ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELECTRÍCA Y ELECTRÓNICA

“PROGRAMACIÓN DE UN ALGORITMO DE PROCESAMIENTO DIGITAL DE LA INFORMACIÓN OBTENIDA POR LOS SENSORES ULTRASÓNICOS PARA SU INTERPRETACIÓN Y ANÁLISIS”

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y AUTOMATIZACIÓN

ROBERTO ENRIQUE PINEDA CAZARES

rpineda.pineda@epn.edu.ec

DIRECTOR: DR. FAUSTO EDUARDO ÁVALOS CASCANTE

eduardo.avalos@epn.edu.ec

Quito, agosto 2022

CERTIFICACIONES

Yo, ROBERTO ENRIQUE PINEDA CAZARES declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Roberto Enrique Pineda Cazares

Certifico que el presente trabajo de integración curricular fue desarrollado por ROBERTO ENRIQUE PINEDA CAZARES, bajo mi supervisión.

Dr. EDUARDO ÁVALOS
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Sr. Roberto Enrique Pineda Cazares

Dr. FAUSTO EDUARDO ÁVALOS CASCANTE

DEDICATORIA

A mi familia: Erick, Consuelo, Roberto, su constante apoyo, ejemplo y voluntad me motivaron cada año lejos de casa, dedico este trabajo en agradecimiento.

AGRADECIMIENTO

A mis padres, Roberto y Consuelo que han puesto el hombro para apoyarme y guiarme constantemente.

A mi hermano Erick, que me ha brindado su amistad, lealtad y honestidad en cada momento de la vida.

A mis amigos por compartir momentos inolvidables que atesoro en mi memoria.

A mis compañeros de la universidad, por compartir no solamente un aula de clase sino una parte de su vida a mi lado y un sueño en común.

A los maestros de la Escuela Politécnica Nacional, que me transmitieron no solo su conocimiento sino la pasión por la ciencia.

ÍNDICE DE CONTENIDO

RESUMEN

El presente documento detalla el desarrollo de un algoritmo para el procesamiento de información de un sistema de adquisición-instrumental para censar la variable caudal usando transductores ultrasónicos, además del monitoreo de energía de un sistema de alimentación fotovoltaico; por lo que la red consta de dos nodos en la comunicación. Para la transmisión de esta información se utilizó interfaz serial-inalámbrica.

De esta manera se busca obtener los datos de cada nodo para filtrarlos, y procesar la información para obtener mediciones estables de caudal con un error mínimo de medición, la información que se requiere supervisar se retransmite a un servidor web desde el Gateway físico usando el módulo ESP8266.

Para el procesamiento se utiliza un procesador virtual alojado en la nube basado en la distribución de Ubuntu 20.04LTS, encargado de la recepción, filtrado, procesamiento, almacenamiento de información y presentación de servicio web de supervisión y control.

Además, la presentación de resultados se realiza en una interfaz construida en Flutter framework basado en lenguaje DART, así los resultados, tendencias en tiempo real y parámetros modificables que son accesibles desde un navegador o una aplicación móvil.

PALABRAS CLAVE: Ultrasónico, ESP8266, Ubuntu, Flutter, framework, DART.

ABSTRACT

This document details the development of an algorithm for the processing information of an instrument-acquisition system for flow census using ultrasonic transducers, in addition to the energy monitoring of a photovoltaic power system; so, the network consists of two nodes in the communication.

For the transmission of this information, a serial-wireless interface was used. In this way it is sought to obtain the data of each node to filter them and process the information to obtain stable flow measurements with a minimum measurement mistake, the information that is required to monitor is retransmitted to a web server from the physical Gateway using the ESP8266 module.

For the processing, a virtual processor hosted in the cloud based on the Ubuntu 20.04LTS distribution is used, in charge of receiving, filtering, processing, storing information and presenting the web service of supervision and control.

In addition, the presentation of results is done in an interface built on Flutter framework based on DART language as well as results, real-time trends and modifiable parameters that are accessible from a browser or a mobile application.

KEYWORDS: Photovoltaic, ESP8266, Ubuntu, Flutter, framework, DART.

1. INTRODUCCIÓN

Los transmisores usados en aplicaciones industriales poseen protocolos de comunicación que permiten el intercambio y recolección de la información de un sin número de variables de campo. [1]

La evolución de la tecnología ha permitido la integración de la información proveniente de diferentes equipos con la finalidad de supervisar el proceso de forma remota, con ello el desarrollo de comunicaciones inalámbricas como WiFi, Zigbee, LoRaWAN, Modbus sobre ethernet se ha popularizado en el mercado. Este proyecto tiene como finalidad desarrollar una solución inalámbrica que consiste en un Gateway para la presentación de información en un servidor web proveniente de dos nodos, el de una DAQ de un sensor ultrasónico de caudal y el de un sistema de alimentación fotovoltaico conectado al canal analógico y a la interfaz serial respectivamente. [1][2].

La necesidad de integrar mediciones de campo a los sistemas de adquisición y control existentes consiste en la ampliación de la red usando cobre o fibra óptica sin embargo para la red de área local las comunicaciones inalámbricas son una alternativa de bajo costo, de rápida integración debido a que los requisitos de latencia no son exigentes, y las distancias no suponen un problema, por lo cual en esta etapa se hará uso de la capa de enlace de manera inalámbrica. [3]

El bus de transmisión de información debe considerar los problemas como la atenuación de las señales, caída de tensión, ruido blanco gaussiano [4] lo que provoca un deterioro en la señal de comunicación por lo que se propone el uso dispositivo conectados en distancias cortas salvaguardando una velocidad de transmisión óptima. [5]

Para el procesamiento de información se requiere de tres etapas: el ordenamiento de datos usando un tipo compuesto llamado lista, el filtrado de información para eliminación de datos atípicos en relación con el conjunto de valores instantáneos aumentando la precisión de la medición por lo que se aplicara la regla de eliminación de rangos Inter cuartiles. Finalmente, la aplicación del criterio matemático a la información estandarizada para en base a los tiempos medidos por el sistema de adquisición – instrumental obtener el flujo de agua que recorre la tubería mediante la relación de tiempo de tránsito de transductores ultrasónicos.

El flujo bidireccional de información entre el servidor de datos, y el cliente [dashboard] que representa a un operador remoto, en conjunto con el dispositivo de campo o DAQ, se controla mediante el Gateway para lo cual se propone usar el módulo ESP8266, donde la implementación del código del servidor de datos se desarrollara en lenguaje Javascript usando el framework Node.js, mientras que la capa de datos y de presentación se desarrolla en el framework Flutter basado en programación en lenguaje DART.[6]

1.1 Objetivo general

El objetivo de la programación del algoritmo de procesamiento es realizar la implementación de un equipo controlador central de información Gateway inalámbrico que permite dar lectura a los datos del sistema de adquisición de medida y al sistema de alimentación fotovoltaico, una vez adquiridos se realiza una fase de ordenamiento y filtrado mediante métodos estadísticos con el fin de normalizar las lecturas, de tal manera que la información obtenida se encuentre dentro de un rango de trabajo determinando el fondo de escala de medición sin perder exactitud. Además, se realizará la presentación de información procesada usando una arquitectura back end - front end para la interacción dinámica con el usuario, haciendo uso de una aplicación híbrida tanto accesible desde un navegador web y también a través de una aplicación móvil.

El esquema físico de montaje para centralizar la información usando dos canales de lecturas de la DAQ se presenta en la Figura. 1.1

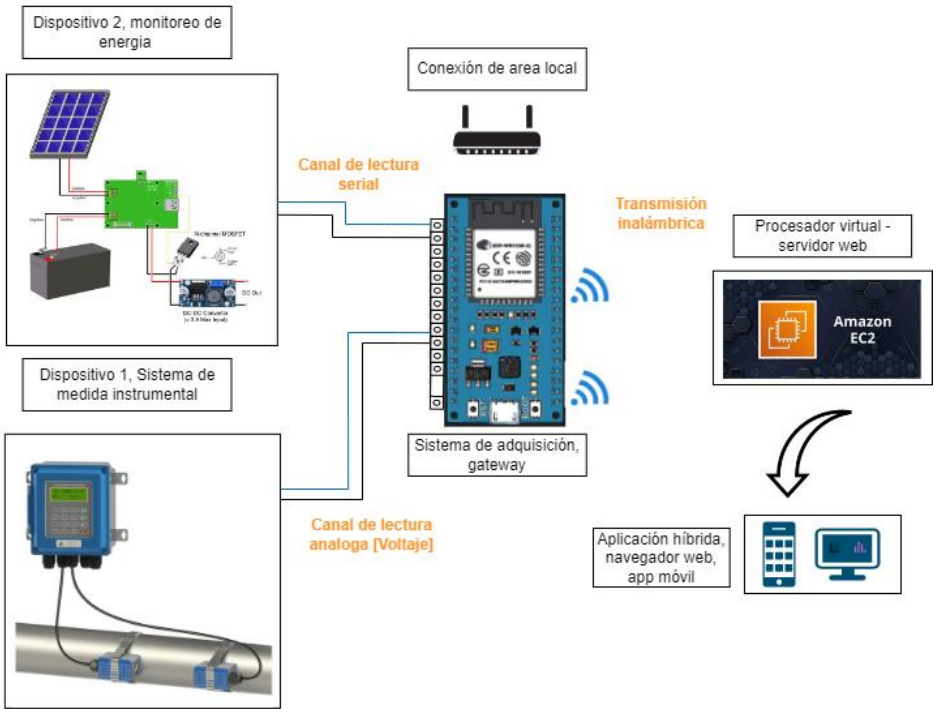


Figura 1.1. Arquitectura funcional del sistema montado en el bus de datos hacia el sistema de procesamiento [4].

1.2 Objetivos específicos.

1. Implementar un servicio web de recepción de valores de lectura desde el sistema de adquisición.

2. Analizar y estudiar las funcionalidades de un Gateway para alojar el servicio web de presentación de resultados.

3. Estudiar los distintos tipos de algoritmos de procesamiento digital de información, clasificarlos y buscar el más adecuado a la presente problemática.

4. Analizar herramientas y lenguajes de programación necesarios para la programación de algoritmos de procesamiento digital de información.

5. Implementar un algoritmo de procesamiento digital de información que satisfaga las necesidades del proyecto.

6. Construir la arquitectura de procesamiento y transmisión de información usando el modelo de capas para manejar las distintas lógicas por separado.

7. Interpretar los resultados obtenidos desde el algoritmo de procesamiento digital de información implementado.

8. Analizar la validez de los resultados obtenidos por medio de la interfaz construida en el servidor web.

1.3 Alcance

El proyecto tuvo como alcance considerar las conexiones de dos tipos de canales para la adquisición de datos, uno de ellos es el acondicionamiento de una entrada analógica de voltaje, y otra de recepción serial asincrónica en un prototipo Gateway inalámbrico. El Gateway permite la interacción del usuario con los valores y tendencias en tiempo real de los dos nodos ocupados por los dispositivos de forma dinámica, el almacenamiento de la información se realiza en un sistema de base de datos orientado a objetos denominado Mongo DB.

Se trasmitirá la gestión de información entre la base de datos y la aplicación web usando un marco de trabajo original de NodeJS el cual se conoce como ExpressJs que permite el desarrollo de aplicaciones en lenguaje Javascript.

Se realizará el análisis de las referencias bibliográficas a fin de fundamentar tanto los programas a implementarse como los requerimientos de software y hardware para lograrlo.

Se realizará una síntesis bibliográfica acerca del algoritmo de filtrado y el ordenamiento de información a usarse, las comunicaciones industriales de dispositivos y sus características, la tecnología usada en el hardware de procesamiento, y el sistema operativo a ejecutarse cuya compatibilidad puede acoplarse con los servidores web.

Se planteará los requerimientos que debe poseer el sistema de procesamiento de información basándose en las multitareas que ejecutará a la vez, tanto monitoreo, como desencriptación, filtrado, modelado, sincronización, y alojamiento del servidor e interfaz, manejo de comunicaciones inalámbricas; comparando placas de desarrollo, sistemas embebidos y procesadores comerciales.

Se seleccionará los lenguajes de programación que permitan el desarrollo del servidor web y/o aplicación móvil en la que se aloje la interfaz a manera de Dashboard para la presentación de resultados permitiendo la supervisión remota del funcionamiento.

Se realizará la programación en la cual se desarrolle el apilamiento de datos, documentos u objetos; se implemente el algoritmo estadístico de clasificación, la evolución del sistema de adquisición de medida de caudal y velocidad superficial del agua, y el envío de información hacia el servidor, además se elaboró pruebas de funcionamiento del sistema de medición para verificar que se obtenga los valores de caudal reales calibrados con un instrumento patrón, de tal forma que además de contar con una plataforma para el monitoreo inalámbrico y local, las lecturas reflejen valores con un bajo umbral de error.

1.4 Marco teórico

1.5 Topología

Se denomina topología de red a la forma en que se encuentra distribuidos físicamente los nodos o componentes de automatización (sensores, actuadores, PLCs, etc.) para su comunicación y el intercambio de datos entre ellos. Los nodos que componen una red son el principal y los secundarios operando de manera subordinada con el primero. Por lo que la influencia de la topología debe considerar la eficiencia de la red ya que está relacionada directamente con el flujo de información (velocidad de transmisión, tiempos y latencias de llegada), y la capacidad de extenderse a un mayor número de dispositivos. [8][9]

La red puede montarse de forma simplificada con dos hilos reduciendo de forma considerable las conexiones, esta configuración es útil cuando se requiere comunicación half duplex y es ampliamente usada en conversores ADC debido a que se usa una referencia en común y un pin con el respectivo acople de impedancias además de un filtro de salida y llegada para mitigar los efectos de ruido, adicionalmente la comunicación serie USART ofrece la opción de manejar interfaces estándares alternativos que se pueden cablear mediante módulos existentes como la norma RS-485 en par diferencial o RS232.[10][11][12][13]

1.6 Gateway [Puerta de enlace]

El Gateway es un dispositivo que cumple la función de habilitar la conectividad entre redes con arquitecturas distintas como lo son los dispositivos de campo con interfaces como USART, SPI, UP_AXI, RS232, RS485, RS422, TTY, Ethernet [10] con los clientes locales y remotos cuyo acceso se realiza desde la nube. En resumen, el Gateway cumple la misión de convertir los protocolos de comunicación modificando el empaquetado de información de origen de cada nivel inferior tomando como referencia al modelo OSI (Figura 1.2) ya que trabaja en el nivel de aplicación de este convirtiéndose en un intermediario para la comunicación remota de transmisores y otros dispositivos que se encuentran en campo con el sistema Supervisorio de control [14][15].

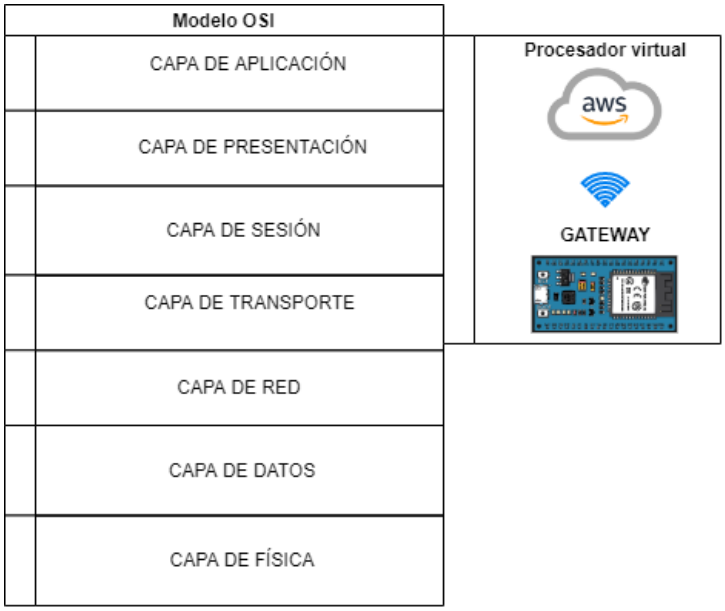


Figura 1.2. Niveles de modelo OSI donde el dispositivo host aloja el Gateway.

1.7 Tecnología WIFI

El uso de la tecnología WiFi proveniente del estándar IEEE 802.11 para redes inalámbricas de área local [WLAN] [13] se ha popularizado por la gran cantidad de módulos y dispositivos existentes que brindan un uso sencillo y se encuentran disponibles en el mercado [2].

La tercera revolución industrial ha llevado de la mano el acceso inalámbrico a aplicaciones cotidianas como negocios, educación, transporte, así como el avance e interacción con tecnología y hardware inteligente. Las tareas de monitoreo se encuentran incluidas en el proceso de optimización de recursos y eficiencia de procesos, debido a que la capacidad computacional ha crecido de la mano de manera exponencial se ha vuelto posible la implementación de tareas de comunicación que requieren velocidad y conexión de dispositivos en redes inalámbricas [17].

Las ventajas que presenta destacan en la capacidad para transferir datos de gran tamaño mientras que la velocidad de transmisión depende del ancho de banda de la conexión de área local y la distancia entre los dispositivos a usarse.

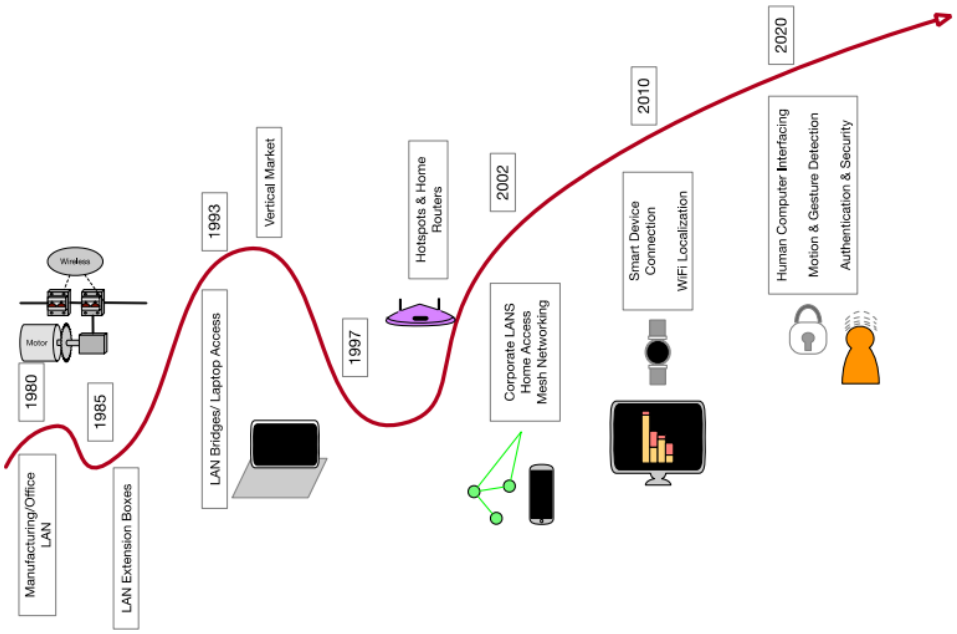


Figura 1.3 Aplicaciones del WiFi y su evolución a lo largo de los años

La pasarela de acceso en conjunto con WiFi se denomina WAG, se encarga de proporcionar enrutamiento de paquetes desde una red cableada a una red inalámbrica, esto es posible mediante la configuración únicamente de software o hardware o una combinación funcional de ambos. [17][18]

1.8 Lenguajes de programación y Marcos de Trabajo

1.8.1. Lenguaje de programación Javascript

En la capa de procesamiento se cumple dos tareas en particular, el acceso a los datos a través de consultas y la manipulación de la información [transformación, validación, normalización y operaciones particulares] [21][22]. Los dispositivos de los cuales proviene la información se encuentran enviando datos en cuanto su solicitud es reconocida generando un volumen de información por lo que se requiere una capacidad de escalabilidad considerable para el análisis o procesamiento en tiempo real.

Uno de los lenguajes multiparadigma más populares para el manejo de datos es Javascript por las herramientas que ofrecen las librerías, este lenguaje de alto nivel es óptimo para la interpretación de patrones de datos complejos.[25].

Una de sus bondades es la ejecución en entornos libres y multiplataforma. La metodología de trabajo en la capa de procesamiento se basa en una computación

estadística para información empaquetada enfatizando en la legibilidad del código con una sintaxis muy clara lo que agiliza las tareas de desarrollo web backend.[26][27].

1.8.2. Framework NodeJs

En la capa de datos se usa un entorno de ejecución (NodeJs). El entorno NodeJs nace con la necesidad de admitir una ejecución prolongada dentro de un servidor de eventos escritos en lenguaje de programación Javascript, las funciones que se proporcionan son la lectura de información desde la red, acceso a bases de datos, útiles para la creación de diferentes tipos de aplicaciones web en tiempo real.[28] En resumen Node.js trabaja como un entorno de tiempo de ejecución para la gestión de entrada y salida en la parte del servidor empleando web sockets sin bloqueo permaneciendo ligero y eficiente en el desarrollo de aplicaciones de servicio escalables utilizando Javascript. [29] [30].

1.8.3. Framework ExpressJs

La capa de negocio es la encargada de manejar la lógica para que los datos que se han calculado cumplan los requerimientos que maneja la base de datos previos a ser enviados al servidor.[31] El framework utilizado se conoce ExpressJs proporciona utilidades para desarrollo web aceptando múltiples peticiones HTTP y respondiéndolas (enrutamiento) de url específicas, además de la configuración del puerto de escucha y la más importante el preprocesamiento de las solicitudes que consiste en desestructurar una petición para la gestión de errores, alcanzando resultados eficientes en la programación de APIs.[31][32]

1.8.4. Base de Datos MongoDB

En base a la compatibilidad de los marcos de trabajo de Javascript con los controladores que proporcionan la interacción con las bases de datos se encuentra MongoDB, base de datos NoSQL orientada a trabajar con objetos, los objetos en particular se denominan documentos y son conformados de un campo y un valor. Los documentos se manejan en una colección de datos relacional, satisfaciendo la consistencia u disponibilidad de información en cada instante de tiempo que se requiera (Figura 1.4) [33][34][35].

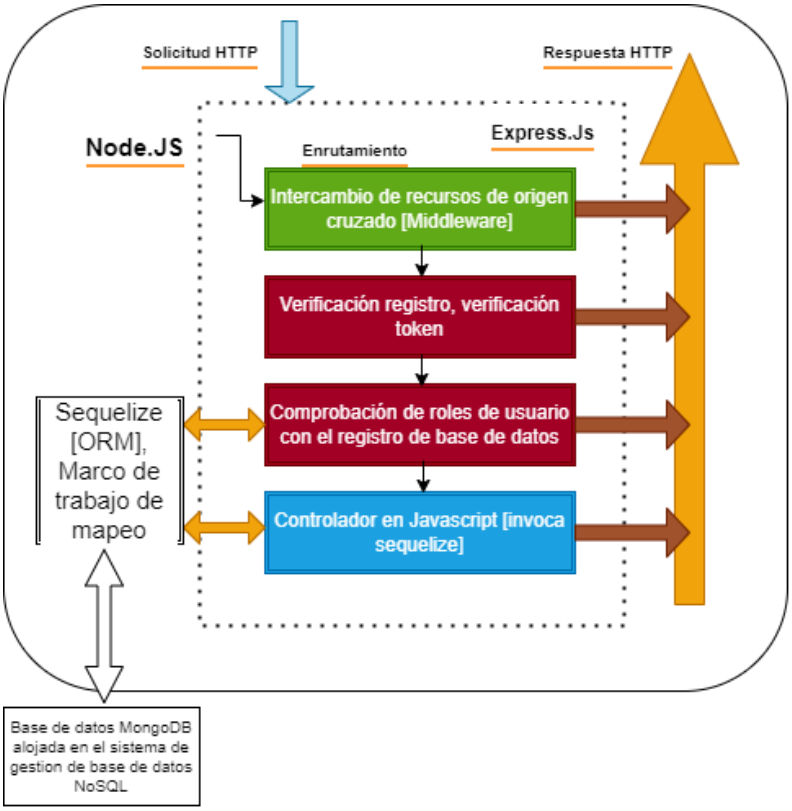


Figura 1.4 Descripción general de procesado en aplicación ExpressJs en la capa de negocio. [34]

1.8.5. Framework Flutter

Relativo al desarrollo en la capa de presentación, es un framework para desarrollo de interfaces web de usuario que utiliza lenguaje de programación DART, ofrece un kit de herramientas de alto rendimiento “nativo” para la creación de aplicaciones web, escritorio y aplicaciones móviles Android. Una aplicación de código nativo tiene la capacidad de definir el tipo de reacción al que los objetos reacción a una solicitud especifican de una o más plataformas a través del propio motor de flutter. Se caracteriza por optimizar la renderización hacia la aplicación evitando cuellos de botella entre los distintos frameworks. [36]

Este marco de trabajo además se considera declarativo, donde según el código se asigna un cierto comportamiento a la interfaz si se ha producido un estado. Es decir, en lugar de recurrir a un ID de un objeto en un cierto momento dada una acción en particular, en flutter se declara la acción que se va a desplegar en un momento, esta estructura se considera método de construcción, cada widget se encuentra anidado ya que es diferente en su estructura [37][38].

Se muestra una aplicación visualmente anidada de objetos en la figura 1.5:

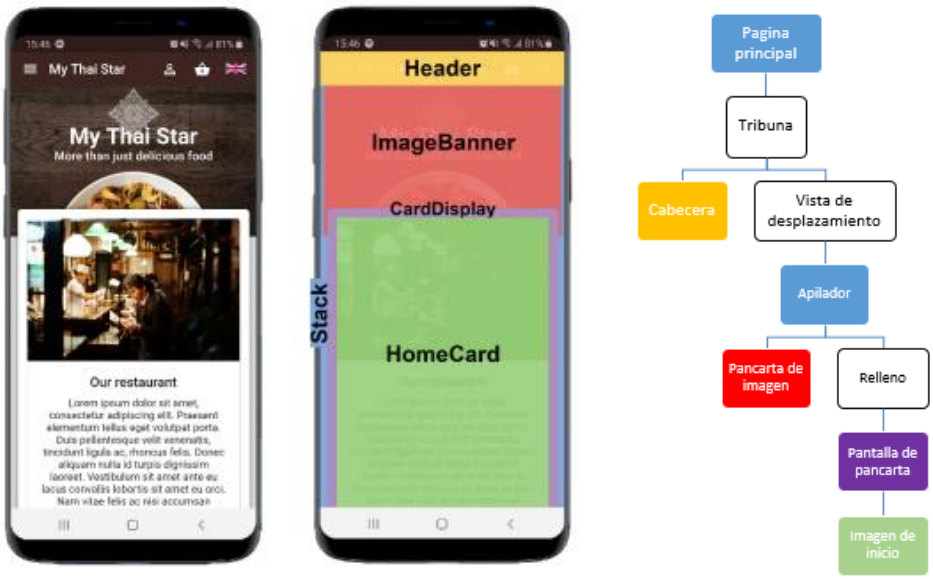


Figura 1.5. Plantilla construida con Flutter como interfaz de usuario con árbol anidado

1.8.6. Lenguaje de Programación DART

El lenguaje de programación DART fue desarrollado por Google en 2011 es un lenguaje de código abierto, se orienta a la programación de objetos y la sintaxis se asemeja a programación en C y tiene la capacidad de ocupar poca memoria admitiendo genéricos tipo cosificados, clases abstractas o interfaces optimizando los tiempos de ejecución. Se caracteriza por priorizar el lado de cliente o front end por lo que se evita uso de bases de datos relacionales reduciendo la sobrecarga entre la capa de negocio y la interfaz [38][41].

1.9 Arquitectura de capas

La arquitectura de capas se usa con enfoque a programación orientada a objetos, cuya característica define niveles de abstracción o capas de tal forma que la estructura del sistema sea funcional con los servicios que ofrece. La idea de proponer un orden jerárquico horizontal para atender a distintas responsabilidades se describe a profundidad en lo que ayuda a realizar cambios en cada una sin alterar el código en la capa de un nivel superior o inferior logrando tener una independencia unidireccional [39][40].

