Diseño de una plataforma IoT para el monitoreo del nivel, velocidad y estado meteorológico en un cauce que permita dar alerta temprana de desastres en puntos estratégicos

Orlando Alberto Moncada Rodríguez y Andrés Felipe Uribe García

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomécanicas

Escuela de Ingeniería de Sistemas e Informática

Bucaramanga

2022

Diseño de una plataforma IoT para el monitoreo del nivel, velocidad y estado meteorológico en un cauce que permita dar alerta temprana de desastres en puntos estratégicos

Orlando Alberto Moncada Rodríguez y Andrés Felipe Uribe García

Trabajo de Grado para optar el título de Ingeniero de Sistemas

Director

Ing. José Geralbert Rubiano

Especialista en Redes de Computadoras

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomécanicas

Escuela de Ingeniería de Sistemas e Informática

Bucaramanga

2022

Dedicatoria

Dedicado a:

Mi madre Balentina Rodríguez Angarita por su cariño y apoyo durante toda mi vida y carrera, igualmente a mi padre José Orlando Moncada Fuentes por ser parte del proyecto y su interés del tema. A mi hermana Diana por su comprensión y colaboración voluntaria con todo lo del día a día.

A mi director de trabajo de grado por ser no solo un instructor sino un amigo que guía y confía en mí.

Y por último agradecer a mis amigos que me acompañaron hasta este punto, que sin ellos hubiese sido algo aburrido y monótono, y sin olvidar por último a mi waifu que me apoyo en todo lo que necesitaba.

Orlando Alberto Moncada Rodríguez

A mi padre Joaquín, quien con su amor, charlas y esfuerzo me han permitido cumplir un sueño más.

Gracias por inculcar en mí el ejemplo de esfuerzo y creatividad.

También a mis hermanas Lady, Carolina y Angie por su cariño, consejos, palabras de aliento y apoyo incondicional.

Andrés Felipe Uribe García

Agradecimientos

A los profesores José Rubiano, Lola Bautista y Gustavo Garzón por su sabiduría, amabilidad y ser excelentes docentes demostrando la preocupación y anhelo por enseñar e inculcar valores que nos han llevado a lo que somos hoy en día.

A nuestros compañeros y amigos, Naruto, Chigüi, Duiti, Fredy, Batman, Sheny y Sasuke por su amistad y compañerismo a lo largo de la carrera, con quien pudimos contar en numerosos proyectos y alegrías que conlleva la vida universitaria.

A cada uno de nuestros familiares por la ayuda incondicional en todo lo que necesitábamos y la motivación a siempre seguir adelante.

Por último, un especial agradecimiento al balneario Las Maravillas que nos permitió hacer las pruebas en el rio Manco con una buena hospitalidad.

Contenido

Pág.

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Glosario

AMI: Imagen de una máquina virtual proporcionada por Amazon para ofrecer sus servicios de AWS.

Angular: Framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página.

AP: Access Point.

ApexChartJs: Librería que provee gráficas en formato SVG

APIs: Interfaz de programación de aplicaciones.

ARS: Altimetría radar por satélite

AWS: Amazon Web Services

AZURE: Servicio de computación en la nube creado por Microsoft para construir, probar, desplegar y administrar aplicaciones y servicios mediante el uso de sus centros de datos.

Backend: Es la parte de un software con la cual no puede interactuar de manera directa el usuario final, su función es acceder a la información solicitada a través de la app, para luego entregarla al usuario final.

Bash: Intérprete de comandos que actúa como interfaz entre el kernel Linux y los usuarios o programas.

Broker: Programa intermediario que sirve de puente de comunicación entre los dispositivos electrónicos y el backend de la aplicación.

Cloud Computing: Es un servicio de hardware o software proporcionado a través de internet por un tercero.

CodeDeploy: Servicio de implementación administrado que automatiza las implementaciones de software en diferentes servicios informáticos.

COM: Interfaz de E/S que permite conectar dispositivos serie a un ordenador.

CRUD: Acrónimo de Create, Read, Update, Delete, las 4 operaciones básicas al operar información en bases de datos.

DAC: Convertidor Analógico Digital.

Daemon: Tipo especial de programa que se ejecuta en segundo plano, en vez de ser controlado directamente por el usuario.

DBMS: Sistemas de software encargados de administrar, crear, recuperar y actualizar datos de forma esquematizada.

DRY: Don't repeat yourself, filosofía de definición de procesos que promueve la reducción de la duplicación especialmente en computación.

EC2: Amazon Elastic Compute Cloud, servicio de computación en la nube de tamaño modificable.

Edge Computing: Es un modelo de negocio en el cual se reparten los recursos informáticos a un centro de datos cerca a los dispositivos físicos que suministran los datos.

EMQX: Broker MQTT diseñado para el uso en IoT.

ENSIVAT: Satélite de observación terrestre construido por la Agencia Espacial Europea

ER: Entidad-Relación.

EV: Estación Virtual.

Express: Framework web transigente, escrito en JavaScript y alojado dentro del entorno de ejecución NodeJS.

Firebase: Plataforma en la nube para el desarrollo de aplicaciones web y móviles perteneciente a Google.

Framework: Esquema de trabajo que ofrece una estructura predefinida para desarrollar un proyecto con objetivos específicos, que sirve como punto de partida para el desarrollo de software.

Frontend: Es la parte de un software que se dedica al diseño visual de un sitio web, desde la estructura del sitio hasta los estilos, tamaños y formas presentes para la interacción con el usuario.

GPIO: *General Purpose Input/Output*, Pin de Entrada/Salida de Propósito General

Heroku: Plataforma de servicios en la nube que permite manejar los servidores y sus configuraciones, escalamiento y la administración.

HTTP: Protocolo de transferencia de hipertexto.

IAM: Identity and Access Management.

IDE: Entorno de desarrollo integrado.

IoT: Internet de las cosas, es un sistema de objetos físicos que llevan incorporados sensores, software y demás tecnologías que le permiten conectarse e intercambiar datos con otros dispositivos y sistemas a través de Internet.

JSON: Javascript Object Notation.

JWT: Json Web Token, es un estándar abierto basado en JSON para la creación de tokens que sirvan para enviar datos entre aplicaciones o servicios garantizando su validez y seguridad.

Mockup: Representación o bosquejo del prototipo que se va a realizar.

MongoDB: Sistema de gestión de bases de datos orientada a documentos.

Mongoose: Biblioteca de JavaScript que permite definir esquemas con datos fuertemente tipados.

MQTT: Message Queuing Telemetry Transport, protocolo de conectividad M2M orientado a IoT.

Node.Js: Entorno de código abierto, para la capa del servidor basado en el lenguaje JavaScript, de ejecución asíncrona, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

NoSQL: Sistemas de gestión de bases de datos que no solo operan bajo el paradigma estructurado proporcionado por SQL.

PLL: Phase-Locked Loop, es un sistema de control basado en la generación de una segunda señal relacionada a la fase de una primera señal de entrada

POO: Programación orientada a objetos.

PUTTY: Emulador de terminal gratuito que admite varios protocolos de red tal como SSH.

PWM: Pulse-width modulation, técnica de transmisión de información por medio de la modulación de una señal periódica.

Responsive Design: Paradigma de diseño web que busca la correcta visualización de un mismo sitio en distintos dispositivo.

REST API: Conjunto de funciones de solicitud y recepción de respuestas a través del protocolo HTTP.

RH: Humedad relativa.

RXJS: Librería de programación reactiva de JavaScript

SASS: Syntactically Awesome Stylesheets, metalenguaje de hojas de estilo en cascada y antecesor de CSS.

SCSS: Herramienta de preprocesamiento de CSS que permite generar hojas de estilo, añadiendo características propias de un lenguaje de programación de las cuales carece CSS

SPA: Single Page Application.

SQL: Lenguaje de computación especializado en conjuntos de datos y las relaciones entre ellos.

SSH: Secure SHell, protocolo de comunicación remota entre servidores.

SSL: Secure Sockets Layer, es un protocolo de seguridad para navegadores web y servidores que permite la autenticación, encriptación y desencriptación de datos transmitidos a través de internet.

STA: Station.

TCP: Protocolo de Control de Transmisión.

WebSocket: Protocolo de red basado en TCP que establece cómo deben intercambiarse datos entre redes.

Resumen

Título: Diseño de una plataforma *IoT* para el monitoreo del nivel, velocidad y estado meteorológico en un cauce que permita dar alerta temprana de desastres en puntos estratégicos.

Autores: Orlando Alberto Moncada Rodriguez, Andrés Felipe Uribe García.

Palabras clave: Desarrollo web, IoT, Control, Aplicación.

Objetivo general: Diseñar e implementar una plataforma *IoT* para el monitoreo del nivel, velocidad y estado meteorológico en un cauce que permita dar alerta temprana de desastres en puntos estratégicos.

Descripción: Los ríos son la principal y más importante fuente de agua potable para la humanidad, pero estas áreas hidrográficas por diversos factores ambientales, tales como la lluvia o los sedimentos, pueden generar un cambio abrupto en el nivel o velocidad del afluente lo cual lleva a producir catástrofes ambientales como inundaciones o deslizamientos de tierra, los cuales generan daños materiales e incluso pérdidas humanas.

Esta problemática dio iniciativa a este proyecto, el cual consta del diseño de una plataforma IoT conformada por un sistema de sensores que permite obtener en tiempo real las mediciones de cantidad de agua por precipitación, nivel del río y velocidad de este, las cuales se verán proyectadas a través de una aplicación web generando una alerta en caso de un cambio significativo en el área hidrográfica.

Abstract

Title: Design of an IoT platform for monitoring the level, speed and meteorological status in a channel that allows early warning of disasters at strategic points.

Authors: Orlando Alberto Moncada Rodriguez, Andrés Felipe Uribe García.

Keywords: Web Development, IoT, Control, Application.

General objective: Design and implement an IoT platform for monitoring the level, speed and meteorological status in a channel that allows early warning of disasters at strategic points.

Description: Rivers are the main and most important source of drinking water for humanity, but these hydrographic areas due to various environmental factors, such as rain or sediments, can generate an abrupt change in the level or velocity of the tributary which leads to produce Environmental catastrophes such as floods or landslides, which cause material damage and even human losses.

This problem gave the initiative to this project, which consists of the design of an IoT platform made up of a sensor system that allows measurements of the amount of water due to precipitation, river level and river speed to be obtained in real time, these measurements will be shown through a web application generating an alert in the event of a significant change in the hydrographic area.

Introducción

El proyecto de investigación “Diseño de una plataforma IOT para el monitoreo del nivel, velocidad y estado meteorológico en un cauce que permita dar alerta temprana de desastres en puntos estratégicos”, que tiene como propósito diseñar una plataforma IOT que permita el monitoreo meteorológico de un cauce de Colombia, con la intención de realizar una prevención temprana de desastres. El desarrollo y diseño de esta propuesta contiene un marco referencial y conceptual que habla sobre los conceptos de los materiales a utilizar tanto en el hardware como en el software, un estado del arte internacional en Paraguay y 2 nacionales, uno en la región de la Orinoquía y otro en el departamento de Antioquia, en donde se han encontrado proyectos hidrológicos y de plataformas similares que ayudan a prevenir desastres mayores.

Para el área de metodología del trabajo a realizar, este se divide en dos grandes partes, en la que comienza una primera parte dividida en 4 grandes fases y otra en el desarrollo del proyecto, estas son las cuatro fases mencionadas anteriormente: primero, la formación tecnológica y el levantamiento de requerimientos, en la que se habló sobre la investigación y análisis de los conceptos fundamentales, en la exploración de las tecnologías apropiadas para el desarrollo del prototipo, la indagación de proyectos similares aplicados en el campo; segundo, el análisis y el diseño de arquitectura en la que observó el estudio de la viabilidad y el alcance del proyecto, la definición y diseño de la plataforma de software, la selección de métricas del proyecto y la definición de la arquitectura; tercero, el desarrollo del prototipo y la plataforma, evidenciamos los diseños de la red y conectividad, de la interfaz del usuario y de la implementación del usuario, la construcción del prototipo a escala, la arquitectura tecnológica

del sistema, la plataforma web y la implementación del esquema de sensores y componentes del hardware; cuarta y última fase, la evaluación de requerimientos del prototipo, arroja la validación, al verificación y el análisis de los resultados de los datos junto con un informe de resultados y correcciones para las interacciones futuras.

Para el desarrollo del proyecto contamos con unas cuatro sub- fases que se van desarrollando de acuerdo a las 4 fases principales, las cuales son : primero, la capacitación de tecnologías; segundo, el análisis y la definición de la arquitectura en la cual se realizarán las comparaciones pertinentes de la arquitectura, los lenguajes para el *Back-end,* las bases de datos y los dispositivos y sensores necesarios para el hardware; tercero, el diseño del hardware y el software en el cual se encontrará, la definición de los requerimientos funcionales, no funcionales y roles, los casos de usos, los diagramas de procesos, el modelo de datos, el diseño del *mockup*, el diseño de la arquitectura y el control de versiones; cuarto, el diseño del hardware y el software donde estará la creación del servidor y la publicación, la base de datos y el cargue a la nube.

Al realizar la aplicación de la metodología propuesta en el proyecto, evidenciamos los resultados de la labor realizada en el río de oro de Girón, en el cual, este será nuestro lugar piloto para desarrollar las muestras de nuestra plataforma. En los resultados evidenciamos gráficas y datos que nos ayudarán a calibrar y mejorar la el nivel, la velocidad de acuerdo con las diferentes condiciones meteorológicas presentadas durante el proceso de exposición del prototipo a escala en este cauce. Para finalizar se encuentran las conclusiones que obtuvimos de todo este trabajo realizado y la invitación a más ingenieros de sistemas y meteorólogos a mejorar y realizar nuevas versiones de acuerdo con los diferentes territorios, entornos, climas y ecosistemas de las demás regiones de Colombia.

1. Objetivos

1.1 Objetivo General

Diseñar e implementar una plataforma IoT para el monitoreo del nivel, velocidad y estado meteorológico en un cauce que permita dar alerta temprana de desastres en puntos estratégicos.

1.2 Objetivos Específicos

Identificar las variables y requerimientos para el diseño de la plataforma.

Definir los componentes a usar para la infraestructura hardware y software requeridos en el desarrollo de la plataforma.

Diseñar la estructura del hardware y software según lo anteriormente planteado.

Evaluar el funcionamiento de la plataforma *IoT* desarrollada mediante la interacción de los distintos módulos de la plataforma con un prototipo funcional.

2. Marco referencial

2.1 Marco conceptual

2.1.1 Pluviómetro

Es un dispositivo que se emplea para medir las precipitaciones que caen en un lugar durante un tiempo determinado. Los datos obtenidos permiten crear un registro climatológico y cómo fluctúan las precipitaciones de dicha zona.

Figura 1. *Pluviómetro tipo balancín*
Fuente: HyQuest Solutions Latin America (2022).

2.1.2 Sensores

Un sensor es un dispositivo que detecta cambios en el entorno y genera una respuesta en base a esta alteración. Este proceso se puede interpretar como la conversión de un fenómeno

físico a un voltaje analógico medible o algunas veces en una señal digital, haciendo posible la lectura de datos y un procesamiento adicional después.

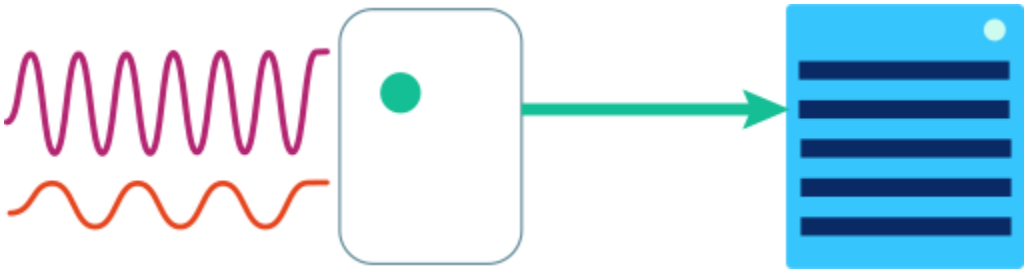


Figura 2. *Entrada y salida en un sensor*

Fuente: elaboración propia.

2.1.3 IoT

El término IoT hace referencia a la palabra anglosajona *Internet of Things*, cuya traducción oficial es Internet de las cosas. Este hace relación a una infraestructura de red inteligente en la cual distintos dispositivos como sensores o artículos cotidianos se interconectan a una red, lo cual permite el intercambio de información sin la necesidad de intervención humana.

Un sistema tradicional de IoT funciona enviando, recibiendo y analizando la información en un círculo constante de realimentación, donde también puede intervenir las personas, inteligencia artificial o el aprendizaje automático dando resolución a la funcionalidad de los equipos.

2.1.4 Cloud Computing

También conocida como computación en la nube, es un modelo de servicios de computación que están disponibles virtualmente bajo demanda, de una manera rápida, sin

intervención por parte del cliente y con un mínimo esfuerzo de gestión e interacción por parte del proveedor.

Dentro de las principales ventajas del Cloud Computing están la flexibilidad rápida brindando al usuario una experiencia dimensionada en base a la demanda desde cualquier ubicación, facilidad de colaboración permitiendo a los involucrados compartir y modificar información en tiempo real, y reducción de costes referidos a inversión en infraestructura, energía eléctrica, seguridad, personal, etc.

2.1.5 Edge Computing

Es un modelo en el cual se desplazan los recursos informáticos de los centros de datos y nubes a dispositivos de baja latencia cerca a la ubicación física de la fuente de datos, obteniendo servicios más confiables y rápidos con menores requerimientos de ancho de banda.

2.1.6 Fog Computing

Es una arquitectura informática descentralizada en la que los datos, comunicaciones, almacenamiento y aplicaciones se distribuyen entre la fuente de datos y la nube, en una estructura horizontal de recursos y servicios compartidos, permitiendo mejorar las prestaciones globales, una reducción en la distancia que recorren los datos en la red y una mayor eficacia.

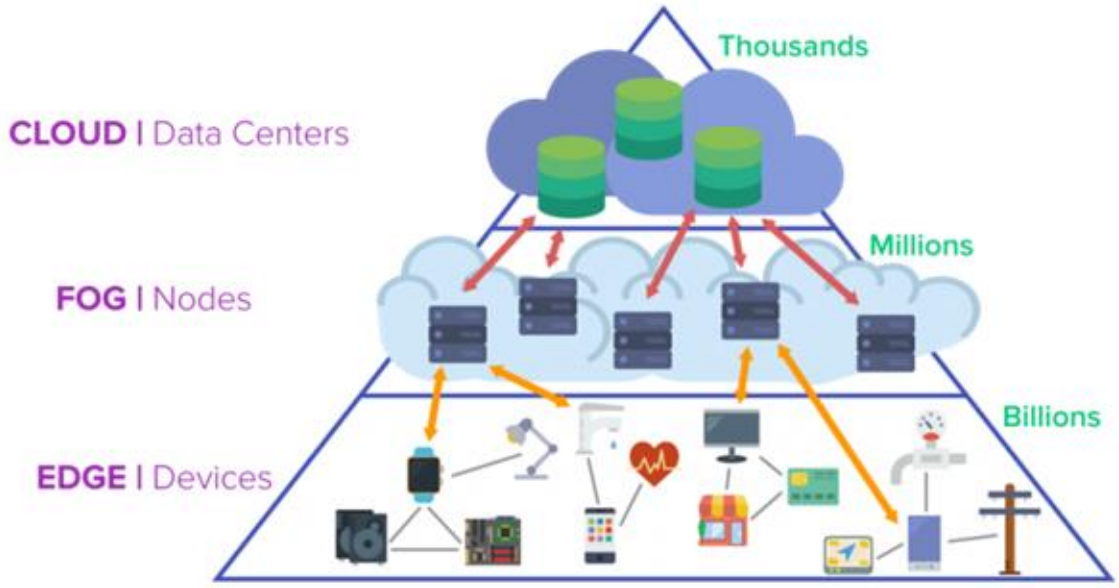


Figura 3. *Distribución de Cloud, Fog y Edge Computing*

Fuente: Fuente: Erpinnews (2018).

2.1.7 Aplicación Web

El concepto de aplicación web está relacionado con el almacenamiento en la nube ya que toda la información de esta se guarda en los servidores de internet popularmente conocidos como hostings los cuales permiten que podamos acceder desde cualquier lugar con solo tener conexión a internet. También es definido como un software que no requiere instalación en el equipo sino por lo general es mostrado a través de un buscador web.

2.1.8 Bases de Datos

Una base de datos es un conjunto autodescriptivo de registros integrados, que nos permiten representar las relaciones entre los datos, almacenados en un conjunto sin redundancias innecesarias cuya finalidad es la de servir a una o más aplicaciones de la manera más eficiente por medio de un sistema gestor de base de datos (DBMS).

La información almacenada permite la posibilidad de ser administrada, modificada, actualizada y eliminada con el objetivo de controlar de manera eficaz los datos.

2.1.9 EC2

Elastic Compute Cloud (EC2) es un servicio web que permite proveer de manera flexible y escalable los recursos computacionales, a los cuales el cliente de la nube tiene acceso y control total. Amazon EC2 permite la creación de servidores virtuales denominados instancias, los cuales tienen la capacidad de provisionar diferentes especificaciones computacionales y distribuciones de sistemas operativos. (Siegel & Gibbons, 2008)

Amazon EC2 proporciona a sus usuarios una amplia selección de instancias optimizadas para adaptarse a cada caso de uso abarcando diferentes combinaciones de CPU, memoria, almacenamiento y redes. Lo cual proporciona una gran flexibilidad en la elección de recursos en cada desarrollo. Además, cada tipo de instancia incluye uno o varios tamaños de instancia, permitiendo escalar recursos según los requisitos de carga de trabajo varían a lo largo del tiempo.

Entre los diferentes tipos de instancias se detallan 6 categorías principales:

- Uso general
- Optimizado para informática
- Optimizado para memoria
- Informática Acelerada
- Optimizado para almacenamiento

2.1.10 AMI

Amazon Web Services a través de su servicio Amazon EC2 ofrece la posibilidad de escoger una Amazon Machine Image (AMI), una plantilla que contiene una configuración de

software previamente establecida en relación al sistema operativo, servidor de aplicaciones y aplicaciones necesarias para lanzar una instancia. Las AMI tienen como objetivo agilizar el proceso de configuración y uso de los recursos computacionales dispuesto por Amazon.

Entre las opciones se encuentran desde aquellas diseñadas por el equipo técnico de Amazon y las diseñadas por la comunidad de usuarios de Amazon EC2, basadas en diferentes distribuciones de sistemas operativos pre-existentes como GNU/Linux, MacOs y Windows especialmente enfocadas a optimizar el uso de recursos en las diferentes arquitecturas disponibles por el servicio como lo son: 32 bits(x86), 64 bits(x86), 64 bits (Arm) y 64 bits (Mac).

2.2 Estado del arte

2.2.1 Red de monitoreo hidrológico virtual sobre la cuenca alta y media del río Orinoco

El proyecto llevado a cabo en el año 2012, fue propuesto como una solución ante la reducción del número de estaciones hidrométricas y, en consecuencia la falta de información para el monitoreo hidrológico de aguas superficiales, en la búsqueda de alternativas en el mejoramiento del monitoreo hidrológico proponen la aplicación de altimetría radar por satélite (ARS) en el monitoreo hidrológico de grandes cauces a través de la implementación de estaciones virtuales (EV), siendo una EV toda aquella intersección entre el barrido del satélite terrestre y un cuerpo de agua, a partir de la cual se deduce una serie que representa la variación de los niveles de agua en el tiempo. El proyecto toma datos provenientes de las misiones satelitales ENVISAT y Jason-2 sobre las cuencas alta y media del río Orinoco (cauces Meta,

Orinoco y Guaviare) para establecer una red virtual de monitoreo hidrológico que permite complementar en espacio y tiempo la red hidrométrica in situ actual.

Como resultado, la aplicación de la altimetría radar aportó mayor cobertura espacial, disponibilidad de la información en tiempo casi-real y complementación de datos in situ faltantes.

El tratamiento de los datos y el control en la variación de los datos de nivel de agua es relevante para el prototipo en pro de facilitar un mejor análisis del comportamiento hidrológico.

2.2.2 Sistema de monitoreo hidrológico en tiempo real para el Paraguay, centro internacional de hidroinformática. Itaipu-UNESCO, Paraguay

El objeto del proyecto Yatro fue el de centralizar, visualizar y proveer los datos mencionados de manera ordenada, teniendo como premisa el acceso libre, la interoperabilidad para el traspaso eficiente de datos y la difusión de la información capturada y generada para varios niveles y sectores de la sociedad con diversos fines.

Yatro captura datos hidrológicos en formatos varios, los depura, procesa y los convierte en gráficos y mapas complementados con líneas características de alertas de niveles, que permiten monitorear el estado y los riesgos asociados a las inundaciones.

El esquema de notificación del sistema es capaz de informar a instituciones claves sobre la evolución del comportamiento del río, de especial importancia para la toma de decisiones respecto a la planificación del riesgo asociado a las inundaciones, es este aspecto en los protocolos de transmisión de la información por el cual se toma como guía para la realización del prototipo a realizar.

2.2.3 Internet de las cosas en los ríos de Antioquia para prevenir desastres

En el año 2017 la Corporación Autónoma Regional de las Cuencas de los Ríos Negro y Nare (Cornare) en colaboración con Centro de Excelencia y Apropiación en Internet de las Cosas (CEA-IoT), Lagash y Microsoft dio a conocer un sistema de monitoreo diseñado con tecnología Azure IoT, para medir los niveles de agua de 60 afluentes ubicados en el Oriente de Antioquia, así como para calcular las cantidades de polución aérea en canteras al aire libre en esta región, no sólo en épocas de invierno sino también en tiempos de sequía.

La integración de los datos del proyecto con la nube y el manejo de la información es un aspecto que se asemeja a las bases tecnológicas que se quieren aplicar en el prototipo, por lo cual es considerado relevante para la implementación del prototipo a realizar.

3. Marco Metodológico

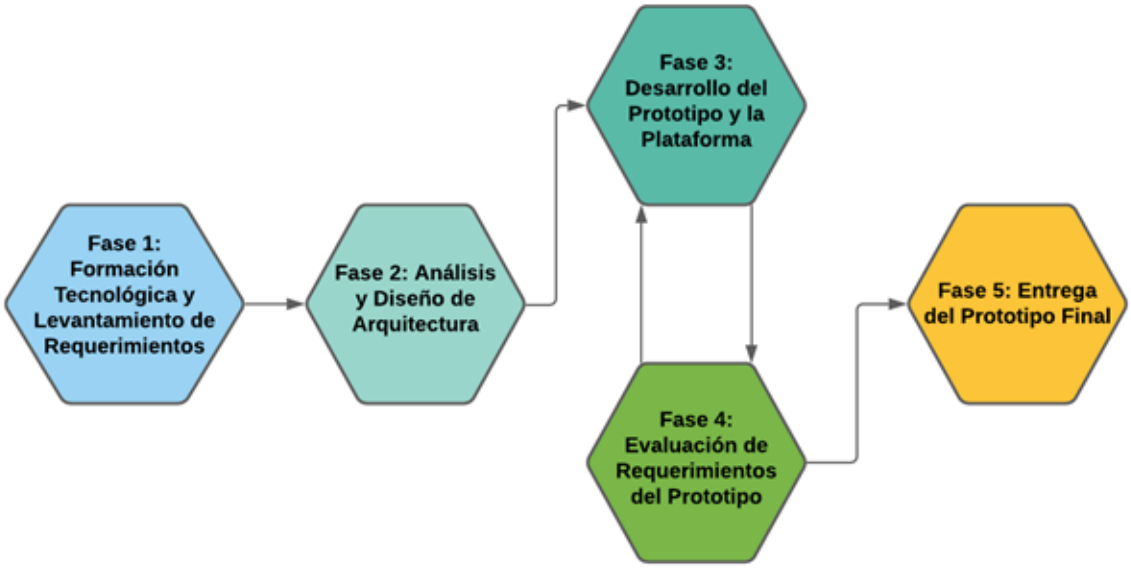


Figura 4. ***Esquema de metodología de trabajo***

Fuente: elaboración propia.

