evidencia en las pruebas realizadas en la Intersección de la Avenida de la Cultura y Calle Haya de la Torre.
- El algoritmo implementado para el control de tráfico vehicular y peatonal, mediante el procesamiento de imágenes, está sujeto a la posición de la cámara web y la iluminación del medio ambiente en donde se coloquen las cámaras web.
- Las variables identificadas para desarrollar el algoritmo son: corrección gamma, transformación blanco- negro e histograma de blanco- negro.
- Se ha comparado la cantidad de píxeles existentes en una imagen base (sin peatones ni vehículos) y una imagen con peatones y vehículos, teniendo en cuenta la iluminación del ambiente físico, es así que este proceso constituye la metodología de trabajo aplicado para el procesamiento de imágenes orientado al proceso de identificación del flujo vehicular y peatonal.
- Se realizó las pruebas de funcionamiento del algoritmo implementado en una maqueta de 60 X 70 centímetros, en donde se le asigna un tiempo mayor de luz verde (pase) a la vía en donde este contenido mayor número o cantidad de pixeles y recíprocamente para la vía en donde esté contenido un menor número o cantidad de pixeles.

Comentario:

Se rescata de esta tesis la idea de poder realizar la implementación de este algoritmo en las cámaras de video vigilancia desplegadas en la ciudad y así poder enviar información sobre el flujo vehicular y peatonal de cada intersección hacia la plataforma IoT a desarrollar y esta enviar configuraciones hacia los controladores semafóricos, de tal manera que se pueda tener una mejor gestión del tránsito por el sistema debido a la constante retroalimentación de los estados en las que se encuentra cada punto de intersección. Todo esto hace que se tenga que tomar en cuenta la capacidad de interacción y escalamiento de la plataforma hacia otras formas de procesamiento y flujo de datos.

D. "***Diseño e Implementación de una Aplicación IoT en la Nube de Azure para el Análisis de Imagen***" (Mora, 2017)

Tesis de maestría presentada en la Escuela Técnica Superior de la Universidad Politécnica de Madrid, España, se obtiene las siguientes conclusiones:

- El objetivo general del TFM (Trabajo de Fin de Maestría) consistía en desarrollar una solución IoT como caso de estudio con el fin de comprobar las facilidades y el coste de implementación y despliegue de dicha solución con servicios en la nube Azure y, además, administrar el ciclo de vida de la aplicación utilizando la suite de herramientas ALM de VSTS. Estos objetivos se ha alcanzado ya que se ha desarrollado una aplicación para IoT que utilizando los servicios cognitivos *Face API* y *Video API* de Microsoft Azure analiza fotos y videos respectivamente a lo cual hay que añadir que se gestiona utilizando la metodología ágil y que se almacenan los fuentes en un repositorio Git. Destacar que no solo los requisitos funcionales y no funcionales se han cumplido, sino que también los de negocio: el desarrollo se ha realizado en apenas 4 meses y con un coste mensual inferior a los 85 €/Mes (aproximadamente 50 €/Mes).

- Por otra parte, se ha podido aprovechar de la ya inminente integración entre IoT y Cloud Computing que auguró Gartner. En particular, la plataforma *cloud Azure* ofrece un conjunto de servicios especializados en lo que respecta a sistemas basados en IoT, tales como el servicio IoT Hub, que ha facilitado en gran medida el desarrollo de la aplicación del caso de estudio y que, además, la dotan de una escalabilidad ilimitada.

  Comentario

  Se destaca de este trabajo la inminente integración entre IoT y el *Cloud Computing*, además de ver como la mayoría de empresas optan por brindar servicios especializados en IoT, que facilitan en gran medida el desarrollo de plataformas de software y la dotan de una gran escalabilidad ilimitada. Esto nos indica que es posible implementar mecanismos que permitan a este proyecto interoperar en base a servicios con otras plataformas de gran reconocimiento como Windows Azure, Amazon, Google, Etc.

  También se rescata la forma en cómo se realiza el desarrollo gestionando una metodología ágil que será un referente útil para el desarrollo de este proyecto.

1.1.3 Objetivos

1.1.3.1 Objetivo principal

Diseñar e Implementar una plataforma IoT para la gestión de los controladores semafóricos en la ciudad del Cusco.

1.1.3.2 Objetivos específicos

- Diseñar una arquitectura IoT para la plataforma.
- Diseñar e Implementar una base de datos de tipo no relacional.
- Diseñar e Implementar una Interfaz de Aplicaciones “API”.
- Diseñar e implementar un servidor de red “*bróker MQTT”*.
- Diseñar e implementar un Cliente Web.

1.1.4 Justificación

El Instituto Nacional de Estadística e Informática del Perú (INEI) en diciembre del 2009, muestra que en el año 2012 la población total en el distrito del Cusco fue de 118052 habitantes, mientras que en el 2013 la cantidad fue un total de 118231 habitantes entre varones y mujeres. 118322 habitantes el 2014 y 118316 habitantes en el 2015, esto demuestra una tasa de crecimiento poblacional en promedio de casi 100 personas por cada año, solo en el distrito del Cusco. (Boletin-18-INEI, 2009)

La Dirección Regional de Comercio Exterior y Turismo en el año 2015; muestra que en el año 2015 el arribo de turistas a la región Cusco experimentó un crecimiento del 4.90% respecto al año 2014. El arribo de extranjeros tuvo un incremento de 9.84%. En el año 2015 el arribo de visitantes a la provincia de Cusco experimentó un crecimiento del 5.89% respecto al año 2014, sustentando principalmente por el aumento de visitantes extranjeros. (DIRCETUR,CUSCO, 2015),

Según el reporte de la Gerencia de Tránsito, Vialidad y Transporte de la Municipalidad Provincial del Cusco en el año 2006, había 35 mil unidades vehiculares contabilizadas, al 2016 son 115 mil, es decir, el número se triplicó (328%) en tan solo diez años. (La Republica, 2016),

Luego de ver y realizar un análisis de los datos estadísticos sobre el crecimiento poblacional, parque automotor y turismo se hace necesaria la implementación de mecanismos que ayuden a la mejor administración del tránsito vehicular y peatonal en la ciudad del Cusco, es por este motivo que la Municipalidad Provincial del Cusco implementa estos mecanismos interconectando cada intersección mediante fibra óptica formando una red de tipo estrella (Ver anexo A), de gran confiabilidad en velocidad y fácil mantenimiento (NoticiasCusco, 2017). También se tiene instalado adaptadores de red que permiten a los controladores semafóricos existentes conectarse en tiempo real mediante TCP/IP (Ver anexo B). Esto crea un contexto en el que es necesaria la implementación

de una plataforma de TI que opere sobre esta red para tener una solución integral en la mejora de la gestión de los controladores semafóricos instalados en cada intersección.

Los datos obtenidos serán de gran importancia tanto para las autoridades, centros de investigación, empresas que brindan servicios públicos, como para los ciudadanos quienes podrían beneficiarse de un mejor servicio del transporte.

Este proyecto pretende sembrar las bases de una plataforma para que Cusco sea el principal promotor en la región en el uso de la tecnología de información y comunicación, poniendo en práctica los conceptos de internet de las cosas IoT.

1.1.5 Alcances, Limitaciones y Delimitaciones

1.1.5.1 Alcances

- Se utilizará los controladores semafóricos existentes de la Municipalidad Provincial del Cusco, así como los protocolos de comunicación física y lógica (Ethernet, TCP/IP).
- Se utilizará los equipos existentes de la Municipalidad Provincial del Cusco (Switches, servidores y otros).
- La presente tesis emplea herramientas de software libre.
- La plataforma deberá ser capaz de gestionar la totalidad de los controladores semafóricos de la Municipalidad Provincial del Cusco.

1.1.5.2 Limitaciones

- Económico: Falta de presupuesto por parte de la Municipalidad Provincial del Cusco, para la adquisición de equipamiento necesaria para la interconexión.
- Consultoría: Falta de profesionales expertos en el tema dentro de la ciudad del Cusco.

1.1.5.3 Delimitaciones Geográficas

La presente tesis comprende el diseño y la implementación de una plataforma IoT, para la gestión de los controladores semafóricos de la ciudad del Cusco, que actualmente son operados por la Gerencia de Transito, Vialidad y Transporte de la Municipalidad Provincial del Cusco.

Fig. 1.1 Macro Localización: Ciudad del Cusco

Fuente: https://maps.google.com/

Fig. 1.2 Micro Localización: Ciudad del Cusco

Fuente: https://maps.google.com/

1.1.6 Metodología

Para esta tesis, se empleó una metodología descriptiva, puesto que se hizo una descripción detallada de las características y conceptos del proyecto. Teniendo también un carácter analítico

para descubrir los elementos y la relación de sus componentes; todo esto para lograr un mejor entendimiento y desarrollo del proyecto.

Los estudios descriptivos buscan especificar las propiedades, las características y los perfiles de personas, grupos, comunidades, procesos, objetos o cualquier otro fenómeno que se someta a un análisis. Es decir, únicamente pretenden medir o recoger información de manera independiente o conjunta sobre los conceptos o las variables a las que se refieren, esto es, su objetivo no es indicar cómo se relacionan éstas. (Hernandez Sampieri, 2014)

A continuación, se muestra las fases que contemplan como guía para el desarrollo del proyecto:

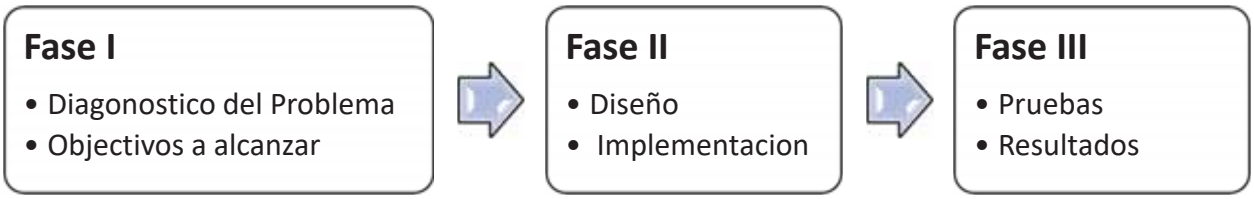


Fig. 1.3 Fases del desarrollo del proyecto

Fuente: Elaboración Propia.

Por otra parte, la metodología de desarrollo que se emplea para la implementación de la plataforma será una metodología ágil, específicamente la metodología Scrum. Esta elección se realizó debido a la característica de poseer un desarrollo incremental, además del compromiso que se tiene hacia el proyecto dado que se ve crecer iteración a iteración (Schwaber & Sutherland, 2014). Y también nos permitirá en cualquier momento realinear el software con los objetivos de la plataforma, ya que puede introducir cambios funcionales o de prioridad en el inicio de cada nueva iteración sin ningún problema.

CAPÍTULO II: MARCO TEORICO CONCEPTUAL

2.1 Internet of things – IoT

Internet de las Cosas, o IoT por sus siglas en inglés, es un conjunto de tecnologías que, a través de la conexión de cualquier objeto a Internet, permiten procesar los datos generados con el fin de entregar información útil para la toma de decisiones relevantes. Este tipo de tecnología se caracteriza por generar grandes volúmenes de datos, que métodos tradicionales de análisis no son capaces de procesar. (Fundación Pais Digital, 2018, pág. 73)

En un mundo cada día más globalizado el internet de las cosas ha resultado ser una gran innovación y de gran utilidad que ha permitido que los objetos estén conectados a internet de modo tal que nos proporcionan información y generan datos que nos ayudaran a tomar decisiones importantes.

(Brea, 2018) Citando a la Agrupación (IERC), define a *Intenet of Things* (IoT) como una infraestructura de red dinámica global capaz de auto configurarse basada en protocolos de comunicación estándar e inter operativos donde tanto las cosas físicas como virtuales tienen identidad propia, atributos físicos, personalidad virtual, e interaccionan de forma inteligente, y se integran de forma transparente, es decir, pasando desapercibidos, en la red de información. (pág.5)

Internet de las cosas es catalogada como una red interactiva que conecta a diferentes dispositivos y que por lo tanto estos adquieren una imagen virtual siendo una evolución tecnológica dado a que no necesita que la presencia del hombre como ejecutor.

Se refieren a la ampliación de la conectividad de red y la capacidad de cómputo a objetos, dispositivos, sensores y elementos que habitualmente no se consideran computadoras. Estos “objetos inteligentes” requieren una mínima intervención humana para generar, intercambiar y

consumir datos; muchas veces tienen conectividad con capacidad de recolección remota, análisis y gestión de datos. (Rose, Eldridge, & Chapin, La Internet de las Cosas-Una breve Reseña, 2015)

Esta nueva tecnología ha permitido darles un valor tecnológico avanzado a todos los objetos que naturalmente no lo eran y que por lo tanto estaban fuera de ser comparados con un computador o cualquier otro componente conectado a una red, sin embargo, en los últimos tiempos esto ha ido cambiado dado que estos objetos "inteligentes "se han convertido en instrumentos útiles brindando mayores servicios.

2.2 Dispositivos IoT

2.2.1 Device IoT

El modelo de comunicación dispositivo a dispositivo representa dos o más dispositivos que se conectan y se comunican directamente entre sí y no a través de un servidor de aplicaciones intermediario. Estos dispositivos se comunican sobre muchos tipos de redes, entre ellas las redes IP o la Internet. Sin embargo, para establecer comunicaciones directas de dispositivo a dispositivo, muchas veces se utilizan protocolos como Bluetooth, *Z-Wave* o *ZigBee*. (Rose, Scott, & Lyman, La internet de las cosas - Una breve reseña , 2015)

El internet de las cosas es la relación o comunicación entre dos o más dispositivos que se entrelazan y comunican entre sí o también mediante un servidor que actuara como intermediador. Las relaciones de los dispositivos se comunican mediante redes de internet, pero si se realiza una comunicación directa se utiliza dispositivos como el Bluetooth, *ZigBee, Z-Wave*. El IoT (Internet of Things) brinda una variedad de soluciones y aplicaciones en la actividad del hombre como servicios de salud y emergencia, calles inteligentes, control de servicios públicos, seguridad, gestión del hogar y del ambiente y congestión de tráfico, como semáforos inteligentes.

El número de aplicaciones y servicios que pueden proporcionar es prácticamente ilimitado y se puede adaptar a muchos campos de la actividad humana, facilitando y mejorando su calidad de vida

en múltiples formas (Santiago Silvestre , 2016).La cantidad de servicios y facilidad que nos puede proporcionar el internet de las cosas IoT (*Internet of Things*) se puede modificar o cambiar muchos campos de sus actividades de trabajo así también como la vida cotidiana, además que es ilimitada. Esto facilita la calidad de vida de los ciudadanos en diferentes campos

En el campo de la investigación se puede encontrar que IoT está brindando soluciones en una alta gama de aplicaciones como: ciudades inteligentes, congestión del tráfico, gestión de residuos, salud, seguridad estructural, servicios de emergencia, logística, ventas al por menor, control industrial, sistemas electrónicos de vehículos, gestión del medio ambiente, gestión del hogar y control de servicios públicos (Paul A, Nuñez A, & Realpe A, 2017).

2.2.2 Controladores semafóricos

Los semáforos inteligentes ofrecen cambios dinámicos de las luces de semáforo en base a mediciones por inducción del flujo del tráfico en la carretera y según la carga se inclinan a favorecer una u otra dirección.

Estos sistemas están conformados por tres componentes: los semáforos, los sensores o cámaras colocadas en las carreteras y la central de control. (Camarena, Contreras, Moreno, Rodriguez, & Salazar, 2018).

Los sensores le indican el estado del tráfico a la central de control y en base a esto toma la decisión para mantener las calles sin congestionamiento. Además, el sistema también utiliza un historial de mediciones para determinar si existe algún tipo de ventaja en realizar el cambio de luz como, por ejemplo, bajar los niveles de contaminación.

2.3 Messages Query Telemetry Transport (MQTT)

MQTT es un protocolo de mensajería basado en el paradigma publicación/suscripción a través de un equipo intermedio denominado bróker. Para su implementación, es necesario el despliegue

de clientes MQTT en el bloque de monitorización que recojan y publiquen la información monitorizada, un bróker y un cliente MQTT que realice la función de suscriptor, procese y almacene la información en el sistema de almacenamiento.

El intermediario en el MQTT es un equipo conocido como bróker y se basa en la publicación de un tema que se le da al suscriptor. En la función es necesario que los clientes MQTT tiene una serie de procesos que monitorizan recogiendo y publicando los temas. El bróker y usuario, que cumple la función de suscriptor, recopile, almacena y procesa la información.

MQTT según los autores Dizdarević, Carpio, Jukan, & Masip-Bruin mencionan que:

> Es uno de los protocolos de mensajería ligeros que siguen el paradigma de publicación-suscripción, lo que lo hace bastante adecuado para dispositivos con recursos limitados y para condiciones de conectividad de red no ideales, como con bajo ancho de banda y alta latencia. Debido a su simplicidad, y un encabezado de mensaje muy pequeño en comparación con otros protocolos de mensajería, a menudo se recomienda como la solución de comunicación de elección en IoT. (2019, pág. 9)

El MQTT siendo el intermediario un equipo conocido como bróker es un conjunto de formalidades de mensajes, que no llevan una vasta cantidad de palabras, que siguen el modelo de publicación-suscriptor, el MQTT es utilizado en dispositivos con escasos recursos, y donde no son ideales la conectividad de red, siendo así el bajo ancho de banda y una alta latencia.

El cliente debe saber el tema en cual está suscrito, una vez realizado se le notificara y recibirá los mensajes respectivos, no obstante, otros usuarios pueden suscribirse al respectivo tema y que le llegue las actualizaciones de nuevos mensajes.

Estos principios también hacen que el protocolo sea ideal para el mundo emergente de las comunicaciones de tipo "máquina a máquina" (M2M) o "Internet de las cosas" de dispositivos conectados, y para aplicaciones móviles, sencillo, ligero y fácil de implantar.

Es ideal para responder a las siguientes necesidades:

- Está especialmente adaptado para utilizar un ancho de banda mínimo.
- Es ideal para utilizar redes inalámbricas.
- Consume muy poca energía.
- Es muy rápido y posibilita un tiempo de respuesta superior al resto de protocolos web actuales.
- Permite una gran fiabilidad si es necesario
- Requiere pocos recursos procesadores y memorias (MQTT un Protocolo Especifico para el Internet de las Cosas, 2015)

2.4 Servidor de red

Según Onyx System menciona al servidor red como:

Un servidor es un equipo informático que forma parte de una red y provee servicios a otros equipos cliente. Se denomina servidor dedicado, aquel que dedica todos sus recursos a atender solicitudes de los equipos cliente. Sin embargo, un servidor compartido es aquel que no dedica todos sus recursos a servir las peticiones de los clientes, sino que también es utilizado por un usuario para trabajar de forma local.

El servidor red sirve como un ordenador que informa la red provee sus servicios o comparte a otros agregados de ordenadores, manteniendo los archivos de forma privada, pero compartida para los que están enlazados en la red. Son de dos formas servidor dedicado, el que da o proporciona recursos y busca atender solicitudes de otros ordenadores. Por el contrario, el servidor compartido, no indica o da todos los recursos de servir las peticiones del usuario, sirve para trabajar de manera local.

Según Claranet; menciona que los tipos de servidores red son:

Servidor de impresiones, servidor de correo, servidor de fax, servidor de la telefonía, servidor proxy, servidor del acceso remoto (RAS), servidor web, servidor de base de datos, servidor de Seguridad. (Claranet , 2012) Estos son algunos tipos de servidores red que podemos que encontramos en el mercado:

Servidor de impresiones: control de uno o más máquinas de impresión e imprime, permitiendo a los usuarios de red enviar, poniéndolo en cola

Servidor de correo: almacenaje, envió, recibe, en ruta y realización de los correos de los usuarios de red

Servidor de fax: gestiona el almacenamiento, decepcionando y almacena el o los faxes

Servidor de la telefonía: Almacena los contestadores, mensajes escritos o de voz, guiando las llamadas del usuario y el control de la red o internet

Servidor proxy: Tiene como como función principal guardar la memoria cache de las respectivas páginas web, que es de acceso para los clientes. También se encarga de las veces que ha utilizado o entrado a la página web, produce servicios de seguridad.

Servidor del acceso remoto: control de las líneas del módem u otras secciones de comunicación de la red, verificando la conexión de la red o peticiones de red.

Servidor web: almacenaje del contenido web como documentos HTML, archivos de texto, escritos, videos e imágenes y este da a los usuarios que piden o solicitan.

Servidor de base de datos: Provee servicios de datos a otras computadoras o programas.

Servidor de seguridad: Teniendo un software especializado que evita a intrusos maliciosos o virus, se utiliza antivirus, *antiadware*, *antispyware*, además que incluye cortafuegos. Se pone en diferentes estratos que evitan ataque de una manera eficiente.

2.4.1 Mosca-MQTT

Debido a que el presente trabajo está dirigido a la comunicación de diferentes dispositivos que requieran de bajo consumo de energía, es necesario la implementación de mecanismos que nos permitan este tipo de comunicación bajo los estándares y protocolos de MQTT.

Se sabe que MQTT es un protocolo de comunicación de dispositivos IoT (uno a uno o uno a mucho), mientras que por el lado de la arquitectura presenta una topología de tipo estrella, haciendo necesaria la implementación de un servidor o *bróker* de comunicaciones por él se gestionará todos los mensajes. En tal sentido se presenta la necesidad de analizar las diferentes herramientas tecnológicas existentes que se pueda utilizar como un servidor de red MQTT.

Existen diversas opciones instalables como *Mosquitto*, *HiveMQ* y *VernMQ* por mencionar algunas. También hay servicios de *bróker MQTT* en la nube como por ejemplo *AWS IoT e IBM Bluemix* listos para usarse, cada una con diferentes características de escalabilidad y útiles en base a diferentes de factores que caracterizan la performance de un servidor MQTT:

- Longitud del mensaje incluido la longitud del tópico y el propio mensaje.
- QoS del mensaje enviado.
- Tráfico de red en un punto y tiempo específico.
- Velocidad del servidor implementado con un *bróker* MQTT.
- El protocolo liviano le permite implementarse en hardware de dispositivos altamente limitados y en redes con ancho de banda de alta latencia/limitado.
- Su flexibilidad hace que pueda soportar varios escenarios de aplicaciones para dispositivos y servicios de IoT.

Los primeros dos factores son factores internos de las cuales podemos tener el control mientras que los otros dos dependen mucho del contexto y los medios de transporte.

Mosca-MQTT es un *bróker* de código libre basado en Node.js, permite la creación de un servidor de red unitario "*standalone*", o como parte de una aplicación más grande. Para el presente proyecto se utilizará como "*standalone*" puesto que este estará funcionando dentro de un contenedor Docker. Este bróker ofrece una gran comodidad para visualización de mensajes, aunque en el instalador se recomienda el añadir una herramienta llamada "*bunyan"* para observar los mensajes de depuración. Esta herramienta, en unión con mosca muestra toda la información de paso de mensajes con un código de colores asociado para facilitar su lectura.

La razón principal por la que al final se opta por este bróker es por unidad de lenguajes, también esté basado en Node.js. Esto facilita el despliegue de la plataforma en cualquier parte puesto que ya no depende del sistema, lo único requerido para su instalación en cualquier sistema es tener Node.js instalado. Mosca puede ser protegido con usuario y contraseña y también es posible cambiar los puertos de escucha.

2.4.2 Transport Layer Security (TLS)

Seguridad en capas de transporte: el protocolo criptográfico que garantiza las comunicaciones en internet, se considera como la siguiente generación del certificado SSL, permite y garantiza el intercambio de datos en un entorno securizado y privado entre dos entes, el usuario y el servidor, mediante aplicaciones como HTTP, POP3, IMAP, SSH, SMTP o NNTP. Se considera al TLS como una versión mejorada de SSL, puesto que este abarcó gran parte de la funcionalidad de SSL, para enfrentar las vulnerabilidades y así poder entregar el cifrado mediante algoritmos más fuertes, en el cual las dos partes implicadas se comunican y negocian las claves que usaran para encriptar y desencriptar el mensaje. Además, se considera como el estándar oficial y el que se recomienda usar siempre.

La encriptación se realiza mediante dos protocolos en capas diferentes: el protocolo de autenticación (llamado *TLS Record Protocol*) y el de mutuo acuerdo (también conocido como *TLS Handshake Protocol*).

a) Record: se lleva a cabo la autenticación para que la transmisión de datos sea mediante una conexión privada y fiable (se negocia la encriptación y la integridad del emisor-receptor)

b) Handshake: se negocia el mensaje de manera segura. En cada mensaje se especifica el protocolo en un campo (llamado *content_type*) y se cifra y empaqueta con un código de autentificación (o MAC).

Por lo tanto, en el protocolo TLS, se lleva a cabo un canal seguro y cifrado entre cliente y servidor en donde se negocia la criptografía del mensaje, se autentifican las claves del cifrado y se realiza una transmisión segura.

El protocolo TLS se distingue por la seguridad con la interoperabilidad (las transmisiones de datos encriptados de diferentes aplicaciones como HTTP, que pasa a ser HTTPS) (IBM, 2018).

2.4.3 Ngx-mqtt

Librería implementada en base a MQTT.js para versiones de Angular mayores a "Angular 2". Utiliza observables, se encarga del manejo de suscripciones y el enrutamiento de mensajes bajo el protocolo MQTT[1]. Esta herramienta es ideal para aplicaciones con muchos componentes y muchos suscriptores. El problema es que si se suscribe regularmente a MQTT con bibliotecas cliente como MQTT.js, aun así todos los mensajes se manejan con un controlador de eventos en el mensaje, por lo que debe enviar los mensajes recibidos por su cuenta. Entonces, si tiene múltiples componentes usando MQTT en su código, solo desea recibir los mensajes para su filtro local. Además, si destruye

[1] Fuente: https://github.com/sclausen/ngx-mqtt

un componente, desea darse de baja de MQTT, pero solo si ningún otro componente usa el mismo filtro.

Esta biblioteca expone un método *observe(filtro)* y *observeRetained(filtro)*, que devuelve un Observable. Si se suscribe a uno de estos observables, se ejecuta la suscripción real de MQTT. El filtro de tema se usa para agregar solo mensajes MQTT coincidentes con el observable. Cada otra ejecución de *observe (filtro)* y *observeRetained*(filtro) con un filtro ya usado devolverá el mismo observable. El observable realiza un seguimiento de los suscriptores y ejecuta un método de cancelación de suscripción MQTT, si todos los suscriptores se han dado de baja del observable. La diferencia entre *observe(filtro)* y *observeRetained*(filtro) es que este último emitirá el último mensaje recibido a nuevos suscriptores. (ngx-mqtt, 2018).

2.5 Sistemas en tiempo real

Según Wilhelm Harder, Zarnett, Montaghami, & Giannikouris mencionan que:

> El software interactivo siempre está sujeto a retrasos. Seguramente ha experimentado la sensación de esperar un segundo para que un procesador de textos le responda ingresando una sola tecla, o el mouse tarda una fracción de segundo más en responder. (Wilhelm Harder, Zarnett, Montaghami, & Giannikouris, 2014)

El sistema a tiempo real produce en el software respuestas dentro de un específico límite de tiempo, siendo algo crucial para el funcionamiento correcto. El software es interactivo, responde a comandos, y siempre se da retrasos, al momento de que el procesador de textos responda cuando se introduce una tecla o en el caso del mouse tarda una fracción de segundos en responder son válidos, pero siempre y cuando sean lo más mínimo posible

Algunos ejemplos que mencionan Wilhelm Harder, Zarnett, Montaghami, & Giannikouris son: Transporte, militar, procesos industriales, médico, telecomunicaciones, hogar, gestión del edificio (Wilhelm Harder, Zarnett, Montaghami, & Giannikouris, 2014, pág. 3).Tenemos como ejemplo

estos sistemas de tiempo real que son: transporte, sistema y control del tráfico de los vehículos, aeronaves, barcos, trenes; militar, sistematización de armas, seguimientos y comunicación; procesos industriales, sistematización y control para la producción, también la química, energía y la fabricación mediante robótica; medica, cuidado del paciente mediante la motorización, desfibriladora y radioterapia; telecomunicaciones, televisión, teléfono, videotelefonía, satélite, redes informáticas, y radio; hogar, control mediante la monitorización de los electrodomésticos y; gestión de edificios, calefacción, ventilación, seguridad

2.5.1 Go

Go, al igual que C y C++, es un lenguaje compilado y concurrente, o en otras palabras: soporta canales de comunicación basados en el lenguaje CSP. Sin embargo, la concurrencia en Go es diferente a los criterios de programación basados en bloqueos como *pthreads*. Los creadores de Go, además, se inspiraron en la versatilidad y las cualidades de otros lenguajes como Python, C++ y Java (entre otros), para conseguir un lenguaje con las siguientes características, algunas únicas, y otras compartidas con otros lenguajes compilados.

Lenguajes como C++, Java o C# son más pesados y voluminosos que Go. La simplicidad es la característica principal de Go. Con una sintaxis clara, limpia y organizada, la idea de este nuevo lenguaje de programación es diferenciarse de la complejidad de C. Comparando ambos lenguajes compilados, vemos que Go utiliza inferencia implícita de tipos para así poder evitar la declaración explícita de variables que tienen lugar en C. (Lenguaje de programación Go y sus característica, 2018)

Características

Compilado: No se necesita instalar ningún programa para que el programa que fue desarrollado funcione en el sistema operativo para el que se compilo.

Estáticamente Tipado: Las variables son tipadas de manera estática, así que, si la variable "x" se define como entera, será entera durante todo su alcance.

Concurrente: Está inspirado en CSP

Uso poco usual de POO: Go no usa clases, no usa herencia y el uso de interfaces se realiza de manera implícita. Esto con el fin de mejorar el rendimiento al momento de diseñar el software.

Uso de paquetes: Se usan los paquetes para organizar el código. Un paquete puede tener varios archivos "*.go*" que permiten definir lo que va a realizar el paquete. Para usar un paquete en el programa, se debe importar.

2.5.2 Node.js

Según Gracia menciona que el Node.JS es: Una tecnología que permite utilizar en el servidor código JavaScript y dejarlo fuera del navegador como si estuviéramos utilizando algunas de los lenguajes de programación ya mencionados como ASP.NET, PHP, Python, etc. (Gracia, 2014).Node.JS sirve como entorno de ejecución de JavaScript que utiliza el motor V8 de Chrome esto nos permite utilizar el servidor código JavaScript con el fin de dejarlo al navegador como estuviera manejando los lenguajes de programación ASP.NET, Python, PHP, etc.

2.5.3 Angular JS

Según los autores Basalo, Angel Alvarez, Hurtado, & Cerdá Angular.JS sirve como una potente herramienta para el desarrollo de aplicaciones de tipo SPA(Simple Page Aplications):

> Este framework pretende que los programadores mejoren el HTML (*HyperText Markup Language*) que desarrollan. Que puedan producir un HTML que, de manera declarativa, genere aplicaciones que sean fáciles de entender incluso para alguien que no tiene conocimientos profundos de informática. El objetivo es producir un HTML

> altamente semántico, es decir, que cuando lo leas entiendas de manera clara qué es lo que hace o para qué sirve cada cosa. (Basalo, Angel Alvarez, Hurtado, & Cerdá, 2014)

Angular.JS permite utilizar HTML (ordena el contenido de una página web) como lenguaje de plantillas, permitiendo extender su sintaxis, eliminando la cantidad de códigos que se tienen que escribir para una aplicación mejorando el HTML además de ayudar a los más inexpertos en conocimientos de informática. Su objetico es generar un HTML bastante entendible de modo que se lea de una manera clara y concisa el cómo funciona, hace o para que sirve una respectiva cosa.

Y todo sucede en el navegador, lo que lo convierte en un socio ideal con cualquier tecnología de servidor. Angular toma otro enfoque. Se trata de minimizar la falta de concordancia entre el documento HTML y cuáles son las necesidades de una aplicación mediante la creación de nuevos bloques HTML. Angular enseña la nueva sintaxis navegador a través de una construcción que llamamos directivas. Los ejemplos incluyen:

- Enlace de datos, como en { { } }.
- Estructuras de control para repetir / ocultar fragmentos de DOM.
- El apoyo a las formas y la validación de formularios.
- Colocación de código subyacente para los elementos DOM.
- Agrupación de HTML en componentes reutilizables.

AngularJs es un marco estructural MVC del lado del cliente escrito en JavaScript. Se ejecuta en un navegador web y en gran medida ayuda a los desarrolladores a escribir modernas single-page, aplicaciones web de estilo AJAX. Es un marco de uso general, pero brilla cuando se utiliza para escribir tipo de aplicaciones web CRUD.

AngularJs es una adición reciente a la lista de marcos MVC del lado del cliente, sin embargo, ha logrado atraer mucha atención, sobre todo debido a su innovador sistema de plantillas, facilidad

de desarrollo y prácticas de ingeniería muy sólidas. De hecho, su sistema de plantillas es único en muchos aspectos:

- Se utiliza HTML como el lenguaje de plantillas.
- No requiere una actualización DOM explícita, AngularJs es capaz de seguir las acciones del usuario, los eventos del navegador, y cambios en el modelo, y de averiguar cuándo y qué plantillas para refrescar.
- Tiene un interesante y extensible subsistema de componentes, y es posible enseñar un navegador cómo interpretar nuevas etiquetas y atributos HTML.

El subsistema de plantillas puede ser la parte más visible de AngularJs, pero no confundir, AngularJs es un marco completo lleno de muchas utilidades y servicios típicamente necesarios en las aplicaciones web de una sola página (Kozlowski & Darwin, 2013).

AngularJs permite escribir aplicaciones web del lado del cliente como si se tuviera un navegador más inteligente. Permite utilizar el buen y antiguo HTML como lenguaje de la plantilla y le permite extender la sintaxis de HTML para expresar los componentes de su aplicación de forma clara y concisa. Sincroniza automáticamente los datos de la interfaz de usuario (la vista) con sus objetos JavaScript (modelo) a través de 2 vías de enlace de datos denominado *databinding*. Para ayudar a estructurar su aplicación mejor y hacerla más fácil de probar, AngularJs enseña al navegador cómo hacer la inyección de dependencia y la inversión de control. Ah sí, y también ayuda con la comunicación del lado del servidor, dominando las devoluciones de llamada asincrónicas con anticipación y retardos; y haciendo que la navegación del lado del cliente y enlaces profundos con direcciones *hashbang* o HTML5.

Para el desarrollo de este proyecto, específicamente para el desarrollo del “Cliente web”, se usara una de las versiones más recientes de este framework “Angular 5”, muy útil no solo por las

bondades mencionadas anteriormente sino también porque supone un avance muy fuerte sobre Angular 1.x e incluye el uso de *TypeScript* como lenguaje de referencia.

Su instalación es bastante sencilla. Basta con instalar Angular-Cli y usar la línea de comandos para la creación de proyectos. El resultado de estas operaciones es la creación de una estructura de archivos, tal como se muestra en la figura 2.1, útiles en un proyecto de Angular 5.

Fig. 2.1 Estructura de archivos en un proyecto de Angular 5

Fuente: https://angular.io/guide/quickstart

Cada uno de estos ficheros tiene un rol para el funcionamiento de un proyecto angular, la tabla siguiente muestra los propósitos para los cuales fueron creados cada archivo y carpeta.

Tabla 2.1 Propósito de las Carpetas y Archivos Angular 5

Archivo	Propósito
e2e/	En esta carpeta se almacenan todos los archivos compilados que serán ejecutados como pruebas del proyecto.
node_modules/	Node.js crea esta carpeta y coloca todos los archivos de las librerías que se van a utilizar en el proyecto y que están registradas en el archivo *package.json*.
.editorconfig	Archivos de configuración
.gitignore	Archivo de configuración en donde se define los archivos que no serán tomados en cuenta por el SVC[2]
angular.json	Configuración para Angular Cli. En este archivo se puede modificar cuales archivos serán incluidos a la hora de compilar el proyecto.
package.json	Archivo de configuración para npm, aquí se registra los paquetes externos que se usan en un proyecto específico.
protractor.conf.js	Archivo de configuración para las pruebas cuando se corre el comando nge2e.
README.md	Documentación básica de la descripción del proyecto
tsconfig.json	Archivo de configuración para el compilador de *TypeScript*
tslint.json	Archivo de configuración para *TSLint* y *Codelyzer*, usado cuando se corre el comando *ng lint*.

Fuente: https://angular.io/guide/quickstart

La figura 2.2 muestra la organización de los archivos dentro de la carpeta "*src"*, esta carpeta es importante puesto que aquí se encuentran todos los componentes de Angular como plantillas,

[2] SVC: Sistema de control de Versiones

estilos, imágenes y cualquier otra cosa que necesite la aplicación. Cualquier archivo fuera de esta carpeta será como apoyo en la construcción de la aplicación.

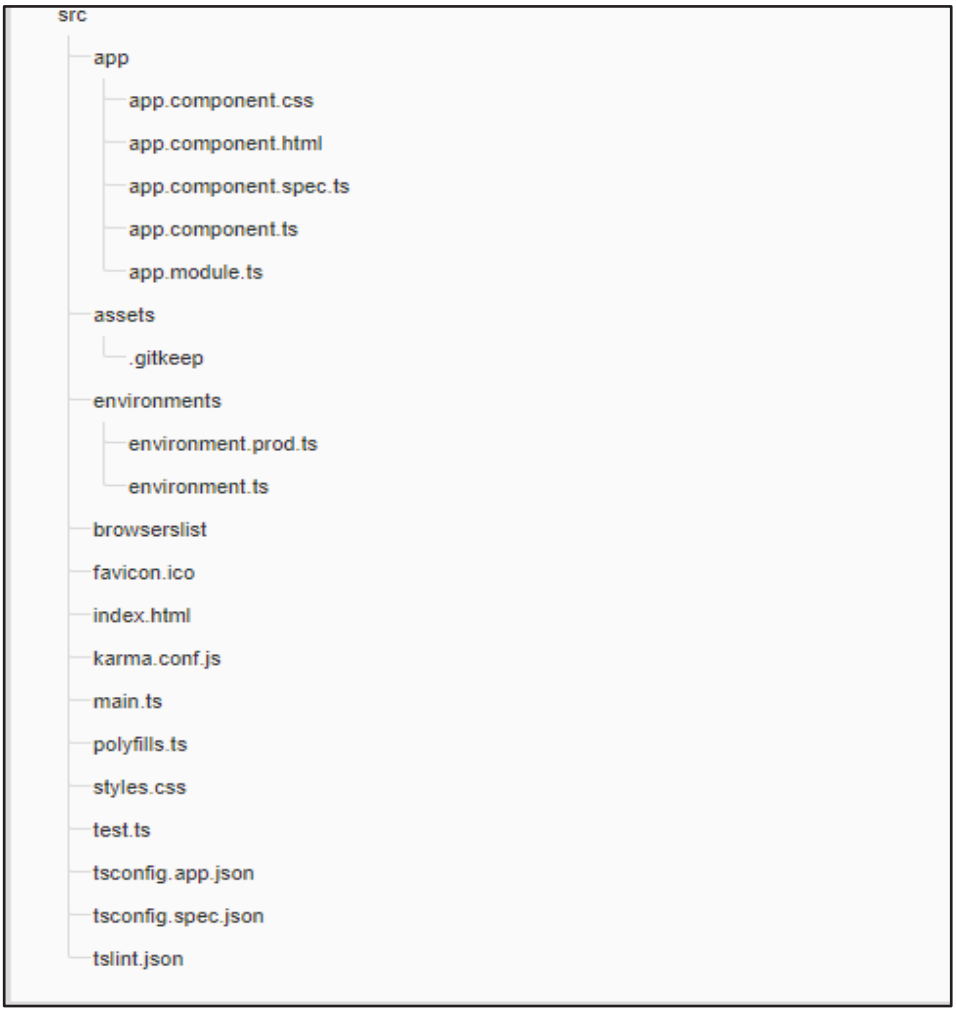


Fig. 2.2 Estructura de archivos en la carpeta "src/"

Fuente: https://angular.io/guide/quickstart

Al igual que los archivos mostrados en la estructura del proyecto principal de Angular 5, la carpe *src/*, también posee archivos y carpetas que tienes un rol específico para el funcionamiento de la aplicación. La tabla 2.2 muestra una lista de propósitos de las carpetas y archivos existentes dentro de la carpeta *src/*.

