UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA:

INGENIERÍA DE SISTEMAS

Trabajo de titulación previo a la obtención del título:

INGENIEROS DE SISTEMAS

TEMA:

“DISEÑO Y DESARROLLO DE UNA PLATAFORMA PROTOTIPO PARA VISUALIZACIÓN, CONTROL Y COMUNICACIÓN CON RASPBERRY PI EN AMBIENTES IOT”

AUTORES:

FRANKLIN FABRICIO CHICAIZA COLA

WILMER JOEL RIVERA ULLOA

TUTOR:

MANUEL RAFAEL JAYA DUCHE

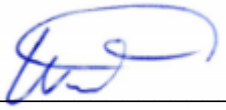
Quito, agosto del 2021

CESIÓN DE DERECHOS DE AUTOR

Nosotros Franklin Fabricio Chicaiza Cola con documento de identificación N° 1723178891 y Wilmer Joel Rivera Ulloa con documento de identificación Nº 2100526736, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación intitulado: "DISEÑO Y DESARROLLO DE UNA PLATAFORMA PROTOTIPO PARA VISUALIZACIÓN, CONTROL Y COMUNICACIÓN CON RASPBERRY PI EN AMBIENTES IOT", mismo que ha sido desarrollado para optar por el título de: INGENIEROS DE SISTEMAS, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Quito, agosto del 2021



Franklin Fabricio Chicaiza Cola

CI: 1723178891

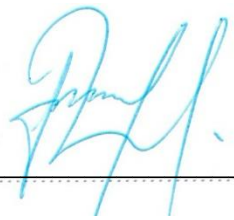
Wilmer Joel Rivera Ulloa

CI: 2100526736

DECLARATORIA DE COAUTORÍA DEL DOCENTE TUTOR/A

Yo declaro que bajo mi dirección y asesoría fue desarrollado el Proyecto Técnico, "DISEÑO Y DESARROLLO DE UNA PLATAFORMA PROTOTIPO PARA VISUALIZACIÓN, CONTROL Y COMUNICACIÓN CON RASPBERRY PI EN AMBIENTES IOT" realizado por Franklin Fabricio Chicaiza Cola y Wilmer Joel Rivera Ulloa, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerado como trabajo final de titulación.

Quito, agosto del 2021



MANUEL RAFAEL JAYA DUCHE

CI: 1710631035

DEDICATORIA

El presente trabajo de titulación esta principalmente dedicado a Dios por darme fuerzas y por brindarme el conocimiento necesario. A mis padres María y Jorge por inculcarme principios y valores, que con amor, esfuerzo, trabajo y confianza permitieron que logre culminar mis estudios universitarios. A mis hermanas Nancy, Silvana, Lizbeth y hermanos Jorge, Edgar, Paul que con sus palabras de apoyo me dieron fuerzas para no rendirme y poder cumplir con mis metas trazadas.

Franklin Chicaiza

A mis padres Coralia y Calixto quienes, con su amor, paciencia y esfuerzo, han sido mi apoyo a lo largo de toda la carrera universitaria y a lo largo de mi vida, me han permitido cumplir un sueño más, gracias por inculcarme el esfuerzo y valentía. A mi familia, por sus consejos y palabras de aliento que de una u otra forma me acompañan en todos mis sueños y metas. Gracias

Wilmer Rivera

AGRADECIMIENTO

A nuestros padres por ser los pilares fundamentales durante el proceso de formación.

A los docentes por compartir sus conocimientos.

A nuestro tutor por ser nuestro mentor.

Franklin Chicaiza y Wilmer Rivera

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente proyecto técnico tiene como objetivo diseñar y desarrollar una plataforma prototipo para visualización, control y comunicación con Raspberry Pi en ambientes IoT (Internet de las Cosas).

Para desarrollar la plataforma se utilizó la metodología Modelo en V o también conocido como Modelo de cuatro niveles, se utilizó el framework Nuxt.js que es basado en Vue.js para el fron-end y Node.js para back-end, el patrón de diseño con el que se trabajo es MVVM (Modelo-Vista-Vista Modelo) ya que es el mismo que utiliza el framework.

La plataforma está diseñada para enviar y recibir datos en forma de mensajes, tiene como cliente la Raspberry Pi. El cliente recibe señales digitales de los dispositivos (sensores, actuadores y controladores) a través de las GPIO (entrada / salida de propósito general). El usuario genera un script que cumple con funciones específicas para MQTT, como la conexión, suscripción y la publicación, en la que se indica el número de GPIO que envía o recibe la señal. Es la encargada de enviar mensajes a la plataforma. MQTT es un protocolo liviano y flexible que comunica al broker a través de mensajes. El broker EMQX gestiona los datos del cliente, recibe los datos publicados y decide quienes deben recibirlos, hace posible la comunicación entre el hardware y software en tiempo real.

ABSTRACT

The objective of this technical project is to design and develop a prototype platform for visualization, control and communication with Raspberry Pi in IoT (Internet of Things) environments.

To develop the platform, the V Model methodology was used or also known as the Four-Level Model, the Nuxt.js framework was used, which is based on Vue.js for the front-end and Node.js for the back-end, the pattern The design I work with is MVVM (Model-View-View Model) since it is the same one used by the framework.

The platform is designed to send and receive data in the form of messages, its client is the Raspberry Pi. The client receives digital signals from the devices (sensors, actuators and controllers) through the GPIOs (general purpose input / output). The user generates a script that fulfills specific functions for MQTT, such as connection, subscription and publication, indicating the number of GPIOs that send or receive the signal. It is in charge of sending messages to the platform. MQTT is a lightweight and flexible protocol that communicates to the broker through messages. The EMQX broker manages customer data, receives published data and decides who should receive it, makes communication between hardware and software possible in real time.

INTRODUCCIÓN

Antecedentes

El uso de sistemas basados en IoT está aumentando en la medida en que se requiere un sistema de seguimiento y adquisición de datos en tiempo real, como ciudad inteligente o tele diagnóstico en áreas médicas. Por lo tanto, se requiere un protocolo de comunicación apropiado para resolver estos problemas. Hoffman, Heimes y Senel (2018) plantean el uso del protocolo MQTT para construir una aplicación de adquisición de datos del sensor de temperatura y humedad con una interfaz móvil, que es Android y basada en la web. El resultado de la prueba indica que el protocolo MQTT tiene la capacidad de transferir datos más rápido que el protocolo HTTP. El uso de MQTT puede ser una opción para la aplicación en tiempo real de adquisición de datos de hardware basada en Internet de las Cosas

Problema

En la actualidad el Internet de las Cosas representa aún un campo desconocido donde hay mucho que investigar, en la cual, las plataformas son clave para el desarrollo de soluciones software y hardware para la interconexión de personas y dispositivos (Cárdenas, Secmotic, 2016). Existe una amplia gama de plataformas: ThingsBoard, Kaaiot, Thinger, OpenRemote; Algunas de estas tienen una cuenta gratuita temporal y otras una cuenta premium que permiten características de administración, siendo este un limitante para las personas o empresas que necesitan visualizar, controlar y comunicarse con los dispositivos que están conectados a un Raspberry Pi, el problema radica en que para administrar los dispositivos se debe pagar una mensualidad, muchas veces necesitamos la plataforma únicamente para hacer pruebas y no sería rentable pagar una mensualidad por cada dispositivo que se

necesite probar. Las plataformas actuales permiten administrar placas que están directamente conectadas, dejando a un lado el control de los dispositivos (sensores, actuadores y controladores) que se conecten a estos. Para este caso no permiten activar o desactivar cuando ocurre un evento, por tanto, es indispensable tener el control de estos dispositivos.

Justificación

Una plataforma web es una parte integral de cualquier producto de IoT, por ende, puede ayudar a acelerar su tiempo de comercialización, minimizar el riesgo, reducir el costo de desarrollo y ayudar a lograr un ajuste entre el producto y el mercado más rápido. El mercado de las plataformas IoT está en continua expansión, siendo esta la base para la interconexión de dispositivos, permitiendo el intercambio de información en tiempo real para optimizar los procesos, por lo tanto, el presente proyecto se enfocará en el desarrollo de una plataforma web para enviar y recibir señales digitales de los dispositivos que están conectados a un Raspberry Pi a través del protocolo MQTT y el broker EMQX para la interacción entre hardware y software.

Objetivos

General

Diseñar y desarrollar una plataforma prototipo para visualización, control y comunicación con Raspberry Pi en ambientes IoT.

Específicos

Investigar los protocolos de comunicación IoT existentes.

Desarrollar una plataforma web que reciba y envíe datos de los dispositivos (sensores, actuadores y controladores).

Crear roles y asignar permisos a usuarios que se conecten a la plataforma.

Administrar los pines de entrada/salida del Raspberry Pi a través de la plataforma permitiendo la visualización del estado de dicho sensor en tiempo real a través de Dashboard.

Realizar pruebas con dispositivos que envíen datos para determinar el correcto funcionamiento de la plataforma.

Metodología

Para el desarrollo del presente proyecto se utilizó el Modelo en V del ciclo de vida conocido como Modelo de cuatro niveles ya que divide el trabajo en niveles individuales los que a continuación se detalla:

Nivel 1: Especificaciones

La plataforma permite la comunicación con los dispositivos conectados a las GPIO de la Raspberry Pi, para establecer la comunicación, el usuario se registra para conectar su dispositivo. Se puede elegir entre tres opciones de widgest: botón, switch on off y un gráfico para mostrar datos. Para crear una plantilla, dependiendo lo que el usuario necesita. Agrega un dispositivo con la plantilla creada, generando las credenciales para la comunicación con el Raspberry Pi. En el Raspberry Pi se crea un script en python donde se especifica las credenciales del dispositivo y el número de GPIO que va a recibir o enviar la señal digital, ya sea para una publicación o suscripción en la plataforma, consiste en clasificar los mensajes según los temas o tópicos que son gestionados por el broker EMQX. Se procede a ejecutar el script y automáticamente se establece la comunicación.

Nivel 2: Análisis funcional

- ✓ Los usuarios podrán registrarse rellenando un sencillo formulario con los campos: Correo, contraseña y nombre de usuario.
- ✓ Los usuarios se autenticarán con el correo y la contraseña introducidos en el registro.
- ✓ El usuario podrá solicitar restablecimiento de contraseña al sistema
- ✓ Los usuarios podrán eliminar las plantillas (widgest: botón, switch on off, gráfico) y dispositivos creados.
- ✓ Los usuarios no registrados no podrán acceder a la plataforma.
- ✓ Los usuarios podrán visualizar únicamente sus dispositivos.

- ✓ El administrador podrá listar todos los usuarios de la plataforma.
- ✓ El administrador podrá activar y desactivar los usuarios.

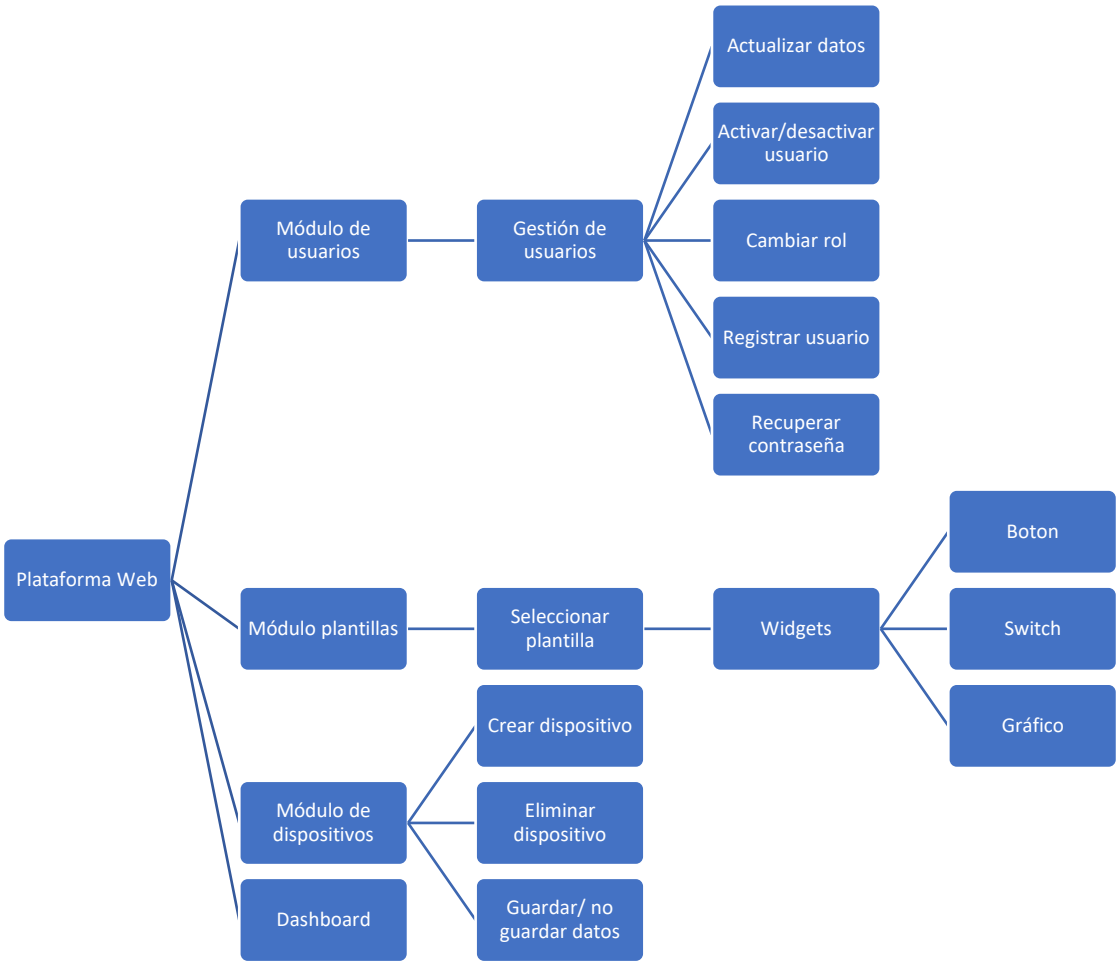
Nivel 3: Arquitectura del sistema

El desarrollo de esta plataforma web se divide en cuatro módulos: usuarios, plantillas, dispositivos y dashboard, los mismos que se describen en el nivel 4. Para el diseño se divide en dos partes: frontend (cliente) y backend (servidor). En el lado del cliente se trabajará con el framework Nuxtjs, está basado en Vuejs, su función es crear aplicaciones universales con escalabilidad por default. Para el lado del servidor utilizaremos nodejs, es una tecnología que se apoya en javascript, la característica más importante es la de no ser bloqueante, es decir que no detiene el hilo de ejecución del programa, esperando que las partes que necesitan más tiempo terminen su ejecución sin detener todo el programa (Contributors, 2021). En los siguientes apartados se detallan las tecnologías utilizadas para el desarrollo de dicha plataforma.

Nivel 4: Módulos del programa

En la Figura 1. Se muestran los módulos y tareas para el desarrollo de la plataforma web:

Figura 1. *Módulos de la plataforma web*



Elaborado por: Los autores.

Módulo de usuarios

- ✓ Tiene como función la gestión de usuarios
- ✓ Cambiar rol: Usuario_invitado y Usuario_premiun
- ✓ Para registrar un usuario nuevo, el rol por defecto será: Usuario_invitado

Módulo de plantillas

- ✓ Crea y elimina plantillas
- ✓ Se puede seleccionar entre tres opciones de widget:

- ✓ Boton: tiene una acción
- ✓ Switch: tiene dos acciones: on y off
- ✓ Gráfico: permite mostrar los datos que envían los sensores, en tiempo real y de forma gráfica.

Módulo de dispositivos

- ✓ Crea, elimina y genera credenciales para la comunicación con el cliente MQTT (Raspberry Pi).
- ✓ Si en la plantilla se elige el widget de gráfico, puede activar o desactivar el guardado de datos
- ✓ Muestra los datos guardados.

Módulo de Dashboard

- ✓ Muestra los datos que envía o recibe el Raspberry Pi.