Tabla 1.1. Modelo de capas y su descripción

Capas de la arquitectura	Descripción de la capa
Capa de presentación	Referente al desarrollo de la interfaz para su funcionalidad de interacción entre el usuario y el sistema, tiene un grado de interacción designado para cada función y objeto. El principal objetivo es la presentación de información, no toma parte o toma mínima parte en la lógica de negocio volviéndola totalmente dependiente de los estados de esta última.
Capa de negocio	En esta capa intermedia se definen los parámetros para la ejecución de los recursos necesarios solicitados por el cliente, es decir la entrega y recepción de datos, los cambios de estado se encuentran definidos y únicamente comunicados con los datos de forma independiente a través de una plataforma anclada a esta capa. [40]
Capa de datos	Se definen las respuestas específicas dadas las consultas que puede realizar el usuario, es decir permite dar consistencia a la información administrable que se le ofrecerá al usuario.
Capa de procesamiento de información - API	Realiza el procesamiento de la información entregada por los esclavos del bus de datos o de las interfaces de la capa física mediante las etapas de procesamiento, depende de servicios externos

Las capas cumplen la función de gestionar las solicitudes atendiendo la capa superior, generar peticiones hasta la capa inferior y controlar el resultado de cada una de las peticiones. Lo que ayuda a su fácil manipulación y portabilidad. [36]. Son cuatro capas las que componen la arquitectura, la capa de presentación, la capa lógica, y la capa de datos, de manera que el desarrollo del programa se realice ordenadamente y con la opción de actualizarlo dependiendo de las necesidades presentes en el camino.

1.10 REST API

El término API proviene de la contracción de “interfaz de programación de aplicaciones” cuyo criterio se basa en una agrupación de procedimiento y funciones que realizan tareas para ser utilizadas por otro programa informático evitando la reprogramación en este último. [43].

REST por su parte se define como una arquitectura que ofrece interoperabilidad entre sistemas informáticos cuyo conjunto de operaciones se representan mediante identificadores uniformes de recursos [URL], en programación los datos u objetos pueden encontrarse en formatos JSON y en formato XML. De esta forma el protocolo más común

dentro de este modo de trabajo es HTTP con una escalabilidad y confiabilidad para servicios web ligeros y no se mantiene rígida a una plataforma específica como tal [44] [45].

Por lo tanto, se busca proporcionar información a un usuario o agente externo a cierta información de un dispositivo y REST hace posible construir un servicio compatible con una operabilidad sencilla para aplicaciones, así las operaciones se ajustan por el protocolo HTTP. En la siguiente Figura. 1.6. Se encuentra cada componente de una API en forma general, se esquematiza por etapas, en primer lugar, el cliente o usuario externo solicita información de forma automática o manual que puede provenir de uno o más dispositivos, pasando a la segunda etapa donde los datos se extraen desde el servidor que su vez desemboca en la etapa de solicitudes desde el sistema de adquisición o almacenamiento de información de el/los dispositivos. Por último, la cuarta etapa consiste en el procesamiento una trama o conjunto de datos para gestionar su óptima transferencia devolviendo los valores procesados desde la API REST al cliente [45] [46].

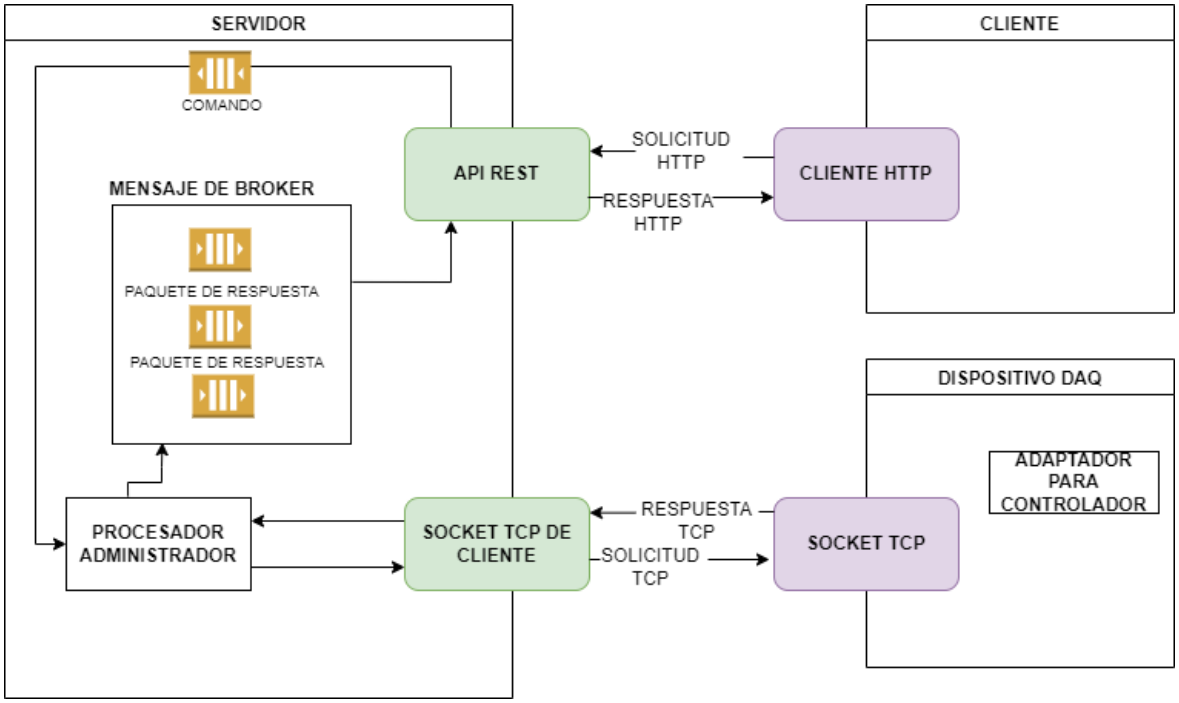


Figura 1.6. Esquema de componentes de REST API para administración de dispositivos.

1.11 Cloud Computing

Se define como un conjunto de servicios computacionales como redes, servicios, aplicaciones, servidores, y almacenamiento extensible, que proveen una interacción y aprovisionamiento rápido con mínimo esfuerzo en cuanto a capacidades TI mediante el uso de internet bajo el costo de consumo que se requiera. [5]

Por lo tanto se define como computación en la nube haciendo alusión a una red subyacente que se encuentra operando por medio de internet, este servicio además se caracteriza en primer lugar por el coste por tiempo de acuerdo a una cantidad de servicio (alquiler) cuya gestión la realiza el proveedor, también el acceso a la información no requiere de la ubicación física del sistema que aloja el servicio web, por ultimo cabe recalcar que los servicios pueden ser públicos, privados, híbridos pero de manera generalista todos poseen servicios de mantenimiento de aplicaciones, arquitecturas de varios usuarios o escalables, e infraestructura informática la cual se aloja en una misma organización. [46] [47]

1.11.1 Servidor Virtual

El servidor virtual hace alusión a un servidor que ha sido creado puramente por software o que se presenta como una versión virtual, lo que hace posible prescindir de la existencia física de un servidor, sin embargo, tiene las mismas características y atributos que el equipo físico. La principal característica es ofrecer recursos propios de un procesador como CPU, memoria RAM, memoria ROM y un entorno de software de un servidor compartidos exclusivamente por uno o más clientes, es decir que las reservas de espacio se encuentran idealmente distribuidas de acuerdo con el alquiler por necesidad. [47]

1.11.2 Tecnología Verde

Acorde a una disminución de recursos computacionales físicos se obtiene inversamente un aumento de eficiencia de estos contribuyendo a reducir el impacto producido en el medio ambiente. El cloud computing entre sus efectos colaterales ha reducido el consumo energético en centros de datos y en su aplicación en empresas como Google, OASIS, W3C, abaratando costes de mantenimiento, dinámica de red y gestión de información. [46]

1.11.3 Amazon Elastic Compute Cloud [AWS EC2]

Se define como un proveedor que ofrece la capacidad informática escalable en base a servicios web de Amazon en la nube. Como cualquier proveedor elimina la necesidad de equipamiento físico o hardware para la gestión de información y despliegue de aplicaciones, EC2 soporta el lanzamiento de tantas máquinas virtuales como sea necesario con estándares de seguridad y administración sencilla de la red de almacenamiento además de distribuir la carga de información entre distintas maquinas; Otra de las bondades que posee es poder ejecutar cualquier software dentro de cualquier máquina virtual montando la imagen ISO a elección del usuario por lo que para desarrolladores se

opta por el uso de Ubuntu o Solaris por su entorno de desarrollo en código libre mediante IDEs de distinto tipo, ejecutando en paralelo un software para administración de bases de datos [MongoDB, MySQL, Oracle, etc.] [48][52]

Figura 1.7. Logotipo de instancia EC2 de servicios de Amazon.

1.11.4 Simple Storage Service [S3]

En el año de 2006 Amazon lanzó el servicio S3 debido a las necesidades de los desarrolladores que buscaban compatibilidad con la plataforma y almacenamiento en línea. Las llamadas web a AWS se realizan mediante interfaces tipo REST o SOAP que para aplicaciones de toda clase son compatibles con sistemas que requieren únicamente el uso de un dispositivo en el lado del cliente. Para intercambio de datos se usa el protocolo estándar HTTP y se agrega comprobaciones periódicas en la integridad de datos, las tres prioridades de S3 se basan en almacenar, recuperar, balancear, y direccionar un volumen considerable de datos en cualquier momento desde la web [48][49].

1.12 Estructura Queue de datos

La manipulación de los elementos dentro de una lista puede generar dos tipos de estructuras diferentes FIFO y LIFO, estas únicamente difieren en el orden en que se va recuperando la información de entrada, en este caso se analizará el formato FIFO. [55] Haciendo alusión a una fila de personas en espera de un servicio de boletería, la primera persona que inicio la fila va a ser la primera persona que alcanza el primer boleto. [56] La representación FIFO se puede ejemplificar con un elemento entrando en el queue (cola) y saliendo en primer lugar como se observa en la Figura. 1.8

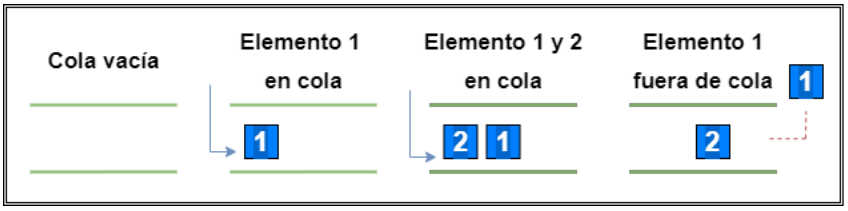


Figura 1.8. Representación de estructura FIFO en una cola.

En lenguaje de programación, la estructura descrita anteriormente precisa de adición y eliminación de elementos bajo los términos de enqueue y dequeue respectivamente. Para enfocar el uso de estas dos operaciones en Python y Javascript se utilizan dos punteros Front y Rear. El puntero Front es el encargado de realizar el seguimiento del primer elemento que se encuentra en la cola mientras que Rear se encarga del tracking del último elemento de la cola.

1.12.1 Lógica para estructura Queue

En un inicio tanto Front como Rear se disponen con valor de -1. (Figura 1.9)

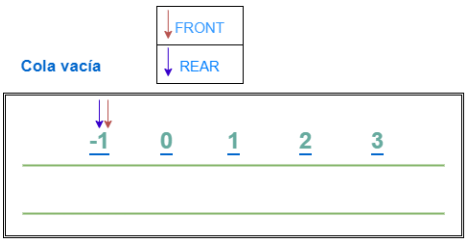


Figura 1.9. Establecimiento inicial de punteros Front y Rear en un Queue.

Después de la verificación de si la lista o queue se encuentra vacío y debido a que el puntero Front se encarga de controlar el posicionamiento del primer elemento, se establece su valor en 0 dentro de la cola (Figura 1.10). [55]

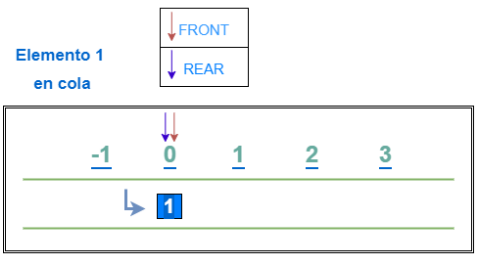


Figura 1.10. Control del primer elemento dentro de la estructura Queue

Para el tracking del último elemento se aumenta el valor del índice del puntero Rear en una unidad a medida que ingresa un nuevo elemento en la cola. (Figura 1.11)

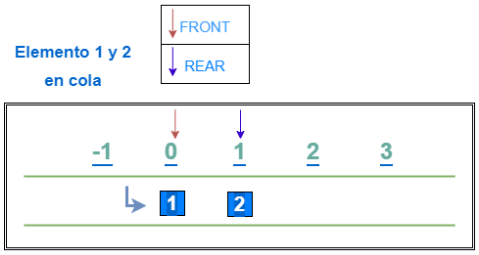


Figura 1.11. Primer tracking del último elemento en la estructura Queue

En el ejemplo presentado se han usado un total de cuatro elementos, por lo que el seguimiento del último elemento resulta en un aumento de 3 unidades para el índice de Rear, el elemento que ocupaba la posición en Rear para un valor “n-1” pasa a ocupar la posición “n”. (Figura 1.12)

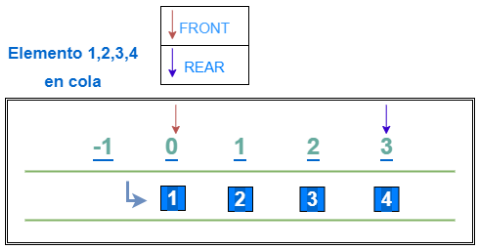


Figura 1.12. Primer tracking del último elemento en la estructura Queue

Para vaciar la estructura Queue se realiza una comprobación adicional del estado de la cola, vacío o lleno. Sí la cola está llena se accede al índice fijo de posición “n” que señala al elemento que ocupa la posición frontal en la cola y se lo extrae, posteriormente se aumenta el índice del puntero Front en una unidad. (Figura 1.13)

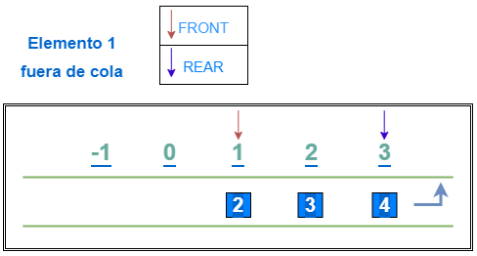


Figura 1.13. Extracción del elemento en la posición frontal en un Queue previamente llenado.

Una vez que el valor de los índices de los punteros Front y Rear son iguales al valor “n”, indicará que se extrajo todos los valores “encolados” en la estructura Queue. (Figura 1.14)

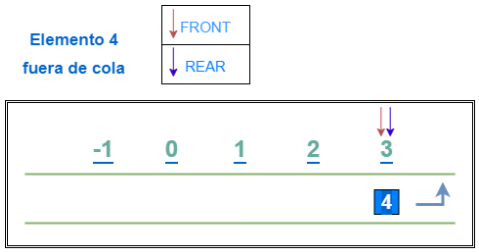


Figura 1.14. Extracción del último elemento del Queue

La salida al retorno se realiza inmediatamente al paso anterior, donde los valores de los índices de ambos punteros, Rear y Front se reestablecen en -1. (Figura 1.15)

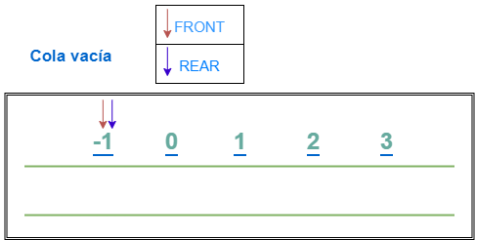


Figura 1.15. Salida de retorno en el vaciado de la estructura Queue.

1.13 Técnica de filtrado de información

La existencia de valores irregulares altera las conclusiones o análisis de información, debido a que los datos dentro de un conjunto pueden tener características robustas o sensibles dependiendo del origen que causa la anomalía o falla de precisión, por lo que la identificación y ajuste de esta sensibilidad permite decidir si se considera uno o más datos anómalos en las consideraciones posteriores.

Cuando en un conjunto determinado de datos existen elementos que desvían su valor cambiando la consistencia de dichos elementos se denominan outliers (valores atípicos). La razón por la que se produce la existencia de un outlier responde a la toma de medidas incorrectas, o a su vez originadas por una población de datos en circunstancias diferentes a la mayoría de los elementos, produciéndose un aumento en la varianza del error y desproporcionando la potencia de análisis estadísticos. Para sustentar y compensar los errores por outliers se desarrollan algoritmos de detección mediante métodos dinámicos de estructuras de datos.

1.13.1 Tipos de valores atípicos

Se clasifican en 4 tipos diferentes de outliers [50] como se explica en la siguiente tabla:

Tabla 1.2. Clasificaciones de outliers, consideraciones en la influencia sobre un espacio muestral

Tipo de valor atípico	Descripción	Consideraciones
Por errores en el procedimiento [error sistemático]	Producidos por errores en el sistema de procesamiento, y en la entrada de valores de un sistema de medida [sistema instrumental]. Se rige a patrones no aleatorios.	La desviación de información de este tipo se subsana añadiendo una etapa de filtrado de datos, de no ser posible requiere la eliminación del conjunto de valores.
Por acontecimientos aleatorios, fuera de lo común	No se encuentra dentro del conjunto valido de datos e información.	Se elimina de forma directa del análisis.
Únicas dentro del conjunto de valores	Aunque esta medida se encuentre dentro de los valores válidos, no tienen la misma naturaleza de la muestra filtrada.	Se recomienda mantener almacenados los datos singulares para estudiar las consecuencias sobre el proceso, o a su vez estimar el modelo que representa.
Por un factor faltante	La información no tiene una procedencia metodológica, por lo que requiere un análisis empírico de su validez. Esto en consecuencia del alcance del proyecto.	Requiere la realización de reportes que contrasten los resultados cuando las observaciones se toman y no en cuenta.

1.13.2 Caracterización de valores atípicos

La distribución de valores adquiridos o medidos se debe separar de forma independiente, es decir una distribución por cada variable. Estos rangos de variables independientes poseen valores atípicos o fuera de un umbral de tolerancia. La asignación del umbral se realiza numéricamente de acuerdo con puntuaciones determinadas. Se sugiere para un espacio muestral de hasta 80 datos un umbral de hasta 2.5 o menos, si el tamaño del espacio es mayor el umbral recomendado es de 3 [51] [53].

1.13.3 Diagrama Boxplot

Conocido como test de Turkey [53], es un método sencillo de detección de valores atípicos basado en cuartiles. Se toma como referencia un rango intercuartílico el cual es la diferencia existente entre el tercer cuartil y el primer cuartil. Para esquematizar el método se tiene una representación en la Fig. 1.16 compuesta de cuatro elementos, un rectángulo, una caja, un par de brazos, y dos ramas.

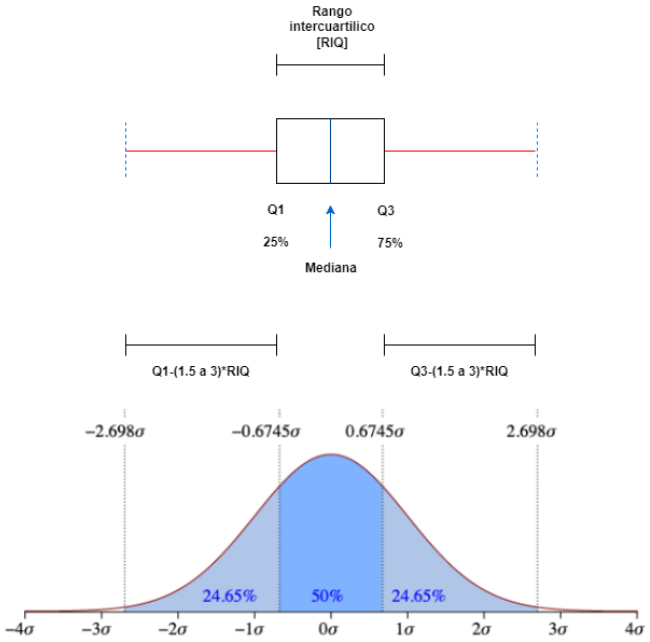


Figura 1.16. Metodología para la detección de valores atípicos Boxplot.

Este método considera la variación de un umbral de entre 1.5 a 3 veces el valor de los cuartiles Q1 y Q3 para ser considerado valor atípico, dada la naturaleza simétrica de población de forma normal. Así se consigue medias robustas para un valor estimado.

$$\hat{\mu}_n = \frac{(Q_1+Q_3)}{2} \tag{1.1}$$

$$\widehat{\sigma_n} = Q_3 - Q_1 \tag{1.2}$$

Donde:

Q_1: Primer cuartil.

Q_3: Tercer cuartil.

$\widehat{\sigma_n}$: Rango intercuartílico.

$\hat{\mu}_n$: Mediana de rango intercuartílico.

Se puede mantener un umbral único, así como dos umbrales para considerar dos tipos de valores atípicos, tanto leves asociados con un umbral de 1.5 como extremos asociados con un umbral de 3. Todo valor que se encuentra fuera de dicho umbral se considera un valor atípico en potencia por lo que no influencia en los resultados por la dependencia de la mediana. [50][53]. Ahora bien, aplicando el concepto de la detección de outliers en primera instancia se puede agregar una herramienta de comparación adicional para un conjunto de n datos univariante, denominada contraste de Grubbs [54].

1.13.4 El contraste de Grubbs

Este método compara a la desviación existente entre un valor perteneciente al conjunto de datos y la media muestral con la desviación estándar de la muestra.

$$z = \frac{|\ \text{Dato[i]} - \bar{x}|}{\sigma} \tag{1.3}$$

Con los valores críticos considerados como límites de z en la sección 2.10.3:

Extremos:

$$(Q_1 - 3\widehat{\sigma_n}, Q_3 + 3\widehat{\sigma_n}) \tag{1.4}$$

Leves:

$$(Q_1 - 1.5\widehat{\sigma_n}, Q_3 + 1.5\widehat{\sigma_n}) \tag{1.5}$$

Se considera que, si el valor de z es mayor al valor de outlier extremo, entonces el valor es un outlier en potencia. Se considera que esta metodología posee un 95[%] de confianza de aceptación o rechazo de un valor atípico. [54]

1.14 Geometría del sistema instrumental

En esta sección se describe un análisis de deducción del modelo matemático que describe la presión acústica de una onda en un medio de tipo fluido líquido confinado en una tubería cilíndrica. [57] Los caudalímetros de tipo ultrasónicos basan la medición en la transmisión de ondas a través del fluido, una de ellas es el tiempo de tránsito de la señal. El principio de medición del caudalímetro ultrasónico se rige por dos características, la disposición física y el número de transductores, en este caso se ha usado en la parte instrumental un receptor y un emisor cuya disposición se encuentra determinada en un eje axial de la tubería con un ángulo de vista [59] (Figura 1.17). La señal ultrasónica es enviada desde el emisor en forma de una cantidad de pulsos en contra del flujo del agua con velocidad v, el receptor recibe los pulsos y dependiendo del número de veces que la señal viaje o se envíe desde el emisor, se realizará el cálculo del tiempo de vuelo o frecuencia de vuelo. [60][58]

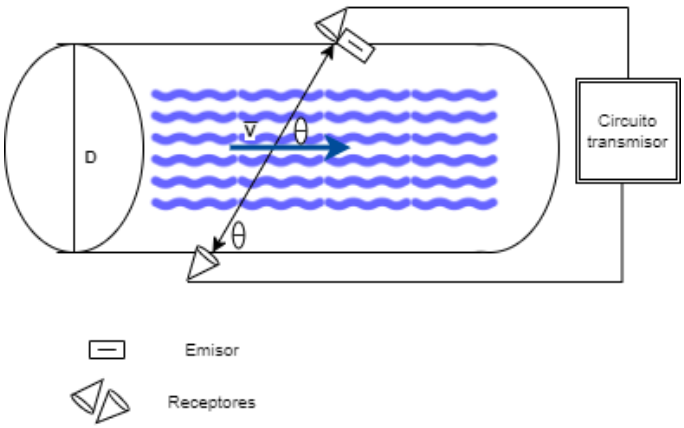


Figura 1.17. Principio de medición, sistema de transducción

El tiempo de vuelo de la señal ultrasónica circulante aguas abajo desde el transmisor se determina por la relación matemática que involucra una tubería circular de diámetro D y tomando en cuenta la velocidad del sonido en el agua c como una constante.

$$t_1 = \frac{D/\sin\,\theta}{c+v\cos\,\theta} \tag{1.6}$$

Mientras que el tiempo de circulación de la onda en contracorriente obedece la relación:

$$t_2 = \frac{D/\sin\,\theta}{c-v\cos\,\theta} \tag{1.7}$$

El ángulo entre el receptor y el emisor debe encontrarse en línea de vista, lo que se refiere a que el ángulo de posición entre el emisor y el receptor determinara la mejor propagación de la señal ultrasónica, este ángulo debe ser verificado de forma experimental. [58]. A continuación, se tiene la variación del tiempo dado por la diferencia entre t_1 y t_2

$$\Delta t = t_1 - t_2 = \frac{2Dv\cos\theta}{\sin\theta(c^2-v^2\cos^2\theta)} \tag{1.8}$$

Para evitar la supresión del término $v^2\cos^2\,\theta$, y guardar la exactitud en el modelo de medida, obtenemos que la velocidad del fluido se determina por:

$$v = \frac{(c^2-v^2\cos^2\theta)\tan\,\theta}{2D\cos\,\theta}\Delta t \tag{1.9}$$

Se puede apreciar que se trata de una ecuación de segundo orden respecto a la variable v, usando la relación para resolver ecuaciones de dicho orden determinamos que la velocidad en función del tiempo de vuelo corresponde a:

$$v = \frac{-\mathrm{D}\mp\sqrt{D^2-c^2\Delta t^2\,\mathrm{sen}\,\theta}}{\Delta\mathrm{t}\cos\,\theta} \tag{1.10}$$

Y atendiendo a las relaciones de dinámica de fluidos relacionamos a la velocidad calculada con el caudal [58]:

$$Q = K_1 \times S \times v = K_1 \frac{\pi D^2}{4} \quad (1.11)$$

Con:

$$K_1 = \frac{2n}{2n+1} \quad (1.12)$$

Donde: n= representa el número de Reynolds y S es la sección transversal de la tubería

1.15 Servidores Web

La utilidad de un servidor web reside en su capacidad de alojar un sitio web permitiendo en la industria el acceso de sistemas de monitorización y control usando conexión a internet. Este tipo de servicios web se conocen como servicios de propósito cuyos requisitos residen en contar con un sistema operativo que soporte la página web dedicada o una aplicación móvil, un hardware que soporte la memoria requerida por la base de datos y a su vez que sea capaz de gestionar la comunicación del servidor. [61]

Las aplicaciones web utilizan la arquitectura cliente-servidor en donde el cliente reside en un navegador o página web cuyo protocolo es HTTP [Protocolo de transferencia de hipertexto] [63]. El cual se encuentra dentro de los protocolos TCP-IP. Este protocolo permite la comunicación entre dispositivos que únicamente deben cumplir el requisito de tener conexión a internet sin importar el hardware o software instalado que posean. El servidor web constituye un programa que se mantiene a la escucha de las solicitudes de los clientes mientras que el cliente es un programa o navegador con el que el usuario solicita cierta información y la obtiene por medio de HTTP como una transferencia de recursos. [64]

Las aplicaciones se clasifican en tres grandes grupos tanto para ordenadores como para dispositivos móviles:

Aplicaciones web

Estas se ejecutan desde un navegador instalado en el equipo, donde su uso depende del acceso a la red de internet dando independencia de la plataforma y del sistema operativo. [65]

Aplicaciones nativas

Estas se desarrollan en base a su futura ejecución en entornos específicos, sistemas operativos compatibles y la revisión con la que cuente.[66]

Aplicaciones hibridas

Son herramientas multiplataforma donde el cómputo se realiza a través de un servidor web. [63], su funcionalidad puede extenderse a los recursos con los que cuente el dispositivo, como ubicación física, cámara, entre otros.