Tabla 2.2 Propósito de los Archivos en src/

Archivo	Propósito
app/app.component.{ts,html,css,spec.ts}	Carpeta para definir el componente principal de la aplicación con una plantilla HTML, Hoja de estilo CSS, clases TS y pruebas unitarias.
app/app.module.ts	Define el *AppModule*, módulo principal que le dice a angular como ensamblar la aplicación.
assets/*	Carpeta donde se pueden colocar imágenes y cualquier otro tipo de archivos estáticos.
environments/*	Esta carpeta contiene un archivo de configuración de variables para el desarrollo y despliegue de la aplicación de Angular
Browserslist	Un archivo de configuración para compartir navegadores de destino entre diferentes herramientas de *front-end*.
favicon.ico	Icono que se verá en la barra de marcadores.
index.html	La página HTML principal que se sirve cuando alguien visita el sitio. La mayoría de las veces, nunca se necesitará editarlo. Angular Cli agrega automáticamente todos los archivos js y css al compilar la aplicación para que nunca se tenga que agregar manualmente ninguna etiqueta <script> o <link>.
karma.conf.js	Configuración de prueba unitaria para el corrector de prueba Karma, que se usa al ejecutar la prueba *ng test*.
main.ts	El principal punto de entrada para la aplicación. Compila la aplicación con *JIT compiler* y arranca el módulo raíz de la aplicación (*AppModule*) para que se ejecute en el navegador. También puede usar el *AOT compiler* sin cambiar ningún código agregando el indicador - *aot* a los comandos *ng build* y *ng serve*.
polyfills.ts	Los diferentes navegadores tienen diferentes niveles de compatibilidad con los estándares web. Los *polyfills* ayudan a normalizar esas diferencias.
styles.css	Archivo donde se definen los estilos globales. La mayoría de las veces se querrá tener estilos locales en los componentes para facilitar el mantenimiento, pero los estilos que afectan a toda su aplicación deben estar en un lugar central.
test.ts	Este es el principal punto de entrada para las pruebas unitarias. Tiene una configuración personalizada que puede ser desconocida, pero no es algo que deba editar.
tsconfig.{app|spec}.json	Configuración del compilador de *TypeScript* para la aplicación Angular (*tsconfig.app.json*) y para las pruebas unitarias *(tsconfig.spec.json)*.

Fuente: https://angular.io/guide/quickstart

Para la implementación del cliente web, se hizo uso de una herramienta que facilite el desarrollo del sistema. En este caso se usó AngularJs 5, framework en base a Java Script basado en componentes para crear *Single Page Application* (SPA), o aplicaciones de página única. Es uno de los framework más populares para desarrollar aplicaciones modernas y escalables en el lado del cliente. Esta herramienta supone un gran avance sobre otras versiones de Angular 1.x e incluye el uso de *TypeScript*[3] como lenguaje de referencia.

La creación del proyecto se hace mediante el uso de la línea de comandos de la herramienta Angular Cli[4], esta es la forma sencilla de ejecutar y que se encarga de crear la organización de los archivos de un proyecto de inicio.

Para la creación de las vistas que serán las que interactúen con los usuarios, es necesario primeramente la creación de los componentes necesarios que formaran parte de ella, estos componentes se crean y codifican de forma independientes, tanto en su visualización, estilo y lógica, luego son referenciados de ser necesarios en un componente más grande que las agrupe y muestre al usuario de forma que se pueda cumplir con la funcionalidad requerida. Para insertar, modificar o eliminar datos en la base de datos se realizan peticiones[5] hacia la API/GraphQL estas peticiones retornaran con resultados en formato JSON que serán operados por las funciones en las clases de cada component.ts, renderizados y mostrados de acuerdo a la organización de la plantilla HTML, esta forma de trabajo se utiliza para la totalidad de componentes que conforman el Sistema Cliente Web, y permitirá cumplir satisfactoriamente con las tareas e Historias de Usuario creadas anteriormente.

[3] Súper conjunto de JavaScript, que esencialmente añade Tipado estático y objetos basados en clases.
[4] La instalación se realiza mediante gestos de paquetes “npm”.
[5] *Query o Mutation*

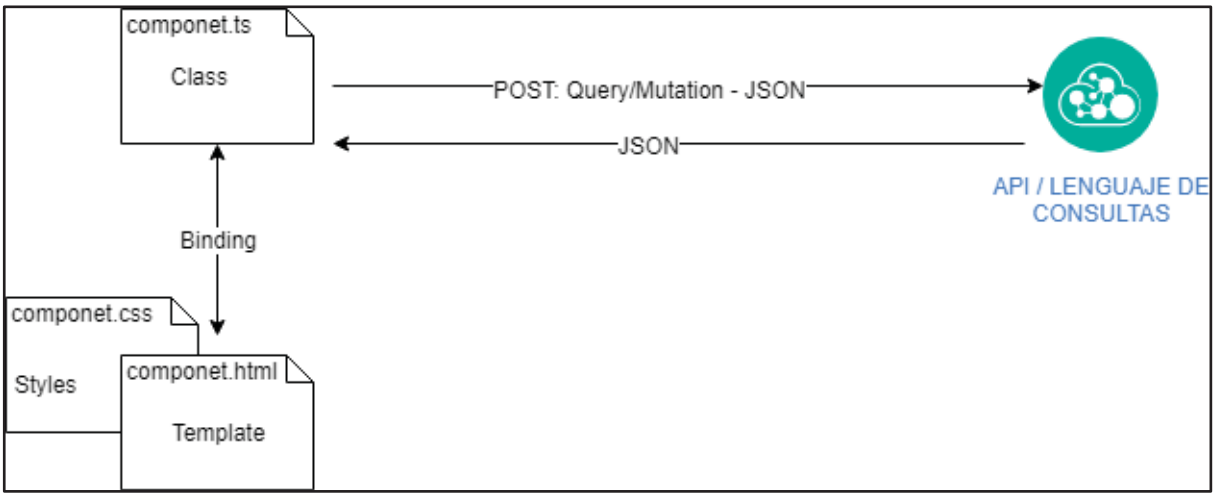


Fig. 2.4. Envío - Recepción datos - API/GraphQL

Fuente: Elaboración Propia

La figura anterior, muestra la forma en cómo se envían y reciben los datos por parte de la API/GraphQL y un determinado componente, esta lógica de trabajo se utilizara en la totalidad de los formularios desarrollados a lo largo del proyecto.

2.6 Base de datos no relacional

2.6.1 MongoDB

Según el autor Garcia Muñoz; Mongo es: una base de datos orientada a documentos. Esto quiere decir que, en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en un formato BSON, que es una representación binaria de JSON. (Garcia Muñoz , 2016, pág. 4).

Según Fernández menciona de MongoDB que:

Una de las diferencias más importantes con respecto a las bases de datos relacionales, es que no es necesario seguir un esquema. Los documentos de una misma colección-concepto similar a una tabla de una base de datos relacional -, pueden tener esquemas diferentes. (2014)

Lo que les diferencia de las bases de datos de tipo relacional, es que no necesariamente se debe seguir un esquema fijo. El mismo puede generar un esquema y que difieren de otros,

pudiendo almacenar información que esta semiestructurada. Los documentos que posean una similar o misma tabla de base de datos relacional puedan que tenga diferentes esquemas.

La base de datos MongoDB está completamente escrita en C++, la principal ventaja de esta base de datos NoSQL es la velocidad de consulta, el motivo de esto es porque la información se la almacena en archivos de formato BSON que son versiones modificadas de JSON, con este formato se puede lograr flexibilidad para datos, mejor búsqueda y localización de información, conversiones de objetos en una aplicación a documento JSON. Esta base de datos contiene para su uso en desarrollo de aplicaciones, índices secundarios, consultas dinámicas, orden en los resultados que se obtiene, agregaciones, almacenamiento de información ilimitada. Las características técnicas que posee la base de datos NoSQL MongoDB hacen que trabaje con grandes cantidades de información y asimismo de usuarios conectados indefinidamente en la aplicación, las características principales son:

- **Escalabilidad horizontal.** El autor Garcia Muñoz menciona como caracteristica la escalabilidad horizontal dando a conocer como parte de su funcion principal:

  Sharding distribuye datos a través de un grupo de máquinas. MongoDB soporta la creación de zonas de datos basadas en la clave *shard*. En un *cluster* equilibrado, MongoDB dirige las lecturas y escrituras cubiertas por una zona sólo a aquellos fragmentos dentro de la zona. (Garcia Muñoz , 2016, pág. 5)

  Distribuye cantidad de datos e información a un grupo de equipos conectados entre sí, Mongo reconoce en sitios de datos que están en clave *sharding*, mediante un *cluster* equilibrado que manda la información de aquellas lecturas a una respectiva zona
- **Consultas ad hoc:** Permite la consulta de información a través de búsquedas de campos, expresiones regulares con comandos de manera rápida y eficaz.
- **Indexación:** MongoDB utiliza indexación para aumentar la eficiencia en la búsqueda de información.

- **Replicación:** MongoDB puede realizar replicación maestro-esclavo (Master – Slave). Master: ejecuta comandos para la lectura y escritura de los datos. Slave: realiza solo lectura de datos pero no los puede modificar, si es que por algún caso no pueda acceder al maestro que se le designo, este esclavo puede elegir otro maestro para seguir trabajando.
- **Blanco de carga:** MongoDB permite que pueda ejecutarse en distintos servidores a la vez, haciendo más fácil su distribución en la carga de usuarios que deseen conectarse así como acceder a la información.
- **Almacenamiento de archivos:** MongoDB puede ser usado como un gestor de archivos e información haciendo que esta base de datos tenga una gran capacidad de almacenar información a través de sus diferentes servidores activos.

Usos en MongoDB: este potente gestor de base de datos se puede implementar en:

- Sistemas de manejo de documentos.
- Juegos
- Aplicaciones móviles
- Registro de eventos.
- Almacenamiento de información en páginas web como locales.
- Administración de estadísticas

Especificaciones de MongoDB.

- Escrito en C++ como primer lenguaje para su creación, se crearon drivers para su manejo más óptimo.
- Mantiene alguna de las propiedades del lenguaje SQL.
- Licencia AGPL apache.
- *Sharding* o replicación maestro – esclavo.
- Utiliza sentencias en JavaScript.
- Realiza un mapeo de los archivos para mejor almacenamiento de la información.

- En sistemas de 32 bits se limita su capacidad a 2.5 Giga Bytes.
- Se agrega el campo de indexación geo espacial.
- Protocolo que trabaja sobre TCP/IP.
- Orientada a documentos almacenados en archivos BSON.

La base de datos contiene específicamente colecciones que a su vez contiene documentos.

2.7 Servicios web

2.7.1 Arquitectura orientada a micro servicios

La arquitectura de micro servicios establece construir una aplicación como un conjunto de servicios, donde cada uno de estos es independiente del otro y hasta pueden ser escritos en lenguajes diferentes y mantenidos por equipos diferentes. Lo anterior permite que el problema de escalar se solucione incrementando el procesamiento necesario para el servicio específico que lo esté requiriendo y no de otros que no lo necesitan. (Barrios, 2017)

Una arquitectura de micro servicios promueve el desarrollo y despliegue de aplicaciones compuestas por unidades independientes, autónomas, modulares y auto- contenidas, lo cual difiere de la forma tradicional o monolítica. (Wolf, 2017)

A continuación, se describe una lista de características que hacen que la arquitectura orientada a micro servicios sea una de las tendencias en el desarrollo de sistemas:

- **Aplicaciones monolíticas**:

En una aplicación monolítica toda la lógica se ejecuta en un único servidor de aplicaciones. Las aplicaciones monolíticas típicas son grandes y construidas por múltiples equipos, requiriendo una orquestación cuidadosa del despliegue para cada cambio. También se consideran aplicaciones monolíticas cuando existen múltiples servicios de API que proporcionan la lógica de negocio, toda

la capa de presentación es una sola aplicación web grande. En ambos casos, la arquitectura de micro servicios puede proporcionar una alternativa. (López & Maya, 2017)

En construir una aplicación como una unidad en la que se construyen usualmente aplicaciones empresariales, como una unión de tres partes: la parte cliente, la parte encargada del manejo de datos y la parte servidor que contiene la lógica de dominio de la aplicación. El emplear un estilo monolítico en aplicaciones puede tener resultados exitosos, sin embargo, el paso del tiempo ha generado que más personas se sienten frustradas, puesto que los cambios constantes en el modelo de negocio (lo que se transforma en cambios de requerimientos) causan que se tenga que modificar toda la aplicación haciéndola más grande, lo que lleva a que mantener este tipo de aplicaciones modulares se vuelva más complicado. Otro tema muy importante —sobre todo en esta época— es el permitir crecer con cierta funcionalidad del sistema, para solucionar esto, se incrementa la capacidad de toda la aplicación mas no del segmento que realmente lo requiere. (Barrios, 2017)

Para explicar el concepto de los micro servicios es útil compararlos con una aplicación monolítica construida como una sola unidad. Las aplicaciones empresariales se construyen a menudo en tres partes principales:

- Una interfaz de usuario del lado del cliente (que consiste en páginas HTML y JavaScript que se ejecutan en un navegador en el ordenador del usuario).
- Una base de datos (que consta de muchas tablas en una base de datos común y por lo general relacional).
- Una aplicación del lado del servidor. La aplicación del lado del servidor se encargará de las peticiones HTTP, ejecutará la lógica de negocio, recuperará y actualizará datos de la base de datos, recuperará y rellenará formularios de las vistas HTML que se envían desde el navegador. Esta aplicación es monolítica (Un único ejecutable lógico) Cualquier cambio en el sistema implica la creación y despliegue de una nueva versión de la aplicación del lado del servidor.

- Las aplicaciones monolíticas pueden tener éxito, pero cada vez más personas se sienten frustradas con ellas, especialmente a medida que más aplicaciones se están desplegando a la nube.
- Un cambio realizado en una pequeña parte de la aplicación, requiere que todo el sistema deba ser reconstruido y desplegado. Con el tiempo, a menudo es difícil mantener una buena estructura modular, por lo que es más difícil mantener los cambios que deberían afectar a un solo módulo dentro de la aplicación.
- El escalamiento requiere el despliegue de toda la aplicación en lugar de solo las partes que requieren un mayor recurso.
- Estas frustraciones han llevado a seguir una arquitectura orientada a micro servicios, construyendo un conjunto de servicios. Así como el hecho de que los servicios sean escalables y posean independencia en el despliegue, puedan ser escritos en distintos lenguajes de programación y a su vez administrados por distintos equipos.
- Desarrollar aplicaciones usando una arquitectura orientada a micro servicios puede ofrecer un número significativo de beneficios (Requena, 2016).

- **Agilidad y productividad en el desarrollo**: El tamaño y la complejidad de un Micro servicio es muy pequeña en comparación con una gran aplicación monolítica. Un Micro servicio también tiene un contexto limitado y está desacoplado de otros servicios. Además, la descomposición de lo que solía ser una aplicación monolítica en una colección de pequeños procesos / funciones reduce drásticamente la complejidad del código y mejora la productividad de programación**.**

  También hace que sea más fácil escalar el desarrollo con múltiples equipos donde cada equipo posee una parte independiente de la aplicación que puede funcionar como un servicio en sí mismo. La única pieza clave que necesitan saber es cómo interactuar con otros servicios.

El riesgo inherente a introducir nuevos cambios se reduce, lo cual permite que la tasa de cambio aumente desembocando en una mayor agilidad y tiempos de respuestas más rápido (Requena, 2016).

- **Flexibilidad en los despliegues:** Debido a que están débilmente acoplados a otros servicios, los micro servicios están versionados de forma independiente y con independencia de despliegue lo que reduce en gran medida el riesgo de fallos que son inherentes a las aplicaciones grandes y monolíticas.

  Por otra parte, el bajo acoplamiento a otros servicios reduce la dependencia de los desarrolladores de otros equipos y les proporciona la agilidad necesaria para liberar y poner a prueba nuevas características, prácticamente a la carta, como los nuevos requisitos y casos de uso que vienen en parte del cliente.

  La arquitectura orientada a micro servicios también permite flexibilidad en la implementación en términos de opciones de hardware para los servicios individuales. Algunos servicios pueden ser altamente informatizados, mientras que otros pueden requerir un uso intensivo de procesamiento o de memoria.

  En lugar de implementar una aplicación entera en un tipo de hardware de computación de alto rendimiento, los micro servicios se pueden implementar en diferentes servidores de configuración de hardware para maximizar la potencia de cálculo. Esto da lugar a adoptar una plataforma en la nube para minimizar el coste, maximizar el rendimiento y la capacidad de escalar a demanda sin afectar a los sistemas en absoluto (Requena, 2016).

- **Escalabilidad :** La arquitectura orientada a micro servicios permite la independencia de escalabilidad, por lo que sólo los servicios que necesitan escalar lo hacen de manera independiente de los otros micro servicios, en lugar de escalar toda la aplicación, lo que puede ahorrar una gran cantidad de recursos informáticos. Por ejemplo, si hay más tráfico a un servicio

de búsqueda en comparación con un servicio de autenticación, sólo escalas el servicio de búsqueda.

Debido a su tamaño, flexibilidad de implementación y el contexto limitado, los micro servicios son más fáciles de escalar y más rápido de crear que una instancia de aplicaciones grandes y monolíticas (Requena, 2016).

- **Identificación y corrección rápida de problemas:** La descomposición y el despliegue de las aplicaciones en servicios más pequeños reducen el tiempo para identificar y corregir fallos. Además, permite el aislamiento de fallos (por ejemplo: pérdidas de memoria) (Requena, 2016).
- **Independencia en la pila de tecnologías:** Al estar débilmente acoplados, los micro servicios también eliminan muchas limitaciones en la pila de tecnología, lo que permite a los desarrolladores ser capaces de elegir el lenguaje de programación en el que sientan más cómodos. Esto hace que sea fácil reemplazar un servicio cuando existan mejores tecnologías disponibles. Por ejemplo, uno micro servicio podría ser escrito en Java, mientras que otro se podría escribir en node.js. A medida que evolucionan mejores patrones en cada pila, el código puede ser fácilmente rediseñado y redistribuido (Requena, 2016).

SOA es más que una arquitectura de software, es una filosofía en la cual se busca modelar un sistema de software tal como ocurre en “la vida real”.

Normalmente se relaciona mucho los servicios web con SOA (*Service Oriented Architecture*) y se llega a tener la errónea idea que SOA es servicios web. Siendo cierto de que SOA no es más que una filosofía en donde, hasta el momento, la tecnología de servicios web son los que mejor se acoplan a la misma. En la arquitectura de SOA se definen los siguientes aspectos:

- Cómo localizar un servicio.
- Cómo conseguir que se comuniquen los diferentes servicios.
- Cómo encaja cada servicio dentro del sistema.

El poder de SOA radica en su capacidad para expresar cuestiones técnicas en términos de negocio y permitir a las empresas crear soluciones.

Dentro de SOA los servicios encapsulan lógica en el interior de sus propios contextos para retener su independencia. El término contexto se refiere a tareas del negocio, entidades del negocio o alguna agrupación lógica. El alcance del contexto puede variar, por lo tanto no hay una forma de medir su extensión. Un buen ejemplo de una forma de automatización del negocio típica es la implementación de procesos de negocio. Estos procesos están constituidos por lógica que dicta las acciones realizadas por la solución. La lógica es descompuesta en una serie de pasos que son ejecutados en una secuencia determinada de acuerdo a las reglas del negocio. En la siguiente figura se muestra el enfoque de servicios aplicado a procesos de negocio donde se ve que hay agrupaciones de lógica encapsuladas en un servicio.

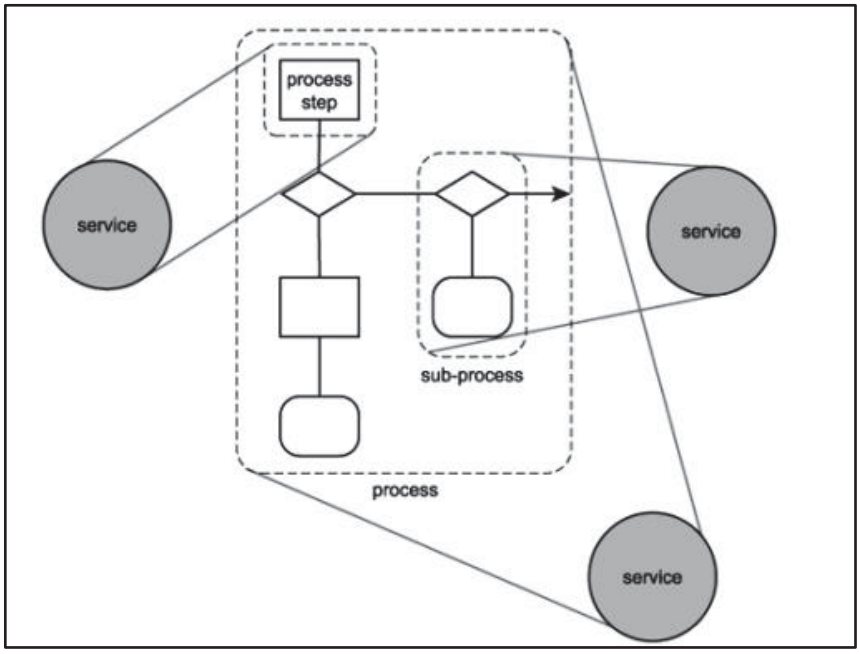


Fig. 2.3 Servicios y un proceso de negocio

Fuente: (Erl-Thomas, 2005)

Los servicios participan todo el tiempo durante la ejecución de los procesos de negocio brindando soporte tecnológico que permita facilitar las tareas y evitar inconsistencias lo cual la hace una arquitectura adecuada para el presente proyecto, además de basarse en la noción de que es

beneficioso descomponer un problema en sub problemas, esto permite que la lógica requerida para resolver el problema sea descompuesta en una colección más pequeña y de fácil solución.

- **Principio de reutilización:** los servicios son inherentemente reusables aun si las especificaciones del requerimiento no contemplen la existencia de ellas. Al aplicar diseños estándares los servicios son potencialmente reusables y permiten que cambios futuros se logren con menor esfuerzo. En otras palabras, debido que un servicio encapsula lógica dentro de un contexto definido y si se aplican diseños estándares dentro de esa lógica, se tiene como resultado un servicio genérico el cual puede ser utilizado por otros servicios o bien dentro de distintos procesos de negocio promoviendo la reusabilidad, interoperabilidad de aplicaciones, composición y la creación de servicios utilitarios.
- **Principio de contrato compartido:** Los contratos de los servicios proveen una definición formal de la terminal del servicio, cada una de sus operaciones, cada entrada y salida soportada por cada operación, reglas y características.
- **Principio del acoplamiento débil**: Es una condición en donde un servicio adquiere conocimiento de otro mientras conserva independencia de ese servicio. Se consigue el acoplamiento débil a través del uso de contratos que permitan a los servicios interactuar dentro de parámetros predefinidos. El acoplamiento débil permite al sistema ser capaz de responder a los cambios imprevistos de forma eficiente. Este principio tiene relación con el de reusabilidad debido que mantiene a los servicios libres de vínculos con otros. Esto hace que sea mucho más sencillo reutilizarlo.
- **Principio de la ocultación de la lógica**: Este principio permite a los servicios actuar como cajas negras, escondiendo sus detalles del mundo exterior.
- **Principio de composición**: Un servicio puede representar cualquier cantidad de lógica de cualquier tipo de fuente, incluyendo otros servicios. El objetivo es asegurarse que los servicios puedan participar en conjunto para cumplir cierta tarea compleja lo cual promueve la

reusabilidad por lo tanto los servicios deben ser creados de forma estándar. También es importante tener un apropiado nivel de granularidad para maximizar las oportunidades de composición.

- **Principio de autonomía**: La autonomía requiere que el rango de la lógica expuesta por el servicio exista dentro de un límite explícito. Esto permite eliminar dependencias de los demás servicios y provee total gobierno de lo que se ejecute dentro de ellos. La autonomía de los servicios es una de las principales consideraciones cuando se decide si la lógica de las aplicaciones debería ser dividida en servicios o si debería ser agrupada dentro de un contexto de un servicio.
- **Principio de sin estado**: Por estado se entiende a la información específica que se encuentra en el servicio mientras se ejecuta la actividad actual y mientras eso ocurra el servicio no está disponible para nadie. Por lo tanto el servicio debería minimizar la cantidad de información del estado que administre. Este principio promueve la reusabilidad y la escalabilidad.
- **Principio de descubrimiento:** Ayuda a evadir la creación accidental de servicios redundantes que implementen lógica redundante ya que provee un mecanismo para encontrar los servicios que están disponibles. Esto se convierte en algo efectivo en la infraestructura de TI y puede soportar numerosas implementaciones de SOA. Puede verse como un directorio telefónico en el cual se ven todos los posibles candidatos (tanto nombre, teléfono y hasta dirección) para algún servicio que se necesite.

2.7.2 API

Una API (*Application Programming Interface*) es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas: sirviendo de interfaz entre programas diferentes de la misma manera en que la interfaz de usuario facilita la interacción humano-software.

Las API son valiosas, ante todo, porque permiten hacer uso de funciones ya existentes en otro software (o de la infraestructura ya existente en otras plataformas) para no estar reinventando la rueda constantemente, reutilizando así código que se sabe que está probado y que funciona correctamente. En el caso de herramientas propietarias (es decir, que no sean de código abierto), son un modo de hacer saber a los programadores de otras aplicaciones cómo incorporar una funcionalidad concreta sin por ello tener que proporcionar información acerca de cómo se realiza internamente el proceso. (Lowe, 2014)

2.7.3 GraphQL

GraphQL es un lenguaje de consulta para el desarrollo de una API, y un tiempo de ejecución del lado del servidor para ejecutar consultas mediante el uso de un sistema de tipo que se defina para los datos. GraphQL no está vinculado a ninguna base de datos específica o motor de almacenamiento y, en cambio, está respaldado por su código y datos existentes.

Un servicio GraphQL es creado por la definición de tipos y campos en esos tipos y luego proporcionando funciones para cada campo[6].

Algo importante a tener en cuenta es que GraphQL no es una librería o *framework*. Es una especificación de cómo implementarlo en cualquier lenguaje y a su vez existen implementaciones ya hechas en lenguajes como:

JavaScript, Ruby, Python, Scala, Java, Clojure, Go, PHP, .NET, etc. También existen clientes para consumir un API GraphQL desde JS, iOS, Android, React, Angular, etc[7].

Normalmente en una API de tipo REST, se realizan peticiones y se consumen mediante métodos tradicionales (usando GET, POST, PUT, DELETE) de ésta forma, cuando se necesita de

[6] *Fuente : https://graphql.org/learn/*
[7] *Fuente: https://platzi.com/blog/introduccion-a-graphql/*

algún recurso, se hace algo cómo: */api/accounts/* para obtener las cuentas asociadas, POST para agregar una nueva cuenta, PUT para actualizarla y DELETE para borrarla.

Con GraphQL sólo se tiene un punto de acceso que de manera predeterminada es /graphql y sólo se interactúa con él con el método POST, para obtener, modificar o crear los datos necesarios de la aplicación, se debe agregar al cuerpo la petición alguna de las siguientes palabras claves:

- Query
- Mutation
- Subscription

- **Query:** Este concepto representa una consulta que contiene los datos pedidos a la aplicación, ésta representación equivaldría al método GET con el que obtendríamos los datos de la aplicación. El resultado es un objeto con, específicamente los datos que se solicitó, sin agregar otros que no son requeridos.
- **Mutation:** Una vez resuelto el problema de la obtención de los datos se debe proveer al cliente de un mecanismo para actualizar o eliminar su cuenta, para eso se usan las mutaciones. Esta trabaja como una función, recibe algunos parámetros y devuelve una respuesta, con los datos necesarios para la actualización del lado del cliente.
- **Subscription:** Otra parte importante de GraphQL son las suscripciones, aunque estás son relativamente nuevas en las librerías para los servidores, ya que no hace mucho se han agregado a la especificación estándar y aún no se incluyen del todo en su código fuente. Estás simplemente agregan un manejador que notifica a quién se haya suscrito a al servidor (a través de conexiones *WebSockets*) y mantener una interacción *RealTime*.
- **Schema:** En las aplicaciones basadas en GraphQL son necesarios los esquemas de datos. Cuando se estructuran alguno de estos esquemas se indica la forma de obtener e interactuar con

los mismos. Un ejemplo de estos sería para la administración de cuentas en una aplicación. En donde se tiene un objeto "*Cuenta* "que contiene los siguientes datos:

- **I**d: un número, parámetro obligatorio
- **nombre:** un texto, parámetro obligatorio
- **Avatar**: un texto, parámetro opcional
- **Keys**: una lista de objetos de tipo *Keys*, que son opcionales

La propiedad *Keys* es una relación de llaves para validación de un cliente. Es un objeto dentro del mismo esquema de la aplicación.

Ahora se necesita una manera de obtener los datos del esquema para ello se define un Objeto con las siguientes especificaciones:

```
type Query {
  getAccount(id: Int!): Account
  getAccounts(): [Account]
}
```

Este permite ejecutar una función para obtener un cliente pasándole un id u obtener todos los clientes, después se definirá las mutaciones, que son las formas de interactuar con los datos.

- **Resolvers:** Al definir los esquemas simplemente se sabe qué hacer en una API, pero se necesita que respondan a los posibles *queries*, mutaciones o suscripciones, para eso son los *resolvers*, funciones que se encargan de procesar la petición a la API y responder solamente con los datos necesarios.

2.8 Despliegue

2.8.1 Contenedores (Docker)

Un contenedor consiste en un entorno de ejecución completo: una aplicación, además de todas sus dependencias, librerías y otros archivos binarios y de configuración necesarios para la ejecución, amarrados en un paquete. Al hacer la contenerización la plataforma de aplicación y sus

dependencias, diferencias en distribuciones de sistemas operativos e infraestructura subyacente son abstraídos.

Los contenedores permiten empaquetar y aislar aplicaciones con todas las librerías y dependencias que necesitan para poder ejecutarse.

Docker es un entorno de código abierto utilizado para el despliegue automático de las aligaciones en contenedores. Fue desarrollado por el equipo de *Docker, Inc. y* lanzado bajo la licencia Apache 2.0.

Docker agrega un motor de despliegue de aplicaciones sobre un entorno de ejecución de contenedor virtualizado. Está diseñado para proporcionar un entorno ligero y rápido en el cual se pueda ejecutar código, así como un eficiente flujo de trabajo para obtener ese código de la computadora de desarrollo a su entorno de prueba y luego en producción. Docker es increíblemente simple. De hecho, se puede comenzar con Docker en un host mínimo que ejecuta nada más que un núcleo Linux compatible y un Docker binario. La instalación de Docker se realiza mediante el gestor de paquetes de la distribución Linux de preferencia:

```
$ sudo apt-get install docker
```

La sentencia anterior, muestra los comandos para la instalación de Docker en el host, para este caso Debian. Una vez instalado se puede hacer uso de la gran cantidad de comandos disponibles, sin embargo, es necesario resaltar algunas más importantes que serán de utilidad para el presente proyecto:

Docker pull: Docker descargará automáticamente cualquier imagen que se le pida. Este comando permite descargar la imagen más reciente de algún repositorio. Sin embargo, también se puede especificar que versión de la imagen que se desea descargar.

Docker run: Permite la ejecución del proceso del contenedor dentro de un host local o remoto, cuando luego este se aísla en el sentido de que tiene su propio sistema de archivos, su propia red y su propio árbol de procesos aislado y separado del host.

Docker build: Este comando permite construir una imagen a partir de un archivo "*Dockerfile*"

Dockerfile: Docker puede generar imágenes automáticamente leyendo las instrucciones de un "*Dockerfile*". Un *Dockerfile* es un documento de texto que contiene todos los comandos que un usuario puede llamar en la línea de comandos para ensamblar una imagen.

La misión de Docker es proporcionar (Turnbull, 2014):

- Una manera fácil y ligera de modelar la realidad
- Una distribución logia de tareas
- Un despliegue rápido y eficiente

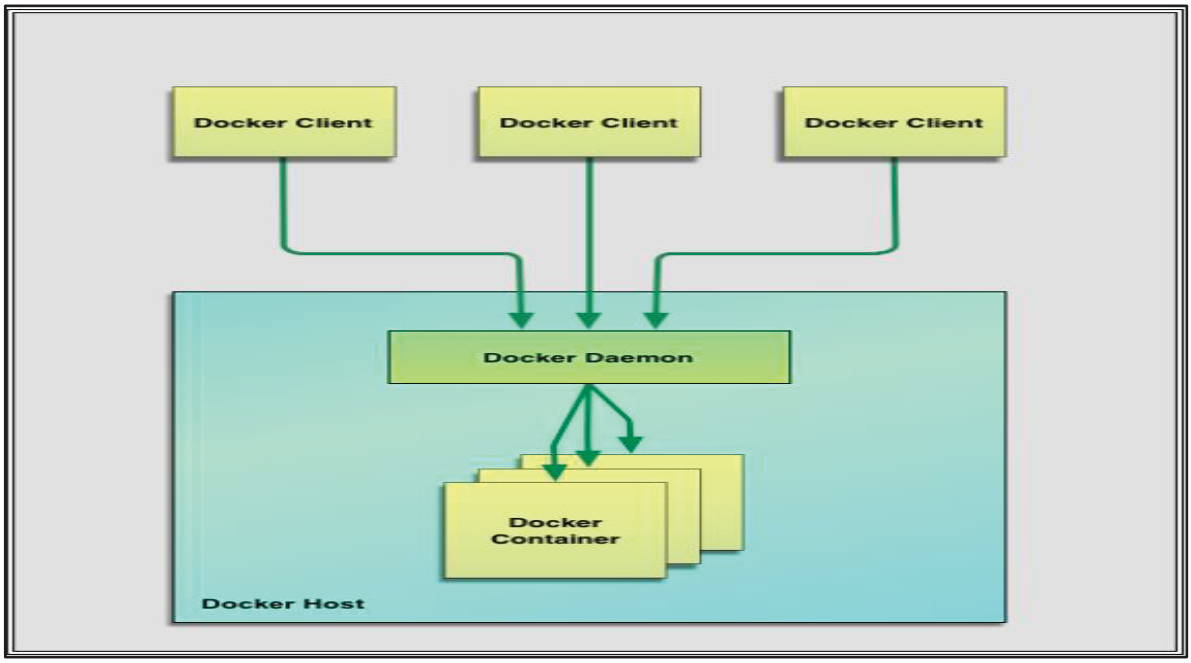


Fig. 2.4 Arquitectura Docker

Fuente: http://osgp88fat.bkt.clouddn.com/books/The%20Docker%20Book.pd

2.8.2 Servidor web (Nginx)

Servidor web se puede referir a hardware o software, o a ambos trabajando juntos.

- En cuanto a hardware, un servidor web es una computadora que almacena los archivos que componen un sitio web (ej. Documentos HTML, imágenes, hojas de estilos CSS y archivo JavaScript) y lo entrega al dispositivo del usuario final. Está conectado a internet y es accesible a través de un nombre de dominio.
- En cuanto a software, un servidor web tiene muchas partes encargadas del control sobre cómo tienen acceso los usuarios a los archivos, por lo menos un servidor HTTP. Un servidor HTTP es una pieza de software que comprende URLs (direcciones web) y HTTP (el protocolo que el navegador usa para ver las páginas web).

Al nivel más básico, siempre que un navegador necesite un archivo almacenado en un servidor web, el navegador lo solicita vía HTTP. Cuando la petición llega al servidor web correcto (hardware), el servidor HTTP (software) envía el archivo antes solicitado, también a través de HTTP (Mozzilla.org, 2018).

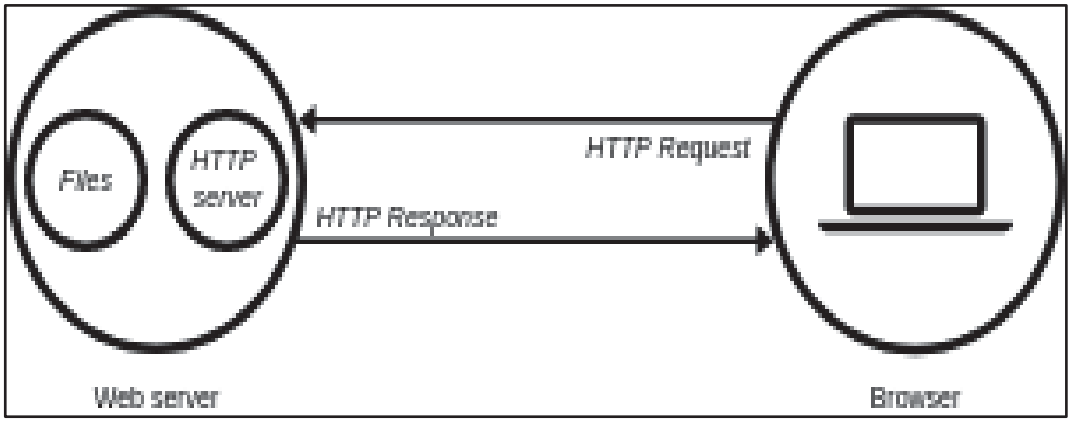


Fig. 2.5 Flujo Cliente - Servidor

Fuente: https://developer.mozilla.org/es/docs/Learn/Common_questions/Que_es_un_servidor_WEB

Dentro de las opciones de servidores web, Nginx se considera como una opción de código abierto de alto rendimiento, muy ligera y flexible frente a otras opciones como Apache.

Nginx es un servidor web de alto rendimiento, capaz de trabajar junto con diversas tecnologías de desarrollo y lenguajes. La asincronía es su característica fundamental, junto con su rapidez, ya que es un servidor web ligero. Todo ello, sin reducir el amplio rango de funcionalidades del servidor: avanzadas y, sobre todo, muy flexibles hace que sea una herramienta adecuada para su utilización como servidor web para el presente proyecto. Es capaz de adaptarse a muchas situaciones distintas y fue pensado desde el primer día para atender grandes necesidades de tráfico.

Puede servir páginas con programación del lado del servidor, con lenguajes como PHP, aunque es muy utilizado también para servir archivos estáticos.

Cuando se habla de Nginx es inevitable compararlo con Apache, ya que tradicionalmente Apache ha sido, y sigue siendo, el servidor web más utilizado. Sin embargo, Nginx ha sido capaz de solucionar algunos asuntos en los que Apache tiene peor rendimiento, como la concurrencia. Nginx es, de entrada y en máquinas similares, capaz de administrar más accesos concurrentes de lo que podría hacer Apache, con menores requisitos de memoria.

Además, como servidor web propiamente dicho, Nginx suele también configurarse como "proxy inverso", sirviendo como capa intermedia entre el cliente y el servidor de aplicaciones. Es muy habitual, por ejemplo, que sirva como capa intermedia a un servidor programado con NodeJS, permitiendo una mejor gestión de los recursos *backend* y aumentando la seguridad. Paralelamente, las configuraciones y casos de uso de Nginx son capaces de ofrecer soluciones encaminadas a la escalabilidad de las aplicaciones, como el balanceo de carga o el cacheo de páginas.

- **Estadística de uso de Nginx vs Apache:** Como se sabe Apache es otro popular servidor web de código abierto. En términos de números en bruto, Apache es el servidor más popular que existe y es usado por el 47% de todos los sitios web con un servidor web conocido,

según W3Teach[8]. Pero en el ranking presentado en octubre del 2018, este porcentaje ha ido disminuyendo en comparación a la tendencia creciente de NGINX.

Si bien Apache es la opción general más popular, Nginx es, en realidad, el servidor web más popular entre los sitios web con mucho tráfico.

Cuando se analizan las tasas de uso por tráfico, NGINX impulsa el: (W3Teach, 2018)

- 58.7% de los 100,000 sitios más populares
- 66.4% de los 10,000 sitios más populares
- 63.9% de los 1,000 sitios más populares

De hecho, Nginx se usan los sitios con los usos más intensivos de recursos que existen, incluyendo Netflix, NASA, e incluso WordPress.com.

El uso de Apache, por otro lado, se mueve en la dirección opuesta a medida que aumente el tráfico web del sitio. Apache impulsa el: (W3Teach, 2018)

- 25.4% de los 100,000s sitios más populares
- 19.8% de los 10,000 sitios más populares
- 17.1% de los 1,000 sitios más populares

Si vemos los términos buscados en Google Search desde 2004 podemos ver que Apache ha estado disminuyendo de forma constante, mientras que NGINX ha experimentado un ligero crecimiento (Google-Analytics, 2018).

2.9 Desarrollo ágil de software.

Es una alternativa en la gestión tradicional de proyectos TI, donde se hace hincapié en el empoderamiento de las personas para colaborar y tomar decisiones en equipo, además potencia la planificación continua, pruebas permanentes y la integración conjunta del código y los despliegues. (EvaluandoSoftware.com, 2018)

[8] *Fuente: https://w3techs.com/technologies/cross/web_server/ranking*

Los proyectos de tecnologías de información (TI), al igual que cualquier proyecto el recurso humano toma importancia en el proceso de obtención del producto, sean bienes o servicios; en el caso de servicios, especialmente; el equipo, grupo o recurso humano cobran especial importancia; lo mismo ocurre en el sector de las tecnologías, incluso con más fuerza, o al menos es lo que propone el desarrollo ágil de software, donde el equipo encargado de producir la aplicación o, software en general, se empodera tanto que no existe una jerarquía en el equipo de trabajo, y se refuerzan aspectos como la colaboración, toma de decisiones, la planificación dentro del equipo; obteniendo resultados agiles.

