UTN FACULTAD REGIONAL MENDOZA
UNIVERSIDAD TECNOLÓGICA NACIONAL

CÁTEDRA DE PROYECTO FINAL / INGENIERÍA EN ELECTRÓNICA

MICRO SMART GRID (MSM)
PROYECTO

Versión *1.2*

INFORMACIÓN DEL PROYECTO

Autores	
Nombre Completo del integrante 1	López Di Paola, Gonzalo Andrés
Legajo	43401
e-mail	gonzalopez2206@gmail.com
Nombre Completo del integrante 2	Suarez, Tomas Octavio
Legajo	41421
e-mail	tomas.suarez.utn@gmail.com

Tutor	Ing. Gustavo Mercado
Director	Ing. Ana Lattuca
Año Académico	24 de Febrero de 2023
Responsable de la cátedra	Ing. Lattuca, Ana

Empresa / Organización / Laboratorio	GridTics
Patrocinador (Sponsor)	GridTics

ÍNDICE

1 INTRODUCCIÓN

1.1 EXECUTIVE RESUME

To develop a Micro Smart Grid system for public buildings with high electricity consumption that are part of a smart city. This modular system characterizes the building's electrical network and provides the data to a web platform for further visualization and analysis

1.2 RESUMEN EJECUTIVO

Desarrollar un sistema *micro smart grid* para edificios de gran consumo eléctrico que formen parte de una *smart city*. El sistema caracteriza modularmente la red eléctrica del edificio y provee los datos a una plataforma web para su posterior visualización y análisis.

Una Micro Smart Grid es una red eléctrica inteligente a pequeña escala que puede generar, distribuir y consumir energía de manera eficiente y sostenible. Estas redes están diseñadas para ser más flexibles y autónomas que las redes eléctricas tradicionales, y pueden integrar diferentes fuentes de energía renovable, como paneles solares y turbinas eólicas. Además, las Micro Smart Grids pueden monitorear y controlar el consumo de energía en tiempo real, lo que permite una gestión más eficiente de la energía y una reducción de los costos de energía y las emisiones de gases de efecto invernadero.

2 DESCRIPCIÓN DEL PROYECTO

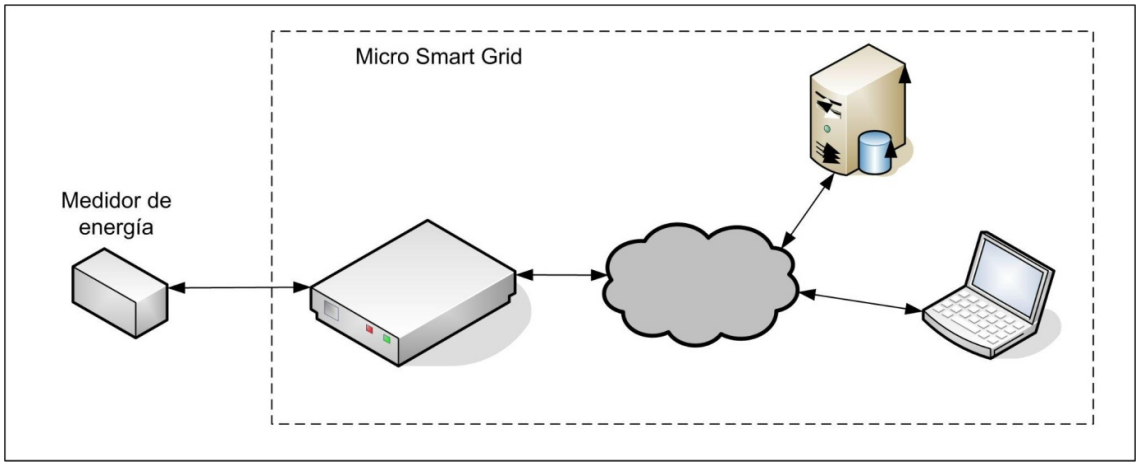
El proyecto consiste en desarrollar un prototipo de un sistema de gran eficiencia energética que a través de un sistema embebido, genere mediciones de cada red eléctrica de baja tensión que conforman al edificio público, obteniendo datos que lo caractericen.

Un conjunto de microcontroladores se encarga de recopilar estos datos y comunicarnos a través de una red interna de 2.4 GHz hacia un dispositivo receptor conectado a internet.

En el microcontrolador receptor, los datos son procesados y enviados a un servidor remoto.

Finalmente, estos datos son procesados y presentados de una forma eficiente y ordenada en una interfaz gráfica interactiva para que un analista de datos pueda posteriormente tomar decisiones que favorezcan la eficiente gestión energética del edificio público y consecuentemente de la *smart city*.

Se elige como edificio de estudio para el prototipo, el campus universitario UTN-FRM.



2.1 OBJETIVO GENERAL

Se busca a largo plazo proveer una herramienta asequible que permita la escalabilidad de las ciudades inteligentes.

2.2 OBJETIVO PARTICULAR

Con este prototipo se busca validar dos conceptos:

- La información entregada por el sistema tiene potencial de acoplarse con un sistema de toma de decisiones que permite ahorrar energía en el edificio.
- El servicio tiene como costo una fracción de la energía eléctrica ahorrada.

3 JUSTIFICACIÓN DEL PROYECTO (CASO DE NEGOCIO)

3.1 ANTECEDENTES DEL PROYECTO

Este proyecto surge a raíz del proyecto PID "*Smart Micro Grid de Campus Universitario - Desarrollo, implementación y prueba.*", originado en el laboratorio GRIDTICS, de la UTN-FRM.

3.2 ESTADO ACTUAL

Las tecnologías para la creación de ciudades inteligentes son cada vez más asequibles. Aún así, no es común verla aplicada en edificios públicos, pues esto supone costos de diseño, instalación y a veces, remodelación de la red eléctrica de edificios.

3.3 NECESIDAD DEL NEGOCIO Y DEFINICIÓN DEL PROBLEMA

Se calcula que para el año 2050 el 85% de la población mundial residirá en ciudades. Actualmente las ciudades son poco eficientes desde un punto de vista energético, pues gran parte de la energía que consumen es desaprovechada y conlleva mucho costo ambiental producirla. Si proyectamos este problema para los próximos 30 años, la calidad de vida humana en las ciudades se verá muy afectada negativamente.

En este contexto surge el la necesidad de una *smart city.* Este concepto hace referencia a ciudades inteligentes, con tecnologías específicas aplicadas para lograr objetivos orientados al bienestar social y ambiental, como el abastecimiento energético renovable, minimizar las emisiones de CO2, maximizar el desarrollo sostenible y por ende, mejorar la calidad de vida.

Para lograr un concepto macro de *smart city,* se tienen que desarrollar primero las tecnologías intrínsecas que lo conformen. Esto evidencia la necesidad de un sistema integral que combine tecnologías de información y comunicación con tecnologías de almacenamiento y ahorro energético. Este proyecto desarrolla las tecnologías necesarias dentro de un edificio público de gran consumo eléctrico para alinearlo con los objetivos de una *smart city*.

3.4 VISIÓN

Se espera lograr en un largo plazo un futuro sostenible donde los sistemas de medición y ahorro de energía sean asequibles. Se entiende que Micro Smart Grid es una primera aproximación con mucho potencial de ser mejorada y orientada al futuro proyectado.

Entender cómo se gasta la energía es un primer paso hacia organizaciones energéticamente sostenibles. Estamos convencidos que comenzar por los organismos públicos es el camino a seguir.

Implementar medidas orientadas a optimizar el desempeño energético de sus instalaciones cumple una función ejemplificadora ante el resto de la sociedad.

3.5 BENEFICIOS DEL PROYECTO

Los patrocinadores, en nuestro caso los laboratorios GRIDTICS e IRESE resultan mayormente beneficiados, ya que obtienen la oportunidad de validar la tecnología pensada para el proyecto.

La organización de Micro Smart Grid también resulta beneficiada en términos académicos.

4 ALCANCE

4.1 ALCANCE

El proyecto busca validar los conceptos en los cuales pueda basarse Micro Smart Grid a futuro, por eso se apuntó a un prototipo no comercial.

Se obtendrán datos de dos circuitos eléctricos del campus universitario y se mostrará esa información en una plataforma web interactiva.

Estudio	200 hs
Diseño preliminar electrónico.	100hs
Diseño e implementación de los sistemas de comunicación y adquisición de datos.	250 hs
Diseño de base de datos.	100 hs
Desarrollo de aplicación.	250 hs
Testeo y realización de cambios mínimos.	50 hs

4.2 LÍMITES O FUERA DE ALCANCE

No se realizará un producto comercial. El prototipo es únicamente con fines de investigación. No se tomarán datos de un edificio completo. No se proveerá un análisis de datos.

El prototipo se limita a:

- Implementar dos dispositivos de medición en redes de baja tensión.
- Implementar dos dispositivos embebidos de envío de datos dentro del edificio.
- Implementar un servidor web remoto para análisis de datos.
- Desarrollar una plataforma web interactiva para visualizar los datos.

4.3 REQUISITOS DE ALTO NIVEL

La siguiente tabla presenta los requisitos que el resultado del proyecto (producto o servicio) debe cumplir como condición necesaria.

Req. #	Descripción del requisito
1	Otorgar datos en tiempo real de al menos un dispositivo o red electrica
2	Almacenar datos de consumo históricos de cada edificio.
3	Implementar una interfaz amigable para el usuario donde se visualicen los datos.

