UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE CUENCA
CARRERA DE INGENIERÍA DE SISTEMAS

INTEGRACIÓN DE CHIRPSTACK CON NODE RED PARA ALOJAR DATOS EN GOOGLE CLOUD PLATFORM

Trabajo de titulación previo a la obtención del
Título de Ingeniero de Sistemas

AUTORES:

ALEX CRISTOPHER CUJI TORRES
DAVID OSWALDO SARUMEÑO AVILA

TUTOR:

ING. ERWIN JAIRO SACOTO CABRERA, PhD.

Cuenca - Ecuador
2022

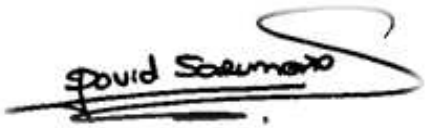
CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Alex Cristopher Cuji Torres con documento de identificación N° 0105315113 y David Oswaldo Sarumeño Avila con documento de identificación N° 0705180016; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Cuenca, 22 de julio del 2022

Atentamente,



Alex Cristopher Cuji Torres	David Oswaldo Sarumeño Avila
0105315113	0705180016

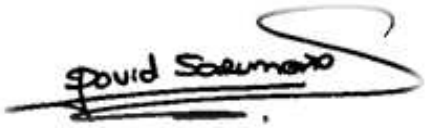
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Alex Cristopher Cuji Torres con documento de identificación N° 0105315113 y David Oswaldo Sarumeño Avila con documento de identificación N° 0705180016, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto Técnico: "Integración de ChirpStack con Node Red para alojar datos en Google Cloud Platform", el cual ha sido desarrollado para optar por el título de: Ingeniero de Sistemas, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, 22 de julio del 2022

Atentamente,



Alex Cristopher Cuji Torres	David Oswaldo Sarumeño Avila
0105315113	0705180016

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Erwin Jairo Sacoto Cabrera con documento de identificación N° 0301185229, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: INTEGRACIÓN DE CHIRPSTACK CON NODE RED PARA ALOJAR DATOS EN GOOGLE CLOUD PLATFORM, realizado por Alex Cristopher Cuji Torres y David Oswaldo Sarumeño Avila, obteniendo cómo resultado final el trabajo de titulación bajo la opción Proyecto Técnico que cumple con todos los requerimientos determinados por la Universidad Politécnica Salesiana.

Cuenca, 22 de julio del 2022

Atentamente,

ERWIN JAIRO SACOTO CABRERA

Firmado digitalmente por ERWIN JAIRO SACOTO CABRERA
Fecha: 2022.10.12 16:49:03 -05'00'

Ing. Erwin Jairo Sacoto Cabrera, PhD.
0301185229

DEDICATORIA

A mi Madre Primeramente, a Dios, por darme salud y permitirme cumplir una meta más en mi vida. A mis padres que fueron un gran apoyo durante todo este proceso, su motivación, sus ánimos y gran esfuerzo para poder terminar mi carrera universitaria. Mis hermanos, que con sus consejos supieron guiarme a donde quería llegar y no rendirme en este proceso.
A nuestro director de tesis al PhD. Erwin Sacoto, que con su ayuda, dedicación y conocimientos brindados se pudo realizar este proyecto de la mejor manera.

Alex Cristopher Cuji Torres

A Dios, por darme salud y fuerza para culminar una etapa mas en mi vida. A mis padres Betty Avila. & Oswaldo Sarumeño., por brindarme su apoyo incondicional, por ser mis pilares fundamental para este trayecto de mi vida.
Mi Hermana Karen Sarumeño; por su compañía en este trayecto de mis estudios y por su apoyo para no rendirme.
Saverlia Cazorla. por estar en todo momento de mi vida, por sus consejos y amor.
En memoria a José Avila. & Juana Avila., por ser mi guía, mi ángel de la guarda.
Gracias a todos mis Familiares y Amigos quienes estuvieron en mi lapso de estudios, por sus consejos, por su aprecio que me han brindado, este logro es hacia ustedes.
A nuestro director de tesis al PhD. Erwin Sacoto, que con su ayuda, dedicación y conocimientos brindados se pudo realizar este proyecto de la mejor manera.

David Oswaldo Sarumeño Avila

Resumen

El Internet de las Cosas (IoT) facilita las actividades diarias, ya que la mayoría de cosas que están a nuestro alrededor se encuentran conectadas a Internet y son administradas mediante plataformas dedicadas a esta tecnología. Una de las plataformas para administrar redes de IoT es ChirpStack, el cual es creado para administrar dispositivos inteligentes, es decir dispositivos que se conectan de manera inalámbrica por medio de la tecnología LoRa, como es el caso de medidores de agua inteligentes, estos dispositivos son capaces de medir el agua y enviar datos a través del Internet a un servidor. Una vez ahí se observarán estos datos, el servidor es capaz de conectarse a dispositivos mediante un gateway LoRa, el cual establece una comunicación entre estos dos.

La ventaja de esta tecnología es la escalabilidad de adaptación con otros sistemas, como lo es Node Red, esta plataforma incluye muchas APIs como ChirpStack, PostgreSQL, GCP, etc. Aquí se podrá realizar gráficas de los datos de consumo del agua ingresados en el servidor ChirpStack, en la que los nodos de PostgreSQL será de gran ayuda para esto, ya que en el servidor se crea una base de datos la cual guardara todos los datos que se reciba por parte de los dispositivos. Mediante una API de dashboard de Node Red y con los nodos de PostgreSQL se podrá realizar consultas a la base de datos y graficarlas en el nodo de dashboard y así poder visualizar los datos de mejor manera en forma de gráficas.

Palabras clave: Servidor, LoRaWAN, ChirpStack, Node-Red, Gateway, IoT, Medidores ultrasónicos, Raspberry Pi, bomba de agua, base de datos, PostgreSQL, dashboard, APIs.

Abstract

The Internet of Things (IoT) facilitates daily activities, since most of the things that are around us are connected to the Internet and are managed through platforms dedicated to this technology. One of the platforms to manage IoT networks is ChirpStack, which is created to manage smart devices, i.e. devices that connect wirelessly through LoRa technology, such as smart water meters, these devices are able to measure water and send data over the Internet to a server. Once this data is observed, the server is able to connect to devices through a LoRa gateway, which establishes a communication between these two.

The advantage of this technology is the scalability of adaptation with other systems, such as Node Red, this platform includes many APIs like ChirpStack, PostgreSQL, GCP, etc. Here it will be possible to make graphs of the water consumption data entered in the ChirpStack server, in which the PostgreSQL nodes will be of great help for this, since a database is created in the server which will store all the data received by the devices. By means of a dashboard API of Node Red and with the PostgreSQL nodes it will be possible to make queries to the database and to graph them in the dashboard node and thus to be able to visualize the data in a better way in the form of graphs.

Keywords: Server, LoRaWAN, ChirpStack, Node-Network, Gateway, IoT, Ultrasonic meters, Raspberry Pi, water pump, database, PostgreSQL, dashboard, APIs

ÍNDICE

Índice de tablas

Índice de figuras

Capítulo I

INTRODUCCIÓN

Según (12; 13), el protocolo de LoRaWAN es una red WAN(Red de área amplia) de baja potencia que se basa en una tecnología LPWAN. Esta tecnología de Internet de las cosas, es un avance grande en la tecnología, la cual permite facilitando la vida de las personas, en donde los dispositivos informáticos inteligentes se transmiten e intercambian información inalámbricamente por medio de Internet, con la finalidad de administrar, monitorear, visualizar procesos que ocurren en estos.

Estos datos que se transmiten entre dispositivos o dispositivos - servidor, pueden ser administrados en plataformas dedicadas a esta tecnología LoRa, por ejemplo el consumo de agua, existen medidores de agua inteligentes capaces de trasmitir información en la red, estos datos se empezaran a mostrar en un servidor como ChirpStack en la que se podrán administrar los dispositivos que estén conectados a la red y visualizar los datos transmitidos.

En cuanto al consumo de agua para tener un mejor control de estos datos es necesario trabajar con empresas que tengan el control del servicio de agua potable en la ciudad como Etapa EP, ya que sé encarga del mantenimiento y operación de los sistemas de agua potable. Sin ellos no sería posible hacer este proyecto, ya que es necesario contar con alta tecnología para implementar una red LoRa dentro de la ciudad el cual ayudara tanto a los

ciudadanos como a la empresa a tener un mejor control del consumo de agua potable que ofrece Etapa EP.

Existen muchas aplicaciones dedicadas a esta tecnología, una de ellas es Node Red, en la que se puede realizar comunicación inalámbrica a servidores de red LoRa, esto mediante protocolos de MQTT, HTTP, HTTPS, etc. También existen muchas APIs de integración a la plataforma, en las cuales se tiene ChirpStack, Google Cloud Platform, InfluxDB, MySQL, entre otras. Una API muy utilizada es de las bases de datos en donde se pueden realizar consultas para recibir información de datos dispositivos administrados por un servidor, además consta de una API de gráficas para realizar dashboard, esto es muy importante para tener un control sobre los datos transmitidos por la red LoRaWAN. Estos datos pueden actualizarse en un tiempo predeterminado, es decir pueden graficarse datos en tiempo real según como transmitan se grafique o simplemente grafícalas durante periodos de tiempo designados por el usuario.

Capítulo II

PROBLEMA

2.1 Antecedentes

En (14) Etapa EP, es la primera empresa creada por la municipalidad para prestar servicios públicos a la ciudad entre ellas el consumo de agua potable, se instaló en el mercado desde el año 1968 y que hasta la fecha ya ha brindado asistencia técnica a muchos municipios y empresas en todo el país. En la ciudad de Cuenca se ofrece este servicio en zonas rurales y urbanas para que todas las personas cuenten con el agua potable en los hogares. Al ser una empresa tan grande se pueden hacer muchos proyectos con respecto a IoT y así aprovechar todo lo que nos ofrece esta tecnología en combinación con lo que ofrece Etapa EP. Si un proyecto así se diera ya no sería necesario personas que revisen consumo de agua de medidores en los hogares, sino con medidores inteligentes, todos los datos de consumo de agua en el hogar estarían disponibles en la red en donde se pueden administrar y controlar mejor estos datos.

En los últimos años, el Internet de las cosas (IoT), es una tecnología que está tomando más fuerza en el desarrollo en la sociedad, que permite múltiples funciones asociadas al confort de las personas la vida diaria, en desarrollo de las ciudades y en los procesos industriales como se menciona en (4; 15). Esta propuesta de trabajo de titulación tiene como

principal objetivo proponer una solución de IoT, a través de la implementación de una plataforma LoRaWAN, utilizando herramientas como un servidor ChirpStack, que permite administrar un sin número gateways y dispositivos LoRaWAN, Node Red que permitirá enviar los datos obtenidos a un servidor en la Nube como Google Cloud Platform o InfluxDB, para el almacenamiento, tratamiento y visualización. Esta propuesta permitirá resolver problemas existentes, en relación con el almacenamiento, procesamiento y visualización de datos para redes privadas LoRaWAN.

2.2 Importancia y Alcances

En la actualidad se han desplegado varias redes LoRaWAN, que mediante un servidor se pueden realizar la administración de dispositivos LoRA, como lo son los medidores de agua inteligentes, para ello se tiene ChirpStack, que es un servidor donde se puede administrar estos dispositivos, esta tecnología es muy innovadora en la actualidad, ya que no es necesario esperar planillas de consumo de agua, sino directamente poder observarlos mediante Internet, y así saber el consumo que tengamos diariamente y mensualmente, lo que sería un proyecto innovador para una mejor comodidad en ese sentido en los hogares de cada familia.

El alcance de este proyecto es facilitar el control de consumo de agua en los hogares, teniendo una visión de cuanta de agua consumen en el día o mensualmente y no tener que esperar una plantilla mensual en donde se detallan todos esos valores, como también podemos tener el control de consumo, ya que mientras más consumo haya más se deberá pagar a empresas que prestan este servicio.

2.3 Delimitación

Lo que se busca es definir, diseñar, desarrollar e implementar una red LoRaWAN, que permita el análisis de consumo de agua en los hogares. Los beneficiarios será la empresa de Etapa EP, porque tendrán un sistema en donde podrán administrar todos los datos que se envíe desde los dispositivos finales, todo esto mediante un servidor el cual realizara distintas tareas para dar un servicio de calidad a sus clientes, otro beneficiado en la implementación de este sistema son los usuarios, ya que su implementación sería sencilla, solo se necesitaría un medidor de agua ultrasónico, el cual se conectara a la red LoRa y este empezara a enviar datos del consumo de agua de los clientes, Etapa EP tendrá un sistema para poder analizar el consumo de agua en los hogares, pudiendo así mejorar el mal uso de agua si no usar lo suficiente, ya que un mal uso sería un gasto adicional, esta plataforma también puede ampliarse para otros tipos de dispositivos que usen esta tecnología y así poder seguir mejorando la calidad de vida de las personas.

Capítulo III

OBJETIVOS GENERALES Y ESPECÍFICOS

3.1 Objetivo General

Integración de ChirpStack con Node Red para alojar datos en Google Cloud o InfluxDB.

3.2 Objetivos Específicos

- Conocer e investigar los fundamentos de LoRaWAN. Fundamentos de servidores como ChirpStack, el uso y aplicación Node Red y Google Cloud Platform.
- Implementar un servidor LoRaWAN.
- Implementar una plataforma para la administración, gestión y visualización de datos utilizando Node Red, Google Cloud Platform o InfluxDB.
- Diseñar e implementar un plan de prueba, para posteriormente poder verificar el acoplamiento del servidor con el sistema y su correcto funcionamiento.

Capítulo IV

REVISIÓN DE LA LITERATURA O FUNDAMENTOS TEÓRICOS

En (16; 3), se define como Internet de las Cosas (IoT), a todo dispositivos que interactúan por medio del Internet, ofrece muchas oportunidades a nivel comercial, que permiten a empresas o negocios crear nuevas estrategias de comercio.

Al respecto, en (1; 17), se indica que esta tecnología permite detectar y transmitir datos en tiempo real, permite cálculos rápidos y toma de decisiones óptimas.

Su principal objetivo es la conectividad entre dispositivos físicos, por ejemplo el controlar el consumo de energía, consumo de agua, cámaras de seguridad, entre muchas otras aplicaciones con la utilización del Internet.

Para la planificación de IoT tal y como se indica (1; 18; 19; 18) el primer paso es la selección de componentes que conformaran la tecnología IoT los cuales pueden ser componentes físicos, luego un protocolo de comunicación de estos dispositivos, el almacenamiento de datos que se translimitan entre dispositivos, y por último el cálculo de estos, como se ve en la Fig. 1.

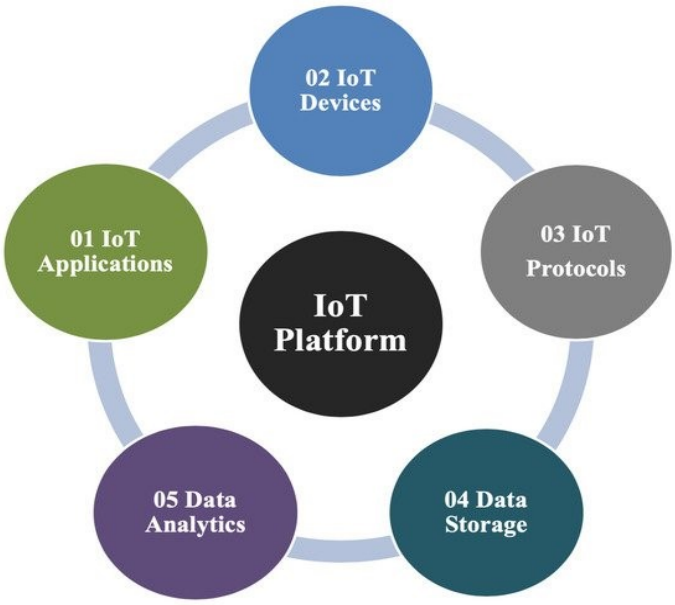


Figura 1. *Componentes de IoT (1).*

4.1 Requisitos de IoT

Para la implementación de IoT, existen muchos problemas que deben abordarse para una eficiente implementación de esta tecnología, según (2) algunas de ellas son falta de arquitectura y protocolos estándar, seguridad y privacidad heterogeneidad de dispositivos, escalabilidad, eficiencia, interoperabilidad y gestión de datos, como se aprecia en la Fig. 2.

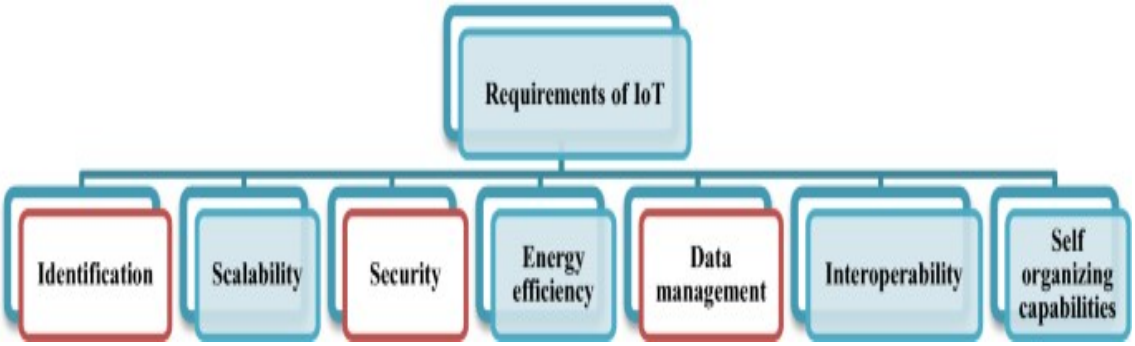


Figura 2. *Requisitos de IoT (2; 3).*

- ***Identificación y escalabilidad:*** Surgen problemas de escalabilidad por la cantidad

de objetos o dispositivos que se interconectan y que están divididos en diferentes niveles como:

1. *Asignación de nombres y direcciones:* el espacio de direcciones debe ser la indicada para la conexión de una gran cantidad de dispositivos.
2. *Comunicación de datos:* gran cantidad de dispositivos interconectados, lo que surge la escalabilidad para la comunicación de datos.
3. *Gestión de la información:* gran cantidad de datos generados, utilización de mecanismos indicados para la extracción útil de datos y almacenamiento.
4. *Aprovisionamiento y gestión de servicios:* gran cantidad de servicio disponibles para IoT.

- ***Capacidad de auto organización:*** A diferencia de computadores, donde el ser humano debe realizar su respectiva configuración, los dispositivos inteligentes deben organizarse, configurarse y adaptarse a situaciones sin necesidad de la intervención de un supervisor. La inteligencia distribuida en IoT, permite que los dispositivos inteligentes reaccionen de diferentes maneras ante cualquier situación que se presente, sin necesidad de la intervención de un usuario supervisor.

- ***Interoperabilidad:*** Para facilitar la comunicación entre dispositivos que se comportan de distinta manera, se requiere un estándar común para su comunicación.

- ***Seguridad y privacidad:*** La seguridad es un requisito fundamental en IoT, debe ser considerada una prioridad en la arquitectura de IoT. Los algoritmos de seguridad deben ser simples y debe haber un número mínimo de intercambio de información debido al ancho de banda limitado.

- ***Eficiencia energética:*** La recolección, conservación y consumo de energía es un problema que debe abordarse en IoT, ya que los dispositivos tienen una energía de

batería limitada, existen técnicas de uso de energía en dispositivos como:

1. *Técnicas de recolección de energía* Materiales piezoeléctricos, termoeléctricos, micro paneles solares.
2. *Técnicas de conservación de energía:* Baterías, pilas de combustible, baterías de polímero.

4.2 Arquitectura de IoT

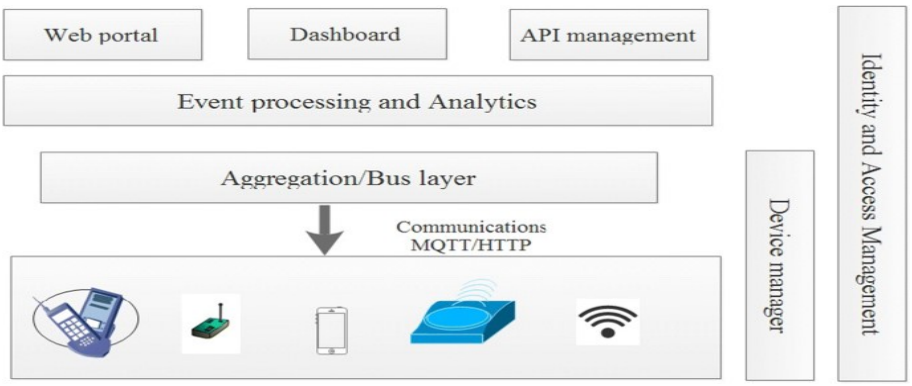


Figura 3. *Arquitectura de IoT (4).*

La arquitectura de IoT, como se observa en la Fig. 3 y se describe en (4; 20), está conformada por:

- ***Capa de dispositivos:*** Capa inferior, representa a dispositivos como celulares, sensores, Bluetooth, que se utilizan en aplicaciones IoT conectados a Internet para su comunicación.
- ***Capa de comunicación:*** Esta capa es la que soporta la conectividad entre dispositivos mediante protocolos de comunicación como Message Queuing Telemetry

Transport (MQTT), Constrained Application Protocol (COAP), etc.

- ***Capa de agregación/bus:*** Esta capa es la responsable de que los protocolos de comunicación como HTTP (Hypertext Transfer Protocol) , MQTT (Message Queuing Telemetry Transport), se conecten a los dispositivos y combina las comunicaciones de diferentes dispositivos para enrutar a las distintas puertas de enlace de comunicación. También es capaz de establecer puentes entre distintos protocolos.

- ***Capa de análisis y procesamiento de eventos:*** Recopila los eventos de la capa de bus y realiza acciones basadas en eventos.

- ***Capa de comunicaciones externas:*** Proporciona interacciones con los sistemas fuera de la red con comunicaciones M2M mediante API.

4.3 Redes LPWAN

Como se menciona en (21), las redes ***Low Power Wide Area Networks*** (LPWAN) están enfocadas para el manejo de redes inalámbricas de una red de área amplia, su principal característica de esta red es que utiliza poca potencia en comparación de la red ***Red de Área Amplia (WAN)*** que están existente, esta puede soportar cantidades de nodos para una sola área.

Esta red LPWAN tiene varias soluciones para la comunicación inalámbrica con los dispositivos IoT, en la actualidad ha existido un sin número de demandas para estos dispositivos especialmente porque cumple un papel fundamental en el bajo consumo de energía, tiene un largo alcance, y la tasa de datos es baja. Se maneja por diferentes tecnologías como lo indica en (5), las más utilizadas son:

- SigFOx
- LoRa/LoRaWan
- NB-IoT

4.3.1 Característica de la Redes LPWAN

La comunicación inalámbrica de las redes LPWAN, como se menciona en (5), últimamente ha tenido una evolución incomparable porque permite la comunicación mediante ***Bluetooth y Wifi***, tiene una mejor transmisión de datos, pero a un corto alcance, existe la posibilidad de mejorar la tasa de datos para un mayor alcance, lo que le impediría es el uso de la alta energía para la transmisión de los datos que realiza los dispositivos finales, para esto se necesitaría la redes de Tercera Generación de Telefonía Móvil (3G), Cuarta Generación (4G), Quinta Generación (5G).

- ***Largo Alcance:***

  Según (5), estas redes LPWAN permite la comunicación de una área geográfica amplia, que tiene como un máximo de unos 10km de alcance, que permitiría cubrir zonas rurales, suburbanas o urbanas. Para la transmisión de estas redes se necesitaría una frecuencia de un radioenlace y una modulación para la transmisión de la información

- ***Bajo Costo y Baja Potencia:***

  Según en (5), el uso de las bandas de frecuencias ISM (Industrial, científica y médica), reduce el costo de operación, por lo que es evitable generar costo extremadamente alto en licencias. Existen otros operadores que requieren una licencia para el uso de la misma como SigFOX, que no es de uso libre como utiliza las redes de LoRaWAN. Con relación a la baja potencia es necesario mantener un costo bajo

en su mantenimiento a próximos proyecto, por el motivo de que existen baterías que tiene de duración un lapso de 10 años o más, para la comunicación de estas redes LPWAN se necesita una topología estrella para que la transmisión de los datos no necesite pasar por otros terminales, como por ejemplo la topología malla, también existe varias alternativas para la transmisión de los datos como un ***Duty Cycling***, cuando la información se transmitirá, se utilizara un ***uplink***, que es la subida de la información hacia el ***gateway***, o un ***downlink***, que es la bajada de la información al ***gateway***, como lo menciona en (5).

4.3.2 SigFOX

Según el documento (22), SigFOX es una red que permite tener conectividad bidireccional global con respecto al Internet de las cosas, tiene como objetivo conectar a todo el mundo.

Esta red tiene varios factores importante que va de la mano con esta tecnología para el uso, y depende del lugar donde se desee emplearlo.

SigFOX va de la mano con la red LPWAN, que conjuntamente ha ganado popularidad dando servicio de cobertura, su principal objetivo es crear redes que permita la comunicación mundial para el Internet de las Cosas(IoT), de manera que existan dispositivos que se conecten fácilmente a la redes.

Actualmente, está desplegado en 65 países con el objetivo de buscar mensajes pequeños y tener una mayor cobertura a nivel mundial.

- ***Bajo Coste:***

  Tiene como prioridad conectar varios volúmenes de cualquier tipo de objeto.

- ***Bajo consumo de energía:***

  Minimiza el impacto ambiental, así como da una larga duración en la batería para reducir costes del mantenimiento.

- ***Facilidad de uso:***

  Tiene integración con varios objetos.

- ***Largo alcance:***

  Puede llegar a tener conectado a todos los objetos, así reducir el coste con respecto a la infraestructura locales.

- ***Manejo:***

  Facilita tener un mejor servicio de monitoreo y a su vez gestionar de mejor manera a los objetos.

- ***Identificación del abonado:***

  Tiene una mejor gestión con respecto a las tarjetas Subscriber Identity Module (SIM), y permite que el costo sea demasiado elevado o exista un adicional.

- ***Profundidad:***

  Permite llegar la comunicación a diferentes lugares, y tener una mejor conectividad con respecto a estructuras estrictas.

Según en (5), actualmente, utiliza un UNB (Ultra NarrowBand), que dependerá de las regiones que se encuentre, comienza desde los 100 Hz hasta los 600 Hz, puede tener una alta tasa de transmisión de 100 a 600 bps respectivamente. Tiene una capacidad para aprovechar todo el espectro dándole una mejor utilidad de aquello.

SigFOX utiliza modulación, también es conocida como Phase Reversal Keying (PRK), que permite tener una robustez antes del ruido y sensibilidad, dependiendo de la longitud del alcance, como se ilustra en la Fig.4.

Figura 4. *Arquitectura SigFOX (5).*

Los mensajes ***uplink*** de SigFOX está conformado por una cabecera que está con datos de capa física como MAC del dispositivo como se puede observar en la Fig.5.

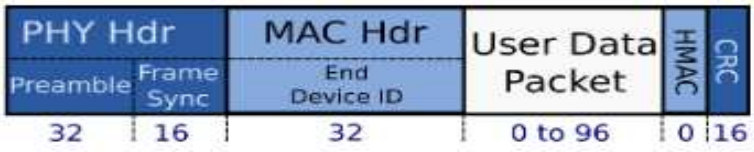


Figura 5. *Estructura de un paquete en SigFOX (5).*

4.3.3 NB-IoT

Según (23), se refiere a una tecnología de IoT de banda estrecha creada por 3GPP(3rd Generation Partnership Project), basado en funcionalidades LTE existentes, en (6) indica que NB-IoT puede ocupar un canal GSM(Global System For Mobile Comunication) de 200 kHz, así mismo este incluye dispositivos de bajo costos, alta cobertura como 20 dB más que GPRS(General Packet Radio Service), duración de 10 años más de batería y capacidad masiva.

NB-IoT cuenta con tres modos de operación:

- Independiente como operador dedicado.
- En banda dentro del ancho de banda ocupado de un operador LTE de banda ancha.

- Dentro de la banda de protección de un LTE existente transportador.

- ***Rendimiento de NB-IoT***

En (6) indica un análisis de rendimiento de NB-IoT, tanto en cobertura, batería, latencia y capacidad, el rendimiento será similar a LTE, lo que varía a esta es la operación de banda de guarda no requerirá que los símbolos OFDM(Multiplexación por División de Frecuencias Ortogonales), se reserven para la región de control heredada. Como se observa en la Fig.6.

Parameter	Stand-alone	In-band
System bandwidth	200 kHz	10 MHz
Frequency band	900 MHz	
eNB transmit power for NB-IoT	43 dBm	35 dBm
MS transmit power	23 dBm	
Propagation channel	TU	
Doppler spread	1 Hz	
Antenna configuration	DL: eNB: 1Tx, MS: 1Rx	DL: eNB: 2Tx, MS: 1Rx
	UL: eNB: 2Rx, MS: 1Tx	
Frequency error	Random from [-50, +50] Hz	
Thermal noise density	-174 dBm/Hz	
eNB receiver noise figure	3 dB	
UE receiver noise figure	5 dB	
Interference margin	0 dB	
Receiver processing gain	0 dB	

Figura 6. *Análisis de rendimiento de NB-IoT (6).*

4.3.4 LoRaWAN

LoRa es la capa física de la red LPWAN se conoce como LoRaWAN, fue creada por la empresa SEMTECH [1] que los primeros pasos fue la creación del chip de LoRa, luego de varios estudios se creó el dispositivo de LoRaWAN que es una capa de acceso al medio que fue desarrollada por una entidad llamada como LoRa Alliance según como lo menciona (7; 24), este chip se ha implementado en diferentes servicios como: transporte, público, en estructura y presta para diferentes campos (25; 26).

El chip LoRa es la capa física de la red LoRaWAN, esta red a nivel mundial está limitada por diferentes canales, en Europa utiliza 10 canales, que no tiene límite de tiempo para la permanencia de canal, pero en el uso de trabajo tiene restricciones, así mismo utiliza el mismo canales para realizar enlaces ascendentes y descendentes, además es utilizada en Norte América que tiene 64 canales, así mismo tiene restricciones en el trabajo y no tiene límite de permanencia en el canal (27).

La red LoRa (7) está conformado por:

- **Dispositivos Finales:**
  - Son dispositivos que se utiliza para realizar una conexión con los objetos de la LoRa.
  - Recolecta información concreta y adecuada, lo transmite en la pasarela.
- **Pasarelas:**
  - Cumple un papel primordial que recibe lo que los dispositivos finales enviaron y lo reenvía a los servidores de la red.

[1] `https://www.semtech.com/`

- **Servidores de Red:**
  - Se encarga de tener una recepción de los datos que ha sido enviado por los dispositivos finales y realiza un procesamiento de los datos.
  - Luego realiza la configuración respectiva de la red y de los dispositivos finales.

4.3.5 Capas de LoRaWAN

- **Capa Física:**
  - Utiliza un dispositivo LoRa que permite la comunicación de largo alcance y de potencia baja.
  - Se necesita emplear un factor de spreading factor (SF), que busca la manera de implementar mecanismo para tener una robustez y velocidad de transmisión.
  - SF permite realizar modificación en el número de canales ortogonales y realizar transmisiones sin ningún tipo de colisión.
  - Utiliza banda ISM inferiores de GHz.
- **Capa MAC:**
  - Permite gestionar el acceso al medio, en los canales de transmisión y la configuración de conexión como lo indica en (7).
  - Existen 3 tipos de clases: Clase A, Clase B, Clase C como se ve en la siguiente Fig.7.

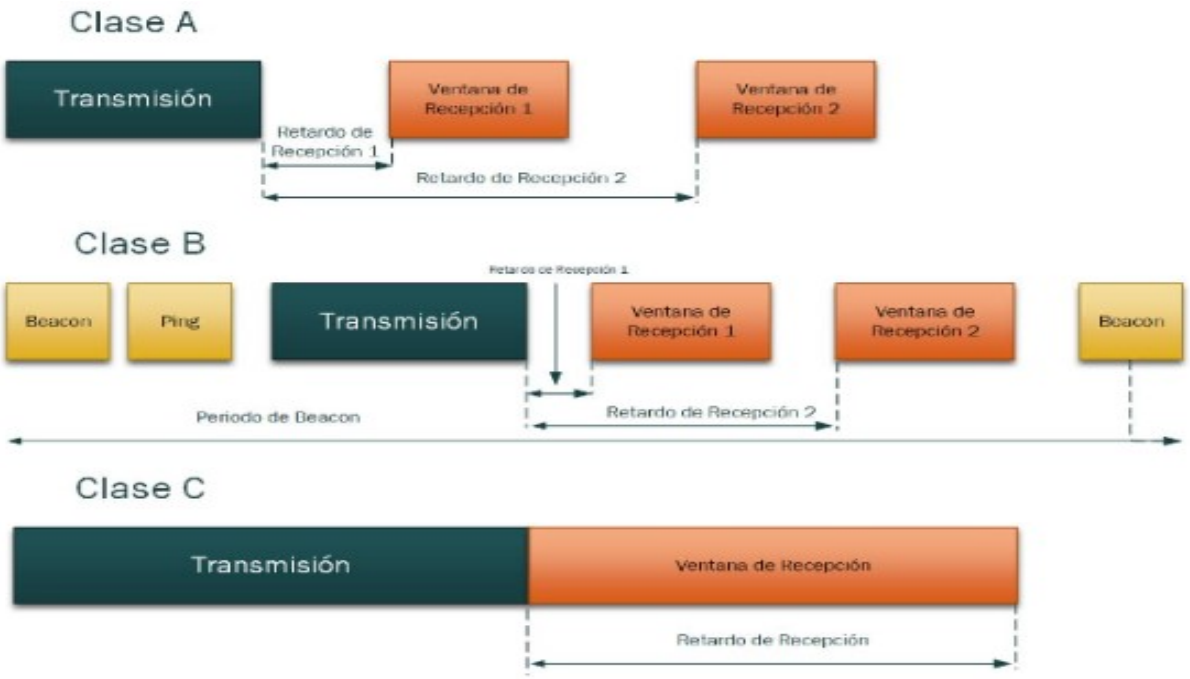


Figura 7. *Clases de nodos (7).*

4.3.6 Tipos de Dispositivos LoRaWAN

1. ***Clase A***

   - Utiliza el protocolo de ALOHA, busca la manera de maximizar el uso de energía, los nodos buscan ventanas para transmitir, pero uno solo tendrá éxito en la transmisión de los datos, una vez terminada la transmisión de los datos se abre dos nuevas ventanas que esta podrá ser escuchadas y se irá bajando desde algún Gateway como lo indica en (7).

2. ***Clase B***

   - Según el autor (7) permite la creación de ventanas sin necesidad que exista una transmisión previa, tendrá un aumento de la capacidad para recibir datos de los dispositivos finales. Está sincronizada con los dispositivos finales con

el fin de tener una mejor sincronización de tiempo para que el dispositivo deba abrir la ventana que sea requerida, y esto realiza un aumento de energía debido al mecanismo de sincronización.

3. ***Clase C***

   - Se encuentra los dispositivos de recepción de los nodos que solo será interrumpido cuando exista una transmisión de los datos, tiene mejor latencia de conexión con los dispositivos finales y las pasarelas debido a su mayor consumo. Esta clase permite tener un balance entre el consumo y la capacidad de recepción, aquí funciona tres tipos de aplicaciones: la que no necesita enviar datos a dispositivos finales, la que tiene una demanda intermedia y la que tiene una gran demanda de envío a los dispositivos finales citado por (7).

4.3.7 Características de LoRaWAN

Característica	*Descripción LoRaWAN*
Calidad de Servicio	LoRaWAN es un protocolo asíncrono que utiliza el espectro sin necesidad de alguna licencia. Para manejar cualquier tipo de intermitencia utiliza la técnica de modulación de Chirp-Spread Spectrum (CSS).
Flexibilidad	LoRaWAN es un protocolo abierto sin requerir de licencia. Además, dispone de posibles soluciones flexibles. Tiene ancho de banda escalable con estabilidad bidireccional para su comunicación.
Consumo de Energía	Como se conoce LoRaWAN es un protocolo asíncrono, los nodos finales pueden estar en standby en un periodo largo o corto segundo el dispositivo de aplicación lo necesite. Tiene una vida útil máxima. Funciona con potencia baja.
Cobertura de red/ Rango	LoRaWAN utiliza la topología estrella que tiene como finalidad que su cobertura llegue áreas externas e internas. La comunicación es más larga que supera a la comunicación de los celulares.
Costo de Políticas	Infraestructura de costo bajos con los respectivos nodos finales. Adecuado para usuarios que necesite aplicar un despliegue de red de menor costo.
Despliegue	Fácil el despliegue de una infraestructura de la red en todo el país. Existe red de celulares que son fáciles de actualizar y de reusar
Escalabilidad	LoRaWAN es muy escalable, porque en una sola pasarela puede cursar uno o miles de usuarios o nodos finales, pero depende de un factor importante que es la carga de tráfico, debido a que dispositivos debe ser nodo bajo.
Rendimiento	Para tener un mejor rendimiento de la red es necesario aumentar un número de sub-bandas que va a permitir que el ciclo de trabajo de ella aumente de manera considerable. Tiene un mejor rendimiento superior a otras tecnologías que hoy en día existe además su complejidad es menor a la que otros equipos.
Seguridad	A futuro las redes IoT evolucionará de manera considerable entonces tendrá una certificación que permita que los dispositivos trabajen de manera considerable sin ningún tipo de interrupción. Usa algoritmo de cifrado que proporciona una seguridad de alto nivel y evitar cualquier tipo de intruso.

Tabla 1. Característica y Descripción de la LoRaWAN.

4.4 Servidor LoRaWAN

Un servidor de red LoRaWAN tal y como se indica en (28) es aquel que integra servicios para el manejo de la red, autenticación, direccionamiento de datos hacia otras aplicaciones y enrutamiento de paquetes entre dispositivos.

Según (29) este servidor cuenta con procesos para el funcionamiento de la red LoRa como:

- Enrutamiento y reenvío de mensajes.
- Eliminación de mensajes duplicados si se reciben por múltiples gateways.
- Encriptación y descifrado de mensajes entre nodos.
- Activación y autorización de nodos a la red LoRaWAN.

Como se describe en (30), hoy en día existen varios servidores para la implementación de una red LoRaWAN, entre las más utilizadas y de código abierto se encuentran: The Thinks Netwrok(TTN), y ChirpStack antes denominado LoRa Server.

4.4.1 ChirpStack

Mediante una conferencia (31) se explica que ChirpStack es un servidor de código abierto, el cual permite el despliegue de un número limitado de puertas de enlace y nodos IoT que conforman la red LoRaWAN. La diferencia con respecto a otro servidor de código abierto es que ChirpStack trabaja con una red privada.

Según (30) las características relevantes de este servidor son:

- Admite dispositivos finales de clase A, B, C.
- Posee una interfaz web que permite la administración de aplicaciones, gateways y dispositivos finales.

- Integración con bases de datos y programas de visualización de estos.
- Registro de tramas en vivo por cada nodo y puerta de enlace.

En su página oficial (32) informa que este servidor, proporciona componentes para la red LoRaWAN, las cuales tenemos las siguientes:

- **ChirpStack Gateway Bridge.**

  Según (33) es un servicio que maneja la comunicación con las puertas de enlace LoRaWAN, es decir la comunicación entre el reenviador de paquetes y el servidor de red ChirpStack, convierte los protocolos LoRa en un formato de datos común de ChirpStack Network Server como lo es JSON, para luego publicarlos en un bróker MQTT.

  Se proporcionan algunos backends de reenviador de paquetes como:

  - *ChirpStack Concentratord:* Es un open-source LoRa(WAN) de código abierto que propone una API basada en ZeroMQ [2] la cual puede ser utilizada por una o varias aplicaciones, permite ejecutar múltiples aplicaciones de reenvío de paquetes simultáneamente.
  - *Semtech UDP Packet Forwarder:* Este reenviador de paquetes se ejecuta en el host de pasarela LoRA, reenvía paquetes de las ondas de radiofrecuencia(RF) recibidos por el concentrador a un servidor a través de un enlace IP/UDP.
  - *Basic Station Packet Forwarder:* Es una implementación LoRaWAN Gateway, la cual es compatible con las clases A, B y C de loRaWAN, compatible con diseños de referencia de concentrador v1.5 y v2, además de potentes protocolos de back-end.

  Además de integraciones como Bróker MQTT genérico, Puente MQTT de GCP Cloud IoT Core y Puente MQTT de Azure IoT Hub.

[2] `https://zeromq.org/`

- **ChirpStack Network Server.**

  Este componente según (34) tiene la responsabilidad de la de duplicación de tramas LoRaWAN, recibidas por las puertas de enlace LoRa, estas tramas recopiladas manejan:

  - Autenticación.

  - LoRaWAN capa mac.

  - Comunicación con el servidor de aplicaciones ChirpStack.

  - Programación de tramas de enlace descendente.

- **ChirpStack Application Server.**

  Se encarga de la parte del inventario de dispositivos de la infraestructura LoRaWAN, según su página principal (8) indica que este maneja solicitudes de unión, manejo y cifrado de las cargas útiles de la aplicación. Ofrece una interfaz web en la que podemos administrar dispositivos, usuarios, aplicaciones y organizaciones, para la integración con servicios externos se tiene gRPC y API Restful. Los datos transmitidos de los dispositivos se pueden enviar o recibir a través de MQTT y HTTP, para poder migrarlos a una base de datos como se aprecia en la Fig.8.

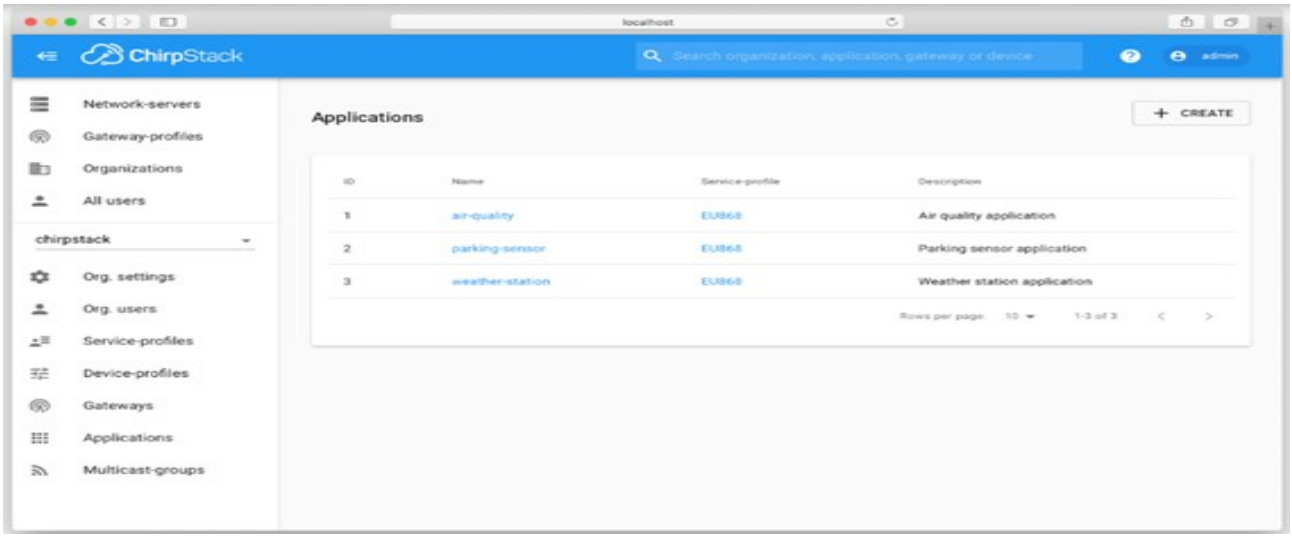


Figura 8. *Interfaz web de chirpstack (8).*

- **ChirpStack Gateway OS.**

  Es un sistema operativo basado en Linux que sirve para ejecutar la pila de ChirpStack, en una puerta de enlace LoRa basada en Raspberry Pi. Según (35) el objetivo de este componente es facilitar el inicio de LoRaWAN y el servidor ChirpStack, en pocos pasos y con mayor facilidad para configurar las puertas de enlace de la red LoRaWAN.

  Tipos de imagen:

  - chirpstack-gateway-os-base: proporciona el ChirpStack Concentratord y el ChirpStack Gateway Bridge preinstalados, incluye una CLI para la configuración de las puertas de enlace.
  - chirpstack-gateway-os-full: proporciona un servidor de ChirpStack completo y con un servidor de aplicaciones que se ejecutan en la puerta de enlace, además de incluir todas las funciones de chirpstack-gateway-os-base.

4.4.2 Arquitectura de ChirpStack

La arquitectura de ChirpStack está compuesta por la siguiente ramas como se ilustra en la Fig.9.

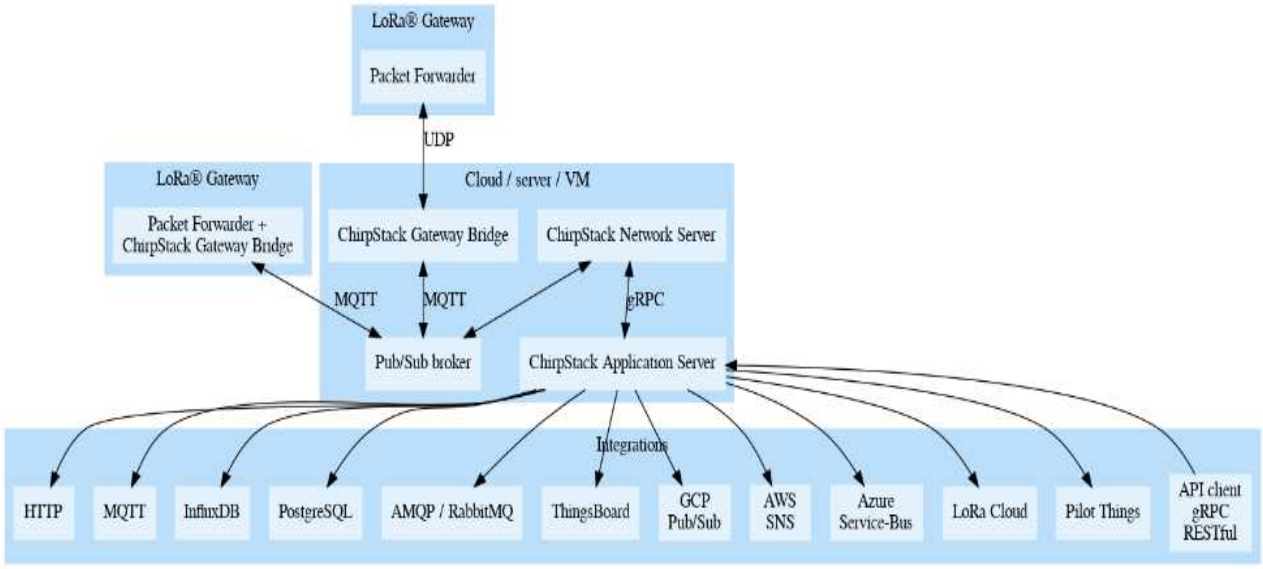


Figura 9. *Arquitectura de ChirpStack (9).*

Según (9) se explica cómo se conectan los componentes de la arquitectura del servidor ChirpStack de la siguiente manera:

- **Dispositivos LoRaWAN.** Son dispositivos que envían datos al servidor ChirpStack mediante gateways LoRa, estos dispositivos pueden ser sensores que tiene una gran cantidad de utilizados entre ellas la medición de calidad del aire, humedad, temperatura, etc.

- **LoRa Gateway.** Generalmente, escucha 8 más canales simultáneamente y reenvía los datos recibidos a ChirpStack. El software encargado de esto es el Packet Forwarder, las implementaciones más comunes para esto son el reenviador de paquetes UDP de Semtech y el reenviador de paquetes de estación básica de Semtech como lo menciona en (35).

- **ChirpStack Gateway Bridge.** Se encuentra entre el reenviador de paquetes y el intermediario MQTT, el cual transforma el formato de reenviador de paquetes a JSON que es utilizado por ChirpStack como no indica en (33).

- **ChirpStack Network Server.** Según los autores (34) administra el estado de la red, maneja solicitudes de unión de nodos. Cuando los datos recibidos son duplicados, este eliminara y los reenviara como carga útil al servidor de aplicaciones de ChirpStack.

- **ChirpStack Application Server.** Según lo mencionado en (8) la interfaz web y API para gestión de usuarios, organizaciones, aplicaciones, gateways y dispositivos.

- **Aplicación final.** Recibe datos del dispositivo a través de una de las integraciones configuradas, las cuales puede ser GCP (Google Cloud Platform), además como indica en (35) el propósito de la aplicación final son los análisis, alertas, visualización de datos, activación, etc.

4.5 Node RED

En (11), se describe a Node-RED como un software de programación visual, que permite conectar dispositivos de hardware, API y servicios en línea. Se basa en JavaScript, creada en la plataforma Node.js y aprovecha al máximo su modelo sin bloqueo basado en eventos.

Como se describe en (36), Node-RED es una herramienta que contiene nodos que están representados por iconos. Los nodos proporcionan distintas funciones, como monitoreo de flujos. Los flujos que se crean se almacenaran en formato JSON. Permite a los desarrolladores conectar nodos ya sea de entrada, salida o procesamiento para la creación de flujos en el que podamos realizar el procesamiento de datos, control, alertas, etc.

Está compuesto por 3 componentes básicos:

– Panel de nodos

– Panel de flujo

– Panel de información y depuración.

De acuerdo con (37), un nodo representa la lógica de un dispositivo o un servicio.

Estos nodos se conectan entre sí cuando necesitan comunicarse para desarrollar alguna tarea que le asignemos. Existen nodos que sirven para una gran variedad de tareas, algunas de ellas es la implementación de funciones, leer datos de una base de datos, comunicación de dispositivos utilizando el protocolo MQTT, etc.

Hoy en día es una herramienta que más se utiliza en proyectos de IoT, según indica en (10), los motivos para utilizar esta herramienta son:

– Software libre

– Se puede incluir en cualquier sistema operativo.

– Se evitan tareas repetitivas, ya que se puede realizar varios procesos simultáneamente como está en la Fig.10.

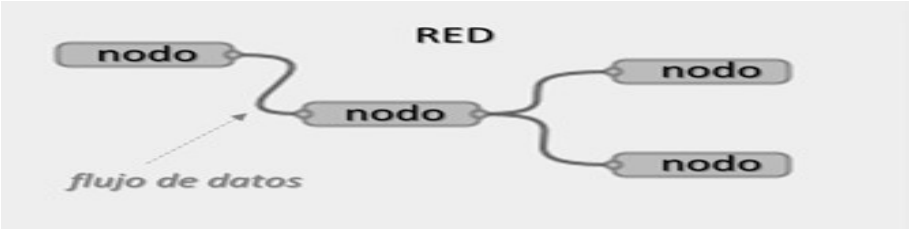


Figura 10. *Flujo de datos entre nodos (10).*

Según (38), Node-RED puede instalarse y ejecutarse en una gran variedad de dispositivos y entornos de despliegue como:

- Dispositivos de bajo coste como Raspberry Pi, el cual actúa como dispositivos edge o pre-edge.
- Despliegue en sistemas operativos Windows, Linux, Mac o incluso en contenedores Docker.

- En una nube que pueden ser Azure, AWS(Amazon Web Service), GCP(Google Cloud Platform), IBM(International Business Machines), etc.

4.5.1 Tipos de nodos en Node-RED

De acuerdo con (10) existen 3 tipos de nodos:

- **Nodos iniciadores.** Inician la ejecución de flujo del proceso y tienen las siguientes características:
  - Poseen 1 puerto de salida.
  - Pueden ser activos, es decir, aquellos flujos que periódicamente supervisan la activación de un evento como un intervalo de tiempo, llegada de un nuevo tópico de MQTT, etc.
  - Pueden ser pasivos, es decir, aquellos flujos que esperan la llegada de un evento de activación generado externamente como una petición de API Rest.
- **Nodos intermedios.** Realizan acciones con la información recolectada en el flujo de datos, posteriormente traspasan los datos resultantes a nodos siguientes. Estos nodos tienen las siguientes características:
  - Poseen 1 un puerto de entrada.
  - Poseen 1 o más puertos de salida.
  - Realizan acciones como: transformación de formato que puede ser en JSON, invocación de APIs, bases de datos, etc.

  Dentro de los nodos intermedios, tenemos otros nodos que realizan la ejecución de código como:

- *function node:* permite la inserción de código de Node.js para tratamiento de datos recibidos o realización de acciones especializadas. Para la utilización de estos nodos es necesario un editor *rich-text* incorporado.
- *exec node:* permite la ejecución de scripts, permite incorporar o reutilizar elementos desarrollados en otros lenguajes de programación como Python, Java, C++, etc.

- **Nodos terminales.** Estos nodos van al final de las ramas del flujo de datos y ejecuta acciones específicas cuando el flujo de datos lo alcanza, cuenta con las siguientes características:
  - Poseen 1 puerto de entrada.
  - Las acciones específicas más frecuentes que se ejecutan son:
    * Ejecución de otros flujos Node-RED.
    * Generación de eventos de negocio en sistemas de mensajería como MQTT, o eventos de notificación en sistemas de mensajería social como Telegram.
    * Invocar otros servicios externos como el envío de mensajes a un backend.
    * Mostrar información con el contenido de datos en el último punto de flujo de datos.

Según la Fig.11 los nodos se puede configurar de diferentes manera simultáneamente.

Figura 11. *Tipos de nodos conectados (11).*

Según su página principal (11), en Node-RED viene incluido los siguientes nodos por defecto según la Fig.12:

Figura 12. *Nodos por defecto en Node RED (11).*

Capítulo V

MARCO METODOLÓGICO

5.1 Requerimientos

Para tener una idea clara de lo que pretende este proyecto, se debe considerar algunos requerimientos que ayudara a iniciar con las actividades planteadas.

5.1.1 Alcance

Facilitar el control de consumo de agua en los hogares, teniendo una visión de cuanta de agua consumen en el día o mensualmente y no tener que esperar una plantilla mensual en donde se detallan todos esos valores, como también podemos tener el control de consumo, ya que mientras más consumo haya más se deberá pagar a empresas que prestan este servicio.

5.1.2 Personal Involucrado

Como se aprecia en la Tabla 2, se tiene todos los involucrados en el desarrollo de este proyecto.

Nombre	**Alex Cristopher Cuji Torres**
Categoria Profesional	Estudiante
Rol	Desarrollador
Responsabilidad	Análisis, diseño, desarrollador
Informe de Contacto	acujit@est.ups.edu.ec
Nombre	**David Oswaldo Sarumeño Avila**
Categoria Profesional	Estudiante
Rol	Desarrollador
Responsabilidad	Análisis, diseño, desarrollador
Informe de Contacto	dsarumeno@est.ups.edu.ec
Nombre	**Erwin Jairo Sacoto Cabrera**
Categoria Profesional	Ingeniero de Sistemas
Rol	Administrador del proyecto.
Responsabilidad	Revisión de cada sprint del proyecto.
Informe de Contacto	jsacotoe@ups.edu.ec

Tabla 2. Personal involucrado en el proyecto..

5.1.3 Requerimientos Funcionales

Como se aprecia en la Tabla 3, se tiene el requerimiento funcional correspondiente a visualización de datos del gateway.

Código	RF01
Nombre	Ingreso de datos para cuenta de Google Cloud.
Descripción	Para la creación de una cuenta de Google Cloud, se requiere ingresar información correspondiente a número de cuenta y contacto.
Entradas	Datos Personales
Proceso	Se creará una cuenta con prueba gratuita en Google Cloud
Salidas	Página principal Google Cloud
Prioridad	Alta

Tabla 3. Requerimiento Funcional 01.

5.1.4 Requerimientos no Funcionales

Como se aprecia en la Tabla 4, se tiene el requerimiento no funcional correspondiente a visualización de datos del gateway.

Código	RN01
Nombre	Datos de RSN y RSSI de gateway.
Descripción	Mediante Node Red se visualiza datos del gateway en vivo.
Entradas	Consulta a base de datos
Proceso	Se visualizarán los datos correspondientes al gateway
Salidas	Datos en vivo de gateway
Prioridad	Baja

Tabla 4. Requerimiento no funcional 01.

Como se aprecia en la Tabla 5, se tiene el requerimiento no funcional correspondiente a visualización de dispositivos conectados al servidor.

Código	RN02
Nombre	Datos de dispositivos
Descripción	Visualización de datos de dispositivos conectados
Entradas	Consulta a base de datos
Proceso	Se visualizarán los dispositivos conectados
Salidas	Datos en vivo de dispositivos
Prioridad	Baja

Tabla 5. Requerimiento no funcional 02.

Como se aprecia en la Tabla 6, se tiene el requerimiento no funcional correspondiente a visualización frecuencia de los dispositivos conectados al servidor.

Código	RN03
Nombre	Datos de frecuencia
Descripción	Visualización de datos de frecuencia de los dispositivos conectados
Entradas	Consulta a base de datos
Proceso	Se visualizarán los datos de la frecuencia de los dispositivos
Salidas	Datos en vivo de frecuencia
Prioridad	Baja

Tabla 6. Requerimiento no funcional 03.

Como se aprecia en la Tabla 7, se tiene el requerimiento no funcional correspondiente a visualización frecuencia de los dispositivos conectados al servidor.

Código	RN04
Nombre	Datos de consumo de agua
Descripción	Visualización de datos en vivo del consumo de agua de los dispositivos
Entradas	Consulta a base de datos
Proceso	Se visualizarán los datos del consumo de agua
Salidas	Datos en vivo del consumo
Prioridad	Alta

Tabla 7. Requerimiento no funcional 04.

5.2 Arquitectura del sistema

La arquitectura se basa en ChirpStack, Google Cloud Platform, PostgreSQL y Node Red. En la siguiente Figura13, se visualiza el sistema completo.