CAPITULO I

MARCO TEÓRICO

1.1. Plataforma IoT

Las plataformas de IoT son una tecnología multicapa para la gestión y automatización de sencillos dispositivos que están conectados a la red, básicamente conecta el software con el hardware, proporcionando funciones listas para usar son la base fundamental en la construcción de sistemas IoT, proporcionando herramientas y centralizando la información. De esta manera facilita la comunicación, la gestión de dispositivos y la administración de aplicaciones. Un sistema completo de IoT necesita un software responsable de analizar los datos que fueron recopilados por los sensores, el verdadero valor de IoT se activa desde el momento en que se integra con los sistemas comerciales, con todos los diferentes tipos de hardware y las diferentes opciones de comunicación para comprender las necesidades del cliente y facilitar la creación de productos nuevos productos. Las empresas inteligentes necesitan soluciones inteligentes, evitando el proceso costoso de contratar desarrolladores de software para construir todo desde cero, Las plataformas IoT son las encargadas de ayudar a resolver este problema de manera rápida y rentable (McClelland, 2020).

1.2. IoT

La evolución de Internet ha permitido la interconexión entre personas a través de aplicaciones en dispositivos inteligentes siendo capaz de obtener información. Está centrado en combinar datos con personas, procesos y objetos. IoT (Internet of Things) se basa en redes de telecomunicación, sensores y en una inteligencia que administra todo el proceso, visto de otra manera se dice que IoT es una interconexión de dispositivos o cosas que están conectada a internet que generan e intercambian datos. Los sensores al detectar vibraciones en el

entorno generan impulsos eléctricos para comunicarse, siendo los encargados de proveer información para la toma de decisiones. Para el desarrollo de un proyecto IoT lo primero que se debe tomar en cuenta es el protocolo IoT de los dispositivos, en la actualidad el internet de las cosas está presente en casi todos los campos, cada vez más empresas optan por esta tecnología, ya que permite convertir los datos en acciones (Hernandez, Mazon, & Escudero, 2018)

1.3. Protocolos de comunicación IoT

Los protocolos de comunicación son sistemas de reglas que utilizan formatos definidos para el intercambio de mensajes, los protocolos de comunicación deben estar acordados por las dos partes. IoT permite el uso de varios protocolos de comunicación, la utilización de un determinado protocolo depende de varios factores, para este proyecto se seleccionó a MQTT y CoAP, los mismos que se detallan más adelante. A continuación, se muestra Figura 2, comparativa con los protocolos más populares:

Figura 2. Comparativa de los principales protocolos para IoT

	Transporte	Modelo	Ámbito de aplicación	Conocimiento del contenido	Datos principales	Seguridad	Prioridad de los datos	Tolerancia a fallos
AMQP	TCP/IP	Intercambio de mensajes punto a punto	D2D D2C C2C	Ninguno	Codificados	TLS	Ninguno	Específica de la implementación
CoAP	UDP/IP	Petición/Respuesta (REST)	D2D	Ninguno	Codificados	DTLS	Ninguno	Descentralizado
DDS	UDP/IP (unicast + mcast) TCP/IP	Publicación/Suscripción Petición/Respuesta	D2D D2C C2C	Enrutamiento basado en el contenido, consultas	Declarados codificados	TLS, DTLS, DDS	Prioridades de transporte	Descentralizado
MQTT	TCP/IP	Publicación/Suscripción	D2C	Ninguno	No definidos	TLS	Ninguno	El nodo central (broker) es el punto único de fallo (SPoF)

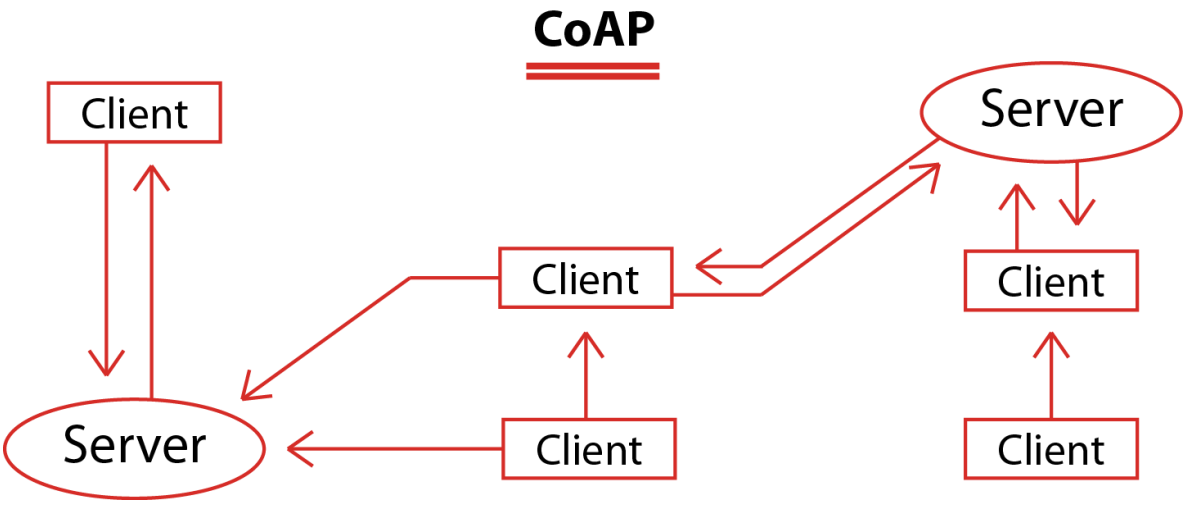
Nota. Características de los principales protocolos utilizados a nivel doméstico e industrial.

Fuente: (Sáez, 2019)

1.4. CoAP

CoAP (Constrained Application Protocol) es un protocolo de transferencia web para usarlo con nodos y redes de baja potencia que pueden comunicarse de forma interactiva a través de internet en general utilizando protocolos similares. Los servicios basados en este protocolo usan la conexión UDP, el intercambio de mensajes es asíncrono, modelo de interacción solicitud/respuesta. El modelo de mensajería intercambia mensajes entre puntos finales a través de UDP, usa un encabezado binario de 4 bytes, para evitar que los mensajes se dupliquen, cada mensaje tiene un id de 16 bits. Se adapta al formato de nodos-sensores, los controladores son de 8 bits con RAM y ROM limitada (Heredia, 2014).

Figura 3. *Arquitectura CoAP*



Nota. Modelo petición/respuesta para el envío y recepción de mensajes.

Fuente: (Pickdata, 2019).

1.5. Modelo Petición/respuesta

La petición y respuesta CoAP se basa en el intercambio de mensajes asíncronos entre el nodo cliente y el nodo servidor, la coordinación utiliza una técnica llamada Piggy-backing

que consiste en el mensaje de respuesta con la respuesta a la consulta realizada. Para enviar las peticiones se hace uso de los métodos GET, PUT, POST Y DELETE sobre un recurso, el cual está identificado por una ruta del paquete. Si el servidor no puede responder inmediatamente una solicitud, se responde con un mensaje vacío para que el cliente deje de retransmitir (Shelby, 2013).

Tabla 1. *Ventajas y Desventajas del protocolo CoAP*

Ventajas	**Desventajas**
Bajo consumo de energía	Topología uno a uno con conexiones directas
Fiabilidad en intercambio de mensajes a través de UDP.	Arquitectura cliente/servidor
Baja latencia	Problema con NAT
Baja saturación	Pocas librerías existentes
Permite implementaciones RESTful	
Tipo de mensaje Asíncrono & Síncrono	

Nota. *Esta tabla contiene los beneficios y limitaciones de CoAP como protocolo IoT.*

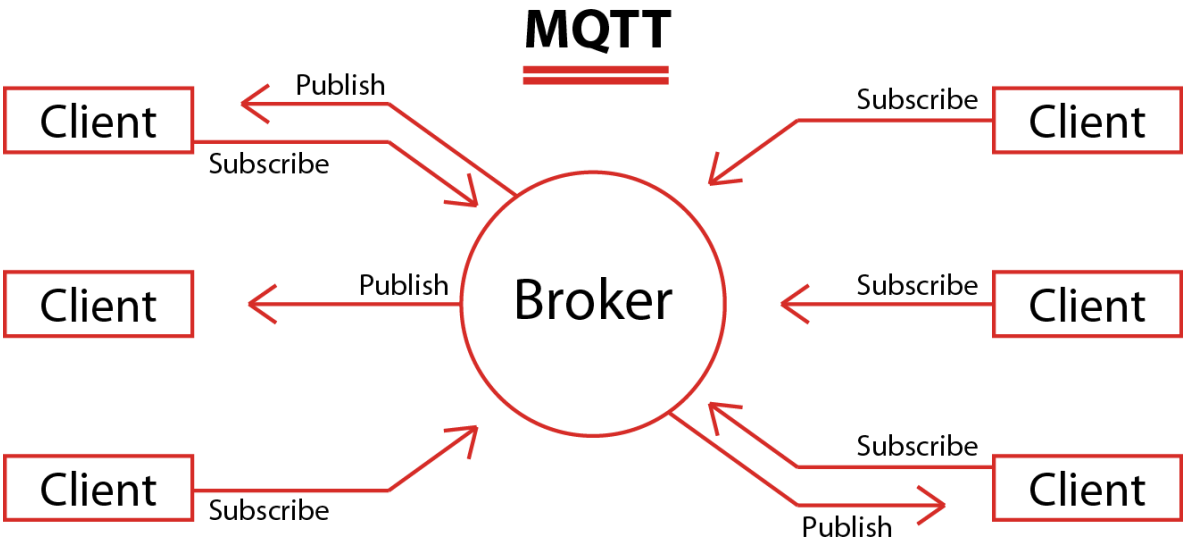
Fuente: (Cortés, 2016).

1.6. MQTT

MQTT (The Message Queuing Telemetry Transport) es un protocolo especial de máquina a máquina, de transporte de mensajería de publicación / suscripción de Cliente Servidor que se ejecuta sobre TCP/IP o sobre protocolos de red con soporte bidireccional, diseñado para conexiones donde el ancho de banda de la red es limitado con un bajo consumo

de energía, una de las características principales es el desacople tridimensional: espacio, tiempo, sincronización. Espacio, publicador y suscriptor no saben de la existencia el uno del otro. Tiempo, en algunos casos no es necesario que coincidan temporalmente. Sincronización, los clientes y broker no se bloquean mientras se publica o suscribe. Las acciones básicas de este protocolo son: conectarse al broker, publicar un mensaje, suscribirse a un topic, cancelar la suscripción de un determinado topic (Biswajeeban Mishra, 2020).

Figura 4. Arquitectura MQTT



Nota. La arquitectura MQTT está basada en el modelo publicación/suscripción a través de un Broker.

Fuente: (Pickdata, 2019).

Tabla 2. *Ventajas y Desventajas del protocolo MQTT*

Ventajas	**Desventajas**
Fiabilidad en intercambio de mensajes a través de TCP	No se puede modificar el formato del mensaje (metadata).
Mensajería asíncrona	Mayor consumo de energía
Arquitectura publicación/suscripción	
Comunicación: M:N	
Topic matching (jerárquico, si se suscribe a un topic, se suscribe a todos los hijos).	
3 QoS diferenciados	

Nota. *Esta tabla contiene los beneficios y limitaciones de MQTT como protocolo de IoT.*

Fuente: (Cortés, 2016).

1.7. Modelo Publicación/Suscripción de MQTT

Este modelo es el medio de comunicación y el intercambio de datos. Para ver las publicaciones, los clientes deben suscribirse a un topic del bróker ya que este reenvía el mensaje a los suscriptores. El topic puede usar "/" para definir los niveles a los que se va a suscribir (EMQX, EMQX, 2019).

1.8. Broker MQTT

Un intermediario MQTT, también conocido como servidor de mensajes MQTT es el encargado de recibir los mensajes de los clientes para luego enrutarlos, ya sea publicación o suscripción. Los principales broker/servidor de código abierto son: EMQX, Eclipse Mosquito, VerneMQ, HiveMQ CE (EMQX, EMQX, 2019).

1.9. Broker EMQX

EMQX es un intermediario de mensajes IoT MQTT de código abierto, diseñado para lograr una alta confiabilidad, admitir conexiones MQTT, realiza enrutamiento de mensajes entre los dispositivos de red física. EMQX broker proporciona un dashboard que permite ver la información básica del servidor sin necesidad de instalar ningún otro software, se puede acceder a través del navegador: http: // localhost: 18083, el usuario predeterminado es *admin* y la contraseña es *public* permitiendo ver la información básica de los clientes conectados. Para establecer la comunicación a través del protocolo MQTT es necesario la autenticación para evitar conexiones ilegales. EMQX admite varias formas de autenticación a través de complementos predefinidos que se pueden configurar desde el dashboard o archivos de configuración, siendo esta una gran ventaja en la fase de desarrollo (EMQX, Docs EMQX, 2020).

1.10. Cliente MQTT

Los clientes MQTT dependen del lenguaje de programación que se esté utilizando, para ser considerado como cliente debe tomar en cuenta las funciones básicas: conexión, publicación, suscripción y desuscripcion. Los clientes más populares son: MQTT C, MQTT Java, MQTT Go, MQTT Erlang, MQTT JavaScript, MQTT Python. Los clientes tienen un ciclo de vida que inicia con una conexión donde se especifica la información básica del broker, Se suscribe a un tema luego que la conexión se estableció correctamente, recibe el mensaje, publica un mensaje a un tema específico, da de baja la suscripción y se desconecta (EMQX, Docs EMQX, 2020).

1.11. Patrón Arquitectónico – MVVM

MVVM (Model/View/ViewModel) es una variación de MVC (Model/View/Controler) se adapta a las plataformas de desarrollo modernas, fue presentada al público en 2005 por John Gossman. MVVM facilita una separación del desarrollo de la interfaz gráfica de usuario con la ayuda de un lenguaje de marcado. Ofrece enlace de datos bidireccional entre vista y modelo de vista, la vista es el punto de partida de la aplicación y su lógica consta de tres componentes:

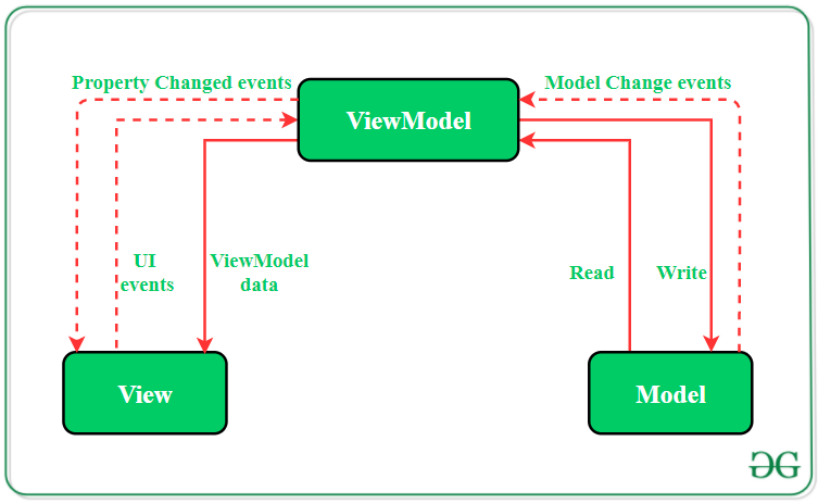
Vista: Muestra los datos que se reciben del controlador como resultado, interfaz de usuario.

Modelo: Almacena los datos y la lógica del negocio.

Vista-Modelo: Encapsula la lógica de la vista y los datos que serán representados.

(Aguirregomezcorta, 2020, pág. 14)

Figura 5. *Diagrama de los componentes del patrón MVVM*



Nota. MVVM se utiliza para abstraer la lógica de las acciones que se pueden realizar en su aplicación.

Fuente: (Mishra, 2020)

1.12. Base de Datos

1.12.1. MongoDB

Es un gestor de base de datos NoSQL de código abierto muy popular que almacena datos de documentos similares a objetos JavaScript Object Notation (JSON). El modelo de esquemas de documentos admite el desarrollo ágil para las aplicaciones modernas basadas en la web (Harrison & Harrison, 2021).

En la Tabla 3, se describen las ventajas y desventajas de la base de datos MongoDB.