El proceso ágil toma en cuenta los siguientes supuestos sobre los proyectos de software:

- Es difícil predecir con antelación cuáles requerimientos persistirán y cuáles cambiarán.
- Para muchos tipos de software el diseño y la construcción se entrelazan.
- Las fases de análisis, diseño, construcción y pruebas no son tan predecibles como se quiere. (Balderas Contreras , 2011)

El desarrollo ágil de software puede tener flaquezas que son tomadas en cuenta en el proceso, por ejemplo, lo poco predictivos que son; sin embargo, esto ocurre por la agilidad que se aplica en el proceso, lo cual a su vez se complementa con la alta adaptabilidad que el producto final de esta metodología tiene.

2.9.1 Metodología Scrum.

2.9.1.1 Generalidades

La metodología Scrum para el desarrollo ágil de software representa un punto de partida de la gestión en cascada. De hecho, Scrum y otro tipo de procesos ágiles se inspiraron en sus limitaciones. La metodología Scrum enfatiza en la comunicación y colaboración, el funcionamiento del software, y la flexibilidad de la que dispone para adaptarse a las emergentes realidades de las empresas - todos los atributos de los que carecía el modelo de cascada. (Urteaga Pecharromán, 2015, pág. 27)

Dentro del desarrollo ágil de software, la metodología Scrum es una de las tantas formas de seguir este proceso, la cual se focaliza en aspectos como la comunicación, colaboración, el funcionamiento del software y la flexibilidad a la adaptación, todo esto dentro de la gestión en cascada, que es la que se usa en la ingeniería de software la misma que se encuentra en función al ciclo de vida de un programa.

"De todas las metodologías ágiles, Scrum es única porque introduce la idea del control empírico de los procesos. Esto significa que Scrum utiliza el progreso real de un proyecto para planificar y concertar los lanzamientos". (Urteaga Pecharromán, 2015). Los procesos en scrum son continuamente evaluados y pasan por continuas mejoras, el resultado no se analiza al final, sino por el contrario estos se analizan constantemente con todos los agentes que intervienen, desde el equipo creador hasta los beneficiarios.

Por lo cual criterios de planificación atienden de manera personalizada a los requerimientos del software o programa en creación; por lo que los errores posibles en el proceso de creación disminuyen, al redirigir y rediseñar constantemente los resultados en cada etapa del proceso, por la búsqueda de un resultado final lo más adecuado posible y que satisfaga las necesidades específicas al que fue direccionado además que tiene una gran adaptabilidad.

"En Scrum, los proyectos se dividen en ritmos de trabajo breves, conocidos como *Sprints*. Normalmente, tienen una, dos o tres semanas de duración". (Urteaga Pecharromán, 2015). En cuanto a terminología, la metodología Scrum introduce un nuevo término, los llamados *Sprints,* los cuales son las fases del proceso, o las subdivisiones de trabajo que comprende la realización de todo el trabajo.

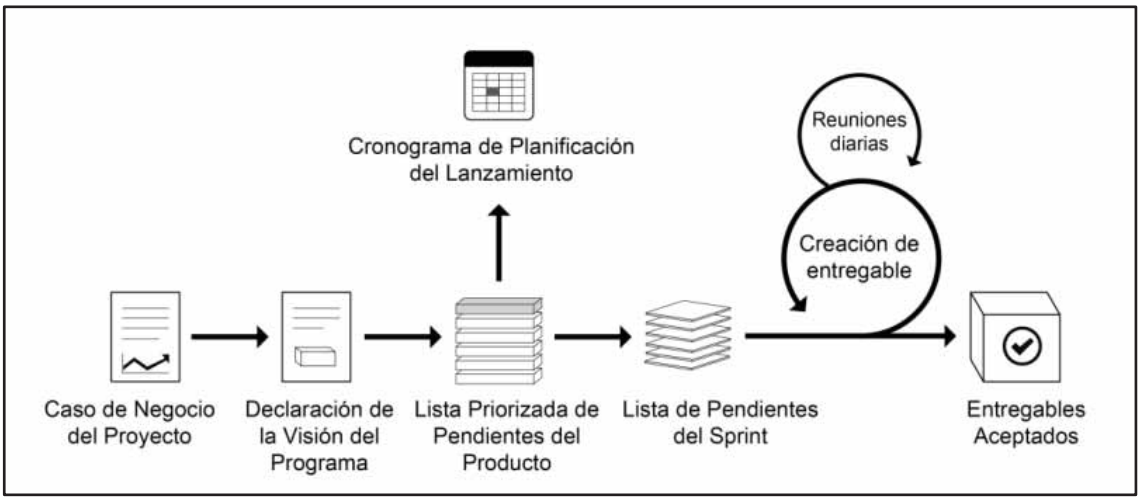


Fig. 2.6 Ciclo de Scrum para un sprint

Fuente: https://www.scrumstudy.com/SBOK/SCRUMstudy-SBOK-Guide-2016-spanish.pdf

Scrum se basa en la teoría de control de procesos empírica o empirismo. El empirismo asegura que el conocimiento procede de la experiencia y de tomar decisiones basándose en lo que se conoce. Scrum emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control del riesgo. Tres pilares soportan toda la implementación del control de procesos empírico: transparencia, inspección y adaptación (Schwaber & Sutherland, 2014).

- **Transparencia:** Transparencia permite que todas las facetas de cualquier proceso de Scrum sean observadas por cualquier persona. (Guía SBOK™, 2013)
- **Inspección:**

  Inspección en Scrum es representado a través de las siguientes acciones:

  - El uso de una Tabla de Scrum común y otros radiadores de información que muestran el progreso del Equipo Scrum en completar las tareas del Sprint actual.
  - La colección de retroalimentación del Cliente y otros *stakeholers* durante los procesos de Desarrollo de Épica(s), Crear la Lista de Pendientes del Producto o, y Realizar la Planificación del lanzamiento.
  - Inspección y la aprobación de los entregables por el *Product Owner* y el Cliente en el proceso de Demostrar y Validar el Sprint. (Guía SBOK™, 2013).

- **Adaptación**: Adaptación sucede cuando el Equipo Principal de Scrum y los *stakeholders* aprenden a través de la transparencia y la Inspección y luego se adaptan al hacer mejoras en el trabajo ya en progreso. (Guía SBOK™, 2013)

Estos tres principios rigen el seguimiento y control que la metodología Scrum tiene, en cuanto al control del proceso, por lo que en cada paso que el equipo da, estos están presentes; donde los mismos participes del proyecto son los que aplican estos principios. El principio de transparencia, de manera individual, hace posible que el seguimiento tenga características de simplicidad en la observación. Por otro lado, el principio de inspección, usa instrumentos estandarizados, los que serán usados por todos aquellos que tengan que ver con el proyecto y deseen generar o analizar los datos. Y finalmente el principio de Adaptación nace como resultado de los anteriores principios, donde los agentes al estar en contacto adaptan sus posibles observaciones, quejas y demás aspectos positivos o negativos, para realizar mejoras, para el proyecto que ya está en marcha, interiorizando así todos los problemas que se generan en el proceso.

2.9.1.2 Roles

Product Owner (propietario del producto), *Scrum Master* (especialista en Scrum) y *Srum Team* (miembros del equipo) (Urteaga Pecharromán, 2015) estos son los tres actores fundamentales en la metodología scrum; los cuales tienen y cumplen determinados roles en el proceso.

A. **Product Owner (propietario del producto):** el *Product Owner* se encarga de comunicar la visión del producto al equipo de desarrollo. Él/ella también debe representar el interés del cliente por medio de los requisitos y la priorización. (Urteaga Pecharromán, 2015)
B. **Scrum Master (especialista en Scrum):** El *Scrum Master* actúa como enlace entre el *Product Owner* y el equipo. El *Scrum Master* no dirige al equipo. Él/ella se encarga de evitar

cualquier barrera que impida al equipo lograr sus objetivos de sprint. (Urteaga Pecharromán, 2015)

C. **Team Member (miembros del equipo):** En la metodología Scrum, el equipo es el responsable de terminar el trabajo. En cada sprint, el equipo es responsable de determinar cómo va a lograr acabar el trabajo. (Urteaga Pecharromán, 2015)

Dentro de los agentes que participan en el proceso, o al menos los que lo hacen activamente, estos agentes están compenetrados e interrelacionados en cuanto a sus funciones porque las es justo por lo que se caracteriza esta metodología. Se observan un equipo que se relaciona directamente con creación del programa; un propietario del proyecto; y un ente que sirve de nexo, uniendo a estos dos grupos, el equipo y propietario.

2.9.1.3 Procesos

Los procesos de Scrum abordan las actividades y el flujo específico de un proyecto Scrum. En total hay diecinueve procesos que se agrupan en cinco fases. (Guía SBOK™, 2013). En general, estos son los procesos que toda la metodología Scrum sigue, sin embargo, esto pude ser adaptado para proyectos de software específicos; es decir se pueden quitar o agregar fases o los mismos procesos de acuerdo a la necesidad; Además que la metodología Scrum es bastante flexible al pasar, durante el proceso, por constantes re direccionamientos; motivo por el cual no puede existir una metodología rígida a seguir.

Tabla 2.3 Fases y Procesos de la Metodología Scrum

Fase	Procesos
Initiate (Iniciar)	1. *Crear la Visión del Producto o* 2. *Identify Scrum Master and Stakeholder(s)* 3. *Formar el Equipo Scrum* 4. *Desarrollode Épica(s)* 5. *Crear la Lista de Pendientes del Producto o* 6. *Realizar la Planificación del Release*
Plan and Estimate (Planear y Estimar)	7. *Crear Historias de Usuarios* 8. *Aprobar, Estimar y Comprometerse a las Historias de los Usuarios* 9. *Crear Tareas* 10. *Estimar el Trabajos* 11. *Crear la Lista de Pendientes de Sprint*
Implement (Implementar)	12. *Crear Entregables* 13. *Realizar un Standup Diario* 14. *Mantenimiento Priorizado de los Pendientes del Producto o*
Review and Retrospect (Revisión y Retrospectiva)	15. *Convocar Scrum de Scrums* 16. *Demostrar y Validar el Sprint* 17. *Retrospectiva del Sprint*
Release (Lanzamiento)	18. *Envío de los Entregables* 19. *Retrospectiva del Proyecto*

Fuente: Adoptado de "Guia para el conocimiento de Scrum" SCRUMstudy, 2013

PROCESOS DE LA FASE INICIAR

- **Crear la Visión del Producto**: En este proceso, el Proyecto Business Case es revisado para crear un Declaración de la Visión del Proyecto que servirá de inspiración y proporcionará un enfoque de todo el Proyecto. El *Product Owner* se identifica en este proceso. (Guía SBOK™, 2013). En todo proyecto la visión planteada es ambiciosa, lo mismo ocurre con esta metodología, donde la visión propuesta será de continua inspiración para el equipo, además de contener el objetivo esperado y fundamental de todo el proyecto.
- **Identificar al Scrum Master y al/a los Stakeholder(s):** En este proceso, el *Scrum Master* y los *stakeholders* se identifican utilizando criterios de selección específicos. (Guía SBOK™, 2013). Este proceso, se define por limitar a los *stakeholders*, es decir a todos los

individuos o grupos que, de manera positiva o negativa, se ven afectados con el proyecto, y esto se realiza por selección específica.

- **Formar el Equipo Scrum:** En este proceso, se seleccionan a los miembros del Equipo Scrum. Normalmente, el *Product Owner* es el responsable principal de la selección de los miembros del equipo, pero a menudo lo hace en Colaboración con el Equipo Scrum. (Guía SBOK™, 2013). Proceso por el cual se ven definidos todo el equipo o recuso humano que serán los encargados de llevar a cabo todos los procesos.
- **Desarrollo de Épica(s):** En este proceso, el Declaración de la Visión del Proyecto sirve como la base para el desarrollo de épicas. Reunión de Grupo de Usuarios pueden tomar lugar para discutir el/los Épica(s) apropiado(s). (Guía SBOK™, 2013)
- **Crear la Lista de Pendientes del Producto:** En este proceso, los Épica(s) son elaborados, refinados, y luego priorizados para crear el *Product Backlog* o del proyecto. Lo que se conoce como Criterio de Terminado también se establece en este punto. (Guía SBOK™, 2013)
- **Realizar la Planificación del Release:** En este proceso, el *Scrum Team* revisa los Historias de Usuarios en el Priorizada *Backlog* Producto o para desarrollar un Cronograma de Planificación del Lanzamiento, que es esencialmente un Programa de implementación por fases que se puede compartir con los *stakeholders* del Proyecto. Los Longitud de los *Sprints* también se determinan en este proceso. (Guía SBOK™, 2013)

La primera fase es donde todo el recurso humano se pone en contacto, el proceso se inicia con el *Product Owner* que define la visión y participa en la primera selección del equipo scrum, ya que, al tener los primeros componentes del equipo, estos son los que participan en la selección del resto de equipo. El desarrollo del producto inicialmente se basa en el desarrollo de épicas, las cuales son las historias de los clientes, o el requerimiento de los clientes finales, de esta manera se dividirá en los futuros procesos en *sprints*.

PROCESOS DE LA FASE PLANEAR Y ESTIMAR

- **Crear Historias de Usuarios:** En este proceso, Historias de Usuarios y sus afines Criterio de Aceptación de la Historia del Usuario se crean. (Guía SBOK™, 2013)
- **Aprobar, Estimar y Comprometerse a las Historias de los Usuarios:** En este proceso, el *Product Owner* aprueba las Historias de Usuarios para un Sprint. (Guía SBOK™, 2013)
- **Crear Tareas:** En este proceso, los Historias de Usuarios Aprobadas, Estimadas y Comprometidas se dividen en tareas específicas y se compilan en una Lista de Tareas. (Guía SBOK™, 2013)
- **Estimar el Trabajos:** En este proceso, el *Scrum Team*, en las reuniones de *Task Estimation*, estima el esfuerzo necesario para realizar cada tarea de la Lista de Tareas. (Guía SBOK™, 2013)
- **Crear la Lista de Pendientes de Sprint:** En este proceso, el *Scrum Team* tiene un Reunión de Planificación del Sprint donde el grupo crea un Pendientes del Sprint que contiene todas las tareas que deben completarse en el Sprint. (Guía SBOK™, 2013)

La fase de planear y estimar se basa en las historias que anteriormente fueron definidas, en esta fase se inicia con la aplicación de criterios para que esas historias sean verídicas, para que posteriormente pasen por la aprobación del *Product Owner*; luego se pasa a generar el listado de tareas; tareas que requieren de trabajo, lo cual se estima; estimada el nivel de esfuerzo requerido se procede a una reunión donde se pone en evidencia los procesos específicos que aún no fueron considerados, (sprints pendientes) y finalmente se delimita el sprint.

PROCESOS DE LA FASE IMPLEMENTAR

- **Crear Entregables:** En este proceso, el Equipo Scrum trabaja en las tareas del Pendientes del Sprint para crear Entregables del Sprint. (Guía SBOK™, 2013)

- **Realizar un Standup Diario:** En este proceso, todos los días se lleva a cabo una reunión *Timeboxed* altamente concentrada llamada Reunión Diaria de *Standup*. (Guía SBOK™, 2013)
- **Mantenimiento Priorizado de los Pendientes del Producto:** En este proceso, *Product Backlog* se actualiza y mantiene continuamente. (Guía SBOK™, 2013)

Esta fase de implementar; se inicia con la creación de entregables, es decir sprints entregables, para facilitar la revisión y las pruebas; para que, en las reuniones diarias, para que las correcciones sean más reales y de fácil comprensión para que el grupo realiza las correcciones, y esto se da de manera constante.

PROCESOS DE LA FASE REVICION Y RETROSPESTIVA

- **Convocar Scrum de Scrums:** En este proceso, los representantes del Equipo Scrum convocan una reunión de *Scrum of Scrums* (SoS) en intervalos predeterminados o cuando sea necesario para colaborar y realizar un seguimiento de sus respectivos progresos, impedimentos, y las dependencias entre los equipos. (Guía SBOK™, 2013)
- **Demostrar y Validar el Sprint:** En este proceso, el Equipo Scrum les demuestra el *Sprint* Deliverable al *Product Owner* y a los relevantes *stakeholders* en una reunión de revisión del Sprint. (Guía SBOK™, 2013)
- **Retrospectiva del Sprint:** En este proceso, el *Scrum Master* y el *Scrum Team* se reúnen para discutir las lecciones aprendidas a lo largo del Sprint. (Guía SBOK™, 2013)

Fase en el cual se llevan a cabo reuniones internas, es decir, los representantes de cada proceso de reuniones para poder explicar las deficiencias y superar problemas a la vez de realizar una evaluación compartida; realizada estas reuniones se inicia con el sustento del producto generado hasta el momento en el que se llevaron las ultimas correcciones, esto ante el *Producto Owner* y los demás agentes que son afectados con el proyecto (*stakeholders*) de acuerdo al orden de impacto de estos.

PROCESOS DE LA FASE LANZAMIENTO

- **Envío de los Entregables:** En este proceso, los Entregables Aceptados se les entregan o trasladan a los *stakeholders* pertinentes. (Guía SBOK™, 2013)
- **Retrospectiva del Proyecto:** En este proceso, que completa el proyecto, los *stakeholders* de la organización y el *Scrum Team* se reúnen para la retrospectiva del Proyecto e identificar, documentar e internalizar las lecciones aprendidas. (Guía SBOK™, 2013)

La fase de lanzamiento se caracteriza inicialmente por la entrega material de los avances (entregables aceptados) a los *stakeholders* más interesados; esta entrega, volverá al proyecto mediante un análisis de retrospección, donde se analizan todos los problemas que se tuvieron en el proceso los mismos que deben ser internalizados para que posteriormente no afecte el proceso de mejoras del programa.

Es muy importante señalar que aunque la metodología comprende 19 procesos agrupados en 5 fases; por la naturaleza y el objetivo de "Diseño e Implementación" de este proyecto solo se utilizaron las primeras 3 fases (Inicio, Planeación y Estimación, Implementación) que se documentaron en los capítulos siguientes, dejando para el futuro la documentación de las 2 restantes de acuerdo a la evolución del proyecto en un proceso de puesta en funcionamiento.

CAPÍTULO III: DESARROLLO DEL PROYECTO

En este capítulo se describe el proceso de diseño e implementación de la plataforma, de acuerdo a los objetivos planteados en esta tesis. Como se describió en el capítulo I, la metodología que sirvió de referencia para el desarrollo de esta plataforma es Scrum, específicamente se realizaron 2 iteraciones o *Sprints*, en las cuales en cada una de ellas se describen 3 fases (Inicio, Planeación e Implementación) y en cada fase los procesos que fueron de utilidad para la obtención de una mejor visión, planeación e implementación del proyecto. Para un mejor control de los procesos de desarrollo de la plataforma, se elaboró un cronograma con los tiempos y las actividades realizadas para cada fase en cada iteración o *Sprint*, tal como se muestra en la figura siguiente:

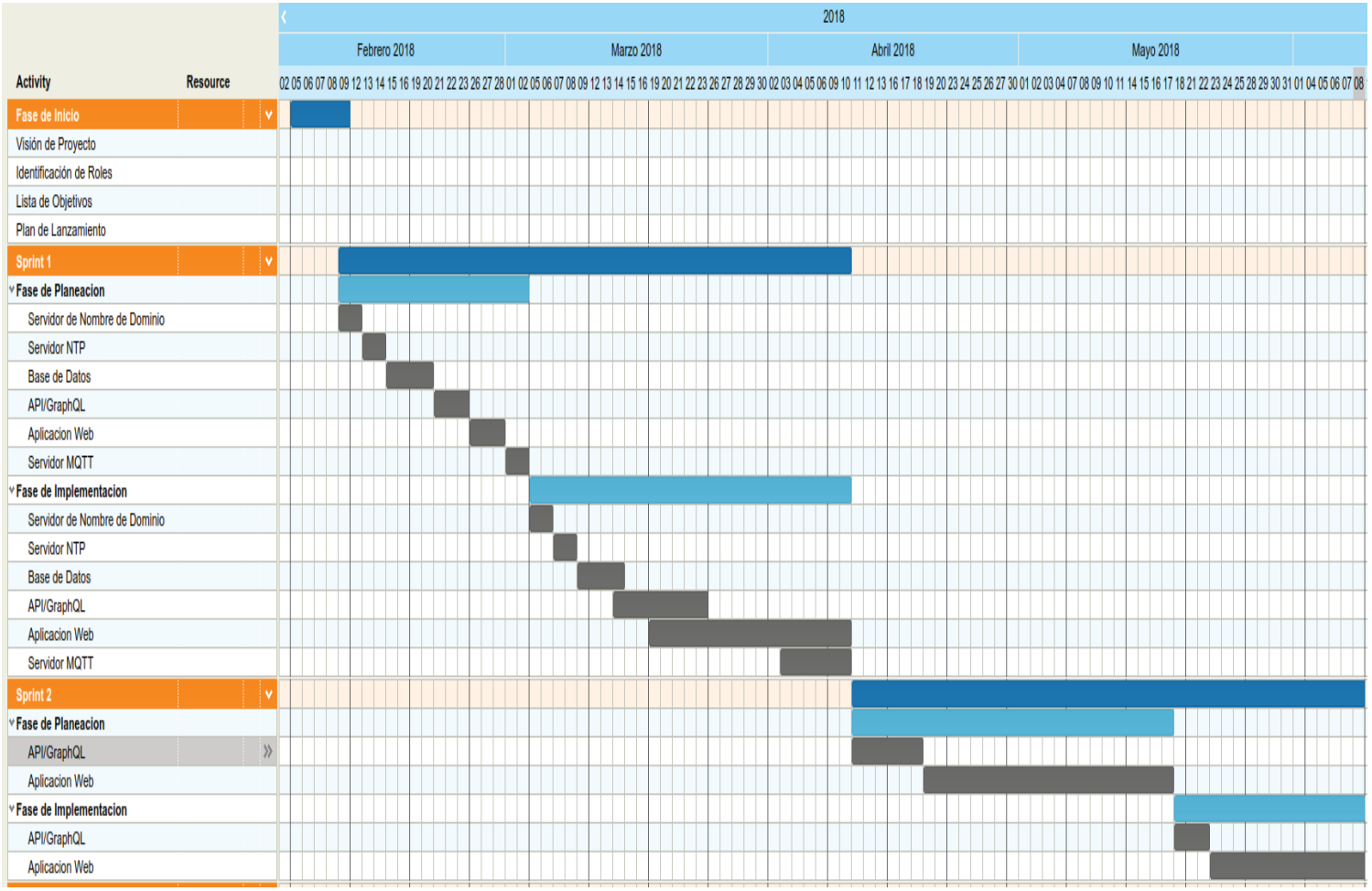


Fig. 3.1 Cronograma de Trabajo

Fuente: Elaboración Propia

3.1 Fase de inicio

3.1.1 Visión del proyecto

En esta etapa se define como visión del proyecto el diseño y/o implementación de un conjunto de componentes que trabajarán de forma conjunta y sistemática para la obtención de los resultados deseados, tanto para el correcto funcionamiento de la plataforma en general y la satisfacción de los requisitos por parte de los usuarios, cada uno de estos componentes estarán desplegados de manera independiente dentro de contenedores Docker y estos a su vez dentro de un solo servidor. A continuación, se describe cada uno de los componentes a diseñar y/o implementar:

1. **Servidor de Nombre de Dominio (DNS):** Es necesario la implementación de un servidor de nombre de dominio para que todas las transacciones sean certificadas por TLS, este protocolo de seguridad hace que los datos entre un servidor y navegador web, y en retroalimentación, viajen de manera íntegra y segura. Un certificado SSL confiable garantiza que los datos están encriptados, en ese momento se podrá asegurar que nadie pueda leer su contenido. Adicionalmente para la implementación de SSL es necesario un nombre de dominio que sea válido dentro de la red de área local en la que funcionara la plataforma.
2. **Servidor NTP:** Este servidor será de utilidad para la sincronización de la fecha y hora estándar UTC/GMT para el Perú, de los relojes instalados en los controladores semafóricos, esta sincronización se realizará periódicamente o cada vez que sea necesario.
3. **Servidor de Red MQTT o *bróker:*** Este componente se encargará de la gestión de la red y transmisión de todos los mensajes obtenidos y enviados hacia los controladores mediante el protocolo MQTT.
4. **API/GraphQL:** Esta interfaz permitirá la comunicación y administración de los datos entre diferentes aplicaciones y la base de datos mediante el uso de servicios que serán consumidos tanto por los clientes propios de la plataforma o terceros. La simplificación del consumo de los

servicios será posible gracias a la implementación de una capara de consultas con el lenguaje de consultas GraphQL.

5. **Base de Datos:** Es necesario la implementación de una base de datos no relacional por la gran cantidad de información que será necesario almacenar y recuperar rápidamente, además de ser este tipo de paradigma el más adecuado para este tipo de sistemas.

Cliente Web: Sistema desarrollado para el usuario final, que se accederá mediante un navegador web instalado en cualquier dispositivo conectado a la red. Este será una de las partes más importantes del proyecto, puesto que será el que más interacción tenga con el usuario final, para lo cual se debe priorizar cumplir tanto con la totalidad de los requerimientos y la usabilidad del sistema.

Luego de un análisis se llegó a la conclusión de que se deberán manejar niveles de acceso de acuerdo a tres perfiles definidos en convenio con los usuarios:

- Nivel Experto
- Nivel Operador
- Nivel Visor

A continuación, se muestra las historias de usuario recolectadas en base a la captura de requerimientos realizado con los usuarios de acuerdo a los perfiles definidos:

Tabla 3.1 Historia de Usuario - Administración de Usuarios

Como **Experto** quiero registrar, actualizar, eliminar los datos (usuario, email, DNI, contraseña, apellidos, nombres) de los usuarios de cuerdo al perfil correspondiente para tener una administración de verificar la identidad de cada personal además de permitir la utilización personalizada de recursos y privilegios de acceso.

Fuente: Elaboración Propia.

Tabla 3.2 Historia de Usuario - Administración de Módulos

Como **Experto** quiero crear, eliminar, actualizar los módulos (subsistemas) que contendrá la plataforma.

Fuente: Elaboración Propia

Tabla 3.3 Historia de Usuario - Administración de Vistas

Como **Experto** quiero crear, eliminar, actualizar las vistas que se mostraran en los módulos de acuerdo al perfil seleccionado para tener una mejor administración en base a los niveles de acceso.

Fuente: Elaboración Propia

Tabla 3.4 Historia de Usuario - Administración de Roles

Como **Experto** quiero asignar los roles que le corresponden a cada módulo para después asignarlos a cada usuario.

Fuente: Elaboración Propia

Tabla 3.5 Historia de Usuario - Administración de Sensores

Como **Experto** quiero registrar los datos de los sensores (Nombre, Código, Abreviación, Tipo) que estarán conectados a un dispositivo para después agregar variables, comandos, estados, configuraciones y así visualizar los datos que estos reporten.

Fuente: Elaboración Propia

Tabla 3.6 Historia de Usuario - Administración de Dispositivos

Como **Experto** quiero registrar los datos de los dispositivos (Nombre, Código, Icono) que pasarán a ser los controladores semafóricos y estarán conectados a la plataforma para agregar configuraciones, sensores y otras variables.

Fuente: Elaboración Propia

Tabla 3.7 Historia de Usuario - Administración de Intersecciones

Como **Operador** quiero registrar los datos de las intersecciones que se conectaran a la plataforma para después realizar las configuraciones de fases, horarios y geolocalización para así poder reportar información sobre el estado de cada intersección.

Fuente: Elaboración Propia

Tabla 3.8 Historia de Usuario - Administración de Configuraciones

Como **Experto** quiero registrar las configuraciones que se podrán realizar a cada controlador semafórico para después registrar sus valores en un panel de configuraciones para cada intersección semafórica.

Fuente: Elaboración Propia

Tabla 3.9 Historia de Usuario - Administración de Firmware

Como **Operador** quiero registrar, editar, eliminar las versiones de firmware disponibles en la plataforma para después enviar las actualizaciones hacia el controlador semafórico de la intersección seleccionada.

Fuente: Elaboración Propia

Tabla 3.10 Historia de Usuario –Actualización de Firmware

Como **Operador y Visor** quiero un mapa asistente tipo "*wizard*", con la geolocalización de las ubicaciones de todos los semáforos conectados a la plataforma para poder tener una vista general de todas las intersecciones y poder realizar acciones sobre algún inconveniente.

Fuente: Elaboración Propia

Tabla 3.11 Historia de Usuario - Enviar Firmware

Como **Operador** quiero enviar y grabar el firmware hacia el controlador semafórico seleccionado, mediante la red o puerto USB de controlador en el que se está trabajando.

Fuente: Elaboración Propia

La arquitectura de un sistema es la organización fundamental que incluye a sus componentes, sus relaciones entre ellos, el ambiente, los principios que dictan el diseño y evolución, así como también el comportamiento específico en función de la colaboración de sus elementos. La figura 3.2, muestra la arquitectura integral de la plataforma propuesta, la descripción de cada componente a desarrollar ha sido expuesta anteriormente y está de acuerdo a la numeración mostrada en la figura 3.2. A sí mismo la figura muestra cómo se permitirá el flujo de datos en forma bidireccional en dos fases; la primera con mecanismos que permita a cualquier controlador semafórico ya sea propio o de terceros que cumplan con las especificaciones del protocolo de comunicación MQTT conectarse y enviar datos, cifrados mediante el protocolo TLS, hacia un servidor de red o *bróker*, mientras que en la segunda fase empezará con la recepción de los datos por parte servidor de red o *bróker*, a partir de este punto se realizará dos procesos en paralelo que completan el flujo de datos de la arquitectura planteada; el primer proceso de persistencia empezando por el paso de datos del *bróker* hacia la API, y finalmente almacenada en una base de datos MongoDB. El segundo proceso consiste en la recepción y envío de datos desde el *bróker*, mediante MQTT, y la API de consultas al cliente o viceversa, esto permitirá la visualización de la información tanto en tiempo real como almacenada

de cada controlador semafórico. El segundo proceso también consta de enviar información almacenada hacia otras plataformas mediante la API de consultas.

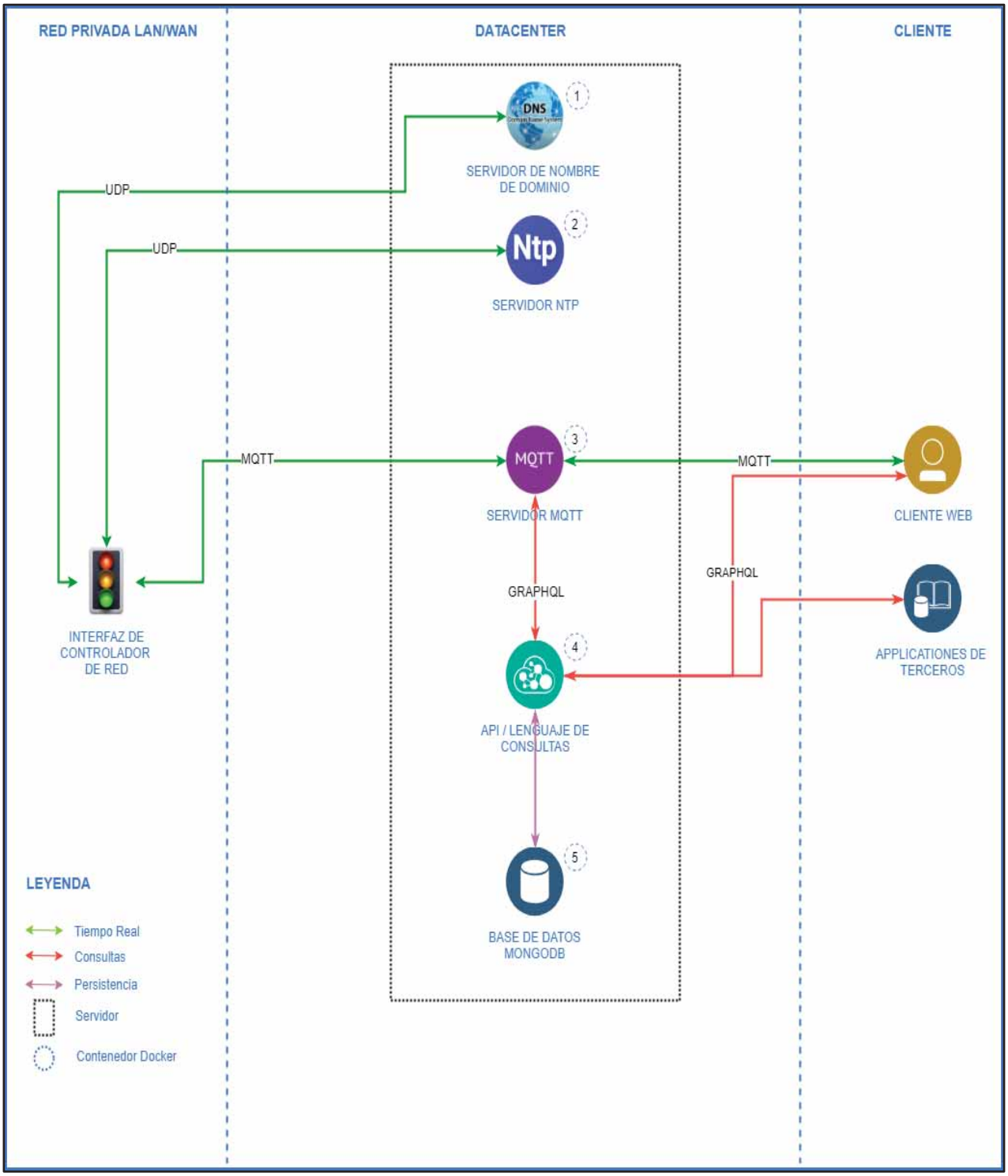


Fig. 3.2 Arquitectura de la Plataforma

Fuente: Elaboración Propia

3.1.2 Identificación de Roles

En esta parte se define e identifica los Scrum roles *y* la formación del equipo Scrum: *Scrum Master, Product Owner y Scrum Team,* tal como se muestra en la tabla 3.12.

Tabla 3.12 Scrum Roles de la Plataforma

Roles	Responsable
Product Owner	Ing. Juan Pablo Vizcardo Zúñiga
Scrum Master	Ing. Ronald Vizcardo Zúñiga
Scrum Team	Bengie Nick Serrano Quispe José Carlos Huallpamaita Quispe

Fuente: Elaboración propia

3.1.3 Lista de objetivos (Product Backlog)

La tabla 3.2, muestra una lista de los objetivos que pasaran a ser las actividades a realizarse, a su vez se hace una separación de tareas en orden de prioridad a nivel de toda la plataforma que se plantea implementar, donde los de prioridad **Alta** son las actividades primordiales y por lo cual se deben priorizar para que se pueda construir la plataforma, mientras que las de prioridad **Media** son las actividades secundarias que se realizaran una vez terminadas las de prioridad Alta.

Tabla 3.13 Lista de objetivos de la plataforma

N°	Objetivo	Prioridad	Tareas
1	Despliegue[9] del servidor de nombre de dominio	Alta	Instalar contenedor "BIND"
			Configurar contenedor "BIND"
2	Despliegue del servidor *NTP*	Alta	Instalar contenedor "*NTP*"
			Configurar contenedor "*NTP*"
3	Implementación de base de datos MongoDB	Alta	Diseño de la arquitectura de documentos
			Implementación de documentos y colecciones
			Despliegue en contenedor Docker
4	Implementación de la API/GraphQL	Alta	Implementación de *Querys*
			Implementación de *Mutations*
			Despliegue en contenedor Docker
	Implementación del Cliente Web	Media	Implementación MQTT Client js
			Vista administración de usuarios
			Vista Administración de módulos
			Vista Administración de vistas
			Vista Administración de roles
			Vista Administración de sensores
			Vista Administración de dispositivos
			Vista Administración de estados
			Vista Administración de Intersecciones
			Vista Administración de configuraciones
			Vista Administración de firmware
			Vista *wizard* mapa geolocalización de semáforos
			Vista enviar firmware
6	Implementación del Servidor MQTT	Media	MQTT sobre TCP Socket
			MQTT sobre *WebSocket*
			Despliegue en contenedor Docker

Fuente: Elaboración propia

[9] Para el contexto de esta tesis se definirá como despliegue al proceso de instalación, configuración y puesta en producción de algún componente de la plataforma.

3.1.4 Plan de lanzamiento

El propósito de la planificación del lanzamiento es la creación de un plan para el despliegue y lanzamiento del proyecto. Para este caso se considera a cada objetivo como un sub proyecto independiente, esta metodología es muy útil en el momento del despliegue y mantenimiento, puesto que cada uno de ellos correrá dentro de un contenedor Docker sobre cualquier distribución de sistema operativo Linux, minimizando así el riesgo de encontrar algún conflicto entre los paquetes necesarios y facilitando más el proceso de mantenimiento.

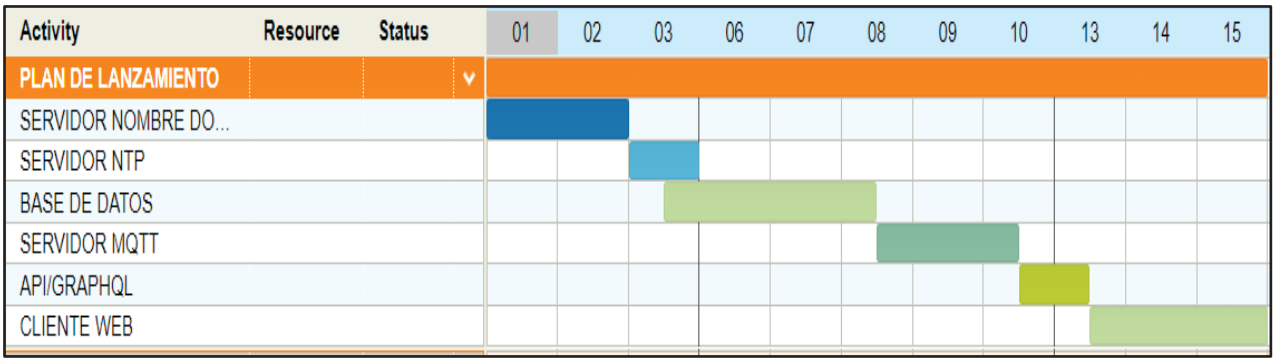


Fig. 3.3 Plan De Lanzamiento de Cada Uno de Los Componentes de la Plataforma en un Total De 15 Días

Fuente: Elaboración Propia

3.1.5 Configuración del entorno de trabajo

A continuación, se detalla las especificaciones técnicas de hardware y software con las que se cuenta tanto en el servidor como en el cliente para la implementación y funcionamiento de la plataforma.

Especificaciones técnicas de hardware y software

- **Servidor**

  Hardware:

  - Procesador Xeon de 4 núcleos.
  - 8 GB de memoria RAM.

- 256 GB de disco SSD (Solid-State Drive), para la partición principal.
- 1 TB de espacio libre en disco, para partición de resguardo de información.
- Pantalla LCD de 17”.
- Dispositivo de señalamiento (ratón) con rueda central.

Software:

- Sistema operativo Linux, cualquier distribución, se recomienda Ubuntu Server 18.04 o Centos 7.
- Docker versión 17.12.1.
  - ✓ Para la instalación del Docker se coloca la siguiente instrucción en la terminal del host, una vez instalado servirá para ser utilizado cuando sea necesario, a lo largo del proyecto:

    ```
    $ sudo apt install Docker
    ```

  - ✓ Las configuraciones para la instalación y configuración de los contenedores, serán explicadas a detalle en la descripción de las tareas correspondientes.
- Librerías adicionales para la implementación del Cliente Web
  - ✓ **encoding/json**: Codificación y decodificación de cadenas de texto con la estructura JSON.
  - ✓ **net/http**: Librería para el manejo de consultas HTTP.
  - ✓ **github.com/graphql-go/graphql**: Librería para el manejo de esquemas (Schema), mutaciones y consultas de GraphQL.
  - ✓ **gopkg.in/mgo.v2**: Librería para la conexión con el gestor de base de datos MongoDB.
  - ✓ **time**: Librería para la consulta de la hora del sistema operativo.
- Firewall. (Configurado correctamente, dependiendo de la distribución).

- **Cliente**

  Hardware

  - Procesador i5.
  - 6 GB de memoria RAM.
  - 256 GB de disco SSD (Solid-State Drive).
  - Pantalla de LCD de 22"
  - Dispositivo de señalamiento (ratón) con rueda central.
  - Tarjeta gráfica con hardware de aceleración 3D en modo color de 32-bit.

  Software

  - Sistema operativo Linux (cualquier distribución), Windows (10, 8, 7), Mac Os (Superior a Mavericks).
  - Navegador Web (se recomienda google Chrome o Firefox).

3.2 Sprint 1

3.2.1 Fase de planeación

En esta fase se contempla el análisis de las historias de usuario, captadas en la fase de inicio y la obtención del conjunto de actividades "**Tareas**", necesarias para el cumplimiento de estos requerimientos. Adicionalmente es necesario enlistar las tareas internas necesarias a ser realizadas en esta iteración, en base a las planteadas en la tabla 3.13. La siguiente tabla muestra la lista de actividades "*Sprint Backlog*" seleccionadas para esta primera iteración.