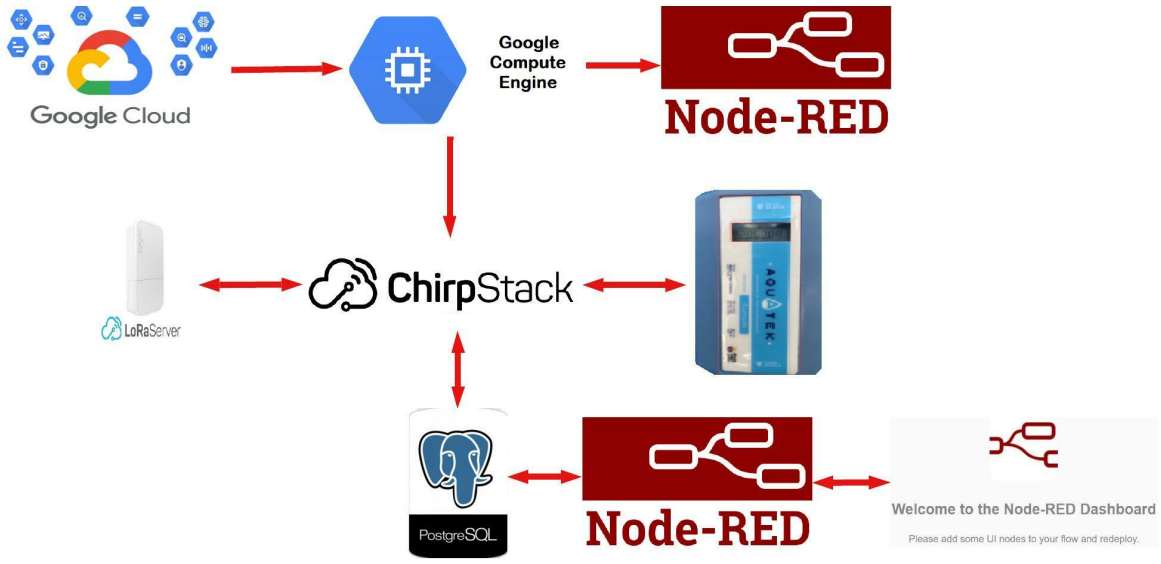


Figura 13. *Arquitectura del Proyecto*.

La arquitectura está conformada por varias fases que permite la comunicación con el medidor de agua ultrasónico, cada fase tiene su papel fundamental para el funcionamiento del mismo.

Primero se creará una máquina virtual en Google Cloud Platform para instalar el servidor de ChirpStack, a su vez se conectará el dispositivo LoRa que tendrá en su configuración el network server del servidor de ChirpStack que permitirá el paso de la data que transmite dicho dispositivo, luego se conectara con el dispositivo de agua ultrasónico que enviara la data al servidor de ChirpStack, esa data enviada será almacenada en la Base de Datos PostgreSQL y luego se procesara a través de node red que recibe los datos en vivo para luego graficarlos en Node Red Dashboard.

5.3 Herramientas utilizadas en el proyecto

Como se aprecia en la Tabla 8, se tiene las herramientas que ha sido utilizada en este proyecto para el procesamiento, almacenamiento y la visualización de la data.

Tecnología	Uso	Descripción
ChirpStack	Procesamiento de Datos	Recibe los datos transmitidos por el dispositivo y lo guarda en la base de datos.
Node Red	Recolección de Datos	Realiza consultas a base de datos mediante nodos enlazados para realizar gráficas estadísticas de los datos.
PostgreSQL	Almacenamiento de Datos	Guarda los datos recibidos en el servidor ChirpStack.
Node Red Dashboard	Visualización de datos en vivo.	Esta aplicación permite visualizar la data recolectada en gráficos estadístico de los datos.

Tabla 8. Herramienta utilizada para el proyecto 01.

5.4 Instalación y configuración herramientas para desarrollo del proyecto

5.4.1 Instalación de máquina virtual en Google Cloud Platform

5.4.1.1 Creación de cuenta en Google Cloud Platform

Como se indica en la Fig.14 se entrara a la página principal de Google Cloud Platform, [1] ahí se dará clic en empezar gratis, se iniciara sesión con una cuenta de Google para continuar.

[1] `https://cloud.google.com/`

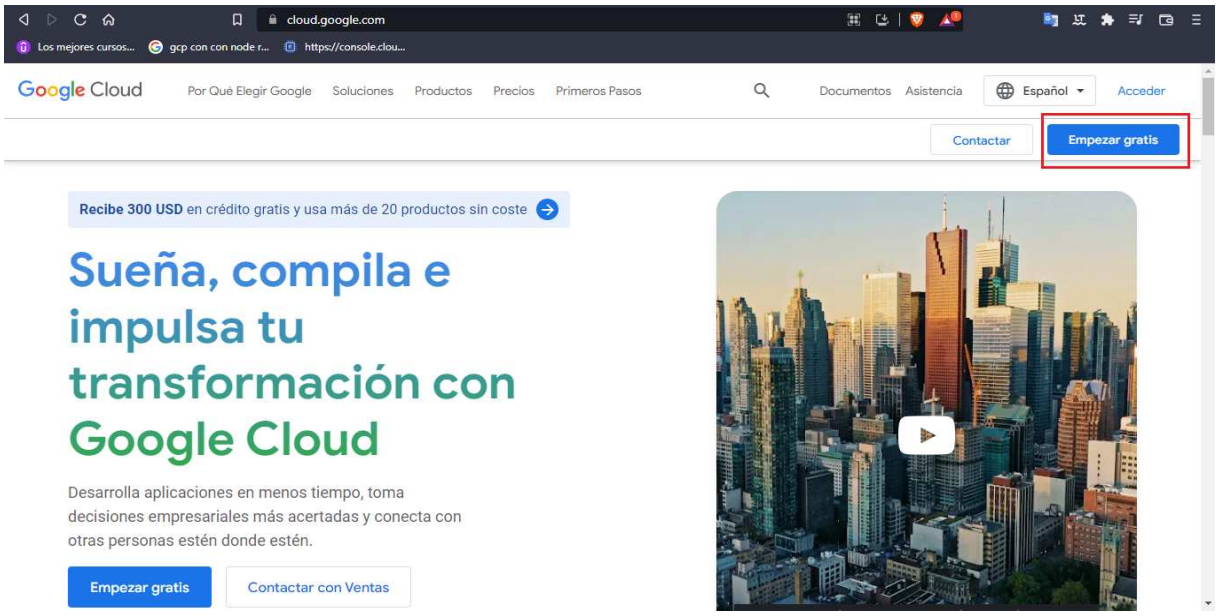


Figura 14. *Página principal Google Cloud Platform.*

Cuando se haya iniciado con una cuenta de Google, se realizaran los siguientes pasos para continuar con una prueba gratuita de la consola de Google Cloud Platform, en la que se tendrá 300 dólares gratis para probar todos los servicios que ofrece esta plataforma.

- **Paso 1, información de la cuenta.**

Se ingresará el país y un proyecto al cual va orientado la creación de esta plataforma, una vez seleccionado estas opciones aceptamos termino y condiciones y damos en continuar como lo indica en la Fig.15.

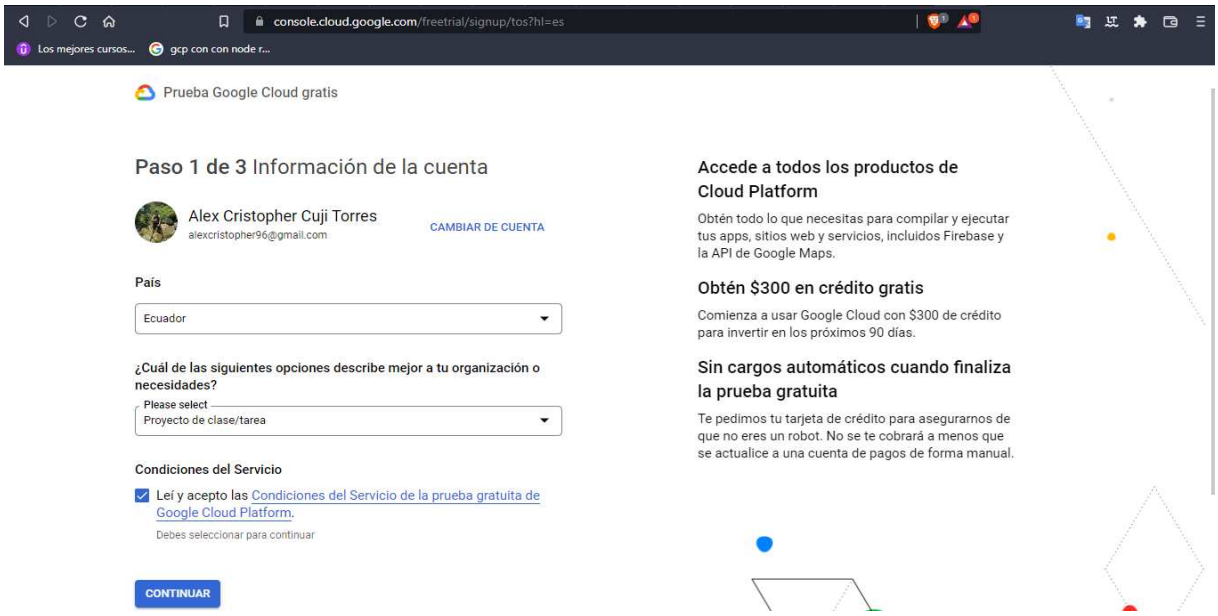


Figura 15. *Ingreso de la información de la cuenta de Google Cloud.*

- **Paso 2, verificación de identidad e información de contacto.**

Se ingresará el número de celular y damos clic en enviar código, en seguida nos enviará un código de verificación al número telefónico ingresado, para luego poder verificarlo en la siguiente página de verificación de identidad como se aprecia en la Fig.16.

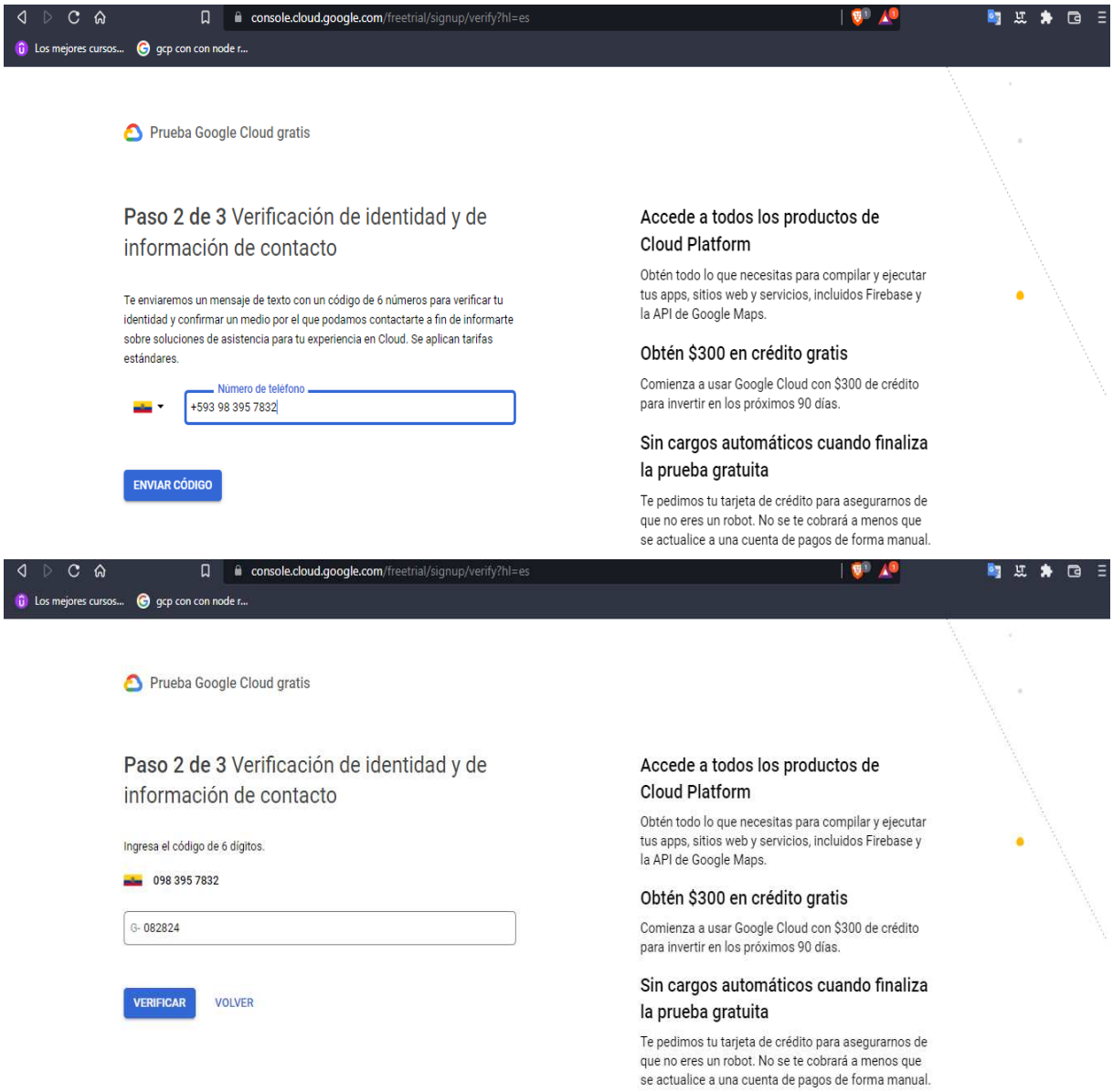


Figura 16. *Verificación de identidad de la cuenta de Google Cloud.*

- **Paso 3, verificación de información de pago.**

En este punto se deberá ingresar un perfil de pago anidado a una tarjeta de pago, en el cual se ingresara el número de la tarjeta, mes y año de caducidad, y el CVC(Card Verification Code) de la tarjeta ingresada. La cuenta de Google estará activa de forma gratuita durante los próximos 3 meses, posteriormente se procederá a cobrar por los servicios que se usará

en esta plataforma como lo indica en la Fig.17.

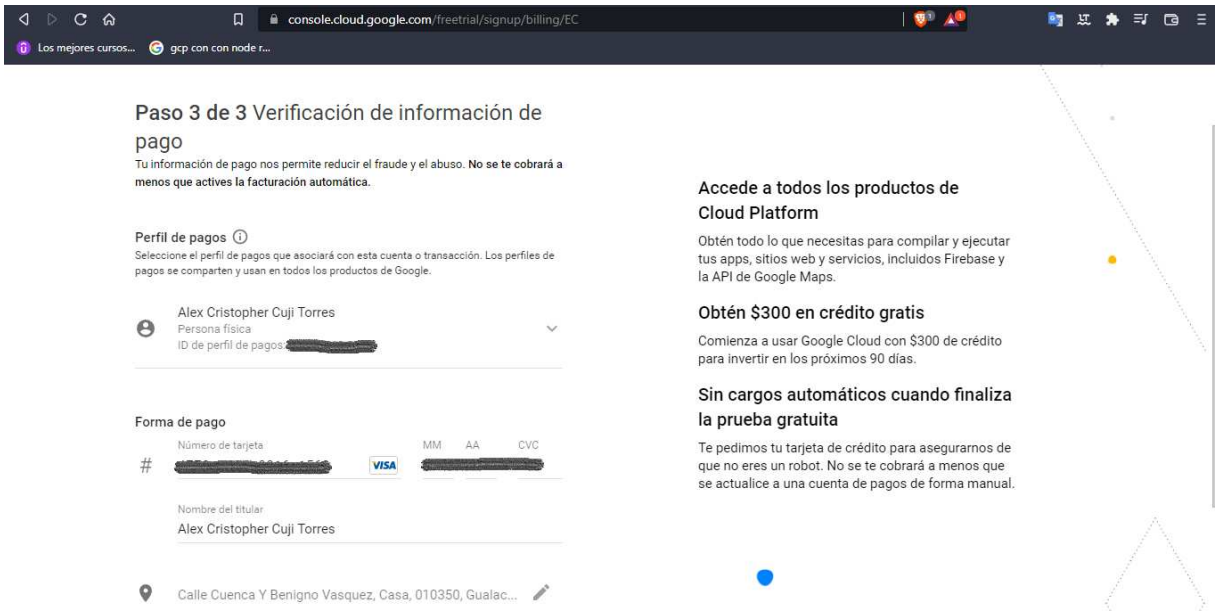


Figura 17. *Verificación de información de pago de Google Cloud.*

Una vez culminado los pasos de creación de la cuenta, se procederá a crear un proyecto en la parte de *My First Project*, para una mejor organización en futuros proyectos en esta plataforma, como se aprecia en la Fig.18.

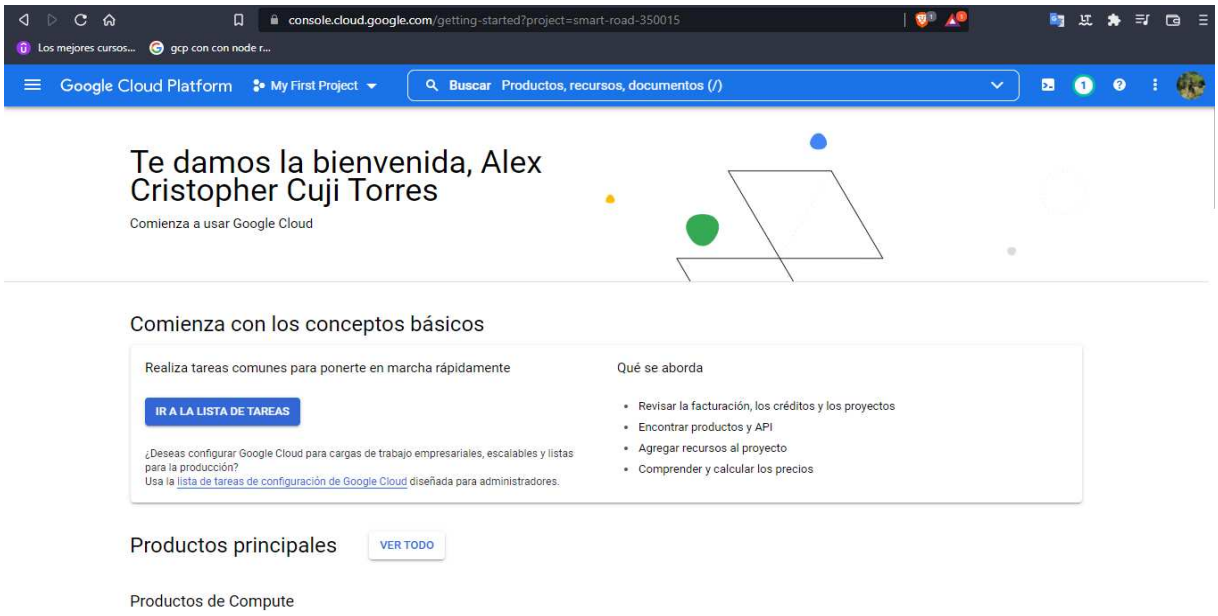


Figura 18. *Página principal de Google Cloud.*

5.4.1.2 Creación de una máquina virtual

- **Activación de Compute Engine API.**

Para ello se ingresará al menú principal donde se encontrará todos los servicios que ofrece Google Cloud, se buscará la opción *Compute Engine*, y se mostrará un submenú donde se seleccionará la opción *Instancias de VM* como lo indica en la Fig.19.

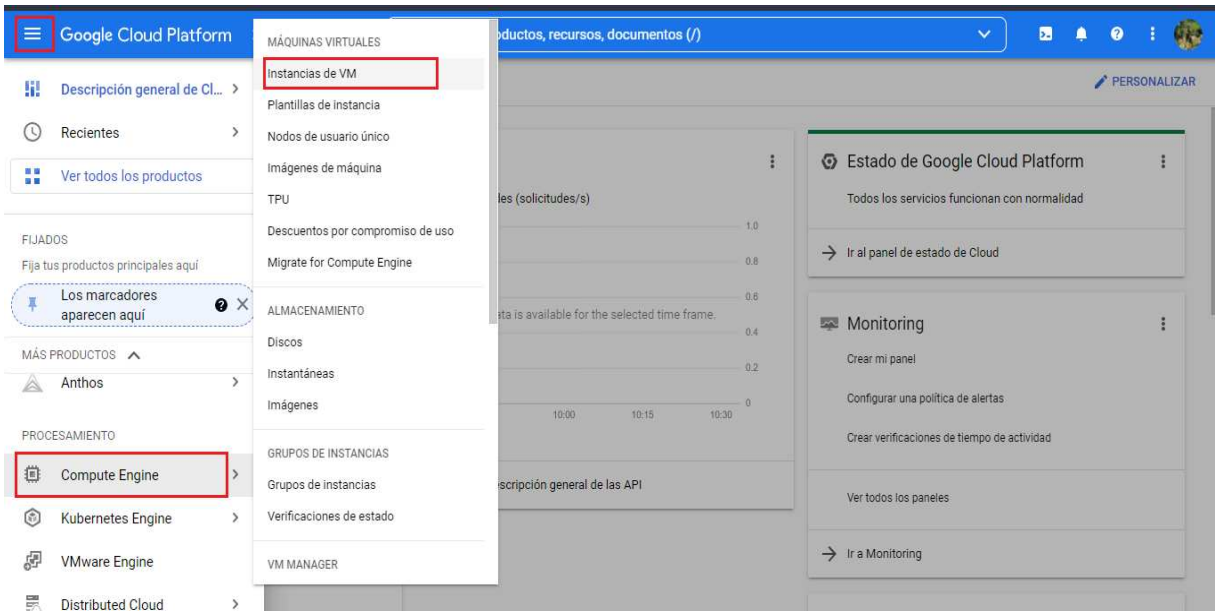


Figura 19. *Instancias de VM*.

Una vez seleccionado la opción de *Instancias de VM*, se habilitará la API para poder crear una instancia de Virtual Machine, como se aprecia en la Fig.20.

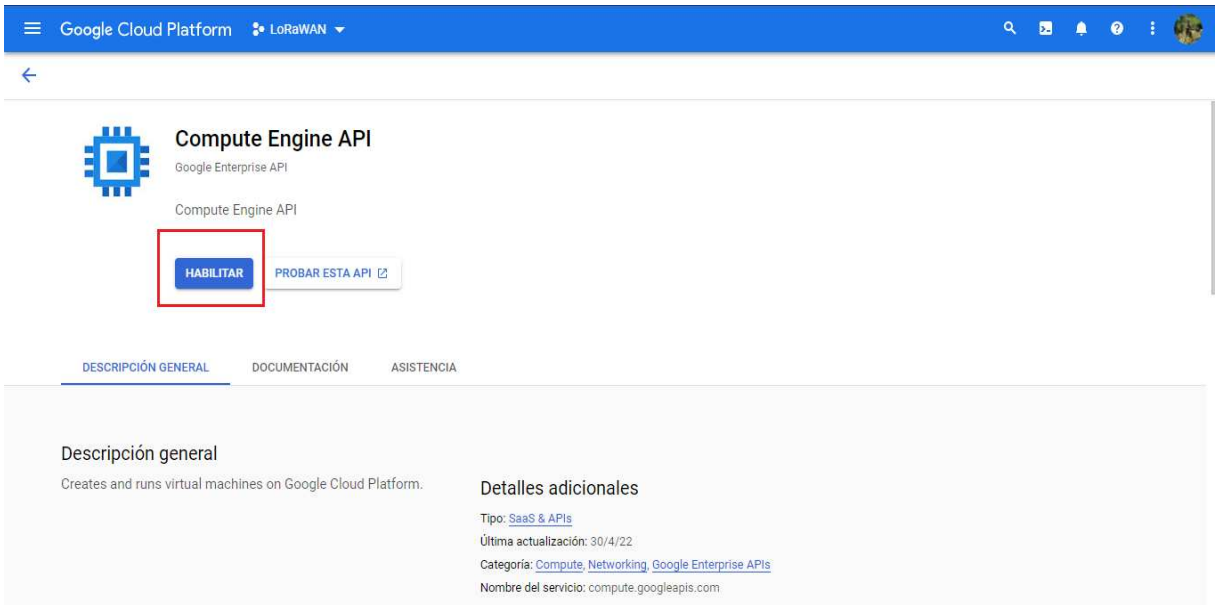


Figura 20. *Habilitación de Compute Engine API*.

- **Creación de instancia de VM.**

Cuando se habilita la API de Compute Engine, se mostrará una ventana de ese servicio en donde podemos crear una instancia y configurarla como lo indica en la Fig.21.

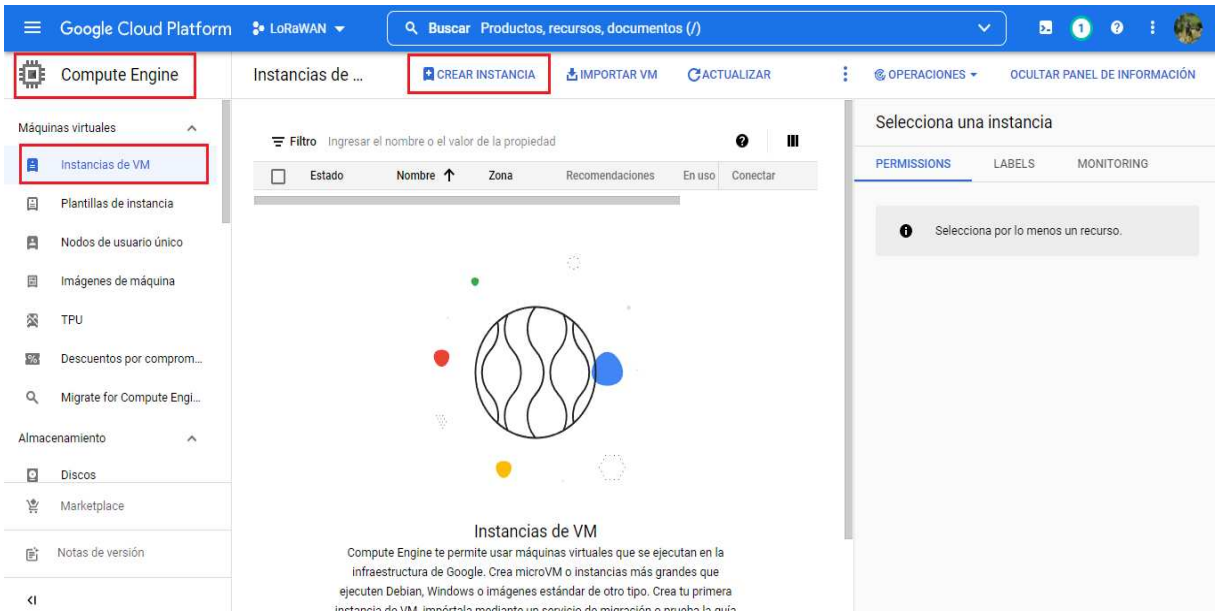


Figura 21. *Ventana principal de Compute Engine*.

Para crear una instancia se deberá dirigir a la parte de crear instancia, a continuación se abrirá una ventana en donde nos pedirá configurar algunos parámetros de la máquina virtual.

Primeramente, se deberá asignar un nombre a la máquina virtual, luego se deberá asignar una región y zona de conexión, en este caso se seleccionará la región *us-central1(lowa)*, y la zona será *us-central1-c*.

Adicional se deberá elegir una serie, la cual será N1, y un tipo de máquina, el cual indicara el número de CPU y cantidad de memoria RAM de la máquina, aquí se puede modificar según la necesidad de cada uno, si no se necesita muchos recursos se podrá de-

jar en un tipo de máquina estándar, la cual cuenta con 1 CPU y 3.75 GB de memoria RAM.

En la parte derecha de la ventana se podrá observar el costo de creación, esto variará según los recursos, zonas, disco, que se asigne a la máquina, este costo se restara con los 300 dólares asignados gratuitamente al inicio de la creación de la cuenta de Google como se aprecia en la Fig.22.

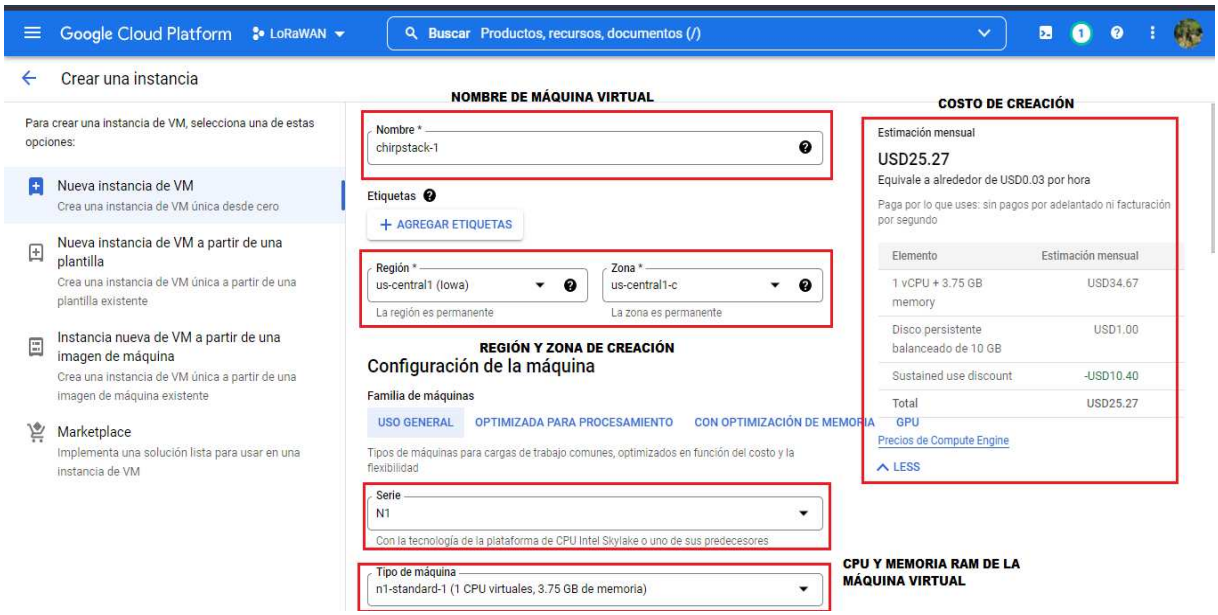


Figura 22. *Creación de VM1.*

Posteriormente, se procederá a elegir el tamaño de disco y el sistema operativo a instalar, el disco estándar para la instalación es de 20 GB, este se podrá modificar y aumentar, eso si el costo aumentara. Este servicio consta de muchos sistemas operativos que se podrá elegir, así mismo de distintas versiones del sistema operativo como lo indica en la Fig.23.

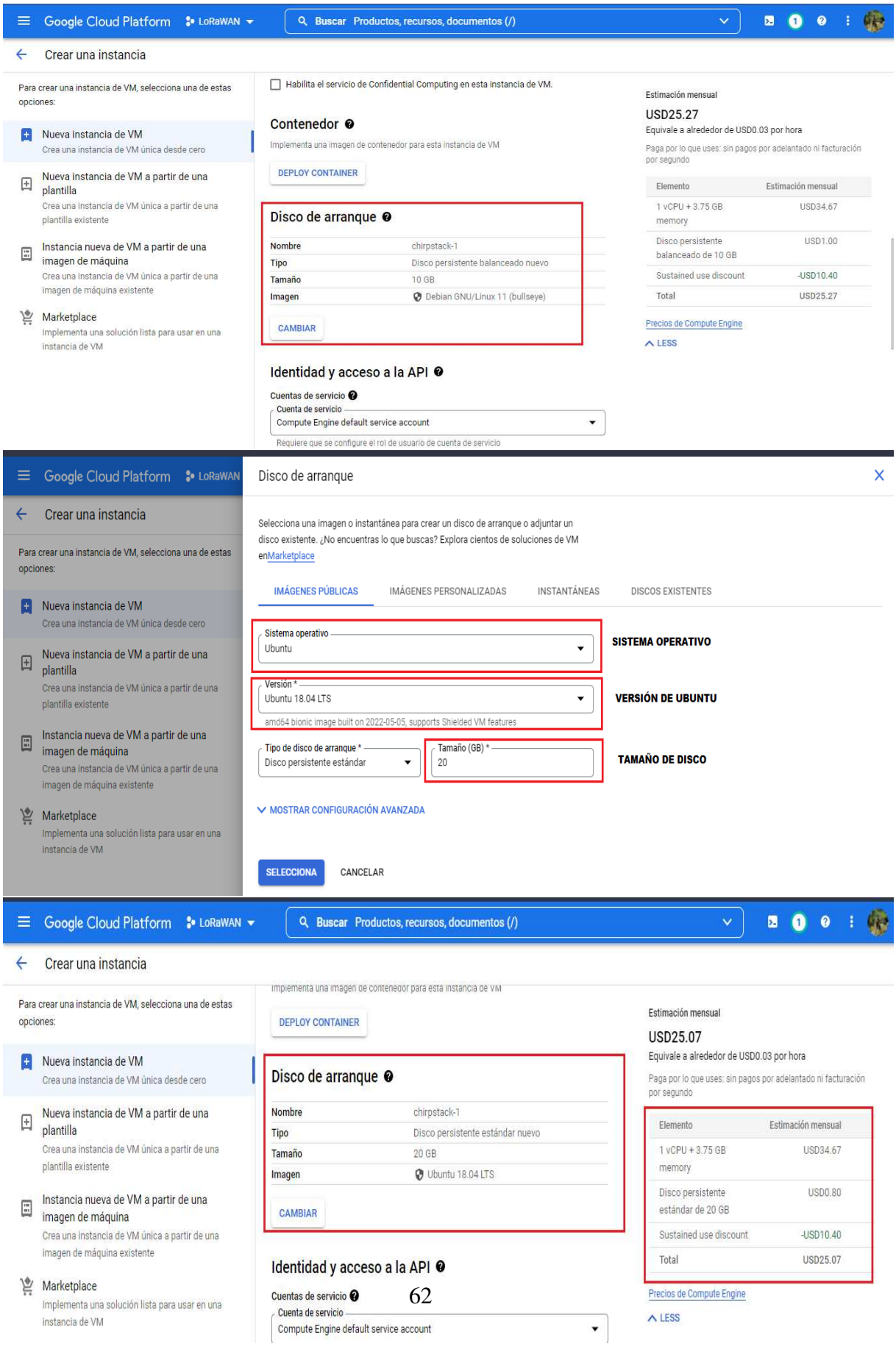


Figura 23. *Asignación de disco y sistema operativo de VM.*

Por último, se pedirá elegir permisos de acceso y firewall para la máquina, en este caso se asignará un tipo de acceso predeterminado y firewall con tráfico HTTP, si se cuenta con tráfico con seguridad adicional se asignará también permisos para tráfico de HTTPS. Una vez configurado los permisos se dará clic en crear y ya se creará la instancia de VM con el sistema operativo asignado como se aprecia en la Fig.24.

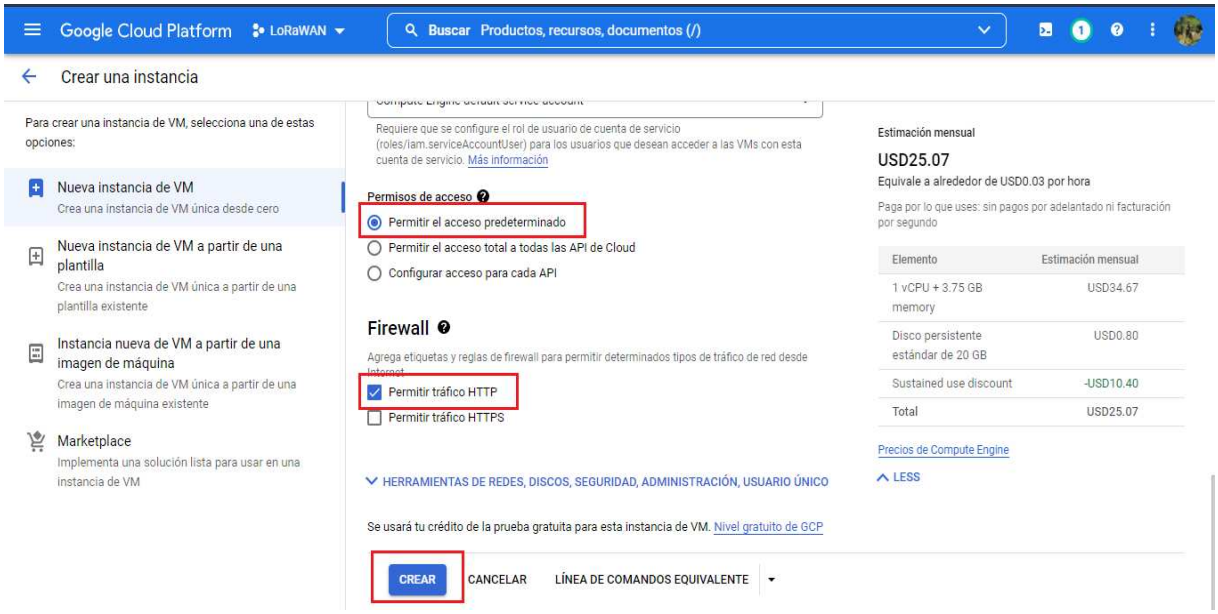


Figura 24. *Asignación de permisos y firewall de VM.*

- **Actualización de VM.**

Según en la Fig.25, una vez creada la máquina virtual, se establecerá una conexión por SSH al sistema operativo designado, en este caso Ubuntu18, ahí se hará un update como se aprecia en la Fig.26 y upgrade para actualización de paquetes como se refleja en la Fig.27.

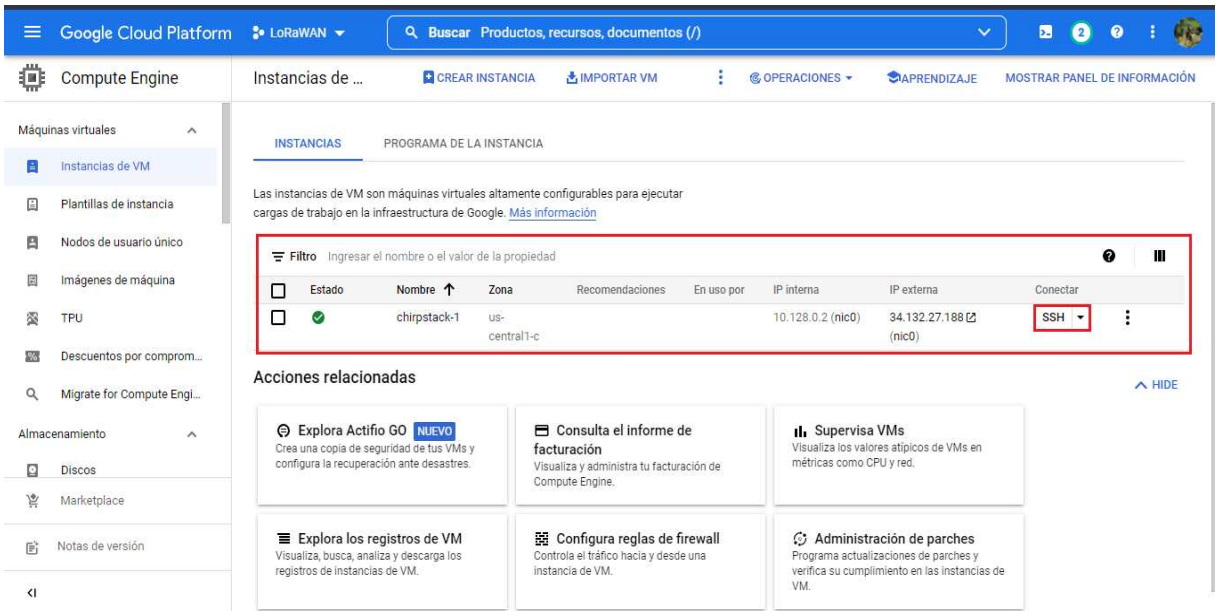


Figura 25. *Conexión a VM por SSH*.

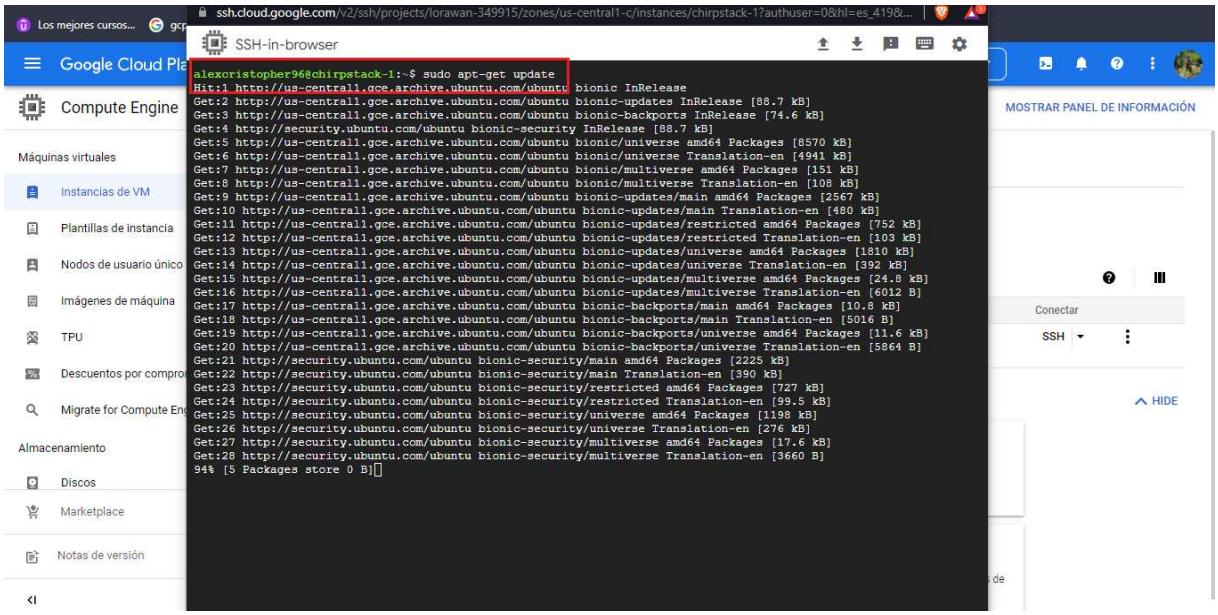


Figura 26. *Update en VM*.

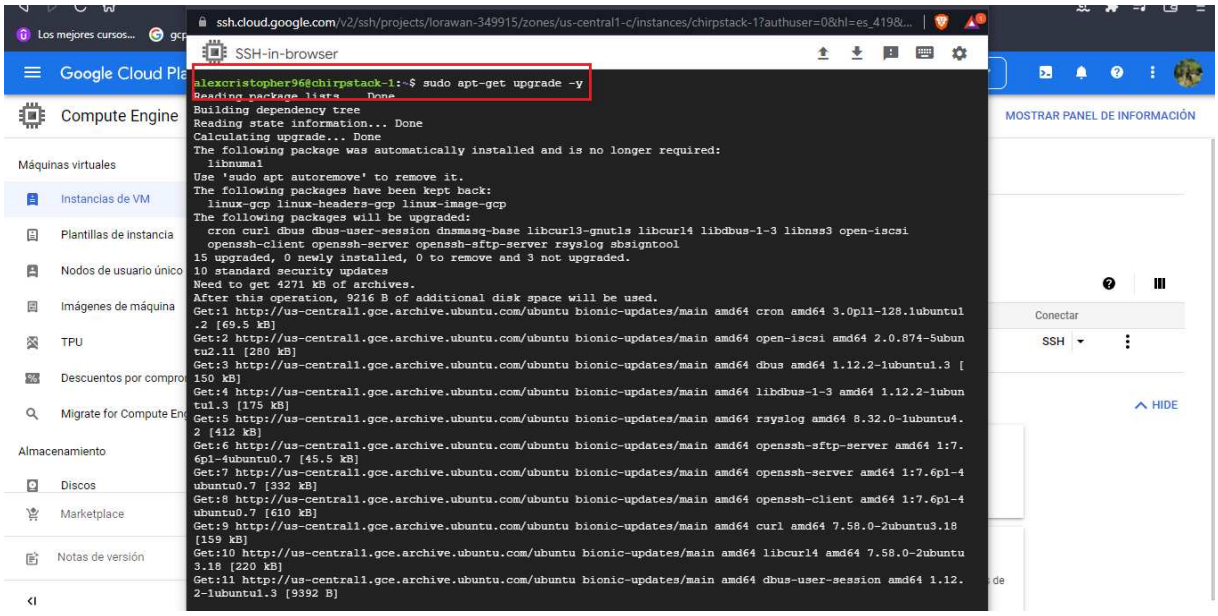


Figura 27. *Upgrade en VM*.

5.4.1.3 Reservar IP externa de conexión a VM

Según como se ilustra en la Fig.28, esto hace para que la IP externa de conexión a la máquina virtual no cambie cada vez que encendemos la VM. Esto servirá para que cuando instalemos servicios dentro de la VM podamos conectarnos siempre por la misma IP, simplemente cambiara los puertos de conexión dependiendo de cada servicio que instalemos.

Para ello se abrirá el menú principal en la parte de Red VPC, y se seleccionará la opción de direcciones IP.

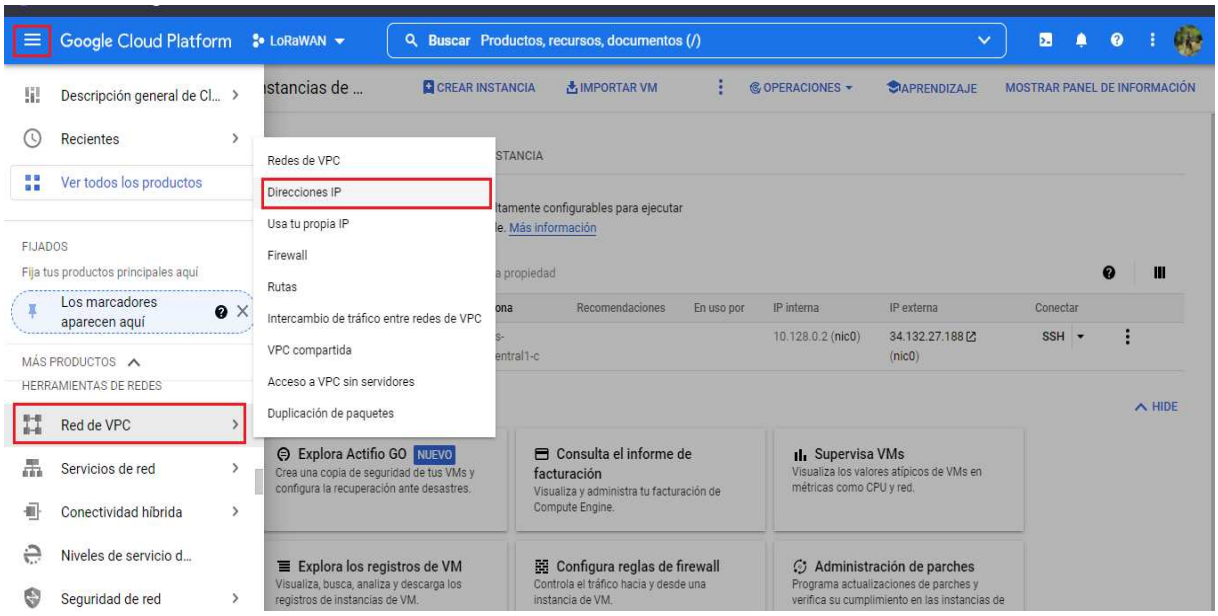


Figura 28. *Direcciones IP de VM*.

Seleccionar la opción de direcciones IP externas, en la cual se mostrara la VM creada recientemente y se seleccionara la opción reservar como se aprecia en la Fig.29.

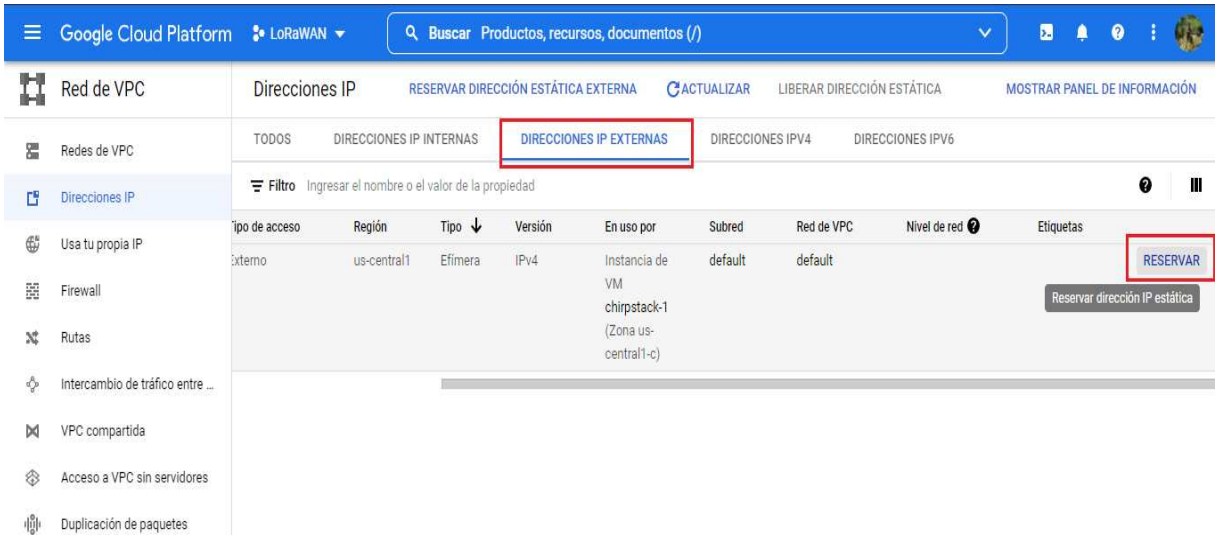


Figura 29. *Reservar IP externa de VM*.

Por último se mostrará en la Fig.30 una ventana donde se ingresara un nombre para la reserva de la dirección IP externa, adicional se podrá ingresar una descripción si se lo

requiere, se dará clic en crear y la IP externa estará reservada para que no cambie cada vez que se encienda la VM.

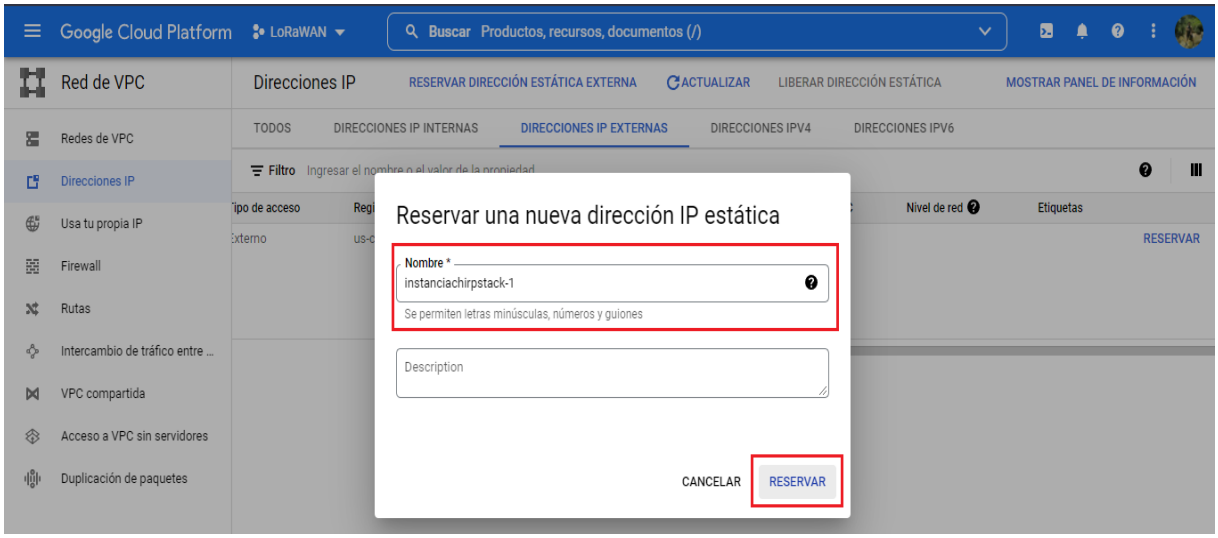


Figura 30. *Nombre de reserva de IP externa de VM.*

5.4.1.4 Creación de regla de firewall para VM

Una regla de firewall servirá para que el tráfico que entre o salga de la VM permita la conexión a distintos puertos, protocolos, rangos de IP, etc.

Primeramente, como se muestra en la Fig.31 se ingresará a la parte de Red VPC, opción firewall. Ahí se seleccionará la opción de crear regla de firewall.

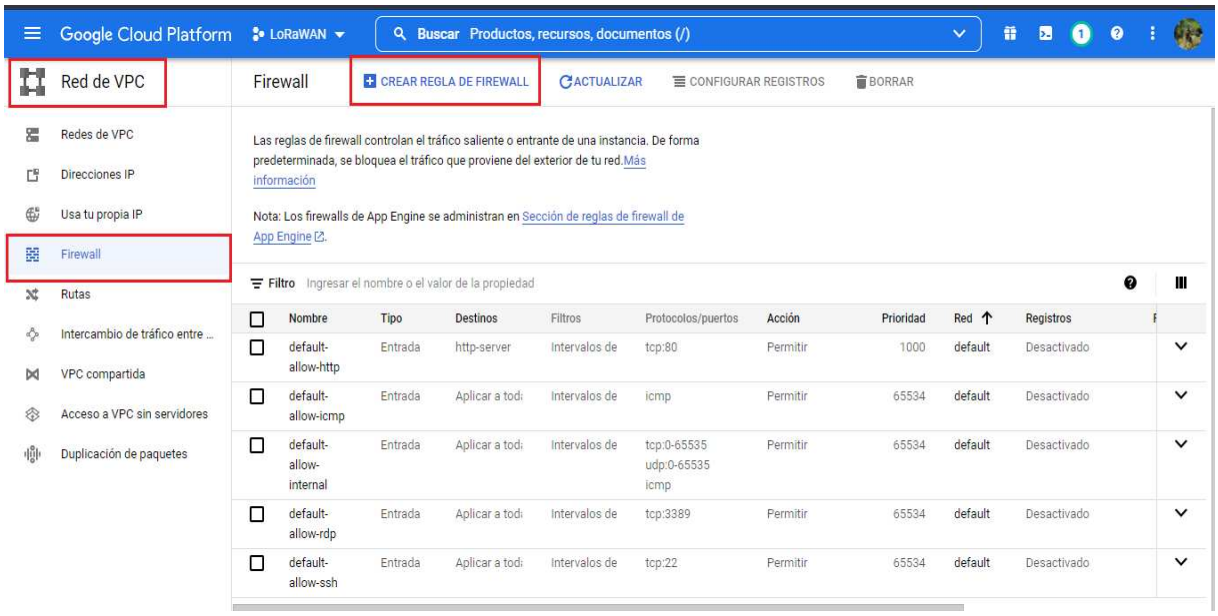


Figura 31. *Reglas de firewall de VM.*

En la ventana de configuración se pedirá ingresar un nombre a la regla, una descripción, activar o desactivar los registros de firewall, este caso se desactivara porque al activar se generara una gran cantidad de registros que pueden aumentar el costo de la VM, luego se seleccionará una red por default y una prioridad de 1000 como se aprecia en la Fig.32.

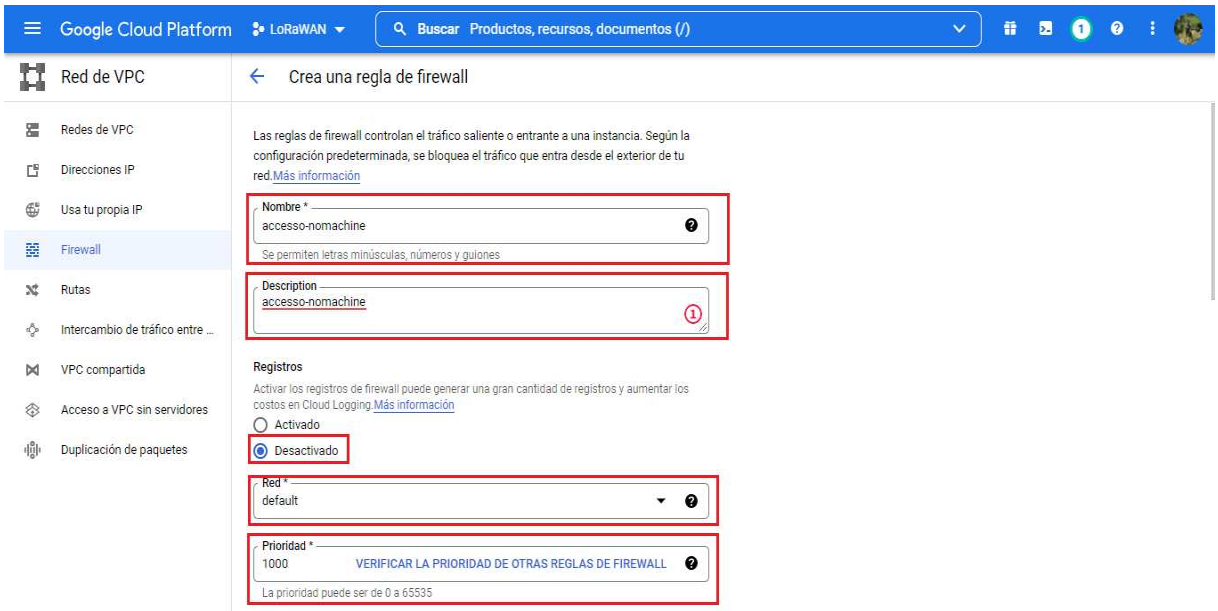


Figura 32. *Creación de reglas de firewall de VM.*

Luego como se observa en la Fig.33, se indicara la dirección de la regla de firewall que puede ser de entrada o salida, se deberá crear una etiqueta de destino en la que se tendrá que anidarla a la configuración de la VM, los rangos de IP será 0.0.0.0/0 para que no tenga limitación y coja todos los rangos de IP que generen tráfico. Por último, podemos permitir ciertos puertos y protocolos que ingresaran como tráfico de datos a la VM o podemos permitir todos los puertos y protocolos de conexión a la VM.