Tabla 3. *Ventajas y desventajas de MongoDB*

Ventajas	**Desventajas**
Es un programa gratuito, multiplataforma.	No admite transacciones.
El rendimiento de las consultas es mucho más eficiente.	Realiza una serie de consultas para extraer datos de varias colecciones.
Base de datos flexible y escalable.	
No tiene un esquema predefinido, por lo que puede almacenar y leer los datos según sea necesario.	

Fuente: (Cookson, 2019)

1.12.2. MySQL

Es una base de datos relacional de código abierto, los datos se almacenan en tablas separadas relacionados entre sí. La estructura relacional permite ejecutar consultas SQL para recuperar o modificar información de la base de datos (Wang, 2018).

En la Tabla 4, se describen las ventajas y desventajas de la base de datos MySQL.

Tabla 4. *Ventajas y desventajas de MySQL*

Ventajas	**Desventajas**
Es un programa gratuito.	Tiene problema de rendimiento al escalar bases de datos muy grandes.
Flexible y fácil de usar.	Debe escribir el esquema para definir una tabla.
Es compatible con varios sistemas operativos.	

Fuente: (Noronha, 2018)

1.13. Node.js

Es un entorno que se encarga de ejecutar JavaScript de código abierto, utiliza el motor de JavaScript Versión 8 de Chrome, el modelo de entrada y salida con eventos asíncronos para evitar bloqueos en operaciones permitiendo que sea rápido y eficiente. Node.js instala por defecto el manejador de paquetes npm que permite utilizar libremente código abierto (Rajiv, 2018).

1.14. Vue.js

Es uno de los frameworks de JavaScript más populares diseñado para crear interfaces de usuario, su objetivo es la adaptabilidad de manera incremental. La biblioteca principal se centra en la capa de vista y es fácil de integrar en proyectos existentes (Rojas, 2020).

La aplicación de una sola página es un concepto nuevo, donde tola la aplicación frontal está a lado del cliente, significando que los navegadores juntan las bibliotecas JavaScript y la página HTML para crear la vista (Pelaez Lopez, 2020).

1.15. Nuxt.js

Es un frameworks de JavaScript basado en Vuejs para el desarrollo de aplicaciones web modernas que pueden ser implementadas en dos modos diferentes universal (SSR) o de una sola página aplicación (SPA) (Tiam kok, 2020).

1.16. Raspberry Pi

Es una minicomputadora de bajo costo diseñada para enseñar a programar a personas de bajos recursos. Tiene todo lo que necesita una computadora normal como puertos USB, Ethernet, HDMI, Wi-Fi pero carece de la capacidad de expansión de memoria y no puede conectar dispositivos integrados. Pi tiene una unidad microSD que puede usarse para ejecutar un sistema operativo, funciona con una fuente de alimentación de 5V (Bell, 2016).

La placa Raspberry tiene el tamaño de una tarjeta de crédito, en la actualidad se la utiliza para crear proyectos IoT. Pi tiene 40 pines de entrada y salida para uso general, estos permiten conectar sensores, actuadores o cualquier dispositivo electrónico. Raspberry tiene librerías preinstaladas, para acceder a los pines emplea el lenguaje de programación Python (Donat, 2018).

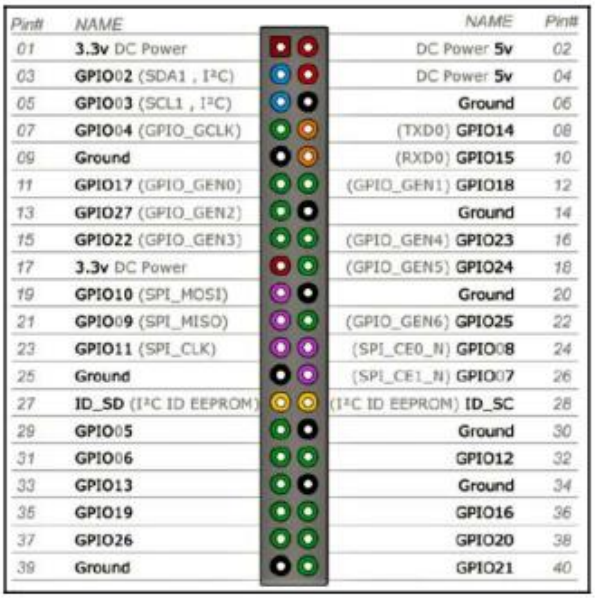
Raspbian es un sistema operativo de código abierto construido en Debian se utiliza oficialmente para las placas Raspberry, iniciado por Mike Thompson y Peter Green. Actualmente ofrece versiones de escritorio y lite (Kurniawan, 2018).

Python es un lenguaje de programación compatible con varios sistemas operativos, se usa ampliamente porque es fácil de comprender. Las reglas de sintaxis permiten crear aplicaciones personalizadas sin necesidad de escribir código adicional (Guillen, 2019).

El encabezado GPIO tiene dos esquemas board y BCM, por lo tanto, hay dos formas de referenciar a los pines en el código. Board numera los pines físicos secuencialmente y la numeración BCM hace referencia a los números del pin de Broadcom que puede variar según la versión del Raspberry Pi (Cicolani, 2021).

En la Figura 6. se muestra los pines de entrada y salida de uso general (GPIO).

Figura 6. *Pines GPIO E/S*



Pin#	NAME		NAME	Pin#
01	**3.3v** DC Power		DC Power **5v**	02
03	**GPIO02** (SDA1 , I²C)		DC Power **5v**	04
05	**GPIO03** (SCL1 , I²C)		**Ground**	06
07	**GPIO04** (GPIO_GCLK)		(TXD0) **GPIO14**	08
09	**Ground**		(RXD0) **GPIO15**	10
11	**GPIO17** (GPIO_GEN0)		(GPIO_GEN1) **GPIO18**	12
13	**GPIO27** (GPIO_GEN2)		**Ground**	14
15	**GPIO22** (GPIO_GEN3)		(GPIO_GEN4) **GPIO23**	16
17	**3.3v** DC Power		(GPIO_GEN5) **GPIO24**	18
19	**GPIO10** (SPI_MOSI)		**Ground**	20
21	**GPIO09** (SPI_MISO)		(GPIO_GEN6) **GPIO25**	22
23	**GPIO11** (SPI_CLK)		(SPI_CE0_N) **GPIO08**	24
25	**Ground**		(SPI_CE1_N) **GPIO07**	26
27	**ID_SD** (I²C ID EEPROM)		(I²C ID EEPROM) **ID_SC**	28
29	**GPIO05**		**Ground**	30
31	**GPIO06**		**GPIO12**	32
33	**GPIO13**		**Ground**	34
35	**GPIO19**		**GPIO16**	36
37	**GPIO26**		**GPIO20**	38
39	**Ground**		**GPIO21**	40

Nota. Se muestra los pines de entrada y salida de uso general (GPIO).

Fuente: (Rao, 2018, pág. 19)

En la Tabla 5, se describe los pines de entrada y salida de uso general (GPIO).

Tabla 5. *Entradas y salidas de las GPIO*

GPIO	**Descripción**
Pines 1 y 17	Proporciona alimentación de + 3,3 V CC.
Pines 2 y 4	Proporcionan alimentación de + 5 V CC.
Pines 6, 9, 14, 20, 25, 30, 34, 39	Asignados para tierra.
GPIO BCM de 0 al 27	Los pines pueden recibir y transmitir datos en formato digital

Fuente: (Rao, 2018, pág. 19)

1.17. UML

UML (Unified Modeling Language), es un lenguaje estándar que puede usarse con diferentes metodologías para representar gráficamente una idea, en base a sus componentes. Es un lenguaje visual que permite modelar procesos, sistemas y software. Ofrece una forma estándar para diagramar los planos de un software. UML es un modelado universal que permite diseñar las diferentes etapas del proyecto utilizando diagramas (Sommerville, 2011, pág. 119).

Diagramas de caso de uso, que permiten identificar el comportamiento de los diferentes actores que van a interactuar con el sistema.

Diagrama de secuencia, representa el intercambio de mensajes entre objetos, también se establecen las clases pertenecientes al sistema y las interacciones que existen entre los actores.

Diagramas de clases, modelan la estructura del sistema que se está diseñando donde se visualiza los objetos y las asociaciones entre estas clases.

1.18. Eclipse Paho Python

Es una biblioteca cliente para versiones 5.0, 3.1.1 y 3.1 de MQTT, desarrollada en el lenguaje Python, bajo el proyecto Eclipse Paho, hace posible la comunicación con el servidor (broker EMQX) a través del protocolo MQTT para publicar mensajes, suscribirse a temas y recibir mensajes publicados. Se instala en el Raspberry Pi utilizando el siguiente comando: pip install paho-mqtt (EMQ, 2020).

1.19. VPS

Un VPS (Servidor Virtual Privado) es un método de particionar un servidor físico en varias máquinas virtuales, asignando recursos exclusivos a cada partición, permite instalar cualquier sistema operativo, alojar sitios web y aplicaciones de software. Los VPS pueden ser administrados o no administrados, si gestiona un no administrado, debe ternes conocimientos avanzados sobre servidores; ya que el cliente debe hacerse responsable de la seguridad del servidor. Los precios varían dependiendo de las características que los clientes necesiten (Segarra, 2021).

En la Tabla 6, se muestran las características del VPS que se va utilizar para desarrollar este proyecto.

Tabla 6. *Características del servicio VPS para el desarrollo de la plataforma web*

Características	**Descripción**
Almacenamiento	30GB SSD
Transferencia mensual	1TB
Memoria RAM	2GB
Núcleos	1
Precio	$15 mensuales

Nota: *Descripción del servicio contratado a la empresa VUCAFLEX.*

Elaborado por: *Los Autores.*

CAPÍTULO II

ANÁLISIS TÉCNICO Y ECONÓMICO

2.1. Análisis técnico

En este apartado se realizarán matrices de priorización para definir el protocolo IoT y la base de datos, ya son los elementos más importantes para el desarrollo de la plataforma. La elección se realizará previo una investigación de las ventajas y desventajas, en la siguiente figura se muestran los puntajes para la evaluación.

Tabla 7. *Puntaje y calificación para la evaluación de los criterios*

Puntaje	**Calificación**
5	Excelente
4	Muy bueno
3	Bueno
2	Regular
1	Malo

Nota. *Los criterios se evalúan de acuerdo con la investigación realizada en el capítulo anterior.*

Elaborada por: *Los autores.*

2.2. Análisis del protocolo de comunicación IoT

En la Tabla 8, se establecen los criterios y pesos para elegir el protocolo de comunicación IoT que se va a utilizar para el desarrollo de esta aplicación.

Tabla 8. *Criterios y pesos para evaluar los protocolos de comunicación IoT.*

Criterios	Peso (%)
Seguridad	15
Transmisión de datos	10
Escalabilidad	15
Fiabilidad	30
Menor consumo de energía	15
Menor consumo de ancho de banda	15

Nota. *En esta tabla se definen los criterios para elegir el protocolo de comunicación IoT.*

Elaborado por: *Los autores.*

Tabla 9. *Matriz de priorización del protocolo IoT*

Criterios	**Peso (%)**	**CoAP**		**MQTT**	
		Puntaje	**%**	**Puntaje**	**%**
Seguridad	0.15	4	0.6	2	0.3
Transmisión de datos	0.10	3	0.3	5	0.5
Escalabilidad	0.15	2	0.3	5	0.75
Fiabilidad	0.30	2	0.6	4	1.2
Menor consumo de energía	0.15	5	0.75	4	0.6
Menor consumo de ancho de banda	0.15	4	0.6	5	0.75
Total	1		3.15		4.1

Nota. *Esta tabla muestra la puntuación de los protocolos IoT.*

Elaborado por: *Los Autores*

En la Tabla 9, se muestra la puntuación de los protocolos IoT, con base a los resultados obtenidos, se determina que el protocolo ideal para el desarrollo del proyecto es MQTT con 4.1 puntos a comparación de CoAP.

En la investigación realizada se determinó que el protocolo de mensajería MQTT es la mejor opción para implementar en la plataforma web ya que está diseñado para trabajar con dispositivos IoT, además proporciona una forma escalable y rentable de conectar dispositivos con un mínimo ancho de banda, garantizando la entrega de mensajes en tiempo real gracias a los niveles QoS (HiveMQ, 2019).

MQTT utiliza el protocolo de capa aplicación para la autentificación y autorización de dispositivos. El protocolo de transporte TCP/IP para agregar seguridad adicional a través del cifrado TLS para garantizar todas las comunicaciones. El bróker MQTT permite conexiones simultaneas con los clientes y la plataforma web (HiveMQ, 2019).

2.3.Análisis de base de datos

En la Tabla 10, se define los criterios más importantes y el porcentaje para determinar la base de datos que se debe utilizar para este proyecto.

Tabla 10. *Criterio y peso para evaluar la base de datos*

Criterios	**Peso (%)**
Integridad	15
Seguridad	20
Rendimiento	15
Escalabilidad	20
Flexibilidad	15
Documentación	15

Nota. *En esta tabla se definen los criterios para elegir la base de datos.*

Elaborado por: *Los autores*

Tabla 11. *Matriz de priorización de base de datos*

Criterios	**Peso (%)**	**MySql**		**MongoDB**	
		Puntaje	**Total**	**Puntaje**	**Total**
Integridad	0.15	4	0.6	4	0.6
Seguridad	0.20	4	0.8	5	1
Rendimiento	0.15	4	0.6	5	0.75
Escalabilidad	0.20	4	0.8	5	1
Flexibilidad	0.15	4	0.6	5	0.75
Documentación	0.15	5	0.75	3	0.45
Total	**1**		**4.15**		**4.55**

Nota. *Esta tabla muestra la puntuación de las bases de datos MongoDB y MySql.*

Elaborado por: *Los autores.*

En base a los resultados de la Tabla 11, se determinó que MongoDB es la base para este proyecto, con una puntuación de 4.55 a comparación de MySql.

En base a la investigación en el capítulo anterior sobre las bases de datos, se llegó a la conclusión que, para el desarrollo de este proyecto se utilizará MongoDB para almacenar los datos. Vergara (2020) plantea que para almacenar información de un dispositivo IoT que continuamente está enviando y recibiendo información en tiempo real, o se trata de guardar gran cantidad de datos para el análisis aplicaciones web en tiempo real, bases de datos flexibles, escalables, de alto rendimiento y alta funcionalidad, a fin de proporcionar magníficas experiencias a los usuarios, MongoDB es la solución.

2.4. Análisis económico

Para hacer una inversión se debe analizar los beneficios del proyecto. Para ello tenemos el VAN (Valor Actual Neto) y el TIR (Tasa Interna de Retorno), son indicadores financieros que consisten en determinar el beneficio y la rentabilidad que reportará el proyecto, una vez hecha la inversión.

VAN

El valor actual neto es un indicador financiero para determinar si un proyecto es viable, para esta consideración se debe tomar en cuenta dos criterios: si el VAN < 0, el proyecto no es rentable y si el VAN > 0 el proyecto es rentable.

Ecuación 1. *Fórmula para calcular el TIR*

$$VAN = \sum_{n=1}^{n} \frac{Fn}{(1+i)^n} - I_0$$

Fuente: (Morales, 2014)

Fn: Flujo de efectivo de cada periodo

n: Periodo de tiempo

i: Tasa de interés

I_0: Inversión Inicial (Morales, 2014)

TIR

La tasa interna de retorno es la rentabilidad que ofrece una inversión, es una medida para la evaluación de proyectos que muestra una medida relativa de rentabilidad tomando en cuenta los siguientes criterios: si TIR > i, el proyecto es aceptado, si TIR < i, el proyecto debe rechazarse (Arias, 2014).