Tabla 3.14 Sprint Backlog 1

N° Actividad	Objetivo
1	Despliegue del servidor de nombre de dominio
	Instalar contenedor "BIND"
	Configurar contenedor "BIND"
2	Despliegue del servidor S*NTP*
	Instalar contenedor "*SNTP*"
	Configurar contenedor "*SNTP*"
3	Implementación de DB MongoDB
	Diseño de la arquitectura de documentos
	Implementación de documentos y colecciones
	Despliegue en contenedor Docker
4	Implementación de la API/GraphQL
	Implementación de *Querys*
	Implementación de *Mutations*
	Despliegue en contenedor Docker
5	Implementación del Cliente Web
	Implementación MQTT Client js
	Vista administración de usuarios
	Vista Administración de módulos
	Vista Administración de vistas
	Vista Administración de roles
	Vista Administración de sensores
	Vista Administración de Intersecciones
	Vista *wizard* mapa geolocalización de semáforos
6	Implementación del Servidor MQTT
	MQTT sobre TCP Socket
	MQTT sobre *WebSocket*
	Despliegue en contenedor Docker

Fuente: Elaboración Propia

3.2.1.1 Servidor de Nombre de dominio

3.2.1.1.1 Elaboración de tareas

Tabla 3.15 Tarea – Instalar contenedor "BIND"

Numero: 1	Responsable: Bengie Serrano
Nombre de Tarea: Instalar contenedor "BIND"	
Descripción: El responsable de la tarea deberá realizar la instalación del servidor de DNS BIND (*Berkeley Internet Name Domain*), dentro de un contenedor Docker con S.O Linux-Ubuntu 18.	

Fuente: Elaboración Propia

Tabla 3.16 Tarea – Configurar contenedor "BIND"

Numero: 2	Responsable: Bengie Serrano
Nombre de Tarea: Configurar contenedor "BIND"	
Descripción: El responsable de la tarea deberá realizar la configuración del servidor de DNS instalado en el contenedor Docker.	

Fuente: Elaboración Propia

3.2.1.2 Servidor NTP

3.2.1.2.1 Elaboración de tareas

Tabla 3.17 Tarea– Instalar contenedor "SNTP"

Numero: 3	Responsable: Bengie Serrano
Nombre de Tarea: Instalar contenedor "SNTP"	
Descripción: El responsable de la tarea deberá realizar la instalación del servidor SNTP (Protocolo simple de hora de red).	

Fuente: Elaboración Propia

Tabla 3.18 Tarea – Configurar contenedor "SNTP"

Numero: 4	Responsable: Bengie Serrano
Nombre de Tarea: Configurar contenedor "SNTP"	
Descripción: El responsable de la tarea deberá realizar la configuración del servidor NTP instalado en el contenedor Docker.	

Fuente: Elaboración Propia

3.2.1.3 Base de Datos "*Mongo DB*"

3.2.1.3.1 Elaboración de tareas

Tabla 3.19 Tarea – Diseño Arquitectura DB

Numero: 5	Responsable: Bengie Serrano, Jose Carlos Huallpamaita
Nombre de Tarea: Diseño de la arquitectura de documentos	
Descripción: El responsable de la tarea deberá diseñar la arquitectura de los documentos y colecciones de la base de datos bajo un concepto no relacional.	

Fuente: Elaboración Propia

Tabla 3.20 Tarea – Implementación de Documentos y Colecciones

Numero: 6	Responsable: Bengie Serrano
Nombre de Tarea de Equipo: Implementación de documentos y colecciones	
Descripción: El responsable de la tarea deberá implementar los scripts para la creación de los documentos y colecciones según la arquitectura diseñada en la tarea anterior.	

Fuente: Elaboración Propia

Tabla 3.21 Tarea – Despliegue de Base de Datos

Numero: 7	Responsable: Bengie Serrano
Nombre de Tarea: Despliegue de Base de Datos	
Descripción: El responsable de la tarea deberá realizar la instalación y configuración de la base de datos en un contenedor Docker.	

Fuente: Elaboración Propia

3.2.1.4 API/GraphQL

3.2.1.4.1 Elaboración de tareas

En esta etapa se definirá las tareas a realizar para poder implementar la API de consultas con GraphQL, donde cada punto de entrada es un nodo que puede conectar con otros nodos para recuperar información (*Query*) o puntos de entrada que modifiquen la base de datos (*Mutation*).

Tabla 3.22 Tarea– Despliegue del Contenedor API/GraphQL

Numero: 8	Responsable: Bengie Serrano
Nombre de Tarea: Despliegue del Contenedor API/GraphQL	
Descripción: El responsable de la tarea deberá realizar el despliegue del contenedor con la API/GraphQL.	

Fuente: Elaboración Propia

Tabla 3.23 Tarea – Consulta para Recuperar Datos de Usuarios

Numero: 9	Responsable: Bengie Serrano
Nombre de Tarea: Consulta para Recuperar Datos de Usuarios	
Descripción: El responsable de la tarea deberá implementar una consulta para la recuperación de los datos de los usuarios almacenados en la base de datos	

Fuente: Elaboración Propia

Tabla 3.24 Tarea– Consulta para Recuperar Datos de Módulos

Numero: 10	Responsable: Bengie Serrano
Nombre de Tarea: Consulta para Recuperar Datos de Módulos	
Descripción: El responsable de la tarea deberá implementar una consulta para la recuperación de los datos de los módulos de la plataforma, almacenados en la base de datos	

Fuente: Elaboración Propia

Tabla 3.25 Tarea – Consulta para Recuperar Datos de las Vistas

Numero: 11	Responsable: Bengie Serrano
Nombre de Tarea: Consulta para Recuperar Datos de las Vistas	
Descripción: El responsable de la tarea deberá implementar una consulta para la recuperación de datos de las vistas, almacenados en la base de datos	

Fuente: Elaboración Propia

Tabla 3.26 Tarea– Consulta para Recuperar Roles

Numero: 12	Responsable: Bengie Serrano
Nombre de Tarea: Consulta para Recuperar Roles	
Descripción: El responsable de la tarea deberá implementar una consulta para la recuperación de los datos de los roles almacenados en la base de datos	

Fuente: Elaboración Propia

Tabla 3.27 Tarea – Consulta para Recuperar Datos de Sensores

Numero: 13	Responsable: Bengie Serrano
Nombre de Tarea: Consulta para Recuperar Datos de Sensores	
Descripción: El responsable de la tarea deberá implementar una consulta para la recuperación de los datos de los sensores almacenados en la base de datos	

Fuente: Elaboración Propia

Tabla 3.28 Tarea – Mutaciones para Modificar Datos de Usuarios

Numero: 14	Responsable: Bengie Serrano
Nombre de Tarea: Mutaciones para Modificar Datos de Usuarios	
Descripción: El responsable de la tarea deberá implementar una mutación para la modificación de los datos de los usuarios almacenados en la base de datos	

Fuente: Elaboración Propia

Tabla 3.29 Tarea – Mutaciones para Modificar Datos de Módulos

Numero: 15	Responsable: Bengie Serrano
Nombre de Tarea: Mutaciones para Modificar Datos de Módulos	
Descripción: El responsable de la tarea deberá implementar una mutación para la modificación de los datos de los módulos de la plataforma, almacenados en la base de datos	

Fuente: Elaboración Propia

Tabla 3.30 Tarea – Mutaciones para Modificar Datos de las Vistas

Numero: 16	Responsable: Bengie Serrano
Nombre de Tarea: Mutaciones para Modificar Datos de las Vistas	
Descripción: El responsable de la tarea deberá implementar una mutación para la modificación de datos de las vistas almacenadas en la base de datos	

Fuente: Elaboración Propia

Tabla 3.31 Tarea– Mutaciones para Modificar Roles

Numero: 17	Responsable: Bengie Serrano
Nombre de Tarea: Mutaciones para Modificar Roles	
Descripción: El responsable de la tarea deberá implementar una mutación para la modificación de los datos de los roles almacenados en la base de datos	

Fuente: Elaboración Propia

Tabla 3.32 Tarea– Mutaciones para Modificar Datos de Sensores

Numero: 18	Responsable: Bengie Serrano
Nombre de Tarea: Mutaciones para Modificar Datos de Sensores	
Descripción: El responsable de la tarea deberá implementar una consulta para la recuperación de los datos de los sensores almacenados en la base de datos	

Fuente: Elaboración Propia

Tabla 3.33 Tarea – Consulta para Recuperar Datos de Intersecciones

Numero: 19	Responsable: José Carlos Huallpamaita
Nombre de Tarea: Consulta para Recuperar Datos de Intersecciones	
Descripción: El responsable de la tarea deberá implementar una consulta para la recuperación de los datos de las intersecciones almacenadas en la base de datos.	

Fuente: Elaboración Propia

Tabla 3.34 Tarea – Mutaciones para Modificar Datos de Intersecciones

Numero: 20	Responsable: José Carlos Huallpamaita
Nombre de Tarea de Equipo: Mutaciones para Modificar Datos de Intersecciones	
Descripción: El responsable de la tarea deberá implementar una mutación para la modificación de los datos de las intersecciones almacenadas en la base de datos.	

Fuente: Elaboración Propia

3.2.1.5 Aplicación web

En esta etapa se considera el desarrollo de las interfaces del cliente obtenidos a partir de las historias de usuario, es por ello que las tablas siguientes muestran una serie de tareas para cada vista.

3.2.1.5.1 Elaboración de tareas

La Tabla 3.35, muestra la tarea que consta la implementación del módulo de conexión entre el controlador semafórico y el cliente web desarrollado en angular mediante el protocolo MQTT.

Tabla 3.35 Tarea– Implementación del MQTT-Client

Numero: 21	Responsable: Bengie Serrano
Nombre de Tarea: Implementación del MQTT-Client	
Descripción: El responsable de la tarea deberá realizar la implementación del módulo en el cliente que conecta los datos enviados desde el controlador semafórico hacia el cliente web mediante el protocolo MQTT.	

Fuente: Elaboración Propia

Tabla 3.36 Tarea– Mantenimiento de Usuarios

Numero: 22	Responsable: Bengie Serrano
Nombre de Tarea: Mantenimiento de Usuarios	
Descripción: El responsable de la tarea deberá implementar una interfaz en el cual un usuario autorizado, podrá realizar operaciones de inserción, actualización y eliminación de los usuarios registrados a la plataforma.	

Fuente: Elaboración Propia

Tabla 3.37 Tarea– Mantenimiento de Módulos

Numero: 23	Responsable: Bengie Serrano
Nombre de Tarea: Mantenimiento de Módulos	
Descripción: El responsable de la tarea deberá implementar una interfaz en el cual un usuario autorizado, podrá realizar operaciones de inserción, actualización y eliminación de los módulos registrados a la plataforma.	

Fuente: Elaboración Propia

Tabla 3.38 Tarea– Mantenimiento de Vistas

Numero: 24	Responsable: Bengie Serrano
Nombre de Tarea: Mantenimiento de Vistas	
Descripción: El responsable de la tarea deberá implementar una interfaz en el cual un usuario autorizado, podrá realizar operaciones de inserción, actualización y eliminación de las vistas registradas a la plataforma.	

Fuente: Elaboración Propia

Tabla 3.39 Tarea– Mantenimiento de Roles

Numero: 25	Responsable: Bengie Serrano
Nombre de Tarea: Mantenimiento de Roles	
Descripción: El responsable de la tarea deberá implementar una interfaz en el cual un usuario autorizado, podrá realizar operaciones de inserción, actualización y eliminación de los roles registrados a la plataforma.	

Fuente: Elaboración Propia

Tabla 3.40 Tarea – Mantenimiento de Sensores

Numero: 26	Responsable: Bengie Serrano
Nombre de Tarea: Mantenimiento de Sensores	
Descripción: El responsable de la tarea deberá implementar una interfaz en el cual un usuario autorizado, podrá realizar operaciones de inserción, actualización y eliminación de los sensores registrados a la plataforma.	

Fuente: Elaboración Propia

Tabla 3.41 Tarea – Mantenimiento de Intersecciones

Numero: 27	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea: Mantenimiento de Intersecciones.	
Descripción: El responsable de la tarea deberá implementar una interfaz en el cual un usuario autorizado, podrá realizar operaciones de inserción, actualización y eliminación de las intersecciones registradas a la plataforma.	

Fuente: Elaboración Propia

Tabla 3.42 Tarea – Mapa de Geolocalización de Semáforos

Numero: 28	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea: Mapa de Geolocalización de Semáforos	
Descripción: El responsable de la tarea deberá implementar una interfaz en el cual un usuario autorizado, podrá visualizar desde un mapa asistente tipo "*wizard*", la geolocalización de las ubicaciones de todos los semáforos conectados a la plataforma.	

Fuente: Elaboración Propia

Tabla 3.43 Tarea – Asistente de Configuración de Intersecciones

Numero: 29	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea: Asistente de Configuración de Intersecciones	
Descripción: El responsable de la tarea deberá implementar una interfaz en el cual un usuario autorizado, podrá configurar mediante un asistente tipo "*wizard*" los datos, fases, horarios y geolocalización de las intersecciones donde están instalados los controladores conectados a la plataforma.	

Fuente: Elaboración Propia

3.2.1.6 Servidor MQTT

3.2.1.6.1 Elaboración de tareas

Tabla 3.44 Tarea – Implementación del Servidor de Red o Bróker

Numero: 30	Responsable: Bengie Serrano
Nombre de Tarea: Implementación del Servidor de Red o Bróker	
Descripción: El responsable de la tarea deberá realizar la implementación del servidor de red o *bróker.*	

Fuente: Elaboración Propia

Tabla 3.45 Tarea– Instalar contenedor del Bróker:

Numero: 31	Responsable: Bengie Serrano
Nombre de Tarea: Despliegue del Contenedor Docker	
Descripción: El responsable de la tarea deberá realizar la instalación y despliegue del servidor de red o *bróker* dentro de un contenedor Docker.	

Fuente: Elaboración Propia

3.2.2 Fase de implementación

3.2.2.1 Servidor de nombre de dominio

- **Tarea 1 – Instalar contenedor "BIND":**

  La información en esta sección explica la configuración del contenedor ***sameersbn/bind:latest***.

  Especificamos un nombre para el contenedor ***sigecs-dns***, dirección IP del servidor DNS default para el contenedor **192.168.71.254**, publicamos y exponemos el puerto 53 con el protocolo UDP, y el puerto 10000 con el protocolo TCP, ambos en la red interna (Docker) y en la red del host, para salvar las configuraciones montamos el directorio ***/data***

como ***/srv/docker/bind***, configuramos la variable de entorno ROOT_PASSWORD con la contraseña deseada.

```
$ docker run
    -d
    --name=dns-app
    --dns=192.168.71.254
    --publish=172.17.0.1:53:53/udp
    --publish=192.168.71.254:53:53/udp
    --publish=172.17.0.1:10000:10000
    --publish=192.168.71.254:10000:10000
    --volume=/srv/docker/bind:/data:z
    --env='ROOT_PASSWORD=password'
    sameersbn/bind:latest
```

Una vez lanzado el comando anterior podemos ver el contenedor en ejecución tal como resalta el rectángulo rojo en la lista contenedores mostrados en la figura 3.4.

Fig. 3.4 Servidor DNS en Ejecución

Fuente: Elaboración Propia

- **Tarea 2 – Configurar contenedor "BIND"**

Se accede mediante un navegador web al panel de configuración.

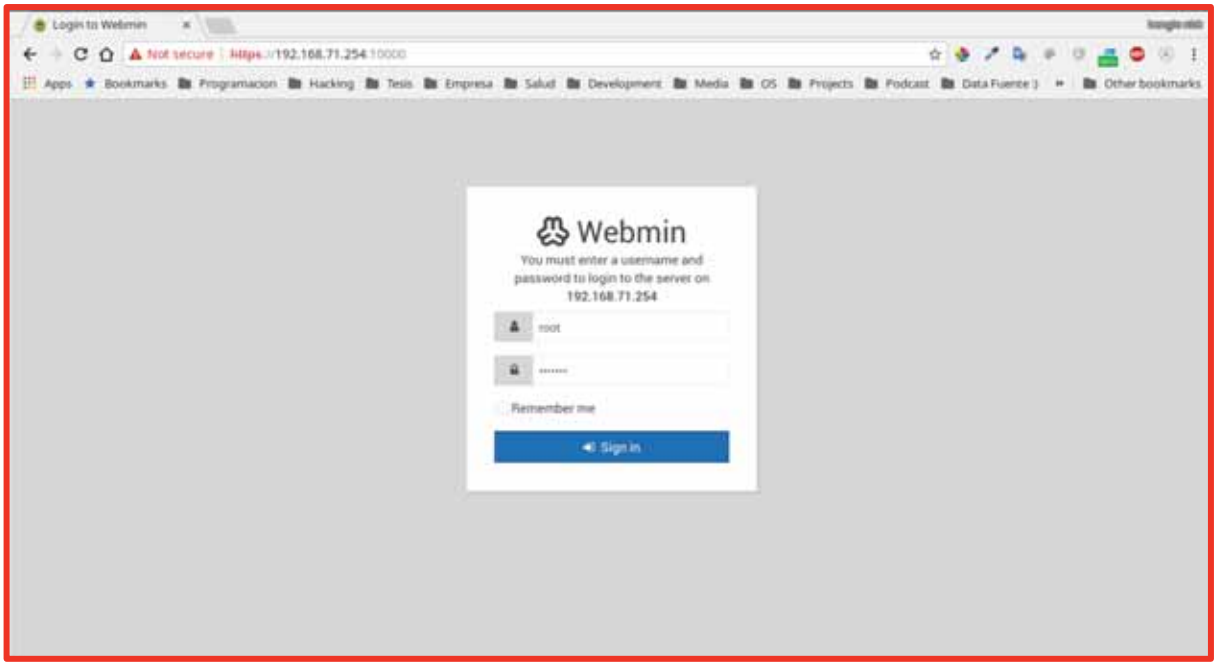


Fig. 3.5 Login Panel Configuración

Fuente: Elaboración Propia

Se configura el nombre de dominio "sigecs.io" y la dirección IP a la que apuntara tal como se resalta con el cuadro rojo mostrado en la figura 3.6.

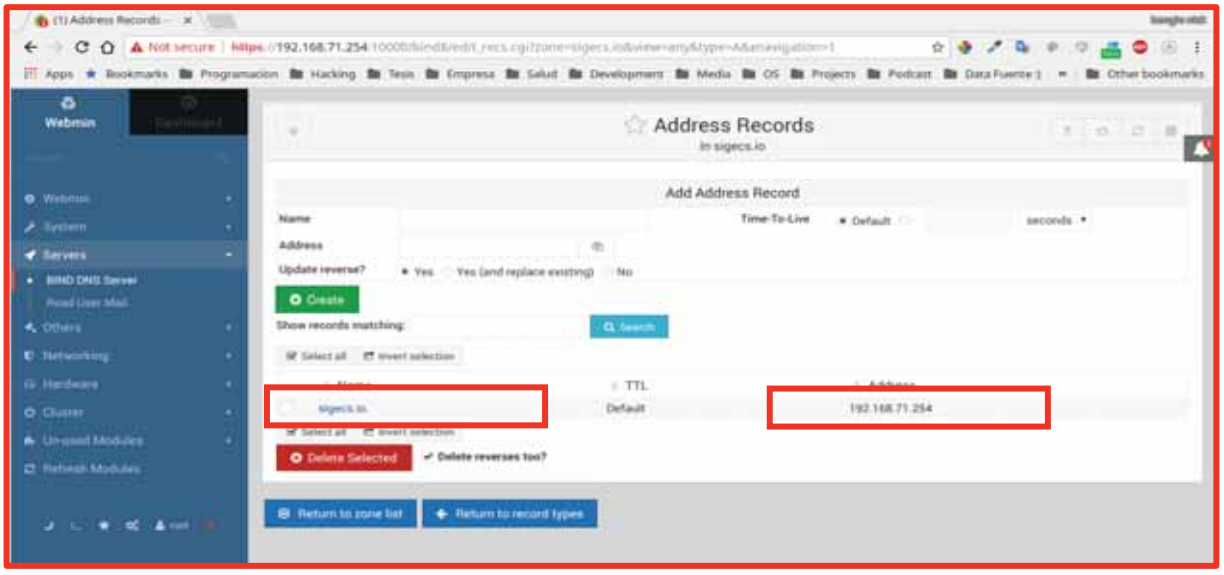


Fig. 3.6 Configuración del Dominio y dirección IP

Fuente: Elaboración Propia

3.2.2.2 Servidor NTP

- **Tarea 3 – Instalar contenedor "SNTP":**

  La información en esta sección explica la configuración del contenedor ***seznam/ntpdsistem***.

  Se especifica un nombre para el contenedor ***sigecs-ntp***, se agrega capacidades de sistema **SYS_TIME**.

```
$ docker run
    -d
    --name=sigecs-ntp
    -- cap-add='SYS_TIME'
    seznam/ntpdsistem
```

  Una vez lanzado el comando anterior se puede ver el contenedor en ejecución tal como resalta el rectángulo rojo en la lista contenedores mostrados en la figura 3.7.

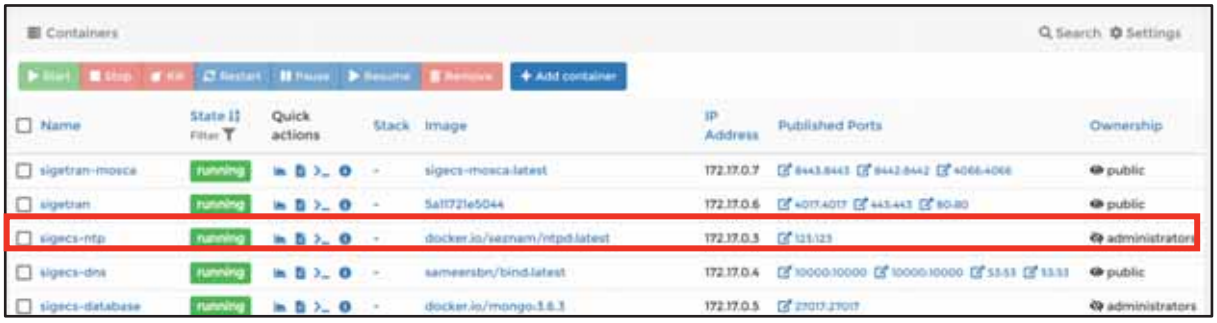


Fig. 3.7 Servidor NTP en Ejecución

Fuente: Elaboración Propia

- **Tarea 4 – Configurar contenedor "SNTP"**

Se accede mediante un navegador web al panel de configuración.

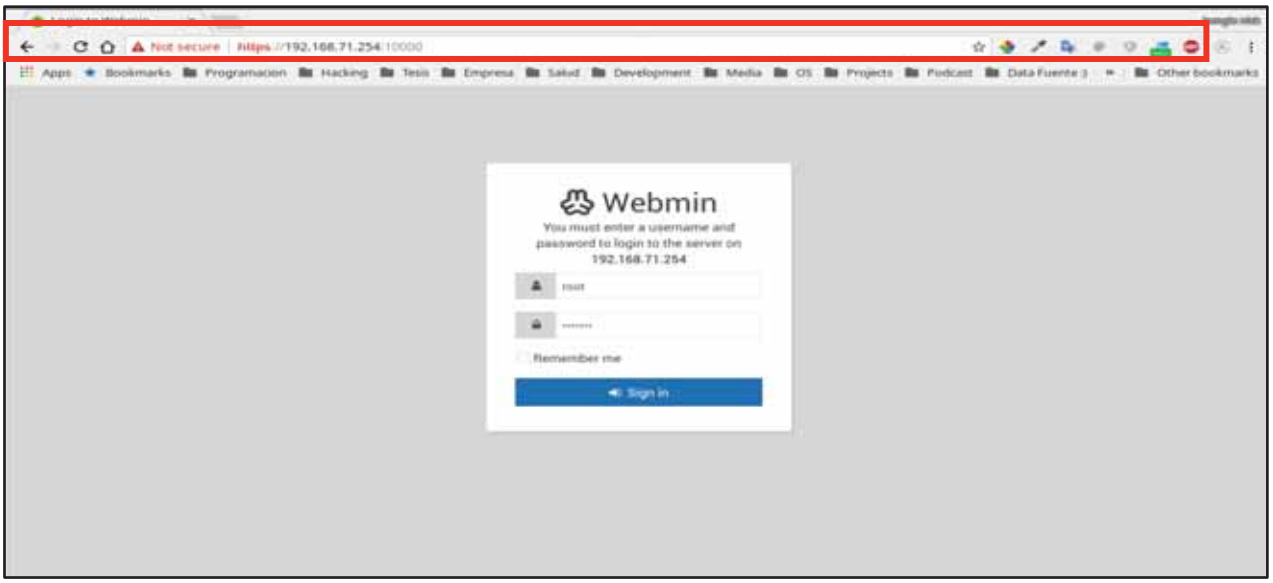


Fig. 3.8 Login Panel Configuración

Fuente: Elaboración Propia

Se configura el nombre de dominio sigecs.io y la dirección IP a la que apuntara tal como se resalta con el cuadro rojo mostrado en la figura 3.9.

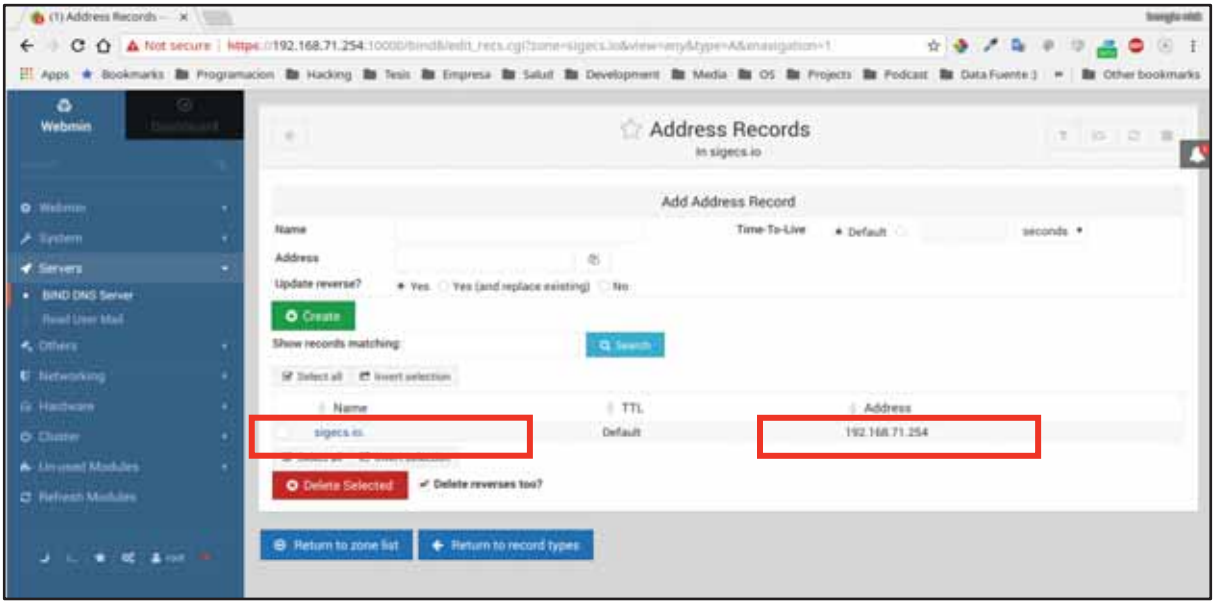


Fig. 3.9 Configuración de Dominio y Dirección IP

Fuente: Elaboración Propia

El diagrama mostrado a continuación es el objetivo de la implementación de un servidor NTP en esta plataforma.

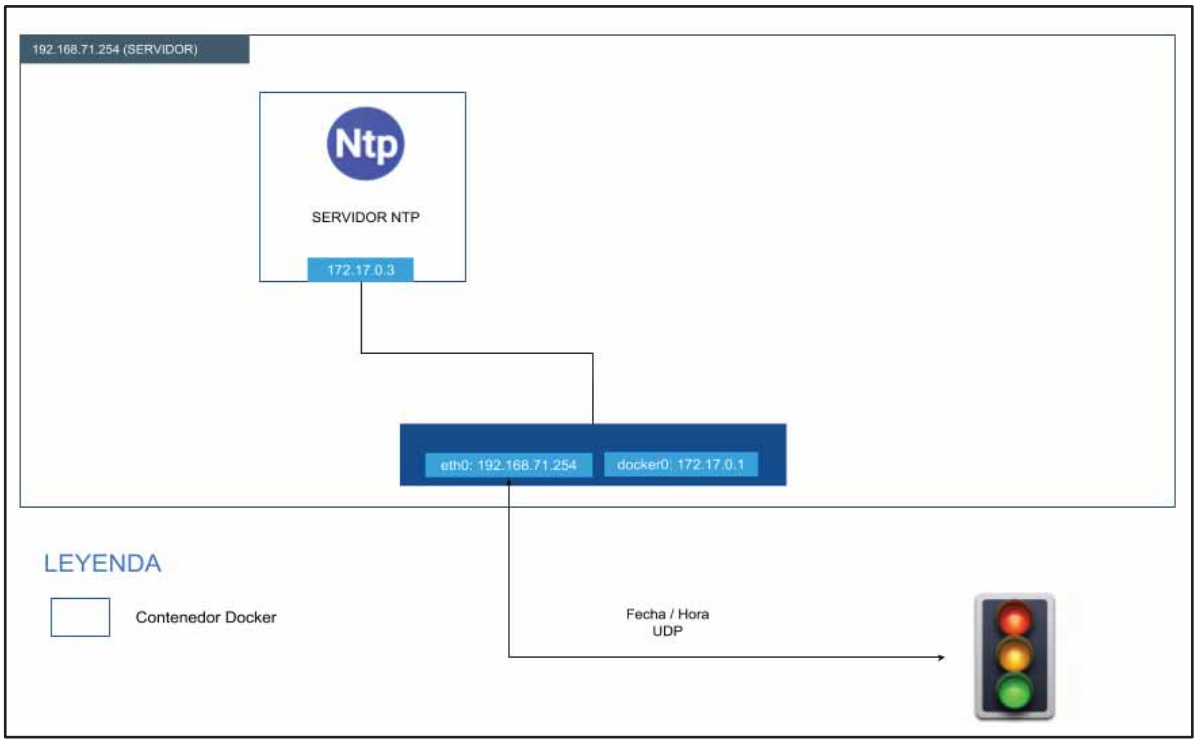


Fig. 3.10 Diseño servidor SNTP

Fuente: Elaboración Propia

3.2.2.3 Base de Datos "Mongo DB"

- **Tarea 5 – Diseño de la estructura de documentos:**

  El diagrama mostrado en el **Anexo C**, muestra la estructura de la base de datos diseñada para la toda la plataforma. La descripción de los campos de cada colección será ampliada en la implementación de las tareas de la API/GraphQL

- **Tarea 6 – Instalar contenedor de "MongoDB 3.6.3":**

  La información en esta sección explica la instalación del contenedor **docker.io/mongo:3.6.3**.

  Se especifica un nombre para el contenedor ***sigecs-db***, se publica y expone el puerto 27017 con el protocolo TCP por default, configurar el usuario root default de la base de

datos para ello la variable de entorno MONGO_INITDB_ROOT_USERNAME será *admin* y la variable de entorno MONGO_INITDB_ROOT_PASSWORD será *password*.

```
$ docker pull mongo:3.6.3
$ docker run
    -d
    --name=sigecs-db
    --publish=27017: 27017
    --env='MONGO_INITDB_ROOT_USERNAME=admin'
    --env='MONGO_INITDB_ROOT_PASSWORD=password'
    docker.io/mongo:3.6.3
```

- **Tarea 7 – Configurar base de datos:**

  La información en esta sección explica la configuración de la base de datos.

  Se conecta a la base de datos usando un acceso por terminal:

  ```
  $ mongo mongodb://admin:password@192.168.71.254:27017/admin
  ```

  Una vez accedido a la base de datos se crea un usuario para el sistema, por cuestiones prácticas usaremos el usuario admin.

```
db.createUser({
user: "admin ",
pwd: "password ",
roles: [{role: "dbOwner ", db: "db_sigecs "}]
})
```

3.2.2.4 API/GraphQL

- **Tarea 8 – Despliegue del Contenedor API/GraphQL:**

  Se toma como base la distribución Ubuntu en su versión 18.

```
FROM ubuntu:18.04
```

  Se actualiza la imagen base.

```
# Install required packages
RUN apt-get update -y
```

  Se copia los certificados CA y se actualiza la imagen.

```
ADD tls/rootCA.crt /usr/local/share/ca-certificates/rootCA.crt
RUN apt-get install -y ca-certificates
RUN update-ca-certificates -y
```

Se crea los directorios para luego copiar los archivos necesarios, con la siguiente estructura.

/opt

/opt/server

/opt/server/build

/opt/server/media

/opt/client

/opt/client/dist

Se le da permiso de ejecución al binario "/opt/server/build/main".

```
RUN mkdir /opt/server
RUN mkdir /opt/client
COPY build /opt/server/build
RUN ["chmod", "+x", "/opt/server/build/main"]
COPY media /opt/server/media
COPY dist /opt/client/dist
```

Se instala y configura nginx, que funcionara como un proxy para resolver las peticiones a la API y servir los archivos estáticos a través de https.

```
RUN apt-get install nginx -y
ADD docker/nginx/nginx.conf /etc/nginx/nginx.conf
ADD docker/nginx/app.conf /etc/nginx/conf.d/default.conf
RUN ln -sf /dev/stdout /var/log/nginx/semfull.error.log
```

Se expone los puertos públicos. (80 y 443 http y https, 4017 acceso a la API).

```
EXPOSE 80
EXPOSE 443
EXPOSE 4017
```

Se configura las variables de entorno, que serán usadas desde el binario del API, para acceder correctamente a la base de datos y servir los servicios (mutaciones y consultas).

```
ENV DB_USER="admin"
ENV DB_PASSWORD="password"
ENV DB_HOST="172.17.0.1"
ENV DB_PORT="27017"
ENV DB_NAME="db_sigecs"
ENV SERVER_PORT="4017"
```

Se copia el script principal y le concedemos permiso de ejecución.

```
ADD docker/run.sh /run.sh
RUN ["chmod", "+x", "/run.sh"]
```

```
#!/bin/bash

echo "RUN NGINX"
exec nginx &

echo "RUN APP"
exec /opt/server/build/main
```

Este script ejecutará el demonio de nginx y ejecutará el binario del API/GraphQL.

Se establece el directorio de trabajo.

```
WORKDIR /opt/server/
```

Se ejecuta el script principal.

```
CMD ["/run.sh"]
```

Para crear el contenedor se ingresa el siguiente comando:

```
$ docker build -t 'sigecs'
```

Para correr el contenedor creado se corre los comandos:

```
$ docker run
    --name=sigetran
    --env='DB_USER=admin'
    --env='DB_PASSWORD=password'
    --env='DB_HOST=172.17.0.1'
    --env='DB_PORT=27017'
    --env='DB_NAME=db_sigecs'
    --env='SERVER_PORT=4017
 Sigecs
```

Como resultado se tiene el contenedor configurado y corriendo, como se muestra en la siguiente imagen

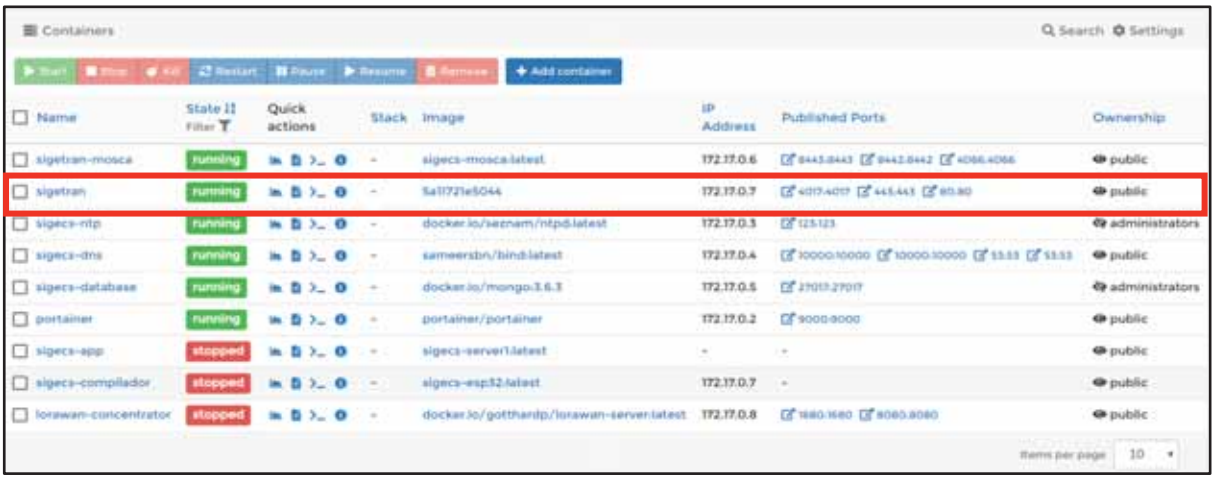


Fig. 3.11 Contenedor de API/GraphQL en Ejecución

Fuente: Elaboración Propia

- **Tarea 9 – Consulta para Recuperar Datos de Usuarios:**

En la figura 3.12 se muestra la consulta para obtener la información de los usuarios con los siguientes campos:

id: Identificación único del usuario.
dni: Numero de documento de identidad del usuario.
email: Dirección de correo electrónico.
names: Nombres del usuario
ln: Apellido paterno
mln: Apellido Materno
nick: Nombre de usuario

```
query {
  users {
    id
    dni
    email
    names
    ln
    mln
    nick
  }
}
```

Fig. 3.12 Query para Recuperar Datos de Usuario

Fuente: Elaboración Propia

En la figura 3.13, se muestra el objeto JSON con los datos solicitados en la consulta de la figura 3.12.

```
{
  "data": {
    "users": [
      {
        "dni": "00000000",
        "email": "sigecs@gmail.com",
        "id": "5a3ed9f12c1e492f45cd2848",
        "ln": "Super",
        "mln": "Super",
        "names": "Super",
        "nick": "super"
      },
      {
        "dni": "00000001",
        "email": "experto@municusco.gob.pe",
        "id": "5b082be8afe59200130e6635",
        "ln": "mpc",
        "mln": "mpc",
        "names": "mpc",
        "nick": "experto"
      },
      {
        "dni": "00000002",
        "email": "operador@municusco.gob.pe",
        "id": "5b0ee78257ed6f1316697c3",
        "ln": "operador",
        "mln": "operador",
        "names": "operador",
        "nick": "operador"
      },
      {
        "dni": "00000003",
        "email": "visor@municusco.gob.pe",
        "id": "5b0ee7ad57ed6f13166997c4",
        "ln": "visor",
        "mln": "visor",
        "names": "visor",
        "nick": "visor"
      }
    ]
  }
}
```


Fig. 3.13 Resultado de la Consulta para Recuperar Datos de Usuario

Fuente: Elaboración Propia

- **Tarea 10 – Consulta para Recuperar Datos de Módulos:**

En la figura 3.14 se muestra la consulta para obtener la información de los módulos con los siguientes campos:

id: Identificador único del módulo.

color: color de fondo del botón para ingresar al módulo.

icono: icono representativo del módulo.

name: nombre del módulo.

```
query {
  modules {
    id
    color
    icon
    name
  }
}
```

Fig. 3.14 Consulta para Recuperar Datos de Módulos

Fuente: Elaboración Propia

En la figura 3.15, se muestra el objeto JSON con los datos solicitados en la consulta de la figura 3.14.

```
{
  "data": {
    "modules": [
      {
        "color": "cat__core__step--danger",
        "icon": "icmn-home",
        "id": "5a3ed8f22c1e492f45cd2842",
        "name": "Admin"
      },
      {
        "color": "cat__core__step--success",
        "icon": "icmn-home",
        "id": "5a61ef0fa310721d7c210ab2",
        "name": "SIGECS"
      },
      {
        "color": "cat__core__step--success",
        "icon": "icmn-home",
        "id": "5b7b15bc797cde1ee27f214f",
        "name": "SIGECAM"
      }
    ]
  }
}
```


Fig. 3.15 Resultado de la Consulta para recuperar Datos de Módulos

Fuente: Elaboración Propia

- **Tarea 11 – Consulta para Recuperar Datos de Vistas:**

  En la figura 3.16 se muestra la consulta para obtener la información de las vistas con los siguientes campos:

  id: Identificador único del módulo.

  name: Nombre del módulo.

  views: Colección de vistas para cada módulo con los campos:

  id: Identificador único de la vista.

  name: Nombre de la vista.

  icon: Icono de la vista.

  url: Ruta de acceso para la vista

  description: Descripción de la vista

```
query {
  modules {
    id
    name
    views {
      id
      name
      icon
      url
      description
    }
  }
}
```