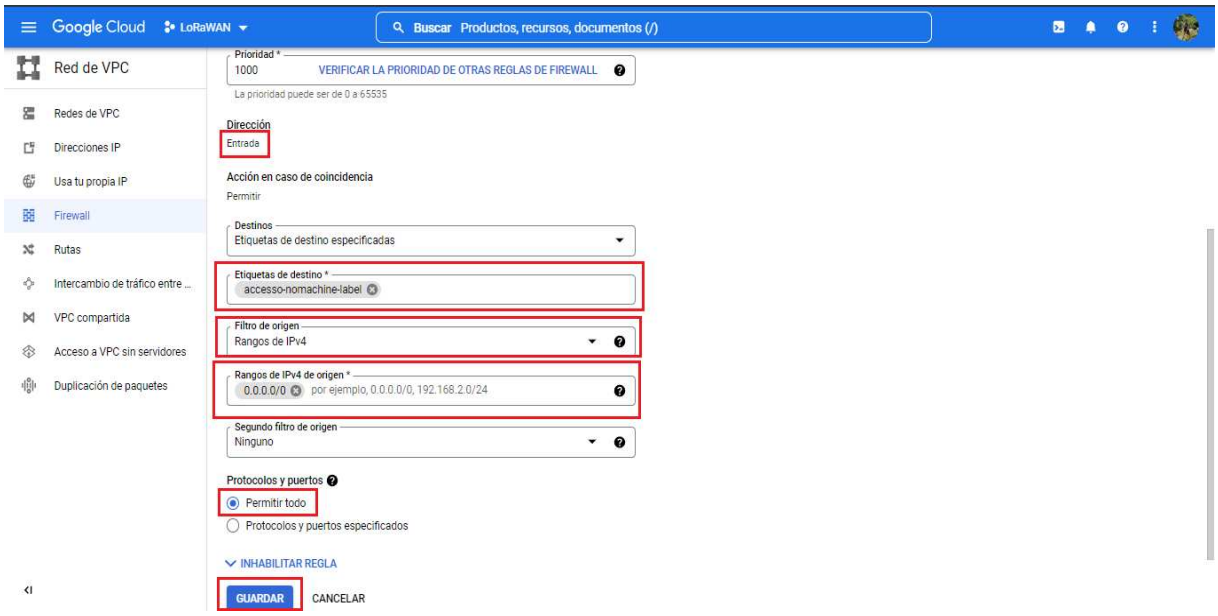


Figura 33. *Configuración de firewall de VM.*

Una vez guardada la regla de firewall la podemos observar en la ventana de firewall como se observa en la Fig.34.

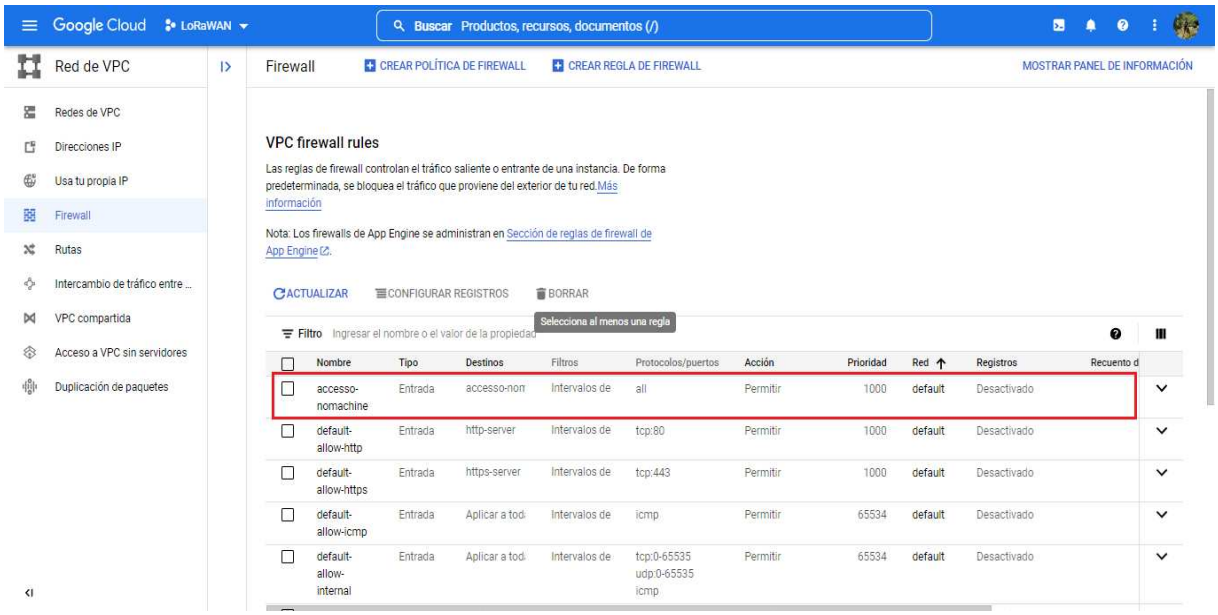


Figura 34. *Regla de firewall creada.*

Una vez creada la regla de firewall, como se observa en la Fig.35 de las ventanas de GCP(Google Cloud Platform), se procede a asignarla a la VM creada, para ello se ingresa a la instancia de VM en la opción editar.

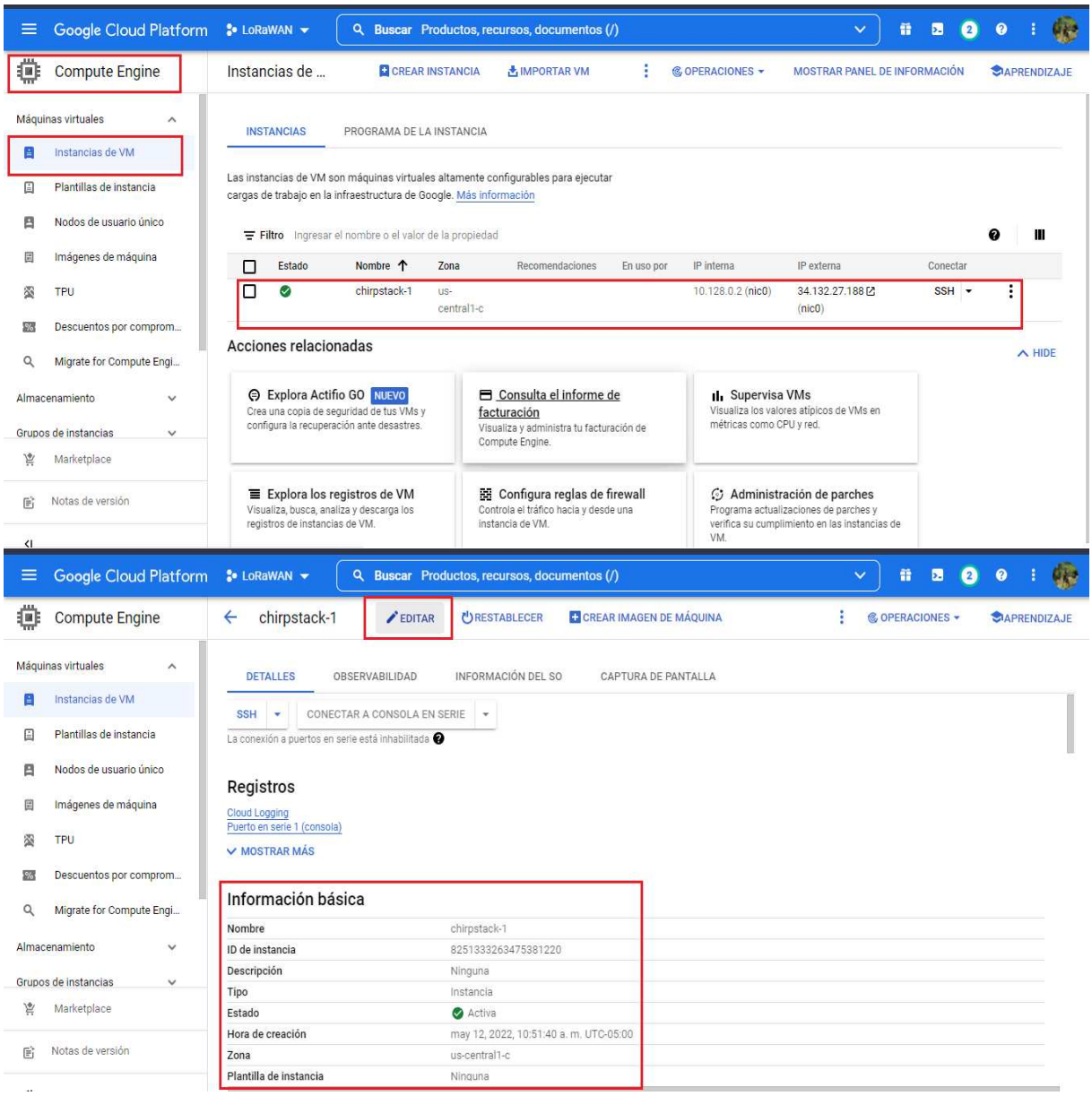


Figura 35. *Editar VM*.

Una vez ingresado a la ventada de editar VM, se procede a dirigir a etiquetas de red, en donde se ingresara la etiqueta creada anteriormente en la creación de la regla de firewall,

este proceso nos ayuda a que se permita el tráfico de todos los puertos y protocolos que ingresen a la VM como lo indica en la Fig.36.

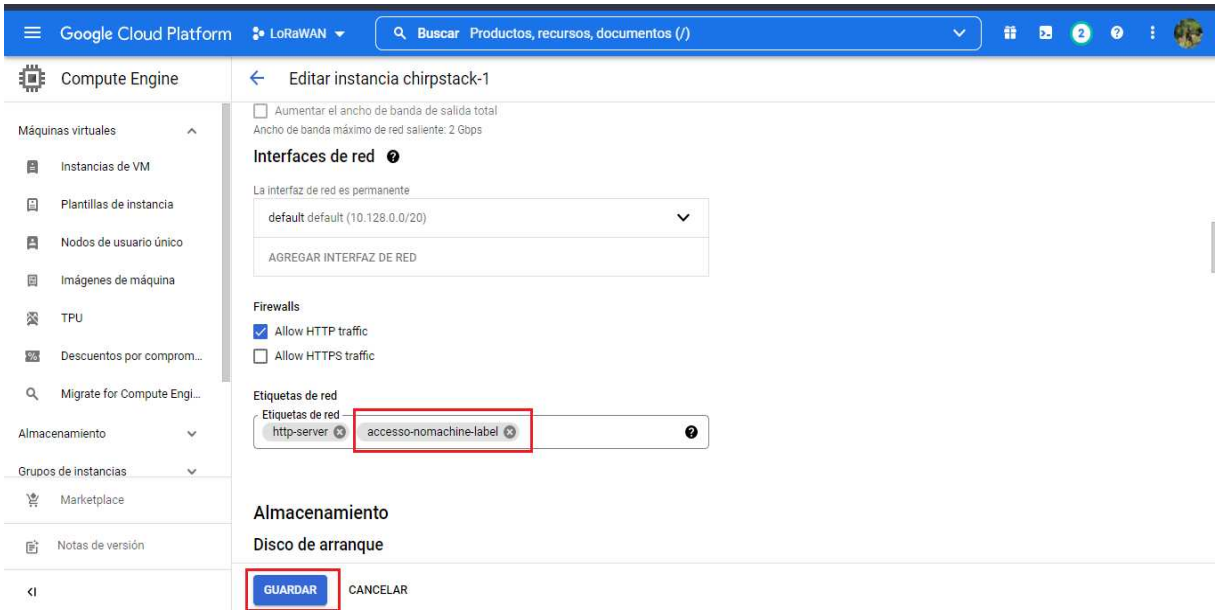


Figura 36. *Asignación de regla firewall a VM.*

Por último se dará clic en guardar y este sería todo el proceso de creación y configuración de VM de Google Cloud Platform.

5.4.2 Instalación y configuración servidor ChirpStack en máquina virtual de Google Cloud Platform

Para la instalación de este servidor es necesario instalar tres servicios que son necesarios para el correcto funcionamiento del mismo:

- **Chirpstack Gateway Bridge:** Maneja la comunicación con los gateway LoRaWAN.
- **Chirpstack Network Server:** Permite crear una red LoRaWAN.
- **Chirpstack Application Server:** Permite tener la interfaz del servidor Chirpstack para registrar los gateways y dispositivos.

Además, es necesario instalar ciertas librerías y componentes iniciales para poder instalar satisfactoriamente el servidor Chirpstack y tenga un correcto funcionamiento, entre estos componentes están:

- **MQTT broker:** Es un protocolo de publicación/suscripción que permite a los usuarios publicar información sobre tramas a los que otros pueden suscribirse. Una implementación popular del protocolo MQTT es Mosquitto, sirve para la comunicación del servidor con otro software de visualización o tratamiento de datos.
- **Redis:** Es una base de datos en memoria utilizada para almacenar datos relativamente transitorios, y que ayudara a la visualización de datos en tiempo real, según el tráfico de datos que el servidor tenga en ese momento.
- **PostgreSQL:** Es una base de datos, la cual almacenara todos los datos que se reciban en el servidor.

Primeramente, como se observa en la Fig.37, se conectará por SSH a la VM creada anteriormente en Google Cloud.

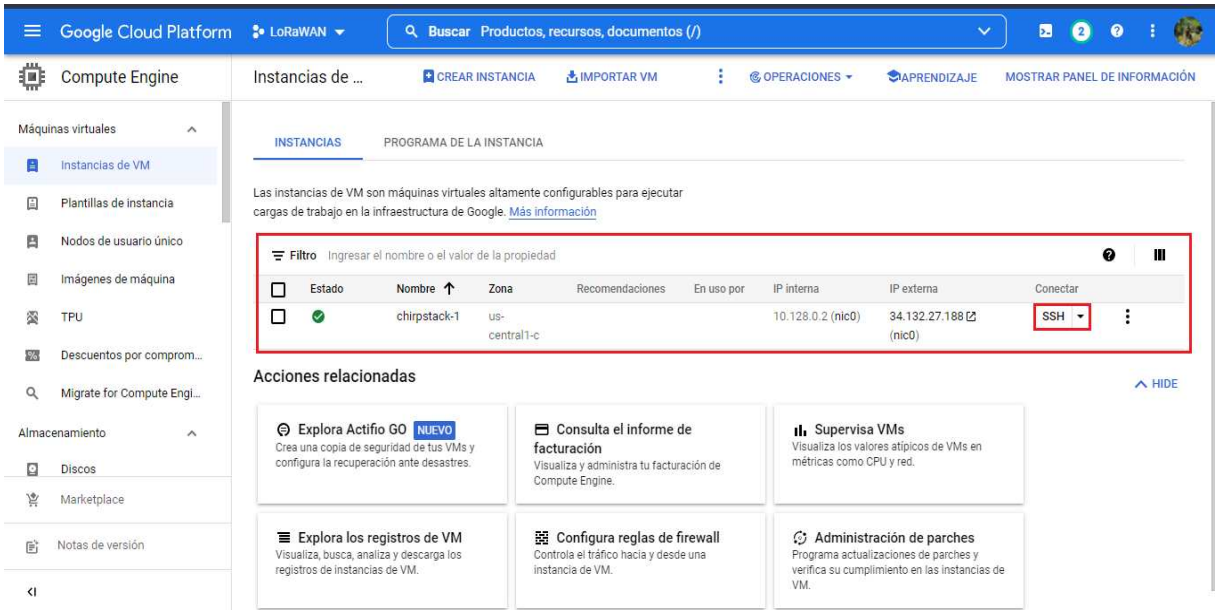


Figura 37. *Conexión a VM por SSH.*

5.4.2.1 Instalación de componentes para servidor ChirpStack

- **Instalación de MQTT Broker, Redis server y PostgreSQL.**

Para la instalación de estos componentes se digitará el siguiente código en la instancia de VM como indica en la Fig.38.

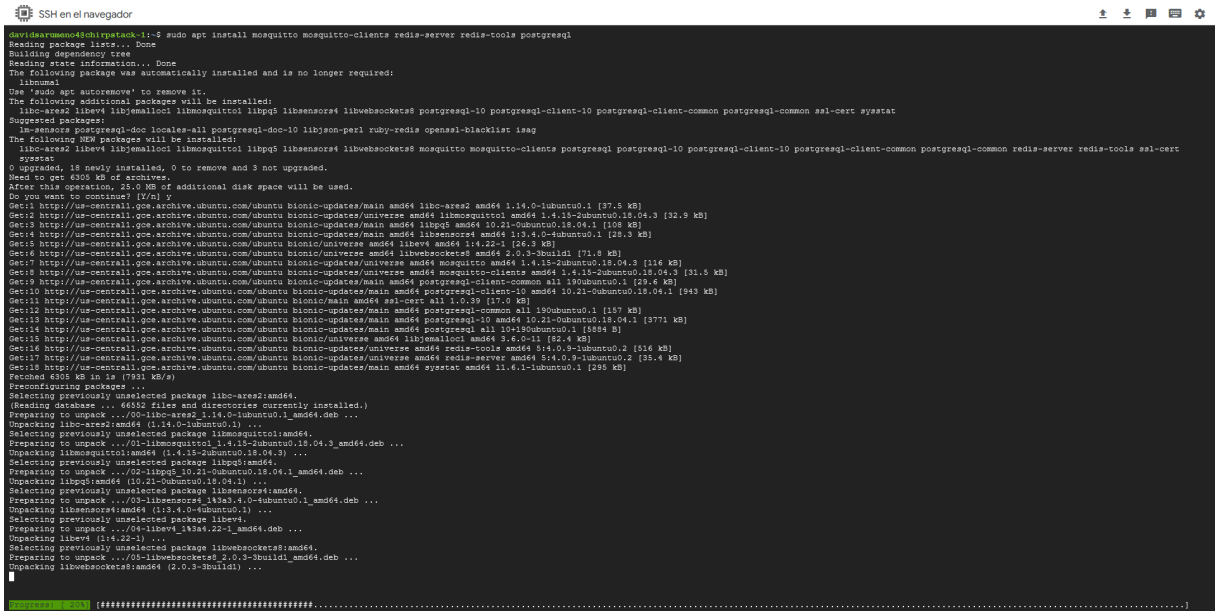


Figura 38. *Instalación de MQTT, Redis y PostgreSQL.*

- **Creación de bases de datos, roles y extensiones en PostgreSQL.**

En este punto se crean roles para el Application Server y Network Server de ChirpStack, estos roles irán anidados a las bases de datos de cada una de estas, tanto el Application como el Network server tienen su propia bases de datos donde enviaran información con respecto a perfiles, gateways, dispositivos, etc.

Además de extensiones como *pgtrgm*, que permitirá crear índices basados en trigrams, que es una secuencia de tres caracteres consecutivos que podemos encontrar en una cadena de texto, y otra extensión llamada *hstore*, el cual ayudara a almacenar conjuntos

de pares clave/valor dentro de un único valor de PostgreSQL, que puede ser útil en varios escenarios, como filas con muchos atributos que rara vez se examinan o datos semi estructurados.

Para realizar la conexión con la base de datos postgres se digitará el siguiente comando *sudo -u postgres psql*, posteriormente se creará las bases de datos como se aprecia en la Fig.39, con sus respectivos con sus roles y extensiones especificadas anteriormente.

Figura 39. *Configuración de base da datos PostgreSQL.*

- **Configuración de repositorio de software de ChirpStack.**

El servidor de ChirpStack proporcionará un repositorio que es compatible con el sistema de paquetes apt de Ubuntu, para ello se necesita instalar *dirmngr* y *apt-transport-https* como se puede observar en la Fig.40.

Figura 40. *Instalación de dirmngr y apt-transport-https.*

Como se ilustra en la Fig.41, una vez instalado se deberá configurar una clave para este nuevo repositorio de la siguiente manera:

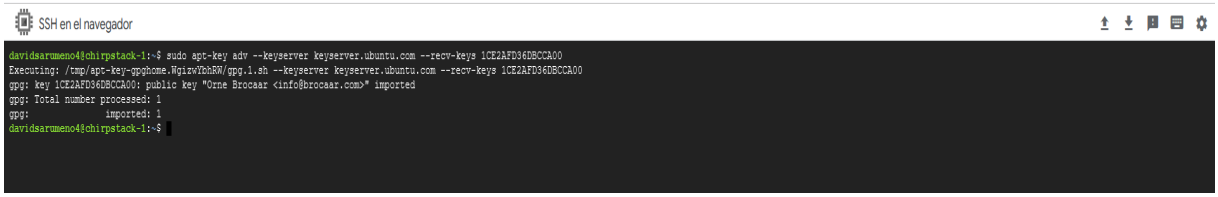


Figura 41. *Configuración de clave para nuevo repositorio*.

Según en la siguiente en la Fig.42, se deberá agregar el repositorio a la lista de repositorios creando un nuevo archivo.

Figura 42. *Agregación de repositorio a lista de repositorios*.

Por último se deberá hacer un update para actualizar el caché del paquete apt como lo indica en la Fig.43.

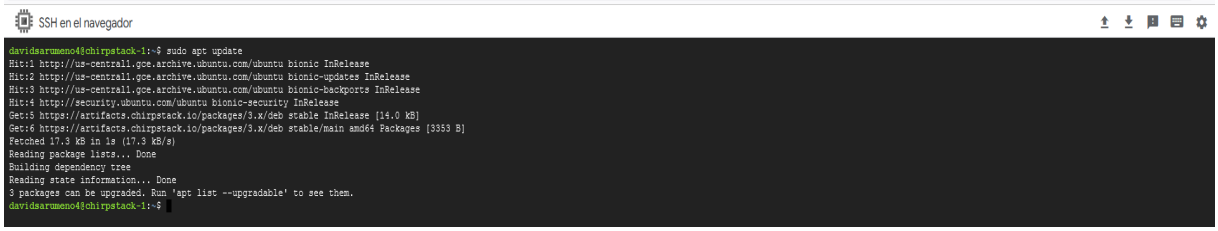


Figura 43. *Actualización de caché del paquete apt*.

- **Instalación de ChirpStack Gateway Bridge.**

ChirpStack Gateway Bridge es un servicio que convierte los protocolos LoRa (Packet Forwarder) en un formato de datos común de ChirpStack Network Server (JSON y Proto-

buf). Este componente es parte de la pila de servidor de red LoRaWAN de código abierto de ChirpStack.

Para instalar este servicio se digitará el comando *sudo apt install chirpstack-gateway-bridge*, se puede visualizar en la Fig.44.

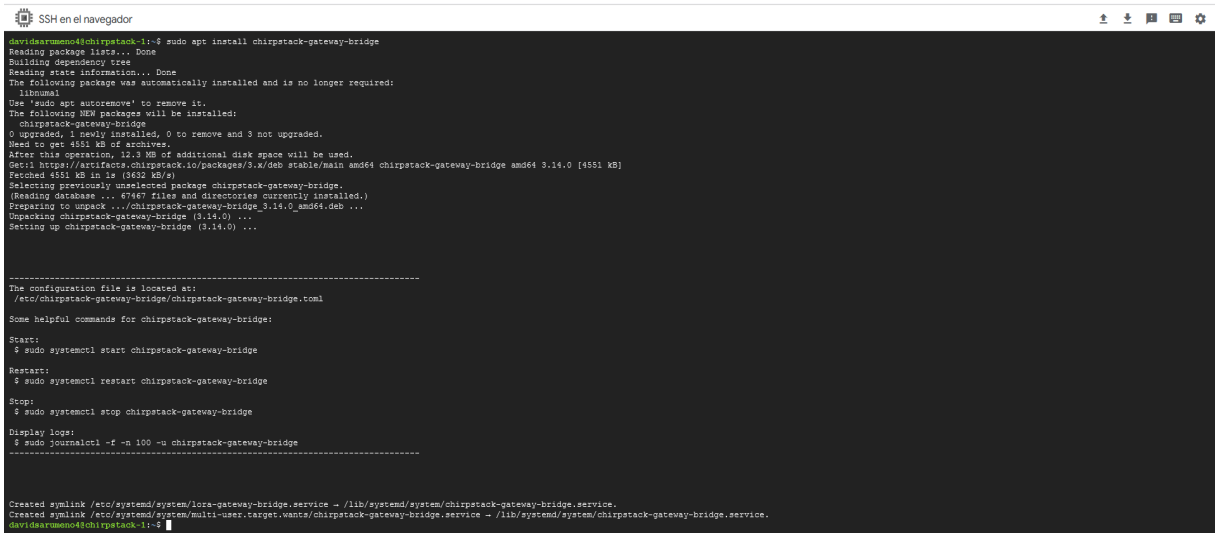


Figura 44. *Instalación de Chirpstack Gateway Bridge*.

En la Fig.45, el archivo de configuración de Gateway Bridge se encuentra en */etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml*; sin embargo, la configuración predeterminada es suficiente, ahora solo quedará habilitar e iniciar el servicio.

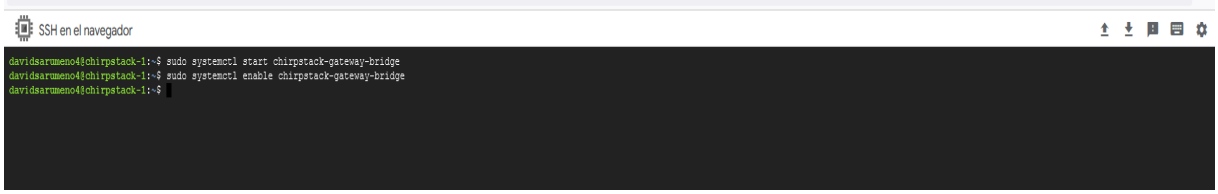


Figura 45. *Iniciar ChirpStack Gateway Bridge*.

- **Instalación de ChirpStack Network Server.**

La función principal de ChirpStack Network Server es la duplicación de las tramas LoRaWAN recibidas por las puertas de enlace LoRa y las tramas recopiladas que manejan:

- Autenticación.
- LoRaWAN capa mac y comandos mac.
- Comunicación con el servidor de aplicaciones ChirpStack.
- Programación de tramas de enlace descendente.

Como se observa en la Fig.46, para instalar este servicio se digitará el comando *sudo apt install chirpstack-network-server*.

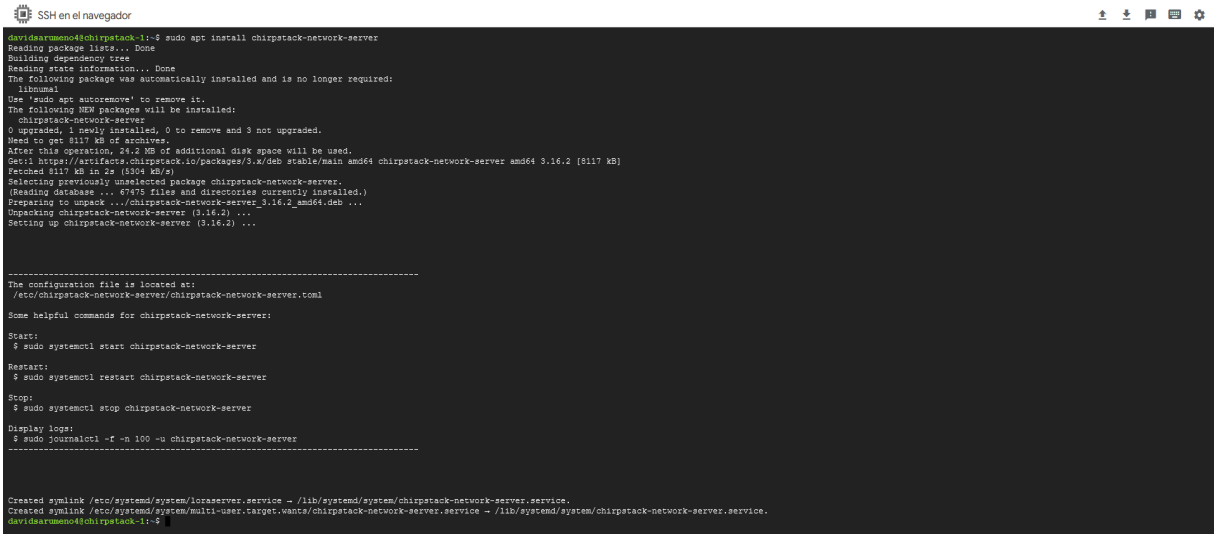


Figura 46. *Instalación de Chirpstack Network Server*.

El archivo de configuración de Gateway Bridge se encuentra en */etc/chirpstack-network-server/chirpstack-network-server.toml*; en la Fig.47 en la que se debe actualizar para que coincida con la base de datos y la configuración de la banda. Primeramente, se habilitará e iniciará el servicio.

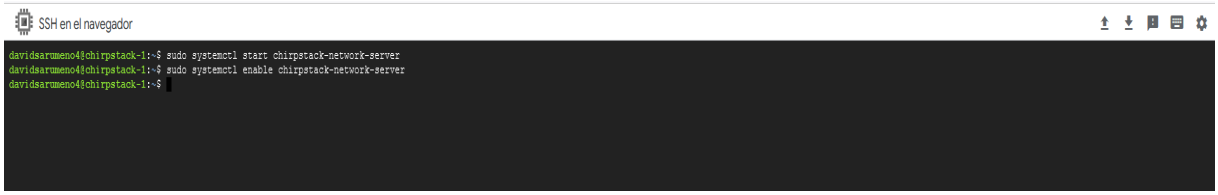


Figura 47. *Iniciar ChirpStack Network Server.*

Ahora se abrirá el archivo de configuración de Network Server en la que se deberá especificar la base de datos que se creó anteriormente, como también la banda de conexión, como lo indica en la Fig.48.

Figura 48. *Archivo de configuración de ChirpStack Network Server.*

Se deberá buscar la línea de dsn, en donde se especificará la base de datos de Network Server con su respectiva contraseña de conexión, como se puede observar en la Fig.49:

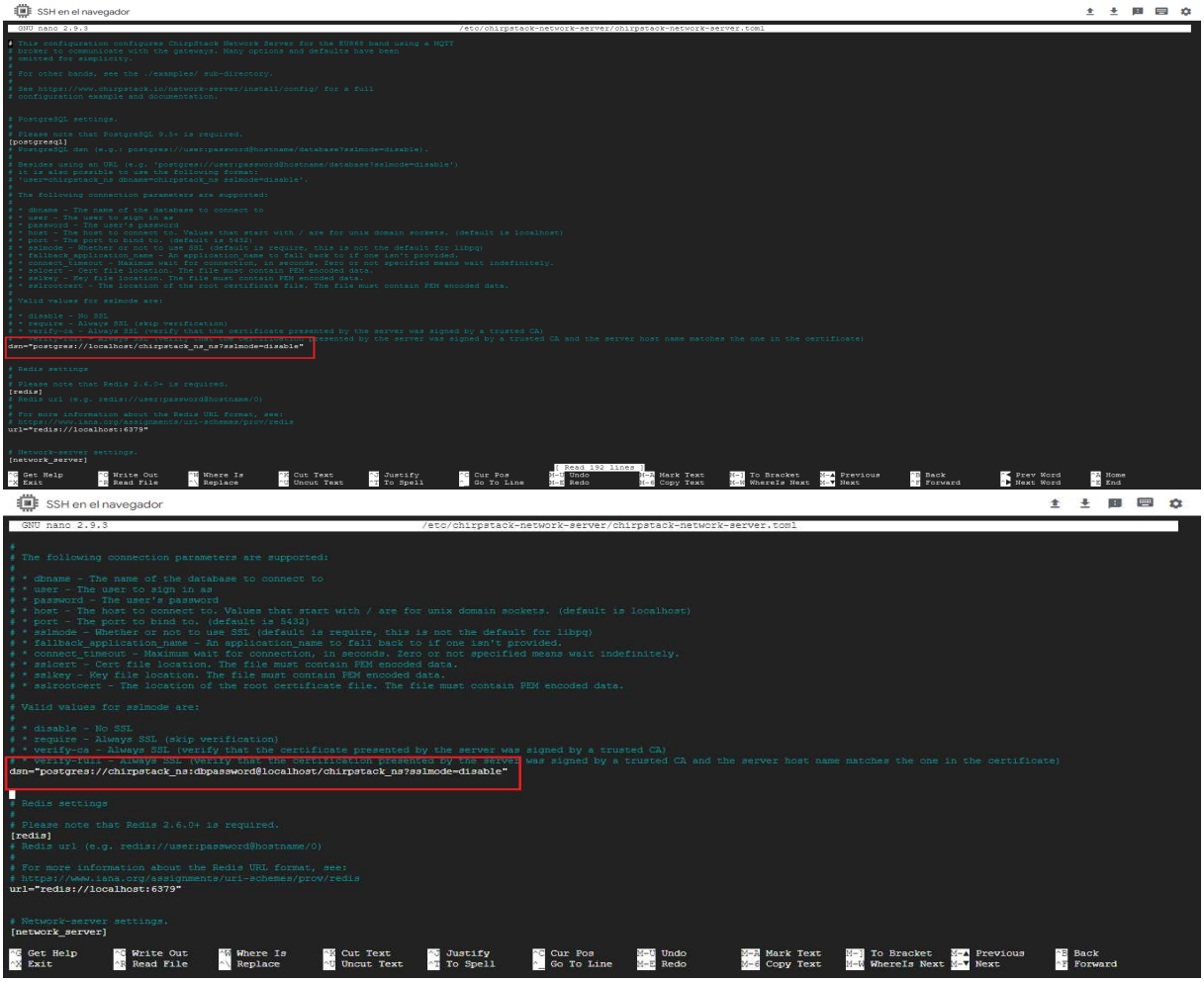


Figura 49. *Configuración de base de datos en archivo de Network Server.*

Ahora se deberá dirigir a la configuración de banda en donde se cambiara por la banda US915, además, se cambiara los canales de uplink, de la siguiente manera en la Fig.50:

Figura 50. *Configuración bandas en archivo de Network Server.*

Por último, como se observa en la Fig.51 se guardará las configuraciones con Ctrl+O, y se saldrá del mismo con Ctrl+X, para finalizar se deberá reiniciar el servicio de ChirpStack Network Server.

Figura 51. *Reinicio de ChirpStack Network Server.*

- **Instalación de ChirpStack Application Server.**

ChirpStack Application Server es el responsable del "inventario" del dispositivo de una infraestructura de red LoRaWAN, el manejo de la solicitud de unión y el manejo y el cifrado de las cargas útiles de la aplicación.

Ofrece una interfaz web donde se pueden administrar usuarios, organizaciones, aplicaciones y dispositivos. Para la integración con servicios externos, ofrece gRPC y API RESTful. Los datos del dispositivo se pueden enviar y/o recibir a través de MQTT, HTTP.

El archivo de configuración de Application Server se encuentra en */etc/chirpstack-application-server/chirpstack-application-server.toml*; en la que se debe actualizar para

que coincida con la base de datos creada anteriormente, además de una clave secreta que se generara automáticamente con un código. Como se aprecia en la Fig.52, para instalar este servicio se digitará el comando *sudo apt install chirpstack-application-server*.

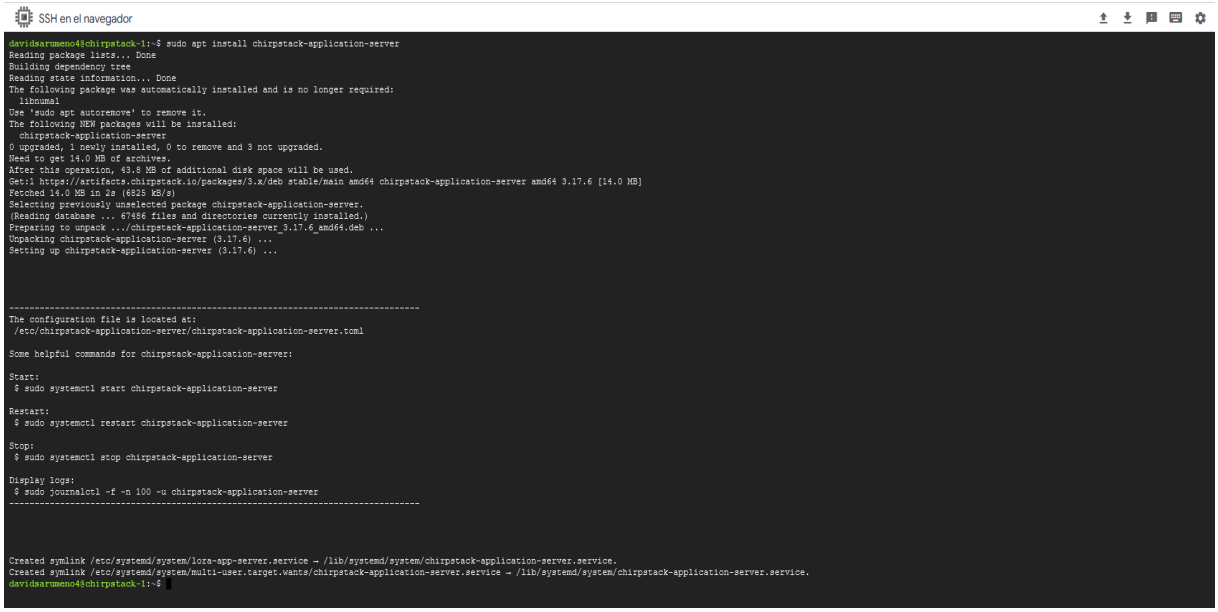


Figura 52. *Instalación de ChirpStack Application Server.*

Ahora se abrirá el archivo de configuración de Application Server en la que se deberá especificar la base de datos que se creó anteriormente para ello se dirigirá a la ruta especificada del archivo de configuración como lo indica en la Fig.53.

Figura 53. *Archivo de configuración de ChirpStack Application Server.*

Como se hizo en la Fig.54 en la configuración de Network Server, se dirigirá a la línea de dsn, y se especificara la base de datos en este caso la base de datos de Application Server.

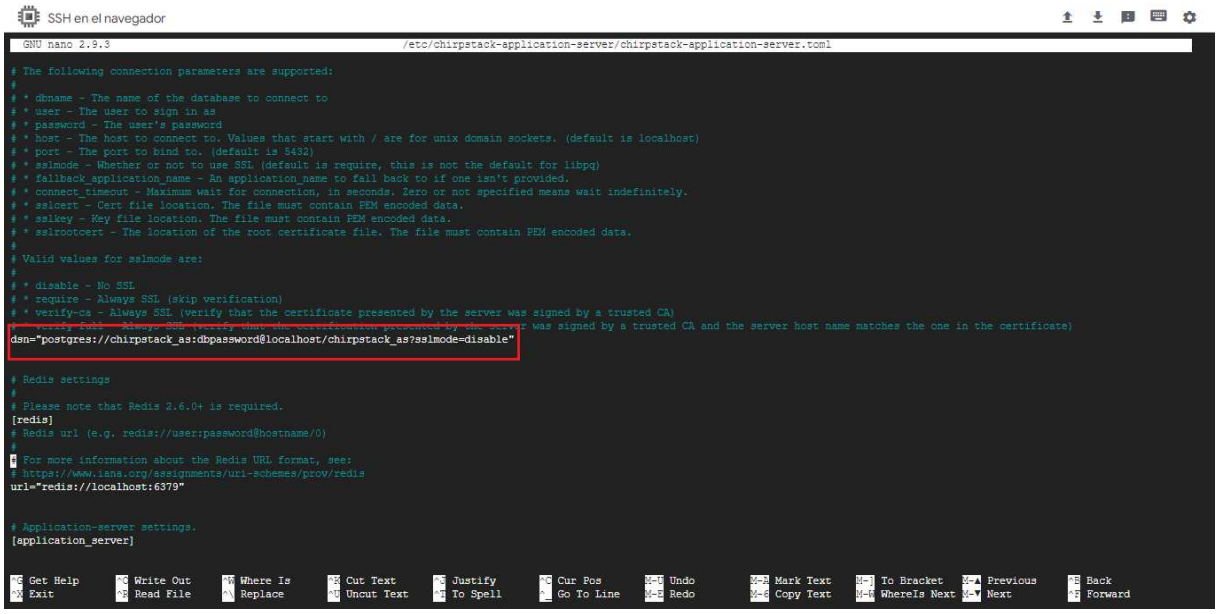


Figura 54. *Configuración de base de datos de Application Server.*

Ahora en el mismo archivo se dirigirá a la línea de jwt secret, en donde se deberá ingresar una clave que se generara mediante un comando, como se visualiza en la Fig.55.

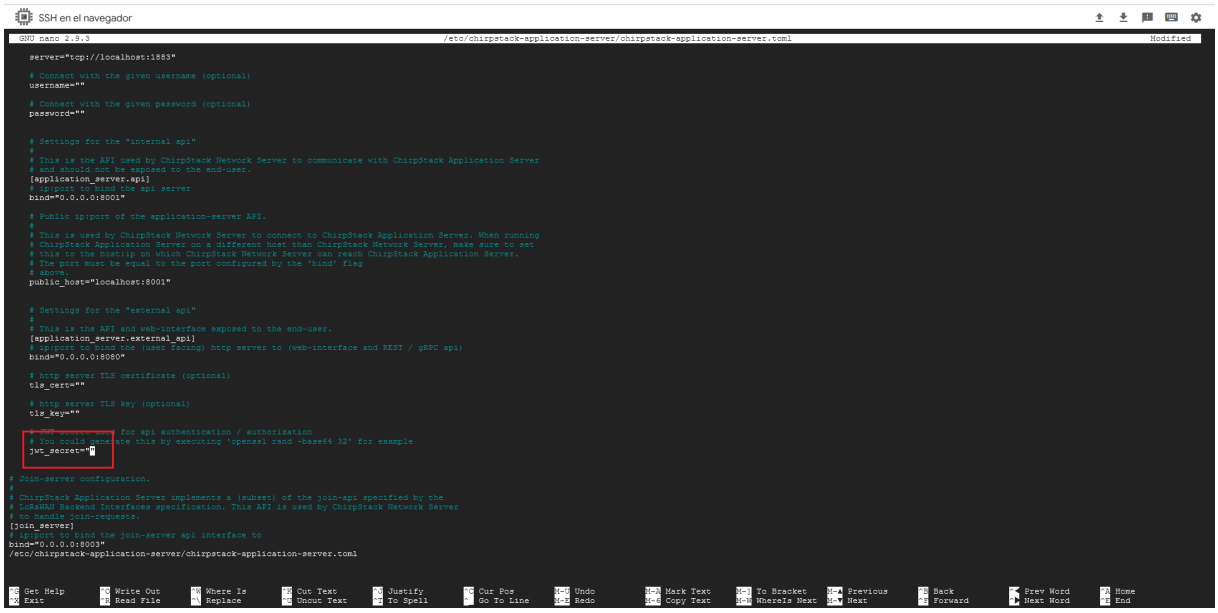


Figura 55. *Configuración de jwt secret de Application Server.*

Como se distingue en la Fig.56 Guardamos la configuración del archivo y salimos del mismo para poder generar la clave mediante el comando *openssl rand -base64 32*, aquella clave generada automáticamente como se visualiza en la Fig.57 se deberá copiar e ingresarla en la línea de jwt secret en el archivo de configuración de ChirpStack Application Server.

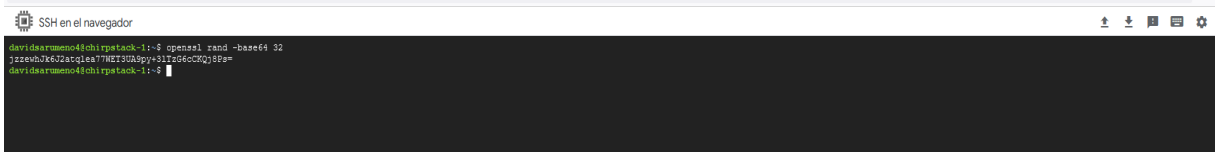


Figura 56. *Generación automática de clave para jwt secret.*

Figura 57. *Configuración de clave secreta de jwt secret en Application Server.*

Una vez ingresado la clave jwt secret, se procederá a guardar y salir del archivo de configuración, por último se habilitara e iniciara el servicio de la siguiente Fig.58:

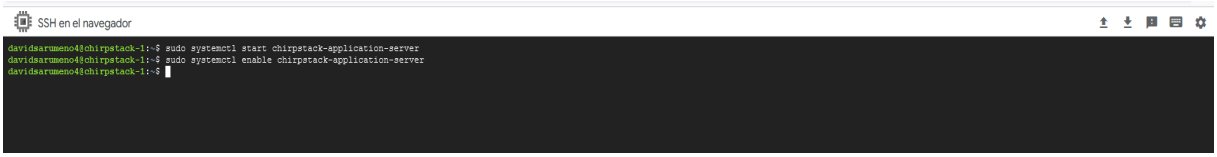


Figura 58. *Inicio de Application Server.*

Ahora se podrá iniciar el servidor de ChirpStack en cualquier navegador, en la siguiente Fig.59 el servicio se iniciará con la IP externa que se reservó en la creación de la VM de Google Cloud y mediante el puerto 8080 que es el puerto de conexión a ChirpStack. El usuario y contraseña de conexión es admin/admin, posteriormente se podrá cambiar de contraseña del servidor, como visualizamos en la Fig.60 de la ventana principal.

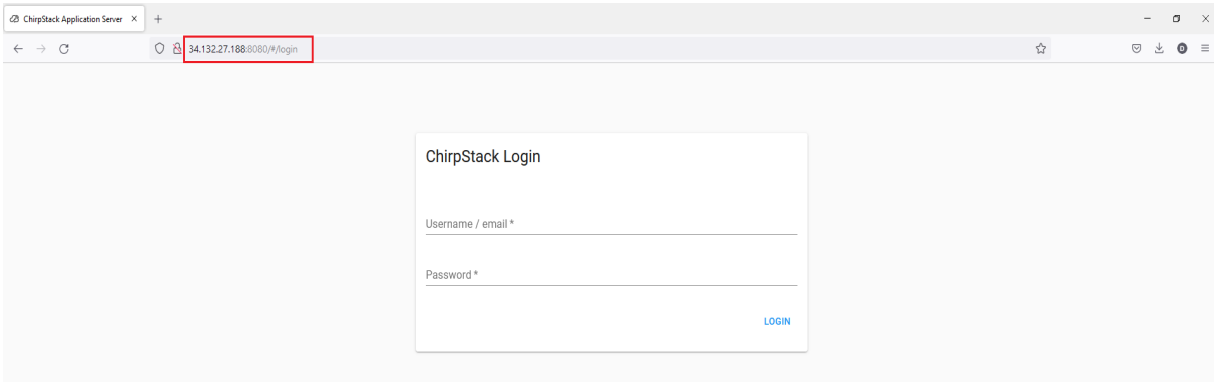


Figura 59. *Inicio de ChirpStack.*

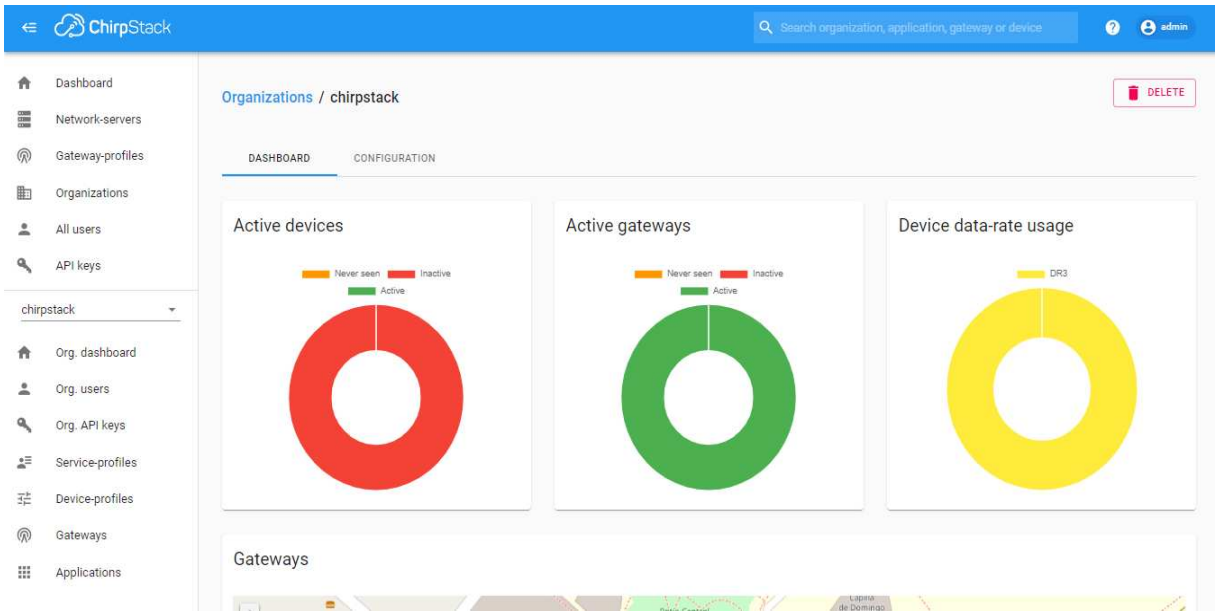


Figura 60. *Ventana principal de ChirpStack.*

5.4.2.2 Creación de base de datos para almacenar los datos que envíen los dispositivos hacia el servidor de ChirpStack

Para la recepción de los datos de la LoRa al Chirpstack y del medidor de agua ultrasónico se creó una base de datos llamada chirpstack_as_events en donde se quedara registrado todos los datos para el uso de gráficas que se creara en node red. En la siguiente Fig.61 vemos el nombre de la base de datos creado:

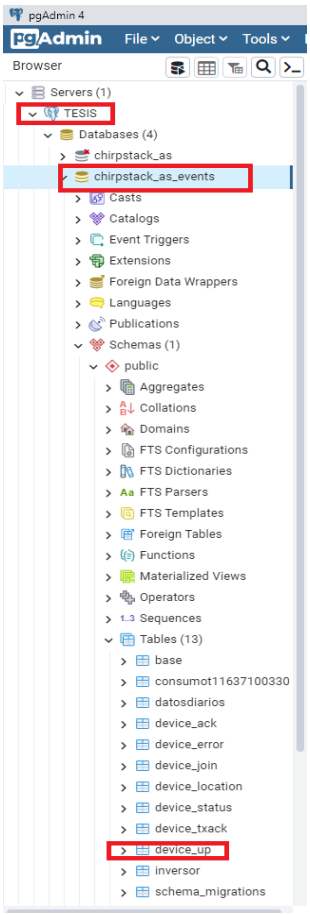


Figura 61. *Base de Datos Creada*.

Una vez que se ha creado la base de datos se crea automáticamente el nombre de la tabla, en este caso se creó con device_up, en esta tabla se quedara registrado todos los datos que sea enviado por la LoRa y por el medidor dentro de esa tabla tiene un número de propiedades como son: *id, received_at, dev_eui, device_name, applitacion_id, etc* son propiedades que se crea conjuntamente con la base dependiendo de que datos se encuentre registrado en el chirpstack, ya que esos datos son enviados a través de la LoRa y del medidor como lo indica en la Fig.62.

	id [PK] uuid	received_at timestamp with time zone	dev_eui bytea	device_name character varying (100)	application_id bigint	application_name character varying (100)	frequency bigint	dr smallint	adr boolean	f_cnt bigint	f_port smallint	tags hstore	data bytea	rx_info jsonb	object jsonb
1	7950528f...	2022-07-06 11:34:55.622...	[binary da...	T11637100330	2	LoraTesis	903300000	0	true	3	2		[binary da...	[{"name": ...	{"Id": 2, "l...
2	2cf51fe7-...	2022-07-06 11:40:08.316...	[binary da...	T11637100330	2	LoraTesis	903100000	3	true	5	2		[binary da...	[{"name": ...	{"Id": 2, "l...
3	9b3f09eb...	2022-07-06 11:45:10.320...	[binary da...	T11637100330	2	LoraTesis	902900000	3	true	6	2		[binary da...	[{"name": ...	{"Id": 2, "l...
4	718e5943...	2022-07-06 11:55:14.323...	[binary da...	T11637100330	2	LoraTesis	903500000	3	true	8	2		[binary da...	[{"name": ...	{"Id": 2, "l...
5	067876ff-...	2022-07-06 12:00:16.328...	[binary da...	T11637100330	2	LoraTesis	903100000	3	true	9	2		[binary da...	[{"name": ...	{"Id": 2, "l...
6	bb5ae311...	2022-07-06 12:05:18.332...	[binary da...	T11637100330	2	LoraTesis	902900000	3	true	10	2		[binary da...	[{"name": ...	{"Id": 2, "l...
7	5b5a3ce9...	2022-07-06 12:15:22.335...	[binary da...	T11637100330	2	LoraTesis	903300000	3	true	12	2		[binary da...	[{"name": ...	{"Id": 2, "l...
8	ed610a6f...	2022-07-06 12:20:24.336...	[binary da...	T11637100330	2	LoraTesis	902900000	3	true	13	2		[binary da...	[{"name": ...	{"Id": 2, "l...
9	5a383e9b...	2022-07-06 12:25:26.341...	[binary da...	T11637100330	2	LoraTesis	902300000	3	true	14	2		[binary da...	[{"name": ...	{"Id": 2, "l...
10	5f580c20...	2022-07-06 12:40:32.348...	[binary da...	T11637100330	2	LoraTesis	902500000	3	true	17	2		[binary da...	[{"name": ...	{"Id": 2, "l...
11	5cf690a3...	2022-07-06 14:13:49.401...	[binary da...	T11637100330	2	LoraTesis	903300000	3	true	36	2		[binary da...	[{"name": ...	{"Id": 2, "l...
12	edcf1f58-...	2022-07-06 14:18:51.402...	[binary da...	T11637100330	2	LoraTesis	902500000	3	true	37	2		[binary da...	[{"name": ...	{"Id": 2, "l...
13	17f653b0...	2022-07-06 14:23:53.406...	[binary da...	T11637100330	2	LoraTesis	902300000	3	true	38	2		[binary da...	[{"name": ...	{"Id": 2, "l...
14	ebfdfa1a-...	2022-07-06 14:28:55.409...	[binary da...	T11637100330	2	LoraTesis	903100000	3	true	39	2		[binary da...	[{"name": ...	{"Id": 2, "l...
15	0b111ec3...	2022-07-06 14:33:57.414...	[binary da...	T11637100330	2	LoraTesis	903100000	3	true	40	2		[binary da...	[{"name": ...	{"Id": 2, "l...
16	4b4e3cbe...	2022-07-06 14:44:12.418...	[binary da...	T11637100330	2	LoraTesis	903700000	3	true	43	2		[binary da...	[{"name": ...	{"Id": 2, "l...
17	275272aa...	2022-07-06 15:09:22.433...	[binary da...	T11637100330	2	LoraTesis	902300000	3	true	48	2		[binary da...	[{"name": ...	{"Id": 2, "l...
18	c0e012f2...	2022-07-06 15:19:26.438...	[binary da...	T11637100330	2	LoraTesis	902300000	3	true	50	2		[binary da...	[{"name": ...	{"Id": 2, "l...
19	3749dafd...	2022-07-06 15:24:28.439...	[binary da...	T11637100330	2	LoraTesis	903100000	3	true	51	2		[binary da...	[{"name": ...	{"Id": 2, "l...
20	ff08ba9e-...	2022-07-06 15:34:43.445...	[binary da...	T11637100330	2	LoraTesis	903500000	3	true	54	2		[binary da...	[{"name": ...	{"Id": 2, "l...
21	04e2ee15...	2022-07-06 15:39:45.447...	[binary da...	T11637100330	2	LoraTesis	902500000	3	true	55	2		[binary da...	[{"name": ...	{"Id": 2, "l...
22	14f9ddfa-...	2022-07-06 15:44:47.451...	[binary da...	T11637100330	2	LoraTesis	902900000	3	true	56	2		[binary da...	[{"name": ...	{"Id": 2, "l...
23	4b27891...	2022-07-07 10:57:00.149...	[binary da...	T11637100330	2	LoraTesis	903100000	0	true	4	2		[binary da...	[{"name": ...	{"Id": 2, "l...

Figura 62. *Propiedades de la Base creada por Chirpstack_as_events*.

Se procederá a crear la base de datos se escribirá el siguiente comando: *sudo -u postgres psql*, esto permitirá ingresar a la base de datos de postgres mediante consola como se ilustra en la Fig.63.

Figura 63. *Comando para ingresar a la Base de Datos*.

Como se observa en la Fig.64,se ingresara a la base de datos, se procederá a colocar el siguiente comando *create role chirpstack_as_events with login password 'dbpassword';* que permitirá crear un rol, este rol permitirá crear el usuario, con este usuario se podrá acceder a la base de datos *chirpstack_as_events* cuando esté creada.

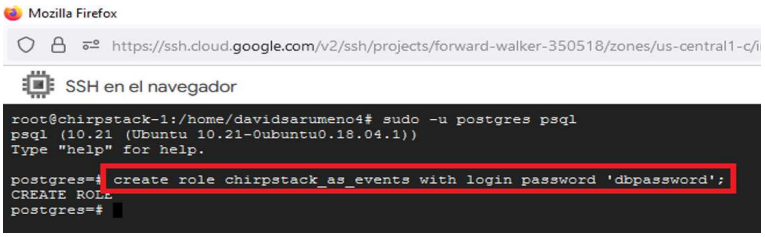


Figura 64. *Creación del Usuario Chirpstack_as_events*.

Como siguiente punto, en la Fig.65 se creara la base de datos *chirpstack_as_envets* con la siguiente línea de comando: *create database chirpstack_as_events with owner chirpstack_as_events;*, a esta base de datos se le pasará como usuario al usuario creado en el paso anterior.