Ecuación 2. *Fórmula para calcular el VAN*

$$VAN = \sum_{n=1}^{n} \frac{Fn}{(1+TIR)^n} - I_0 \ = 0$$

Fuente: (Arias, 2014)

Fn: Flujo de efectivo de cada periodo

n: Periodo de tiempo

TIR: Tasa de interés

Io: Inversión Inicial

En este apartado se calcula el VAN y TIR para determinar si el proyecto es factible, a continuación, se detalla la inversión inicial del proyecto:

Tabla 12. Detalle de la inversión inicial del proyecto

Rubro	**Cantidad**	**Costo unitario**	**Costo total**	**Detalle**
VPS	6	15	90	VUCAFLEX
Raspberry Pi3 B+	1	65	65	Mercado libre
Sensores			30	Mercado libre
Desarrollo			2000	Costo desarrollo
Materiales de implementación		20	20	Mercado libre
Total			**2205**	

Nota. *Esta tabla muestra las herramientas y servicios que se utilizan para el desarrollo del proyecto.*

Elaborado por: *Los autores.*

En la Figura 13, se muestran los ingresos, egresos, valor neto durante tres años, este tiempo se considera como vida útil de la aplicación. El costo por usuario se calculó entre los planes más vendidos de las plataformas: Kaaiot y Thinger. Boral (2021) muestra una lista de las mejores plataformas IoT, el software se conecta con el hardware IoT con sus herramientas para desarrolladores.

Tabla 13. *Ingresos, egresos, valor neto*

Año	Usuarios	Precio/usuario	Ingresos	Egresos	Valor neto
1	15	160	2400	2205	195
2	25	160	4000	800	3200
3	40	160	6400	1000	5400
Total			12800	4005	8795

Elaborado por: *Los autores.*

Para calcular el VAN se aplica la ecuación 1, con los valores de la tabla 12 y 13, con una tasa de interés del 12%, dando un resultado de $4363,74. Para el cálculo del TIR se aplica la ecuación 2 con los valores de la tabla 12 y 13, dando como resultado 74%. Con los resultados obtenidos, se determina que el proyecto es viable y rentable, ya que el TIR $>$ tasa de interés y el VAN > 0.

CAPÍTULO III

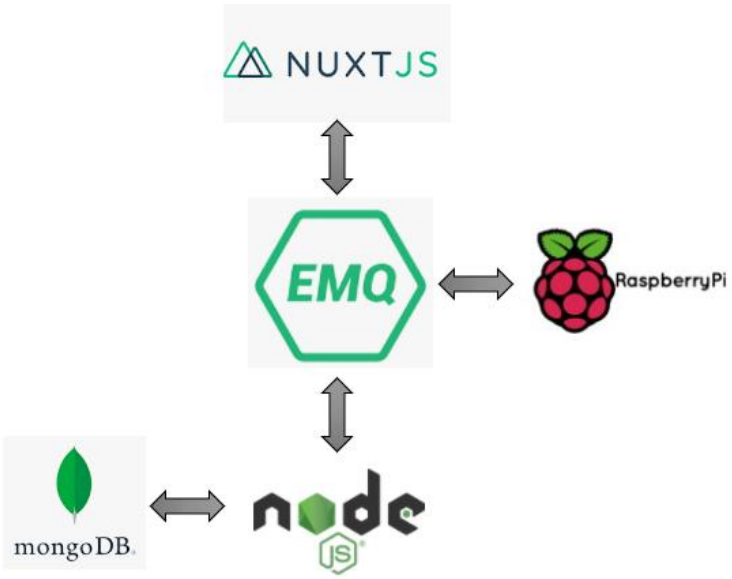
DISEÑO Y DESARROLLO

En este capítulo, se detallará la arquitectura del proyecto, las clases abstractas, el Raspberry Pi como cliente de la plataforma IoT y la base de datos del sistema.

3.1. Arquitectura del proyecto

En la Figura 7, se muestra la arquitectura del proyecto, para el front end se trabajará con el framework Nuxt js, el back end con Node js, el broker será EMQ X y Mongo db para almacenar los datos. Para los clientes se va a utilizar la librería paho, este proyecto comprende el desarrollo de la plataforma para clientes Raspberry Pi, para poder comunicar el software con el hardware, los usuarios deben registrarse en la plataforma, loguearse, crear una plantilla con el widget de su preferencia para luego crear un dispositivo, que genera las credenciales para la comunicación con el cliente Raspberry Pi, El cliente debe generar un script con estructura MQTT, importar la librería paho, escribir las credenciales generadas, especificar la GPIO con la que va enviar o recibir los datos, por último se ejecuta el script y se establece la comunicación.

Figura 7. *Arquitectura general del sistema*

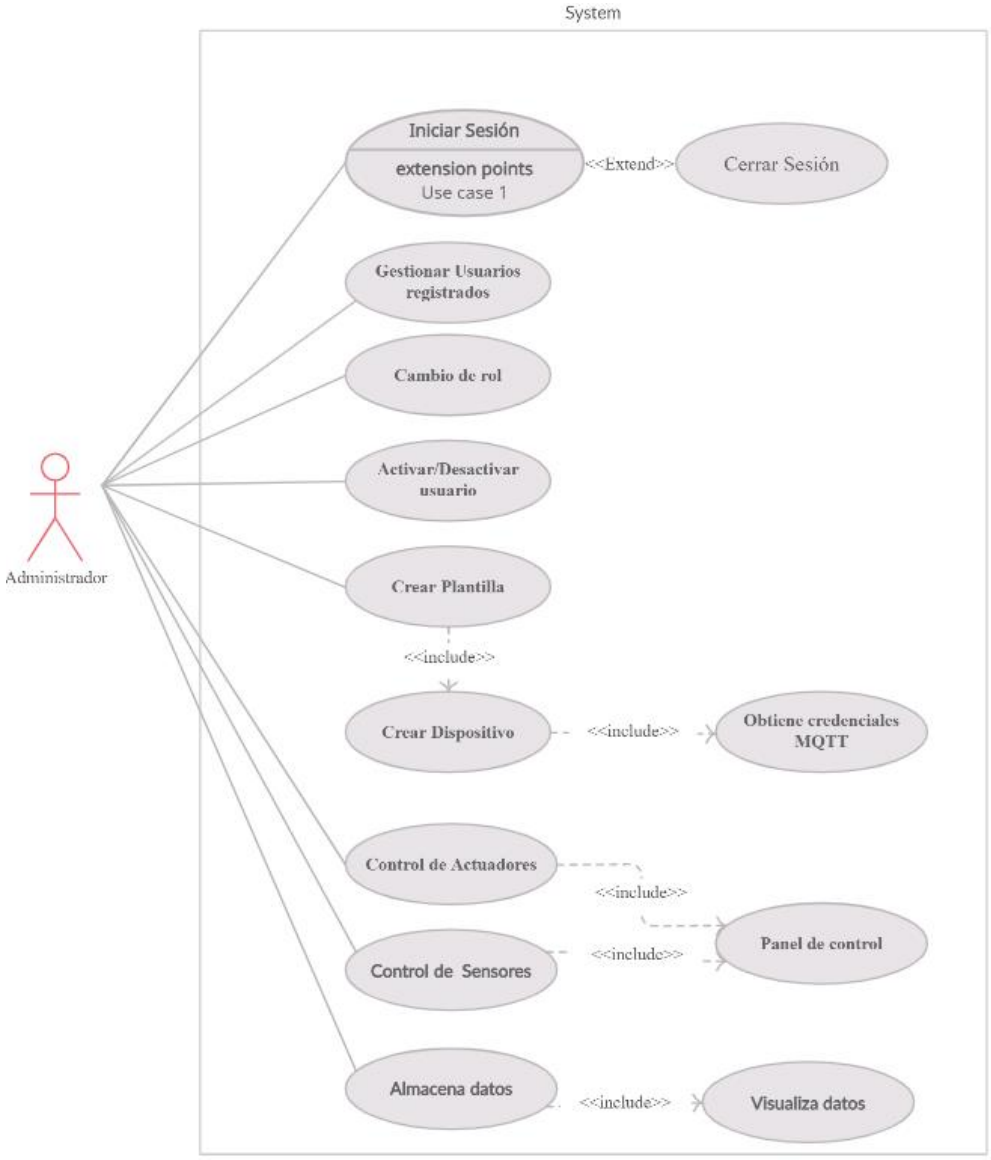


Elaborado por: Los autores

3.1. Diagramas de caso de uso

En la Figura 8, se observa el diagrama general utilizado para el desarrollo de la aplicación, donde se define el actor administrador y las acciones que realiza.

Figura 8. *Caso de uso para Administrador*



Nota. Interacción del Administrador con la plataforma.

Elaborado por: Los autores.

En la Tabla 14, se describe las precondiciones que debe cumplir para ejecutar los flujos de tareas, los mismo que permiten al usuario administrador realizar la gestión de usuario como modificar, activar o desactivar.

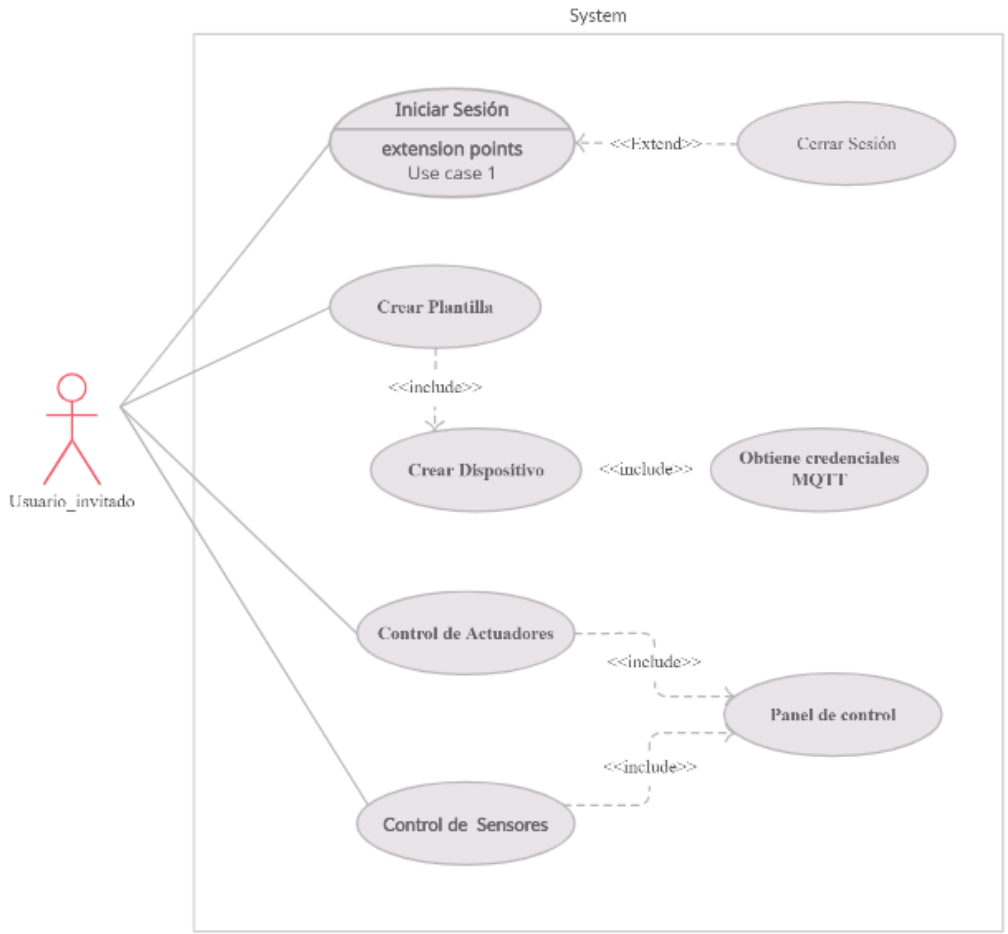
Tabla 14. *Descripción del caso de uso para el usuario Administrador*

Caso de uso: Gestión de Usuario	CDU01
Actor: Usuario Administrador	
Descripción: Permite activar, desactivar y modificar datos de los usuarios.	
Precondiciones: ✓ Debe estar registrado en la plataforma ✓ Debe iniciar sesión en la plataforma	
Flujo Principal: 1. Inicia sesión en la plataforma. 2. Validar credenciales. 3. Seleccionar la opción usuarios. 4. Cambia de rol a los usuarios. 5. Activa o desactiva usuarios. 6. Crea plantillas. 7. Crea dispositivos. 8. Visualiza el dashboard. 9. Visualiza los datos.	
Flujo secundario: 1. En el paso 2 verifica que las credenciales sean correctas caso contrario muestra un mensaje de error.	
Postcondiciones 1. El administrador modifica, activa o desactiva usuarios en la plataforma.	

2. El administrador crea plantillas, dispositivos los mismos que puede almacenar y visualizar en el dashboard.

Elaborado por: *Los autores.*

Figura 9. *Caso de uso para Usuario_invitado*



Nota. Interacción del Usuario_invitado con la plataforma

Elaborado por: Los autores.

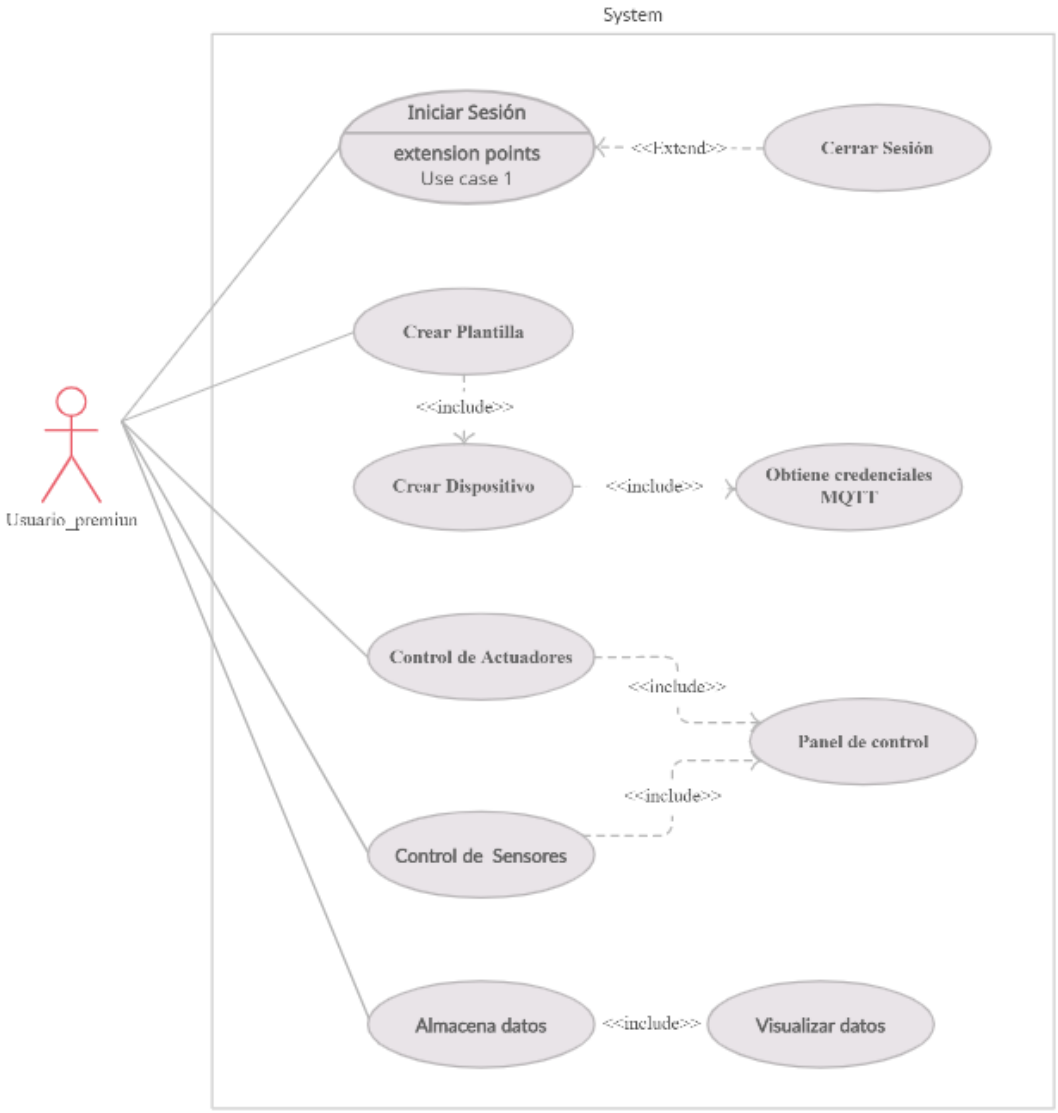
En la Tabla 15, se describe las precondiciones que debe cumplir para ejecutar los flujos de tareas, los mismo que permiten al usuario invitado crea plantillas, dispositivos los mismos que puede visualizar en el dashboard.