Fig. 3.16 Consulta para Recuperar Datos de Vistas

Fuente: Elaboración Propia

En la figura 3.17, se muestra el objeto JSON con los datos solicitados en la consulta de la figura 3.16.

```
{
  "data": {
    "modules": [
      {
        "id": "5a3ed8f22c1e492f45cd2842",
        "name": "Admin",
        "views": [
          {
            "description": "Administracion de Modulos",
            "icon": "icmn-list",
            "id": "5b0f24391ede1d1ac3dea63a",
            "name": "A. Modulos",
            "url": "/admin-modules"
          },
          {
            "description": "Administracion de usuarios",
            "icon": "icmn-user",
            "id": "5b0f378a1ede1d1ac3dea63c",
            "name": "Super A. Usuarios ",
            "url": "/admin-users"
          },
          {
            "description": "Administracion de Usuarios",
            "icon": "icmn-user",
            "id": "5b0f41101ede1d1ac3dea640",
            "name": "A. Usuarios",
            "url": "/admin-users2"
          }
        ]
      },
```


Fig. 3.17 Resultado de la Consulta para recuperar Datos de Vistas

Fuente: Elaboración Propia

- **Tarea 12 – Consulta para Recuperar Datos de Roles:**

En la figura 3.18 se muestra la consulta para obtener la información de los roles con los siguientes campos:

id: Identificador único del módulo.

name: Nombre del módulos.

roles: Colección de roles con los campos:

id: Identificador único del rol.

name: nombre del rol.

description: Descripción breve sobre el rol

```
query {
  modules {
    id
    name
    roles {
      id
      name
      description
    }
  }
}
```


Fig. 3.18 Query para recuperar Datos de Roles

Fuente: Elaboración Propia

En la figura 3.19, se muestra el objeto JSON con los datos solicitados en la consulta de la figura 3.18.


```
{
  "data": {
    "modules": [
      {
        "id": "5a3ed8f22c1e492f45cd2842",
        "name": "Admin",
        "roles": [
          {
            "description": "Super Administrator",
            "id": "5a3ed8f22c1e492f45cd2843",
            "name": "Super"
          },
          {
            "description": "Administracion de usuarios",
            "id": "5b0f3d331ede1d1ac3dea63e",
            "name": "Admin Usuarios"
          }
        ]
      },
      {
        "id": "5a61ef0fa310721d7c210ab2",
        "name": "SIGECS",
        "roles": [
          {
            "description": "Role super Admin",
            "id": "5a61ef0fa310721d7c210ab3",
            "name": "Admin"
          },
          {
            "description": "Operador Experto",
            "id": "5b0440616c5f3000134e982c",
            "name": "Operador Experto"
          },
          {
            "description": "Operador",
            "id": "5b0ee3e557ed6f13166997ba",
            "name": "Operador"
          },
          {
            "description": "Visor",
            "id": "5b0ee3ef57ed6f13166997bb",
            "name": "Visor"
          }
        ]
      },
      {
        "id": "5b7b15bc797cde1ee27f214f",
        "name": "SIGECAM",
        "roles": [
          {
            "description": "Operador Experto",
            "id": "5b7b16b7797cde1ee27f2151",
            "name": "Operador Experto"
          }
        ]
      }
    ]
  }
}
```


Fig. 3.19 Resultado de la Consulta para recuperar Datos de Roles

Fuente: Elaboración Propia

- **Tarea 13 – Consulta para Recuperar Datos de Sensores:**

  En la figura 3.20, se muestra la consulta para obtener la información de los sensores con los siguientes campos:

id: Identificador único del sensor.

name: Nombre del sensor.

code: Código del sensor.

abdrev: Abreviatura representativa del sensor.

type: Tipo de sensor.

```
query {
  sensors {
    id
    name
    code
    abbrev
    type
  }
}
```

Fig. 3.20 Consulta para recuperar Datos de Sensores

Fuente: Elaboración Propia

En la figura 3.21, se muestra el objeto JSON con los datos solicitados en la consulta de la figura 3.20.


```
{
  "data": {
    "sensors": [
      {
        "abbrev": "V",
        "code": "LL001",
        "id": "5a406f14a4cee6281eec484c",
        "name": "Lampara Led Vehicular",
        "type": "led"
      },
      {
        "abbrev": "P",
        "code": "LL002",
        "id": "5a4071cda4cee6281eec4850",
        "name": "Lampara Led Peatonal",
        "type": "led"
      },
      {
        "abbrev": "G",
        "code": "LL003",
        "id": "5a4071e0a4cee6281eec4851",
        "name": "Lampara Led Giro",
        "type": "led"
      },
      {
        "abbrev": "D",
        "code": "TFT 1.8 SPI",
        "id": "5a83e9f2c3a972336cf93685",
        "name": "TFT 1.8 SPI",
        "type": "display"
      },
      {
        "abbrev": "",
        "code": "esp32",
        "id": "5b10531b2258802c52b9914d",
        "name": "ESP 32",
        "type": "chip"
      },
      {
        "abbrev": "",
        "code": "DS3231",
        "id": "5b105a772258802c52b9914f",
        "name": "RTC DS3231",
        "type": "rtc"
      },
      {
        "abbrev": "",
        "code": "MCP23017",
        "id": "5b1200150dfe873d0d43cb5f",
        "name": "MCP23017",
        "type": "i2c"
      }
    ]
  }
}
```


Fig. 3.21 Resultado de la Consulta para recuperar Datos de Sensores

Fuente: Elaboración Propia

- **Tarea 14 – Mutaciones para Modificar Datos de Usuarios:**

En la figura 3.22 se muestra la consulta para agregar la información de un usuario, con los siguientes campos como parámetros de entrada:

names: Nombres de usuario.
ln: Apellido Paterno.
mln: Apellido Materno.
email: Dirección de correo electrónico.
Nick: Nombre de usuario.
password: Contraseña para el usuario.
Dni: Numero de documento de identidad del usuario

Fig. 3.22 Mutación para Agregar Nuevo Usuario

Fuente: Elaboración Propia

En la figura 3.23 se muestra la consulta para actualizar la información de un usuario, con los siguientes campos como parámetros de entrada:

id: Identificador único del usuario a actualizar
names: Nombres de usuario.
ln: Apellido Paterno.
mln: Apellido Materno.
email: Dirección de correo electrónico.
Nick: Nombre de usuario.
password: Contraseña para el usuario.
Dni: Numero de documento de identidad del usuario

Fig. 3.23 Mutación para Actualizar Usuario

Fuente: Elaboración Propia

En la figura 3.24 se muestra la consulta para eliminar la información de un usuario, con los siguientes campos como parámetros de entrada:

id: identificador único del usuario a eliminar.

Fig. 3.24 Mutación para Eliminar un Usuario

Fuente: Elaboración Propia

- **Tarea 15 – Mutaciones para Modificar Datos de Módulos:**

En la figura 3.25 se muestra la consulta para agregar la información de un módulo, con los siguientes campos como parámetros de entrada:

name: Nombre del módulo.
icono: Icono representativo del módulo.
color: Color de fondo del botón para ingresar al módulo.

Fig. 3.25 Mutación para Agregar Nuevo Modulo

Fuente: Elaboración Propia

En la figura 3.26 se muestra la consulta para actualizar la información de un módulo, con los siguientes campos como parámetros de entrada:

id: identificador único del módulo a actualizar.
name: Nombre del módulo.
icono: Icono representativo del módulo.
color: Color de fondo del botón para ingresar al módulo.

Fig. 3.26 Mutación para Actualizar Módulos

Fuente: Elaboración Propia

En la figura 3.27 se muestra la consulta para eliminar la información de un módulo, con los siguientes campos como parámetros de entrada:

id: identificador único del módulo a eliminar.

Fig. 3.27 Mutación para Eliminar un Modulo

Fuente: Elaboración Propia

- **Tarea 16 – Mutaciones para Modificar Datos de Vistas:**

En la figura 3.28 se muestra la consulta para agregar la información de una vista, con los siguientes campos como parámetros de entrada:

idModule: Identificador único del módulo al que se le agregara la vista.

name: Nombre de la vista.

description: Descripción breve de la vista.

icon: icono representativo de la vista.

url: dirección de enlace para la vista.

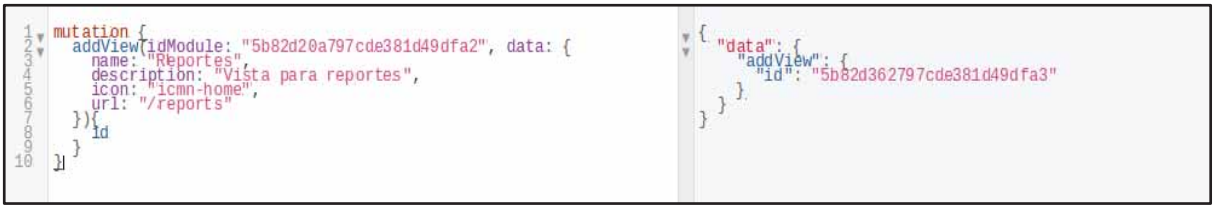


Fig. 3.28 Mutación para Agregar Nueva Vista

Fuente: Elaboración Propia

En la figura 3.29, se muestra la consulta para agregar la información de una vista, con los siguientes campos como parámetros de entrada:

id: Identificador único de la vista al que se desea actualizar.

name: Nombre de la vista.

description: Descripción breve de la vista.

icon: icono representativo de la vista.

url: dirección de enlace para la vista.

Fig. 3.29 Mutación para Actualizar una Vista

Fuente: Elaboración Propia

En la figura 3.30, se muestra la consulta para eliminar la información de una vista, con los siguientes campos como parámetros de entrada:

id: identificador único de la vista a eliminar.

Fig. 3.30 Mutación para Eliminar una Vista

Fuente: Elaboración Propia

- **Tarea 17 – Mutaciones para Modificar Datos de Roles:**

En la figura 3.31, se muestra la consulta para agregar la información de un rol, con los siguientes campos como parámetros de entrada:

idModule: Identificador único del módulo al que se le agregara el rol.
name: nombre del rol.
description: Descripción del rol.

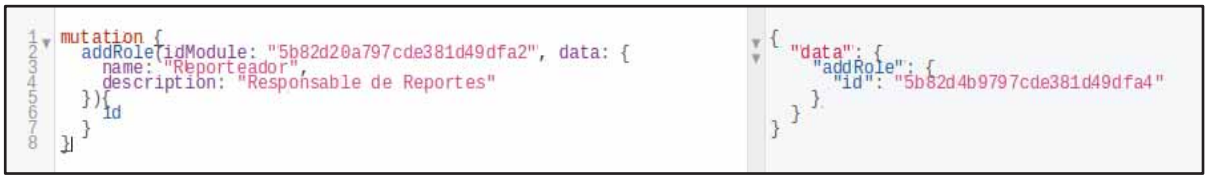


Fig. 3.31 Mutación para Agregar un Rol a un Modulo

Fuente: Elaboración Propia

En la figura 3.32 se muestra la consulta para actualizar la información de un rol, con los siguientes campos como parámetros de entrada:

id: Identificador único del rol a actualizar.
name: nombre del rol.
description: Descripción del rol.

Fig. 3.32 Mutación para Modificar un Rol

Fuente: Elaboración Propia

En la figura 3.33, se muestra la consulta para eliminar la información de un rol, con los siguientes campos como parámetros de entrada:

id: identificador único del rol a eliminar.

Fig. 3.33 Mutación para Eliminar Rol

Fuente: Elaboración Propia

- **Tarea 18 – Mutaciones para Modificar Datos de Sensores:**

En la figura 3.34, se muestra la consulta para agregar la información de un sensor, con los siguientes campos como parámetros de entrada:

name: Nombre del Sensor.

code: Codigo del Sensor.

type: tipo de Sensor

Fig. 3.34 Mutación para Agregar Nuevo Sensor

Fuente: Elaboración Propia

- **Tarea 19 – Consulta para Recuperar Datos de Intersecciones:**

En la figura 3.35, se muestra la consulta para obtener la información de las intersecciones con los siguientes campos:

id: Identificador único de la intersección.
code: Código semafórico de la intersección.
name: nombre de la intersección.
lat: latitud de geolocalización.
lngt: Longitud de geolocalización.
day1: Tipo de configuración para el día 1.
day2: Tipo de configuración para el día 2.
day3: Tipo de configuración para el día 3.
day4: Tipo de configuración para el día 4.
day5: Tipo de configuración para el día 5.
day6: Tipo de configuración para el día 6
day7: Tipo de configuración para el día
ethernetSetting: Configuración para la conexión por la red con los campos:
address: dirección IP.
dns: Dirección de DNS.
gateway: Puerta de enlace.
host: dirección del host.
netmask: mascara de red.
port: puerto por el que se realiza la conexión

```
query {
  intersections {
    id
    code
    name
    lat
    lngt
    day1
    day2
    day3
    day4
    day5
    day6
    day7
    ethSetting {
      address
      dns
      gateway
      host
      netmask
      port
    }
    installed
  }
}
```

Fig. 3.35Consulta para recuperar Datos de Intersecciones

Fuente: Elaboración Propia

En la figura 3.36, se muestra el objeto JSON con los datos solicitados en la consulta de la figura 3.35.


```
{
  "data": {
    "intersections": [
      {
        "code": "SEM001",
        "day1": "type1",
        "day2": "type1",
        "day3": "type1",
        "day4": "type1",
        "day5": "type1",
        "day6": "type1",
        "day7": "type1",
        "ethSetting": {
          "address": "192.168.71.141",
          "dns": "192.168.71.254",
          "gateway": "192.168.71.1",
          "host": "sigecs.io",
          "netmask": "255.255.255.0",
          "port": 8442
        },
        "id": "5b68f07a797cde3e803.7dd6b",
        "installed": true,
        "lat": -13.525868858085462,
        "lng": -71.98757909242914,
        "name": "Almudena"
      },
      {
        "code": "SEM002",
        "day1": "type1",
        "day2": "type1",
        "day3": "type1",
        "day4": "type1",
        "day5": "type1",
        "day6": "type1",
        "day7": "type1",
        "ethSetting": {
          "address": "192.168.71.142",
          "dns": "192.168.71.254",
          "gateway": "192.168.71.1",
          "host": "sigecs.io",
          "netmask": "255.255.255.0",
          "port": 8442
        },
        "id": "5b68f406797cde3e803.7dd6c",
        "installed": true,
        "lat": -13.526986027560271,
        "lng": -71.9842658936974,
        "name": "Patacalle - Antonio Lorena"
      },
```


Fig. 3.36 Resultado de la Consulta para recuperar Datos de Intersecciones

Fuente: Elaboración Propia

En la figura 3.37, se muestra la consulta para obtener la información complementaria de las fases que posee las intersecciones con los siguientes campos:

id: Identificador único de la intersección.

code: Código semafórico de la intersección.

name: Nombre de la intersección.

phase: Fase vehicular con los campos:

- name: Nombre de la fase.
- enable: Estado de activado o desactivado.
- lat: Latitud de geolocalización de la fase.
- lng: Longitud de geolocalización de la fase.
- deg: Angulo giro

```
query {
  intersections {
    id
    code
    name
    phaseV1{
      name
      enable
      lat1
      lng1
      deg1
      lat2
      lng2
      deg2
      lat3
      lng3
      deg3
    }
    phaseV2{
      name
      enable
      lat1
      lng1
      deg1
      lat2
      lng2
      deg2
      lat3
      lng3
      deg3
    }
    phaseV3{
      name
      enable
      lat1
      lng1
      deg1
      lat2
      lng2
      deg2
      lat3
      lng3
      deg3
    }
  }
}
```


```
{
  "data": {
    "intersections": [
      {
        "code": "SEM001",
        "id": "5b68f07a797cde3e8037dd6b",
        "name": "Almudena",
        "phaseV1": {
          "deg1": 20,
          "deg2": 200,
          "deg3": 0,
          "enable": true,
          "lat1": -13.525833375330627,
          "lat2": -13.5259076980974,
          "lat3": 0,
          "lng1": -71.98758646822942,
          "lng2": -71.98755428410024,
          "lng3": 0,
          "name": ""
        },
        "phaseV2": {
          "deg1": 290,
          "deg2": 110,
          "deg3": 0,
          "enable": true,
          "lat1": -13.525792954167041,
          "lat2": -13.525890095338939,
          "lat3": 0,
          "lng1": -71.98754891817246,
          "lng2": -71.98756836478258,
          "lng3": 0,
          "name": ""
        },
        "phaseV3": {
          "deg1": 0,
          "deg2": 0,
          "deg3": 0,
          "enable": true,
          "lat1": 0,
          "lat2": 0,
          "lat3": 0,
          "lng1": 0,
          "lng2": 0,
          "lng3": 0,
          "name": ""
        }
      },
```


Fig. 3.37 Consulta para recuperar Datos de las Fases Vehiculares en las Intersecciones

Fuente: Elaboración Propia

En la figura 3.38, se muestra la consulta para obtener la información complementaria de los ciclos que posee las intersecciones para cada tipo, con los siguientes campos:

id: Identificador único de la intersección.

code: Código semafórico de la intersección.

name: Nombre de la intersección.

type: Tipo de configuración con los campos:

cycles: Número de ciclos.

start: hora de inicio.

end: Hora de finalización.

phaseV1: Fase vehicular 1.

phaseV2: Fase vehicular 2.

phaseV3: Fase vehicular 3.

phaseP1: Fase peatonal 1.

phaseP2: Fase peatonal 2.

phaseG1: Fase de giro 1.

phaseG2: Fase de giro 2.

Fig. 3.38 Consulta para recuperar Datos de los Tipos en las Intersecciones

Fuente: Elaboración Propia

- **Tarea 20 – Mutaciones para Modificar Datos de Intersecciones:**

  En la figura 3.39 se muestra la consulta para agregar la información de una nueva intersección para configurarlas posteriormente, con los siguientes campos como parámetros de entrada:

code: Código semafórico de la intersección.

name: nombre de la intersección.

lat: latitud de geolocalización.

lngt: Longitud de geolocalización.

ethernetSetting: Configuración para la conexión por la red con los campos:

address: dirección IP.

dns: Dirección de DNS.

gateway: Puerta de enlace.

host: dirección del host.

netmask: mascara de red.

port: puerto por el que se realiza la conexión.

switchD: dirección de switch nodo D

switchE: dirección de switch nodo E

```
mutation {
  addIntersection(data: {
    name:"Santiago - Siete Mascarones",
    code:"SEM500",
    lat:-13.525373747178868,
    lng:-71.96341911659583,
    ethSetting:{
      address:"192.168.71.11",
      gateway:"192.168.71.1",
      netmask:"255.255.255.0",
      dns:"192.168.71.254",
      host:"sigecs.io",
      port:8442,
      switchD: "192.168.7.10",
      switchE: "192.168.7.10"
    },
    day1:"type1",
    day2:"type1",
    day3:"type1",
    day4:"type1",
    day5:"type1",
    day6:"type1",
    day7:"type1",
    type_1_1:{
      start:"00:00:00",
      end:"23:59:59",
      cycles:2,
      phaseV1:"1A1D",
      phaseV2:"1A1D",
      phaseV3:"2D",
      phaseP1:"2D",
      phaseP2:"2D",
      phaseG1:"2D",
      phaseG2:"2D",
      phaseG3:"2D"
    },
    type_1_2:{
      start:"",
      end:"",
      cycles:0,
      phaseV1:"",
      phaseV2:"",
      phaseV3:"",
      phaseP1:"",
      phaseP2:"",
      phaseG1:"",
      phaseG2:"",
      phaseG3:""
    },
    phaseV1:{
      name:"",
      enable:true,
      lat1:-13.525331370070575,
      lng1:-71.96336075686443,
      deg1:20,
      lat2:0,
      lng2:0,
      deg2:0,
      lat3:0,
      lng3:0,
      deg3:0
    },
    phaseV2:{
      name:"",
      enable:true,
      lat1:0,
      lng1:0,
      deg1:0,
      lat2:0,
      lng2:0,
      deg2:0,
      lat3:0,
      lng3:0,
      deg3:0
    },
    installed:true
  }){
    id
  }
}
```

```
{
  "data": {
    "addIntersection": {
      "id": "5b82e3de797cde4292a97335"
    }
  }
}
```

Fig. 3.39 Mutación para Agregar Nueva Intersección

Fuente: Elaboración Propia

En la figura 3.40, se muestra la consulta para agregar la información de una nueva intersección para configurarlas posteriormente, con los siguientes campos como parámetros de entrada:

id: Identificador único de la intersección.
code: Código semafórico de la intersección.
name: nombre de la intersección.
lat: latitud de geolocalización.
lngt: Longitud de geolocalización.
ethernetSetting: Configuración para la conexión por la red con los campos:
address: dirección IP.
dns: Dirección de DNS.
gateway: Puerta de enlace.
host: dirección del host.
netmask: mascara de red.
port: puerto por el que se realiza la conexión.
switchD: dirección switch nodo D
switchE: dirección switch nodo E

```
mutation {
  updateIntersection(id: "5b82e3de797cde4292a97335", data: {
    name:"Santiago - Siete Mascarones",
    code:"SEM500",
    lat:-13.525373747178868,
    lng:-71.96341911659583,
    ethSetting:{
      address:"192.168.71.11",
      gateway:"192.168.71.1",
      netmask:"255.255.255.0",
      dns:"192.168.71.254",
      host:"sigecs.io",
      port:8442,
      switchD: "192.168.7.10",
      switchE: "192.168.7.10"
    },
    day1:"type1",
    day2:"type1",
    day3:"type1",
    day4:"type1",
    day5:"type1",
    day6:"type1",
    day7:"type1",
    type_1_1:{
      start:"00:00:00",
      end:"23:59:59",
      cycles:2,
      phaseV1:"1A1D",
      phaseV2:"1A1D",
      phaseV3:"2D",
      phaseP1:"2D",
      phaseP2:"2D",
      phaseG1:"2D",
      phaseG2:"2D",
      phaseG3:"2D"
    },
    type_1_2:{
      start:"",
      end:"",
      cycles:0,
      phaseV1:"",
      phaseV2:"",
      phaseV3:"",
      phaseP1:"",
      phaseP2:"",
      phaseG1:"",
      phaseG2:"",
      phaseG3:""
    },
    phaseV1:{
      name:"",
      enable:true,
      lat1:-13.525331370070575,
      lng1:-71.96336075686443,
      deg1:20,
      lat2:0,
      lng2:0,
      deg2:0,
      lat3:0,
      lng3:0,
      deg3:0
    },
    phaseV2:{
      name:"",
      enable:true,
      lat1:0,
      lng1:0,
      deg1:0,
      lat2:0,
      lng2:0,
      deg2:0,
      lat3:0,
      lng3:0,
      deg3:0
    },
    installed:true
  }){
    id
  }
}
```

```
{
  "data": {
    "updateIntersection": {
      "id": "5b82e3de797cde4292a97335"
    }
  }
}
```

Fig. 3.40 Mutación para Actualizar una Intersección

Fuente: Elaboración Propia

En la figura 3.41, se muestra la consulta para eliminar la información de una intersección, con los siguientes campos como parámetros de entrada:

id: identificador único del firmware a eliminar.

```
mutation {
  removeIntersection(id: "5b82e3de797cde4292a97335"){
    id
  }
}
```

```
{
  "data": {
    "removeIntersection": {
      "id": "5b82e3de797cde4292a97335"
    }
  }
}
```

Fig. 3.41 Mutación para Eliminar Intersección

Fuente: Elaboración Propia

- **Tarea 21 – Implementación del MQTT-Client:**

Para implementar esta tarea se hizo uso de la librería ***ngx-mqtt***. Para poder hacer uso de esta herramienta primero se descarga los paquetes necesarios mediante npm[10] desde los repositorios oficiales:

```
$npm install ngx-mqt
```

Lo siguiente es realizar las configuraciones necesarias para que se pueda conectar al servidor de red o *bróker.*

La siguiente figura, muestra el código de configuración para el MQTT-Client; donde el *hostname* apunta al dominio "sigecs.io" del servidor MQTT que escucha por el puerto 8443, en la ruta "*/mqtt*" y utiliza el protocolo "*wss*" [11]

```
export const MQTT_SERVICE_OPTIONS: IMqttServiceOptions = {
    hostname: 'sigecs.io',
    port: 8443,
    path: '/mqtt',
    protocol: 'wss',
    rejectUnauthorized: true
};
```

Fig. 3.42 Configuración MQTT-Cli

Fuente: Elaboración Propia

Una vez configurado lo siguiente es construir procedimiento que utilice las funciones disponibles en la librería cada vez que un determinado componente requiera información

[10] Npm: *Node Package Manager* (Gestor de paquetes para node.js)
[11] Wss: WebSocket encriptado

en tiempo real del servidor de red o bróker. La figura siguiente muestra un segmento de código de un procedimiento para obtener el estado en tiempo real de las fases de todos los controladores semafóricos conectados, haciendo una suscripción al tópico "*{cod_semaforo}/ou/states*".

```
this.utilService.get(query, variables: {}, callbackOk: (result: any) => {
    this.intersections = this.utilService.clone(result.data.intersections);
    this.intersections.forEach( callbackFn: (intersection) => {
        this.stationService.disconectIntersection(intersection);
        this.stationService.loadIconMarkers2Intersection(intersection, callbackAddMarker: () => {
            this.addMarker(intersection);
        }, callbackFinish: () => {
            // subscription to mqtt
            let topic = `/${intersection.code}/ou/state/`;
            let sub = this._mqttService.observe(topic).subscribe( next: (message) => {
                let msg = JSON.parse(message.payload.toString());
                let updateIcon = msg['v1'] + '_M' != intersection['stateRT'];
                this.stationService.updateStationFromMsg(intersection, message.payload.toString());
                if (updateIcon) {
                    intersection['marker'].setIcon(intersection['markerIcons'][intersection.stateRT]);
                }
                setTimeout( callback: () => {
                    let dif = (new Date()).getTime() - intersection.lastConnected.getTime();
                    let seconds = Math.abs( x: dif / 1000);
                    if (seconds >= StationService.TIMEOUT_DISCONNECT - 0.3) {
                        if ('N_M' != intersection['stateRT']) {
                            intersection['marker'].setIcon(intersection['markerIcons'][intersection.stateRT]);
                        }
                        this.stationService.disconectIntersection(intersection);
                    }
                }, StationService.TIMEOUT_DISCONNECT);
            });
            this.subsMessage$.push(sub);
        })
    });
});
```

Fig. 3.43 Segmento de Código para Obtener los Estados en Tiempo Real de Todas las Intersecciones

Fuente: Elaboración Propia

- **Tarea 22 – Mantenimiento de Usuarios**

Para realizar esta tarea se crearon dos componentes principales con sus respectivas clases ".ts", plantillas ".html" y hojas de estilo ".css", el primer componente de la figura siguiente, muestra en un formulario la lista de todos los usuarios registrados al sistema y las acciones de actualización, eliminación y creación para nuevos usuario, también se muestra otro sub componente que muestra los roles asignados a cada usuario, el mismo que será explicado con más detalle en la implementación de la tarea 12. Los datos

mostrados en el formulario se obtienen realizando la consulta implementada en la tarea N°9.

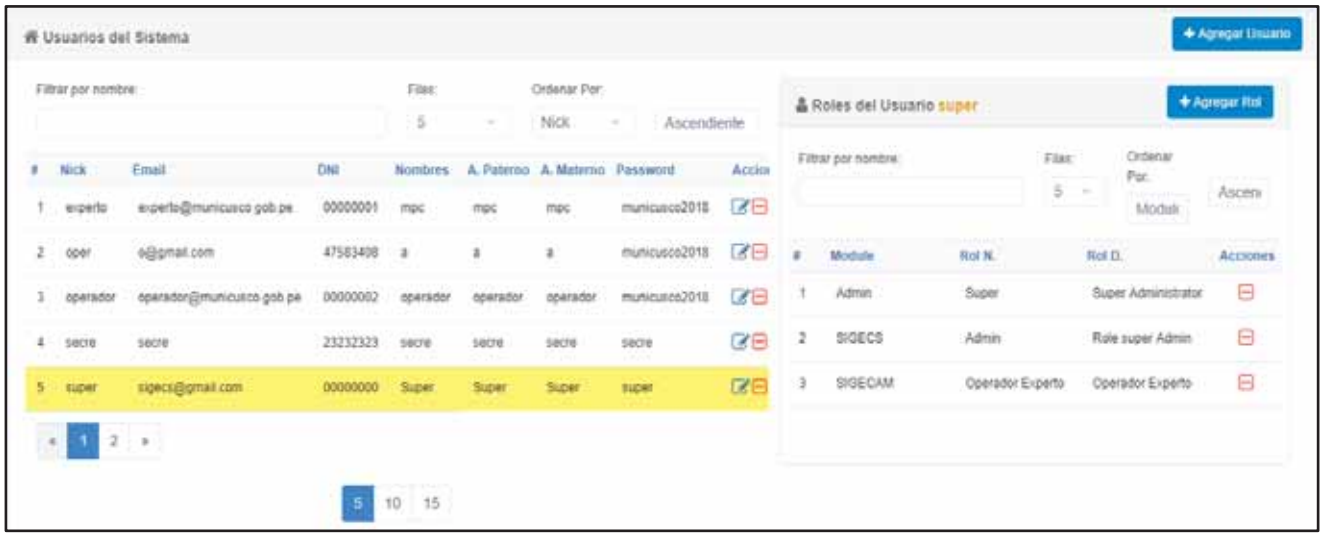


Fig. 3.44 Formulario Principal Mantenimiento Usuarios

Fuente: Elaboración Propia

Los formularios mostrados en la figura 3.45 y 3.46, muestran los componentes para realizar las operaciones de creación y actualización de usuarios respectivamente, los datos registrados en estos formularios son enviados bajo una petición con los parámetros de "*nick, email, dni, nombres, paterno, materno, pawsword*" y almacenados en la base de datos con las consultas implementadas en la tarea N° 14

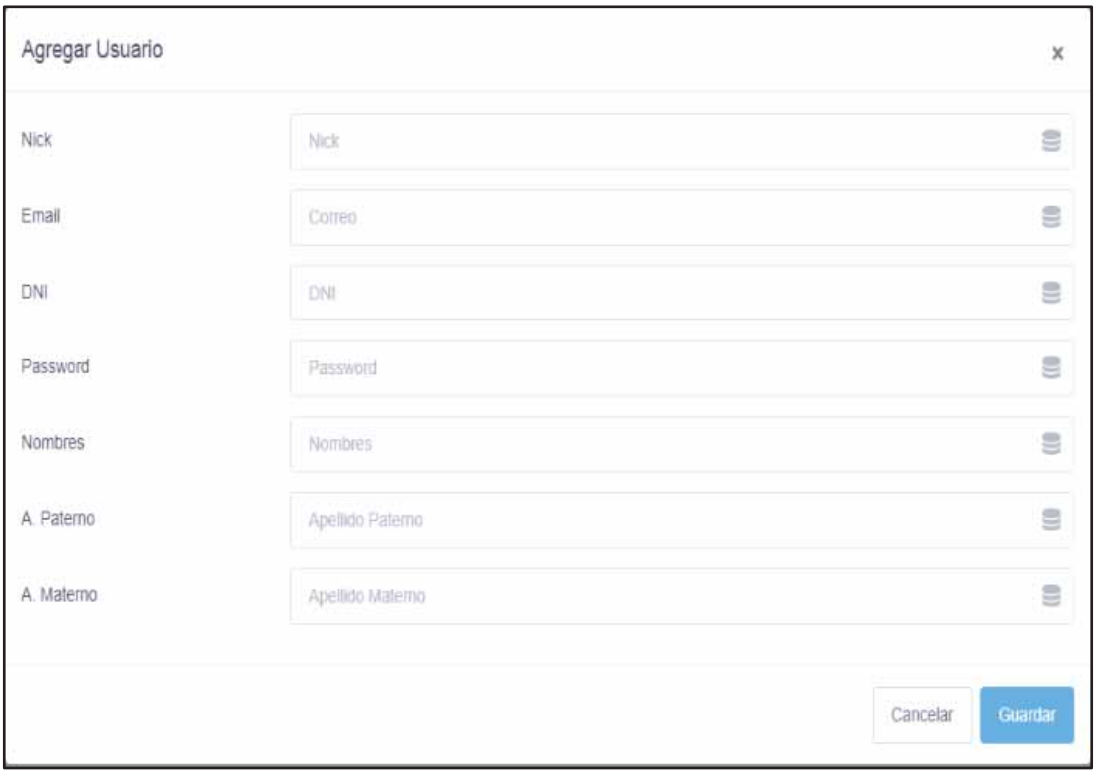


Fig. 3.45 Formulario Agregar Usuario

Fuente: Elaboración Propia

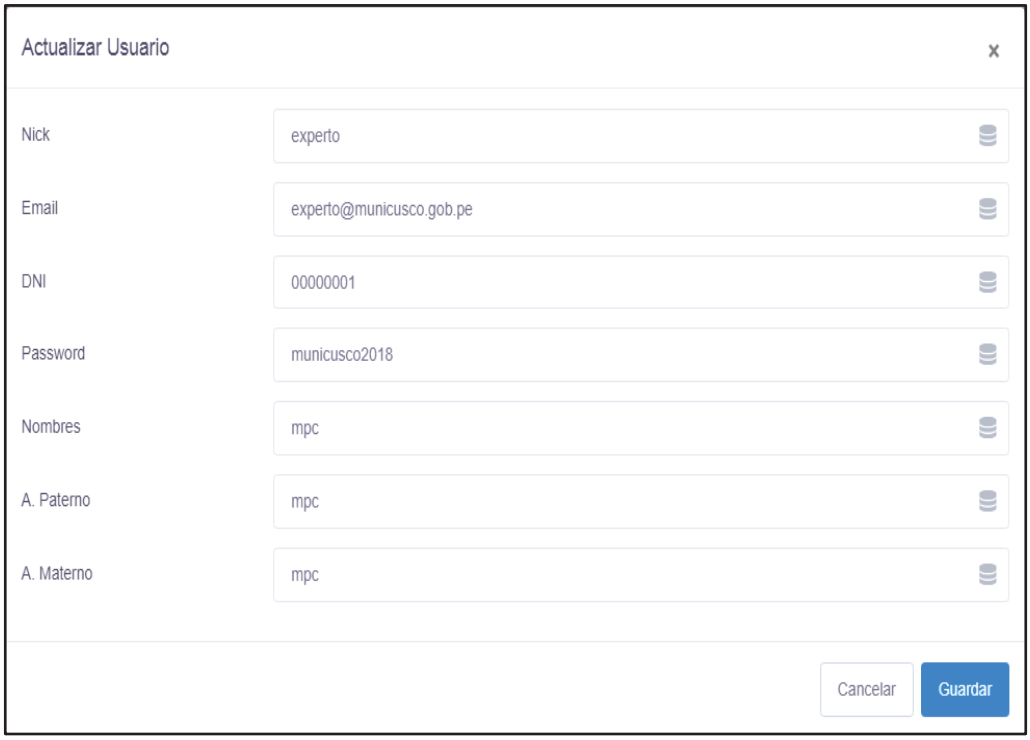


Fig. 3.46 Formulario Actualizar Usuario

Fuente: Elaboración Propia

- **Tarea 23 – Mantenimiento de Módulos**

  Similar a la tarea 22, se creó dos componentes con sus respectivos clases ".ts", plantillas ".html" y hojas de estilo ".css", el primer componente de la figura siguiente, muestra un formulario donde se lista todos los módulos disponibles en la plataforma con sus respectivas acciones de actualización, eliminación y creación para nuevos módulos, también se muestra otro sub componente que muestra las vistas disponibles para cada módulo. Los datos mostrados en el formulario se obtienen realizando las consultas implementadas en la tarea N° 10

Fig. 3.47 Formulario Principal Mantenimiento Módulos

Fuente: Elaboración Propia

Los formularios mostrados en las figuras 3.48 y 3.49, muestran los componentes para realizar las operaciones de creación y actualización de módulos respectivamente, los datos registrados en estos formularios son enviados bajo una petición con los parámetros "*nombre, icono y color*" y almacenados en la base de datos utilizando las consultas implementadas en la tarea N°15.

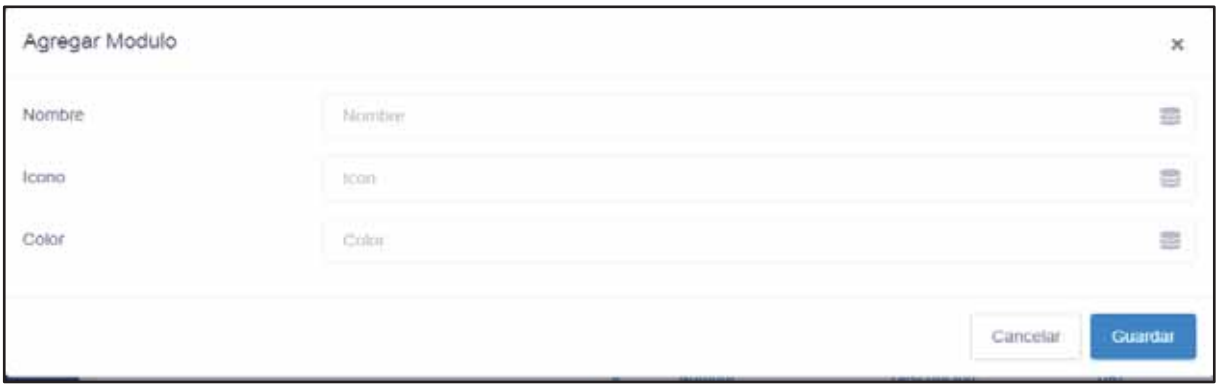


Fig. 3.48 Formulario Agregar Modulo

Fuente: Elaboración Propia

Fig. 3.49 Formulario Actualizar Modulo

Fuente: Elaboración Propia

- **Tarea 24 – Mantenimiento de Vistas**

  Se creó un componente con sus respectivas clases ".ts", plantillas ".html" y hojas de estilo ".css", el primer componente de la figura siguiente, muestra un formulario donde se lista todas las vistas de un determinado modulo con sus respectivas acciones de actualización, eliminación y creación para nuevas vistas. Los datos mostrados en el formulario se obtienen realizando las consultas implementadas en la tarea N°11.

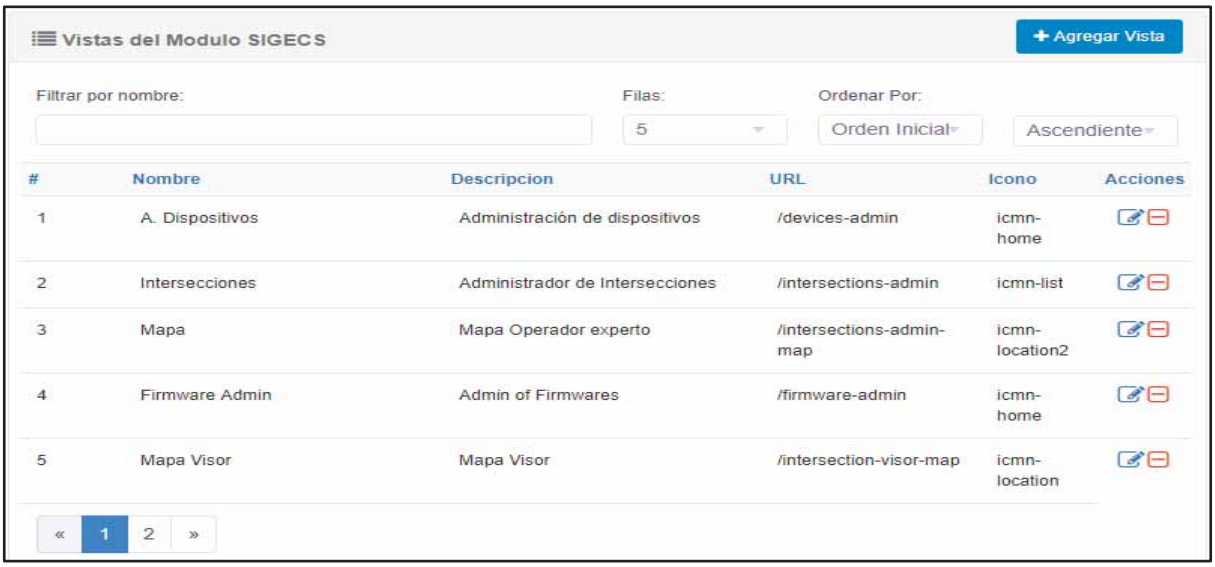


Fig. 3.50 Formulario Principal Mantenimiento Vistas

Fuente: Elaboración Propia

Los formularios mostrados en las figuras 3.51 y 3.51, muestran los componentes para realizar las operaciones de creación y actualización de vistas respectivamente, los datos registrados en estos formularios son enviados y almacenados en la base de datos utilizando las consultas implementadas en la tarea N°16

Fig. 3.51 Formulario Agregar Vista

Fuente: Elaboración Propia

Fig. 3.52 Formulario Actualizar Vista

Fuente: Elaboración Propia

- **Tarea 25 – Mantenimiento de Roles**

Se creó un componente con sus respectivas clases ".ts", plantillas ".html" y hojas de estilo ".css", el primer componente de la figura siguiente, muestra un formulario donde se lista todos los roles de un determinado usuario con sus respectivas acciones de actualización, eliminación y creación para nuevos roles. Los datos mostrados en el formulario se obtienen realizando las consultas implementadas en la tarea N°12.