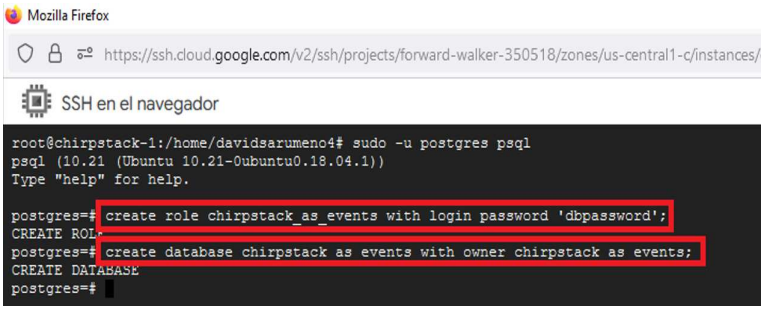


Figura 65. *Creación de la Base creada Chirpstack_as_events*.

Se le dará todos los privilegios tanto al usuario como la base de datos *chirpstack_as_events*, como lo indica en la Fig.66, esto significa que los privilegios son de un superusuario para evitar estar dando permisos a otros usuarios que ya tiene el mismo privilegio, solo que los otros usuario si cuenta con el mismo privilegio ellos pueden manipular la base de datos creada.

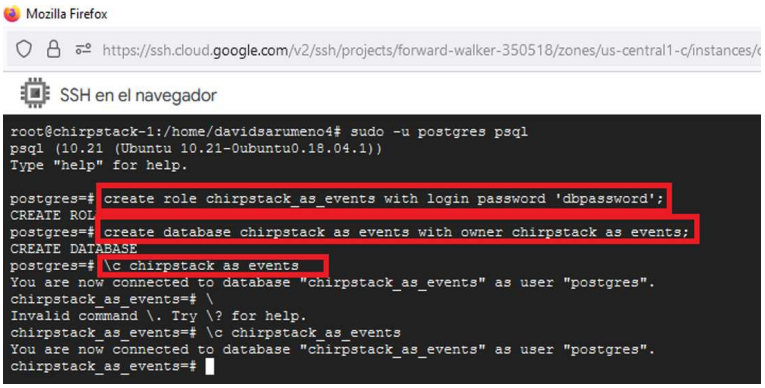


Figura 66. *Privilegios a Chirpstack_as_events.*

Se creará una extensión con la siguiente línea de código: *create extension hstore;* que permite almacenar varios conjuntos de valores dentro de un valor único, esto es muy necesario, ya que existen filas o columnas con atributos que se examina o tienen datos que están semi estructurado, para luego salir de la terminal del psql se procede a colocar la siguiente línea \q como se aprecia en la Fig.67.

Figura 67. *Almacenar conjuntos de valores en Chirpstack_as_events.*

Como se contempla en la Fig.68 se procederá a conectar a la base de datos para verificar si el usuario y la base de datos ha sido creada de manera satisfactoria con la siguiente línea de comando: *psql -h localhost -U chirpstack_as_events -W chirpstack_as_events*.

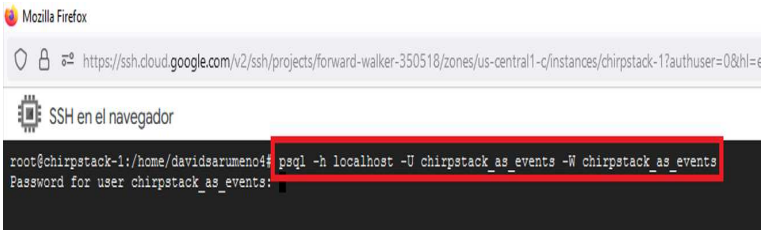


Figura 68. *Verificación de Usuario y Contraseña en Chirpstack_as_events*.

Una vez ejecutado el comando anterior se pedirá que se ingrese el password o contraseña que ha sido creada en el rol del usuario, como se ilustra en la Fig.69, una vez ingresado de manera correcta se procederá a conectar a la base de datos del PostgreSQL específicamente a la base que se creó en los pasos anteriores *chirpstack_as_events*.

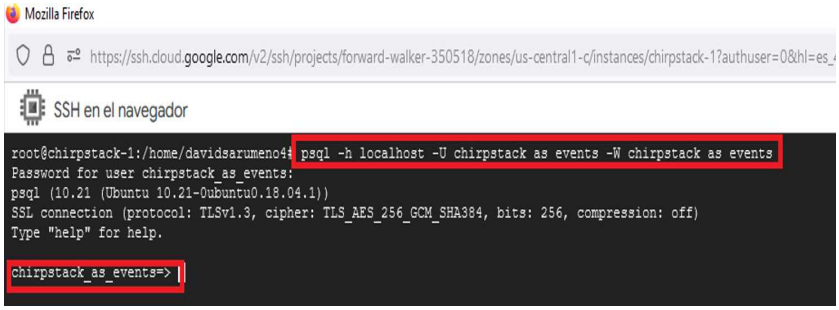


Figura 69. *Verificación de Datos correctos en Chirpstack_as_events*.

Se ingresará a la configuración de la siguiente línea de comando: *nano /etc/chirpstack-application-server/chirpstack-application-server.toml* , este archivo de configuración permitirá que ChirpStack comience a escribir los datos de eventos en la Base de Datos de Postgres, como puede divisar en la Fig.70.

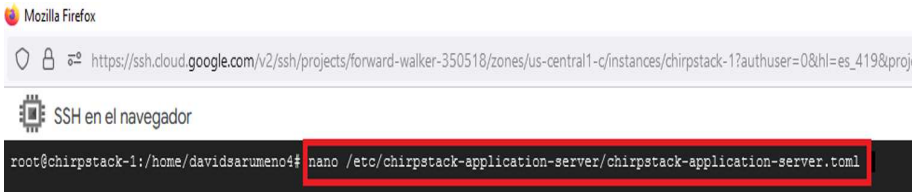


Figura 70. *Configuración de archivo nano /etc/chirpstack-application-server/chirpstack-application-server.toml* .

Como se mencionó en el paso anterior se debe configurar el archivo de *nano /etc/chirpstack-application-server/chirpstack-application-server.toml*, se configura en la línea que dice:

 # Enable integrations

enabled="mqtt", "postgresql"

y debajo de Post se colocará las siguientes líneas:

[application_server.integration.postgresql]

dsn="postgres://chirpstack_as_events:dbpassword@localhost/chirpstack_as_events?sslmode=disable"

luego se procede a guardar los datos configurados para posteriormente verificar si estos datos se encuentra ya registrados en la base de datos, como se nota en la Fig.71.

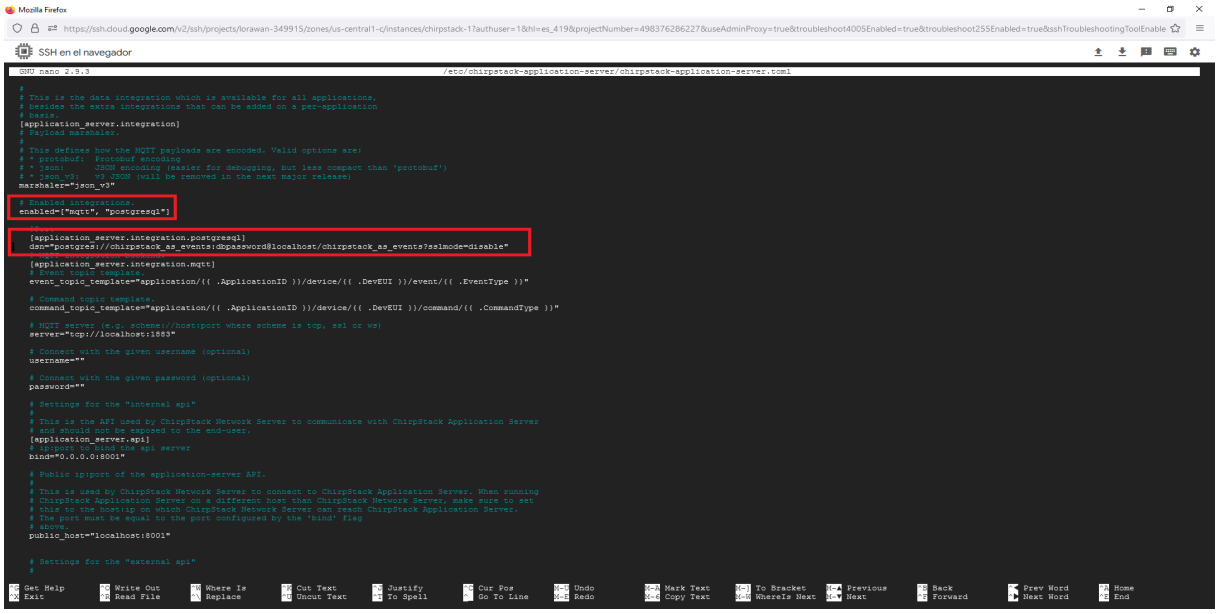


Figura 71. *Configuración de archivo nano.*

5.4.2.3 Cambio de zona horaria de base de datos ChirpStack en PostgreSQL

Cuando se instaló PostgreSQL, la zona horaria estaba configurado de otro país, se procedió a seguir unos pasos de configuración, cuando se instala PostgreSQL viene con un archivo de configuración que trae configurado de otra zona.

Se colocará la siguiente línea para entrar al archivo de configuración: *nano /etc/postgresql/10/main/postgresql.conf* como se verifica en la Fig.72.

Figura 72. *Archivo de configuración nano.*

En la siguiente Fig.73, se buscará la línea de timezone y se cambiara por otra zona horaria en este caso de Ecuador, la cual seria "America/Guayaquil", se actualizara la zona horaria y así se tendrá la hora correspondiente a esa zona horaria.

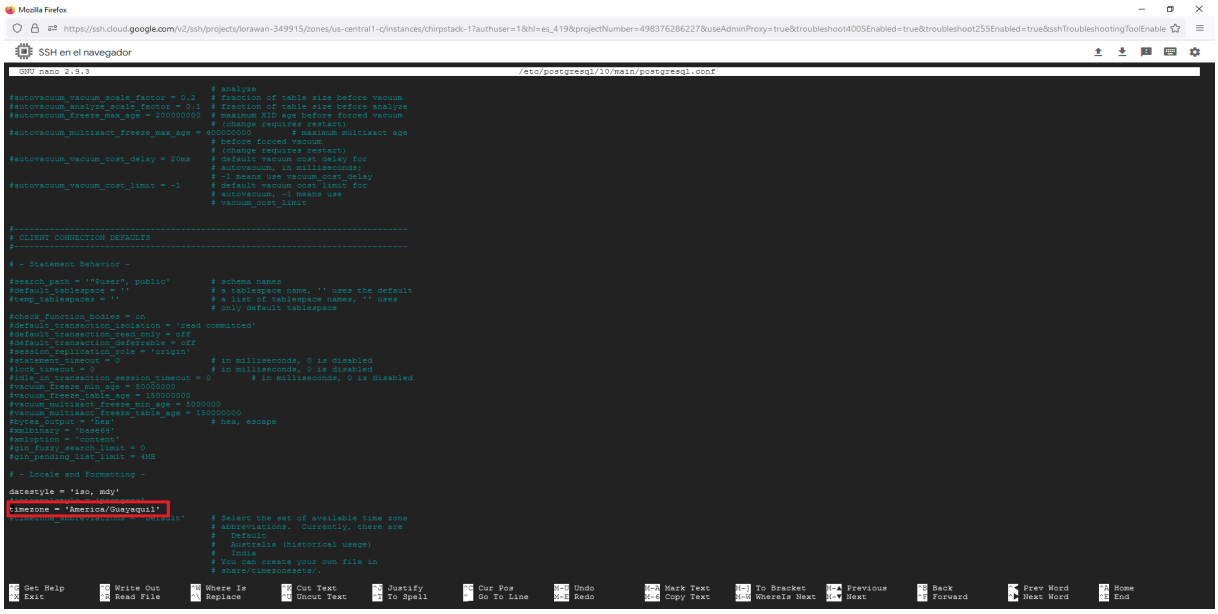


Figura 73. *Cambio de la zona horaria.*

Así mismo, como se observa en la Fig.74, se buscara en el mismo archivo long_timezone y se cambiara por el país que corresponde, porque es la fecha y hora que se guardara en los logs, para verificar la zona horaria que nos compete se podrá ingresar a https://www.php.net/manual/es/timezones.america.php, y se podrá verificar la zona horaria para PostgreSQL.

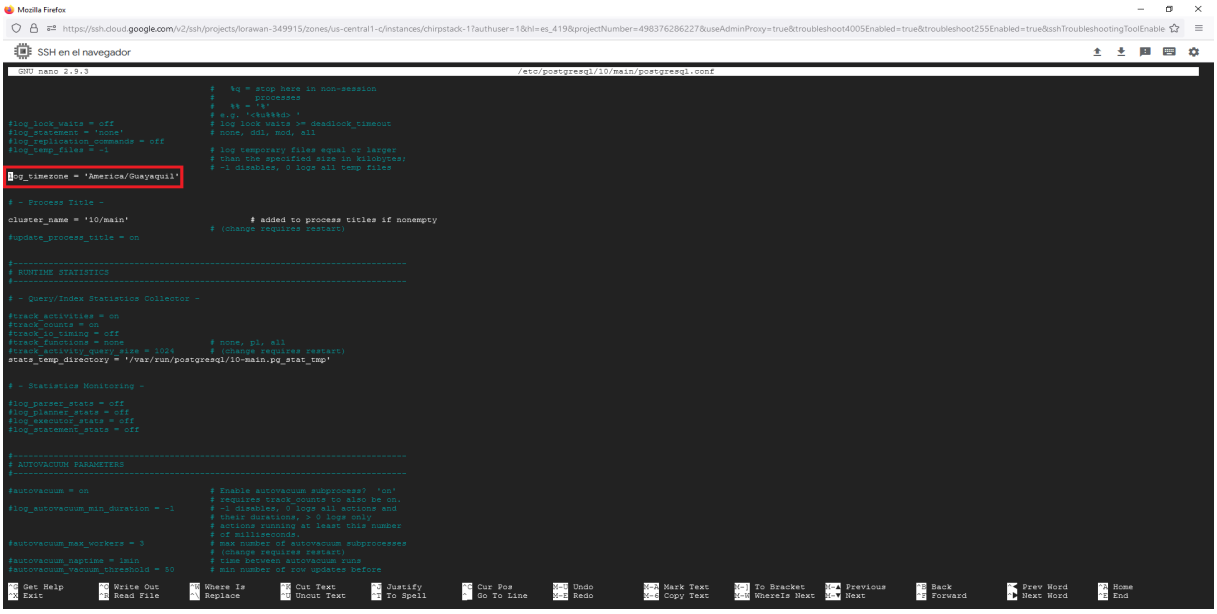


Figura 74. *Cambio de la zona horaria long_timezone*.

Después de realizar los cambios se procede al reinicio de PostgreSQL para qué sé que restablezca los servicios de servidor y tener un cambio de zona horaria, para el reinicio se coloca: */etc/init.d/postgresql restart* , como indica en la Fig.75.

Figura 75. *Reiniciar el servicio de PostgreSQL.*

Para comprobar que el cambio se ha realizado con éxito, se entrara en modo de cliente en PostgreSQL como se aprecia en la Fig.76.

Figura 76. *Ingreso a terminal de PostgreSQL.*

Se verificará la zona horaria con el comando: *show timezone;* como lo indica en la Fig.77.

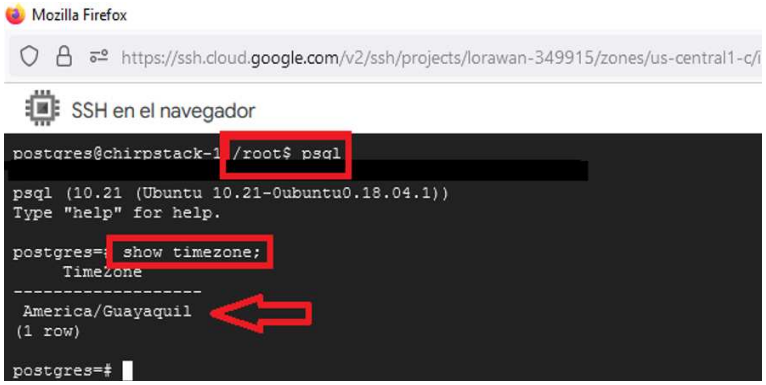


Figura 77. *Verificación de hora y fecha cambiada.*

Luego se ingresara a PgAdmin y se realizara una consulta con *select current_timestamp*, ahí se podrá observar si la hora y fecha corresponde a la nueva zona horaria que se ingresó en los archivos de configuración de PostgreSQL como se observa en la Fig.78.

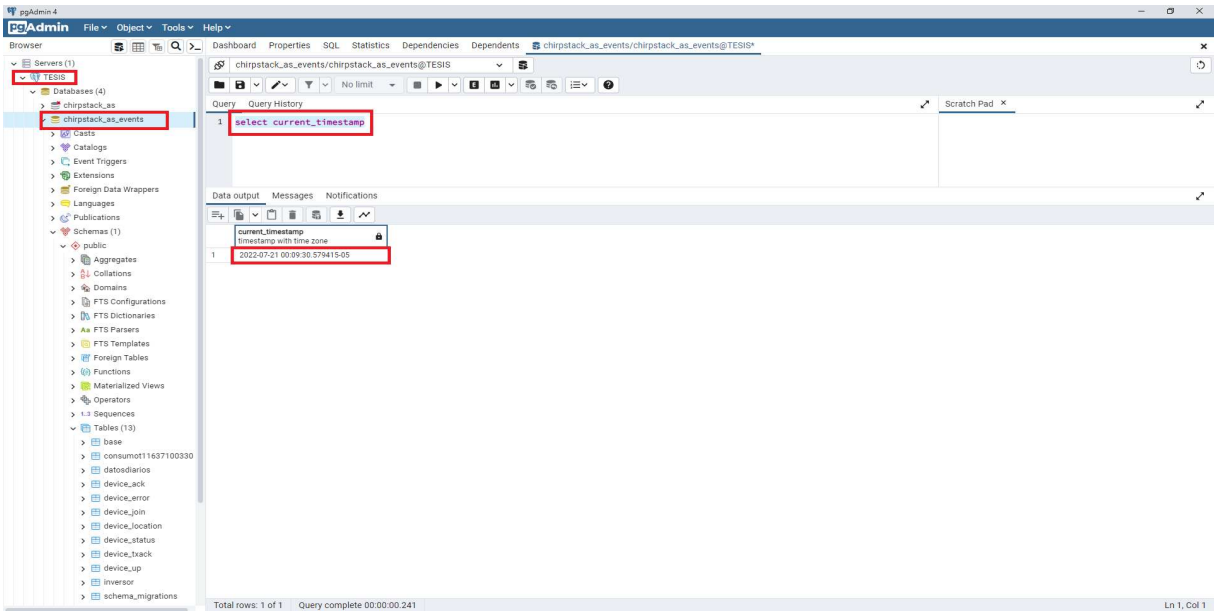


Figura 78. *select current_timestamp.*

Como se puede distinguir en la Fig.79, existen varios comandos para verificar hora y

fecha como por ejemplo: *show timezone;* que nos permite visualizar la región que hemos configurado en pasos anteriores.

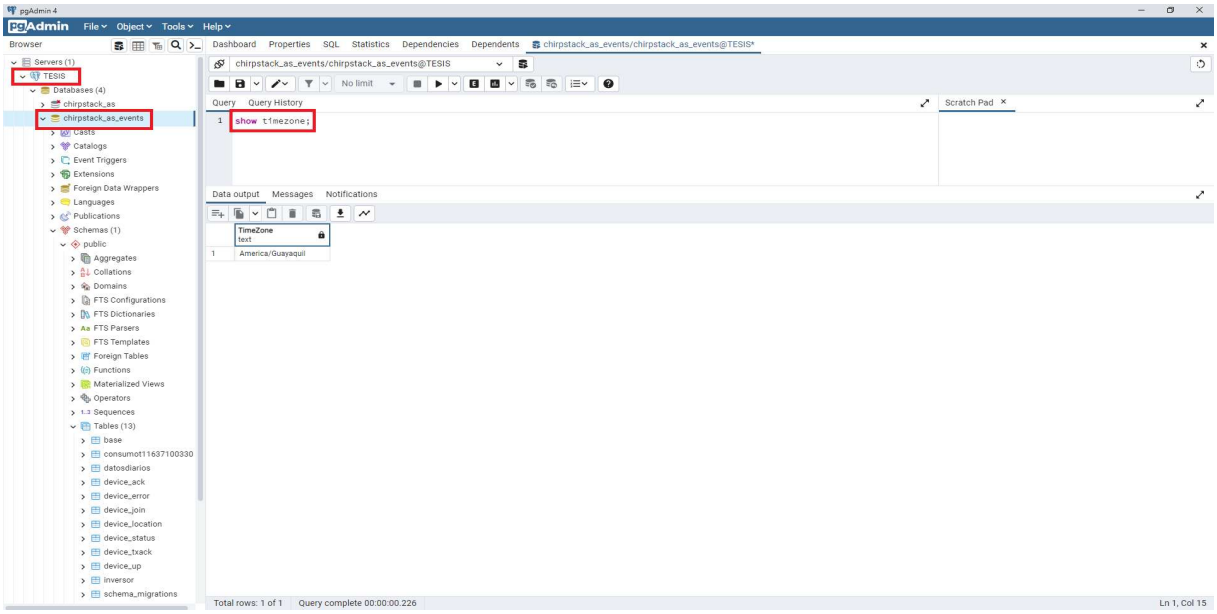


Figura 79. *show timezone*.

‘

5.4.2.4 Instalación de extensión PgAgent para tareas programadas en PostgreSQL

PgAgent Job servirá para realizar tareas programadas mediante scripts, este componente ayudará para crear, insertar, actualizar, eliminar datos de una base de datos. Para instalar se digitará el comando *sudo apt-get -y install pgAgent*, como se visualiza en la Fig.80.

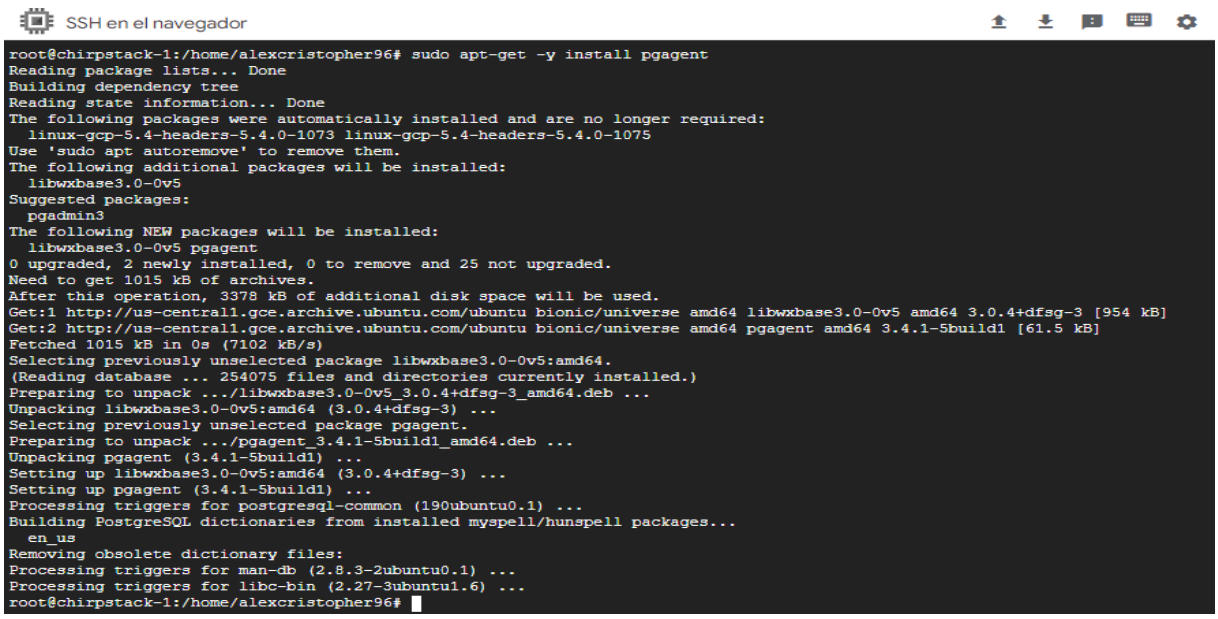


Figura 80. *Instalación de pgAgent.*

Una vez instalado se deberá agregar a la base de datos que se desea insertar esta extensión, en este caso se insertara a la base de datos de eventos antes creada, ya que en aquella base de datos llega toda la información correspondiente a los distintos dispositivos creados en el servidor de ChirpStack, y así poder realizar tareas automáticas en aquellos datos almacenados en aquella base de datos.

Para ello se deberá realizar la conexión a esa base de datos, esto lo realizaremos con el comando *"psql -h 34.132.27.188 -U chirpstack_as_events -d chirpstack_as_events"*, en este comando se especificará la IP de conexión la cual será la IP externa de la VM, además el nombre de la base de datos a la que se desea agregar esta extensión, posteriormente se pedirá la contraseña de conexión a la base de datos y se conectara a la misma.

Como se puede divisar en la Fig.81,cuando se realice la conexión a la base de datos, creamos la extensión de pgagent, una vez hecho esto se verificara en la aplicación PgAdmin de PostgreSQL la cual se podrá instalar localmente y conectarse de manera remota

con la base de datos creada en la VM de Google Cloud.

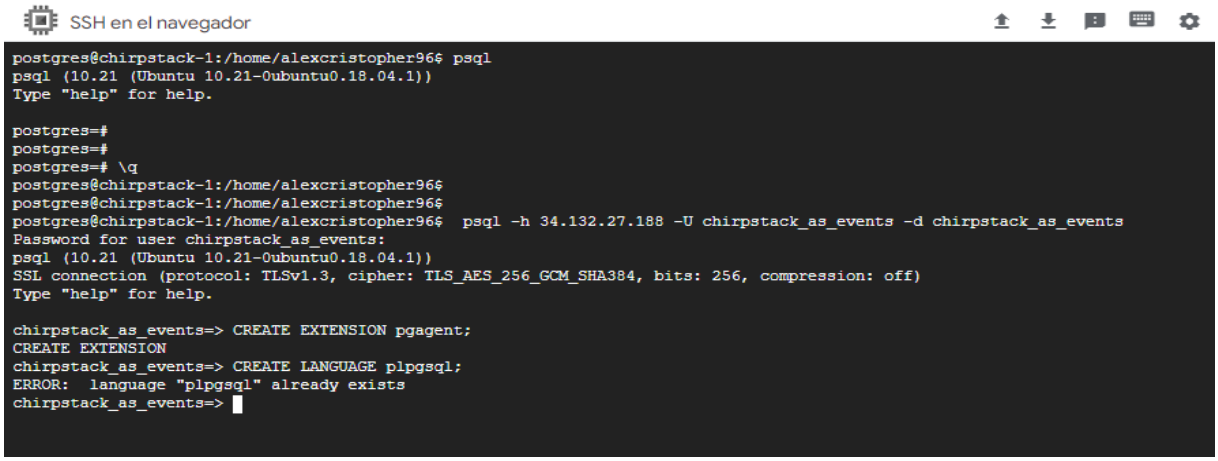


Figura 81. *Creación de extensión pgAgent en base de datos.*

5.4.2.5 Conexión remota a base de datos de ChirpStack

Para la visualización de datos del Network Server, Application Server y la de eventos se podrá realizar mediante la aplicación de PostgreSQL llamada PgAdmin, la cual se podrá conectar a las bases de datos de ChirpStack creadas anteriormente de manera remota esto con la IP externa de la VM y puerto de conexión de PostgreSQL.

Se deberá tener instalado PgAdmin en la máquina local, en la que se registrará un servidor de base de datos, para ello se dará clic derecho en servidores y registrar server, como se observa en la Fig.82.

Figura 82. *Creación servidor de base de datos.*

Como se puede apreciar en la Fig.83 se abrirá una ventana de configuración del servidor, primeramente en la pestaña de general se pondrá un nombre para el servidor.

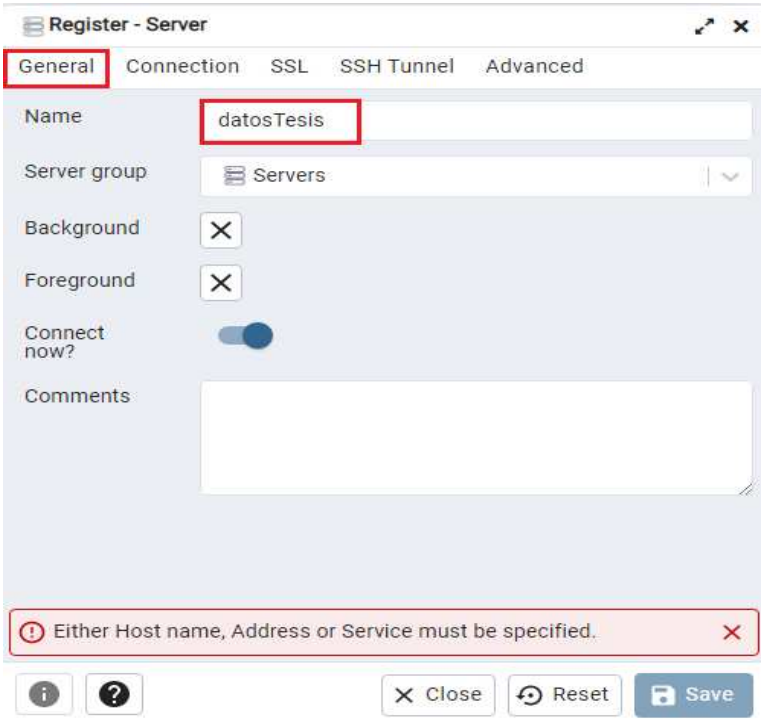


Figura 83. *Nombre del servidor de base de datos*.

En la siguiente pestaña de connection, se ingresará la IP de conexión a la base de datos en este caso la IP externa de la VM de Google Cloud y el puerto de conexión de postgres que sería el 5432, por último se ingresara el nombre de la base de datos a la cual se quiere conectar, aunque con la IP y el puerto de conexión se conectará a todas las bases de datos creadas en esa instancia de VM, se guarda y se conectara a las bases de datos creadas como se indica en la Fig.84, de los pasos realizados anteriormente.

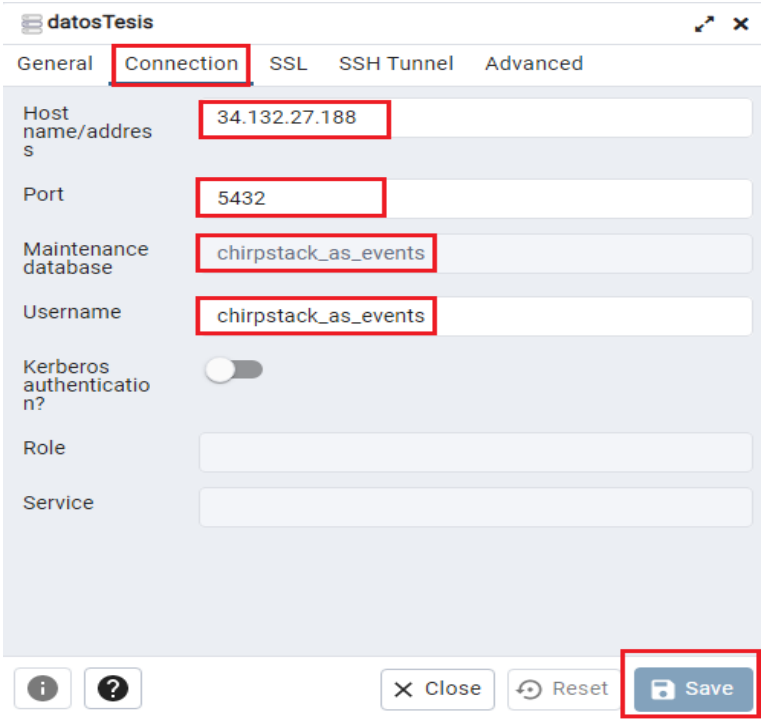


Figura 84. *Configuración de conexión remota a bases de datos de ChirpStack.*

Como se observa en la siguiente Fig.85, se conectó a todas las bases de datos que se creó anteriormente tanto la de Application Server, Network Server y la de Eventos, además se puede observar que se agregó la extensión de PgAgent Job.

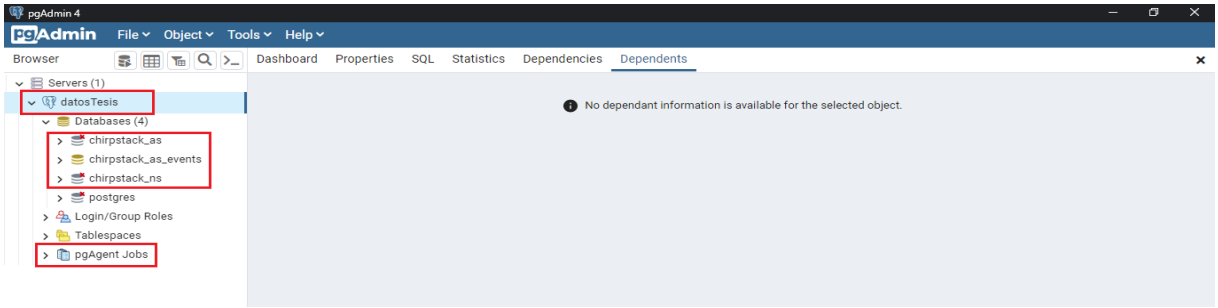


Figura 85. *Conexión satisfactoria a bases de datos de ChirpStack.*

5.4.2.6 Instalación de Node Red en VM de Google Cloud

Esta aplicación servirá para leer los datos que se guarden en la base de datos PostgreSQL de eventos del servidor ChirpStack, y así poder realizar un dashboard con los datos enviados a aquella base de datos para una mejor visualización de estos.

Como se ilustra en la siguiente Fig.86, se iniciará la VM de Google Cloud, y se digitará el comando *"bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)"* para la instalación de Node-Red. Este comando nos instalará paquetes de Node.js, npm, entre otros componentes que se necesitan para la ejecución de Node-Red.

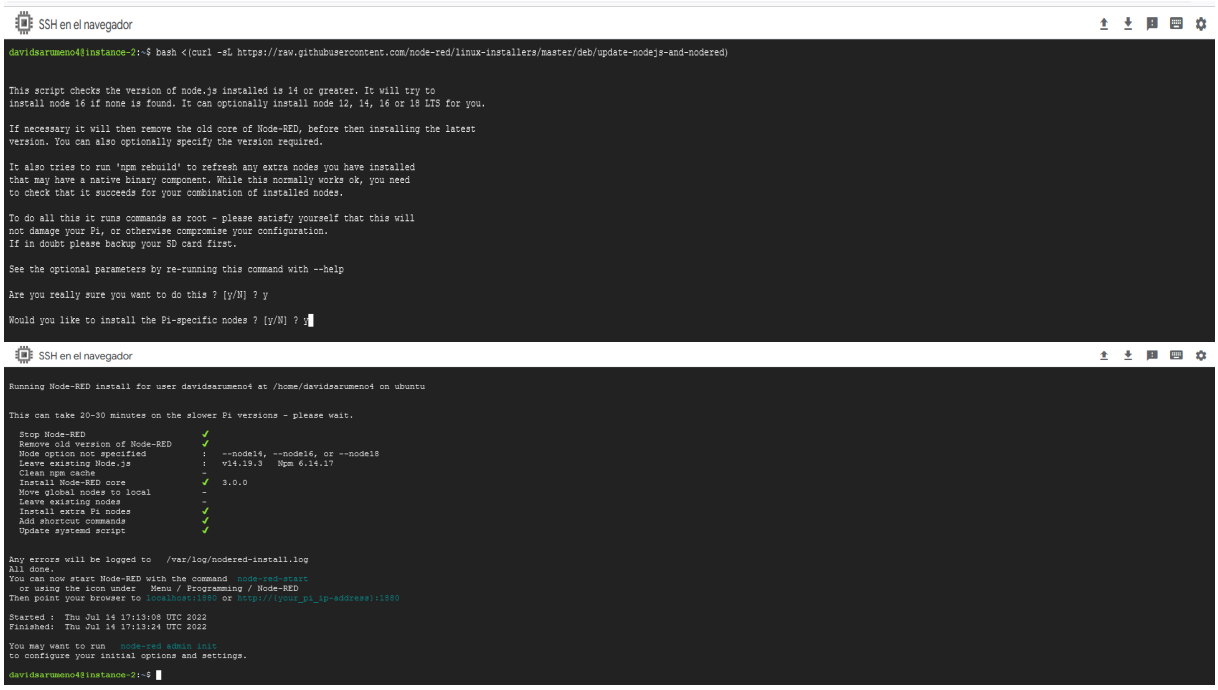


Figura 86. *Instalación de Node-Red*.

Una vez instalado, se podrá ejecutar simplemente al escribir node-red en la línea de comandos que está en la Fig.87.

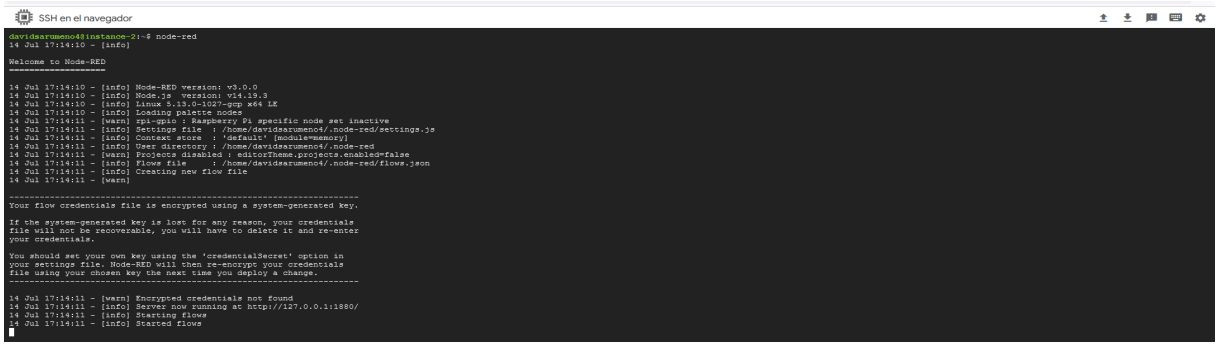


Figura 87. *Ejecución de Node-Red.*

Para poder observar la aplicación en el navegador, apreciar en la siguiente Fig.88, basta con escribir la IP externa de la VM de Google con el puerto 1880 que es el puerto de conexión a Node-Red.

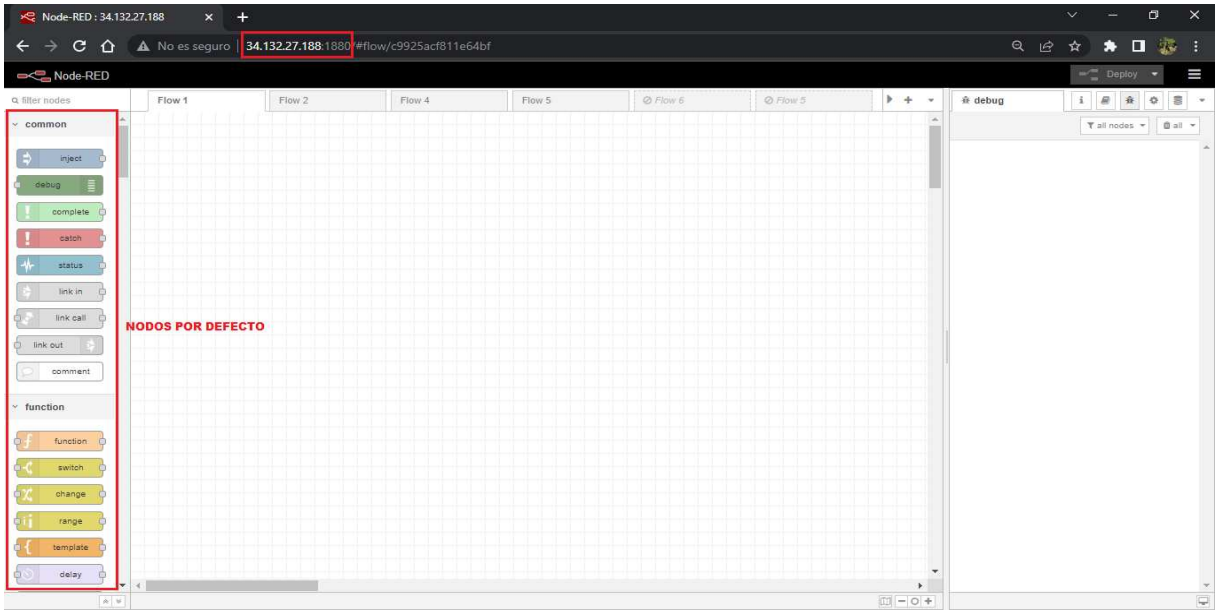


Figura 88. *Ventana de trabajo de Node-Red.*

5.4.2.7 Inicio Automático de Node Red

El inicio automático ayudará para que cada vez que se inicia la VM de Google, no se tendrá que ejecutar el comando node-red para iniciar el servicio, sino que automáticamente

el servicio se iniciara y podemos ingresar a él mediante la IP y el puerto de conexión.

Para ello se tendrá que instalar un paquete llamado pm2, el cual permite implementar demonios en aplicaciones para que puedan funcionar en segundo plano como servicios, para instalar este paquete se hará uso de npm el cual se instaló conjuntamente con node-red, como lo indica en la Fig.89.

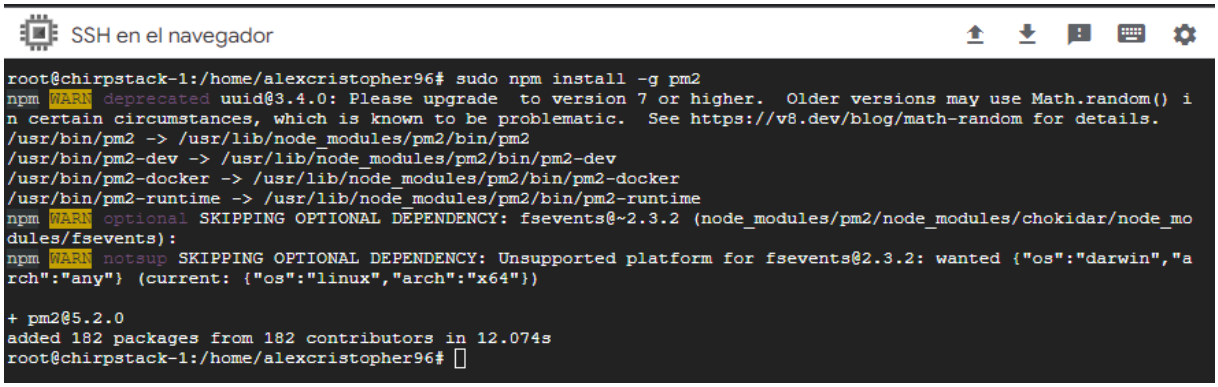


Figura 89. *Instalación de pm2*.

Una vez instalado el paquete pm2, se debe observar en la Fig.90, se indicara que node-red se ejecutara como un servicio en segundo plano, indicando la ruta y el archivo de ejecución de node red con el comando *pm2 start /usr/bin/node-red – -v*.

Figura 90. *Configuración de inicio automático node-red.*

Para verificar que node-red está online como servicio, se ejecutara *pm2 info node-red*, y se podrá observar en la Fig.91 que node-red está online y que cada vez que se inicie la VM se ejecutara automáticamente.

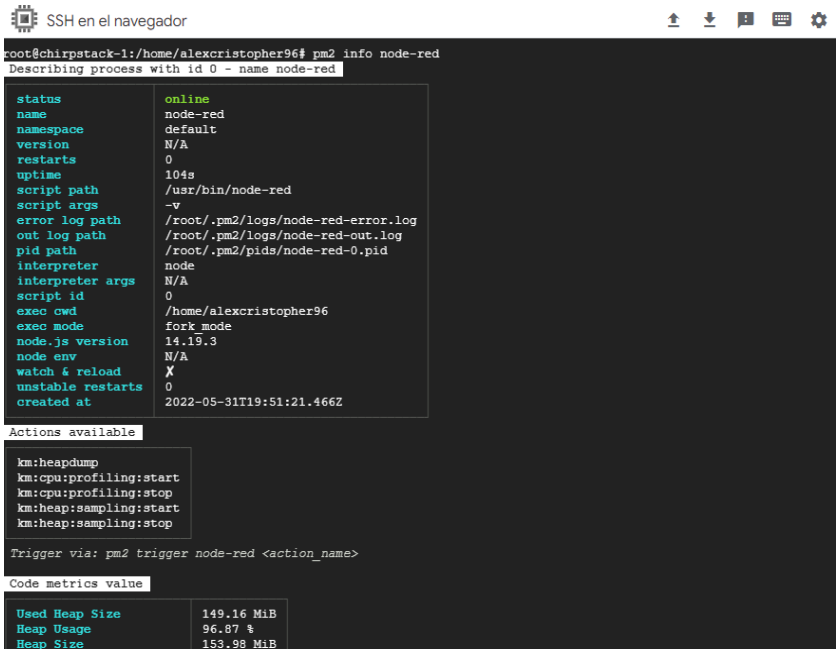


Figura 91. *Inicio automático node-red.*

Por último guardamos todos los procesos que realizamos con el paquete de pm2, como se aprecia en la Fig.92.

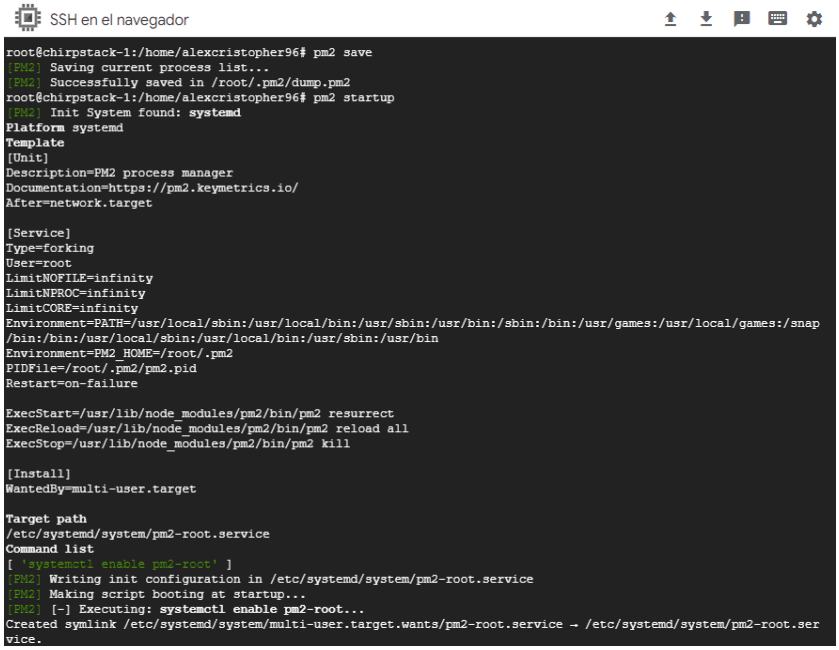


Figura 92. *Guardar cambios de pm2.*

5.5 Desarrollo de la red LoRaWAN para leer datos de dispositivos de ChirpStack y mostrarlos en dashboard de Node Red en tiempo real

5.5.1 Implementación de Gateway LoRa con el servidor ChirpStack

Primeramente, se debe conectar a la red del Gateway Mikrotik, una vez conectado se ingresará al menú de configuración mediante un navegador web, en el que se ingresara el gateway de la conexion, en seguida aparecerá una pantalla de logeo, en el que el usuario y contraseña son admin/admin como se aprecia en la Fig.93.

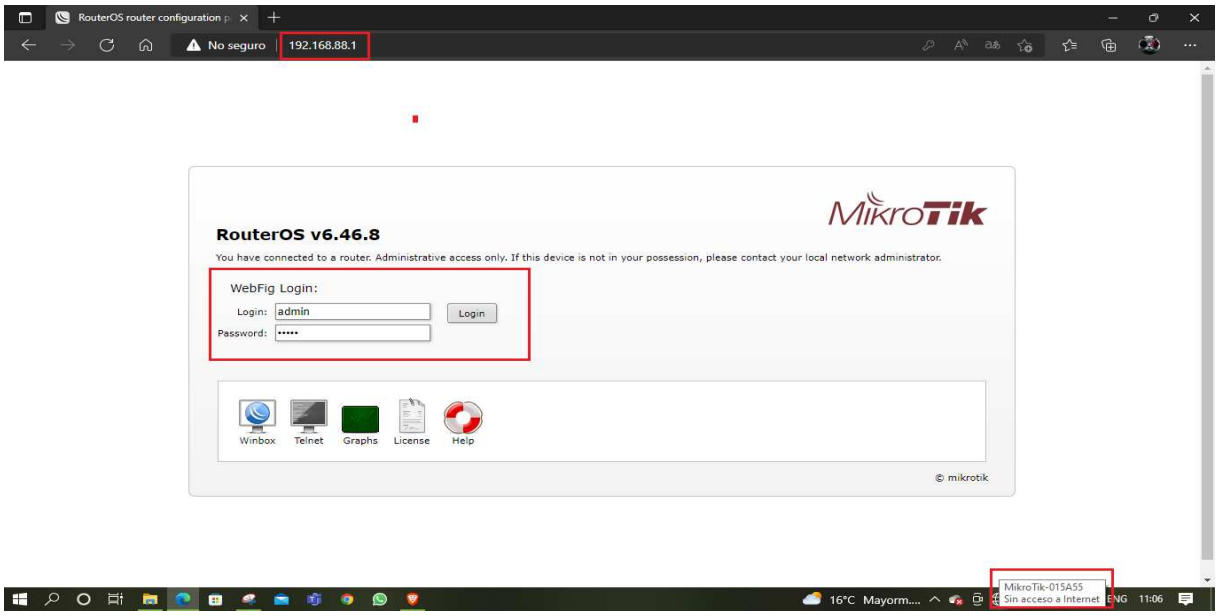


Figura 93. *Ingreso a pantalla login del gateway*.

Una vez ingresado a la ventana de configuración, se ingresará a la opción LoRA, ahí se puede agregar un servidor para poder conectarse con el Gateway, para registrar el servidor se pedirá el nombre, el cual será el Network Server del servidor de ChirpStack, este componente se creara posteriormente en el servidor, así que el nombre que se asigne

en el gateway será el mismo en el servidor. Luego se pedirá la IP de conexión al servidor, como se aprecia en la Fig.94, se creó el servidor en el gateway.

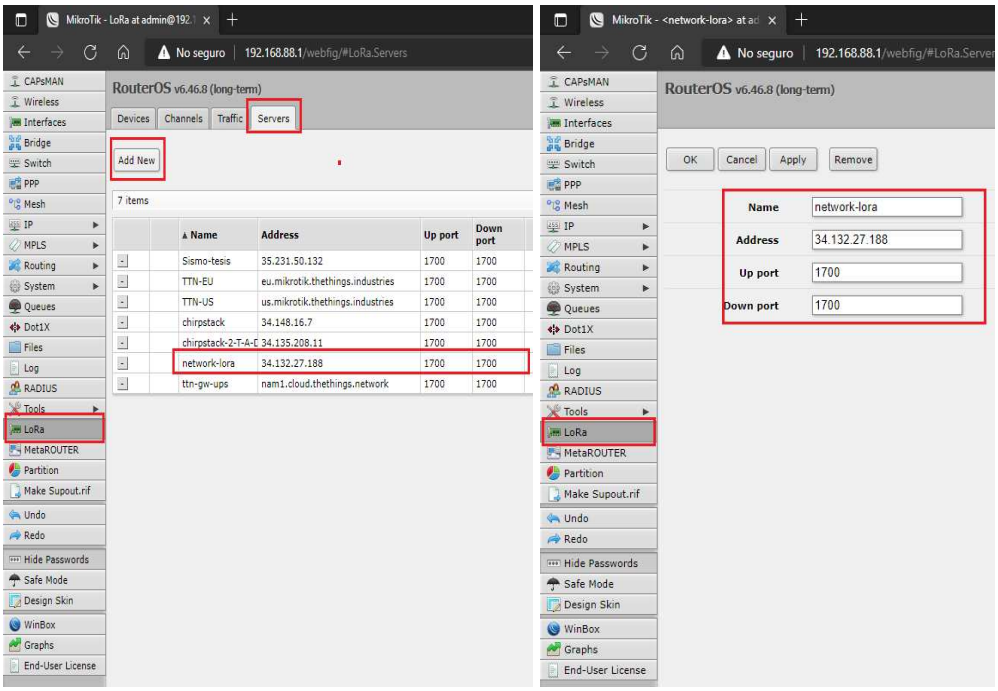


Figura 94. *Ingreso de servidor al gateway*.

Por último se deberá agregar el servidor creado en el gateway, a la lista de servidores que están en cola para conectarse con el gateway como se aprecia en la Fig.95.

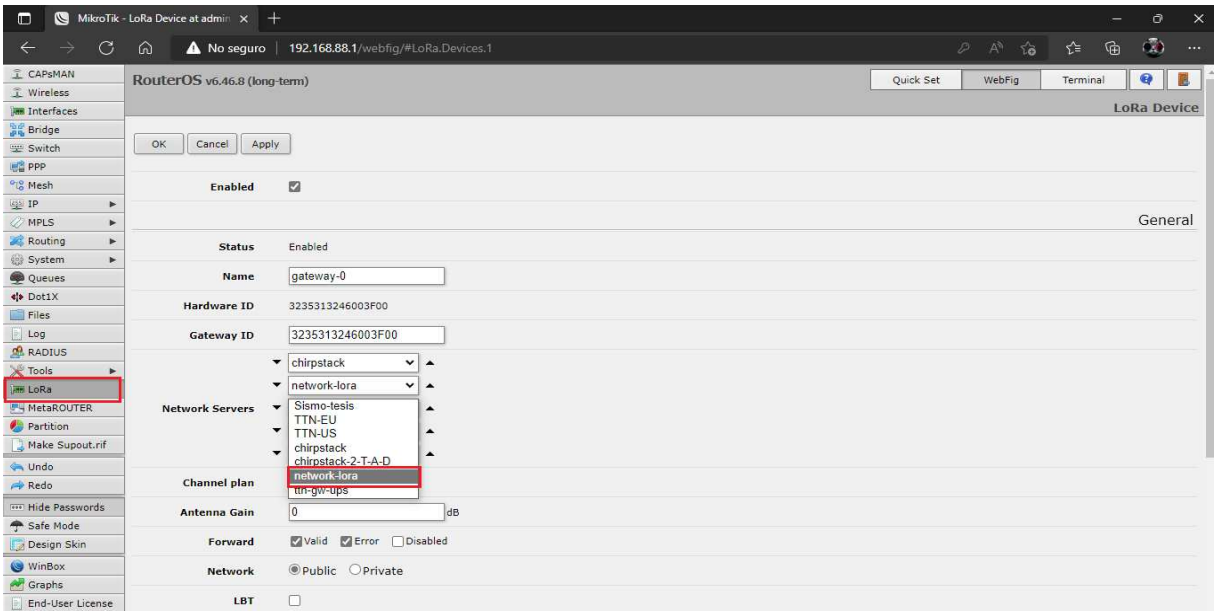


Figura 95. *Ingreso de servidor a cola de conexión con gateway.*

5.5.2 Creación y configuración de componentes en ChirpStack

Para insertar dispositivos y trasmitir datos en el servidor ChirpStack se debe seguir una secuencia de pasos.

5.5.2.1 Creación de Network Server

Primero se debe crear un servidor de red el cual corresponde al servidor de red instalado en el servidor como se aprecia en la Fig.96, para ello se dirigirá a la opción network servers en donde se digitará un nombre para el network server y la IP con el puerto de conexión a ChirpStack, una vez llenado estas opciones se guardara.

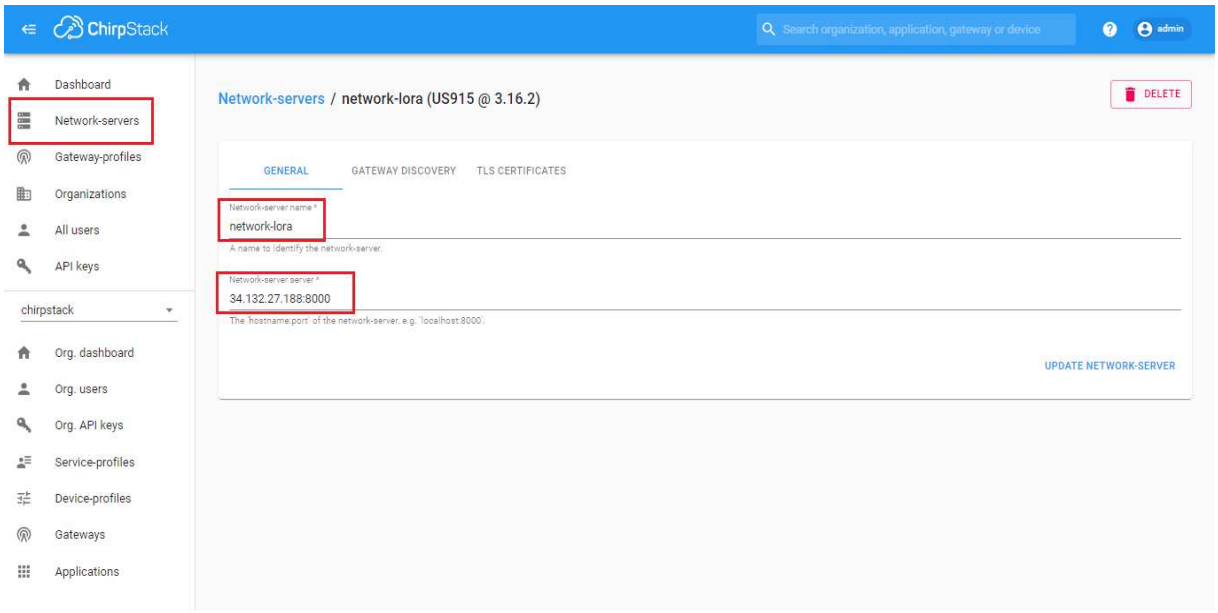


Figura 96. *Creación de Network Server.*

5.5.2.2 Creación de Gateway Profile

Es necesario crear un perfil de gateway, para posteriormente asignar a una puerta de enlace, además es necesario especificar un plan de canales el cual define el ancho de banda de radio de la siguiente manera:

- Canal de 500 kHz = ancho de banda de radio de 1,1 MHz.
- Canal de 250 kHz = ancho de banda de radio de 1 Mhz.
- Canal de 125 kHz = ancho de banda de radio de 0,925 MHz.