Tabla 15. *Descripción del caso de uso para Usuario_invitado*

Caso de uso: Usuario invitado	CDU02
Actor: Usuario_invitado	
Descripción: Permite crear, eliminar plantillas y dispositivos.	
Precondiciones: ✓ Debe estar registrado en la plataforma ✓ Debe iniciar sesión en la plataforma	
Flujo Principal: 1. Inicia sesión en la plataforma. 2. Validar credenciales. 3. Crea plantillas. 4. Crea dispositivos. 5. Visualiza el dashboard.	
Flujo secundario: 1. En el paso 2 verifica que las credenciales sean correctas caso contrario muestra un mensaje de error.	
Postcondiciones 1. El usuario invitado crea plantillas, dispositivos los mismos que puede visualizar en el dashboard.	

Elaborado por: *Los autores*

Figura 10. *Caso de uso para Usuario_premiun*



Nota. Interacción del usuario_premiun con la plataforma.

Elaborado por: Los autores.

En la Tabla 16, se describe las precondiciones que debe cumplir para ejecutar los flujos de tareas, los mismo que permiten al usuario_premiun crea plantillas, dispositivos los mismos que puede almacenar y visualizar en el dashboard.

Tabla 16. *Descripción de caso de uso para Usuario_premiun*

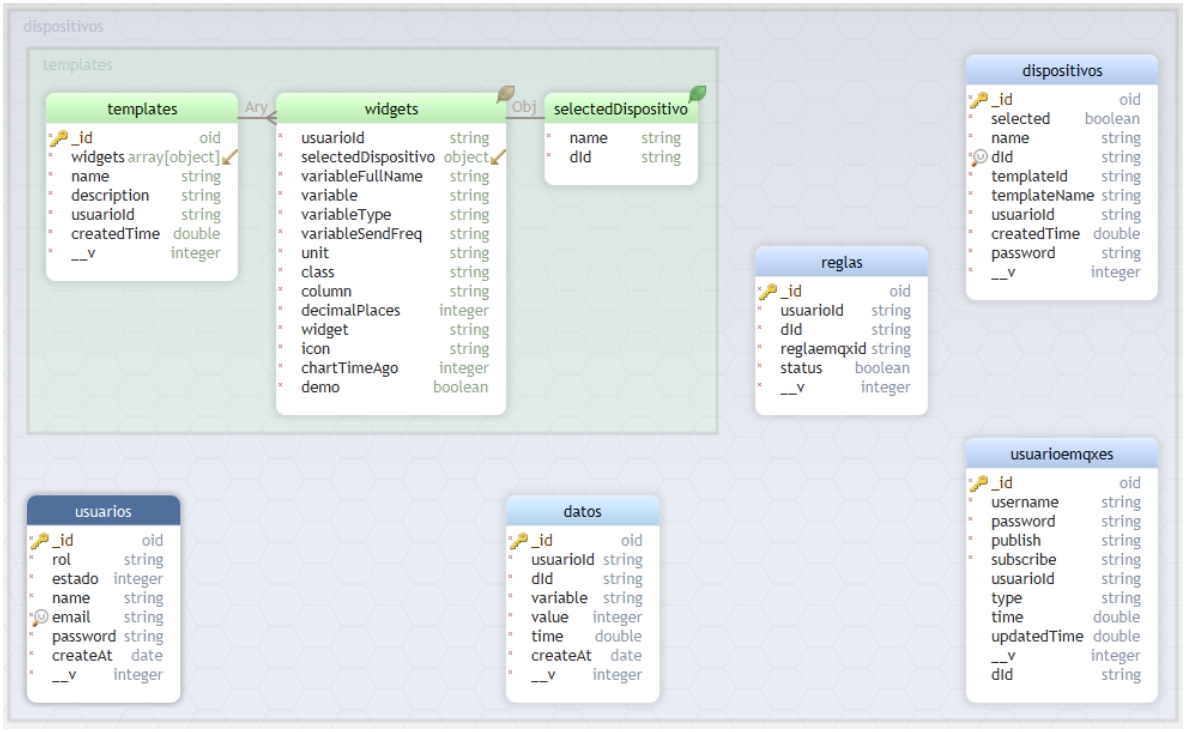
Caso de uso: Usuario_premiun	CDU03
Actor: Usuario_premiun	
Descripción: Permite crear, eliminar plantillas y dispositivos.	
Precondiciones: ✓ Debe estar registrado en la plataforma ✓ Debe iniciar sesión en la plataforma	
Flujo Principal: 1. Inicia sesión en la plataforma. 2. Validar credenciales. 3. Crea plantillas. 4. Crea dispositivos. 5. Visualiza el dashboard. 6. Visualiza los datos.	
Flujo secundario: 1. En el paso 2 verifica que las credenciales sean correctas caso contrario muestra un mensaje de error.	
Postcondiciones 1. El usuario_premiun crea plantillas, dispositivos los mismos que puede almacenar y visualizar en el dashboard.	

***Elaborado por**: Los Autores.*

3.2. Diagrama de base de datos

Mongodb no maneja estándares gráficos como en las bases de datos SQL, Para tener una abstracción grafica del diagrama de base de datos se utilizó la herramienta DbSchema, permitiendo entender de mejor manera el esquema de la base de datos.

Figura 11. *Diagrama de base de datos*



Nota. MongoDB al ser una base no relacional funciona con datos no estructurados, siendo esta una solución flexible para el desarrollo de este sistema.

Elaborado por: Los autores.

3.3. Interfaces abstractas

En la Figura 12, se muestra se muestra la pantalla para ingresar al sistema, para acceder se debe estar previamente registrado.

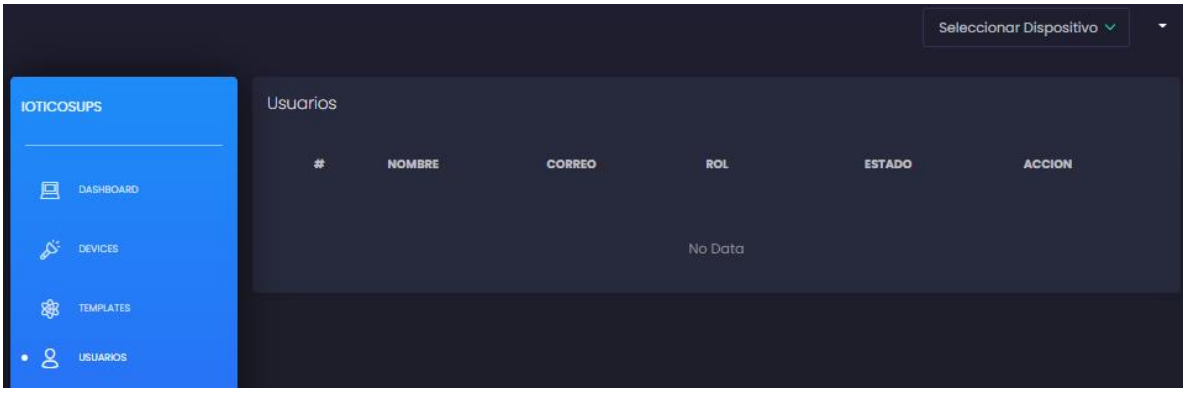
Figura 12. *Representación abstracta del login de la plataforma*

Nota. Interfaz de login para acceder a la plataforma.

Elaborado por: Los autores.

En la Figura 13, se muestra la interfaz de usuarios, la misma que permite: listar, activar, desactivar, modificar. La interfaz de usuarios la puede ver únicamente el administrador.

Figura 13. *Representación abstracta de la interfaz de usuarios*



Nota. Interfaz de gestión de usuarios

Elaborado por: Los autores.

En la Figura 14, se presenta la interfaz de dispositivos, crea y lista los dispositivos por cada usuario, asignando credenciales para la conexión con el Raspberry.

Figura 14. *Representación abstracta de la interfaz de dispositivos*

Nota. Pantalla de gestión de dispositivos.

Elaborado por: Los autores.

3.4. Desarrollo

En este apartado se detallará las partes más importantes que se utilizaron para el desarrollo del sistema, se mostrará parte del código, conexión a la base de datos y comunicación con el cliente MQTT.

3.5. Conexión de MongoDB con EMQX

En la Figura 15, se muestra la conexión de MongoDB con el broker EMQX, Para habilitar la autenticación de MongoDB, debe configurar lo siguiente en etc/plugins/emqx_auth_mongo.conf

Figura 15. *Conexión del broker EMQX con MongoDB*

```
## MongoDB Topology Type.
##
## Value: single | unknown | sharded | rs
auth.mongo.type = single
## The set name if type is rs.
## Value: String
## auth.mongo.rs_set_name =
## MongoDB server list.
## Value: String
## Examples: 127.0.0.1:27017,127.0.0.2:27017...
auth.mongo.server = 104.218.52.111:27017
## MongoDB pool size
## Value: Number
auth.mongo.pool = 8
## MongoDB login user.
## Value: String
auth.mongo.login = ioticosups
## MongoDB password.
## Value: String
auth.mongo.password = Parxxxxxx
## MongoDB AuthSource
## Value: String
## Default: mqt
auth.mongo.auth_source = admin
## MongoDB database
## Value: String
#auth.mongo.database = mqtt
auth.mongo.database = ioticosups
```

Nota. Configuración del plugin mongodb en EMQX.

Elaborado por: Los autores.

En la figura 16, se muestra la función para recuperar las credenciales MQTT desde el Raspberry Pi, mismas que se obtienen con el id del dispositivo y la contraseña.

Figura 16. *Función para obtener las credenciales MQTT*

```
router.post("/getData", async (req, res) => {
  try {
    const dId = req.body.dId;
    const password = req.body.password;
    const dispositivo = await Dispositivo.findOne({ dId: dId });
    if (password != dispositivo.password) {
      return res.status(401).json();
    }
    const usuarioId = dispositivo.usuarioId;
    var credentials = await GetData(dId, usuarioId);
    var template = await Template.findOne({ _id: dispositivo.templateId });
    var variables = [];
    const response = {
      username: credentials.username,
      password: credentials.password,
      topic: usuarioId + "/" + dId + "/",
      variables: variables
    };
    res.json(response);
    setTimeout(() => {
      GetData(dId, usuarioId);
    }, 10000);
  } catch (error) {
    console.log(error);
    res.sendStatus(500);
  }
});
```

Nota. Con esta función se obtiene las credenciales MQTT tales como: usuario y topic. Estas credenciales permiten la comunicación del Raspberry con la plataforma.

Elaborado por: Los autores.

En la Figura 17, se define la estructura de los topics, tanto para publicación como para suscripción, separando sus niveles con el signo (/) las mismas que hacen posible el envío y recepción de mensajes.

Figura 17. *Topic y sus parámetros*

```
publish: [usuarioId + "/" + dId + "/+/sdata"],
subscribe: [usuarioId + "/" + dId + "/+/actdata"],
```

Nota. Topic publicación/suscripción y estructura.

Elaborado por: Los autores

En la Figura 18, muestra la codificación, decodificación y tiempo de expiración del token, misma que define el mecanismo para realizar una conexión de forma segura, verificando si el remitente es quien dice ser y asegurar que el mensaje sea legítimo.

Figura 18. *Estructura del token*

```
export default {
    encode: async (_id,rol,email) => {
        const token = jwt.sign({_id:_id,rol:rol,email:email},'securePasswordHere',{expiresIn: '1d'});
        return token;
    },
    decode: async (token) => {
        try {
            const {_id} = await jwt.verify(token,'securePasswordHere');
            const user = await Usuario.findOne({_id,estado:1});
            if (user){
                return user;
            } else{
                return false;
            }
        } catch (e){
            const newToken = await checkToken(token);
            return newToken;
        }
    }
}
```

Nota. Codificación y decodificación del token.

Elaborado por: Los autores.

CAPÍTULO IV

PRUEBAS Y RESULTADOS

A continuación, se presenta las pruebas y resultados de funcionamiento de la plataforma, envió y recepción de mensajes, comunicación entre hardware y software, sensores analógicos y digitales, carga de datos en las diferentes interfaces.

4.1. Pruebas de funcionamiento

En esta sección se muestra el funcionamiento de la plataforma, cargando datos en todas las interfaces e interactuando con el cliente MQTT (Raspberry py).

4.1.1. Pantalla de registro de usuario

En la Figura 19, se presenta la interfaz de registro de usuarios, para registrar un usuario nuevo se necesitan tres datos: nombre, correo, contraseña. El nuevo usuario tiene asignado por defecto el rol usuario_invitado, que limita el número de dispositivos y el acceso a los datos guardados.

Figura 19. *Registro usuario nuevo*

Nota. Crea un usuario nuevo y le asigna un rol por defecto.

Elaborado por: Los autores.

4.1.2. Interfaz de inicio de sesión

En la Figura 20, se observa el formulario de inicio de sesión, dicho formulario esta validado para que los campos sean obligatorios, luego de iniciar sesión se accede a la pestaña dashboard, donde se muestran los dispositivos conectados.

Figura 20. *Inicio de sesión*

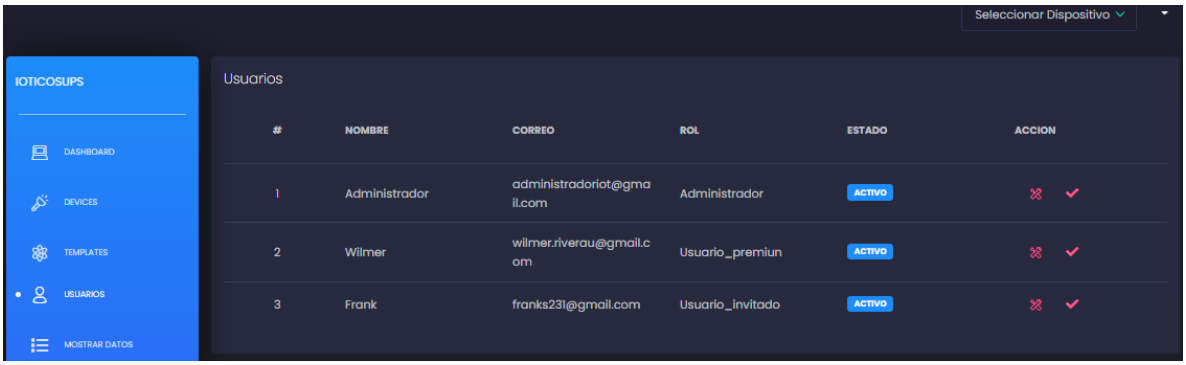
Nota. Pantalla de login.

Elaborado por: Los autores.

4.1.3. Interfaz de usuarios

En la Figura 21, se presenta la pantalla que lista los usuarios de la plataforma, permite actualizar los datos, estado y cambiar rol. El rol administrador es el único que tiene privilegios para realizar dichos cambios.

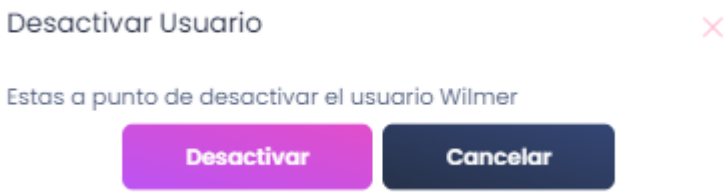
Figura 21. *Lista de usuarios creados*



Nota. Gestión de usuarios

Elaborado por: Los autores.