1.15.1 Servidor web embebido para transmisores

La infraestructura espacial de datos se denomina a la información que los sensores o transmisores en la industria proveen en función del tiempo. Esta información en tiempo real se considera una fuente de datos la cual debe ser soportada con recursos adicionales a nivel de aplicación, actualmente se toma como soporte los SWE para el acceso y control de estas fuentes de datos usando internet. [66]

Esta extensión de un sensor/transmisor web a un sistema de servicios web embebido se define como un procesador que contiene un conjunto de interfaces codificadas que trabajan de tal manera que la incorporación de la capa física puede abarcar múltiples formas de datos a un modelo común de aplicación basada en estándares web y de comunicación con el fin de obtener una representación de información normalizada. [66]

La integración de estos servicios web se basa en la filosofía de no depender de forma directa de software o configuraciones específicas [uso de controladores en una PC] para soportar la información proporcionada por un dispositivo (Plug and Play), por lo que se rige a cumplir con las siguientes bondades:

- Muestreo determinado por la capacidad del sensor asegurando medidas de buena exactitud y repetibilidad.
- Parámetros accesibles y modificables desde el software hacia el sistema instrumental.
- Mediciones proporcionadas en tiempo real y valores temporales con codificación construida bajo un estándar.
- Notificación de eventos a nivel de monitoreo del sistema sensor.
- Funciones de acceso remoto a través de página web.

Por lo tanto, la naturaleza de un SWE incrementa la interoperabilidad de un transmisor con aplicaciones industriales facilitando el razonamiento y la comprensión de información por sus servicios de despliegue, generación, almacenamiento y control además de reducir los costes de recursos informáticos ya que los requisitos de un servicio web son reducidos para su ejecución y manejo con la ventaja del acceso remoto de los usuarios. [65]

1.16 Criterio de uso de Gateway

El Gateway desempeña la función de establecer el puente entre el servidor web y la aplicación. Como se ha explicado con anterioridad, en el lado del servidor se ha de invocar

a la aplicación usando solicitudes HTTP del cliente, de esta manera se referencia o invoca objetos que se ha escrito en el framework de la aplicación. [66]. Las instancias u objetos en programación pueden ser una función o un método invocable para poder ser llamados varias veces mediante solicitudes.

Cuando la aplicación recibe el llamado desde el servidor, este genera un código de estado HTTP con encabezado generando la respuesta HTTP. [74]. El servidor presenta el resultado combinando el código de estado, encabezado y cuerpo resultado de la lógica que se haya programado en la aplicación, al cliente.

La aplicación como el servidor puede encontrarse en distintos formatos, lenguajes o estructuras, por lo que el Gateway deberá modificar el empaquetado de la información para responder a las solicitudes del cliente, entonces tanto la capa de negocio como la capa de cliente abarcan las multitareas que el Gateway físico debe procesar. [66] [67]

En otras palabras, el Gateway es un dispositivo que permite consolidar la información proveniente de las interfaces físicas, procesar dicha información en la aplicación además de permitir la conexión a una red LAN en la cual el servidor pueda intercambiar información a través de los objetos, atendiendo a solicitudes desde navegadores, aplicaciones móviles, entre otros. [66] [69]

1.17 Páginas web dinámicas

La utilización de las páginas web dinámicas incrustadas en los servidores web actualmente se enfocan en aumentar su rendimiento, la programación en la que se basan para cumplir este objetivo es de tipo front-end. La forma en la que se gestiona la información es a través de dos elementos la base de datos, y las páginas activas [caracterizadas por contener fragmentos del contenido de la base de datos en archivos javascript]. De esta manera el programa puede comunicarse con la base de datos de forma asincrónica para realizar modificaciones en la página. [70]

Para dinamizar el contenido de la información basados en navegadores web o aplicaciones móviles es necesario tomar en cuenta el uso de scripting en el lado del cliente por lo que la capa involucrada en esta tarea es la de presentación, algunas de las acciones comunes son el pasar el ratón en un campo, scroll en la pantalla, o a su vez acciones automáticas de tiempo. [68][70]

1.18 Dashboard para monitoreo y control [HMI/MMI]

Una interfaz máquina-hombre es un mecanismo que permite la interacción del humano con una máquina y determinar magnitudes físicas o estados de un proceso presentes en el campo de un proceso industrial. La presentación de los resultados y valores de medición

se pueden desarrollar en una interfaz gráfica con representaciones esquemáticas de un sistema individual de medida o conjunto mixto de mediciones, de esta manera se facilita la disponibilidad de lecturas de información comprensible para el usuario usando herramientas como tendencias en tiempo real, widgets, categorías de navegación, o división de información en varias pantallas las cuales deben ser visualmente cómodas como se especifica en [19][20].

1.19 Desarrollo basado en Flutter

Flutter se considera un marco de trabajo para la elaboración de interfaces web de usuario dinámicas y aplicaciones móviles. [36] priorizando soluciones para navegación, animaciones, tipografía entre otras, que representa la creación de aplicaciones en base a herramientas prototipo ya que permiten que la compilación se realice de manera directa en el procesador de un dispositivo. Flutter ofrece una comunicación fácil entre distintas plataformas, sin necesidad de un cambio de lenguaje entre códigos por lo que Javascript funciona como un puente que ofrece buen rendimiento en cuanto a ejecución y compilación. [37]

2. METODOLOGÍA

Se plantea un primer esquema de las operaciones secuenciales y en paralelo que se realiza en el desarrollo de la arquitectura del procesamiento, se opta por usar un servicio virtual por fines de abaratar costos y verificar la potencialidad computacional puramente de software frente a una plataforma que use hardware. (Figura 2.1)

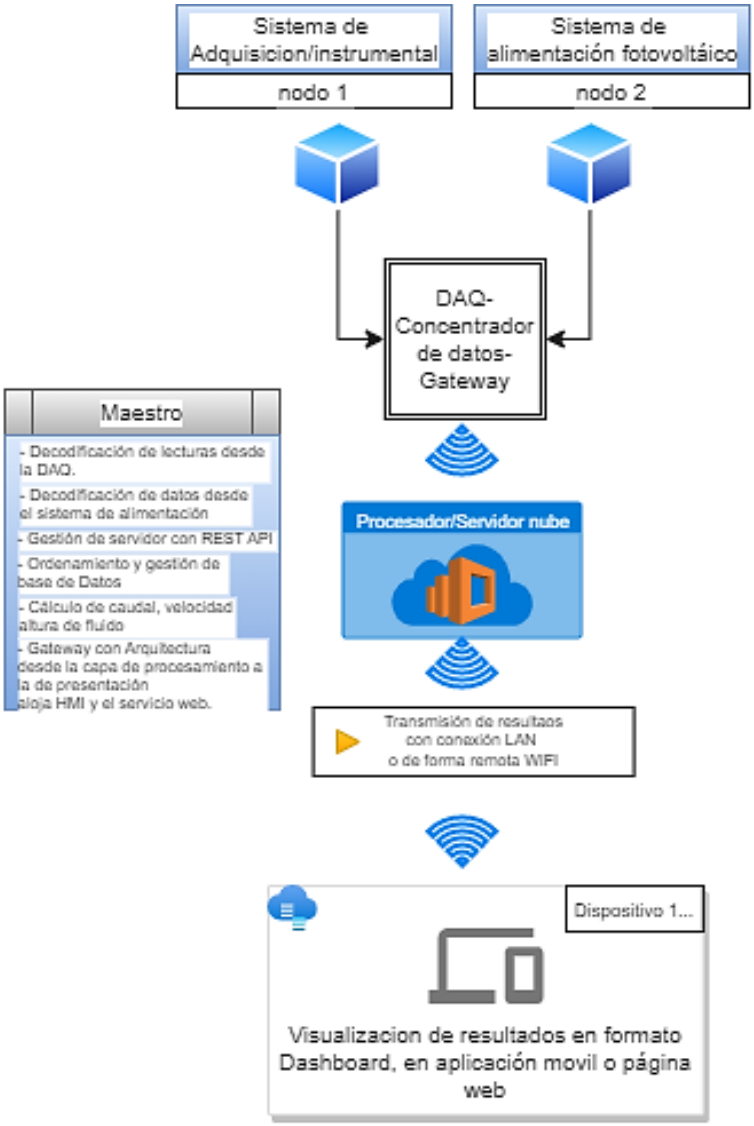


Figura 2.1. Operaciones desarrolladas por el procesador virtual.

2.1. Implementación de Gateway

En la figura 2.2 se identifica cada uno de los bloques de los que se compone el hardware; se ha seleccionado la tarjeta de desarrollo ESP8266 como el concentrador de datos WiFi/buses desarrollado por Arduino la cual utiliza una parte de la capa de datos con lenguaje de programación similar a C++ enfocado a controladores mientras que en la capa de procesamiento se utiliza un procesador virtual a través del uso de una instancia de los

servicios virtuales de Amazon cuyas características se describen en la Tabla2.1, la otra parte de la capa de datos hace uso de Javascript para la creación de las API REST y el enlace a la base de datos MongoDB. Se programa la tarjeta a través del sistema operativo Ubuntu versión 18.04 (versión modificada y adaptada para tarjetas de desarrollo) que es una distribución de Linux de software libre cuyo origen se basa en Debian, sin embargo, se programa desde la consola por ser una instancia de versión gratuita. [7].

Finalmente, la capa de presentación se programa en lenguaje orientado a objetos DART. Las peticiones que se realizan desde el lado del cliente utilizan los REST de la capa de negocio deserializando los documentos JSON que se generan en el servidor y se apuntan a la base de datos.

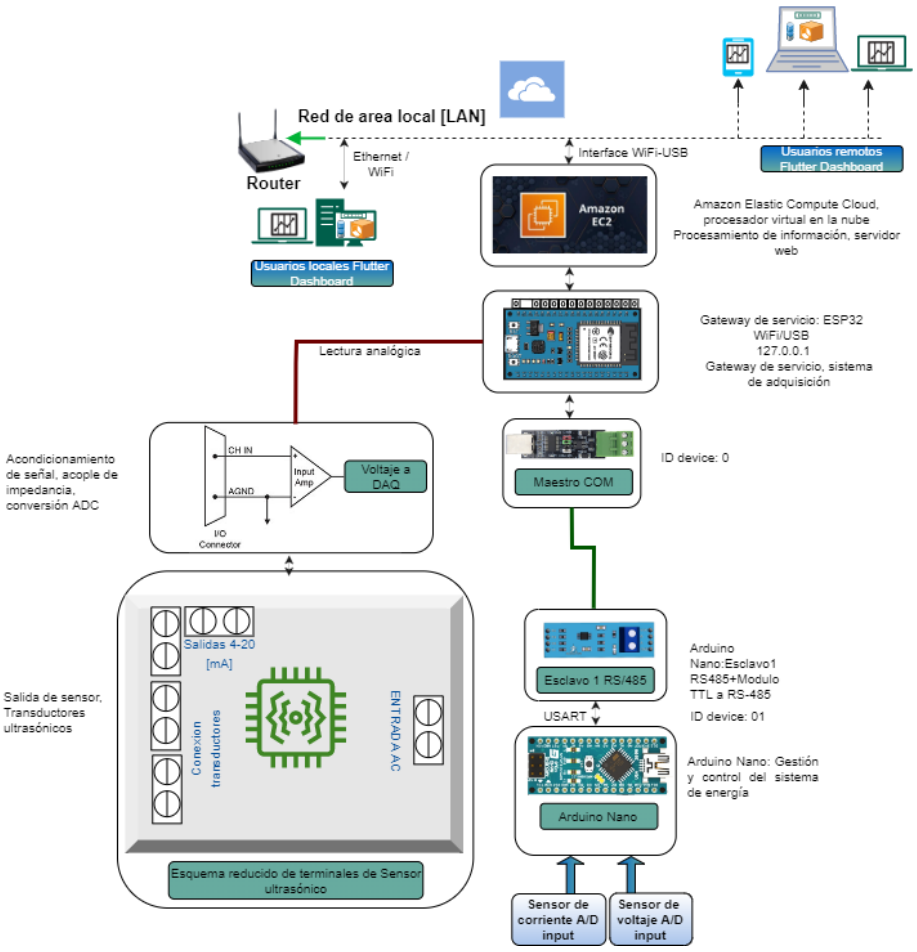


Figura 2.2. Montaje de interfaces del sistema de procesamiento virtual.

La funcionalidad del Gateway WiFi/virtual se basa en la adquisición de información de dos nodos montados en el bus de la interfaz física de manera indirecta. De forma general cada nodo usa una comunicación serial. En el primer caso el sistema de adquisición de información del sistema de transducción ultrasónico cuenta con una unidad que controla el fondo de escala en la medición como el encapsulamiento de información por separado a través de comunicación por documentos JSON hacia el servidor virtual.

En el segundo caso se cuenta con una unidad que controla el sistema de alimentación fotovoltaico a través de mediciones usando los puertos de entradas analógicas. Por lo que para una integración de mayor numero de dispositivos es necesario la disponibilidad de los pines seriales o canales analógicos de lectura del concentrador ESP8622. Para la recepción de la información proveniente de los nodos se utiliza una interfaz de tipo USB, la decodificación de los datos y procesamiento realizado por el maestro (Servidor de instancia EC2 t2.micro) que mediante peticiones HTTP y documentos JSON escribe, edita y genera la información previa a la presentación de resultados escrita en lenguaje de programación DART el cual se desarrolla en el framework Flutter enfocada al desarrollo de objetos que la interfaz web y la aplicación móvil ejecutan de forma paralela al back end del servidor.

2.2. Mecanismo de servidor virtual

La arquitectura que se construyó fue la de cliente-servidor debido a que uno o más clientes conectados a la aplicación móvil o interfaz del navegador a través de las credenciales establecidas realicen los requerimientos que el servidor tiene por medio de las solicitudes, se usa el protocolo HTTP para la actualización de los servicios codificando y decodificando documentos tipo JSON para el despliegue grafico de información (Figura 2.3). El mecanismo para realizar la interconexión de estos servicios se sustenta en la sección 1.6. debido a que REST API ofrece prácticas óptimas para su desarrollo.

Figura 2.3. Mecanismo de servicio REST API cliente-servidor.

Debido a que se prescinde de un servidor físico como tal se opta por el uso de una computadora virtual en este caso alojada en los servicios web de Amazon, las características se describen en la tabla 2.1.

Tabla 2.*1*. Características de procesador virtual para el servidor

Características	Descripción
Sistema Operativo	Ubuntu Server 20.04 LTS

Cantidad de núcleos	1
Espacio de memoria RAM	1GB
Microarquitectura CPU	64 bits
Familia	T2.micro
Velocidad de procesamiento	2.5 [GHz]

El objetivo de la aplicación se destina a un uso privado entre usuarios (estudiantes u operadores) dependiendo de la ubicación del sensor; por lo que se procesa las variables de caudal, velocidad superficial y altura en documentos de tipo JSON ya que el funcionamiento de tendencias debe mantener una evolución en el tiempo sin interrupciones por lo que los documentos en este formato aseguran que la transmisión de información instrumental/energía/aplicación web y la tarjeta DAQ se mantenga estable si es que existe un punto de conexión a internet.

2.3. Creación de servidor virtual en instancia EC2 de AWS

Los servicios informáticos de AWS se configuran en la consola de administración en base a la elección de una instancia EC2, la cual soporta el servicio de la máquina virtual como se observa en la Fig. 3.4

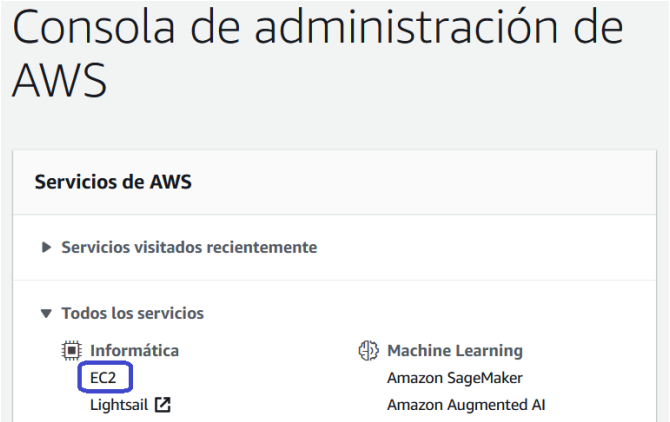


Figura 2.4. Elección de servicio EC2 de AWS

Una vez elegido el servicio EC2, accedemos a las instancias que nos permitirán elegir las combinaciones de la capacidad de memoria, procesamiento y almacenamiento para las aplicaciones del servicio web y los recursos que demanda. Nos ubicamos en el botón lanzar instancias:

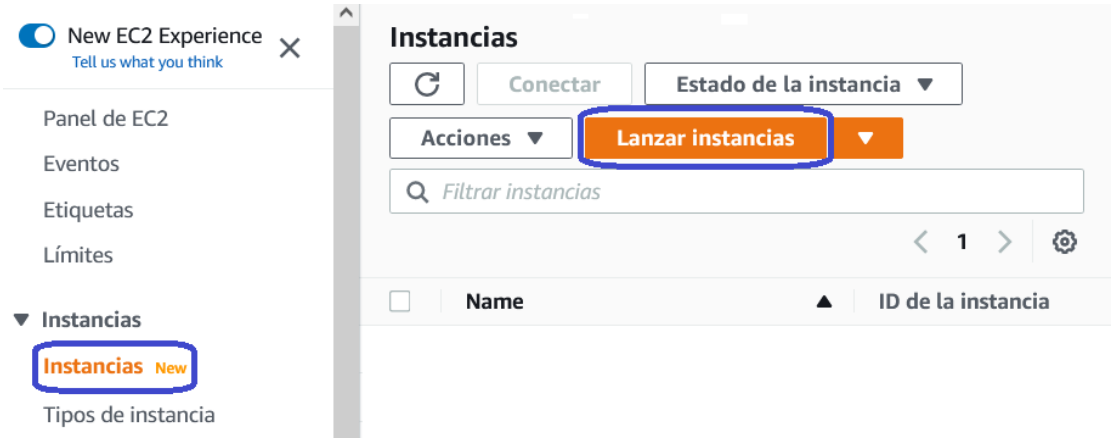


Figura 2.5. Lanzamiento de instancias en el soporte EC2 de AWS

La imagen ISO del sistema operativo se debe elegir para ser instalada en el servidor. En este caso elegimos el servidor de Ubuntu de propósito general.

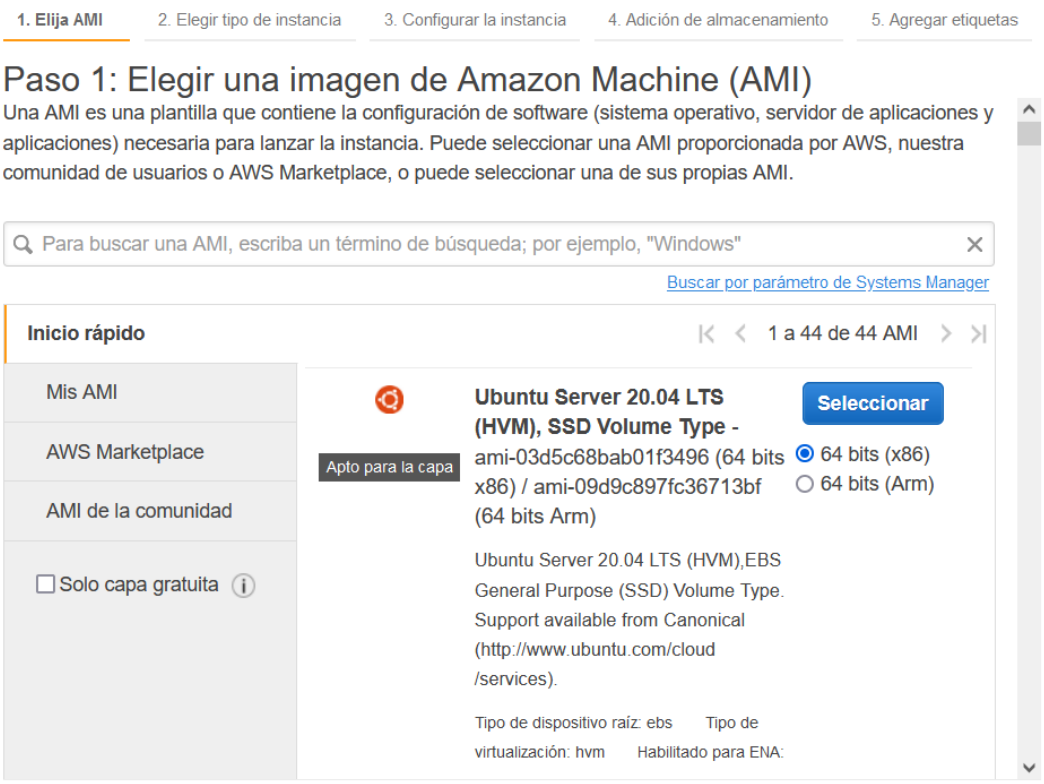


Figura 2.6. Elección de máquina virtual de Amazon

La elección de procesadores con la memoria RAM, número de núcleos, tasabilidad de CPU por hora y precio de la instancia por hora de demanda se elige en función del consumo de los REST del servidor. El proyecto busca abaratar los costos de implementación a nivel de software, se elige la versión t2.micro.

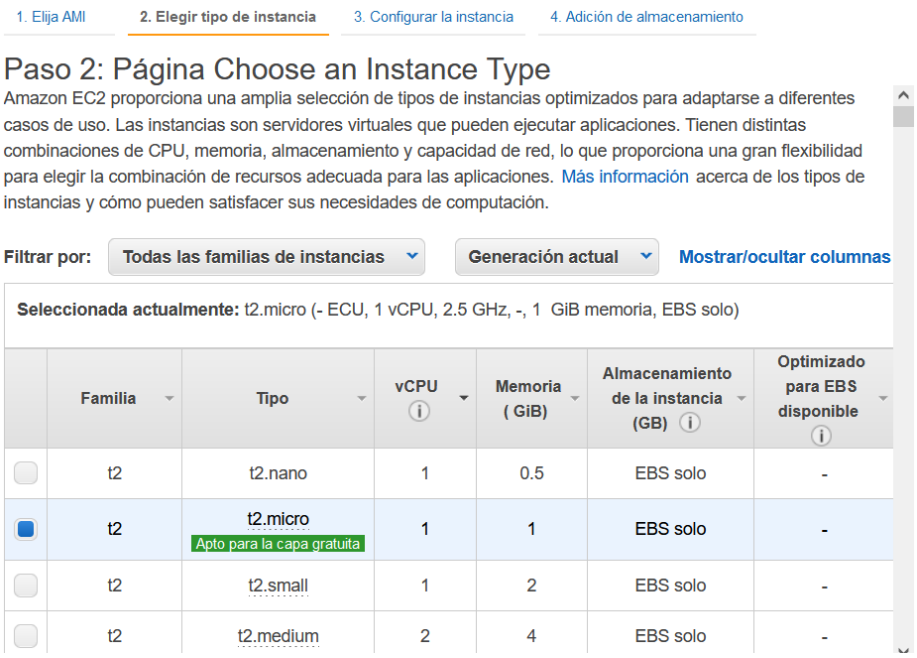


Figura 2.7. Recursos de procesador virtual, elección de capa gratuita.

Una vez finalizado el paso 2, se configuran los parámetros de internet para el servidor, los que se han de modificar únicamente son la nube virtual privada [VPC], las subredes pertenecientes a la VPC y por último la IP pública para brindar acceso al servidor con una red cualquiera.

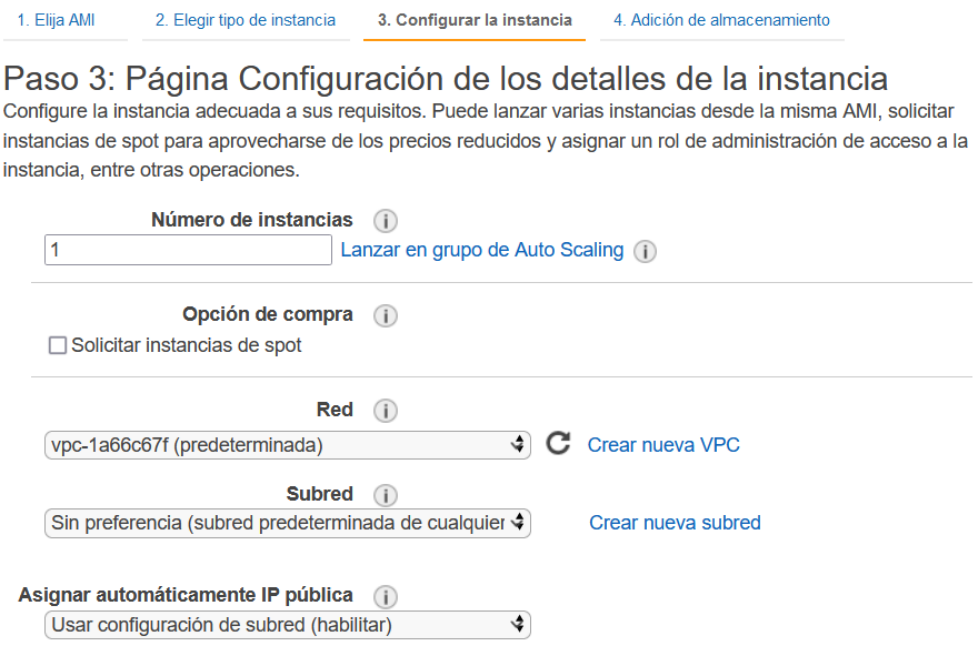


Figura 2.8. Configuraciones de red para el servidor.

Se debe asignar el volumen que abarcará la instancia, esto en base a los requisitos de diseño de las REST que se proyecta en el servidor, sin embargo, el volumen superado no implica un desbordamiento de información, EC2 asigna recursos adicionales al límite de

capacidad asignado de forma automática, funcionando con una o más máquinas virtuales de ser necesario. Asignamos un límite de 8 GiB.

Figura 2.9. Asignación de volumen de almacenamiento de servicio virtual.

Para resguardar la seguridad a la información que se acceda de forma remota a través del servidor se selecciona un grupo de seguridad de protocolo SSH el cual posee una encriptación robusta y seleccionamos el puerto HTTP para que el servidor se mantenga a la escucha.

Figura 2.10. Selección de protocolo de seguridad y puerto de servidor.

Finalmente elegiremos la opción de revisar y lanzar, se despliega las características y configuraciones que se establecieron para el servidor.

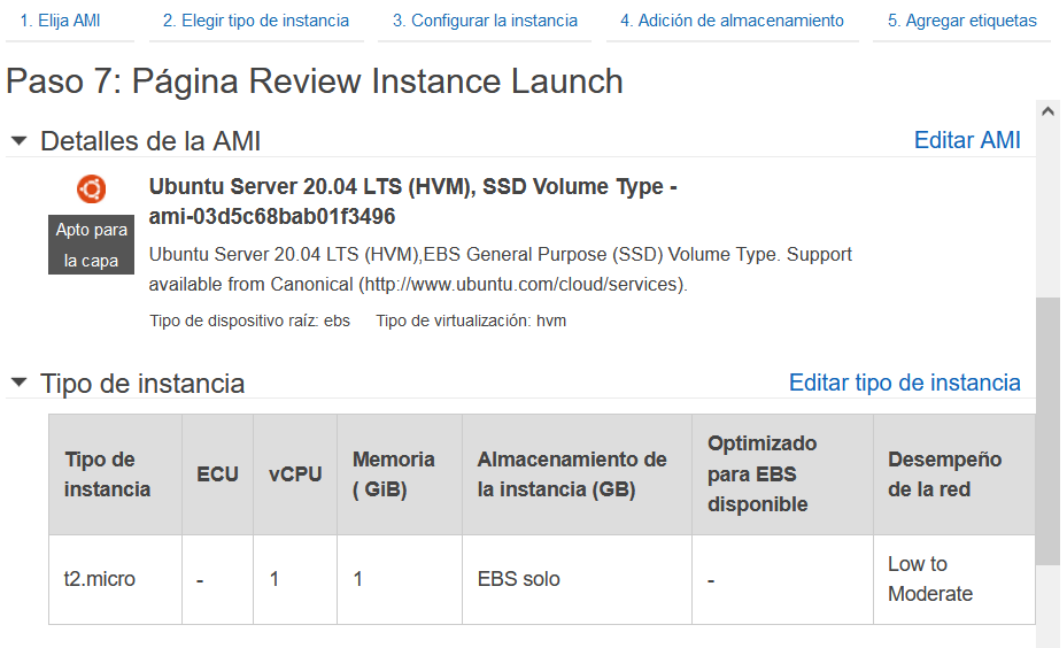


Figura 2.11. Características de servidor configuradas en EC2, AMI de ubuntu

La llave para acceder al servidor es usada para las conexiones SSH, sin ellas no será posible que se acceda de forma remota a este último, este proceso debe realizarse por el usuario que requiera usar el servidor de forma particular para administrar el intercambio de información.