Para la creación de este perfil se tendrá que dirigir a la opción Gateway profile y se creara con las opciones descritas anteriormente, como se observa en la Fig.97.

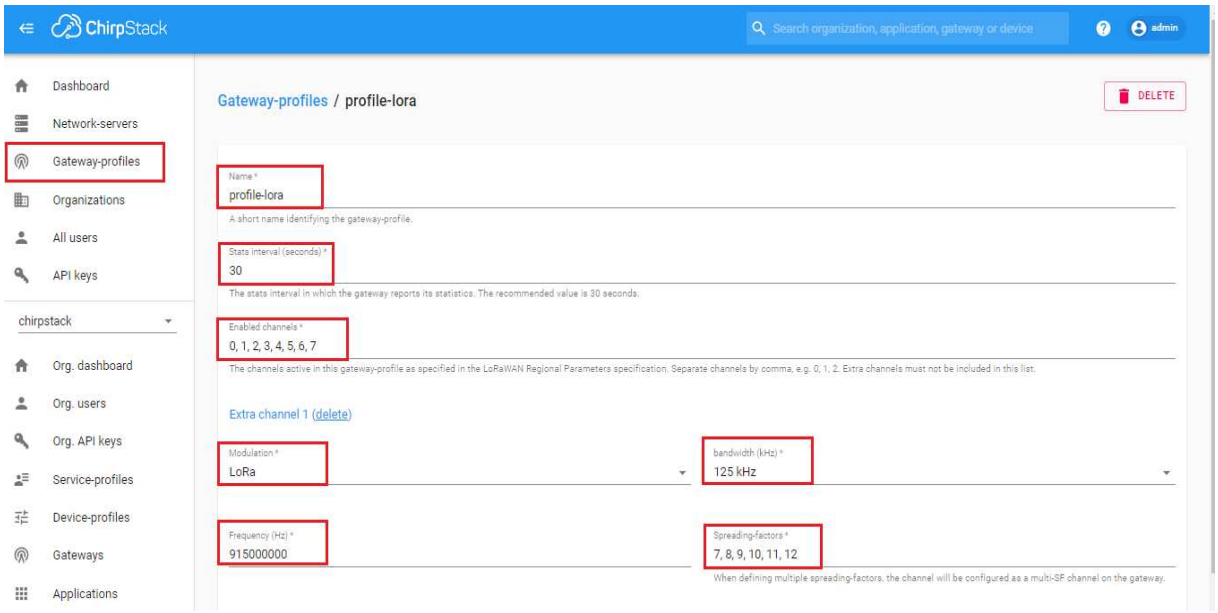


Figura 97. *Creación de Gateway Profile*.

5.5.2.3 Creación de Service Profile

El perfil de servicio puede definirse como el contrato entre un usuario y la red. Como se describe en la Fig.98, las funciones que están habilitadas para los usuarios del perfil de servicio y la tasa de mensajes que se pueden enviar a través de la red, para ello se dirigirá a la opción service profile y se creará de la siguiente manera.

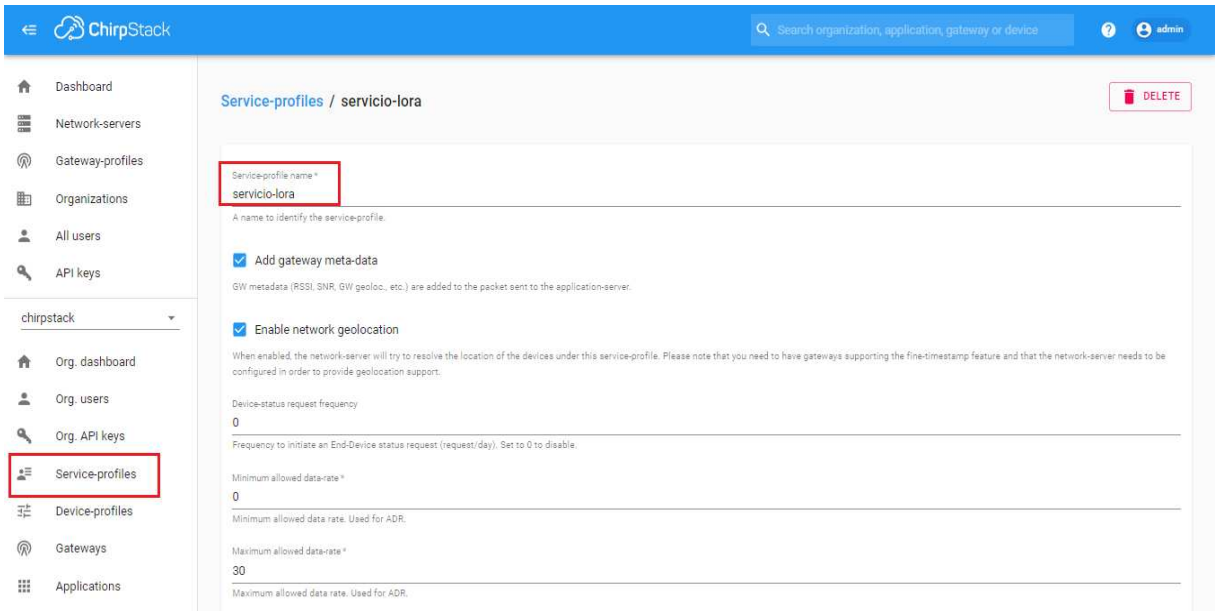


Figura 98. *Creación de Service Profile*.

5.5.2.4 Creación de Device Profile

Un perfil de dispositivos define las capacidades del dispositivo y los parámetros de arranque que necesita el servidor de red para configurar el servicio de acceso, es decir permitirá reconocer a los dispositivos por el gateway. Para ello se dirigirá a la opción device profile y se creará de la siguiente manera, como se ilustra en la siguiente Fig.99.

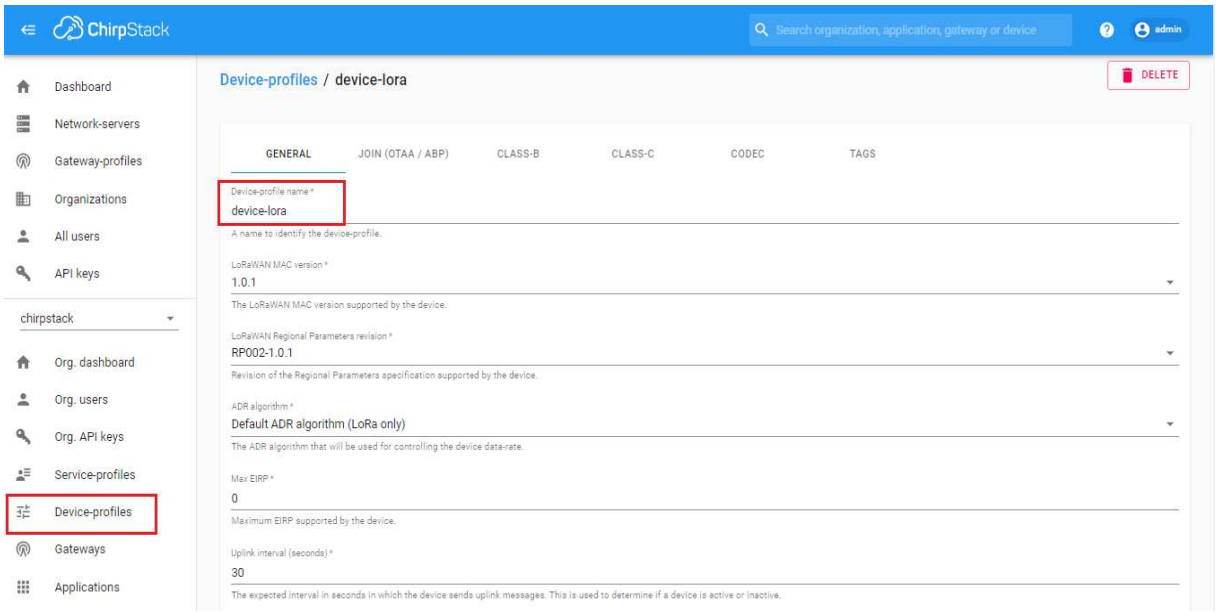


Figura 99. *Creación de Device Profile*.

En este punto como se visualiza en la Fig.100, se debe hacer una configuración extra en el CODEC, que son funciones de JavaScript que sirve para decodificar la data recibida por los dispositivos, aquí se ingresara un código de JavaScript que nos ayudara con este proceso. Para ello se ingresará a la pestaña de CODEC y se ingresará el código de JavaScript en la parte de decode, este código lo podemos observar en el siguiente enlace h https://drive.google.com/file/d/1sgyFa3_knSEpNt6JwAszTUuVdMMvzZfO/view?usp=sharing.

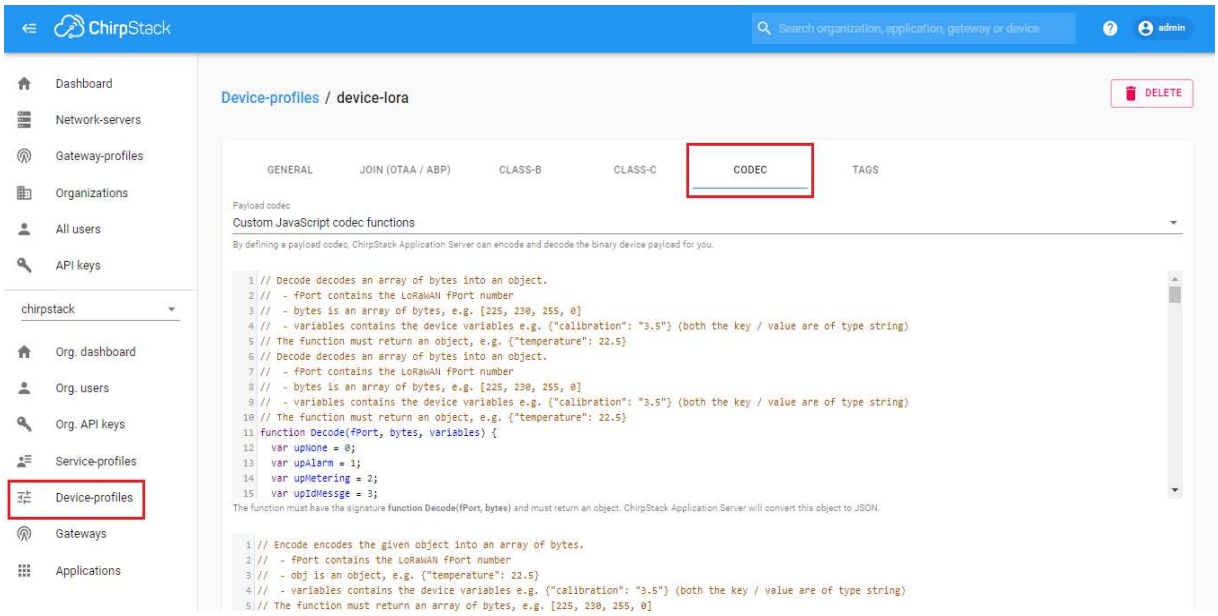


Figura 100. *Ingreso de código JavaScript a CODEC.*

5.5.2.5 Creación de Gateway

La creación de un gateway LoRA va acorde a los datos del gateway a registrar. Una cosa muy importante a tomar en cuenta es que el GatewayId es la MAC del mismo. En este punto es necesario haber creado todos los componentes anteriores como el gateway profile y el sevice profile, como se aprecia en la Fig.101.

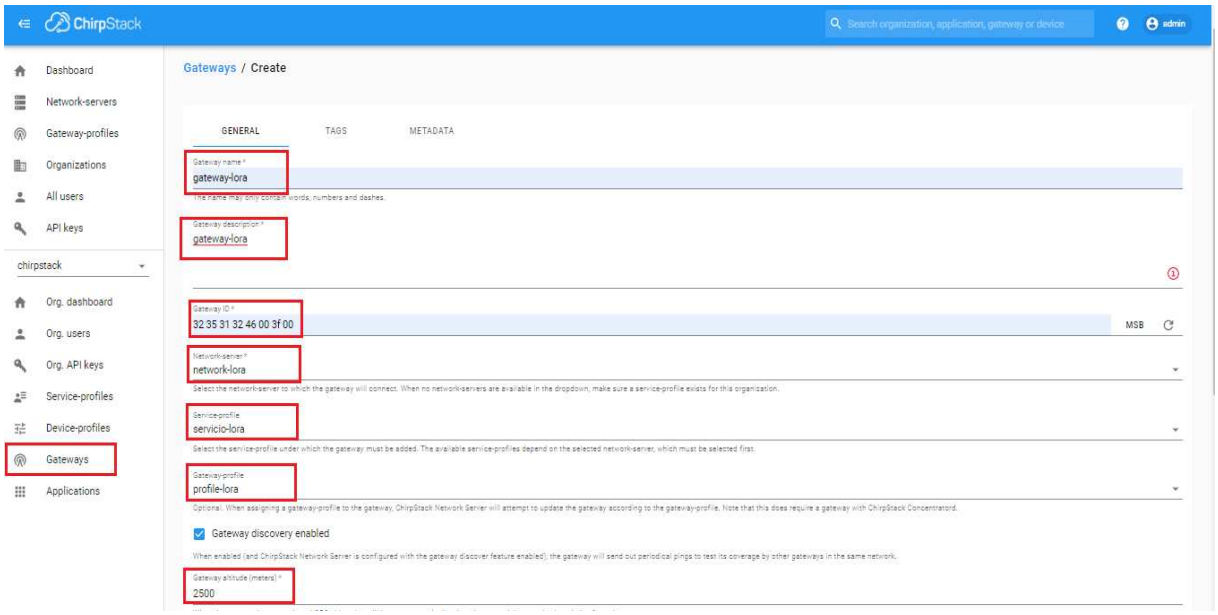


Figura 101. *Creación de Gateway*.

También se puede definir una ubicación del gateway, simplemente como se ve en la Fig.102 se navegará en el mapa y pondremos un punto de referencia de donde está ubicado el gateway y lo guardamos junto con el resto de configuraciones de registro de gateway.

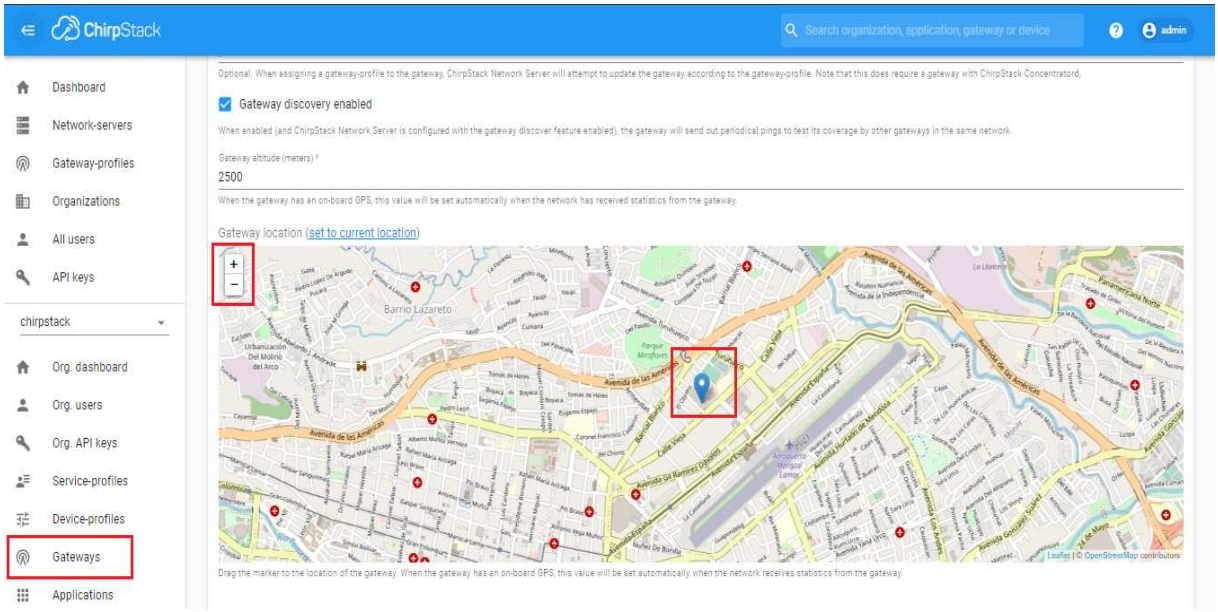


Figura 102. *Ubicación de Gateway*.

Una vez guardada la configuración se esperará un momento para que el gateway enganche y se establezca una conexión con el servidor ChirpStack. En la pestaña de dashboard de ChirpStack se puede observar el gateway en estado online lo que significa que se estableció la conexión gateway-servidor, como se visualiza en la Fig.103.

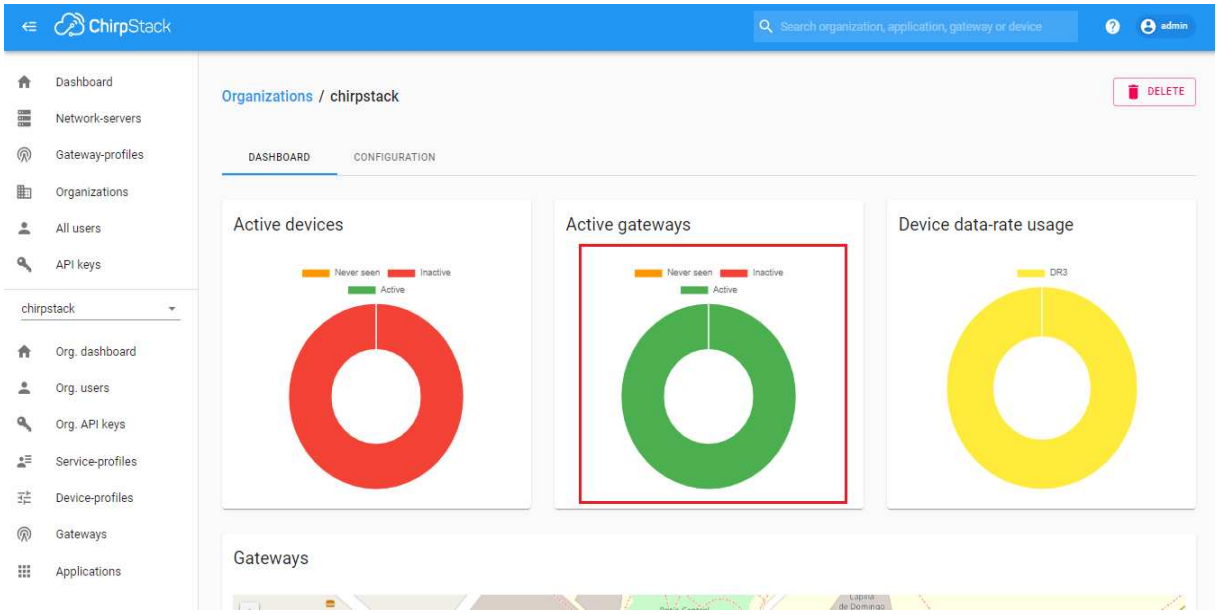


Figura 103. *Estado de Gateway*.

Así mismo se puede ver en la Fig.104, un resumen del estado del gateway en la opción gateway del menú principal, en ella se puede observar la ubicación, el ID, altitud, coordenadas de ubicación, última conexión o dato recibido por parte de ChirpStack. Además, en la Fig.105, de gráficas con detalles de datos recibidos y enviados, frecuencia transmitida y recibida, DR transmitido y recibido y por último el estado de transmisión, como se indica en la Fig.106.

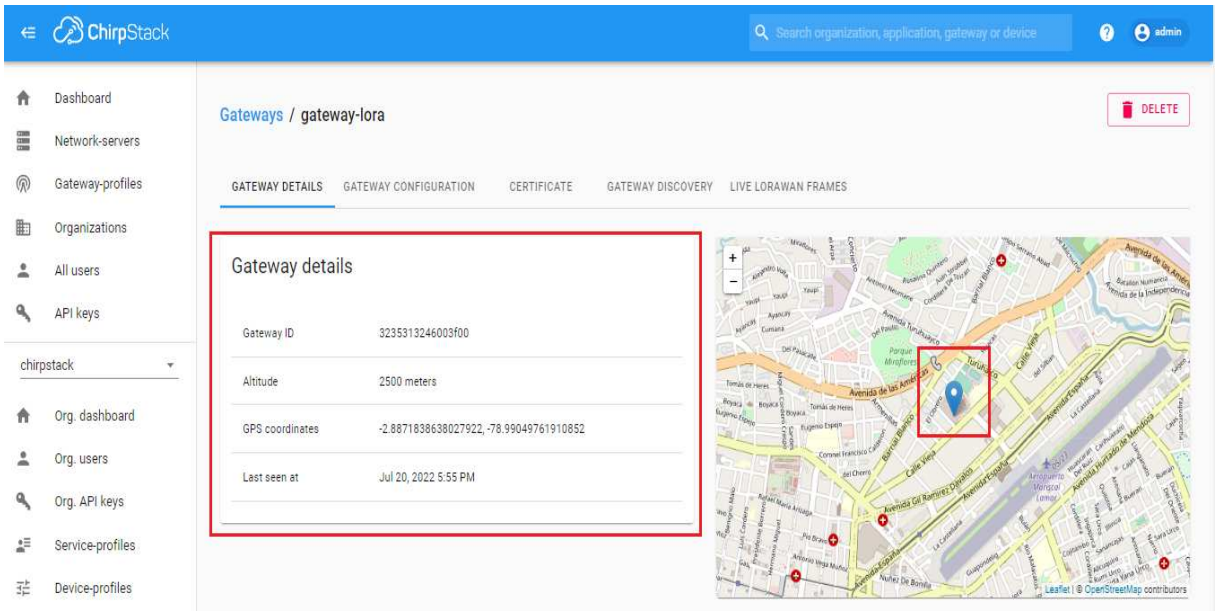


Figura 104. *Destalles de Gateway*.

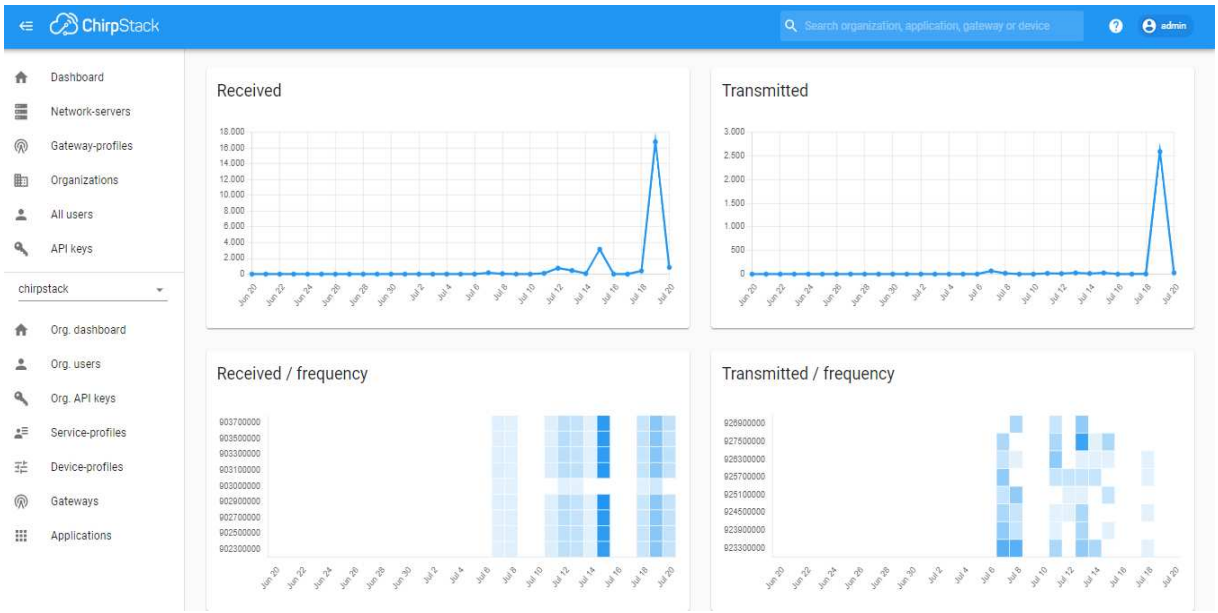


Figura 105. *Gráficas de datos y frecuencias de gateway*.

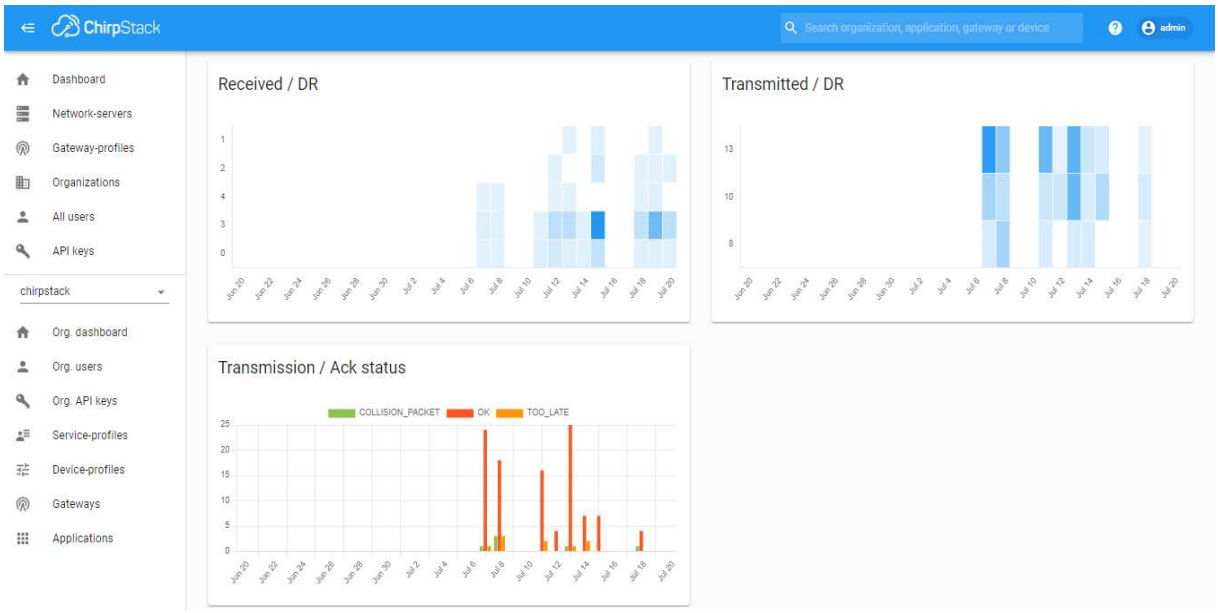


Figura 106. *Gráficas de transmisión de datos de gateway*.

También se puede observar en la Fig.107, los datos transmitidos en vivo en la opción de *LIVE LORAWAN FRAMES*, en donde se podrá observar los datos recibidos, ahí se podrá encontrará datos de frecuencia, relación señal/ruido, intensidad de señal rssi, altitud, etc en la Fig.108.

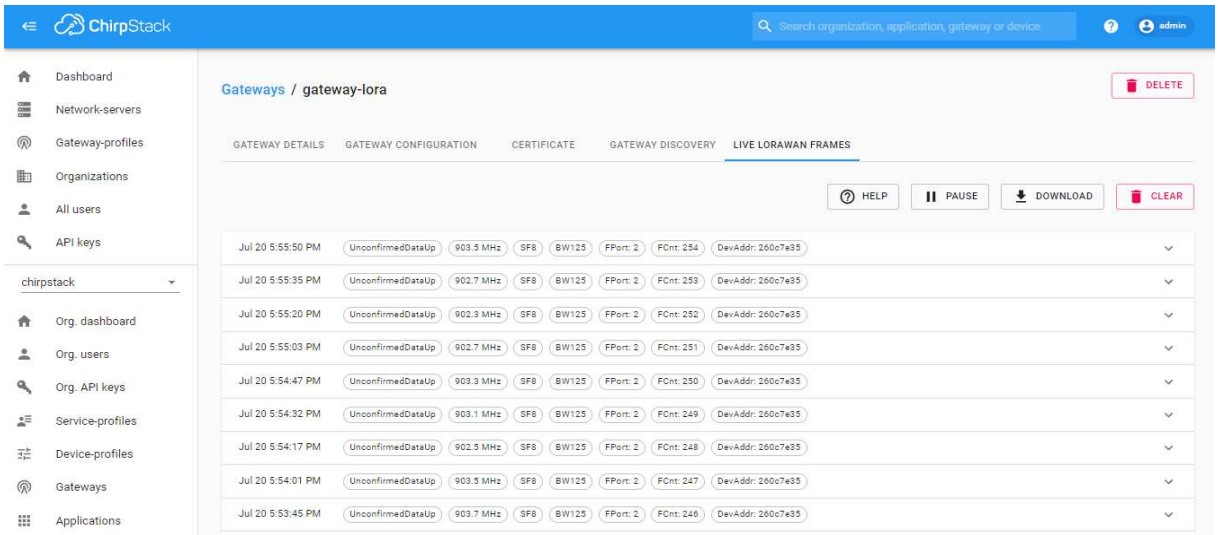


Figura 107. *Live LoraWan Frames*.

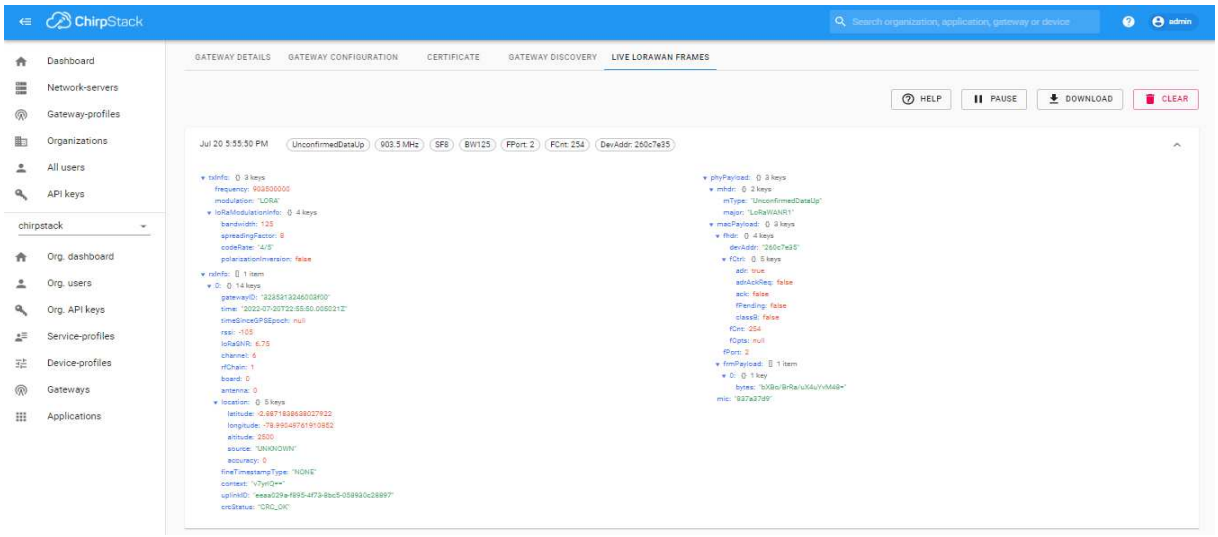


Figura 108. *Datos de Live LoraWan Frames*.

5.5.2.6 Creación de un Dispositivo

Para la creación de un dispositivo, como se indica en la Fig.109, primeramente se debe crear una aplicación que es una colección de dispositivos con el mismo propósito o del mismo tipo. En la creación del application se pedirá un service profile el cual se creó anteriormente.

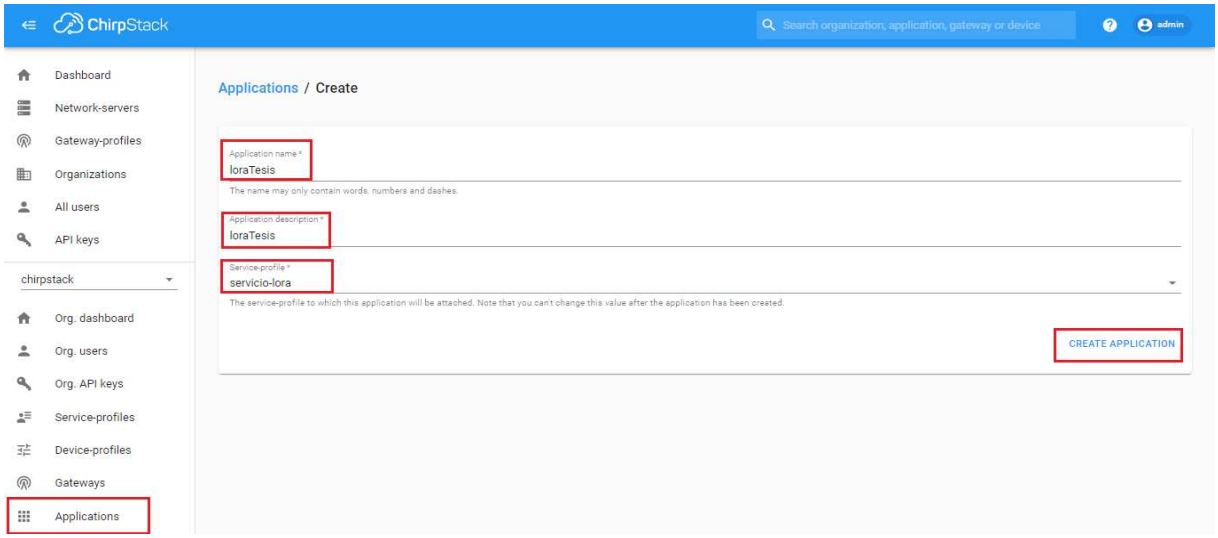


Figura 109. *Creación de un Application*.

Una vez creado el application, como se observa en la Fig.110, se procede a crear un dispositivo, es este caso se crearan dispositivos que miden el consumo de agua, es decir medidores inteligentes que enviaran datos sobre el consumo cada vez que pase agua por ellos.

Figura 110. *Dispositivo a registrar.*

Para la creación de estos dispositivos se dirigirá a la application creado anteriormente y se dará clic en create, en esta pestaña de creación de dispositivos se pedirá un nombre de dispositivo, una descripción, un device EUI el cual estará especificado en el mismo dispositivo, y por último un device profile el cual se creó anteriormente, como se aprecia en la Fig.111.

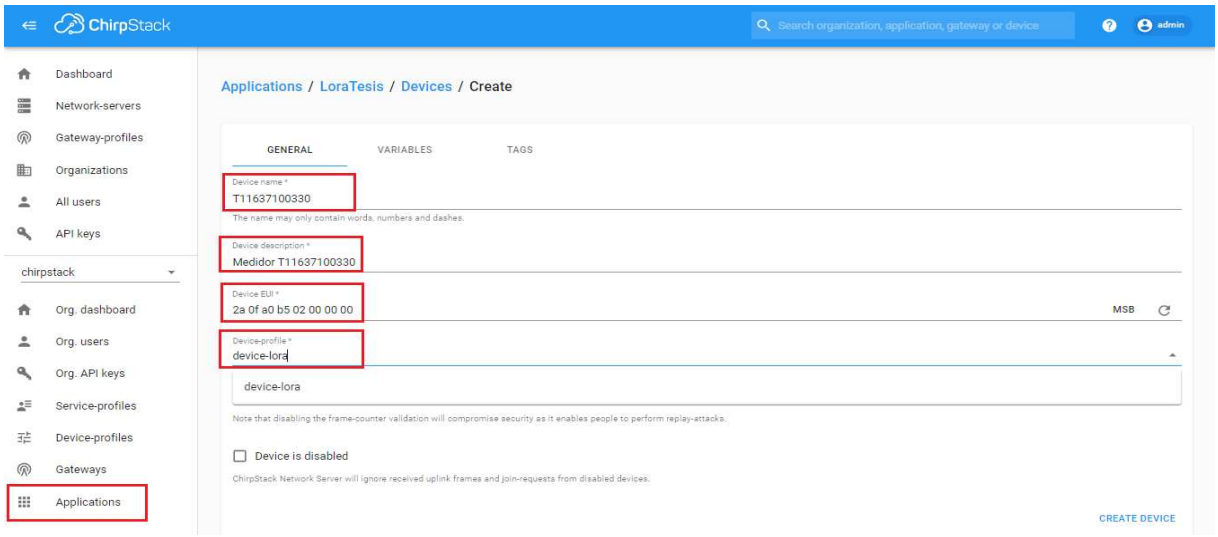


Figura 111. *Creación de un dispositivo.*

Como se ilustra en la Fig.112, cuando se haya creado el dispositivo, lo podemos ver en la lista de dispositivos del application creado anteriormente.

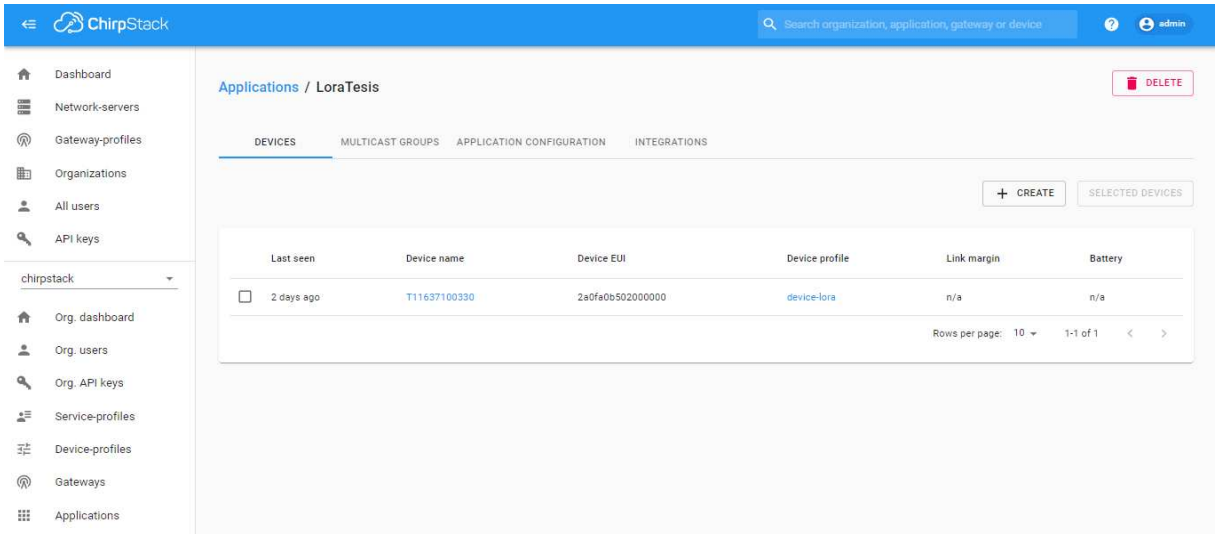


Figura 112. *Resumen de dispositivos creados.*

El servidor de aplicaciones ChirpStack admite dispositivos de tipo OTAA(activación por aire) y dispositivos ABP(activación por personalización). En este caso se está utilizando dispositivos tipo OTAA, por lo que es necesario ingresar la llave de activación

para que el dispositivo se sincronice y funcione correctamente en el servidor.

Para ingresar la clave OTAA, se ingresará al dispositivo creado anteriormente, una vez ahí se dará clic en la opción KEYS(OTAA), y se ingresará la clave correspondiente al dispositivo que se creó, como lo indica en la Fig.113.

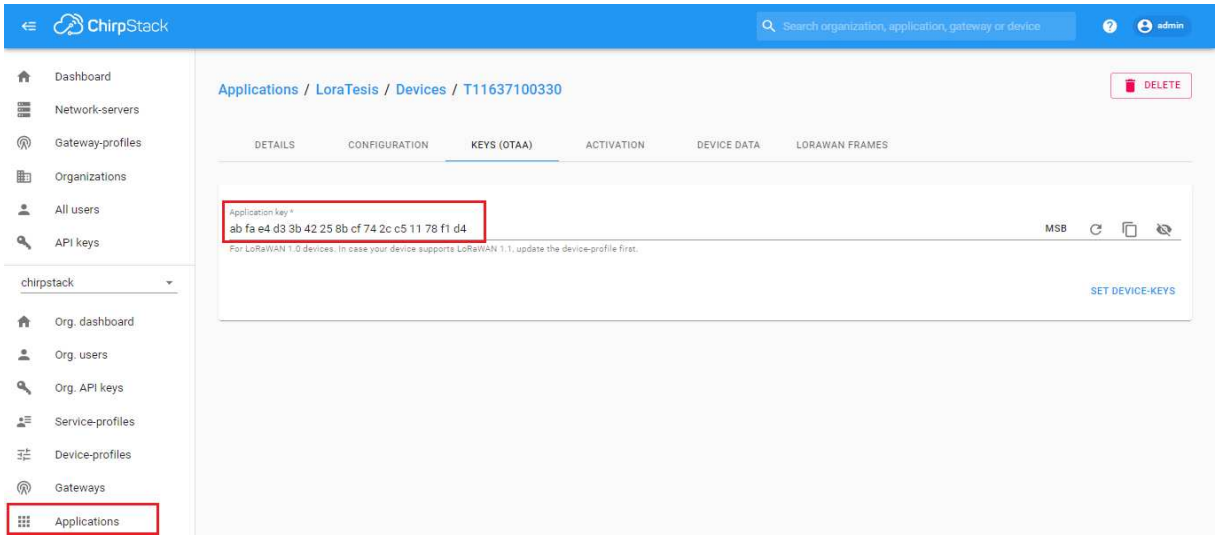


Figura 113. *Ingreso de clave OTAA*.

En la pestaña de details, como se podrá observar en la Fig.114, un resumen del dispositivo como el nombre, descripción, el device profile en donde se registró, la fecha y hora de último registro y el estado. Además, en la Fig.115 de gráficas de datos recibidos, errores, el SNR (Relación señal/ruido), RSSI(Intensidad de señal recibida) y frecuencia.

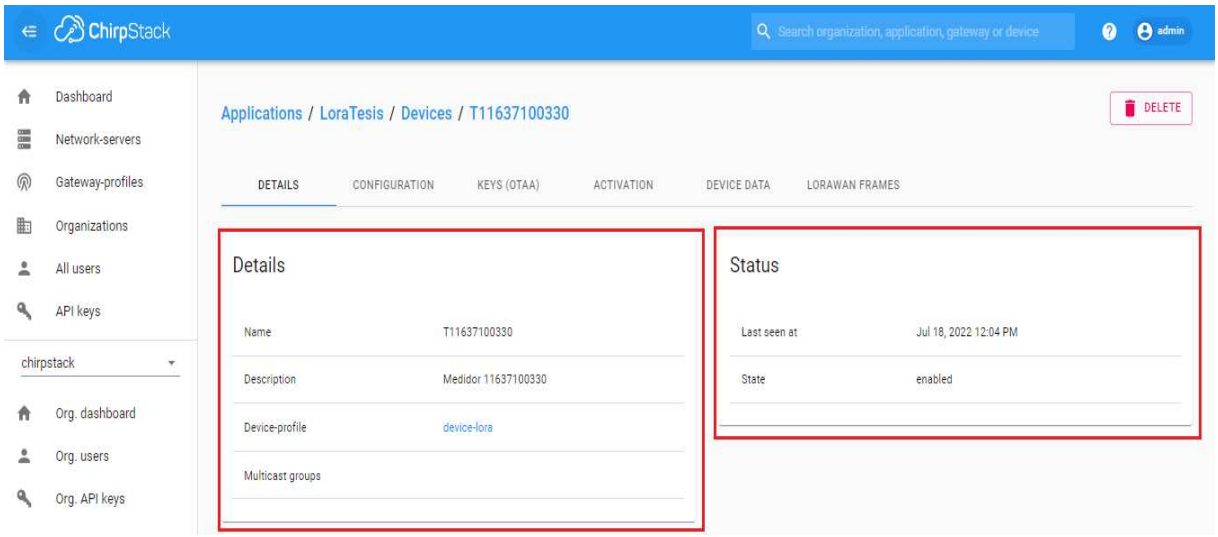


Figura 114. *Resumen de dispositivo*.

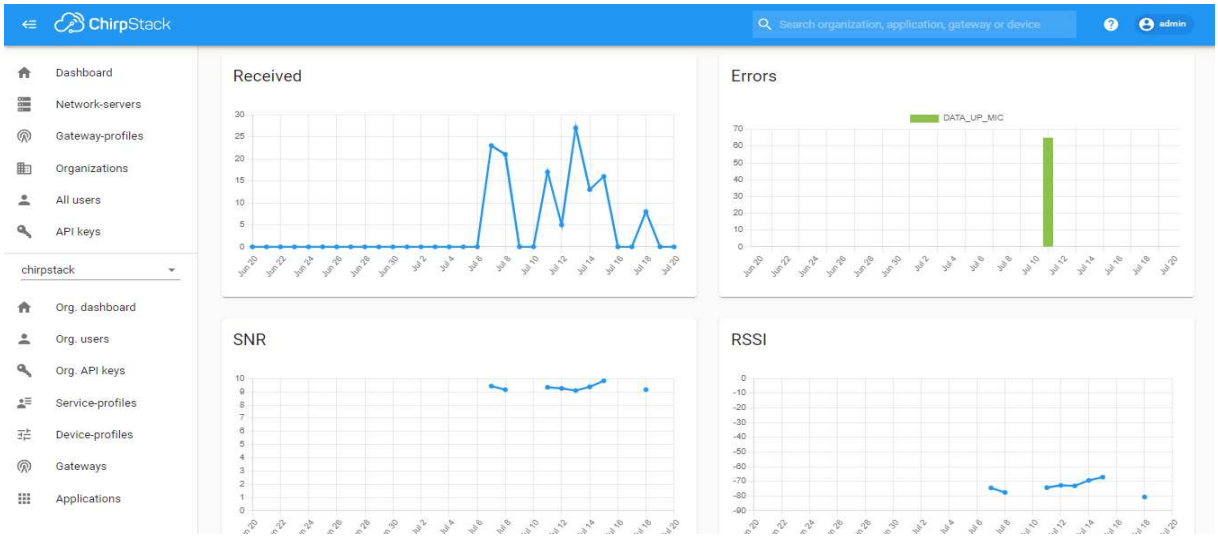


Figura 115. *Gráficas de datos recibidos de dispositivo*.

Como en el gateway, podemos ver en la Fig.116 los datos en vivo en la opción de *DEVICE DATA*, aquí se irán viendo en la Fig.117 los datos que se vaya recibiendo según pase el tiempo y el servidor reciba datos del dispositivo cada vez que pase agua o haya algún consumo de agua sobre él. Dentro cada uno de estos datos se podrá ver el eui, frecuencia, consumo, etc.

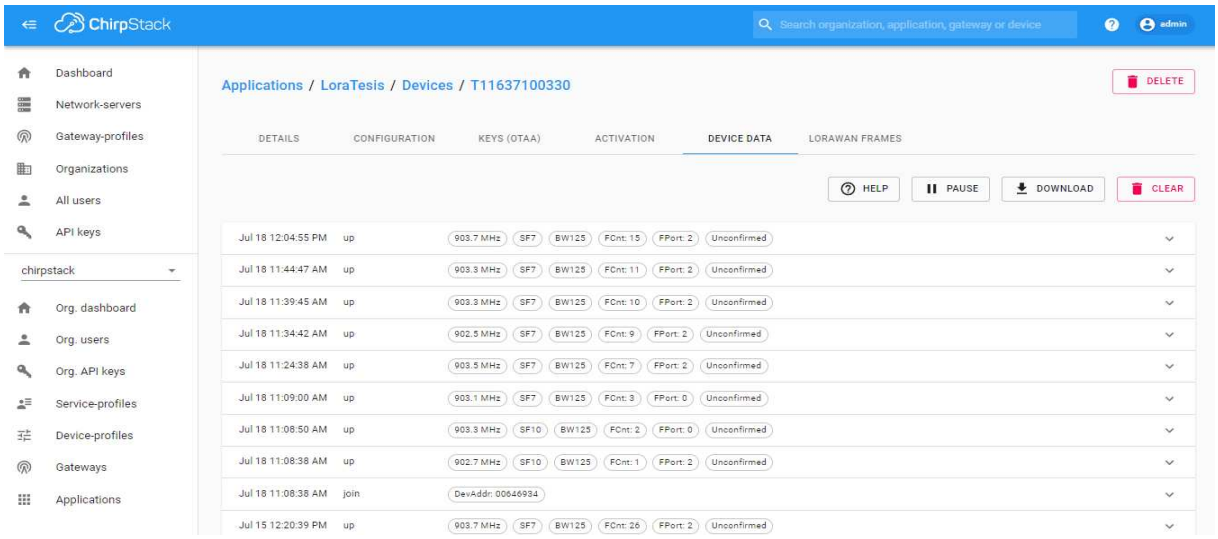


Figura 116. *Device Data*.

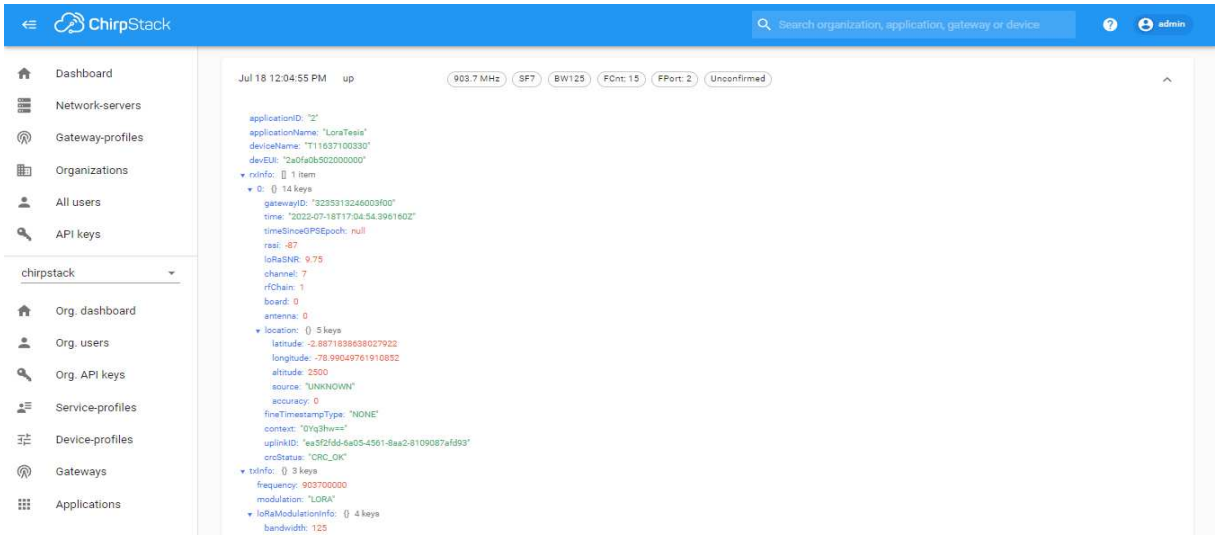


Figura 117. *Datos recibidos en Device Data*.

En la siguiente Fig.118, estos datos cada vez que lleguen al servidor se guardaran en la base de datos de eventos creada anteriormente, la tabla que contiene todos los datos de los dispositivos y gateway se llama device_up.

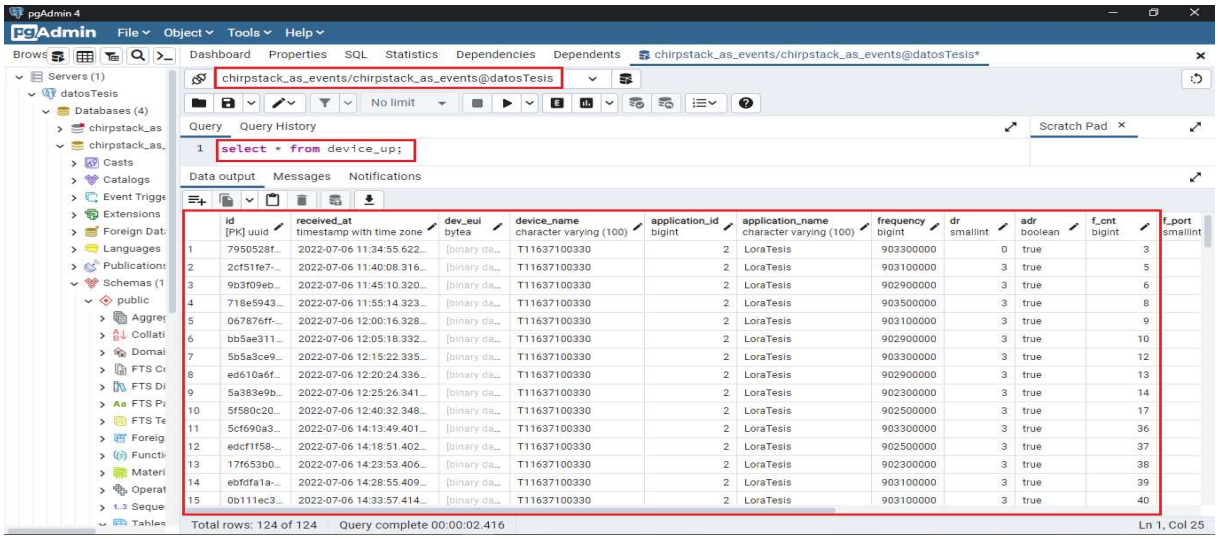


Figura 118. *Datos guardados en base de datos de eventos.*

En la columna *rx_info*, se podrá ver los datos recibidos por parte del gateway en formato JSON, como se observa en la Fig.119.

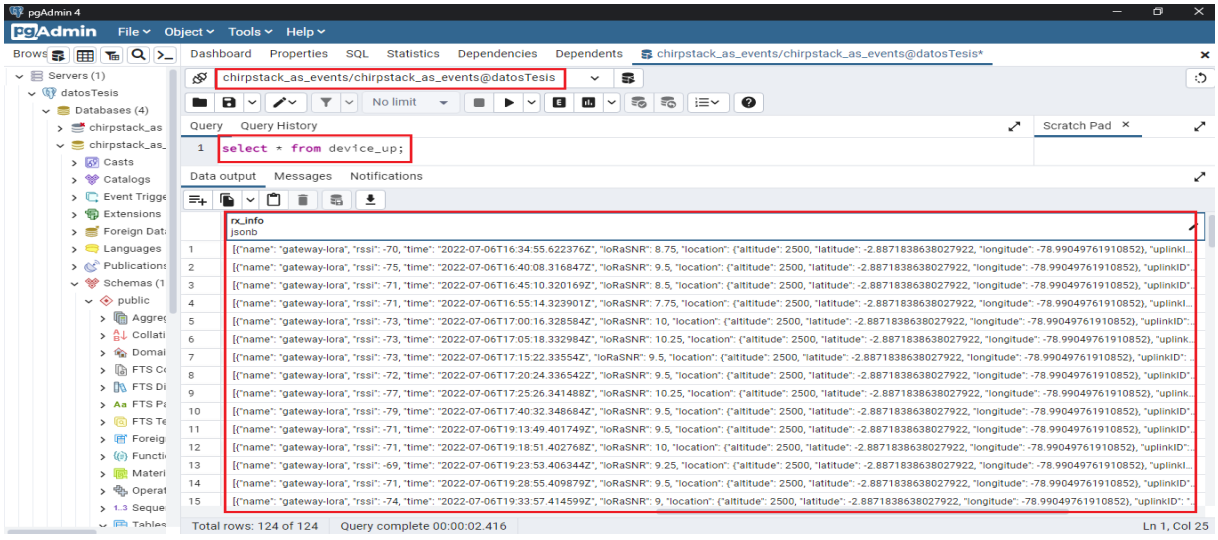


Figura 119. *Datos de gateway en base de datos.*

En cambio, en la Fig.120, los datos del dispositivo se verán en todas las filas, pero el dato más relevante para el consumo de agua se verá en la columna*Object*, donde se tendrá un dato llamado "Consumo" en formato JSON, este dato es el número que aparece en la

pantalla del medidor de agua el cual ira subiendo cada vez que pase agua sobre él.