3.1 Fase 1: Formación Tecnológica y Levantamiento de Requerimientos

Se investigará acerca de las tecnologías fundamentales para la implementación de conceptos como IoT, Cloud computing, además del software y hardware asociados al proyecto y el manejo de herramientas tecnológicas que aporten a la realización del prototipo tales como Angular, Amazon Web Services, Node.js y Arduino. A su vez, se levantará la información de requerimientos que se desean cubrir.

Actividades

- A.1.1. Investigación y análisis de los conceptos fundamentales.

- A.1.2. Exploración de las tecnologías apropiadas para el desarrollo del prototipo.
- A.1.3. Indagación de proyectos similares aplicados en campo.
- A.1.4. Búsqueda de herramientas de relevantes en la realización del proyecto
- A.1.5 Definición de requerimientos del software y hardware.
- Resultados
- R.1.1. Dominio de conceptos fundamentales.
- R.1.2. Estado del arte.
- R.1.3. Tecnologías aplicables en la realización del proyecto.
- R.1.4. Requerimientos del prototipo.

3.2 Fase 2: Análisis y Diseño de Arquitectura

Se analizará la viabilidad de las diferentes opciones tecnológicas disponibles y se seleccionará la más adecuada para los objetivos y realización del proyecto con el fin de definir los componentes de hardware y software necesarios para el monitoreo, adquisición y estudio de datos requeridos por la plataforma.

Actividades

- A.2.1. Estudio de la viabilidad y alcance del proyecto.
- A.2.2. Definición y diseño de la plataforma de software
- A.2.3. Selección de métricas del proyecto
- A.2.4. Definición de la arquitectura (Dispositivos, Conectividad, Infraestructura, Aplicación)
- Resultados

- R.2.1. Requerimientos de la arquitectura de IoT

3.3 Fase 3: Desarrollo del Prototipo y la Plataforma

Una vez definidas las tecnologías y los diseños de la plataforma y el prototipo, se procederá al desarrollo en conjunto realizando simulaciones que nos permitan determinar la eficacia y eficiencia de los diferentes componentes tanto de hardware como de software y su comportamiento en bloque, comprobando regularmente su aporte al cumplimiento de los objetivos planteados, con lo cual se determinará si es necesario volver a la fase de análisis y diseño de la arquitectura.

Actividades

- A.3.1. Diseño y desarrollo de la interfaz de usuario.
- A.3.2. Implementación del esquema de sensores y componentes de hardware.
- A.3.3 Construcción del prototipo a escala.
- A.3.4. Diseño de la red de conectividad de los dispositivos y su conexión con la plataforma web.
- A.3.5. Diseño e implementación de la base de datos.
- Resultados
- R.3.1. Arquitectura tecnológica del sistema.
- R.3.2. Base de datos y plataforma web.
- R.3.3. Prototipo funcional y apto para validaciones.

3.4 Fase 4: Evaluación de requerimientos del prototipo

En esta etapa se hará la verificación del estado de los requerimientos realizando pruebas de ejecución y se analizarán los resultados con el fin de comprobar el cumplimiento de los objetivos anteriormente planteados, identificando posibles problemas e identificando sus mejoras para versiones futuras

Actividades

- A.4.1 Almacenamiento de datos en la nube e interacción con ellos.
- A.4.2. Verificación de la lógica e implementación.
- A.4.3. Validación del sistema mediante la realización de pruebas (funcionales, unitarias, entre otras).
- A.4.4. Análisis del resultado de las pruebas realizadas.
- Resultados
- R.4.1. Informe de resultados y correcciones para las iteraciones futuras

3.5 Fase 5: Entrega del Prototipo Final

Se realizará la documentación formal del proyecto y su posterior presentación ante la Escuela de Ingeniería de Sistemas e Informática de la Universidad Industrial de Santander culminando de esta manera el proyecto de grado.

Actividades

- A.5.1. Entrega del prototipo final con los ajustes realizados.
- Resultados

- R.5.1. Prototipo final

4. Desarrollo del proyecto

4.1 Capacitación de tecnologías

Con el objetivo de desarrollar con brevedad y eficacia el proyecto, se realizaron diversos cursos a través de plataformas pagas tales como Udemy y Platzi, como también gratuitas siendo ejemplo de estas Youtube. Cabe destacar de los cursos vistos uno con el nombre de IOT MASTERCLASS el cual, por su contenido, era muy afín con el proyecto planteado, ya que presentaba temas como Broker (MQTT), internet de las cosas, NodeJS y de gran importancia, Arduino.

Además de los cursos vistos, también se hizo una consulta exhaustiva acerca del Cloud Computing, lo cual dio pie para manejar de forma más fácil los servicios de AWS. Tanto documentación como videos tutoriales facilitados en YouTube, fue posible el rápido aprendizaje de esta tecnología y concepto. En cuanto a los lenguajes de programación referentes a la plataforma se tomaron los cursos del instructor Fernando Herrera los cuales facilitaron el desarrollo ágil de esta. (Ver Figuras 5, 6 y 7)

Figura 5. *Curso Angular de cero a experto*

Fuente: Udemy . (2021). *Angular: De cero a experto (Legacy)*. https://www.udemy.com/course/angular-2-fernando-herrera/

Figura 6. *Curso Node de cero a experto*

Fuente: Udemy. (2022). *Node: De cero a experto*. https://www.udemy.com/course/node-de-cero-a-experto/

Figura 7. *Curso IoT Bootcamp! Nuxt*

Fuente: Udemy. (2022). *IoT Bootcamp! Nuxt - Node - Mongo - Emqx - ¡Más de 60 horas!* . https://www.udemy.com/course/iot-god-level/

4.2 Análisis y definición de arquitectura

4.2.1 Comparación y servicios de arquitectura

Partiendo de la existencia múltiple de compañías prestadoras de servicios en la nube, se halló que las 5 principales plataformas en la actualidad son:

Amazon Web Services, Microsoft Azure, Google Cloud, Alibaba Cloud y Oracle Cloud.

Después de identificadas las mejores plataformas, se decidió estimar el costo de la adquisición de estos servicios. (Ver Tabla 1)

Tabla 1. *Amazon EC2 estímación*

Precios de Amazon Elastic Block Storage (EBS) (monthly)	3.00 USD
Instancias Amazon EC2 Instance Savings Plans (monthly)	30.75 USD
Costo total mensual:	**33.75 USD**

Fuente: elaboración propia.

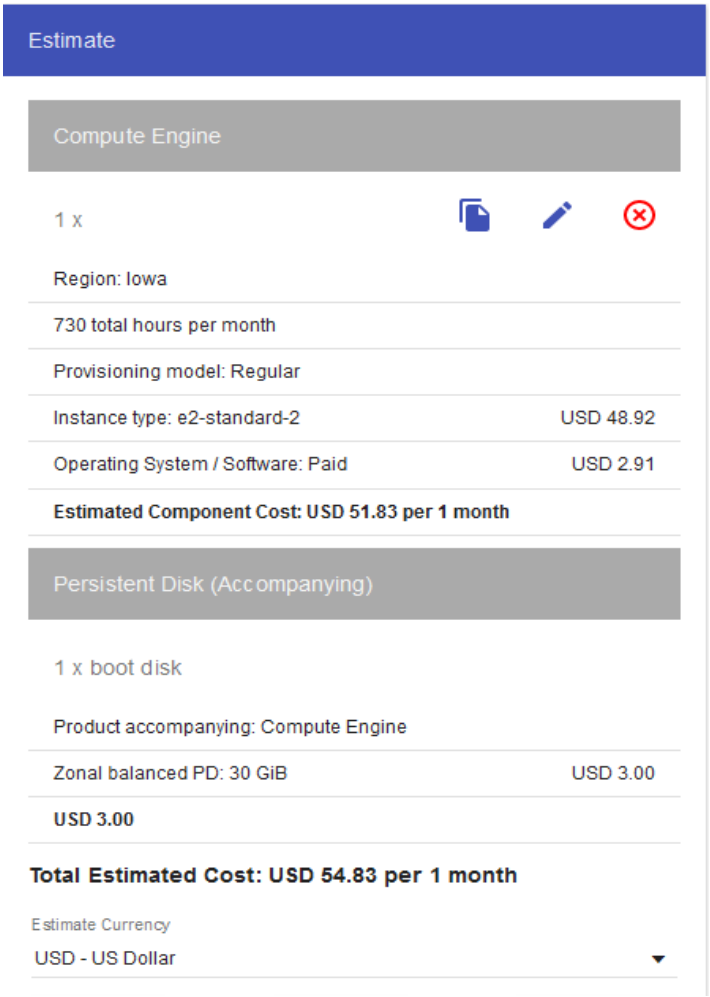
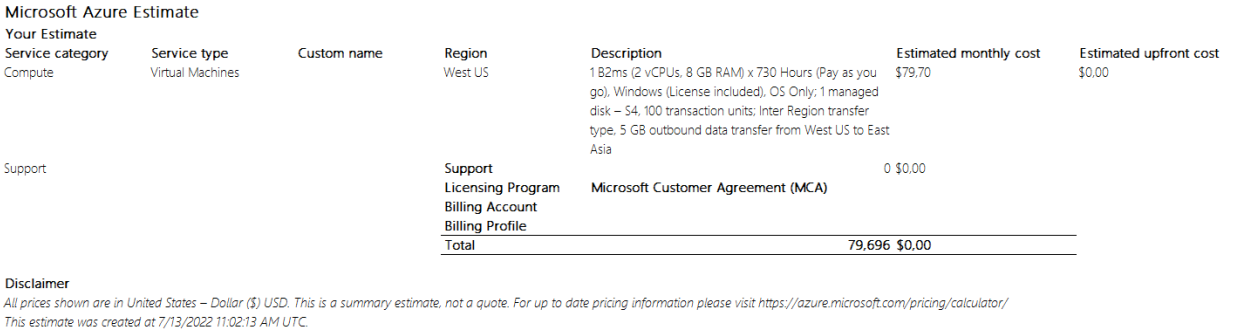


Figura 8. *Calculadora de EC2*

Fuente: Amazon Services. (2022). *Calculadora de EC2*. https://calculator.aws/#/addService



Microsoft Azure Estimate

Your Estimate

Service category	Service type	Custom name	Region	Description	Estimated monthly cost	Estimated upfront cost
Compute	Virtual Machines		West US	1 B2ms (2 vCPUs, 8 GB RAM) x 730 Hours (Pay as you go), Windows (License included), OS Only; 1 managed disk – S4, 100 transaction units; Inter Region transfer type, 5 GB outbound data transfer from West US to East Asia	$79,70	$0,00
Support			Support		0 $0,00	
			Licensing Program	Microsoft Customer Agreement (MCA)		
			Billing Account			
			Billing Profile			
			Total		79,696	$0,00

Disclaimer

All prices shown are in United States – Dollar ($) USD. This is a summary estimate, not a quote. For up to date pricing information please visit https://azure.microsoft.com/pricing/calculator/
This estimate was created at 7/13/2022 11:02:13 AM UTC.

Figura 9. *Calculadora de Google Cloud*

Fuente: Google. (2022). *Calculadora de Google Cloud*. https://azure.microsoft.com/en-us/pricing/calculator/

A modo de conclusión realizamos una tabla comparativa de los precios y características dadas por cada proveedor a la hora de crear un servidor virtual. (Ver Tabla 2)

Tabla 2. *Comparación de precios entre proveedores de instancias*

Característica	Amazon	Google Cloud	Azure
Almacenamiento	30 GiB	30 GiB	30 GiB
RAM	8 GB	8 GB	8 GB
CPU Cores	2	2	2
Uso	24 horas/dia	24 horas/dia	730 hrs/mes
Precio total	33,73 USD por mes	54,83 USD por mes	79,696 USD por mes

Fuente: elaboración propia.

Para efecto del prototipo, también se buscó su capa gratuita para cada proveedor, y se obtuvo lo siguiente. (Ver Tabla 3)

Tabla 3. *Comparación de precios en versión gratuita entre proveedores de instancias*

Característica	Amazon	Google Cloud	Azure
Almacenamiento	30 GB	30 GB	4 GB
RAM	1 GB	1 GB	1 GB
CPU Cores	1	2	1
Uso	750 hrs/mes	750 hrs/mes	730 hrs/mes
Popularidad	Muy alta	Alta	Alta

Fuente: elaboración propia.

Dado los datos anteriores y otras características encontradas sobre los proveedores, se tomó como decisión usar Amazon Web Services para la gestión cloud del back-end, ya que sus servicios adicionales en la capa gratuita presentaban una mejor comodidad y rendimiento en cuanto al proyecto se refiere, ya que, cuenta con herramientas IoT más avanzadas que los demás, y su integración con DevOps es más sencilla.

4.2.2 Comparación de lenguajes para el Back-end

El Back-end, donde la lógica del servidor es vital para el correcto funcionamiento, donde se integra la base de datos o el origen de la información procesada para el Front-end. Partiendo de lo anterior descrito, se necesita un proceso riguroso de selecciones de lenguaje, ya que dependiendo del nivel de rigurosidad, seguridad o escalabilidad es necesario optar por uno u otro lenguaje. Los principales exponentes en que está desarrollado la mayoría de Back-end son: JavaScript, Python, Ruby, PHP, Java, C#, Perl, C + +, Kotlin, Scala.

JavaScript se ha tornado el lenguaje más popular debido a su facilidad de aprender y la gran comunidad que aporta, ya sea con la creación de librerías o simplemente en la interacción en foros como StackOverflow. En el entorno de back-end crea la posibilidad de estructurar el proyecto de forma fácil haciendo uso de los Router y Middlewares.

En cuanto a Python, uno de los lenguajes de alto nivel más poderosos en cuanto a ciencia de datos, y en los últimos años el más usado para desarrollo de inteligencia artificial. Es un lenguaje que también cuenta con una gran comunidad y por ende numerosas librerías que permiten el ahorro de trabajo a la hora de crear un proyecto. Para Java y otros lenguajes más antiguos que los mencionados previamente, su popularidad ha decaído pero su nivel de organización, ya sea basado en un paradigma de programación orientada objetos o funcional,

tienen una estructura bien definida que se ha pulido a través de los años lo cual los hace una oferta tentativa a la hora de elegir un lenguaje base de programación para cualquier proyecto.

Por los motivos antes mencionados se ha decidido optar por JavaScript que aporta al proyecto el uso de múltiples librerías como express.js, la cual ayuda en el desarrollo ágil de APIs posteriormente consumidas por el Front-end. Además de la facilidad de solución de conflictos tanto en el compilado como en la lógica estructural de las funciones que realice el Back-end del proyecto. Para simplificar aún más el proyecto se decidió usar Node.js como entorno de desarrollo del Back-end aumentando la optimización y convirtiendo el código de máquina más rápido.

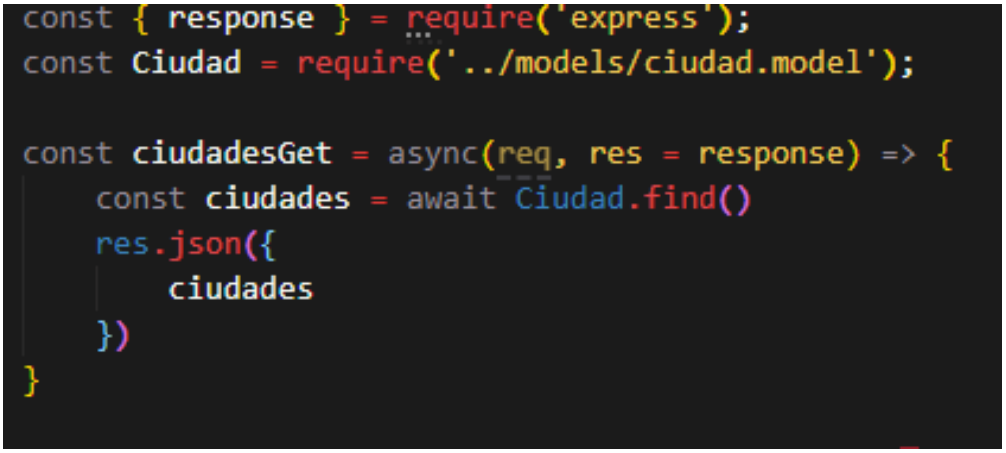


Figura 10. *Código para el Back-end*

Fuente: elaboración propia.

4.2.3 Comparación de bases de datos

Una apropiada gestión de la información implica tanto la definición de una estructura para el almacenamiento como también la disposición de un mecanismo para la adecuada manipulación de datos. Además, un sistema de bases de datos debe proporcionar fiabilidad en la información almacenada, a pesar de las posibles caídas del sistema o fallas en el de acceso y

modificación de la información, evitando al máximo posibles resultados anómalos. Existen diversas formas de almacenar información y esto da lugar a distintos modelos de organización de la base de datos, entre los modelos de bases de datos más conocidos se encuentran dos clasificaciones: aquellos basados en objetos como: El modelo Entidad-Relación (ER), Orientado a Objetos, Semánticos y Funcional, mientras que por el otro lado se encuentran aquellos basados en registros como: El Relacional, de Red y el Jerárquico. Los modelos relacionales son particularmente relevantes debido a su uso extendido en la industria del desarrollo y los beneficios que provee tales como: simplicidad, facilidad de uso, períodos cortos de aprendizaje y consultas de información fáciles de especificar.

Una base de datos relacional es aquella donde los datos están organizados estrictamente en tablas de valores estructurados, y en donde todas las operaciones operan sobre estas tablas. Estas bases de datos requieren de una colección de relaciones normalizadas normalmente hasta el nivel 5 con el fin de eliminar redundancias sistemáticas y depurar las relaciones entre las tablas.

Figura 11. *Ejemplo del esquema de tablas de una base de datos relacional*

Fuente: elaboración propia.

Para un correcto funcionamiento de una base de datos relacional se necesita de un sistema gestor de bases de datos, los cuales se encargan del almacenamiento, modificación y extracción de la información en la base de datos, por medio de las operaciones CRUD (Create, Read, Update, Delete) valiéndose de sentencias SQL. Algunos de los más conocidos a la fecha son:

- MySQL
- MariaDB
- MariaDB
- SQLite
- PostgreSQL
- Microsoft SQL Server
- Oracle.

Por otro lado, están las bases de datos no-relacionales, comúnmente llamadas NoSQL al no utilizar el lenguaje SQL para la realización de consultas, caracterizadas por la no utilización de esquemas rígidos y almacenamiento de tipo desestructurado, es decir, sus tablas no poseen una estructura fija, guardando toda la información en documentos de texto plano no relacionados entre sí, permitiendo una gran adaptación de este tipo de bases de datos a las necesidades específicas de los proyectos de una forma más sencilla que los modelos Entidad-Relación.

```
{
        "persona": {
                "nombre": "Alberto",
                "edad": "23"
        }
}
```

Figura 12. *Ejemplo de la estructura de un documento en JSON*

Fuente: elaboración propia.

Las ventajas de este tipo de bases de datos residen en su flexibilidad, escalabilidad, alto rendimiento, gran optimización hacia modelos de datos específicos y patrones de acceso, y su alta funcionalidad.

Entre las bases de datos NoSQL más conocidas podemos destacar:

- MongoDB
- Redis
- Cassandra
- CouchDB
- RavenDB
- ObjectDB
- Google BigTable
- Amazon DynamoDB

En concreto, MongoDB es la elección dentro de la categoría de bases de datos no relacionales que posteriormente vamos a implementar para la gestión de datos, debido a su estructura flexible e integración con las tecnologías Angular, Amazon Web Services, Node.js y Arduino resultado de relación tan cercana con el lenguaje JavaScript.

4.2.4 Comparación de dispositivos y sensores necesarios para el hardware

4.2.4.1 Placas de desarrollo

Para este proyecto en concreto se tuvieron en cuenta las placas de desarrollo NodeMCU-32S, NodeMCU 0.9, NodeMCU 1.0, Arduino Nano y Rapsberry Pi Pico.

- NodeMCU-32S, basada en el módulo ESP32 tiene un procesador Tensilica Xtensa LX6 de doble núcleo a 32bits operando entre los 160 a los 240 MHz, puede soportar temperaturas entre los -40°C y 125°C, compatible con los modos AP, STA y AP +STA, cuenta con

conexión de modo dual a Wi-Fi y Bluetooth de 2,4 GHz, 448KiB de memoria ROM y 520 KiB de memoria RAM, 23 pines digitales y 12 de analógicos, 2 de ellos de salida (DAC), un convertidor de USB a CP2102 que facilita su programación a través de puertos COM, alimentación de 3,3 y 5 V a un costo en el mercado de aproximadamente 9 a 10 dólares. (Ver Figura 13)

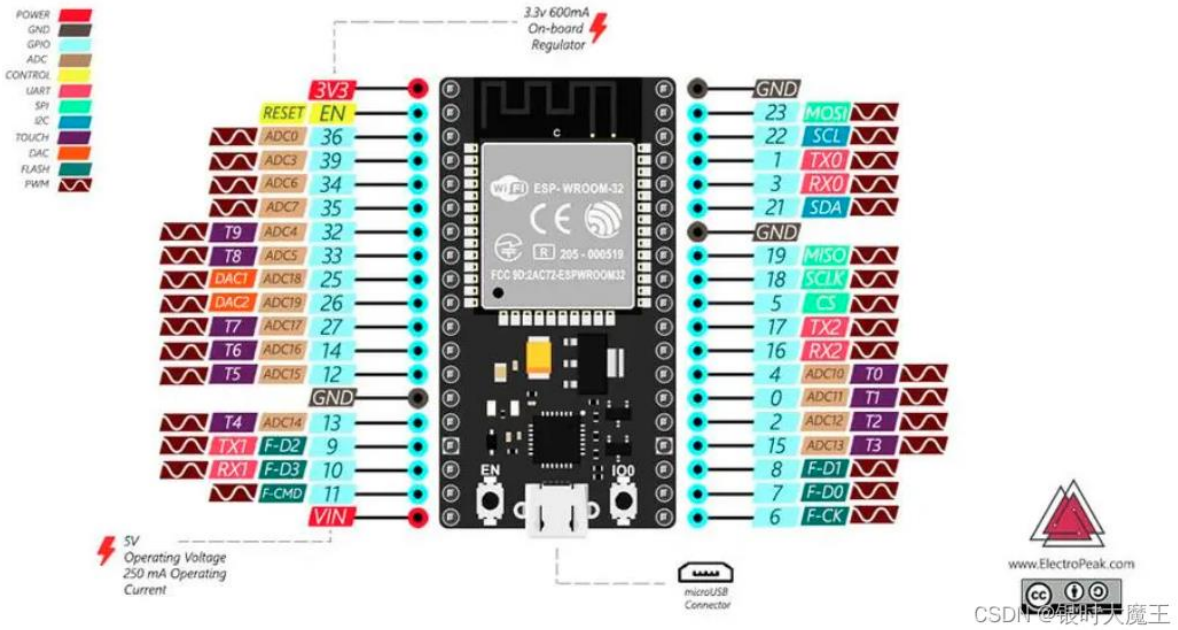


Figura 13. *Diagrama de pines del NodeMCU 32S*

Fuente: Arduino, S.R.L. (2020). *Arduino Sketcvh on NodeMCU 32S, ESP32*. https://forum.arduino.cc/t/arduino-sketcvh-on-nodemcu-32s-esp32/656784

- NodeMCU 0.9 y NodeMCU 1.0, ambos basados en el módulo ESP8266 en sus versiones 12 y 12 E tienen un procesador Tensilica Xtensa LX106 de doble núcleo a 32bits operando entre los 80 a los 160 MHz, pueden soportar temperaturas entre los -40°C y 125°C, cuentan con conexión de modo dual a Wi-Fi y Bluetooth de 2,4 GHz, 64 KiB de memoria ROM y 96 KiB de memoria RAM, 16 pines GPIO de propósito general de los cuales 6 se encuentran

reservados en la versión 0.9, además de un convertidor de USB a CP2102, alimentación de 3,3V a un costo de 4 y 5 dólares respectivamente. (Ver Figura 14)

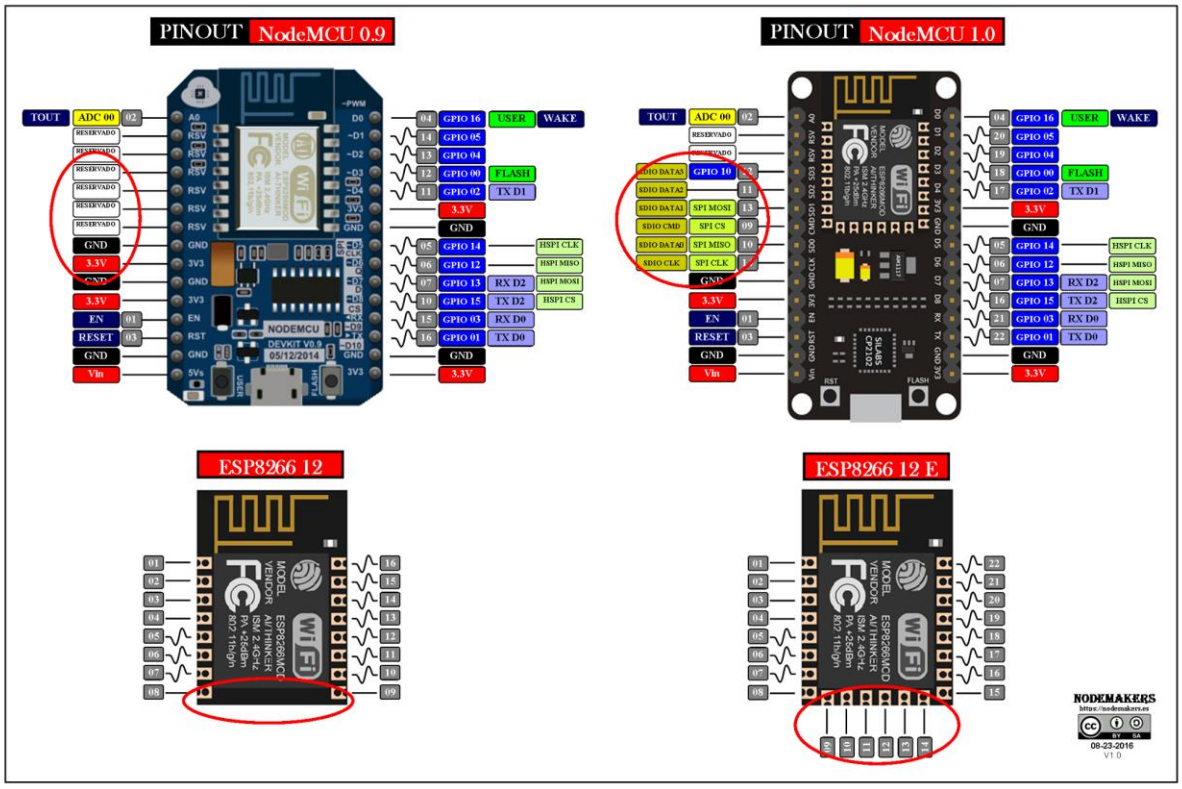


Figura 14. *Diagrama de comparación entre las placas NodeMCU 0.9 y NodeMCU 1.0*

Fuente: Esploradores. (2022). *Comparación de las placas NodeMCU* . https://www.esploradores.com/comparacion-de-placas-nodemcu_/