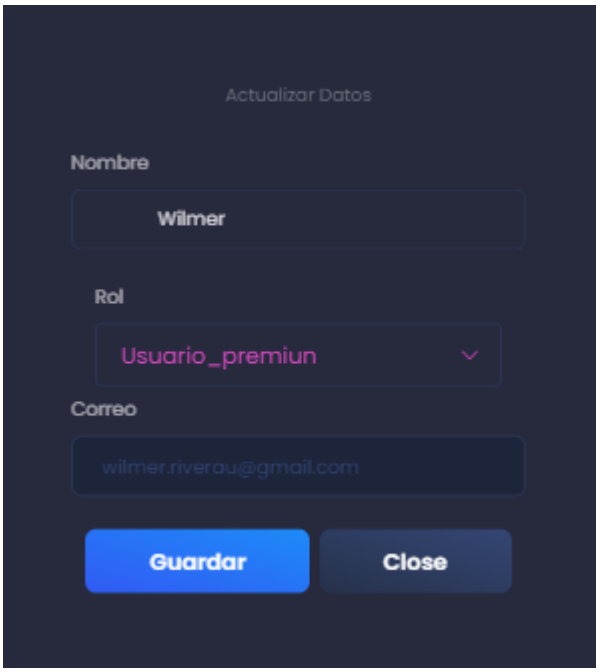
Figura 22. *Mensaje de alerta*



Nota. Mensaje de alerta para desactivar los usuarios.

Elaborado por: Los usuarios.

Figura 23. *Interfaz para actualizar datos*



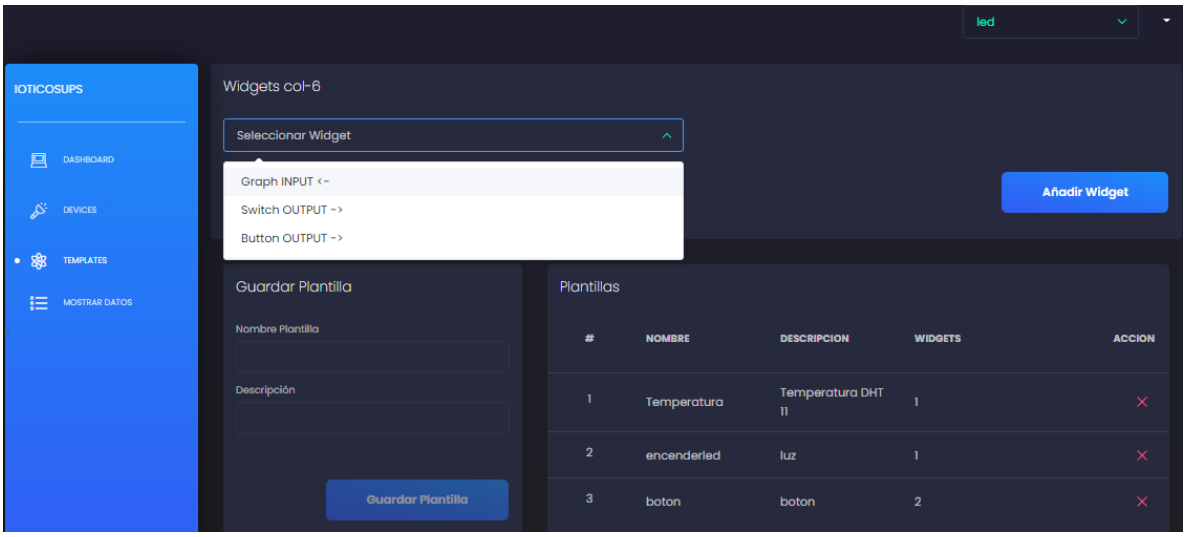
Nota. Actualiza los datos del usuario.

Elaborado por: Los Autores.

4.1.4. Interfaz de templates

En la Figura 24, Se observa la pantalla de templates, puede elegir tres opciones de template: grafica, switch y botón. A demás lista todas las plantillas creadas por el usuario.

Figura 24. *Interfaz para crear plantilla*



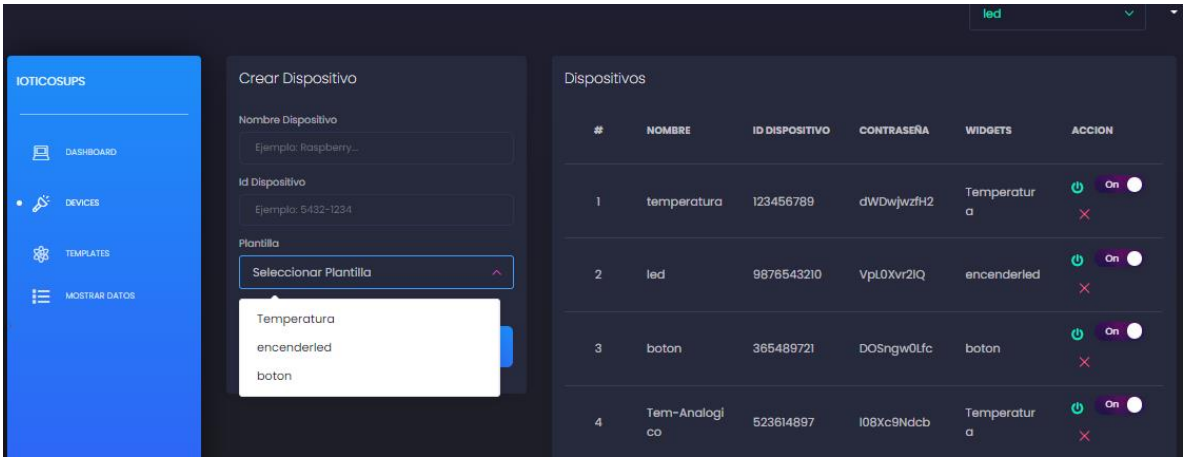
Nota. Elige entre tres widgets para mostrar o interactuar con el Raspberry.

Elaborado por: Los autores.

4.1.5 Interfaz dispositivo

En la Figura 25, se muestra la pantalla de dispositivos, para crear un dispositivo es necesario seleccionar una plantilla,

Figura 25. *Interfaz para crear dispositivo*



Nota. Crea los dispositivos y genera las credenciales para la comunicación con el cliente.

Elaborado por: Los autores.

4.2. Comunicación con el Raspberry Pi

4.2.1. Comunicación del cliente Raspberry Pi Con la plataforma web

En la Figura 26, se muestra las variables de conexión con las credenciales generadas en la plataforma web, dichas credenciales son llamadas por la función connect_mqtt para establecer la conexión.

Figura 26. *Conexión del Raspberry Pi con la plataforma web*

```
pin = 4 #Pin en la raspberry donde conectamos el sensor
DeviceId = "123456789" # Id del dispositivo creado en la plataforma
ClientDevicePassword = "0jfNf2OopL" #Contraseña del dispositivo

#validacion de credenciales
CredentialsMqtt = "http://ioticosups.tk:3001/api/getData"
broker = 'ioticosups.tk'
port = 1883
data = {'dId': DeviceId , 'password': ClientDevicePassword}
res = requests.post(url = CredentialsMqtt, data=data)
con = res.status_code
if con < 0:
    print("Error de credenciales")
    exit()
if con != 200:
    print("Error de credenciales")
    exit()
if con == 200:
    print("Bienvenido")
lista = []
lista.append(res.json())

for elemento in lista:
 username = elemento['username']
 password = elemento['password']
 str_topic = elemento['topic']

for i in range(len(elemento['variables'])):
    vari1 = elemento['variables'][0]['variable']
client_id = "device_"+DeviceId+"_"+f'{random.randint(0, 1000)}'

topic1 = str_topic + vari1+'/sdata'

def connect_mqtt():
    def on_connect(client, userdata, flags, rc):
        if rc == 0:
            print("Conexion MQTT exitosa!")
        else:
            print("Conexion fallida!! %d\n", rc)

    client = mqtt_client.Client(client_id)
    client.username_pw_set(username, password)
    client.on_connect = on_connect
    client.connect(broker, port)
    return client
```

Nota. Si las credenciales son correctas, se establece la conexión, caso contrario, se debe revisar las que las credenciales sean correctas.

Elaborado por: Los autores.

En la Figura 27, se muestra la función que envía los mensajes con los datos obtenidos con el sensor DHT11.

Figura 27 *Publicación de datos a la plataforma*

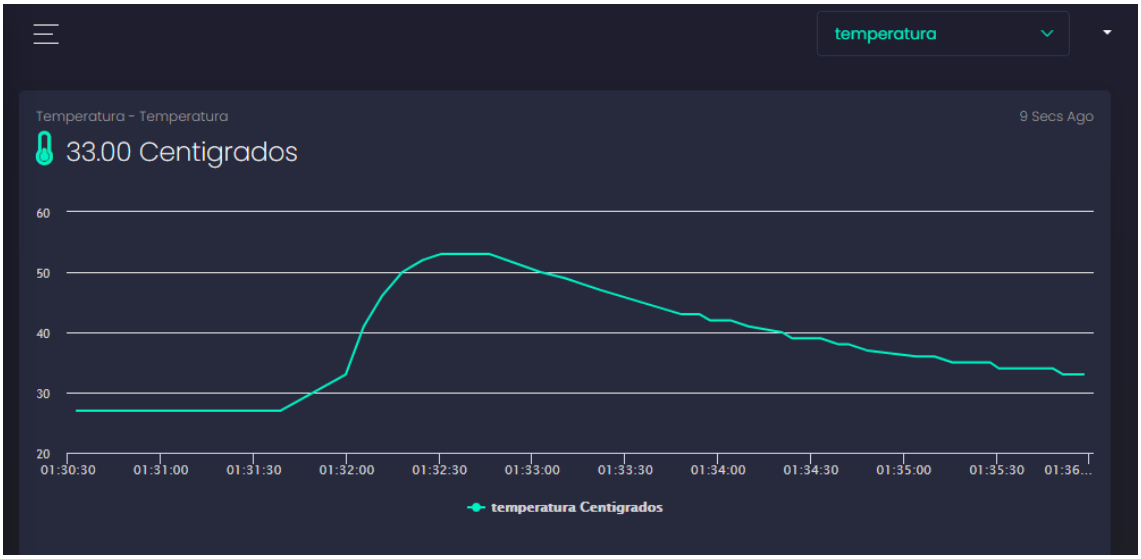
```
def publish(client):

    while True:
        #lectura de datos desde el sensor
        sensor = Adafruit_DHT.DHT11
        humedad, temperatura = Adafruit_DHT.read_retry(sensor, pin)
        print (humedad,temperatura)
        temp= str(temperatura)
        hum= str(humedad)
        time.sleep(0.25)
        #envio de datos a la plataforma
        msg1 = '{"save": 1, "value":'+temp+' }'
        client.publish(topic1, msg1)
def run():
    client = connect_mqtt()
    client.loop_start()
    publish(client)

if __name__ == '__main__':
    run()
```

Elaborado por: Los autores.

4.2.2. Gráfico de temperatura con sensor digital

En la Figura 28, se muestran los datos obtenidos desde la Raspberry Pi con el sensor digital DHT11.

Figura 28. *Visualización de los datos obtenidos del sensor DHT11*



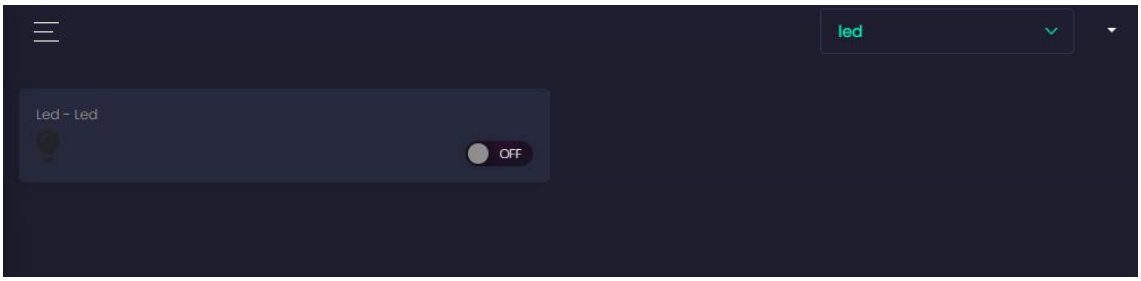
Nota. Visualización en tiempo real de los datos enviados desde la Raspberry Pi en forma de mensajes.

Elaborado por: Los autores.

4.2.3. Control de led con switch

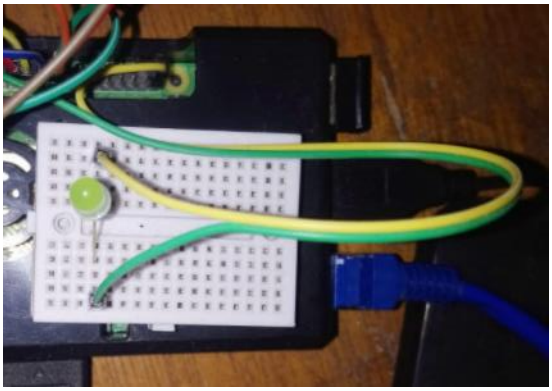
En la Figura 29, se presenta el control de led a través de un switch, se prende y apaga el led al presionar el botón.

Figura 29. *Switch off*



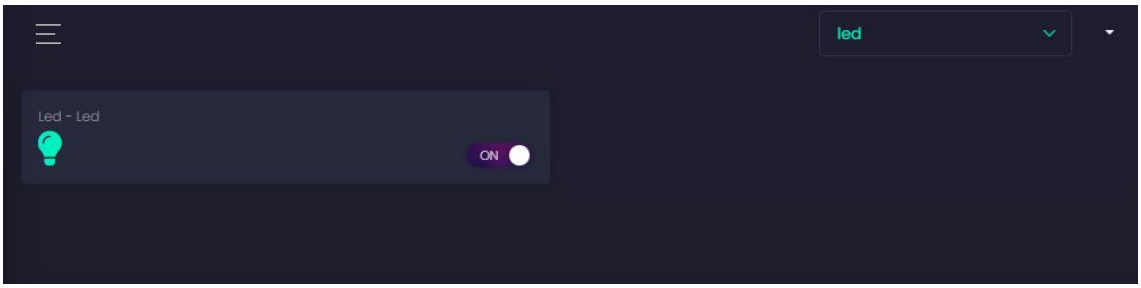
Elaborado por: Los autores

Figura 30. *Led apagado*



Elaborado por: Los autores.

Figura 31. *Switch on*



Elaborado por: Los autores.

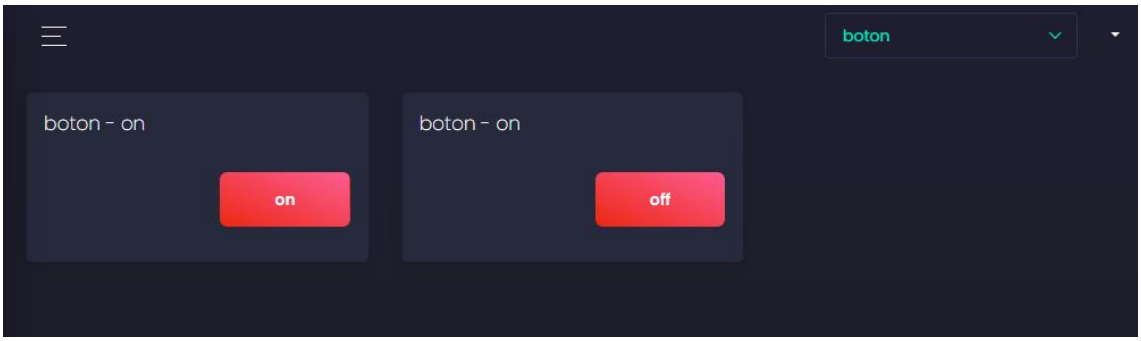
Figura 32. *Led encendido*

Elaborado por: Los autores.

4.2.4. Control de led con botones

En la Figura 33, se muestra el control de un led con dos botones, uno enciende y el otro apaga, cada botón envía un mensaje diferente (on/off).