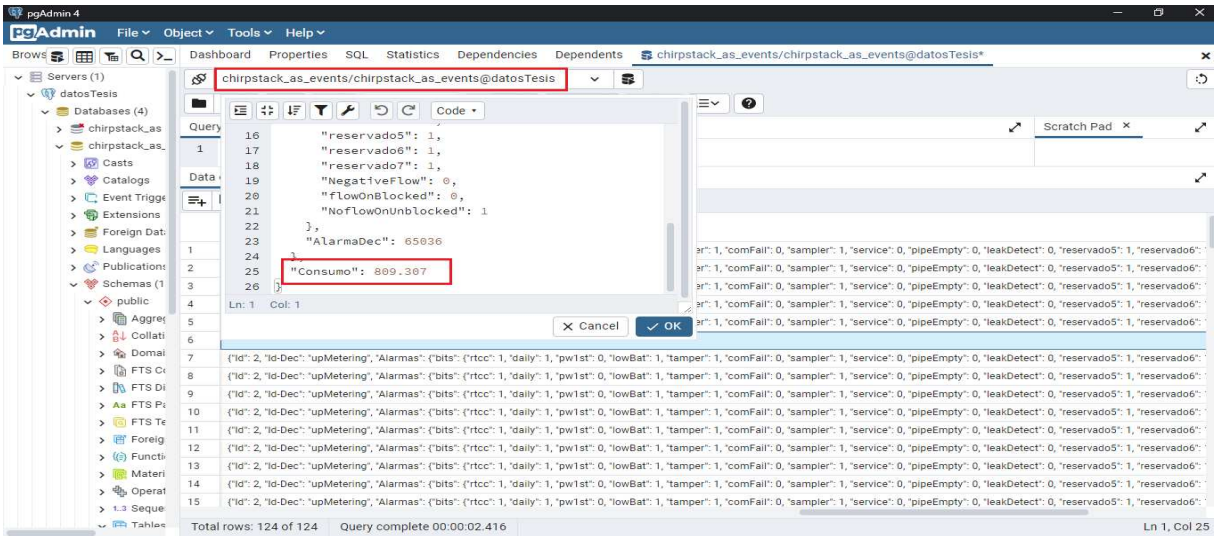


Figura 120. *Dato de consumo del medidor de agua.*

5.5.3 Creación de dashboard para mostrar datos de gateway y dispositivos mediante Node Red

Para la creación de un dashboard se necesita la instalación de varios nodos que nos ayudaran a que los datos recibidos en la base de datos de eventos de ChirpStack se muestren mediante gráficas generadas por Node-Red.

5.5.3.1 Instalación y explicación de nodos a usar en la implementación de dashboard

Para instalar nodos se tendrá que ir al menú principal en la opción de Manage Palette, ahí se podrá instalar nuevos nodos que no vengan por defecto en la instalación, así también revisar todos los nodos que se tendrá instalado en el sistema, como se visualiza en la FIG.121

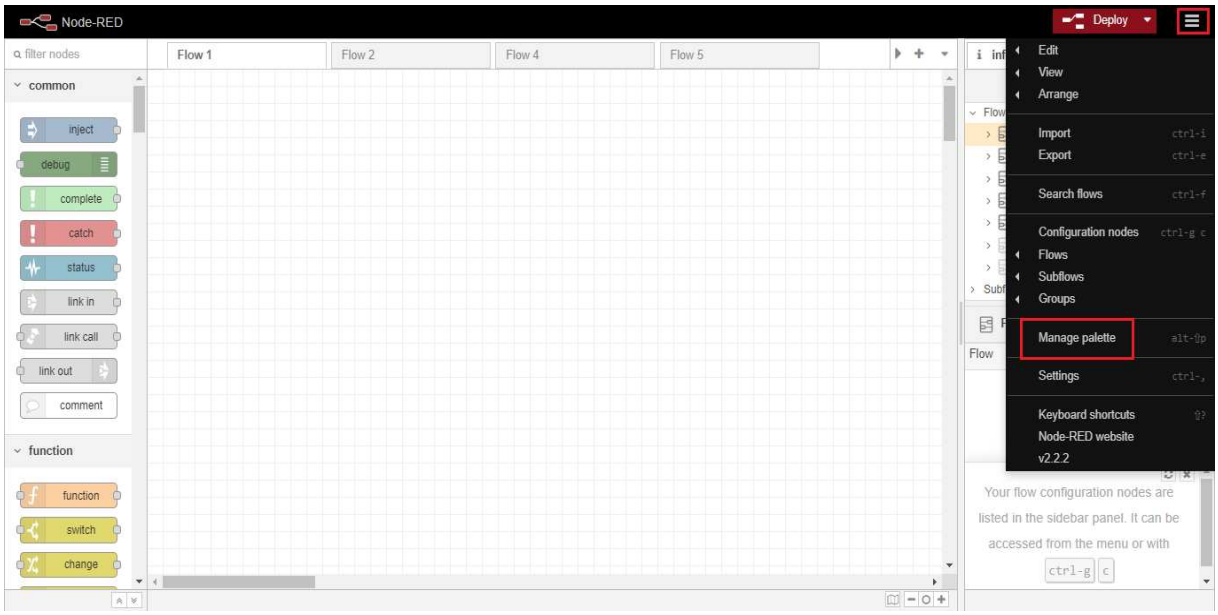


Figura 121. *Menú de Instalación de nodos*.

En la pestaña de Nodes se podrá observar los nodos que tenemos instalados, en cambio, en la pestaña de install podremos instalar nuevos nodos.

- **Instalación de node-red-contrib-postgresql**

Este nodo sirve para realizar consultas a una base de datos PostgreSQL, una vez hecha una consulta a una base de datos la salida o la respuesta (filas) se proporciona msg.payload como una matriz. El msg.payload es una variable administrada por node red en donde se almacenara los datos.

Para instalar este nodo se digitará postgresql en la barra de búsqueda de instalaciones, ahí se mostrara en la FIG.122, el nodo y se presionara en install.

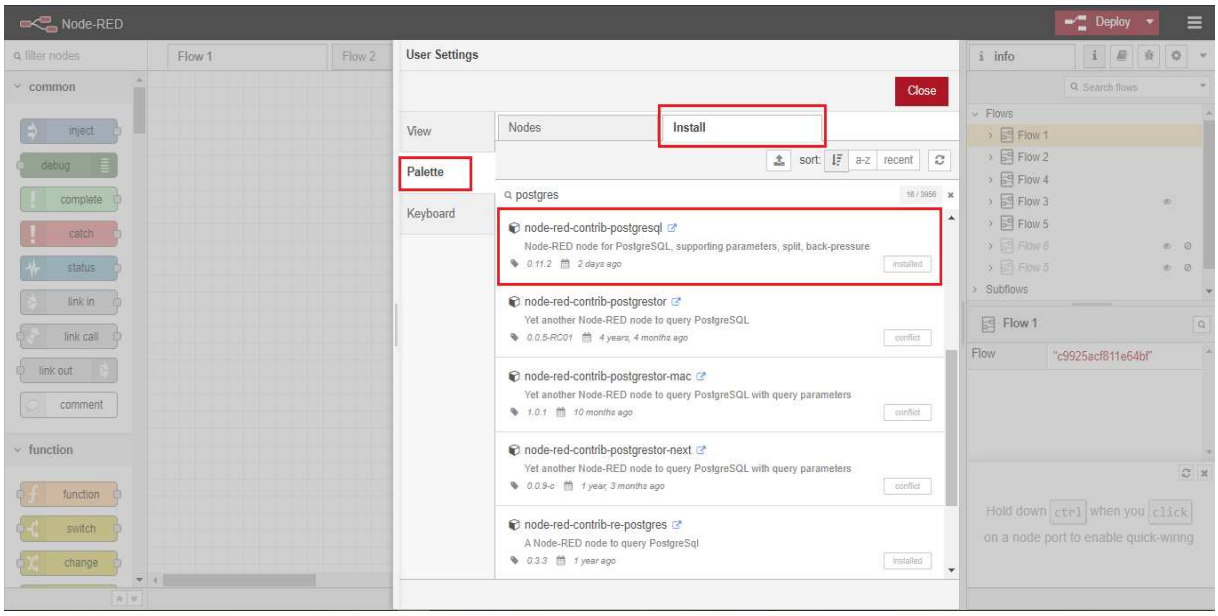


Figura 122. *Instalación de node-red-contrib-postgresql.*

Una vez instalado se podrá observar en la FIG.123 el nodo en la barra de nodos.

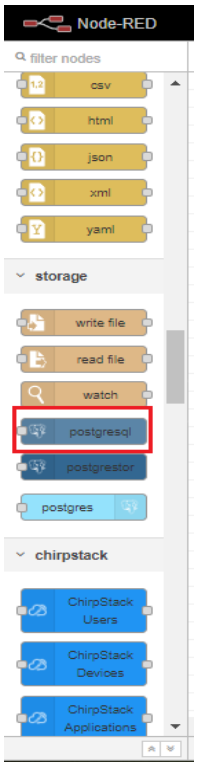


Figura 123. *node-red-contrib-postgresql instalado.*

- **Instalación de node-red-dashboard, node-red-contrib-ui-media y node-red-node-ui-table**

Los nodos de Dashboard permite crear rápidamente un tablero de datos en vivo, así mismo el nodo de media y table, que es parte de los nodos de dashboard, en este caso son widgets que se debe instalar de manera independiente a los nodos de dashboard, el nodo media se usa para insertar imágenes en la ventana de gráficos, mientras que el table se usa para insertar tablas.

Para ingresar a la ventana de dashboard, como se ve en la FIG.124 se debe ingresar la IP de conexión con el puerto de Node-Red, se debe verificar en la FIG.125 la instalación de media, y junto a ello un slash como está en la FIG.126, y se escribirá ui de la siguiente FIG.127 de esta manera:

34.132.27.188:1880/ui

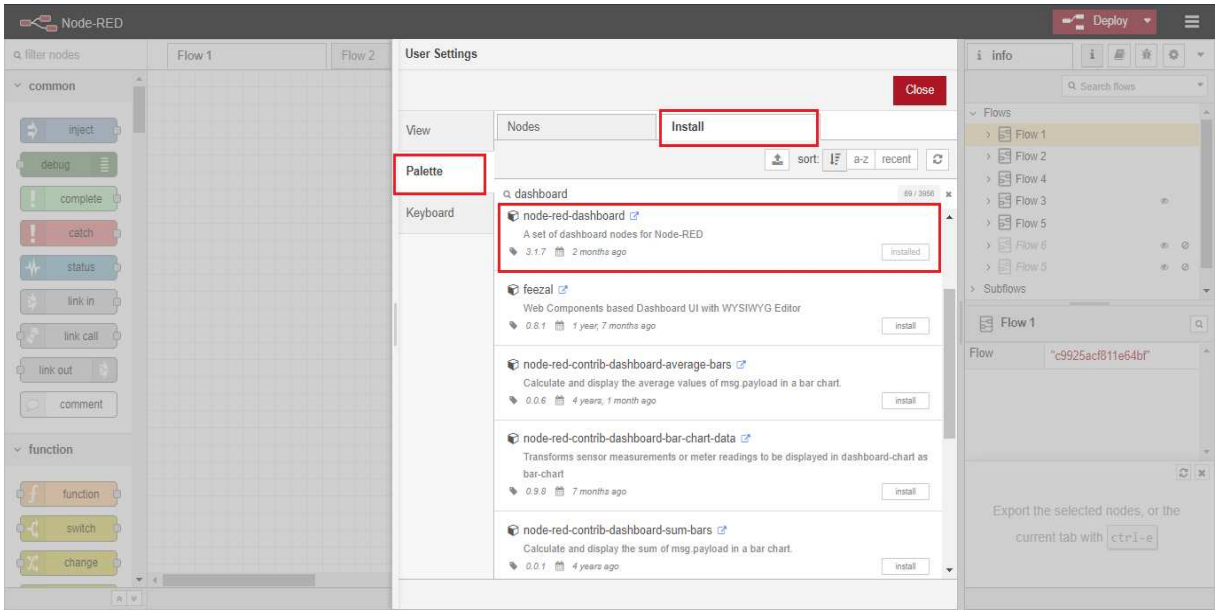


Figura 124. *Instalación de node-red-dashboard.*

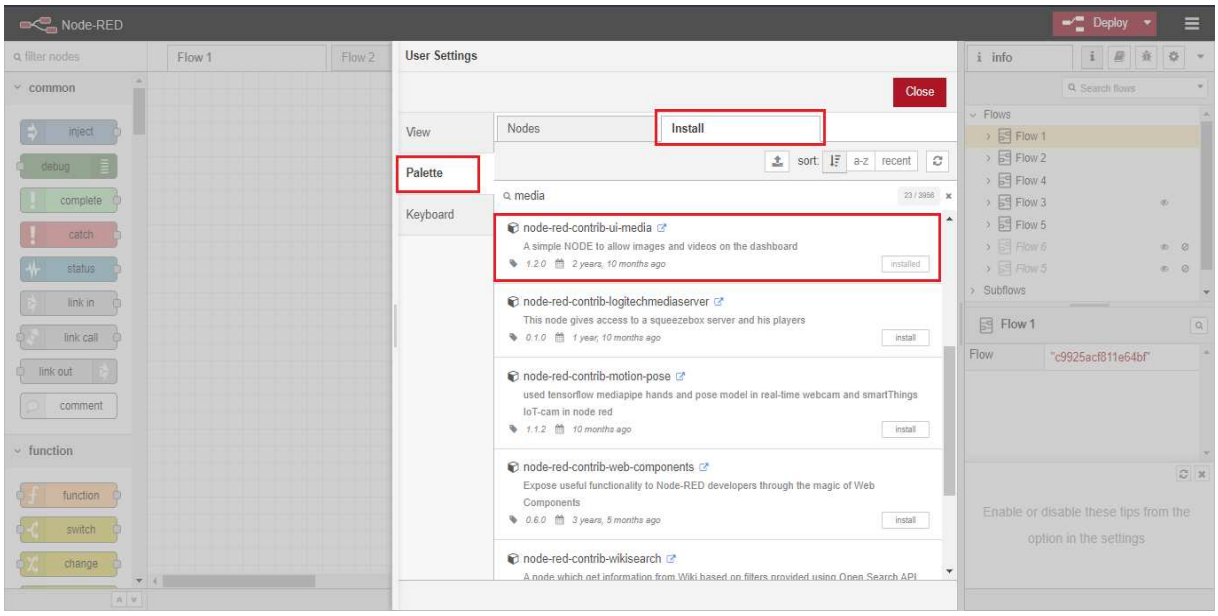


Figura 125. *Instalación de node-red-contrib-ui-media.*

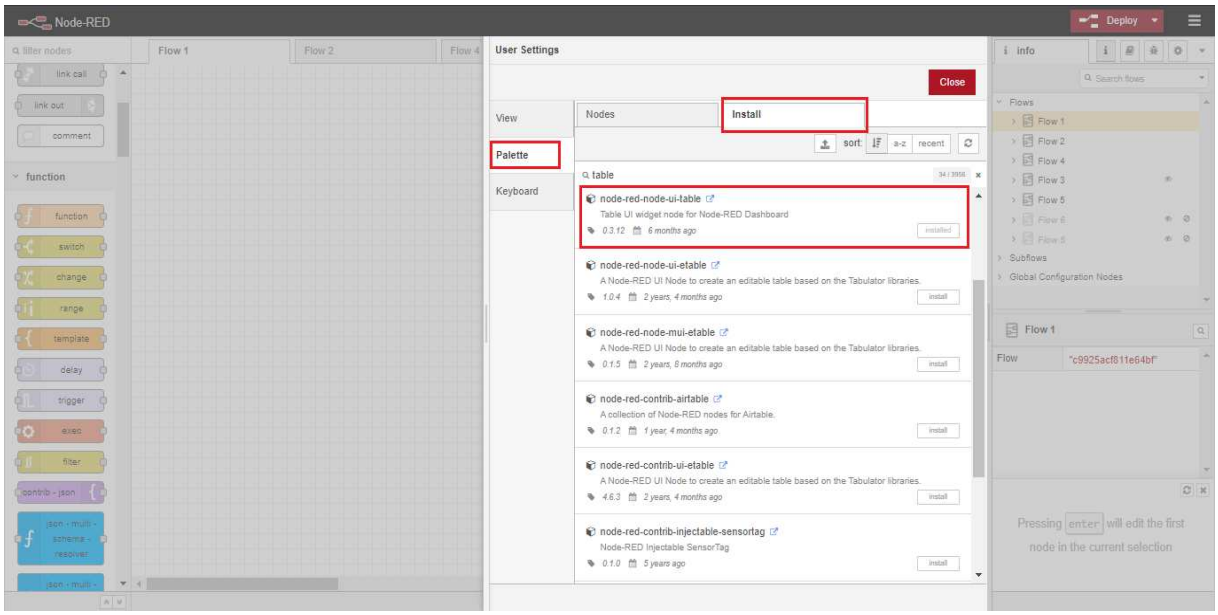


Figura 126. *Instalación de node-red-node-ui-table.*

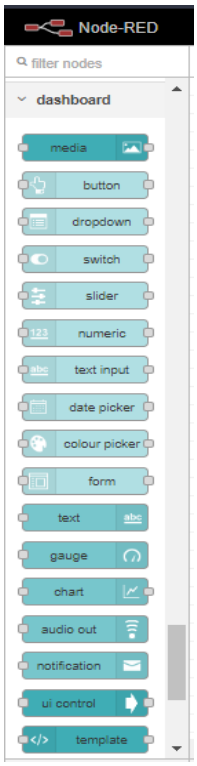


Figura 127. *Nodos de dashboard, media y table instalados*.

- **Instalación de node-red-contrib-remote**

Este nodo es usado para realizar una conexión remota a una aplicación de celular, y así poder observar el dashboard de manera remota en nuestro dispositivo móvil, como se aprecia en la FIG.128.

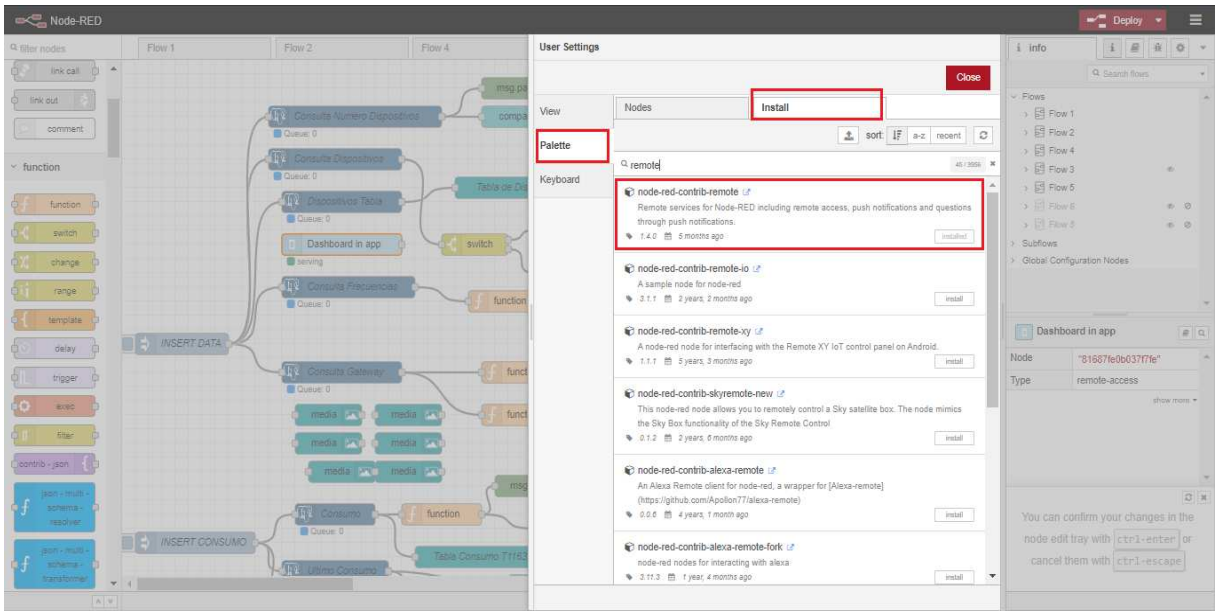


Figura 128. *Instalación de node-red-contrib-remote*.

- **Nodo inject**

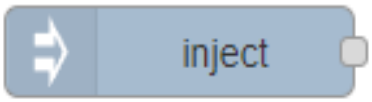


Figura 129. *Nodo inject*.

En la FIG.129, se aprecia que este nodo está instalado por defecto en Node-Red, se puede usar para activar manualmente un flujo de datos haciendo clic en el botón del nodo dentro del editor. También se puede utilizar para inyectar datos automáticamente configurando un intervalo de tiempo de ejecución.

- **Nodo debug**

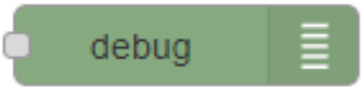


Figura 130. *Nodo debug*.

Este nodo está instalado por defecto en Node-Red, se puede usar para mostrar mensajes en la barra lateral de debug dentro del editor, cuando realicemos una consulta con el nodo de PostgreSQL podemos ver el resultado de la consulta con este nodo, como lo indicia en la FIG.130.

- **Nodo function**

Figura 131. *Nodo function*.

Este nodo está instalado por defecto en Node-Red, como se aprecia en la FIG.131 permite que el código JavaScript se ejecute en los mensajes que se pasan a través de él, este nodo ayudara para que los datos recibidos en una consulta del nodo PostgreSQL, cambiarlos la forma de visualización, para que en los nodos de dashboard puedan interpretarlos y poder graficarlos.

- **Nodo switch**

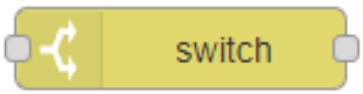


Figura 132. *Nodo switch*.

Este nodo como se aprecia en la FIG.132 está instalado por defecto en Node-Red, permite que los mensajes se enruten a diferentes ramas de un flujo mediante la evaluación de un conjunto de reglas para cada mensaje, funciona similar a un if en java, que se puede tener varias salidas dependiendo de las reglas que se ingresen en él.

- **Nodo change**

Figura 133. *Nodo change*.

Este nodo está instalado por defecto en Node-Red, se puede utilizar para modificar las propiedades de un mensaje y establecer propiedades de contexto, este nodo es una variante al nodo function, pero sin realizar código JavaScript, como se ve en la FIG.133.

- **Nodo text de dashboard**

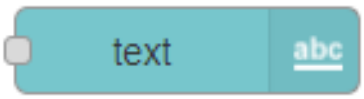


Figura 134. *Nodo text*.

Este nodo como se puede apreciar en la FIG.134 se instaló conjuntamente con los nodos de dashboard, es usado para insertar texto en la ventana de gráficas de dashboard, se puede configurar un widget de solo lectura, el diseño de la etiqueta y el valor.

- **Nodo gauge de dashboard**

Figura 135. *Nodo gauge*.

Este nodo se instaló conjuntamente con los nodos de dashboard, es usado para insertar distintos tipos de gráficas, tiene 4 modos: estándar (calibre simple), dona (360° completo), brújula y onda. También puede especificar la gama de colores de los calibres estándar y de dona, como se indica en la FIG.135.

- **Nodo chart de dashboard**

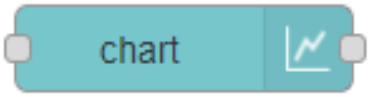


Figura 136. *Nodo chart*.

En esta FIG.136, este nodo se instaló conjuntamente con los nodos de dashboard, es usado para insertar distintos tipos de gráficas, tiene modos de gráfico de líneas, de barras y circular. Además, las etiquetas del eje X se pueden personalizar mediante una cadena de formato de fecha.

5.5.3.2 Creación de menús y grupos

Para poder crear distintos menús de navegación en el dashboard, se puede dirigir a la parte derecha a lado del menú de Node-Red, en donde se abrirá una opción de dashboard en donde se puede crear distintas pestañas de menú y grupos pertenecientes a cada menú, como lo indica en la FIG.137.

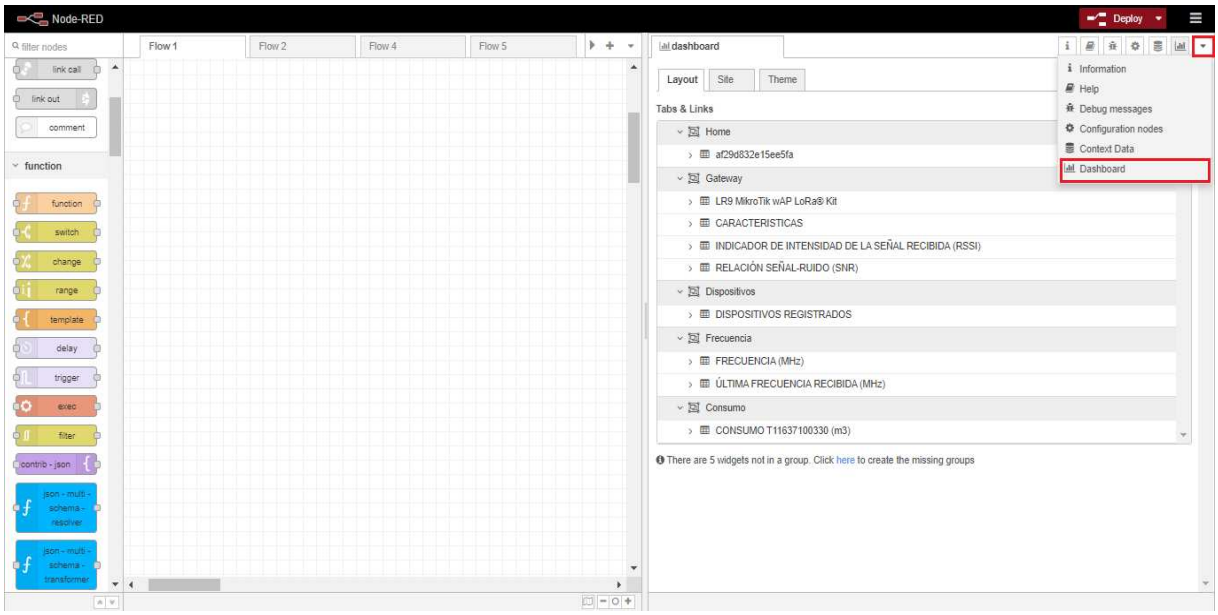


Figura 137. *Menú de dashboard.*

En esta sección, en la FIG.138, se creará varios menús para separar los distintos gráficos que se vayan a realizar en el dashboard, para crear un menú se dará clic en el botón + tab, cada menú puede tener varios grupos dentro de ella, dentro de este grupo se puede haber varios gráficos, para crear un grupo dentro de un menú se dará clic en el botón + group.

Los menús que se creara son:

- Home
- Gateway
- Dispositivos
- Frecuencia
- Consumo

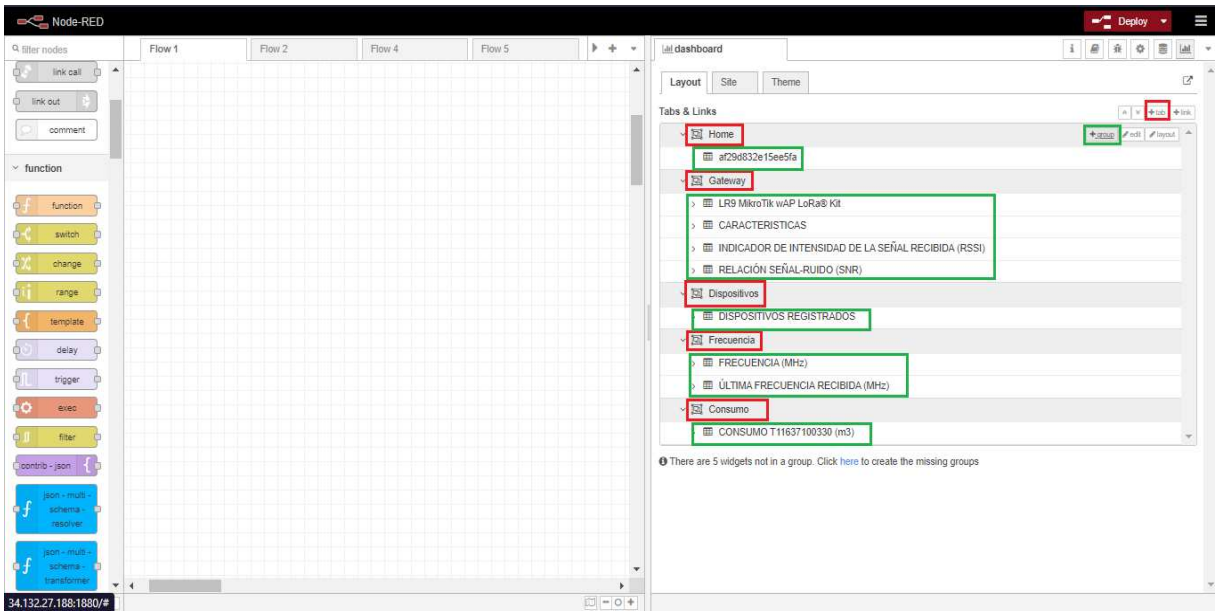


Figura 138. *Creación de menús y grupos en dashboard.*

Además, como se observa en la FIG.139 y en la FIG.140, se puede configurar el icono de cada menú, para ello se dará clic en editar menú y mediante una página de *Material Design icon* se puede observar un sinnúmero de iconos que se puede insertar para cada menú, simplemente se escribirá él nombre del icono en la opción de icon del menú.

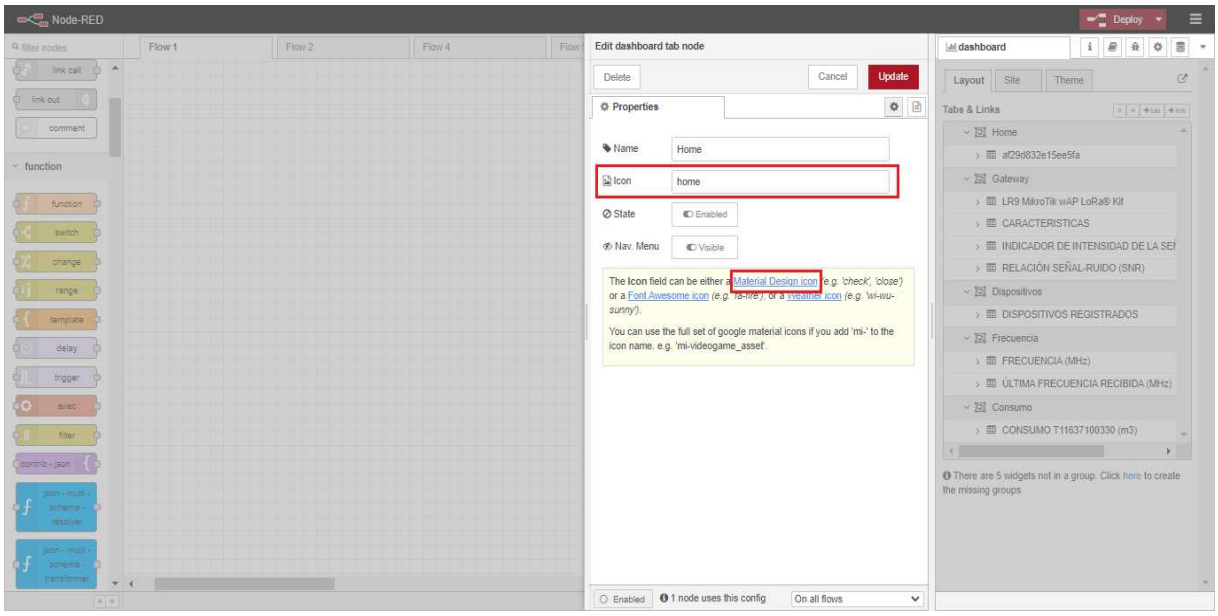


Figura 139. *Edición de icono de menu.*

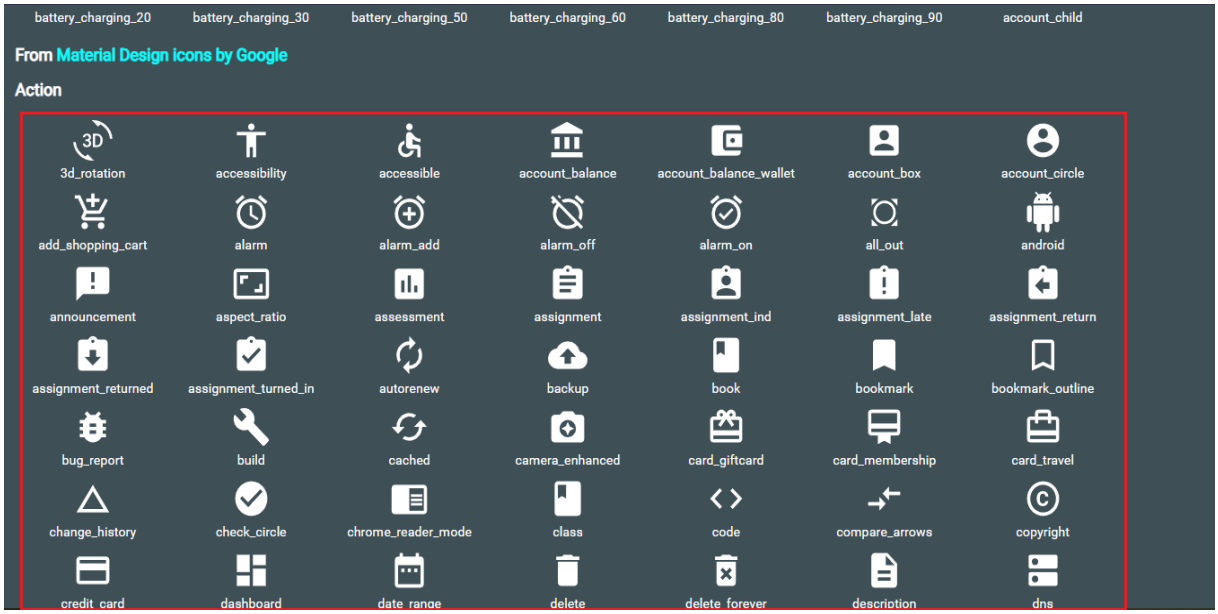


Figura 140. *Página de iconos para menú de dashboard.*

También en varias FIG.141existen otras funciones para personalizar nuestra ventana de dashboard, como el nombre del sitio web, el color de la página, tema y estilo de letra que se desee aplicar, como se observa en la FIG.142.

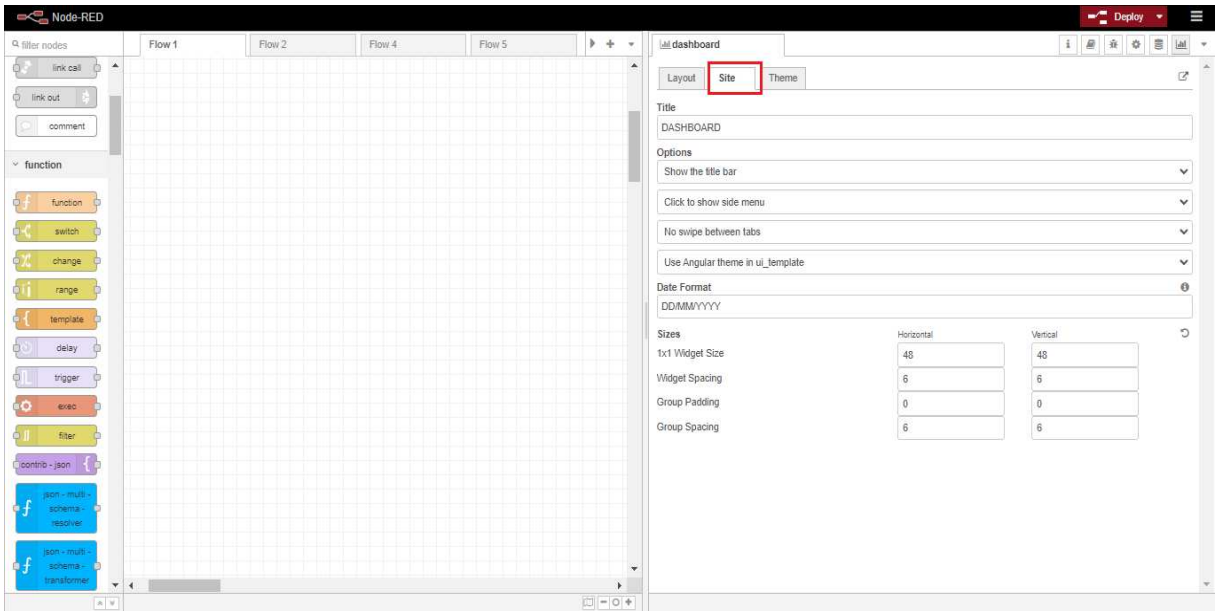


Figura 141. *Personalización de sitio web de dashboard.*

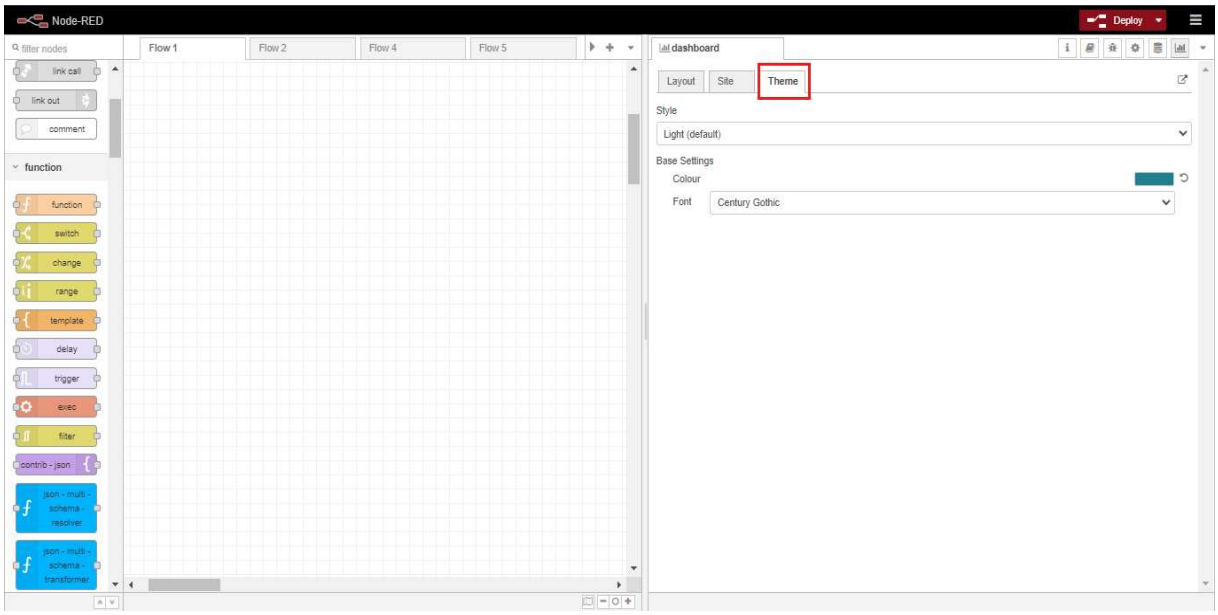


Figura 142. *Personalización de tema de ventana de dashboard.*

5.5.3.3 Implementación de nodos para graficar datos en dashboard

En este punto se hará uso de los distintos nodos explicados anteriormente y así poder ir armando el dashboard con los datos más relevantes del gateway y dispositivos registrados en el servidor de ChirpStack.

- **Menú Home**

En esta pestaña, como se ilustra en la FIG.143 se realizará una pequeña carátula del dashboard, para ello se hará uso de los nodos de media y text.

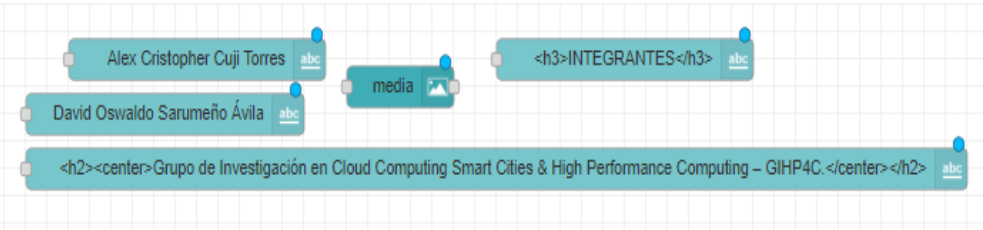


Figura 143. *Nodos del menu Home*.

Para los nodos como se puede observar en la FIG.144, de text pueden ser de dos tipos, mediante un label y texto o solo un label, en este caso solo se utilizara la opción de label para insertar un texto específico, para ello se ingresara a la configuración del text dando doble clic sobre el nodo, ahí se deberá elegir el menú y grupo donde se insertara el nodo, y si queremos que sea solo un label se escribirá en la opción de label y se dejara en blanco la opción de value format, además se eligiera la opción de label/value centrado. Cabe recalcar que estos tipos de nodos funcionan similar a un texto en html, es decir podemos configurar con comando en html, con etiquetas como <h3>, , etc.

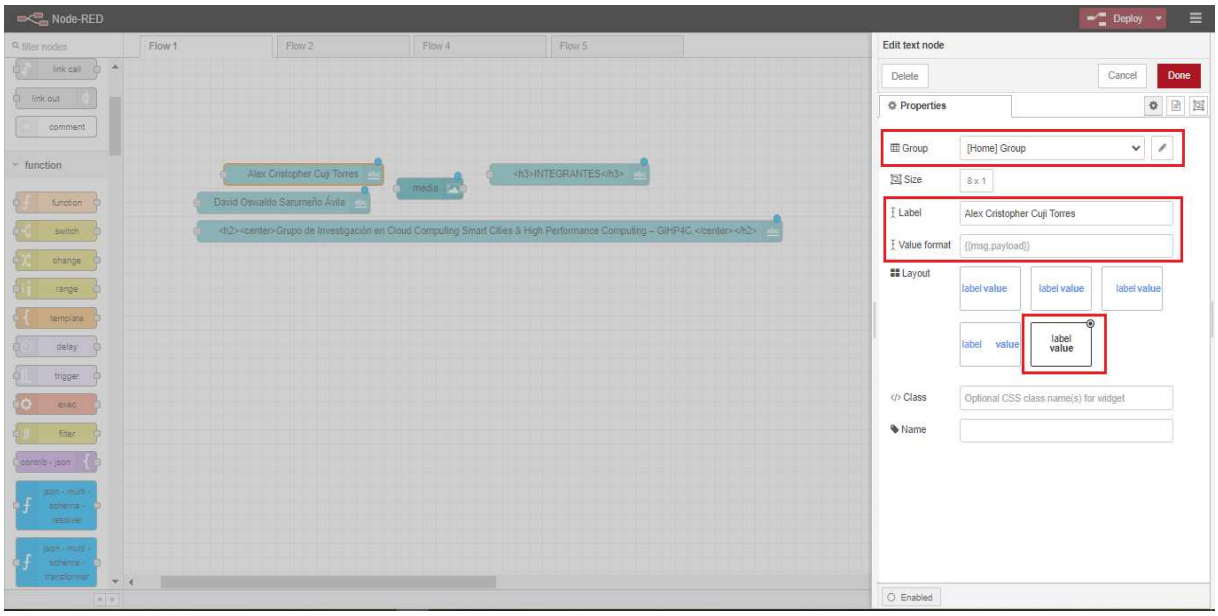


Figura 144. *Edición de nodo text*.

Así mismo con el nodo de media, se tendrá que escoger el menú y grupo de donde se quiere que se muestre ka imagen, además si la imagen sea centrada, en mosaico, expandida, etc. En la sección de files, se tendrá que subir la imagen al sistema, primeramente creado una categoría para luego subir la imagen en esa categoría, como se indica en la FIG.145, y en la FIG.146.

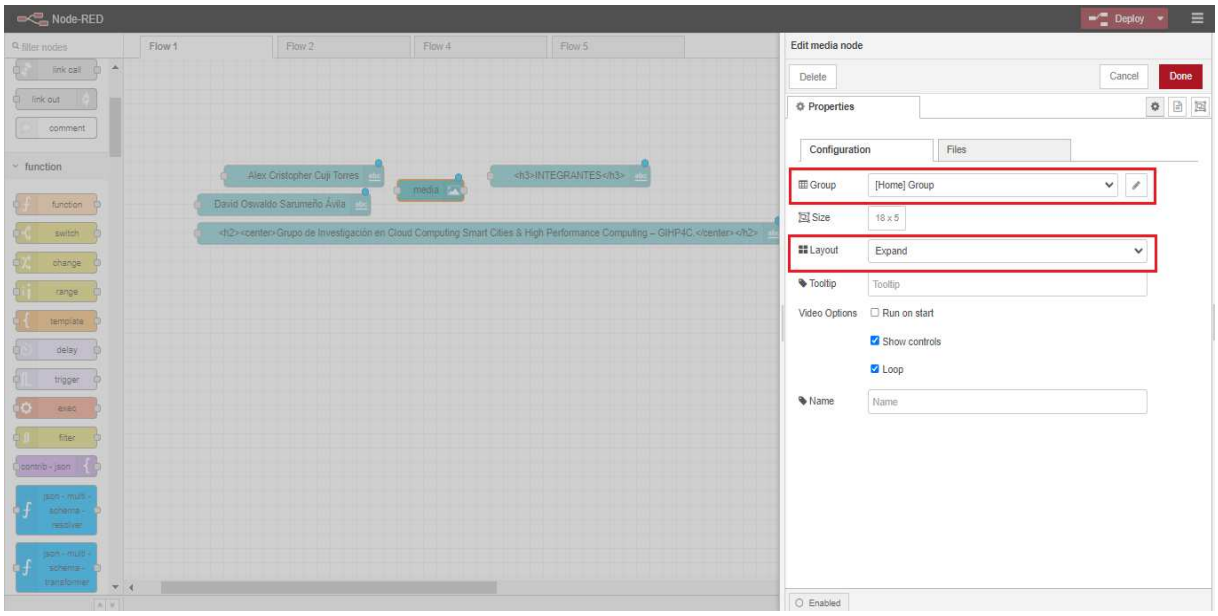


Figura 145. *Edición de nodo media.*

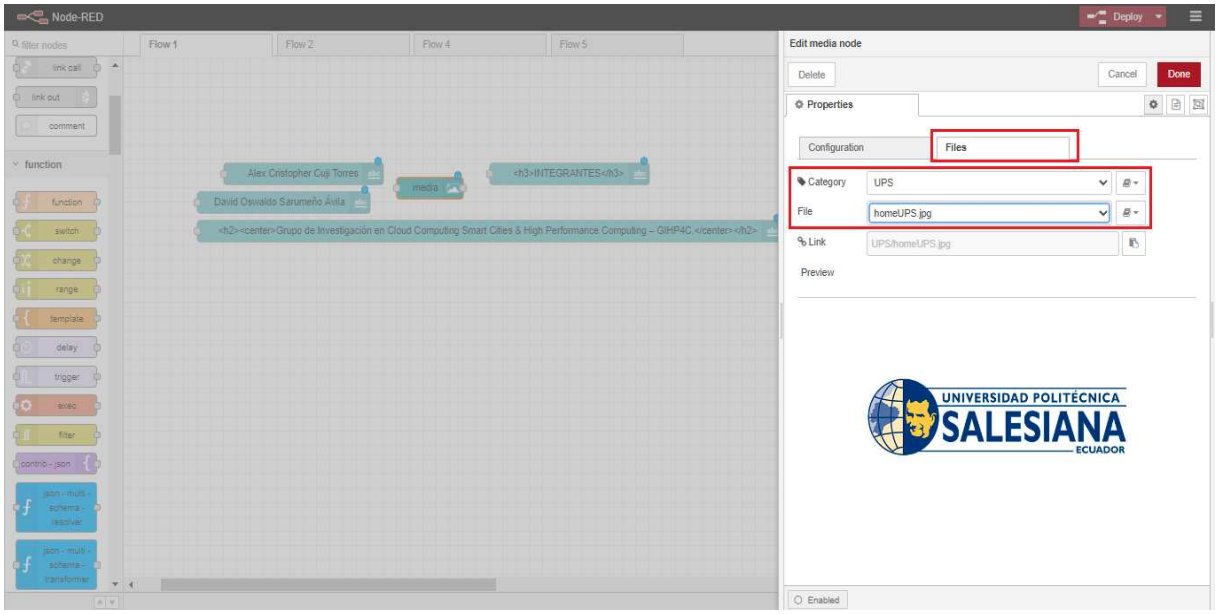


Figura 146. *Subir imagen al sistema del nodo media.*

En el menú de dashboard, como se ve en la FIG.147, se podrá configurar el tamaño y ubicación de los nodos ingresados en ese grupo, para ello se dirigirá al menú en la opción de layout, como se observa en la FIG.148.

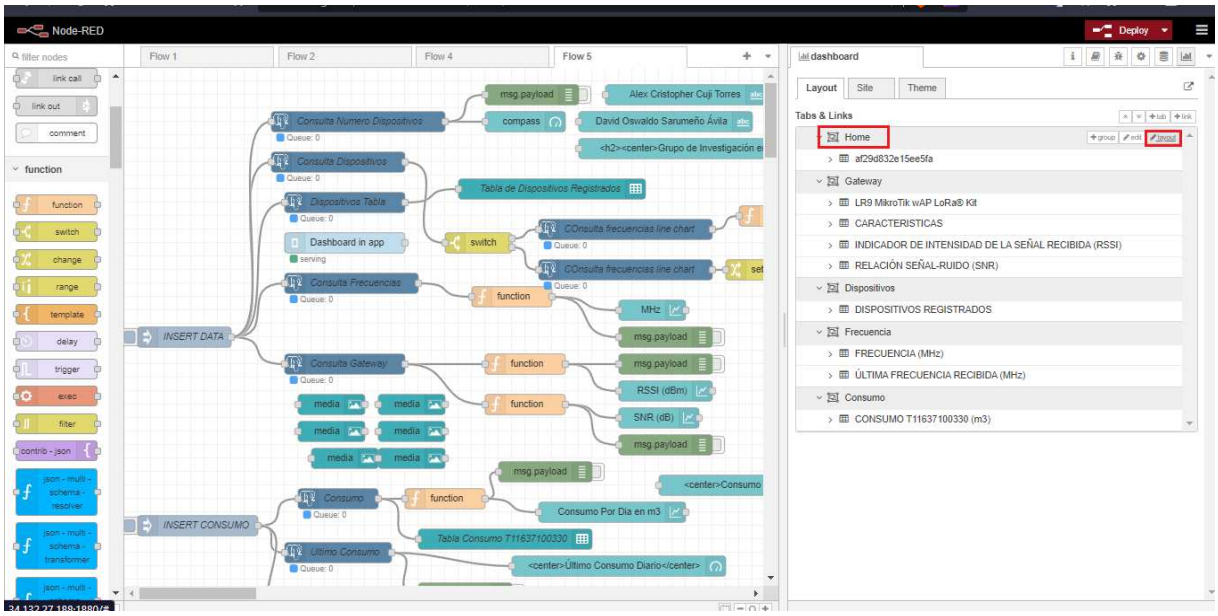


Figura 147. *Ingreso a layout de un menu.*

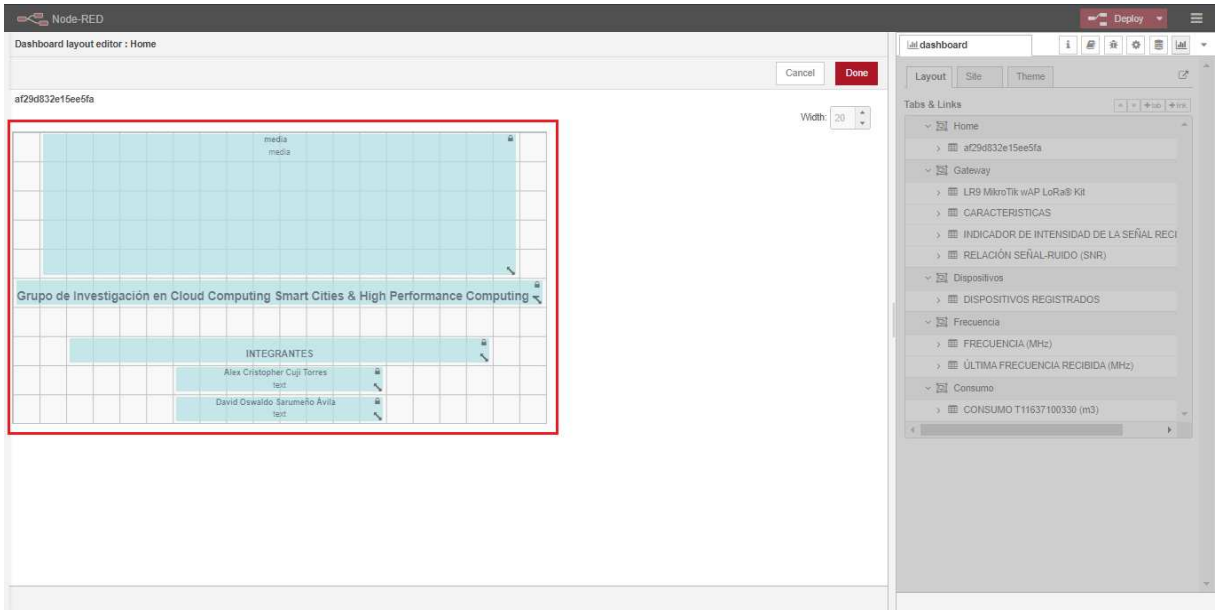


Figura 148. *Configuración de tamaño y ubicación de los nodos.*

Una vez realizado todas las configuraciones, en la FIG.149, se podrá dirigir a la ventana de dashboard mediante la IP 34.132.27.188:1880/ui, ahí se podrá ver creado los distintos menús y la ventana Home configurada recientemente.

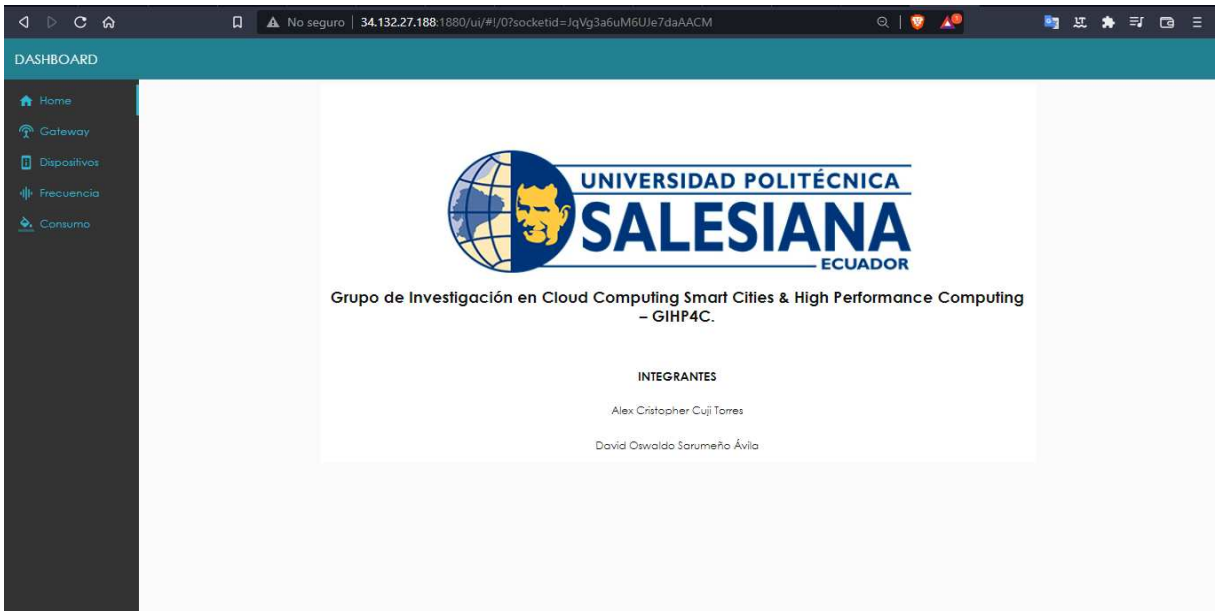


Figura 149. *Ingreso a menú Home en dasboard.*

- **Menú Gateway**

En esta pestaña se realizará una pequeña descripción del tipo de gateway que se usó para la red LoRaWAN, para ello se utilizó el nodo media para insertar imágenes, además de dos gráficas que representan el INDICADOR DE INTENSIDAD DE LA SEÑAL RECIBIDA (RSSI) y la RELACIÓN SEÑAL-RUIDO (SNR), para ello se usó los nodos de inject, postgresql, function, chart y debug, como se aprecia en la Fig.150.

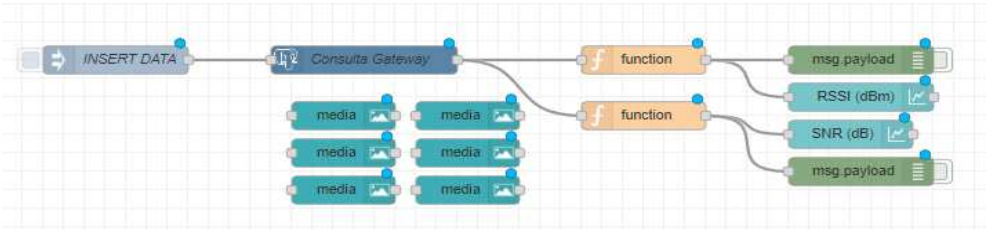


Figura 150. *Nodos del menú Gateway.*

En el nodo ***inject***, se establece el tiempo de ejecución de los nodos, es decir cada vez que se inyecten datos se actualizara los nodos ***chart***, ya que los dispositivos envían

datos cada 20 minutos aproximadamente, entonces las gráficas de líneas de RSSI y SNR se actualizaran y mostraran nuevos datos dentro de ella. Pero estas gráficas tienen un formato para poder reconocer los datos que se vaya a graficar, los datos que se reciben del nodo ***postgresql*** son una lista de objetos, en el que cada objeto contiene una tupla de la base de datos, pero las gráficas de líneas, requieren un formato diferente en el que mediante el nodo ***function***, se podrá cambiar el formato de mensaje y así tener un objeto, dentro de este habrá 3 arrays llamados *series, data y labels*, las series serán los nombres de los dispositivos que se tenga registrado en el servidor, la data son los datos de la gráfica como el RSSI y SNR, mientras que los labels serán la hora y fecha de recibo del dato en el servidor, se establecerá un query en el nodo ***postgresql*** donde se obtendrán esos datos requeridos. Con el nodo ***debug***, se podrán observar como llegan los mensajes a Node-Red.

Como se puede observar en la Fig.151, se establece un tiempo de ejecución del nodo inject.