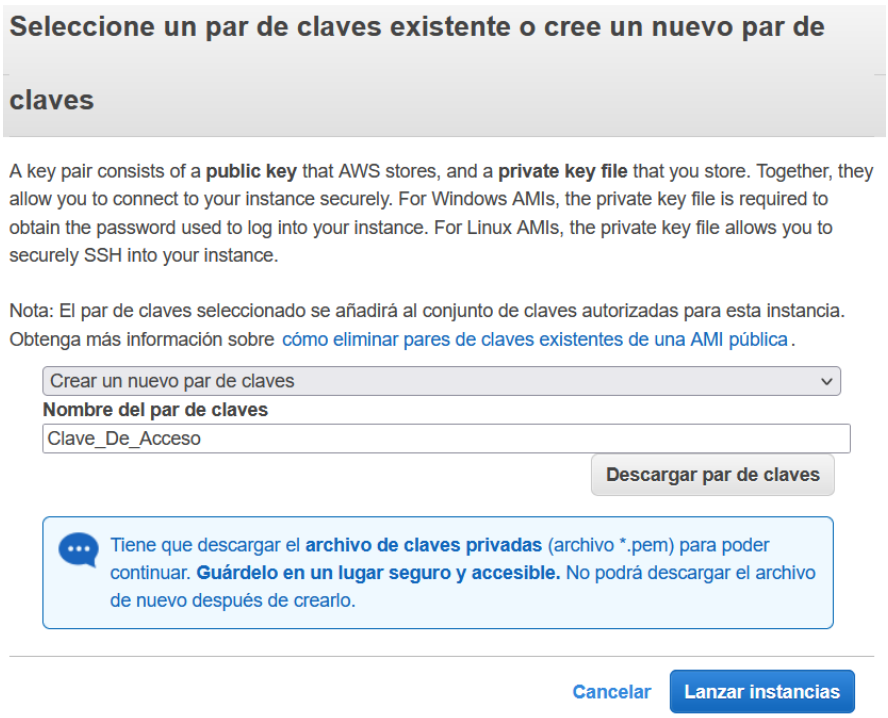


Figura 2.12. Llave de acceso única de instancia.

2.4. Desarrollo de REST API

El desarrollo de la codificación en el lado del servidor se realiza a través del procesador virtual de Amazon, en la consola de la imagen se lanzará la instancia a través de la dirección IP privada importando las llaves de acceso que se obtuvo en la sección 2.3. En la Figura .2.13 puede verificarse el % de espacio usado por la API que se ha creado, la IP de conexión ethernet, y los usuarios que administran el servidor.

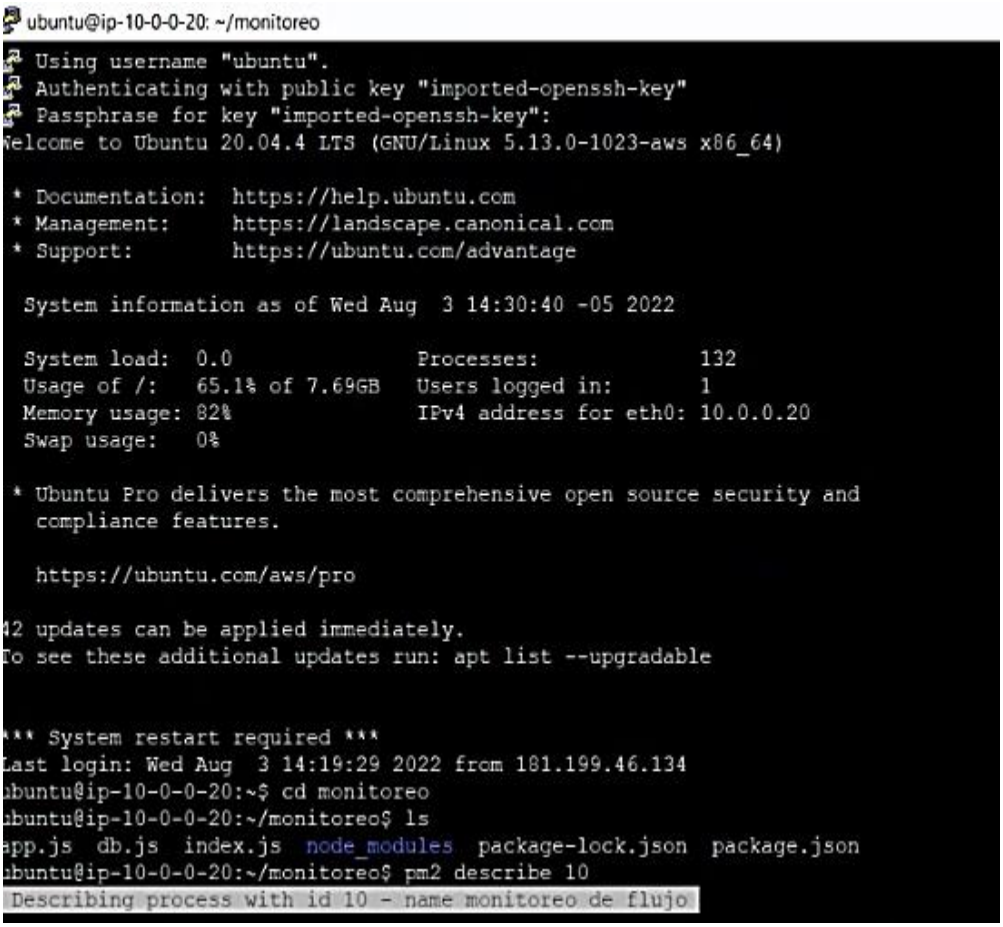


Figura 2.13. Información de versión de Ubuntu que soporta la API

Para poder mostrar la lista de los directorios que se crearon en Linux de la API, se utiliza el nombre de la API + el comando ls, obteniendo los archivos en orden alfabético, en este caso el directorio que contiene los subdirectorios es el de nombre: app.js. La tabla 2.2 muestra

Usando el comando describe X, se presentan los metadatos de las tablas que se traerán desde la base de datos. Para la tabla no relacional que se referencia desde la base de datos se obtiene la descripción:

Tabla 2.2. Características de procesador virtual para el servidor

Status	**online**
Name	monitoreo de flujo
namespace	Default
Version	1.0.0
restarts	5
Uptime	14D
script path	/home/ubuntu/monitoreo/index.js
error log path	/home/ubuntu/.pm2/logs/monitoreo-de-flujo-error.log
out log path	/home/ubuntu/.pm2/logs/monitoreo-de-flujo-out.log
pid path	/home/ubuntu/.pm2/pids/monitoreo-de-flujo-10.pid
interpreter	Node
interpreter args	N/A
script id	10
exec cwd	/home/ubuntu/monitoreo
exec mode	Fork_mode

El procesador virtual de Amazon si bien cuenta con una versión de Ubuntu 20.04.4 LTS no hay un sistema operativo en el cual se pueda instalar un IDE por lo que se programa únicamente a través de la consola, esto debido a que la versión gratuita no incluye dicho sistema operativo, se observa los documentos de la aplicación de JavaScript: app.js y las rutas de los documentos del servidor con características adicionales en la Fig.2.13. Usando el comando describe se despliega el estado e información del proceso o API que se desarrolló. En la Tabla 3.2 se puede observar los metadatos alojados en el servidor.

En primer lugar el código de cabecera escrito en Javascript declara algunas constantes empezando por express encargada de cargar los módulos y exportaciones de Node, similar a la función include en C, bodyparser se usa debido a que el cuerpo requiere peticiones tipo POST cuando un cliente va a crear o actualizar un registro [PUT], es decir es un middleware para acceder a la información en la base de datos, db se encarga de crear la nueva instancia dentro de la base datos, app llama a una función express a manera de objeto en una aplicación, jwt admitirá la autorización para el intercambio de información creando la codificación y decodificación para los datos, cors es un mecanismo para HTTP para que el usuario obtenga acceso a los recursos del servidor es decir desde el front end, por último tenemos ObjectID el cual se encarga de la creación de identificadores para los documentos JSON de la base de datos. El código descrito con anterioridad se presenta en Figura 2.14.

GNU nano 4.8

```
‘use strict’
const express = require ('express');
const bodyParser = require('body-parser');
const db = require (". /db′);
```

```
const app = express ();
const. jwt = require('jsonwebtoken');
const cors = require ('cors');
const. Objectid = require('mongodb'). Objectid;
```

Figura 2.14. Declaración de constantes necesarias del programa

Adicionalmente, se tienen las estructuras monitoreo de flujo, usuarios y datos los cuales son los documentos asignados para los registros que almacena la base de datos. Por último, las estructuras bodyParser le indican al sistema que use los documentos JSON y que se realice un algoritmo sencillo para un análisis superficial, las funciones request, response y next son las devoluciones o callbacks. El código se describe en la Figura 2.15

```
const base = 'Monitoreodeflujo'
const coleccionUsuarios = 'usuarios'
const coleccionDatos = 'datos'
const secret =
'knfajnfjkmbfrqogqrojqrqwrsayfjbsjfbRhEOUYebDKNVbsdlyt
app. use (bodyParser.urlencoded( ( extended: false J));
app. use (bodyParser.json());
```

Figura 2.15. Declaración de documentos JSON y funciones HTTP

Las cabeceras res.header permiten que el navegador enlace su solicitud con un método ‘GET, POST, Allow’, comprobando el acceso a un servicio que se encuentra en el servidor a través del navegador desde un dominio diferente, de esta forma se obtiene una respuesta mediante un método concediendo o no el acceso. El código de la Figura 2.16 declara las cabeceras de los servicios.

```
App.use((req, res, next) => {
res.header(‘Access-Control-Allow-Origin’,’*’);
res.header(‘Access-Control-Allow-Headers’,’Authorization, X-API-KEY, Origin, X-
Requested-With, Content-Type, Access-Control-Allow-Request-Method’);
res.header(‘Access-Control-Allow-Methods’,’GET’,’POST’);
res.header(‘Allow’,’GET’,’POST’);
Next();
});
```

Figura 2.16. Declaración de cabeceras de llamado de servicios en API

En los siguientes diagramas se presenta la estructura lógica del programa en el lado del servidor, adicionalmente se describe como se encuentra construida la conexión entre el sistema de adquisición, la base de datos, y el front end de la aplicación.

2.4.1. REST de verificación de estado de conexión

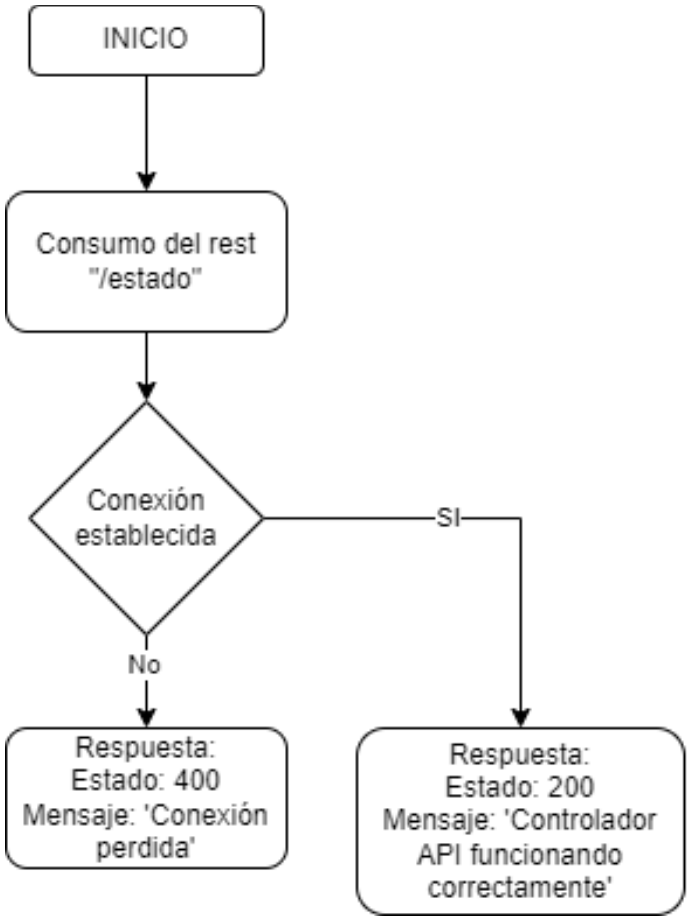


Figura 2.17. Diagrama de estado del REST a modo de verificación

La figura Figura. 2.17 muestra cómo se realiza el servicio de verificación codificado, en primera instancia si la conexión no es correcta entonces las peticiones posteriores en el servidor no se van a ejecutar, esto puede deberse a varios factores, pero la razón principal es que el servidor mostrará error cuando la porción de almacenamiento en la memoria se ha superado. El código implementado se encuentra en la descripción del código “REST estado”. El código implementado en JavaScript del REST descrito en la figura 2.17 se implementa en la figura 2.18.

```
app.get('/estado', function (req, res) {
return res.status(200).send('Controlador API funcionando correctamente')
});

app.patch('/login', function (req, res) {
if (!req.body.user ||!req.body.password) {
return res.status(400).send({ mensaje: "Parametros incompletos"});
}
let DB = db.get().db(base).collection(coleccionUsuarios);

new Promise((resolve, reject) => {
DB.aggregate([{ $match: { user: req.body.user, password: req.body.password}
}]).toArray(function(err, result) {
if (err){
reject("Access Denegado")
})
```

Figura 2.18. Código de verificación de estado de la API REST

2.4.2. REST para creación de usuarios en base de datos MongoDB

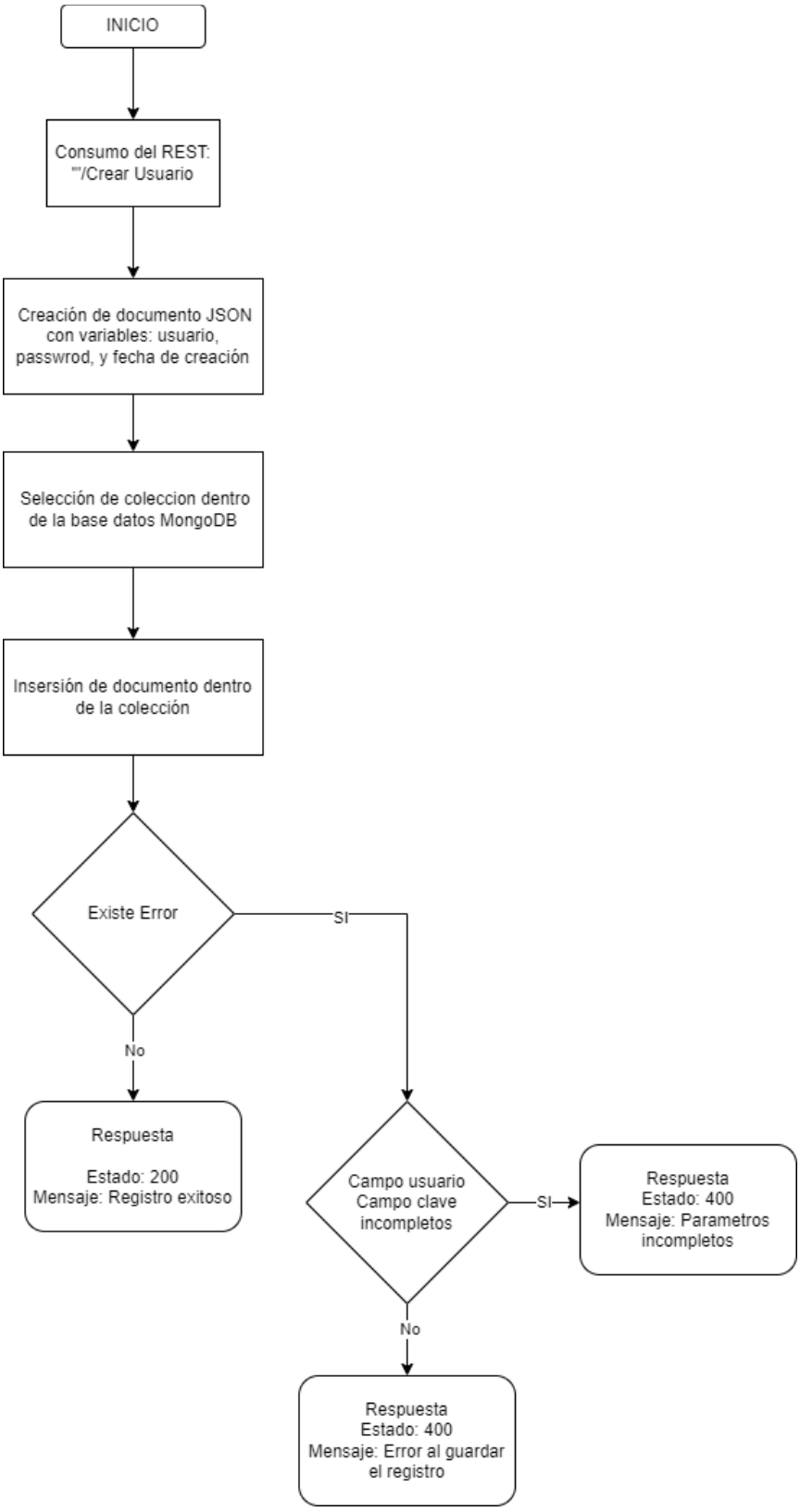


Figura 2.19. Diagrama de flujo para registro de usuarios en la base de datos MongoDB

La figura 2.19 describe el proceso de registro de usuarios dentro de la base de datos para el acceso con credenciales, cada usuario tiene acceso a los históricos de las variables alojadas en el documento JSON, de presentarse o no errores se notifica con los códigos de estado HTTP 200 y 400 respectivamente. El código implementado del REST solicitud-respuesta para la creación de usuario se muestra en la figura 2.20.

```
app.post ('/crearUsuario', async function (req, res) {
if (!req.body.user II !req.body.password) (
return res.status(400). send({ message: "Parametros incompletos" });
}
let doc = { user: req.body.user, password: req.body.password, createdAt:
Date.now(),}

let.DB = db.get ().db (base). collection(coleccionUsuarios);
auth(req, res).then(
result => {
if (result) {
DB.insertone(doc, (err), => {
if (err) {
return res.status(400). send({ message: "Error al guardar el registro" })
} else {
return res.status(200). send({ message: "Registro exitoso" });
}
})
} else
return res.status(400). send({ message: "Acceso denegado" });
},
error => {
return res.status(400). send({ message: error });
})});
```

Figura 2.20. Código de registro de usuarios en la base de datos del servidor.

2.4.3. REST para login de usuarios en el lado del servidor

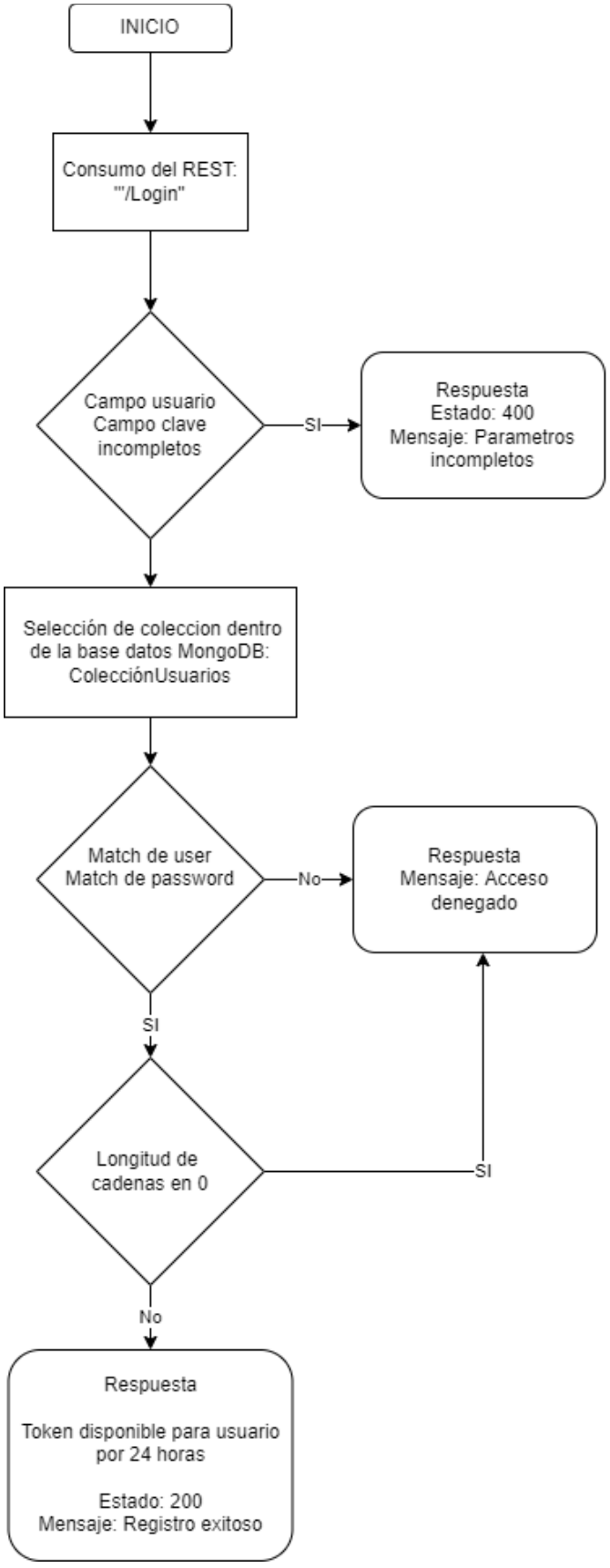


Figura 2.21. Diagrama de flujo para Login de usuarios en el lado del servidor.

En el diagrama de la Figura. 2.21 se puede observar la lógica de verificación para el login de un usuario en la interfaz web hacia el lado del servidor, el REST ”Login” realiza la verificación cifrada tanto del usuario como de la clave, una vez que las cadenas no vacías coinciden con las que se encuentran previamente almacenadas en la colección de la base de datos se genera un jwt de intercambio de información, mientras los datos sean los esperados haciendo un $match bajo la información específica tipeada por el cliente se habrá realizado el acceso de inicio de sesión haciendo que el método de promesa se mantenga en ejecución secuencial del código con el ultimo valor asincrónico. El código implementado para el REST de login de usuarios que se registran en la base de datos a petición del cliente se encuentran en la figura 2.22.

```
app.post ('/login', function (req, res) {
if (!req.body.user II !req.body.password) (
return res.status(400). send({ message: "Parametros incompletos" });
}

let.DB = db.get ().db (base). collection(coleccionUsuarios);

new Promise((resolve, reject) => {
DB.aggregate([{ $match: {user: req.body.user, password:
req.body.password}}]).toArray(function(err, result){
if(err){
reject(“Acceso Denegado”)
} else {
if (result.length == 0) {
reject(“Acceso Denegado”)
}else{
resolve(result[0]);
}}})
}).then(

result => {
let token = jwt.sign({id: result._id}, secret, {expiresIn: ‘24h’});
return res.status(200).send({token: token});
);
error => {
return res.status(400).send({message:error});
})
```

Figura 2.22. Código de REST para login de usuarios por POST en la base de datos.

Para la operación asincrónica se realiza a base de la función “promise” que es la estructura que retorna un objeto para la finalización de la operación asincrónica del REST login.

```
new Promise((resolve, reject) => {
DB.aggregate([{ $match: {user: req.body.user, password:
req.body.password}}]).toArray(function(err, result){
if(err){
reject("Acceso Denegado")
} else {
if (result.length == 0) {
reject("Acceso Denegado")
}else{
resolve(result[0]);
}}})
```

Figura 2.*23*. Código de REST para login de usuarios, finalización con método promesa.

Se ha usado la aplicación URL encoded para realizar las verificaciones de los usuarios. A continuación, se tiene dos pruebas de operatividad de los parámetros usuario y contraseña, en caso de no realizar el login de forma correcta, el match de la colección debe buscar el usuario particular: admin, y contraseña: abc123. El caso del Código para Login de usuarios registrados sucede en caso de no hacer el match con las cadenas almacenadas en la base de datos, Fig.2.24, mientras que el constructor promesa de login con ejecución asincrónica sucede en caso de tener alguna cadena vacía, Fig.2.25.

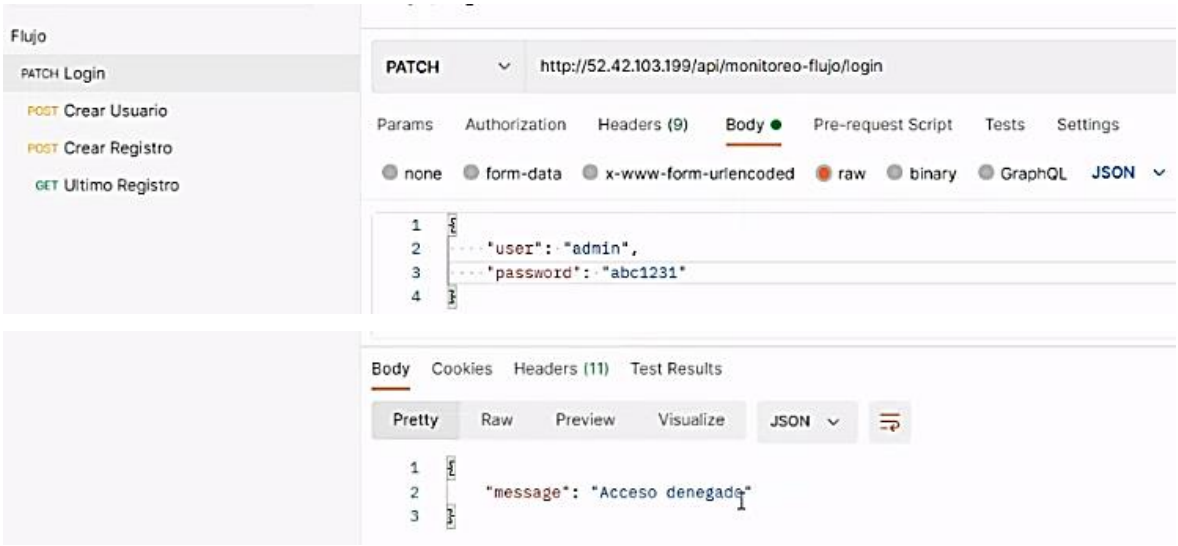


Figura *2.24*. Caso de verificación de credenciales de usuario, acceso denegado

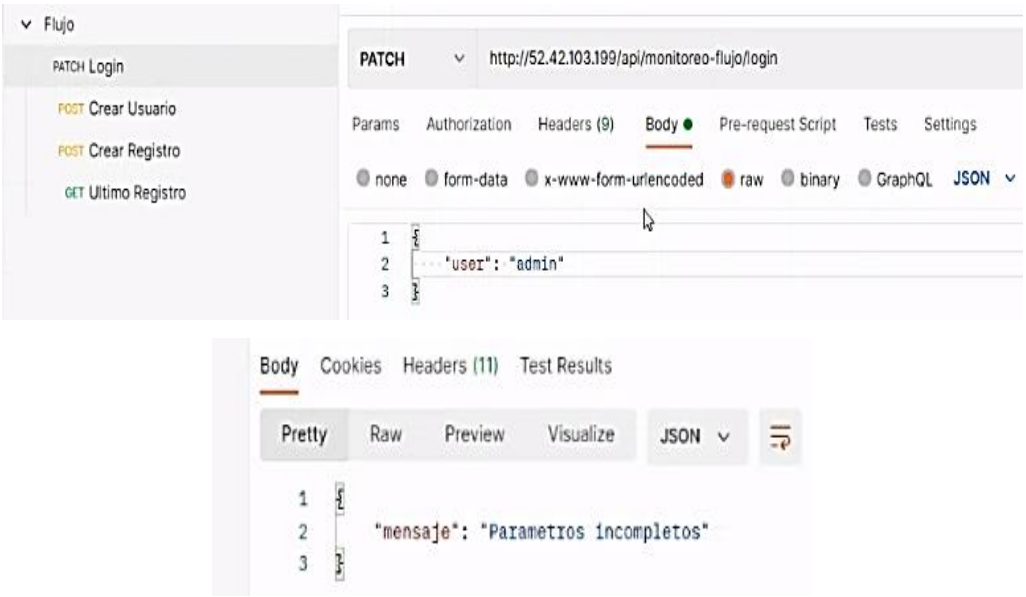


Figura *2.25*. Caso de verificación de credenciales de usuario, parámetros incompletos.