Figura 33. *Control de led con botones*

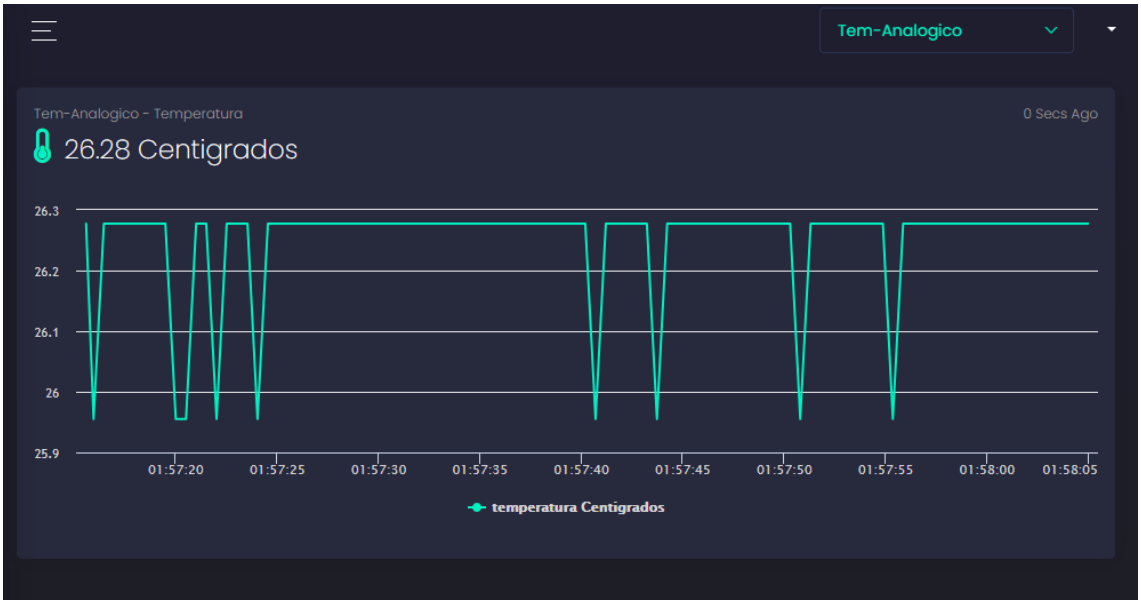


Elaborado por: Los autores.

4.2.5. Gráfico de temperatura con sensor analogico

En la Figura 34, se muestra la gráfica de la tempreratura en tiempo real, la plataforma admite unicamente señales digitales, para esta prueba usamos el sensor LM35 con un convertidor analogo-digital MCP 3008.

Figura 34. *Temperatura sensor LM35*

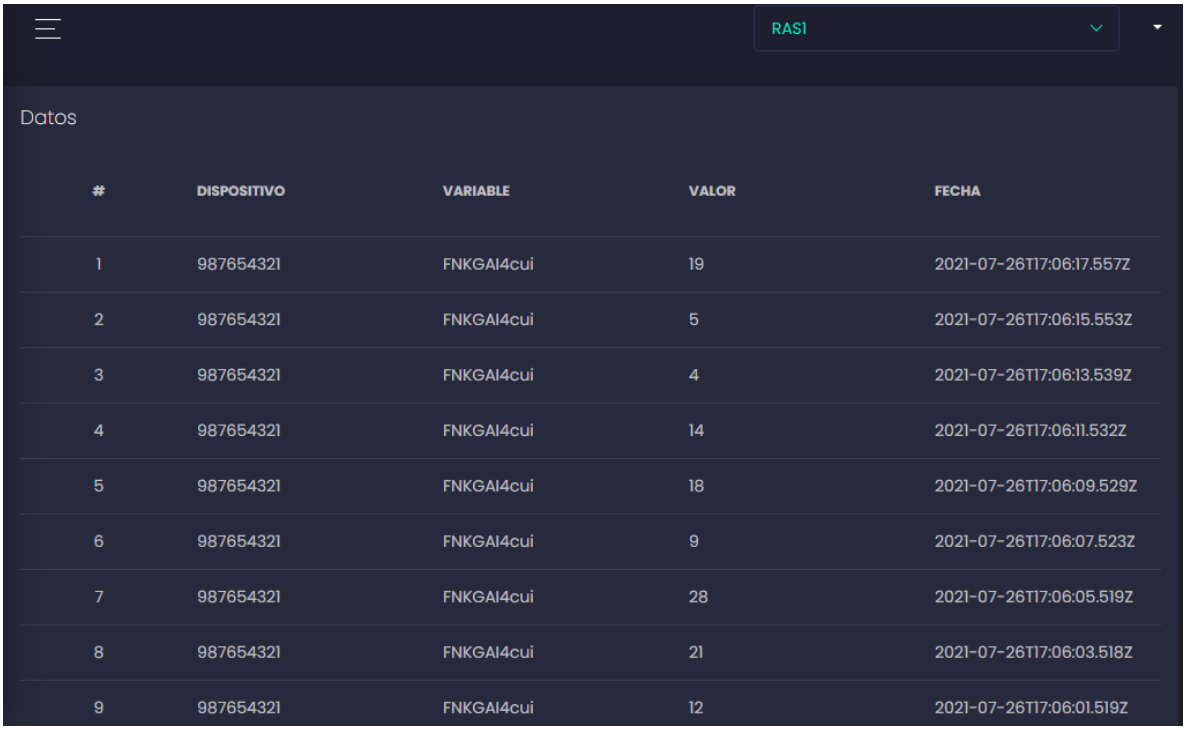


Elaborado por: Los autores.

4.2.6. Datos guardados

En la Figura 35, se muestra los valores guardados de cada sensor,

Figura 35. *Valores guardados*



Nota. Muestra los datos enviados por los sensores.

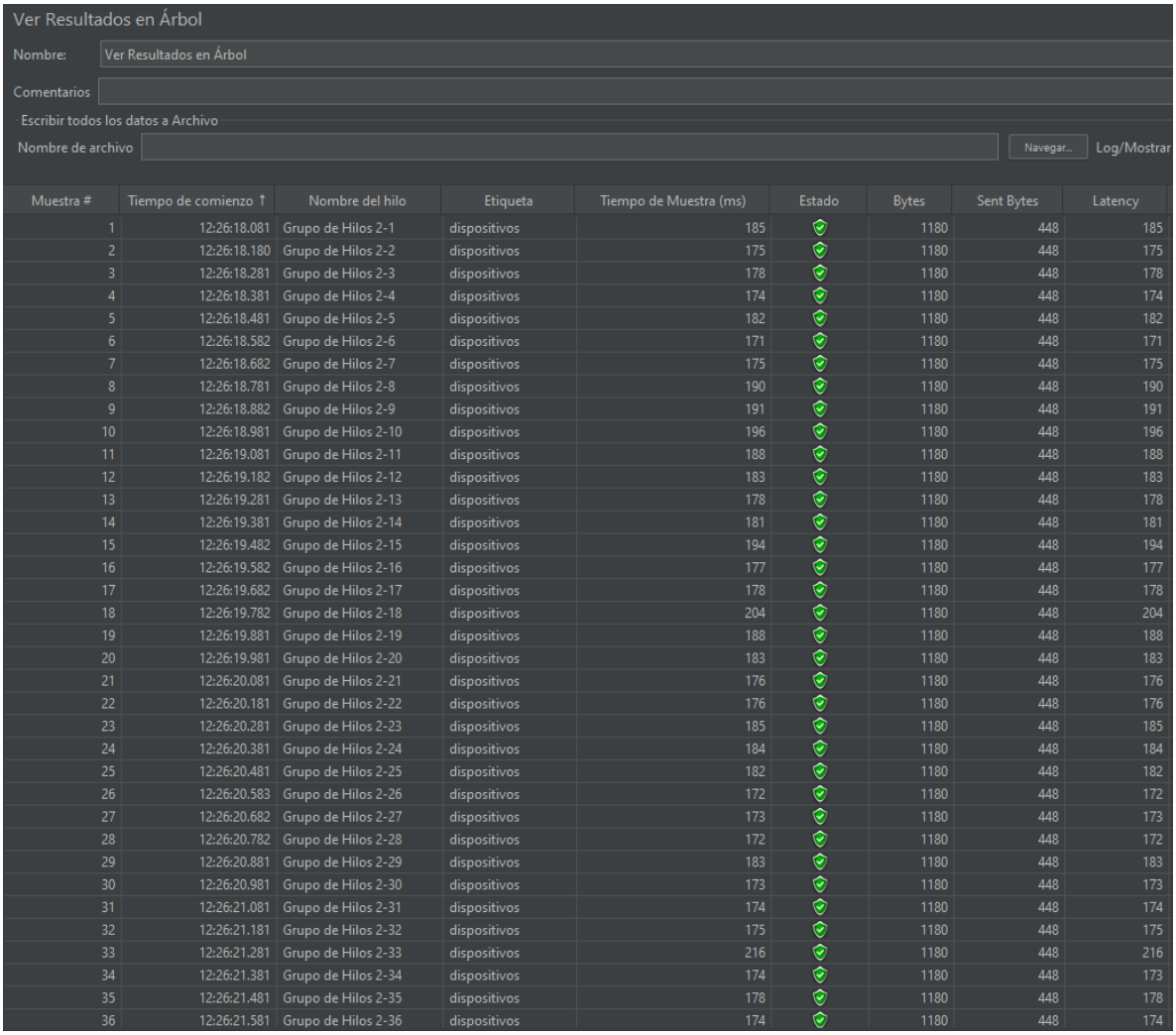
Elaborado por: Los autores.

4.3. Pruebas de carga

En este apartado, se muestran las pruebas de carga en la plataforma, utilizando la herramienta “Jmeter”, las pruebas fueron evaluadas con 500 peticiones cada 50 segundos obteniendo los siguientes resultados.

En la Figura 36, se muestra la ejecución de cada hilo que simula el módulo de dispositivos por 500 peticiones http, donde se visualiza el estado en color verde y que la latencia es aceptable.

Figura 36. *Prueba de carga modulo dispositivos*

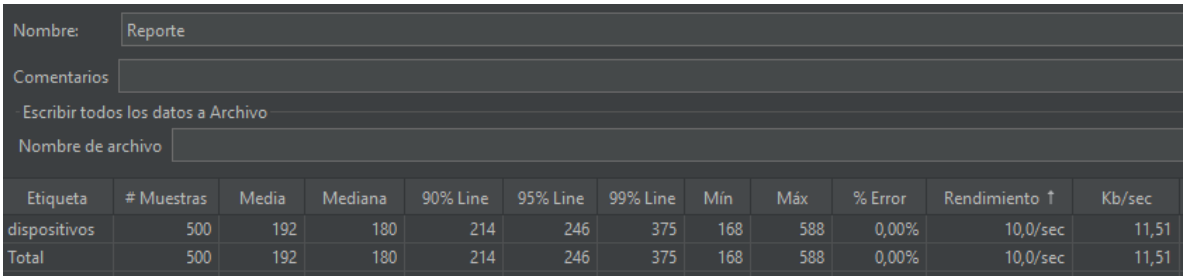


Muestra #	Tiempo de comienzo	Nombre del hilo	Etiqueta	Tiempo de Muestra (ms)	Estado	Bytes	Sent Bytes	Latency
1	12:26:18.081	Grupo de Hilos 2-1	dispositivos	185		1180	448	185
2	12:26:18.180	Grupo de Hilos 2-2	dispositivos	175		1180	448	175
3	12:26:18.281	Grupo de Hilos 2-3	dispositivos	178		1180	448	178
4	12:26:18.381	Grupo de Hilos 2-4	dispositivos	174		1180	448	174
5	12:26:18.481	Grupo de Hilos 2-5	dispositivos	182		1180	448	182
6	12:26:18.582	Grupo de Hilos 2-6	dispositivos	171		1180	448	171
7	12:26:18.682	Grupo de Hilos 2-7	dispositivos	175		1180	448	175
8	12:26:18.781	Grupo de Hilos 2-8	dispositivos	190		1180	448	190
9	12:26:18.882	Grupo de Hilos 2-9	dispositivos	191		1180	448	191
10	12:26:18.981	Grupo de Hilos 2-10	dispositivos	196		1180	448	196
11	12:26:19.081	Grupo de Hilos 2-11	dispositivos	188		1180	448	188
12	12:26:19.182	Grupo de Hilos 2-12	dispositivos	183		1180	448	183
13	12:26:19.281	Grupo de Hilos 2-13	dispositivos	178		1180	448	178
14	12:26:19.381	Grupo de Hilos 2-14	dispositivos	181		1180	448	181
15	12:26:19.482	Grupo de Hilos 2-15	dispositivos	194		1180	448	194
16	12:26:19.582	Grupo de Hilos 2-16	dispositivos	177		1180	448	177
17	12:26:19.682	Grupo de Hilos 2-17	dispositivos	178		1180	448	178
18	12:26:19.782	Grupo de Hilos 2-18	dispositivos	204		1180	448	204
19	12:26:19.881	Grupo de Hilos 2-19	dispositivos	188		1180	448	188
20	12:26:19.981	Grupo de Hilos 2-20	dispositivos	183		1180	448	183
21	12:26:20.081	Grupo de Hilos 2-21	dispositivos	176		1180	448	176
22	12:26:20.181	Grupo de Hilos 2-22	dispositivos	176		1180	448	176
23	12:26:20.281	Grupo de Hilos 2-23	dispositivos	185		1180	448	185
24	12:26:20.381	Grupo de Hilos 2-24	dispositivos	184		1180	448	184
25	12:26:20.481	Grupo de Hilos 2-25	dispositivos	182		1180	448	182
26	12:26:20.583	Grupo de Hilos 2-26	dispositivos	172		1180	448	172
27	12:26:20.682	Grupo de Hilos 2-27	dispositivos	173		1180	448	173
28	12:26:20.782	Grupo de Hilos 2-28	dispositivos	172		1180	448	172
29	12:26:20.881	Grupo de Hilos 2-29	dispositivos	183		1180	448	183
30	12:26:20.981	Grupo de Hilos 2-30	dispositivos	173		1180	448	173
31	12:26:21.081	Grupo de Hilos 2-31	dispositivos	174		1180	448	174
32	12:26:21.181	Grupo de Hilos 2-32	dispositivos	175		1180	448	175
33	12:26:21.281	Grupo de Hilos 2-33	dispositivos	216		1180	448	216
34	12:26:21.381	Grupo de Hilos 2-34	dispositivos	174		1180	448	173
35	12:26:21.481	Grupo de Hilos 2-35	dispositivos	178		1180	448	178
36	12:26:21.581	Grupo de Hilos 2-36	dispositivos	174		1180	448	174

Elaborado por: Los autores, a través de la herramienta Jmeter.

En la Figura 37, se muestra un reporte de resultados para el módulo de dispositivos, las 500 peticiones http tardan un tiempo mínimo de 182 milisegundos, un tiempo máximo de 454 milisegundos y su media de 209 milisegundos, sin porcentaje de error con un tiempo de rendimiento de 10 segundos.