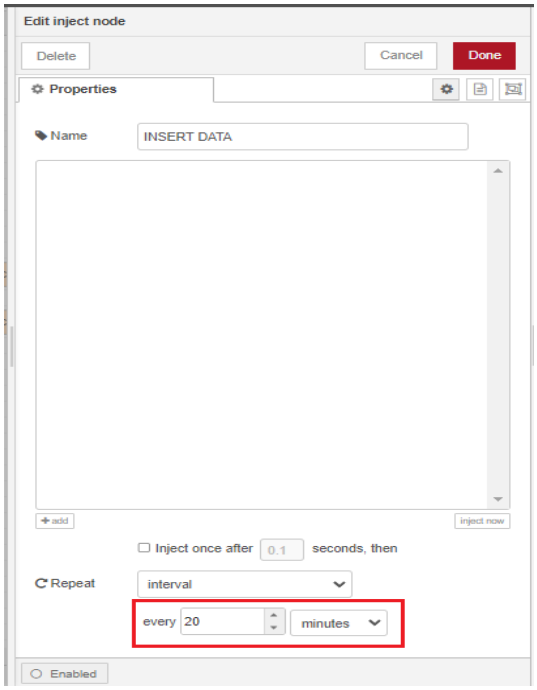


Figura 151. *Tiempo de ejecución de nodo inject.*

En las Fig.152 y Fig.153, se puede observar el ingreso del query y configuración de conexión a la base de datos en el nodo postgresql.

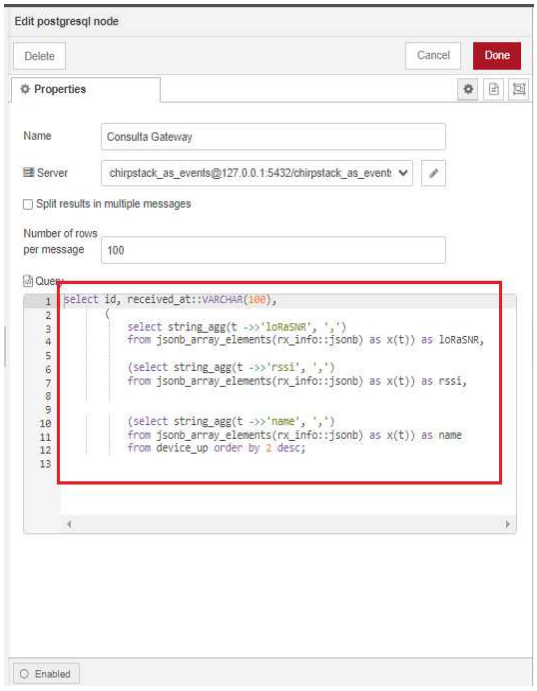


Figura 152. *Ingreso de query al nodo postgresql.*.

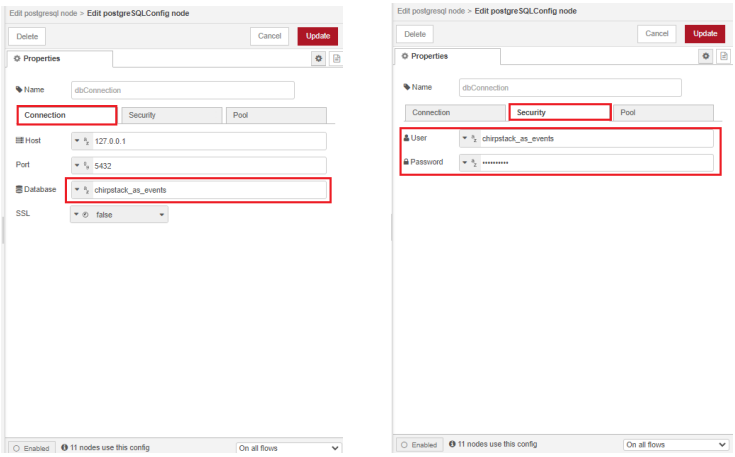


Figura 153. *Configuración de conexión a base de datos*.

En las Fig.154 y Fig.155, se puede observar el resultado del query ingresado en PgAdmin y como llegan los datos a Node-Red.

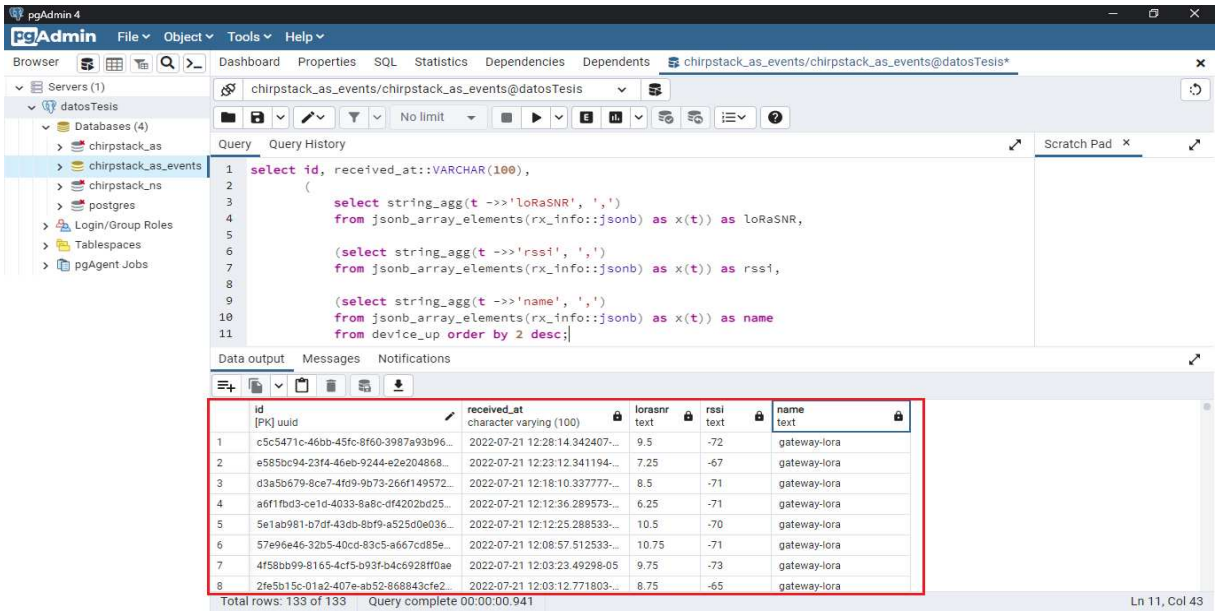


Figura 154. *Resultado de query en PgAdmin.*

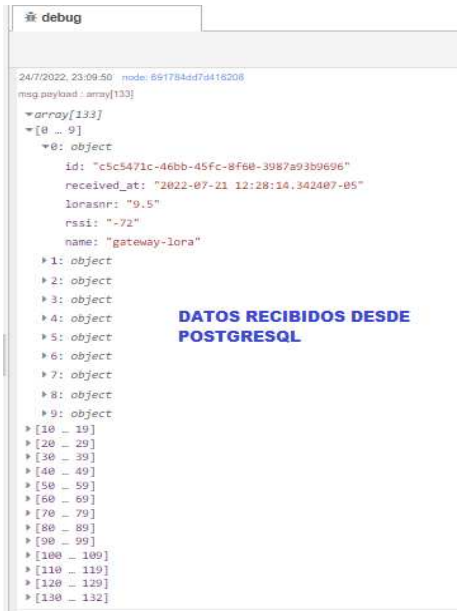


Figura 155. *Datos recibidos en Node-Red.*

Mediante el nodo *function*, se cambia de formato el mensaje recibido directamente del nodo postgresql, para ello mediante código JavaScript, se crean tres matrices de series, data y labels, en el que se debe recorrer la lista de datos que se recibió desde la base

de datos, estos datos se almacenan en una variable llamada *msg.payload*, para ello se guarda esos datos en una variable, en la que mediante un for se recorrerá la lista y se irán guardando los datos en las matrices pertenecientes a cada variable, el nombre del dato que se recibió es el nombre de la columna de cada dato en la base de datos, por último se volverá a guardar en msg.payload como se observa en la Fig.156.

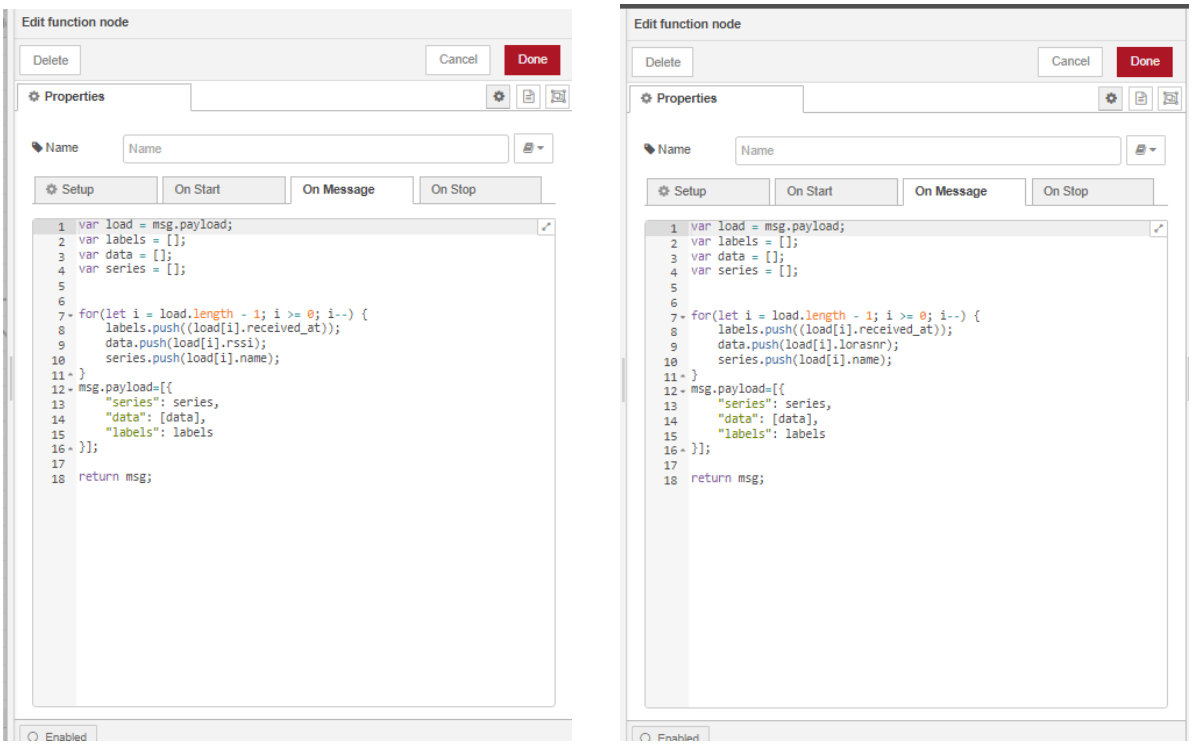


Figura 156. *Cambio de formato de datos para SNR y RSSI mediante nodo function.*

Ahora en la Fig.157, se puede ver como se cambió los datos mediante el nodo function.

Figura 157. *Formato cambiado mediante nodo function.*

Una vez cambiado de formato bastará con conectar estos nodos con los nodos chart para insertar la gráfica, solamente se tendrá que elegir el grupo en donde se quiera que grafique como se observa en la Fig.158.

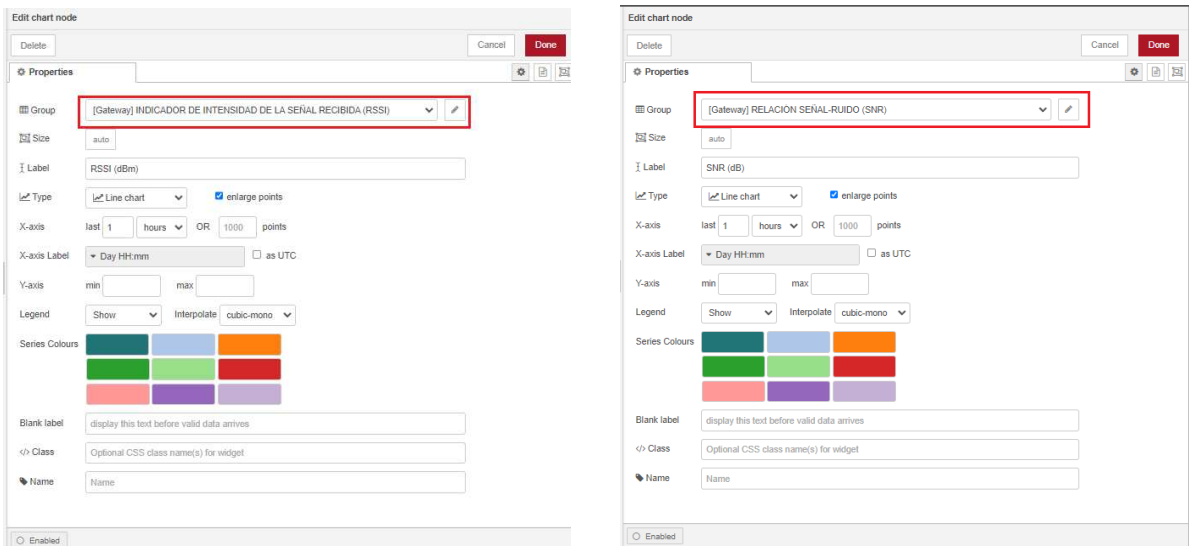


Figura 158. *Configuración del Nodo Chart.*

En la Fig.159, se puede observar el layout del menú Gateway, en donde se cuenta con varios grupos. 2 de ellos contiene solo nodos media donde se ingresó imágenes correspondientes al gateway que se usó, y en los dos siguientes grupos contienen las

gráficas relacionadas con datos de SNR y RSSI.

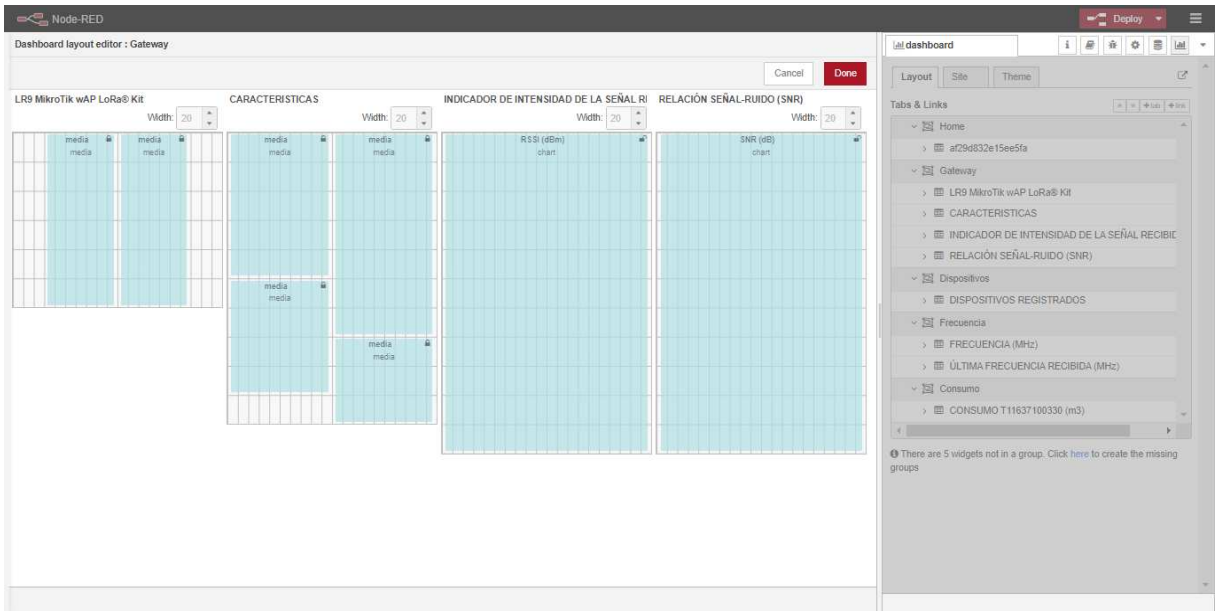


Figura 159. *Layout de menú Gateway*.

Por último, como se observa en la Fig.160, la ventana de dashboard quedaría de la siguiente manera:

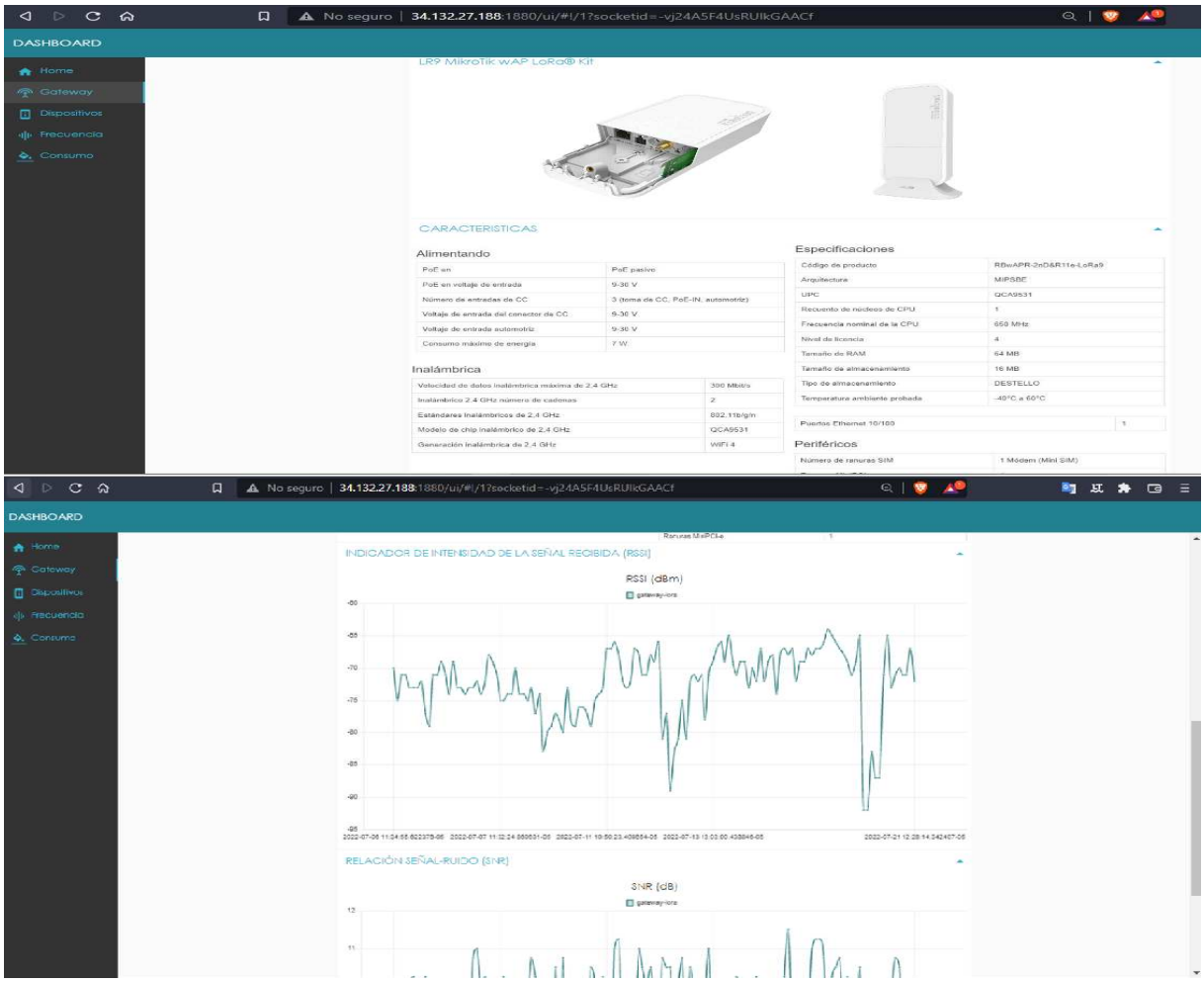


Figura 160. *Ventana de menú Gateway*.

- **Menú Dispositivos**

En esta pestaña se mostrará una gráfica que indique el número de dispositivos que se encuentran registrados en el servidor, y una tabla con destalles de los dispositivos registrados.

En la siguiente Fig.161, se observa los nodos que contiene el menú dispositivos.

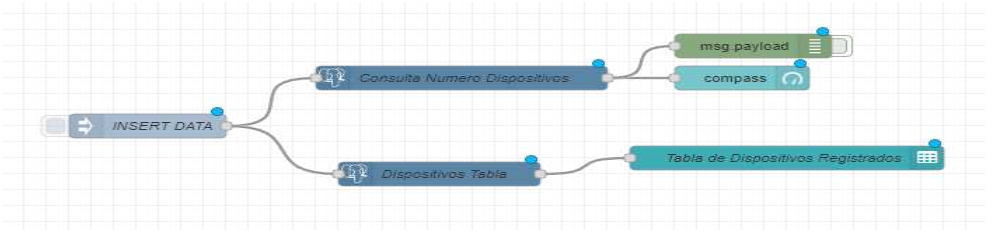


Figura 161. *Nodos de menú Dispositivos.*

Se realizará 2 query para esta ventana como se observa en la Fig.162.

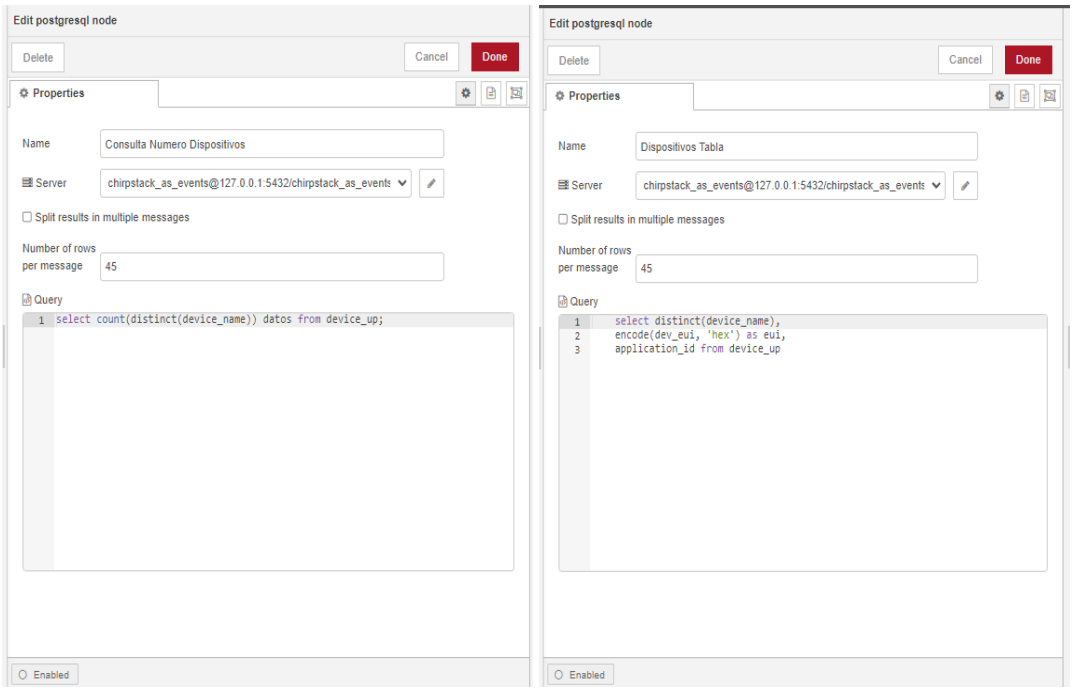


Figura 162. *Querys de menú Dispositivos.*

En la gráfica del número de dispositivos registrados, se utiliza un chart de tipo compass, en este tipo de gráfica recibe únicamente un número, es por eso que se debe imprimir solo ese número en el nodo, para ello en la opción de value format, se imprime el mensaje almacenado en msg.payload, el cual contiene el número traído desde la base de datos, así como muestra la Fig.163.

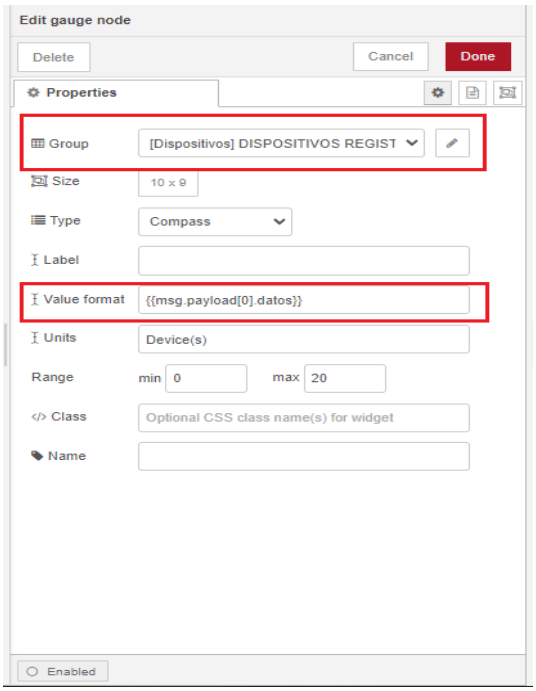


Figura 163. *Ingreso de dato en chart compass*.

Ahora para insertar datos en una tabla se podrá utilizar lenguaje HTML, el nodo table cuenta con una sección donde se puede insertar código HTML, ahí, se imprimirá los datos en la tabla como muestra la Fig.164.

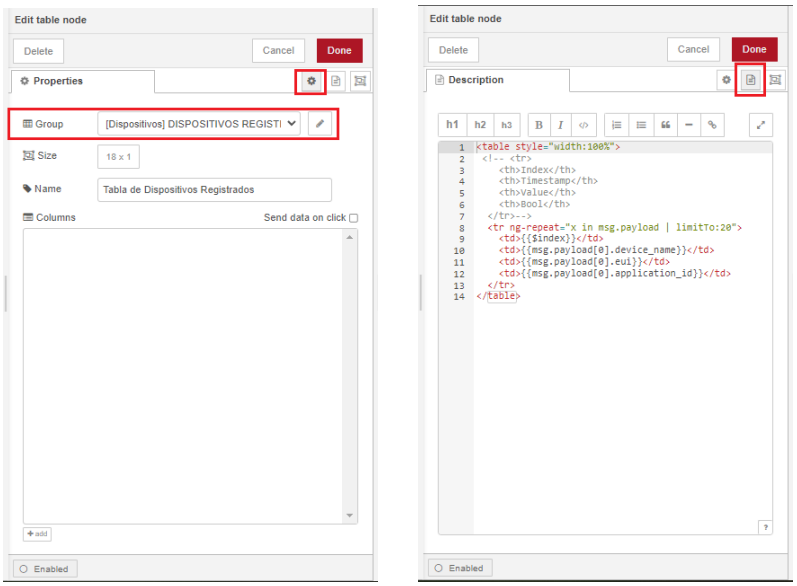


Figura 164. *Código html en table*.

En la Fig.165, se observa el layout del menú de dispositivos.

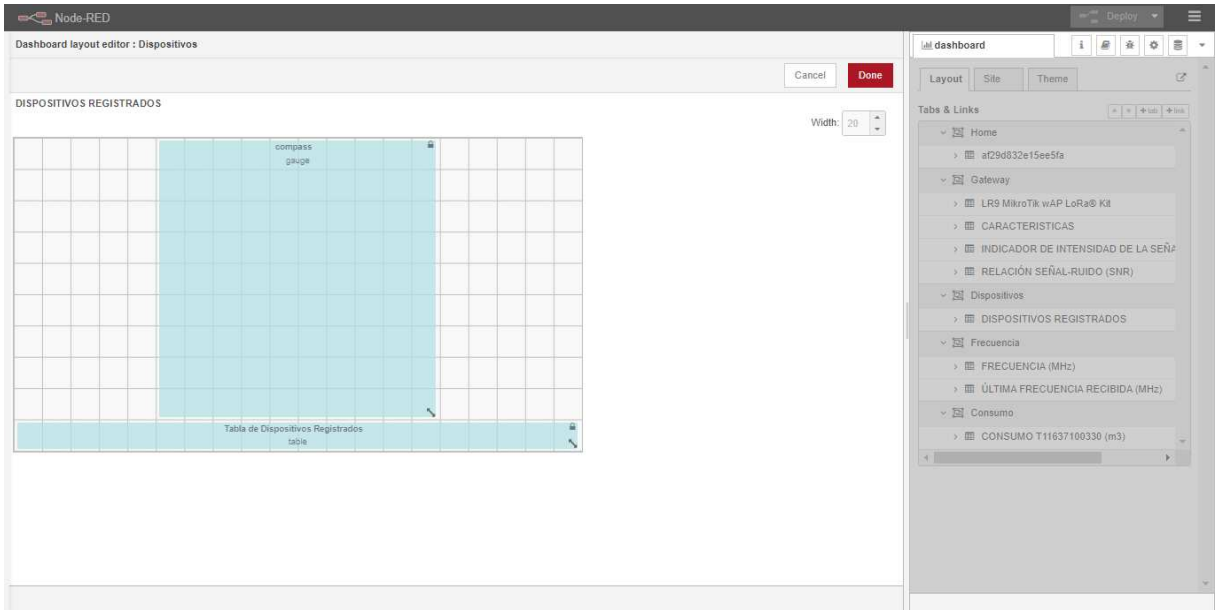


Figura 165. *Layout de menú de Dispositivos*.

Una vez ingresado los datos en la as gráficas, se podrá observar en la ventana de dashboard como indica la Fig.166.

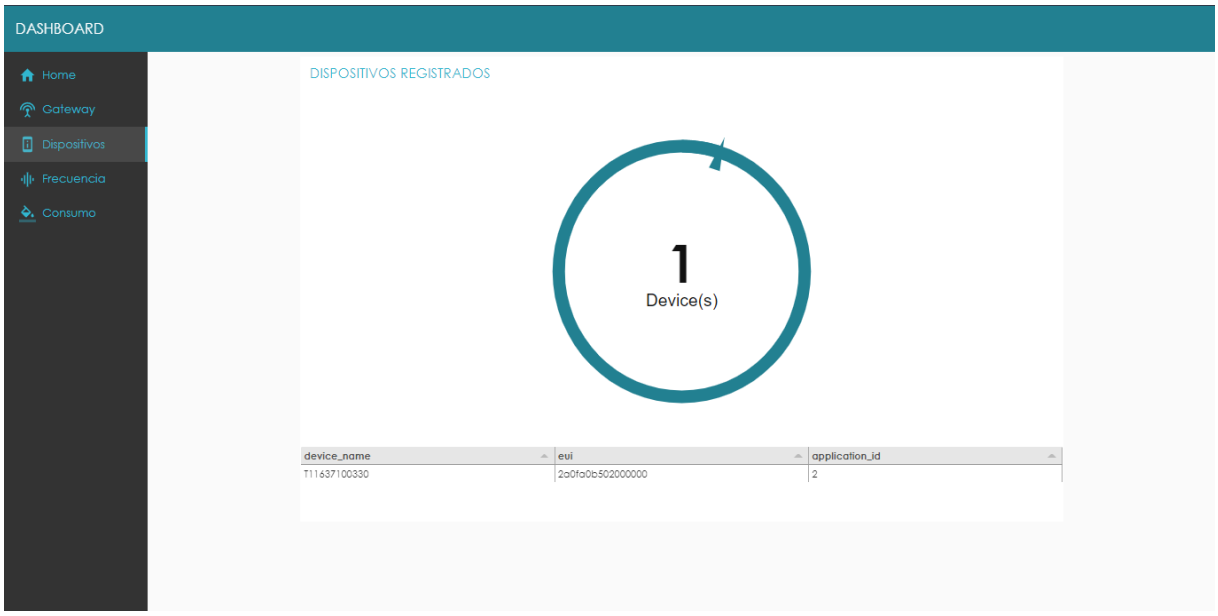


Figura 166. *Ventana de dashboard de menú de Dispositivos.*

- **Menú Frecuencia**

En esta pestaña se mostrará una gráfica de líneas que indique la frecuencia que transmiten los dispositivos, y otra gráfica de barras que representa la última frecuencia de cada dispositivo.

En la siguiente Fig.167, se observa los nodos que contiene el menú frecuencia.

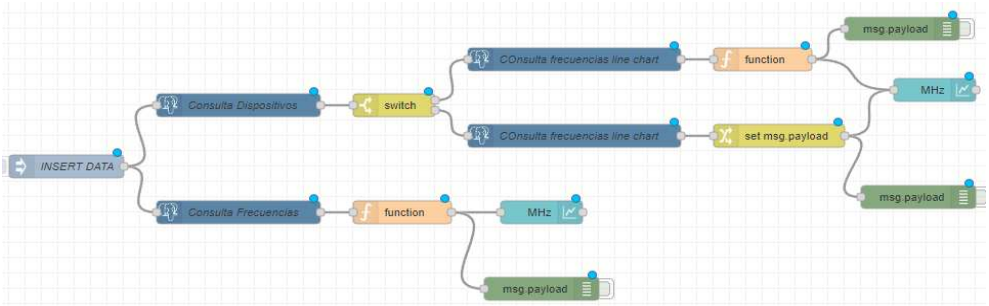


Figura 167. *Nodos de menú Frecuencia.*

Primeramente, se cuenta con un nodo ***inject***, que al igual que en el menú dispositivos y gateway, se ejecutara cada 20 minutos para inyectar nuevos datos a las gráficas, este

inject se divide en 2:

- *Consulta Frecuencias*, es un query que nos devolverá valores de la última frecuencia que recibió el servidor de todos los dispositivos, como sabemos para poder graficar se tiene que cambiar a un formato que la gráfica pueda leer, es por ello que se añade un nodo function para poder cambiar de formato y así poder graficar. Como se aprecia en las Fig.168 y Fig.169, se hace la consulta en el nodo postgresql y se cambia de formato con el nodo function para así poder mostrar datos en un gráfico de barras del nodo chart.

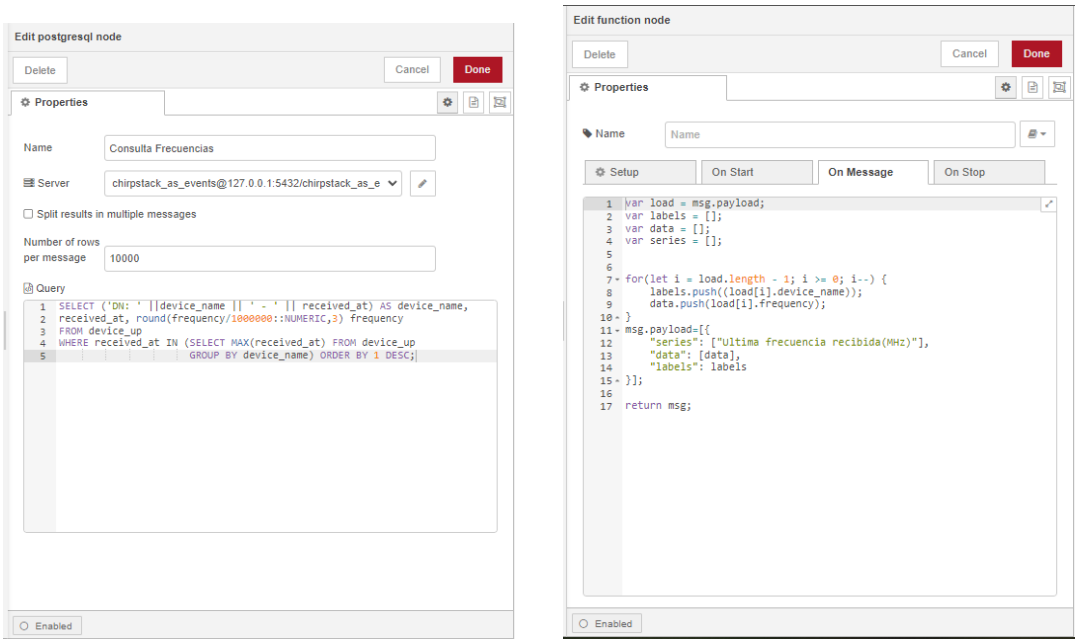


Figura 168. *Ingreso de query y cambio de formato de mensaje mediante function.*

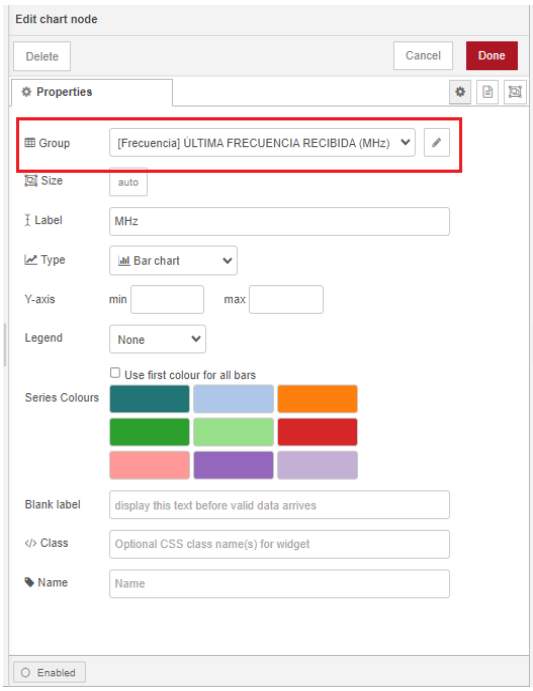


Figura 169. *Configuración de chart*.

- *Consulta Dispositivos*, como se observa en la Fig.170, es un query que nos devolverá el número de dispositivos registrados en el servidor.

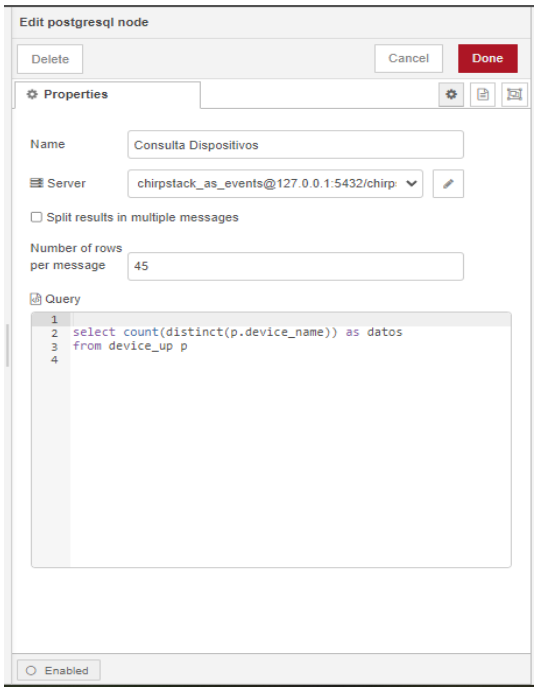


Figura 170. *Query número de dispositivos registrados*.

El nodo ***switch***, evaluará el número de la consulta en el que se tendrá dos opciones de salida, si el número es igual a 1 o mayor a 1, esto se hace, ya que si existe datos de más de un dispositivo, el formato para ingreso de datos en el nodo chart cambia, así que antes de realizar un query para imprimir los datos de la frecuencia se hace este proceso, como se puede observar en la Fig.171, se evaluara la variable msg.payload la columna de salida llamada datos la cual se especifica en el query postgresql.

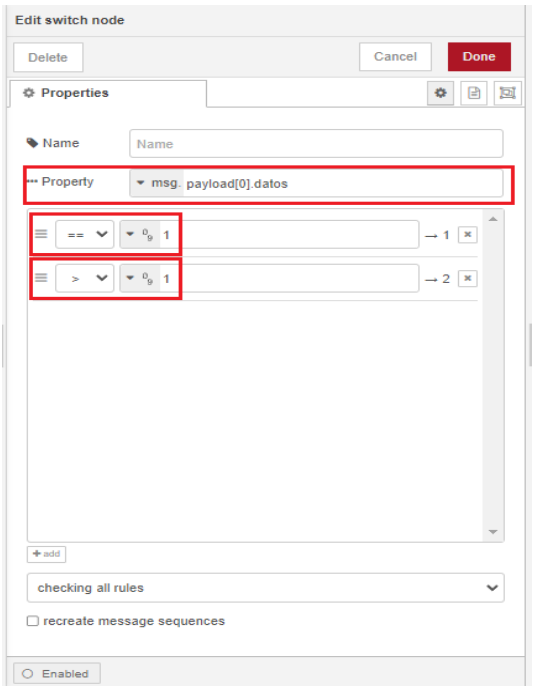


Figura 171. *Configuración de nodo switch.*

Una vez analizado el dato en switch, se realiza nuevamente un query para mostrar datos de la frecuencia, aquí se tendrá 2 opciones:

- Si existe un dispositivo, en este caso se graficaría de la misma manera como se hizo en las gráficas del menú de gateway, se separaran los datos mediante el nodo function, aquí como se trata de un solo dispositivo la lista de series sería un object en específico, como se puede observar en la Fig.172.

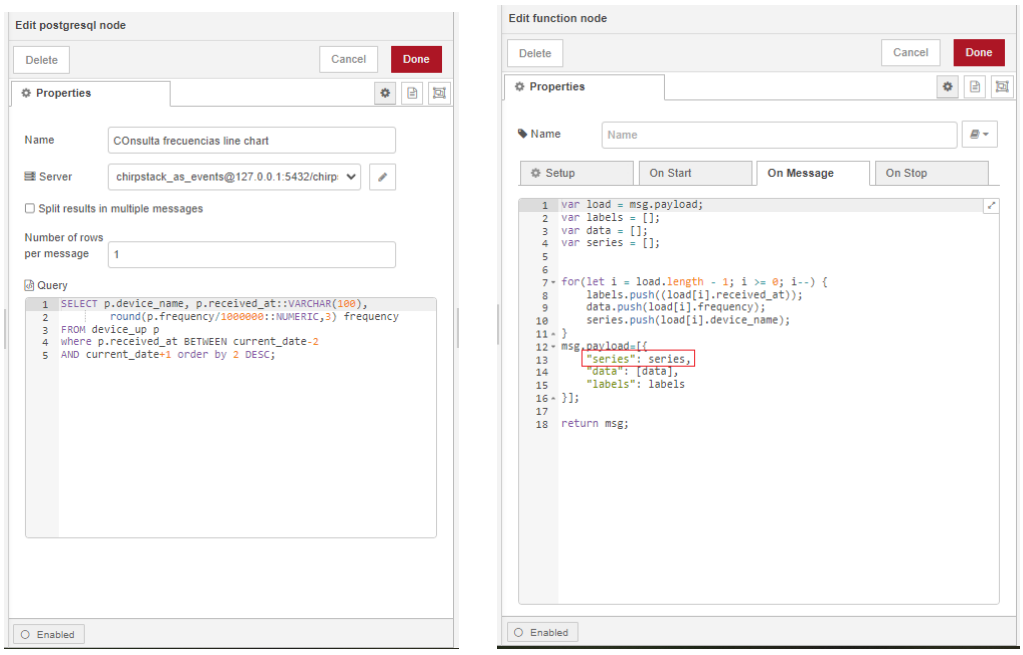


Figura 172. *Consulta de frecuencia y cambio de formato cuando se trata de 1 dispositivo.*

- Cuando hay datos de más de un dispositivo, la consulta de postgresql sería la misma para el otro nodo, lo que cambia es el formato de ingresar datos al nodo chart, en este caso se usa el nodo *change*, el cual es una variante al nodo function, pero sin usar código JavaScript, aquí debemos hacer una operación en donde indique que data pertenece a que dispositivo, para ello se usa un comando each, en donde decimos que tal dispositivo pertenece tal frecuencia en la gráfica como se observa en la Fig.173.

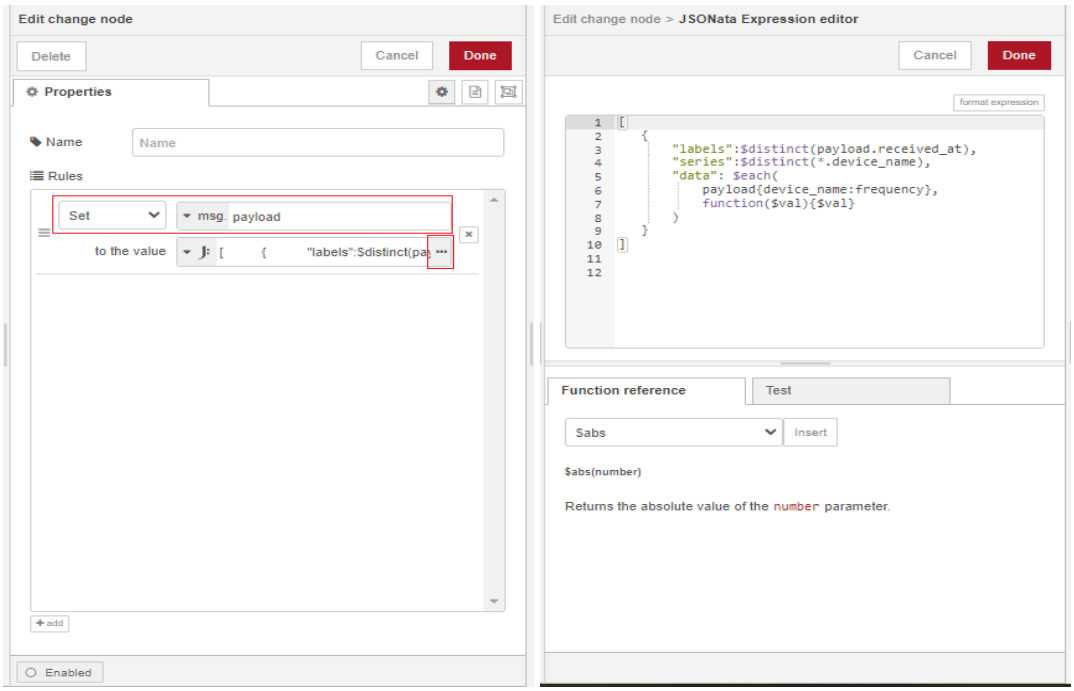


Figura 173. *Cambio de formato en change*.

Por último una vez cambiado el formato ya sea con uno o más dispositivos, se conectara al nodo chart en donde se graficará todos los puntos que se hizo en la consulta postgresql como se observa en la Fig.174.

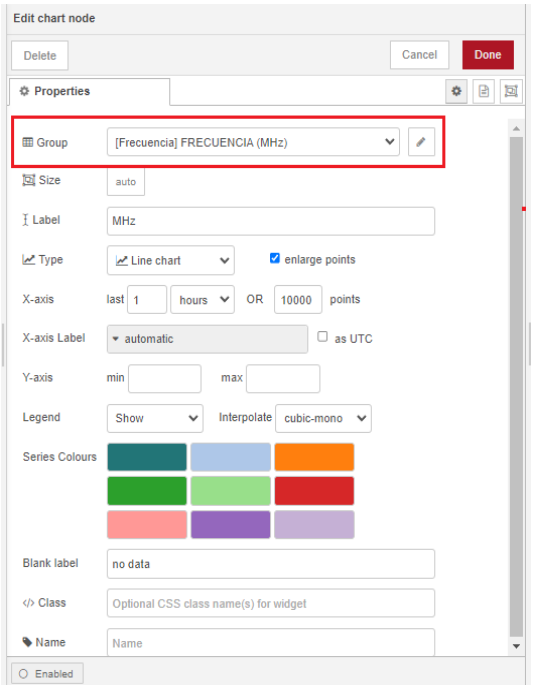


Figura 174. *Configuración de nodo chart de frecuencia.*

En las Fig.175 y Fig.176, se aprecia el layout del menú de frecuencia y la ventana de dashboard donde aparecen las gráficas con datos insertados.

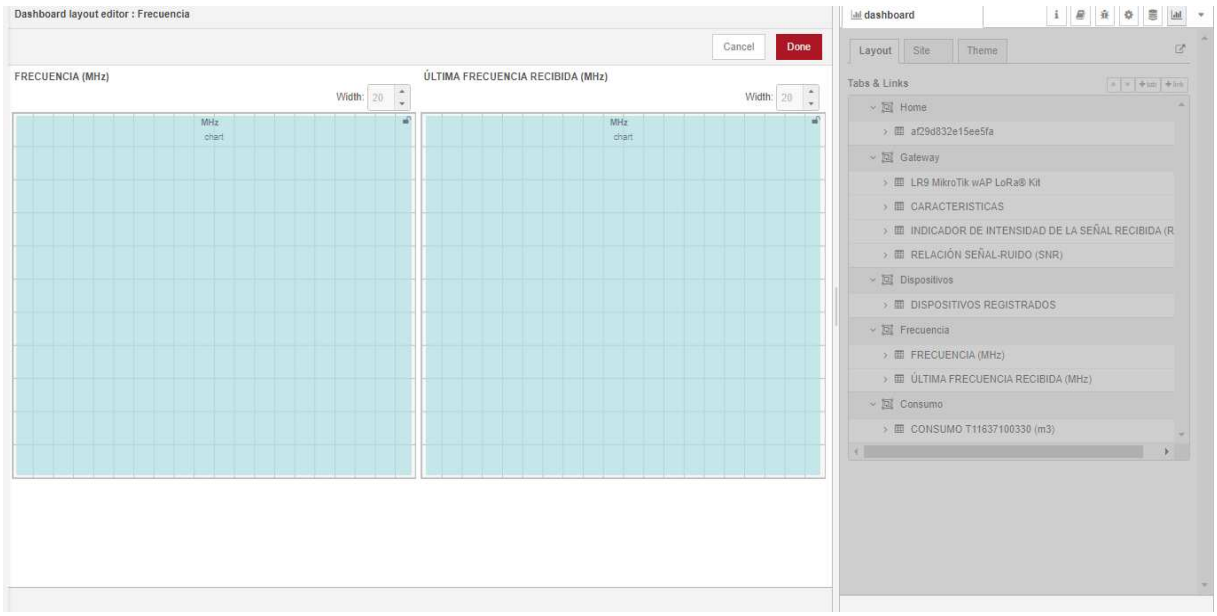


Figura 175. *Configuración de layout de menú frecuencia.*

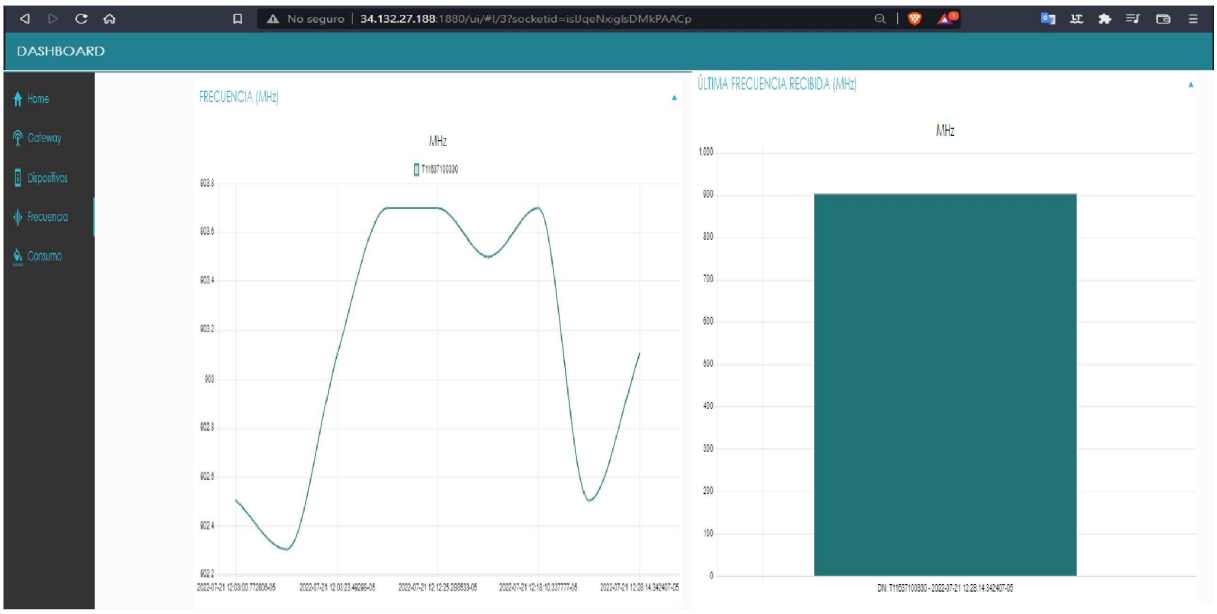


Figura 176. *Ventana de dashboard de menú frecuencia*.

- **Menú Consumo**

En esta pestaña se mostrarán 4 tipos de gráficas correspondiente a cada dispositivo, primeramente se tiene un chart de tipo level, donde se muestra el último consumo que se registró en el día, una tabla con todos los resultados del cálculo de consumo en metros cúbicos de todos los días, un gráfico de líneas que también graficará el consumo calculado de todos los días, y por último una gráfica de pastel que es el consumo mensual del dispositivo igualmente en metros cúbicos.

Como se observa en la Fig.177, estos son los nodos que conforman esta ventana de dashboard.

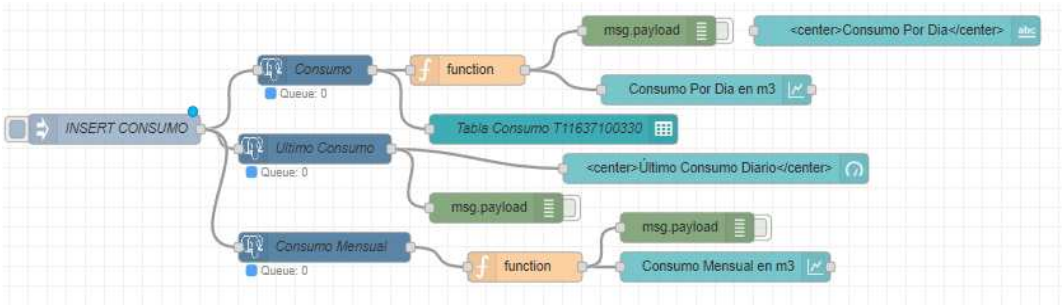


Figura 177. *Nodos del menú consumo*.

Para saber el consumo que hubo durante un día, se deberá realizar una consulta a la tabla device_up, aquella tabla tiene una columna en formato JSON llama object, ahí se encontrara una variable denominada "Consumo". Aquel consumo viene dado en un valor separado por un punto, el número antes del punto representa los dm^3 del dato enviado, el número después del punto represéntalos cm^3 , tal y como muestra la Fig.178.

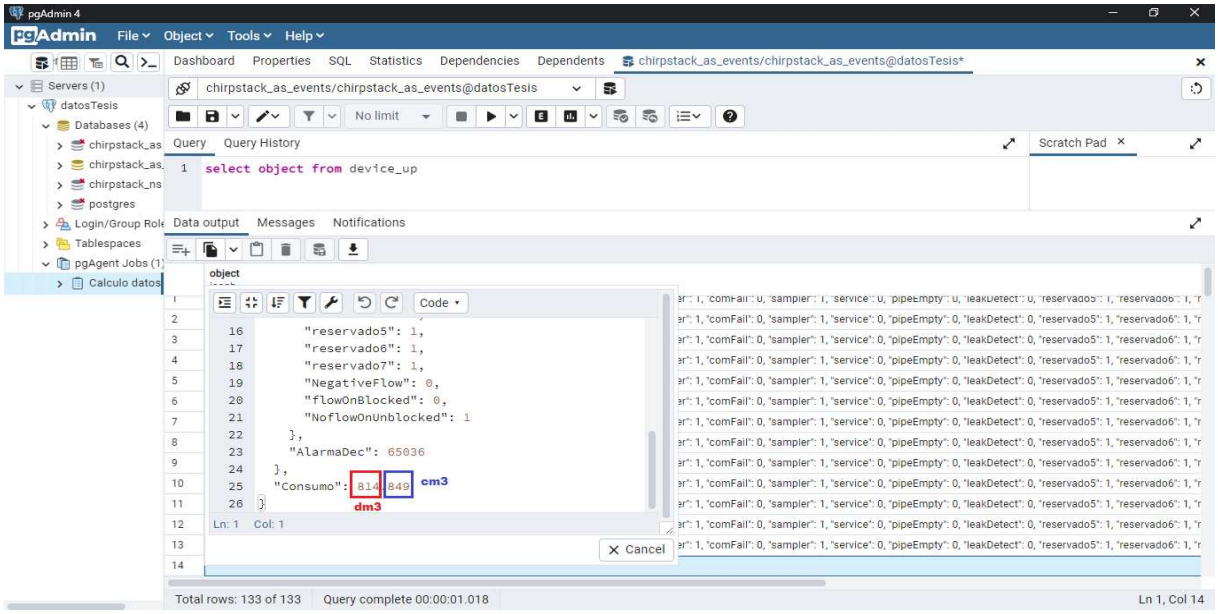
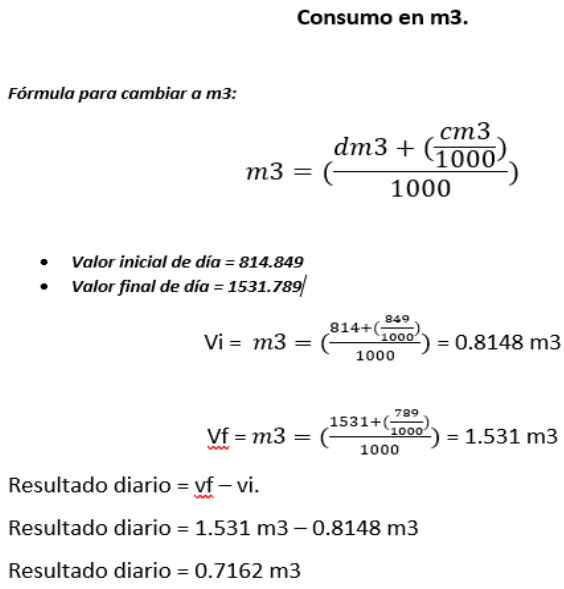


Figura 178. *Resultado del consumo en PgAdmin*.

Para hacer el cálculo a m^3 es necesario una fórmula, la siguiente Fig.179, muestra un ejemplo de cálculo de consumo durante un día, en el que se debe coger el primer dato

recibido en el servidor y el último dato en el día, después se convierte esos valores a m^3 y se hace la resta de estos dos y así se obtendrá el consumo en m^3 durante un día.