El dispositivo físico que concentra los datos provenientes de uno o más nodos de información es la tarjeta ESP8266, esta tarjeta cuenta con la capacidad de configurarse para contar con conexión wifi, y con la configuración que se implementa por código se han formado paquetes JSON para realizar el POST de datos, basta con utilizar las librerías de servicio URL que ofrecen los servicios de internet de la base de datos MongoDB.

2.4.4. REST para creación de registros de información de la DAQ

El flujo de información desde el Access point hacia el servidor virtual, se utiliza el protocolo HTTP, la creación de los documentos JSON encriptados tienen la función de almacenar la lista de información que se ha muestreado en el hardware de adquisición para transmitirse a manera de peticiones GET hacia el servidor y escribirse en la base de datos a manera de POST. La información por encapsular contiene los datos de caudal, energía, estado las cuales se registrarán en la base de datos en las colecciones correspondientes a “Colección de usuarios” creada en esta. Si existe anomalías en los callbacks de los objetos HTTP, se desplegará el mensaje 400. El diagrama 2.26 esquematiza esta operación:

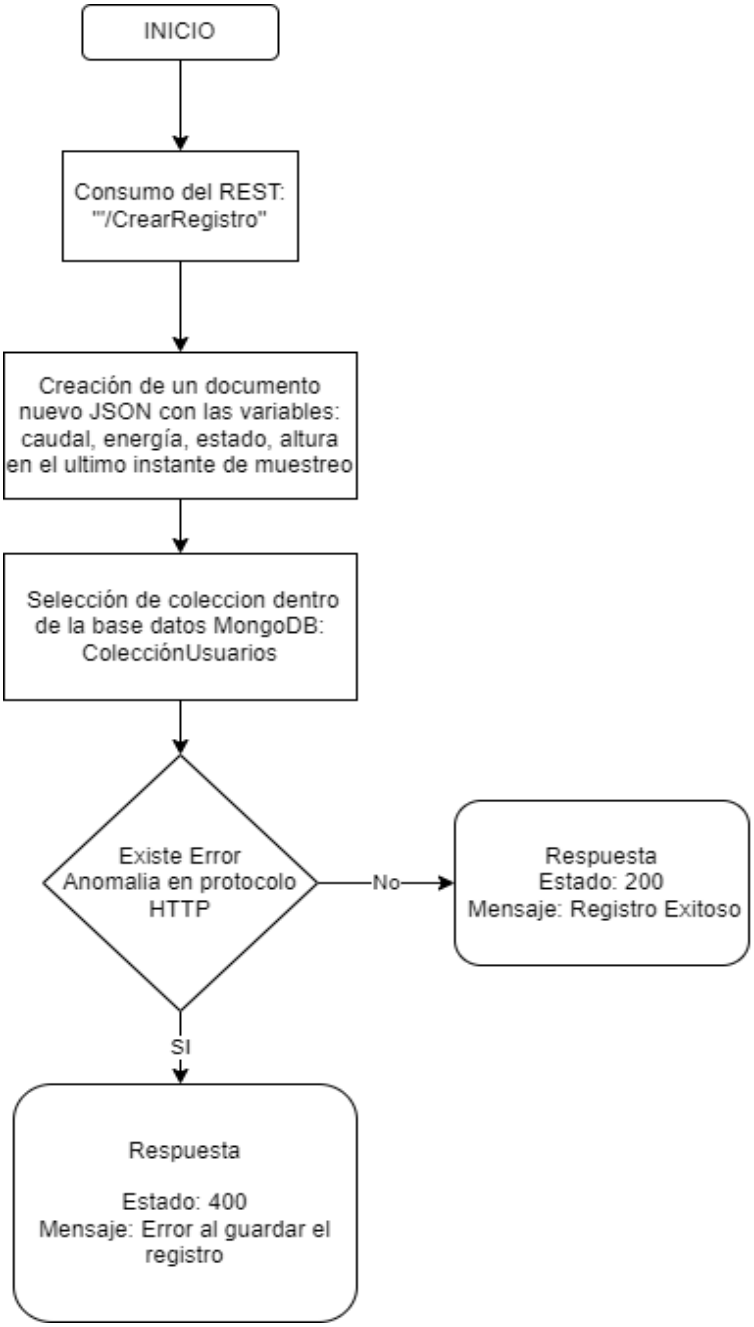


Figura *2.26*. Diagrama de flujo para escritura de información proveniente del sistema

El código para la obtención de los documentos que encapsulan los datos desde el sistema DAQ se puede observar en la figura 2.27.

```
app,post ('/crearRegistro', function (req, res) {
if ('req.body. caudal || !req.body,bateria || |req.body.estado) {
return.res.status(400).send({message: “Parametros incompletos”});
}
let doc = {
caudal: req.body.caudal,
bateria: req.body.bateria,
estado: req.body.estado,
createdAt: new Date(),
}
let DB=db.get().db(base).collection(coleccionDatos);

DB insertOne(doc, (err) => {
if (err) {
return res.status(400).send({message: “Error al guardar el registro”});
}else{
return res.status(200).send({message: “Registro Exitoso”});
}})});
```

Figura *2.27* Código para escritura de información proveniente del sistema DAQ

2.4.5. REST para ordenamiento FIFO de información

La estructura que se maneja dentro de la colección de datos en la capa de procesamiento se realiza por agrupación de valores (como se ha descrito en la sección 1.9) para el análisis a lo largo del tiempo.

En la base de datos MongoDB se utiliza operadores para este fin, en primer lugar se utiliza el comando $sort el cual devuelve los documentos JSON de entrada ordenados con un valor de operador id:-1 que significa que el orden de entrega de la información será de forma descendente, adicional a ello se usa el método 'limit' en un valor de 1 que se encarga de optimizar el código en cuanto a la memoria se refiere pues se obtendrá un límite en los espacios disponibles en la cadena de datos de 1 de elementos dentro de la memoria. De esta forma se obtuvo una estructura de entrada-ordenamiento y entrega de datos donde el primer elemento que llega será el primero en salir o mostrarse en la capa de presentación, esta lógica se observa en el diagrama de la Fig. 2.28.

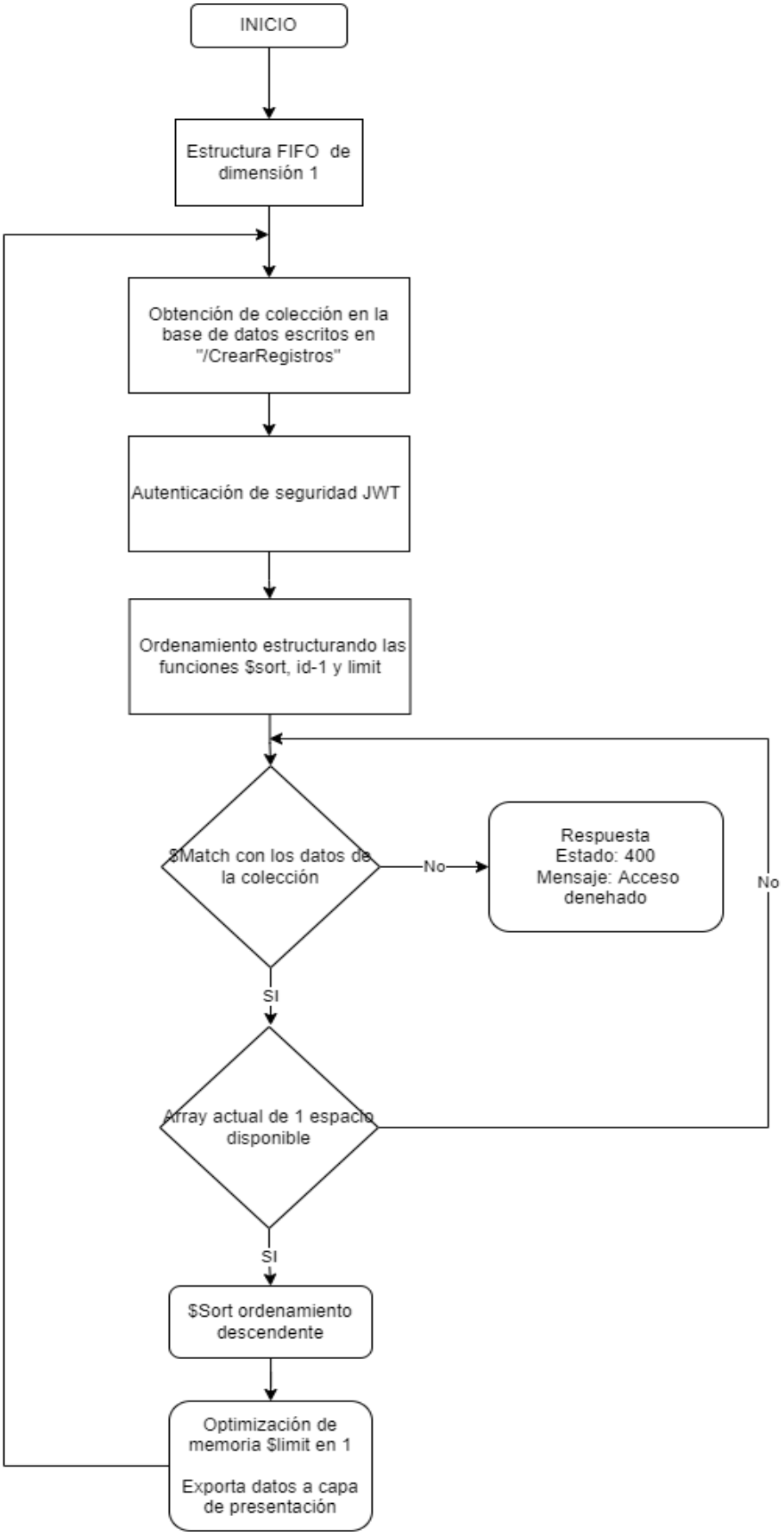


Figura *2.28* Diagrama de flujo ordenamiento tipo FIFO con optimización de memoria

El código para la agregación de información desde la base de datos al servidor se documenta en el código de canalización ´ObtenerUltimoRegistro´ del REST de la API, que se puede ver en la figura 2.29.

```
app,post ('/obtenerUltimoRegistro', function (req, res) {
let DB=db.get().db(base).collection(coleccionDatos);
auth(req, res).then(
result => {
DB.aggregate([{ $match: {} },{$sort: {_id:-1}}, {$limit:1}]).toArray(function
(err,data){
if(err){
return res.status(400).send({message: “Acceso denegado”})
}else{
return res.status(200).send(data[0]);
}})}
else
return res.status(400).send({message: “Acceso denegado”});
},
error => {
return res.status(400).send({message:error});
})});
module.exports = app;
```

Figura *2.29* Código para extracción de ultimo registro de la base de datos.

2.4.6. REST para filtrado de información

Para el filtrado de datos se implementó la técnica bajo los criterios de las secciones descritas en 1.10.3 y 1.10.4, con las herramientas que proporciona NodeJs en el lado del servidor. En la figura 2.30 se describe en primera instancia el acceso a la colección de datos del grupo $caudal respectivamente que es la información que proviene de la lectura análoga digital de la DAQ. Una vez se accede a los documentos JSON de este grupo particular se toma en orden decreciente las últimas veinte lecturas y se agrupan en un array.

Se calcula el promedio del arreglo y finalmente se realiza la diferencia con cada uno de los elementos del vector, esta diferencia absoluta bajo el criterio de 1.10.4 se ha escogido valores entre los que debe mantenerse la desviación de los valores típicos instantáneos de 0.25 a 1.25. Los registros de caudal se vuelven a escribir en la base de datos si cumplen esta condición y retorna al acceso de las siguientes veinte lecturas, de no cumplirse la condición se mantiene el ultimo valor de caudal filtrado y retorna al acceso del grupo caudal en el documento JSON.

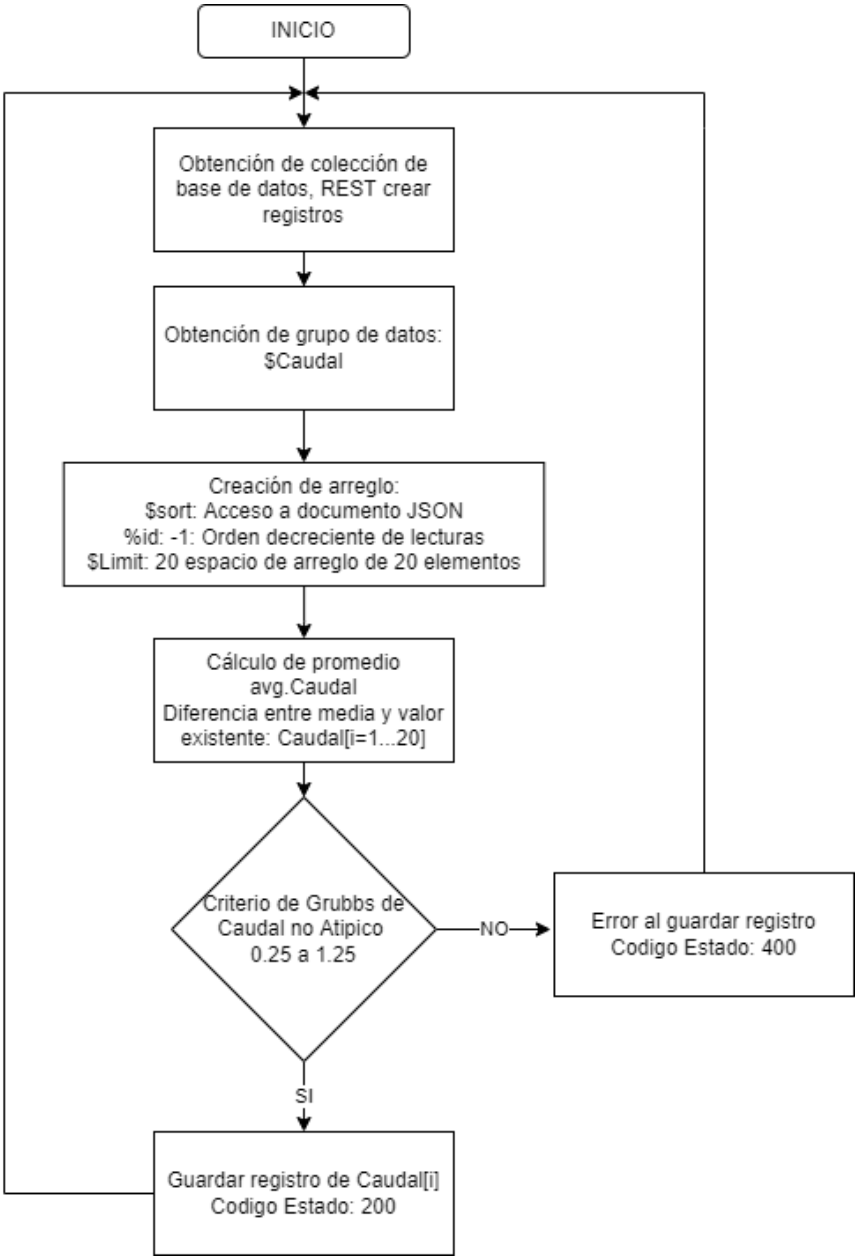


Figura *2.30* Diagrama para filtrado de información estadístico de valor $caudal.

El código del REST de filtrado se encuentra implementado dentro de la creación de registros, y en la figura 2.31 se muestra su codificación en el lado del servidor.

```
let DB = db.get().db(base).collection(coleccionDatos);
let group = { _id: null, avg: { $avg: "$caudal"}}
new Promise((resolve, reject) =>) {
DB.aggregate([{$match:{}}, {$sort: {id:-1}}, {$limit: 20}, {$group:
group}]).toArray(function(err,result){
if(err) { reject("No data") }else{
resolve(result);
}})
}).then(result=> {if (doc.caudal/result[0].avg<0.75 || doc.caudal/result[0].avg
>1.25) doc.caudal=result [0].avg
```

Figura 2.*31* Código implementado para filtrado de información de $caudal.

2.4.7. Función de autenticación HTTP del servidor

El código en el cual se implementa la solicitudes y respuestas HTTP se construye en la función ´Auth´, esta permite devolver el JWT firmado de los objetos hacia el cliente, es el código para funcionamiento JWT en ExpressJs

```
Function auth(req, res){
if(!req.headers.authorization){
return Promise.reject(“Parametros incompletos”);
}
let user_id = jwt.decode(req.headers.authorization, secret).id;
if(!user_id){
return Promise.reject(“Acceso denegado”);
}
let DB = db.get().db(base).collection(coleccionUsuarios);
return new Promise((resolve, reject) => {
DB.aggregate([{ $match: {_id: new ObjectId(user_id)}
}]).toArray(function(err,result){
if (err) {
reject(“Acceso denegado”)
}else{
if(result.length==0){
reject(“Acceso denegado”)
}else{
resolve(result[0]);
}}}}).then(
result => {
return result._id;
},
error => {
return undefined
})};
```

Figura *2.32* Código implementado para solicitudes-respuestas HTTP.

2.4.8. Conexión entre servidor y base de datos MongoDB

Para la conexión con la base de datos se usó el método llamado “Db” que es el objeto asignado de Mongo, de esta forma la conexión se realiza usando una instancia que puede entender el servidor. Después de realizar la consulta de campos nulos en los documentos de entrada se comprueba que la conexión con la url del servidor siempre sea la misma para la escucha GET y finalmente cerramos la conexión.

```
var MongoClient = require('mongodb').MongoClient
var state = { db: null }
exports.connect = function(url, done) {
if(state.db) return done()

MongoClient.connect(url, function(err, db) {
if(err) return done(err)
state.db = db
done()
})}
exports.get function(){
return state.db
}
exports.close = function(done) {
if (state.db){
state.db.close(function(err, result) {
state.db = null
state.mode = null
done (err)
})}}
```

Figura *2.33* Código de conexión servidor NodeJs-Base de datos MongoDB.

2.5. REST Implementados en el Servidor

En primer lugar, se tiene el patch de login para el acceso y comprobación de usuarios.

Tabla 2*.3.* Datos que se generan para REST de login de usuario

Tipo	Descripción
Método HTTP	PATCH
URL:	http://52.42.103.199/api/monitoreo-flujo/login

La API que se encarga de la creación de los registros en la colección de usuarios es el REST crear usuario.

Tabla 2*.4.* Datos que se generan para REST de login de usuario

Tipo	Descripción
Método HTTP	POST
URL:	http://52.42.103.199/api/monitoreo-flujo/crearUsuario

Adicionalmente, se tiene la API que se encarga de la creación de registros para el ordenamiento, filtrado y escritura de registros en la base de datos y el servidor.

Tabla 2.*5.* Datos que se generan para REST de creación de registros de datos.

Tipo	Descripción
Método HTTP	POST
URL:	http://52.42.103.199/api/monitoreo-flujo/crearRegistro

Por último, se tiene la API de ordenamiento donde se extrae y lleva el último registro mediante la respuesta get del servidor por la solicitud del cliente.

Tabla 2.*6.* Registro de salida hacia la interfaz del cliente.

Tipo	Descripción
Método HTTP	**GET**
URL:	http://52.42.103.199/api/monitoreo-flujo/obtenerUltimoRegistro

2.6. Servicio de alojamiento web [Hosting]

Amazon provee de un hosting que trae un dominio particular para stacks de organizaciones sin fines de lucro, debido a que la modificación de estos puede implicar gastos adicionales por lo que se utiliza la dirección web que EC2 entrega al usuario para la gestión de su información, como observa en la Tabla 2.7

Tabla 2.*7.* Last point de entrada de servicio web de Amazon EC2

Descripción	Dirección URL de página web
IPV4 last point	http://monitoreo-de-flujo.s3-website-us-west-2.amazonaws.com/#/dashboard

2.7. Desarrollo FrontEnd de Interfaz Web/Aplicación Móvil

La capa de presentación es la estructura visual para la interacción entre el usuario y la información proveniente de la base de datos a través del servidor. El framework que se ha elegido permite el desarrollo de aplicaciones mixtas que pueden ejecutarse como aplicaciones web y aplicaciones móviles. La plantilla que se presenta en la Figura. 2.34 dispone de cuatro secciones que muestran valores instantáneos tanto de la base de datos como los que se han programado directamente en el lado del framework Flutter, adicionalmente el boceto presenta características de desplazamiento con scroll.

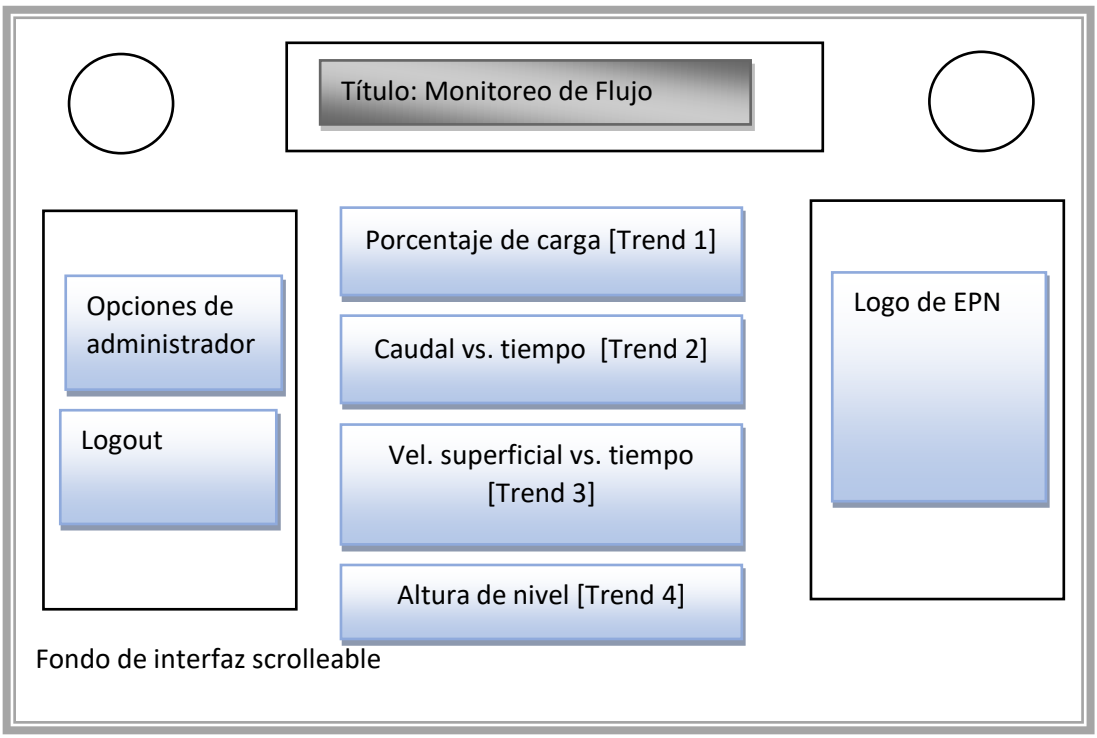


Figura *2.34* Plantilla de interfaz web de usuarios

La visualización de la interfaz debe seguir las buenas prácticas de desarrollo de tal manera que el usuario pueda contar con un despliegue limpio y directo de variables e información, para ello las tendencias en tiempo real de variables físicas ofrecen información en un rango determinado de tiempo, donde al hacer clic sobre dichas tendencias se puede observar su valor, adicionalmente se cuenta con campos alfanuméricos de visualización y campos de edición de parámetros para obtener cambios en la evolución de las variables como en el ingreso de credenciales requeridos en la pantalla de Login de la Figura. 2.35

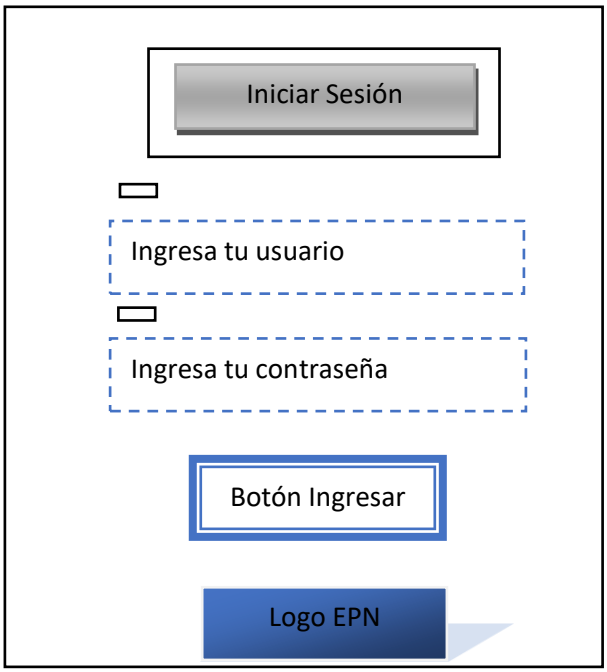


Figura *2.35* Plantilla de interfaz de Login de usuarios.

2.7.1. Diseño de Aplicación Móvil

La distribución de secciones y valores de la interfaz móvil se acoplaron a partir de la interfaz web diseñada previamente para que resulte una versión equivalente para pantallas de entre 4,5 a 5,5 pulgadas estándares en teléfonos. La navegación mantiene la versión horizontal de desplazamiento por scroll cuando se realiza la interacción con el dedo sobre la pantalla. Para la comodidad visual se mantiene el uso de un color oscuro en el background con widgets creados en armonía con el color de este último.

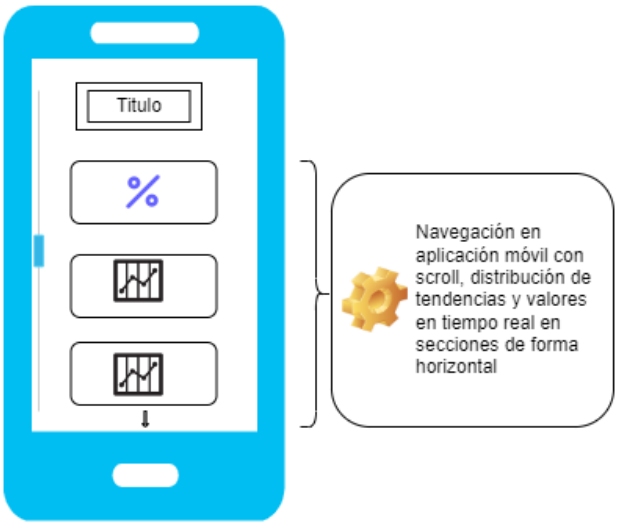


Figura *2.36* Plantilla de interfaz de aplicación móvil.

2.8. Programación de Interfaz Web

Para aplicaciones nativas que necesitan ser exportadas a un framework móvil se utiliza Flutter que ofrece el servicio de agregación a la aplicación, esto quiere decir que a través del código escrito en lenguaje de programación DART se podrá invocar el archivo main de Flutter en paralelo con la aplicación nativa [lado de servidor], donde además se vuelve posible integrar bibliotecas de terceros lo que permite centrarse únicamente en el apartado de interfaz o front end de usuario.

Los diagramas que se presentan a continuación describen la invocación de los objetos en la interfaz y la diferenciación para la conexión con los datos que reporta el servidor a través de los documentos JSON entre la aplicación móvil y la interfaz de navegación web.

El diagrama de la Fig. 2.37 describe como se define el diseño que se va a desplegar con la comprobación de las dimensiones de pantalla donde se va a ejecutar el programa, debido a que Flutter maneja el sistema back end y front end por separado se usan variables u objetos preprogramados para poder ejecutarse usando un tipo especial de variable "runtime" necesario para funcionar.