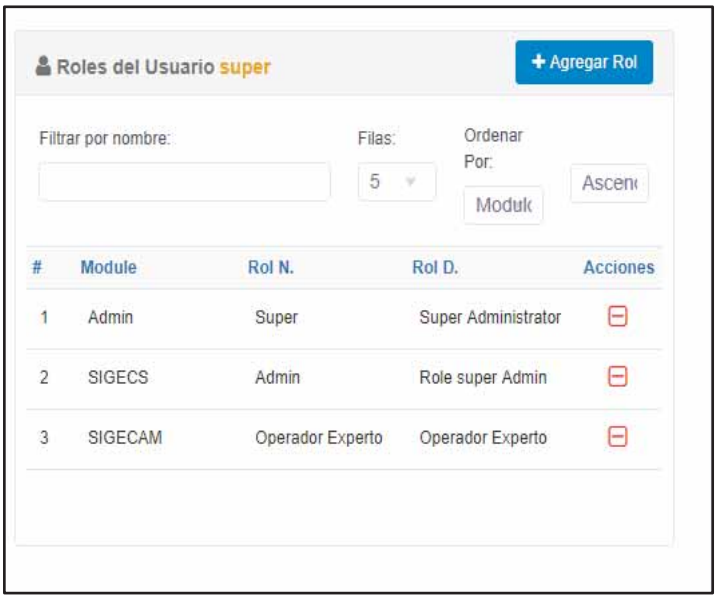


Fig. 3.53 Formulario Principal Mantenimiento Roles

Fuente: Elaboración Propia

El formulario mostrado en la figura 3.54, muestra un componente para realizar las operaciones de creación y actualización de roles, los datos registrados en estos formularios son enviados y almacenados en la base de datos utilizando las consultas implementadas en la tarea N°17.

Fig. 3.54 Formulario Agregar Rol

Fuente: Elaboración Propia

- **Tarea 26 – Mantenimiento de Sensores**

  Se crearon dos componentes principales con sus respectivas clases ".ts", plantillas ".html" y hojas de estilo ".css", el primer componente de la figura siguiente, muestra un formulario la lista de todos los sensores registrados al sistema y las acciones de actualización, eliminación y creación para nuevos sensores, también se muestra otro sub componente que muestra los datos adicionales para cada sensor tales como las variables, comandos, estados y configuraciones. Los datos mostrados en el formulario se obtienen utilizando las consultas implementadas en la tarea N°13.

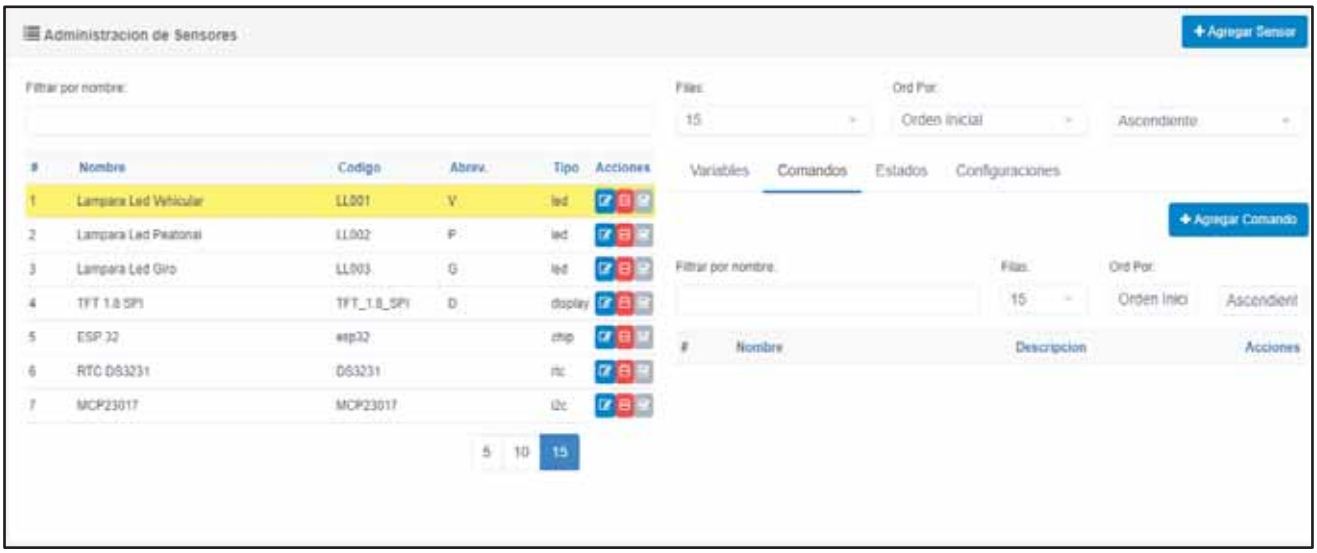


Fig. 3.55 Formulario Principal Mantenimiento Sensor

Fuente: Elaboración Propia

El formulario mostrado en la figura 3.56, muestra un componente para realizar las operaciones de creación y actualización de sensores, los datos registrados en estos formularios son enviados y almacenados en la base de datos utilizando las consultas implementadas en la tarea N°18.

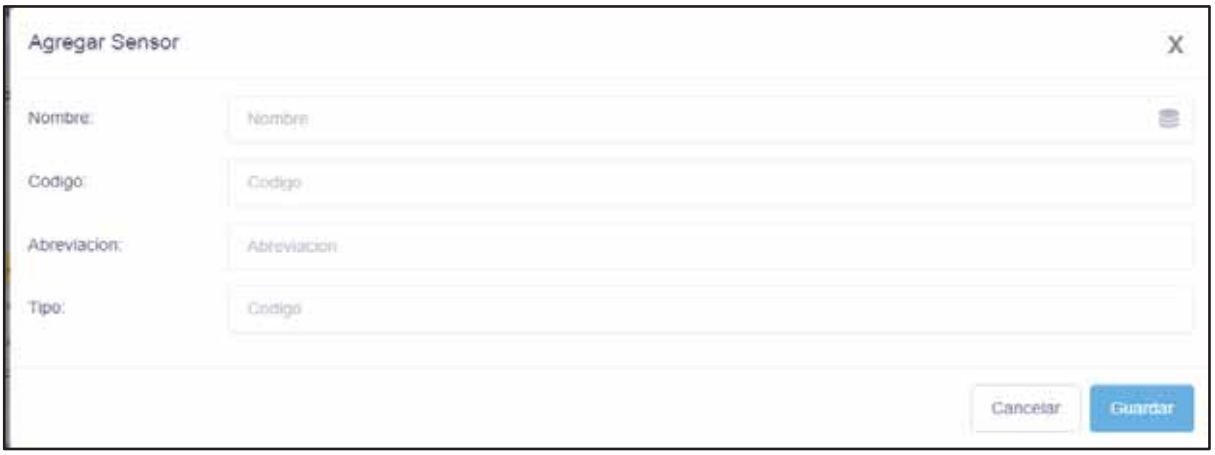


Fig. 3.56 Formulario Agregar Sensor

Fuente: Elaboración Propia

- **Tarea 27 – Mantenimiento de Intersecciones**

Se creó componentes con sus respectivas clases ".ts", plantillas ".html" y hojas de estilo ".css". El formulario de la figura siguiente muestra una lista con las intersecciones registradas al sistema y el estado en tiempo real de todas las fases de cada intersección, también se muestra los controles que permitirán realizar las operaciones de actualización, eliminación, agregación de un nuevo estado, sincronización de fecha y hora, envió de configuraciones prioritarias y actualización de firmware. Los datos mostrados en el formulario se obtienen realizando las consultas implementadas en la tarea N°19

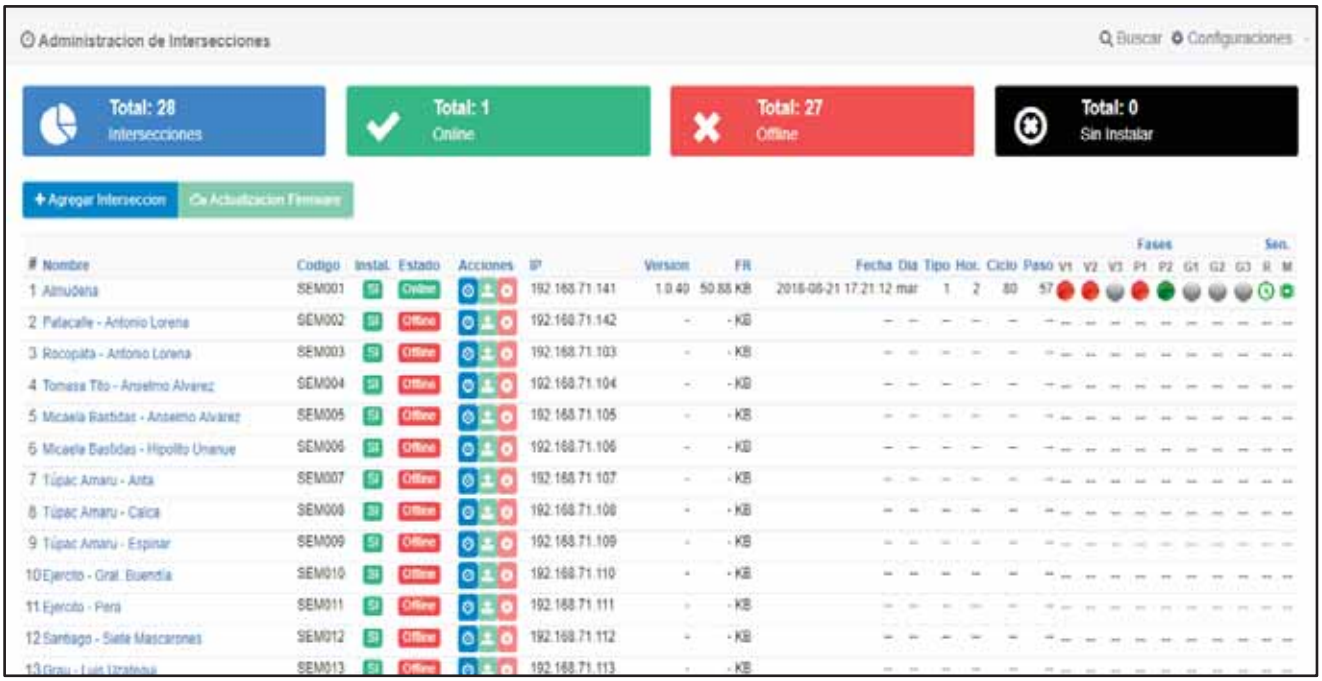


Fig. 3.57 Formulario Principal de Mantenimiento de Intersecciones

Fuente: Elaboración Propia

También se crearon componentes para la configuración[12] de una determinada intersección como se muestra en la figura 3.58, se pueden realizar configuraciones para poder actualizar los datos generales ingresados al momento de crear la intersección.

[12]Es el conjunto de instrucciones que se graban en la memoria del controlador semafórico y que define la lógica de la sincronización, comportamiento y características de cada semáforo conectado a la plataforma.

También se puede observar botones con otras funcionalidades que serán descritas más adelante.

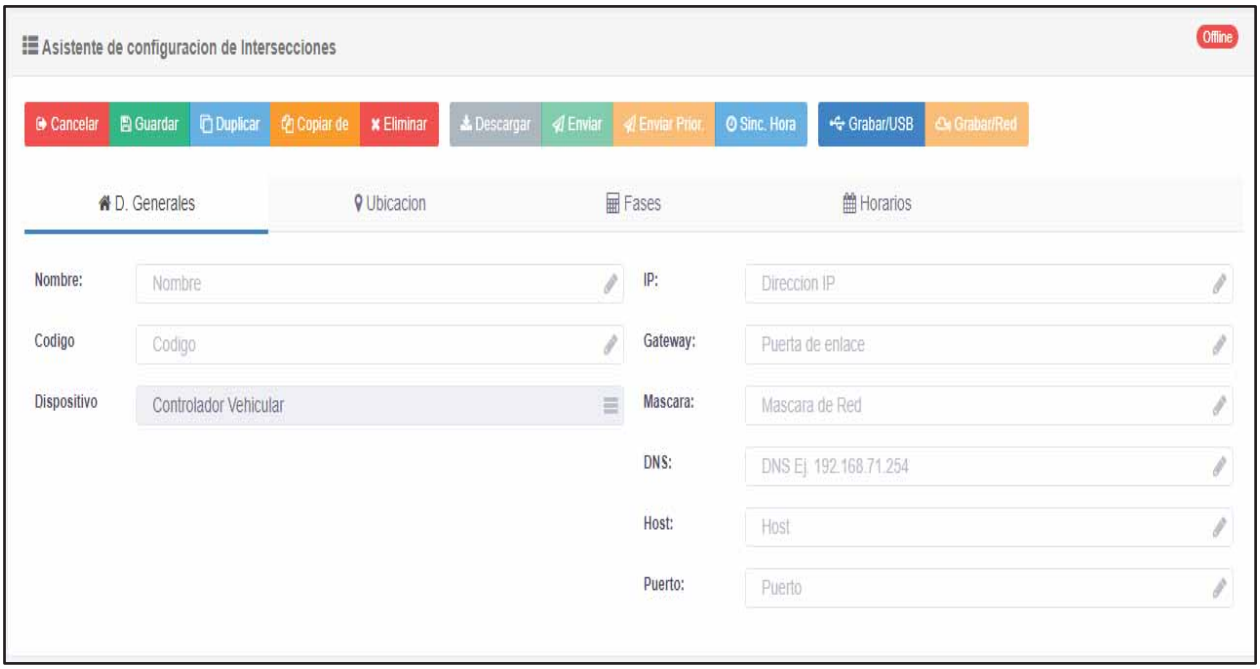


Fig. 3.58 Datos Generales

Fuente: Elaboración Propia

Adicionalmente estos componentes permiten registrar la geolocalización de una determinada intersección arrastrando y soltando el marker a través del mapa, tal como se muestra en la figura 3.59.

Fig. 3.59 Geolocalización Intersecciones

Fuente: Elaboración Propia

También es posible configurar la geolocalización de cada una de las fases que conforman una intersección.

Fig. 3.60 Geolocalización de Semáforos

Fuente: Elaboración Propia

La figura 3.61, muestra un formulario en el que se configuran cada una de las fases que conforman una intersección, habilitar o deshabilitar una fase, registrar el nombre para cada fase. También es posible asignar los intervalos de tiempo para cada tipo de programación horaria.

Fig. 3.61 Configuración de Fases

Fuente: Elaboración Propia

Mientras que la figura 3.62, muestra un formulario en el que se configuran la secuencia para cada fase en cada uno de los horarios definidos en los formularios mostrados en la figura 3.61. También es posible asignar el tipo de programación para los 7 días de la semana.

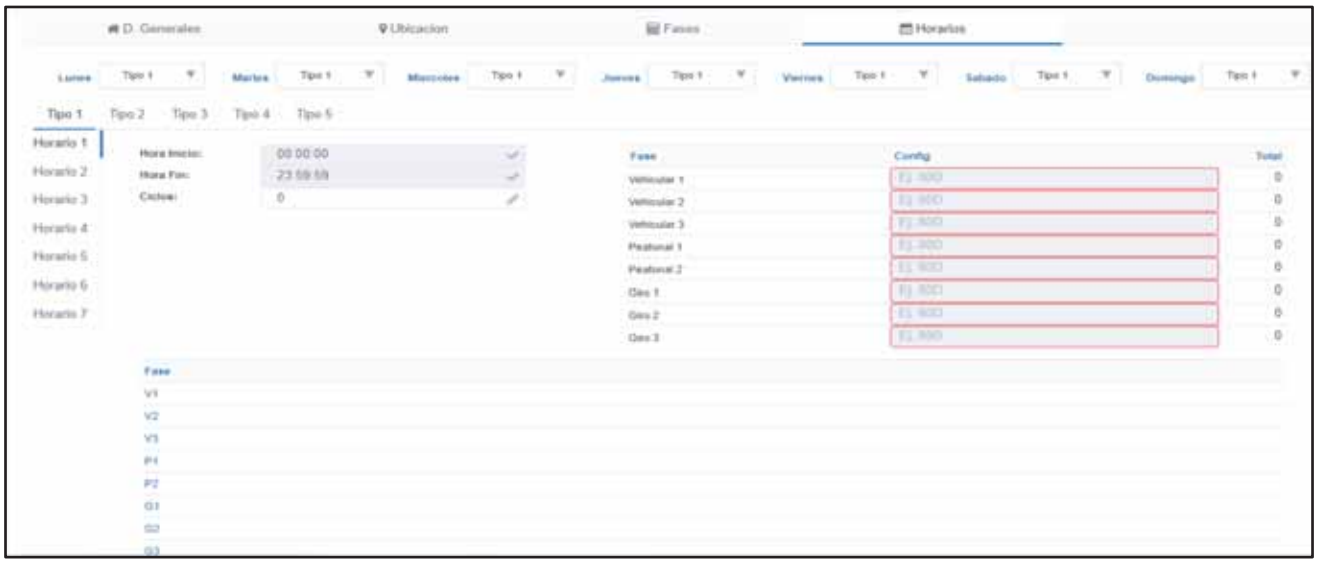


Fig. 3.62 Configuración de Horarios

Fuente: Elaboración Propia

- **Tarea 28 – Mapa de Geolocalización de Semáforos**

Se creó un componentes con sus respectivos clases ".ts", plantillas ".html" y hojas de estilo ".css", el primer componente de la siguiente figura, muestra un mapa con implementado a partir de las imágenes de "*open Street map*"[13], sobre esto se colocan *markers* con colores, de acurdo al estado de la fase principal, representando la geolocalización de las intersecciones semafóricas. Los puntos GPS[14] de cada intersección, se obtiene realizando las consultas implementadas en la tarea N°19

[13] Proyecto colaborativo para crear mapas editables y libres.
[14] GPS: Sistema de Posicionamiento Global

También se muestra el estado en tiempo real de todas las fases de todas las intersecciones. Esta información se obtiene desde el servidor de red o *bróker* haciendo uso del servicio que se implementó en el MQTT-Client.

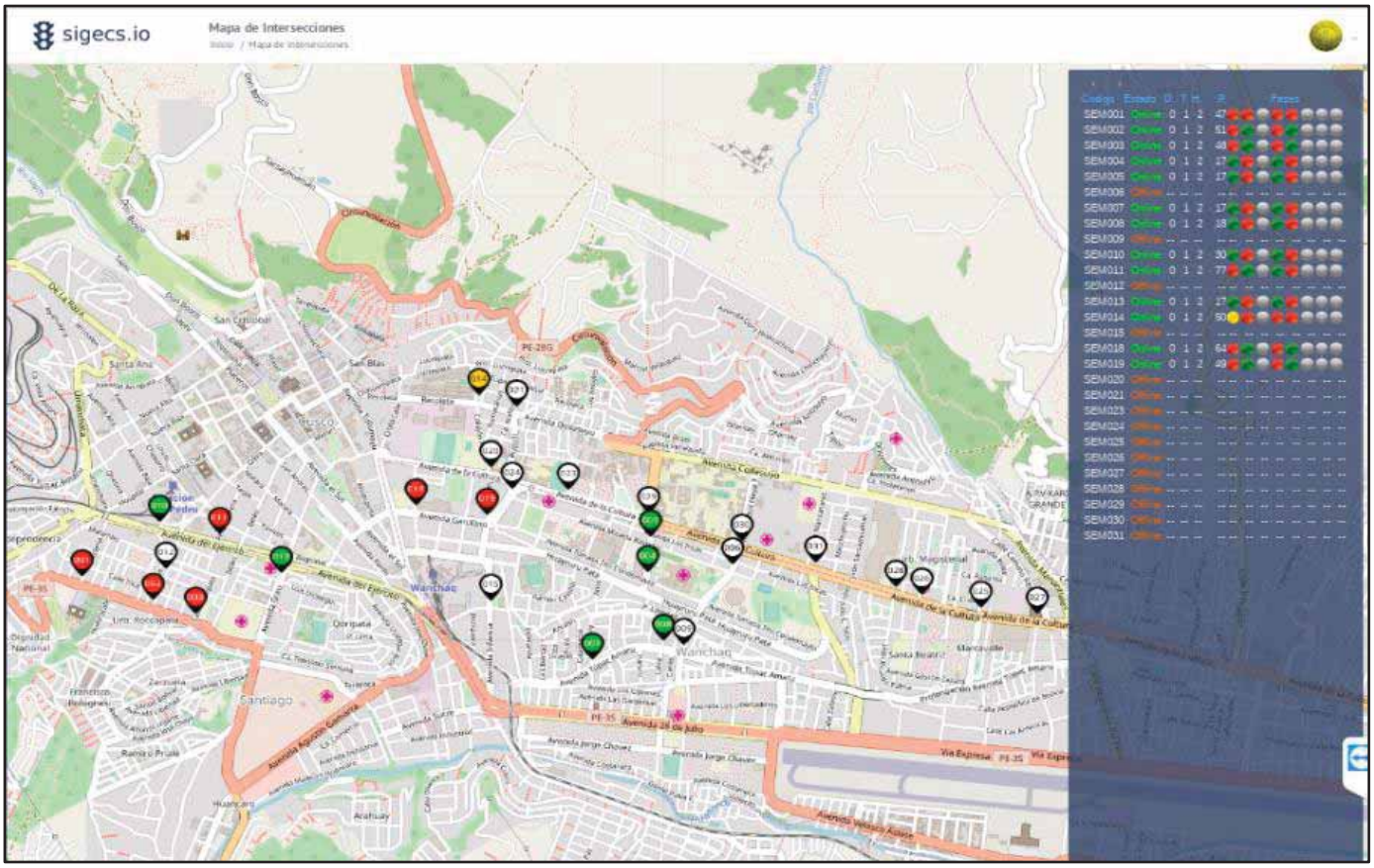


Fig. 3.63 Formulario Mantenimiento de Configuraciones

Fuente: Elaboración Propia

- **Tarea 29 – Asistente de Configuración de Intersecciones**

  Se creó componentes con sus respectivas clases ".ts", plantillas ".html" y hojas de estilo ".css". El formulario mostrado en la figura siguiente es un asistente de configuración que facilita la configuración y otras operaciones auxiliares que se pueden realizar en una intersección. Para ello se crearon componentes para:

  - Duplicar la configuración de la intersección actual hacia una nueva (Fig. 3.65).

- Copiar la configuración desde otra intersección hacia la intersección actual (Fig. 3.66).
- Descargar una configuración en formato JSON,
- Enviar la configuración hacia el controlador semafórico
- Enviar una configuración con mayor prioridad frente al programado (Fig. 3.67 y 3.68)
- Cancelar la configuración prioritaria, regresando a la configuración normal
- Sincronizar la fecha y hora del controlador semafórico con la del servidor NTP.

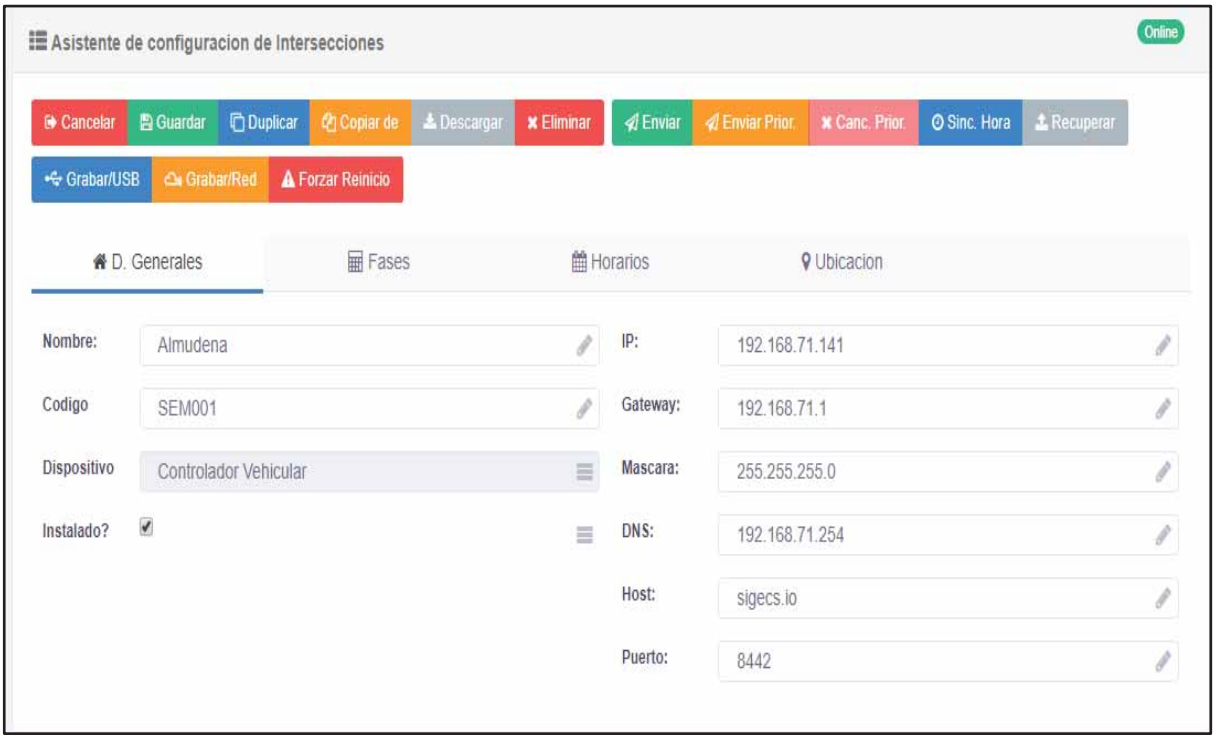


Fig. 3.64 Formulario Asistente de Configuración

Fuente: Elaboración Propia

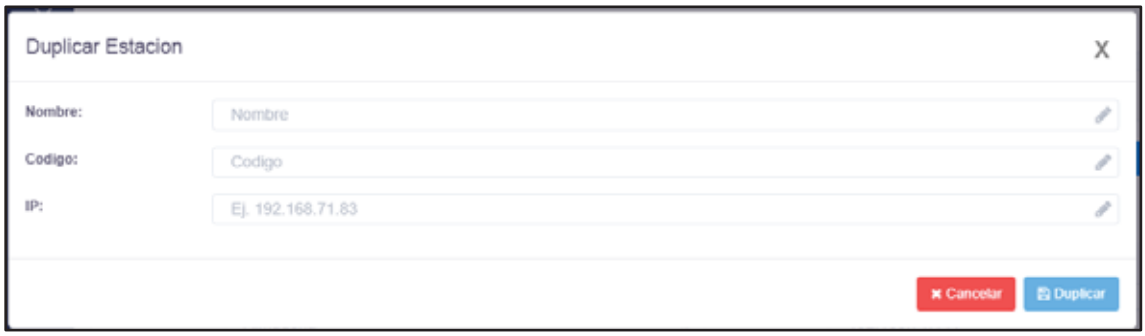


Fig. 3.65 Formulario Duplicar Intersección

Fuente: Elaboración Propia

Fig. 3.66 Formulario Copiar Configuración

Fuente: Elaboración Propia

Fig. 3.67 Mapa de Geolocalización de Semáforos

Fuente: Elaboración Propia

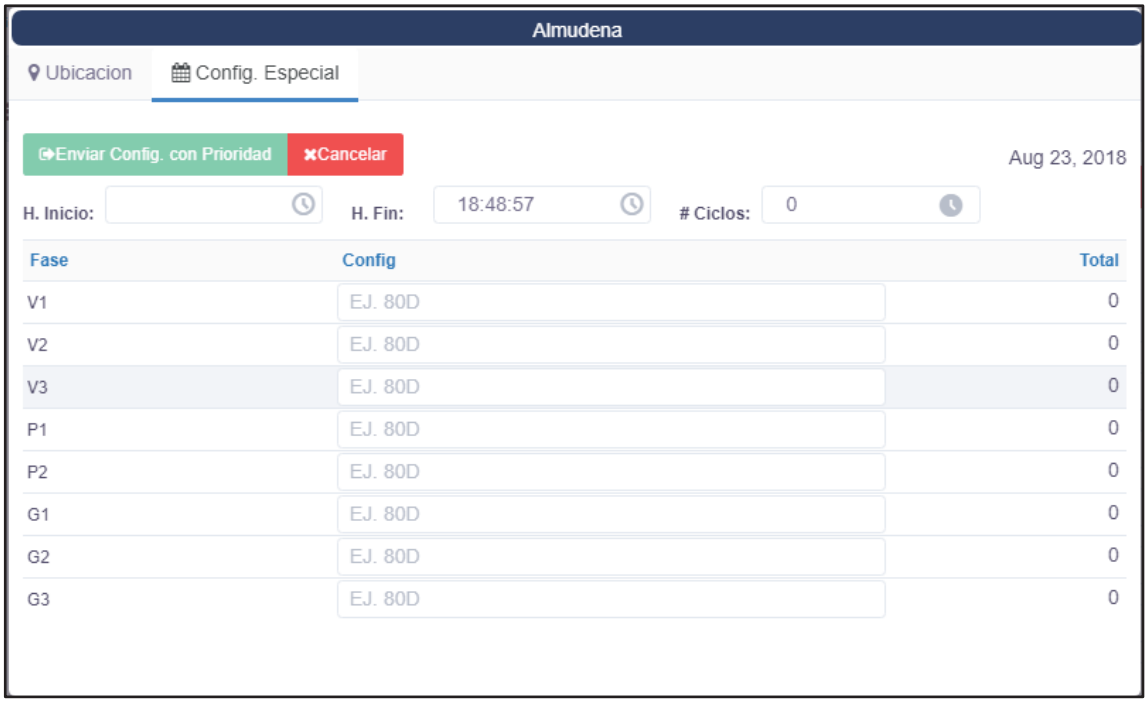


Fig. 3.68 Formulario Envió Prioritario

Fuente: Elaboración Propia

3.2.2.5 Servidor MQTT

- **Tarea 30 – Implementación del Servidor de Red o *Bróker*:**

  El bróker implementado se basó en la librería *https://github.com/mcollina/mosca*.

  Para instalar la librería usamos el comando npm (*Node Package Manager*).

  ```
  $ npm install mosca
  ```

  Se importa la librería en el proyecto.

```
3    import * as mosca from 'mosca';
```

Fig. 3.69 Importación de "mosca" al Proyecto

Fuente: Elaboración Propia

Se lee las variables de entorno, las cuales servirán para las configuraciones del bróker.

```
7   let DB_USER = process.env.DB_USER;
8   let DB_PASSWORD = process.env.DB_PASSWORD;
9   let DB_HOST = process.env.DB_HOST;
10  let DB_PORT = process.env.DB_PORT;
11  let DB_NAME = process.env.DB_NAME;
12  let API_HOST = process.env.API_HOST;
13  let API_PORT = process.env.API_PORT;
```

Fig. 3.70 Lectura de las Variables de Entorno

Fuente: Elaboración Propia

Realizar las configuraciones necesarias para la librería, donde se especifica el certificado TLS 1.2, lo cual nos permite conectarnos al *bróker* haciendo uso del protocolo "*mqtts*" por el puerto 8442, y mediante el protocolo "*wss*" por el puerto 8443, los dispositivos que se conectan a este *bróker* hacen uso del protocolo "*mqtts*", mientras que los clientes web, se conectan haciendo uso de web sockets seguros (*wss*).

```
54      private ascoltatore = {
55          type: 'mongo',
56          url: `mongodb://${DB_USER}:${DB_PASSWORD}@${DB_HOST}:${DB_PORT}/${DB_NAME}`,
57          pubsubCollection: 'ascoltatori',
58          mongo: {}
59      };
60
61      private moscaSettings = {
62          port: 4066,
63          backend: this.ascoltatore,
64          secure: {
65              port: 8442,
66              keyPath: SECURE_KEY,
67              certPath: SECURE_CERT,
68          },
69          https: {
70              port: 8443,
71              bundle: true,
72              static: './'
73          },
74          allowNonSecure: true
75      };
```

Fig. 3.71 Configuraciones para Conectar al bróker

Fuente: Elaboración Propia

```
87          this.server = new mosca.Server(this.moscaSettings);
```

Fig. 3.72 Código para Iniciar el Servidor

Fuente: Elaboración Propia

- **Tarea 31 – Instalar contenedor de Bróker:**

  Se crea el script "*Dockerfile*" para la creación de la imagen del contenedor del bróker.

  Se toma como base la distribución Centos en su versión 7.

```
FROM centos:centos7
```

  Se actualiza la imagen base.

```
RUN yum update -y; yum clean all
```

  Se copia los certificados CA y se actualiza la imagen.

```
ADD tls/rootCA.pem /etc/pki/ca-trust/source/anchors/
RUN update-ca-trust
RUN update-ca-trust extract
```

  Se expone los puertos públicos. (8442 mqtts y 8443 mqtt sobre wss (websocket secure)).

```
EXPOSE 4066
EXPOSE 8442
EXPOSE 8443
```

Se configura las variables de entorno, que serán usadas desde el binario del API, para acceder correctamente a la base de datos y servir los servicios (mutaciones y consultas).

```
ENV DB_USER="admin"
ENV DB_PASSWORD="password"
ENV DB_HOST="172.17.0.1"
ENV DB_PORT="27017"
ENV DB_NAME="mosca"
ENV SERVER_PORT="4017"
ENV API_HOST="172.17.0.1"
ENV API_PORT="4017"
ENV NODE_ENV="production"
```

Se instala nodejs (dependencia para ejecutar mosca).

```
RUN curl --silent --location https://rpm.nodesource.com/setup_8.x | bash -
RUN yum -y install nodejs
```

Se instala PM2 (nos permite ejecutar un aplicativo desarrollado en nodejs como un demonio).

```
RUN npm i -g pm2
```

Se crea los directorios para luego copiar los archivos necesarios, con la siguiente estructura.

/opt
 /opt/mosca
  /opt/mosca/build
  /opt/mosca/tls
  /opt/mosca/node_modules

```
RUN mkdir /opt/mosca
COPY build /opt/mosca/build
COPY tls /opt/mosca/tls
COPY node_modules /opt/mosca/node_modules
```

Copiamos el script principal y le concedemos permiso de ejecución.

```
ADD docker/run.sh /run.sh
RUN ["chmod", "+x", "/run.sh"]
```

```
#!/bin/bash

echo "RUN MOSCA"
pm2 start /opt/mosca/build/index.js --no-daemon
```

Este script ejecutará el bróker (mosca) como un demonio.

Se ejecuta el script principal.

```
CMD ["/run.sh"]
```

El cual es creado con el siguiente comando:

```
$ docker build –t sigecs-mosca
```

Para lanzar el contenedor usamos el siguiente comando:

```
$ docker run
    --name=sigetran
    --env='DB_USER=admin'
    --env='DB_PASSWORD=password'
    --env='DB_HOST=172.17.0.1'
    --env='DB_PORT=27017'
    --env='DB_NAME=mosca'
    --env='API_HOST=172.17.0.1'
    --env='API_PORT=4017'
    --env='NODE_ENV=production'
  sigetran-mosca
```

Una vez lanzado se puede observar el contenedor en ejecución.

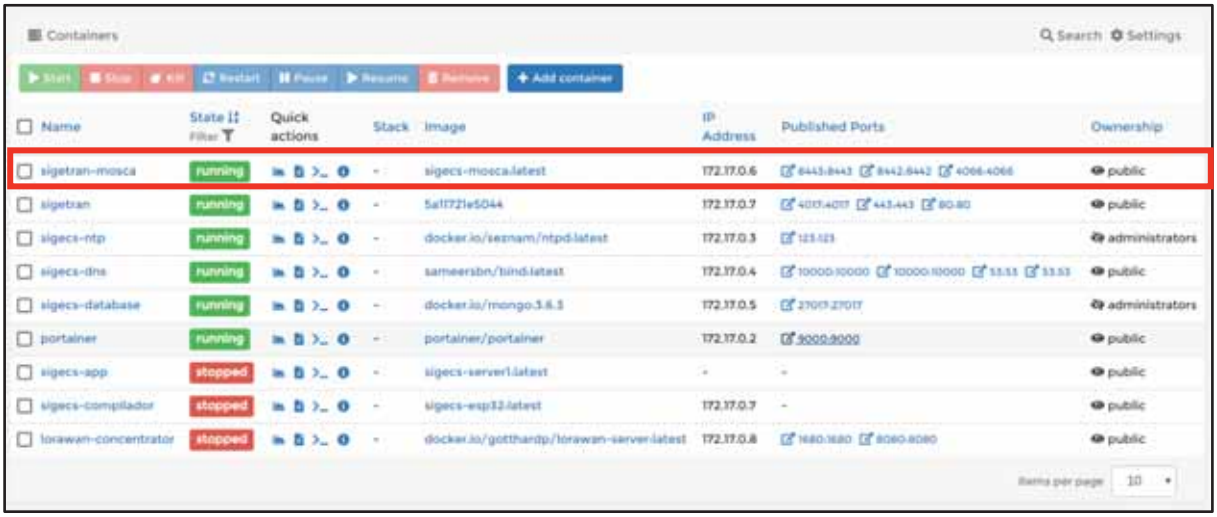


Fig. 3.73 Contenedor del Servidor de Red en Ejecución.

Fuente: Elaboración Propia

3.2.3 Pruebas y resultados

Para realizar la validación del funcionamiento de la plataforma IoT para la gestión de los controladores semafóricos, es necesario realizar pruebas tanto para evaluar la funcionalidad como la usabilidad, para ello se coordinó con la Gerencia de Transito Vialidad y Transporte de la Municipalidad Provincial del Cusco para poder desplegar los contendores que conforman la plataforma dentro de los servidores que ellos administran. Luego se procedió a registrar y validar haciendo uso de las vistas de mantenimiento básicas implementadas, los datos necesarios para poder hacer uso de la plataforma.

En esta parte solo se mostrará las pruebas y los resultados obtenidos de los procesos más importantes implementados para esta iteración además que definen la esencia y la importancia de este proyecto.

La figura 3.74 muestra toda la información del estado, "*Rojo, Ámbar y Verde*", en tiempo real en el que se encuentra todas las fases de cada una de las intersecciones registradas a la plataforma y como estos van cambiando de color de acuerdo a la configuración establecida para cada controlador. También se puede observar otros datos que reporta cada controlador como la cantidad de memoria RAM que usa cada controlador, la fecha y hora, el tipo de configuración semanal, el horario en el que se encuentra, el número de ciclos programados y el paso actual en el que se encuentra.

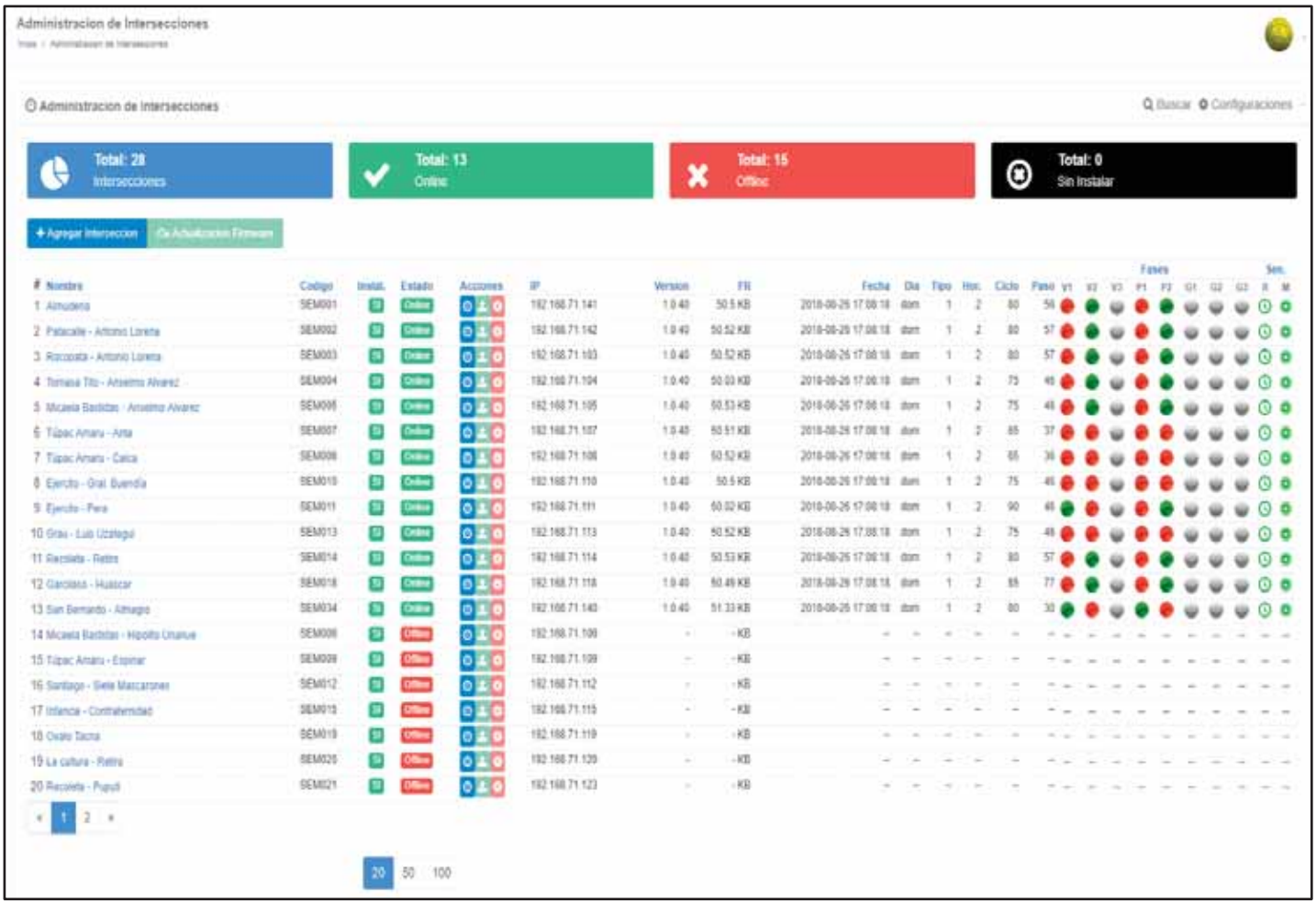


Fig. 3.74 Gestión en Tiempo Real

Fuente: Elaboración Propia

Otro de los procesos que es necesario validar es el funcionamiento y usabilidad de la plataforma en la configuración de intersecciones, para ello se muestra una interfaz que simplifica dicha configuración mediante vistas que están agrupadas y distribuidas de tal forma que para realizar este proceso el encargado utilice el menor tiempo posible y esta pueda ser enviado desde una estación base y grabado en la memoria del controlador, todo esto de forma remota. En la figura 3.75, se puede visualizar una parte de la interfaz "Asistente de Configuración de Intersecciones", específicamente la configuración de horarios, allí se observa cual es el tipo y la hora de programación en segmentos de tiempo que sigue el controlador semafórico así como también la

programación que tiene cada una de las fases de un determinada intersección. Estos datos se encuentran almacenados tanto en la base de datos como en la memoria del controlador semafórico.