4.4 SOLUCIONES Y ENTREGABLES PRINCIPALES

La siguiente tabla muestra un listado de los entregables del proyecto (productos o servicios)

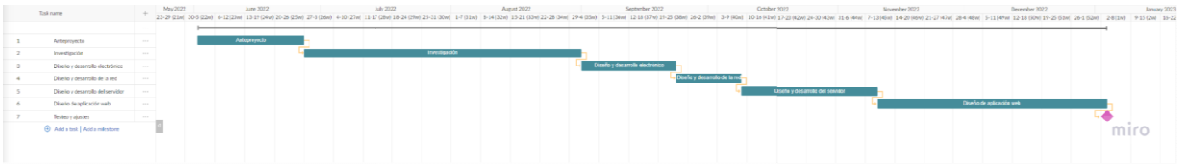
Entregables principales	Descripción del entregable
Sistema embebido de medición eléctrica	Implementación de un dispositivo de medición en una red eléctrica física.
Medidor (Adquirido)	Placa que adquiere datos y hace cálculos
Comunicación entre dispositivo medidor y emisor de datos	Desarrollo e implementación del protocolo de comunicación entre el dispositivo de medición de la red eléctrica y sistema emisor de datos.
Comunicación entre receptor y servidor	Realizar e implementar un protocolo de comunicación entre el sistema receptor de datos y el servidor.
Servidor	Realización de un servidor remoto para el almacenamiento de datos.
Aplicación web para análisis de datos	Realizar interfaz gráfica para la presentación de los datos obtenidos.

5 PLANIFICACIÓN DEL PROYECTO

5.1 PROCESOS

El equipo de Micro Smart Grid se gestiona dividiendo el flujo de trabajo a través de la metodología Kanban. Se utilizará la plataforma Trello. El rol de Project Manager no quedará definido por una sola persona.

5.2 CRONOGRAMA (LINK EN IMAGEN)



5.3 HITOS

La tabla muestra un listado de hitos generales del proyecto y el cronograma estimado de finalización

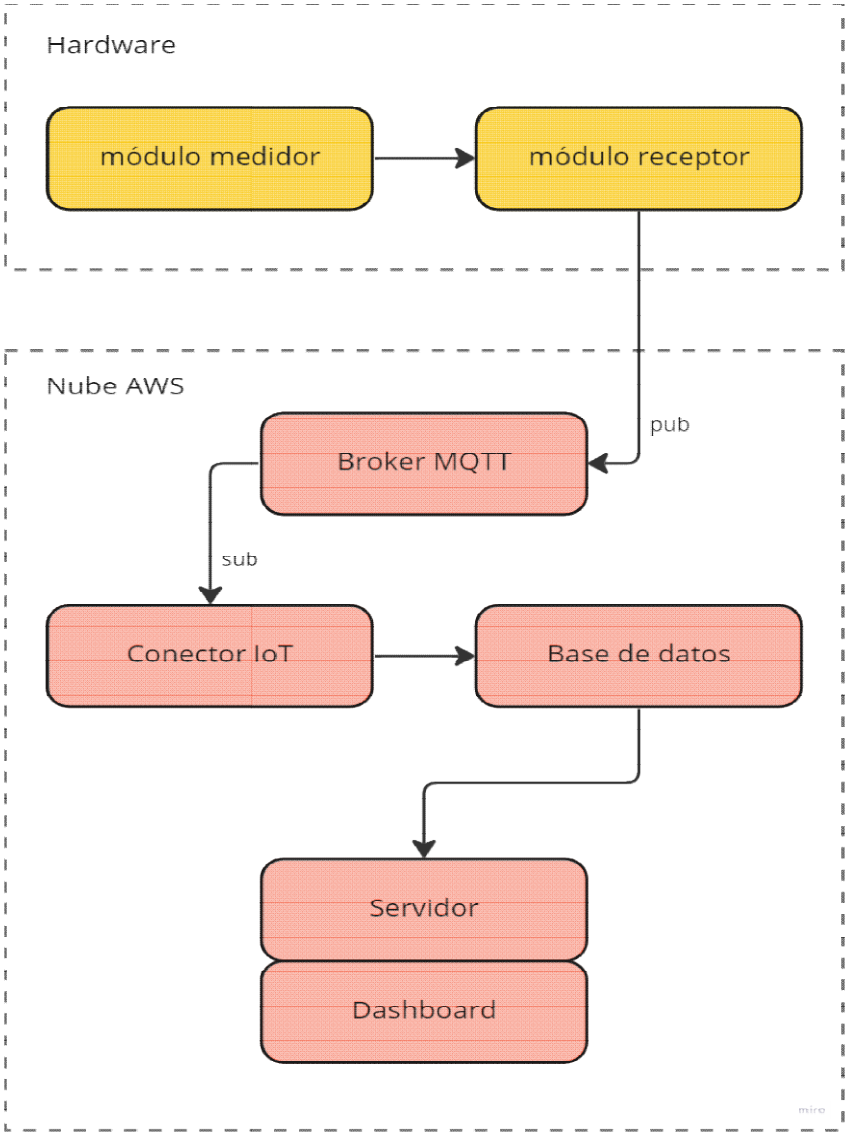
Hitos	Fecha de finalización
Documentación del anteproyecto	**26 de Junio de 2022**
Investigación.	**30 de Agosto de 2022**
Diseño y desarrollo electrónico.	**20 de Septiembre de 2022**
Diseño y desarrollo de la red.	**5 de Octubre de 2022**
Diseño y desarrollo de servidor	**5 de Noviembre de 2022**
Diseño de aplicación web	**29 de Diciembre de 2022**
Testeo y ajustes	**8 de Febrero de 2022**

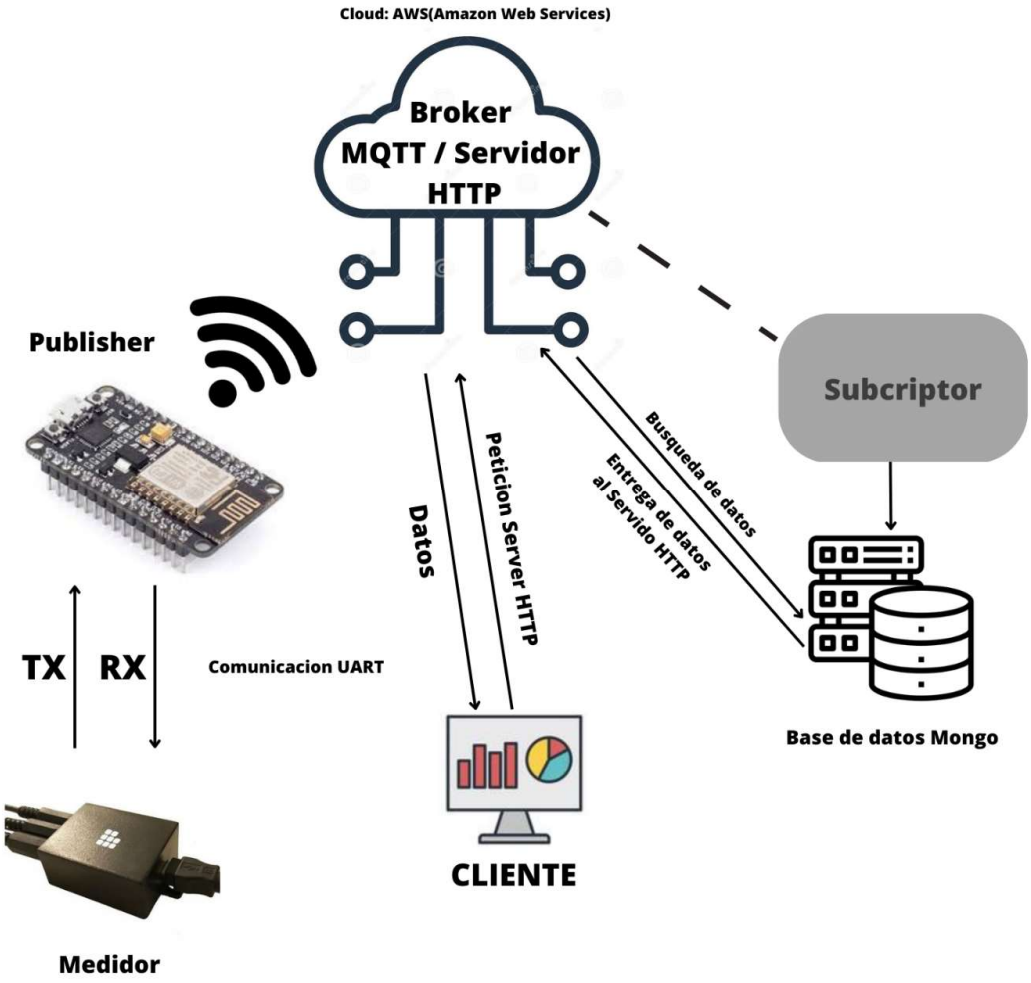
6 DESARROLLO DEL PROYECTO

6.1 DESARROLLO TÉCNICO

INTRODUCCIÓN - RESÚMEN

El fin del proyecto es mostrar de manera clara en un panel tipo dashboard un resúmen de los datos obtenidos por un medidor de energía eléctrica en una fecha determinada.
A continuación se muestra un esquema con los módulos que conforman al proyecto y debajo se describe cada uno indicando su función, qué tecnología utiliza, entrada y salida de datos, limitaciones, desafíos y criterios.





MÓDULO HARDWARE

MÓDULO MEDICIÓN DE ENERGÍA: MEDIDOR DE ENERGÍA ELÉCTRICA

El proyecto implicó la adquisición de un medidor de energía eléctrica. Se optó por uno de la marca SolverBox. La razón por la cual se optó por este medidor fue debido a que posee la capacidad de medir Potencia activa, Corriente RMS, THD, Nivel de armónicos hasta el 7mo y energía acumulada estos datos fueron considerados como suficientes para una correcta medición de la calidad de energia. Este dispositivo obtiene datos de la corriente eléctrica a través de pinzas amperométricas no invasivas, que tienen una capacidad de medición de hasta 100 Amperes. Para adquirir los datos, las pinzas se colocan en forma periférica alrededor del cable a medir.

Una vez procesados los datos de la señal de corriente, el medidor proporciona información valiosa, incluyendo: Identificador de fase, Voltaje RMS, Corriente RMS, Corriente pico, Corriente máxima (histórico), 7 componentes armónicas de la señal, Distorsión armónica total, Potencia activa y energía acumulada (desde el momento de encendido del medidor).
Esta información se muestra a través del puerto de comunicaciones del mismo en formato de una trama tipo String.

Criterios, limitaciones y desafíos: Se eligió puntualmente este medidor por la facilidad que tiene de transmitir datos. Puede transmitirlos por interfaz USB, o directamente por un un pin TX a 115200 baudios. Cabe destacar que este módulo está fuera del alcance del proyecto y es intercambiable por cualquier medidor de energía que brinde esos datos, debiendo adaptar la trama de datos que se envíe.

