Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

DISEÑO DE UN MONITOR INTELIGENTE DE CONSUMO ENERGÉTICO DOMÉSTICO CON CONEXIÓN WIFI Y VISUALIZACIÓN DE DATOS EN DISPOSITIVOS ANDROID

José Manuel Otzoy Chacón
Asesorado por la Inga. Ingrid Salomé Rodríguez García de Loukota

Guatemala, octubre de 2022

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DISEÑO DE UN MONITOR INTELIGENTE DE CONSUMO ENERGÉTICO DOMÉSTICO CON CONEXIÓN WIFI Y VISUALIZACIÓN DE DATOS EN DISPOSITIVOS ANDROID

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

JOSÉ MANUEL OTZOY CHACÓN
ASESORADO POR LA INGA. INGRID SALOMÉ RODRÍGUEZ GARCÍA DE LOUKOTA

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN ELECTRÓNICA

GUATEMALA, OCTUBRE DE 2022

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANA	Inga. Aurelia Anabela Cordova Estrada
VOCAL I	Ing. José Francisco Gómez Rivera
VOCAL II	Ing. Mario Renato Escobedo Martínez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Kevin Vladimir Armando Cruz Lorente
VOCAL V	Br. Fernando José Paz González
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANA	Inga. Aurelia Anabela Cordova Estrada
EXAMINADOR	Ing. José Aníbal Silva de los Angeles
EXAMINADOR	Ing. Hugo Leonel Tiul Valenzuela
EXAMINADOR	Ing. Christian Antonio Orellana López
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DISEÑO DE UN MONITOR INTELIGENTE DE CONSUMO ENERGÉTICO DOMÉSTICO CON CONEXIÓN WIFI Y VISUALIZACIÓN DE DATOS EN DISPOSITIVOS ANDROID

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha 20 de septiembre de 2021.

José Manuel Otzoy Chacón

Guatemala 15 de junio 2022

Ingeniero
Julio César Solares Peñate
Coordinador del Área de Electrónica
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC.

Apreciable Ingeniero Solares,

Me permito dar aprobación al trabajo de graduación titulado ¨**Diseño de un monitor inteligente de consumo energético doméstico con conexión WIFI y visualización de datos en dispositivos Android**¨, del señor **José Manuel Otzoy Chacón**, por considerar que cumple con los requisitos establecidos.

Por tanto, el autor de este trabajo de graduación y, yo, como su asesora, nos hacemos responsables por el contenido y conclusiones de este.

Sin otro particular, me es grato saludarle.

Atentamente,

Inga. Ingrid Rodríguez de Loukota
Colegiada 5,356
Asesora

Ingrid Rodríguez de Loukota
Ingeniera en Electrónica
colegiado 5356

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

Guatemala, 22 de julio de 2022

Señor director
Armando Alonso Rivera Carrillo
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC

Estimado Señor director:

Por este medio me permito dar aprobación al Trabajo de Graduación titulado: **DISEÑO DE UN MONITOR INTELIGENTE DE CONSUMO ENERGÉTICO DOMÉSTICO CON CONEXIÓN WIFI Y VISUALIZACIÓN DE DATOS EN DISPOSITIVOS ANDROID,** desarrollado por el estudiante **José Manuel Otzoy Chacón**, ya que considero que cumple con los requisitos establecidos.

Sin otro particular, aprovecho la oportunidad para saludarlo.

Atentamente,

ID Y ENSEÑAD A TODOS

Ing. Julio César Solares Peñate
Coordinador de Electrónica

ESCUELA DE INGENIERÍA MECÁNICA ELÉCTRICA
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

REF. EIME 54.2022.

El Director de la Escuela de Ingeniería Mecánica Eléctrica, después de conocer el dictamen del Asesor, con el Visto Bueno del Coordinador de Área , al trabajo de Graduación del estudiante José Manuel Otzoy Chacón: DISEÑO DE UN MONITOR INTELIGENTE DE CONSUMO ENERGÉTICO DOMÉSTICO CON CONEXIÓN WIFI Y VISUALIZACIÓN DE DATOS EN DISPOSITIVOS ANDROID, procede a la autorización del mismo.

Ing. Armando Alonso Rivera Carrillo

Guatemala, 29 de agosto de 2022.

Decanato
Facultad de Ingeniería
24189101- 24189102
secretariadecanato@ingenieria.usac.edu.gt

LNG.DECANATO.OI.694.2022

La Decana de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica, al Trabajo de Graduación titulado: **DISEÑO DE UN MONITOR INTELIGENTE DE CONSUMO ENERGÉTICO DOMÉSTICO CON CONEXIÓN WIFI Y VISUALIZACIÓN DE DATOS EN DISPOSITIVOS ANDROID,** presentado por: **José Manuel Otzoy Chacón,** después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANA
FACULTAD DE INGENIERÍA

Inga. Aurelia Anabela Cordova Estrada
Decana

Guatemala, octubre de 2022

AACE/gaoc

ACTO QUE DEDICO A:

Dios	Por brindarme la inteligencia necesaria para poder finalizar mis estudios universitarios y no desistir en el camino por falta de motivación.
Mis padres	Por brindarme todo el apoyo necesario para superar los retos y adversidades que la vida me ha puesto enfrente.
Mis amigos	Por su ayuda académica y compañía durante la etapa que nos fue posible compartir.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por permitirme acceder a la educación superior de manera gratuita.
Escuela de Mecánica Eléctrica Facultad de Ingeniería	Por hacer posible que los estudiantes tengan la oportunidad de estudiar las carreras de la cuales dispone la escuela.
Catedráticos de la Facultad de Ingeniería	Especial agradecimiento a los catedráticos que priorizan el aprendizaje por encima de todo, e imparten su clase de una forma amena en la cual el estudiante puede pasar un momento agradable durante el aprendizaje, despertando así el interés por los temas de estudio.
Inga. Ingrid Rodríguez	Por su ayuda en la culminación de este trabajo de graduación, por ser una catedrática ejemplar que me transmitió muchos conocimientos interesantes del mundo de la electrónica, y a su vez me inspiro a seguir aprendiendo.
Comunidades de ayuda en Internet	Usuarios de Stack Overflow, Arduino Forum, GitHub, entre otras. Por la ayuda brindada para la resolución de algunos errores en el desarrollo de software.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
A	Amperio
HP	Caballo de fuerza
$	Dólar
W	Energía
GB	Gigabyte
°C	Grado Celsius
J	Joules
kWh	Kilowatt hora
φ	Phi
%	Porcentaje
P	Potencia activa
S	Potencia aparente
Q	Potencia reactiva
s	Segundos
t	Tiempo
V	Voltio
VA	Voltio-amperio
VAr	Voltio-amperio reactivo
W	Watt

GLOSARIO

AES	Del inglés *Advanced Encryption Standard*. Esquema de cifrado.
APT	Herramienta de paquetes avanzados en sistemas basados en Linux.
ARM	Arquitectura avanzada de set de instrucciones reducido para procesadores.
Bluetooth	Tecnología de red inalámbrica de corto alcance.
Byte	Unidad de medida de información.
CAD	Diseño asistido por computadora.
Calve pública	Clave usada en criptografía asimétrica para cifrar mensajes.
CAM	Fabricación asistida por computadora.
Cifrar	Alterar el contenido de la información para ser incomprensible.
Clave privada	Clave usada en criptografía asimétrica para descifrar mensajes.

Corriente	Flujo de electrones.
CPU	Unidad central de procesamiento.
Credencial	Información usada para autenticar inicios de sesión.
DBMS	Gestor de base de datos.
Descifrar	Proceso realizado posterior al cifrado, cuyo propósito es convertir información incomprensible a información comprensible.
Firmware	Programa encargado de controlar los componentes electrónicos a un nivel lógico bajo.
Fragment	Recurso modular de Android Studio.
Framework	Entorno de trabajo.
GTQ	Quetzal Guatemala.
Hash	Sucesión alfanumérica que representa a un conjunto de datos.
IDE	Entorno de desarrollo integrado.
IoT	Internet de las cosas.
IP	Protocolo de internet.

IPv4	Protocolo de internet versión 4.
IPv6	Protocolo de internet versión 6.
Kernel	Software fundamenta de un sistema operativo que accede a los recursos de hardware de manera segura y los conecta con el software que los solicite.
Layout	Recurso gráfico de Android Studio.
Memoria *flash*	Tipo de alta velocidad memoria no volátil.
Memoria RAM	Memoria de acceso aleatorio.
MQTT	Protocolo de telemetría.
MySQL	Gestor de base de datos licenciado bajo Oracle.
NEMA	Asociación estadunidense encargada de normar estándares de construcción eléctrica.
NVMe	Memoria exprés no volátil.
OAEP	Esquema de relleno de cifrado.
QoS	Calidad de servicio.
RSA	Sistema criptográfico asimétrico.

SDK	Software que ejecuta programas diseñados para un sistema diferente al del anfitrión.
SHA 512	Función hash criptográfica.
SI	Sistema Internacional de Medidas.
Sistema mksa	Hace referencia las nomenclaturas metro, kilogramo, segundo y amperio.
SoC	Sistema en chip.
SQL	Lenguaje de consultas estructurado.
SSD	Unidad de almacenamiento de estado sólido.
SSH	Protocolo de conexión remota segura.
SSH Key	Clave utilizada para autenticar conexiones SSH.
TLL	Lógica transistor a transistor.
TLS	Protocolo encargado de establecer conexiones seguras.
UI	Interfaz de usuario.
URL	Dirección de un sitio web.

USD	Dólar Estados Unidos.
Usuario *root*	Usuario con privilegios administrativos.
Voltaje	Diferencia de potencial eléctrico.
WiFi	Tecnología de red inalámbrica.
3D	3 dimensiones.

RESUMEN

Los conceptos teóricos, diseño y funcionamiento del monitor energético presentado en este trabajo de graduación se describe por medio de los capítulos correspondientes a este.

El primer capítulo explica las bases teóricas para comprender como actúa la potencia eléctrica en entornos suministrados por corriente alterna, además de describir los tipos de potencia según su naturaleza.

El segundo y tercer capítulo describen la definición y funcionamiento de bases de datos y protocolo MQTT haciendo énfasis en temas relevantes a fin de entender el funcionamiento de este monitor energético.

El cuarto capítulo muestra el diseño físico, ensamble de componentes, conexión de dispositivos electrónicos y código de programación correspondientes al dispositivo de medición.

El quinto capítulo describe el desarrollo de la aplicación Android dedicada a la configuración del dispositivo de medición y el monitoreo energético.

El sexto capítulo describe la creación, la configuración y el funcionamiento del servidor remoto que hace posible la interacción de los dispositivos por medio de Internet.

El séptimo capítulo muestra un manual de usuario para dar a conocer la forma de uso de este monitor energético.

OBJETIVOS

General

Diseñar un sistema capaz de medir el consumo energético con el fin de almacenarlo en una base de datos y visualizarlo en tiempo real por medio de dispositivos Android.

Específicos

1. Calcular y mostrar al usuario el valor monetario correspondiente a la energía eléctrica consumida durante un determinado período de tiempo.

2. Acceder de forma remota a la información brindada por el monitor energético.

3. Aportar información útil para las personas que se encuentren desarrollando aplicaciones IoT similares a este monitor energético a fin de incentivar la implementación total o parcial del sistema mostrado en este trabajo de graduación.

INTRODUCCIÓN

Un monitor energético es útil en caso se quiera llevar un mejor control en el consumo de los dispositivos, gracias al concepto de Internet de las cosas (IoT), es posible dotar de funciones extras a este monitor energético, estas funciones consisten en el monitoreo desde cualquier parte del mundo (con acceso a Internet), por medio de dispositivos Android, y se consigue incorporando tecnología WiFi al dispositivo de medición, permitiéndole así obtener una dirección IP con la cual será capaz de alcanzar redes externas.

Para entender de mejor manera el diseño de este monitor energético es preciso dividirlo en 3 secciones fundamentales, la primera consiste en un dispositivo de medición el cual se encarga de medir las variables eléctricas del entorno y su vez enviarlas por medio de internet para ser visualizadas (cuando es requerido), y almacenarlas en una base de datos, la segunda sección consiste en el monitoreo utilizando dispositivos Android por medio de una aplicación dedicada y desarrollada específicamente para este monitor energético, la tercera sección es invisible desde el punto de vista del usuario, esta consiste en el servidor remoto encargado de resolver las peticiones de los dispositivos, con el propósito de realizar las tareas requeridas por los clientes.

El diseño de este monitor energético pretende incentivar al lector a implementar, realizar mejoras o agregar otras funcionalidades al sistema, el diseño cuenta con las descripciones necesarias para poder extraer fragmentos o ideas con la finalidad de ser implementadas en otro proyecto similar o diferente al mostrado, aportando así información a la comunidad de desarrollo IoT u otras ramas de interés.

1. POTENCIA

La potencia se define como la cantidad de energía utilizada en un determinado tiempo y se representa con la siguiente ecuación:

Figura 1. **Fórmula potencia**

$$P = \frac{W}{t}$$

Fuente: elaboración propia, realizado con Adobe Illustrador.

- P = potencia (*watts* W)
- *W*= energía (*joules* J)
- t = tiempo (segundos s)

La energía es la capacidad de realizar un trabajo, por ejemplo: en un motor es capaz de generar trabajo mecánico o cuando fluye corriente por los filamentos de una resistencia esta energía se transforma en calor. (Floyd L, 2007)

La unidad de potencia es el watt (W), pero también se puede medir en caballos de fuerza (hp) que es como usualmente se hace con los motores, es importante hacer énfasis en que un caballo de fuerza es equivalente a 746 watts (1 hp = 746 W), en el Sistema Internacional (SI), la unidad de medida para energía es el joule (J), entonces el watt es la cantidad de *joules* consumidos durante determinados segundos por lo tanto 1 watt es igual a 1 joule consumido durante 1 segundo. Con la información anteriormente descrita se puede representar el

consumo de energía como la potencia consumida durante cierto periodo de tiempo.

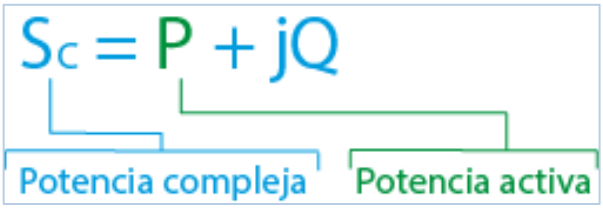
Las empresas que se encargan de suministrar electricidad realizan cobros proporcionales a la energía consumida por el usuario por lo que surgen otras unidades de energía como lo son los watts consumidos durante cierto tiempo. Por la cantidad de energía que se suele suministrar a los usuarios el kilowatt-hora es la unidad de energía más práctica y así la empresa suministradora puede asignar una tarifa por cada 1 000 watts consumidos durante una hora.

En corriente alterna (régimen permanente), la potencia se comporta de forma distinta a como se comporta en corriente directa, debido a que en corriente directa la potencia es un valor que pertenece al conjunto de los números reales y en corriente alterna es un valor que pertenece al conjunto de los números complejos, es importante recalcar que se puede dar el caso en el que la potencia sea puramente imaginaria o real, estas definiciones dan origen a lo que se conoce como potencia activa, reactiva, potencia aparente, triángulo de potencia y factor de potencia.

1.1. Potencia activa

Normalmente se representa con la letra mayúscula P y su unidad de medida es el watt W perteneciente al sistema mksa. Esta potencia tiene un valor que pertenece al conjunto de los números reales y en un circuito de corriente DC por su naturaleza continua esta es la única potencia que resulta del consumo de sus cargas.

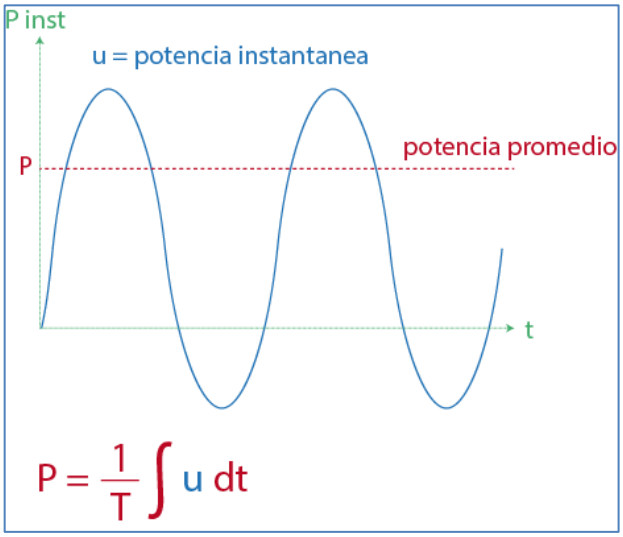
Figura 2. **Potencia activa**



Fuente: elaboración propia, realizado con Adobe Illustrator.

En un circuito AC la potencia activa es la parte real de la potencia compleja, también se conoce como potencia promedio puesto que uno de los métodos para encontrar su valor es a partir del valor promedio de la potencia instantánea (el producto del voltaje por la corriente da como resultado la potencia instantánea). La integral durante un periodo de la potencia instantánea por el producto del inverso del periodo da como resultado la potencia media o promedio.

Figura 3. **Potencia promedio**

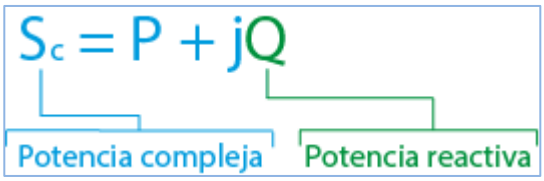


Fuente: elaboración propia, realizado con Adobe Illustrator.

1.2. Potencia reactiva

La parte imaginaria de la potencia compleja se conoce como potencia reactiva y se representa con la letra mayúscula Q y su unidad es el voltio-amperio reactivo VAR en el sistema mksa.

Figura 4. **Potencia reactiva**



Fuente: elaboración propia, realizado con Adobe Illustrator.

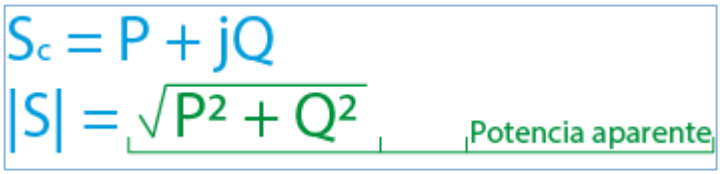
El signo de la potencia reactiva depende de qué tipo de carga predomine en el circuito, si en el circuito predomina la carga inductiva entonces la potencia reactiva será positiva y por el contrario el signo será negativo si la carga capacitiva predomina en el circuito.

1.3. Potencia aparente

La magnitud del vector de potencia compleja se conoce como potencia aparente y se representa mediante la letra mayúscula S su unidad es el voltio-amperio del sistema mksa, matemáticamente se define como la raíz cuadrada de la suma de la potencia activa y reactiva ambas elevadas al cuadrado, es esa la razón por la que también se le conoce como potencia total, su valor pertenece al conjunto de los números reales.

La potencia aparente será igual a la potencia activa en el caso en el que la potencia reactiva sea nula y de la misma manera la potencia aparente será igual a la potencia reactiva si la potencia activa es nula.

Figura 5. **Potencia aparente**

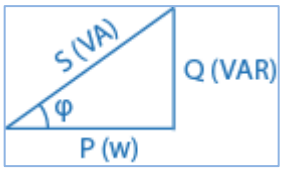


Fuente: elaboración propia, realizado con Adobe Illustrator.

1.4. Triángulo de potencia

Es un método geométrico en el cual se encuentran los valores de potencia activa, reactiva y aparente. Tiene el fin de poder visualizar las potencias por medio de un triángulo rectángulo que resulta realmente útil para realizar cálculos. (Edminister J, 1985).

Figura 6. **Triángulo de potencia**



Fuente: elaboración propia, realizado con Adobe Illustrator.

El cateto vertical pertenece a la potencia reactiva y la dirección a la que apunta este depende del tipo de carga, si es inductiva apunta hacia arriba con

respecto al cateto horizontal y si es capacitiva apunta hacia abajo. El cateto horizontal siempre apunta hacia la misma dirección y toma el valor de la potencia activa o real. La hipotenusa del triángulo tiene el valor de la potencia aparente.

Al ser un triángulo rectángulo se puede hacer uso de la geometría a la que se rige dicha figura y como consecuencia definir ecuaciones de potencia en función de sus catetos, hipotenusa o el ángulo entre la hipotenusa y cualquiera de sus catetos. La figura 7 muestra algunas de esas ecuaciones.

Figura 7. **Ecuaciones de potencia en función de los catetos del triángulo de potencia**

Fuente: elaboración propia, realizado con Adobe Illustrator.

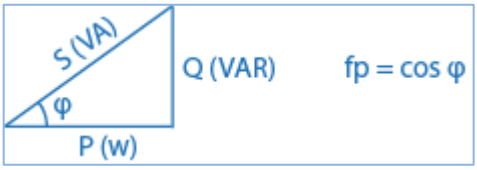
1.5. Factor de potencia

El factor de potencia es un valor adimensional que toma valores entre 0 y 1, este valor se encuentra en función al triángulo de potencia y se calcula obteniendo el valor del coseno del ángulo formado entre la hipotenusa y el cateto correspondiente a la potencia activa (este ángulo es comúnmente representado por la letra griega phi).

El factor de potencia puede interpretarse como un indicador de eficiencia, proporciona el dato de que tan eficientemente se está aprovechando la potencia entregada, un factor de potencia con valor 1 indica que toda la potencia entregada

se está aprovechando para generar un trabajo y no existen desperdicios, contrario a un factor de potencia menor a 1 en donde una porción de la potencia entregada no está usándose para generar un trabajo real y está representada en la potencia reactiva.

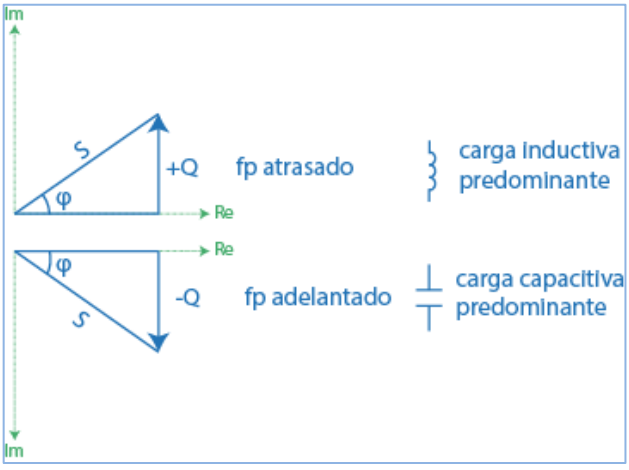
Figura 8. **Factor de potencia**



Fuente: elaboración propia, realizado con Adobe Illustrator.

El factor de potencia puede estar atrasado o retrasado dependiendo del tipo de carga que predomine en el circuito, en la figura 9 se observa cómo las cargas afectan al factor de potencia.

Figura 9. **Triangulo de potencia carga inductiva y capacitiva**

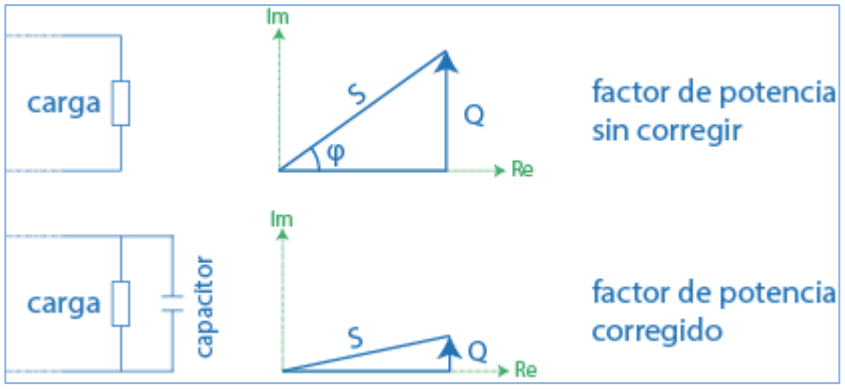


Fuente: elaboración propia, realizado con Adobe Illustrator.

En el sector industrial está penalizado consumir grandes cantidades de potencia reactiva y es un indicador del uso ineficiente de los recursos energéticos porque de la energía suministrada o contratada solo se está usando una porción para generar trabajo real y el resto se está disipando en forma de potencia reactiva, las penalizaciones que se aplican son monetarias y el indicador para ejecutar las es el factor de potencia, lo que quiere decir que si una empresa tiene un factor de potencia menor al estipulado será penalizada monetariamente por el proveedor de servicios energéticos.

En la industria usualmente predomina un factor de potencia atrasado debido a las cargas inductivas como lo son: motores, transformadores, lámparas incandescentes, entre otros. Para corregir este factor de potencia (corriente AC monofásica), se debe de acoplar una carga capacitiva (banco de capacitores) en paralelo a la carga tal que la competente reactiva Q disminuya en magnitud y así adelante el factor de potencia y lo lleve lo más próximo a 1 para así evitar penalizaciones económicas.

Figura 10. **Corrección del factor de potencia (AC monofásica)**

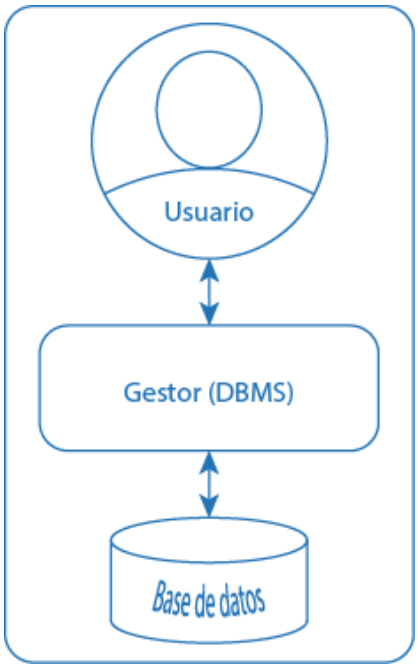


Fuente: elaboración propia, realizado con Adobe Illustrator

2. BASES DE DATOS

En informática una base de datos es un espacio virtual destinado a almacenar datos y con la facilidad de posteriormente acceder a ellos una vez almacenados, al ser únicamente un espacio virtual no es capaz de hacer funciones de almacenamiento o consultas por si sola y por consiguiente, se necesita de un sistema que sea capaz de ejecutar dichas tareas requeridas por el usuario, al sistema encargado de realizar la interacción entre la base de datos y el usuario se le conoce como gestor de base de datos o DBMS (del inglés *Database Management System*).

Figura 11. **Resumen sistema base de datos**



Fuente: elaboración propia, realizado con Adobe Illustrator.

Actualmente existen muchos gestores de bases de datos en el mundo de la informática, pero los más usados son: Microsoft SQL, MySQL, Oracle, PostgreSQL y MongoDB.

La elección del gestor de base de datos depende de la necesidad o aplicación a la que está destinada la base de datos, si bien el objetivo principal de los gestores es realizar una interacción entre el usuario y la base de datos, la forma en la que se realiza la interacción es una de las diferencias de algunos gestores y es un factor muy importante por tomar en cuenta.

2.1. Bases de datos no relacionales

Es un tipo de base de datos poco usado, razón por la cual aún no ha sido estandarizado por ningún sistema como lo es el caso de las relacionales, se difieren principalmente por la forma en la que se almacenan los datos pues lo hacen en forma de documentos, en ellas se puede almacenar datos sin seguir ningún tipo de estructura y esto las hace muy útiles a la hora de tratar con información de la que no se tenga la noción del tipo de dato al que pertenece.

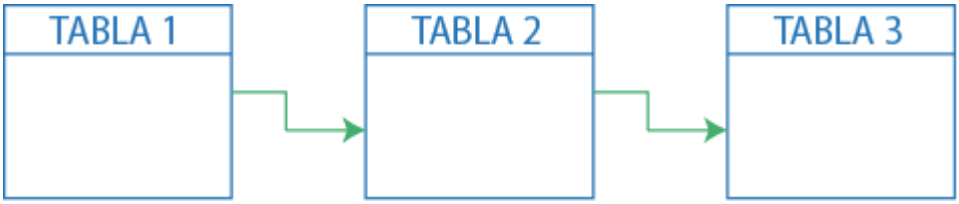
Las bases de datos no relacionales están diseñadas para interactuar con una gran cantidad de datos y ser eficientes con ellos, además de poseer una gran escalabilidad horizontal. (Meirer A y Kaufmann M, 2019)

2.2. Bases de datos relacionales

Este modelo de base de datos es el más popular y usado, ofrece una estructuración por medio de relaciones y está regido por el lenguaje estructurado de consultas SQL (del inglés *Structured Query Language*), los datos almacenados y sus características se guardan en las filas de una tabla, esas filas

se conocen como tuplas y cada tupla representa un dato de la colección almacenada en la tabla. (Oppel A y Sheldon R, 2010)

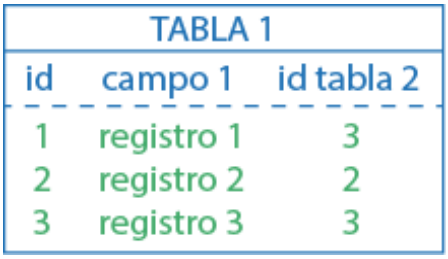
Figura 12. **Estructura relacional**



Fuente: elaboración propia, realizado con Adobe Illustrator.

Componentes de una tabla en la estructura relacional:

Figura 13. **Componentes de una tabla en el modelo relacional**



Fuente: elaboración propia, realizado con Adobe Illustrator.

- Tabla: elemento principal en este modelo y en ella se guardan los datos denominados registros, el nombre de una tabla no puede repetirse dentro de una base de datos.

- Campos: a las columnas de cada tabla se le conoce como campos y sirven para determinar el tipo de registro que almacenan.

- Registros: comprenden las filas de cada tabla y son estos registros los que conforman la información, cada registro representa un dato de la colección contenida en la tabla.

- Claves primarias: forma en la que se identifica cada registro del resto es asignándole un identificador único denominado clave primaria, cada tabla debe tener obligatoriamente un campo destinado para almacenar la clave primaria y este registro no se puede repetir dentro del campo.

- Claves foráneas: son calves primarias pertenecientes a otra tabla y esto permite registros repetidos en un campo asignado a almacenar claves foráneas. Estas claves hacen referencia a un campo almacenado en otra tabla relacionada.

En una base de datos relacional las asociaciones o relaciones se pueden llevar a cabo de distintas maneras, las formas más usuales de llevar a cabo esas relaciones son:

- Una a una: cuando la relación se da entre un registro y solamente un registro de una tabla ajena, entonces se efectúa una relación una a una.

- Una a varias: relación una a varias se efectúa cuando un registro de una primera tabla se referencia a varios registros de una segunda tabla.

- Varias a varias: este tipo de relación se compone de varios registros de una primera tabla relacionados con varios registros de una segunda tabla.

2.2.1. Base de datos local

Una base de datos local consiste en centralizar los datos en un solo equipo ubicado dentro de la red local, esto significa que solamente podrá ser accesible desde puntos dentro de la misma red y no desde redes externas, al no estar distribuida en distintas partes la carga de trabajo representada por las gestiones realizadas en la base de datos radica en un solo equipo y esto puede llegar a reducir su eficiencia a nivel lógico.

2.2.1.1. Ventajas

- Seguridad: debido a que toda la información está centralizada en un solo lugar es más fácil restringir los accesos a personas no deseadas lo cual disminuye las vulnerabilidades.

- Resolución de problemas: al ser un servicio centralizado se sabe con certeza de que punto proviene el fallo y esto disminuye el tiempo de identificación del punto de falla.

2.2.1.2. Desventajas

- Accesibilidad: so se puede acceder desde una red externa y por lo tanto no es enrutable hacia la red de internet.

- Redundancia: al ser de naturaleza local solo existe una ruta de acceso y en caso falle el servicio estará inhabilitado dado que no se cuenta con una ruta de respaldo.

- Indisponibilidad: al carecer de rutas de respaldo el tiempo de indisponibilidad es igual al tiempo que se demore la resolución de la falla lo que genera una indisponibilidad bastante considerable.

2.2.2. Base de datos remota

Este tipo de base de datos es muy usado debido a su alto nivel de disponibilidad y hay muchos proveedores en la Internet que ofrecen paquetes de alojamiento para bases de datos remotas, funcionan utilizando una copia local de la base de datos remota la cual actualizan a cada cierto tiempo y por su naturaleza son altamente escalables, también se caracterizan por ser muy eficientes con el rendimiento. Son alcanzables desde cualquier parte del mundo por medio de una dirección IP pública lo que las hace una buena opción a implementar para el caso de organizaciones multinacionales que requieran acceder a los mismos datos desde cualquier parte del mundo sin importar la red local a la que se pertenezca.

2.2.2.1. Ventajas

- Accesibilidad: por ser una base de datos enrutable hacia la red de Internet, se puede acceder desde cualquier parte del mundo siempre y cuando se cuente con las credenciales.

- Disponibilidad: al ser un servicio en red puede tener enlaces redundantes lo cual aumenta la disponibilidad, en caso falle una de las rutas para alcanzar la base de datos habrá una ruta de respaldo que puede funcionar mientras se resuelve el fallo en la ruta principal o por defecto.

- Eficiencia: debido a que la información puede estar distribuida en distintos puntos los servicios se pueden segmentar según la necesidad, esto disminuye el tráfico y la carga introducida por cada servicio.

2.2.2.2. Desventajas

- Seguridad: es un servicio alcanzable desde la red de Internet la información está expuesta a ciber ataques y la privacidad de datos puede resultar comprometida por vulnerabilidades sin resolver.

- Disponibilidad de red: una de las desventajas más notables es el hecho de que se necesita una conexión estable a Internet o de lo contrario no se podrá acceder a la información, fallos en el servicio brindado por el proveedor de Internet pueden ocasionar indisponibilidad de los datos y esto significa que este tipo de implementación es vulnerable a fallos ajenos dentro de la red de la organización.

2.2.3. Consultas SQL

Una vez creadas las tablas y almacenados los datos se utiliza la instrucción SELECT (perteneciente al lenguaje SQL) para recuperar los datos anteriormente guardados, la sintaxis de la instrucción SELECT consta de cláusulas de búsqueda de las cuales las más importantes son:

- WHERE: se utiliza para establecer condiciones en la búsqueda efectuada.

- GROUP BY: en esta cláusula puede agrupar según criterios del usuario los datos devueltos por una consulta.

- HAVING: con el uso de esta cláusula se puede establecer un grupo de condiciones en la búsqueda efectuada.

- ORDER BY: ordena los resultados devueltos por una consulta según criterios establecidos por el usuario.

Figura 14. **Consultas SQL**

Fuente: elaboración propia, realizado con Adobe Illustrator.

Tal como lo muestra la figura 14 después de escribir la palabra reservada SELECT se debe indicar sobre que campos o columnas se aplicará la instrucción y bien puede ser solo una columna o un conjunto de columnas, para seleccionar todas las columnas existentes en una tabla se debe escribir un asterisco seguido de la palabra SELECT y la instrucción tomará en cuenta todas las columnas que contenga la tabla.

Las siguientes palabras después de SELECT y su cláusula hacen referencia al lugar de donde se va a tomar la información en este caso quiere decir que sobre la tabla existente se hará la consulta, esto se hace escribiendo la palabra reservada FROM seguido del nombre de la tabla. Por último, se deben especificar las cláusulas con las que se efectuara la consulta, cada cláusula cuenta con condiciones y criterios de búsqueda y el uso de una o de otra depende del tipo de búsqueda que se quiera efectuar, por ejemplo, usando la cláusula WHERE se puede definir una condición que muestre solo los registros que pertenecen al mes de diciembre y usando la cláusula ORDER BY se puede ordenar las fechas de mayor a menor o viceversa según sea la necesidad del caso.

3. PROTOCOLO MQTT

MQTT es un protocolo cuya función es transportar mensajes entre dispositivos y se caracteriza por ser ligero debido a su bajo consumo de ancho de banda, está diseñado para cubrir las necesidades del Internet de las cosas IoT (del inglés *Internet of Things*), actualmente es usando en las industrias de automovilismo, logística, manufactura, casas inteligentes, entre otros. (Steve's Internet Guide, 2022)

La gran escalabilidad de este protocolo hace posible la conexión de millones de dispositivos remotos con gran facilidad, soporta conexiones bidireccionales y tiene la capacidad de cifrar mensajes implementando la tecnología TLS (del inglés *Transport Layer Security*), además de usar autenticación del lado del cliente a través de protocolos modernos como lo es el caso de OAuth (del inglés *Open Authorization*).

3.1. ¿Cómo funciona?

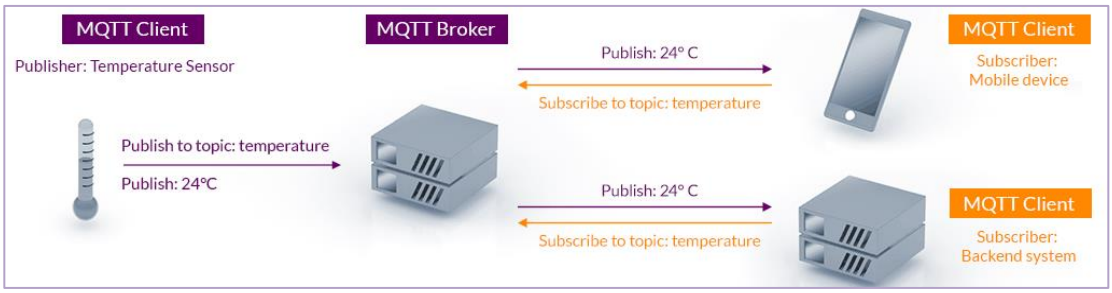
MQTT utiliza la arquitectura publicador/suscriptor para la comunicación entre dispositivos, el encargado de enlazar al publicador con el suscriptor correcto recibe la denominación de bróker y también se le conoce como el intermediario entre publicador – suscriptor.

- *Topic*: la comunicación se lleva a cabo mediante *topics*, el publicador envía el mensaje al *topic* designado y los suscriptores lo reciben, cuando un cartero reparte la correspondencia necesita saber a qué buzón depositar la carta en este caso un *topic* cumple la misma función que un buzón, pero

con la diferencia de que existen varios usuarios que usan este mismo buzón llamado *topic* y por ello se le puede definir como un canal de comunicación.

La figura 15 muestra la arquitectura publicador/suscriptor en ella se puede observar que tanto el publicador como el suscriptor son clientes MQTT, el publicador es un sensor de temperatura que comparte sus mediciones por medio del *topic* “temperatura”, luego por medio del *broker* los suscriptores (dispositivos) del *topic* son capaces de recibir los mensaje enviados por el sensor de temperatura.

Figura 15. **Arquitectura MQTT**



Fuente: MQTT (2022). *MQTT Publish / Subscribe Architecture.* Consultado el 4 de enero de 2022. Recuperado de https://mqtt.org.

3.2. *Broker*

Es el encargado de la distribución de los mensajes y opera del lado del servidor, se caracteriza por ser liviano y de bajo consumo de recursos, es capaz de soportar la interacción de muchos publicadores y suscriptores. Actualmente, existen varios servicios de bróker, entre los más populares se tienen:

- Cassandana
- EMQ X
- HiveMQ
- Moquette
- Mosca
- Mosquitto

El *broker* Mosquitto es bastante utilizado por ser una solución de código abierto que cuenta con una licencia EPL/EDL, se puede implementar tanto en microcontroladores como hasta en servidores y ofrece las versiones 5.0, 3.1.1 y 3.1 del protocolo MQTT.

3.3. Publicador

En la arquitectura de mensajería MQTT el publicador es el emisor del mensaje, este envía su información en forma de mensajes a los *topics* destinados, normalmente del lado del publicador se encuentran los sensores y los sistemas que se encargan de transmitir la información de estos sensores.

Al ser MQTT un protocolo de mensajería bidireccional un mismo dispositivo puede funcionar como suscriptor y publicador, este es el caso de los dispositivos usados en casas inteligentes, pues estos dispositivos necesitan recibir y enviar información de su estado con el propósito de recibir nuevas instrucciones.

3.4. Suscriptor

Al momento que un dispositivo se suscribe a un *topic* este estará habilitado para recibir la información enviada por los publicadores y pasará a recibir la designación de suscriptor, un dispositivo puede suscribirse a varios *topics* y recibir la información de muchos publicadores.

El suscriptor puede ser cualquier dispositivo capaz de alcanzar el servidor donde está alojado el *broker*, esto permite diversidad de dispositivos como lo son: celulares, tabletas, microcontroladores, servidores completos, entre otros. Debido al avance del Internet de las cosas IoT cada vez más los dispositivos traen incorporado *hardware* para tener acceso a la red de Internet y como consecuencia son configurables como un cliente MQTT.

3.5. Estructura de un mensaje

Un menaje MQTT se compone de 3 partes principales que son: el encabezado fijo, encabezado opcional y la carga. El tamaño mínimo que un mensaje puede tener es de 2 bytes, que es muy liviano.

Figura 16. **Estructura de un mensaje MQTT**

Fuente: elaboración propia, realizado con Adobe Illustrator.

- Encabezado fijo: es obligatorio y su longitud esta entre 2 y 5 bytes, se divide en el encabezado de control y la longitud del paquete.
    - Encabezado de control: contiene la información del tipo de mensaje que se está enviado pudiendo ser del tipo publicación, conexión, desconexión u otro.

    - Longitud del paquete: en esta división del encabezado fijo se encuentra la información de la longitud que tiene el mensaje.

- Encabezado opcional: puede no estar presente ya que la información que puede almacenar depende del tipo de mensaje, hay ciertos mensajes que pueden requerir información adicional y para ello usan el encabezado opcional.

- Carga: esta parte del mensaje contiene la información que se quiere transportar hacia los suscriptores.

3.6. Calidad de servicio (QoS)

La calidad de servicio QoS (del inglés *Quality of Service*), es un sistema utilizado para priorizar y garantizar el correcto funcionamiento de los servicios en los que se implementa, MQTT es un protocolo que soporta QoS en sus mensajes y es capaz de segmentarlos en niveles de prioridad. Los niveles implementados en MQTT son:

- QoS 0: este nivel lleva el nombre de *at most one* (como máximo uno), porque envía el mensaje una sola vez sin importar si el mensaje fue recibido o no, usando QoS 0 no hay opción de corregir los mensajes en los que ocurrió un fallo durante el transporte o recepción.

- QoS 1: tiene el nombre de *at least one* (al menos uno), tiene la capacidad de corregir fallos en el envío de mensajes y lo hace por medio de confirmaciones de parte de los suscriptores con lo cual se garantiza la recepción, pero esto puede provocar que por algún fallo se reciba el mismo mensaje 2 veces.

- QoS 2: llamado *exactly one* (exactamente uno) funciona de la misma manera que QoS 1 pues tiene la posibilidad de reenviar el mensaje en caso ocurra un error, pero se asegura que el mensaje solo sea recibido una vez por el suscriptor y no dos veces en caso de un fallo. Este nivel de calidad de servicio puede consumir más recursos que QoS 0 y QoS 1.

4. DISEÑO DISPOSITIVO DE MEDICIÓN

El dispositivo de medición de este monitor energético está diseñado para ser de fácil uso y de conexión sencilla, por medio de modelado 3D se desarrolló un contenedor que permite tener un dispositivo compacto con todos los componentes electrónicos y eléctricos en su interior. Con el fin de cumplir con los criterios de diseño determinados por las funciones del dispositivo se ha elegido implementar un controlador capaz de establecer conexiones a internet por medio de Wifi.

4.1. Funciones dispositivo de medición

La función principal es obtener el consumo energético y enviarlo a un servidor mediante internet, pero también debe de ser capaz de guardar el consumo energético y la información relacionada a este. Para aclarar de mejor manera cómo funciona el dispositivo, se describen sus funciones:

4.1.1. Guardar información de usuario

La información de usuario se guarda en la memoria no volátil del dispositivo de medición, y se comparte al servidor remoto para ser almacenada en base de datos. Si al momento de recibir la información de usuario el dispositivo de medición no cuenta con conexión a Internet el envío al servidor se llevará a cabo cuando se logre establecer la conexión. La manera en la que el usuario realiza la configuración (basada en su información), del dispositivo de medición es por medio de un dispositivo Android, con la aplicación correspondiente previamente instalada.