Consumo en m3.

Fórmula para cambiar a m3:

$$m3 = (\frac{dm3 + (\frac{cm3}{1000})}{1000})$$

- ***Valor inicial de día = 814.849***
- ***Valor final de día = 1531.789***

$$Vi = m3 = (\frac{814+(\frac{849}{1000})}{1000}) = 0.8148\ m3$$

$$Vf = m3 = (\frac{1531+(\frac{789}{1000})}{1000}) = 1.531\ m3$$

Resultado diario = vf – vi.

Resultado diario = 1.531 m3 – 0.8148 m3

Resultado diario = 0.7162 m3

Figura 179. *Ejemplo de cálculo de consumo en* m^3 *al día.*

Para realizar este proceso se necesitó hacer una separación de los datos de un solo día en una tabla llamada "datosDiarios" como se observa en la Fig.180, ahí se obtendrá los datos durante ese día y el cálculo en m^3 de cada valor que nos devuelva la consulta.

Query | Query History

```
1  ---------------DATOS DIARIOS DEL CONSUMO DE TODOS LOS MEDIDORES---------------
2  delete from datosDiarios*;
3
4  insert into datosDiarios(received_at, device_name, consumo, dcm3, cm3, m3)
5  select d.received_at, d.device_name, d.object ->> 'Consumo' as consumo,
6  split_part( d.object ->> 'Consumo', '.', 1)::DECIMAL as dcm3,
7  split_part( d.object ->> 'Consumo', '.', 2)::DECIMAL as cm3,
8  round((split_part( d.object ->> 'Consumo', '.', 1)::DECIMAL+
9         (split_part( d.object ->> 'Consumo', '.', 2)::DECIMAL/1000))/1000,4)as m3
10 from device_up d
11 where d.object ->> 'Consumo' is not null and d.object ->> 'Consumo' <> ''
12 and d.received_at BETWEEN current_date and current_date+1 order by 2 DESC;
```

Figura 180. *Separación de datos del día en tabla datosDiarios.*

Una vez separado los datos en aquella tabla, se procede a realizar una consulta para saber el primer y último valor de día, los valores que nos devuelva la consulta se almacenaran en otra tabla llamada "separadorT11637100330", esto dependiendo del dispositivo a realizar tal y como se aprecia en la Fig.181.

Query Query History

```
1
2  ------------------ACTUALIZACION CONSUMO INICIAL Y FINAL DE T11637100330----------------
3  delete from separadorT11637100330*;
4
5  insert into separadorT11637100330(received_at, device_name, m3)
6  select received_at, device_name, m3 from datosDiarios
7  where received_at=(select min(received_at) from datosDiarios)
8  and device_name='T11637100330'
9  UNION
10     select received_at, device_name, m3 from datosDiarios
11 where received_at=(select max(received_at) from datosDiarios)
12 and device_name='T11637100330'
```

Data output Messages Notifications

	received_at timestamp with time zone	device_name character varying (100)	m3 numeric
1	2022-07-18 11:24:38.374...	T11637100330	2.0887
2	2022-07-18 12:04:54.396...	T11637100330	2.3114

Figura 181. *Separación de primer y último dato del día.*

Las tablas ***datosDiarios y separadorT11637100330***, se actualizarán todos los días, eliminarán los datos que estén guardados en ese momento para ingresar nuevos del día siguiente, lo contrario pasa con la tabla ***consumoT11637100330***, el cual va a ir ingresando datos cada día, de esta tabla se tomara datos para graficar en el dashboard tal y como muestra la Fig.182.

Query Query History

```
1
2  ------------------DATOS CONSUMO DIARIO EN M3 DE T11637100330 ------------------
3  INSERT INTO consumoT11637100330 (received_at, device_name, medicion)
4  select received_at::DATE, device_name, max (m3) -min (m3) as medicion from separadorT11637100330
5  where received_at::DATE=(select max (received_at::DATE) from separadorT11637100330)  group by 1,2;
6
```

Data output Messages Notifications

	received_at date	device_name character varying (100)	medicion numeric
9	2022-07-14	T11637100330	0.5441
10	2022-07-15	T11637100330	0.3364
11	2022-07-16	T11637100330	0.1102
12	2022-07-17	T11637100330	0
13	2022-07-18	T11637100330	0.2227
14	2022-07-19	T11637100330	0.4156
15	2022-07-20	T11637100330	0.6325
16	2022-07-21	T11637100330	0.2256
17	2022-07-22	T11637100330	0.1485
18	2022-07-23	T11637100330	0.1012
19	2022-07-24	T11637100330	0
20	2022-07-25	T11637100330	0.5023

Figura 182. *Insertar datos de consumo en tabla consumoT11637100330.*.

Estas tablas se actualizarán automáticamente para ello se hará uso de *PgAgent Job*, el cual nos ayuda a realizar tareas programadas mediante scripts como se muestra en las Fig.183, Fig.184, y Fig.185.

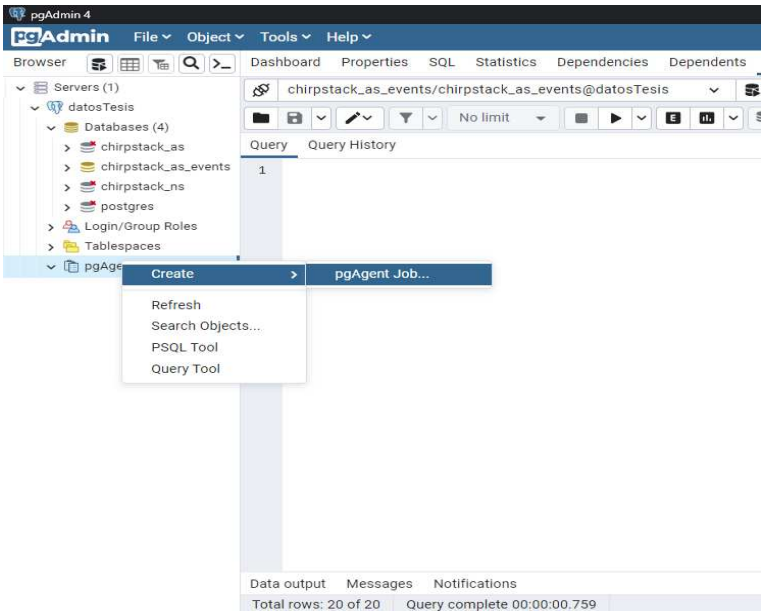


Figura 183. *Creación de PgAgent Job*.

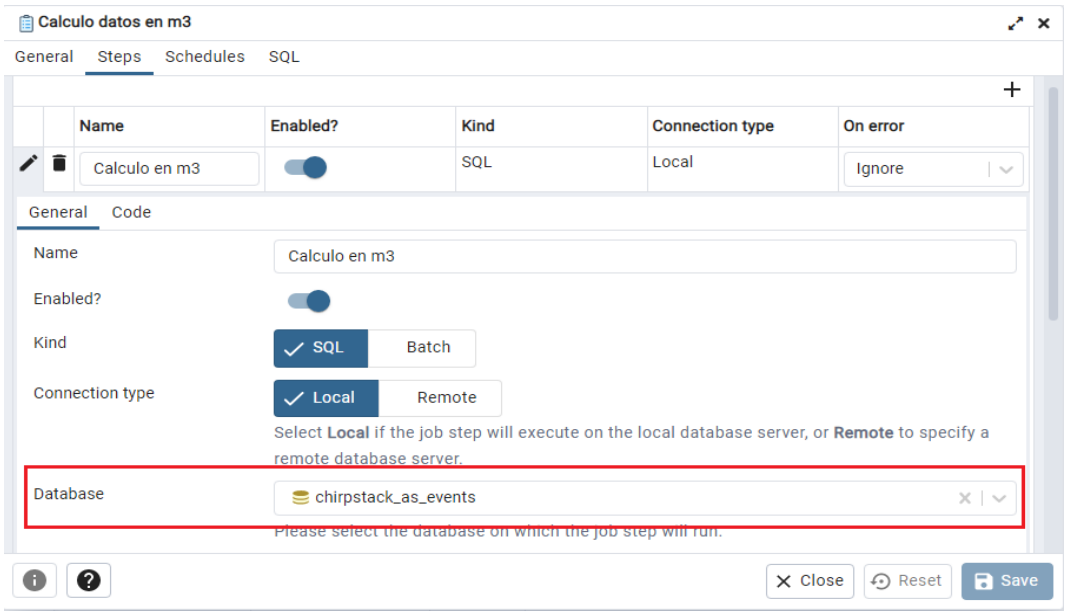


Figura 184. *Creación de Tarea programada.*

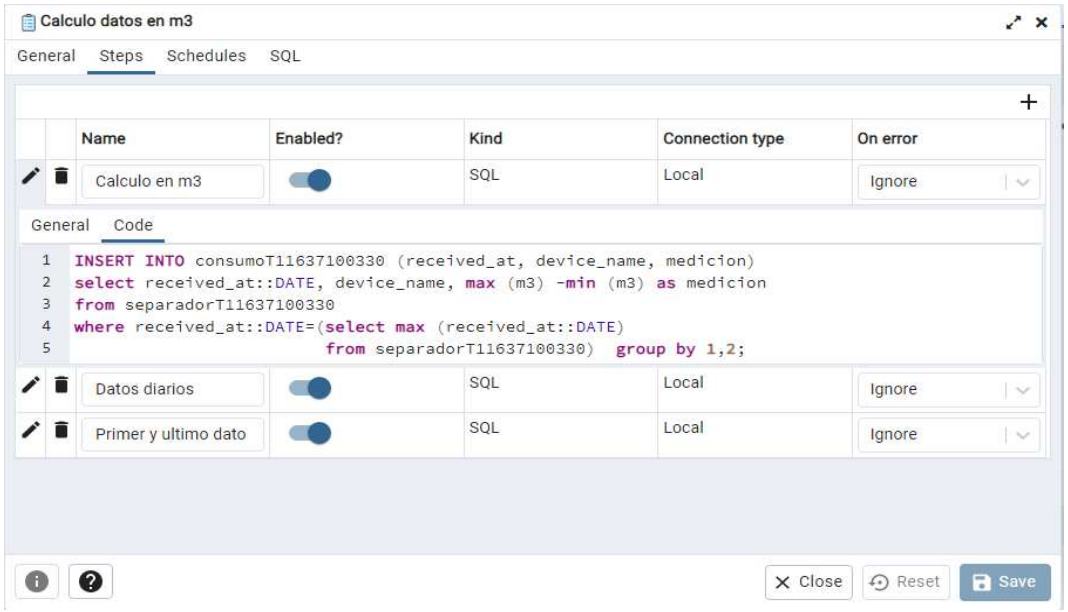


Figura 185. *Creación de Tarea programada.*

Para graficar el dashboard en la ventana de consumo, se lo realizará de la misma manera que en las gráficas explicadas anteriormente, ya que se implementa gráficas de tipo chart y el formato de entrada sería el mismo que las anteriores.

Como se observa en las Fig.186 y Fig.187, tenemos el layout del menú de consumo y el ventana en la página de dashboard.

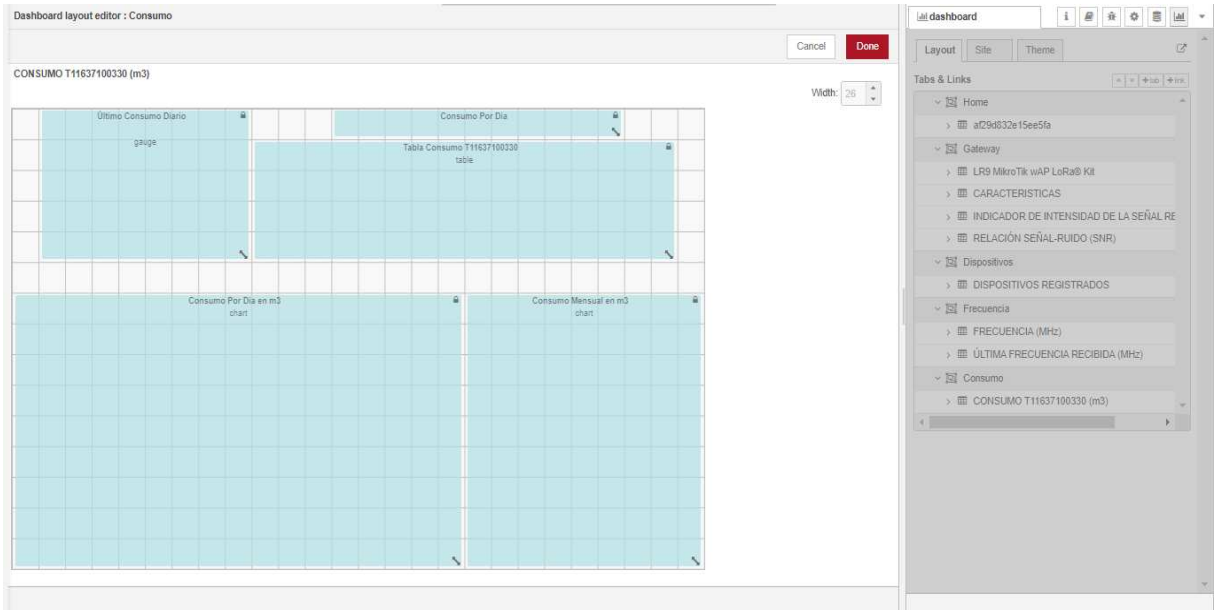


Figura 186. *Layout del menú de consumo.*

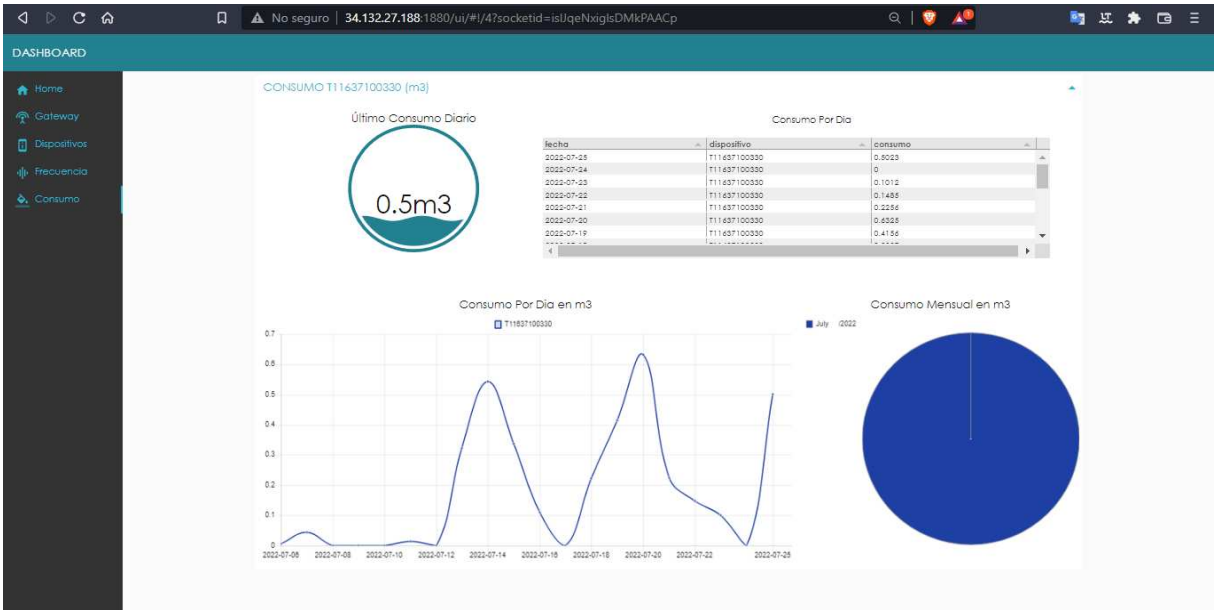


Figura 187. *Ventana del menú de consumo en dashboard.*

Capítulo VI

RESULTADOS

6.1 Prueba de funcionamiento, detección de errores y resultados

En este capítulo se complementa de manera detallada el funcionamiento de los dispositivos, cuya implementación será totalmente documentada para que en un futuro no existan inconvenientes a la hora de implementar más dispositivos para distintas zonas urbanas o rurales.

Se ha implementado una maqueta la misma que simula el consumo de agua en los diferentes dispositivos. Para su implementación, se ha requerido de los siguientes componentes: monitor, Raspberry Pi, medidor de agua ultrasónicos, teclado, mouse, pantalla, llaves de agua con su respectivo lavamanos, motor de agua, tubos, entre otros materiales, tal como se aprecia en la Fig.188.

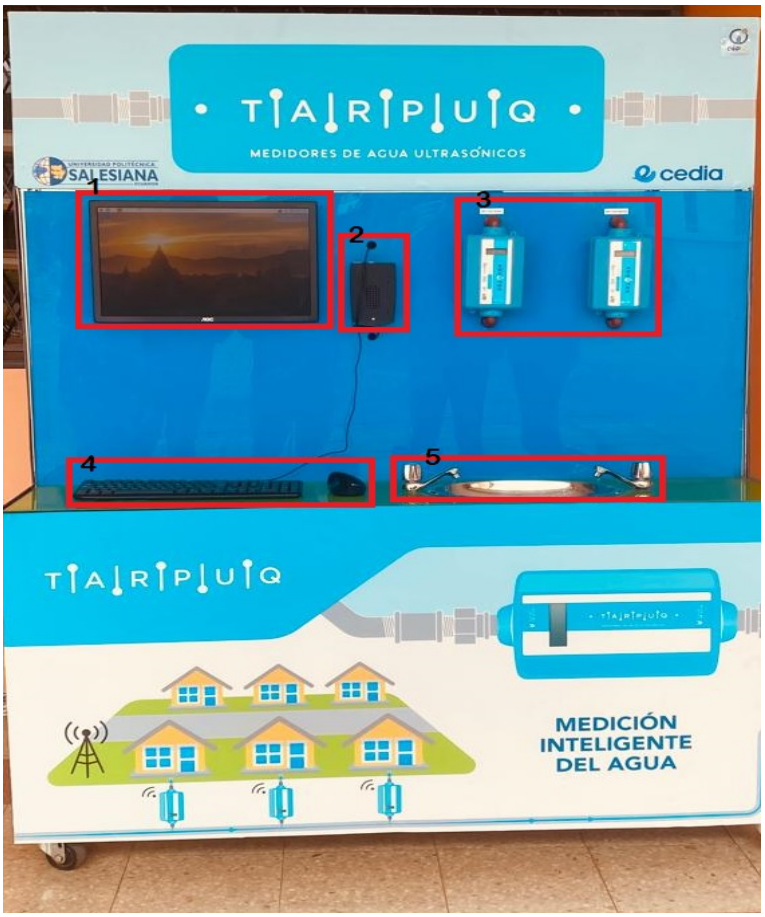


Figura 188. *Maqueta para el flujo de agua a través de medidores ultrasónicos*.

En la conexión para el flujo de agua, se realizó mediante tubos PVC, ver Fig.189, la misma que se conecta a una bomba que ayuda a tener una presión de agua que circula por los tubos, fluyendo por los medidores de agua con el objetivo de que puedan medir el consumo.

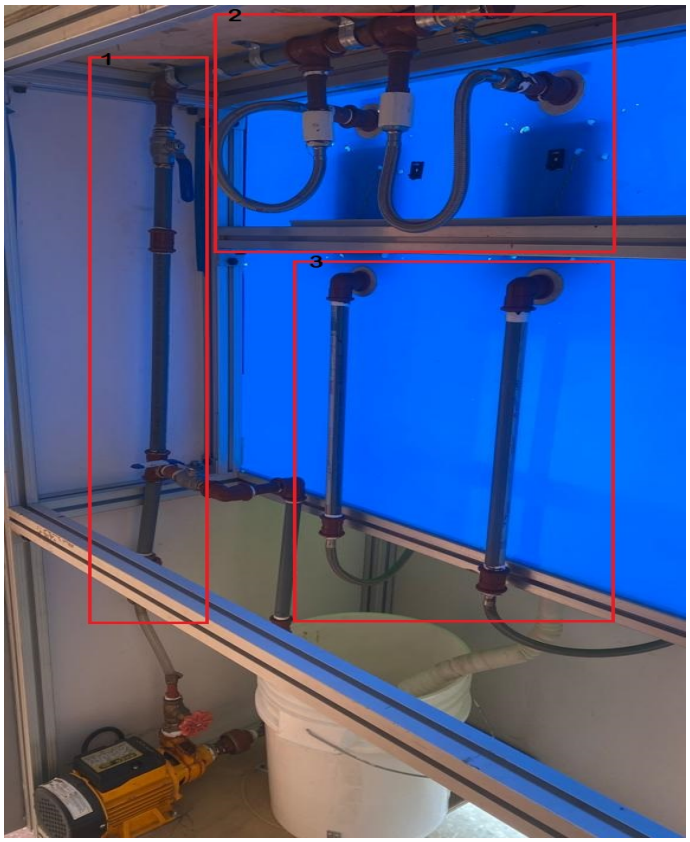


Figura 189. *Conexión para el Flujo de Agua*.

Los medidores envían datos del consumo de agua al servidor, en cuya pantalla se observa un número que son los dm^3 y cm^3 del consumo de agua. Estos datos se envían al servidor cada 20 minutos aproximadamente, para luego mediante node-red hacer el cálculo respectivo en m^3 para conocer el consumo real en el día. Estos valores mostrados en la pantalla de los dispositivos se pueden observar en la Fig.190.

Figura 190. *Medidor de Agua Ultrasónico*.

Una vez hecha todas las conexiones mediante los tubos, se procede a conectar los demás dispositivos de la maqueta, en el que se tiene un *Raspberry Pi*, que sirve para mostrar los datos del dashboard en la pantalla, además de poder conectar a Internet el gateway mediante un puerto de Ethernet del Raspberry. Para ello se realizó algunas configuraciones internas para así poder habilitar el Internet en un puerto Ethernet y que entre ellos se conecten.

6.2 Configuración de Raspberry Pi para dar salida de Internet por Ethernet

Primero se procede a actualizar el sistema de Raspberry PI mediante el comando *sudo apt-get update* y *sudo apt-get upgrade* como se observa en la Fig.191.

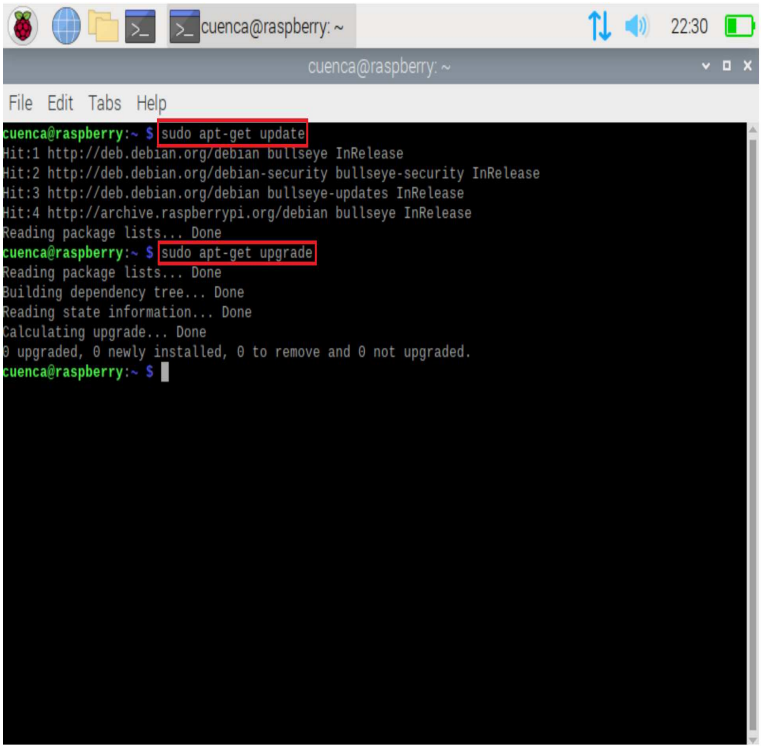


Figura 191. *Actualización de Sistema Operativo*.

Luego se instala *sudo apt-get install network-manager network-manager-gnome*, que permite simplificar el uso de la redes del dispositivo Raspberry Pi, aunque esto también funciona para todos los sistemas operativos, especialmente Linux, como se observa en la Fig.192.

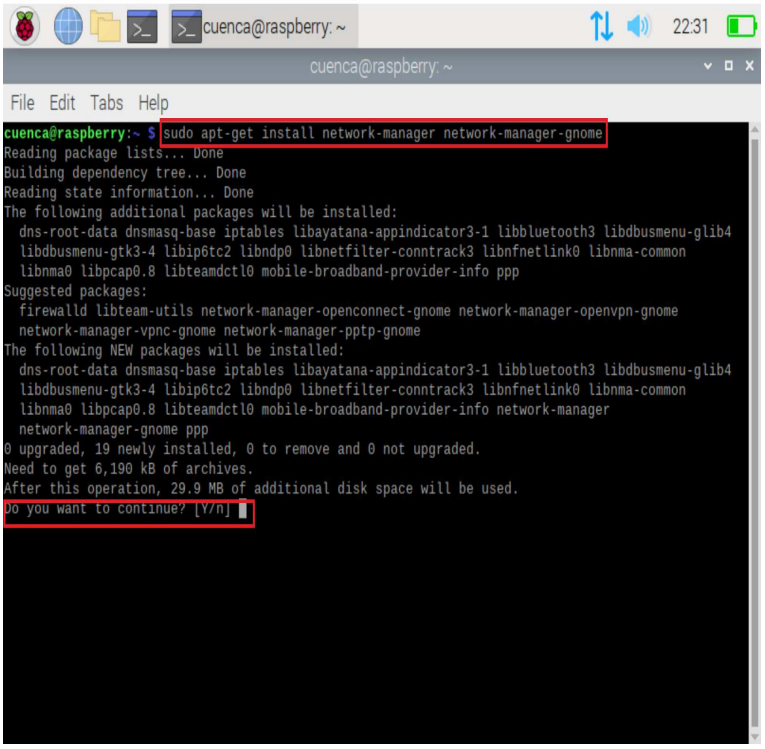


Figura 192. *Simplificar el Uso de Redes*.

En la Fig.193, se observa el comando *systemctl list-units dhcp**, que establece varias direcciones IP asignadas por el servidor de manera aleatoria a varios clientes que se encuentren con la configuración DHCP.

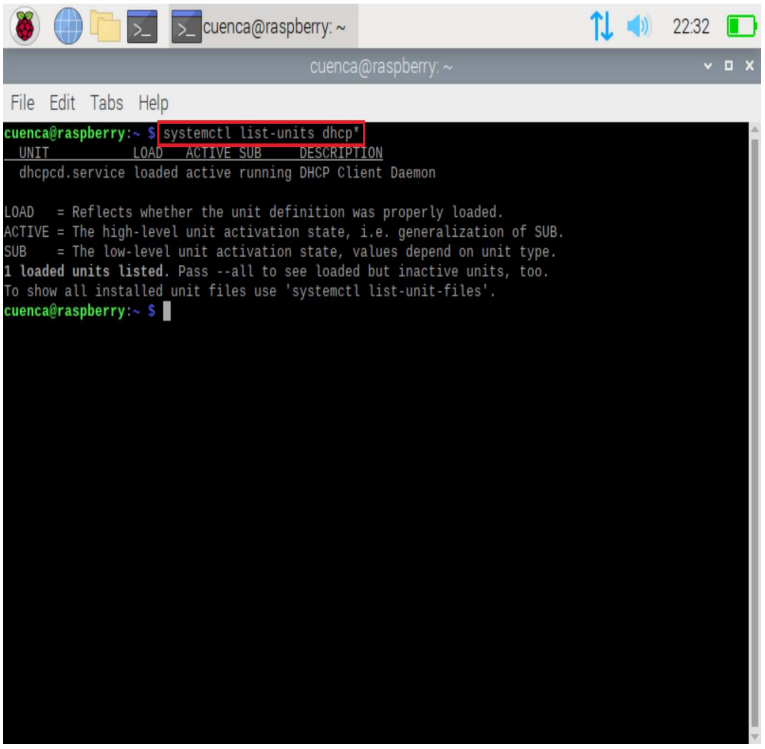


Figura 193. *Establecer varias direcciones IP*.

Inmediatamente, se procede a detener y deshabilitar el servicio de DHCP para que cuando la máquina se inicie, se restablezca la dirección IP a todos los clientes DHCP que se encuentren conectados a este, tal como se muestra en la Fig.194.

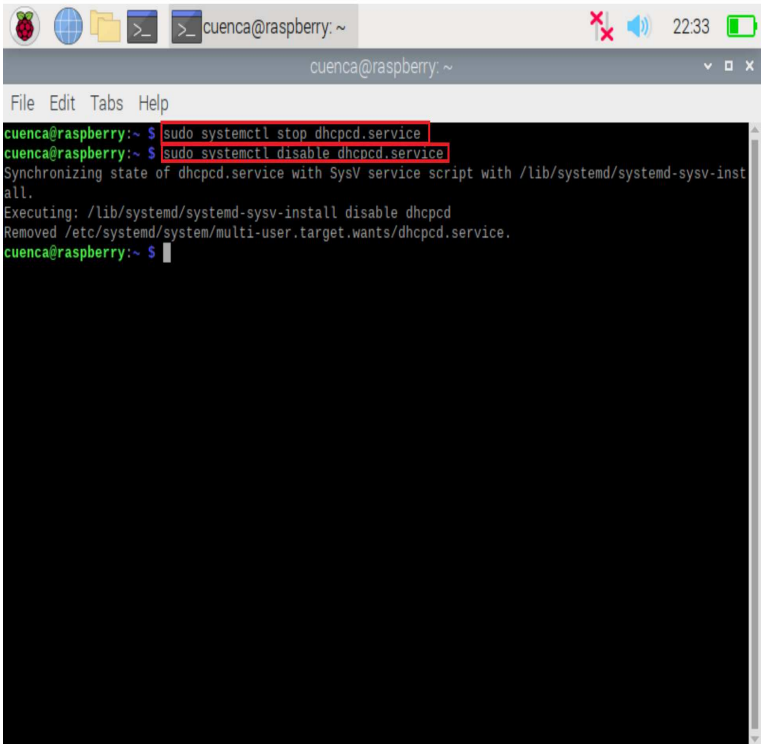


Figura 194. *Detener Servicios de DHCP*.

Como se ve en la Fig.195, el comando *systemctl enable NetworkManager.service*, permitirá reducir el uso de la redes, además es una forma de monitorizar los recursos de la red, también ofrece información super detallada de la interfaz de red, la cual puede ser determinada para diferentes configuraciones.

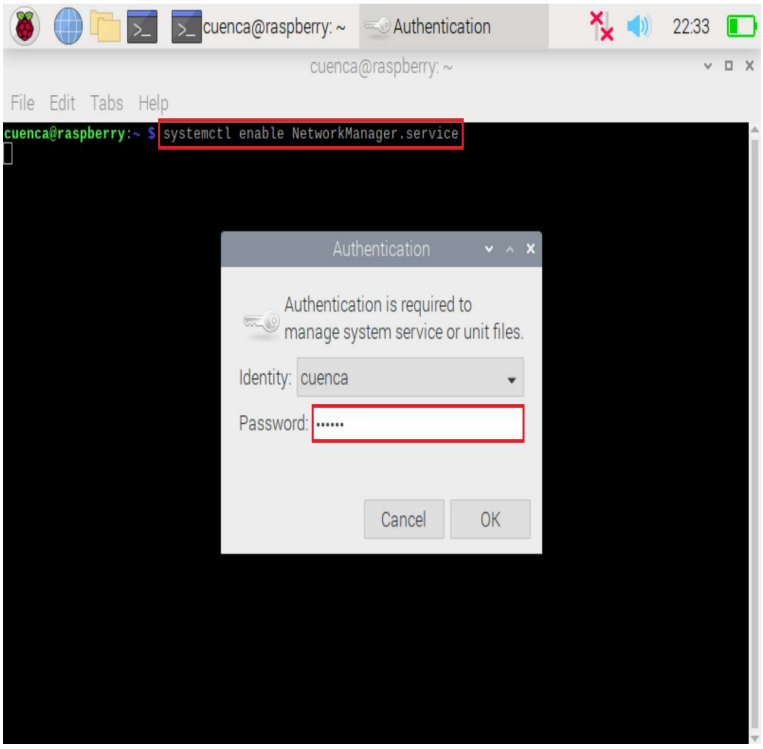


Figura 195. *Levantar Servicios de NetworkManager.*

Por último se levanta el servicio mediante el comando *systemctl start NetworkManager.service*, permite la conexión a diferentes redes, para ello conectamos el puerto Ethernet del Raspberry Pi el cual tendrá salida a Internet y así poder conectar el dispositivo LoRa.

Las Fig.196 y Fig.197, muestra como iniciar el servicio de conexión mediante Ethernet y comprobación del mismo.

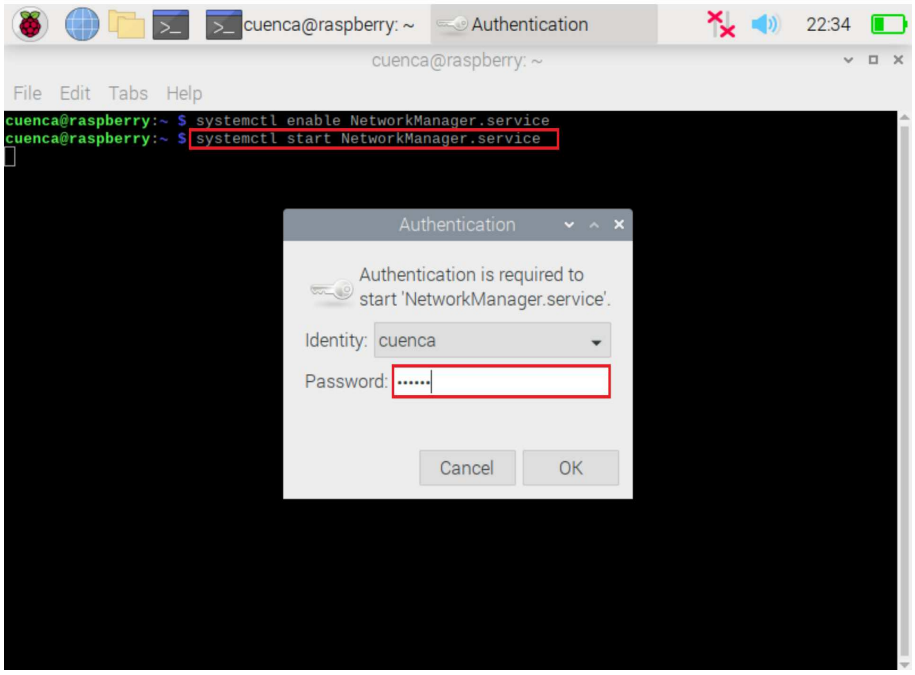


Figura 196. *Iniciar Servicios de NetworkManager.*

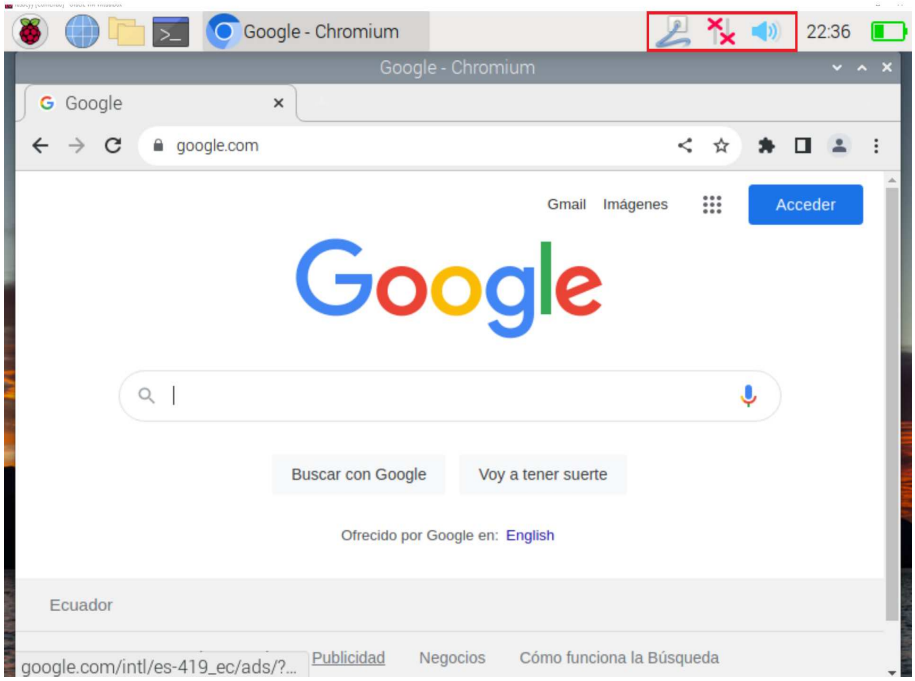


Figura 197. *Servicio de Ethernet con salida a Internet.*

6.3 Detección de Fallas en la maqueta

6.3.1 Falla de bomba de Agua

En un lapso de tiempo la bomba de agua comenzó a generar fallas como recalentamiento, la presión del flujo de agua era muy baja, hubo en un determinando tiempo que ya no recorría el agua por el tubo de PVC, así que decidimos cambiar la bomba de agua, pero vimos obligados a realizar otros cambios.

La Fig.198 muestra la bomba de agua con falla.

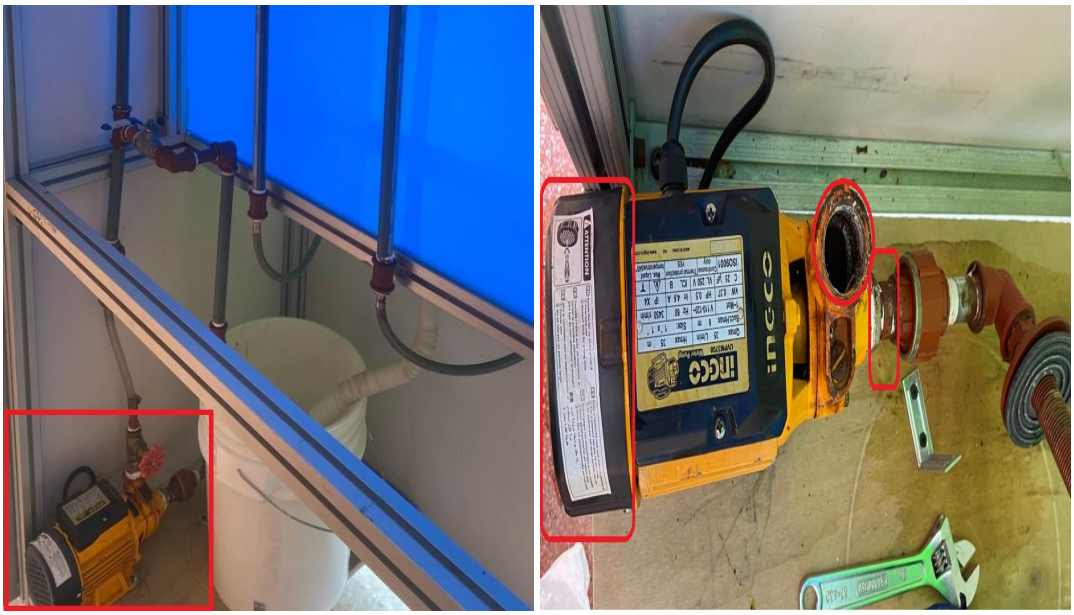


Figura 198. *Bomba de Agua con Falla*.

La Fig.199, muestra la nueva bomba de agua instalada en la maqueta, la cual funciona de excelente manera con una buena presión de agua lo que arreglo el problema inicial.

Figura 199. *Bomba de Agua con correcto Funcionamiento* .

6.3.2 Falla en llave de paso

Se verificó que en la llave existía obstrucción por la vía donde circulaba el agua, ya que la llave se encontraba abierta, pero en ella existía teflón dentro del paso de agua de la llave, y el eje que permite hacer la rotación de abrir o cerrar se encontraba atascado y no permitía ni cerrar ni abrir la llave, así que se procedió el cambio de la misma para el correcto flujo del agua, ya que esa llave es la vía principal para que el flujo de agua que sale por la bomba circule por ella. En la siguiente Fig.200, se observa la llave de paso dañada.

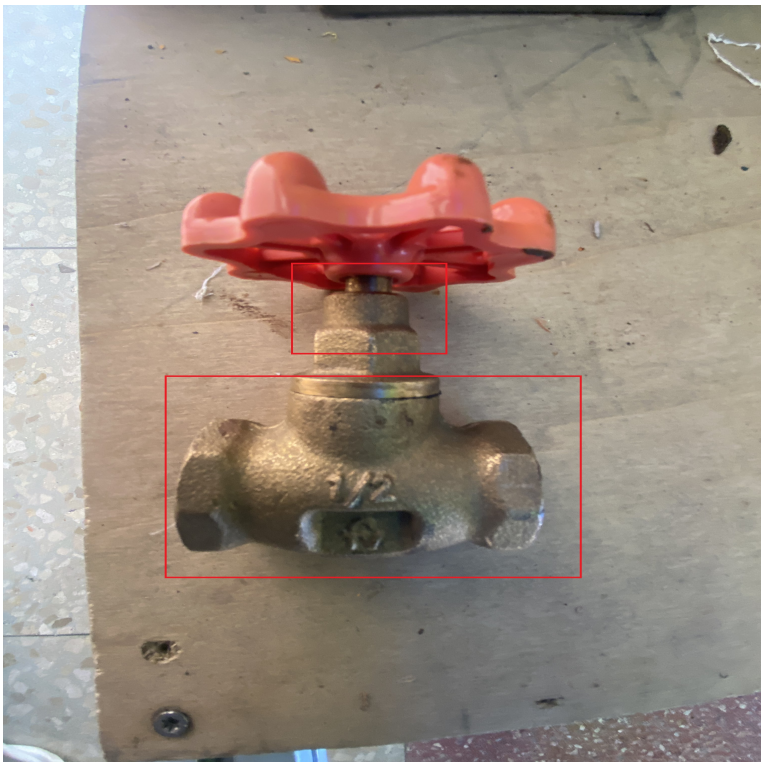


Figura 200. *Llave de paso Agua con falla.*

Al momento de colocar las uniones se observó que existían uniones sin rosca la cual se tuvo que cambiar por la razón que filtraba agua por los costados además se le colocó teflón para evitar que con el tiempo exista algún tipo de filtrado de agua y pueda ocasionar cualquier inconveniente.

6.4 Resultados

6.4.1 Funcionamiento de la maqueta

Después de varios procesos de arreglo por diferentes motivos en la maqueta, se obtiene como resultado una maqueta completamente funcional como lo podremos observar en la Fig.201.

Figura 201. *Maqueta completamente funcional.*

Por último, se puede observar cómo los medidores envían datos al servidor Chirp-Stack, mediante el gateway con conexión a Internet, y así con la ayuda de Node-Red, poder armar un dashboard con estos datos. La Fig.202, muestra el dashboard de consumo

en la pantalla de la maqueta, el cual se irá actualizando cada vez que haya flujo de agua y se inserten nuevos datos a la base de datos del servidor ChirpStack.

Figura 202. *Dashboard en pantalla de la maqueta*.

Capítulo VII

CRONOGRAMA

Como se observa en la Tabla 9, se tiene el diagrama de actividades seguido durante todo el proyecto.

Actividad	Fecha	Horas
Estudio de los fundamentos de LoRaWAN	11/04/2022 - 14/04/2022	20
Estudio del servidor ChirpStack	15/04/2022 - 19/04/2022	15
Estudio del software Node Red y su implementación con ChirpStack	20/04/2022 - 22/04/2022	15
Estudio del software de alojamiento de datos Google Cloud Platform	25/04/2022 - 29/04/2022	25
Estudio de los medidores inteligentes	02/05/2022 - 03/05/2022	10
Estudio de los métodos de comunicación entre cliente y servidor	04/05/2022 - 06/05/2022	15
Implementación de servidor ChirpStack en máquina virtual de Google Cloud Platform	09/05/2022 - 03/06/2022	100
Implementación de Node Red en máquina virtual Ubuntu de Google Cloud Platform	06/06/2022 - 08/06/2022	18
Instalación de APIs, diseño y desarrollo de nodos para creación de dashboard en Node Red	09/06/2022 - 08/07/2022	132

Diseño del plan de prueba del sistema	11/07/2022 - 12/07/2022	12
Verificación de sistema y detección de posibles fallas o errores	13/07/2022 - 15/07/2022	14
Corrección de errores y verificación	18/07/2022 - 21/07/2022	24

Tabla 9. Cronograma de Actividades.

- **Horas totales de Alex Cristopher Cuji Torres:** 400
- **Horas totales de David Oswaldo Sarumeño Avila:** 400

Capítulo VIII

PRESUPUESTO

DENOMINACIÓN	Cantidad	Costo Unitario	Costo Total
	Unidades	Dólares	Dólares
1.Bienes			
Copias	588	0,10	58,80
Impresiones	196	0,12	23,52
Empastados	4	12	48
2.Tecnológico			
Computador Portátil	2	900	1800
Celular Inteligente	2	400	800
Servidor en la Nube	1	50	50
3.Servicios			
Servicio de Transporte	80	1.40	112
Servicio de Internet	1	30	30
Taxis	0	0	0
Alimentación	140	2,25	315
4.Personal			
Estudiante Investigador	0	0	0
Asesoría especializada	1	0	0
5.Otros			
Imprevistos	1	130	130
Total	1016	$1.525,87	$3.367,32

Tabla 10. Presupuesto Utilizado para el Desarrollo del Proyecto..

Capítulo IX

CONCLUSIONES

- Se utilizó un medidor de agua ultrasónico y un dispositivo LoRa, para la lectura de datos en el servidor de ChirpStack, esto con la finalidad de administrar estos datos como el consumo, frecuencia, relación de señal a ruido, intensidad de señal, etc. Para la visualización de estos datos mediante gráficas en un dashboard se hizo uso de Node-Red, el cual mediante consultas a la base de datos donde se guarda toda la información de los dispositivos, se pudo graficar estos datos y poder visualizarlas de mejor manera para realizar un análisis de estos.

- Para que el Gateway y el servidor ChirpStack estén conectados y pasando datos entre sí, primeramente se tuvo que crear un Network Server en ChirpStack, para que ese componente se pueda ingresar en la lista de servidores en la configuración del Gateway, una vez ingresado al Gateway, se sincronizó y se realizó la conexión entre estos dos y así poder recibir datos de dispositivos registrados en el servidor.

- Todos estos datos recibidos en el Gateway se guarda directamente en una base de datos, esta base de datos tendrá toda la información correspondiente al gateway como a los dispositivos registrados, se insertara datos en la tabla cada vez que el medidor envíe un dato hacia el servidor. Para saber el consumo de estos medidores

se procedió a realizar una fórmula de cálculo, en donde en una columna de tipo JSON, se encuentra el dato del consumo en donde se tendrá que mostrar el primer y último dato del día enviado hacia el servidor, sabiendo esto se procedió a separar por partes cada dato, ya que este dato tiene en sí los dm^3 y cm^3 del consumo, en donde el valor antes del punto son los dm^3 y después del punto son los cm^3 , y mediante estos datos se realizó la fórmula de cálculo para mostrar el consumo en m^3 , cuando se haya convertido estos dos datos del día en m^3 , se procedió a restar estos dos valores, el último dato y el primer dato del día, en donde el resultado será el consumo de agua durante un día.

- Este proyecto sirvió mucho para enriquecer nuestros conocimientos, conociendo nuevas plataformas que son de mucha ayuda para tecnologías basadas en IoT, ya que estas se usan mucho en la actualidad y con enorme proyección a futuro. Los medidores de agua inteligentes y con una tecnología que se basa en Internet, ayudara mucho a la administración de consumo de agua en los hogares, simplemente al ingresar a una página donde se mostrara gráficas del consumo del medidor se podrá tener un control sobre este, lo cual resulta muy factible.

- Una vez implementado esta tecnología, tendrá muchos beneficios tanto como a la empresa Etapa EP como para la Universidad, ya que al ser una tecnología que funciona por internet no es necesario personal que se encargue de tomar lectura de los datos en los medidores que se encuentran en los hogares, como también seguir incentivando con nuevos proyectos y tecnologías que están creciendo en la actualidad, así como para la universidad, ya que se siguen desarrollando nuevos proyectos el cual enorgullece a la institución y que posteriormente se sigan desarrollando nuevos y mejores proyectos con distintas empresas grandes como Etapa EP.

Capítulo X

RECOMENDACIONES

- Es importante que el dispositivo Gateway esté en un lugar estratégico con vista de línea directa para tener una comunicación estable, y que no se pierda la comunicación entre el dispositivo Gateway con el medidor de agua ultrasónico.
- Para crear un Gateway-Profile de ChirpStack, se debe tener en cuenta los canales en que trabaja los dispositivos, el ancho de banda, frecuencia y los factores de dispersión, para que el dispositivo Gateway pueda realizar el envío de datos caso contrario no lo realiza.
- Se debe tener en cuenta que la comunicación entre dispositivos con el servidor se usan muchos puertos de conexión por lo que es necesario una conexión a la red sin muchas restricciones, si es el caso se obtendrá un error de ***WebSocket not connecting***
- Los medidores de agua ultrasónico no pueden estar registrados en varios servidores, porque no permitiría enlazarse con el servidor de ChirpStack donde se esté haciendo las configuraciones.

REFERENCIAS

[1] N. Hossein Motlagh, M. Mohammadrezaei, J. Hunt, and B. Zakeri, "Internet of things (iot) and the energy sector," *Energies*, vol. 13, no. 2, p. 494, 2020.

[2] C. Sobin, "A survey on architecture, protocols and challenges in iot," *Wireless Personal Communications*, vol. 112, no. 3, pp. 1383–1429, 2020.

[3] A. Sanchis-Cano, J. Romero, E. J. Sacoto-Cabrera, and L. Guijarro, "Economic feasibility of wireless sensor network-based service provision in a duopoly setting with a monopolist operator," *Sensors*, vol. 17, no. 12, p. 2727, 2017.

[4] K. K. Patel, S. M. Patel, *et al.*, "Internet of things-iot: definition, characteristics, architecture, enabling technologies, application & future challenges," *International journal of engineering science and computing*, vol. 6, no. 5, 2016.

[5] E. E. Carrasco Galdame, "Metodología para selección de tecnologías lpwan para diversas aplicaciones de internet de las cosas," 2020.

[6] R. Ratasuk, B. Vejlgaard, N. Mangalvedhe, and A. Ghosh, "Nb-iot system for m2m communication," in *2016 IEEE wireless communications and networking conference*, pp. 1–5, IEEE, 2016.

[7] I. Ordóñez Monfort, "Estudio de la arquitectura y el nivel de desarrollo de la red lorawan y de los dispositivos lora.," 2017.

[8] "Introduction - chirpstack open-source lorawan® network server." https://www.chirpstack.io. Accedido: 2022-4-20.

[9] "The chirpstack project - chirpstack open-source lorawan® network server." https://www.chirpstack.io/project/. Accedido: 2022-4-20.

[10] M. A. Moral-Pérez, "Node-red como herramienta visual de dispositivos iot," 2021.

[11] "Node-RED." https://nodered.org/. Accessed: 2022-4-20.

[12] E. Sisinni, P. Ferrari, D. F. Carvalho, S. Rinaldi, P. Marco, A. Flammini, and A. Depari, "Lorawan range extender for industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5607–5616, 2019.

[13] D. P. Rodriguez Alvarado and E. J. Sacoto-Cabrera, "Implementation and analysis of the results of the application of the methodology for hybrid multi-cloud replication systems," in *The International Conference on Advances in Emerging Trends and Technologies*, pp. 273–286, Springer, 2021.

[14] "Empresa publica municipal de telecomunicaciones.Agua Potable, Alcantarillado y saneamiento de Cuenca." http://ftp.eeq.com.ec/upload/empresas-publicas-eficientes/Cultura%20Empresarial/Oswaldo%20Larriva%20-%20ETAPA%20copy.pdf. Accedido: 2022-09-10.

[15] J. Aranda, E. J. Sacoto Cabrera, D. Haro Mendoza, and F. Astudillo Salinas, "5g networks: A review from the perspectives of architecture, business models, cybersecurity, and research developments," *Novasinergia*, vol. 4, 2021.

[16] K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim, "Internet of things (iot) for next-generation smart systems: A review of current challenges, future trends

and prospects for emerging 5g-iot scenarios," *Ieee Access*, vol. 8, pp. 23022–23040, 2020.

[17] E. Sacoto-Cabrera, J. Rodriguez-Bustamante, P. Gallegos-Segovia, G. Arevalo-Quishpi, and G. León-Paredes, "Internet of things: Informatic system for metering with communications mqtt over gprs for smart meters," in *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, pp. 1–6, IEEE, 2017.

[18] E. J. Sacoto-Cabrera, L. Guijarro, J. R. Vidal, and V. Pla, "Economic feasibility of virtual operators in 5g via network slicing," *Future Generation Computer Systems*, vol. 109, pp. 172–187, 2020.

[19] E. J. Sacoto Cabrera, S. Palaguachi, G. A. León-Paredes, P. L. Gallegos-Segovia, and O. G. Bravo-Quezada, "Industrial communication based on mqtt and modbus communication applied in a meteorological network," in *The International Conference on Advances in Emerging Trends and Technologies*, pp. 29–41, Springer, 2020.

[20] V. Vimos and E. J. S. Cabrera, "Results of the implementation of a sensor network based on arduino devices and multiplatform applications using the standard opc ua," *IEEE Latin America Transactions*, vol. 16, no. 9, pp. 2496–2502, 2018.

[21] B. E. Zavala Soledispa, J. M. Romero Arguello, *et al.*, "Diseño de una red lpwan basada en tecnología lora para las estaciones hidrometeorológicas," B.S. thesis, Espol, 2018.

[22] I. Terán Lozano, "Internet de las cosas: una primera aproximación utilizando la tecnología sigfox," Master's thesis, Universitat Politècnica de Catalunya, 2015.

[23] R. S. Sinha, Y. Wei, and S.-H. Hwang, "A survey on lpwa technology: Lora and nb-iot," *Ict Express*, vol. 3, no. 1, pp. 14–21, 2017.

[24] E. J. Sacoto Cabrera, L. Guijarro, and P. Maillé, "Game theoretical analysis of a multi-mno mvno business model in 5g networks," *Electronics*, vol. 9, no. 6, p. 933, 2020.

[25] P. Chaudhari, A. K. Tiwari, S. Pattewar, and S. Shelke, "Smart infrastructure monitoring using lorawan technology," in *2021 International Conference on System, Computation, Automation and Networking (ICSCAN)*, pp. 1–6, IEEE, 2021.

[26] E. J. Sacoto-Cabrera, G. León-Paredes, and W. Verdugo-Romero, "Lorawan: application of nonlinear optimization to base stations location," in *Communication, Smart Technologies and Innovation for Society*, pp. 515–524, Springer, 2022.

[27] J. M. Marais, R. Malekian, and A. M. Abu-Mahfouz, "Lora and lorawan testbeds: A review," *2017 Ieee Africon*, pp. 1496–1501, 2017.

[28] J. P. M. Álvarez, V. I. R. Abdalá, F. M. R. M. Barboza, and F. R. C. Soria, "Estudio descriptivo de lorawan y aplicaciones específicas," *Difu100ci@, Revista de difusión científica, ingeniería y tecnologías, volume=15, number=1, pages=8–17, year=2021*.

[29] J. P. Villén Macías, "Diseño e implantación de red telemática de sensores inalámbricos para optimización de riego," 2020.

[30] E. F. Avila Cueva and M. Á. Parra Ordóñez, "Desarrollo de un prototipo de red lpwan con tecnología lora para la detección de intrusos en las viviendas de una zona residencial," B.S. thesis, Quito, 2021., 2021.

[31] M. Meli, E. Gatt, O. Casha, I. Grech, and J. Micallef, "A low cost lora-based iot big data capture and analysis system for indoor air quality monitoring," in *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 376–381, IEEE, 2020.

[32] "Chirpstack, open-source lorawan® network server stack." https://www.chirpstack.io. Accedido: 2022-4-20.

[33] "Introduction - ChirpStack open-source LoRaWAN® Network Server." https://www.chirpstack.io/gateway-bridge/. Accedido: 2022-4-20.

[34] "Introduction - ChirpStack open-source LoRaWAN® Network Server." https://www.chirpstack.io/network-server/. Accedido: 2022-4-20.

[35] "Introduction - ChirpStack open-source LoRaWAN® Network Server." https://www.chirpstack.io/gateway-os/. Accedido: 2022-4-20.

[36] M. Lekić and G. Gardašević, "Iot sensor integration to node-red platform," in *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pp. 1–5, IEEE, 2018.

[37] D. Clerissi, M. Leotta, G. Reggio, and F. Ricca, "Towards an approach for developing and testing node-red iot systems," in *Proceedings of the 1st ACM SIGSOFT International Workshop on Ensemble-Based Software Engineering*, pp. 1–8, 2018.

[38] P. Sancho, "Fundamentos de node-red."

[39] "Agua Potable." https://www.etapa.net.ec/principal/agua-potable. Accedido: 2022-09-10.

[40] J. Mocnej, A. Pekar, W. K. Seah, and I. Zolotova, *Network traffic characteristics of the IoT application use cases*. School of Engineering and Computer Science, Victoria University of . . . , 2018.

[41] F. Delgado-Ferro, J. Navarro Ortiz, N. Chinchilla Romero, J. J. Ramos Muñoz, *et al.*, "Arquitectura lorawan para entornos sin cobertura," 2021.

[42] E. M. J. Coloch Tahuico, *Diseño e implementación de tarjeta electrónica para pluviómetros de balancín para comunicación en red LoRaWAN y visualización en tiempo real*. PhD thesis, Universidad de San Carlos de Guatemala, 2021.