- Arduino Nano, basado en un procesador ATmega328P en operación a 16MHz con una frecuencia máxima de operación de 20MHz, voltaje de entrada de 1.8 a 5.5V sobre 3 timers posibles, cuenta con 28KiB de memoria flash y 2KiB de RAM, 14 pines de entrada y salida de los cuales 6 pueden ser usados analógicamente con PWM a un costo en el mercado de aproximadamente 7 a 8 dólares. (Ver Figura 15)

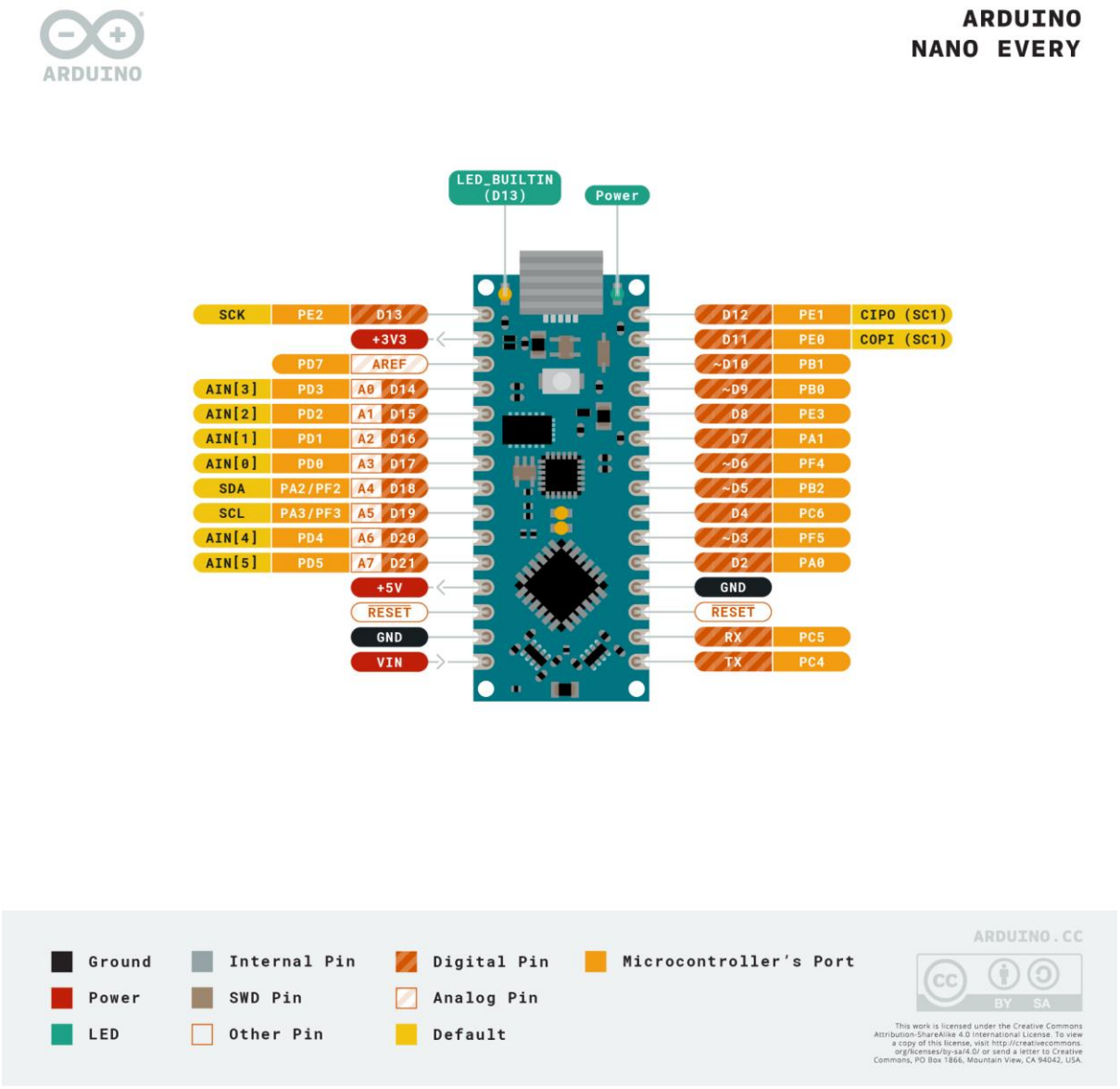


Figura 15. *Diagrama de pines del Arduino Nano*

Fuente: Arduino S.R.L. (2022). *Arduino Nano Every*. https://store-usa.arduino.cc/products/arduino-nano-every

- Raspberry Pi Pico, equipado con un procesador dual core ARM Cortex M0+ funcionando a 133 MHz, acompañado de 264 KiB de RAM y 2 MiB de almacenamiento integrado, cuenta con 26 puertos de entrada y salida (GPIO) 3 de ellos con capacidad ADC de 3.3V, puerto de alimentación tipo micro-USB B programable por canal flash, frecuencia del núcleo variable

a través del PLL del chip, Flash Quad-SPI externo con eXecute in Place (XIP) a un costo de 4 a 5 dólares. (Ver Figura 16)

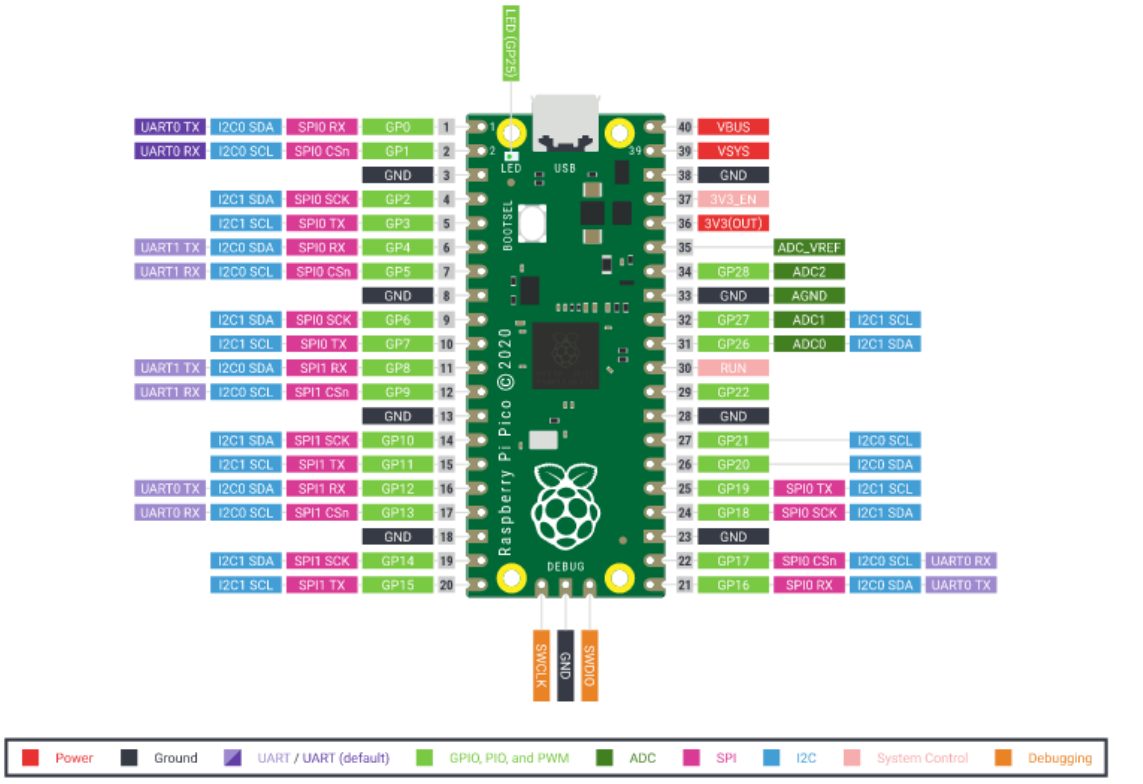


Figura 16. *Diagrama de pines del Raspberry Pi Pico*

Fuente: Raspberry Pi. (2022). *Raspberry Pi Pico – Raspberry Pi*. https://www.raspberrypi.com/products/raspberry-pi-pico/

Teniendo en consideración la gran cantidad de documentación al respecto y el sustento que da la gran cantidad de librerías suministradas por el framework de Arduino se optó por usar la placa NodeMCU 1.0, que además de cumplir con todos las funciones necesarias para llevar a cabo el proyecto a cabalidad permite replicar de manera sencilla el modelo de estaciones en placas de desarrollo como la ESP32s o la NodeMCU 0.9, con la única condición de compartir su módulo principal, la ESP8266 o su homónimo más reciente el ESP32.

4.2.4.2 Sensores

Como datos principales para el desarrollo de este proyecto se han escogido la temperatura, humedad, nivel del agua, velocidad del agua y precipitaciones, para este fin se han buscado

En cuanto a temperatura se han tenido en cuenta los sensores:

PT100 - RTD de Platino, cuenta con un rango de entre -20°C a 100°C, con una variación de resistencia de entre ± 0.005°C, funcional a 5V, aislado por fibra de vidrio y equipado con una sonda impermeable a un costo de 10 dólares. (Ver Figura 17)

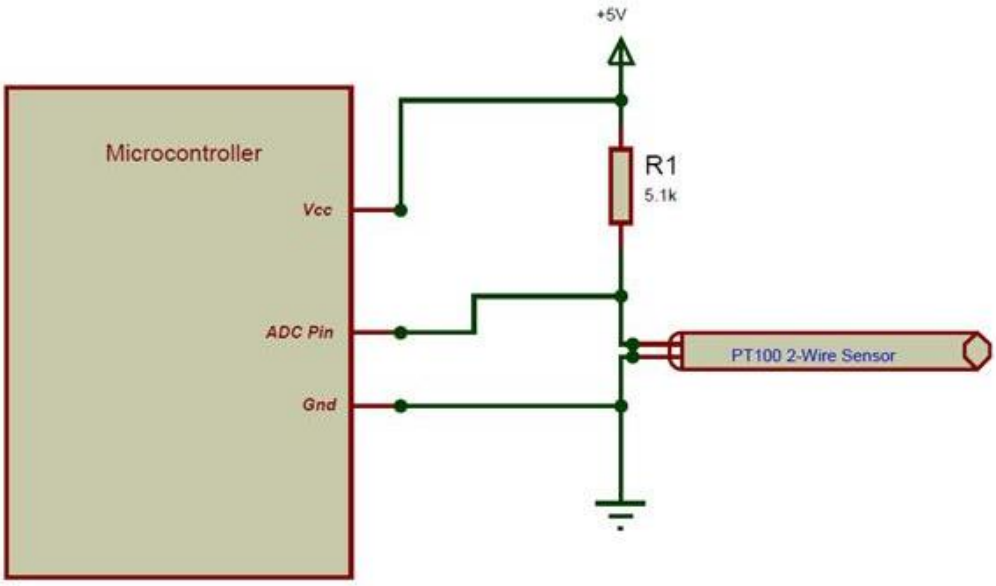


Figura 17. *Diagrama del sensor PT100 - RTD*

Fuente: DataSheet. (2022). *PDF Pt100 (Hoja de datos)*. http://www.datasheet.es/PDF/900325/Pt100-pdf.html

- DH11 y DHT22 los cuales presentan diferencias mínimas en cuanto a precisión, estabilidad y precio, los cuales cuentan con una salida de señal digital, resistencia de temperatura entre los

0°C y 50°C, alimentación de 3.5V a 5V, consumo de 2,5 mA y precisión de ± 2ºC cada 25°C a un costo de 1 dólar. (Ver Figura 18)

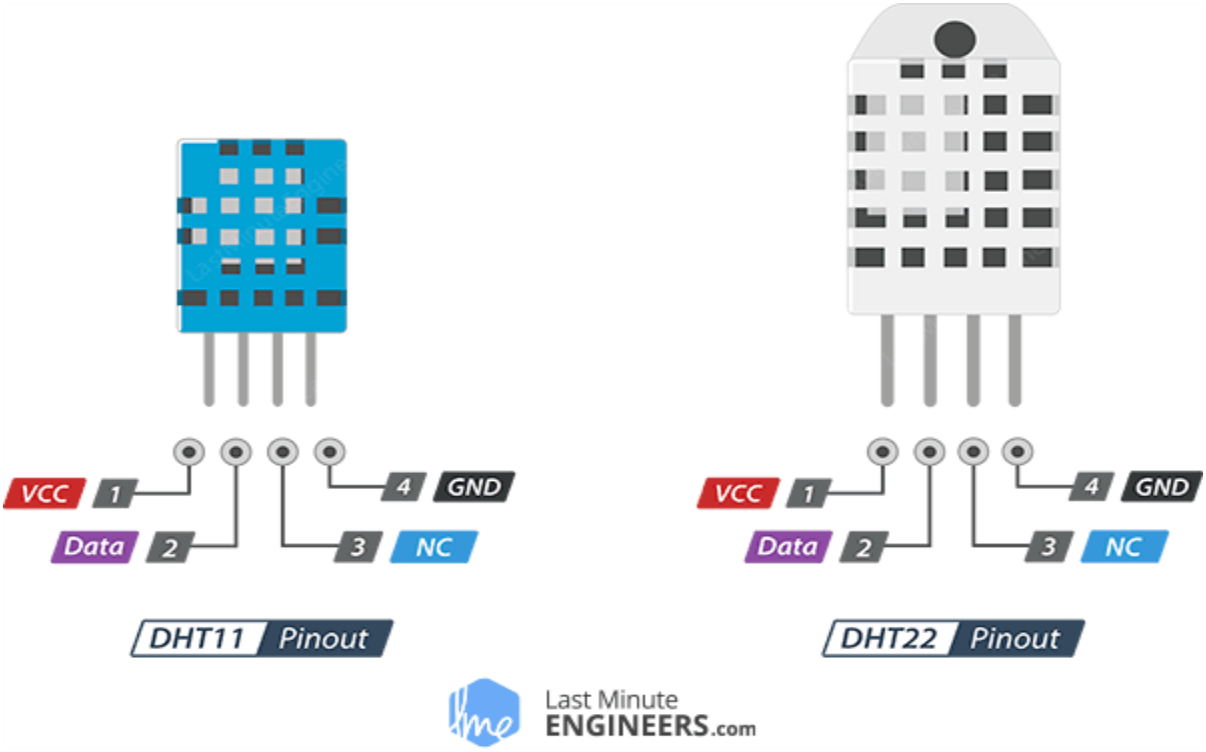


Figura 18. *Diagrama de pines de los módulos DHT-11 y DHT-22*

Fuente: Last Minute Engineers. (2022). *How DHT11 DHT22 Sensors Work & Interface With Arduino*. https://lastminuteengineers.com/dht11-dht22-arduino-tutorial/

En cuanto a humedad se han tenido en cuenta los sensores:

- DH11 el cual tiene un rango de entre 20% a 90% RH, alimentación de 3.5V a 5V, consumo de 2,5 mA y precisión de ±1% a un costo de 1 dólar.
- TRICAN HTD2800, cuenta con un rango de entre 0% a 100% RH, operando con una fuente de alimentación de 4.5 a 5V, consumo de 150mA y precisión de ±0.5% a un costo de 185 dólares. (Ver Figura 19)

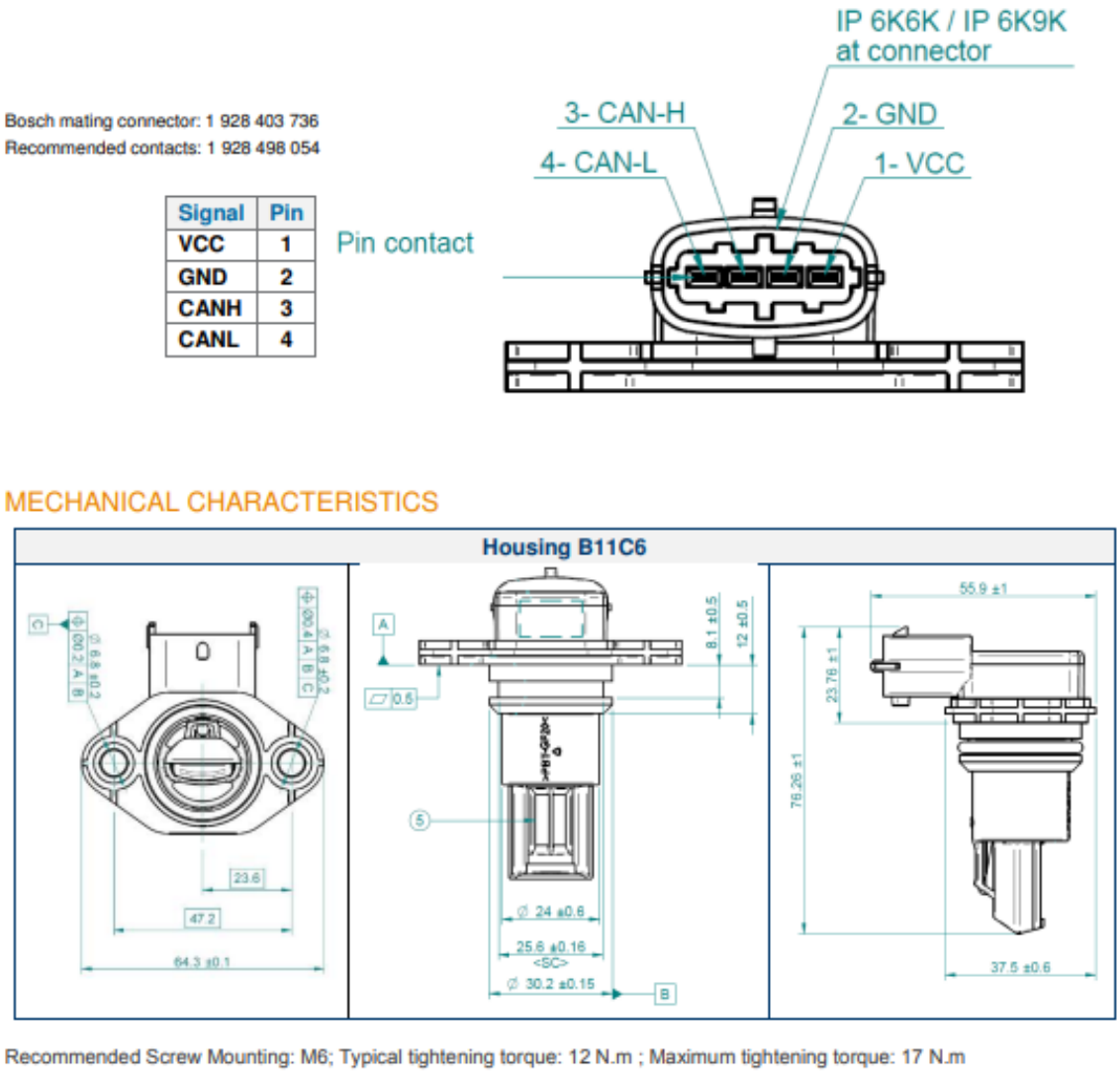


Figura 19. *Diagrama del sensor TRICAN HTD2800*

Fuente: Digi-Key Electronics. (2022). *Trican Engine Sensor HTD 2800*. https://www.digikey.com/catalog/en/partgroup/trican-engine-sensor-htd2800/87522

Para medir las precipitaciones se tuvieron en cuenta los sensores:

- HR0043, de señal análoga operante a 5V y 20 mA, en condiciones de humedad de entre 10% a 90% y de temperatura entre 10°C a 30°C a un costo de 1 dólar.

- WS-601SS2 de señal análoga operante a 12V, con un consumo de 230mA, funcional entre temperaturas de -15°C a 55°C y humedad de 5% a 98% a un costo de 45 dólares. (Ver Figura 20)

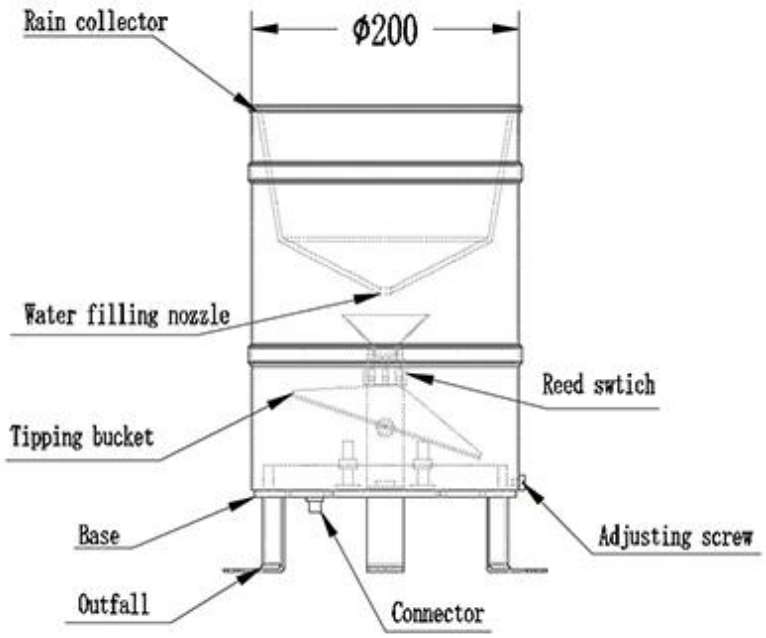


Figura 20. *Diagrama del sensor RK400-01*

Fuente: Rika Sensor. (2019). *Sensor RK400-01*. https://www.rikasensor.com/rk400-01-tipping-bucket-rainfall-sensor-accurate-rain-gauge.html

Por último, para medir el nivel del río se tuvieron los sensores de ultrasonido:

- HC-SR04, el cual cuenta con la capacidad de medir la distancia entre él y un objeto por medio del envío y recepción de una onda de ultrasonido de hasta 40Hz, cuenta con una señal análoga, operante a 5V y corriente de hasta 15mA, con un rango de medición de entre 2cm a 450cm con una precisión de 3mm, a un costo de 9 dólares.

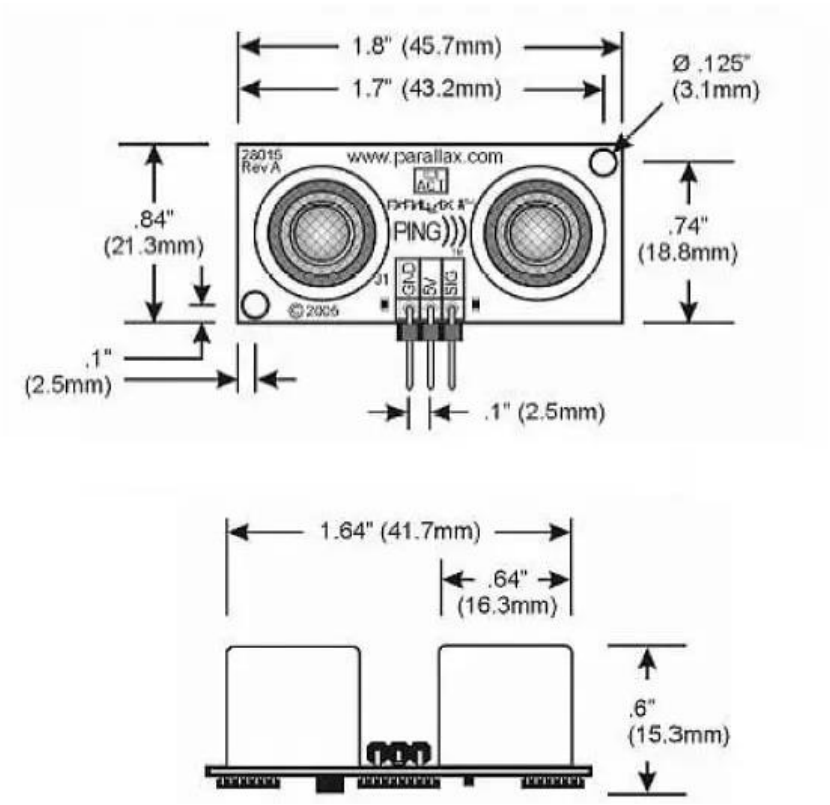


Figura 21. *Diagrama del sensor HC-SR04*

Fuente: All Data Sheet. (2022). *HC-SR04 Datasheet(PDF) - List of Unclassifed Manufacturers*. https://www.alldatasheet.com/datasheet-pdf/pdf/1132203/ETC2/HC-SR04.html

DL-MBX, con una frecuencia de ultrasonido de señal análoga, operante a 12V de forma independiente con baterías alcalinas tipo C por hasta 6 años de autonomía, con un rango de medición de hasta 10m con una precisión de 1mm a un costo de 50 dólares.

Debido a motivos de presupuesto y practicidad se ha optado para el desarrollo del prototipo el uso del sensor DHT11 de humedad y temperatura, el HR0043 de precipitaciones y el HC-SR04 de ultrasonido para el nivel del rió los cuales suministran los valores indicados con una precisión aceptable.

4.3 Diseño del Hardware y Software

4.3.1 Definición de requerimientos funcionales, no funcionales y roles

4.3.1.1 Roles

Se definieron unos roles en base a las múltiples opciones que presenta la plataforma para determinar la necesidad de los permisos que cada funcionalidad requiera. (Ver Tabla 4)

Tabla 4. *Definición de roles*

Rol	Descripción
Administrador	Tiene acceso a todos los módulos del sistema
Técnico	Puede ver la ubicación de las estaciones, tanto en tiempo real como a manera de reporte, además puede añadir estaciones y crear reportes de mantenimiento.

Fuente: elaboración propia.

4.3.1.2 Requerimientos funcionales

Con el fin de satisfacer los objetivos específicos del plan, se describieron los requerimientos funcionales mínimos que la plataforma deberá tener para un correcto funcionamiento. (Ver Tabla 5)

Tabla 5. *Requerimientos funcionales*

No	Descripción	Roles
RF-1	El sistema debe ser capaz de identificar, autentificar y autorizar el acceso según el rol asignado.	Administrador, Técnico
RF-2	Se debe observar el número de estaciones con sus respectivos datos.	Administrador, Técnico
RF-3	Se debe poder actualizar datos de cada estación	Administrador, Técnico

RF-4	Se debe poder eliminar una estación de los registros del sistema	Administrador, Técnico
RF-5	Se debe poder crear un usuario con sus distintos datos	Administrador
RF-6	Se debe poder actualizar los distintos datos de cada usuario	Administrador
RF-7	Se debe poder eliminar un usuario	Administrador
RF-8	Se debe poder ver geográficamente las estaciones, con los datos ambientales de cada uno en tiempo real.	Cualquier usuario
RF-9	La aplicación debe notificar en tiempo real una alerta emitida por la posibilidad de desborde	Ningún usuario
RF-10	La estación debe generar una alerta vía SMS en caso de un posible desborde	Ningún usuario
RF-11	Se debe poder agregar un registro de mantenimiento en caso de daño de la estación	Administrador, Técnico
RF-12	Se debe poder observar los registros de mantenimientos creados.	Administrador, Técnico

Fuente: elaboración propia.