Si aún no se ha configurado el dispositivo de medición, este no se podrá utilizar porque la información brindada por el usuario se utiliza para realizar procedimientos fundamentales en el monitoreo energético. La información de usuario solicitada para realizar una configuración es la siguiente:

- Credenciales Wifi
- Tarifa energética
- Moneda
- Día de pago
- Nombre del dispositivo al que se le está realizando la medición

4.1.2. Enviar información energética para la visualización en tiempo real en dispositivos Android

Si el dispositivo de medición cuenta con conexión a internet y algún dispositivo Android solicita la visualización en tiempo real entonces las mediciones realizadas por el sensor energético serán enviadas a cada segundo al servidor MQTT, si no hay ningún dispositivo Android solicitando la visualización en tiempo real esta se suspende hasta recibir una nueva solicitud de visualización. Para que un dispositivo Android pueda funcionar como visualizador es necesario haber instalado previamente la aplicación correspondiente a este monitor energético. Las mediciones que se envían al servidor para su visualización son las siguientes:

- Voltaje
- Corriente
- Potencia
- Energía consumida
- Valor monetario equivalente a la energía consumida

4.1.3. Enviar información para ser almacenada en base de datos

Con una frecuencia de 1 vez cada 10 minutos se solicita por medio del servidor la información de consumo de todos los dispositivos de medición conectados a Internet y luego esta se almacena en base de datos para su futura consulta por parte del usuario (las consultas se realizan por medio de la opción “consultar historial” de la aplicación Android). Todos registros se almacenan en el sistema durante un año, lo que significa que el usuario solo puede realizar consultas de historial no mayores a un año de antigüedad. La información que se envía al servidor por medio de esta función es la siguiente:

- Nombre dispositivo de medición
- Nombre del dispositivo al que se le está realizando la medición
- Tarifa energética
- Moneda
- Día de pago
- Voltaje
- Corriente
- Potencia
- Energía consumida
- Valor monetario equivalente a la energía consumida
- Fecha y hora
- Zona horaria

4.2. Diseño 3D

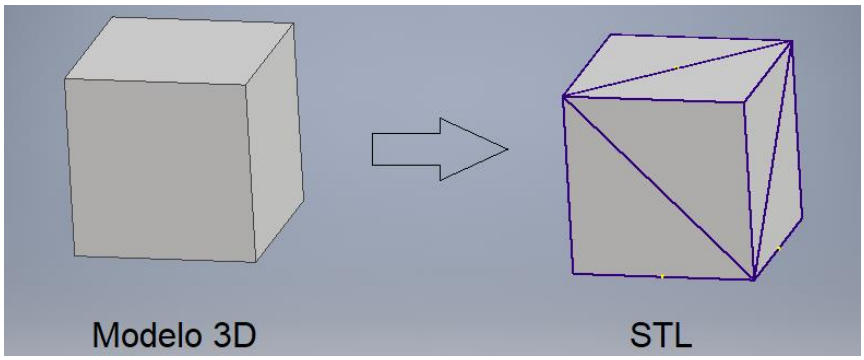
El diseño de objetos en 3 dimensiones o 3D es una técnica para representar gráficos a través de un espacio tridimensional, usualmente asistido por computadora, empleado en varias ramas de la ingeniería y arquitectura para el desarrollo de prototipos, representación visual de objetos a escala, proyección de modelos en construcción estructural, fabricación de piezas mecánicas, entre otros.

El diseño 3D es muy importante en la industria actual debido al avance tecnológico de la última década. Hoy en día es posible simular situaciones reales a partir de objetos 3D y estudiar su interacción tomando en cuenta las variables establecidas por el entorno, esta gran ventaja permite optimizar los recursos y disminuir las situaciones de riesgo a la hora de implementar proyectos en la vida real puesto que previamente pueden ser simulados y estudiados.

4.2.1. Archivo STL

Es uno de los archivos más populares en el diseño 3D asistido por computadora, compatible con la mayoría de software de diseño CAD. Un archivo STL es una versión liviana de un archivo CAD, se caracteriza por eliminar el interior de objetos sólidos dejando únicamente el área superficial, las figuras en formato STL están formadas a partir de triángulos, lo que significa que un archivo STL de alta resolución está formado por un mayor número de triángulos en comparación a un STL de resolución normal. (Regidor A, 2016)

Figura 17. **Representación STL de un modelo 3D**



Fuente: elaboración propia, realizado con Autodesk Inventor.

En la figura 17 se puede observar como el cubo del modelo 3D está formado por cuadrados en la superficie, mientras que la versión STL de ese mismo cubo está formado por triángulos, cada cara contiene 2 triángulos, entonces, para formar el cubo STL completo se necesita de 12 triángulos, esto hace que el cubo STL se convierta en una versión simplificada del cubo 3D original.

La ventaja que presenta un archivo STL contra la versión original del modelo 3D es la lectura por parte de la mayoría de software y herramientas CAD, usualmente el modelo 3D original se limita únicamente al software de creación, lo que impide que pueda ser leído en otro entorno de diseño.

4.2.2. Software de diseño 3D

Existen una variedad bástate extensa de software de diseño 3D enfocado a la arquitectura, construcción, ingeniería, diseño mecánico, entre otros. Si bien todo software de diseño 3D es capaz de representar objetos tridimensionales, cada uno tiene características diferentes que lo hacen útil para una tarea en

específico. A continuación, se presenta una lista del software de diseño 3D más usados en la actualidad:

- TinkerCAD: es un servicio Web gratuito con enfoque educativo orientado a principiantes del diseño 3D, cuenta con una interfaz de diseño fácil de usar, principalmente empleado en la creación de piezas imprimibles, además de contar con otras herramientas de interés como lo son la simulación electrónica de circuitos y el desarrollo de programas que automatizan el diseño 3D.

- SketchUP: desarrollado por Trimple, está enfocado a profesionales y aficionados, cuenta con una curva de aprendizaje baja lo que permite a usuarios principiantes aprender rápidamente a crear modelos sofisticados y elaborados. Ofrece varios paquetes de paga y uno gratuito que se ejecuta en la nube por medio de la versión Web del programa.

- Fusion 360: es una solución Web de paga implementada por Autodesk, enfocada a la industria de manufactura cuenta con herramientas CAD, CAM, CAE y la posibilidad de diseñar circuitos impresos integrados. Soporta diseño generativo, colaboración en la nube, renderización y documentación fotorrealista.

- Inventor: usado en el diseño mecánico 3D, ofrece una solución profesional en ingeniería de productos y diseño CAD, cuenta con automatización de diseño y herramientas dedicadas para el modelado de tuberías, tubos, chapas de metal y diseño de estructuras. Una de las funciones más importantes de este software es el modelado de ensamblajes a partir de piezas previamente importadas al entorno de trabajo. Este programa de

diseño únicamente cuenta con versiones de paga distribuidas por Autodesk.

- AutoCAD: se caracteriza por implementar diseño asistido por computadora en 2D y 3D, especialmente usado en la industria de la construcción para dibujar planos arquitectónicos y modelar en 3D todo tipo de estructuras e interiores. Es una opción de paga desarrollada por Autodesk.

4.2.3. Impresión 3D

La impresión 3D es una técnica usada para crear objetos reales a partir de modelos 3D, el proceso se realiza imprimiendo capas consecutivas a manera de formar las áreas transversales de las que se compone el modelo, esta técnica también es llamada manufactura por adición y realmente la idea no es nada nueva, las primeras patentes de este proceso se registraron en los años 70, pero no fue hasta 1984 que se generó un producto usando la técnica anteriormente mencionada. Los avances tecnológicos han permitido mejorar y automatizar los procesos de impresión 3D hasta llegar a las facilidades de uso que se tienen hoy día.

Actualmente, la impresión 3D está haciendo aportes significativos a la humanidad, prueba de ello son las prótesis e implantes médicos fabricados empleando esta técnica, y prototipos que hace un tiempo atrás hubieran sido muy complejos o costosos de fabricar, sin duda alguna hoy día el proceso de impresión 3D suele ser más accesible en comparación a otros procesos de fabricación de piezas. Los materiales más usados para imprimir son los plásticos y metales.

Los plásticos usados como material de impresión usualmente se les puede encontrar en forma de filamento y pueden ser de distintos colores y materiales. Según el tipo de material el objeto impreso puede presentar mayor resistencia en exteriores, mayor flexibilidad, transparencia u otro tipo de propiedades. Algunos de los materiales plásticos más usados en impresión 3D se describen en la siguiente lista:

- PLA: es uno de los materiales más usados para la impresión 3D debido a que es reciclable y hecho a partir de materia prima orgánica, se caracteriza por su nula emisión de gases tóxicos y su baja resistencia térmica. (Impresoras3D.com, 2018)

- ABS: material hecho a base de petróleo, se caracteriza por ser muy resistente y estable a temperaturas de entre -90 °C y 80 °C, para trabajar con este material se debe tomar todas las precauciones correspondientes como lo es un ambiente ventilado ya que produce gases nocivos a la salud. Es principalmente usado en la fabricación de piezas industriales y de automoción. (Impresoras3D.com, 2018)

- PETG: compuesto por PET y Glicol, entre sus propiedades se encuentra, la flexibilidad, alta transparencia y una muy buena adhesión entre capas. Entre sus desventajas se tiene la no biodegradabilidad y su leve toxicidad. (Impresoras3D.com, 2018)

- TPE: fabricado con un compuesto formado por plástico y caucho, presenta una muy buena amortiguación ante colisiones, se caracteriza por su suavidad y lo hace útil para la producción de elementos protectores, decoradores y elementos elásticos, aunque con el paso del tiempo tiende a perder su elasticidad. (Impresoras3D.com, 2018)

- HIPS: ofrece una alta resistencia contra impactos, es reciclable y debido a que no desprende gases tóxicos puede ser usado en la fabricación de cubiertos y envases para alimentos, por su composición no es capaz de soportar ambientes a la intemperie. (Impresoras3D.com, 2018)

4.2.4. Piezas 3D del dispositivo de medición

Para el diseño del dispositivo de medición de este monitor energético se ha empleado el modelado 3D con el propósito de crear un *case* o caja para contener y mantener los dispositivos electrónicos resguardados del exterior y así evitar la manipulación directa de los mismos, el diseño de la caja se caracteriza por ser compacto, práctico y aporta al usuario una experiencia agradable al momento de usar y configurar el dispositivo.

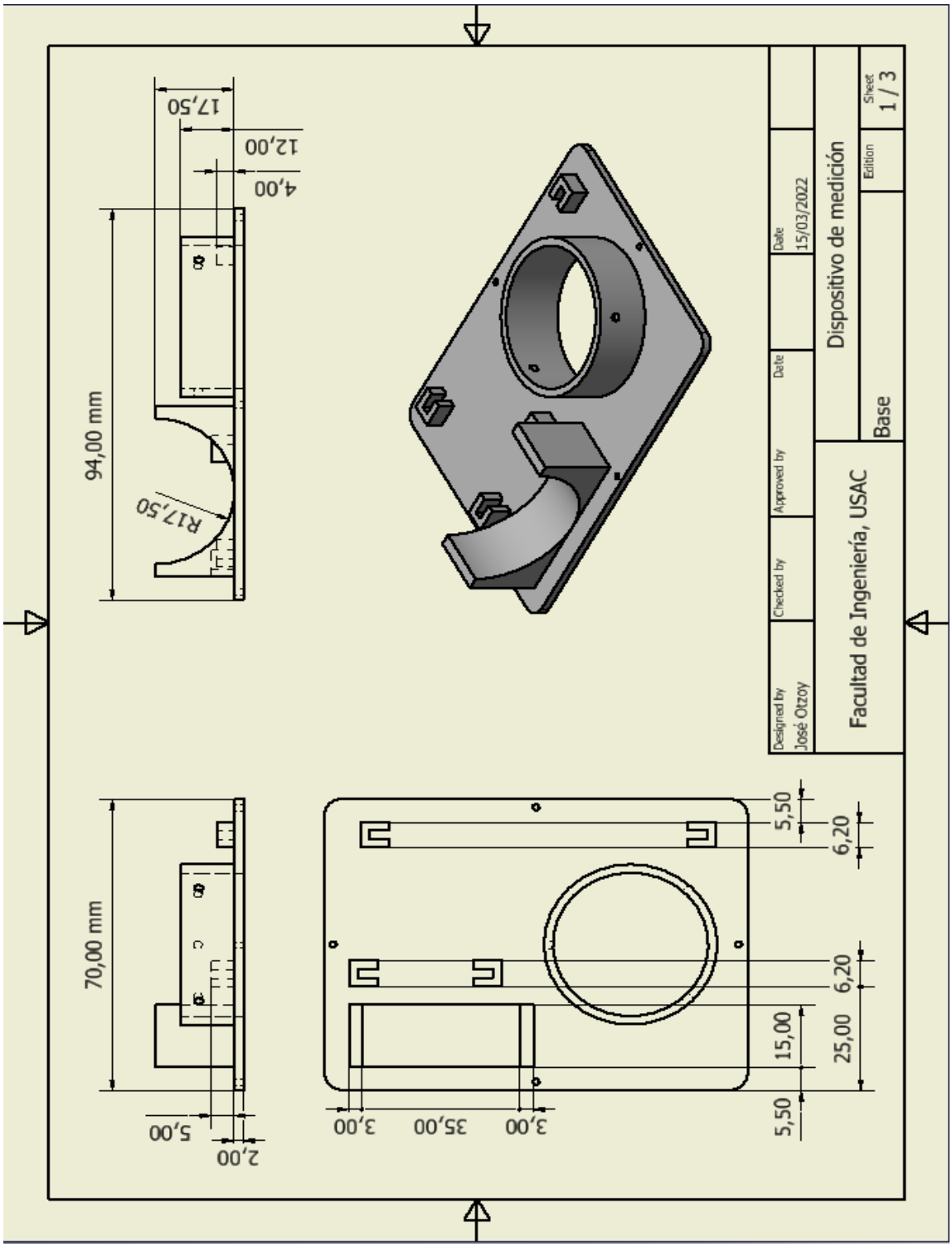
Los modelos empleados están diseñados de tal forma que las piezas puedan ser creadas a partir de impresión 3D, para luego ser ensambladas y aseguradas con los tornillos adecuados, para los dispositivos electrónicos se cuenta con sujetadores sin embargo es recomendable usar un pegamento apropiado al material de impresión para asegurar los componentes, si la impresión 3D se realiza utilizando PLA un pegamento ideal podría ser el de silicona. Las piezas que conforman la caja o *case* para el dispositivo de medición están descritas en la tabla I.

Tabla I. **Piezas requeridas por el dispositivo de medición**

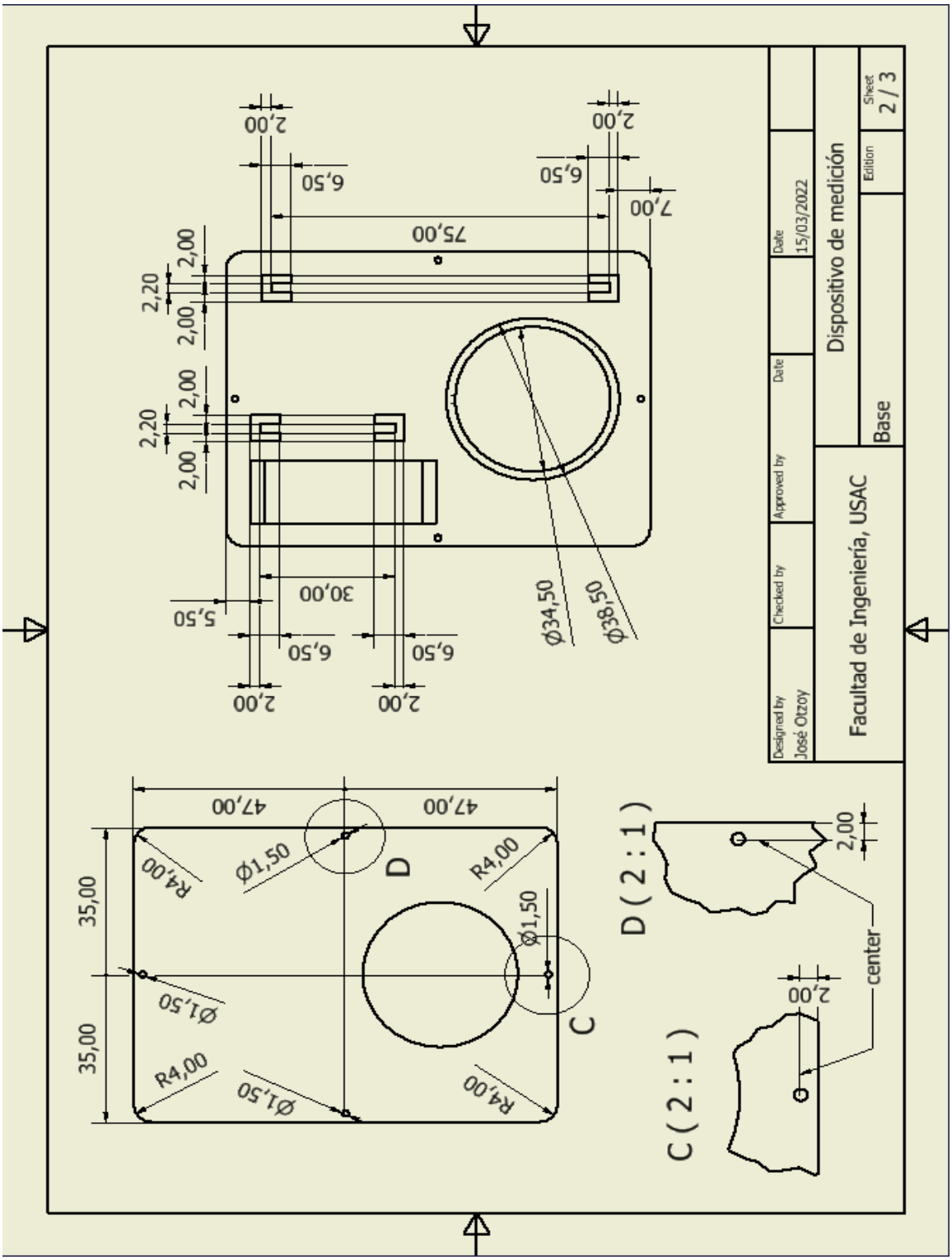
Nombre de la pieza	Cantidad	Descripción
Base	1	Destinada a fijar los componentes electrónicos y eléctricos que conforman el dispositivo de medición, cuenta con una abertura diseñada para introducir la clavija de una espiga marga EAGLE la cual se utiliza para realizar la conexión de corriente eléctrica.
Tapa	1	Situada en la parte superior del dispositivo y cuenta con una abertura alineada de tal forma que permite sobresalir una toma de corriente marca BTICINO MAGIC, que corresponde al enchufe tipo B que permite insertar la clavija del dispositivo que estará bajo monitoreo.
Paredes	1	Sirve para unir la base y la tapa mediante tornillos formando así una superficie cerrada que resguarda los componentes del dispositivo de medición, cuenta con seguros diseñados para fijar el soporte de una placa BTICINO MAGIC 503/3SR, es importante aclarar que para que la placa sea compatible con este diseño deberá contar con una modificación, que consiste en cortar las aberturas correspondientes a los módulos situados a los extremos de la placa dejando así únicamente la abertura del centro.

Fuente: elaboración propia.

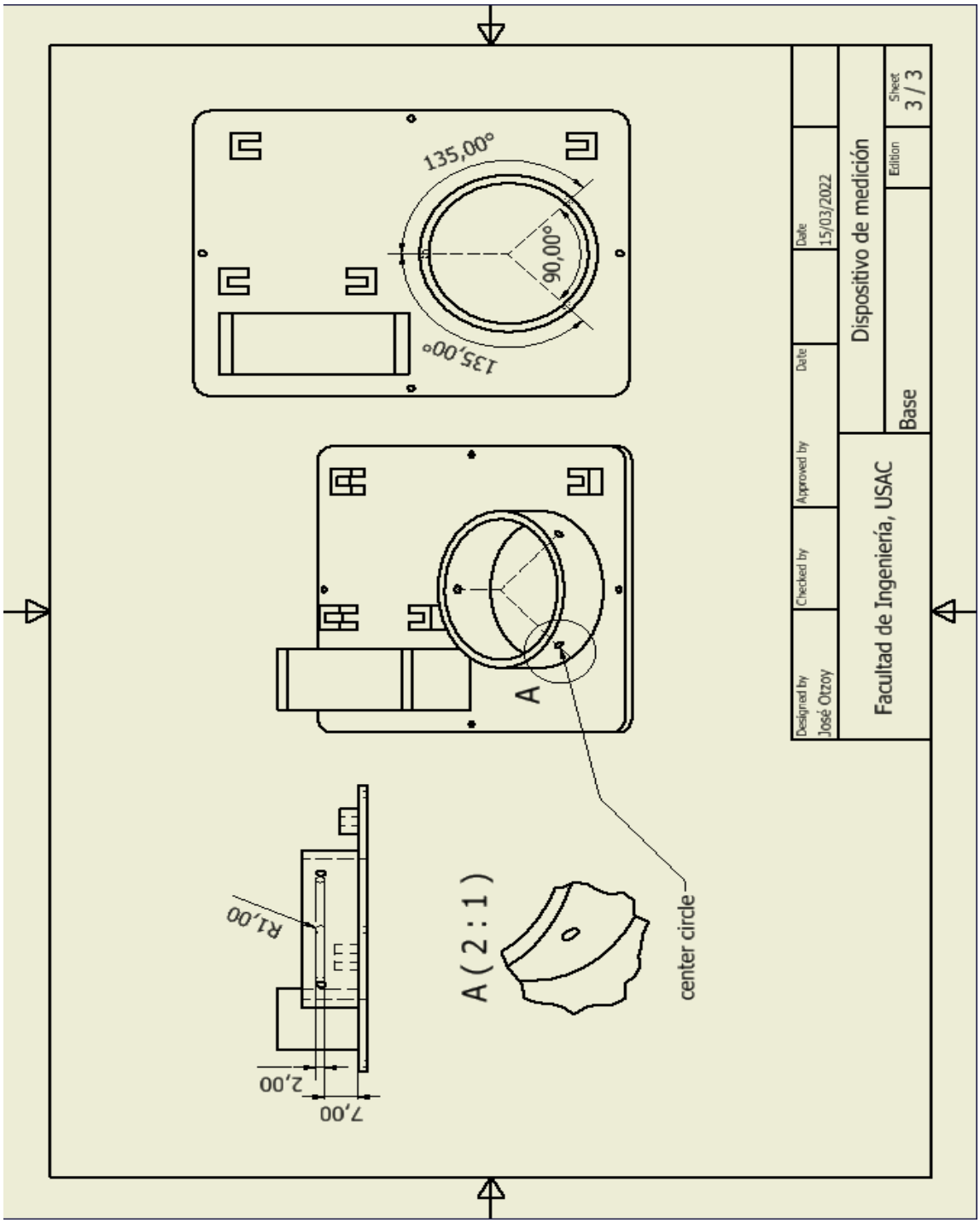
Figura 18. **Base (planos de modelado 3D)**



Continuación de la figura 18.

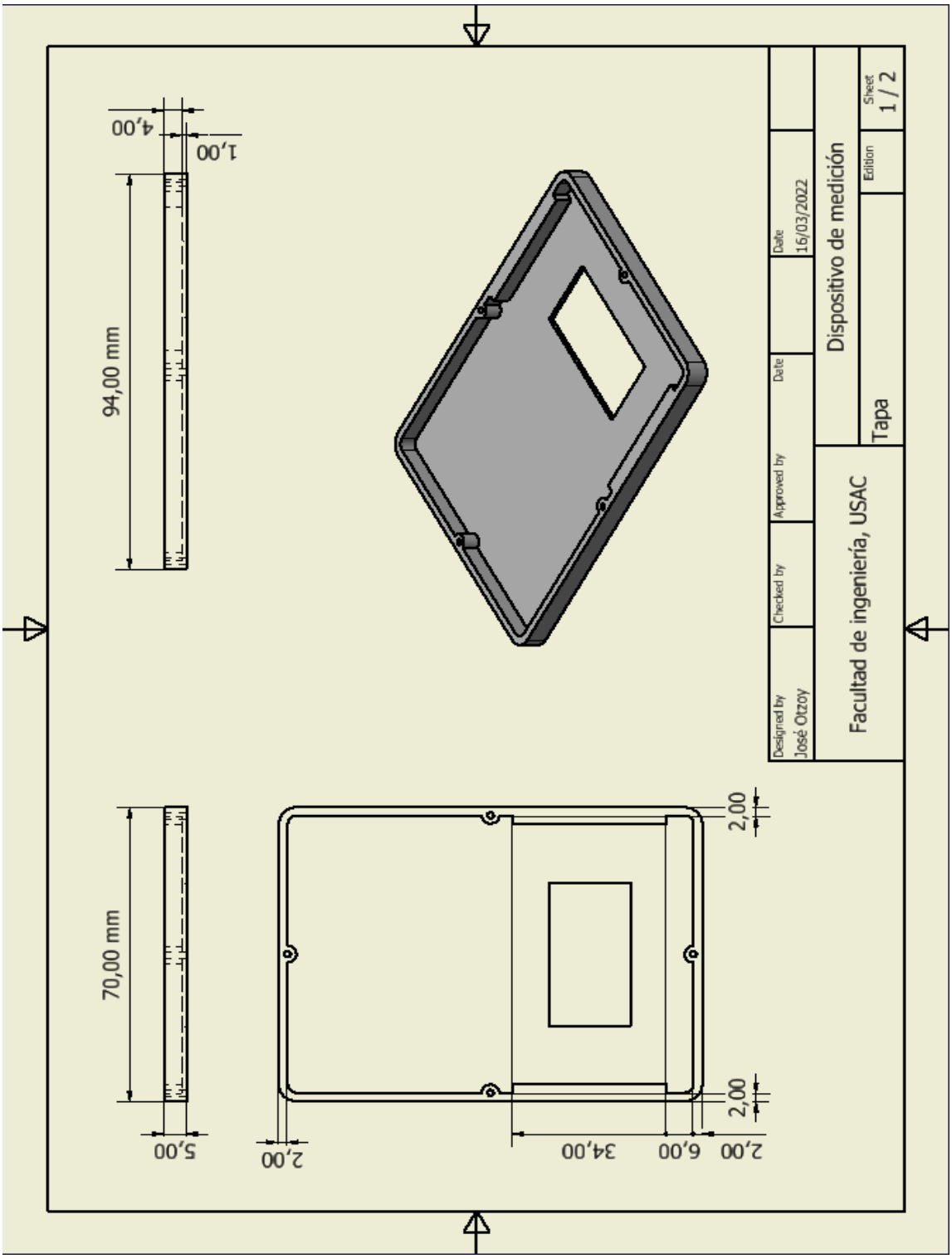


Continuación de la figura 18.

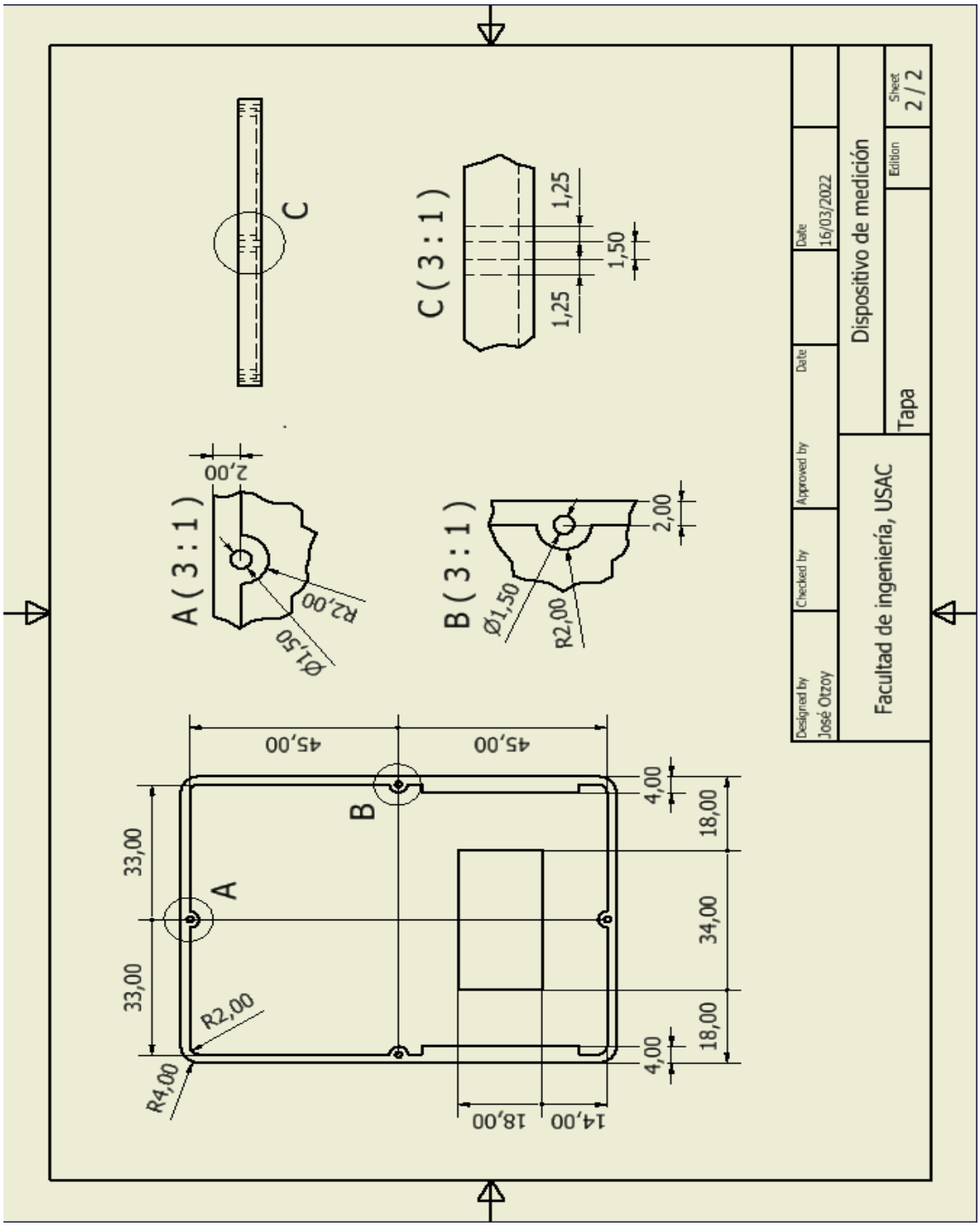


Fuente: elaboración propia, realizado con Autodesk Inventor.

Figura 19. **Tapa (planos de modelado 3D)**

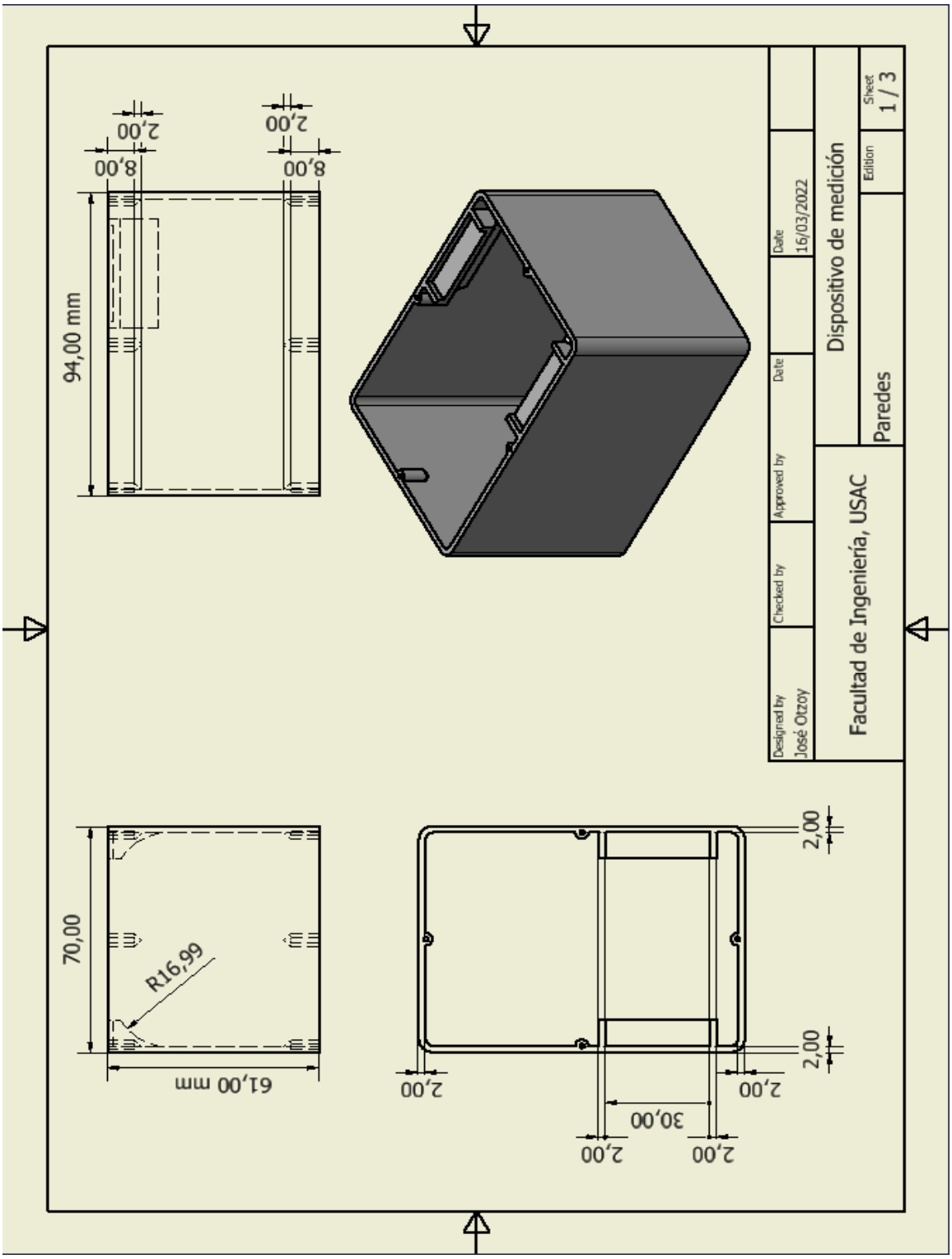


Continuación de la figura 19.

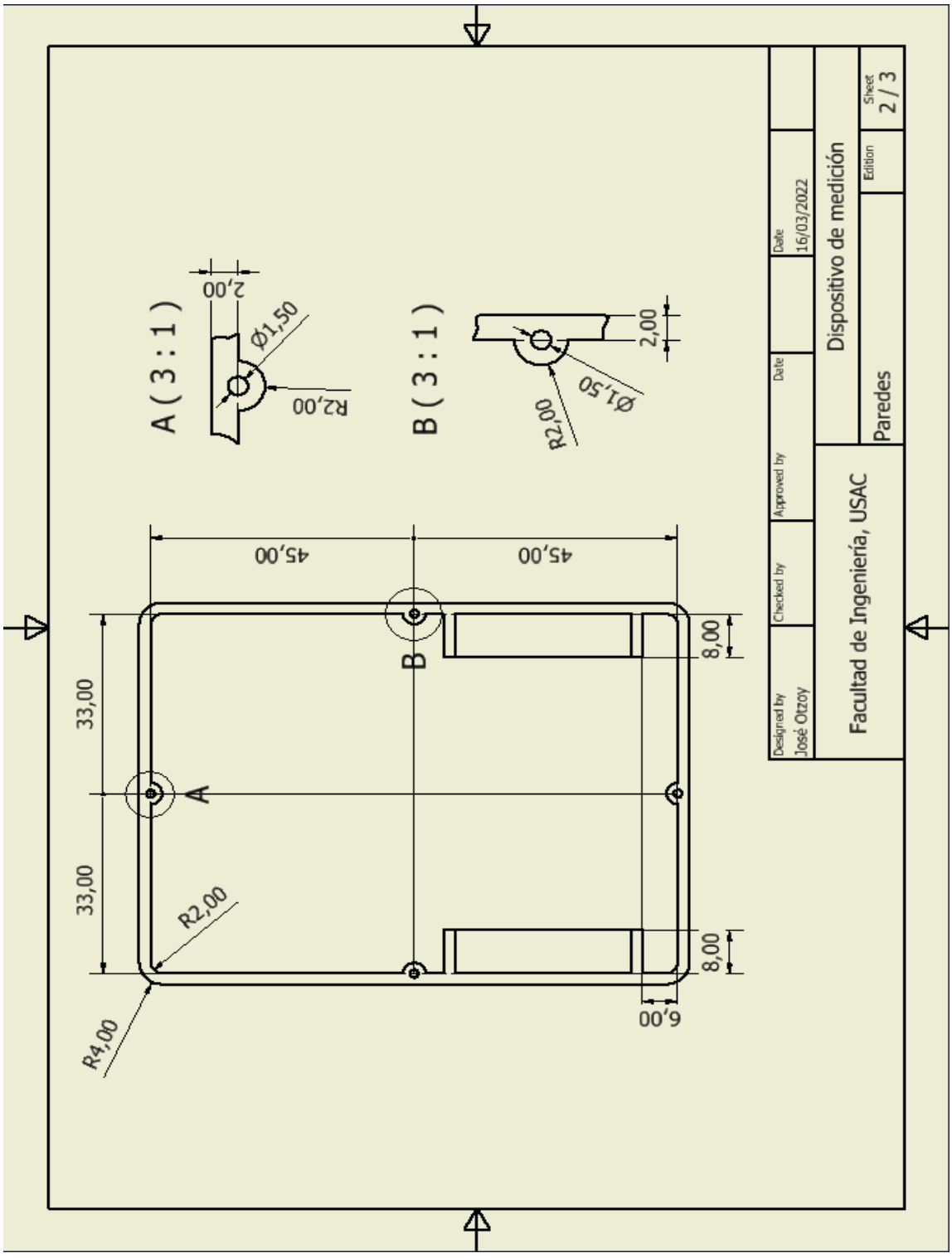


Fuente: elaboración propia, realizado con Autodesk Inventor.

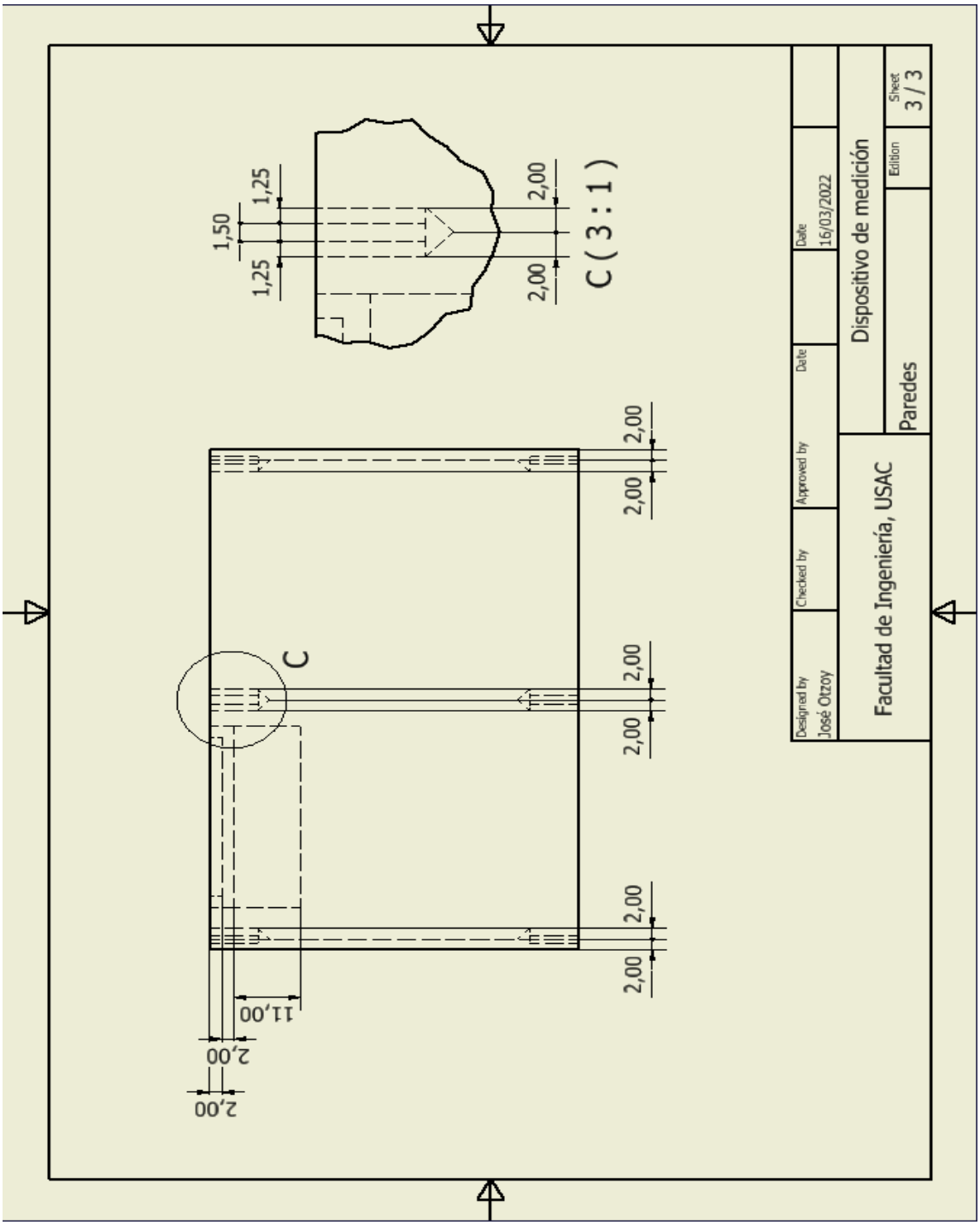
Figura 20. **Paredes (planos de modelado 3D)**



Continuación de la figura 20.



Continuación de la figura 20.



Fuente: elaboración propia, realizado con Autodesk Inventor.

4.3. Componentes electrónicos

Los componentes del dispositivo de medición se han elegido de manera tal que cumpla con todos los requisitos necesarios para trabajar de una forma óptima facilitando así la interacción con el usuario por medio de la configuración vía Bluetooth y la visualización por medio de Wifi.

4.3.1. Clavija tipo B

Este tipo de clavija pertenece al estándar americano NEMA 5-15, principalmente usado en regiones de Norte y Centro América, cuenta con 3 pines de conexión para: línea viva, neutro y tierra, en este estándar el pin de tierra es más largo en comparación a los otros 2, de esta manera se establece una conexión aterrizada a tierra antes de energizar cualquier circuito. Por la región en donde se ha desarrollado este dispositivo se ha elegido la clavija tipo B.

El dispositivo de medición está diseñado para permitir el ensamble de una clavija tipo B de espiga (15 amperios), de la marca EAGLE, aunque cualquier otra clavija que cumpla con las medidas y forma del espacio asignado puede ser usada sin ningún problema no importando la marca de esta.

4.3.2. Enchufe tipo B

A manera de mantener la compatibilidad entre conectores se ha elegido un enchufe tipo B para energizar el dispositivo que estará bajo monitoreo por parte del dispositivo de medición, este tipo de enchufe al igual que la clavija (tipo b) están normados para conducir una corriente máxima de 15 amperios (según norma NEMA 5-15). (Larry, 2020)

El espacio para montar el enchufe fue diseñado a partir de una toma de corriente BTICINO MAGIC 1M 2P+T y un soporte para la placa MAGIC 3M 503/3SR BTICINO, para que el soporte de la placa sea compatible con el diseño debe ser modificado con el propósito de encajar en el espacio asignado, dicha modificación consiste en cortar el espacio para los módulos de los extremos dejando únicamente el espacio para el módulo del medio, por medio de la figura 21 se puede observar en que consiste dicha modificación.

Figura 21. **Modificación soporte BTICINO**

Fuente: elaboración propia, realizado con Paint 3D.

4.3.3. Módulo PZEM-004T

Este módulo cuenta con un sensor energético capaz de realizar distintas mediciones en corriente alterna, cuenta con una interfaz TTL para comunicación

y lectura. La medición se puede llevar a cabo conectando directamente la carga en paralelo a la placa del módulo, pero de esta forma los rangos de medición están limitados, otra forma de realizar la medición es conectando un transformador de corriente (ya incluido en el paquete), a la placa del módulo y haciendo pasar el cable (fase neutra) de alimentación de la carga por el agujero del transformador, de estar forma se logra obtener el máximo rango de medición. Las mediciones que este módulo puede realizar son las siguientes:

- Voltaje: rango de medición de 80 - 260 voltios a una resolución de 0,1 voltios con 0,5 % de exactitud en la medición.

- Corriente: rango de medición de 0 – 100 amperios a una resolución de 0,001 amperios con 0,5 % de exactitud en la medición.