Figura 37. *Reporte de resultados módulo de dispositivos*



Nombre: Reporte

Comentarios

Escribir todos los datos a Archivo

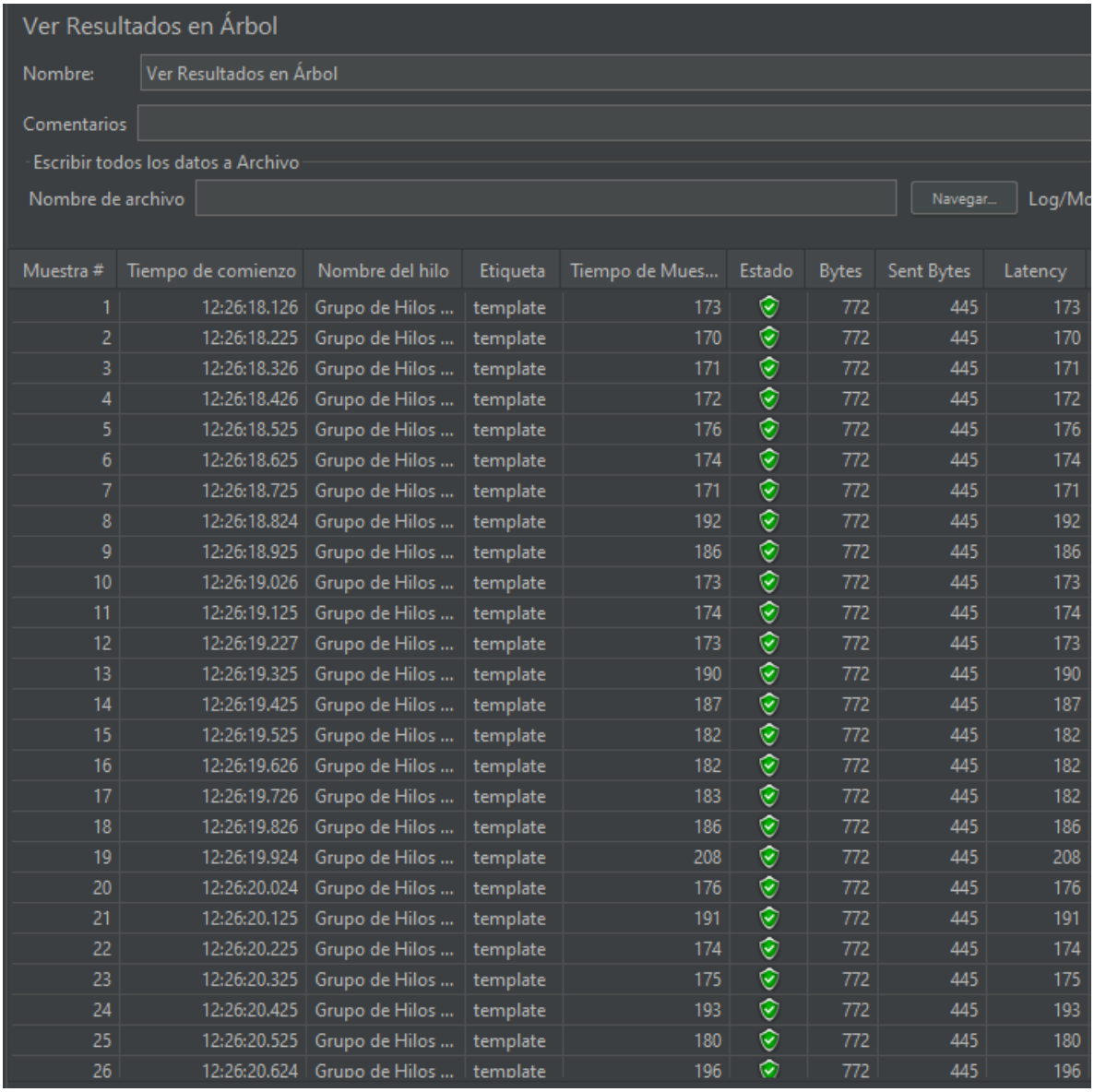
Nombre de archivo

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento ↑	Kb/sec
dispositivos	500	192	180	214	246	375	168	588	0,00%	10,0/sec	11,51
Total	500	192	180	214	246	375	168	588	0,00%	10,0/sec	11,51

Elaborado por: Los autores, a través de la herramienta Jmeter.

En la Figura 38, se muestra la ejecución de cada hilo que simula el módulo de templates con 500 peticiones http, donde se visualiza el estado en color verde y que la latencia es aceptable.

Figura 38. *Prueba de carga módulo de templates*

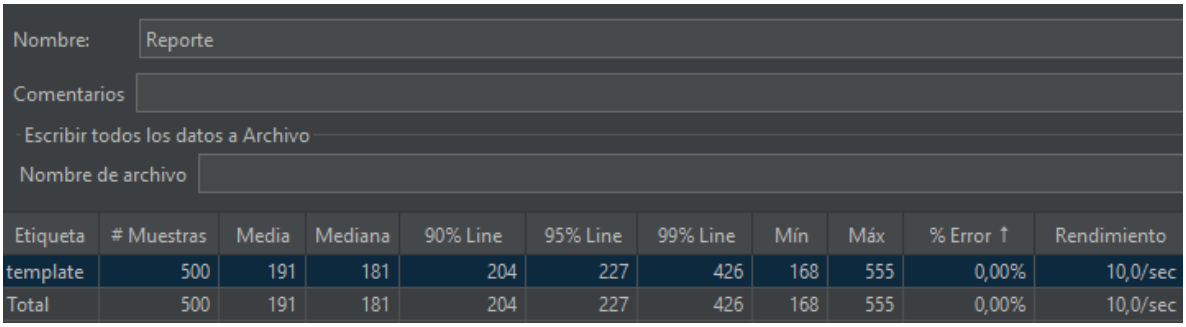


Muestra #	Tiempo de comienzo	Nombre del hilo	Etiqueta	Tiempo de Mues...	Estado	Bytes	Sent Bytes	Latency
1	12:26:18.126	Grupo de Hilos ...	template	173		772	445	173
2	12:26:18.225	Grupo de Hilos ...	template	170		772	445	170
3	12:26:18.326	Grupo de Hilos ...	template	171		772	445	171
4	12:26:18.426	Grupo de Hilos ...	template	172		772	445	172
5	12:26:18.525	Grupo de Hilos ...	template	176		772	445	176
6	12:26:18.625	Grupo de Hilos ...	template	174		772	445	174
7	12:26:18.725	Grupo de Hilos ...	template	171		772	445	171
8	12:26:18.824	Grupo de Hilos ...	template	192		772	445	192
9	12:26:18.925	Grupo de Hilos ...	template	186		772	445	186
10	12:26:19.026	Grupo de Hilos ...	template	173		772	445	173
11	12:26:19.125	Grupo de Hilos ...	template	174		772	445	174
12	12:26:19.227	Grupo de Hilos ...	template	173		772	445	173
13	12:26:19.325	Grupo de Hilos ...	template	190		772	445	190
14	12:26:19.425	Grupo de Hilos ...	template	187		772	445	187
15	12:26:19.525	Grupo de Hilos ...	template	182		772	445	182
16	12:26:19.626	Grupo de Hilos ...	template	182		772	445	182
17	12:26:19.726	Grupo de Hilos ...	template	183		772	445	182
18	12:26:19.826	Grupo de Hilos ...	template	186		772	445	186
19	12:26:19.924	Grupo de Hilos ...	template	208		772	445	208
20	12:26:20.024	Grupo de Hilos ...	template	176		772	445	176
21	12:26:20.125	Grupo de Hilos ...	template	191		772	445	191
22	12:26:20.225	Grupo de Hilos ...	template	174		772	445	174
23	12:26:20.325	Grupo de Hilos ...	template	175		772	445	175
24	12:26:20.425	Grupo de Hilos ...	template	193		772	445	193
25	12:26:20.525	Grupo de Hilos ...	template	180		772	445	180
26	12:26:20.624	Grupo de Hilos ...	template	196		772	445	196

Elaborado por: Los autores, a través de la herramienta Jmeter

En la Figura 39, se muestra un reporte de los resultados del módulo de templates por las 500 peticiones http indicando que el tiempo mínimo es 180 milisegundos, el tiempo máximo de 372 milisegundos y su media de 202 milisegundos, sin porcentaje de error con un tiempo de rendimiento de 10 segundos.

Figura 39. *Reporte de resultados módulo templates*



Nombre: Reporte

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error ↑	Rendimiento
template	500	191	181	204	227	426	168	555	0,00%	10,0/sec
Total	500	191	181	204	227	426	168	555	0,00%	10,0/sec

Elaborado por: Los autores, a través de la herramienta Jmeter

Con los resultados obtenidos se puede decir que, con las características del VPS actual, (30 GB de almacenamiento, 2GB de RAM y 1 núcleo) se puede tener hasta 500 peticiones http con un tiempo promedio de 202 milisegundos, si se necesita tener más peticiones, es necesario aumentar la capacidad del VPS.

CONCLUSIONES

- ✓ La solución desarrollada en este proyecto para la comunicación IoT entre hardware y software usando un protocolo MQTT permite tener el control sobre los clientes, tópicos y datos. Al tener la plataforma y el broker en un VPS, no se puede apreciar lentitud en la publicación y subscripción de mensajes. Adicionalmente se puede garantizar su disponibilidad permanente.
- ✓ Una buena selección de herramientas de desarrollo en base a criterios técnicos, garantizan un correcto desempeño de cualquier sistema, y son estas herramientas las que aseguran la factibilidad del proyecto.
- ✓ Lo más importante del desarrollo de esta plataforma fue determinar el protocolo IoT, para esto se hizo una matriz de priorización entre CoAP y MQTT, siendo estos los protocolos que batallan por ser el mejor para Internet de las Cosas. MQTT demostró ser más flexible en el modo de transmisión, ya que el broker EMQX está instalado en el VPS, permitiendo gestionar los datos publicados para los suscriptores, la comunicación es de manera inmediata al momento de ejecutar el script en el Raspberry pi.
- ✓ La solución desarrollada en este proyecto permite que la plataforma genere las credenciales MQTT, dichas credenciales cambian cada vez que se actualiza la página, siendo esta una medida de seguridad para evitar el acceso a usuarios mal intencionados. Los resultados de las pruebas realizadas demuestran que el desarrollo de este proyecto cumple con los objetivos planteados.

RECOMENDACIONES

- ✓ Antes de implementar la plataforma, se debe considerar un análisis sobre las características del VPS, tomando en cuenta los tipos de dispositivos con los que se va a trabajar y la información que se necesite guardar.

- ✓ La plataforma está diseñada para que los usuarios utilicen tres tipos de widgest, se recomienda realizar un diseño donde el usuario administrador pueda cargar diferentes widgest para que los usuarios puedan conectar otros dispositivos.

- ✓ Solo en el caso de necesitar seguridad en la comunicación con los dispositivos, se recomienda contratar la versión Enterprise de EMQX, ya que la versión gratuita permite la conexión de hasta 100000 dispositivos.

LISTA DE REFERENCIAS

Pelaez Lopez, D. (2020). *Full-Stack Web Development with Jakarta EE and Vue.js.* Medellin, Colombia: Apress.

Aguirregomezcorta, J. d. (Julio de 2020). *Repositorio UAM.* Obtenido de https://repositorio.uam.es/bitstream/handle/10486/692833/Lis_Aguirregomezcorta_juan de_tfg.pdf?sequence=1

Arias, A. S. (15 de Julio de 2014). *Economipedia*. Obtenido de https://economipedia.com/definiciones/tasa-interna-de-retorno-tir.html#referencia

Bell, C. (2016). *Windows 10 for the Internet of Things.* Warsaw, Virginia, USA: Apress.

Biswajeeban Mishra, A. K. (04 de Noviembre de 2020). The Use of MQTT in M2M and IoT Systems: A Survey. *IEEE Xplore*. doi:10.1109/ACCESS.2020.3035849

Boral, S. (21 de Mayo de 2021). *IoT Techtrends*. Obtenido de https://www.iottechtrends.com/best-iot-platforms/

Cárdenas, A. (28 de Noviembre de 2016). *secmotic*. Recuperado el 15 de Septiembre de 2020, de https://secmotic.com/plataforma-iot/

Cárdenas, A. (28 de Noviembre de 2016). *Secmotic*. Obtenido de https://secmotic.com/plataforma-iot/#gref

Cicolani, J. (2021). *Beginning Robotics with Raspberry Pi and Arduino* (Second Edition ed.). Pflugerville, TX, USA: Apress.

Contributors, M. (6 de Julio de 2021). *MDN Web Docs*. Obtenido de https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction

Cookson, R. (08 de Aug de 2019). *virtual-dba*. Obtenido de virtual-dba: https://www.virtual-dba.com/blog/pros-and-cons-of-mongodb/

Cortés, J. A. (Octubre de 2016). *Universidad Carlos III de Madrid.* Obtenido de https://e-archivo.uc3m.es/handle/10016/27437

Donat, W. (2018). *Learn Raspberry Pi Programming with Python.* Palmdale, California, USA: Apress.

EMQ. (2020). *EMQ Docs*. Obtenido de https://docs.emqx.io/en/broker/v4.3/development/python.html#paho-python-usage-example

EMQX. (6 de Noviembre de 2019). *EMQX*. Obtenido de https://emqx.medium.com/mqtt-broker-server-9b17cb8eaccb

EMQX. (2020). *Docs EMQX*. Obtenido de https://docs.emqx.io/en/broker/v4.3/advanced/auth.html#authentication-method

EMQX. (2020). *EMQX*. Obtenido de https://docs.emqx.io/en/broker/v4.3/

Guillen, G. (2019). *Sensor Projects with Raspberry Pi.* Ciudad de Mexico, Mexico: Apress.

Harrison, G., & Harrison, M. (2021). *MongoDB Performance Tuning.* Australia: Apress.

Heredia, J. C. (Septiembre de 2014). *Repositorio UPCT.* Obtenido de https://repositorio.upct.es/bitstream/handle/10317/4163/pfc5908.pdf;sequence=1

Hernandez, D., Mazon, B., & Escudero, C. (Junio de 2018). *researchgate*. Recuperado el 14 de Septiembre de 2020, de https://www.researchgate.net/publication/327702411_Capitulo_3_Internet_de_las_cosas_IoT

HiveMQ. (22 de Octubre de 2019). *hivemq*. Obtenido de hivemq: https://www.hivemq.com/blog/15-frequently-asked-mqtt-questions/

Hoffman, J. B., Heimes, P., & Senel, S. (11 de Octubre de 2018). *IEEE*. Obtenido de https://ieeexplore.ieee.org/document/8489905/citations#citations

Kurniawan, A. (2018). *Raspbian OS Programming with the Raspberry Pi.* Depok, Indonesia: Apress.

McClelland, C. (10 de Enero de 2020). *iotforall*. Recuperado el 14 de Septiembre de 2020, de https://www.iotforall.com/what-is-an-iot-platform/

Mishra, R. (31 de Octubre de 2020). *Geeksforgeeks*. Obtenido de https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/

Morales, V. V. (15 de Junio de 2014). *Economipedia*. Obtenido de https://economipedia.com/definiciones/valor-actual-neto.html#referencia

Noronha, T. (17 de Abril de 2018). *Resellerclub*. Obtenido de Resellerclub: https://blog.resellerclub.com/mongodb-vs-mysql-comparison/

Pickdata. (21 de Octubre de 2019). *Pickdata*. Obtenido de https://www.pickdata.net/es/noticias/mqtt-vs-coap-mejor-protocolo-iot

Rachmad, A., Riantini, R., & Hasin, M. (Mayo de 2017). *Researchgate*. Recuperado el 10 de Septiembre de 2020, de https://www.researchgate.net/publication/317391853_IoT_real_time_data_acquisition_using_MQTT_protocol

Rajiv, Y. (2018). *Developing Turn-Based Multiplayer Games.* Bangalore, Karnataka, India: Apress.

Rao, M. (2018). *Internet of Things with Raspberry Pi 3.* BIRMINGHAM - MUMBAI: Packt Publishing.

Rojas, C. (2020). *Building Native Web Components.* Medellin, Colombia: Apress.

Sáez, I. P. (07 de Febrero de 2019). *Incibe-cert*. Obtenido de https://www.incibe-cert.es/blog/iot-protocolos-comunicacion-ataques-y-recomendaciones

Segarra, F. (22 de Junio de 2021). *Hostinger*. Obtenido de https://www.hostinger.mx/tutoriales/que-es-un-vps

Shelby, Z. (28 de Junio de 2013). *Datatracker*. Obtenido de https://datatracker.ietf.org/doc/html/draft-ietf-core-coap-18#page-9

Sommerville, I. (2011). *Ingeniería de software.* Mexico: PEARSON EDUCACION. Obtenido de https://www.academia.edu/25063155/Ingenieria_de_Software_Somerville?email_work_card=view-paperhttps://www.academia.edu/25063155/Ingenieria_de_Software_Somerville?email_work_card=view-paper

Thinger. (2021). *Thinger*. Obtenido de https://pricing.thinger.io/#!/cloud

Tiam kok, L. (2020). *Hands-on Nuxt.js Web Development.* BIRMINGHAM - MUMBAI: Packt Publishing Ltd.

Vergara, S. (14 de Enero de 2020). *ITDO*. Obtenido de https://www.itdo.com/blog/que-modelo-de-base-de-datos-se-adapta-a-mi-proyecto/

Wang, K. (2018). *Systems Programming in Unix/Linux.* Pullman, WA, USA: Springer.