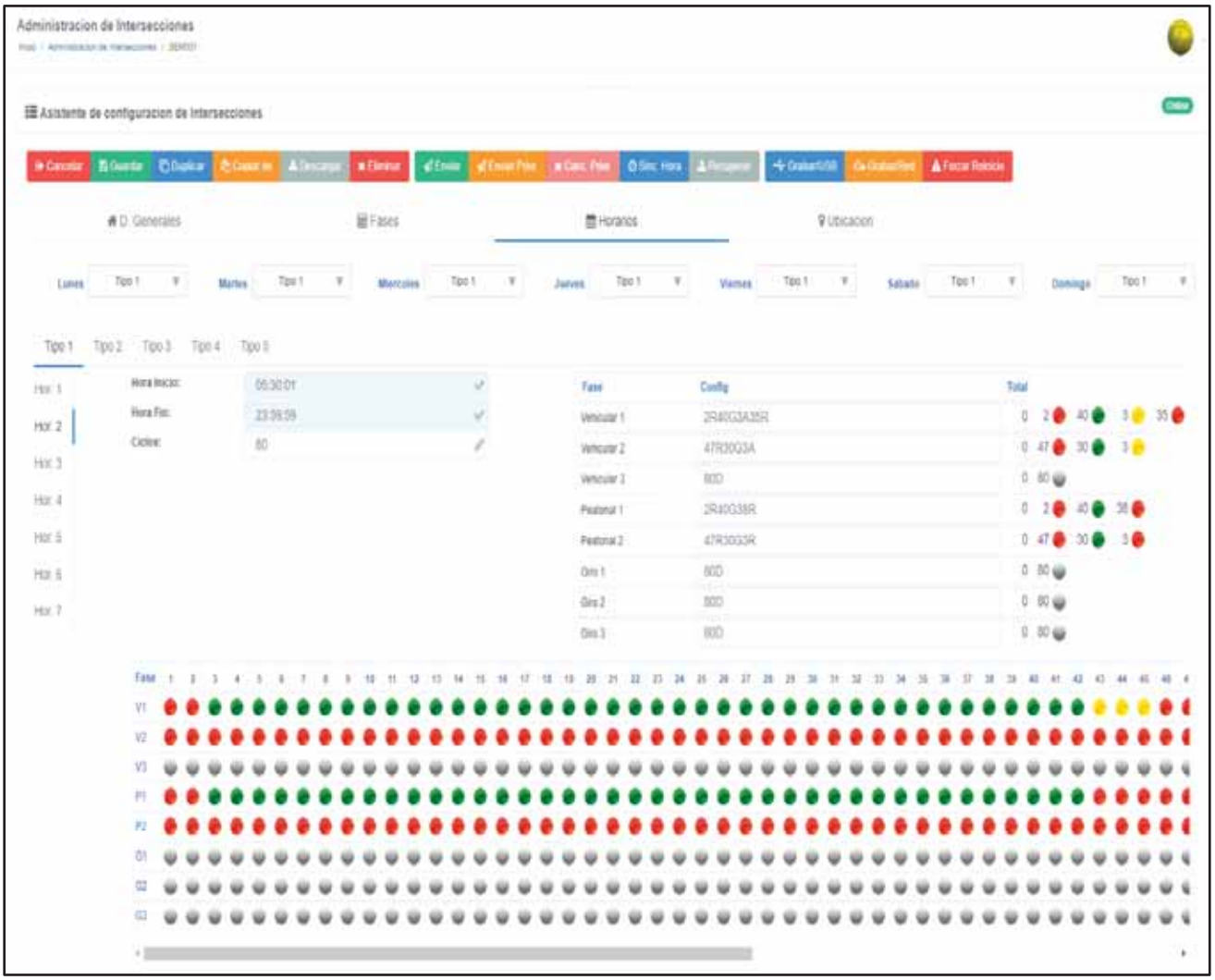


Fig. 3.75 Prueba Configuración de Intersecciones

Fuente: Elaboración Propia

Como parte de los objetivos de este proyecto es poder gestionar la actividad de las intersecciones semafóricas en la ciudad del Cusco; para lo cual se diseñó e implemento vistas que permitan visualizar lo más real posible el estado de cada intersección, es por ello que la figura 3.76, muestra la distribución de todas las intersecciones en un mapa y como estos van cambiando de color en tiempo real de acuerdo al estado de la fase principal *"Fase Vehicular V1"*, validando así que los

datos de geolocalización se ha registrado correctamente en la base de datos y como el servidor de red administra óptimamente los datos que vienen reportando los controladores semafóricos.

Por otro lado, la figura 3.77, muestra la posibilidad de hacer un *Zoom* hacia una intersección en, este caso a la intersección nombrada como **Almudena**, y así poder visualizar una abstracción de la posición y color del estado real de todas las fases (semáforos) que tiene dicha intersección.

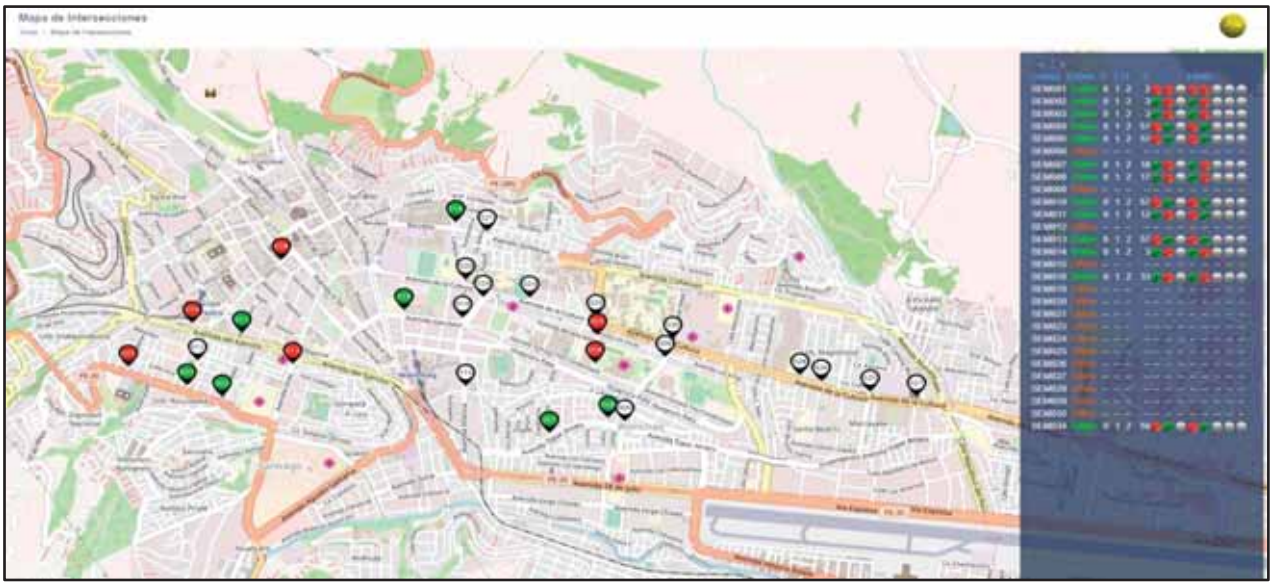


Fig. 3.76 Prueba – Mapa en Tiempo Real

Fuente: elaboración Propia.

Fig. 3.77 Prueba – Mapa Estado de las Fases.

Fuente: Elaboración Propia

Parte del objetivo en gestión que se pretende lograr, es la capacidad de actuar oportunamente en base a alguna eventualidad que se pudiera presentar en la administración del tránsito de la ciudad el Cusco, es por este motivo que se implementa interfaces en las cuales se pueda enviar una configuración especial que se ejecutara con mayor prioridad al establecido diariamente. Los datos de esta configuración se enviaran desde una vista en el Cliente Web, y será almacenado tanto en la base de datos y grabado en la memoria del controlador seleccionado, esta prueba la intersección nombrada como **Almudena**, la configuración especial enviada es ejecutada después de que el contador de ciclos de la configuración normal llegue a su fin, haciendo posible actuar oportunamente ante cualquier situación de emergencia. Véase la figura 3.78.

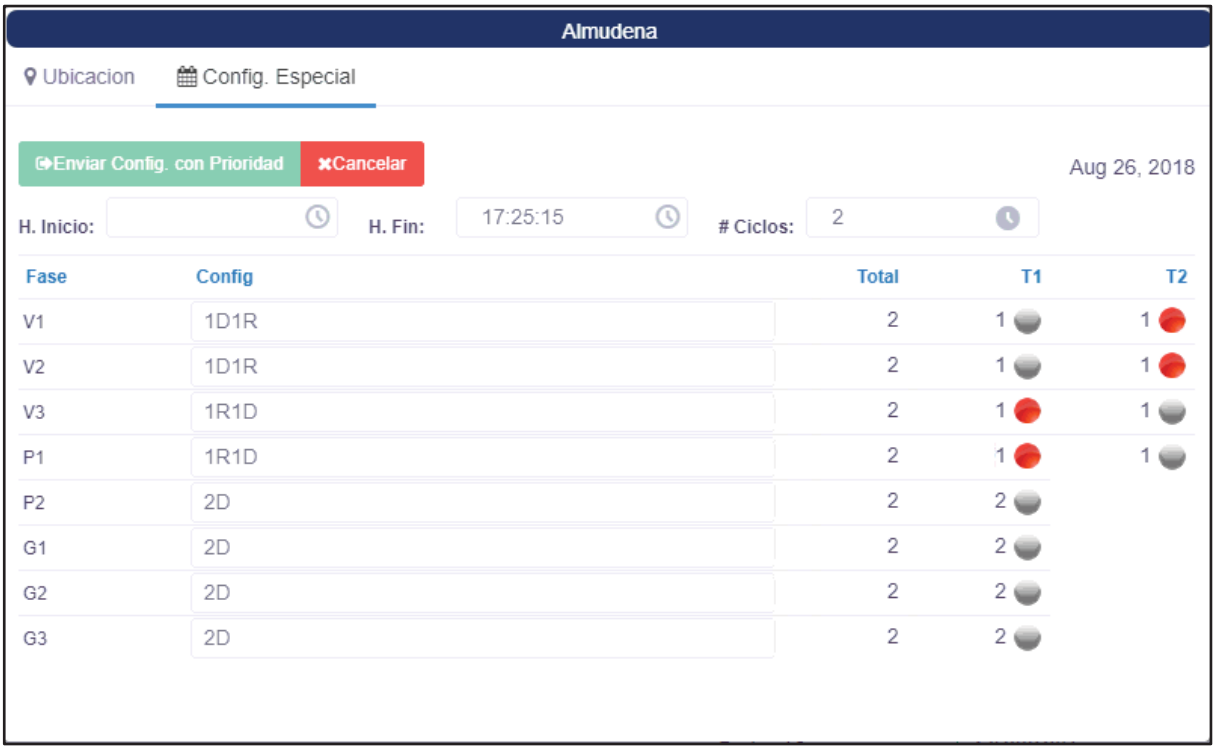


Fig. 3.78 Prueba – Estado de las Fases en Tiempo Real.

Fuente: Elaboración Propia

3.3 Sprint 2

3.3.1 Fase de planeación

En esta fase se contempla el análisis de las historias de usuario captadas en la fase de inicio y obtener el conjunto de actividades "**Tareas**" necesarias para el cumplimiento de estos requerimientos. Adicionalmente es necesario enlistar las tareas internas necesarias a ser realizadas en esta iteración, en base a las planteadas en la tabla 3.13. Y las nuevas que surgieron después del lanzamiento del primer sprint. La siguiente tabla muestra la lista de actividades "Sprint Backlog" seleccionadas para esta segunda iteración.

Tabla 3.46 Sprint Backlog 2

N° Actividad	Objetivo
1	Implementación de la API/GraphQL
	Implementación de *Querys*
	Implementación de *Mutations*
2	Cliente Web
	Vista Administración de dispositivos
	Vista Administración de estados
	Vista Administración de configuraciones
	Vista Administración de firmware
	Mejorar Vista Administración de Intersecciones
	Mejorar Vista *wizard* mapa geolocalización de semáforos
	Vista enviar firmware
	Vista de Auditoria de eventos

Fuente: Elaboración propia

3.3.1.1 API/GraphQL

En esta etapa se definirá las tareas a realizar para poder implementar la API de consultas con GraphQL, donde cada punto de entrada es un nodo que puede conectar con otros nodos para recuperar información (*Query*) o puntos de entrada que modifiquen la base de datos (*Mutation*).

Tabla 3.47 Tarea – Consulta para Recuperar Datos de Dispositivos

Numero: 32	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea: Consulta para Recuperar Datos de Dispositivos	
Descripción: El responsable de la tarea deberá implementar una consulta para la recuperación de los datos de los dispositivos almacenados en la base de datos.	

Fuente: Elaboración Propia

Tabla 3.48 Tarea – Consulta para Recuperar Datos de Estados

Numero: 33	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea: Consulta para Recuperar Datos de Estados	
Descripción: El responsable de la tarea deberá implementar una consulta para la recuperación de los datos de los estados almacenados en la base de datos.	

Fuente: Elaboración Propia

Tabla 3.49 Tarea – Consulta para Recuperar Datos de Configuraciones

Numero: 34	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea: Consulta para Recuperar Datos de Configuraciones	
Descripción: El responsable de la tarea deberá implementar una consulta para la recuperación de los datos de las configuraciones almacenadas en la base de datos.	

Fuente: Elaboración Propia

Tabla 3.50 Tarea – Consulta para Recuperar Datos de Firmware

Numero: 35	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea: Consulta para Recuperar Datos de Firmware	
Descripción: El responsable de la tarea deberá implementar una consulta para la recuperación de los datos de la versión del firmware almacenadas en la base de datos.	

Fuente: Elaboración Propia

Tabla 3.51 Tarea – Mutaciones para Modificar Datos de Dispositivos

Numero: 36	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea: Mutaciones para Modificar Datos de Dispositivos	
Descripción: El responsable de la tarea deberá implementar una mutación para la modificación de los datos de los dispositivos almacenados en la base de datos.	

Fuente: Elaboración Propia

Tabla 3.52 Tarea – Mutaciones para Modificar Datos de Estados

Numero: 37	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea de Equipo: Mutaciones para Modificar Datos de Estados	
Descripción: El responsable de la tarea deberá implementar una consulta para la recuperación de los datos de los estados almacenados en la base de datos.	

Fuente: Elaboración Propia

Tabla 3.53 Tarea – Mutaciones para Modificar Datos de Configuraciones

Numero: 38	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea de Equipo: Mutaciones para Modificar Datos de Configuraciones	
Descripción: El responsable de la tarea deberá implementar una mutación para la modificación de los datos de las configuraciones almacenadas en la base de datos.	

Fuente: Elaboración Propia

Tabla 3.54 Tarea – Mutaciones para Modificar Datos de Firmware

Numero: 39	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea de Equipo: Mutaciones para Modificar Datos de Firmware	
Descripción: El responsable de la tarea deberá implementar una mutación para la modificación de los datos de la versión del firmware almacenadas en la base de datos.	

Fuente: Elaboración Propia

Tabla 3.55 Tarea – Consulta para Recuperar Historial de Eventos

Numero: 40	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea de Equipo: Consulta para Recuperar Historial de Eventos	
Descripción: El responsable de la tarea deberá implementar una consulta para recuperar un historial de eventos realizados sobre la configuración de las intersecciones.	

Fuente: Elaboración Propia

3.3.1.2 Aplicación web

Tabla 3.56 Tarea – Mantenimiento de Dispositivos

Numero: 41	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea: Mantenimiento de Dispositivos	
Descripción: El responsable de la tarea deberá implementar una interfaz en el cual un usuario autorizado, podrá realizar operaciones de inserción, actualización y eliminación de los dispositivos registrados a la plataforma.	

Fuente: Elaboración Propia

Tabla 3.57 Tarea – Mantenimiento de Estados

Numero: 42	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea: Mantenimiento de Estados	
Descripción: El responsable de la tarea deberá implementar una interfaz en el cual un usuario autorizado, podrá realizar operaciones de inserción, actualización y eliminación de los estados registrados a la plataforma.	

Fuente: Elaboración Propia

Tabla 3.58 Tarea – Mantenimiento de Configuraciones

Numero: 43	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea: Mantenimiento de Configuraciones	
Descripción: El responsable de la tarea deberá implementar una interfaz en el cual un usuario autorizado, podrá realizar operaciones de inserción, actualización y eliminación de las configuraciones registradas a la plataforma.	

Fuente: Elaboración Propia

Tabla 3.59 Tarea – Mantenimiento de Firmware

Numero: 44	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea: Mantenimiento de Firmware	
Descripción: El responsable de la tarea deberá implementar una interfaz en el cual un usuario autorizado, podrá realizar operaciones de inserción, actualización y eliminación de las versiones de firmware del controlador semafórico registradas a la plataforma.	

Fuente: Elaboración Propia

Tabla 3.60 Tarea – Asistente de Instalación de Módulos

Numero: 45	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea: Asistente de Instalación de Módulos	
Descripción: El responsable de la tarea deberá implementar una interfaz en el cual un usuario autorizado, podrá registrar los datos de los módulos que albergan los controladores semafóricos que serán conectados a la plataforma.	

Fuente: Elaboración Propia

Tabla 3.61 Tarea – Enviar Firmware

Numero: 46	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea: Enviar Firmware	
Descripción: El responsable de la tarea deberá implementar una interfaz en el cual un usuario autorizado, podrá enviar una actualización del firmware que será quemado en la memoria del controlador semafórico conectado a la plataforma.	

Fuente: Elaboración Propia

Tabla 3.62 Tarea – Historial de Eventos

Numero: 47	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea: Historial de Eventos	
Descripción: El responsable de la tarea deberá implementar una interfaz en el cual un usuario autorizado, podrá visualizar un reporte con los eventos realizados sobre la configuración de las intersecciones semafóricas entre dos fechas.	

Fuente: Elaboración Propia

Tabla 3.63 Tarea – Mejorar Geolocalización de Intersecciones

Numero: 48	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea: Mejorar Geolocalización de Intersecciones	
Descripción: El responsable de la tarea deberá mejorar el mapa de geolocalización de las intersecciones, con iconos más representativos y que muestren mayor información además de los estados de la fase principal.	

Fuente: Elaboración Propia

Tabla 3.64 Tarea – Mejorar Mapa de Geolocalización de Semáforos

Numero: 49	Responsable: Jose Carlos Huallpamaita
Nombre de Tarea: Mejorar Mapa de Geolocalización de Semáforos	
Descripción: El responsable de la tarea deberá mejorar el mapa en el que se configura la geolocalización de los semáforos con iconos más representativos y que muestren mayor información de cada fase.	

Fuente: Elaboración Propia

3.3.2 Fase de implementación

3.3.2.1 API/GraphQL

- **Tarea 32 – Consulta para Recuperar Datos de Dispositivos:**

  En la figura 3.79, se muestra la consulta para obtener la información de los dispositivos con los siguientes campos:

  id: Identificador único del dispositivo.
  name: Nombre del dispositivo.
  code: Código del dispositivo.
  img: Imagen representativa en base 64 del dispositivo.
  states: Colección de estados que conforman el dispositivo con los campos:
  id: Identificador único del estado.
  Key: clave característica el estado

```
query {
  devices {
    id
    name
    code
    img
    states {
      id
      key
      # thumbnail
    }
  }
}
```

Fig. 3.79 Consulta para recuperar Datos de Dispositivos

Fuente: Elaboración Propia

En la figura 3.80, se muestra el objeto JSON con los datos solicitados en la consulta de la figura 3.79.


```
{
  "data": {
    "devices": [
      {
        "code": "CV001",
        "id": "5a407347a4cee6281eec4853",
        "img": "media/devices/a5c4420e-46a9-4b71-90f4-dcec3dda9bb8.jpeg",
        "name": "Controlador Vehicular",
        "states": [
          {
            "id": "5a9498986cbf672f85775249",
            "key": "R"
          },
          {
            "id": "5a9498986cbf672f8577524a",
            "key": "G"
          },
          {
            "id": "5a9498986cbf672f8577524b",
            "key": "A"
          },
          {
            "id": "5a9498986cbf672f8577524c",
            "key": "D"
          },
          {
            "id": "5acb121e6d02103e7abf75c6",
            "key": "N"
          },
          {
            "id": "5ad098acbf3c9f0e17f65da0",
            "key": "R_M"
          },
          {
            "id": "5ad098acbf3c9f0e17f65da1",
            "key": "G_M"
          },
          {
            "id": "5ad098acbf3c9f0e17f65da2",
            "key": "A_M"
          },
          {
            "id": "5ad098acbf3c9f0e17f65da3",
            "key": "D_M"
          },
          {
            "id": "5ad098acbf3c9f0e17f65da4",
            "key": "N_M"
          },
          {
            "id": "5b04b0f03ecabe0f0dbcc20e",
            "key": "U"
          }
        ]
      }
    ]
  }
}
```


Fig. 3.80 Resultado de la Consulta para Recuperar Datos de Dispositivos

Fuente: Elaboración Propia

- **Tarea 28 – Consulta para Recuperar Datos de Firmware:**

En la figura 3.81, se muestra la consulta para obtener la información de los las versiones de firmware con los siguientes campos:

id: Identificador único del firmware.

app: Nombre del aplicativo.

versión: Versión del archivo.

description: descripción de la versión del firmware.

date: fecha de lanzamiento

bin: Binario del firmware en base 64.

```
query {
  firmwares {
    id
    app
    version
    description
    date
    #bin
  }
}
```

Fig. 3.81Consulta para Recuperar Datos de Firmware

Fuente: Elaboración Propia

En la figura 3.82, se muestra el objeto JSON con los datos solicitados en la consulta de la figura 3.81.


```
{
  "data": {
    "firmwares": [
      {
        "app": "sigecs",
        "date": "",
        "description": "release stable 25-05-18",
        "id": "5b16175a554d820f98eee5b5",
        "version": "1.0.13"
      },
      {
        "app": "sigecs",
        "date": "",
        "description": "test i2c bus",
        "id": "5b314384c2b34e27eec74a79",
        "version": "1.0.33"
      },
      {
        "app": "sigecs",
        "date": "",
        "description": "firmware release pre mundial",
        "id": "5b220a6e9594b30013509 1e6",
        "version": "1.0.31"
      },
      {
        "app": "sigecs",
        "date": "",
        "description": "release i2c ok",
        "id": "5b345792a14a51001373060a",
        "version": "1.0.35"
      },
      {
        "app": "sigecs",
        "date": "",
        "description": "Beta Test",
        "id": "5b515164cf9c28001336733a",
        "version": "1.0.38"
      },
      {
        "app": "sigecs",
        "date": "",
        "description": "test sigecs droid",
        "id": "5b59f133f3e8ea0013b52e1e",
        "version": "1.0.40"
      }
    ]
  }
}
```


Fig. 3.82 Resultado de la Consulta para Recuperar Datos de Firmware

Fuente: Elaboración Propia

- **Tarea 30 – Mutaciones para Modificar Datos de Dispositivos:**

En la figura 3.83, se muestra la consulta para agregar la información de un nuevo dispositivo, con los siguientes campos como parámetros de entrada:

name: nombre del dispositivo.
code: Código del dispositivo.
img: Imagen representativa del dispositivo en base 64.

Fig. 3.83 Mutación para Agregar Nuevo Dispositivo

Fuente: Elaboración Propia

En la figura 3.84, se muestra la consulta para actualizar la información de un dispositivo, con los siguientes campos como parámetros de entrada:

id: Identificador único del dispositivo a actualizar.
name: Nombre del dispositivo.
code: Código del dispositivo.
img: Imagen representativa del dispositivo en base 64.

Fig. 3.84 Mutación para Actualizar Dispositivo

Fuente: Elaboración Propia

En la figura 3.85, se muestra la consulta para eliminar la información de un dispositivo, con los siguientes campos como parámetros de entrada:

id: identificador único del dispositivo a eliminar.

Fig. 3.85 Mutación para Eliminar Dispositivo

Fuente: Elaboración Propia

- **Tarea 31 – Mutaciones para Modificar Datos de Firmware:**

En la figura 3.86, se muestra la consulta para agregar la información de un nuevo firmware, con los siguientes campos como parámetros de entrada:

app: Nombre del aplicativo.
versión: Versión del archivo.
description: descripción de la versión del firmware.
date: fecha de lanzamiento
bin: Binario del firmware en base 64.

Fig. 3.86 Mutación para Agregar Nuevo Firmware

Fuente: Elaboración Propia

En la figura 3.87, se muestra la consulta para agregar la información de un nuevo firmware, con los siguientes campos como parámetros de entrada:

id: Identificador único del firmware
app: Nombre del aplicativo.
versión: Versión del archivo.
description: descripción de la versión del firmware.
date: fecha de lanzamiento
bin: Binario del firmware en base 64.

Fig. 3.87 Mutación para Actualizar Firmware

Fuente: Elaboración Propia

En la figura 3.88, se muestra la consulta para eliminar la información de un firmware, con los siguientes campos como parámetros de entrada:

id: identificador único del firmware a eliminar.

Fig. 3.88 Mutación para Eliminar Firmware

Fuente: Elaboración Propia

- **Tarea 37 – Consultas para Recuperar Historial de Eventos:**

  En la figura 3.89, se muestra la consulta para recuperar la información del historial de eventos, con los siguientes campos:

  id: Identificador único del evento.
  date: Fecha del evento.
  info: Información sobre el evento.
  name: Nombre del evento.
  user: Usuario que realiza el evento con los siguientes campos:
    id: identificador único del usuario.
    email: dirección de correo del usuario.

```
query {
  historyEvents {
    id
    date
    # info
    name
    user {
      id
      email
    }
  }
}
```

```
{
  "data": {
    "historyEvents": [
      {
        "date": "2018-10-19T15:39:00-05:00",
        "id": "5bca40e4797cde3e30879a53",
        "name": "sincronizacion de corredor",
        "user": {
          "email": "sigecs@gmail.com",
          "id": "5a3ed9f12cle492f45cd2848"
        }
      },
      {
        "date": "2018-10-19T16:22:39-05:00",
        "id": "5bca4b1f797cde4620a1ad70",
        "name": "configuracion de controlador",
        "user": {
          "email": "sigecs@gmail.com",
          "id": "5a3ed9f12cle492f45cd2848"
        }
      },
```

Fig. 3.89 Consulta para Recuperar Historial de Eventos

Fuente: Elaboración Propia

3.3.2.2 Cliente web

- **Tarea 41 – Mantenimiento de Dispositivos**

  Se creó un componente con sus respectivas clases “.ts”, plantillas “.html” y hojas de estilo “.css”, el primer componente de la figura siguiente, muestra un formulario donde se lista todos los dispositivos registrados en el sistema, con sus respectivas acciones de actualización, eliminación y creación para nuevos dispositivos, también se muestra otro sub componente que muestra las configuraciones que se puedan realizar a cada dispositivo. Los datos mostrados en el formulario se obtienen realizando las consultas implementadas en la tarea N°32.

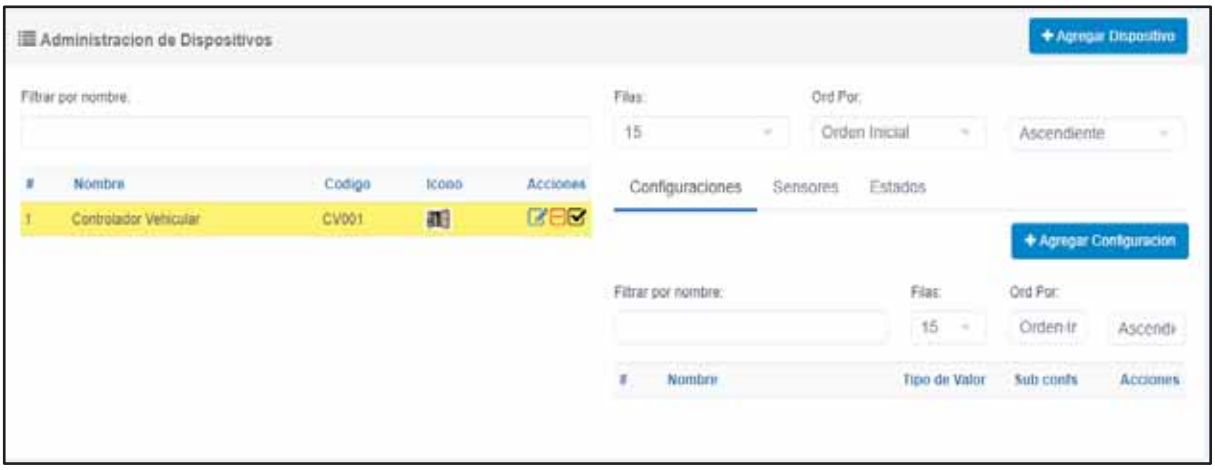


Fig. 3.90 Formulario Administración de Dispositivos

Fuente: Elaboración Propia

Los formularios mostrados en las figuras 3.91 y 3.92, muestran los componentes para realizar las operaciones de creación y actualización de vistas respectivamente, los datos registrados en estos formularios son enviados y almacenados en la base de datos realizando las consultas implementadas en la tarea N°36

Fig. 3.91 Formulario Agregar Dispositivo

Fuente: Elaboración Propia

Fig. 3.92 Formulario Actualizar Dispositivo

Fuente: Elaboración Propia

- **Tarea 39 – Mantenimiento de Estados**

  Se creó componentes con sus respectivas clases ".ts", plantillas ".html" y hojas de estilo ".css". El formulario de la figura siguiente muestra una lista con los estados que puede tomar una determinada fase, también se muestra los controles que permitirán realizar las operaciones de actualización, eliminación y agregación de un nuevo estado. Los datos mostrados en el formulario se obtienen realizando las consultas implementadas en la tarea N°33.

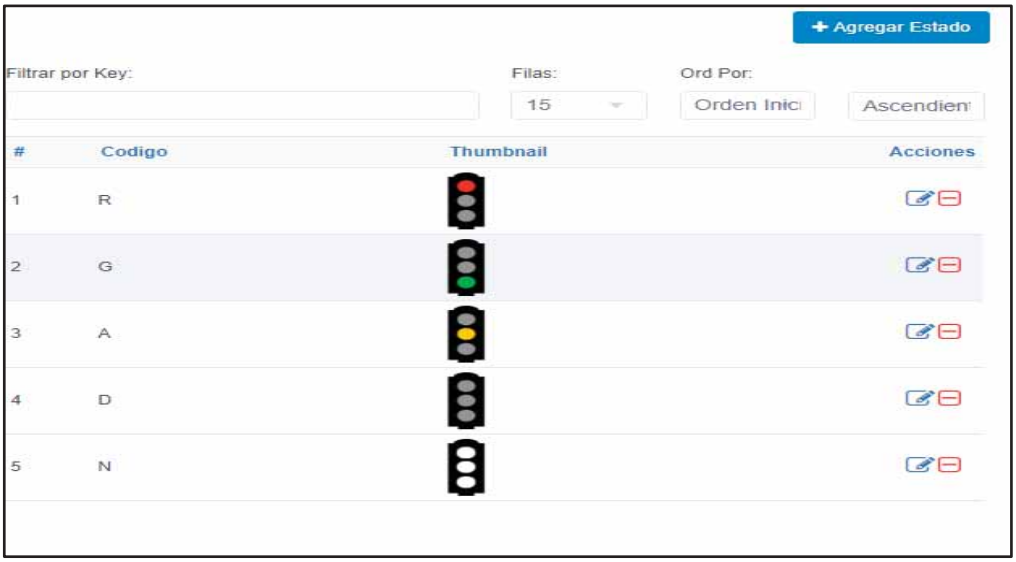


Fig. 3.93 Formulario Mantenimiento Estados

Fuente: Elaboración Propia

- **Tarea 41– Mantenimiento de Configuraciones**

Se creó componentes con sus respectivas clases ".ts", plantillas ".html" y hojas de estilo ".css". El formulario de la figura siguiente muestra una lista con las configuraciones para un determinado sensor, también se muestra los controles que permitirán realizar las operaciones de actualización, eliminación y agregación de una nueva configuración. Los datos mostrados en el formulario se obtienen realizando las consultas implementadas en la tarea N°34.

La ejecución del software tiene como fundamento estudios estadísticos previos, donde es necesaria la realización de un estudio cuantitativo de contabilización vehicular, de acuerdo a afluencia de vehículos, el tiempo que demoran a la espera de cada cambio y el tiempo que toma que todos los vehículos que estuvieron esperando en luz roja pasen el cruce vial.

Estos indicadores darán cabida a determinar el número de segundos que el semáforo deberá marcar en rojo, ámbar y verde, para así evitar la congestión vehicular y volver eficaz los tiempos de paso de cada vehículo.

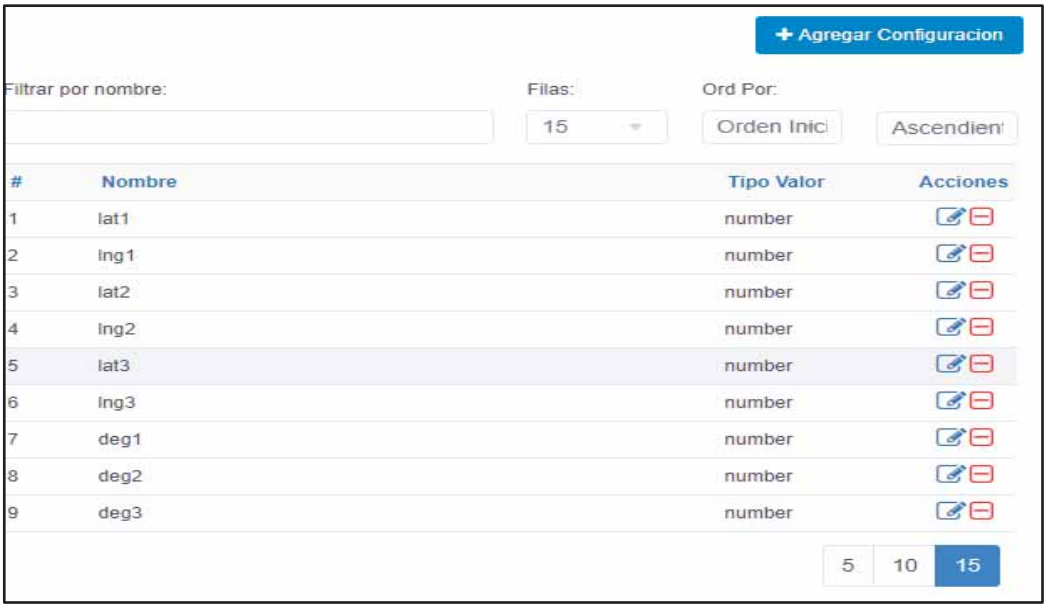


Fig. 3.94 Formulario Mantenimiento de Configuraciones

Fuente: Elaboración Propia

- **Tarea 42 – Mantenimiento de Firmware**

  Se creó un componente con sus respectivas clases ".ts", plantillas ".html" y hojas de estilo ".css", el primer componente de la figura siguiente, muestra un formulario donde se lista todas las versiones de firmware disponibles en la plataforma, con sus respectivas acciones de actualización, eliminación y creación para nuevas versiones. Los datos mostrados en el formulario se obtienen realizando las consultas implementadas en la tarea N° 35.

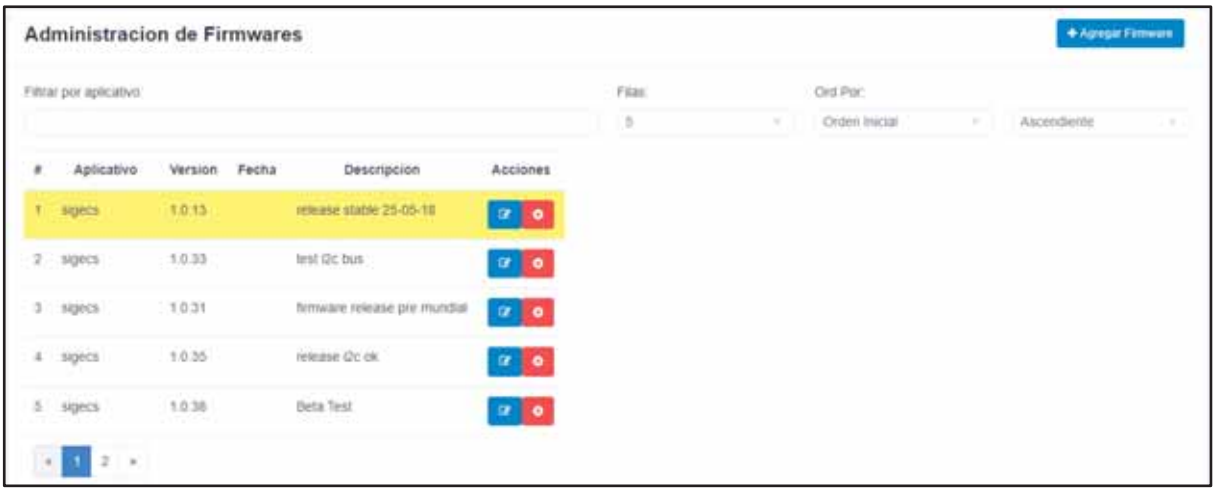


Fig. 3.95 Formulario Mantenimiento de Firmwares

Fuente: Elaboración Propia

Los formularios mostrados en la primera y segunda figura, muestran los componentes para realizar las operaciones de creación y actualización de firmware respectivamente, los datos registrados en estos formularios son enviados bajo una petición con los parámetros "*aplicativo, fecha, versión, descripción, binario*" y almacenados en la base de datos utilizando las consultas implementadas en la tarea N°39.

Fig. 3.96 Formulario Agregar Firmware

Fuente: Elaboración Propia

Fig. 3.97 Formulario Actualizar Firmware

Fuente: Elaboración Propia

- **Tarea 46 – Enviar Firmware**

  Se creó dos componentes con sus respectivas clases ".ts", plantillas ".html" y hojas de estilo ".css". El formulario mostrado en la figura siguiente, permite el grabado del firmware con la configuración inicial de la intersección seleccionada mediante el puerto USB del controlador semafórico siempre y cuando esté conectada a una estación de trabajo[15] con el programa de grabado de firmware instalado.

[15] PC-Escritorio, Laptop, etc.

Fig. 3.98 Mapa de Geolocalización de Semáforos

Fuente: Elaboración Propia

Por otro lado, el formulario Mostrado en la siguiente figura, permite el envío y grabado del firmware hacia la intersección seleccionada, mediante la red en la que se está trabajando.

Fig. 3.99 Mapa de Geolocalización de Semáforos

Fuente: Elaboración Propia

- **Tarea 47 – Historial de Eventos**

Se creó componentes con sus respectivas clases “.ts”, plantillas “.html” y hojas de estilo “.css”. El formulario de la figura siguiente muestra una tabla con los eventos realizados en una fecha seleccionada y entre dos espacios de tiempo, también se puede apreciar cual es el usuario que realizo la acción y la descripción de dichas acciones.