- Potencia activa: rango de medición de 0 – 23 kilowatts a una resolución de 0,1 watt con 0,5 % de exactitud en la medición.

- Factor de potencia: rango de medición 0,00 – 1,00 a una resolución de 0,01 con 1 % de exactitud en la medición.

- Frecuencia: rango de medición de 45 – 65 Hertz a una resolución de 0,1 Hertz con 0,5 % de exactitud en la medición.

- Energía activa: rango de medición de 0 – 9 999,99 kilowatt hora a una resolución de 1 watt hora con 0,5 % de exactitud en la medición.

Si bien el módulo PZEM-004T cuenta con rangos de medición bastante amplios, en un ambiente doméstico no se logran aprovechar todas sus características y como consecuencia de esto la mayoría de los rangos en el

dispositivo de medición se reducen de una manera considerable, también es importante recordar que la corriente soportada por la clavija y enchufe elegidos no puede superar los 15 amperios por lo tanto se limita de una manera considerable el rango de medición de corriente del módulo energético.

El módulo cuenta con una memoria del tipo no volátil utilizada para almacenar el consumo de energía activa, con esto se consigue conservar la medición en caso exista una ausencia en la alimentación principal debida a desconexiones provocadas o interrupciones causada por fallas en la red eléctrica. Estas son las razones por las que el módulo se ha elegido para conformar el dispositivo de medición.

Figura 22. **PZEM-004T**

Fuente: ESPHome. *Peacefair PZEM-004T V3 Energy Monitor*. Consultado el 1 de marzo de 2022. Recuperado de https://esphome.io/components/sensor/pzemac.html.

4.3.4. ESP32

Es un chip SoC desarrollado por Espressif Systems, monta un microprocesador Xtensa LX6 que puede ser de núcleo único o doble, cuenta con un sistema híbrido que incorpora Wifi y Bluetooth en un mismo chip, por sus características de conexión es fuertemente usado en proyectos IoT e incluso en pruebas de ciber seguridad.

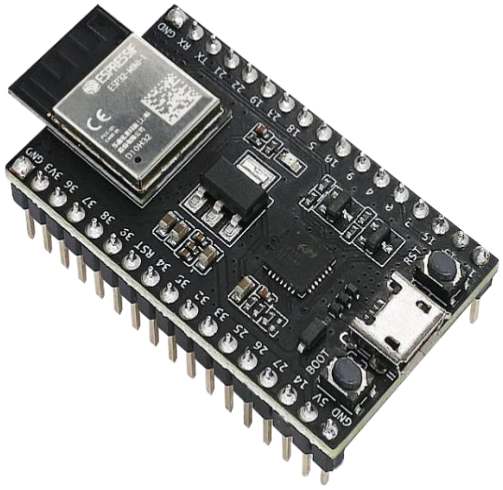
El chip se puede encontrar mondando en un *kit* de desarrollo que incluye componentes extras que permiten la comunicación serial para la carga de archivos de programa desde un computador, también se puede encontrar únicamente el chip para ser soldado en una placa de circuito impreso, uno de los inconvenientes de contar únicamente con el chip es que se debe de contar con un circuito adicional de comunicación serial para cargar archivos de programa a la memoria del chip.

Las rutinas para el procesamiento de información y la comunicación con el servidor remoto (por parte del dispositivo de medición), están desarrolladas a partir de código de programación compatible con el microprocesador de un chip ESP32. Se implementa una lógica de programación a dos hilos, utilizando los dos núcleos que ofrece el microcontrolador.

- El primer núcleo se encarga de leer el módulo energético y manejar las conexiones con el servidor remoto para procesar el envío de datos.

- El segundo núcleo procesa la información proveniente de las configuraciones del dispositivo de medición brindadas por el usuario.

Figura 23. ***Kit* de desarrollo ESP32-DevKitM-1**



Fuente: ESPRESSIF. *ESP32-DevKitM-1*. Consultado el 1 de marzo de 2022. Recuperado de https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/user-guide-devkitm-1.html.

4.3.5. Fuente de alimentación conmutada

Esta fuente es la encargada de suminístrale energía al chip ESP32 y al circuito TTL del módulo PZEM-004T, la fuente se conecta a la toma principal de energía del dispositivo de medición en donde se obtiene corriente alterna la cual se transforma y regula para obtener corriente continua a la salida de la fuente.

Las fuentes conmutadas tienen la característica de ser muy eficientes y compactas, por esa razón se ha elegido este tipo de fuente para alimentar los dispositivos electrónicos.

Las características de la fuente son un factor muy importante para tomar en cuenta pues si se excede la carga soportada por la fuente se podría producir calentamiento en los conductores u otro efecto proveniente de una sobre intensidad.

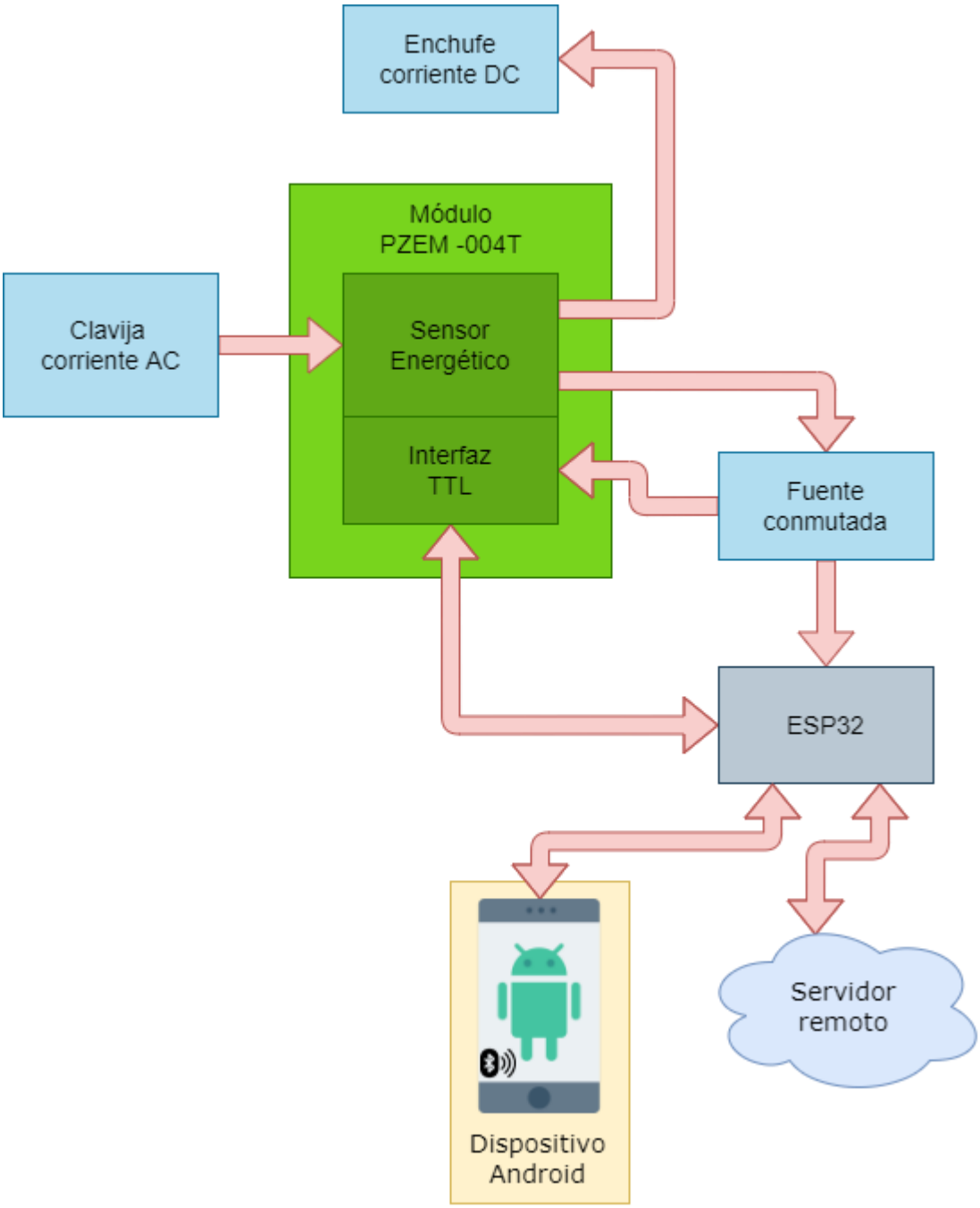
Para este dispositivo de medición se recomienda usar una fuente conmutada que sea capaz de suministrar una corriente de mínimo 500 miliamperios con un voltaje de entre 3.5 a 5 voltios.

4.4. Diagrama de bloques y conexiones

En este apartado se explica de manera gráfica la conexión de los compontes que conforman el dispositivo de medición al igual que los componentes que interactúan con este.

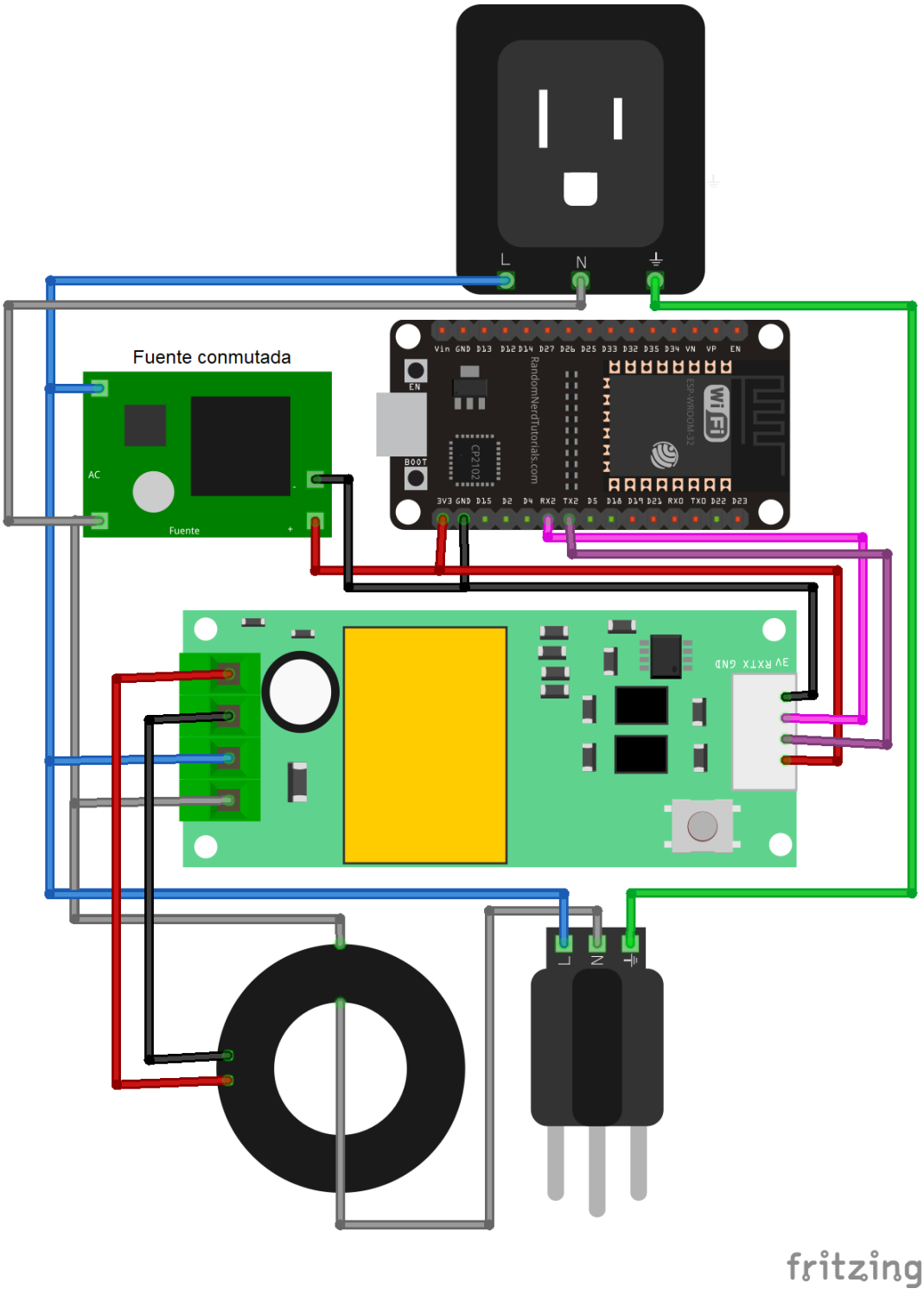
Por medio del diagrama de conexiones (25), se muestra de una forma detallada y explícita el circuito que hace posible la ejecución de las funciones requeridas por el dispositivo, de la misma manera el diagrama de bloques (24), explica el flujo de interacción entre los competentes propios del dispositivo y otros servicios externos.

Figura 24. **Diagrama de bloques dispositivo de medición**

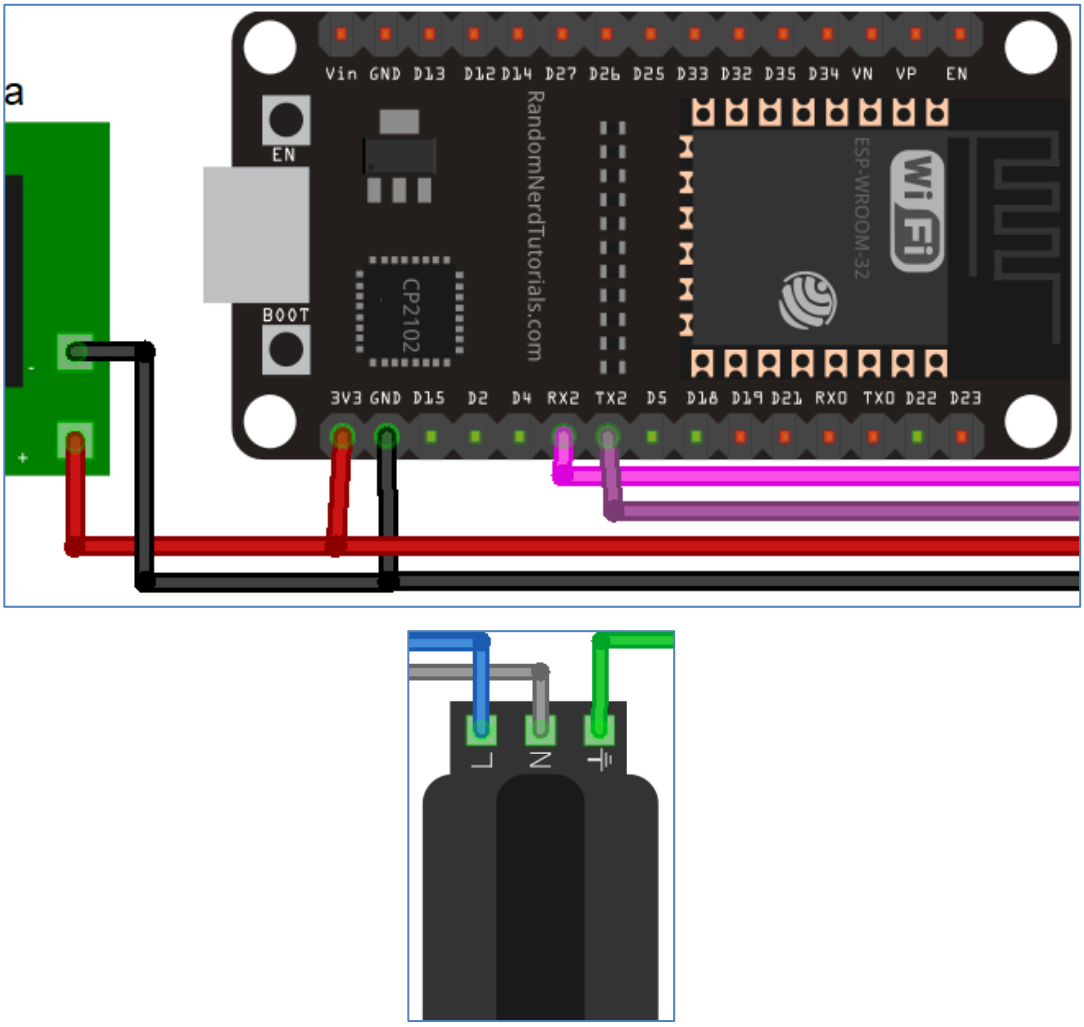


Fuente: elaboración propia, realizado con Draw.io.

Figura 25. **Conexiones dispositivo de medición**



Continuación de la figura 25.



Fuente: elaboración propia, realizado con Fritzing.

4.5. Ensamble de piezas y componentes electrónicos

Por medio de planos ensamble se muestra el acoplamiento de piezas y componentes electrónicos que conforman el dispositivo de medición, la clavija y enchufe descritos en los planos son representados a partir de productos con

algunas modificaciones a fin de eliminar piezas sobrantes y adaptar las piezas funcionales al diseño requerido.

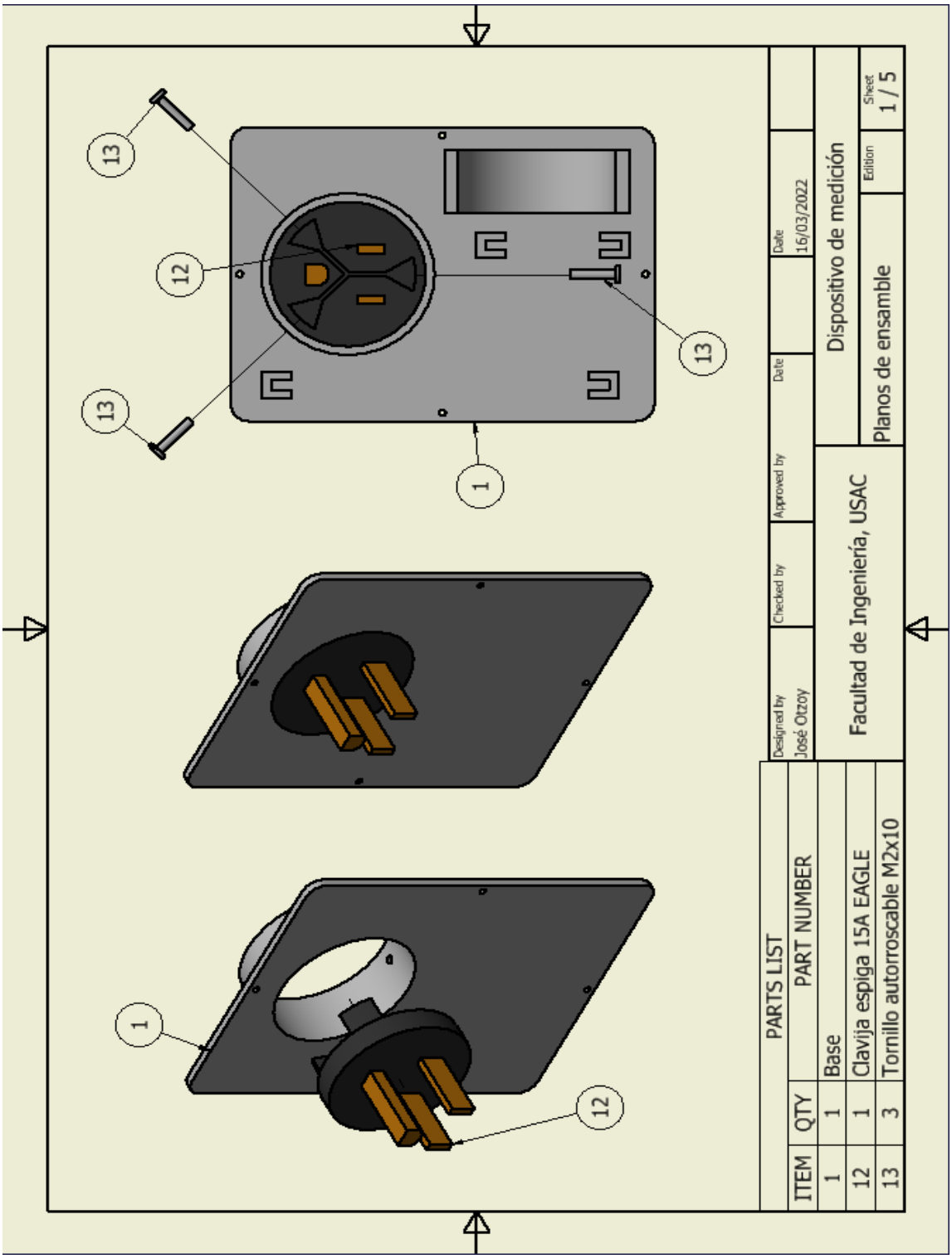
En el caso de los componentes electrónicos se muestra el lugar y la orientación que les corresponde, pero debido a la aleatoriedad del caso no se muestra la ubicación de los diferentes cables de conexión requeridos para la interacción de los compontes por lo que se debe tener en cuenta que los cables deben ser acomodados en ubicaciones que no causen interferencia al sensor energético, un ejemplo de una ubicación que no contribuye ninguna interferencia podría ser la cara interior de las paredes.

La interferencia al sensor energético es debida a conductores que atraviesan el transformador de corriente del módulo PZEM-004T, ya que el único cable que debe atravesar dicho transformador es el conductor (fase neutra) que alimenta los componentes que requieran corriente alterna y el dispositivo al cual se está monitoreando.

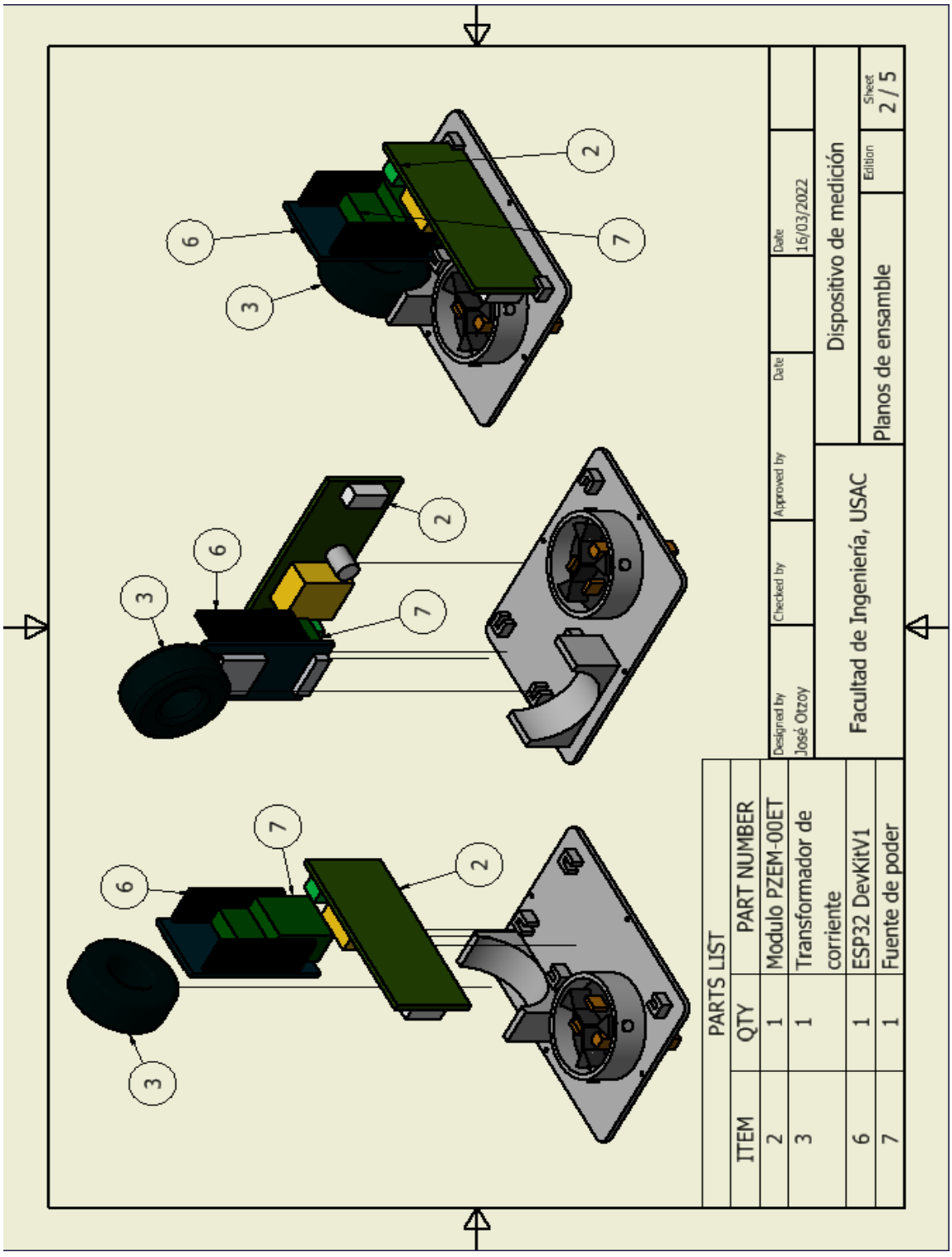
Es importante mencionar que para las conexiones donde circule corriente alterna se debe usar conductores que soporten la corriente máxima permitida por el dispositivo de medición, en este caso 15 amperios, si se utiliza un conductor no apto a las condiciones se podrían ocasionar accidentes debidos a corto circuitos o calentamiento.

La manera en la que se realiza la unión de las piezas impresas en 3D es por medio de tornillos. La pieza “base” cuenta con sujetadores para los componentes electrónicos, pero siempre es recomendable usar pegamento de silicona (o pegamento adecuado a los materiales), a fin de mejorar la adherencia entre los objetos y evitar la movilidad de los componentes dentro del contenedor del dispositivo de medición.

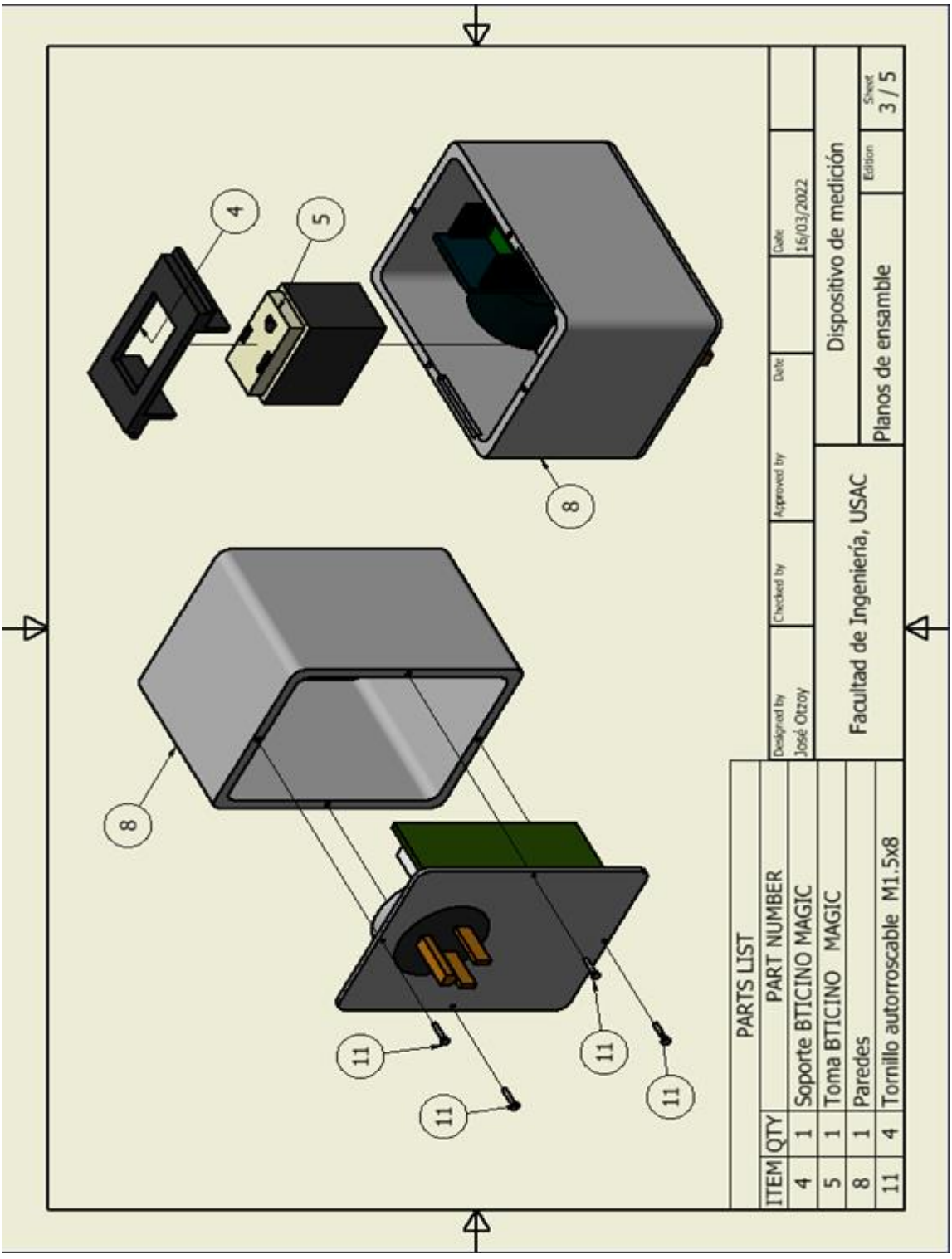
Figura 26. **Planos de ensamble dispositivo de medición**



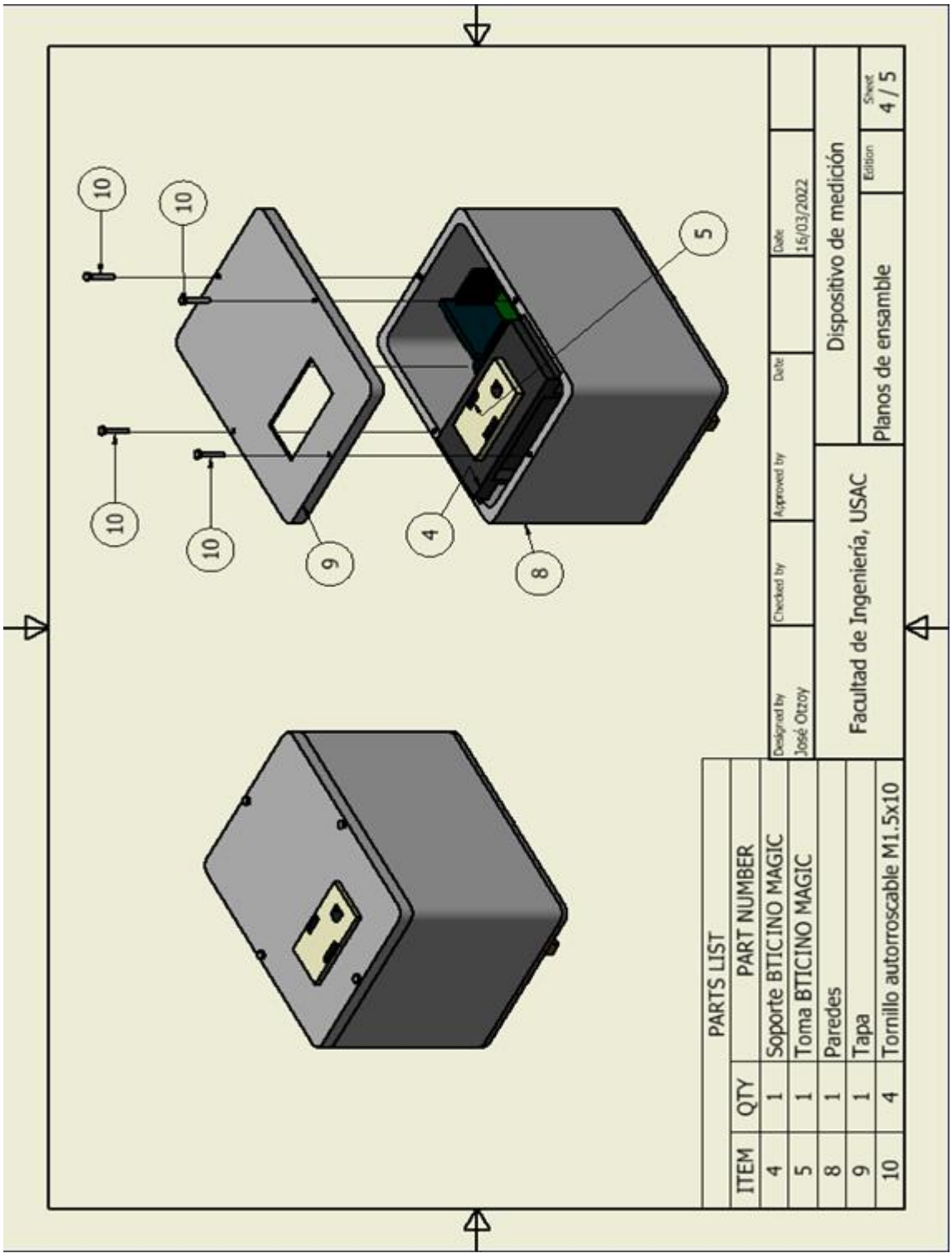
Continuación de la figura 26.



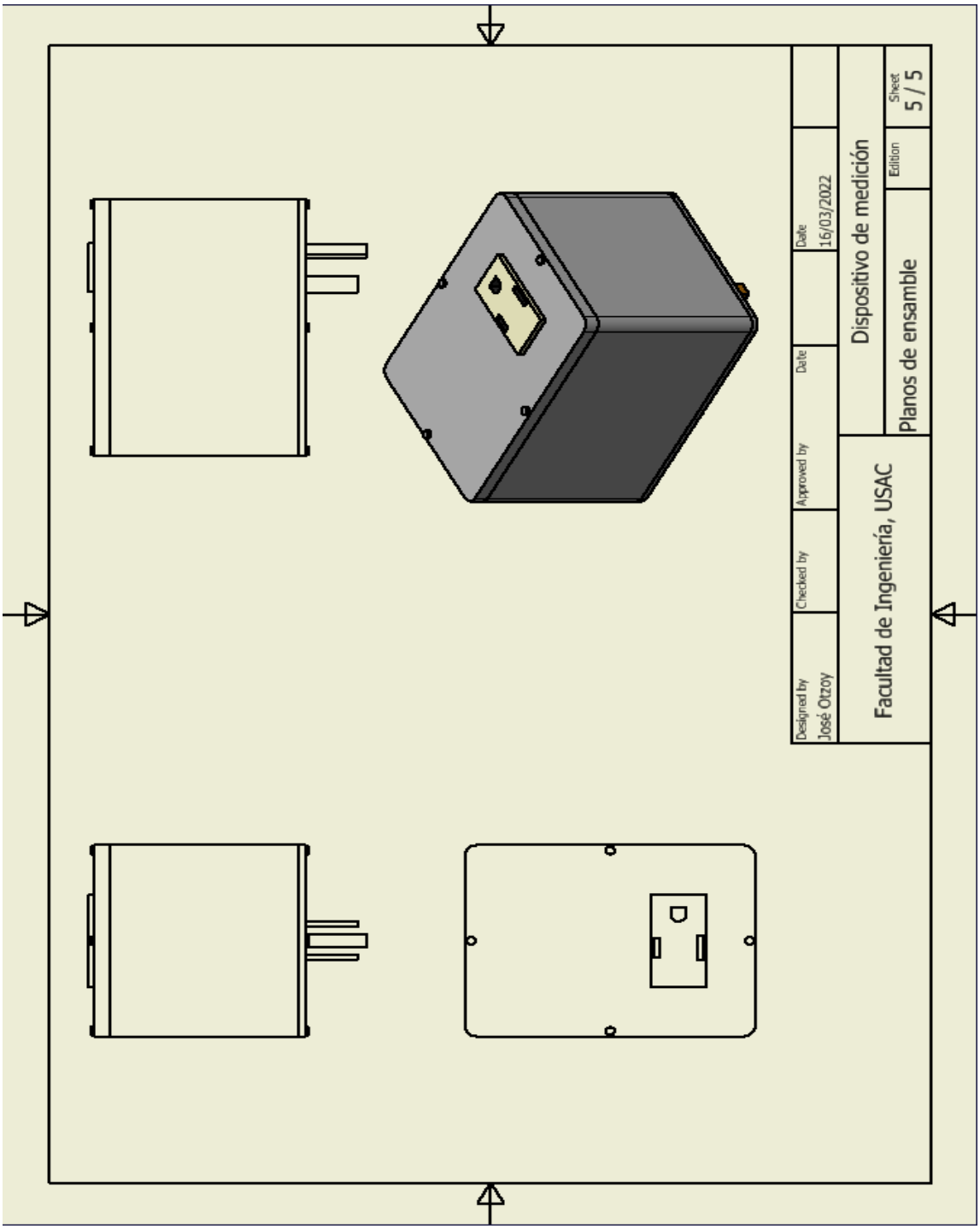
Continuación de la figura 26.



Continuación de la figura 26.



Continuación de la figura 26.



Fuente: elaboración propia, realizado con Autodesk Inventor.

4.6. Programación ESP32

El chip SoC ESP32 es el encargado de controlar los sensores y ejecutar la lógica de funcionamiento del dispositivo de medición realizando el procesamiento de datos de manera local y remota, de manera análoga se le podría comparar con un cerebro debido a que es el encargado de coordinar todas las acciones del sistema por medio de un algoritmo de programación que define la rutina de funcionamiento.

Con el propósito de indicarle al chip cuales son las tareas que debe de realizar por medio del microcontrolador es necesario tener un código de programa en donde se describa el algoritmo a ejecutar, el código se desarrolla por medio de un IDE que permita transformar el código de programa a un archivo con instrucciones legibles para el microcontrolador, las instrucciones se cargan al chip utilizando comunicación serial y se almacenan en la memoria flash para su persistencia.

4.6.1. Software de programación

Comúnmente denominado IDE, es un entorno de desarrollo que cuenta con herramientas tales como: editor de código, depurador, compilador, entre otras. (RedHat, 2019) Los IDE más usados para el desarrollo de código compatible con el chip ESP32 son Arduino IDE y los basados en el intérprete MicroPython, aunque existen otras opciones que implementan librerías de Arduino que permiten usar su sintaxis para el desarrollo en un IDE diferente, también existen los IDEs que permiten el desarrollo por medio del *framework* oficial para productos ESPRESSIF denominado IoT Development Framework, ejemplo de esto es el complemento para IDEs llamado PlatformIO que permite el

desarrollo de código para chips ESP32 por medio de IoT Development Framework y Arduino.

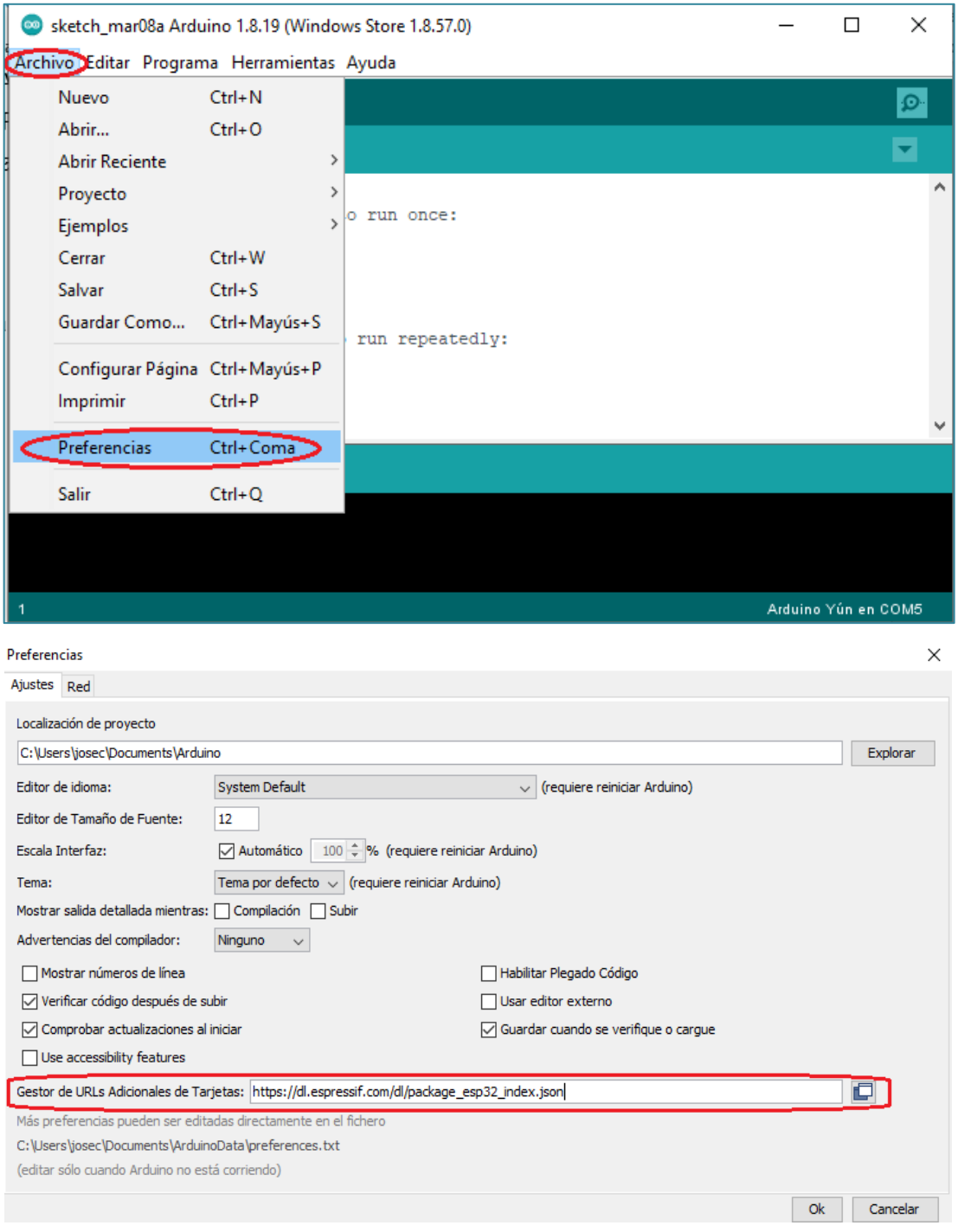
4.6.1.1. Arduino IDE

Es el software de programación oficial para placas Arduino, también es un IDE de código abierto basado en lenguaje C++ que cuenta con una gran comunidad de usuarios que aportan librerías y recursos de programación. (Lozano R, 2021) Por todo ello es uno de los IDE con mejor soporte y documentación, gracias a las numerosas librerías con las que cuenta es posible desarrollar código compatible con otras placas ajenas a Arduino como fabricante lo que permite agregar al gestor de tarjetas de Arduino placas basadas en el chip ESP32. En el sitio web de Arduino (www.arduino.cc), se encuentra habilitada la descarga del IDE de manera gratuita.

Pasos para agregar placas ESP32 al entorno Arduino IDE:

- Paso 1: agregar los repositorios ESP32 al gestor de URLs adicionales de tarjetas. Para hacer este procedimiento es necesario navegar al menú de ajustes a través de Archivo > Preferencias > Ajustes, luego pegar la dirección https://dl.espressif.com/dl/package_esp32_index.json en el apartado Gestor de URLs Adicionales de Tarjetas.

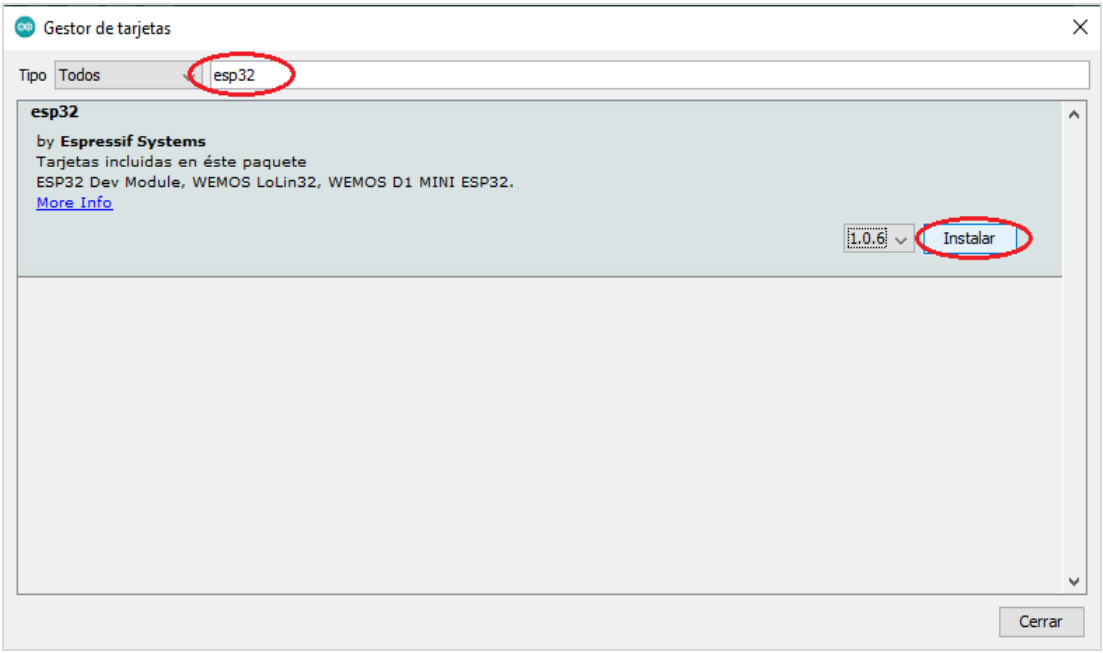
Figura 27. **Agregar repositorios ESP32 en Arduino IDE**



Fuente: elaboración propia, realizado con Paint3D.