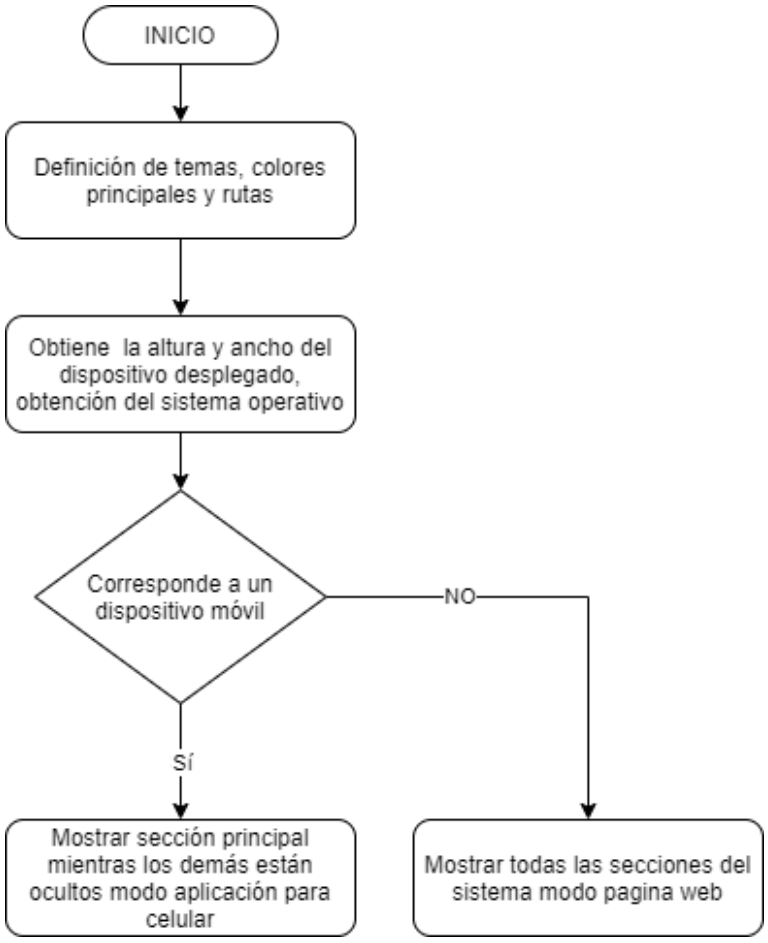


Figura 2.*37* Diagrama de flujo de despliegue de objetos runtime en aplicación hibrida.

Para el desarrollo de aplicaciones hibridas: móviles y web, se recupera información de manera asincrónica del back end nativo, y lo mismo ocurre con la comunicación, esto permite obtener mayor fluidez en el despliegue de la interfaz visual debido a que la velocidad de respuesta del servidor puede tener retardos respecto a la velocidad de despliegue de información.

Una diferenciación importante entre el despliegue de la interfaz es que en la versión web se muestra la pantalla activa, en la versión móvil las demás pantallas permanecerán ocultas.

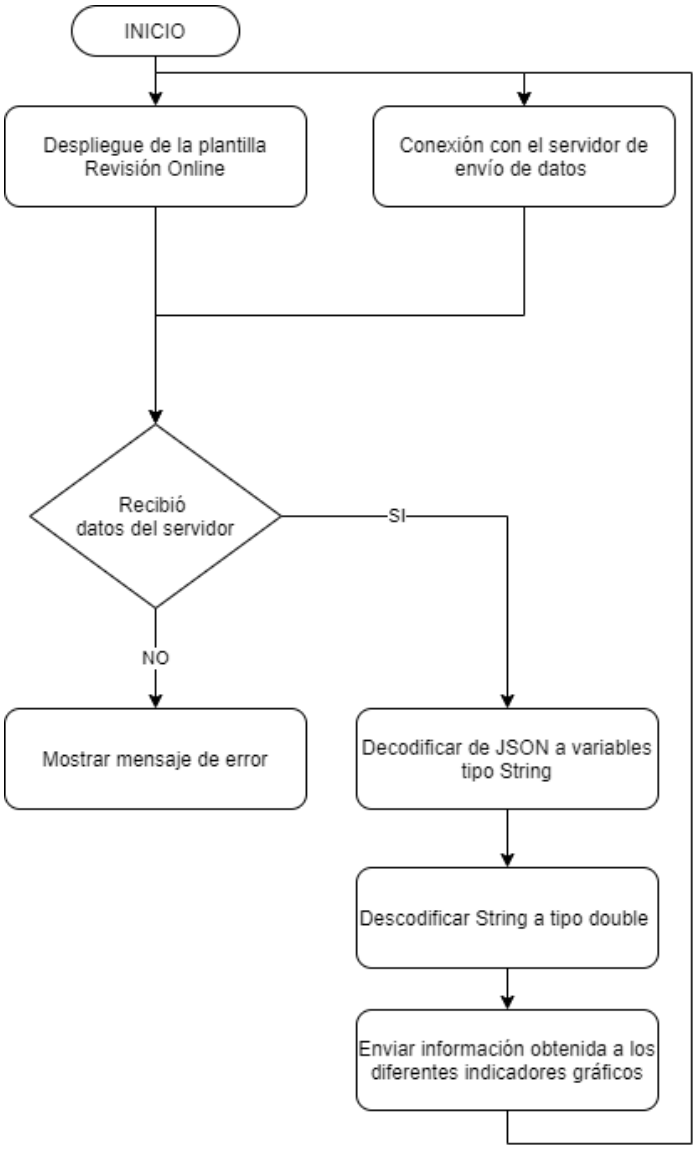


Figura *2.38* Diagrama de flujo de adquisición de documentos JSON, codificación y decodificación de listas.

El concepto que se maneja para realizar la representación gráfica de datos de documentos JSON en el framework de Flutter es la ‘deserialización’; esta operación representa la

conversión del objeto JSON que después de una operación cíclica a una lista DART, para ello se define un constructor que analizará el archivo JSON de forma dinámica y devuelve un archivo de salida donde finalmente se usa un callback que invoca un método de carga de archivos a la aplicación gráfica siempre y cuando los datos no sean NULL.

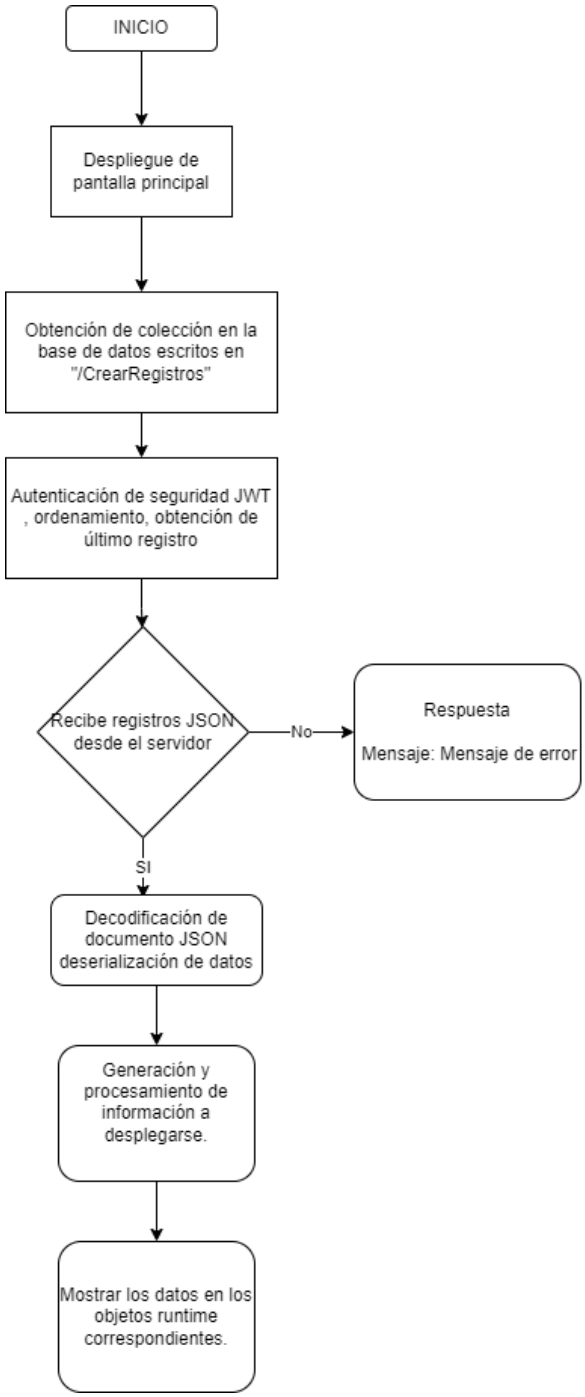


Figura *2.39* Diagrama de flujo de adquisición de documentos JSON para conversión y despliegue en interfaz.

Para dar respuesta al cliente HTTP se utiliza el paquete importado en lenguaje DART, estas respuestas contendrán los datos recibidos de forma exitosa ante una petición, las fuentes de datos asincrónicas se trabajan además con una clase o constructor que devuelve valores más discretos como una canalización estándar en base al último valor obtenido sin errores desde el servidor.

3. RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1. Resultados

3.1.1. Ensayos en laboratorio de hidráulica, toma de información, caudal-batería.

Se ha realizado cuatro ensayos de toma de medidas en el laboratorio de Hidráulica de la Escuela Politécnica Nacional para verificar la linealidad de la medida en un rango de 0 a 100 [m3/h] debido a que la potencia de la bomba es de 7.5 [kW] y la disponibilidad del agua en los depósitos de almacenamiento posee una capacidad aproximada de entre 6000 a 7000 litros, por lo que es necesario limitar la apertura de la válvula para no desbordar el flujo de agua en el canal. (ver Figura 3.1)

Figura *3.1* Laboratorio de Hidrodinámica, Facultad de Ingeniería Civil, Escuela Politécnica Nacional

Figura *3.2* Bomba Guiseley Motors 10 Hp

Las cuatro mediciones se realizaron de forma continuada en rangos específicos de tiempo, con caudales constantes, caudales promedio de operación, caudales incrementales y caudales mínimos de medida. En la Fig. 3.3 se observa la gráfica que contiene toda la información de caudal en función del tiempo y los rangos en que se divide la toma de información. (ver Figura 3.3)

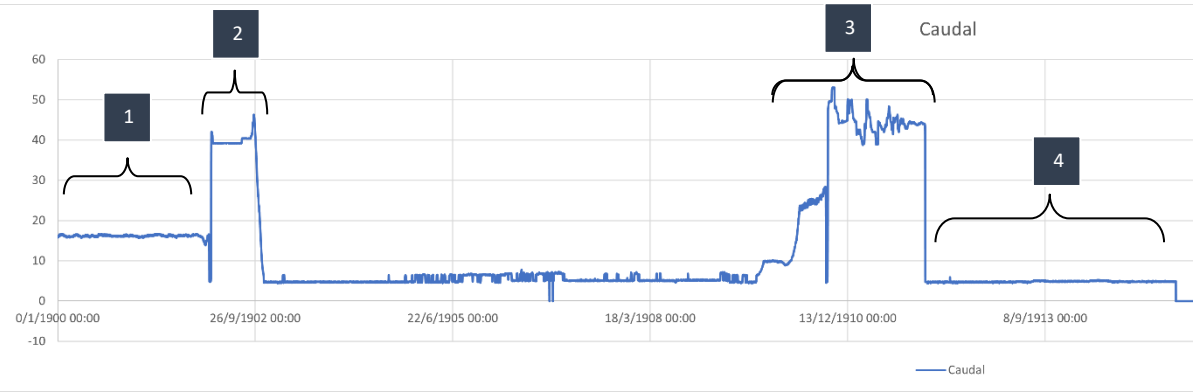


Figura *3.3* Mediciones de caudal, toma de datos en rangos de tiempo

3.1.2. Resultados de tendencias de Caudal

El primer rango de medición se obtuvo con un caudal constante de entre 16.1 a 16.5 [m3/h], se amplía el rango realizando el tratamiento de datos desde la base de datos hacia Excel y después exportándolo a Matlab, la evolución temporal de caudal se realiza en un primer intervalo de aproximadamente 800 segundos con un tiempo de refresco desde la interfaz de un segundo. Se observa la tendencia en la Figura. 3.4

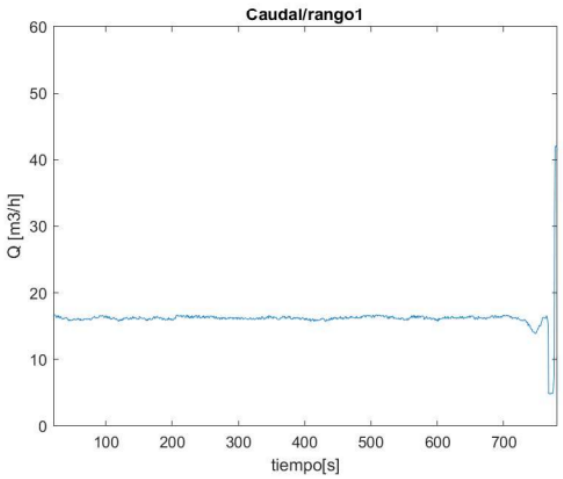


Figura *3.4* Mediciones de caudal, primer rango de medida, operación constante.

El segundo rango de medición se observa en la Figura. 3.5 en un intervalo aproximado de 300 segundos para la evolución de caudal, se ha realizado la desconexión del servidor en el segundo 768 para obtener un valor de trabajo constante de medida de 40 [m3/s], se observa que antes de estabilizarse la medición en este valor promedio de trabajo se obtiene picos por la turbulencia que se genera en la apertura de la medida y la caída de presión que eso conlleva, adicionalmente se observa que en el tiempo 1000 a 1050 [s] (Figura. 3.6), la pendiente de la curva en el rango decreciente de lectura de caudal respeta la linealidad que se ha ajustado dentro del rango de 0 a 100 [m3/h] logrando que aunque se haya realizado ajustes en el sistema de acondicionamiento y configuración del sensor, el software de presentación mantiene una naturaleza lineal en rangos de trabajo promedios para una capacidad de agua considerable

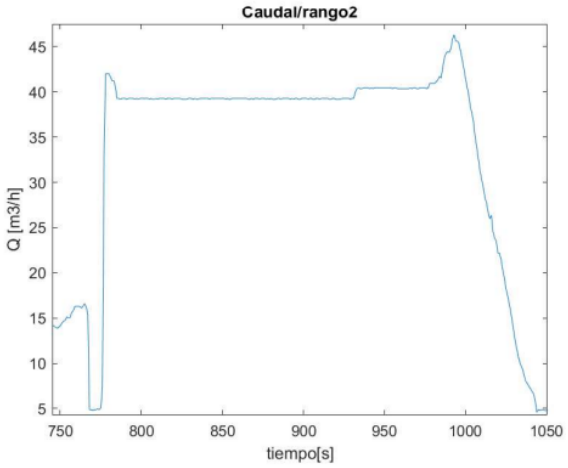


Figura *3.5* Mediciones de caudal, segundo rango de medida, operación promedio.

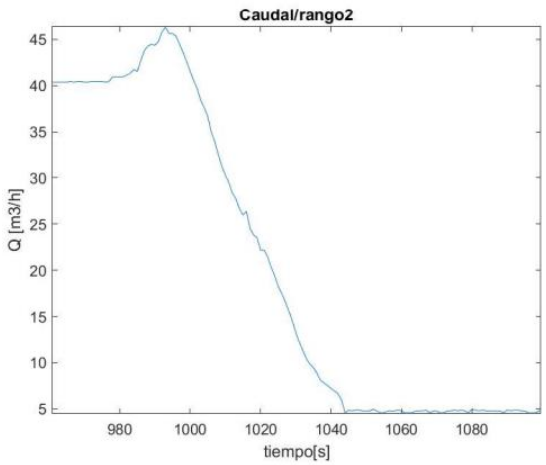


Figura *3.6* Mediciones de caudal, segundo rango de medida, medida decreciente, evolución lineal.

El tercer rango de medición se realiza en 500 segundos para operación incremental de caudal con aperturas de válvula en el segundo 3550, 3750, y 3850, se puede ver que para cambios ligeros en la adquisición del valor de caudal, el servidor recoge los datos que se registran en la base de datos mostrando tendencias suaves y lineales, sin embargo cuando los cambios en el sistema de adquisición son bruscos se produce un desfase entre la información que toma el servidor y la que llega a mostrar la interfaz web, ya que el tiempo de refresco es menor que el de muestreo, adicionalmente en los últimos 500 [s] se observa como la señal posee ruido, esto se debe a la exactitud que se maneja cuando se utiliza un rango limitado de fondo de escala en el sistema de medición, sin embargo la señal termina de estabilizarse en su valor promedio de trabajo a 45 [m3/h], como se observa en la Figura. 3.7

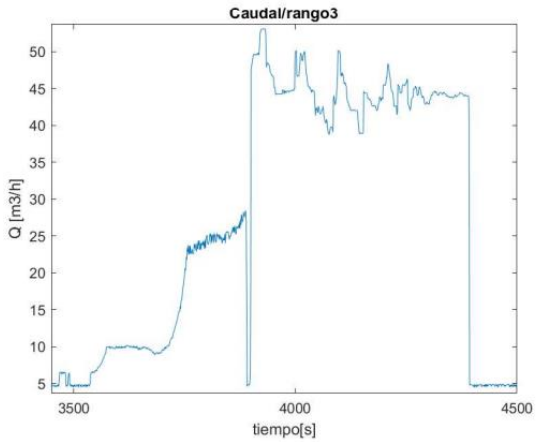


Figura *3.7* Mediciones de caudal, tercer rango de medida, medida creciente, evolución lineal, con ruidos

En la Figura. 3.8 se puede observar la visualización de un valor mínimo de medición a 4.5 [m3/s], los valores promedio y constantes de trabajo presentan una variación mínima en el despliegue de variables de las tendencias en función del tiempo. Algunos de los valores por rango de medida se presentan en el ANEXO II.

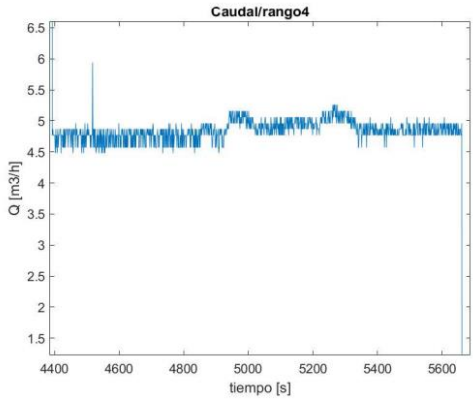


Figura *3.8* Mediciones de caudal, cuarto rango de medida, medida constante.

3.1.3. Resultados de tendencia de Batería

La supervisión del sistema de monitoreo de baterías se realizó de manera intermitente con la conexión y desconexión del sistema a diferentes tiempos con un valor promedio de entre 60% a 50% de carga total, se analizará únicamente el intervalo de 0 a 600 [s]. Figura. 3.9

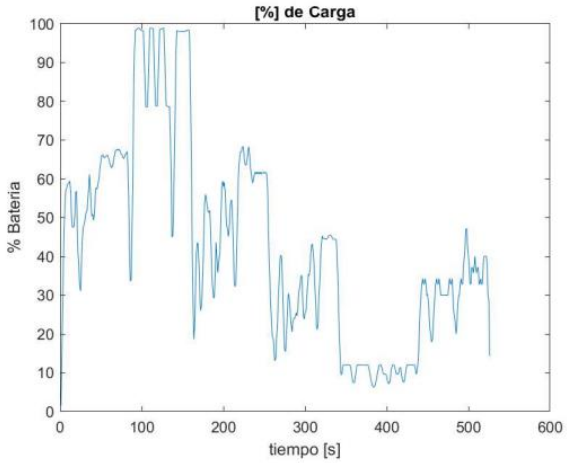


Figura *3.9* Mediciones de batería, rangos intermitentes de carga

La Figura. 3.9 muestra que en el momento de la conexión de referencias desde el sistema de adquisición al emisor asincrónico se obtiene una medida constante promedio de carga, debido a que se mantenía el sistema de medición/adquisición y una carga de 110 [Vac] conectados al mismo tiempo se muestra la tendencia del nivel de batería como se presenta dentro de la interfaz con ligeras variaciones debido a que el servidor no obtiene información entre tiempos de 20 segundos desde el la comunicación del sistema fotovoltaico, lo que causa una espera al valor siguiente de medición, sin embargo se observan valores estables promedio para la interfaz. La tabla con algunos de los valores de medida obtenidos de la base de datos se adjunta en los Anexos.

3.1.4. Documentos generados en la Base Datos

Los valores tomados se exportan desde la base de datos MongoDB con los documentos que se crean en las colecciones de información desde el servidor. En la Fig.3.10 se observa las colecciones de usuario como de datos y en la Fig. 3.11 se despliega los documentos con la información que se recoge del servidor.

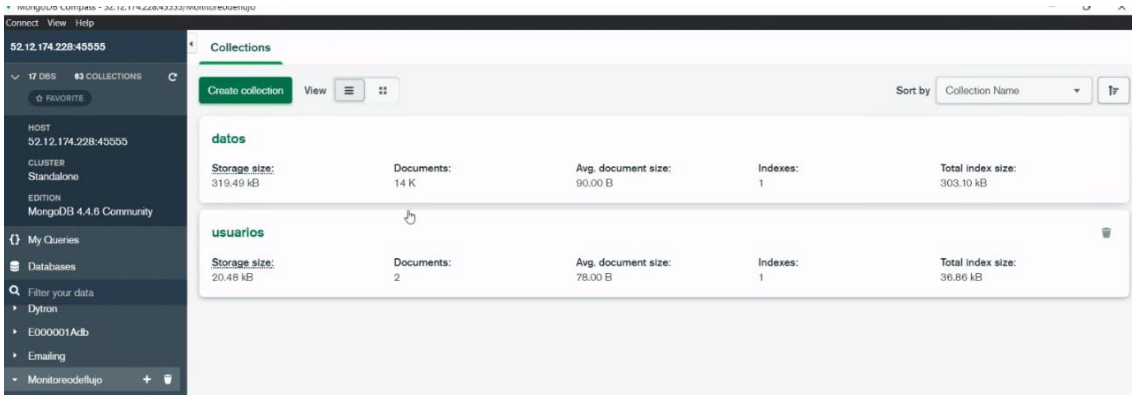


Figura *3.10* Colección de mediciones en base de datos MongoDB

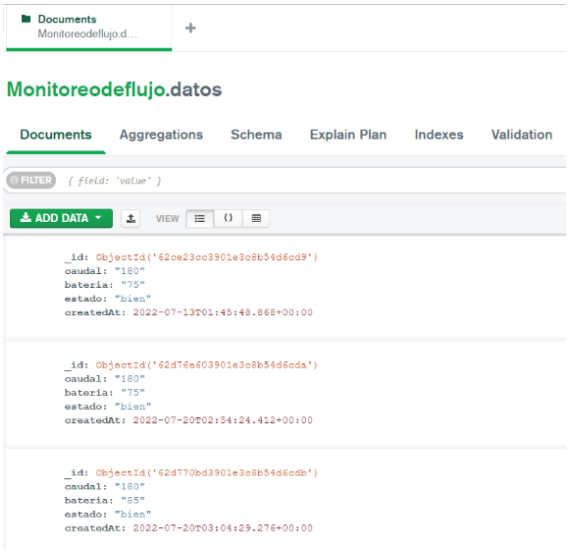


Figura *3.11* Colección valores de monitoreo en base de datos MongoDB

3.1.5. Interfaz web de supervisión-aplicativo móvil.

En la Fig. 3.12 se observa la interfaz del sistema de supervisión en la versión web donde se puede mostrar los indicadores tanto del nivel de batería, el caudal del sistema de medición-adquisición, la velocidad superficial y la altura del agua dentro de la tubería, adicionalmente existen campos de modificación de información como el diámetro, el porcentaje de inclinación de la tubería y un factor de material para el cálculo actual de la altura.

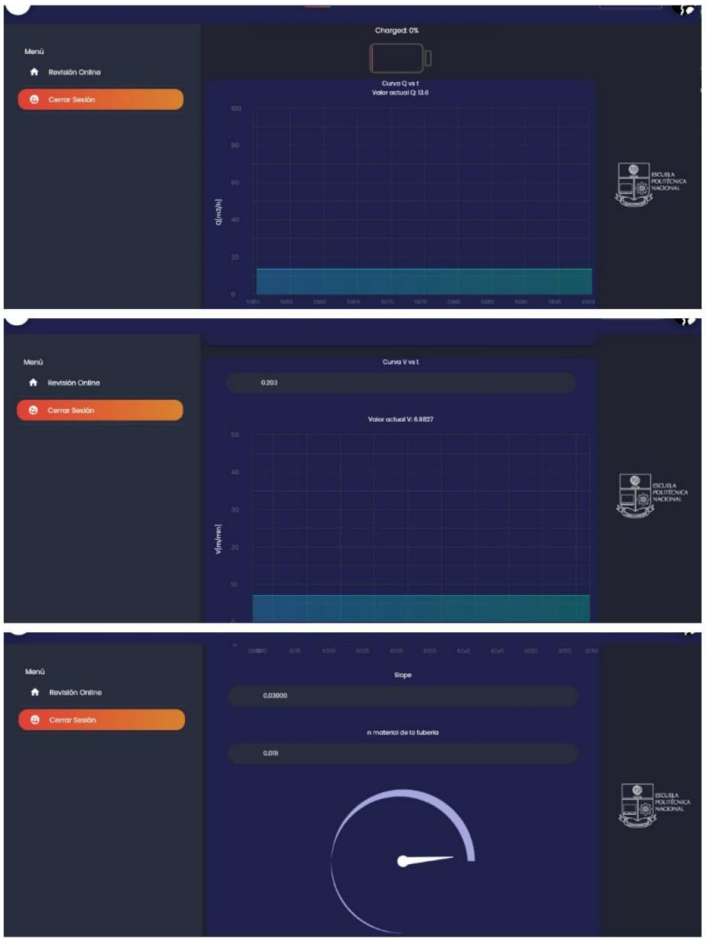


Figura *3.12* Interfaz web de monitoreo construida para verificación de valores en tiempo real

La navegación dentro de la interfaz se realiza por scroll vertical y el acceso a los valores instantáneos se hacen ubicando el cursor sobre un instante en las tendencias en función del tiempo, sin embargo, existen indicadores numéricos en la parte superior de cada widget. A continuación, se presenta el equivalente aplicativo para teléfonos móviles, se presenta un diseño similar con diseño, colores, widgets de interacción con el usuario únicamente con dimensiones diferentes a las que se presentaron con anterioridad.

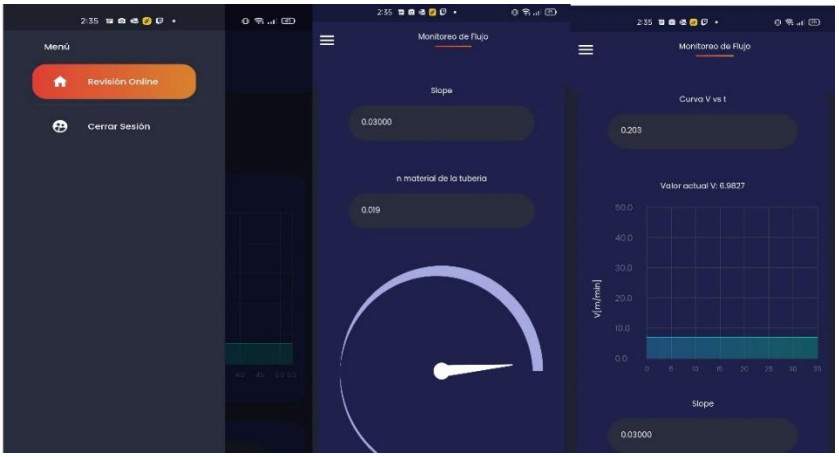


Figura *3.13* Aplicación móvil de monitoreo de tendencias y valores, compatible con Android.