Input: señal de corriente eléctrica
Output: String con los datos procesados

Medidor SolversBox

MÓDULO DE ADQUISICIÓN Y TRANSMISIÓN DE DATOS: NODEMCU 1.0 - ESP8266

NodeMCU 1.0 es una placa de desarrollo basada en el chip ESP8266, que es un microcontrolador de bajo costo y bajo consumo de energía con capacidades Wi-Fi integradas. Se utiliza para desarrollar proyectos de Internet de las cosas (IoT, por sus siglas en inglés).

NodeMCU 1.0 incluye el chip ESP8266 y una serie de componentes adicionales, como un microcontrolador, memoria flash, EEPROM, puertos USB y GPIO, entre otros. Estos componentes permiten a los usuarios programar y controlar el dispositivo a través de una conexión Wi-Fi y puerto serie, lo que lo hace ideal para aplicaciones de IoT como sensores, dispositivos de control, sistemas de automatización del hogar, entre otros. Además, NodeMCU 1.0 se puede programar utilizando lenguajes de programación como Lua o C + + y se integra con una amplia variedad de herramientas y bibliotecas, lo que permite crear aplicaciones de IoT complejas y personalizadas.

Este módulo se programó en C++ utilizando PlatformIO integrado en el editor de texto Visual Studio code. PlatformIO es un entorno de desarrollo integrado de código abierto que se utiliza para la programación de microcontroladores. Se destaca por su capacidad de integración con diferentes plataformas, incluyendo Arduino.

Esto significa que se puede usar PlatformIO para programar y depurar proyectos de Arduino, ofreciendo una serie de ventajas en comparación con el IDE de Arduino tradicional. Algunas de estas ventajas incluyen una mejor gestión de librerías y dependencias, soporte para múltiples plataformas y una mayor eficiencia en el desarrollo y la depuración.

La función del módulo receptor es obtener la trama de datos del medidor de energía, procesarla y enviarla a través de

una red Wi-Fi hacia un broker MQTT. Después de procesar los datos del medidor, se identifican y organizan de manera que sea conveniente enviarlos al broker y se agrega la fecha en que se tomaron los datos.

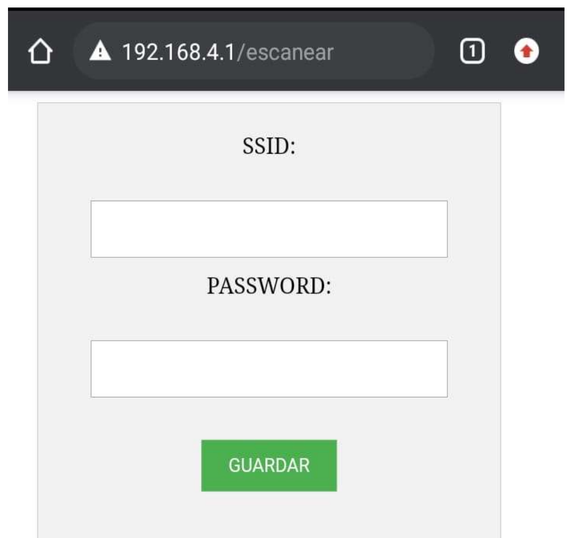
Node MCU ESP8266

FUNCIÓN: CORTE DE LUZ - INTERRUPCIÓN DE CONECTIVIDAD

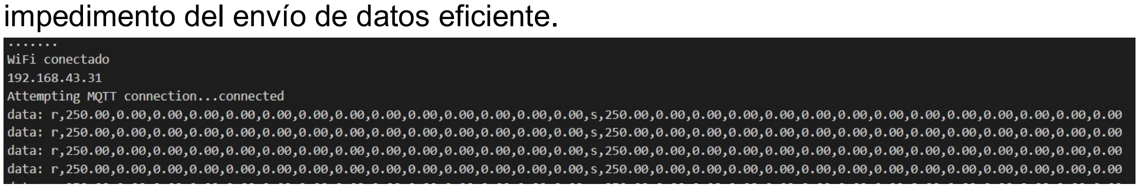
Los módulos *medidor* y *receptor* cuentan con un sistema de alimentación que los protege ante una interrupción de energía eléctrica. El mismo consta de un sistema de baterías que lo siguen alimentando cuando no hay servicio eléctrico.
En estos casos, no podría enviar los datos hacia el broker MQTT, ya que no habría conectividad.
Por esto, está programado para cuando suceden estos eventos, toma las mediciones, calcula fecha y hora, pero en lugar de ser enviadas al broker, las almacena en la memoria. Luego, cuando hay conectividad Wi-Fi y establece la conexión con el broker, envía todos los datos almacenados en la memoria. El sistema hacia aguas del broker MQTT puede identificar cuándo se produjeron interrupciones de servicio en el edificio o circuito a monitorear.

FUNCIÓN: CONFIGURACIÓN DE WI-FI

El NodeMCU posee una memoria EEPROM, donde guarda las credenciales de la red Wi-Fi a la cual debe conectarse.
Al inicio, intenta conectarse a esa red, pero si no la encuentra, el equipo entra en modo Access Point. De este modo, genera su propia red Wi-Fi para que el usuario pueda colocar las credenciales del edificio donde se va a instalar.Para garantizar la seguridad, esta red Wi-Fi cuenta con una clave privada que impide la configuración por parte de usuarios no autorizados y protege contra posibles acciones malintencionadas, como la desconexión de la red y el impedimento del envío de datos eficiente.



Interfaz de usuario para configurar red Wi FI



```
.......
WiFi conectado
192.168.43.31
Attempting MQTT connection...connected
data: r,250.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,s,250.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
data: r,250.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,s,250.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
data: r,250.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,s,250.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
data: r,250.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,s,250.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
```

Medidor conectado a red Wifi y al Broker, Trama enviada al broker MQTT.

<u>Criterios de selección, limitaciones y desafíos:</u>

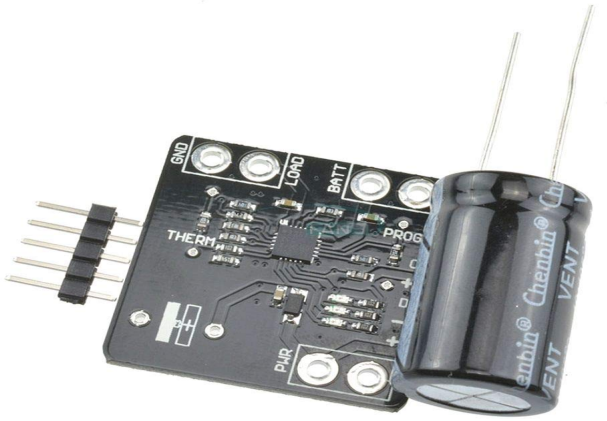
La placa de desarrollo NodeMCU 1.0 fue seleccionada para este proyecto debido a su facilidad de uso y bajo costo. Otros módulos basados en el chip ESP8266 también fueron considerados, incluyendo el ESP-01. Sin embargo, este último no fue elegido debido a la necesidad de un circuito externo para programación, en

USB. Además, la diferencia de precio entre ambos módulos es mínima. Cabe destacar que el ESP-01 presenta un tamaño significativamente reducido en comparación con el NodeMCU 1.0.

MÓDULO DE ALIMENTACIÓN:

Para la alimentación de respaldo del circuito se utilizó la placa cargadora **MCP 73871**. Este es un circuito integrado diseñado para cargar baterías de iones de litio y polímero de litio. Es un controlador de carga de batería de litio que se encarga de supervisar y controlar el proceso de carga de la batería, asegurando que se realice de manera segura y eficiente.

El MCP73871 recibe energía eléctrica externa y la utiliza para cargar la batería. Durante el proceso de carga, realiza un monitoreo constante de la tensión y la corriente de la batería para asegurarse de que no se sobrecargue ni se descargue demasiado. Si se detecta una situación anormal, como una tensión de carga excesivamente alta o una corriente de carga demasiado baja, el MCP73871 interrumpe el proceso de carga para proteger la batería.

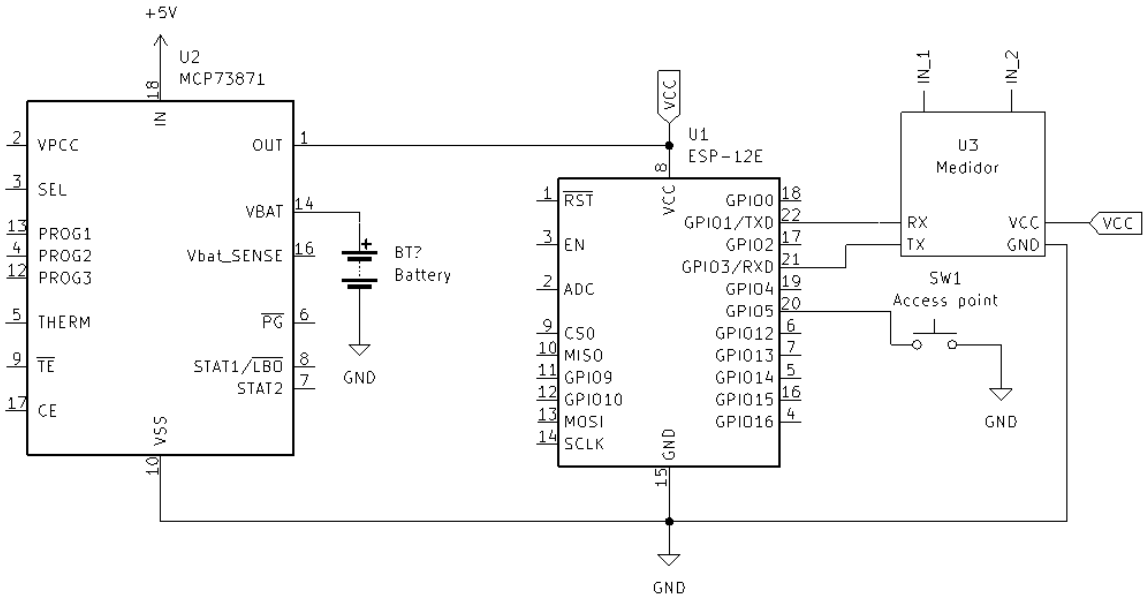


Placa cargadora MCP73871

Una de las ventajas del MCP 73871 es que es un controlador de carga de batería completamente integrado, lo que significa que se requiere muy poco hardware externo para utilizarlo. Además, es muy fácil de utilizar y es compatible con una amplia variedad de aplicaciones, desde sistemas de alimentación portátiles hasta dispositivos electrónicos alimentados por batería.