- Paso 2: descargar y cargar las dependencias y librerías necesarias usando el gestor de tarjetas. En este paso se descargan y agregan las placas ESP32 a Arduino IDE navegando al menú del gestor de tarjetas por medio de Herramientas > Placa > Gestor de tarjetas, luego en el gestor de tarjetas buscar e instalar el paquete llamado esp32.

Figura 28. **ESP32 en gestor de tarjetas Arduino IDE**

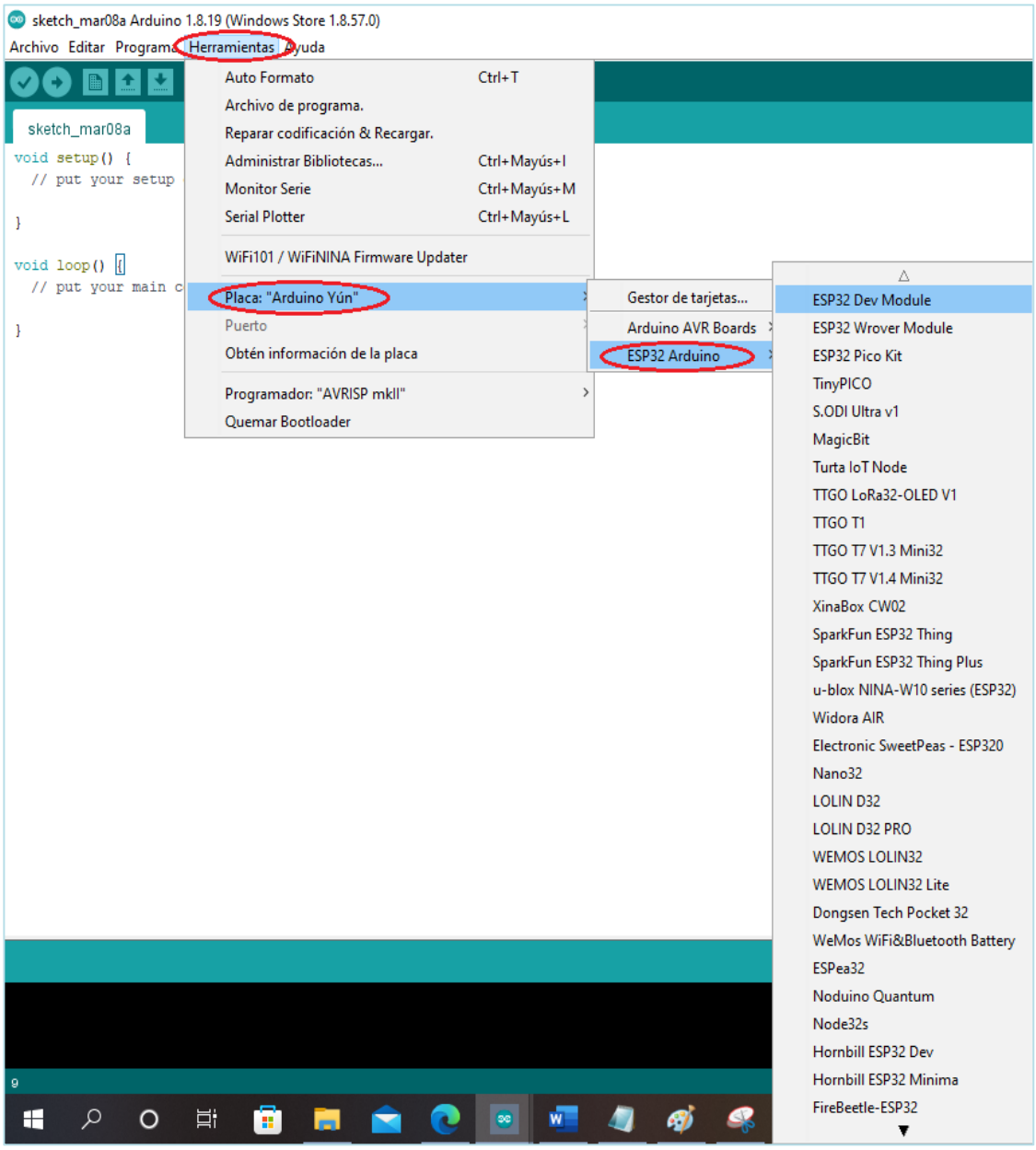


Fuente: elaboración propia, realizado con Paint3D.

- Paso 3: seleccionar la tarjeta para la cual se va a desarrollar. Para mostrar las tarjetas disponibles se debe ir al menú correspondiente navegando por: Herramientas > Placa > ESP32 Arduino > Seleccionar la placa correspondiente. Después de haber seleccionado la placa ya se podrá desarrollar el código correspondiente al algoritmo de interés y también se podrán aplicar diferentes configuraciones de hardware, por ejemplo: seleccionar el tipo de esquema de partición, asignar un tamaño y modo de

funcionamiento para la memoria *flash*, seleccionar la frecuencia de CPU, entre otros.

Figura 29. **Selección de placa ESP32 Arduino IDE**



Fuente: elaboración propia, realizado con Paint3D.

4.6.1.2. IDEs basados en MicroPython

MicroPython es un intérprete de código que ejecuta una versión de Python 3 diseñada para microcontroladores y sistemas embebidos con recursos limitados, es compatible con numerosas arquitecturas basadas en ARM por lo que es una de las soluciones más usadas para proyectos IoT en conjunto con *kits* de desarrollo basados en ESP32, para el caso de los chips ESP32 antes de poder desarrollar código se debe de borrar la memoria flash y posteriormente cargar el *firmware* correspondiente para el desarrollo con el lenguaje Python, dado que de fábrica el chip no incorpora las herramientas para permitir este tipo de acciones.

Los IDEs que implementan el intérprete MicroPython se caracterizan por contar con una terminal que permite enviarle instrucciones al microcontrolador sin necesidad de cargar ningún archivo, función que es muy útil para la ejecución de pruebas, la terminal también muestra el estando del microcontrolador facilitando así la depuración de errores. Los IDEs más conocidos que utilizan MicroPython son:

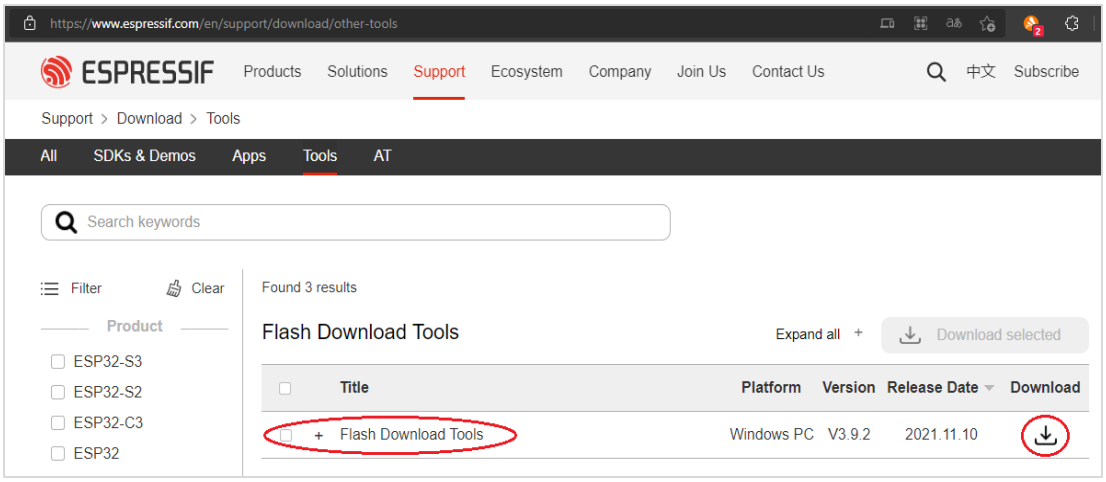
- Mu Editor
- uPyCraft IDE
- Thonny IDE
- VS Code + Pymakr
- PyCharm

Pasos para cargar el *firmware* de MicroPython en tarjetas ESP32:

- Paso 1: descargar la herramienta denominada Flash Download Tools proporcionada por ESPRESSIF por medio del siguiente enlace:

    - www.espressif.com/en/support/download/other-tools

Figura 30. **Descarga herramienta Flash Download Tools de ESPRESSIF**

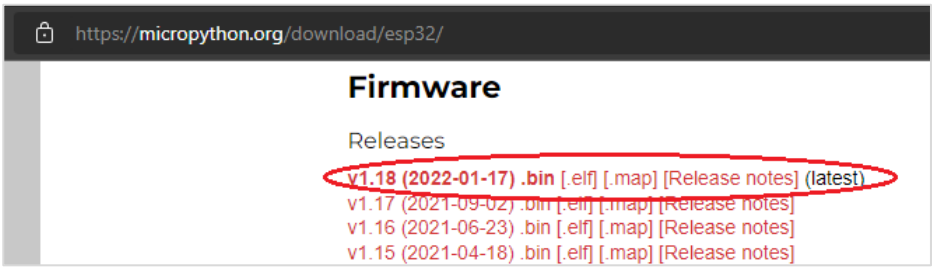


Fuente: elaboración propia, realizado con Paint3D.

- Paso 2: descargar el archivo binario correspondiente al *firmware* de MicroPython por medio del sitio oficial (se recomienda descargar la última versión estable) con el siguiente enlace:

    - https://micropython.org/download/esp32/

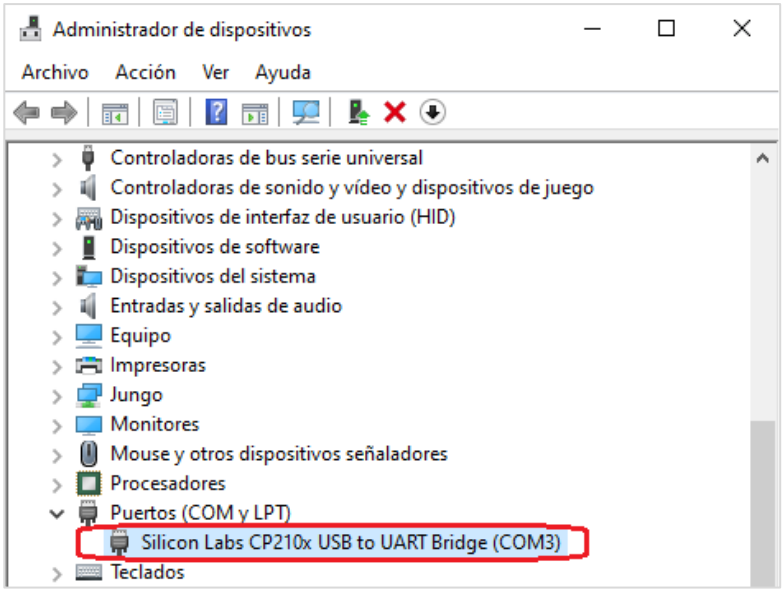
Figura 31. **Descarga *firmware* oficial MicroPython**



Fuente: elaboración propia, realizado con Paint3D.

- Paso 3: Conectar la tarjeta ESP32 al computador y verificar el puerto COM asignado, para visualizar los puertos COM en Windows es necesario abrir el administrador de dispositivos y verificar el puerto asignado al dispositivo de interés.

Figura 32. **Puertos COM Administrador de dispositivos Windows**

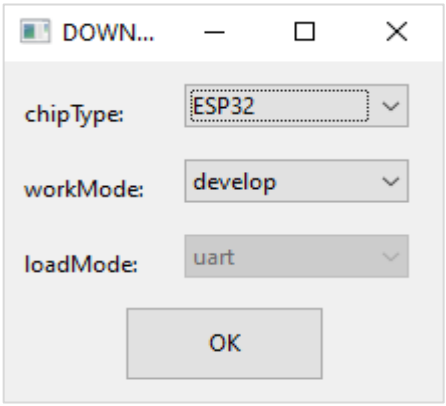


Fuente: elaboración propia, realizado con Paint3D.

- Paso 4: abrir la herramienta proporcionada por ESPRESSIF para la carga de firmware, rellenar los campos solicitados y hacer clic en el botón OK, los campos deben ser rellenados de la siguiente manera:

    - chipType: ESP32
    - workMode: develop
    - loadMode: uart

Figura 33. **Herramienta Flash Download Tools de ESPRESSIF**



Fuente: elaboración propia, realizado con captura de pantalla

- Paso 5: cargar el archivo binario correspondiente al *firmware* de interés por medio de la pestaña SPIDownload, asignar la dirección de memoria “0x1 000” y seleccionar la línea correspondiente al archivo binario.

- Paso 6: rellenar los campos solicitados por el apartado SpiFlashConfig, de la siguiente manera:

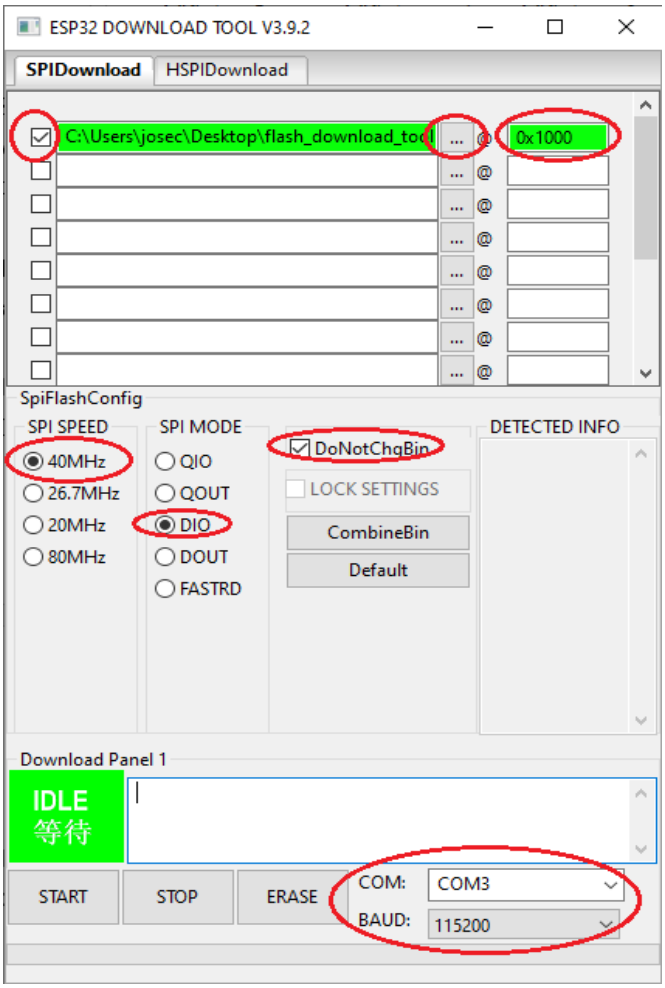
    - SPI SPEED: 40 MHz
    - SPI MODE: DIO

- Seleccionar DoNotChBin

- Paso 7: rellenar los campos del apartado Download Panel 1 de la siguiente manera:

  - COM: puerto COM asignado al dispositivo ESP32
  - BAUD: 115200

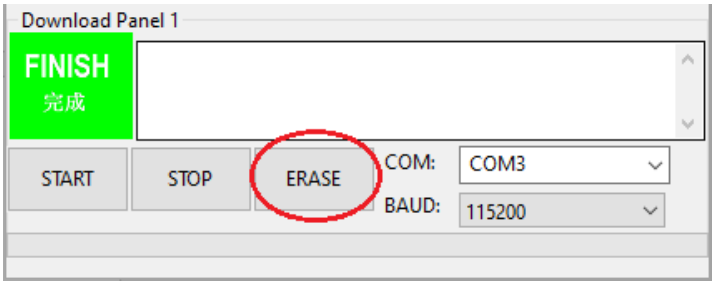
Figura 34. **Configuración Flash Download Tools ESP32**



Fuente: elaboración propia, realizado con Paint3D.

- Paso 8: borrar la memoria *flash* haciendo clic en el botón ERRASE y esperar hasta que el cuadro verde del apartado Download Panel 1 muestre la palabra FINISH.

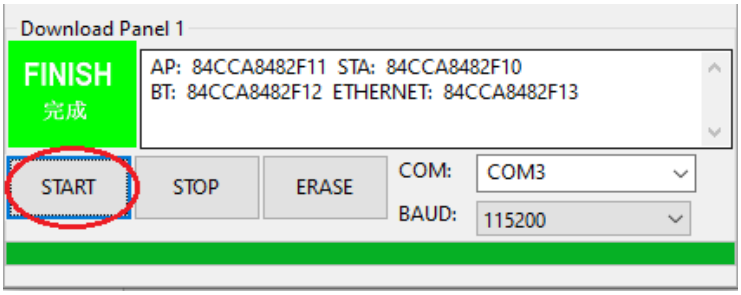
Figura 35. **Memoria flash borrada con éxito ESP32**



Fuente: elaboración propia, realizado con Paint3D.

- Paso 9: cargar el firmware haciendo clic en el botón START, esto hará que se inicie el proceso de carga y se mostrara la palabra *Download* en el cuadro verde del apartado Download Panel 1, es necesario esperar hasta que el cuadro verde indique que el proceso ha terminado por medio de la palabra *FINISH*.

Figura 36. ***Firmware* cargado con éxito ESP32**



Fuente: elaboración propia, realizado con Paint3D.

Si todos los pasos se ejecutaron con éxito la placa ESP32 ya será capaz de ejecutar algoritmos desarrollados por medio de cualquier IDE que soporte la implementación del interprete MicroPython.

4.6.1.3. PlatformIO

Es una multiplataforma diseñada para programar sistemas embebidos y microcontroladores en un entorno profesional, debido a que soporta varias arquitecturas de hardware es compatible con más de 800 tarjetas de desarrollo. PlatformIO se ofrece como una extensión o complemento para entornos de programación profesional tales como: Atom, Eclipse, NetBeans, SublimeText, Visual Studio Code, entre otros. (PlatformIO, 2014)

A continuación, se muestran los pasos para programar un dispositivo ESP32 usando PlatformIO y Visual Studio Code previamente instalado:

- Paso 1: buscar e instalar la extensión PlatformIO IDE por medio de la tienda de extensiones de Visual Studio Code.
    - Paso 1.1: abrir la tienda de extensiones
    - Paso 1.2: escribir “PlatformIO en el campo de búsqueda
    - Paso 1.3: identificar la extensión anteriormente escrita e instalarla por medio del botón Install.
- Paso 2: abrir el menú inicial por medio del botón PlatformIO Home ubicado en la barra inferior.
- Paso 3: crear un nuevo proyecto rellenando los siguientes campos:

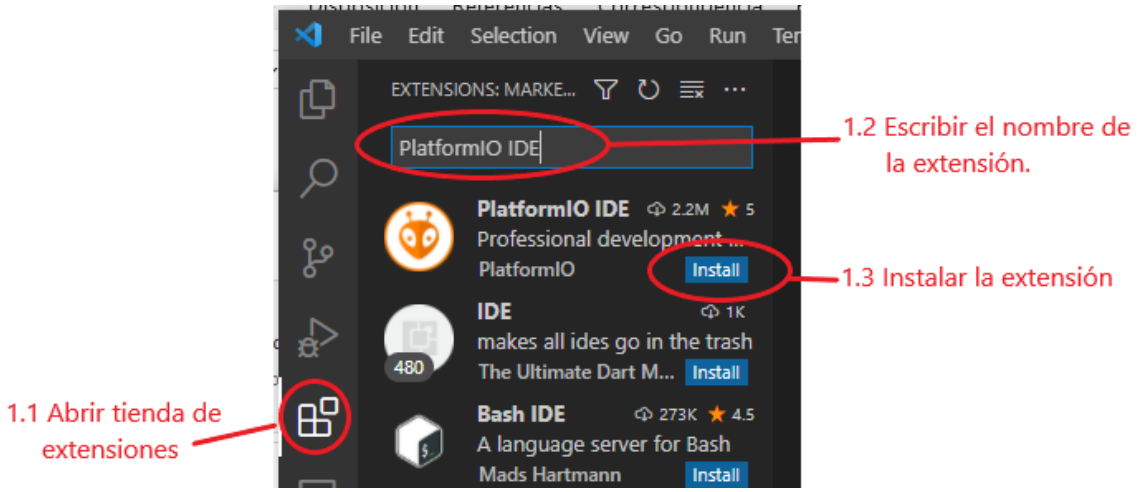
- Name: nombre del proyecto

- Board: en este campo se debe de seleccionar la tarjeta de desarrollo con la que se cuente.

- Framework: para las tarjetas basadas en el chip ESP32 se cuenta con los *frameworks* Arduino y ESPRESSIF IoT Development *Framework*.

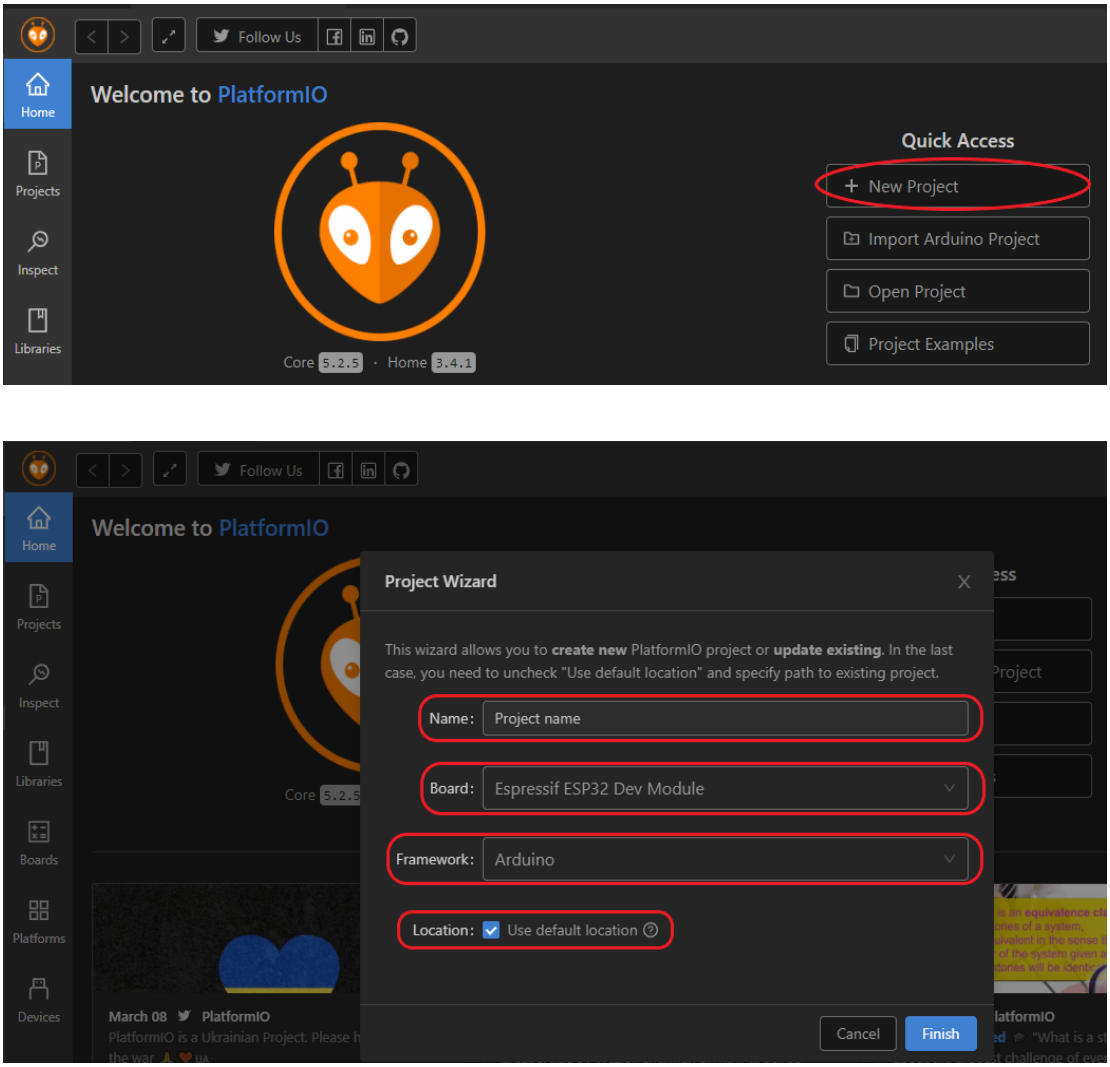
- Location: permite seleccionar la ubicación en donde se guardará el proyecto.

Figura 37. **Instalación PlatformIO en VSCode**



Fuente: elaboración propia, realizado con Paint3D.

Figura 38. **Crear nuevo proyecto PlatformIO**



Fuente: elaboración propia, realizado con Paint3D.

4.6.2. Código de programación dispositivo de medición

El desarrollo del algoritmo de funcionamiento se realizó utilizando código propio, pero también el proporcionado por las diferentes librerías desarrolladas por la comunidad de usuarios y las ya incluidas por defecto en el *framework*

Arduino, dichas librerías se encuentran descritas por su importación en el encabezado del archivo fuente (figura 40).

La lógica del dispositivo de medición funciona a partir de 2 archivos, el primero de ellos contiene el código necesario para almacenar de manera persistente credenciales e identificadores, este se carga al chip ESP32 una sola vez, luego se elimina y se carga el archivo principal que contiene el código fuente del programa, esto se hace como medida de seguridad al evitar mostrar información sensible por medio de la escritura en código y en su lugar esta se obtiene leyendo los datos (credenciales e identificadores), anteriormente escritos en la memoria del chip.

Cada dispositivo de medición cuenta con un identificador único que está formado por la palabra ENER-ESP32- seguido de un número de 3 dígitos que representa el número de serie, si bien este identificador no es un dato sensible se almacena en la memoria del chip debido a que es un dato que nunca cambiará, y sirve para facilitar la preparación de nuevos dispositivos de medición debido a que de esta manera se evita modificar el código fuente cada vez que se tenga que preparar un nuevo dispositivo de medición adoptando así un archivo estándar para la producción.

El desarrollo y depuración del código de programación se llevó a cabo utilizando el entorno profesional conformado por Visial Studio Code y la extensión PlatformIO IDE usando Arduino como *framework*.

Figura 39. **Código archivo para guardar la configuración inicial en un nuevo dispositivo de medición**

```
#include <Arduino.h>
#include <Preferences.h>
Preferences preferences;

void setup() {
  //save mqtt credentials
  preferences.begin("mqtt_cred", false);
  preferences.putString("mqtt_server","xxx.xxx.xxx.xxx"); //ipV4 dir
  preferences.putInt("mqtt_port",1883);
  preferences.putString("mqtt_user","user"); //user
  preferences.putString("mqtt_pass","pass");  //pass
  preferences.end();
  //save info device
  preferences.begin("info_device", false);
  //measurement device name
  preferences.putString("device","ENER-ESP32-###");
  preferences.end();
}

void loop() {

}
```

Fuente: elaboración propia, realizado con captura de pantalla.

Figura 40. **Código archivo principal dispositivo de medición**

```
#include <Arduino.h>
#include <PZEM004Tv30.h>
#include "BluetoothSerial.h"
#include <Preferences.h>
#include <PubSubClient.h>
#include <WiFi.h>

#define PZEM_RX_PIN 16
#define PZEM_TX_PIN 17
#define PZEM_SERIAL Serial2

PZEM004Tv30 pzem(PZEM_SERIAL, PZEM_RX_PIN, PZEM_TX_PIN);
BluetoothSerial SerialBT;
WiFiClient ENER_ESP32_CLIENT;
Preferences preferences;
PubSubClient MqttClient(ENER_ESP32_CLIENT); //mqtt MqttClient
TaskHandle_t TaskCore1;  //task for core 1

int one_second= 1000;
unsigned long now = 0;
int vis_count = 60;

String device, device_topic, meas_topics[5];
const String root_topic = "energy/devices";
const char *update_topic = "energy/update";

void core1(void *parameter);
void setup_wifi(String WiFi_SSID, String WiFi_PASS, boolean new_cred);
void connectMqtt();
void callback(char* topic, byte* payload, unsigned int length);
void ask_wifi_credentials();
void save_conf(String name_space, String key, String value);
void set_update_server(bool updat);
void send_info_server(String dest);
void get_device();

void setup() {
  get_device();

  //Create a new task and assign to core 1
```

Continuación de la figura 40.

```
41    xTaskCreatePinnedToCore(
42    core1,
43    "Task_1",
44    3000,
45    NULL,
46    1,
47    &TaskCore1,
48    1);
49
50    Serial.begin(115200);
51    SerialBT.begin(device); //Bluetooth device name
52    ask_wifi_credentials();
53  }
54
55  void loop() {
56    if (WiFi.status() == WL_CONNECTED && !MqttClient.connected()) {
57      //if there is not conexion with mqtt server try to reconnect
58      connectMqtt();
59    }
60
61    //send data only if there is any android device listening
62    if (vis_count < 60 && millis() >= now+one_second) {
63      now = millis();
64      send_info_server("energy_measu");
65      //wait for 60 seconds before to stop sending data to visualization
66      //in case there is any android divece listening
67      vis_count++;
68    }
69
70    MqttClient.loop();
71  }
72
73  void core1(void *parameter){       //Paralel task in core 1
74    String WiFi_SSID, WiFi_PASS, rate, currency,
75           payday, name, timezone, key;
76    char bt_payload;
77    while (true) {
78      while (SerialBT.available()){
79        bt_payload = char(SerialBT.read());
80        if (bt_payload == '\n'){
```

Continuación de la figura 40.

```
81          if (key=="pass:"){
82            //Connect to Internet
83            setup_wifi(WiFi_SSID, WiFi_PASS, true);
84            WiFi_SSID=""; WiFi_PASS="";
85          }else if (key=="timezone:"){
86            //save energy configuration
87            save_conf("energy_info","rate",rate);
88            save_conf("energy_info","curre",currency);
89            save_conf("energy_info","payday",payday);
90            save_conf("energy_info","name",name);
91            save_conf("energy_info","timezone",timezone);
92            SerialBT.println("ok");
93            rate=""; currency=""; payday=""; name=""; timezone="";
94            //put a request to update server
95            set_update_server(0);
96          }
97          key="";
98        } else if(bt_payload == '?'){
99          //Android device is consulting energy information saved
100         send_info_server("android_config");
101         key="";
102       }else if (key=="ssid:"){
103         WiFi_SSID+=bt_payload;
104       }else if (key=="pass:"){
105         WiFi_PASS+=bt_payload;
106       }else if (key=="rate:"){
107         rate+=bt_payload;
108       }else if (key=="currency:"){
109         currency+=bt_payload;
110       }else if (key=="payday:"){
111         payday+=bt_payload;
112       }else if (key=="name:"){
113         name+=bt_payload;
114       }else if (key=="timezone:"){
115         timezone+=bt_payload;
116       }else{
117         key+=bt_payload;
118       }
119     }
120   }
```

Continuación de la figura 40.

```
121     vTaskDelay(10);
122   }
123
124   void setup_wifi(String WiFi_SSID, String WiFi_PASS, boolean new_cred){
125     WiFi.disconnect();
126     delay(50);
127     // Connect to Wifi Network
128     Serial.println();
129     Serial.print("Conneting to: ");
130     Serial.println(WiFi_SSID);
131
132     WiFi.begin(WiFi_SSID.c_str(), WiFi_PASS.c_str());
133
134     int con = 0;
135     unsigned long t_WiFi = millis();
136     while (WiFi.status()!=WL_CONNECTED && con<8) {
137       if (millis() > t_WiFi+one_second){
138         t_WiFi = millis();
139         Serial.print(".");
140         con=con+1;
141       }
142     }
143
144     if (WiFi.status() == WL_CONNECTED) {
145       Serial.println("Connected to WiFi");
146       Serial.print("IP: ");
147       Serial.println(WiFi.localIP());
148       SerialBT.println("I");
149
150       if (new_cred){   //if credentials are correct they will be save
151         save_conf("wifi_cred", "ssid", WiFi_SSID);
152         save_conf("wifi_cred", "pass", WiFi_PASS);
153       }
154     } else {
155       Serial.println("Error: it could not connect to WiFi");
156       SerialBT.println("i");
157       if (new_cred){
158         SerialBT.println("cred_wrg");
159       }
160     }
```

Continuación de la figura 40.

```
161   }
162
163   void connectMqtt() {
164     //get MQTT credential from memory
165     preferences.begin("mqtt_cred", false);
166     const String mqtt_server = preferences.getString("mqtt_server", "");
167     const int mqtt_port = preferences.getInt("mqtt_port", 0);
168     const String mqtt_user = preferences.getString("mqtt_user", "");
169     const String mqtt_pass = preferences.getString("mqtt_pass", "");
170     preferences.end();
171
172     int con = 0;
173     unsigned long t_mqtt = millis();
174     while (!MqttClient.connected() && con<4 && WiFi.status()==WL_CONNECTED){
175       if (millis() > t_mqtt+one_second){
176         t_mqtt = millis();
177
178         Serial.print("Connecting to Mqtt...");
179         // Create a ID MqttClient
180         String clientId = device;
181         // Try to connect
182         MqttClient.setServer(mqtt_server.c_str(), mqtt_port);
183         MqttClient.setCallback(callback);
184         if (MqttClient.connect(clientId.c_str(),
185                                mqtt_user.c_str(),
186                                mqtt_pass.c_str())){
187           if(MqttClient.subscribe(device_topic.c_str()) and
188             MqttClient.subscribe(root_topic.c_str())){
189             Serial.println("Subscription ok");
190           }else{
191             Serial.println("Subscription failed");
192           }
193           Serial.println("Connected!");
194         } else {
195           Serial.print("failed with error -> ");
196           Serial.print(MqttClient.state());
197           Serial.println("Trying connecting again");
198         }
199         con=con+1;
200       }
```

Continuación de la figura 40.

```
201     }
202   }
203
204   void callback(char* topic, byte* payload, unsigned int length){
205     String incoming = "";
206     for (int i = 0; i < length; i++) {
207       incoming += (char)payload[i];
208     }
209     incoming.trim();
210     if (incoming=="reset"){
211       pzem.resetEnergy();   //reset counter
212       MqttClient.publish(device_topic.c_str(), "ok-reset");
213     } else if (incoming=="save-consum"){
214       send_info_server("server_update");  //save info in server
215     } else if (incoming=="ok-update"){
216       set_update_server(1);
217     } else if (incoming=="visualization"){
218       vis_count = 0;
219     }
220   }
221
222   void ask_wifi_credentials(){
223     preferences.begin("wifi_cred", false);
224     String WiFi_SSID = preferences.getString("ssid", "");
225     String WiFi_PASS = preferences.getString("pass", "");
226     preferences.end();
227     if (WiFi_SSID != "" && WiFi_PASS != ""){
228       setup_wifi(WiFi_SSID, WiFi_PASS, false);
229     }
230   }
231
232   void save_conf(String name_space, String key, String value){
233     preferences.begin(name_space.c_str(), false);
234     //save value if is not saved yet
235     if (!(preferences.getString(key.c_str(), "")==value)){
236       preferences.putString(key.c_str(),value);
237       if(key=="name" || key=="curre" || key=="payday"
238           || key =="timezone"){
239         //reset energy count if some of those keys changed
240         pzem.resetEnergy();
```

Continuación de la figura 40.

```
241       }
242     }
243     preferences.end();
244   }
245
246   void set_update_server(bool updat){
247     preferences.begin("energy_info", false);
248     //save value if is not saved yet
249     if (!(preferences.getBool("server_update", false)==updat)){
250       preferences.putBool("server_update",updat);
251     }
252     preferences.end();
253   }
254
255   void send_info_server(String dest){
256     String values[6];
257     //read values of energy sensor
258     values[0] = String(pzem.voltage(),2);
259     values[1] = String(pzem.current(),2);
260     values[2] = String(pzem.power(),2);
261     values[3] = String(pzem.energy(),3);
262
263     //get user data saved
264     preferences.begin("energy_info", false);
265     String rate = preferences.getString("rate", "");
266     String currency = preferences.getString("curre", "");
267     String payday = preferences.getString("payday", "");
268     String med_name = preferences.getString("name", "");
269     String timezone = preferences.getString("timezone", "");
270     bool is_server_update = preferences.getBool("server_update", false);
271     preferences.end();
272
273     if (dest=="android_config"){
274       if (WiFi.status() == WL_CONNECTED) {
275       SerialBT.println("I");
276       }else{
277         SerialBT.println("i");
278       }
279       delay(75);
280       SerialBT.println("rate:"+rate);
```

Continuación de la figura 40.

```
281       delay(75);
282       SerialBT.println("currency:"+currency);
283       delay(75);
284       SerialBT.println("payday:"+payday);
285       delay(75);
286       SerialBT.println("name:"+med_name);
287       delay(75);
288     } else {
289       //verify all data to send
290       boolean is_val;
291       is_val = rate!=""&&currency!=""&&payday!=""&&med_name!=""&&timezone!="";
292       for (String val:values){
293         if (val=="nan"){
294           is_val = false;
295           break;
296         }
297       }
298       if (is_val && MqttClient.connected()) {
299         // calcule rate
300         values[4] = String(values[3].toFloat()*rate.toFloat(),3)
301                     +" "+currency;
302         values[5] = String(values[3].toFloat()*rate.toFloat(),3);
303 
304         if (dest=="energy_measu"){
305           //send measurements to Android devices every second
306           for(int i=0;i<=4;i++){
307             MqttClient.publish(meas_topics[i].c_str(), values[i].c_str());
308           }
309         } else if (dest=="server_update") {
310           //to send information to server in order to save
311           //the energy registers in data base
312           if (!is_server_update){
313             //check if device configuration saved in server need to be updated
314             String data_update = "req"+String('\n')+device+String('\n')+
315                                  med_name+String('\n')+rate+String('\n')+currency+
316                                  String('\n')+payday+String('\n')+timezone;
317             //send an update of device configuration to server
318             MqttClient.publish(update_topic, data_update.c_str());
319           }
320 
```

Continuación de la figura 40.

```
321             String data_server = "req"+String('\n')+device+String('\n')+
322                     med_name+String('\n')+rate+String('\n')+currency+
323                     String('\n')+payday+String('\n')+timezone+String('\n')+
324                     values[0]+String('\n')+values[1]+String('\n')+values[2]+
325                     String('\n')+values[3]+String('\n')+values[5];
326             //send consumption data to server
327             MqttClient.publish(device_topic.c_str(), data_server.c_str());
328           }
329         }
330       }
331     }
332
333     void get_device(){ //init. vars with the name obtained from memory
334       preferences.begin("info_device", false);
335       device = preferences.getString("device", "");
336       preferences.end();
337       device_topic = root_topic+"/"+device;
338       meas_topics[0]=device_topic+"/voltage";
339       meas_topics[1]=device_topic+"/current";
340       meas_topics[2]=device_topic+"/power";
341       meas_topics[3]=device_topic+"/energy";
342       meas_topics[4]=device_topic+"/charge";
343     }
```

Fuente: elaboración propia, realizado con captura de pantalla.

5. APLICACIÓN ANDROID PARA CONFIGURACIÓN Y MONITOREO

Por medio de esta aplicación se establecen 2 tipos de conexiones inalámbricas para cumplir las funciones requeridas por el sistema, dichas conexiones se establecen a través de WiFi y Bluetooth Classic.

La conexión Bluetooth establecida entre el dispositivo de medición y un dispositivo Android se da con el fin de enviar la información energética del usuario al dispositivo de medición, dicha información es guardada y luego usada para definir las variables necesarias para realizar los distintos cálculos energéticos requeridos para ejecutar el algoritmo del sistema. La segunda conexión consiste en acceder a Internet por medio de WiFi con el propósito de establecer la comunicación entre el servidor remoto MQTT (*broker*), y un dispositivo Android que servirá como monitor para la visualización de consumo energético (en tiempo real), y consulta de historial.

La aplicación fue desarrollada en Android Studio utilizando Java como lenguaje de programación, para gestionar la comunicación MQTT y manejo de mensaje se utilizó Eclipse Paho como cliente MQTT, así como también código propio. La aplicación cuenta con una funcionalidad fuera de línea en la cual se encuentran limitadas las funciones a únicamente la configuración de dispositivos de medición, cuando se cuenta con conexión a internet se permite el monitoreo por medio de la visualización energética en tiempo real. La aplicación está diseñada para ser compatible con la mayoría de los teléfonos y tabletas que usan Android como sistema operativo.

5.1. Android Studio

Es el IDE oficial para el desarrollo de aplicaciones Android, está basado en el IDE Intellij IDEA de JetBrains, anteriormente las aplicaciones se desarrollaban en un entorno Eclipse, pero esto cambió cuando Google lanzó su propio IDE y lo convirtió en el oficial para el desarrollo de aplicaciones Android, actualmente soporta Java y Kotlin como lenguajes de programación, aunque también ofrece la posibilidad de agregar código C y C++. (Android developers, 2021)

Una de las características más notables de este entorno es su compilación por medio de Gradle, el cual realiza procedimientos avanzados para la automatización y administración de la compilación de código, y tiene la capacidad de ejecutar máquinas virtuales (previamente instaladas), para la simulación de proyectos.

5.1.1. Instalación Android Studio

La versión Bumblebee de Android Studio cuenta con soporte para sistemas operativos Windows, MacOs, Linux y Chrome Os, aunque es impórtate revisar que se cumpla con los requisitos mínimos de hardware y software según el sistema operativo de interés. Para ver las opciones de descarga disponibles y sus respectivos requisitos es necesario consultar el sitio oficial de descargas.

5.1.1.1. Instalación en Windows

Los pasos para realizar la instalación de Android Studio Bumblebee 2021.1.22 en un sistema operativo Windows, antes de iniciar la instalación se debe estar seguro de que se cumplen con los requisitos mínimos., ver tabla II.

Tabla II. **Requisitos mínimos Android Studio Bumblebee (Windows)**

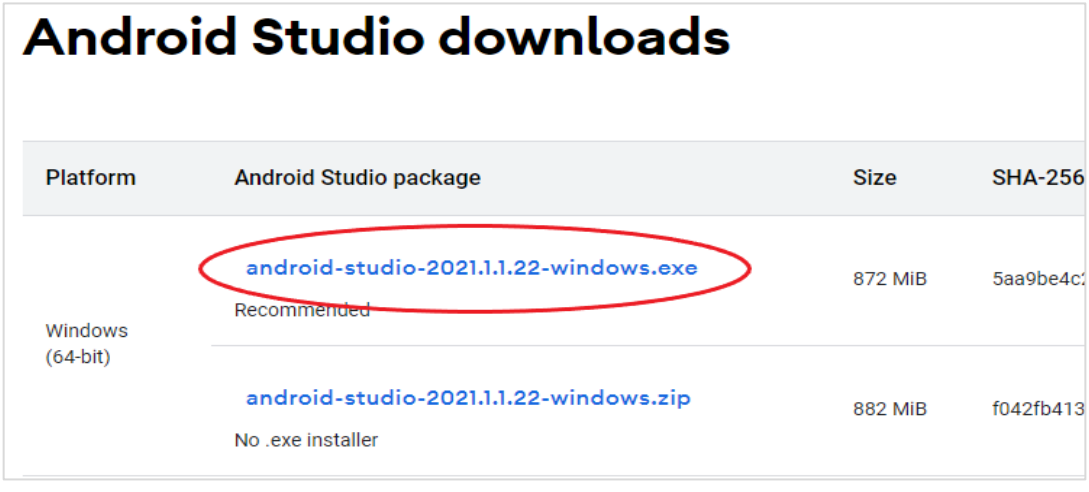
Microsoft Windows	8/10 64-bit
Procesador	Arquitectura x86_64, Intel Core de 2da generación o superior, AMD con soporte para Windows Hypervisor
Memoria Ram	8 GB
Espacio en disco duro	8 GB
Resolución de monitor	1280x800

Fuente: elaboración propia.