4.3.1.3 Requerimientos no funcionales

Con el fin de satisfacer un óptimo funcionamiento se describieron los requerimientos funcionales mínimos que la plataforma deberá tener. (Ver Tabla 6)

Tabla 6. *Requerimientos no funcionales*

No	Nombre	Descripción
RF-1	Escalabilidad	● Diseñar de forma modular los componentes para su posterior reutilización ● Permitir la fácil integración con librerías o servicios.
RF-2	Usabilidad	● Interfaz limpia y de fácil comprensión para el usuario usando iconografía. ● La aplicación debe contar con un diseño responsivo. ● La aplicación debe tener un sistema de notificación de errores
RF-3	Soporte	● La aplicación será soportada en varios navegadores (Chrome, Mozilla y edge)
RF-4	Seguridad	● La aplicación contará con certificado SSL. ● Las contraseñas se encontrarán encriptadas. ● La aplicación contará con un servicio de tokens para la autorización y autentificación de usuarios.
RF-5	Conectividad	● Se requiere conexión a internet de forma inalámbrica. ● Se requiere una cuenta de un proveedor de servicios en la nube

Fuente: elaboración propia.

4.3.2 Plan de pruebas

Tabla 7. *Ingreso del usuario*

#	Cliente	Software	Datos	Revisar
1		Cargar la página de bienvenida con los botones de “Login” y “ver mapa”		

2	Puede visualizar la información de bienvenida Clickear en el botón de "Login" Ingresar correo electrónico Ingresar contraseña		Correo@falso.com 12345678	- Que se cargue adecuadamente el formulario Que no deje ingresar un correo no registrado en la base de datos Que la contraseña coincida con la registrada
3	Entrar			Validar que los datos de usuario disponibles sean los correspondientes
4		Cargar la página para escoger uno de los 5 menús disponibles		
5				**Observaciones:** Estos 5 menús están únicamente disponibles para los usuarios
	Errores – Descripción		**Corregido**	**Fecha de Corrección**

Fuente: elaboración propia.

La segunda prueba por realizar será que cualquier usuario (técnico o administrador) pueda visualizar adecuadamente el menú estaciones. (Ver Tabla 8)

Tabla 8. *Prueba cargar menú de estaciones*

#	Cliente	Software	Datos	Revisar
1		Cargar el menú Estaciones		
2				Visualizar adecuadamente la información correspondiente a las diferentes estaciones

3		Opción de agregación Opción de edición Opción de desactivación		
4	Seleccionar la opción de agregación Seleccionar la opción de edición Seleccionar la opción de desactivación	Mostrar formulario de agregación de una estación Mostrar formulario de edición de la estación -		Se carga adecuadamente la nueva estación Se cargan adecuadamente los datos editados -
5				**Observaciones:** las opciones de agregación y desactivación están disponibles para todos los roles
	Errores – Descripción		**Corregido**	**Fecha de Corrección**

Fuente: elaboración propia.

La tercera prueba por realizar será con un usuario que cuente con el rol de tipo administrador quien puede visualizar y editar los usuarios existentes en el menú usuarios. (Ver Tabla 9)

Tabla 9. *Prueba cargar menú de usuario para visualizar y editar*

#	Cliente	Software	Datos	Revisar
1		Cargar el menú Usuario		
2				Visualizar adecuadamente la información correspondiente a los diferentes usuarios
3		Grilla de usuarios Opción de agregación Opciones de edición		

		Opciones de eliminación		
4	Seleccionar la opción de agregación Seleccionar las opciones de edición Seleccionar las opciones de eliminación	Mostrar formulario de agregación de un usuario Mostrar formulario de edición del usuario Eliminación del usuario		Se carga adecuadamente la nueva estación Se cargan adecuadamente los datos editados Se elimina adecuadamente el usuario seleccionado
5				**Observaciones:** las opciones de agregación, edición y eliminación de usuarios está disponible solo para el rol administrador
#	**Errores – Descripción**		**Corregido**	**Fecha de Corrección**

Fuente: elaboración propia.

La cuarta prueba por realizar será con un usuario que cuente con el rol de tipo técnico quien puede únicamente visualizar los usuarios existentes en el menú usuarios. (Ver Tabla 10)

Tabla 10. *Prueba cargar el menú de usuarios existentes*

#	Cliente	Software	Datos	Revisar
1		Cargar el menú Usuario		
2				Visualizar adecuadamente la información correspondiente a los diferentes usuarios

3		Grilla de usuarios		
4				**Observaciones:** las opciones de agregación, edición y eliminación de usuarios no deben estar disponibles para el rol técnico
#	**Errores – Descripción**		**Corregido**	**Fecha de Corrección**

Fuente: elaboración propia.

La quinta prueba por realizar será que cualquier usuario (técnico o administrador) pueda visualizar adecuadamente el menú reportes. (Ver Tabla 11)

Tabla 11. *Prueba cargar el menú de reportes*

#	Cliente	Software	Datos	Revisar
1		Cargar el menú Reportes		
2	Seleccionar una o más de las estaciones disponibles en el select			Que se seleccionen adecuadamente las estaciones disponibles
3		Mostrar adecuadamente las diferentes gráficas disponibles		
4	Interactuar y desactivar de forma individual en cada una de las gráficas los diferentes dispositivos			La correcta visualización de los datos suministrados por cada dispositivo
5				**Observaciones:** la información deberá mantenerse en constante actualización por

				parte del dispositivo
#	**Errores – Descripción**		**Corregido**	**Fecha de Corrección**

Fuente: elaboración propia.

La sexta prueba por realizar será que cualquier usuario (técnico o administrador) pueda visualizar adecuadamente el menú de mantenimientos. (Ver Tabla 12)

Tabla 12. *Prueba cargar el menú de mantenimientos*

#	Cliente	Software	Datos	Revisar
1		Cargar el menú Mantenimientos		
2				Visualizar adecuadamente los registros correspondientes los mantenimientos de las estaciones en orden cronológico
3		Opción de agregación Opción de aprobación Opción de eliminación		
4	Seleccionar la opción de agregación Seleccionar la opción de aprobación Seleccionar la opción de eliminación	Mostrar formulario de agregación de un mantenimiento Cambia el estado del mantenimiento a aprobado Elimina el mantenimiento de la grilla		Se carga adecuadamente la nueva estación Se cargan adecuadamente el nuevo estado Se elimina el registro del mantenimiento
5				**Observaciones:** las opciones de agregación,aprobación y

				eliminación están disponibles para todos los roles
#	**Errores – Descripción**		**Corregido**	**Fecha de Corrección**

Fuente: elaboración propia.

La séptima prueba por realizar será que cualquier usuario sin ingresar al módulo principal vea los cambios de los datos en las estaciones. (Ver Tabla 13)

Tabla 13. *Prueba cargar la página de mapa*

#	Cliente	Software	Datos	Revisar
1	Clickear en el botón "Ver mapa"			
2		Cargar la página de Mapa		Que las tarjetas correspondientes demuestren los estados de riesgo e indicadores según haya disminuido, aumentado o permanecido
3	Interactuar con las diferentes tarjetas accediendo al mapa correspondiente a cada ubicación	Alterar la tarjeta mostrando la posición correspondiente a la estación		Que la tarjeta cambie de forma dinámica mostrando la información indicada
4				**Observaciones:** Las diferentes tarjetas deben cambiar constantemente según reciban datos de sus respectivas estaciones

#	Errores – Descripción	Corregido	Fecha de Corrección

Fuente: elaboración propia.

4.3.3 Casos de uso

Con el fin de revisar las funcionalidades que tendría la plataforma para los distintos roles, se creó un diagrama de casos de uso donde se identifican las acciones realizadas por los mismos de una forma clara. (Ver Figura 22)

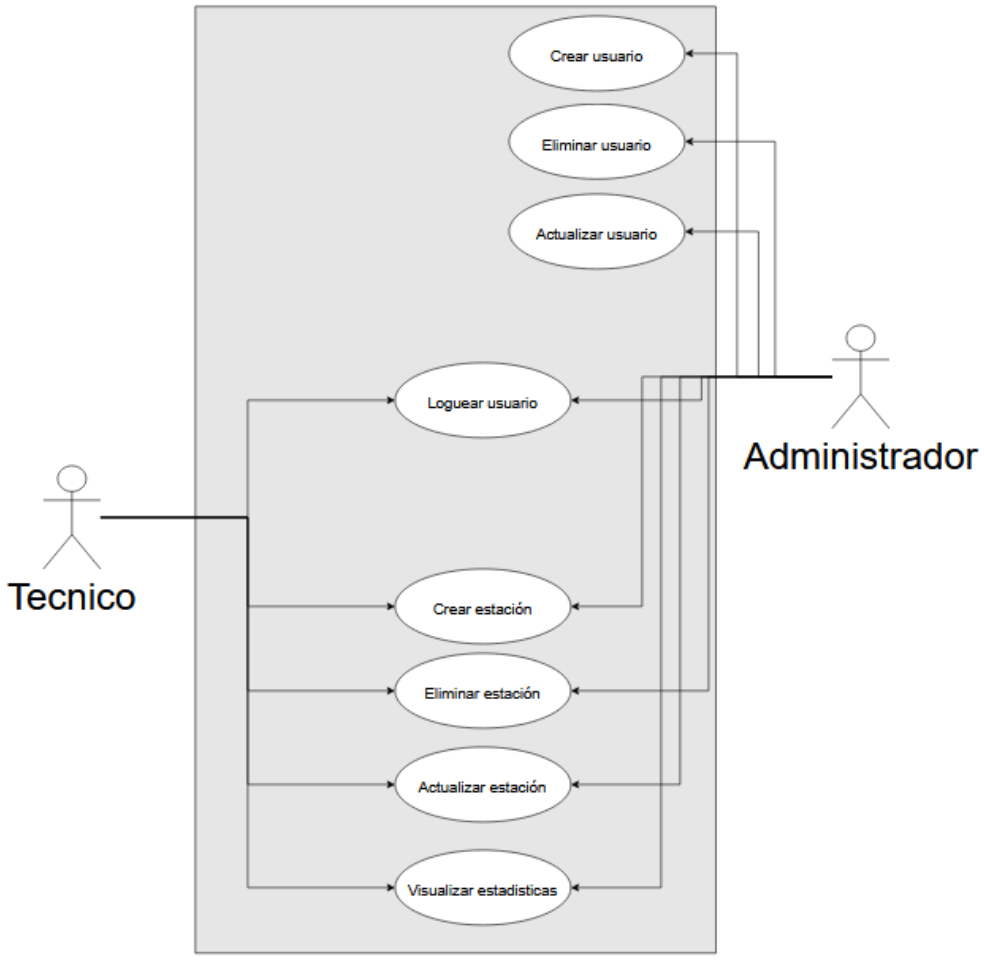


Figura 22. *Casos de usos*
Fuente: elaboración propia.

4.3.4 Diagramas de procesos

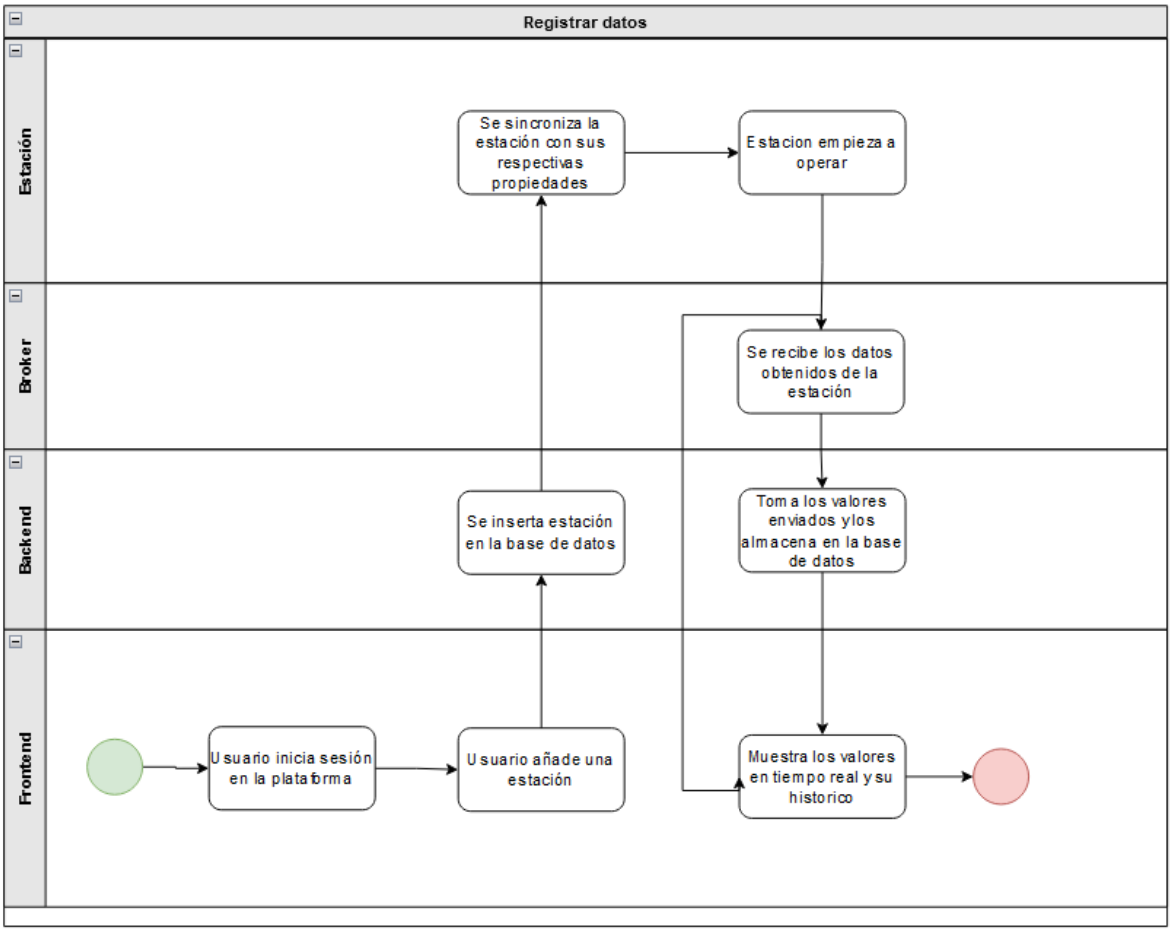


Figura 23. *Diagrama de proceso de registro de datos*

Fuente: elaboración propia.

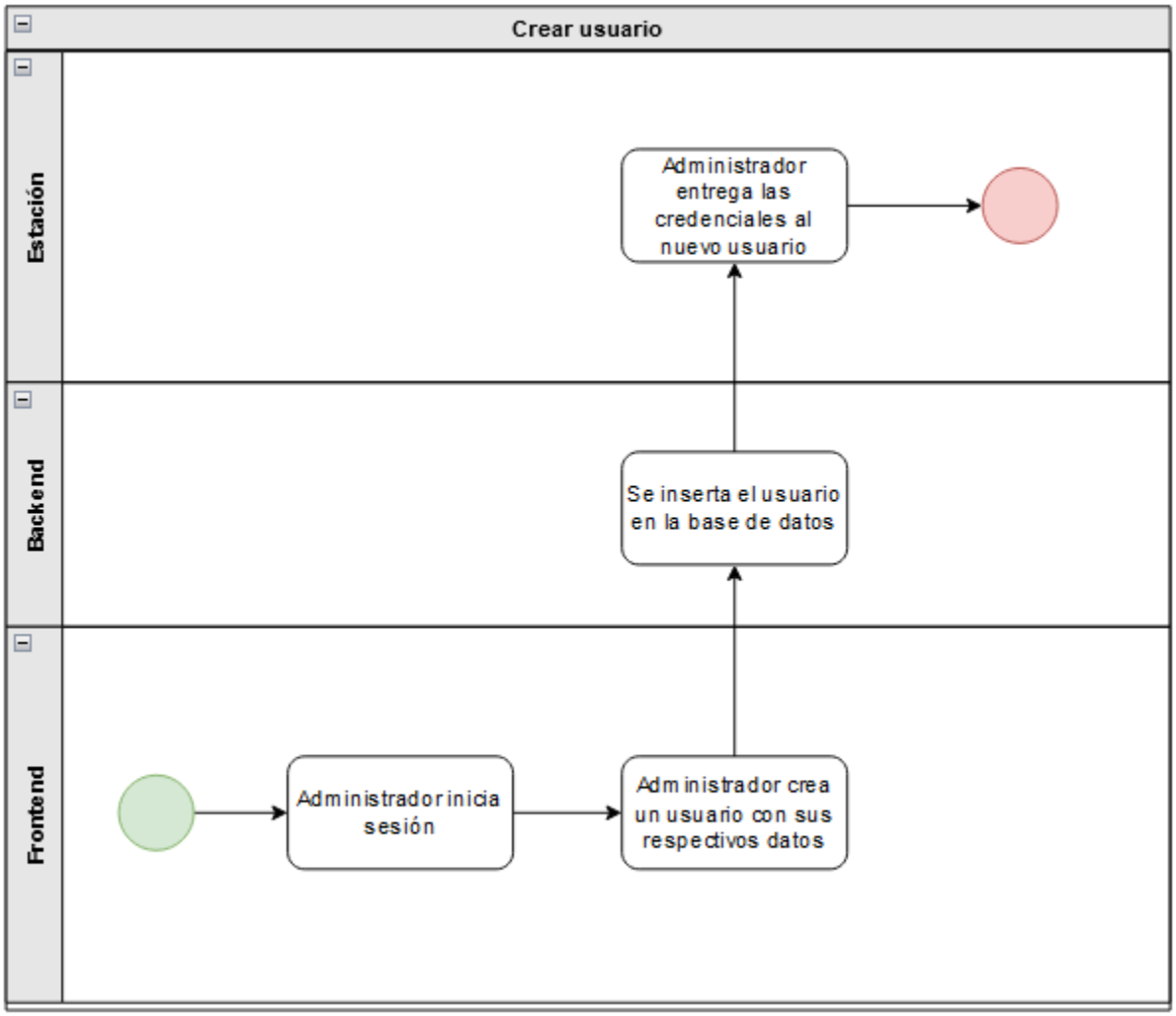


Figura 24. *Diagrama de procesos de creación de usuarios*

Fuente: elaboración propia.

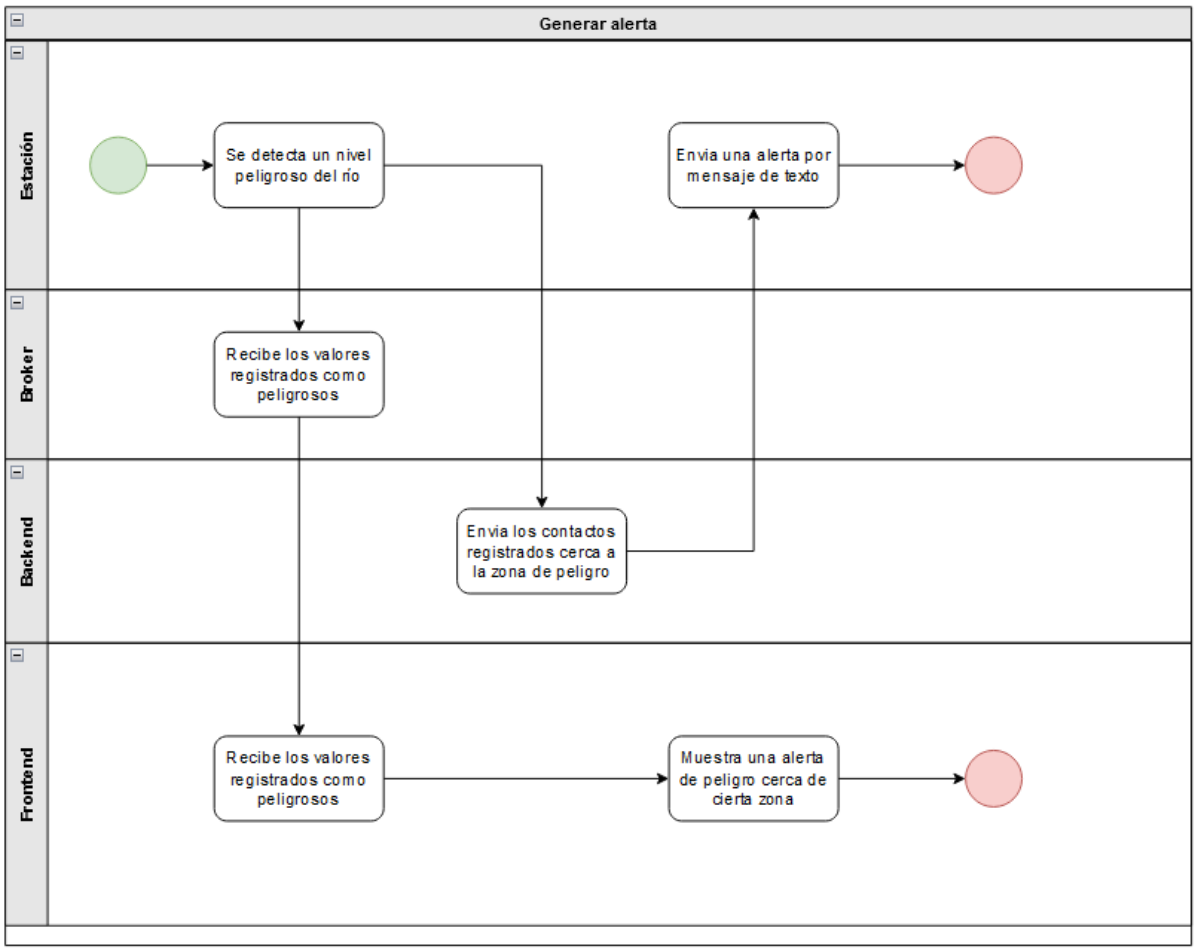


Figura 25. *Diagrama de procesos de alerta general*

Fuente: elaboración propia.

4.3.5 Modelo de datos

Para una visión clara y estructural de la base de datos no relacional en MongoDB, se construyó un diagrama de modelo de datos, para la correcta interacción en base a referencias entre tablas. Dicho modelo se puede observar establecido o reglamentado en la sección de modelos en el backend donde la librería mongoose permite crear los denominados esquemas y su patrón de información según tipo de dato y características de requerido o con un valor por defecto.

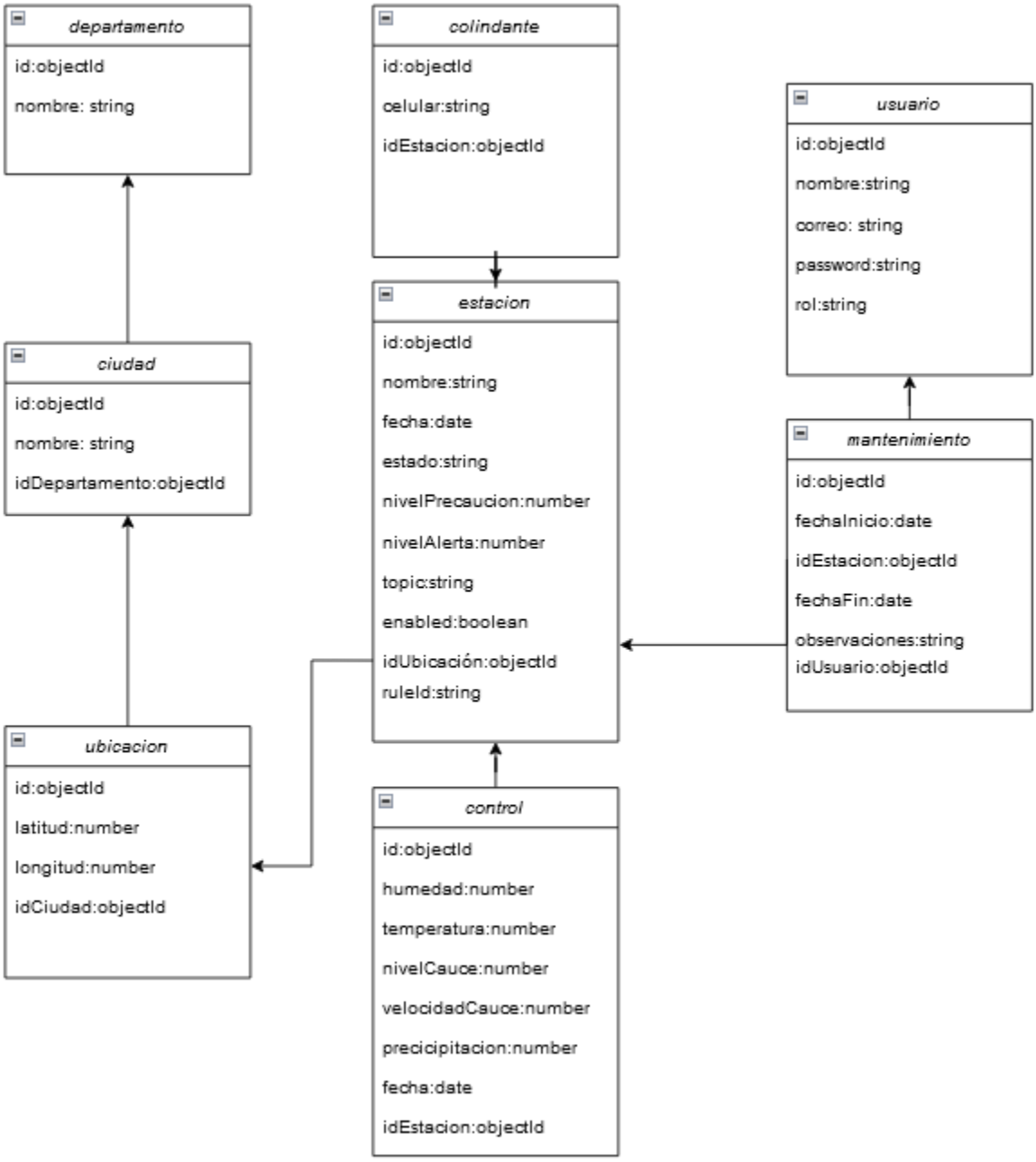


Figura 26. *Modelo de datos*

Fuente: elaboración propia.

4.3.6 Diseño del mockup

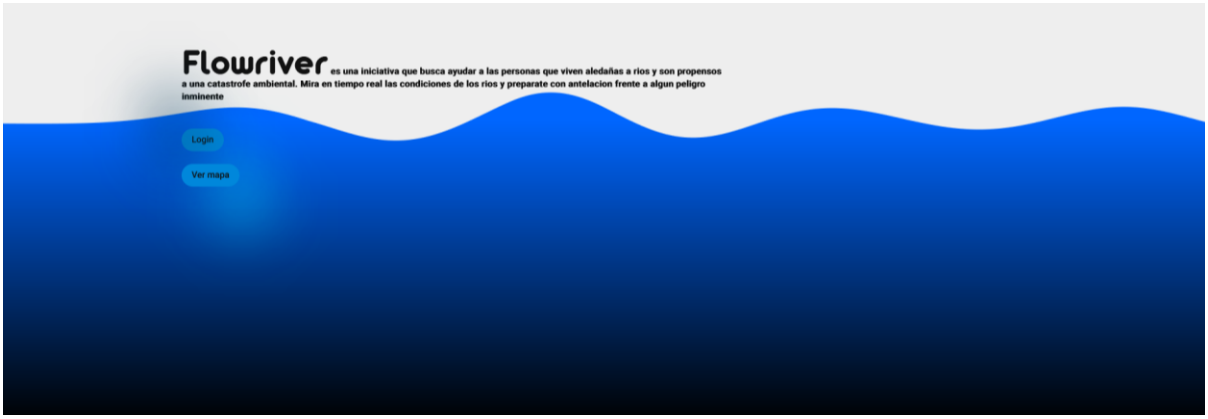


Figura 27. *Vista de inicio*

Fuente: elaboración propia.