Otra de las ventajas que supone la placa cargadora es que puede intercambiar entre la alimentación directa y la alimentación de la batería, permitiendo que, en caso de que la batería se encontrase descargada, es circuito nunca quede sin alimentación en caso de que vuelva la tensión de alimentación principal

Prototipo realizado "EnergyEye"



Esquema circuital del prototipo desarrollado.

MÓDULO SOFTWARE

MÓDULO BROKER MQTT

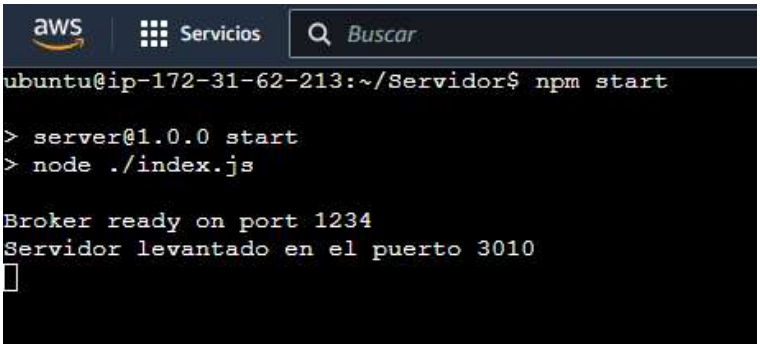
Un broker MQTT es un intermediario esencial en un sistema de comunicación basado en MQTT, un protocolo de mensajería que utiliza un esquema de publicación y suscripción para permitir la transmisión de datos en tiempo real entre dispositivos conectados a Internet.

En un sistema de comunicación basado en MQTT, el broker actúa como un intermediario confiable y eficiente entre los dispositivos publicadores y suscriptores. Los publicadores envían mensajes a un tópico específico, mientras que los suscriptores se registran para recibir mensajes de ese tópico o tópicos específicos.

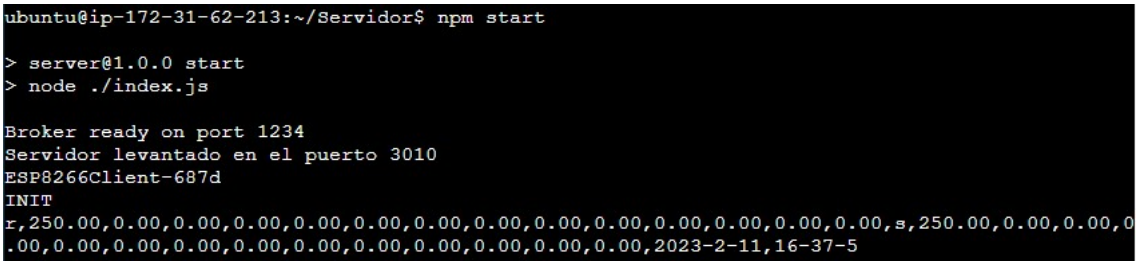
El papel del broker MQTT es fundamental en aplicaciones y sistemas IoT que requieren comunicación en tiempo real y confiabilidad. Por ejemplo, en un sistema

dispositivos, como sensores, actuadores y otros elementos, permitiendo la transmisión de información en tiempo real de manera eficiente y segura.

Para este proyecto se optó por hacer nuestro propio broker MQTT. El mismo fue programado en Express.js utilizando la librería “mosca”, que permite correr un broker en un puerto específico.
Express.js está escrito en JavaScript, y utiliza la programación asíncrona y “event-driven” de JavaScript para crear aplicaciones web rápidas y escalables.



Broker MQTT funcionando y esperando la conexión de un medidor



Medidor conectado al broker y enviando datos

Criterios, limitaciones y desafíos:

A la hora de elegir un protocolo de comunicación, se contempló la posibilidad de usar CoAP (Constrained Application Protocol). Es un protocolo de aplicación diseñado específicamente para aplicaciones IoT y dispositivos restringidos. Es muy similar a HTTP y, por lo tanto, es fácil de implementar en dispositivos con limitaciones de recursos. CoAP es muy eficiente en términos de ancho de banda y es adecuado para aplicaciones en las que la frecuencia de actualización es alta.

Se eligió MQTT debido a la escalabilidad y la facilidad de desarrollo, pues al ser más utilizado, tiene más documentación en internet. MQTT es un protocolo altamente escalable, debido a su arquitectura basada en publicador-suscriptor y su capacidad para manejar una gran cantidad de dispositivos conectados de manera eficiente. El protocolo MQTT permite a los clientes publicar mensajes en un tópico y a los servidores responder a esos mensajes, lo que lo hace adecuado para aplicaciones con grandes cantidades de datos y dispositivos.

Por otro lado, CoAP es un protocolo diseñado para ser utilizado en redes de baja potencia y limitadas en términos de ancho de banda y recursos, por lo que no es

escalable en comparación al protocolo MQTT. CoAP es adecuado para aplicaciones con pocos dispositivos y una cantidad limitada de datos, pero no es la mejor opción para aplicaciones que requieren una gran cantidad de dispositivos o una gran cantidad de datos.

En resumen, MQTT es una mejor opción si se requiere escalabilidad y una gran cantidad de dispositivos y datos, mientras que CoAP es adecuado para aplicaciones con requisitos más limitados en términos de escalabilidad.

También se optó por construir un broker MQTT propio sobre un servidor en vez de contratar un broker privado, ya que al poder desarrollarlo, nos brindó mayor libertad y control sobre el broker. Además de añadir seguridad extra en comparación a broker públicos como mosquito.

```
var broker = new mosca.Server(settings);

broker.on('ready', () => {
   console.log('Broker ready on port 1234');
});

broker.on('published', (packet) => {
   console.log(packet.payload.toString());
});
```

En este caso se puede observar un ejemplo de código JS del broker.

El principal obstáculo a superar es garantizar la disponibilidad activa del intermediario en un servidor que se encuentre en ejecución en un sistema informático. Inicialmente se experimentó con la utilización de equipos locales, sin embargo, esto resultó en una implementación restringida debido a la dependencia del proveedor de servicios de Internet (ISP) para abrir los puertos del servidor al público. Para solucionar este problema, se optó por utilizar una máquina virtual con el sistema operativo Ubuntu en la nube de Amazon Web Services (AWS), permitiendo así ejecutar todo el proyecto de manera adecuada.

MÓDULO CONECTOR IOT: MQTT TO MONGODB

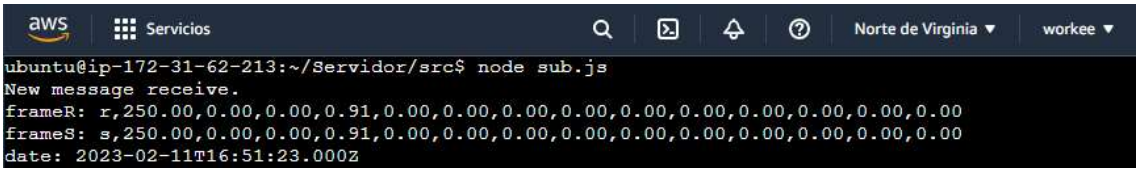
Este módulo es responsable de recibir los mensajes que llegan a un tópico específico en el broker MQTT. Este tópico está asociado con un piso específico de un edificio que es usado por un usuario que ha contratado el servicio. El módulo toma los datos que llegan en los mensajes, los formatea según un formato especificado, y finalmente los guarda en la base de datos MongoDB.
Está programado en node.js utilizando la librería “mqtt”, la cual tiene amplia documentación y ejemplos. Corre sobre el mismo servidor que el broker MQTT en la nube de AWS.

```
client.on('message', (topic, message) => {//cuando recibo un mensaje lo parsep a string
    let status = false;
    message = message.toString();
    status = match(message);

    if (status === true) {
        console.log("New message received.");
        parseFrame(message);
    }
    else {
        console.log("Frame Error");
        console.log(message);
    }
});
```

Ejemplo de suscriptor implementado en JS

Entrada de datos: mensaje string proveniente del broker MQTT con la trama de datos de una medición.
Salida de datos: objeto con una estructura de datos propia que se publica en la base de datos mongoDB



Datos procesados por el suscriptor, en este caso se puede apreciar las dos tramas enviadas por el medidor separadas con su correspondiente fecha.

MÓDULO BASE DE DATOS

Como se mencionó antes, se utilizó Mongodb en lugar de SQL como base de datos. MongoDB es una base de datos NoSQL (sin esquema) de tipo documental. Esto significa que en lugar de almacenar datos en tablas con filas y columnas como lo hacen las bases de datos relacionales, MongoDB almacena datos en documentos en un formato similar a JSON, donde cada documento es un registro independiente con su propio conjunto de campos y valores.
Hay varias razones por las que MongoDB es considerado mejor que SQL en algunos casos:

1. Flexibilidad en la estructura de datos: en MongoDB, los documentos pueden tener campos diferentes, lo que permite una mayor flexibilidad en la modelación de los datos.
2. Escalabilidad horizontal: MongoDB es altamente escalable, lo que significa que se puede distribuir fácilmente en múltiples servidores para manejar grandes volúmenes de datos y alta demanda de procesamiento.
3. Desempeño: MongoDB fue diseñado desde el principio para manejar grandes volúmenes de datos y alta concurrencia, lo que lo hace muy adecuado para aplicaciones que requieren un alto rendimiento.

4. Facilidad de uso: MongoDB utiliza un lenguaje de consulta sencillo y fácil de usar, lo que lo hace más accesible para los desarrolladores que no tienen experiencia en bases de datos relacionales.