- Paso 1: descargar el instalador aceptando los términos y condiciones, enlace sitio oficial de descargas:

  - https://developer.android.com/studio#downloads

Figura 41. **Descarga de Android Studio para Windows**



Fuente: elaboración propia, realizado con Paint3D.

- Paso 2: ejecutar como administrador el instalador y hacer clic en el botón Next para continuar con la instalación.

    - Paso 2.1: elección de componentes, esta opción permite instalar un dispositivo Android virtual (Emulador), junto a Android Studio.

    - Paso 2.2: localización para la instalación, aquí se pregunta en que localización del disco duro se desea realizar la instalación, se recomienda usar la instalación predeterminada por Windows. Después de elegir la localización hacer clic en el botón Next.

    - Paso 2.3: selección de carpeta del menú de inicio, sirve para seleccionar la carpeta donde se guardarán los atajos del programa, se recomienda crear la carpeta por defecto propuesta por el programa de instalación, para continuar pulsar el botón Install.

    - Paso 2.3: esperar a que el instalador termine el proceso, cuando el instalador muestre la palabra *Completed* ya se podrá hacer clic en el botón Next.

- Paso 3: finalizar la instalación ejecutando Android Studio

- Paso 4: importar configuraciones, si se tiene un archivo de configuraciones utilizado anteriormente este se puede importar a la nueva versión o bien no importar ninguna, presionar el botón OK, para continuar.

- Paso 5: bienvenida, el IDE dará la bienvenida al entorno informando que hará un análisis para determinar si falta algún software para poder iniciar,

si falta algún componente este dará el aviso en los siguientes pasos, pulsar el botón Next, para continuar.

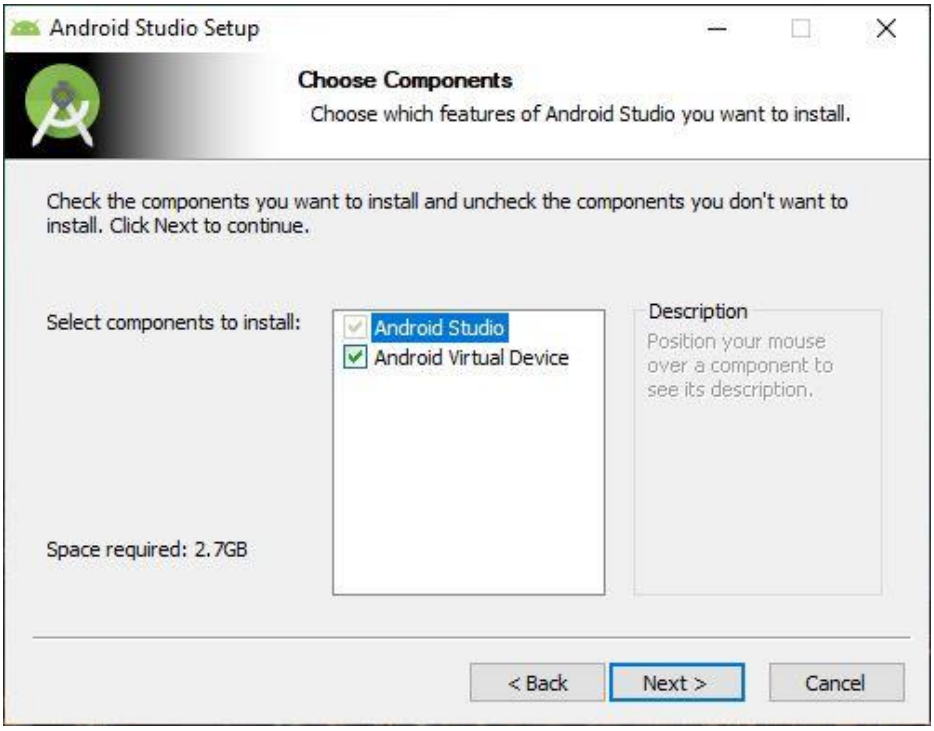
- Paso 6: selección de configuración inicial, se tiene las opciones de elegir entre una configuración estándar o una personalizada (para usuarios avanzados), se recomienda elegir la configuración estándar, avanzar haciendo clic en el botón Next.

- Paso 7: selección del tema UI, se puede elegir entre un tema oscuro o uno claro, después de haber elegido el tema hacer clic en el botón Next.

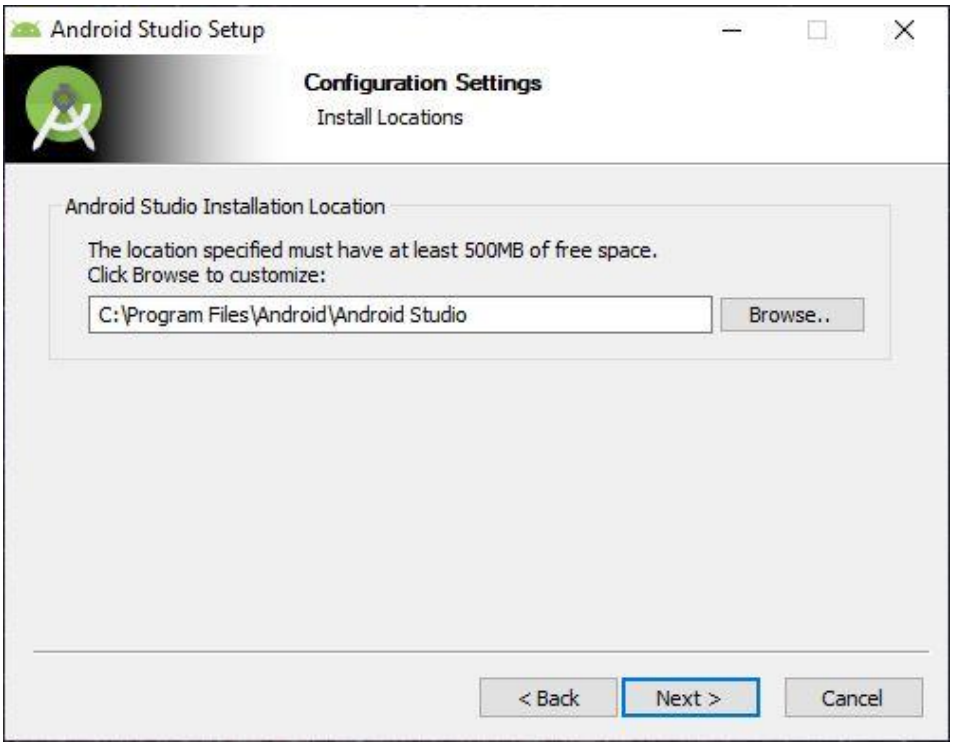
- Paso 8: se muestra un resumen de la configuración y el software faltante que se necesita para ejecutar el programa, si se desea cambiar alguna configuración se tiene la opción de retroceder con el botón Previous, de lo contrario hacer clic en el botón Next, para continuar.

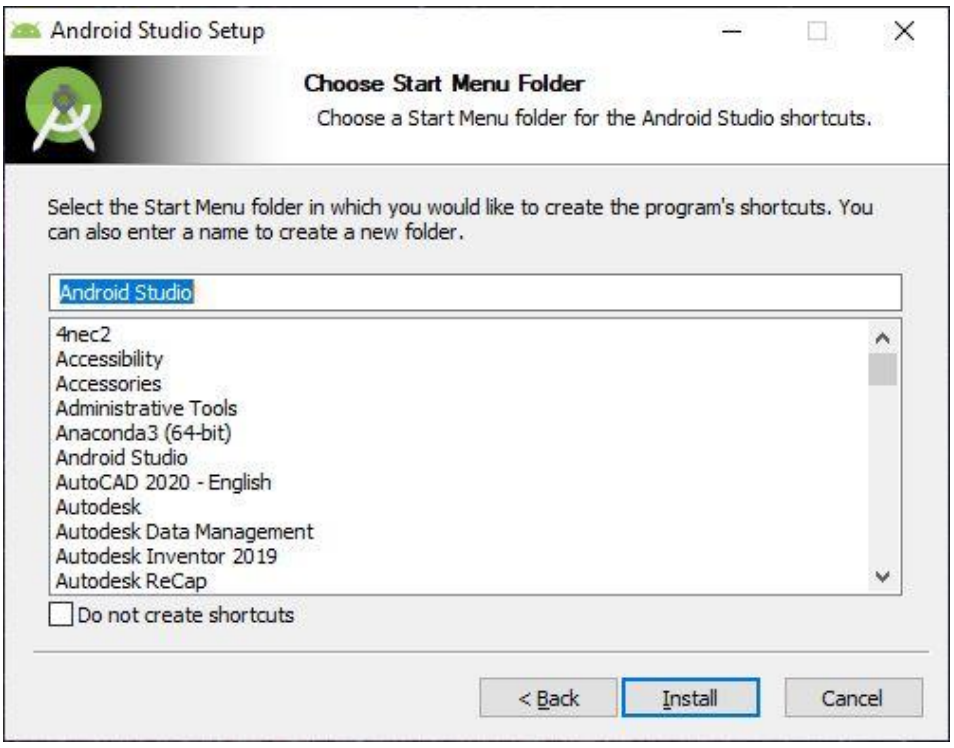
- Paso 9: se mostrará una ventana en donde se debe aceptar los términos y condiciones en caso se haya elegido instalar un emulador junto al IDE. Hacer clic en el botón Finish, para finalizar el proceso, en caso haya software faltante se iniciará la descarga de esté y una vez finalizada se abrirá el mensaje de bienvenida al IDE Android Studio, lo que indica que la instalación fue realizada satisfactoriamente.

Figura 42. **Instalación de Android Studio en Windows**

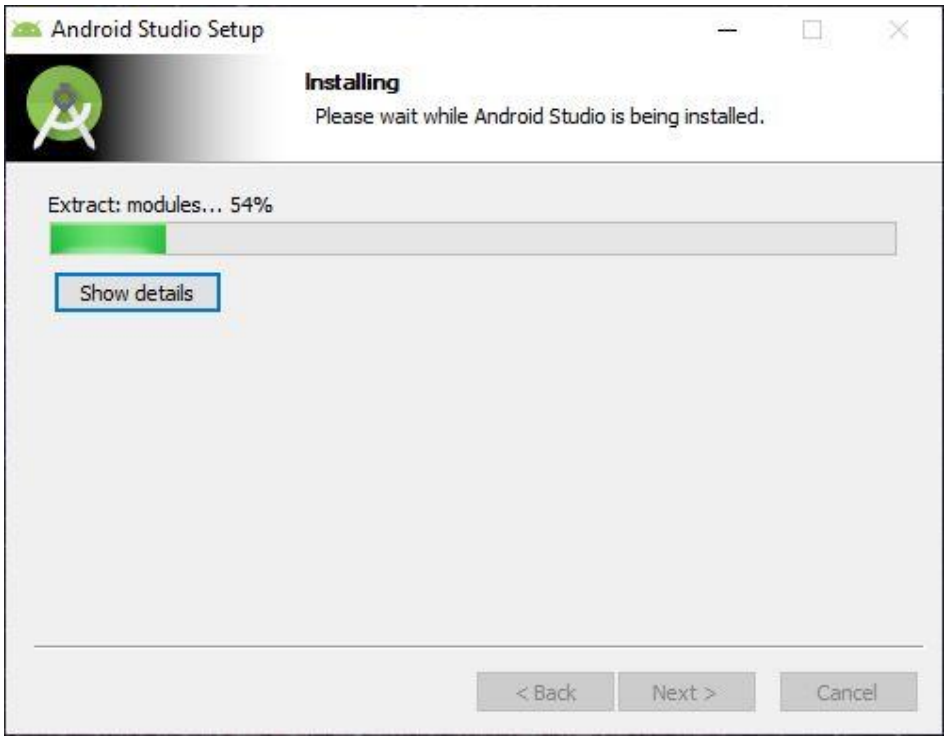


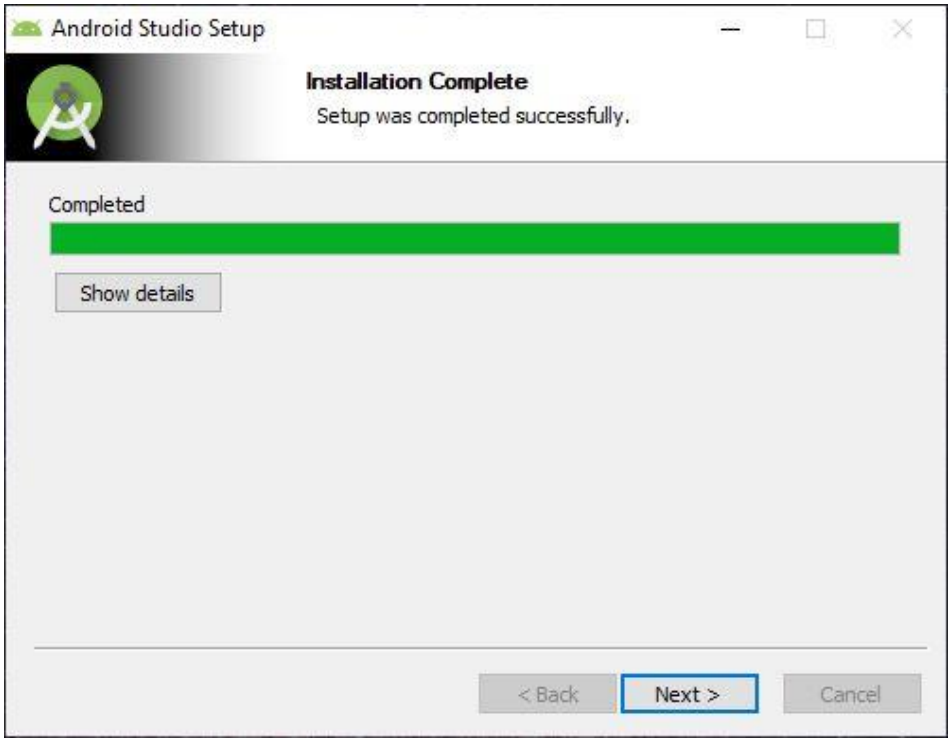
Continuación de la figura 42.



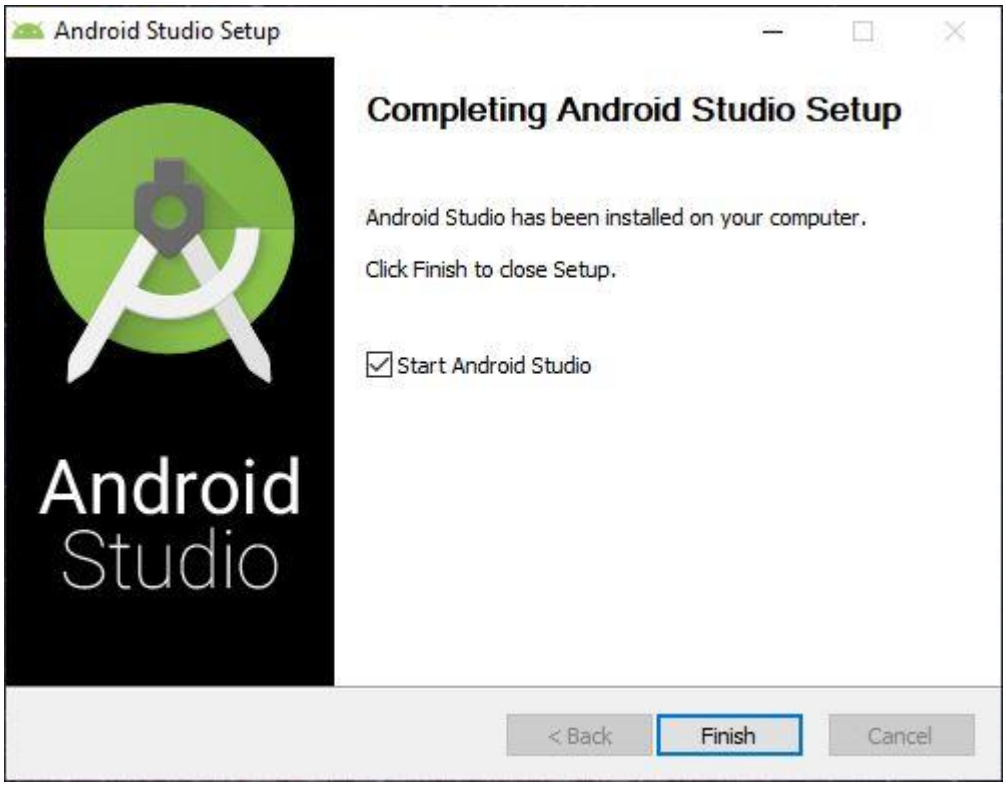


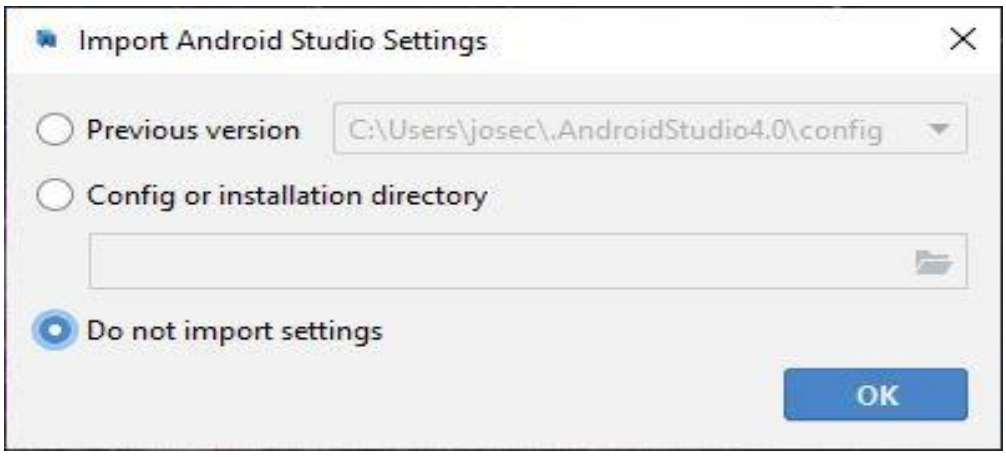
Continuación de la figura 42.



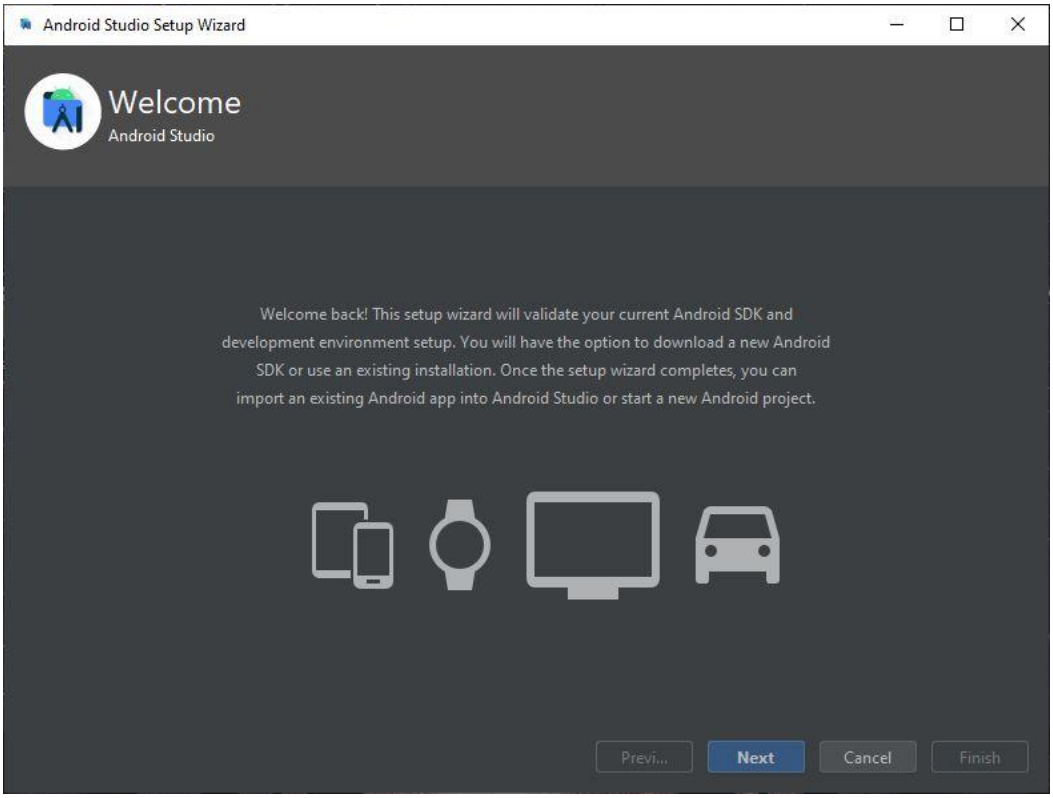


Continuación de la figura 42.

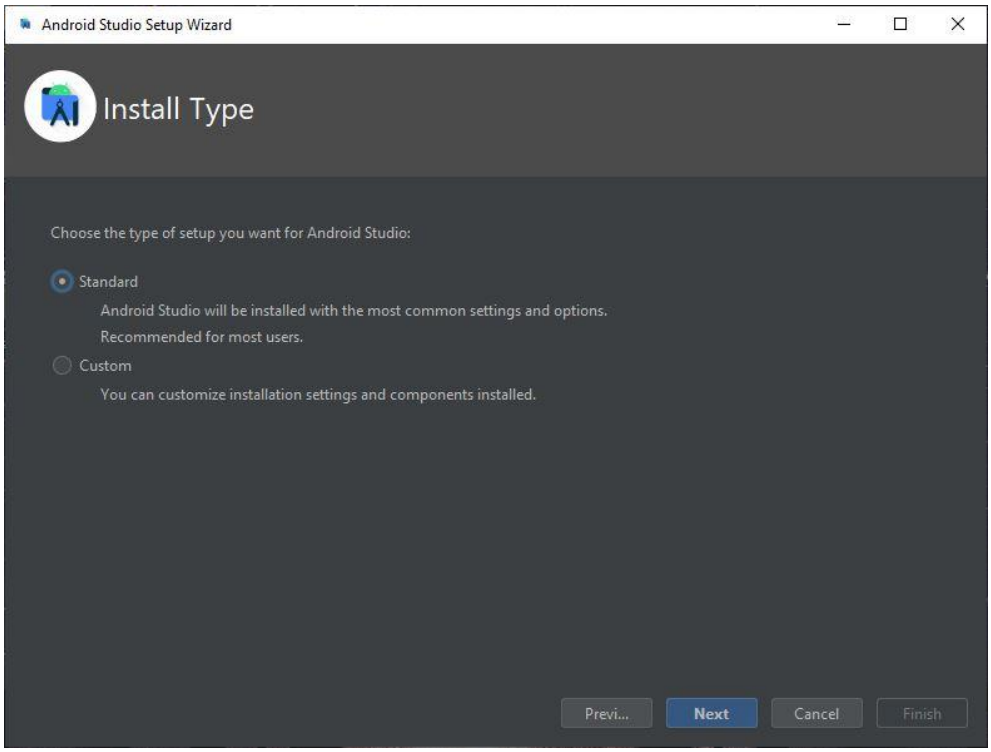


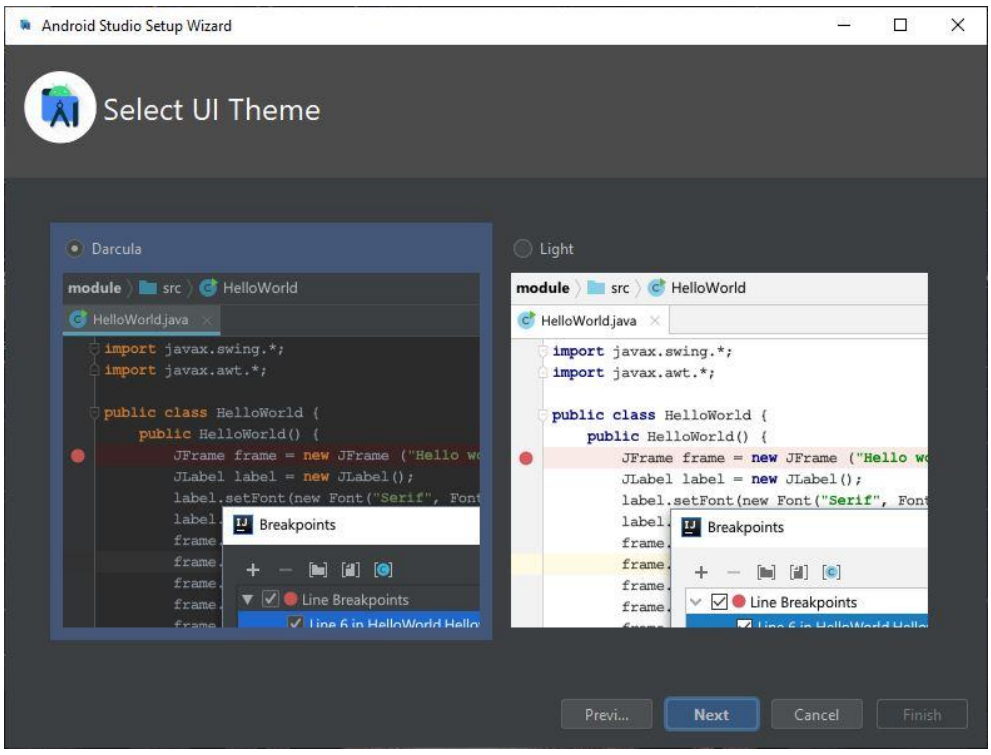


Continuación de la figura 42.

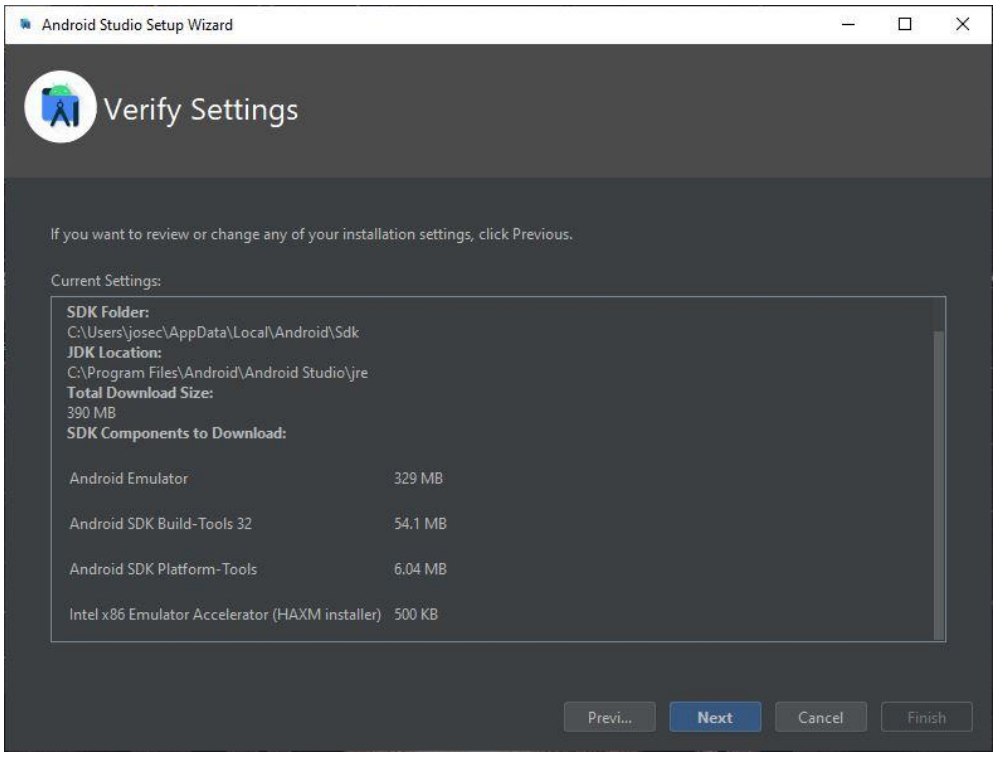


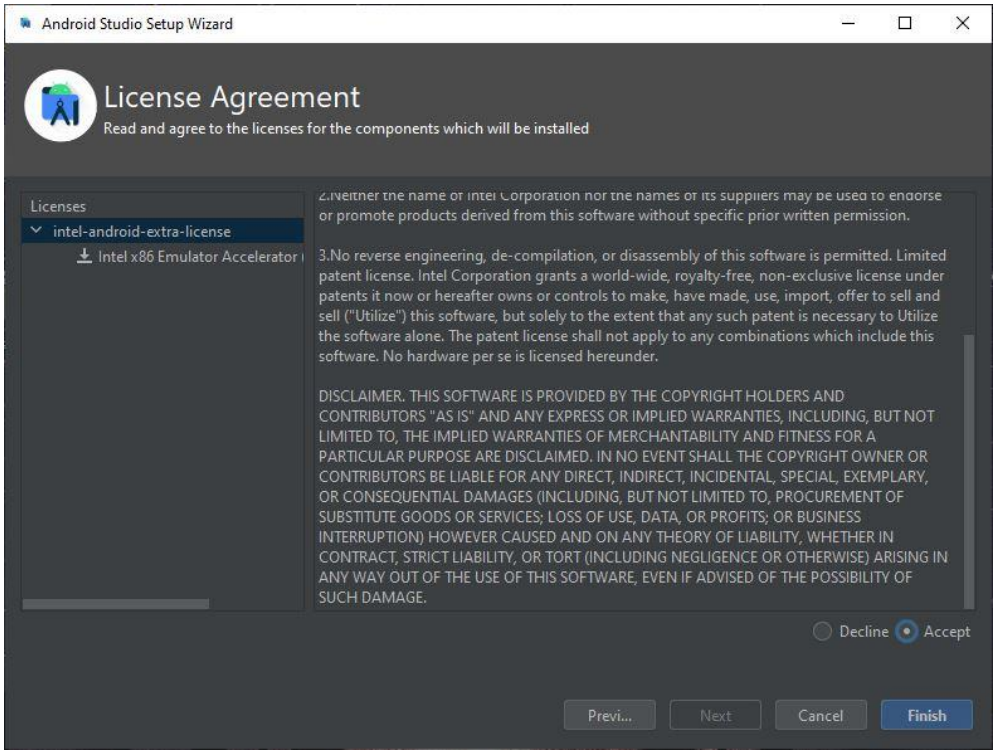
Continuación de la figura 42.



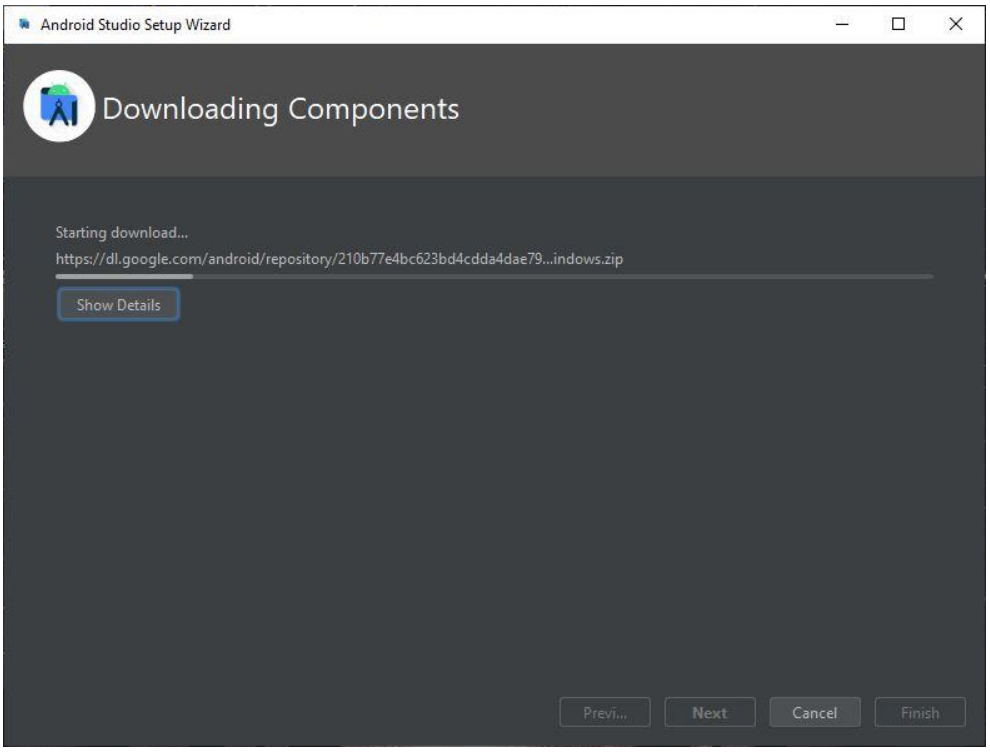


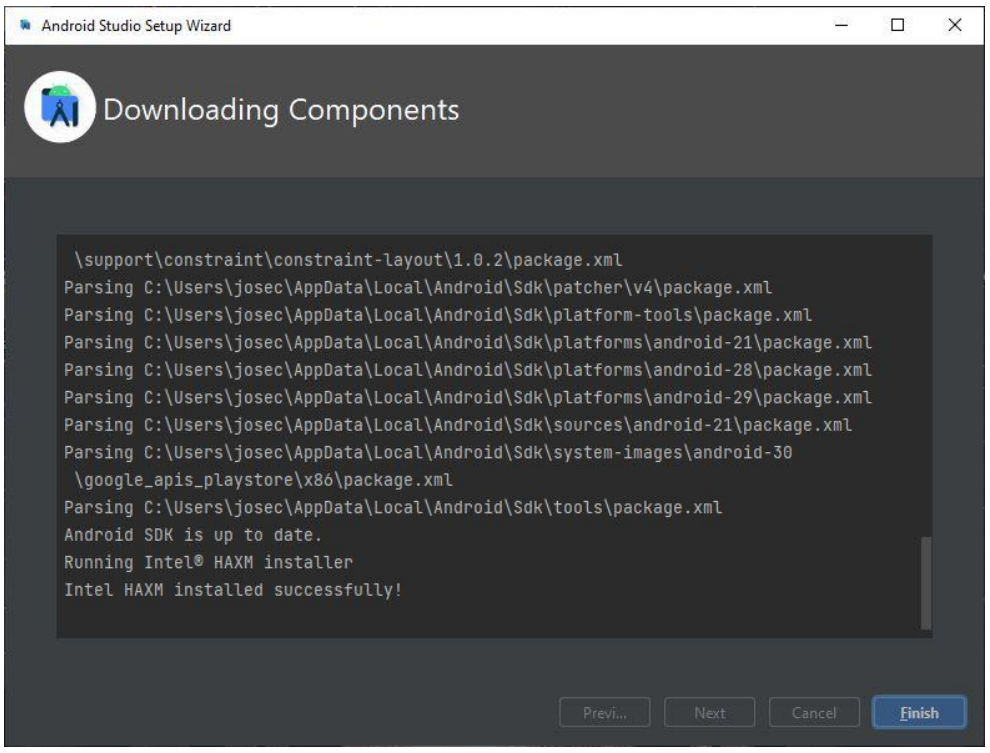
Continuación de la figura 42.



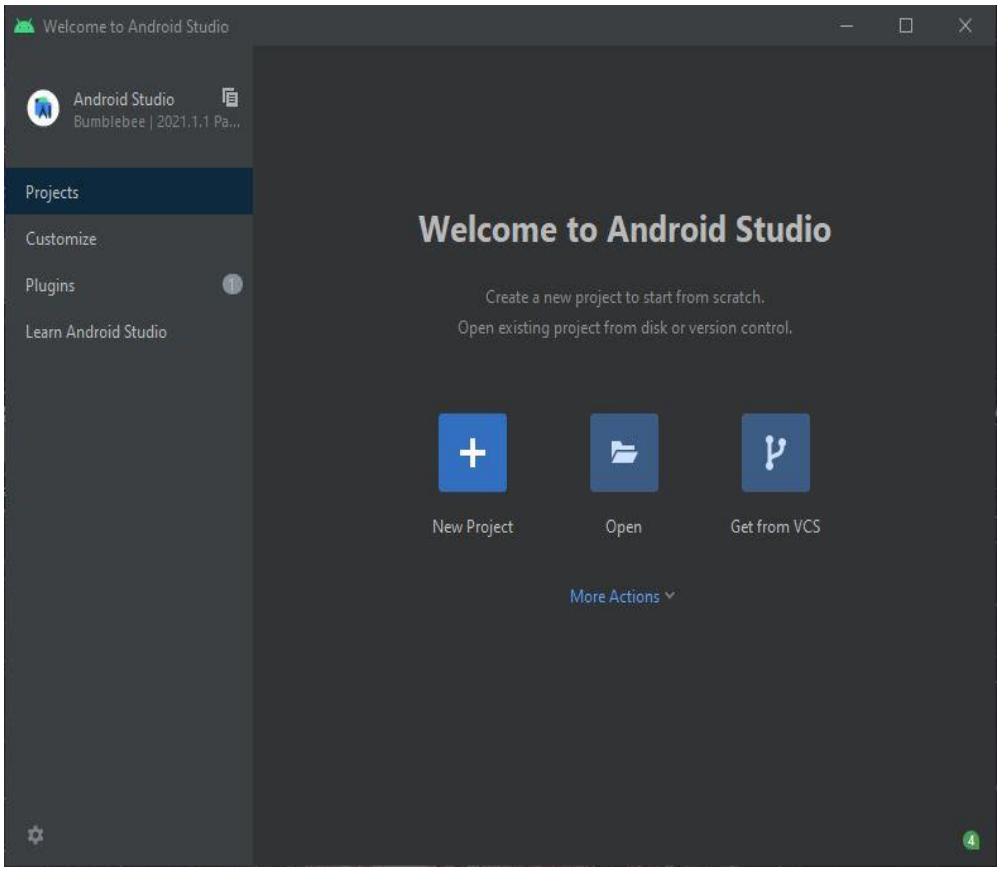


Continuación de la figura 42.





Continuación de la figura 42.



Fuente: elaboración propia, realizado con Paint3D.

5.1.2. Creación de un proyecto

Android Studio permite crear aplicaciones por medio de proyectos, la versión Bumblebee del IDE cuenta con la posibilidad de desarrollar proyectos para dispositivos inteligentes tales como: tabletas, celulares, relojes y automóviles. (Android developers, 2021)

A continuación, se muestran los pasos a seguir para crear un proyecto con soporte para celulares y tabletas:

- Paso 1: estando en la pantalla de bienvenida de Android estudio pulsar el botón New Project.

- Paso 2: seleccionar la plantilla *Phone and Tablet,* situada al lado izquierdo de la ventana, el lado derecho de la ventana permite seleccionar el tipo de actividad a implementar, esta actividad depende del tipo de aplicación a desarrollar, por lo que para esta demostración se seleccionará *Empty Activity,* que crea una actividad vacía.

- Paso 3: rellenar los campos correspondientes a algunas propiedades del proyecto. Los campos son los siguientes:

    - Name: determina el nombre de la aplicación

    - Package name: sirve para asignar un nombre al conjunto de archivos y recursos de la aplicación.

    - Save location: con esta opción se puede elegir la ubicación en donde se guardará el proyecto.

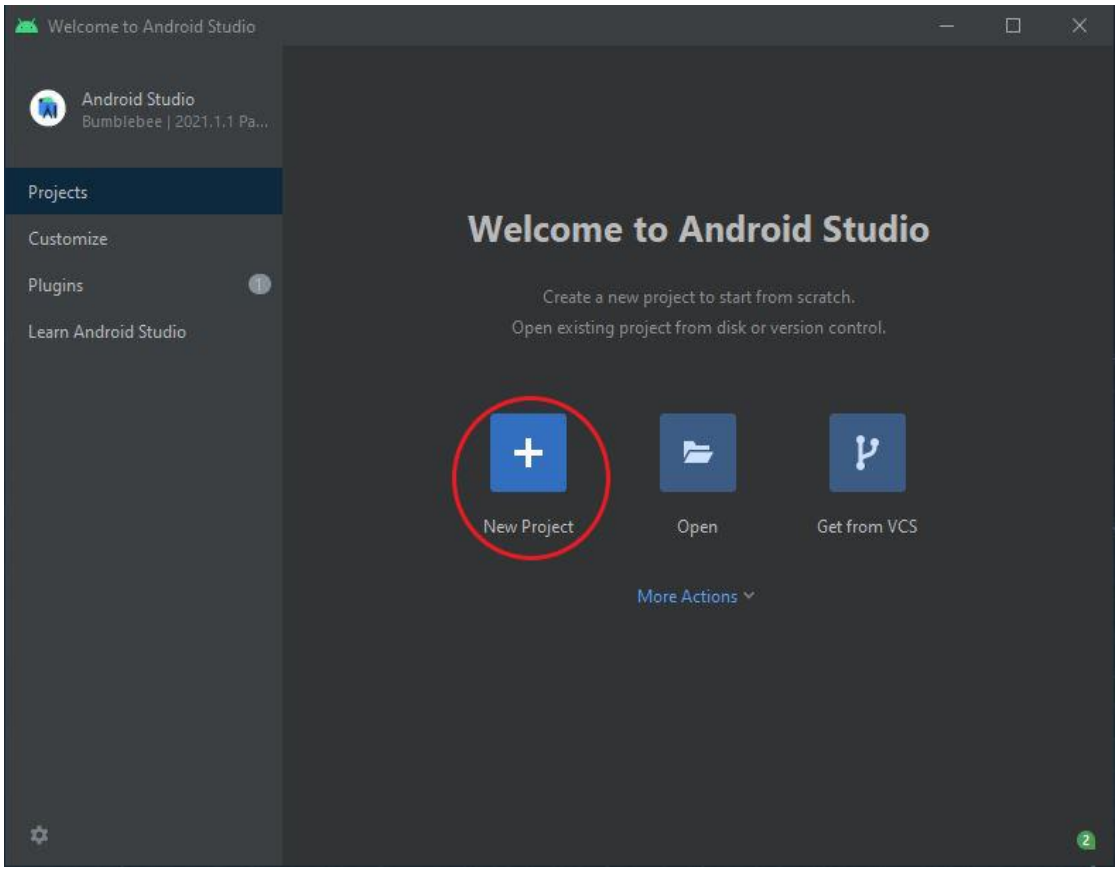
    - Language: permite elegir el lenguaje de programación, las opciones disponibles son Java y Kotlin.

    - Minimun SDK: determina la versión mínima de Android en la que se podrá ejecutar la aplicación.

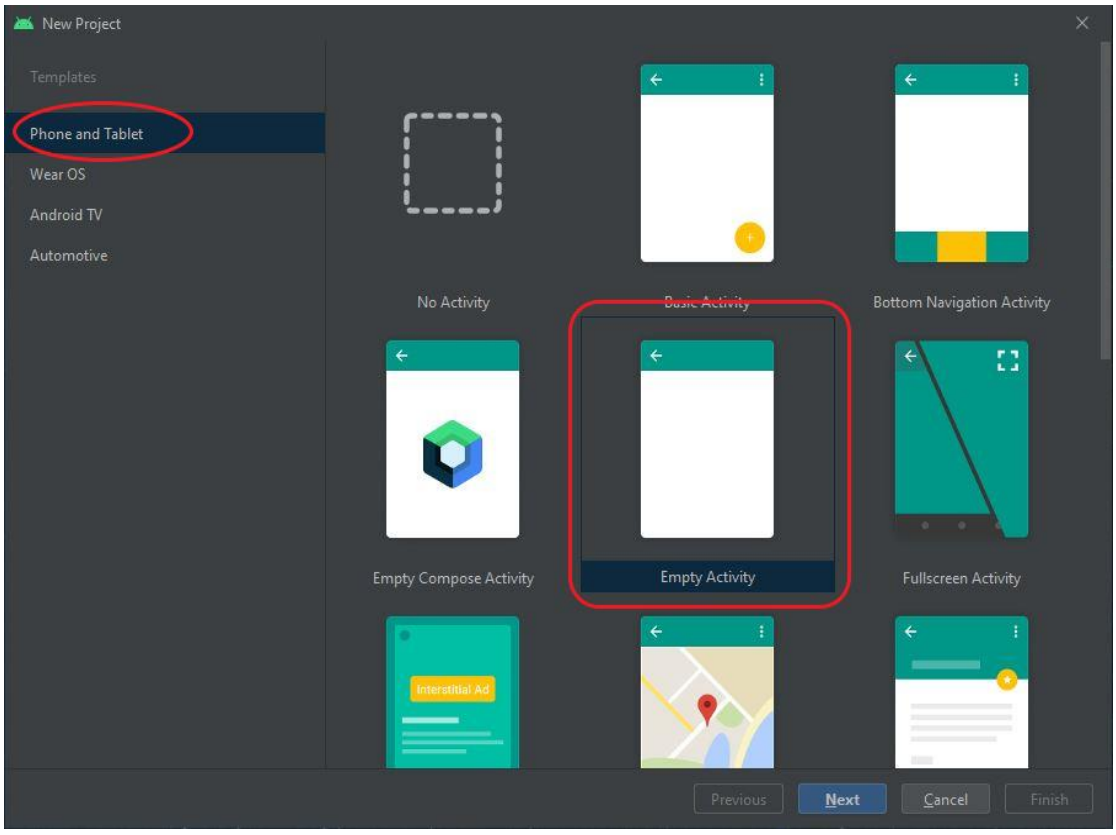
- Use legacy: esta opción sirve para dar soporte a librerías antiguas permitiendo prevenir el uso de las ultimas librerías brindadas por Play Services y Jetpack.