3.2. Conclusiones

Es importante señalar que una vez implementado el equipo controlador central de información Gateway inalámbrico se podría dar lectura a los datos del sistema de adquisición de medida, así como al sistema de alimentación fotovoltaico, ya que una vez adquirido se puede realizar una fase de ordenamiento y filtrado en base a métodos estadísticos mismo que facilite normalizar lecturas de tal forma que la información obtenida este dentro del rango de trabajo determinando para que así el fondo de escala de medición no este pierda exactitud.

Una vez concluido el tema investigativo se puede puntualizar que el uso de algoritmos en el procesamiento de datos para la eliminación de valores atípicos ayuda aumentando la eficiencia de las medidas entregadas por el sistema de medición ya que este permite el procesamiento digital de la información y a la vez su clasificación.

Es necesario señalar que el desarrollo de una interfaz de visualización web elimina la necesidad de contar con un dispositivo o sistema operativo específico, ya que disponer de este instrumento facilita la lectura y visualización de datos permitiendo que se vuelva más accesible al usuario y contrastando la información con el instrumento patrón.

Se puede mencionar que el desarrollo de un sistema que utiliza una arquitectura de capas desacopla al mismo en distintos módulos, por lo que permite realizar operaciones independientes de cada uno y a su vez tener la escalabilidad más sencilla para actualización de tal forma que la estructura del sistema sea funcional con los servicios que ofrece.

Los lenguajes de programación seleccionados para el tratamiento de los metadatos de la aplicación se seleccionaron acorde a la magnitud del proyecto y el tipo de base de datos no relacional utilizada. Javascript y DART se enfocan puntualmente en programación orientada a objetos guardando relación directa entre la lógica del back end y el front end.

El análisis de la arquitectura que se planteó en un inicio se resolvió en el desarrollo del esquema REST para la capa de negocio, esto en base a los patrones que permite utilizar los paquetes npm dentro de los servicios web de Amazon, de esta forma el intercambio de información se mantiene seguro y unidireccional en los casos que se requiera cifrado adicional garantizando la seguridad respecto a la información que el cliente consume.

3.3. Recomendaciones

Basados en las pruebas que se realizaron en el entorno gráfico y la base de datos se obtuvo un antecedente en aplicaciones web dentro de servicios virtuales, donde para el tratamiento de información se puede optar con un esquema amplio de formatos de datos jerárquicos que se acoplan a lenguajes desde Javascript, Python, Ruby, que dependiendo de la necesidad puede optimizarse el tiempo y a su vez los costos de implementación con lo que respecta al back end o capa de procesamiento/negocio.

Se recomienda capacitar con los manuales de configuración del servidor y la base de datos a futuros clientes o usuarios de la aplicación. Adicionalmente el paquete de servicios virtuales debe elegirse en base a credenciales personales para evitar filtraciones para próximos trabajos por lo que dependerá de la capacidad de servicio que se desee contratar.

Se recomienda mantener el desarrollo de la aplicación en el entorno de Flutter debido al despliegue asincrónico de los objetos ya que existe documentación y soporte en internet además de objetos previamente creados que pueden optimizar el desarrollo del front end y evitar problemas de compilación.

En base a la estructura de los servicios web REST que se programaron en el lado del servidor se puede incorporar un mayor número de funcionalidades con códigos ampliados para solicitudes HTTP, de esta manera las consultas se pueden volver más dinámicas a la par de las URL de respuestas.

Se sugiere para un futuro trabajo la creación de una extensión de la interfaz a dispositivos con sistema operativo iOS, de esta manera no se limita el software que se maneja en los teléfonos cubriendo una mayor gama de posibles clientes.

En base al desarrollo de la aplicación web, se sugiere añadir la funcionalidad del prototipo con horarios, registros de información a modo de tablas, además de ampliar el alcance de mediciones con un mayor número de nodos si es posible capturando información con tecnologías como LoRA.

4. REFERENCIAS BIBLIOGRÁFICAS

[1] Barandica, A. y Guevara, A. (2015). Pasarela para usar transmisores HART desde una red DeviceNet. En: Ingeniería, Vol. 20, No. 1, pp. 95–109. Universidad Distrital Francisco José de Caldas. [Online]. Disponible: http://dx.doi.org/10.14483/udistrital.jour.reving.2015.1.a06

[2] Ortiz, M. y Caiza, J. (2020). Desarrollo de una red de sensores inalámbricos utilizando tecnología LORA para el monitoreo de un sistema. En: Ingeniería de Prototipos, Universidad Politécnica Salesiana, Quito. [En linea]. Disponible: https://dspace. ups.edu.ec/handle/ 123456789/18469? mode=full

[3] Shankar V. Achanta, Andy Kivi, y Ben Rowland. Extending SCADA Networks Using Wireless Communications, Schweitzer Engineering Laboratories, Inc., Raleigh, Carolina del Norte, Artículo presentado 2nd Annual PAC World Americas Conference, agosto de 2015. [En línea]. Disponible: https://cms-cdn.selinc. com/assets /Literature /Publications/Technical% 20Papers /6680_Extending SCADA_ STW_20150123_ Web2.pdf?v=20170303-171956

[4] Adler Instruments. (2020). ¿Qué es el Ruido Blanco Gaussiano Aditivo AWGN?. En: Adler-instrumentos.es. [En linea]. Disponible: https://www.adler-instrumentos.es/wp-content/uploads/2020/09/Qué-es-AWGN-Ruido-Blanco-Gaussiano-Aditivo.pdf

[5] F. Saravia. (2020). "DISEÑO DE UN SISTEMA DE COMUNICACIÓN INDUSTRIAL MULTICAPA MODBUS TCP – RS485 WIRELESS QUE PERMITA ENLAZAR ESTACIONES REMOTAS", Trabajo de Grado. En: Electrónica y Telecomunicaciones, Universidad Nacional de Piura, Piura, Perú. [En línea]. Disponible: https://repositorio. unp.edu. pe/bitstream/handle/20.500.12676/2765/IEYT-SAR-JUA-2020.pdf?sequence=1&isAllowed=y

[6] D. Mancheno. “Diseño e implementación de un prototipo de Gateway IOT industrial y adquisición de datos y su monitoreo desde un navegador web”, Escuela Politécnica Nacional, Facultad de Ingeniería Eléctrica y Electrónica. Enero. 2022. Quito. Ecuador. [En línea]. Disponible en: Repositorio Digital - EPN: https://bibdigital.epn.edu.ec/handle/15000/22346

[7] L. Cordero. (2018). ¿Qué es Ubuntu 18.04 LTS?. Trastetes., Iniciando Ubuntu. [En línea]. Disponible: https://trastetes.blogspot.com/2018/04/1-que-es-ubuntu-1804-lts.html

[8] P. Donato. (2019). SCADA-buses de campo. En: Introducción a la electrónica industrial. Univeridad Técnica del Norte. Studylib. Ibarra. Ecuador. [En linea]. Disponible: https://studylib.es/doc/8575914/introducción-a-la-electrónica-industrial-scada-–-buses-de.

[9] O. Ram. (2018). "Redes de comunicación: Topologia y enlaces", Sistemas Industriales Distribuidos., Universidad de Valencia, Valencia, España, 2015. [En línea]. Disponible: https://www.uv.es/rosado/courses/sid/Capitulo2_rev0.pdf

[10] "Interfases y Protocolos de Comunicación". (2012). En: Eveliux.com/mx. [En línea]. Disponible: http://tecsup-r5-ac-interfaces-y-protocolos.blogspot.com/

[11] "Convertidor de analógico a digital". (2016). En: Docs.espressif.com. [En línea]. Disponible: https://docs.espressif.com/projects/esp-idf/en/v4.2.1/esp32/api-reference/peripherals/adc.html

[12] J. Beningo. "USART vs UART: Conoce la diferencia". (2015). En: EDN.com. [En línea]. Disponible: https://www.edn.com/usart-vs-uart-know-the-difference/

[13] G. Tate, "Proper Use of Digitizer Front-End Signal Conditioning", SPECTRUM, p. 6. [En línea]. Disponible: https://spectrum-instrumentation.com/dl/an_optimizing_digitizer_performance_by_proper_front_end_signal_conditioning.pdf

[14] "Puerta de enlace". Adaptix Networks. (2018). [En línea]. Disponible: https://www.adaptixnetworks.com/puerta-de-enlace/#:~:text=Un%20gateway%20(puerta%20de%20enlace,redes%20con%20arquitecturas%20completamente%20distintas.

[15] ¿Qué es un gateway y por qué debería usarla en mis aplicaciones de IoT? Instrumining. [En línea]. Disponible: https://www. instrumining. com/altus-que-es-un-gateway-y-por-que-deberia-usarla-enisaplicaciones- de-iot/

[16] K. Pahlavan, Evolution and Impact of Wi-Fi Technology and Applications: A Historical Perspective. Springer Science+Business Media. (2020). [En línea]. Disponible: https://doi.org/10.1007/s10776-020-00501-8

[17] "Wi-Fi Access Gateways". Juniper Networks, Inc. (2021). [En línea]. Disponible: https://www.juniper. net/documentation /us/en/software/ junos/subscriber-mgmt-access/topics/topic-map/wifi-access-gateways.html

[18] M. Bradley. "Wireless gateway". Wikipedia. (2014). [En línea]. Disponible: https://en.wikipedia.org/wiki/Wireless_gateway

[19] "Dashboard Overview". Juniper Networks, Inc. (2021). [En línea]. Disponible: https://www.juniper.net/documentation/us/en/software/jweb-srx21.2/jweb-srx/topics/topic-map/j-web-security-dashboard.html

[20] S. Subotin. "Dashboard Design: Considerations and Best Practices". Designers. (2017). [En línea]. Disponible: https://www. toptal.com/designers/data-visualization/dashboard-design-best-practices

[21] CAPA DE PROCESAMIENTO DE DATOS". Arquitectura aplicaciones distribuidas. (2013) [En línea]. Disponible: http://arquitecturaaplicacionesdistribuidas.blogspot.com/2013/02/23-capa-de-procesamiento-de-datos_22.html

[22] "Capa de procesamiento y análisis". AWS Well-Architected. (2022). [En línea]. Disponible: https://docs.aws. amazon.com/es_es/ wellarchitected/latest/ analytics-lens/processing-and-analytics-layer.html

[23] "Aprendiendo Arduino. Arquitecturas IoT". (2018). [En línea]. Disponible: https://aprendiendoarduino.wordpress.com/tag/capa-de-procesamiento/

[24] R. Paffenroth, (2015). "Python in Data Science Research and Education", PROC. OF THE 14th PYTHON IN SCIENCE CONF, p. 7. [En línea]. Disponible: https://pdfs.semanticscholar.org/5c49/64faa7fbff1491c400b836eb2f0cd3bada80.pdf?_ga=2.185271084.406736796.165422728941712094.1654227289

[25] Dobesova, Z. (2011). Programming Language Python for Data Processing. Proceedings of International Conference on Electrical and Control Engineering (ICECE), Yichang, China, Volume 6 Institute of Electrical and Electronic Engineers (IEEE), CFP 1173J-PRT, 4866-4869 p.ISBN 978-1-4244-8163-7. [En línea]. Disponible: http://dobesova.upol.cz/wp-content/uploads/fulltext/2011_Python.pdf

[26] E. Mezna. (2019). "Arquitectura de Aplicaciones Distribuidas". [En línea]. Disponible: http://arquitecturaaplicacionesdistribuidas.blogspot.com/

[27] S. Tilkov, (2010). "Node.js: Using JavaScript to Build High-Performance Network Programs", The Functional Web, p. 4. [En línea]. Disponible: http://steve.vinoski.net/pdf/IC-Node.js.pdf

[28] T. Capan. (2017). "¿Por qué demonios usaría Node.js? Un tutorial caso por caso". Developers. [En línea]. Disponible: https://www.toptal.com/nodejs/por-que-demonios-usaria-node-js-un-tutorial-caso-por-caso

[29] "What is Node.js?" TutorialsTeachers. [En línea]. Disponible: https://www.tutorialsteacher.com/nodejs/what-is-nodejs

[30] S. Ramírez. (2017). "Diseño de Clean Architecture en NodeJS". Izertis. [En línea]. Disponible: https://www.izertis.com/es/-/blog/diseno-de-clean-architecture-en-nodejs

[31] B. Leroux. (2021). "EXPRESS.JS: PARA QUÉ SE USA Y CUÁNDO, DÓNDE USARLO PARA EL DESARROLLO DE TU APP DE NEGOCIO". Startechup. [En línea]. Disponible: https://www.startechup.com/es/blog/express-js-what-it-is-used-for-and-when-where-to-use-it-for-your-enterprise-app-development/

[32] Y. Antonieta. (2021). "Arquitectura de capas para NodeJs". Ctrly. [En línea]. Disponible: https://ctrly.blog/es/arquitectura-capas/

[33] Y. Chango. (2016). "BASES DE DATOS NO RELACIONALES: UTILIZACIÓN DE MONGO DB COMO BASE DE DATOS NO RELACIONAL EMPLEANDO FORMATO GEO JSON", Ingenieria en sistemas, p. 10. [En línea]. Disponible: https://dspace.ups.edu.ec/bitstream/123456789/17241/4/UPS%20-%20ST002829.pdf

[34] L. Schaefer. (2019). "Connect to a MongoDB Database Using Node.js". Mongodb. [En línea]. Disponible: https://www.mongodb.com/blog/post/quick-start-nodejs-mongodb-how-to-get-connected-to-your-database

[35] "MongoDB JavaScript tutorial". (2022). Zetcode. [En línea]. Disponible: https://zetcode.com/javascript/mongodb/

[36] M. Miroslav. (2020). "Using Flutter framework in multi-platform application implementation", Bachelor's, Universidad Masaryk. [En línea]. Disponible: https://is.muni.cz/th/on45r/bachelors_thesis.pdf

[37] E. Müller. (2021) "Web Technologies on the Desktop: An Early Look at Flutter", Bachelorarbeit, Universidad de Stuttgart. [En línea]. Disponible: https://d-nb.info/1234985535/34

[38] S. Faust. (2020). "Using Google´s Flutter Framework for the Development of a Large-Scale Reference Application", Trabajo de Grado, Universidad de Cologne. [En línea]. Disponible: https://epb.bibl.th-koeln.de/frontdoor/deliver/index/docId/1498/file/flutter-for-the-dev-of-large-scale-ref-app.pdf

[39] E. W. Dijkstra. (1968). 'The structure of the "THE"- multiprogramming system'. [En línea]. Disponible: https://www.cs.utexas.edu/users/EWD/transcriptions/EWD01xx/EWD196.html

[40] B. Martin. (2015). 'The Principles of Clean Architecture', King Street, Norwich. [En línea]. Disponible: https://www.techtarget.com/searchapparchitecture/tip/A-primer-on-the-clean-architecture-pattern-and-its-principles

[41] D. Villegas. (2020). "DESARROLLO DE UNA APLICACIÓN MÓVIL UTILIZANDO FLUTTER SDK DE GOOGLE PARA PROMOCIONAR EL ARTE DE LA CIUDAD DE GUAYAQUIL", Trabajo de Grado, Universidad de Guayaquil, Guayaquil, 2020. [En línea]. Disponible: http://repositorio.ug.edu.ec/bitstream/redug/48911/1/B-CISC-PTG-1746-2020%20Loor%20Muñoz%20Kevin%20Alexander%20-%20Villegas%20Salazar%20Diego%20Armando.pdf

[42] P. Franco. (2020). "DESARROLLO DE UN PROTOTIPO MÓVIL BASADO EN REALIDAD AUMENTADA COMO HERRAMIENTA DE APRENDIZAJE INTERACTIVA EN EL ÁREA DE FINANZAS APLICADO EN ESTUDIANTES DE BACHILLERATO", Trabajo de Grado, Universidad Autónoma De Bucaramanga, Bucaramanga. [En línea]. Disponible: https://repository.unab.edu.co/bitstream/handle/20.500.12749/14410/2020_Tesis_Bibiana_Patricia_Franco_Florez.pdf?sequence=1&isAllowed=y

[43] B. H. B. H. L. Chevarría. (2018). "Rest Api for management of electronic devices," Universidad Peruana de Ciencias Aplicadas (UPC), Lima, Perú. [En línea]. Disponible: https://doi.org/10.19083/tesis/624358

[44] G. Valencia. (2018). "ANÁLISIS DE FRAMEWORKS DE DESARROLLO DE API REST Y SU IMPACTO EN EL RENDIMIENTO DE APLICACIONES WEB CON ARQUITECTURA SPA", Tesis de Masterado, Universidad Técnica del Norte, Ibarra. [En línea]. Disponible: http://repositorio.utn.edu.ec/bitstream/123456789/8264/1/PG%20659%20TESIS.pdf

[45] T. Nguyen. (2021). "Node.js Express Login example with MySQL database". dev.to. [En línea]. Disponible: https://dev.to/tienbku/node-js-express-login-example-with-mysql-database-2n51

[46] L. Ortiz. (2014). "Propuesta para la oferta del servicio de cloud computing por parte de la empresa computadores y equipos COMPUEQUIP DOS S.A en la ciudad de Cuenca.", Trabajo de Grado, Universidad Politécnica Salesiana de Cuenca, Cuenca. [En línea]. Disponible: https://dspace.ups.edu.ec/bitstream/123456789/5856/1/UPS-CT002815.pdf

[47] A. Folch. (2011). "Interface development for Eucalyptus based cloud", Trabajo de Grado, VILNIUS GEDIMINAS, Vilnius. [En línea]. Disponible: https://upcommons.upc.edu/bitstream/handle/2099.1/14597/70010.pdf

[48] R. Saini. (2013). An Introduction to AWS – EC2 (Elastic Compute Cloud). Proceedings of the International Conference on Research in Management & Technovation. [En línea]. Disponible: https://www.researchgate.net/publication/348638365_An_Introduction_to_AWS-EC2_Elastic_Compute_Cloud

[49] E. Mungai. (2011). "Cloud Computing With Amazon S3 Using the Zend Framework", Trabajo de Grado, Universidad Metropolitana de Helsinky. [En línea]. Disponible: https://www.theseus.fi/ bitstream/ handle/10024/30580 /finalthesis.pdf?sequence=1&isAllowed=y

[50] F. Ocaña. (2005). "Tratamiento estadistico de outliers y datos faltantes", Materiales Doctorado UGR, p. 6. [En línea]. Disponible:

https://www.ugr.es/~fmocan/MATERIALES%20DOCTORADO/Tratamiento%20de%20outliers%20y%20missing.pdf

[51] Tukey, John W. (2011). Exploratory Data Analysis. Addison-Wesley. (1977) ISBN 0-201-07616-0. OCLC 3058187. [En línea]. Disponible: https://www.stat.berkeley.edu/~brill/Papers/EDASage.pdf [52] E. Mungai. (2011). "Cloud Computing With Amazon S3 Using the Zend Framework", Trabajo de Grado, Universidad Metropolitana de Helsinky. [En línea]. Disponible: https://www. theseus.fi/ bitstream/ handle/10024/30580 /finalthesis.pdf?sequence=1&isAllowed=y

[53] V. Yepes Piqueras. (2022). "¿Qué hacemos con los valores atípicos (outliers)?", Universidad Politécnica de Valencia, p. 2. [En línea]. Disponible: https://victoryepes.blogs.upv.es/2022/02/21/que-hacemos-con-los-valores-atipicos-outliers/

[54] L. Willerman. (1991). "Exploración de datos: Contraste de Grubbs". BioMates. [En línea]. Disponible: https://www. sgapeio.es/ INFORMEST/VICongreso/taller/applets/biomates/explora/explora_grubbs/explora_grubbs.htm

[55] "Queue Data Structure and Implementation in Java, Python and C/C++". Programiz: Learn to Code for Free. [En línea]. Disponible: https://www.programiz.com/dsa/queue

[56] A. Vassaloti. (2022). "queue-clase de cola sincronizada — documentacion". 3.10.6 Documentation. [En línea]. Disponible: https://docs.python.org/es/3/library/queue.html

[57] M. Willatzen. (2004). "Flow acoustics modelling and implications for ultrasonic flow measurement based on the transit-time method", Ultrasonics, vol. 41, n. º 10, pp. 805–810. [En línea]. Disponible: https://doi.org/10.1016/j.ultras.2003.12.044

[58] L. Guosheng y L. Mingwei. (2021). "Research on Improving the Accuracy of the Ultrasonic Flow-Meter with Time Difference Method", IOP Conference Series: Earth and Environmental Science, vol. 844, n. º 1, p. 011001. [En línea]. Disponible: https://doi.org/10.1088/1755-1315/844/1/011001

[59] "Conceptos sobre Linea de Vista". WNI Mexico - Wireless Solutions.[En línea]. Disponible: https://www. wni.mx/index. php/wireles stech/50-los

[60] R.-d. Wang, Q. Liu, C.-t. Du y L. Yao. (2013) "A Signal Pre-processing Algorithm Applied for Ultrasonic Flow-Meter", TELKOMNIKA Indonesian Journal of Electrical Engineering, vol. 11. [En línea]. Disponible: https://doi.org/10.11591/telkomnika.v11i9.3282

[61] V. Adajania, M. Agrawal. (2011). Embedded Web Server Application Based Automation and Monitoring System. 2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies. p. 634–637. [En línea]. Disponible: doi:10.1109/icsccn.2011.6024628

[62] A. W. S. Inc., “Tipos de instancias de Amazon EC2,” Amazon Web Service Inc., 2020. [En línea]. Disponible en: https://aws. amazon.com/es/ec2/instancetypes/#General _Purpose. [Último acceso: 19-Mar-2020].

[63] A. Aguirre. (2020). "Desarrollo de un sistema protoipo web y móvil para la notificación de eventos académicos a estudiantes que pertenecen a las carreras del departamento de Eletrónica

y Telecomunicaciones y redes de información de la Escuela Politécnica Nacional empleando el protocolo MQTT.", Trabajo de Grado, Escuela Politécnica Nacional, Quito.

[64] A. Wasserman, “Software Engineering Issues for Mobile Application Development,” Septiembre 2010. [En línea]. Disponible en: https://dl.acm.org/doi/pdf/10.1145/1882362.1882443.

[65] P, Giménez. (2015). "Aplicaciones de SWE en entornos industriales", Tesis doctoral, Universidad Politécnica de Valencia. [En línea]. Disponible: https://riunet.upv.es/bitstream/handle/10251/51282/GIMÉNEZ%20-%20Aplicaciones%20de%20SWE%20en%20entornos%20industriales.pdf?sequence=1

[66] K. Sreenivasa. (2013). "Diseño e implementación de una arquitectura de servidor web embebido para red de sensores inalámbricos.", Revista Internacional de Investigación en Ingeniería y Tecnología. [En línea]. Disponible: https://www.academia.edu/6934991/DESIGN_AND_IMPLEMENTATION_OF_AN_ANCRCHITECTURE_OF_EMBEDDED_WEB_SERVER_FOR_WIRELESS_SENSOR_NETWORK

[67] P. J. Eby. (2010). "Python Web Server Gateway Interface v1.0". peps.python.org. [En línea]. Disponible: https://peps.python.org/pep-0333/

[68] Guerrero, E. (2020). Diseño y desarrollo de un sistema de prototipo de reconocimiento facial sobre infraestructura FOG Computing en una arquitectura de microservicios montada en contenedores docker ejecutadas en una instancia de infraestructura distribuida de en Opennebula, aplicación movil android a los medios de transporte público. Trabajo de Grado, Universidad Politécnica Salesiana de Cuenca. [En línea]. Disponible: https://dspace.ups.edu.ec/bitstream/123456789/19854/4/UPS-CT008973.pdf

[69] "Build forum from scratch: Web Server Gateway Interface". (2020). Developaper. [En línea]. Disponible: https://developpaper.com/build-forum-from-scratch-2-web-server-gateway-interface/

[70] J. Torres del Rey. (2014). "La localización de webs dinámicas: objetos, métodos, presente y futuro", Investigación, Universidad de Salamanca. [En línea]. Disponible: https://www.jostrans.org/issue21/art_torres_rodrigue.pdf

5. ANEXOS

ANEXO I

Conjunto de Datos de Batería provenientes de la base MongoDB

Debido a la cantidad masiva de datos se opta por presentar un conjunto de datos de la adquisición y monitoreo para tiempos de 30 [s] en el caso del porcentaje de batería.

tiempo [s]	Battery [%]
0	1.431552588
1	8.964106845
2	26
3	36.91368948
4	47.4
5	54.22153589
6	56.91619366
7	57.6
8	58.51285476
9	58.80083472
10	59
11	59.48547579
12	56.07796327
13	50.72203673
14	47.48631052
15	47.71285476
16	47.54357262
17	50.92203673
18	56.47796327
19	56.82153589
20	48.74357262
21	40.83489149
22	36.94106845
23	32.96761269
24	31.01168614
25	36.00083472
26	43.47796327
27	47.17429048
28	47.88297162
29	48.91619366
30	49.82821369

ANEXO II.

Conjunto de Datos de Caudal provenientes de la base MongoDB

Tiempo [s]	Caudal [m3/h]
775	18.51715359
776	25.9291202
777	33.17310184
778	38.67230551
779	41.27209349
780	41.58106177
781	41.25470451
782	40.80233222
783	40.32717863
784	39.90515359
785	39.54693823
786	39.32529549
787	39.24684641
788	39.24430718
789	39.24684641
790	39.252
791	39.252
792	39.252
793	39.252
794	39.24684641
795	39.23915359
796	39.234
797	39.23915359
798	39.24684641
799	39.25715359
800	39.26484641
801	39.26484641
802	39.25715359
803	39.24684641
804	39.234
805	39.22115359
806	39.216
807	39.216
808	39.22115359
809	39.234
810	39.24684641
811	39.252
812	39.252
813	39.252
814	39.252

Caudal medido, obtenido de la base de datos.

ANEXO III.