En esta vista se muestra información relacionada a las estaciones como las variables geográficas tomadas recientemente y su variación positiva o negativa.

Figura 28. *Vista de mapa por estaciones*

Fuente: elaboración propia.

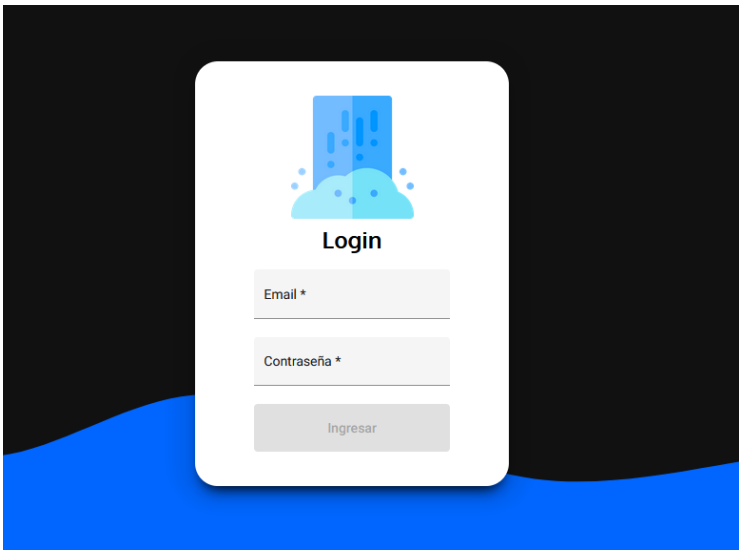


Figura 29. *Vista de login*

Fuente: elaboración propia.

Vistas del dashboard

En este apartado, donde solo técnicos y administradores pueden acceder, se pueden ver la mayoría de estadísticas proporcionadas por la base de datos, tales como la tabla de mantenimientos, tabla de usuarios y reportes de las variables obtenidas de los sensores en cada estación. (Ver Figura 30)

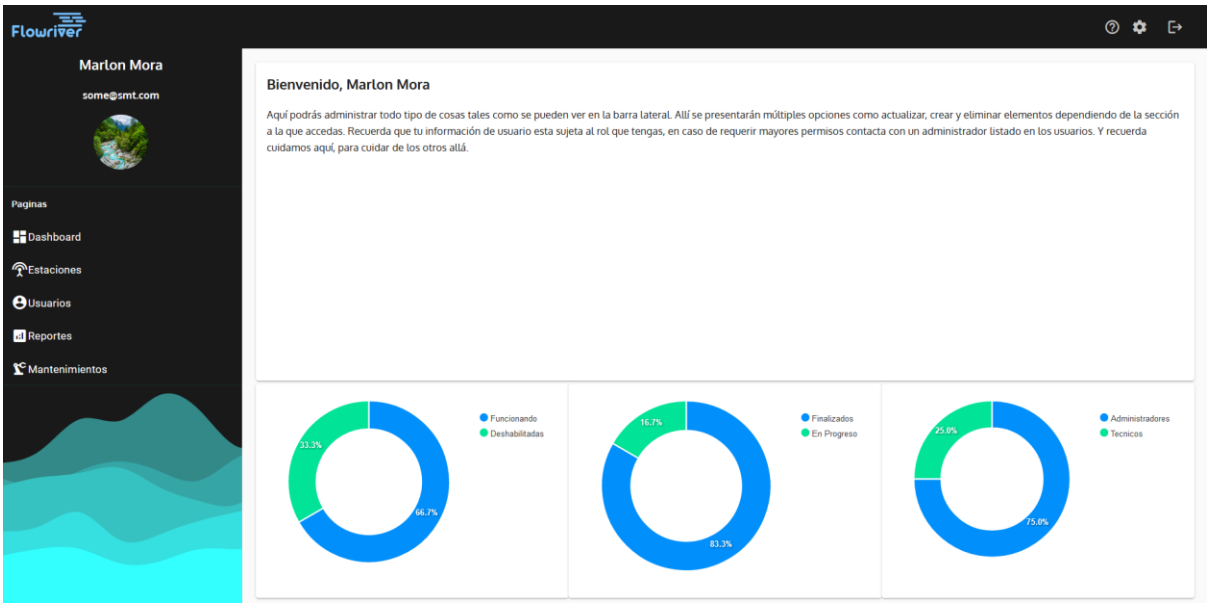


Figura 30. *Vistas del dashboard - Inicio*

Fuente: elaboración propia.

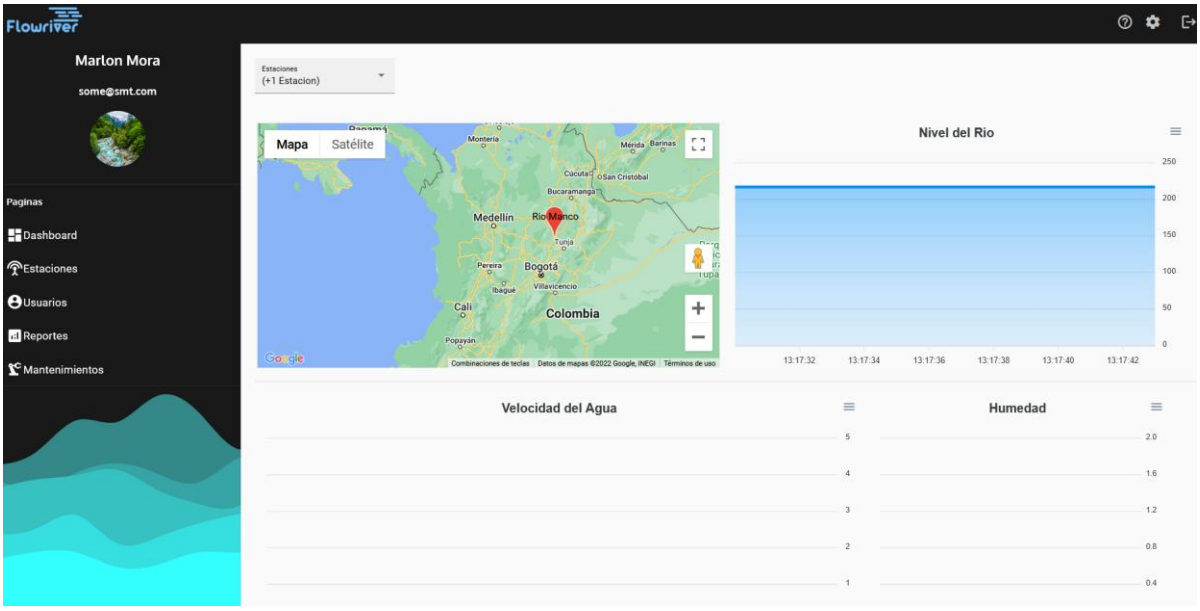


Figura 31. *Vistas del dashboard - Reportes*

Fuente: elaboración propia.

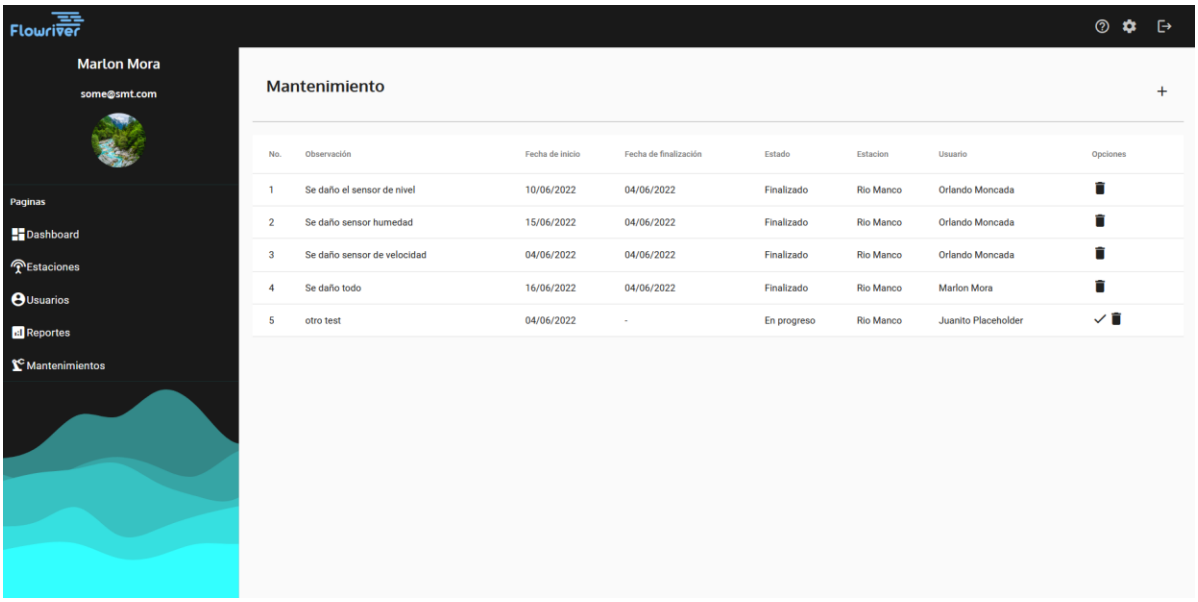


Figura 32. *Vistas del dashboard - Mantenimientos*

Fuente: elaboración propia.

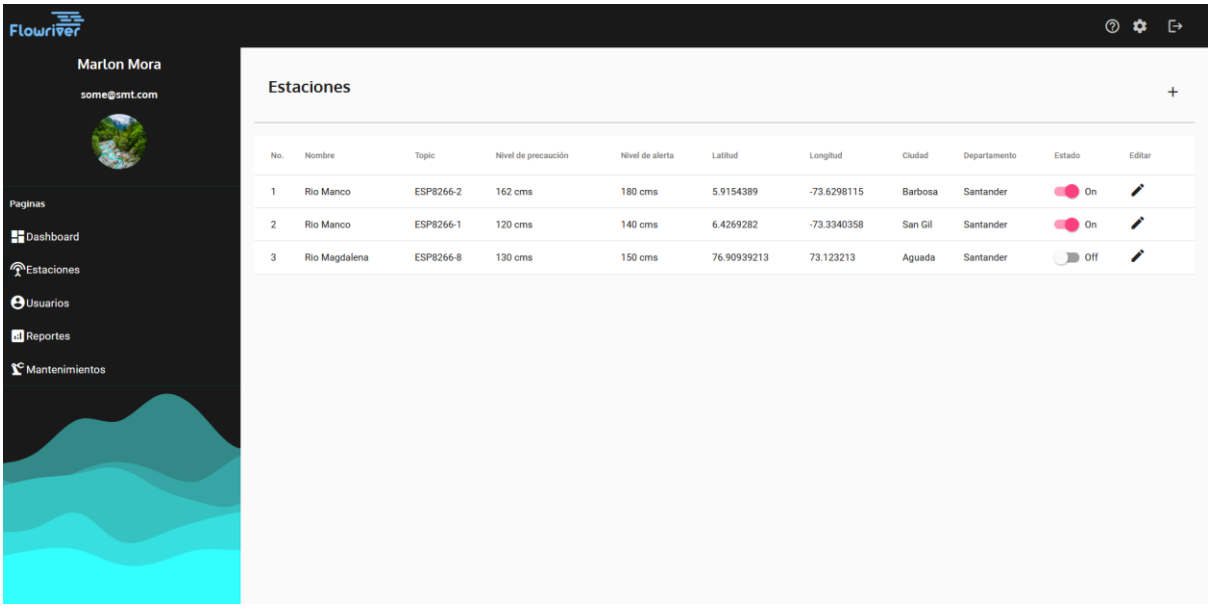


Figura 33. *Vistas del dashboard - Estaciones*

Fuente: elaboración propia.

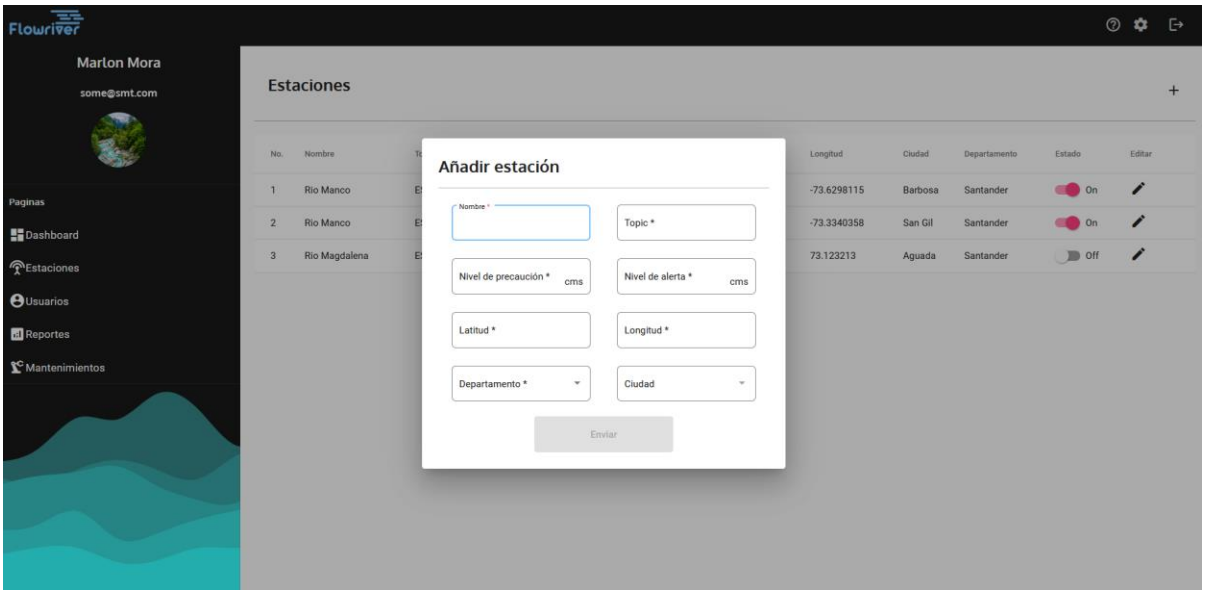


Figura 34. *Vistas del dashboard - Agregar estación*

Fuente: elaboración propia.

En esta vista solo el administrador tiene vista del icono de editar y eliminar usuario.

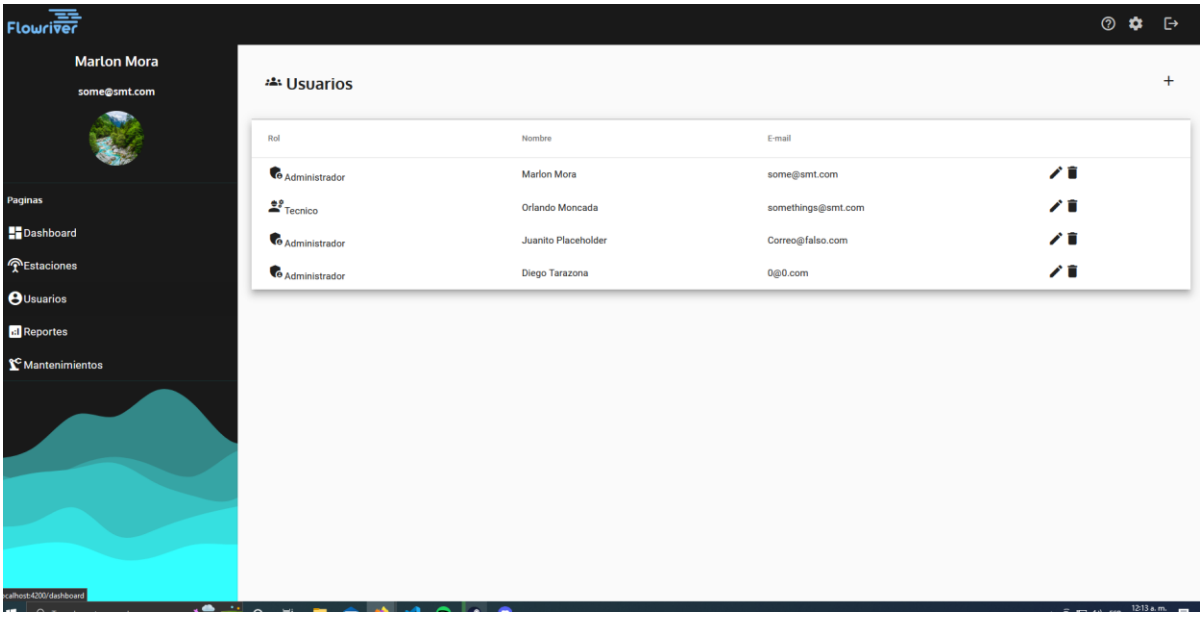


Figura 35. *Vistas del dashboard – Usuarios*

Fuente: elaboración propia.

4.3.6 Diseño de la arquitectura

Es importante definir las herramientas de trabajo después de tener estructurados los mockups y en base a los requerimientos solicitados. Para facilitar esto, se realiza el diseño de arquitectura el cual permite ver a gran escala la complejidad en la unión de las partes y así lograr tener un sistema organizado. (Ver Figura 36)

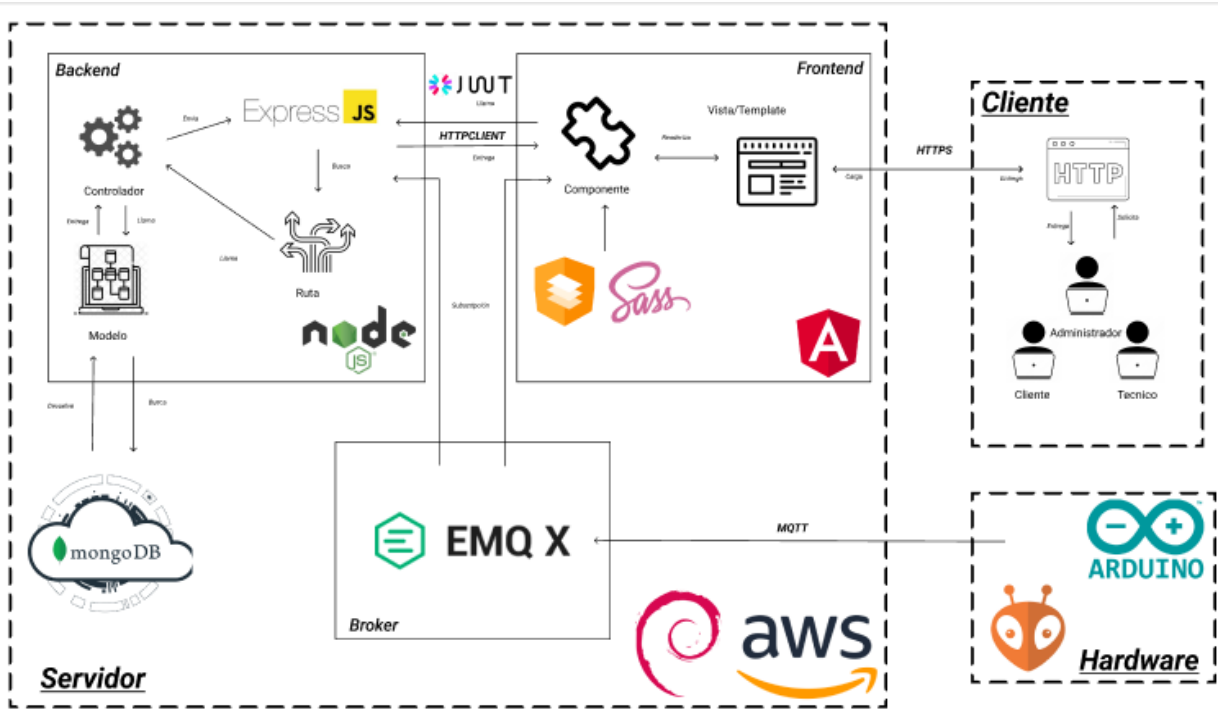


Figura 36. *Diseño de la arquitectura del sistema*

Fuente: elaboración propia.

Como se puede observar en la figura, el diseño arquitectónico consta de 3 subdivisiones las cuales son:

- Hardware: Esta sección hace referencia a los sensores y el microcontrolador que permite la transmisión de los datos hacia el servidor.

- Servidor: Esta sección hace referencia a la unión de broker, el frontend y el backend, y como los datos son transformados, trasladados y proporcionados al cliente.
- Cliente: Esta sección hace referencia al navegar donde el usuario accede a la página web.

Así mismo se puede descomponer en 3 el servidor como se describió anteriormente. En cuanto al Backend, podemos hacer referencia que a través de las rutas se pueden hacer peticiones API las cuales son fáciles de crear gracias a la librería Express, que redirige la petición al controlador quien realizar el proceso de transformación u obtención de la información por medio de los modelos, los cuales crean una interfaz de información renderizada por mongoose, librería cuya finalidad de uso es adaptar los objetos de javascript al esquema de datos de MongoDB.

Por parte del frontend, al hacer uso de un framework SPA como lo es Angular, es importante adoptar la visión de trabajar por componentes, por lo cual Angular Material ayuda en gran manera a la hora de tener componentes estilizados tales como inputs o selects. Debido a que el proyecto también maneja un carácter estadístico, Apex Charts es una librería que agiliza la creación de gráficos. En cuanto a la organización esta se divide en vistas alimentadas por los componentes antes mencionados y renderizados con sass y assets, las cuales van cambiando dependiendo de la ruta especificada en el navegador, lo cual es comprendido como router o el componente router-outlet. Además, con ayuda de HTTPClient, una librería de Angular que permite realizar las peticiones API y obtener los datos dados por cada endpoint se crearán los servicios para cada modelo de datos maquetados como interfaces de typescript.

4.3.7 Control de versiones

Para una gestión fácil de la creación de archivos en el software es necesario una herramienta para el control de versiones como lo es GitHub que permite numerosas integraciones como Heroku o Firebase. Además, y siendo lo más importante con ayuda de las ramas, se puede mantener un orden en la integración del código de cada uno de los desarrolladores.

4.3.7.1 Repositorios

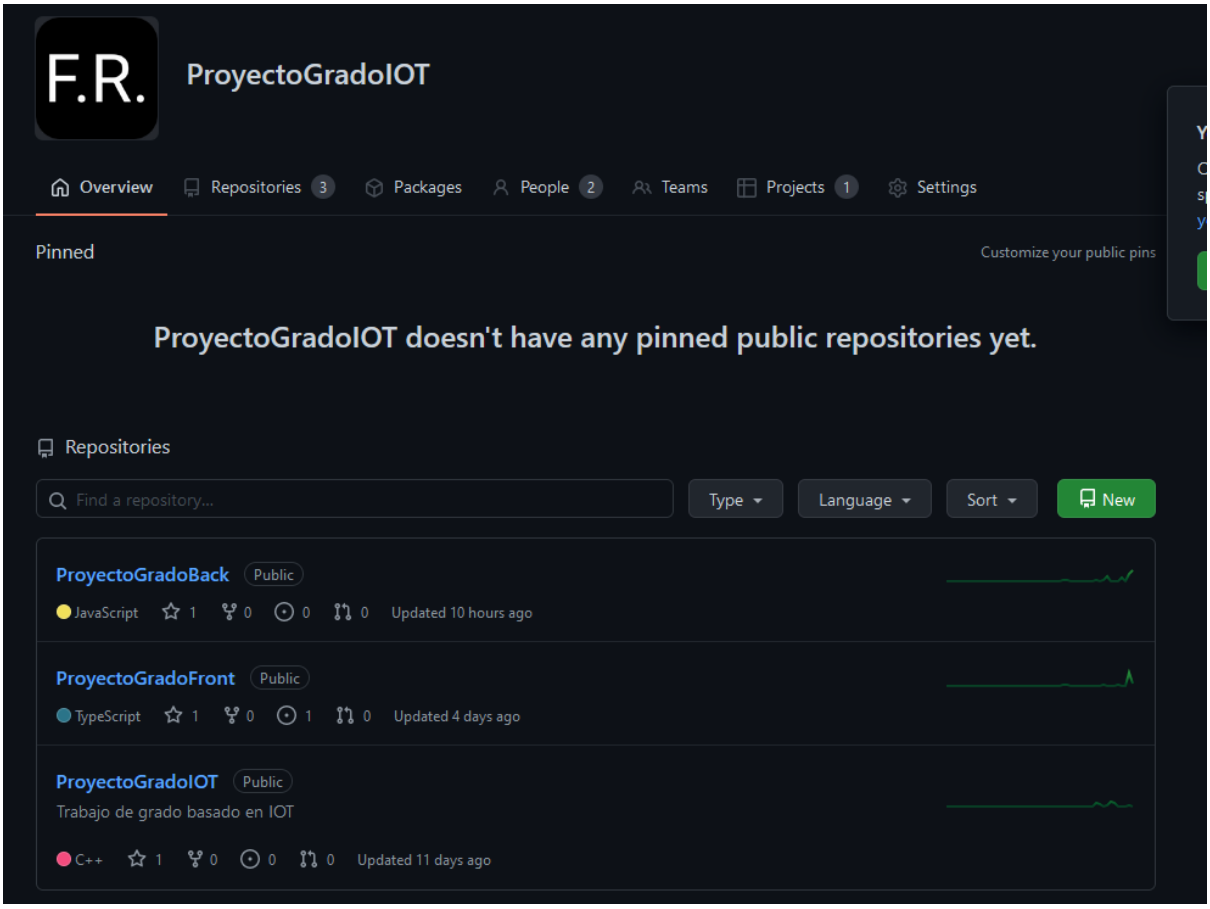


Figura 37. *Repositorios*

Fuente: GitHub. (2022). *Repositorios*. https://github.com/ProyectoGradoIOT

4.3.7.2 Grafica de commits

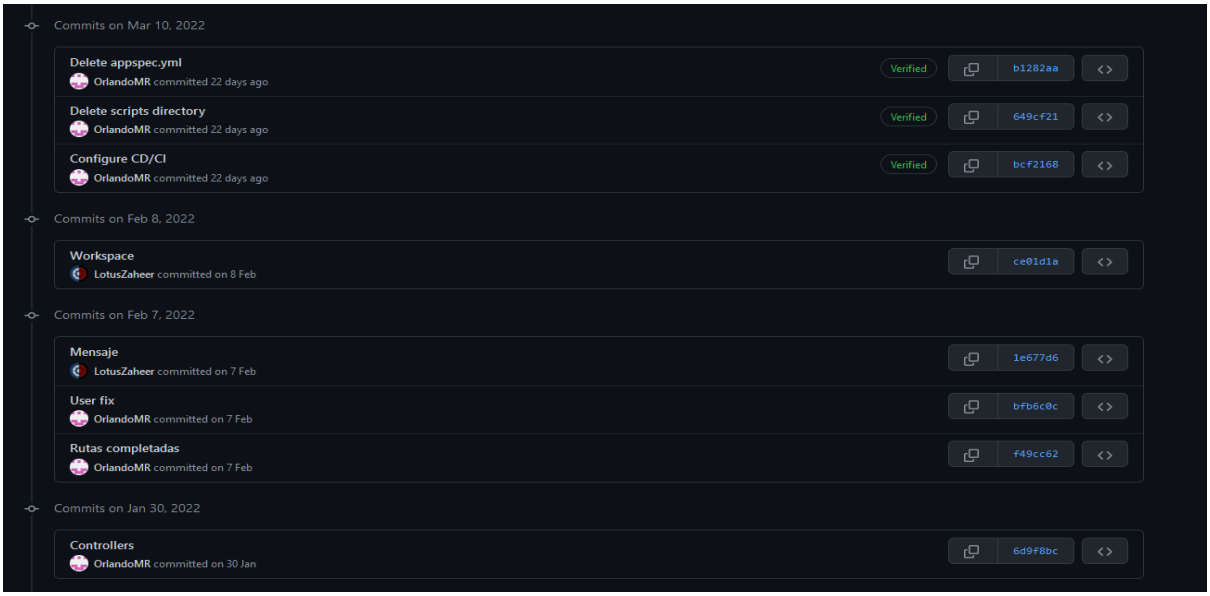


Figura 38. *Grafica de commits*

Fuente: GitHub. (2022). *Repositorios*. https://github.com/ProyectoGradoIOT

4.3.7.3 Panel de gestión de actividades

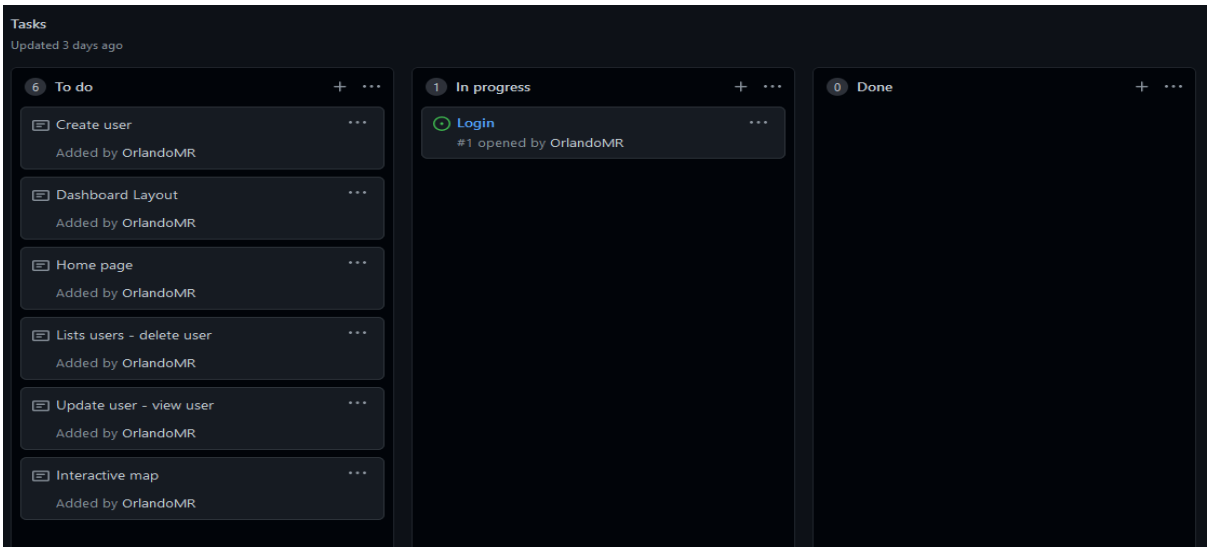


Figura 39. *Panel de gestión de actividades*

Fuente: GitHub. (2022). *Panel de gestión de actividades*.
https://github.com/ProyectoGradoIOT/ProyectoGradoFront/issues