- Paso 4: hacer clic en el botón Finish, como consecuencia se empezarán a crear los archivos del proyecto, al finalizar, si el resultado del procedimiento fue exitoso los archivos ya serán visibles en la sección “Project” ubicada en la parte izquierda de la ventana.

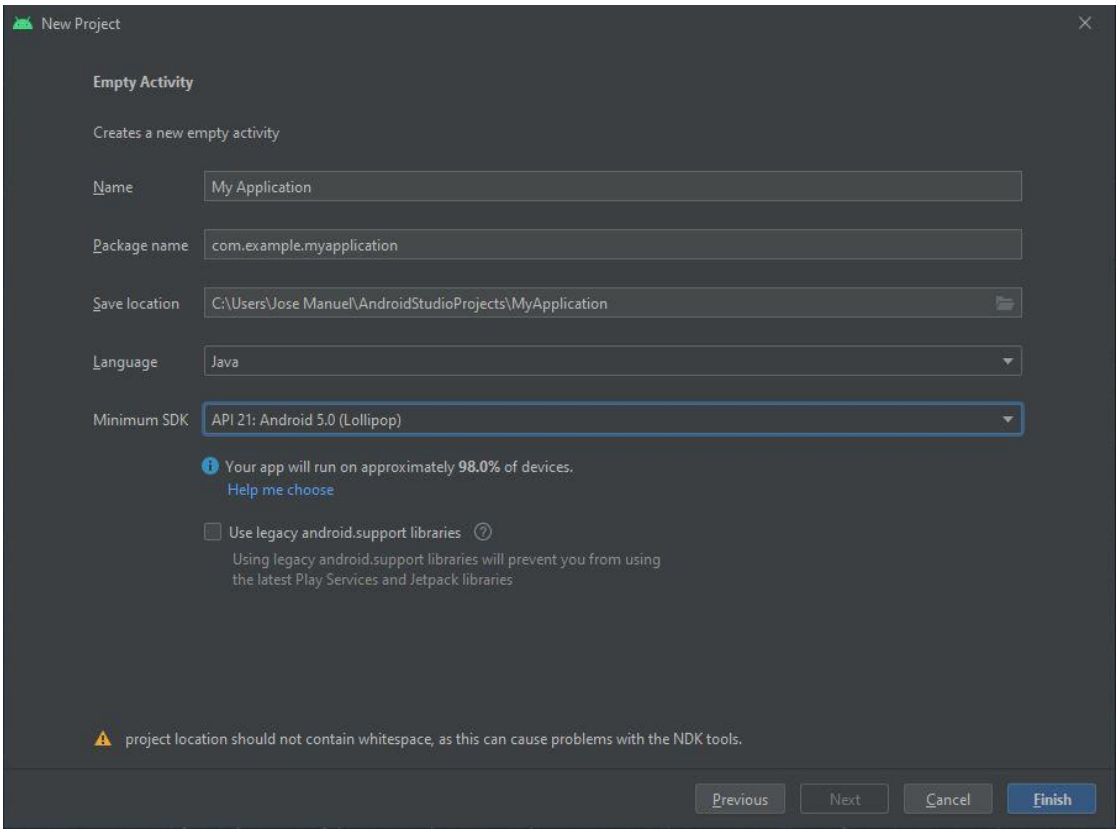
Figura 43. **Crear nuevo proyecto Android Studio**

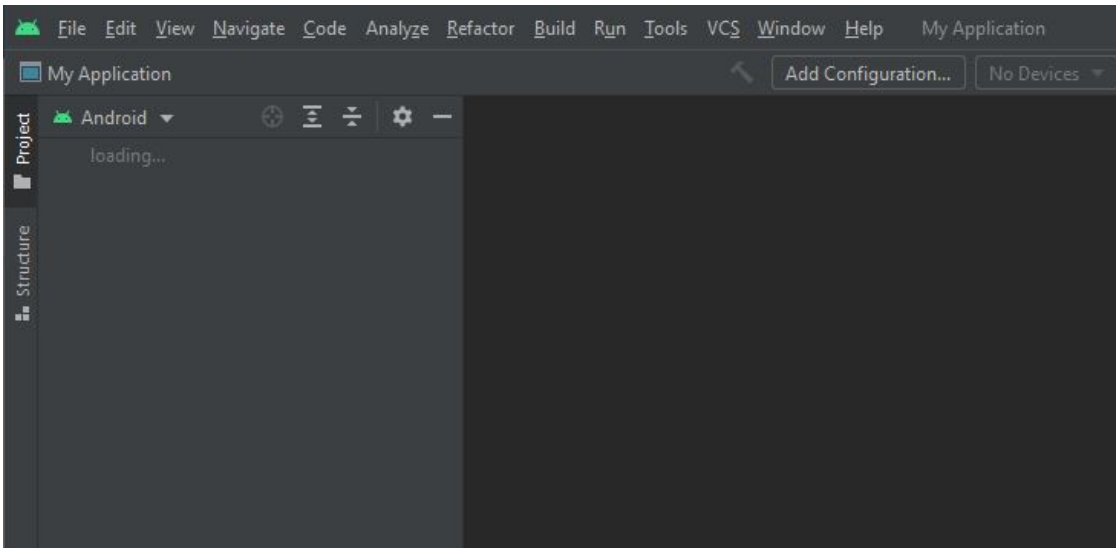


Continuación de la figura 43.

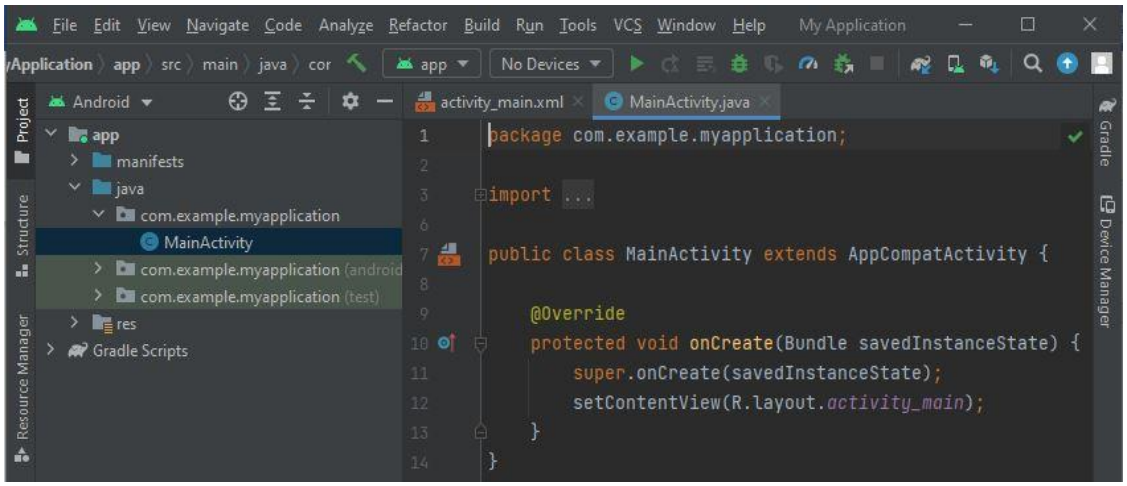


Continuación de la figura 43.





Continuación de la figura 43.



Fuente: elaboración propia, realizado con Paint3D.

5.1.2.1. Módulos de un proyecto

Un módulo dentro de un proyecto funciona como un contenedor para almacenar recursos y código, un proyecto puede tener tantos módulos como sean necesarios. Un nuevo módulo es útil cuando se requiere crear una biblioteca o desarrollar un conjunto de aplicaciones con soporte a varios tipos de dispositivos, y se tiene la posibilidad de compilar y depurar individualmente cada módulo, al crear un nuevo proyecto de forma predeterminada se crea un módulo determinado por el tipo de aplicación elegida. Android Studio Bumblebee dispone de los siguientes tipos:

- Módulo de teléfono y Tablet
- Módulo de Wear OS
- Módulo de Android TV
- Módulo de Glass

5.1.2.2. Vistas y archivos de un proyecto

Por medio de la vista Android, se agrupan los archivos de un proyecto en conjuntos los cuales facilitan el acceso a los archivos más utilizados durante el desarrollo, esta vista difiere de la vista Project ordenada por jerarquía, al usar la vista Android, los archivos no se muestran con la misma estructura con la que están almacenados en el disco e incluso se ocultan algunos directorios y archivos irrelevantes. (Android developers, 2022)

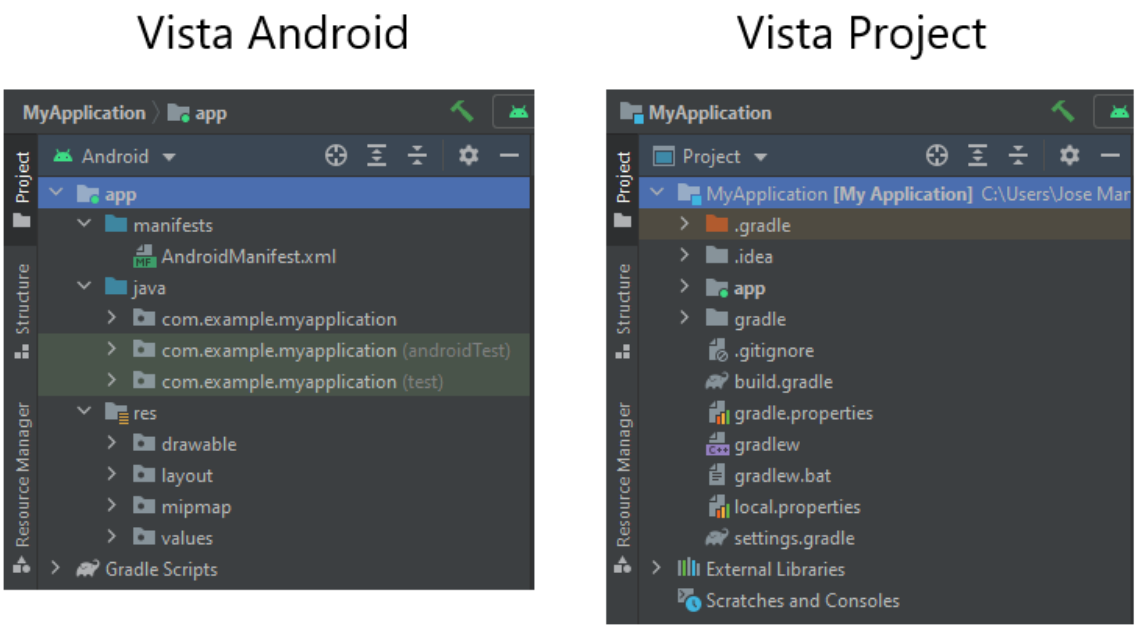
La vista “Android” organiza en 3 grupos los archivos dentro de cada módulo perteneciente a un proyecto. Estos grupos se detallan a continuación:

- manifests: contiene el archivo AndroidManifest.xml, que describe información y permisos de la aplicación.

- java: almacena los archivos de código fuente de Java (clases, interfaces, funciones, métodos, entre otros).

- res: se encuentran los recursos (sin código), de la aplicación, por lo general se pueden encontrar archivos de diseños XML, variables IU, temas, gráficos, entre otros recursos.

Figura 44. **Comparación entre vistas de archivos en Android Studio**



Fuente: elaboración propia, realizado con Paint3D.

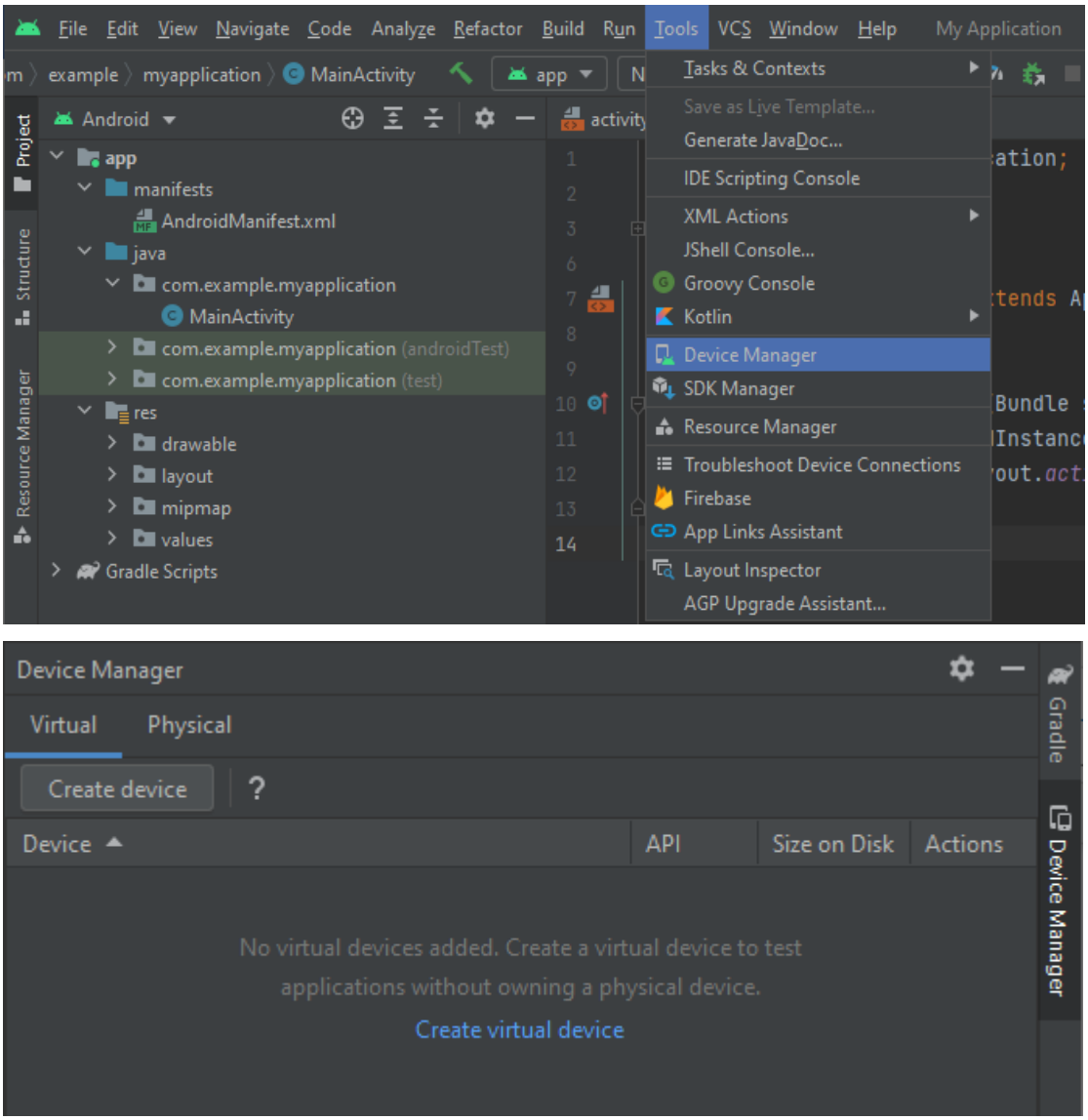
5.1.3. Crear medio de ejecución

Un medio de ejecución es útil para probar el funcionamiento de aplicaciones y realizar tareas de depuración, las maneras en las que opera un medio de ejecución son virtualmente y sobre dispositivos de hardware.

5.1.3.1. Crear emulador

Para crear un emulador funcional en Android Studio es necesario ir a la barra superior y navegar a Tools > Device Manager, esto abrirá el administrador de dispositivos, consecuente a esto se mostrarán los dispositivos virtuales y físicos.

Figura 45. **Abrir administrador de dispositivos Android Studio**

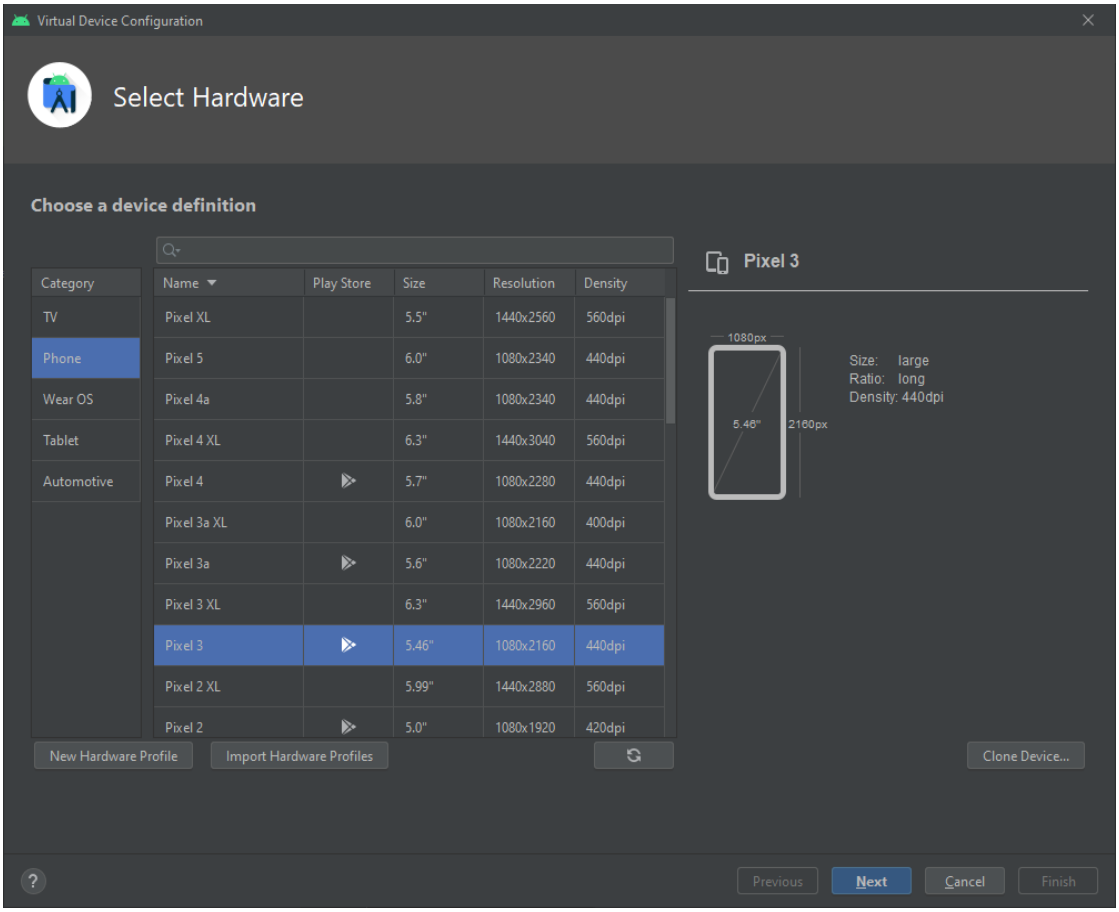


Fuente: elaboración propia, realizado con captura de pantalla.

Hacer clic en el botón Create device, para abrir la ventana de selección de hardware en donde se elige la categoría del dispositivo (TV, Phone, Wear OS, Tablet y Automotive), y las características del hardware, es posible elegir

hardware basado en dispositivos reales, crear un perfil personalizado e importar un perfil de hardware previamente creado.

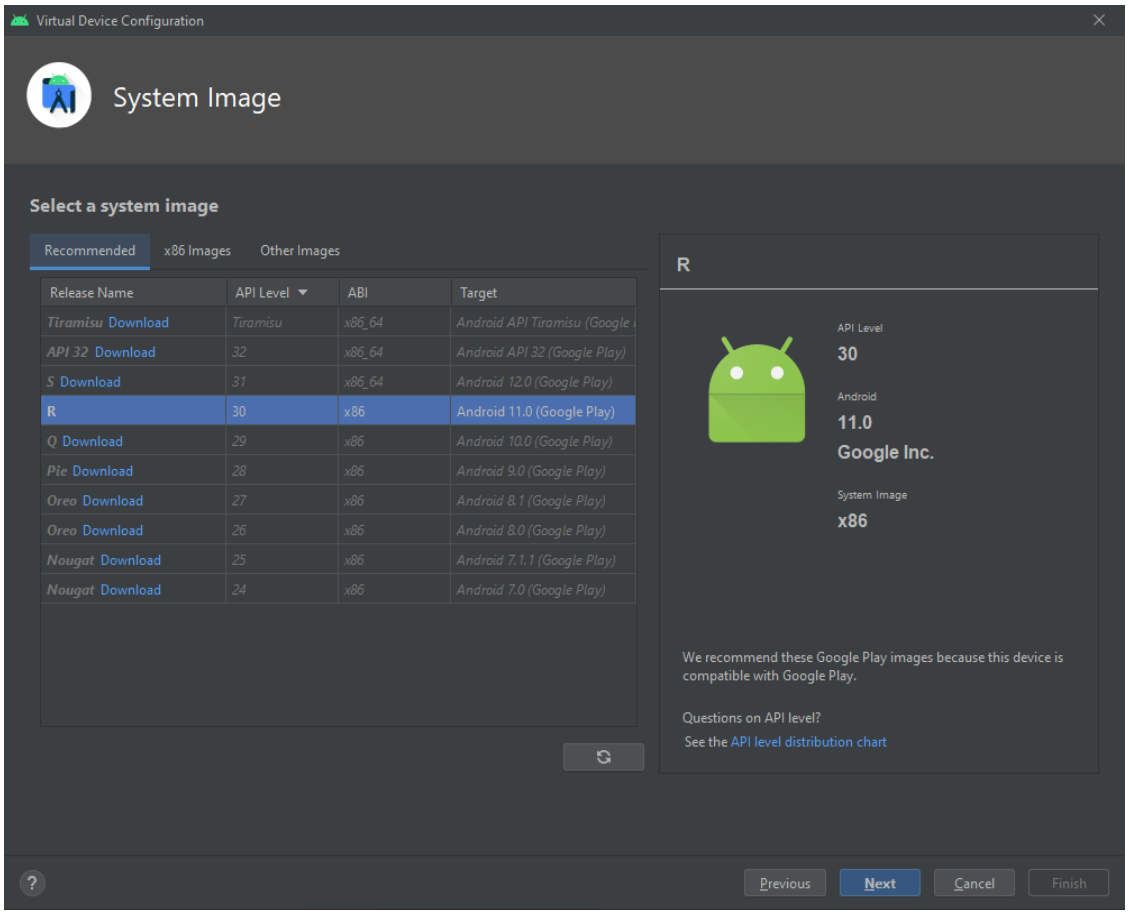
Figura 46. **Ventana selección de hardware (crear emulador Android Studio)**



Fuente: elaboración propia, realizado con captura de pantalla.

Después de elegir el perfil de hardware se pregunta por la imagen del sistema que consiste en la versión de Android que utilizara el emulador, si no se cuenta con la imagen se debe descargar por medio de la opción Download.

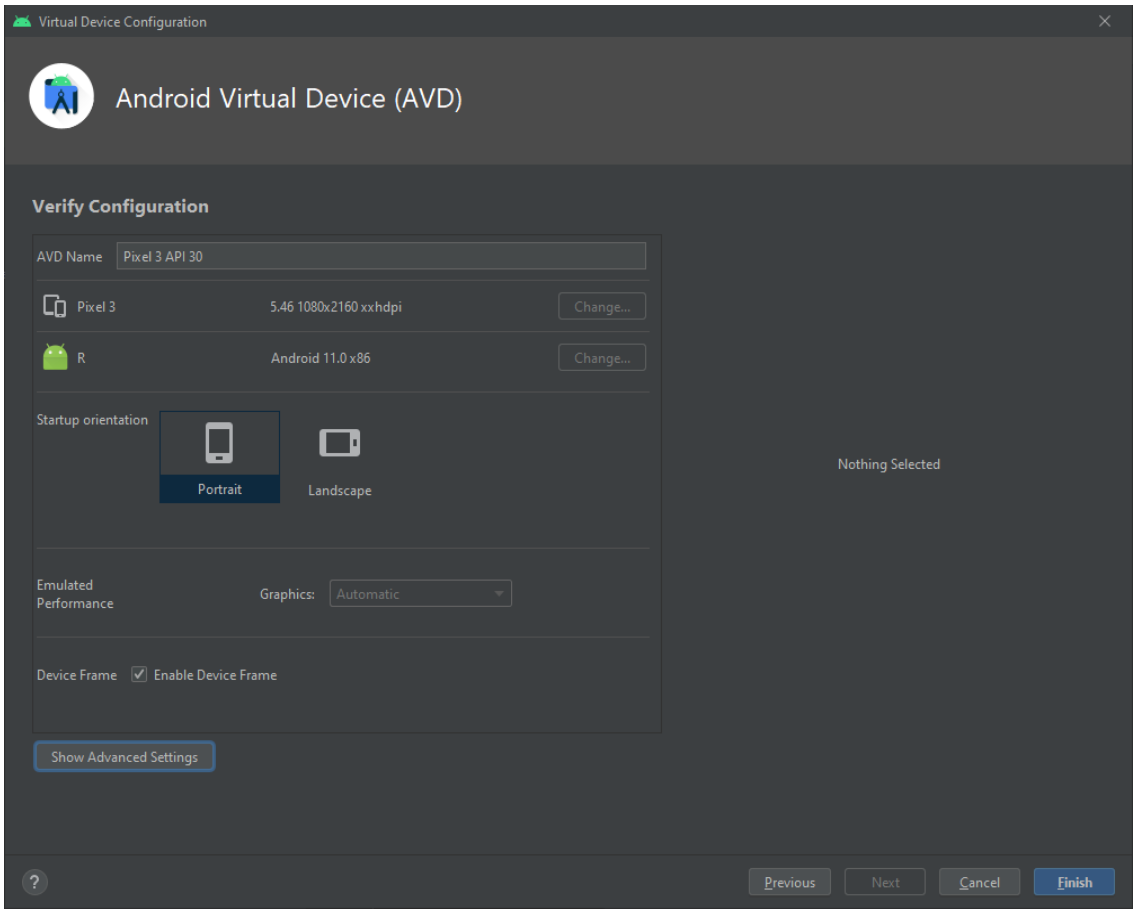
Figura 47. **Ventana de selección de imagen (crear emulador Android Studio)**



Fuente: elaboración propia, realizado con captura de pantalla.

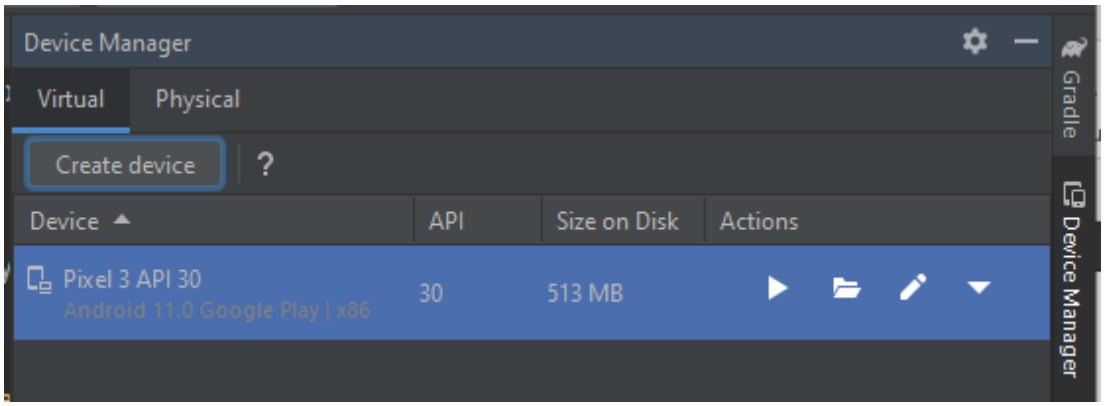
Por medio de la ventana de verificación de configuración se puede elegir la orientación del emulador, las configuraciones avanzadas de esta sección permiten elegir cuanto recurso de hardware del sistema anfitrión se le asignará al emulador. Una vez asignadas y verificadas todas las configuraciones hacer clic en el botón Finish, para crear el emulador, si todo el procedimiento ha finalizado con éxito se podrá observar el dispositivo virtual por medio del administrador de dispositivos.

Figura 48. **Ventana de verificación de configuración (crear emulador Android Studio)**



Fuente: elaboración propia, realizado con captura de pantalla.

Figura 49. **Dispositivo virtual (Emulador) creado con éxito Android Studio**



Fuente: elaboración propia, realizado con captura de pantalla.

5.1.3.2. Configurar opciones de desarrollador en dispositivos Android

El administrador de dispositivos es capaz de detectar dispositivos físicos conectados al computador por medio de una conexión USB o por medio de una red WiFi siempre y cuando el dispositivo Android cuente con las opciones de desarrollador activadas, en caso no se cuenten con estas opciones habilitadas abrir las configuraciones del dispositivo > acerca del teléfono > pulsar 7 veces en Número de compilación, seguido de esto aparecerá un mensaje indicando que ya se han activado estas opciones.

Con las opciones de configuración ya habilitadas se procede a activar la depuración, ir a Configuraciones > Sistema > Avanzado > Opciones para desarrollador > pulsar en Depuración USB, si se desea activar la depuración por WiFi entonces pulsar en Depuración inalámbrica, después de esto el administrador de dispositivos ya será capaz de detectar el dispositivo. Los

procedimientos anteriormente descritos son válidos para versiones de Android 9, para versiones anteriores o posteriores consultar el sitio web oficial para desarrolladores a fin de obtener documentación e instrucciones, sitio oficial de la documentación para activar las opciones de desarrollador en Android: https://developer.android.com/studio/debug/dev-options.

5.1.4. Ejecutar aplicación

La ejecución de la aplicación es un proceso fundamental en el desarrollo pues permite probar el código con la finalidad de detectar errores o funcionamiento inesperado que conduzca a una depuración de código. Para ejecutar una aplicación en Android Studio ir a la barra superior y navegar por Run > Run 'App' o presionar la combinación de teclas Mayus+F10 y la aplicación iniciará su ejecución en el dispositivo que se encuentre seleccionado, si se desea depurar la aplicación navegar por Run > Debug 'App' o presionar la combinación de teclas Mayus+F9 y la aplicación iniciará su ejecución en modo depuración.

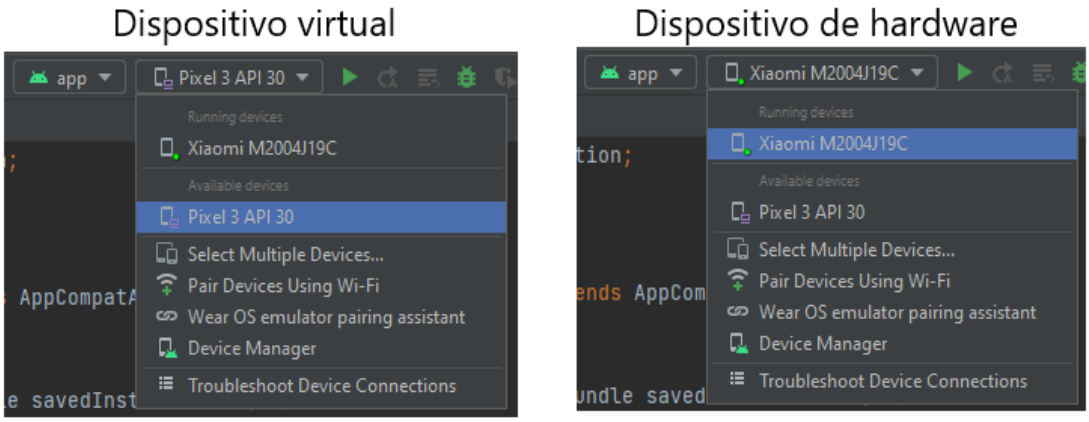
5.1.4.1. Ejecución en emulador

Para ejecutar aplicaciones usando un dispositivo virtual (emulador), es necesario haberlo seleccionado antes de ejecutar una depuración, una de las principales ventajas de usar emuladores es la facilidad de probar la aplicación en varios dispositivos y versiones Android disponibles en el IDE, por el contrario una de las desventajas más notorias para equipos con recursos demasiado ajustados a los requerimientos mínimos es la lentitud de ejecución debido a los recursos demandados por el emulador lo cual conlleva a una experiencia poco agradable en ejecución y depuración.

5.1.4.2. Ejecución en dispositivo de hardware

La ejecución en dispositivos de hardware es muy versátil y ofrece una experiencia más fluida para equipos de bajos recursos dado que se utiliza un dispositivo físico para ejecutar aplicaciones e interactuar con el IDE, para usar este tipo de ejecución el dispositivo debe de estar configurado para permitir la depuración USB o inalámbrica por medio de una red WiFi, antes de iniciar la ejecución es necesario seleccionar un dispositivo físico en donde se instalará la aplicación requerida.

Figura 50. **Seleccionar dispositivo de ejecución Android Studio**



Fuente: elaboración propia, realizado con Paint3D.

5.2. Desarrollo aplicación Android de configuración y monitoreo energético

La aplicación de configuración y monitoreo está diseñada para funcionar en celulares y tabletas que cuenten con WiFi y Bluetooth porque este proyecto funciona sobre estas tecnologías, las distintas tareas demandadas por las

funciones de la aplicación se ejecutan por medio del recurso llamado *fragment,* dicho recurso tiene la capacidad de definir su propio diseño y cuenta con su propio ciclo de vida por estas razones se define un *fragment* por cada función permitiendo mostrar en la interfaz de usuario con un diseño acorde a la tarea solicitada.

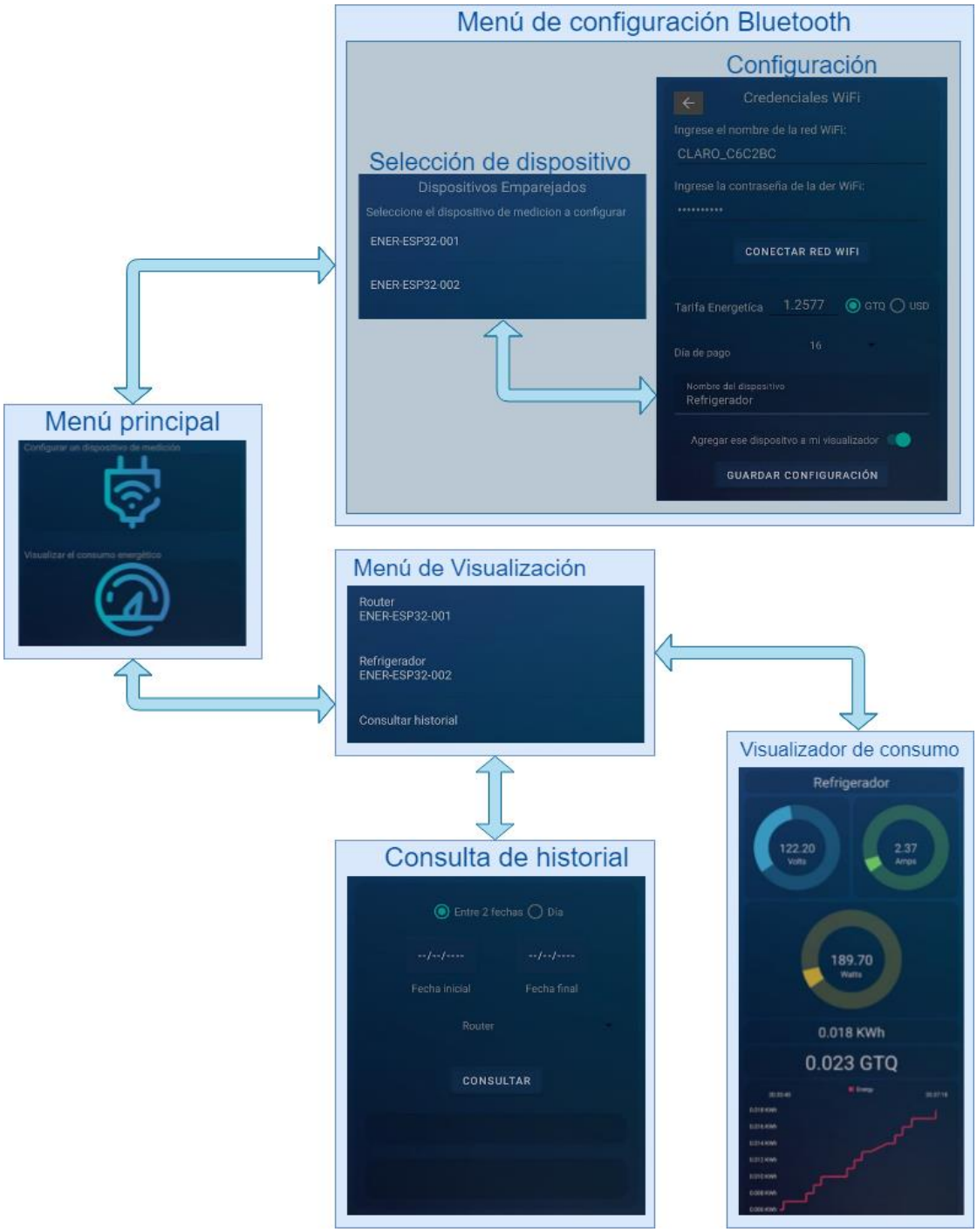
5.2.1. Diagrama de navegación

Al iniciar la aplicación el primer *fragment* mostrado es el correspondiente al menú principal el cual contiene las opciones para configurar un dispositivo de medición y visualizar el consumo energético, la primera opción sirve para guardar o modificar credenciales WiFi y datos energéticos en el dispositivo de medición usando una conexión Bluetooth, la segunda opción sirve para visualizar el consumo energético en tiempo real de todos los dispositivos energéticos asociados y también permite consultar el historial de consumo energético.

Si al momento de abrir la aplicación no se cuenta con una conexión a internet la opción para visualizar el consumo energético estará deshabilitada puesto que será imposible realizar las tareas solicitadas y solo se contará con la opción para realizar configuraciones.

Por medio del diagrama de navegación de la figura 51 se puede observar los diferentes menús y tareas representados por cada *fragment* de la aplicación, dicho diagrama también cuenta con indicadores en forma de flechas para trazar el flujo de navegación y conexiones existentes entre las tareas de cada función.

Figura 51. **Diagrama de navegación aplicación Android**



Fuente: elaboración propia, realizado con Draw.io.

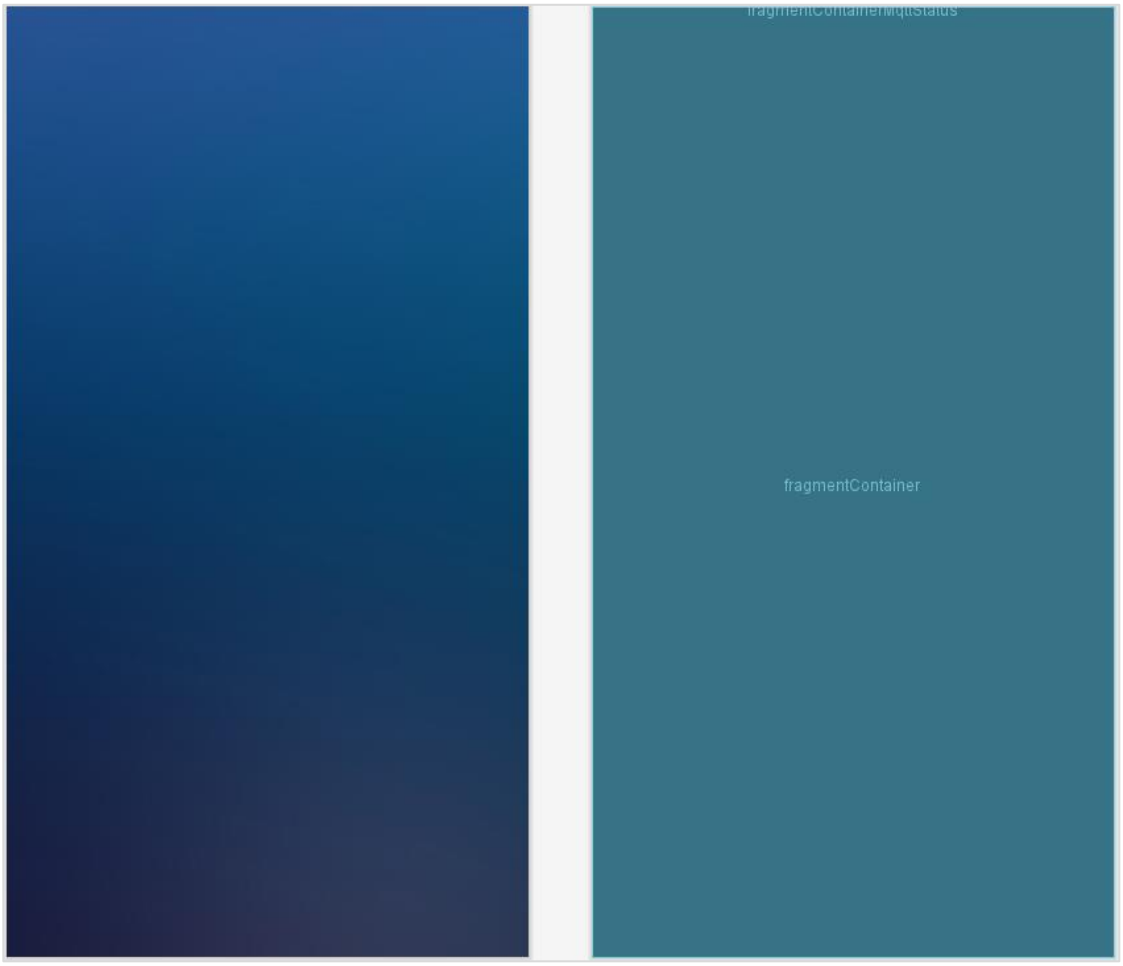
5.2.2. Archivos y código del proyecto

En esta sección se describe el funcionamiento de los diferentes *fragments* utilizados por la aplicación, se ilustran los archivos de diseño y código utilizados para realizar las tareas requeridas por el proyecto. Para el caso de *fragmetns* y MainActivity se muestra el código y el diseño de cada recurso, para clases e interfaces Java se muestra el código correspondiente.

5.2.2.1. Actividad principal (MainActivity)

La actividad principal es la encargada de administrar los diferentes *fragments* pertenecientes al proyecto, esta actividad está presente durante toda la ejecución de la aplicación y cuenta con dos contenedores de *fragments*, el primer contenedor alberga el *fragment* encargado de establecer comunicaciones MQTT, gráficamente cuenta con un indicador de conexión que únicamente es visible en el menú principal por lo que la mayoría del tiempo permanece oculto, el segundo contenedor funciona como visualizador y normalmente ocupa todo el espacio disponible en la pantalla, por medio de este se muestra al usuario las interfaces gráficas (*layouts*), pertenecientes a los menús y funciones presentes en la aplicación.