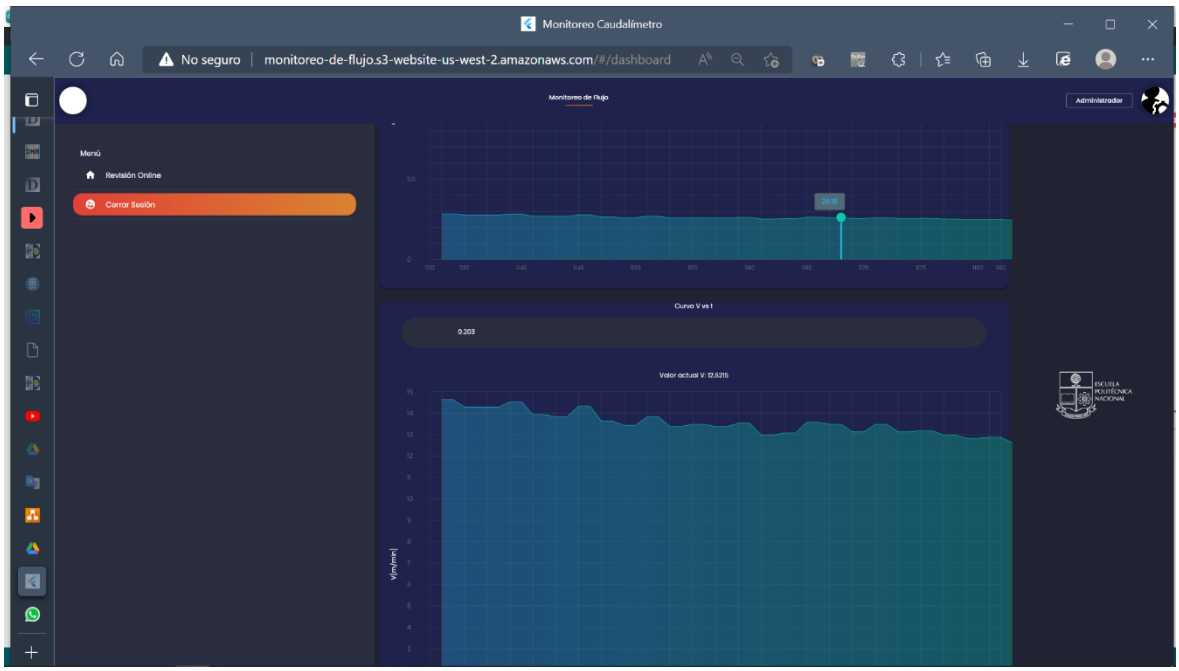
Conjunto de Datos de velocidad superficial provenientes de interfaz gráfica

Tiempo [s]	Q[m3/h]	V/Sup [m/h]
3721	11.26033222	53.87718766
3722	11.414	54.61244019
3723	11.60145242	55.50934173
3724	11.81115359	56.5126966
3725	12.04057262	57.61039532
3726	12.27027379	58.70944397
3727	12.48627379	59.74293679
3728	12.6988798	60.76019043
3729	12.91684641	61.80309287
3730	13.13170117	62.83110607
3731	13.35490484	63.89906623
3732	13.60027379	65.07308033
3733	13.86	66.31578947
3734	14.114	67.53110048
3735	14.3571202	68.69435503
3736	14.58342738	69.77716449
3737	14.78139399	70.72437316
3738	14.99351085	71.73928637
3739	15.24742738	72.95419799
3740	15.56290484	74.46365953
3741	15.90627379	76.10657316
3742	16.30866444	78.03188728
3743	16.74394157	80.11455296
3744	17.21578464	82.37217532
3745	17.69854758	84.68204583
3746	18.19857262	87.07451015
3747	18.71548581	89.54777899
3748	19.22933556	92.00639024
3749	19.69884641	94.25285364
3750	20.17727045	96.54196388
3751	20.68588314	98.97551741
3752	21.20578464	101.4630844
3753	21.70854758	103.8686487
3754	22.2388798	106.4061234
3755	22.63428047	108.2979927
3756	22.86578464	109.4056681
3757	22.97994157	109.9518735
3758	23.04706177	110.2730228
3759	23.0388798	110.2338746
3760	23.01945242	110.1409207
3761	23.02742738	110.1790784
3762	22.99618197	110.0295788
3763	22.97360267	109.9215439
3764	22.92579132	109.6927814

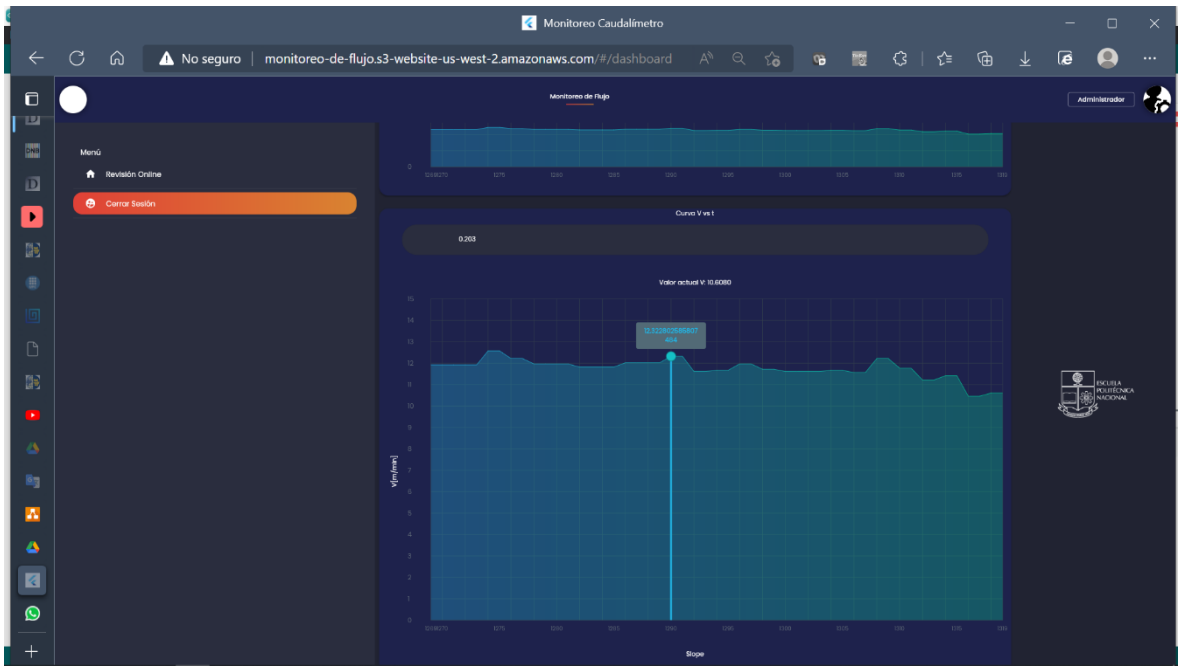
ANEXO IV

Ensayos de laboratorio y pruebas de Aplicación he interfaz web

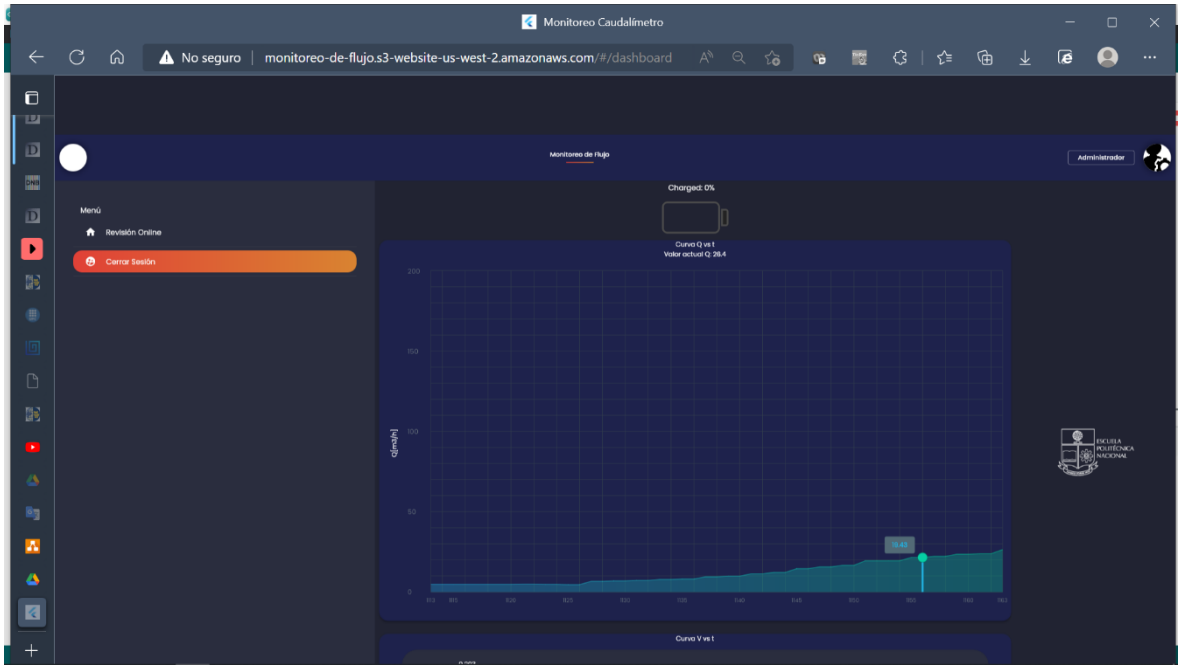
Se presentan algunos de los ensayos donde se encuentra las tendencias de caudal, velocidad superficial y altura de fluido en la tubería.



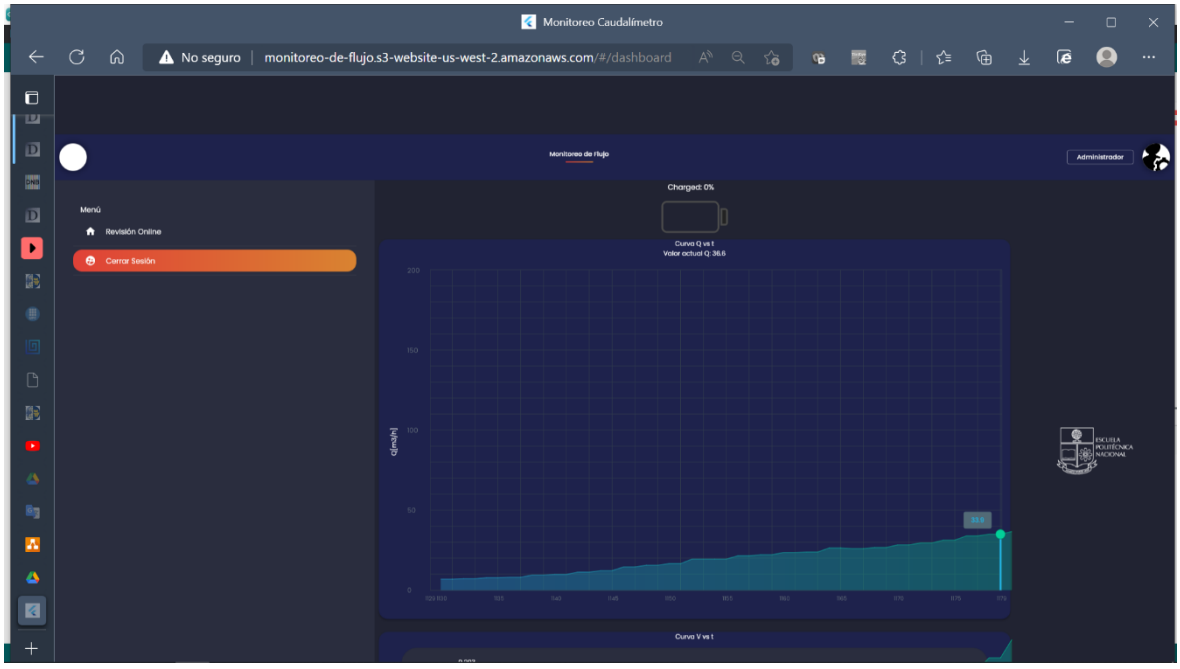
Toma de información en interfaz, caudal=26.8[m3/s].



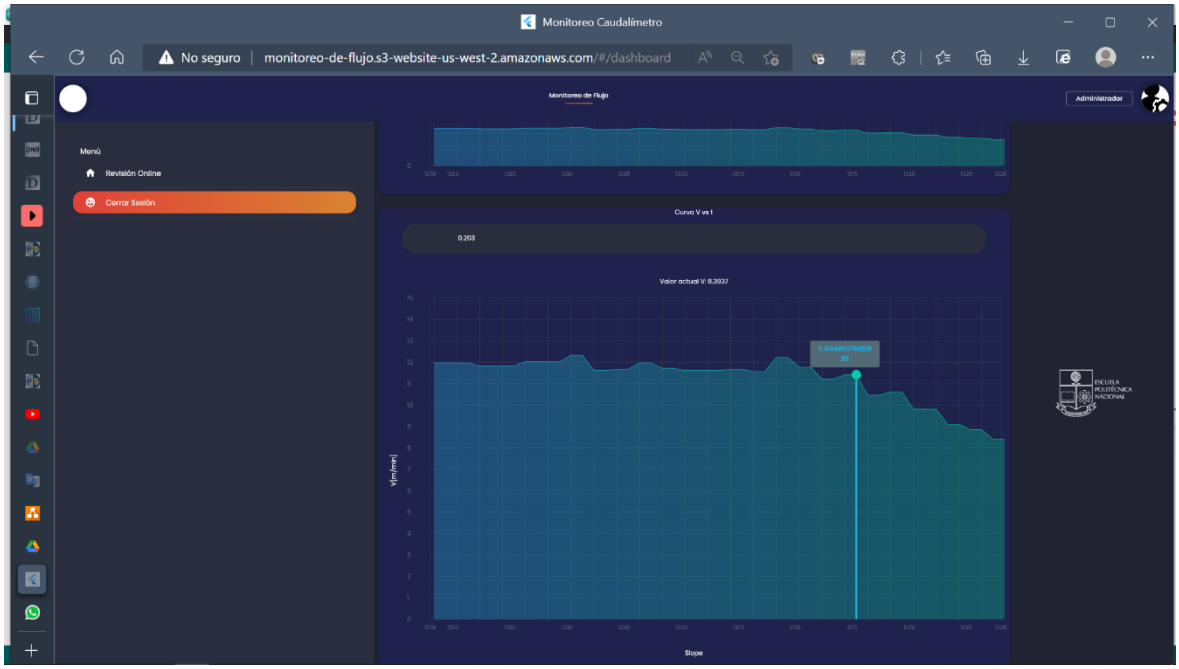
Toma de información en interfaz, velocidad superficial=12.32 [m/s].



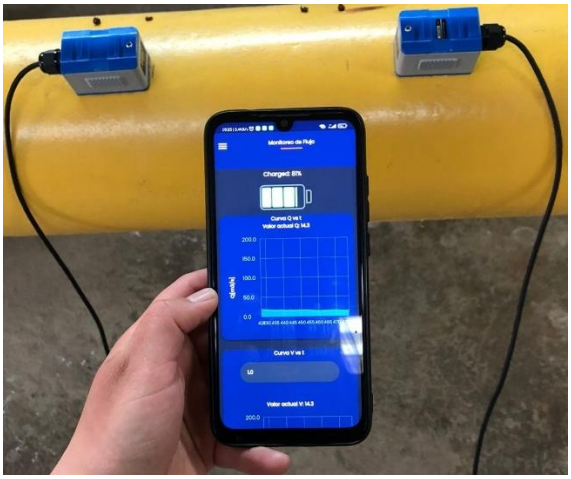
Toma de información en interfaz, caudal=19.43[m3/s].



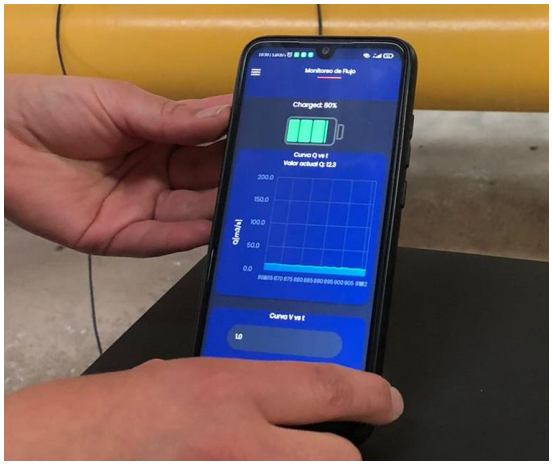
Toma de información en interfaz, caudal=33.9[m3/s].



Toma de información en interfaz, velocidad superficial=11.41 [m/s].



Toma de información en interfaz móvil, batería=81 [%]



Toma de información en interfaz móvil, batería=80 [%]

Toma de información en interfaz móvil, batería=80 [%]

ANEXO V

MANUAL DE USUARIO

Descripción General

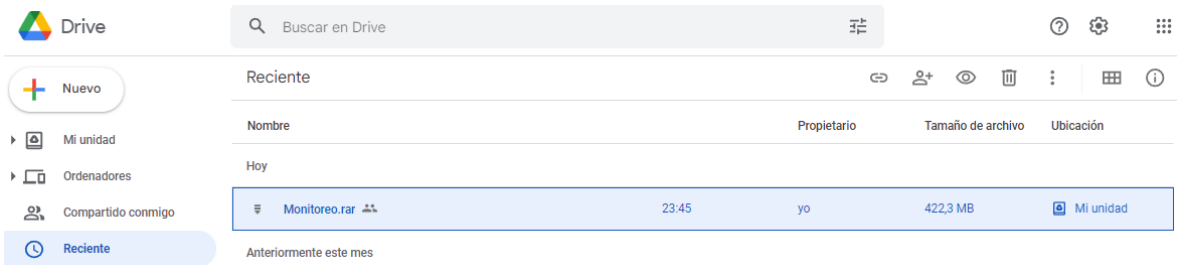
La interfaz web o aplicación móvil a utilizar tiene como objetivo prescindir de forma general de un dispositivo en especifico o sistema operativo puntual para ejecutarse de forma adecuada, por lo que se detalla tanto la instalación de la aplicación como el acceso con credenciales unicas a la pagina web. Sin embargo de requerirse acceso o modificaciones directamente en el servidor y la base de datos se detalla como se debe realizar el acceso paso a paso.

- **Instalación de aplicación**

El enlace en donde se ha dejado disponible tanto el codigo de la interfaz gráfica como la aplicación que se ha lanzado desde Visual Studio Code se puede acceder mediante el siguiente enlace:

https://drive.google.com/file/d/1DaGmQse3SqabE8eZ8fHf0VHwV7h8MlZw/view?usp=sharing

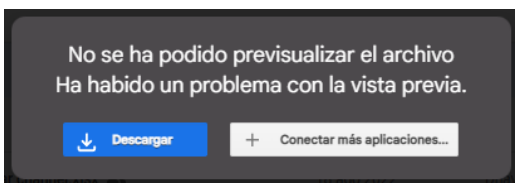
Al dar clic sobre el enlace se despliega en el navegador la ventana de Google Drive con el archivo llamado “Monitoreo” cuyo contenido tiene dos elementos: una carpeta con nombre: Monitoreo_Flujometro que contiene el código de todo el componente gráfico y la conexión JSON de comunicación de documentos con el servidor, y ademas la aplicación previamente lanzada a través de Visual Studio Code y NodeJS.



Archivo Comprimido de aplicación y código de interfaz gráfica.

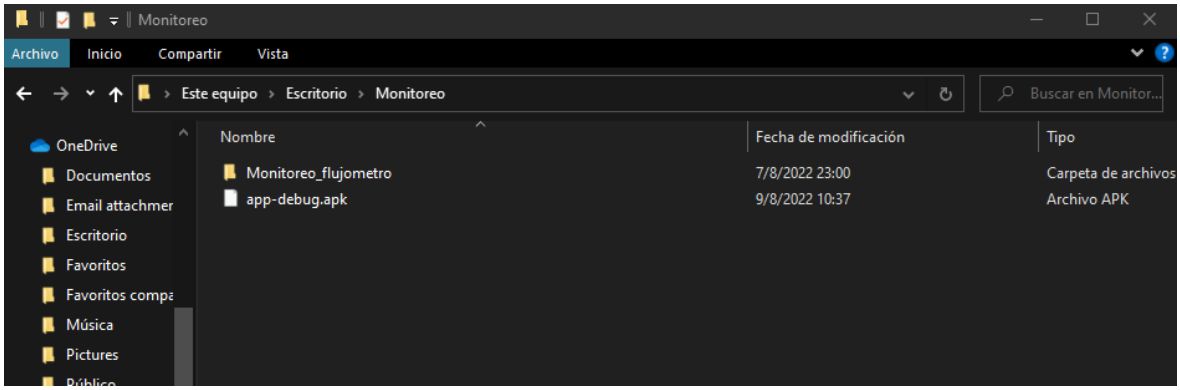
Paso 1:

Se deberá hacer clic sobre el archivo y se presionará descargar, el archivo tardará unos minutos en mostrarse en la carpeta descargas, y posteriormente se lo descomprimirá



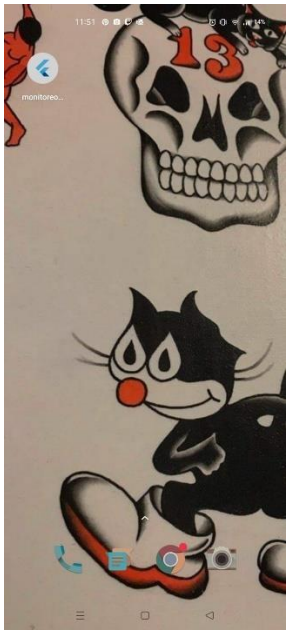
Paso 2:

Una vez descomprimido se tiene dos archivos, seleccionamos el archivo “appp-debug.apk” cuya extensión nos permite instalarlo dentro de dispositivos Android.



Paso 3:

Una vez que se obtiene la aplicación basta con transferirla al telefono por cable USB e instalarla desde el almacenamiento interno, donde el el icono de la aplicación previa a ejecutarse se debe visualizar:



Paso 4:

Al presionar sobre el icono de la aplicación se mostrará la primera pantalla de Login, la información que se requiere se ha establecido para dos usuarios, sin embargo para el uso libre de la aplicación se proporciona el siguiente nombre y contraseña generico:

Usuario	admin
Contraseña	abc123

Paso 5:

Dentro de la aplicación se puede desplazar de forma vertical las 4 tedencias de información para verificar los valores en tiempo real si el sistema se encuentra en sincronización con el dispositivo DAQ, caso contrario se visualizará el último valor medido y almacenado en la base de datos del servidor.

Paso 6:

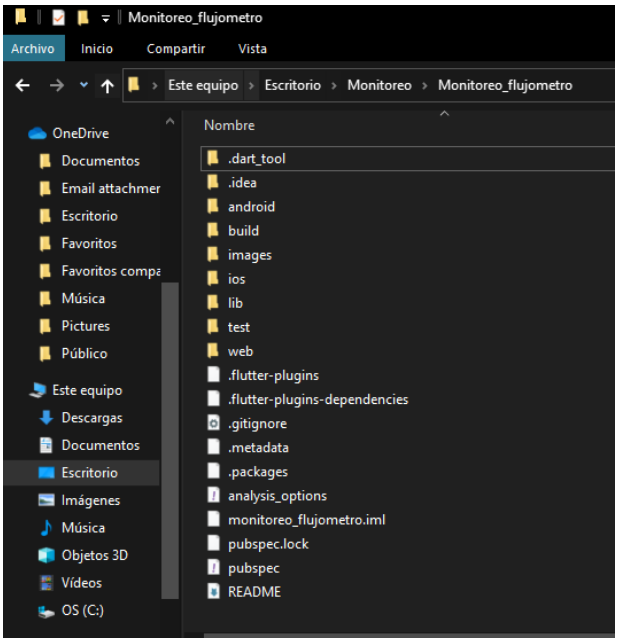
Si no se cuenta con un móvil, tambien se tiene la alternativa web mediante un navegador, el siguiente enlace generado desde el servidor se usa para ingresar directamente a la página web mediante las mismas credenciales indicadas en el paso 4:

Monitoreo Caudalímetro (monitoreo-de-flujo.s3-website-us-west-2.amazonaws.com)

- **Acceso a modificaciones en la interfaz web**

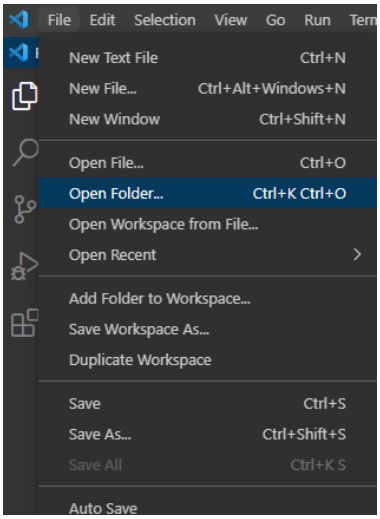
Paso 1:

De requerirse cambios como agregar, modificar, quitar y esquematizar desde cualquier punto los componentes de la interfaz gráfica se accede a la carpeta descomprimida: monitoreo_flujometro, el contenido puede abrirse para su adición desde cualquier IDE compatible con Javascript, sin embargo se recomienda al usuario final usar Visual Code Studio por su compatibilidad con el node package manager.



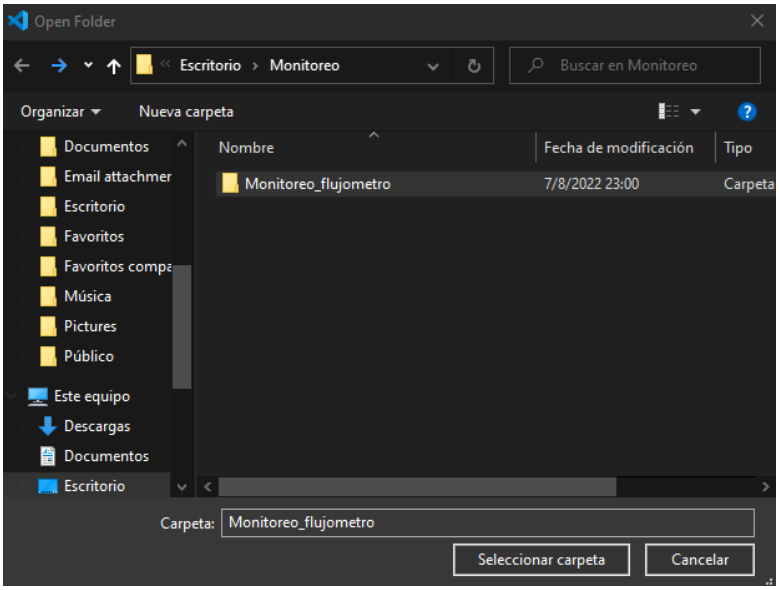
Paso 2:

La experiencia en el uso de Visual Code Studio queda a estudio del lector ya que abarca toda una temática respecto a distintas aplicaciones que salen de la capacidad de este texto, sin embargo si el usuario cuenta con la aplicación instalada basta con acceder a la pestaña File, y seleccionar: Open Folder, donde se buscará la ruta del código descargado anteriormente.



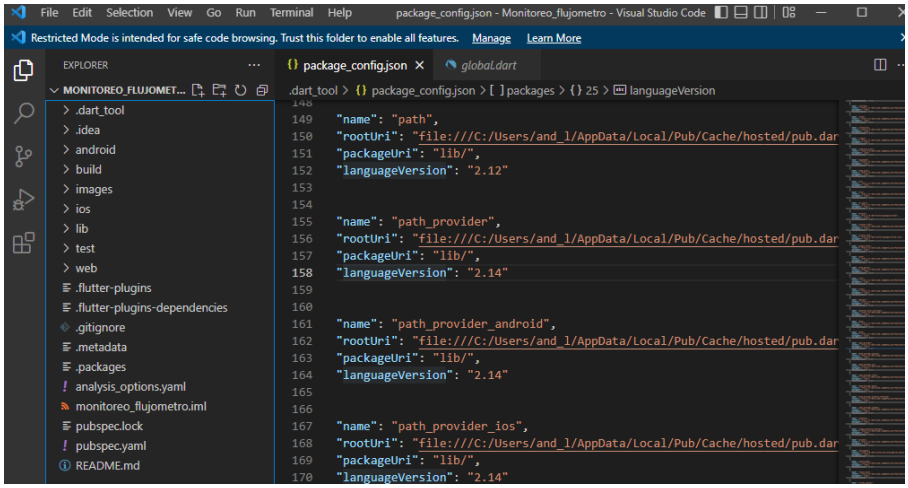
Paso 3:

Una vez ubicados sobre la ruta indicada, presionamos el boton de seleccionar carpeta.



Paso 4:

Los metadatos que se han creado se encuentran en el arbol de busqueda que se despliega en la parte izquierda de la ventana, la estructura del código se realiza en base a programación orientada a objetos, el contenido de cada una de las secciones se encuentra ordenado por las herramientas necesarias para la programación, el paquete de documentos JSON leidos desde el servidor, librerias útiles de protocoloo HTTP, de lenguaje DART, y de construcción de elementos gráficos, pantallas de tendencias, las imagenes anexadas, entre otros. El código se libera de manera global para la manipulación de la aplicación


File Edit Selection View Go Run Terminal Help
package_config.json - Monitoreo_flujometro - Visual Studio Code
Restricted Mode is intended for safe code browsing. Trust this folder to enable all features. Manage Learn More
EXPLORER
MONITOREO_FLUJOMET...
.dart_tool
.idea
android
build
images
ios
lib
test
web
.flutter-plugins
.flutter-plugins-dependencies
.gitignore
.metadata
.packages
analysis_options.yaml
monitoreo_flujometro.iml
pubspec.lock
pubspec.yaml
README.md
package_config.json
global.dart
.dart_tool > package_config.json > [ ] packages > { } 25 > languageVersion
"name": "path",
"rootUri": "file:///C:/Users/and_l/AppData/Local/Pub/Cache/hosted/pub.dar
"packageUri": "lib/",
"languageVersion": "2.12"
"name": "path_provider",
"rootUri": "file:///C:/Users/and_l/AppData/Local/Pub/Cache/hosted/pub.dar
"packageUri": "lib/",
"languageVersion": "2.14"
"name": "path_provider_android",
"rootUri": "file:///C:/Users/and_l/AppData/Local/Pub/Cache/hosted/pub.dar
"packageUri": "lib/",
"languageVersion": "2.14"
"name": "path_provider_ios",
"rootUri": "file:///C:/Users/and_l/AppData/Local/Pub/Cache/hosted/pub.dar
"packageUri": "lib/",
"languageVersion": "2.14"