4.4 Diseño del Hardware y Software

4.4.1 Creación del servidor y publicación

4.4.1.1 EC2

Para la creación de un servidor en AWS primero se debe contar con una cuenta en Amazon Web Services e iniciar sesión en la consola. (Ver Figura 40)

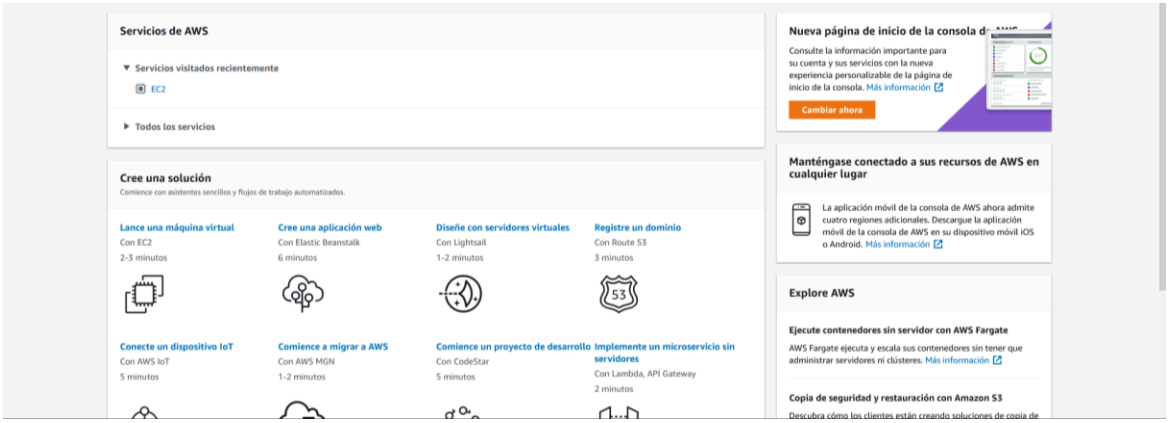


Figura 40. *Consola de administración de AWS*

Fuente: elaboración propia.

Una vez allí, en la barra superior se busca el servicio llamado EC2, así encontrando este e ingresando para configurar la instancia. Se puede observar el panel de opciones que existen para la configuración de nuestra máquina virtual para un óptimo funcionamiento en concordancia con nuestro objetivo. (Ver Figura 41)

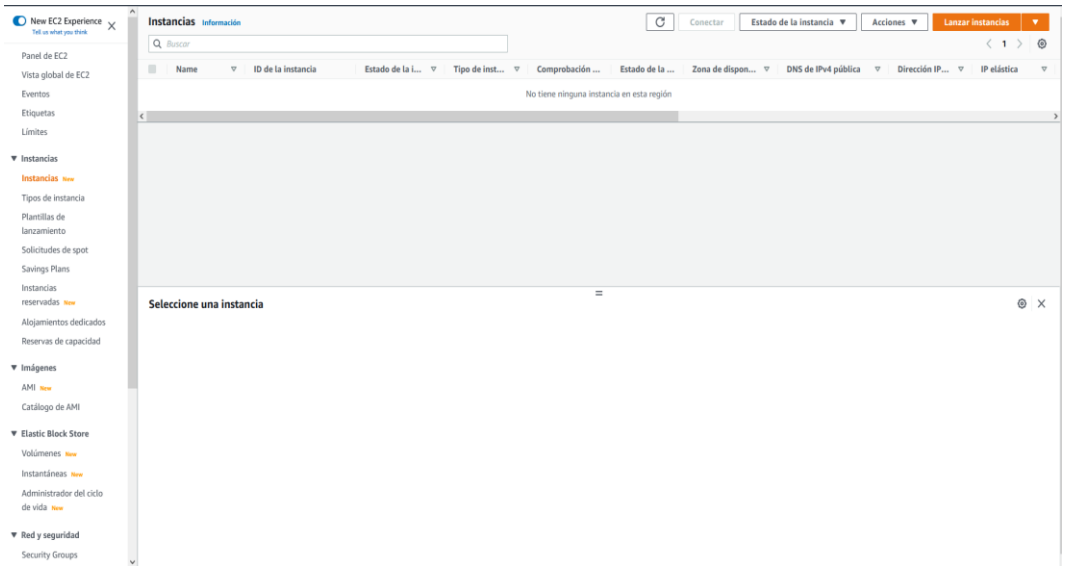


Figura 41. *Panel de opciones*

Fuente: elaboración propia.

Para la creación de una instancia se debe seguir un flujo de opciones empezando desde el botón ubicado en la esquina superior derecha llamado “Lanzar instancia”, el cual nos va a dirigir a los siguientes pasos:

El primero consta de elegir una AMI o en palabras sencillas una imagen de sistema operativo, para este caso, se decidió optar por Ubuntu 20.04 LTS x64 ya que permite la fácil integración con el broker MQTT. (Ver Figura 42)

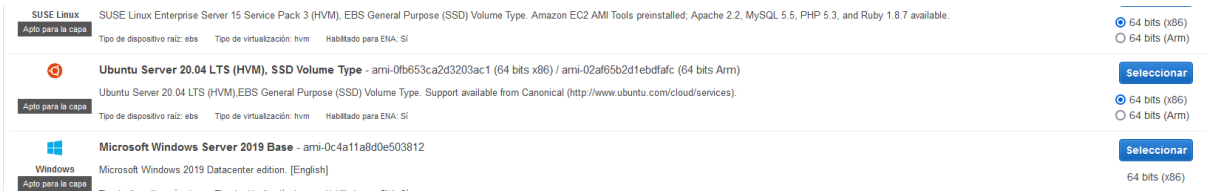


Figura 42. *Lanzar instancia*

Fuente: elaboración propia.

Luego elegimos el tipo de instancia dependiendo de las combinaciones de CPU, memoria almacenamiento y capacidad de red, para este caso se optó por la instancia gratuita, ya que el prototipo no requiere un hardware de grandes capacidades. (Ver Figura 43)

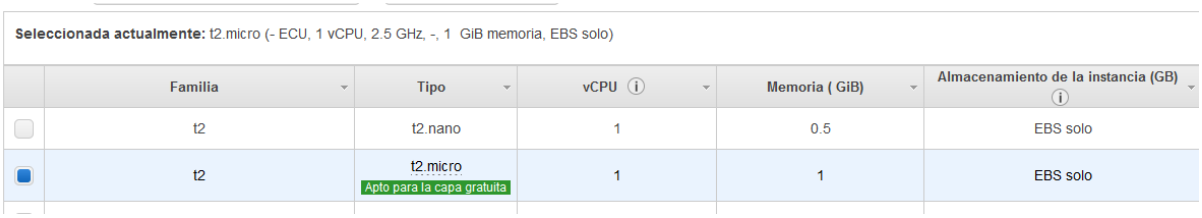


Figura 43. *Tipo de instancia*

Fuente: elaboración propia.

El siguiente paso fue añadir un rol de IAM y datos de usuario para facilitar posteriormente el pipeline de integración continua. (Ver Figura 44)

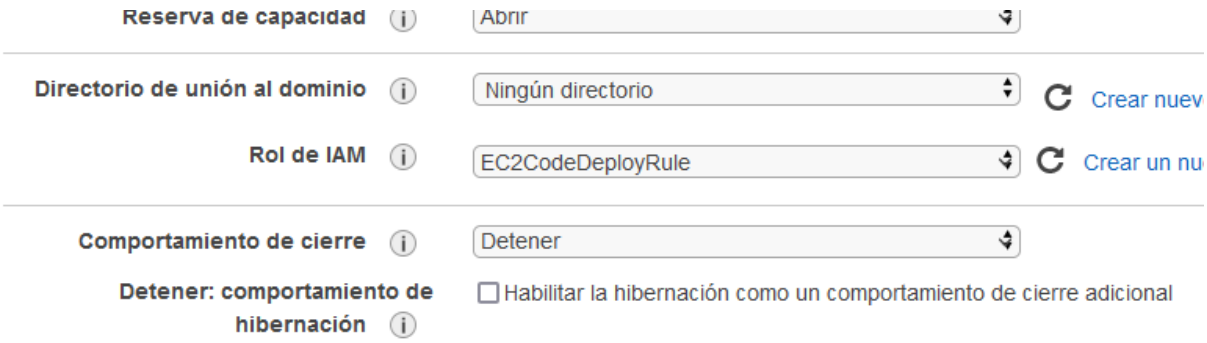


Figura 44. *Rol de la IAM*

Fuente: elaboración propia.

Una vez agregada esta información procedemos a la adición de almacenamiento, para este caso añadimos 16GB de tipo SSD para una alta velocidad a la hora de grabar/leer en disco. (Ver Figura 45)

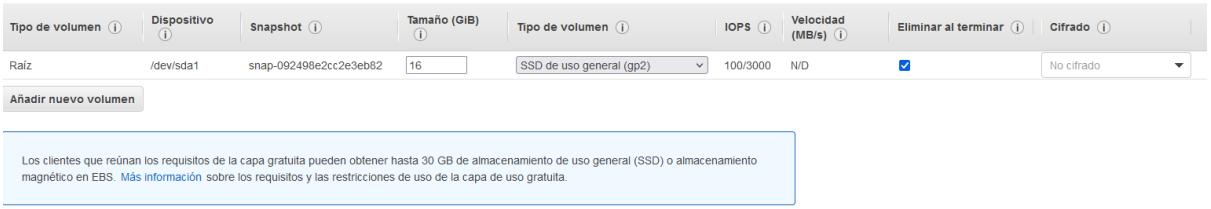
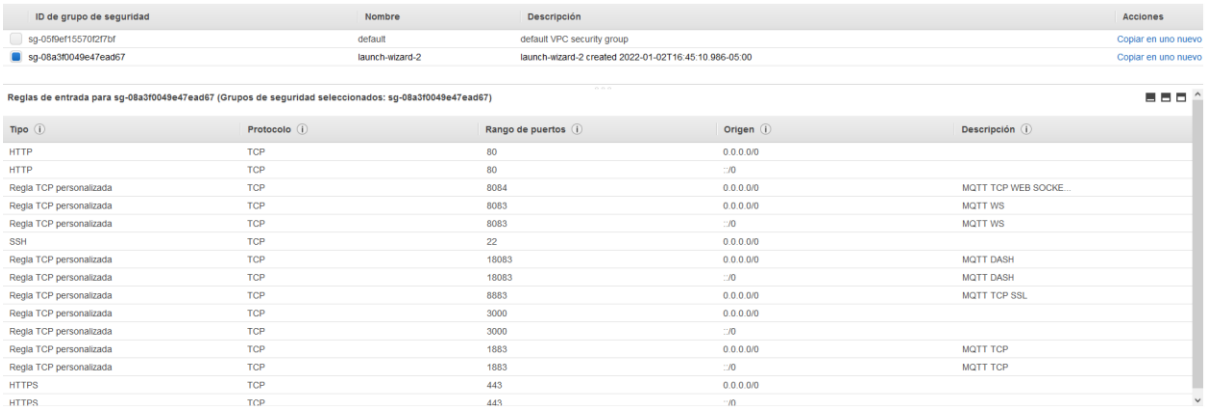


Figura 45. *Adición de almacenamiento*

Fuente: elaboración propia.

En el siguiente paso se recomienda añadir una etiqueta o tag, la cual consta de una clave-valor. Pero debido a que después en otra configuración se requiere crear otras etiquetas este paso es descartado y se continúa.

Para el último paso, se debe configurar el security group, el cual permite definir los puertos que tendrán conexión. Cabe aclarar que el security group fue creado previamente, pero en este paso se puede añadir directamente. Como ejemplo podemos crear una regla TCP para el puerto 22 que nos permitirá la conexión mediante SSH de un computador a la consola de esta instancia. (Ver Figura 46)



ID de grupo de seguridad	Nombre	Descripción	Acciones
sg-05f9ef15570f2f7bf	default	default VPC security group	Copiar en uno nuevo
sg-08a3f0049e47ead67	launch-wizard-2	launch-wizard-2 created 2022-01-02T16:45:10.986-05:00	Copiar en uno nuevo

Reglas de entrada para sg-08a3f0049e47ead67 (Grupos de seguridad seleccionados: sg-08a3f0049e47ead67)

Tipo	Protocolo	Rango de puertos	Origen	Descripción
HTTP	TCP	80	0.0.0.0/0	
HTTP	TCP	80	::/0	
Regla TCP personalizada	TCP	8084	0.0.0.0/0	MQTT TCP WEB SOCKE...
Regla TCP personalizada	TCP	8083	0.0.0.0/0	MQTT WS
Regla TCP personalizada	TCP	8083	::/0	MQTT WS
SSH	TCP	22	0.0.0.0/0	
Regla TCP personalizada	TCP	18083	0.0.0.0/0	MQTT DASH
Regla TCP personalizada	TCP	18083	::/0	MQTT DASH
Regla TCP personalizada	TCP	8883	0.0.0.0/0	MQTT TCP SSL
Regla TCP personalizada	TCP	3000	0.0.0.0/0	
Regla TCP personalizada	TCP	3000	::/0	
Regla TCP personalizada	TCP	1883	0.0.0.0/0	MQTT TCP
Regla TCP personalizada	TCP	1883	::/0	MQTT TCP
HTTPS	TCP	443	0.0.0.0/0	
HTTPS	TCP	443	::/0	

Figura 46. *configuración del security group*

Fuente: elaboración propia.

Por último, AWS nos muestra un breve resumen de la configuración de nuestra instancia a lanzar seguido del lanzamiento de esta. Una vez iniciada ya podemos acceder mediante SSH como se describió previamente.

4.4.1.2 Dominio

Para conectar con un dominio personalizado se hizo uso de la plataforma Hostinger, la cual da precios asequibles durante el primer año, así que se procedió a crear la cuenta mediante Google Social Media Connect, una vez iniciada la sesión se nos ofrece distintas opciones de

compra, tal como un servicio cloud o tan solo un dominio, luego de escoger el dominio buscamos un nombre para este, en este caso se optó por flowriver online e inmediatamente se dispone a realizar el pago. Después de atravesar por la pasarela de pagos y dar nuestra información financiera se nos proveen distintas opciones para la configuración del dominio. El único ajuste será agregar la ruta de la dirección a la que hará referencia nuestro dominio, para este caso tomamos la ip obtenida de nuestro EC2.

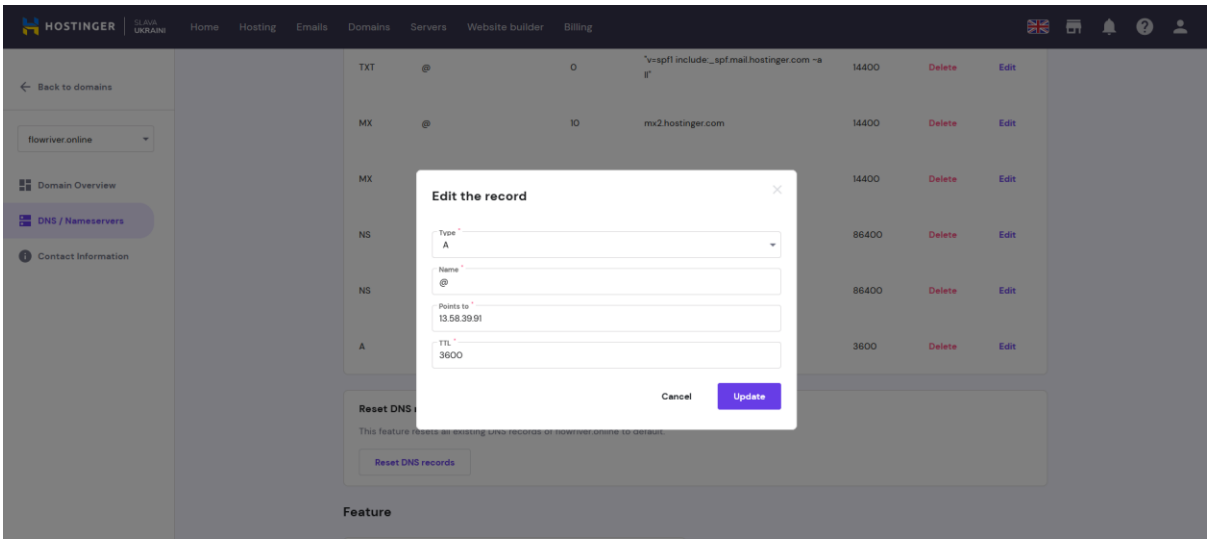


Figura 47. *Dominio*

Fuente: elaboración propia.

4.1.1.3 Instalación EMQX

Para la instalación del broker en nuestro servidor, nos conectamos a la terminal del servidor mediante SSH al puerto 22, usando la key generada por Amazon dada al finalizar la creación del servidor EC2. Para ello usamos PUTTY en Windows. (Ver Figura 48)

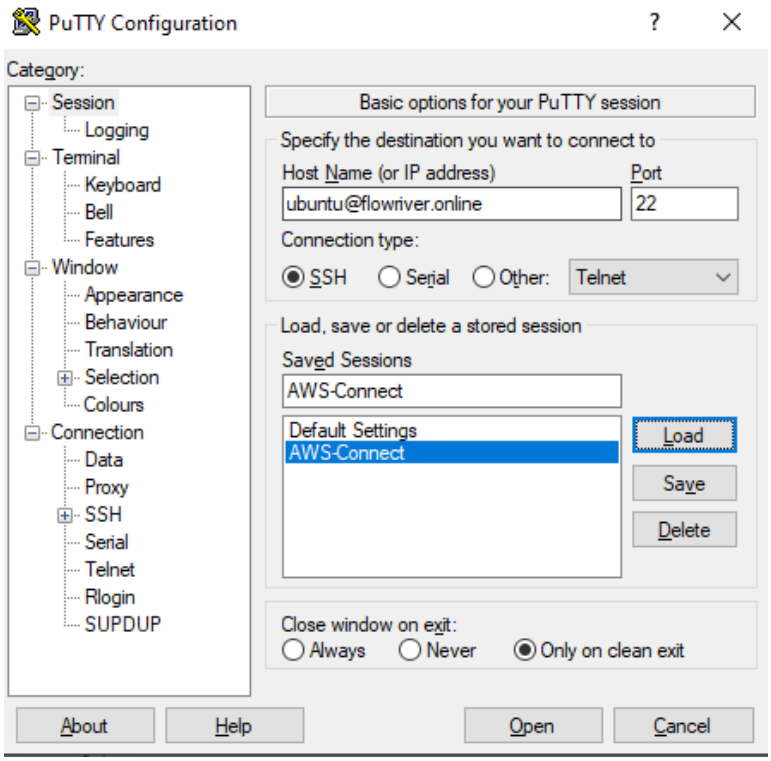


Figura 48. *Creación del servidor EC2*

Fuente: elaboración propia.

Luego de conectarnos a la terminal ingresamos los siguientes comandos para la descarga e instalación de EMQX.

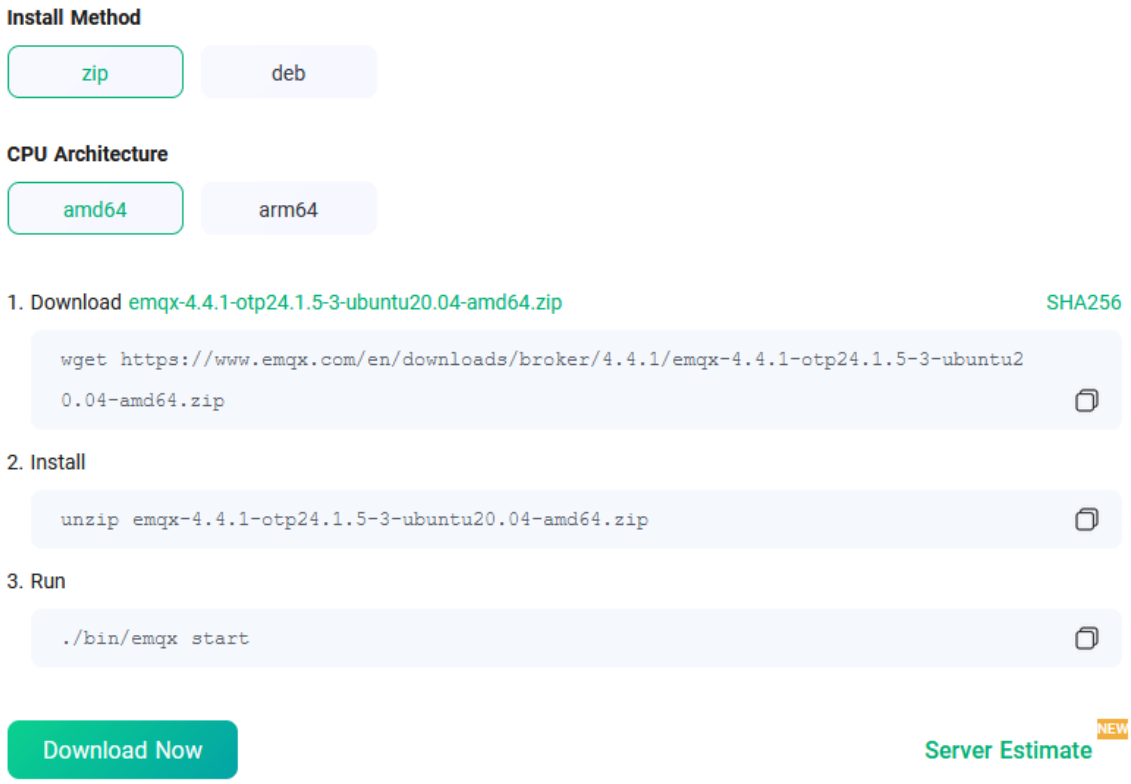


Figura 49. *descarga e instalación de EMQX*

Fuente: elaboración propia.