Figura 52. **Diseño y código XML MainActivity (App Android)**



activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imageViewError"
        android:layout_width="wrap_content"
```

Continuación de la figura 52.

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="@+id/fragmentContainer"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/fragmentContainerMqttStatus"
        app:srcCompat="@drawable/locked" />

    <FrameLayout
        android:id="@+id/fragmentContainer"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:visibility="visible"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

    </FrameLayout>

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/fragmentContainerMqttStatus"
        android:name="com.example.monitorenergtico.MQTTservice"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="5dp"
        android:layout_marginEnd="5dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Fuente: elaboración propia, realizado con captura de pantalla.

Figura 53. **Código Java MainActivity (App Android)**

MainActivity.java

```java
package com.example.monitorenergtico;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentContainerView;
import androidx.fragment.app.FragmentTransaction;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;

public class MainActivity extends AppCompatActivity implements Listener{

    FragmentTransaction transaction;
    Fragment fragmentLogin, fragmentNewAccout,fragmentOptions,fragmentConfigureDeviceBt,
             fragmentViewDevices,fragmentViewEnergyCons, fragmentConsumptionHistory;
    FragmentContainerView fragmentContainerMqttStatus;
    ImageView imgError;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        View decorView = getWindow().getDecorView();
        // Hide both the navigation bar and the status bar.
        // SYSTEM_UI_FLAG_FULLSCREEN is only available on Android 4.1 and higher, but as
        // a general rule, you should design your app to hide the status bar whenever you
        // hide the navigation bar.

        fragmentLogin = new Login();
        fragmentNewAccout = new NewUser();
        fragmentOptions = new Options();
        fragmentConfigureDeviceBt = new ConfigureDeviceBt();
        fragmentViewDevices = new ViewDevices();
        fragmentViewEnergyCons = new ViewEnergyConsumption();
        fragmentConsumptionHistory = new ConsumptionHistory();
        fragmentContainerMqttStatus = findViewById(R.id.fragmentContainerMqttStatus);
        imgError = findViewById(R.id.imageViewError);
    }

    @Override
    public void setConfDevice() {
        fragmentContainerMqttStatus.setVisibility(View.GONE);
        transaction=getSupportFragmentManager().beginTransaction();
        transaction.replace(R.id.fragmentContainer,fragmentConfigureDeviceBt);
        transaction.addToBackStack(null);
        transaction.commit();
    }
```

Continuación de la figura 53.

```
49
50      @Override
51      public void setViewDevices() {
52          fragmentContainerMqttStatus.setVisibility(View.GONE);
53          transaction=getSupportFragmentManager().beginTransaction();
54          transaction.replace(R.id.fragmentContainer,fragmentViewDevices);
55          transaction.addToBackStack(null);
56          transaction.commit();
57      }
58
59      @Override
60      public void setViewEnergyCons(String name, String topic) {
61          Bundle data = new Bundle();
62          data.putString("name",name);
63          data.putString("topic",topic);
64          fragmentViewEnergyCons.setArguments(data);
65          transaction=getSupportFragmentManager().beginTransaction();
66          transaction.replace(R.id.fragmentContainer,fragmentViewEnergyCons);
67          transaction.addToBackStack(null);
68          transaction.commit();
69      }
70
71      @Override
72      public void setConsHistory() {
73          transaction=getSupportFragmentManager().beginTransaction();
74          transaction.replace(R.id.fragmentContainer,fragmentConsumptionHistory);
75          transaction.addToBackStack(null);
76          transaction.commit();
77      }
78
79      @Override
80      public void setOptions() {
81          fragmentContainerMqttStatus.setVisibility(View.GONE);
82          transaction=getSupportFragmentManager().beginTransaction();
83          transaction.replace(R.id.fragmentContainer,fragmentOptions).commit();
84      }
85
86      @Override
87      public void InfConn(boolean isConnected) {
88          Bundle data = new Bundle();
89          data.putString("MqttCon",String.valueOf(isConnected));
90          fragmentOptions.setArguments(data);
91          transaction=getSupportFragmentManager().beginTransaction();
92          transaction.replace(R.id.fragmentContainer,fragmentOptions);
93          transaction.addToBackStack(null);
94          transaction.commit();
95      }
96  }
```

Fuente: elaboración propia, realizado con captura de pantalla.

5.2.2.2. *Fragment* MQTTservice

Al iniciar la aplicación es el encargado de realizar la conexión MQTT con el servidor y una vez establecida la conexión administrará e interpretará todos los mensajes salientes y entrantes, también realizar suscripciones a *topics* de interés y está presente (oculto la mayoría del tiempo), en MainActivity durante todo el tiempo de ejecución.

Figura 54. **Código XML *fragment* MQTTservice (App Android)**

fragment_m_q_t_tservice.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MQTTservice">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:id="@+id/textStatus"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="30dp"
        android:text="Status"
        android:textAlignment="center"
        android:textColor="#B1B1B1"
        android:textSize="16sp" />

</FrameLayout>
```

Fuente: elaboración propia, realizado con captura de pantalla.

Figura 55. **Código Java *fragment* MQTTservice (App Android)**

MQTTservice.java

```java
package com.example.monitorenergtico;

import android.content.Context;
import android.graphics.Color;
import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentResultListener;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import org.eclipse.paho.android.service.MqttAndroidClient;
import org.eclipse.paho.client.mqttv3.IMqttActionListener;
import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.IMqttToken;
import org.eclipse.paho.client.mqttv3.MqttCallback;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

public class MQTTservice extends Fragment {

    TextView status;
    SecureData secureData;
    MqttAndroidClient mqttClient;
    Listener listener;
    String deviceTopic=null,deviceTopicHis=null,dateHis=null;
    public MQTTservice() {
        // Required empty public constructor
    }

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

```

Continuación de la figura 55.

```
43          getParentFragmentManager().setFragmentResultListener( requestKey: "req_measur",
44                  lifecycleOwner: this, new FragmentResultListener() {
45              @Override
46              public void onFragmentResult(@NonNull String requestKey,
47                                           @NonNull Bundle bundle) {
48                  //save topic and subscribe
49                  String device = "energy/devices/"+bundle.getString( key: "topic");
50                  try {
51                      mqttClient.publish(device,"visualization".getBytes(),
52                              qos: 0, retained: false);
53                  } catch (MqttException e) {
54                      e.printStackTrace();
55                  }
56              }
57          });
58
59          getParentFragmentManager().setFragmentResultListener( requestKey: "topic",
60                  lifecycleOwner: this, new FragmentResultListener() {
61              @Override
62              public void onFragmentResult(@NonNull String requestKey,
63                                           @NonNull Bundle bundle) {
64                  //save topic and subscribe
65                  deviceTopic = "energy/devices/"+bundle.getString( key: "topic");
66                  setSubscription(deviceTopic+"/+");
67              }
68          });
69
70          getParentFragmentManager().setFragmentResultListener( requestKey: "unsuscribe",
71                  lifecycleOwner: this, new FragmentResultListener() {
72              @Override
73              public void onFragmentResult(@NonNull String requestKey,
74                                           @NonNull Bundle bundle) {
75                  if (!deviceTopic.equals(null)) {
76                      deleteSubscription( topic: deviceTopic + "/+");
77                      deviceTopic = null;
78                  }
79              }
80          });
81
82          getParentFragmentManager().setFragmentResultListener( requestKey: "consult",
83                  lifecycleOwner: this, new FragmentResultListener() {
84              @Override
85              public void onFragmentResult(@NonNull String requestKey,
86                                           @NonNull Bundle bundle) {
87                  deviceTopicHis = "energy/devices/" +bundle.getString( key: "device");
```

Continuación de la figura 55.

```
88                  setSubscription(deviceTopicHis);
89                  String dateIni = bundle.getString( key: "dateIni");
90                  String type = bundle.getString( key: "type");
91                  String name = bundle.getString( key: "name");
92                  String message="";
93                  if (type=="req-data-day") {
94                      message = type+'\n'+name+'\n'+dateIni;
95                      dateHis=dateIni;
96                  }else if(type=="req-data"){
97                      String dateFinal = bundle.getString( key: "dateFinal");
98                      message = type+'\n'+name+'\n'+dateIni+'\n'+dateFinal;
99                      dateHis=dateIni+"-"+dateFinal;
100                 }
101                 try {
102                     mqttClient.publish(deviceTopicHis,message.getBytes(),
103                             qos: 0, retained: false);
104                 } catch (MqttException e) {
105                     e.printStackTrace();
106                 }
107             }
108         });
109     }
110
111     @Override
112 @   public View onCreateView(LayoutInflater inflater, ViewGroup container,
113                             Bundle savedInstanceState) {
114         // Inflate the layout for this fragment
115         return inflater.inflate(R.layout.fragment_m_q_t_tservice, container,
116                 attachToRoot: false);
117     }
118
119     @Override
120     public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
121         super.onViewCreated(view, savedInstanceState);
122
123         secureData = new SecureData();
124         status = view.findViewById(R.id.textStatus);
125
126         connMqtt();
127     }
128
129     @Override
130     public void onAttach(@NonNull Context context) {
131         super.onAttach(context);
132         if (context instanceof Listener){
```

Continuación de la figura 55.

```
133             listener = (Listener) context;
134         }
135     }
136
137     public void connMqtt(){
138
139         String clientId = MqttClient.generateClientId();
140         mqttClient = new MqttAndroidClient(getActivity().getApplicationContext(),
141                                            getCredentialsMqtt()[0],
142                                            clientId);
143         MqttConnectOptions options = new MqttConnectOptions();
144         options.setUserName(getCredentialsMqtt()[1]);
145         options.setPassword(getCredentialsMqtt()[2].toCharArray());
146
147         try {
148             status.setText("Estableciendo conexion con el servidor...");
149             IMqttToken token = mqttClient.connect(options);
150             token.setActionCallback(new IMqttActionListener() {
151                 @Override
152                 public void onSuccess(IMqttToken asyncActionToken) {
153                     // We are connected
154                     status.setTextColor(Color.parseColor( colorString: "#32CD32"));
155                     status.setText("Conexion establecida");
156                     listener.InfConn( isConnected: true);
157                 }
158
159                 @Override
160                 public void onFailure(IMqttToken asyncActionToken, Throwable exception) {
161                     // Something went wrong e.g. connection timeout or firewall problems
162                     listener.InfConn( isConnected: false);
163                     status.setTextColor(Color.parseColor( colorString: "#F13A37"));
164                     status.setText("Error, no se pudo establecer la conexión con el " +
165                             "servidor, verifique su conexión a Internet. No se podra " +
166                             "visualizar el consumo energético");
167                     //Toast.makeText(getActivity(),"error",Toast.LENGTH_LONG).show();
168                 }
169             });
170         } catch (MqttException e) {
171             e.printStackTrace();
172         }
173
174         mqttClient.setCallback(new MqttCallback() {
175             @Override
176             public void connectionLost(Throwable cause) {
177
```

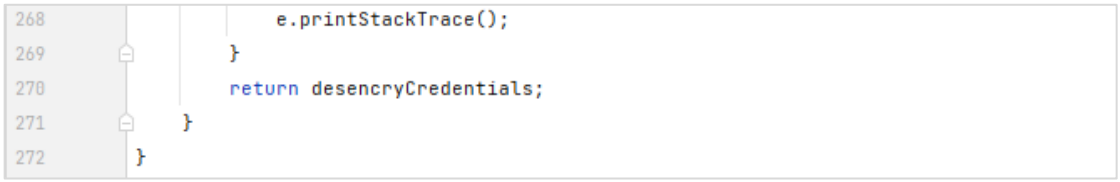
Continuación de la figura 55.

```
178             }
179
180             @Override
181             public void messageArrived(String topic, MqttMessage message)
182                     throws Exception {
183                 //subText.setText(new String(message.getPayload()));
184                 handMessage(topic,new String(message.getPayload()));
185             }
186
187             @Override
188             public void deliveryComplete(IMqttDeliveryToken token) {
189
190             }
191         });
192     }
193
194     private void setSubscription(String topic){
195         try{
196             mqttClient.subscribe(topic, qos: 0);
197         }catch (MqttException e){
198             e.printStackTrace();
199         }
200     }
201
202     private void deleteSubscription(String topic){
203         try{
204             mqttClient.unsubscribe(topic);
205         }catch (MqttException e){
206             e.printStackTrace();
207         }
208     }
209
210 @   public void handMessage(String topic, String msg){
211         String[] info = msg.split(String.valueOf('\n'));
212         if(topic.equals(deviceTopic+"/voltage")){
213             Bundle loginBundle = new Bundle();
214             loginBundle.putString("voltage",info[0]);
215             getParentFragmentManager().setFragmentResult( requestKey: "voltage",loginBundle);
216         } else if (topic.equals(deviceTopic+"/current")){
217             Bundle loginBundle = new Bundle();
218             loginBundle.putString("current",info[0]);
219             getParentFragmentManager().setFragmentResult( requestKey: "current",loginBundle);
220         } else if (topic.equals(deviceTopic+"/power")){
221             Bundle loginBundle = new Bundle();
222             loginBundle.putString("power",info[0]);
```

Continuación de la figura 55.

```
223             getParentFragmentManager().setFragmentResult( requestKey: "power",loginBundle);
224         } else if (topic.equals(deviceTopic+"/energy")){
225             Bundle loginBundle = new Bundle();
226             loginBundle.putString("energy",info[0]);
227             getParentFragmentManager().setFragmentResult( requestKey: "energy",loginBundle);
228         } else if (topic.equals(deviceTopic+"/charge")){
229             Bundle loginBundle = new Bundle();
230             loginBundle.putString("charge",info[0]);
231             getParentFragmentManager().setFragmentResult( requestKey: "charge",loginBundle);
232         }
233         if(topic.equals(deviceTopicHis) && info[0].equals("res-data") &&
234                 info[1].equals(dateHis)){
235             dateHis=null;
236             Bundle loginBundle = new Bundle();
237             loginBundle.putString("energy",info[2]);
238             loginBundle.putString("charge",info[3]);
239             getParentFragmentManager().setFragmentResult( requestKey: "history",loginBundle);
240         }
241     }
242
243     private String[] getCredentialsMqtt(){
244         String[] encryCredentials;
245         String[] desencryCredentials = {"","","",""};
246         InputStream mqtt_credentials =
247                 this.getResources().openRawResource(R.raw.mqtt_credentials);
248         BufferedReader reader =
249                 new BufferedReader(new InputStreamReader(mqtt_credentials));
250         String line,info="";
251         try {
252             while ((line = reader.readLine()) != null) {
253                 info=info+line+'\n';
254             }
255             reader.close();
256         } catch (IOException e) {
257             e.printStackTrace();
258         }
259         encryCredentials = info.split(String.valueOf('\n'));
260         try {
261             desencryCredentials[0]=
262                     secureData.desencrypt(encryCredentials[0],encryCredentials[3]);
263             desencryCredentials[1]=
264                     secureData.desencrypt(encryCredentials[1],encryCredentials[3]);
265             desencryCredentials[2]=
266                     secureData.desencrypt(encryCredentials[2],encryCredentials[3]);
267         }catch (Exception e){
```

Continuación de la figura 55.

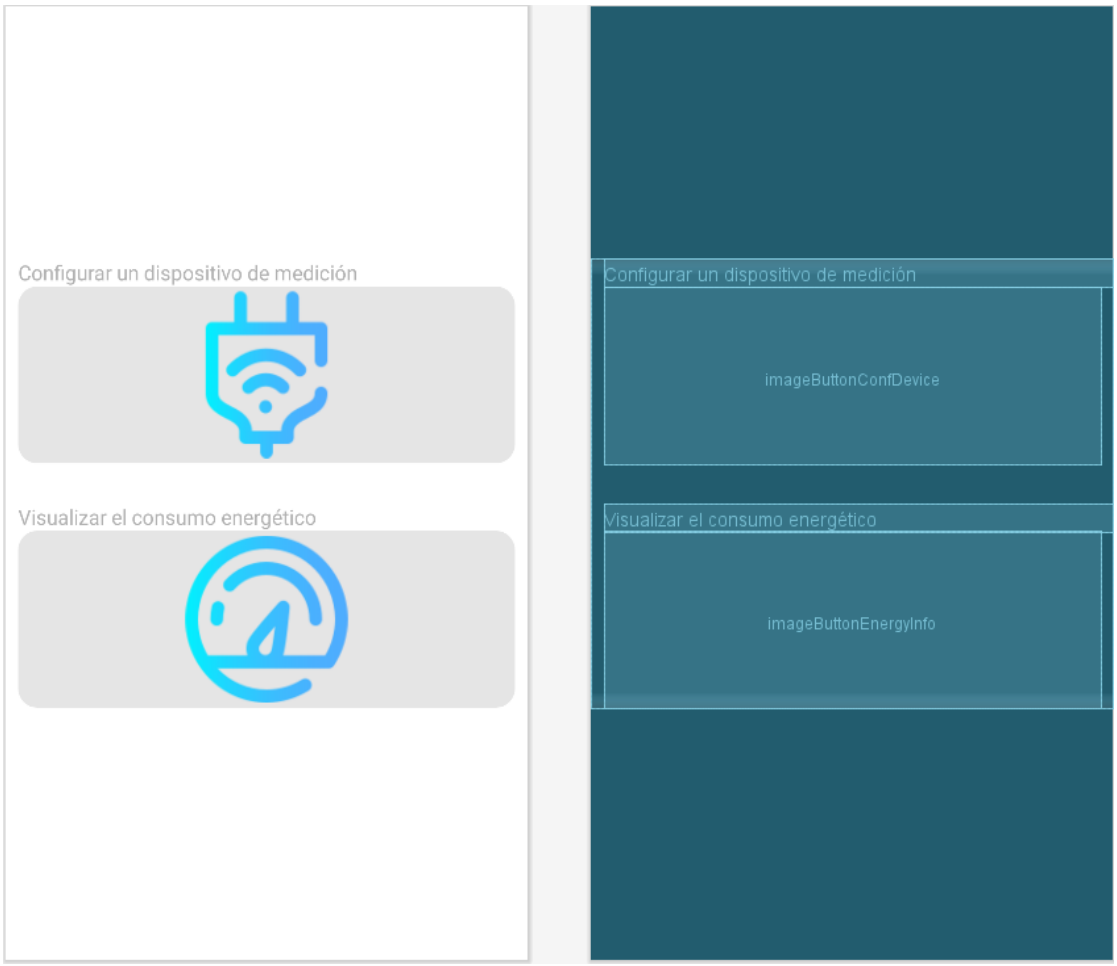


Fuente: elaboración propia, realizado con captura de pantalla.

5.2.2.3. *Fragment* Options

Funciona como menú principal porque es el primer *fragment* visto por el usuario al iniciar la aplicación, cuenta con las opciones para configurar un dispositivo de medición o visualizar el consumo energético (disponible si la conexión MQTT fue exitosa), cuando el usuario selecciona una opción se envía la instrucción correspondiente a MainActivity con el propósito de reemplazar el *fragment* actual (menú principal) por el requerido por el usuario.

Figura 56. **Diseño y código XML *fragment* Options (App Android)**



fragment_options.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:orientation="vertical"
    tools:context=".Options">

    <TextView
        android:id="@+id/textView4"
```

Continuación de la figura 56.

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:text="Configurar un dispositivo de medición"
        android:textColor="#B1B1B1"
        android:textSize="16sp" />

    <ImageButton
        android:id="@+id/imageButtonConfDevice"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:background="@drawable/round_corners"
        android:backgroundTint="#80000000"
        android:src="@drawable/socket"
        tools:ignore="SpeakableTextPresentCheck" />

    <TextView
        android:id="@+id/textViewEnergyInfo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="30dp"
        android:text="Visualizar el consumo energético"
        android:textColor="#B1B1B1"
        android:textSize="16sp" />

    <ImageButton
        android:id="@+id/imageButtonEnergyInfo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:background="@drawable/round_corners"
        android:backgroundTint="#80000000"
        android:src="@drawable/speedometer"
        tools:ignore="SpeakableTextPresentCheck" />

</LinearLayout>
```

Fuente: elaboración propia, realizado con captura de pantalla.

Figura 57. **Código Java *fragment* Options (App Android)**

Options.java

```java
package com.example.monitorenergtico;

import android.content.Context;
import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageButton;
import android.widget.TextView;

/**
 * A simple {@link Fragment} subclass.
 * Use the {@link Options#newInstance} factory method to
 * create an instance of this fragment.
 */
public class Options extends Fragment {

    Listener listener;
    ImageButton imgBtnConfDevice,imgBtnEnergyInfo;
    TextView textViewEnergyInfo;

    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "MqttCon";

    // TODO: Rename and change types of parameters
    private String mMqttCon;

    public Options() {
        // Required empty public constructor
    }

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param MqttCon Parameter 1.
     * @return A new instance of fragment Options.
     */
    // TODO: Rename and change types and number of parameters
    public static Options newInstance(String MqttCon) {
        Options fragment = new Options();
```

Continuación de la figura 57.

```
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, MqttCon);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mMqttCon = getArguments().getString(ARG_PARAM1);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_options, container, attachToRoot: false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        imgBtnConfDevice = view.findViewById(R.id.imageButtonConfDevice);
        imgBtnEnergyInfo = view.findViewById(R.id.imageButtonEnergyInfo);
        textViewEnergyInfo = view.findViewById(R.id.textViewEnergyInfo);

        if (!Boolean.parseBoolean(mMqttCon)){
            imgBtnEnergyInfo.setVisibility(view.GONE);
            textViewEnergyInfo.setVisibility(view.GONE);
        }
        imgBtnConfDevice.setOnClickListener(v -> listener.setConfDevice());
        imgBtnEnergyInfo.setOnClickListener(v -> listener.setViewDevices());
    }

    @Override
    public void onAttach(@NonNull Context context) {
        super.onAttach(context);
        if (context instanceof Listener){
            listener = (Listener) context;
        }
    }
}
```

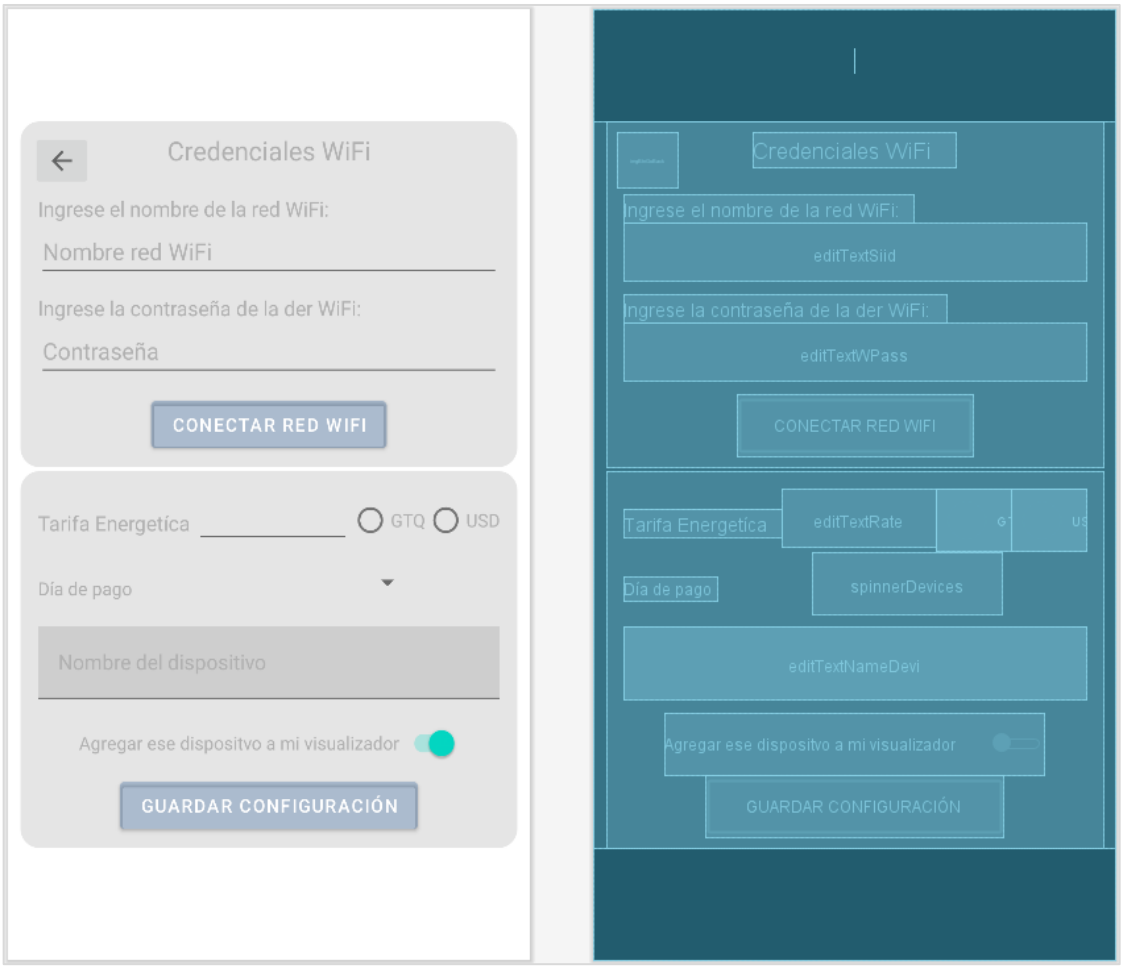
Fuente: elaboración propia, realizado con captura de pantalla.

5.2.2.4. *Fragment* ConfigureDeviceBt

Este *fragment* hace uso de las funciones Bluetooth del dispositivo Android con el propósito de establecer conexiones, enviar y recibir información a través de un socket Bluetooth y acceder a la lista de los dispositivos emparejados, el primer recurso gráfico mostrado es una lista de los dispositivos bluetooth emparejados aplicando un filtro se muestran solo los elementos que correspondan a un dispositivo de medición de este monitor energético.

Esta lista es mostrada con el propósito de seleccionar un dispositivo de medición para el cual se realizará una configuración, cuando ya se realizó la selección del dispositivo de medición a configurar se procede a ocultar la lista anteriormente mencionada y se muestra un menú de configuración que contiene campos para introducir credenciales WiFi y datos energéticos.

Figura 58. **Diseño y código XML *fragment* ConfigureDeviceBt (App Android)**



fragment_configure_device_bt.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      tools:context=".ConfigureDeviceBt">
9
10     <androidx.constraintlayout.widget.ConstraintLayout
```

Continuación de la figura 58.

```
        android:id="@+id/ConstraintLayoutBtOk"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:id="@+id/constraintLayoutConf"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginStart="10dp"
            android:layout_marginTop="3dp"
            android:layout_marginEnd="10dp"
            android:background="@drawable/round_corners"
            android:backgroundTint="#80000000"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/constraintLayoutWifi">

            <TextView
                android:id="@+id/textView14"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="10dp"
                android:layout_marginTop="25dp"
                android:text="Tarifa Energetica "
                android:textColor="#B1B1B1"
                android:textSize="16sp"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent" />

            <RadioGroup
                android:id="@+id/radioGroup"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="10dp"
                android:layout_marginEnd="10dp"
                android:orientation="horizontal"
                app:layout_constraintEnd_toEndOf="parent"
```

Continuación de la figura 58.

```
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toTopOf="parent">

            <RadioButton
                android:id="@+id/radioButtonGTQ"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="GTQ"
                android:textColor="#B1B1B1" />

            <RadioButton
                android:id="@+id/radioButtonUSD"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="USD"
                android:textColor="#B1B1B1" />
        </RadioGroup>

        <EditText
            android:id="@+id/editTextRate"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:ems="10"
            android:inputType="numberDecimal"
            android:textAlignment="center"
            android:textColor="#B1B1B1"
            android:textColorHint="#B1B1B1"
            app:layout_constraintEnd_toStartOf="@+id/radioGroup"
            app:layout_constraintStart_toEndOf="@+id/textView14"
            app:layout_constraintTop_toTopOf="parent"
            tools:ignore="SpeakableTextPresentCheck,TouchTargetSizeCheck" />

        <TextView
            android:id="@+id/textView15"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="10dp"
            android:layout_marginBottom="10dp"
            android:text="Día de pago"
            android:textColor="#B1B1B1"
            app:layout_constraintBottom_toBottomOf="@+id/spinnerDevices"
            app:layout_constraintStart_toStartOf="parent" />

```

Continuación de la figura 58.

```
98      <Spinner
99          android:id="@+id/spinnerDevices"
100         android:layout_width="150dp"
101         android:layout_height="wrap_content"
102         android:minHeight="48dp"
103         android:popupBackground="@color/colorBackgroundSpinner"
104         app:layout_constraintEnd_toEndOf="parent"
105         app:layout_constraintStart_toEndOf="@+id/textView15"
106         app:layout_constraintTop_toBottomOf="@+id/radioGroup"
107         tools:ignore="SpeakableTextPresentCheck" />
108
109     <com.google.android.material.textfield.TextInputLayout
110         android:id="@+id/textInputLayout"
111         android:layout_width="0dp"
112         android:layout_height="wrap_content"
113         android:layout_marginStart="10dp"
114         android:layout_marginTop="10dp"
115         android:layout_marginEnd="10dp"
116         android:textColorHint="#B1B1B1"
117         app:hintTextColor="#B1B1B1"
118         app:layout_constraintEnd_toEndOf="parent"
119         app:layout_constraintStart_toStartOf="parent"
120         app:layout_constraintTop_toBottomOf="@+id/spinnerDevices">
121
122         <com.google.android.material.textfield.TextInputEditText
123             android:id="@+id/editTextNameDevi"
124             android:layout_width="match_parent"
125             android:layout_height="wrap_content"
126             android:background="#1A000000"
127             android:hint="Nombre del dispositivo"
128             android:textColor="#B1B1B1" />
129     </com.google.android.material.textfield.TextInputLayout>
130
131     <Switch
132         android:id="@+id/switchAddDevi"
133         android:layout_width="wrap_content"
134         android:layout_height="wrap_content"
135         android:layout_marginTop="10dp"
136         android:checked="true"
137         android:minHeight="48dp"
138         android:text="Agregar ese dispositvo a mi visualizador"
139         android:textColor="#B1B1B1"
140         app:layout_constraintEnd_toEndOf="parent"
141         app:layout_constraintHorizontal_bias="0.5"
```

Continuación de la figura 58.

```
142             app:layout_constraintStart_toStartOf="parent"
143             app:layout_constraintTop_toBottomOf="@+id/textInputLayout" />
144
145         <Button
146             android:id="@+id/btnSaveConf"
147             android:layout_width="wrap_content"
148             android:layout_height="wrap_content"
149             android:layout_marginBottom="5dp"
150             android:text="Guardar Configuración"
151             app:layout_constraintBottom_toBottomOf="parent"
152             app:layout_constraintEnd_toEndOf="parent"
153             app:layout_constraintHorizontal_bias="0.5"
154             app:layout_constraintStart_toStartOf="parent"
155             app:layout_constraintTop_toBottomOf="@+id/switchAddDevi" />
156
157     </androidx.constraintlayout.widget.ConstraintLayout>
158
159     <androidx.constraintlayout.widget.ConstraintLayout
160         android:id="@+id/constraintLayoutWifi"
161         android:layout_width="match_parent"
162         android:layout_height="wrap_content"
163         android:layout_marginStart="10dp"
164         android:layout_marginEnd="10dp"
165         android:background="@drawable/round_corners"
166         android:backgroundTint="#80000000"
167         app:layout_constraintEnd_toEndOf="parent"
168         app:layout_constraintStart_toStartOf="parent"
169         app:layout_constraintTop_toTopOf="parent">
170
171         <TextView
172             android:id="@+id/textView16"
173             android:layout_width="wrap_content"
174             android:layout_height="wrap_content"
175             android:layout_marginTop="5dp"
176             android:text="Credenciales WiFi"
177             android:textColor="#B1B1B1"
178             android:textSize="20sp"
179             app:layout_constraintEnd_toEndOf="parent"
180             app:layout_constraintStart_toStartOf="parent"
181             app:layout_constraintTop_toTopOf="parent" />
182
183         <TextView
184             android:id="@+id/textView12"
185             android:layout_width="wrap_content"
```

Continuación de la figura 58.

```
186             android:layout_height="wrap_content"
187             android:layout_marginStart="10dp"
188             android:layout_marginTop="20dp"
189             android:text="Ingrese el nombre de la red WiFi:"
190             android:textColor="#B1B1B1"
191             android:textSize="16sp"
192             app:layout_constraintStart_toStartOf="parent"
193             app:layout_constraintTop_toBottomOf="@+id/textView16" />
194
195         <EditText
196             android:id="@+id/editTextSiid"
197             android:layout_width="0dp"
198             android:layout_height="wrap_content"
199             android:layout_marginStart="10dp"
200             android:layout_marginEnd="10dp"
201             android:ems="10"
202             android:hint="Nombre red WiFi"
203             android:inputType="textPersonName"
204             android:textColor="#B1B1B1"
205             android:textColorHint="#B1B1B1"
206             app:layout_constraintEnd_toEndOf="parent"
207             app:layout_constraintStart_toStartOf="parent"
208             app:layout_constraintTop_toBottomOf="@+id/textView12"
209             tools:ignore="TouchTargetSizeCheck" />
210
211         <TextView
212             android:id="@+id/textView13"
213             android:layout_width="wrap_content"
214             android:layout_height="wrap_content"
215             android:layout_marginStart="10dp"
216             android:layout_marginTop="10dp"
217             android:text="Ingrese la contraseña de la der WiFi:"
218             android:textColor="#B1B1B1"
219             android:textSize="16sp"
220             app:layout_constraintStart_toStartOf="parent"
221             app:layout_constraintTop_toBottomOf="@+id/editTextSiid" />
222
223         <EditText
224             android:id="@+id/editTextWPass"
225             android:layout_width="0dp"
226             android:layout_height="wrap_content"
227             android:layout_marginStart="10dp"
228             android:layout_marginEnd="10dp"
```

Continuación de la figura 58.

```
229                android:ems="10"
230                android:hint="Contraseña"
231                android:inputType="textPassword"
232                android:textColor="#B1B1B1"
233                android:textColorHint="#B1B1B1"
234                app:layout_constraintEnd_toEndOf="parent"
235                app:layout_constraintStart_toStartOf="parent"
236                app:layout_constraintTop_toBottomOf="@+id/textView13"
237                tools:ignore="TouchTargetSizeCheck" />
238
239            <Button
240                android:id="@+id/btnConnW"
241                android:layout_width="wrap_content"
242                android:layout_height="wrap_content"
243                android:layout_marginTop="10dp"
244                android:layout_marginBottom="5dp"
245                android:text="Conectar red WiFi"
246                app:layout_constraintBottom_toBottomOf="parent"
247                app:layout_constraintEnd_toEndOf="parent"
248                app:layout_constraintHorizontal_bias="0.5"
249                app:layout_constraintStart_toStartOf="parent"
250                app:layout_constraintTop_toBottomOf="@+id/editTextWPass" />
251
252            <ImageButton
253                android:id="@+id/imgBtnGoBack"
254                android:layout_width="wrap_content"
255                android:layout_height="wrap_content"
256                android:layout_marginStart="5dp"
257                android:layout_marginTop="5dp"
258                android:src="?attr/actionModeCloseDrawable"
259                app:layout_constraintStart_toStartOf="parent"
260                app:layout_constraintTop_toTopOf="parent"
261                tools:ignore="TouchTargetSizeCheck,SpeakableTextPresentCheck" />
262        </androidx.constraintlayout.widget.ConstraintLayout>
263    </androidx.constraintlayout.widget.ConstraintLayout>
264
265    <androidx.constraintlayout.widget.ConstraintLayout
266        android:id="@+id/ConstraintLayoutConnBt"
267        android:layout_width="0dp"
268        android:layout_height="match_parent"
269        android:layout_marginTop="30dp"
270        android:visibility="gone"
271        app:layout_constraintBottom_toBottomOf="parent"
```

Continuación de la figura 58.

```
272         app:layout_constraintEnd_toEndOf="parent"
273         app:layout_constraintHorizontal_bias="0.5"
274         app:layout_constraintStart_toStartOf="parent"
275         app:layout_constraintTop_toTopOf="parent">
276
277         <TextView
278             android:id="@+id/textView10"
279             android:layout_width="wrap_content"
280             android:layout_height="wrap_content"
281             android:text="Dispositivos Emparejados"
282             android:textSize="20sp"
283             app:layout_constraintEnd_toEndOf="parent"
284             app:layout_constraintHorizontal_bias="0.5"
285             app:layout_constraintStart_toStartOf="parent"
286             app:layout_constraintTop_toTopOf="parent" />
287
288         <TextView
289             android:id="@+id/textView11"
290             android:layout_width="0dp"
291             android:layout_height="wrap_content"
292             android:layout_marginStart="10dp"
293             android:layout_marginTop="10dp"
294             android:layout_marginEnd="10dp"
295             android:text="Seleccione el dispositivo de medicion a configurar"
296             android:textSize="16sp"
297             app:layout_constraintEnd_toEndOf="parent"
298             app:layout_constraintStart_toStartOf="parent"
299             app:layout_constraintTop_toBottomOf="@+id/textView10" />
300
301         <ListView
302             android:id="@+id/listViewPairedDevi"
303             android:layout_width="0dp"
304             android:layout_height="0dp"
305             app:layout_constraintBottom_toBottomOf="parent"
306             app:layout_constraintEnd_toEndOf="parent"
307             app:layout_constraintStart_toStartOf="parent"
308             app:layout_constraintTop_toBottomOf="@+id/textView11" />
309
310     </androidx.constraintlayout.widget.ConstraintLayout>
311
312     <androidx.constraintlayout.widget.ConstraintLayout
313         android:layout_width="wrap_content"
314         android:layout_height="wrap_content"
```

Continuación de la figura 58.

```
315          android:layout_marginTop="30dp"
316          app:layout_constraintEnd_toEndOf="parent"
317          app:layout_constraintHorizontal_bias="0.5"
318          app:layout_constraintStart_toStartOf="parent"
319          app:layout_constraintTop_toTopOf="parent">
320
321          <TextView
322              android:id="@+id/textViewStatusWifi"
323              android:layout_width="wrap_content"
324              android:layout_height="wrap_content"
325              android:textAlignment="center"
326              app:layout_constraintEnd_toEndOf="parent"
327              app:layout_constraintTop_toTopOf="parent" />
328
329          <ImageView
330              android:id="@+id/imageViewStatusWifi"
331              android:layout_width="wrap_content"
332              android:layout_height="wrap_content"
333              android:layout_marginTop="2dp"
334              app:layout_constraintEnd_toStartOf="@+id/textViewStatusWifi"
335              app:layout_constraintHorizontal_bias="1.0"
336              app:layout_constraintStart_toStartOf="parent"
337              app:layout_constraintTop_toTopOf="parent" />
338      </androidx.constraintlayout.widget.ConstraintLayout>
339  </androidx.constraintlayout.widget.ConstraintLayout>
```

Fuente: elaboración propia, realizado con captura de pantalla.

Figura 59. **Código Java *fragment* ConfigureDeviceBt (App Android)**

ConfigureDeviceBt.java

```
package com.example.monitorenergtico;

import android.app.ProgressDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.AsyncTask;
import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.RadioButton;
import android.widget.Spinner;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;
import java.util.TimeZone;
import java.util.UUID;

public class ConfigureDeviceBt extends Fragment {

    Listener listener;
```

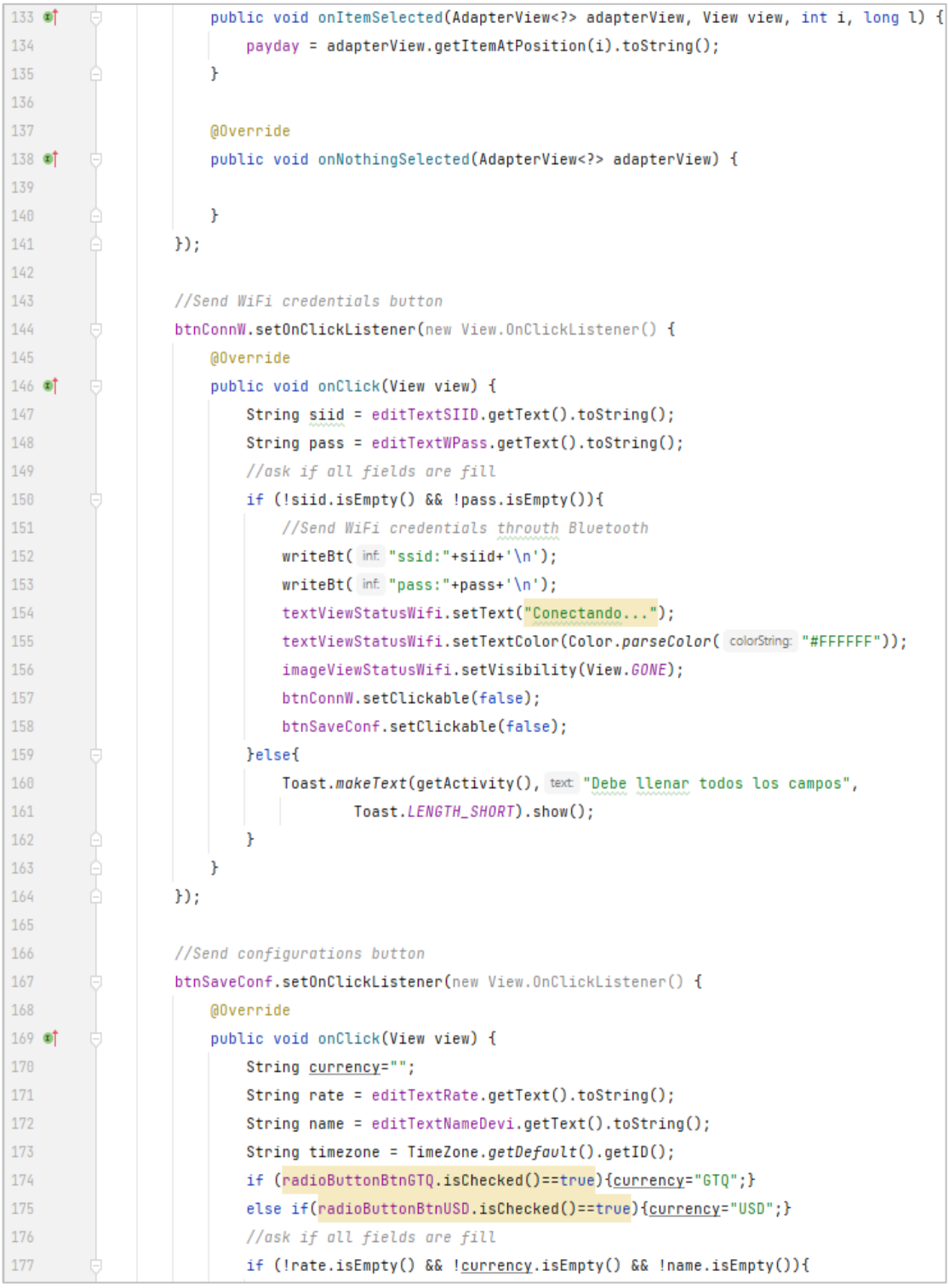
Continuación de la figura 59.

```
43
44          ConstraintLayout constraintLayoutConnBt, constraintLayoutBtOk;
45          ListView listViewPairedDevices;
46
47          private BluetoothAdapter myBluetooth = null;
48          private Set<BluetoothDevice> pairedDevices;
49          BluetoothSocket btSocket = null;
50          private boolean isBtConnected = false;
51          static final UUID myUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
52          String device = null, address = null;
53
54          private ProgressDialog progress;
55
56          EditText editTextSIID, editTextWPass, editTextRate, editTextNameDevi;
57          RadioButton radioButtonBtnGTQ, radioButtonBtnUSD;
58          Spinner SpiPayday;
59          String payday="1";
60          Switch switchAddDevi;
61          Button btnConnW, btnSaveConf;
62          ImageButton imgBtnGoBack;
63          TextView textViewStatusWifi;
64          ImageView imageViewStatusWifi;
65          String msgBt="";
66          String confMeasDevi[];
67
68          public ConfigureDeviceBt() {
69              // Required empty public constructor
70          }
71
72          @Override
73          public void onCreate(Bundle savedInstanceState) {
74              super.onCreate(savedInstanceState);
75
76          }
77
78          @Override
79          public View onCreateView(LayoutInflater inflater, ViewGroup container,
80                                   Bundle savedInstanceState) {
81              // Inflate the layout for this fragment
82              return inflater.inflate(R.layout.fragment_configure_device_bt,
83                      container,  attachToRoot: false);
84          }
85
86          @Override
87          public void onDestroyView() {
```

Continuación de la figura 59.