En una base de datos de aplicaciones IoT en MongoDB, se pueden tener diferentes colecciones para diferentes tipos de objetos, como sensores, actuadores, eventos, usuarios, etc. Cada objeto se representa mediante un documento JSON, que puede contener diferentes campos con diferentes tipos de datos. Por ejemplo, un objeto "sensor" puede contener campos como "ID", "ubicación", "tipo", "estado", etc.
Además, MongoDB permite una fácil escalabilidad horizontal y la adición de nuevos nodos al clúster sin necesidad de modificar la estructura de la base de datos. Esto es útil en aplicaciones de IoT donde se pueden tener diferentes fuentes de datos que cambian con el tiempo.
En resumen, la estructura de la base de datos en MongoDB para aplicaciones de IoT se basa en colecciones de documentos JSON que representan objetos en la aplicación. Cada objeto se representa mediante un documento que contiene diferentes campos con diferentes tipos de datos. La escalabilidad horizontal y la capacidad de adaptarse a diferentes tipos de datos hacen de MongoDB una opción popular para aplicaciones de IoT.

En el caso del proyecto se utilizó la siguiente estructura de datos para ser almacenada:

```
const meassureSchema = mongoose.Schema({
   meassure: String,
   date: Date
});
```

Esquema de base de datos en Mongodb

Esto permite guardar la trama recibida por el suscriptor y la fecha en la cual fue adquirida sin mayores complicaciones a la hora de tener un manejo de datos eficiente.
A continuación se muestra un ejemplo de los datos almacenados en la base de datos:

Imagen capturada de base de datos Mongodb

Para el manejo de datos en mongodb se utilizó la librería **mongoose**.

Mongoose es una biblioteca de modelado de objetos de MongoDB para Node.js. Es un enrutador que actúa como una capa intermediaria entre la aplicación y la base de datos MongoDB, permitiendo interactuar con la base de datos a través de objetos en lugar de tener que escribir manualmente código de consulta a la base de datos.
En resumen, MongoDB es una buena opción cuando se requiere una mayor flexibilidad en la estructura de datos, escalabilidad y un mejor desempeño en aplicaciones con grandes volúmenes de datos y alta demanda de procesamiento. Sin embargo, para aplicaciones más tradicionales que requieren una estructura de datos rígida y relaciones complejas entre datos, una base de datos relacional como SQL puede ser una mejor opción.

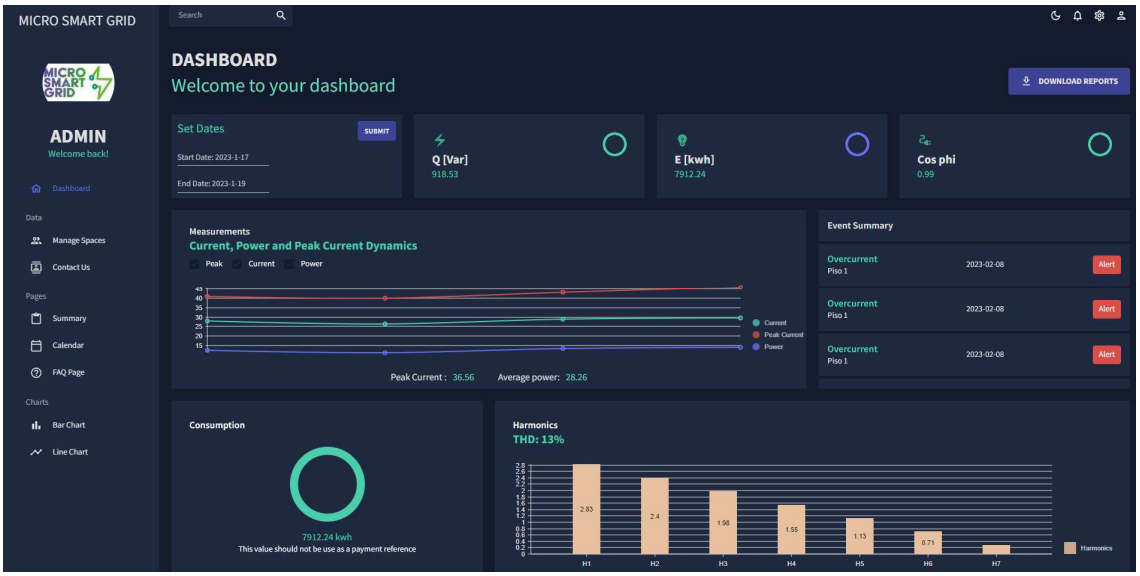
PANEL DASHBOARD

En el contexto de sistemas de monitorización de consumo eléctrico, el dashboard es una herramienta que permite visualizar los datos adquiridos por los medidores instalados en tiempo real y de forma organizada. El objetivo principal de esta herramienta es proporcionar una representación clara y concisa del consumo eléctrico de uno o varios edificios con el fin de permitir a los usuarios expertos tomar decisiones informadas basadas en la información proporcionada.
El dashboard recopila, procesa y muestra los datos adquiridos por los medidores, permitiendo a los usuarios expertos analizar y comparar los patrones de consumo eléctrico a lo largo del tiempo, identificar tendencias y patrones anómalos, y ajustar la gestión de la energía en consecuencia. Además, la interfaz del dashboard debe ser intuitiva y fácil de usar para permitir a los usuarios no expertos entender rápidamente los datos y tomar decisiones en consecuencia.
En resumen, el dashboard es una herramienta importante en los sistemas de monitorización de consumo eléctrico, permitiendo a los usuarios expertos tomar decisiones informadas en función de los datos recopilados y procesados. La interfaz del dashboard debe ser intuitiva y fácil de usar, lo que permite a los usuarios no expertos entender rápidamente la información y tomar decisiones en consecuencia.

FRONTEND

Del lado del Frontend tenemos la página web desarrollada la cual muestra datos procesados por el servidor relacionados con la energía eléctrica. Estos son mostrados en distintas gráficas de forma tal que pueda sacar una rápida conclusión sobre lo que está ocurriendo en el edificio bajo medición.
Los datos presentados son extraídos según las fechas que se indiquen en la página web son promediados y por último mostrados. Estos incluyen la energía acumulada, el factor de distorsión armónica total (THD), la potencia reactiva, el coseno phi, los armónicos del 1 al 7 con sus respectivos valores en un gráfico de

barras.

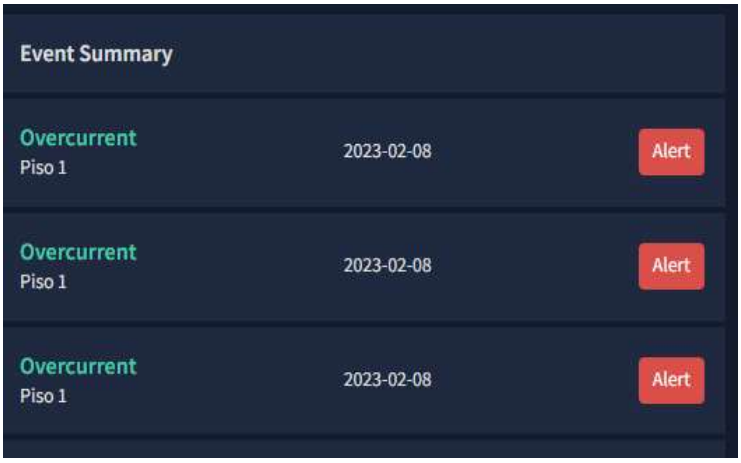


Dashboard completo con datos reales ilustrados.

Además, la página también presenta información sobre la potencia activa, la corriente rms y la corriente pico en tiempo real.

Corriente pico, RMS y potencia activa en el tiempo

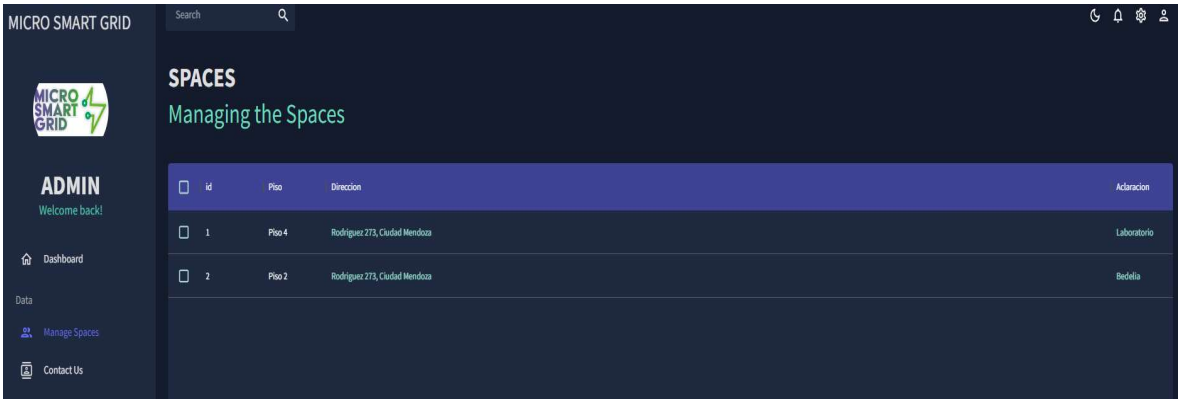
Para brindar una mayor funcionalidad, la página cuenta con un indicador de eventos, que alerta al usuario en caso de picos en el consumo de energía eléctrica, los cuales pueden ser configurados en la pestaña "summary".



Resumen de eventos que han superado un valor determinado por el usuario

Entre otras funcionalidades que posee se encuentra la pestaña de "*manage spaces*" la cual permite al usuario seleccionar de qué piso desea ver la información. Es una

herramienta útil para la administración de diversos pisos en los que se encuentre los medidores adquiriendo datos.



Pestaña "Manage Spaces" la cual permite controlar qué dispositivo o piso se están mostrando los datos.

Otra funcionalidad a destacar en la pestaña "summary" es que permite la comparación de los datos medidos entre diferentes pisos. En el caso de la imagen se puede observar la comparativa entre dos pisos en los cuales se realizó la medición de energía.