4.4.3 Creación de la base de datos y cargue a la nube

Mongo Atlas es un servicio de creación y alojamiento de bases de datos en la nube, lo cual permite acceder a los datos desde cualquier lado del mundo simplemente con las credenciales. Su creación es fácil e intuitiva empezando con un registro de cuenta en la plataforma de Mongo Atlas, luego al entrar al dashboard nos ofrece la posibilidad de crear un clúster eligiendo parámetros como el lugar de alojamiento (AWS, Azure, etc), capacidad y tipo de almacenamiento. (Ver Figura 50)

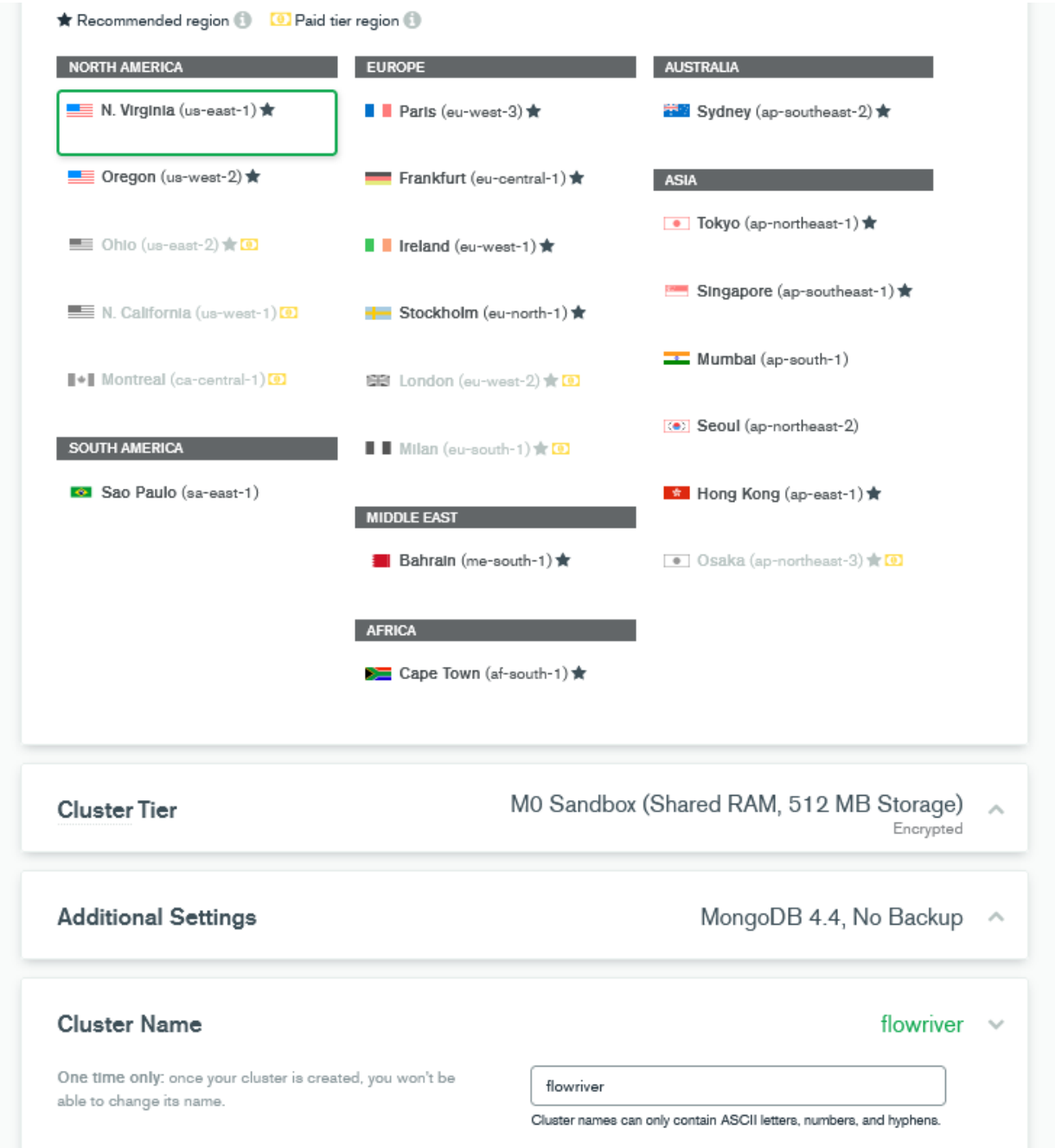


Figura 50. *Creación de la base de datos y cargue a la nube*

Fuente: elaboración propia.

Una vez creado el clúster se procede a conectar el servidor de Node con la base de datos, para ello, se usa la librería Mongoose que además de realizar la conexión permite interactuar con ella haciendo posible crear peticiones de forma segura y eficaz. (Ver Figura 51)

```
const mongoose = require("mongoose");

const dbConnection = async() => {
    try {

        await mongoose.connect(process.env.MONGODB_CNN, {
            useNewUrlParser: true,
            useUnifiedTopology: true,
            useCreateIndex: true,
            useFindAndModify: false
        });
        console.log('Base de datos online')
    } catch (error) {
        throw new Error('Error a la hora de conectar la base de datos');
    }
}

module.exports = {
    dbConnection
}
```

Figura 51. *Librería Mongoose*

Fuente: elaboración propia.

4.4.4 Construcción del Hardware

Para el desarrollo del hardware de cada una de las estaciones fue requerido como módulo principal la placa de desarrollo NodeMCU V3 basada en el chip ESP8266, un sensor de humedad y temperatura DHT-22, un sensor del nivel del agua por ultrasonido HC-SR04, un pluviómetro totalizador basado en el sensor HW-038 y un sensor de caudal FS300A adaptado. (Ver Figura 52)

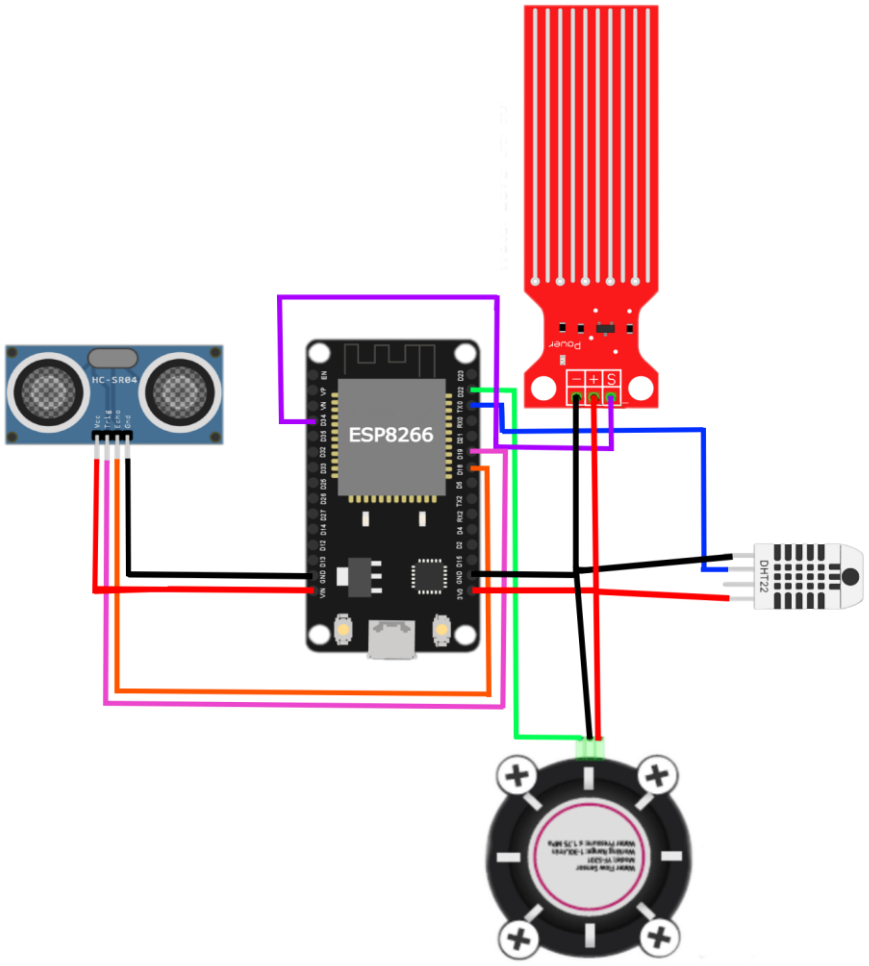


Figura 52. *Simulación del circuito*

Fuente: elaboración propia.

Por medio de la IDE Platformio se programó el módulo ESP8266 a través del Framework de Arduino, el cual se codifica en el lenguaje C++ , para recibir los datos suministrados por los diferentes sensores y enviarlos al broker EQM-X utilizando un websocket que aprovecha el protocolo TCP/IP suministrado por la placa de desarrollo.

4.4.5 Configuración del Hardware y conexión al broker

El hardware de cada estación está configurado para enviar la información en formato JSON a el Broker EQM-X, utilizando el protocolo de mensajería MQTT, para conseguir dicha conexión de forma segura fue necesario especificar por defecto datos de dirección del servidor en este caso flowriver.online y un puerto de acceso autorizado el 1883, además para cada estación de forma independiente se le asigna un usuario único compuesto por la referencia de la placa de desarrollo y un número de serie asignado según su orden de agregación al sistema y una contraseña de acceso. (Ver Figura 53)

```
//----------- MQTT CONFIG -----------//
const char *mqtt_server = "flowriver.online";
const int mqtt_port = 1883;
const char *mqtt_user = "ESP8266-2";
const char *mqtt_pass = "flowriver";
const char *root_topic_subscribe = "flowriver/#";
const char *root_topic_publish = "flowriver/ESP8266-2";
```

Figura 53. *Configuración del Hardware y conexión al broker*

Fuente: elaboración propia.

Para poder realizar la conexión al sistema MQTT es necesario primero configurar un cliente HTTP que aproveche el protocolo TCP/IP, para ello se utilizará la librería de Arduino ESP8266HTTPClient con la cual podremos crear una rutina de configuración, conexión y callback periódico que permita evaluar el estado de la conexión al servidor de forma periódica.

Una vez realizada la conexión al sistema MQTT cada estación el dispositivo se suscribe a la ruta principal del proyecto "flowriver/#" desde la cual recibirá órdenes que podrán afectar desde la taza de publicación de la información como también enviar notificaciones vía sms a los usuarios más cercanos, el dispositivo también se enlazará a una ruta única asignada para cada dispositivo la cual se compondrá por la ruta principal del proyecto flowriver/ concatenada al usuario asignado a cada dispositivo.

4.4.6 Desarrollo del backend

Mediante una arquitectura REST API se realizó la lógica del programa para optimizar la transformación de la información obtenida en la base datos, a su vez siendo este proceso seguro ya que el cliente no puede acceder a este de forma fácil, evitando numerosos problemas de seguridad.

4.4.6.1 Modelos

Debido a la integración entre MongoDB y Node es necesario crear modelos comúnmente denominados esquemas, para evitar errores de tipado, manejar validaciones y sobre todo crear una conexión entre la base y el backend. Para agilizar estas tareas se cuenta con la librería mongoose, la cual crea los esquemas con diversas validaciones. (Ver Figura 54)

```
const { Schema, model } = require('mongoose');


const UsuarioSchema = Schema({
    nombre: {
        type: String,
        required: [true, 'El nombre es obligatorio']
    },
    correo: {
        type: String,
        required: [true, 'El correo es obligatorio'],
        unique: true
    },
    password: {
        type: String,
        required: [true, 'La contraseña es obligatoria']
    },
    rol: {      You, 5 months ago • Models
        type: String,
        required: true,
        emun: ['ADMIN_ROL', 'TEC_ROL']
    }
})

UsuarioSchema.methods.toJSON = function() {
    const { __v, password, _id, ...user } = this.toObject();
    user.uid = _id;
    return user;
}

module.exports = model('Usuario', UsuarioSchema);
```

Figura 54. *Estructura de un esquema de mongoose*

Fuente: elaboración propia.

Como se puede observar en la figura la misma librería nos brinda el manejo de errores cuando no se cumple una validación. Este modelo es importante ya que las mismas reglas

creadas para cada modelo son transmitidas a la base de datos, haciendo posible una base organizada.

4.4.6.2 Controladores

Los controladores diseñados para cada tipo de solicitud HTTP (GET, POST, PUT, DELETE) son agrupados por cada tipo de modelo, aquí es donde se realiza el proceso de obtención, transformación y retorno de la información, así estructurando con rapidez el CRUD básico y demás funciones requeridas por el cliente.

4.4.6.3 Middleware

Los middlewares hacen referencia a todo tipo de funciones que puedan ser reutilizadas para cada tipo de controlador, así eliminando la redundancia y atomizando el código en cada archivo.

Los principales archivos creados en esta sección fueron la validación del JSON WEB TOKEN (JWT) y la validación del rol del usuario que envía la solicitud para obtener o crear información en la base de datos, así los permisos para el dashboard son asegurados a nivel de backend en caso de un error en la validación del front-end.

4.4.6.4 Helpers

En esta sección del código, se crearon funciones relacionadas a la base de datos y la seguridad del token. Aquí se crea el JSON Web Token (JWT) y también se verifica la conexión

con la base de datos, para proteger la integridad de esta misma. Estas funciones por lo general no irían en las demás secciones ya que manejan un nivel de seguridad más importante y pueden ser llamadas simultáneamente por los controladores lo cual rompería el principio DRY (Don't repeat yourself).

4.4.6.5 Rutas

Las rutas hacen referencia a la entrada de las solicitudes realizadas por el cliente, estas son manejadas gracias a la librería express que permite manejar de forma fácil estas solicitudes. Tomando los datos de entrada y pasándolos al controlador para realizar su respectiva función y devolver exitosamente una respuesta o bien manejando un error en el procedimiento asignado.

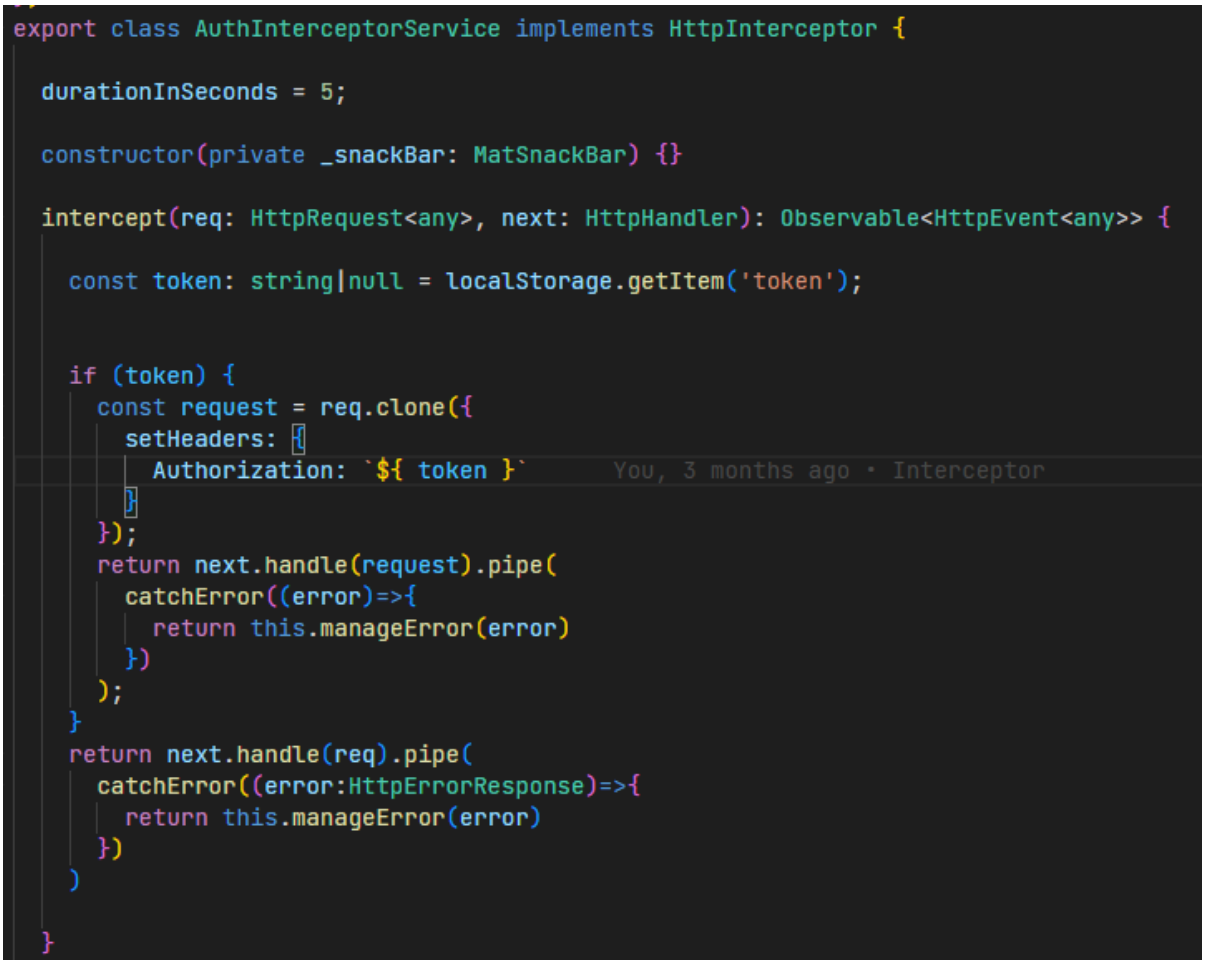
4.4.7 Desarrollo del Frontend

En el desarrollo frontend se tuvieron varios aspectos para la organización y creación de este con el fin de optimizar, automatizar y asegurar el proyecto.

4.4.7.1 Guards e interceptors

Para la seguridad del acceso y permisos en las rutas, se crearon los guards los cuales permiten, que el usuario al intentar acceder a una sección de la aplicación web y no tiene los permisos suficientes, este es enviado a otra página, en la mayoría de casos hacia el inicio. Contando con los *interceptors* estos nos permiten poner en el header *authorization* el token de acceso del usuario generado al momento de loguearse. Además, otra utilidad de este, es el fácil

manejo de los errores ya que con la ayuda de un componente *toast* despliega si ocurrió un error en el servicio. (Ver Figura 55)



```
export class AuthInterceptorService implements HttpInterceptor {

  durationInSeconds = 5;

  constructor(private _snackBar: MatSnackBar) {}

  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {

    const token: string|null = localStorage.getItem('token');


    if (token) {
      const request = req.clone({
        setHeaders: {
          Authorization: `${ token }`      You, 3 months ago · Interceptor
        }
      });
      return next.handle(request).pipe(
        catchError((error)=>{
          return this.manageError(error)
        })
      );
    }
    return next.handle(req).pipe(
      catchError((error:HttpErrorResponse)=>{
        return this.manageError(error)
      })
    )

  }
```

Figura 55. *Interceptor*

Fuente: elaboración propia.

4.4.7.2 Material Angular

Además de contar con archivos SCSS, una derivación de SASS (*Syntactically Awesome Stylesheets*), la cual es un preprocesador CSS que permite generar, de manera automática, hojas de estilo, añadiéndoles características tales como variables, funciones, selectores anidados, herencia, etcétera. También se añadieron componentes de material angular, una librería de

componentes desarrollada por Google, la cual permite tener botones, campos de entrada, entre otros componentes. A su vez también permite manejar de forma más fácil el responsive y la paleta de colores usada creando una interfaz moderna, intuitiva y agradable para el usuario.

4.4.7.3 Graficas

Para poder mostrar la información a través de gráficas, haciendo cómoda la visualización del usuario. Por ende, se optó por la librería Apex Charts que tiene un diseño bastante moderno, gratuito y además cuenta con una incorporación a Angular bastante buena. Su personalización y manejo es bastante amplio, además de contar con distintos tipos de gráficos y animaciones a la hora de hacer actualizaciones en estas. (Ver Figura 56)

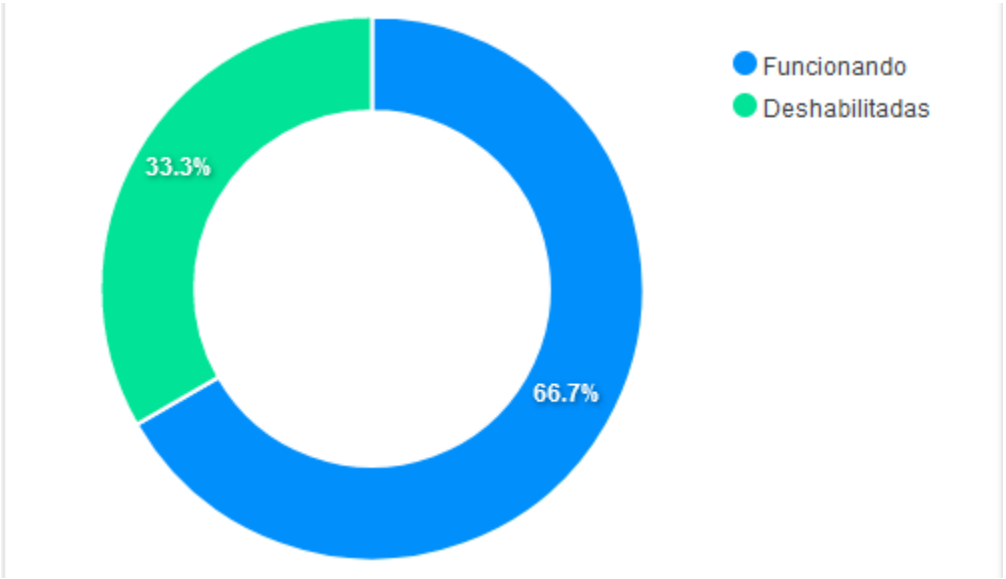


Figura 56. *Estado de estaciones*

Fuente: elaboración propia.

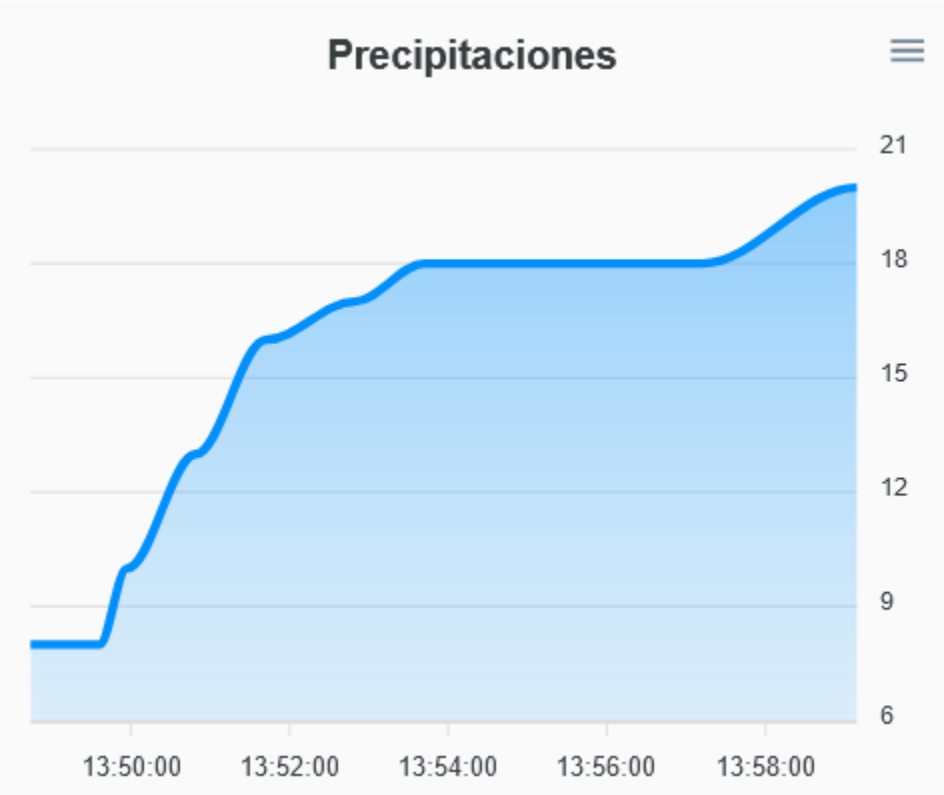


Figura 57. *Nivel de precipitaciones*

Fuente: elaboración propia.

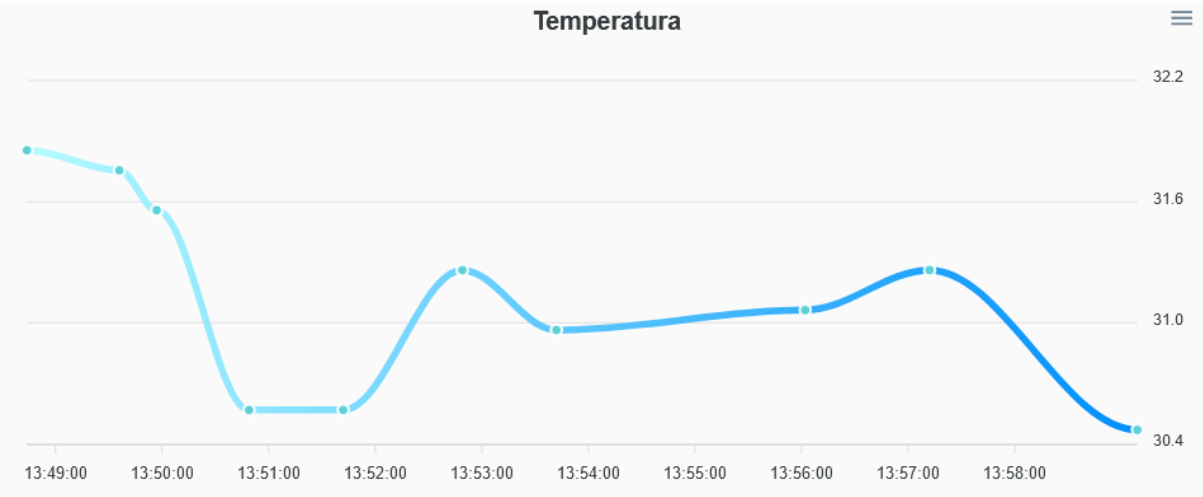


Figura 58. *Temperatura del agua*

Fuente: elaboración propia.

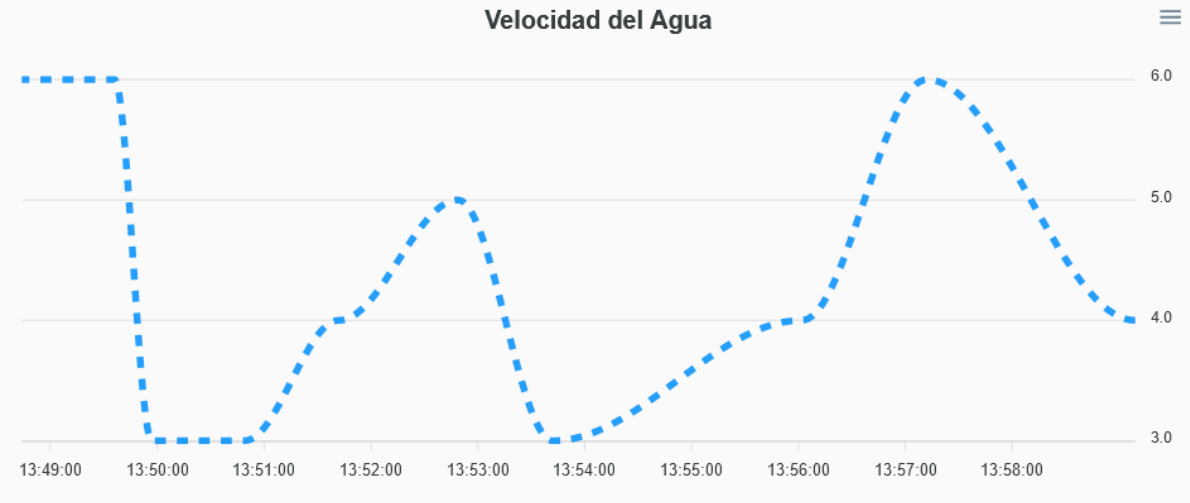


Figura 59. *Velocidad del agua*

Fuente: elaboración propia.