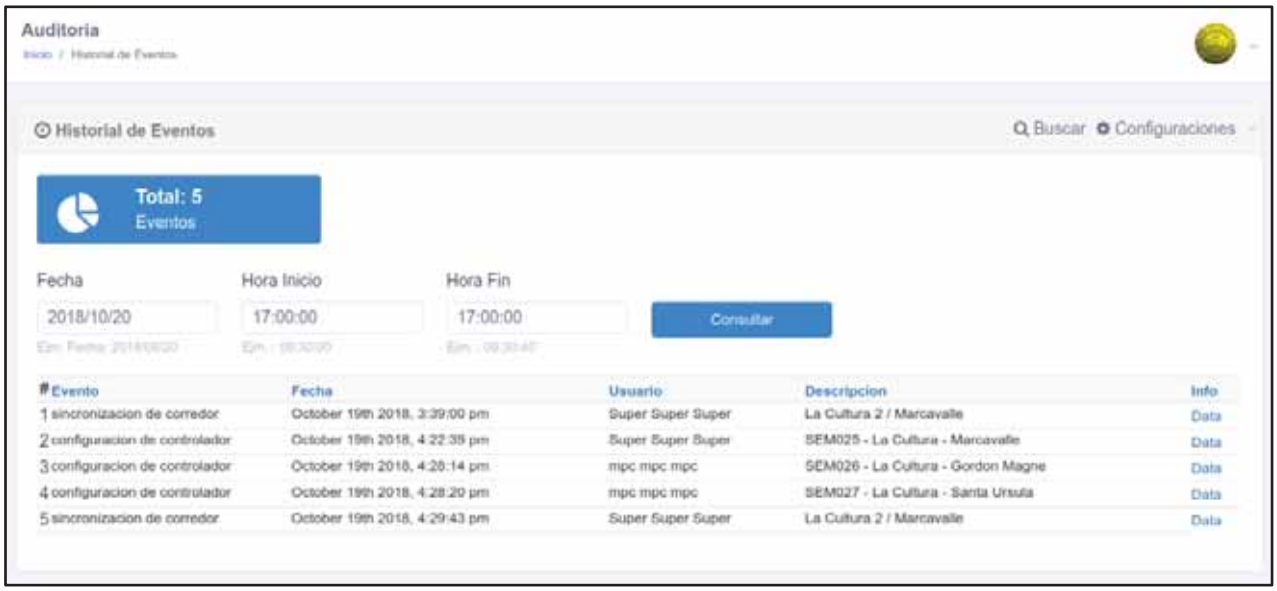


Fig. 3.100 Historial de Eventos

Fuente: Elaboración Propia

En la parte derecha de la tabla, se muestra un enlace que lanzara una ventana en forma modal con el detalle de la información del evento, tal como se muestra en la figura 3.102:

Fig. 3.101 Detalle de Historial de Eventos

Fuente: Elaboración Propia

- **Tarea 48 – Mejorar Geolocalización de Intersecciones**

Se actualizo la presentación de los iconos que representan el estado de la fase principal; cambiando los marker de colores por otros de tipo viñeta; en el que el borde representa el estado de la fase principal y la barra debajo del número o código del semáforo, representa el tipo de configuración que se está ejecutando: configuración normal o configuración especial, tal como se muestra la figura 3.103

Fig. 3.102 Actualización de los Puntos de Intersecciones

Fuente: Elaboración Propia

- **Tarea 49 – Mejorar Mapa de Geolocalización de Semáforos**

Fig. 3.103 Mejora de Geolocalización de Semáforos

Fuente: Elaboración Propia

3.3.3 Pruebas y resultados

Otra de las características que es necesario resaltar en la gestión de controladores semafóricos, es la posibilidad de poder actualizar el código fuente "*firmware*" que corre en cada controlador. Las figuras 3.104 y 3.105 muestran una interfaz en el cual es permite enviar remotamente y en tiempo real el código binario que ejecuta el procesador del controlador semafórico, eliminando así el esfuerzo de ir y actualizar manualmente hasta el punto de la intersección.

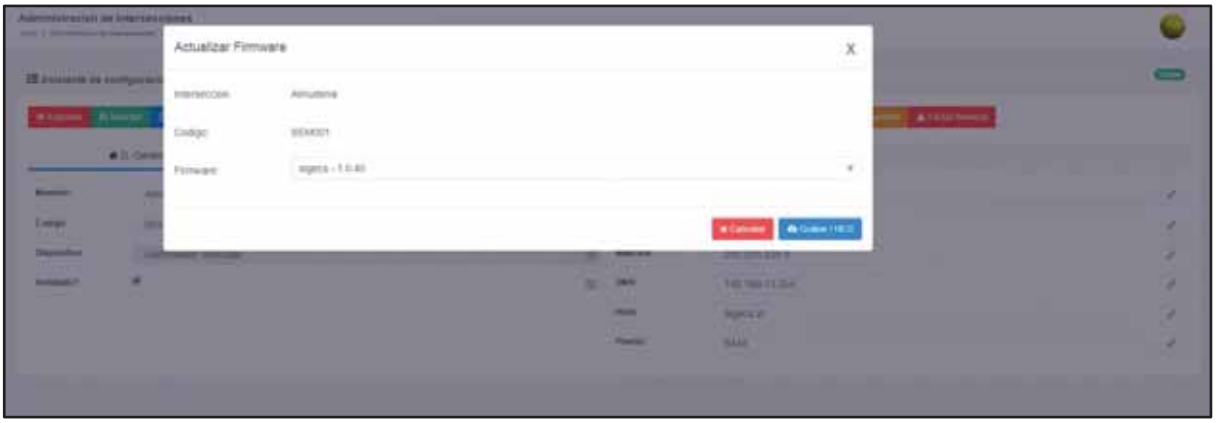


Fig. 3.104 Actualización de Firmware en tiempo real.

Fuente: Elaboración Propia

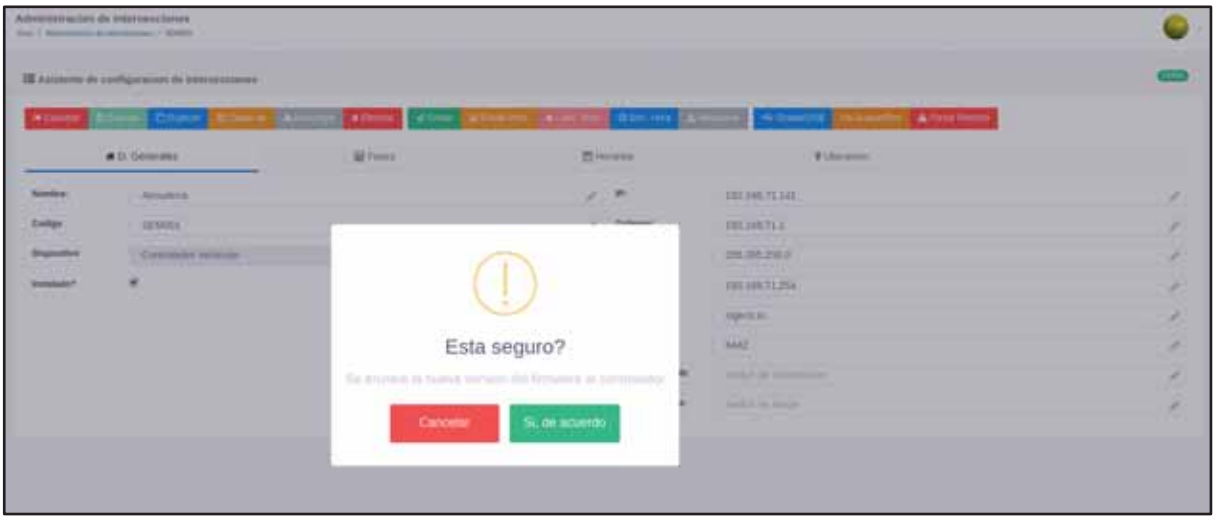


Fig. 3.105 Dialogo de Confirmación para la Actualización de Firmware.

Fuente: Elaboración Propia

Como resultado de toda la información recolectada desde los controladores se puede realizar reportes que muestren el comportamiento tanto de cada controlador como de la plataforma en general, todo esto para hacer una validación y sobretodo tomar decisiones en base a información real. Las figuras 3.106 y 3.107, muestran un reporte del historial de ejecución de las fases vehiculares para una fecha y entre un intervalo de tiempo, en este caso se muestra el resultado de consultar el historial para la fecha "*26-08-2018*" entre las horas "*16:55:48 a 16:57:28*"

Fig. 3.106 Panel Selección Intervalo de Fechas

Fuente: Elaboración Propia

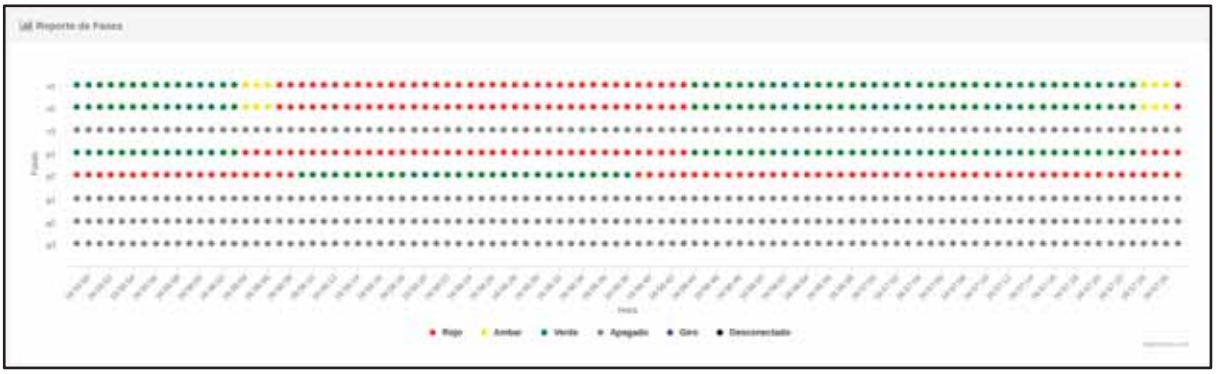


Fig. 3.107 Historial de Ejecución de Fases Vehiculares Entre dos Fechas.

Fuente: Elaboración Propia

Otro de los resultados que es posible obtener de la plataforma es el historial con el porcentaje de intersecciones conectadas en intervalos de tiempo tal como muestra el grafico de tortas en la figura 3.108.

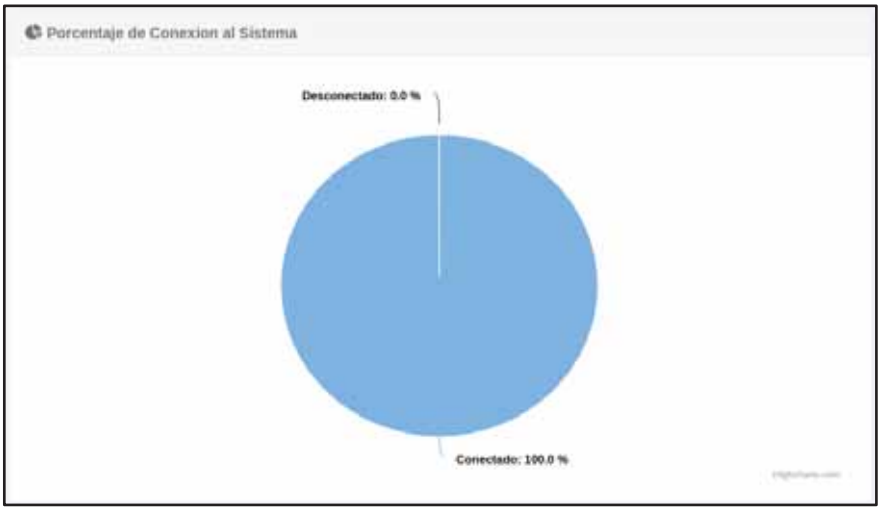


Fig. 3.108 Historial de Conexión al Sistema Entre dos Fechas.

Fuente: Elaboración Propia

También se puede obtener un historial de la cantidad de memoria RAM libre de los controladores semafóricos entre dos fechas, dicha información ayudara a diagnosticar posibles errores en la versión del firmware en ejecución. Véase la figura 3.109.

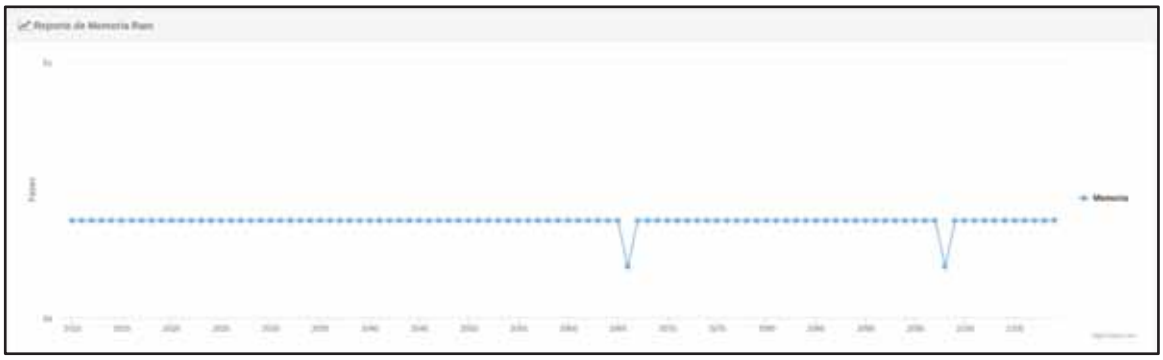


Fig. 3.109 Historial de Memoria RAM Libre en un Controlador.

Fuente: Elaboración Propia

Otra de las mejoras o actualizaciones que se hizo para esta iteración, es el rediseño del mapa de geolocalización de intersecciones, priorizando una mejor representación del estado de la fase principal, con más información que indique una representación más real.

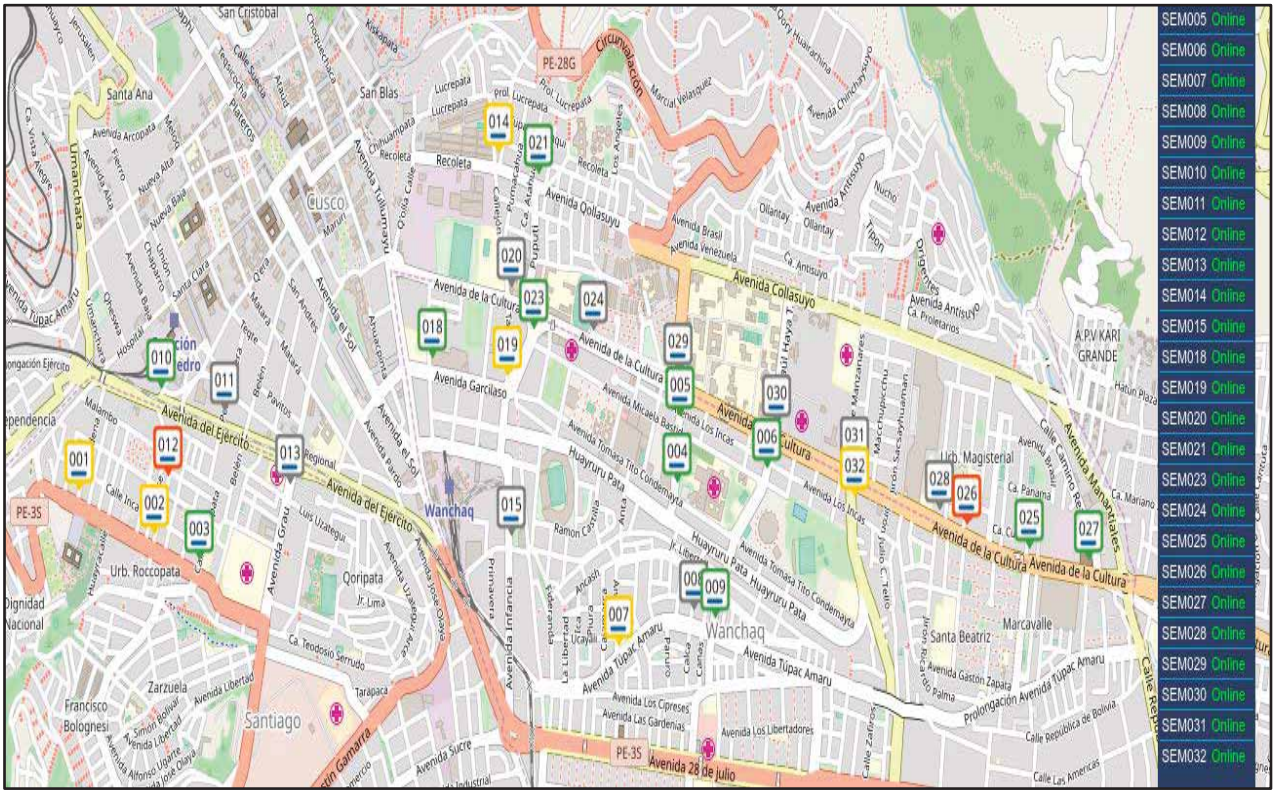


Fig. 3.110 Mapa de Geolocalización Mejorada.

Fuente: Elaboración Propia

CONCLUSIONES

- Es posible aplicar los conceptos del Internet de las Cosas (IoT), para el diseño e implementación de plataformas que se encarguen de la gestión de controladores semafóricos instalados y conectados a una red WAN.

- Se diseñó una arquitectura que cumple los requerimientos que demanda el Internet de las Cosas: ser distribuido, escalable, flexible, seguro y robusto.

- Los datos obtenidos en aplicaciones relacionadas al Internet de las Cosas son complejas y necesitan ser accedidos por múltiples usuarios, es en ese contexto que el diseño e implementación de bases de datos de tipo no relacional resultan una opción eficaz para este tipo de aplicaciones.

- Para sistemas en los cuales importante la interoperabilidad, el diseñar e implementar una interfaz de aplicaciones “API”, representa un método ideal y este se repotencia más si se implementa una capa de consultas con GraphQL; complementando la esencia de ser escalable y flexible, haciendo que la plataforma conserve su integridad ante futuras actualizaciones.

- Para proyectos relacionados con Internet de las Cosas, Mosca-MQTT, resulta una herramienta ideal para este tipo de soluciones, siendo MQTT el protocolo por defecto para una eficaz implementación de un servidor de red “Broker MQTT”; por su ligereza, poco consumo de memoria y batería.

RECOMENDACIONES

- Dada la gran cantidad de datos se recomienda la aplicación de técnicas de Big Data dentro de los grandes volúmenes de información provenientes de los controladores semafóricos para descubrir patrones que sean útiles en la mejora del tránsito vehicular.

- Es posible utilizar este proyecto como base para que no sólo se pueda gestionar controladores semafóricos, sino también cualquier otro tipo de dispositivos, sensores o actuadores que cumplan con las especificaciones del IoT.

- En base a la tesis desarrollada, se recomienda la adición de nuevas funcionalidades aplicando conceptos de inteligencia artificial para el comportamiento autónomo de los controladores semafóricos.

BIBLIOGRAFIA

Aleksovska, Taneva, & Ganchey. (2015). *Traffic Lights Control Via Web Application.* Sofia,Bulgaria: Department of Control Systems Technical University.

Balderas Contreras , T. (2011). *Desarrollo Agil.* México. Obtenido de INAODE.

Barrios, C. D. (2017). Arquitectura de Microservcios . *Tecnologia, Investigación y Academia*, 37.

Basalo, A., Angel Alvarez, M., Hurtado, P., & Cerdá, X. (2014). *Desarrolloweb.com.* Obtenido de Desarrolloweb.com: https://programacion.net/files/code/20161021121055_manualdeangularjs-manualcompleto.pdf

Boletin-18-INEI. (2009). *Estimaciones y Proyecciones de Poblacion por Sexo, Segun Departamento, Provincia y Distrito 200-2015.* Lima,Peru: INEI.

Brea, S. V. (2018). *Internet de las Cosas.Horizonte 2050.* Madrid: Instituto Español de Estudios Estrategicos.

Camarena, J., Contreras, L., Moreno, K., Rodriguez, M., & Salazar, C. (2018). *Aplicación del IoT para el control de congetión vehivular*. Chiriquí: III Congreso Internacional de Ciencia y Tecnológia para el Desarrollo Sostenible.

Cisco. (2013). *Connections Counter: The Internet of Everything in Motion*. Obtenido de The Network:https://newsroom.cisco.com/featurecontent?type=webcontent&articleId=1208342

Cisco. (2017). *Internet of Things*. Obtenido de Cisco: http://www.cisco.com/web/solutions/trends/iot/overview.html

Claranet . (08 de 09 de 2012). *Claranet* . Obtenido de Claranet : https://www.claranet.es/about/news/que-tipos-de-servidores-hay.html

DIRCETUR,CUSCO. (2015). *Boletin Estadistico de Turismo.* Cusco,Peru: DIRCETUR.

Dizdarević, J., Carpio, F., Jukan, A., & Masip-Bruin, X. (2019). Una encuesta de protocolos de comunicación para internet de las cosas y los desafíos relacionados con la niebla y la computación en la nube integración. *Encuestas de computación ACM*.

Dodson, S. (October de 2003). *The internet of things*. Obtenido de The Guardian: https://www.theguardian.com/technology/2003/oct/09/shopping.newmedia

Erl-Thomas. (2005). *ERL, Thomas. Service-Oriented Architecture.*

EvaluandoSoftware.com. (6 de Febrero de 2018). *EvaluandoSoftware.com*. Obtenido de EvaluandoSoftware.com: https://www.evaluandosoftware.com/desarrollo-software-agil/

Fernández, R. (3 de 2 de 2014). *Genbeta.* Obtenido de Genbeta: https://www.genbeta.com/desarrollo/mongodb-que-es-como-funciona-y-cuando-podemos-usarlo-o-no

Fundación Pais Digital. (2018). *CÓMO EMPRENDER EN INTERNET DE LAS COSAS:CONCEPTOS PRÁCTICOS.* Santiago de Chile.

Garcia Muñoz , C. (2016). *Informática IES Gonzalo Nazareto.* Obtenido de Sharding in MongoDB: http://informatica.gonzalonazareno.org/proyectos/2016-17/Cluster_Sharding_MongoDB_Carlos_Garcia.pdf

Gartner. (2014). *Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020*. Obtenido de Gartner: https://www.gartner.com/newsroom/id/2636073

Gershenfeld. (2015). *What Is the "Internet of Things"?* Obtenido de Scientific American: https://www.scientificamerican.com/article/what-is-the-internet-of-things/

Google-Analytics. (2018). *Google Trends*. Obtenido de https://trends.google.com: https://trends.google.com/trends/explore?date=all&q=nginx,apache

Gracia, O. (2014). Aplicacion de tiempo real con node.js . *Northware*.

Guía SBOK™. (2013). *Una guía para el Conocimiento de Scrum.* Arizona.

Hernandez Sampieri, R. (2014). *Metodologia d ela investigacion* . Mexico: Mc Graw Hill Education.

Hernández, L. d. (2017). *Arduino y los dispositivos del IoT. Recuperado de* . Obtenido de Programar Facil: https://programarfacil.com/podcast/61-arduino- y-los-dispositivos-del-iot.

Iacono, & Godoy. (2012). *Estudio de la Integracion entre WSN y Redes TCP/IP*. Argentina: Facultad de Ingenieria de Mendoza.

IBM. (2018). *Conceptos de TLS*. Obtenido de IBM®: https://www.ibm.com/support/knowledgecenter/es/SSFKSJ_9.0.0/com.ibm.mq.sec.doc/q009920_.htm

Kozlowski, P., & Darwin, B. (2013). *Mastering Web Application Development with AngularJS.* Packt Publishing, Limited.

La Republica. (09 de Mayo de 2016). Parque automotor de la ciudad de Cusco aumentó en 328% los últimos 10 años. *Diario la Republica*, pág. 1.

Lenguaje de programación Go y sus característica. (25 de 4 de 2018). Obtenido de Keepcofing.io: 2018

Ley-N°27972. (2003). *Ley Organica de Municipalidades.*

López , D., & Maya, E. (2017). *Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web.* Imbabura.

Lowe, R. (12 de Julio de 2014). *TICbeat*. Obtenido de https://www.ticbeat.com/tecnologias/que-es-una-api-para-que-sirve/

Mendez. (16 de 12 de 2009). *Sistema de Comunicaciones Basado en Ethernet para el Control de Sistemas Empotrados.* Mexico: Universidad Tecnologica de Mixteca. Obtenido de Yumpu: http://jupiter.utm.mx/~tesis_dig/10992.pdf

Mora. (2017). *Diseño e Implementación de una Aplicación IoT en la Nube de Azure para el Análisis de Imagen.* Madrid.España: E.T.S Ingenieria de Sistemas Informaticos - UPM.

Mozzilla.org. (2018). *Que es un servidor WEB?* Obtenido de MDN web docs: https://developer.mozilla.org/es/docs/Learn/Common_questions/Que_es_un_servidor_WEB

MQTT ORG. (2017). *Frequently Asked Questions*. Obtenido de Mqtt.org: http://mqtt.org/faq

MQTT un Protocolo Especifico para el Internet de las Cosas. (2015). Obtenido de Digital Dimension Solutions: http://www.digitaldimension.solutions/es/blog-es/opinion-de-expertos/2015/02/mqtt-un-protocolo-especifico-para-el-internet-de-las-cosas/

ngx-mqtt. (2018). *Ngx-mqtt Home Page*. Obtenido de sclausen.github.io: https://sclausen.github.io/ngx-mqtt/

NoticiasCusco. (08 de Julio de 2017). *Noticias Cusco*. Obtenido de http://noticias.ruedadenegocios.com.pe: http://noticias.ruedadenegocios.com.pe/municipalidad-de-cusco-trabaja-en-mejora-de-la-semaforizacion-de-la-ciudad/

Phan, M. (2015). *Web Application Programming Interface Design for a Customer Portal.* Aalto University School of Science.

Requena, C. (2016). *Arquitectura Orienda a Microservicios*. Obtenido de cjrequena.github: https://cjrequena.github.io/micro-services/2016/09/20/micro-services-architecture-es.html

Research, A. (2013). *More Than 30 Billion Devices Will Wirelessly Connect to the Internet of Everything in 2020.* Obtenido de Abi Research: https://www.abiresearch.com/press/more-than-30-billion-devices-will-wirelessly-conne/

Roldan, & Ccapatinta. (2015). *Implementación de un Algoritmo para el Control de Tráfico Vehicular y Peatonal Mediante el Procesamiento de Imágenes.* Cusco,Peru: Escuela Profesional de Ingeniería Informatica y de Sistemas - UNSAAC.

Rose, K., Eldridge, S., & Chapin, L. (2015). *La Internet de las Cosas-Una breve Reseña.* Internet Society.

Rose, K., S. E., & L. C. (2015). *La internet de las cosas - Una breve reseña* .

Salcedo, J. L. (2014). *Sistemas de Tiempo Real.*

Sampieri, H. (2014). *Metodologia de la Investigacion.* Mexico: McGraw Hill.

Santiago Silvestre , J. S. (2016). *Internet de las cosas.* Erasmus.

Schwaber, & Sutherland. (2014). *The Scrum Guide.* Scrum.org and Scruminc.

Secretaría de Movilidad de Medellín. (2013). Obtenido de https://www.medellin.gov.co/movilidad/documents/seccion_senalizacion/cap7_semaforos.pdf

Tipos de Servidores. (s.f.). Obtenido de Claranet.es: https://www.claranet.es/about/news/que-tipos-de-servidores-hay.html

Turnbull, J. (2014). *the Docker Book.*

Urteaga Pecharromán, A. (2015). *Aplicación de la metodología de.* Madrid.

W3Teach. (31 de 10 de 2018). *Usage of Web Servers Broken Down by Ranking*. Obtenido de Web Technology Survey: https://w3techs.com/technologies/cross/web_server/ranking

Wilhelm Harder, D., Zarnett, J., Montaghami, V., & Giannikouris, A. (2014). *Una introducción práctica a los sistemas en tiempo real para la ingenería de pregrado.*

Wolf, E. (2017). *Microservices.* United States: Person Education.

ANEXOS

A. Instalación de Fibra Óptica En La Ciudad Del Cusco

Fig. 7.1 Distribución de la Red de Fibra Óptica en la Ciudad del Cusco

Fuente: Municipalidad Provincial del Cusco

Fig. 7.2 Instalación de Fibra Óptica

Fuente: Municipalidad Provincial del Cusco

B. Adaptador para la Conexión a una Red Mediante TCP/IP

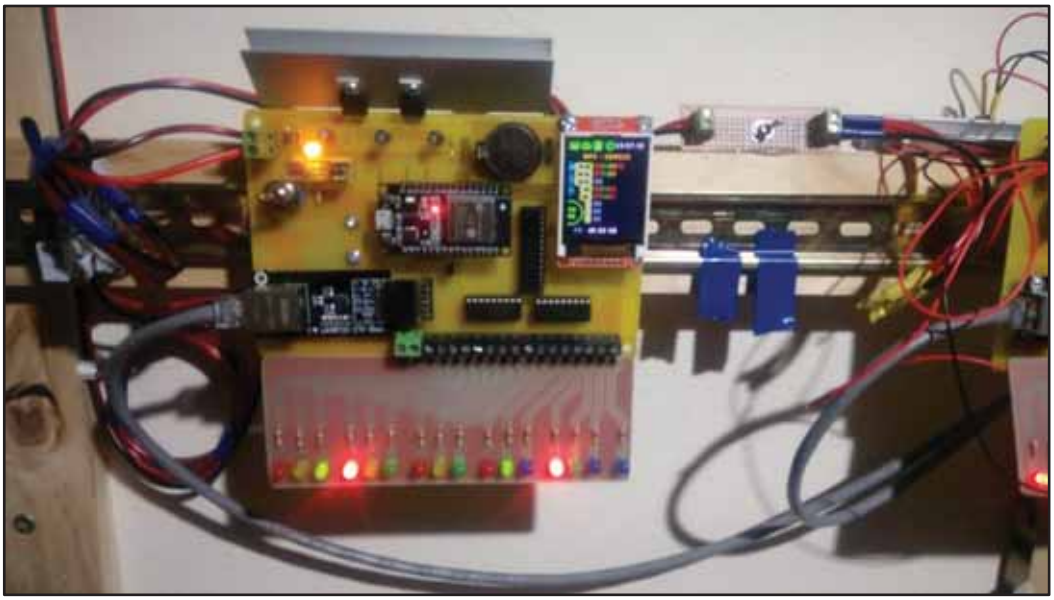


Fig. 7.3 Adaptador de Red para los Controladores Semafóricos

Fuente: Municipalidad Provincial del Cusco

Fig. 7.4 Instalación de los Adaptadores de Red en el Gabinete

Fuente: Municipalidad Provincial del Cusco

C. Descripción y Estructura de la Arquitectura de la Base de Datos para la Plataforma

Colección: ScheduleType		
Campo	Tipo	Descripción
cycles	Int	Numero de ciclos
direction	String	Dirección del Horario, subida o bajada.
end	String	Hora de inicio del horario en formato HH:mm:ss
start	String	Hora fin del horario en formato HH:mm:ss

Colección: TypeSchedule		
Campo	Tipo	Descripción
Name	String	Nombre del Tipo de Horario.
Schedules	[ScheduleType]	Lista de Horarios del Tipo de Horario.

Colección: Runner		
Campo	Tipo	Descripción
day1	String	Tipo de configuración del dia lunes.
day2	String	Tipo de configuración del dia martes.
day3	String	Tipo de configuración del dia miercoles.
day4	String	Tipo de configuración del dia jueves.
day5	String	Tipo de configuración del dia viernes.
day6	String	Tipo de configuración del dia sabado.
day7	String	Tipo de configuración del dia domingo.
Id	String	Identificador unico.
intersections	[Intersection]	Lista de intersecciones de un corredor.
name	String	Nombre del corredor.
types	[TypeSchedule]	Lista de tipos de horario.

Colección: Intersection		
Campo	Tipo	Descripción
cameras	[Camera]	Lista de cámaras de la Intersección.
code	String	Código de la intersección.
day1	String	Tipo de configuración del dia lunes.
day2	String	Tipo de configuración del dia martes.
day3	String	Tipo de configuración del dia miercoles.
day4	String	Tipo de configuración del dia jueves.
day5	String	Tipo de configuración del dia viernes.
day6	String	Tipo de configuración del dia sabado.
day7	String	Tipo de configuración del dia domingo.
ethSetting	ETHSetting	Configuración de red de la intersección.
id	String	Identificador unico.
installed	Boolean	Flag que indica si la intersección fue instalada.
lat	Float	Latitud de la ubicación.
lng	Float	Longitud de la ubicación.
name	String	Nombre de la intersección.
phaseG1	PhaseSetting	Configuración de la phase de Giro 1.
phaseG2	PhaseSetting	Configuración de la phase de Giro 2.
phaseG3	PhaseSetting	Configuración de la phase de Giro 3.
phaseP1	PhaseSetting	Configuración de la phase de Peatonal 1.
phaseP2	PhaseSetting	Configuración de la phase de Peatonal 2.
phaseV1	PhaseSetting	Configuración de la phase de Vehicular 1.
phaseV2	PhaseSetting	Configuración de la phase de Vehicular 2.
phaseV3	PhaseSetting	Configuración de la phase de Vehicular 3.
type_1_x	Schedule	Configuración de horario “x” del tipo 1.
type_2_x	Schedule	Configuración de horario “x” del tipo 2.

type_3_x	Schedule	Configuración de horario “x” del tipo 3.
type_4_x	Schedule	Configuración de horario “x” del tipo 4.
type_5_x	Schedule	Configuración de horario “x” del tipo 5.

Colección: Camera		
Campo	Tipo	Descripción
address	String	Dirección IP de la cámara.
deg	Int	Grados de inclinación del icono de la cámara.
description	String	Descripción de la cámara.
id	String	Identificador unico.
lat	Float	Latitud de la ubicación.
lng	Float	Longitud de la ubicación.
traffic	Boolean	Flag que indica si la cámara es de tráfico.

Colección: ETHSetting		
Campo	Tipo	Descripción
address	String	Direccion IP.
dns	String	Nombre de dominio.
gateway	String	Dirección IP de Gateway.
host	String	Dominio de sistema.
netmask	String	Mascara de red.
port	String	Puerto de sistema.
switchD	String	Switch de Distribucion.
switchE	String	Switch de Borde.

Colección: PhaseSetting		
Campo	Tipo	Descripción
deg1	Float	Grados de inclinación (1) del icono de la fase.
deg2	Float	Grados de inclinación (2) del icono de la fase.
deg3	Float	Grados de inclinación (3) del icono de la fase.
enable	Boolean	Flag para indicar si la fase está activa.
lat1	Float	Latitud (1) del icono de la fase.
lat2	Float	Latitud (2) del icono de la fase.
lat3	Float	Latitud (3) del icono de la fase.
lng1	Float	Longitud (1) del icono de la fase.
lng2	Float	Longitud (2) del icono de la fase.
lng3	Float	Longitud (3) del icono de la fase.
name	String	Nombre de la fase.

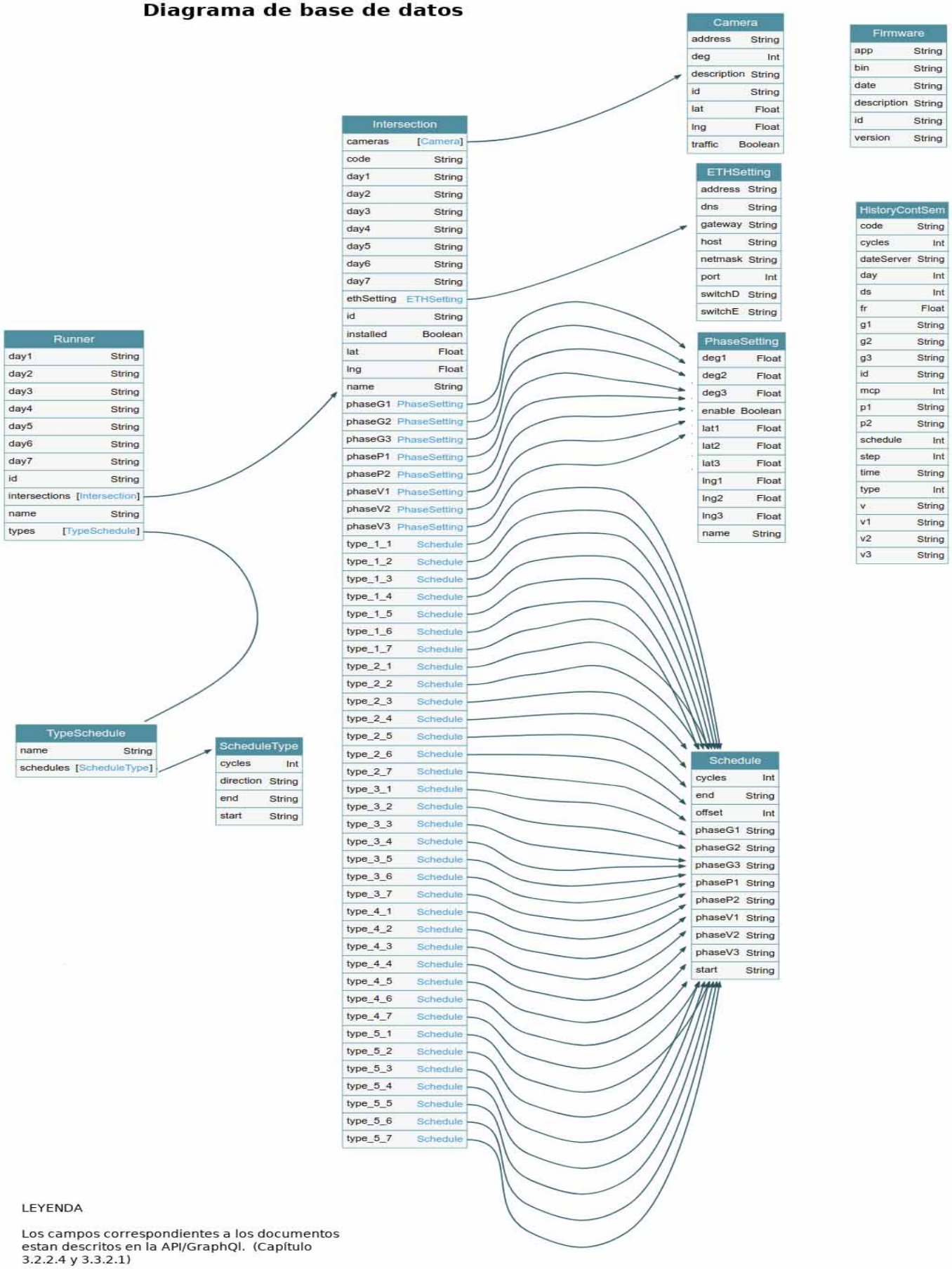
Colección: Schedule		
Campo	Tipo	Descripción
cycles	Int	Número de ciclos del horario.
end	String	Hora fin del horario en formato HH:mm:ss
offset	Int	Segundos de diferencia para la sincronización.
phaseG1	String	Configuración de la fase de giro 1.
phaseG2	String	Configuración de la fase de giro 2.
phaseG3	String	Configuración de la fase de giro 3.
phaseP1	String	Configuración de la fase peatonal 1.
phaseP2	String	Configuración de la fase peatonal 2.
phaseV1	String	Configuración de la fase vehicular 1.
phaseV2	String	Configuración de la fase vehicular 2.
phaseV3	String	Configuración de la fase vehicular 3.

start	String	Hora de inicio del horario en formato HH:mm:ss

Colección: HistoryContSem		
Campo	Tipo	Descripción
code	String	Código de la intersección.
cycles	Int	Numero de ciclos.
dateServer	String	Fecha de servidor.
day	Int	Día de la semana.
ds	Int	Sensor de Hora activo.
fr	Float	Memoria ram libre.
g1	String	Fase de giro 1.
g2	String	Fase de giro 2.
g3	String	Fase de giro 3.
id	String	Identificador único.
mcp	Int	Sensor expansor activo.
p1	String	Fase peatonal 1.
p2	String	Fase peatonal 2.
schedule	Int	Horario activo.
step	Int	Paso activo.
time	String	Hora de la intersección.
type	Int	Tipo activo.
v	String	Versión de firmware.
v1	String	Fase vehicular 1.
v2	String	Fase vehicular 2.
v3	String	Fase vehicular 3.

Colección: Firmware		
Campo	Tipo	Descripción
app	String	Nombre de aplicativo de firmware.
bin	String	Binario en formato base 64.
date	String	Fecha de subida de firmware.
description	String	Descripción de firmware.
id	String	Identificador único.
version	String	Versión de firmware.

Diagrama de base de datos



LEYENDA

Los campos correspondientes a los documentos estan descritos en la API/GraphQl. (Capítulo 3.2.2.4 y 3.3.2.1)