Además de presentar información visual sobre la energía eléctrica, la página web desarrollada en React también se comunica con un backend desarrollado en Node.js. La comunicación entre la página web y el backend se realiza mediante la utilización de la biblioteca Axios.

Axios es un cliente HTTP basado en promesas que permite hacer solicitudes HTTP desde el lado del cliente de una manera sencilla y eficiente. Esto permite que la página web obtenga los datos necesarios del backend de manera dinámica, actualizando en tiempo real la información presentada en la página. La combinación de React para el desarrollo de la interfaz y Node.js para el backend, junto con Axios para la comunicación, brinda una solución completa y eficiente para la visualización y monitoreo de datos relacionados con la energía eléctrica.

Tecnología utilizada: Para el desarrollo de la página web se utilizó el framework **React.** El cual es una biblioteca de JavaScript desarrollada por Facebook que se utiliza para construir interfaces de usuario en aplicaciones web.
En términos simples, React permite crear componentes de la interfaz de usuario que se actualizan y renderizan automáticamente cuando cambian los datos asociados. Esto permite crear aplicaciones web dinámicas y altamente interactivas sin tener que recargar la página completa cada vez que se produce un cambio. En resumen, React es una herramienta valiosa para los desarrolladores web que buscan crear aplicaciones web modernas y eficientes, y es por esto que se decidió utilizarla.

BACKEND

El Backend es la parte de una aplicación web que se encarga de las tareas y operaciones que ocurren en el servidor. Este es el encargado de procesar y almacenar datos, y de ofrecer servicios a la parte del cliente (frontend) de la aplicación.
Express es una popular librería de Node.js que se utiliza para crear aplicaciones web y servicios API en el lado del servidor. Es un framework minimalista y flexible que brinda una gran cantidad de características y herramientas para crear aplicaciones web con Node.js de manera rápida y fácil.

La comunicación entre el frontend y el backend se lleva a cabo a través de solicitudes HTTP (por ejemplo, una solicitud GET). Cuando llega una solicitud GET con querys, Express la procesa y puede acceder a los datos enviados en la consulta. Luego, puede utilizar estos datos para realizar alguna operación en el servidor (por ejemplo, recuperar información de una base de datos) y enviar una respuesta al frontend. La respuesta puede ser en formato JSON o cualquier otro tipo de formato de datos que se haya acordado entre el frontend y el backend.

Además de lo mencionado anteriormente, es común que en el backend se cree una API (Application Programming Interface) para comunicar el frontend con la base de datos. La API actúa como un intermediario entre el frontend y la base de datos, permitiendo que el frontend realice solicitudes a la base de datos a través de la API en lugar de hacerlo directamente.
Una API es un conjunto de reglas y protocolos que definen cómo dos sistemas informáticos se deben comunicar entre sí. En el caso de las aplicaciones web, una API define cómo el frontend puede acceder y utilizar los servicios y funcionalidades proporcionados por el backend. La API especifica qué solicitudes HTTP se pueden realizar (por ejemplo, GET o POST), qué parámetros se pueden enviar y qué formato de respuesta se recibirá. De esta manera, el frontend puede interactuar con el backend de manera consistente y predecible.

En el proyecto desarrollado utilizaron las herramientas mencionadas anteriormente. La API desarrollada permite comunicar el Frontend con la base de datos.
En la comunicación el Frontend realiza una petición GET el cuál envía a una ruta específica 4 parámetros importantes: Las fechas de inicio y fin, el mes y el piso del cual se quieren obtener los datos.

Con estos datos podemos realizar la consulta a la base de datos para obtener los datos relacionados con las fechas relacionadas. Es importante destacar que el formato de la fecha recibido es convertido al formato *2023-02-08T17:00:00.000+00:00* según la norma ISO 8601. Es importante este formato ya que permite realizar búsquedas eficientes en las bases de datos.

Una vez obtenidas todas las mediciones entre las fechas seleccionadas se extraen los datos y son promediados en el caso del coseno phi, potencia reactiva, tensión, corriente y pico de corriente.

Todos estos datos son agregados a un objeto JSON y es enviado al Frontend.

A continuación se muestra un ejemplo del JSON enviado, el cual consta de 4 mediciones obtenidas de la base de datos y los parámetros previamente procesados.


```
{
  meassure: [
    {
      _id: new ObjectId("63e6a3129770284a730f0745"),
      meassure: 'r,250.00,27.88,41.00,32.73,2.79,2.37,1.95,1.53,1.12,0.70,0.28,0.13,12.41,7906.49',
      date: 2023-02-08T17:00:00.000Z
    },
    {
      _id: new ObjectId("63e6a34f9770284a730f0746"),
      meassure: 'r,250.00,26.56,40.00,32.73,2.66,2.26,1.86,1.46,1.06,0.66,0.27,0.13,11.23,7896.48',
      date: 2023-02-08T17:15:00.000Z
    },
    {
      _id: new ObjectId("63e6a38b9770284a730f0747"),
      meassure: 'r,250.00,28.98,43.00,32.73,2.90,2.46,2.03,1.59,1.16,0.72,0.29,0.13,13.42,7917.33',
      date: 2023-02-08T17:30:00.000Z
    },
    {
      _id: new ObjectId("63e6a3c09770284a730f0748"),
      meassure: 'r,250.00,29.63,46.00,32.73,2.96,2.52,2.07,1.63,1.19,0.74,0.30,0.13,14.04,7928.66',
      date: 2023-02-08T17:45:00.000Z
    }
  ],
  max: 29.63,
  average: 28.2625,
  harmonics: [
    2.8274999999999997,
    2.4025,
    1.9775,
    1.5525,
    1.1324999999999998,
    0.7050000000000001,
    0.28500000000000003,
    0.13,
    7912.24,
    12.775,
    0.9915139938498094,
```

```
    918.53125
  ]
}
```

En conclusión, Node.js y Express son una combinación poderosa para desarrollar aplicaciones web y servicios API. Node.js es un entorno de ejecución de JavaScript en el servidor que brinda una alta performance y escalabilidad, mientras que Express es un framework minimalista y flexible que facilita la creación de aplicaciones y servicios en Node.js. Juntos, ofrecen una solución eficiente y fácil de usar para el desarrollo de aplicaciones web y servicios API, lo que lo hace una buena opción para una amplia variedad de proyectos. Además, con la creación de una API, se puede lograr una buena separación entre el frontend y el backend, lo que mejora la escalabilidad y el mantenimiento a largo plazo de la aplicación.

CONTROL DE VERSIONES

Para llevar a cabo el proyecto, se contó con el trabajo de varios desarrolladores. Esto presentó un desafío en términos de coordinación, ya que cada módulo del proyecto depende del correcto funcionamiento de otros módulos. Por ejemplo, para comprobar el funcionamiento del módulo receptor, es necesario contar con un servidor MQTT que reciba los mensajes, pero este servidor puede estar en constante desarrollo por otra persona.

Para resolver estos problemas de coordinación y simplificar el desarrollo, se optó por utilizar un sistema de control de versiones GIT en GitHub y ejecutar todos los servidores y programas en una computadora en la nube de AWS (Amazon Web Services). De esta manera, cada desarrollador puede trabajar en su propia copia local del proyecto sin interferir en el trabajo de los demás y sin detener el funcionamiento del proyecto.

GIT

GIT (Global Information Tracker) es un software de control de versiones de código fuente. Esto significa que puede llevar un registro de todos los cambios realizados en un proyecto de software y permite revertir a una versión anterior si es necesario. GIT también permite trabajar en equipo en un mismo proyecto, permitiendo a cada desarrollador tener una copia local de la última versión y enviar sus cambios a un servidor central para ser revisados e integrados en el proyecto.

GITHUB

GitHub es una plataforma web que aloja proyectos de software y permite a los desarrolladores colaborar en ellos. GitHub utiliza GIT para controlar las versiones del código y proporciona una interfaz en línea para que los desarrolladores puedan trabajar juntos, hacer seguimiento de los problemas y alojar documentación. Además, GitHub permite a los desarrolladores publicar sus proyectos y compartirlos con la comunidad de manera abierta o privada.

Hay varias plataformas que utilizan el sistema de versionado GIT, como GitLab, SourceForge, Bitbucket, etc. Se decidió utilizar GitHub debido a su gran popularidad y fácil integración con el editor de texto Visual Studio code, que fue el editor de texto elegido para el desarrollo de este proyecto.

Conclusión: fue de gran utilidad utilizar GitHub, ya que permitió por un lado un desarrollo ordenado del software y por otro lado permitió ver un progreso del proyecto en cada actualización del repositorio.

IMPLEMENTACIÓN EN LA NUBE

Para mejorar la solución a los problemas descritos en un capítulo anterior, se decidió ejecutar todos los programas en una computadora en la nube de AWS, excepto el módulo receptor. Esta decisión permitió que los módulos relevantes puedan funcionar de manera constante y que cualquier desarrollador pueda acceder a ellos en cualquier momento sin necesidad de coordinación con otros desarrolladores del proyecto.

Además, esta solución también permitió a los desarrolladores evadir las restricciones impuestas por los proveedores de internet locales en cuanto a las reglas de firewall. En su lugar, se tuvo la ventaja de poder editar fácilmente los grupos y reglas de seguridad en la nube de Amazon, sin requerir conocimientos técnicos específicos.