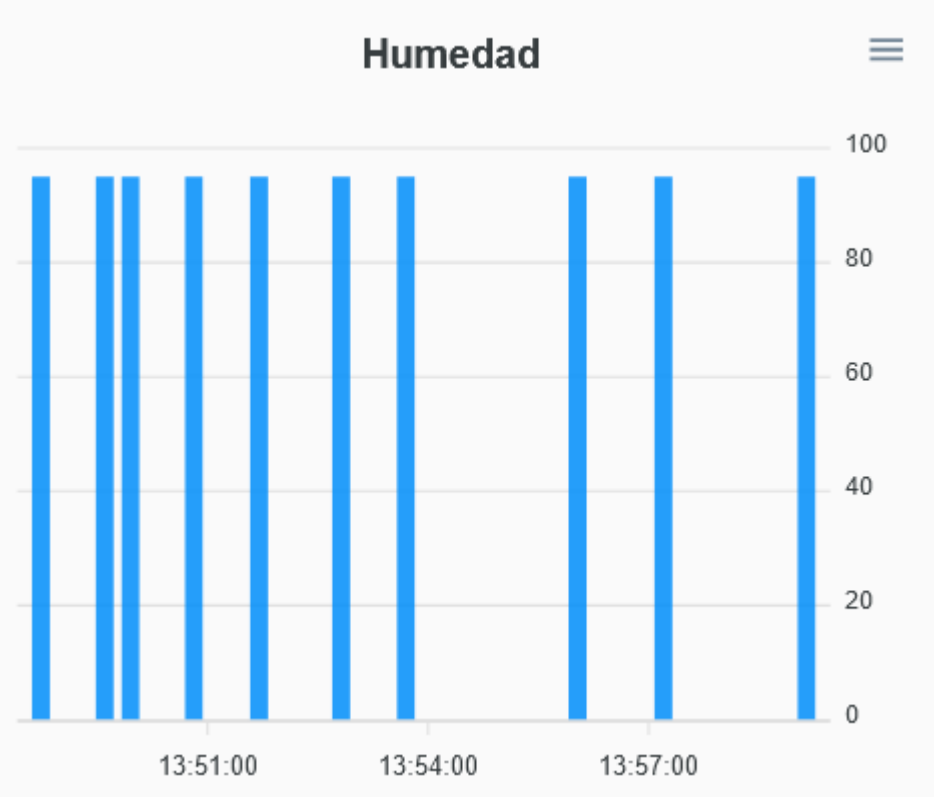


Figura 60. *Humedad*

Fuente: elaboración propia.

4.4.7.4 Servicios y subscripciones

En cuantos a los llamados del cliente a la API creada en el backend, se usó la librería por defecto de Angular ya que esta misma está basada en Express, a su vez el uso de esta es fácil e intuitivo por lo cual es la opción más usada a nivel mundial en cuanto a un proyecto basado en Angular se refiere. Su punto fuerte es la creación de una suscripción la cual conlleva a tener características propias de un elemento de RXJS, el cual se autodenomina como la extensión reactiva a Javascript. Al ser un servicio reactivo, este permite que se realicen funciones sin tener que refrescar o actualizar la página, lo cual permite trabajar de forma asíncrona.

4.4.7.4.1 Web Socket

Para manejar los datos en tiempo real, se requería el uso de un websocket, que permitiera obtener los datos cuando la estación enviará un dato al broker, por ellos se hace uso de la librería de MQTT, que tiene incorporada la función del websocket. (Ver Figura 61)

```
export const MQTT_SERVICE_OPTIONS: IMqttServiceOptions = {
  hostname: 'www.flowriver.online',
  connectTimeout: 4000,
  clientId: 'emqx',
  keepalive: 60,
  port: 8083,
  path: '/mqtt',
  clean: true,
};
```

Figura 61. *Configuración de la librería MQTT*

Fuente: elaboración propia.

```
this.subscription = this._mqttService
  .observe('flowriver/#')
  .subscribe((message: IMqttMessage) => {
    this.getData();
  });
```

Figura 62. *Uso de la función del websocket*

Fuente: elaboración propia.

4.4.8 Desarrollo del Hardware

Para el desarrollo del módulo ESP8266 se hizo uso del IDE Platformio a través del Framework de Arduino, utilizando las librerías de comunicación como lo fueron; <WiFi.h> encargada de gestionar la conexión vía WiFi del dispositivo, <ESP8266HTTPClient.h> encargada de gestionar las peticiones de tipo Post, Put, Get y Delete con los servicios de API Restfull, <ArduinoJson.h> encargada del manejo de ficheros tipo Json, <PubSubClient.h> la cual permite obtener una conexión por medio del protocolo MQTT con el ESP8266, además fue necesaria la librería <DHT.h> para el uso del sensor DHT-11.

Teniendo una vez desarrollado el código necesario y utilizando adecuadamente las librerías requeridas se carga el código a la placa por medio del puerto COM utilizando un cable microUSB. (Ver Figura 63)

```
void loop()
{
    if (!client.connected())
    {
        reconnect();
    }

    if (client.connected())
    {
        String str = "{'NivelAgua':" + String(LevelAgua())
        + ",'VelAgua':" + String(WaterFlow()) // L/min
        + ",'Temp':" + String(DHTTemperatura())
        + ",'Hum':" + String(DHTHumedad())
        + ",'Prec':" + String(LevelPrecipitaciones())
        + ",'Fecha':'" + Fecha() + "'}";
        str.toCharArray(msg, 125);
        client.publish(root_topic_publish, msg);
        nivelPrecaucion = NivelPrecaucion();
        nivelAlerta = Alerta();

        if (LevelAgua() > nivelAlerta){
        delay(9000);//Hay cada 10 segundos
        }
        else{
        delay(59000);//cada minuto
        }
    }

    client.loop();
}
```

Figura 63. *Porción del código utilizado*

Fuente: elaboración propia.

4.4.9 Incorporación del CI/CD

Para el fácil despliegue a producción fue incorporado al proyecto pipelines con el fin de actualizar automáticamente el backend y frontend.

4.4.9.1 CodeDeploy

Debido a que el servidor backend se encuentra alojado en una instancia EC2, era bastante útil usar otra herramienta proporcionada por Amazon Web Service que permitiera la creación de pipelines y ajustar el CI/CD a la aplicación de Node.js. Por ende, se optó por el uso de CodeDeploy que actúa como un *daemon* revisando en tiempo real los cambios en la rama *main* del repositorio del backend. Dado este evento, se procede a ejecutar los scripts elaborados en Bash. (Ver Figura 64)

```
OrlandoMR, 4 months ago | 1 author (OrlandoMR)
#!/bin/bash

#give permission for everything in the express-app directory
sudo chmod -R 777 /home/ubuntu/express-app

#navigate into our working directory where we have all our github files
cd /home/ubuntu/express-app

#add npm and node to path
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"  # loads nvm bash_completion (no

#install node modules
npm install

#start our node app in the background
node app.js > app.out.log 2> app.err.log < /dev/null &
```

Figura 64. Script de ejecución de la aplicación

Fuente: elaboración propia.

```
#!/bin/bash
#Stopping existing node servers
echo "Stopping any existing node servers"
sudo pkill node 2>/dev/null
```

Figura 65. *Script de finalización del servicio*

Fuente: elaboración propia.

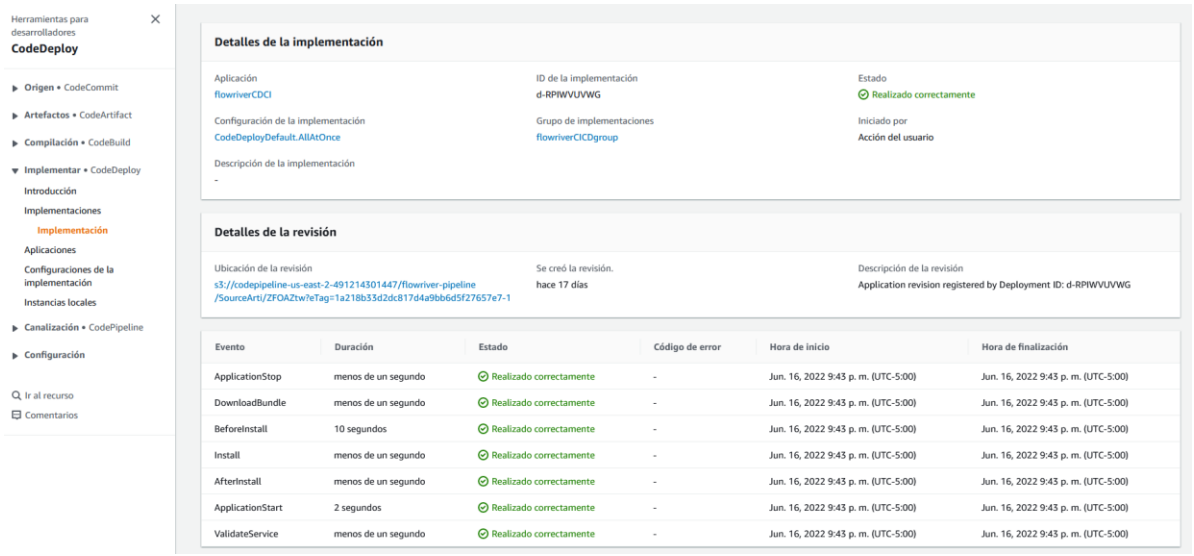


Figura 66. *CodeDeploy eventos*

Fuente: elaboración propia.

4.4.9.2 Github Actions

Para la incorporación de CI/CD en el proyecto de Frontend, al momento de alojar la aplicación web en Firebase, este servicio añade manualmente esta funcionalidad con solo dar acceso al repositorio en github, mediante scripts.

A través de la consola se fueron creando estos scripts según las opciones presentadas, tales como el manejo de una SPA (Single Page Application), o el acceso al repositorio.

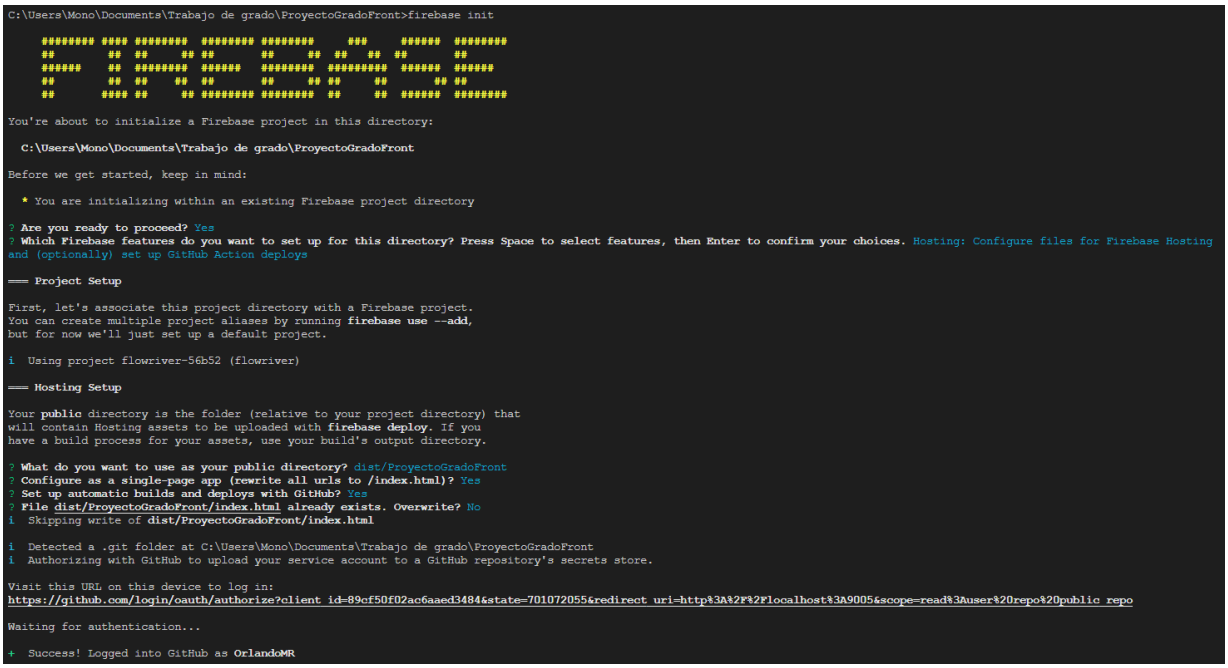


Figura 67. *Creación de los scripts mediante comandos*

Fuente: elaboración propia.

```yaml
# This file was auto-generated by the Firebase CLI
# https://github.com/firebase/firebase-tools

name: Deploy to Firebase Hosting on merge
'on':
  push:
    branches:
      - main
jobs:
  build_and_deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - run: npm ci && npm run build
      - uses: FirebaseExtended/action-hosting-deploy@v0
        with:
          repoToken: '${{ secrets.GITHUB_TOKEN }}'
          firebaseServiceAccount: '${{ secrets.FIREBASE_SERVICE_ACCOUNT_FLOWRIVER_56B52 }}'
          channelId: live
          projectId: flowriver-56b52
```

Figura 68. *Script creado para el flujo de CI/CD*

Fuente: elaboración propia.

5. Resultados

Basados en las fases propuestas inicialmente para la realización del proyecto, este se guio para la recopilación, análisis, comprensión, desarrollo y ejecución de información, los resultados obtenidos conforme a las actividades planteadas fueron: (Ver Tabla 14)

Tabla 14. *Formación Tecnológica y Levantamiento de Requerimientos*

Fase 1: Formación Tecnológica y Levantamiento de Requerimientos	
Investigación y análisis de los conceptos fundamentales.	Cumplida
Exploración de las tecnologías apropiadas para el desarrollo del prototipo	Cumplida
Indagación de proyectos similares aplicados en campo.	Cumplida
Búsqueda de herramientas relevantes en la realización del proyecto.	Cumplida
Definición de requerimientos del software y hardware.	Cumplida

Fuente: elaboración propia.

Tras el cumplimiento de las actividades de la Fase 1, se obtuvo información actualizada sobre el estado del arte y tecnologías aplicables al desarrollo del proyecto, también se realizó el planteamiento inicial de los requerimientos del prototipo. (Ver Tabla 15)

Tabla 15. *Análisis y Diseño de Arquitectura*

Fase 2: Análisis y Diseño de Arquitectura	
Estudio de la viabilidad y alcance del proyecto.	Cumplida
Definición y diseño de la plataforma de software	Cumplida
Selección de métricas del proyecto	Cumplida
Definición de la arquitectura	Cumplida

Fuente: elaboración propia.

Tras el cumplimiento de las actividades de la Fase 2, se plantearon los requerimientos para la arquitectura de IoT del proyecto. (Ver Tabla 16)

Tabla 16. *Desarrollo del Prototipo y la Plataforma*

Fase 3: Desarrollo del Prototipo y la Plataforma	
Diseño y desarrollo de la interfaz de usuario.	Cumplida
Implementación del esquema de sensores y componentes de hardware.	Cumplida
Construcción del prototipo a escala.	Cumplida
Diseño de la red de conectividad de los dispositivos y su conexión con la plataforma web.	Cumplida
Diseño e implementación de la base de datos.	Cumplida

Fuente: elaboración propia.

Tras el cumplimiento de las actividades de la Fase 3, se desarrolló la arquitectura tecnológica del sistema, la base de datos, la plataforma web y el prototipo funcional apto para validaciones con el sistema. (Ver Tabla 17 y Figura 69)

Figura 69. *Prototipo funcional*

Fuente: elaboración propia.

Tabla 17. *Evaluación de requerimientos del prototipo*

Fase 4: Evaluación de requerimientos del prototipo	
Almacenamiento de datos en la nube e interacción con ellos.	Cumplida
Verificación de la lógica e implementación.	Cumplida
Validación del sistema mediante la realización de pruebas (funcionales, unitarias, entre otras).	Cumplida
Análisis del resultado de las pruebas realizadas.	Cumplida

Fuente: elaboración propia.

Tras el cumplimiento de las actividades de la Fase 4, se realizó un informe de resultado correspondiente y aplicaron las correcciones pertinentes para iteraciones futuras. (Ver Figuras 70 y 71)

Figura 70. *Realización de pruebas del prototipo en campo 1*

Fuente: elaboración propia.

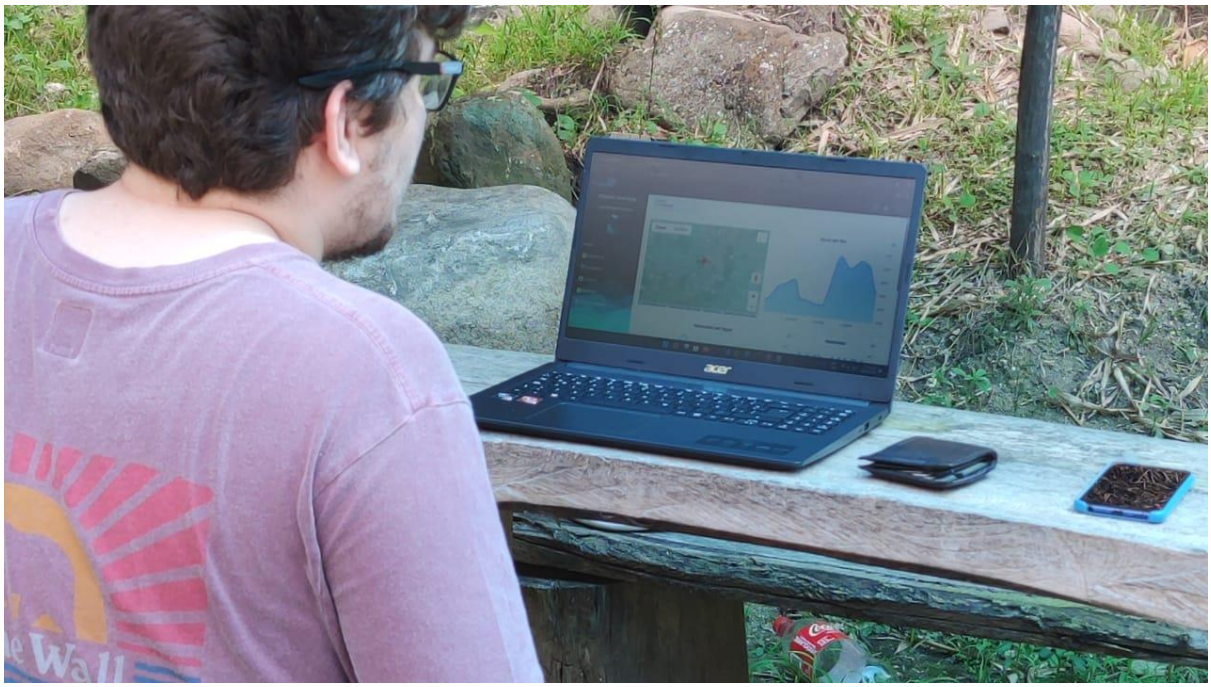


Figura 71. *Realización de pruebas del prototipo en campo 2*

Fuente: elaboración propia.

Tabla 18. *Entrega del Prototipo Final*

Fase 5: Entrega del Prototipo Final	
Entrega del prototipo final con los ajustes realizados.	Cumplida

Fuente: elaboración propia.

6. Conclusiones

Al terminar el proyecto titulado “diseño de una plataforma IOT para el monitoreo del nivel, velocidad y estado meteorológico en un cauce que permita dar alerta temprana de desastres en puntos estratégicos”, hemos encontrado favorables resultados a la hora de aplicar el prototipo, en el río Manco del municipio de Piedecuesta, Santander.

Teniendo en cuenta los objetivos de la investigación, obtuvimos lo siguiente: En primer lugar, se identificó las variables que jugarían un papel importante en la recolección de datos por parte de las estaciones. Teniendo en cuenta las necesidades iniciales presentadas y a través de la fase de análisis, se observó los posibles requerimientos que necesitaría la plataforma a la hora de hacer uso de esta. A través de esta dinámica repetitiva, se fueron añadiendo otros pocos requerimientos logrando así fortalecer la arquitectura de esta.

En segundo lugar, al tener el conocimiento básico de los principales requerimientos, se fue desglosando las herramientas a usar, para la creación de la aplicación con base a la capacitación obtenida y a los conocimientos previos, dando así visibilidad a los límites del proyecto, ya que se definieron unos parámetros al seleccionar las características pertinentes para la ejecución de este.

En tercer lugar, se desarrolló el proyecto de manera eficaz ya que se contó con una fase de definición sólida y concreta. En esta fase, además, se fue revisando el cumplimiento de los requerimientos establecidos mediante el plan de pruebas establecido, para así, observar falencias o dificultades que podrían desencadenar en el no cumplimiento de los requerimientos. Esto fue realizado variadamente para revisar si las correcciones satisfacían los objetivos.

En cuarta instancia, tras el análisis y definición de arquitectura se optó por limitar el presupuesto invertido para el hardware, teniendo en cuenta que la escala del proyecto así lo permitía, por lo cual se sacrificó parte de la precisión en las lecturas de datos para poder trabajar con sensores de costo, más adelante, por medio de las pruebas se concluyó que las lecturas de datos se encontraban dentro de lo deseado, satisfaciendo el objetivo del proyecto de dar una alerta temprana de desastres en puntos estratégicos.

Por último, se pudo verificar el cumplimiento de los requerimientos expuestos, ya que, el funcionamiento del prototipo y a su vez la plataforma, arrojó el desempeño esperado en cuanto a las características solicitadas, lo cual muestra, que el proyecto fue un éxito y que con ayuda de elementos más sofisticados permitan mejorar el rendimiento y calidad del producto en una posible versión llevada a escenarios más complejos.

7. Recomendaciones

En cuanto a la conectividad de las estaciones es posible que existan conflictos con la red WiFi móvil o cobertura brindada por el proveedor. Por ello como alternativa, es la implementación del módulo de SIM Card para enviar mensajes SMS a través del dispositivo (Arduino), añadiendo también alertas de zona como son manejadas en USA para los tornados inminentes en dicha área. Además de esto, sería de gran ayuda establecer conexión a internet mediante datos móviles.

Por parte de la plataforma existen numerosos features incorporables, tales como, una personalización amplia de usuario por parte de los técnicos ya que solo el administrador puede configurar y administrar los usuarios.

La creación de una aplicación móvil permitirá que el técnico instale las estaciones de forma más cómoda, ya que puede ser tedioso el manejo de la página web visto desde un celular o tablet.

Por último, la accesibilidad es indispensable en esta era por ende para incluir el acceso cómodo a la plataforma es útil contar con detalles como la visualización de palabras en lenguaje de señas mediante GIFs, la lectura de texto mediante un bot comúnmente denominado TextToSpeech, y el aumento y disminución del tamaño de la fuente.

Referencias

Alibaba. (2022). *Empower Your Business in USA & Canada with Alibaba Cloud's Cloud Products & Services*. https://us.alibabacloud.com/es

All Data Sheet. (2022). *HC-SR04 Datasheet (PDF) - List of Unclassifed Manufacturers*. https://www.alldatasheet.com/datasheet-pdf/pdf/1132203/ETC2/HC-SR04.html

Amazon Services. (2022). *Calculadora de EC2*. https://calculator.aws/#/addService

Amazon Web Service. (2022). *Elastic compute cloud (EC2) de capacidad modificable en la nube*. https://aws.amazon.com/es/ec2/

Anita, M. (2022). *PT100 RTD Temperature Sensor ：What is RTD sensor and PT100?* https://www.easybom.com/blog/a/pt100-rtd-temperature-sensor-datasheet-pinout-wiring-diagram

Arduino S.R.L. (2022). *Arduino Nano Every*. https://store-usa.arduino.cc/products/arduino-nano-every

Arduino, S.R.L. (2020). *Arduino Sketcvh on NodeMCU 32S, ESP32*. https://forum.arduino.cc/t/arduino-sketcvh-on-nodemcu-32s-esp32/656784

Back4App. (2021). *The Best Ten Backend Languages To Code*. https://blog.back4app.com/best-backend-language/

Celis, R. (2017). *¿Qué es Heroku? ¿Cómo funciona la plataforma y para qué sirve?* https://platzi.com/blog/que-es-heroku/

Chand, M. (2021). *Top 10 Cloud Service Providers In 2021*. https://www.c-sharpcorner.com/article/top-10-cloud-service-providers/

DataSheet. (2022). *PDF Pt100 (Hoja de datos)*. http://www.datasheet.es/PDF/900325/Pt100-pdf.html

Datasheet,. (2013). Ultrasonic Ranging Module HC - SR04. *Indoware*, 1-4.

Decentlab. (2022). *Ultrasonic distance / level sensor for LoRaWAN®*. https://www.decentlab.com/products/ultrasonic-distance-/-level-sensor-for-lorawan

Digi-Key Electronics. (2022). *Trican Engine Sensor HTD 2800*. https://www.digikey.com/catalog/en/partgroup/trican-engine-sensor-htd2800/87522

Emq Technologies Co., Ltd. (2022). *Descarga el repositorio EMQX*. https://www.emqx.io/downloads?os=Ubuntu

Erpinnews . (2018). *Computación en la niebla vs computación perimetral*. https://erpinnews.com/fog-computing-vs-edge-computing/

Esploradores. (2022). *Comparación de las placas NodeMCU* . https://www.esploradores.com/comparacion-de-placas-nodemcu_/

Fox, P. (2021). *Sensores*. https://es.khanacademy.org/computing/ap-computer-science-principles/x2d2f703b37b450a3:computing-innovations/x2d2f703b37b450a3:monitoring-innovations/a/sensor-types

GitHub. (2022). *Panel de gestión de actividades*. https://github.com/ProyectoGradoIOT/ProyectoGradoFront/issues

GitHub. (2022). *Repositorios*. https://github.com/ProyectoGradoIOT

Google. (2022). *Calculadora de Google Cloud*. https://azure.microsoft.com/en-us/pricing/calculator/

Google. (2022). *Google Cloud*. https://cloud.google.com/

Grokhotkov, I. (2022). *Biblioteca ESP8266 WiFi*. https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html

Herrera, F. (2021). *Angular: De cero a experto (Legacy)*. https://www.udemy.com/course/angular-2-fernando-herrera/

HyQuest Solutions Latin America. (2022). *TB3 - Pluviómetro* . https://www.directindustry.es/prod/kisters-hyquest-solutions/product-235669-2384212.html

Last Minute Engineers. (2022). *How DHT11 DHT22 Sensors Work & Interface With Arduino*. https://lastminuteengineers.com/dht11-dht22-arduino-tutorial/

Maloy, G. (2020). *¿Qué es un Sensor y Qué Hace?* https://dewesoft.com/es/daq/que-es-un-sensor

Microsoft . (2022). *Servicios de informática en la nube* . https://azure.microsoft.com/es-es/

Miguel, R. (2022). *El verdadero significado del término daemon*. https://blog.desdelinux.net/el-verdadero-significado-del-termino-daemon/

Mozilla. (2022). *JavaScript* . https://developer.mozilla.org/es/docs/Web/JavaScript

Ochoa, M. (2018). *¿Qué es Edge Computing y por qué es relevante para las empresas?* . https://www.itmastersmag.com/noticias-analisis/que-es-edge-computing-y-por-que-es-relevante-para-las-empresas/

Open Webinars. (2022). *¿Qué es Sass? ventajas, desventajas y ejemplos de desarrollo*. https://openwebinars.net/blog/que-es-sass-ventajas-desventajas-y-ejemplos-de-desarrollo/

Oracle. (2022). *Descubre la plataforma en la nube de próxima generación*. https://www.oracle.com/co/cloud/

Portillo, G. (2021). *Pluviómetro*. https://www.meteorologiaenred.com/pluviometro.html

Raspberry Pi. (2022). *Raspberry Pi Pico – Raspberry Pi*. https://www.raspberrypi.com/products/raspberry-pi-pico/

Red Hat. (2022). *¿Qué es el Internet de las cosas (IoT)?* https://www.redhat.com/es/topics/internet-of-things/what-is-iot

Rika Sensor. (2019). *Sensor RK400-01*. https://www.rikasensor.com/rk400-01-tipping-bucket-rainfall-sensor-accurate-rain-gauge.html

Udemy . (2021). *Angular: De cero a experto (Legacy)*. https://www.udemy.com/course/angular-2-fernando-herrera/

Udemy. (2022). *IoT Bootcamp! Nuxt - Node - Mongo - Emqx - ¡Más de 60 horas!* . https://www.udemy.com/course/iot-god-level/

Udemy. (2022). *Node: De cero a experto*. https://www.udemy.com/course/node-de-cero-a-experto/