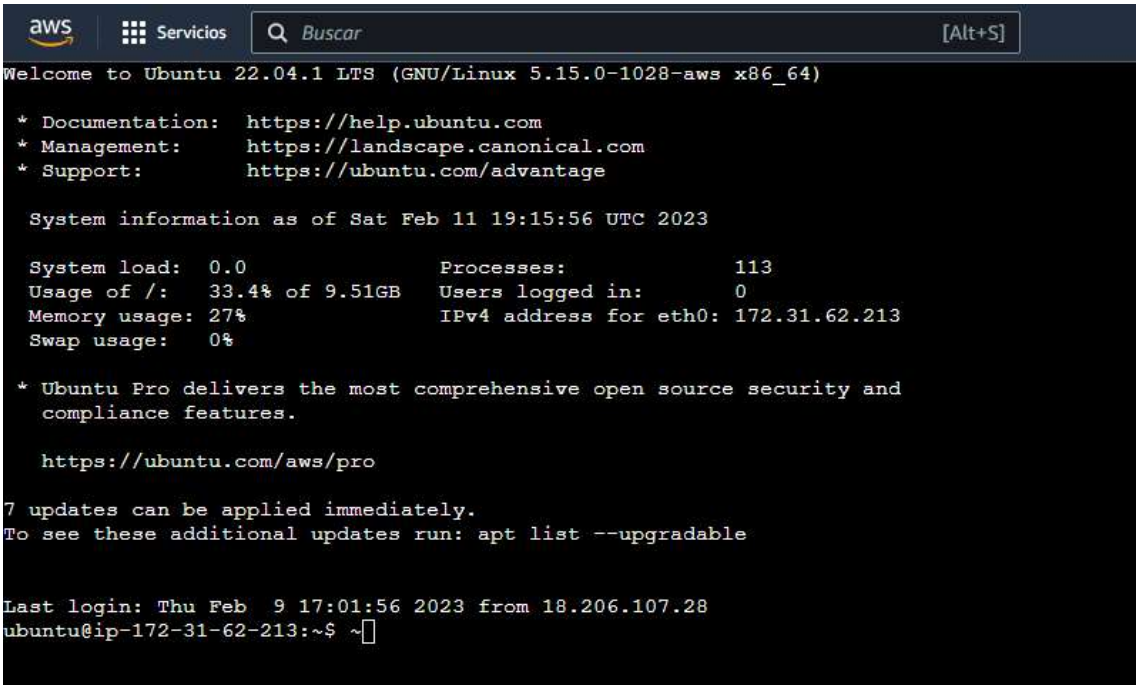
EC2

EC2 es uno de los servicios más populares de Amazon Web Services (AWS) y es una plataforma de computación en la nube que permite a los usuarios lanzar y escalar aplicaciones de manera rápida y sencilla. EC2 (Amazon Elastic Compute Cloud) brinda a los usuarios la capacidad de correr máquinas virtuales en la nube, donde pueden instalar y ejecutar aplicaciones, servicios y bases de datos.

Además, EC2 también brinda a los usuarios la capacidad de elegir diferentes tipos de instancias (que varían en términos de CPU, memoria, almacenamiento, entre otros) y regiones geográficas para satisfacer sus requisitos de desempeño y costo. Esto significa que los usuarios pueden elegir el tipo de instancia que se adapte a sus necesidades específicas y pueden escalar fácilmente si es necesario.

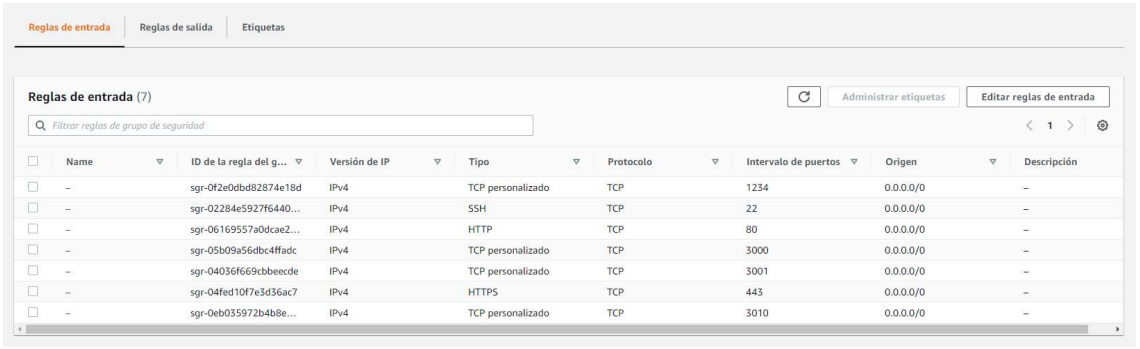
Para correr este proyecto se eligió una máquina virtual con Ubuntu Server 22.04 con 1 GB de ram, 10 GB de disco SSD y procesador de un núcleo virtual.

Se puede acceder a la VM a través de una consola online que brinda AWS o por conexión SSH, teniendo las claves correctas.



Consola de acceso a la VM

La plataforma de EC2 permite fácilmente editar las reglas de seguridad para permitir o denegar el tráfico de datos a través de sus puertos.



Reglas de entrada/salida para los puertos TCP utilizados

INTEROPERABILIDAD SEMÁNTICA

En la gestión de redes eléctricas inteligentes (smart grids), la interoperabilidad y la seguridad son aspectos fundamentales para el intercambio de información entre dispositivos y sistemas. Open Smart Grid Protocol (OSGP) es un estándar de comunicación que busca abordar estas necesidades al proporcionar una comunicación segura y confiable entre los diferentes dispositivos y sistemas de la red eléctrica.

OSGP utiliza un modelo de comunicación de dos vías para el intercambio de información en tiempo real, y se basa en tecnologías de comunicación existentes

como TCP/IP y GPRS. Además, es un estándar abierto y cuenta con una amplia comunidad de desarrolladores y usuarios que trabajan juntos para mejorar y promover el estándar.
Entre las características clave de OSGP se encuentran la interoperabilidad, la capacidad de soportar diferentes aplicaciones y servicios de red inteligente, como la gestión de la demanda y la integración de energías renovables, y la seguridad. Estas características permiten una gestión más eficiente y sostenible de la red eléctrica, lo que es fundamental para lograr un futuro energético más sostenible y limpio. En resumen, OSGP es un estándar de comunicación clave para el desarrollo de redes eléctricas inteligentes más eficientes y sostenibles.

Para estandarizar el envío de datos, deberían enviarse a través de MQTT en formato XML (eXtensible Markup Language), que es un lenguaje de marcado que se utiliza para describir y estructurar datos.
El objetivo de XML es proporcionar una forma de intercambiar datos entre aplicaciones y sistemas informáticos, independientemente del lenguaje de programación y el sistema operativo que se utilicen.
XML utiliza etiquetas para describir los datos, de manera similar a HTML. Sin embargo, en lugar de describir la presentación de los datos, XML se centra en describir su estructura y significado. Esto significa que los datos descritos en XML pueden ser fácilmente procesados y utilizados por aplicaciones informáticas, como una base de datos o un servicio web.

El Open Smart Grid Protocol (OSGP) utiliza XML como formato de intercambio de datos porque es un lenguaje estandarizado y flexible que permite la descripción y el intercambio de datos complejos y estructurados.
En el caso de un sistema de gestión de energía inteligente, los datos pueden ser muy complejos y variados, y pueden incluir información sobre medidores, aparatos, sistemas de distribución eléctrica, etc. XML permite describir esta información de manera clara y estructurada, lo que facilita su procesamiento y utilización por parte de aplicaciones y servicios.

Para el caso de este proyecto, los datos deberían salir del medidor en formato XML a través de MQTT de la siguiente manera:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<measurement>
  <phase>{phase}</phase>
  <RMSVoltage>{RMSVoltage}</RMSVoltage>
  <RMSCurrent>{RMSCurrent}</RMSCurrent>
  <PeakCurrent>{PeakCurrent}</PeakCurrent>
  <MaximumCurrent>{MaximumCurrent}</MaximumCurrent>
  <HarmonicCurrent1>{HarmonicCurrent1}</HarmonicCurrent1>
  <HarmonicCurrent2>{HarmonicCurrent2}</HarmonicCurrent2>
  <HarmonicCurrent3>{HarmonicCurrent3}</HarmonicCurrent3>
  <HarmonicCurrent4>{HarmonicCurrent4}</HarmonicCurrent4>
  <HarmonicCurrent5>{HarmonicCurrent5}</HarmonicCurrent5>
  <HarmonicCurrent6>{HarmonicCurrent6}</HarmonicCurrent6>
```

```
  <HarmonicCurrent7>{HarmonicCurrent7}</HarmonicCurrent7>
  <THD>{THD}</THD>
  <ActivePower>{ActivePower}</ActivePower>
  <AccumulatedEnergy>{AccumulatedEnergy}</AccumulatedEnergy>
  <timestamp>{timestamp}</timestamp>
</measurement>
```

Luego, para continuar utilizando la norma, los datos deberían guardarse en la base de datos mongoDB con la siguiente estructura:

```
{
  "_id": ObjectId("..."),
  "phase": 1,
  "RMSVoltage": 220,
  "RMSCurrent": 0.5,
  "PeakCurrent": 0.7,
  "MaximumCurrent": 0.8,
  "HarmonicCurrent1": 0.1,
  "HarmonicCurrent2": 0.2,
  "HarmonicCurrent3": 0.3,
  "HarmonicCurrent4": 0.4,
  "HarmonicCurrent5": 0.5,
  "HarmonicCurrent6": 0.6,
  "HarmonicCurrent7": 0.7,
  "THD": 3.0,
  "ActivePower": 500,
  "AccumulatedEnergy": 1000,
  "timestamp": ISODate("2023-02-11T12:00:00Z")
}
```

SEGURIDAD

El proyecto en cuestión utiliza el módulo ESP8266, el cual presenta medidas de seguridad para evitar la lectura del código fuente. El ESP8266 utiliza un sistema de protección de memoria para prevenir la copia del código fuente, lo que aumenta la seguridad en el uso de este dispositivo.

Además, en este proyecto se emplea la seguridad UART para la conexión entre el medidor y el ESP, lo que garantiza una conexión segura entre estos componentes. La comunicación UART utiliza un protocolo de comunicación seguro, el cual asegura que la información transmitida sea auténtica y no haya sido modificada en tránsito.

En cuanto a la conexión a una red WiFi, el módulo ESP8266 implementa diferentes mecanismos de seguridad para garantizar la confidencialidad e integridad de los datos transmitidos. Este módulo ofrece soporte para diferentes protocolos de cifrado, como WPA, WPA2 y WEP, los cuales garantizan la seguridad de la red WiFi.

El proyecto también incluye la implementación de medidas de seguridad en otros componentes críticos. En particular la transferencia de información desde el ESP al broker a través del protocolo MQTT. Para garantizar la seguridad en la transferencia de datos, el proyecto utiliza medidas de seguridad en el protocolo MQTT.
El protocolo MQTT proporciona varias medidas de seguridad para garantizar la confidencialidad e integridad de los datos durante la transferencia. Por ejemplo, se puede configurar el protocolo para utilizar cifrado TLS/SSL en la transferencia de datos, lo que garantiza que la información transmitida esté encriptada y protegida contra posibles ataques.
Además, el proyecto implementa medidas de seguridad en el broker MQTT, el cual está implementado en Node.js. Para garantizar la seguridad del broker, se pueden implementar diferentes medidas de seguridad, como la autenticación de usuarios, la limitación de acceso a ciertos temas (topics) y la configuración de permisos de acceso a los diferentes clientes.

Otro módulo importante en el proyecto es el que involucra la implementación de medidas de seguridad en la página web y el servidor. La seguridad en la página web y el servidor es crítica para garantizar la privacidad y la integridad de los datos, así como para prevenir posibles ataques de terceros.
En la página web, se pueden implementar diferentes medidas de seguridad para prevenir ataques de hackers, como el Cross-Site Scripting (XSS) o la inyección SQL. Estas medidas pueden incluir la validación de entradas de usuario, el uso de token de sesión para prevenir el CSRF (Cross-Site Request Forgery) y la utilización de SSL/TLS para cifrar la comunicación entre el cliente y el servidor.
En el servidor, se pueden implementar medidas de seguridad similares, como la validación de entradas de usuario, la autenticación de usuarios, la configuración de permisos de acceso, y el uso de SSL/TLS para cifrar la comunicación entre el servidor y los clientes.
Además, es importante mantener el servidor actualizado y utilizar software de seguridad actualizado para prevenir posibles vulnerabilidades conocidas. También se pueden implementar firewalls y monitoreo de seguridad para detectar posibles intrusiones y prevenir posibles ataques.

Es importante destacar que la implementación del servidor y la página web están corriendo en AWS (Amazon Web Services) que puede proporcionar beneficios adicionales en términos de seguridad. AWS proporciona una amplia gama de servicios de seguridad y cumplimiento, lo que significa que la infraestructura y los datos están protegidos por varias capas de seguridad física y virtual. Estos servicios incluyen la protección DDoS (ataques de denegación de servicio distribuido), la seguridad de la red, la autenticación y el control de acceso, el cifrado y la monitorización de seguridad.
Además, AWS permite una fácil implementación de la seguridad de los servidores, incluyendo la configuración de firewalls, la configuración de políticas de seguridad y la protección de datos en tránsito y en reposo. También se pueden implementar soluciones de seguridad personalizadas para cumplir con las necesidades específicas del proyecto.
En resumen, la implementación de la página web y el servidor en AWS proporciona beneficios adicionales en términos de seguridad gracias a los servicios de seguridad

y cumplimiento proporcionados por la plataforma. Esto incluye la protección DDoS, la seguridad de la red, la autenticación y el control de acceso, el cifrado y la monitorización de seguridad. Además, AWS permite una fácil implementación de medidas de seguridad personalizadas para cumplir con las necesidades específicas del proyecto.

CONCLUSIÓN

Se desarrolló una herramienta práctica y de simple uso para poder llevar a cabo la administración energética de un edificio. Esto permite una gran escalabilidad ya que es completamente plausible agregar más medidores en un mismo edificio y llevar un completo control del consumo de estos. Permitiendo realizar comparaciones y recibir alertas para poder actuar rápidamente ante diversas situaciones. Además la herramienta desarrollada permite una gran escalabilidad ya que sumamente sencillo ir agregando funcionalidades y en un próximo desarrollo permitir el control de actuadores en forma remota utilizando la interfaz de usuario.

PROYECCIONES A FUTURO

Debido a la gran extensión que abarca el proyecto hubieron varias funcionalidades que no pudieron ser desarrolladas pero que se agregaran en futuras versiones.
Entre ellas se encuentra:

- Página de Login para que distintos usuarios puedan acceder con una clave personal.
- Agregar seguridad en los servidores.
- Agregar la capacidad de tomar decisiones en base a los datos obtenidos y entregar un reporte completo.
- Generar un sistema de login automático para los medidores. De forma tal que al conectarse al broker se crea automáticamente el suscriptor correspondiente.
- Añadir medidores de forma dinámica en la página web.
- Añadir la posibilidad de que los medidores puedan realizar acciones a través de comandos..
- Añadir la posibilidad de controlar los actuadores en caso de ser necesario.

BIBLIOGRAFÍA

Desarrollo web:

- MDN Web Docs: https://developer.mozilla.org/es/docs/Web
- W3Schools: https://www.w3schools.com/
- OpenBootcamp: https://open-bootcamp.com/

Desarrollo de servidores en Node JS:

- Documentación oficial de Node.js: https://nodejs.org/es/docs/
- https://www.w3schools.com/nodejs/
- https://www.w3schools.com/nodejs/nodejs_http.asp
- https://nodejs.dev/learn/creating-a-http-server-in-nodejs
- https://www.digitalocean.com/community/tutorials/how-to-create-a-web-server-in-node-js-with-the-http-module

- https://www.geeksforgeeks.org/simple-web-server-using-node-js/
- https://medium.com/@joe.reidy/how-to-build-a-basic-http-server-in-node-js-8c5eb1f5ff5d

Programación en ESP con WIFI:

- Documentación oficial de ESP8266: https://docs.espressif.com/projects/esp8266-rtos-sdk/en/latest/
- Tutorial de programación de ESP8266 en Instructables: https://www.instructables.com/Getting-Started-With-ESP8266WiFi-Module/

Protocolo MQTT:

- Documentación oficial de MQTT: https://mqtt.org/documentation/
- Tutorial de MQTT en Adafruit: https://learn.adafruit.com/mqtt-adafruit-io-and-you/overview
- https://www.hivemq.com/blog/how-to-install-mqtt-broker/
- https://github.com/mqttjs/MQTT.js
- https://www.emqx.io/blog/how-to-set-up-mqtt-broker-on-nodejs
- https://mosca.io/
- https://www.npmjs.com/package/aedes

Broker MQTT usando MOSCA:

- Documentación oficial de Mosca: https://github.com/mcollina/mosca/wiki
- Tutorial de Mosca en Medium: https://medium.com/@ilkerb/mosca-an-open-source-mqtt-broker-that-scales-7f424d878aba
- https://www.hivemq.com/blog/how-to-install-mqtt-broker/
- https://github.com/mqttjs/MQTT.js
- https://www.emqx.io/blog/how-to-set-up-mqtt-broker-on-nodejs
- https://mosca.io/
- https://www.npmjs.com/package/aedes

6.2 FACTIBILIDAD ECONÓMICA: VAN & TIR

SE CALCULÓ CON **11 SERVICIOS** EN EL AÑO 1 MANTENIENDO LAS VENTAS AÑO A AÑO.

	0	1	2	3	4	5
Ingreso		$ 24,200	$ 24,200	$ 24,200	$ 24,200	$ 24,200
Costos variables		$ -3,729	$ -3,729	$ -3,729	$ -3,729	$ -3,729
Costos fijos		$ -18,486	$ -22,686	$ -25,686	$ -29,886	$ -32,886
Depreciación maquinaria		$ -260	$ -260	$ -260	$ -260	$ -260
Impuestos por ventas		$ 1,210	$ 1,210	$ 1,210	$ 1,210	$ 1,210
Utilidad		$ 2,935	$ -1,265	$ -4,265	$ -8,465	$ -11,465
Impuesto a las ganancias		$ -881	$ 380	$ 1,280	$ 2,540	$ 3,440
Utilidad neta		$ 2,055	$ -886	$ -2,986	$ -5,926	$ -8,026
Depreciación maquinaria		$ 260	$ 260	$ 260	$ 260	$ 260
Capital de trabajo	$ -4,839					$ 4,839
Maquinaria	$ -1,300					
Servidor y aplicacion web	$ -300					
Valor de desecho						$ 650
Valor de liquidacion						$ 20,000
Flujo del proyecto	$ -6,439	$ 2,315	$ -626	$ -2,726	$ -5,666	$ 17,724
VAN	-$ 354.82					
TIR	11%					

En un segundo análisis se CALCULÓ CON **11 SERVICIOS** EN EL AÑO 1 INCREMENTANDO 25% LAS VENTAS CADA AÑO.

	0	1	2	3	4	5
Ingreso		$ 24,200	$ 30,250	$ 37,813	$ 47,266	$ 59,082
Costos variables		$ -3,729	$ -4,661	$ -5,827	$ -7,283	$ -9,104
Costos fijos		$ -18,486	$ -22,686	$ -25,686	$ -29,886	$ -32,886
Depreciación maquinaria		$ -260	$ -260	$ -260	$ -260	$ -260
Impuestos por ventas		$ 1,210	$ 1,513	$ 1,891	$ 2,363	$ 2,954
Utilidad		$ 2,935	$ 4,155	$ 7,931	$ 12,200	$ 19,786
Impuesto a las ganancias		$ -881	$ -1,247	$ -2,379	$ -3,660	$ -5,936
Utilidad neta		$ 2,055	$ 2,909	$ 5,551	$ 8,540	$ 13,850
Depreciación maquinaria		$ 260	$ 260	$ 260	$ 260	$ 260
Capital de trabajo	$ -4,839					$ 4,839
Maquinaria	$ -1,300					
Servidor y aplicacion web	$ -300					
Valor de desecho						$ 650
Valor de liquidacion						$ 20,000
Flujo del proyecto	$ -6,439	$ 2,315	$ 3,169	$ 5,811	$ 8,800	$ 39,599
VAN	**$ 30,352.13**					
TIR	**78%**					

6.3 PAYBACK O PLAZO DE RECUPERACIÓN

$$\square\square\square\square\square\square\square = 2 + (6439 - 11295)/5811 = 2.16$$

6.4 PRODUCTOS Y SERVICIOS DE OTROS FABRICANTES

Para la realización del proyecto son necesarios los medidores de la empresa SolverBox. Este medidor homologado cuenta con todas las mediciones necesarias para la realización del prototipo.