```
88          super.onDestroyView();
89          if (isBtConnected) {
90              isBtConnected = false;
91              listViewPairedDevices.setAdapter(null);
92              try {
93                  btSocket.close();
94              } catch (IOException e) {
95                  e.printStackTrace();
96              }
97          }
98      }
99
100     @Override
101     public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
102         super.onViewCreated(view, savedInstanceState);
103
104         editTextSIID = view.findViewById(R.id.editTextSiid);
105         editTextWPass = view.findViewById(R.id.editTextWPass);
106         editTextRate = view.findViewById(R.id.editTextRate);
107         editTextNameDevi = view.findViewById(R.id.editTextNameDevi);
108         radioButtonBtnGTQ = view.findViewById(R.id.radioButtonGTQ);
109         radioButtonBtnUSD = view.findViewById(R.id.radioButtonUSD);
110         SpiPayday = view.findViewById(R.id.spinnerDevices);
111         switchAddDevi = view.findViewById(R.id.switchAddDevi);
112         btnConnW = view.findViewById(R.id.btnConnW);
113         btnSaveConf = view.findViewById(R.id.btnSaveConf);
114         imgBtnGoBack = view.findViewById(R.id.imgBtnGoBack);
115         imageViewStatusWifi = view.findViewById(R.id.imageViewStatusWifi);
116         textViewStatusWifi = view.findViewById(R.id.textViewStatusWifi);
117
118         constraintLayoutConnBt = view.findViewById(R.id.ConstraintLayoutConnBt);
119         constraintLayoutBtOk = view.findViewById(R.id.ConstraintLayoutBtOk);
120         listViewPairedDevices = view.findViewById(R.id.listViewPairedDevi);
121
122         //Show paired devices to connect
123         constraintLayoutConnBt.setVisibility(View.VISIBLE);    //hide paired devices list
124         constraintLayoutBtOk.setVisibility(View.GONE);  //show configuration layout
125
126         //Fill payday spinner
127         ArrayAdapter<CharSequence> adapterSpi = ArrayAdapter.createFromResource(getActivity(),
128                 R.array.days, R.layout.spinner_item);
129         adapterSpi.setDropDownViewResource(R.layout.spinner_item);
130         SpiPayday.setAdapter(adapterSpi);
131         SpiPayday.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
132             @Override
```

Continuación de la figura 59.



```
        public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
            payday = adapterView.getItemAtPosition(i).toString();
        }

        @Override
        public void onNothingSelected(AdapterView<?> adapterView) {

        }
    });

    //Send WiFi credentials button
    btnConnW.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String siid = editTextSIID.getText().toString();
            String pass = editTextWPass.getText().toString();
            //ask if all fields are fill
            if (!siid.isEmpty() && !pass.isEmpty()){
                //Send WiFi credentials throuth Bluetooth
                writeBt( inf: "ssid:"+siid+'\n');
                writeBt( inf: "pass:"+pass+'\n');
                textViewStatusWifi.setText("Conectando...");
                textViewStatusWifi.setTextColor(Color.parseColor( colorString: "#FFFFFF"));
                imageViewStatusWifi.setVisibility(View.GONE);
                btnConnW.setClickable(false);
                btnSaveConf.setClickable(false);
            }else{
                Toast.makeText(getActivity(), text: "Debe llenar todos los campos",
                        Toast.LENGTH_SHORT).show();
            }
        }
    });

    //Send configurations button
    btnSaveConf.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String currency="";
            String rate = editTextRate.getText().toString();
            String name = editTextNameDevi.getText().toString();
            String timezone = TimeZone.getDefault().getID();
            if (radioButtonBtnGTQ.isChecked()==true){currency="GTQ";}
            else if(radioButtonBtnUSD.isChecked()==true){currency="USD";}
            //ask if all fields are fill
            if (!rate.isEmpty() && !currency.isEmpty() && !name.isEmpty()){
```

Continuación de la figura 59.

```
178                    //Send all configurations throuth Bluetooth
179                    writeBt( inf: "rate:"+rate+'\n');
180                    writeBt( inf: "currency:"+currency+'\n');
181                    writeBt( inf: "payday:"+payday+'\n');
182                    writeBt( inf: "name:"+name+'\n');
183                    writeBt( inf: "timezone:"+timezone+'\n');
184                    //Save this device in app
185                    if (switchAddDevi.isChecked()){
186                        AddDevice(name+'\n'+device.trim()+'\n'+address);
187                    }else{
188                        ErraseDevice(name+'\n'+device.trim()+'\n'+address);
189                    }
190                }else{
191                    Toast.makeText(getActivity(), text: "Debe llenar todos los campos",
192                            Toast.LENGTH_SHORT).show();
193                }
194            }
195        });
196
197        imgBtnGoBack.setOnClickListener(v -> listener.setOptions());
198
199        startBt();
200
201    }
202
203    @Override
204    public void onAttach(@NonNull Context context) {
205        super.onAttach(context);
206        if (context instanceof Listener){
207            listener = (Listener) context;
208        }
209    }
210
211    public void startBt(){
212        ArrayList Devices = new ArrayList();
213        ArrayList AddressDevices = new ArrayList();
214
215        myBluetooth = BluetoothAdapter.getDefaultAdapter();
216
217        if(!myBluetooth.isEnabled())
218        {
219            //Ask to the user turn the bluetooth on
220            Intent turnBTon = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
221            startActivityForResult(turnBTon, requestCode: 1);
222        }
```

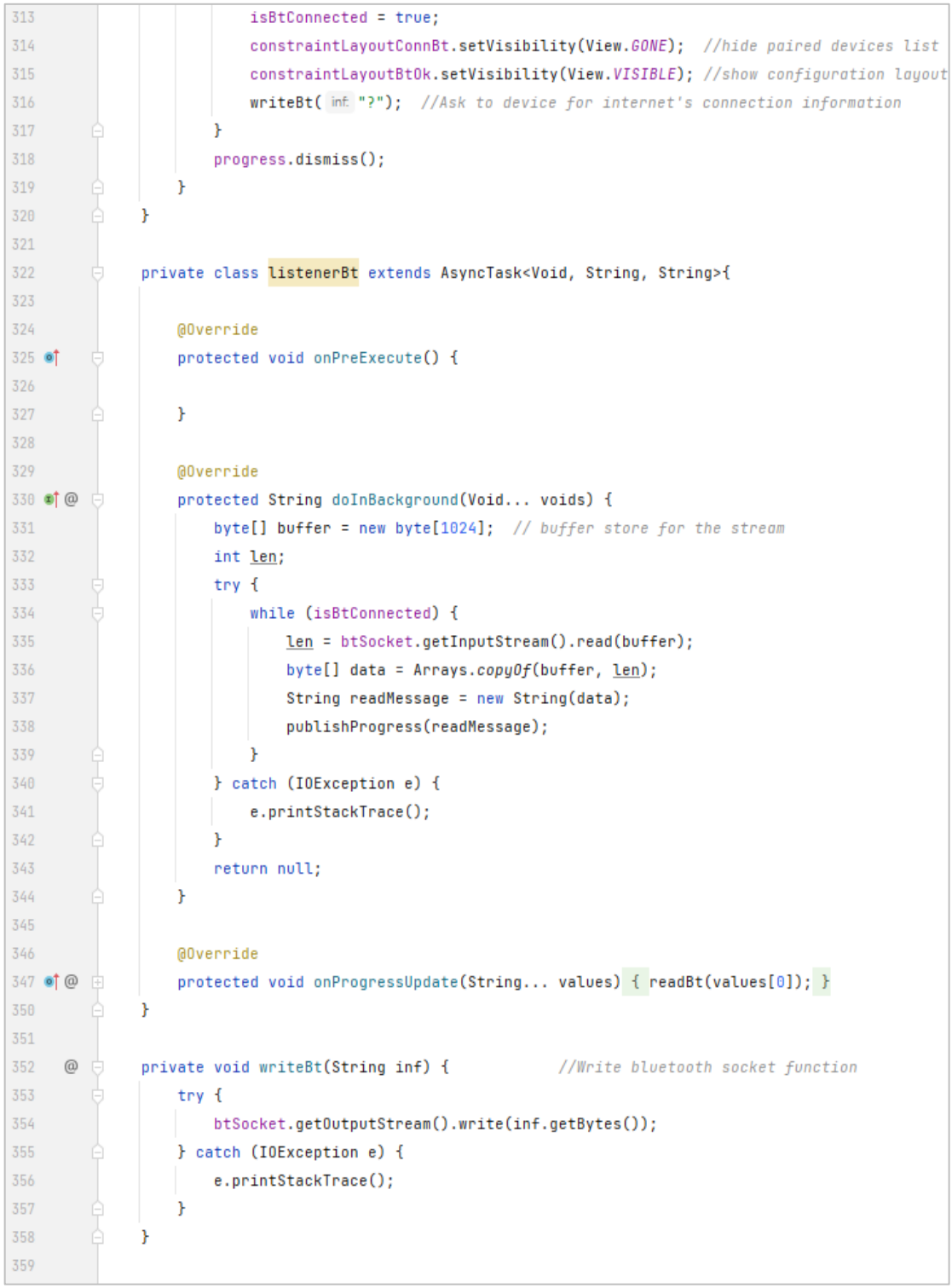
Continuación de la figura 59.

```
223
224         pairedDevices = myBluetooth.getBondedDevices();
225
226         if (pairedDevices.size()>0)
227         {
228             for(BluetoothDevice bt : pairedDevices)
229             {   //Show only measurement devices
230                 if (bt.getName().length()>=14) {
231                     if (bt.getName().substring(0,11).equals("ENER-ESP32-")){
232                         Devices.add("\n"+ bt.getName() + "\n"); //Save name devices
233                         AddressDevices.add(bt.getAddress());      //Save MAC address devices
234                     }
235                 }
236             }
237         }
238         else
239         {
240             Toast.makeText(getActivity(),
241                     text: "Primero debe emparejar el dispositivo a configurar",
242                     Toast.LENGTH_LONG).show();
243         }
244
245         final ArrayAdapter adapter = new ArrayAdapter(getActivity(),
246                 android.R.layout.simple_list_item_1, Devices);
247         listViewPairedDevices.setAdapter(adapter);
248         listViewPairedDevices.setOnItemClickListener(new AdapterView.OnItemClickListener() {
249             @Override
250             public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
251                 address = AddressDevices.get(i).toString();
252                 device = Devices.get(i).toString();
253                 new ConnectBT().execute(); //Call the class to connect
254             }
255         });
256     }
257
258     private class ConnectBT extends AsyncTask<Void, Void, Void>  // UI thread
259     {
260         private boolean ConnectSuccess = true; //if it's here, it's almost connected
261
262         @Override
263         protected void onPreExecute()
264         {
265             //show a progress dialog
266             progress = ProgressDialog.show(getActivity(),  title: "Conectando...",
267                     message: "Espere mientras se establece la conexón");
```

Continuación de la figura 59.

```
268         }
269
270         //while the progress dialog is shown, the connection is done in background
271         @Override
272         protected Void doInBackground(Void... devices)
273         {
274             try
275             {
276                 if (btSocket == null || !isBtConnected)
277                 {
278                     //get the mobile bluetooth device
279                     myBluetooth = BluetoothAdapter.getDefaultAdapter();
280                     //connects to the device's address and checks if it's available
281                     BluetoothDevice dispositivo = myBluetooth.getRemoteDevice(address);
282                     //create a RFCOMM (SPP) connection
283                     btSocket = dispositivo.createRfcommSocketToServiceRecord(myUUID);
284                     BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
285                     btSocket.connect();//start connection
286                 }
287             }
288             catch (IOException e)
289             {
290                 ConnectSuccess = false;//if the try failed, you can check the exception here
291             }
292             return null;
293         }
294
295         //after the doInBackground, it checks if everything went fine
296         @Override
297         protected void onPostExecute(Void result)
298         {
299             super.onPostExecute(result);
300
301             if (!ConnectSuccess)
302             {
303                 Toast.makeText(getActivity(),
304                         text: "¡Error! No se pudo establecer la conexión. Intente de nuevo",
305                         Toast.LENGTH_LONG).show();
306
307             }
308             else
309             {
310                 Toast.makeText(getActivity(), text: "Bluetooth conectado con éxito",
311                         Toast.LENGTH_LONG).show();
312                 new listenerBt().execute();  //Start Bluetooth listener
```

Continuación de la figura 59.

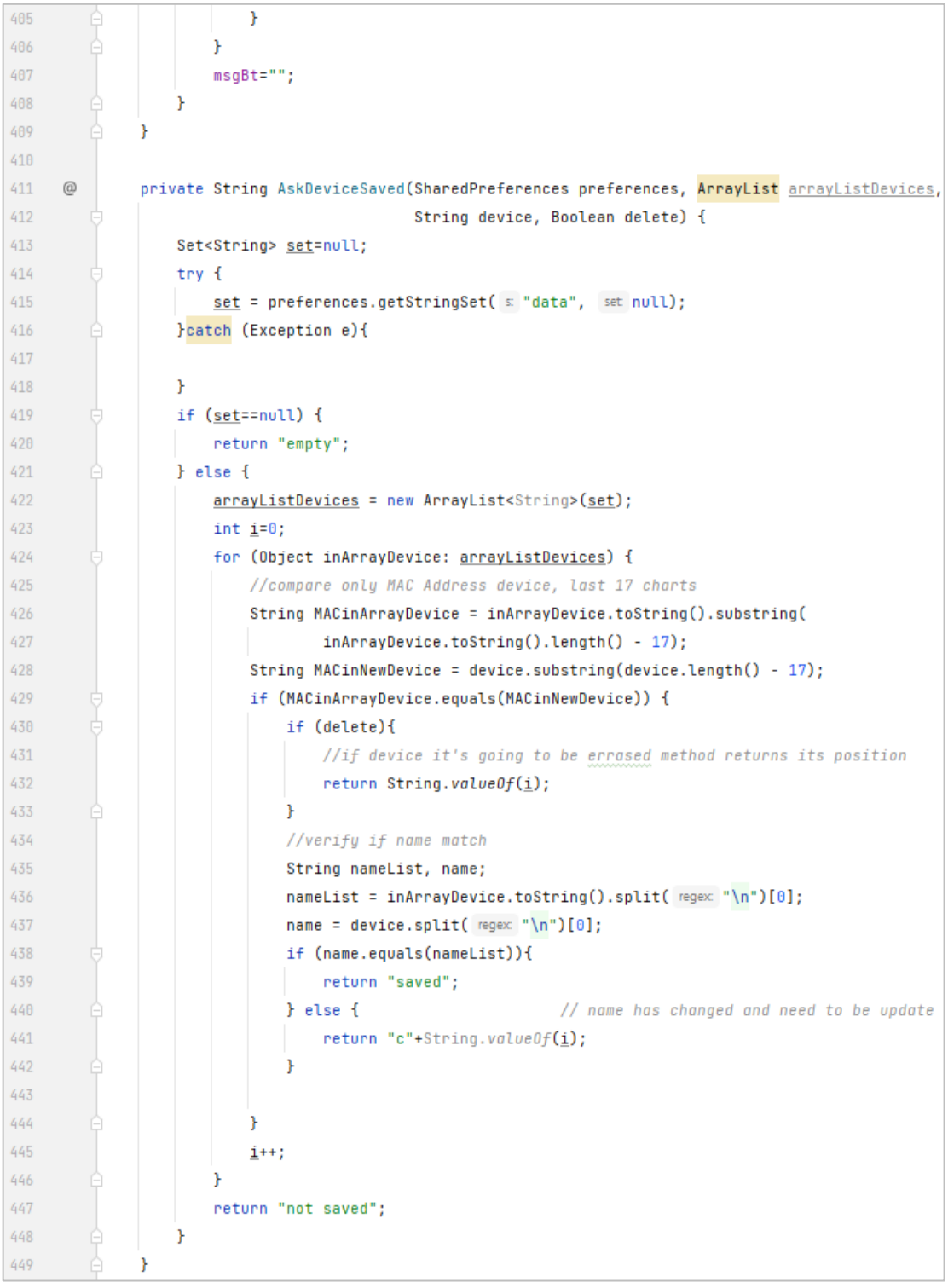


```
313                 isBtConnected = true;
314                 constraintLayoutConnBt.setVisibility(View.GONE);  //hide paired devices list
315                 constraintLayoutBtOk.setVisibility(View.VISIBLE); //show configuration layout
316                 writeBt( inf: "?");  //Ask to device for internet's connection information
317             }
318             progress.dismiss();
319         }
320     }
321
322     private class listenerBt extends AsyncTask<Void, String, String>{
323
324         @Override
325         protected void onPreExecute() {
326
327         }
328
329         @Override
330         protected String doInBackground(Void... voids) {
331             byte[] buffer = new byte[1024];  // buffer store for the stream
332             int len;
333             try {
334                 while (isBtConnected) {
335                     len = btSocket.getInputStream().read(buffer);
336                     byte[] data = Arrays.copyOf(buffer, len);
337                     String readMessage = new String(data);
338                     publishProgress(readMessage);
339                 }
340             } catch (IOException e) {
341                 e.printStackTrace();
342             }
343             return null;
344         }
345
346         @Override
347         protected void onProgressUpdate(String... values) { readBt(values[0]); }
350     }
351
352     private void writeBt(String inf) {           //Write bluetooth socket function
353         try {
354             btSocket.getOutputStream().write(inf.getBytes());
355         } catch (IOException e) {
356             e.printStackTrace();
357         }
358     }
359
```

Continuación de la figura 59.

```
360     private void readBt(String inf) {       //Read bluetooth socket function
361         msgBt=msgBt+inf;
362         if (msgBt.charAt(msgBt.length()-1) == '\n') {
363             msgBt = msgBt.replaceAll( regex: "\n",  replacement: "").trim();
364             confMeasDevi = msgBt.split( regex: ":");
365             if (msgBt.equals("I")) {                        //Bt device has Internet access
366                 textViewStatusWifi.setText("Con conexión a Internet");
367                 imageViewStatusWifi.setImageResource(R.drawable.green_point);
368                 textViewStatusWifi.setTextColor(Color.parseColor( colorString: "#32CD32"));
369                 imageViewStatusWifi.setVisibility(View.VISIBLE);
370                 btnConnW.setClickable(true);
371                 btnSaveConf.setClickable(true);
372             } else if (msgBt.equals("i")) {              //Bt device has not Internet access
373                 textViewStatusWifi.setText("Sin conexión a Internet");
374                 imageViewStatusWifi.setImageResource(R.drawable.red_point);
375                 textViewStatusWifi.setTextColor(Color.parseColor( colorString: "#F13A37"));
376                 imageViewStatusWifi.setVisibility(View.VISIBLE);
377                 btnConnW.setClickable(true);
378                 btnSaveConf.setClickable(true);
379             } else if (msgBt.equals("cred_wrg")) {
380                 Toast.makeText(getActivity(),  text: "Nombre de red o contaseña incorrectos",
381                         Toast.LENGTH_LONG).show();
382                 btnConnW.setClickable(true);
383             } else if (msgBt.equals("ok")) {
384                 //configuration has been saved
385                 Toast.makeText(getActivity(),  text: "Configuración guardada con éxito",
386                         Toast.LENGTH_LONG).show();
387                 listener.setOptions();
388             }else if (confMeasDevi.length>1) {
389                 if (confMeasDevi[0].equals("rate")) {
390                     //Bt device has send its rat
391                     editTextRate.setText(confMeasDevi[1]);
392                 } else if (confMeasDevi[0].equals("currency")) {
393                     //Bt device has send its currency
394                     if (confMeasDevi[1].equals("GTQ")) {
395                         radioButtonBtnGTQ.setChecked(true);
396                     } else if (confMeasDevi[1].equals("USD")) {
397                         radioButtonBtnUSD.setChecked(true);
398                     }
399                 } else if (confMeasDevi[0].equals("payday")) {
400                     //Bt device has send its daypay
401                     SpiPayday.setSelection(Integer.parseInt(confMeasDevi[1]) - 1);
402                 } else if (confMeasDevi[0].equals("name")) {
403                     //Bt device has send its name
404                     editTextNameDevi.setText(confMeasDevi[1]);
```

Continuación de la figura 59.

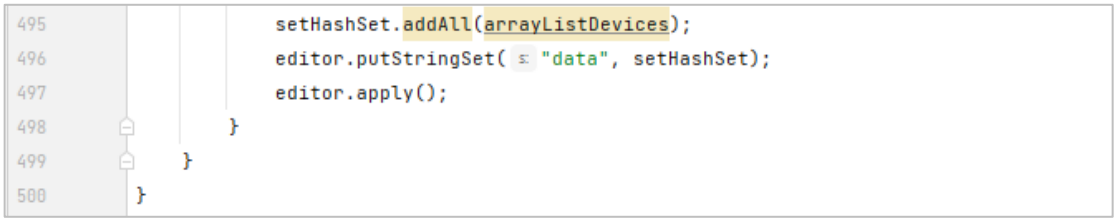


```
405                 }
406             }
407             msgBt="";
408         }
409     }
410
411 @   private String AskDeviceSaved(SharedPreferences preferences, ArrayList arrayListDevices,
412                                   String device, Boolean delete) {
413         Set<String> set=null;
414         try {
415             set = preferences.getStringSet( s: "data",  set: null);
416         }catch (Exception e){
417
418         }
419         if (set==null) {
420             return "empty";
421         } else {
422             arrayListDevices = new ArrayList<String>(set);
423             int i=0;
424             for (Object inArrayDevice: arrayListDevices) {
425                 //compare only MAC Address device, last 17 charts
426                 String MACinArrayDevice = inArrayDevice.toString().substring(
427                         inArrayDevice.toString().length() - 17);
428                 String MACinNewDevice = device.substring(device.length() - 17);
429                 if (MACinArrayDevice.equals(MACinNewDevice)) {
430                     if (delete){
431                         //if device it's going to be errased method returns its position
432                         return String.valueOf(i);
433                     }
434                     //verify if name match
435                     String nameList, name;
436                     nameList = inArrayDevice.toString().split( regex: "\n")[0];
437                     name = device.split( regex: "\n")[0];
438                     if (name.equals(nameList)){
439                         return "saved";
440                     } else {                    // name has changed and need to be update
441                         return "c"+String.valueOf(i);
442                     }
443
444                 }
445                 i++;
446             }
447             return "not saved";
448         }
449     }
```

Continuación de la figura 59.

```
450     private void AddDevice(String device){
451         SharedPreferences preferences= getActivity().getSharedPreferences(
452                 name: "devices",Context.MODE_PRIVATE);
453         ArrayList arrayListDevices = new ArrayList();
454         Set<String> setHashSet = new HashSet<>();
455
456         //Verify if device is already saved or there are any data saved in phone
457         String status = AskDeviceSaved(preferences, arrayListDevices, device, delete: false);
458
459         if (!status.equals("saved")){              //save device if isn't still saved
460             SharedPreferences.Editor editor = preferences.edit();
461             //if there are any data store in phone then load data and write a new value
462             if (status.equals("not saved") || status.charAt(0)=='c'){
463                 Set<String> setArray=null;
464                 setArray = preferences.getStringSet( s: "data",  set: null);
465                 arrayListDevices = new ArrayList<String>(setArray);
466             }
467             if (status.charAt(0)=='c'){
468                 int index = Integer.parseInt(status.substring(1));  //get index to update
469                 arrayListDevices.set(index,device);     //update data
470             }else {
471                 arrayListDevices.add(device);          //write new value
472             }
473             setHashSet.addAll(arrayListDevices);
474             editor.putStringSet( s: "data", setHashSet);
475             editor.apply();
476         }
477     }
478
479     private void ErraseDevice(String device){
480         SharedPreferences preferences= getActivity().getSharedPreferences(
481                 name: "devices",Context.MODE_PRIVATE);
482         ArrayList arrayListDevices = new ArrayList();
483         //Verify if device is already saved or there are any data saved in phone
484         String status = AskDeviceSaved(preferences, arrayListDevices, device, delete: true);
485
486         if (!status.equals("empty") && !status.equals("not saved")) {
487             Set<String> setHashSet = new HashSet<>();
488
489             SharedPreferences.Editor editor = preferences.edit();
490             Set<String> setArray=null;
491             setArray = preferences.getStringSet( s: "data",  set: null);
492             arrayListDevices = new ArrayList<String>(setArray);
493             arrayListDevices.remove(Integer.parseInt(status));        //erase device
494
```

Continuación de la figura 59.

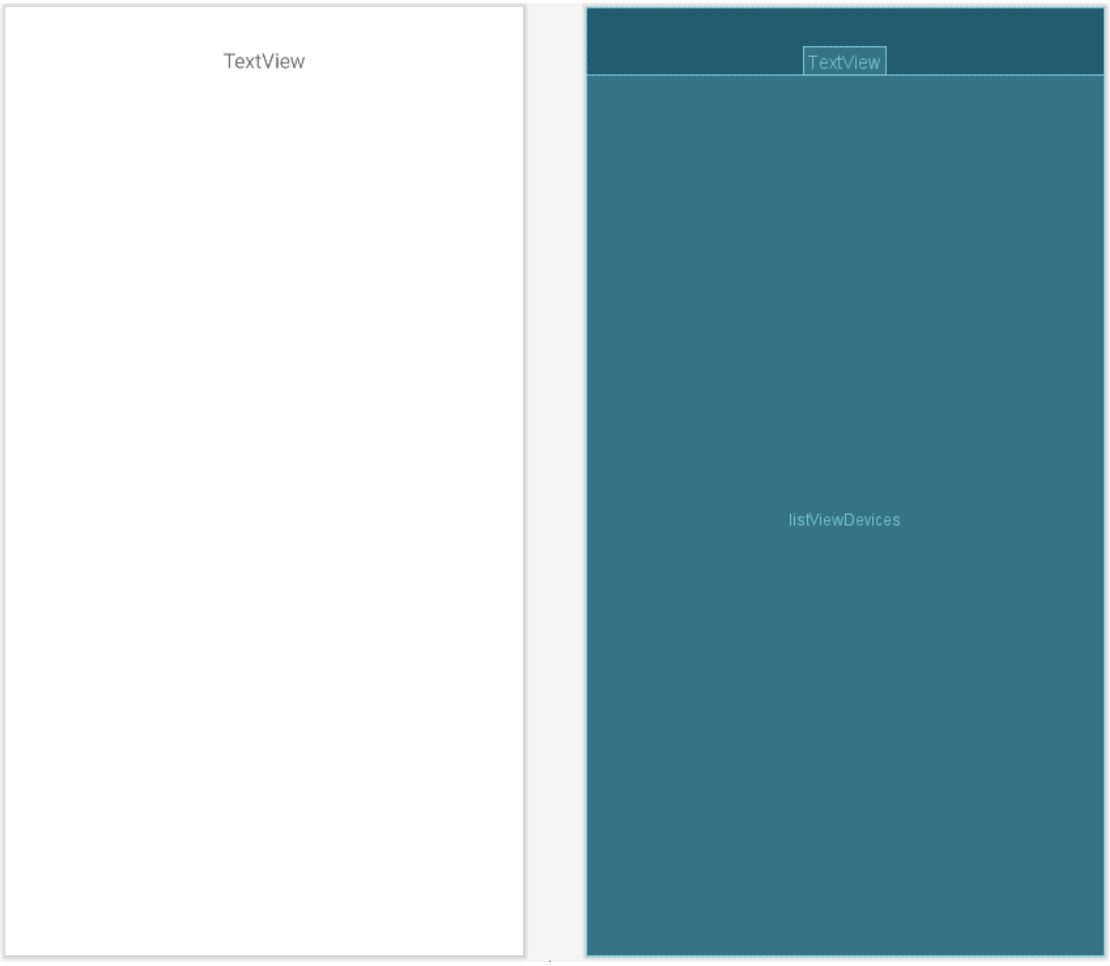


Fuente: elaboración propia, realizado con captura de pantalla.

5.2.2.5. *Fragment* ViewDevices

Para acceder a este recurso es necesario haber establecido conexión con el servidor MQTT, gráficamente este recurso únicamente muestra una lista con los dispositivos de medición previamente agregados y una opción que permite consultar el historial de consumo de los dispositivos, en caso no se tenga agregado ningún dispositivo se mostrará un anuncio indicando la ausencia de estos y se ocultarán las opciones de visualización. Según la opción seleccionada por el usuario se mandará a llamar al *fragment* correspondiente a la función solicitada y se reemplazará el actual.

Figura 60. **Diseño y código XML *fragment* ViewDevices (App Android)**



fragment_view_devices.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/frameLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ViewDevices">

    <ListView
        android:id="@+id/listViewDevices"
```

Continuación de la figura 60.

```
12          android:id="@+id/listViewDevices"
13          android:layout_width="0dp"
14          android:layout_height="0dp"
15          app:layout_constraintBottom_toBottomOf="parent"
16          app:layout_constraintEnd_toEndOf="parent"
17          app:layout_constraintStart_toStartOf="parent"
18          app:layout_constraintTop_toBottomOf="@+id/textViewInfo" />
19
20      <TextView
21          android:id="@+id/textViewInfo"
22          android:layout_width="wrap_content"
23          android:layout_height="wrap_content"
24          android:layout_marginTop="30dp"
25          android:text="TextView"
26          android:textAlignment="center"
27          android:textSize="16sp"
28          app:layout_constraintEnd_toEndOf="parent"
29          app:layout_constraintHorizontal_bias="0.5"
30          app:layout_constraintStart_toStartOf="parent"
31          app:layout_constraintTop_toTopOf="parent" />
32  </androidx.constraintlayout.widget.ConstraintLayout>
```

Fuente: elaboración propia, realizado con captura de pantalla.

Figura 61. **Código Java *fragment* ViewDevices (App Android)**

ViewDevices.java

```java
package com.example.monitorenergtico;

import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import java.util.ArrayList;
import java.util.Set;

public class ViewDevices extends Fragment {

    Listener listener;

    ListView listViewDevices;
    TextView textViewInfo;
    public ViewDevices() {
        // Required empty public constructor
    }


    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_view_devices, container, attachToRoot: false);
    }

    @Override
    public void onAttach(@NonNull Context context) {
        super.onAttach(context);
```

Continuación de la figura 61.

```
46          if (context instanceof Listener) {
47              listener = (Listener) context;
48          }
49      }
50
51      @Override
52      public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
53          super.onViewCreated(view, savedInstanceState);
54
55          listViewDevices = view.findViewById(R.id.listViewDevices);
56          textViewInfo = view.findViewById(R.id.textViewInfo);
57
58          textViewInfo.setText("Seleccione el dispositivo a visualizar:");
59
60          SharedPreferences preferences= getActivity().getSharedPreferences(
61                  name: "devices", Context.MODE_PRIVATE);
62          ArrayList arrayListDevices = new ArrayList();
63          Set<String> setArray=null;
64          try {
65              setArray = preferences.getStringSet( s: "data",  set: null);
66              arrayListDevices = new ArrayList<String>(setArray);
67          }catch (Exception e){
68
69          }
70          if (arrayListDevices.isEmpty()){
71              textViewInfo.setText("Para visualizar un dispositivo primero debe " +
72                      "configurarlo y agregarlo a su visualizador.");
73          }else{
74              ArrayList arrayListShowDevi = new ArrayList();
75              for(Object item:arrayListDevices){
76                  String[] nameDevi = item.toString().split( regex: "\n");
77                  arrayListShowDevi.add('\n'+nameDevi[0]+'\n'+nameDevi[1]+'\n');
78              }
79              arrayListShowDevi.add('\n'+"Consultar historial"+'\n');
80              final ArrayAdapter adapter = new ArrayAdapter(getActivity(),
81                      android.R.layout.simple_list_item_1, arrayListShowDevi);
82              listViewDevices.setAdapter(adapter);
83
84              listViewDevices.setOnItemClickListener(new AdapterView.OnItemClickListener() {
85                  @Override
86                  public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
87                      if (adapterView.getItemAtPosition(i).toString().trim().equals(
88                              "Consultar historial")){
89                          listener.setConsHistory();
90                      }else {
```

Continuación de la figura 61.

```
91                  String name = adapterView.getItemAtPosition(i).toString();
92                  String topic = adapterView.getItemAtPosition(i).toString();
93                  name = name.split( regex: "\n")[1];
94                  topic = topic.split( regex: "\n")[2];
95                  //send info to get device's mqtt messages
96                  Bundle bundle = new Bundle();
97                  bundle.putString("topic",topic);
98                  getParentFragmentManager().setFragmentResult( requestKey: "topic",bundle);
99
100                 listener.setViewEnergyCons(name,topic);
101             }
102         }
103     });
104 }
105 }
106 }
```

Fuente: elaboración propia, realizado con captura de pantalla.

5.2.2.6. *Fragment* ViewEnergyConsumption

Este recurso resulta como consecuencia de una llamada por parte del *fragment* ViewDevices y su función corresponde a una de las más importantes dentro de este monitor energético y por medio de sus recursos gráficos se lleva a cabo la visualización energética en tiempo real, la información mostrada es actualizada con una frecuencia de 1 vez por segundo, este recurso se comunica constantemente con el *fragment* MQTTservice con el propósito de solicitar y obtener los valores de mediciones transmitidos a través del *topic* MQTT dedicado a dicha tarea.

Figura 62. **Diseño y código XML *fragment* ViewEnergyConsumption (App Android)**

fragment_view_energy_consumption.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/frameLayout2"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ViewEnergyConsumption">

```

Continuación de la figura 62.

```
    <TextView
        android:id="@+id/textViewName"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"
        android:layout_marginTop="30dp"
        android:layout_marginEnd="10dp"
        android:background="@drawable/round_corners"
        android:backgroundTint="#B3000000"
        android:text="NAME DEVICE"
        android:textAlignment="center"
        android:textColor="@color/white"
        android:textSize="24sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/textViewCharge"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"
        android:layout_marginTop="6dp"
        android:layout_marginEnd="10dp"
        android:background="@drawable/round_corners"
        android:backgroundTint="#B3000000"
        android:text="0"
        android:textAlignment="center"
        android:textColor="@color/white"
        android:textSize="34sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.662"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textViewEnergy" />

    <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/ConsLayVolt"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"
        android:layout_marginTop="6dp"
        android:layout_marginEnd="3dp"
        android:background="@drawable/round_corners"
```

Continuación de la figura 62.

```
54      android:backgroundTint="#B3000000"
55      app:layout_constraintEnd_toStartOf="@+id/ConsLayCorr"
56      app:layout_constraintHorizontal_bias="0.5"
57      app:layout_constraintStart_toStartOf="parent"
58      app:layout_constraintTop_toBottomOf="@+id/textViewName">
59
60      <ProgressBar
61          android:id="@+id/circular_pb_volt"
62          android:layout_width="150dp"
63          android:layout_height="150dp"
64          android:layout_marginStart="10dp"
65          android:layout_marginTop="10dp"
66          android:layout_marginEnd="10dp"
67          android:layout_marginBottom="10dp"
68          android:indeterminateOnly="false"
69          android:progress="0"
70          android:progressDrawable="@drawable/pb_circular_determinative"
71          app:layout_constraintBottom_toBottomOf="parent"
72          app:layout_constraintEnd_toEndOf="parent"
73          app:layout_constraintHorizontal_bias="0.5"
74          app:layout_constraintStart_toStartOf="parent"
75          app:layout_constraintTop_toTopOf="parent" />
76
77      <TextView
78          android:id="@+id/textViewVolt"
79          android:layout_width="wrap_content"
80          android:layout_height="wrap_content"
81          android:text="0"
82          android:textColor="@color/white"
83          android:textSize="20sp"
84          app:layout_constraintBottom_toBottomOf="@+id/circular_pb_volt"
85          app:layout_constraintEnd_toEndOf="@+id/circular_pb_volt"
86          app:layout_constraintHorizontal_bias="0.503"
87          app:layout_constraintStart_toStartOf="@+id/circular_pb_volt"
88          app:layout_constraintTop_toTopOf="@+id/circular_pb_volt"
89          app:layout_constraintVertical_bias="0.504" />
90
91      <TextView
92          android:id="@+id/textView9"
93          android:layout_width="wrap_content"
94          android:layout_height="wrap_content"
95          android:text="Volts"
96          android:textColor="@color/white"
```

Continuación de la figura 62.

```
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textViewVolt" />

    </androidx.constraintlayout.widget.ConstraintLayout>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/ConsLayCorr"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="3dp"
        android:layout_marginTop="6dp"
        android:layout_marginEnd="10dp"
        android:background="@drawable/round_corners"
        android:backgroundTint="#B3000000"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toEndOf="@+id/ConsLayVolt"
        app:layout_constraintTop_toBottomOf="@+id/textViewName">

        <ProgressBar
            android:id="@+id/circular_pb_curr"
            android:layout_width="150dp"
            android:layout_height="150dp"
            android:layout_marginStart="10dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="10dp"
            android:layout_marginBottom="10dp"
            android:indeterminateOnly="false"
            android:progress="0"
            android:progressDrawable="@drawable/pb_circular_current"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/textViewCurrent"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="0"
```

Continuación de la figura 62.

```
            android:textAlignment="center"
            android:textColor="@color/white"
            android:textSize="20sp"
            app:layout_constraintBottom_toBottomOf="@+id/circular_pb_curr"
            app:layout_constraintEnd_toEndOf="@+id/circular_pb_curr"
            app:layout_constraintStart_toStartOf="@+id/circular_pb_curr"
            app:layout_constraintTop_toTopOf="@+id/circular_pb_curr" />

        <TextView
            android:id="@+id/textView17"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Amps"
            android:textColor="@color/white"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textViewCurrent" />

    </androidx.constraintlayout.widget.ConstraintLayout>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/ConstLayPower"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"
        android:layout_marginTop="6dp"
        android:layout_marginEnd="10dp"
        android:background="@drawable/round_corners"
        android:backgroundTint="#B3000000"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/ConsLayVolt">

        <ProgressBar
            android:id="@+id/circular_pb_power"
            android:layout_width="175dp"
            android:layout_height="175dp"
            android:layout_marginTop="10dp"
            android:layout_marginBottom="10dp"
            android:indeterminateOnly="false"
            android:progress="0"
            android:progressDrawable="@drawable/pb_circular_power"
```

Continuación de la figura 62.

```
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/textView18"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Watts"
            android:textColor="@color/white"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textViewPower" />

        <TextView
            android:id="@+id/textViewPower"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="0"
            android:textColor="@color/white"
            android:textSize="24sp"
            app:layout_constraintBottom_toBottomOf="@+id/circular_pb_power"
            app:layout_constraintEnd_toEndOf="@+id/circular_pb_power"
            app:layout_constraintStart_toStartOf="@+id/circular_pb_power"
            app:layout_constraintTop_toTopOf="@+id/circular_pb_power" />

    </androidx.constraintlayout.widget.ConstraintLayout>

    <TextView
        android:id="@+id/textViewEnergy"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"
        android:layout_marginTop="6dp"
        android:layout_marginEnd="10dp"
        android:background="@drawable/round_corners"
        android:backgroundTint="#B3000000"
        android:text="0 KWh"
        android:textAlignment="center"
        android:textColor="@color/white"
```

Continuación de la figura 62.

```
226         android:textSize="24sp"
227         app:layout_constraintEnd_toEndOf="parent"
228         app:layout_constraintHorizontal_bias="0.5"
229         app:layout_constraintStart_toStartOf="parent"
230         app:layout_constraintTop_toBottomOf="@+id/ConstLayPower" />
231
232     <androidx.constraintlayout.widget.ConstraintLayout
233         android:id="@+id/ConstLayGraph"
234         android:layout_width="0dp"
235         android:layout_height="0dp"
236         android:layout_marginStart="10dp"
237         android:layout_marginTop="6dp"
238         android:layout_marginEnd="10dp"
239         android:layout_marginBottom="10dp"
240         android:background="@drawable/round_corners"
241         android:backgroundTint="#B3000000"
242         app:layout_constraintBottom_toBottomOf="parent"
243         app:layout_constraintEnd_toEndOf="parent"
244         app:layout_constraintStart_toStartOf="parent"
245         app:layout_constraintTop_toBottomOf="@+id/textViewCharge">
246
247         <com.github.mikephil.charting.charts.LineChart
248             android:id="@+id/linechart"
249             android:layout_width="match_parent"
250             android:layout_height="match_parent"
251             app:layout_constraintBottom_toBottomOf="parent"
252             app:layout_constraintEnd_toEndOf="parent"
253             app:layout_constraintStart_toStartOf="parent"
254             app:layout_constraintTop_toTopOf="parent" />
255
256     </androidx.constraintlayout.widget.ConstraintLayout>
257
258 </androidx.constraintlayout.widget.ConstraintLayout>
```

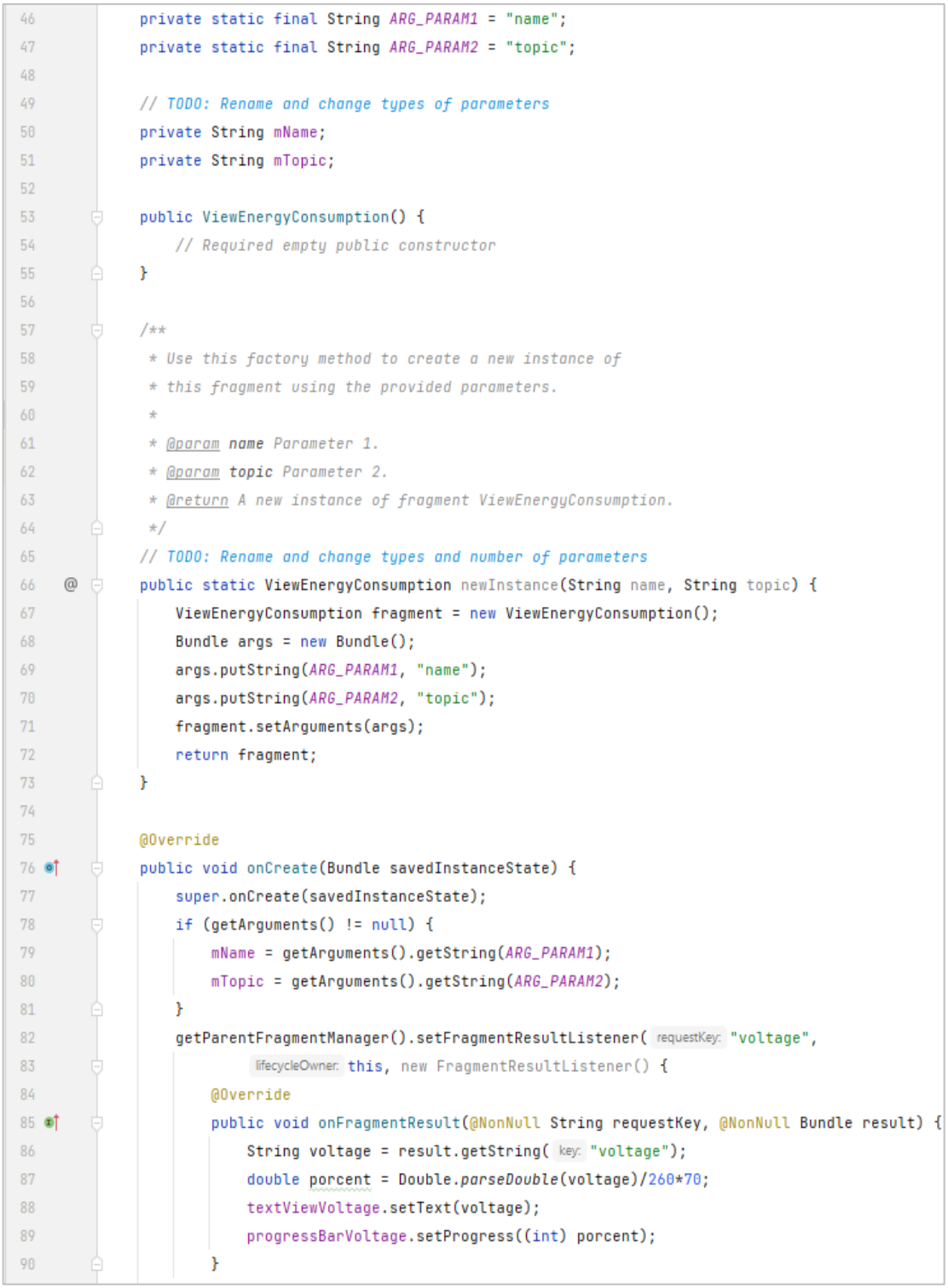
Fuente: elaboración propia, realizado con captura de pantalla.

Figura 63. **Código Java *fragment* ViewEnergyConsumption (App Android)**

ViewEnergyConsumption.java

```
package com.example.monitorenergtico;

import android.graphics.Color;
import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentResultListener;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ProgressBar;
import android.widget.TextView;
import com.github.mikephil.charting.charts.LineChart;
import com.github.mikephil.charting.components.AxisBase;
import com.github.mikephil.charting.components.Description;
import com.github.mikephil.charting.components.Legend;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.data.LineData;
import com.github.mikephil.charting.data.LineDataSet;
import com.github.mikephil.charting.formatter.ValueFormatter;
import com.github.mikephil.charting.interfaces.datasets.ILineDataSet;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

/**
 * A simple {@link Fragment} subclass.
 * Use the {@link ViewEnergyConsumption#newInstance} factory method to
 * create an instance of this fragment.
 */

public class ViewEnergyConsumption extends Fragment {

    TextView name;
    TextView textViewVoltage, textViewCurrent, textViewPower, textViewEnergy, textViewCharge;
    ProgressBar progressBarVoltage,progressBarCurrent,progressBarPower;
    LineChart mpLineChart;
    ArrayList<Entry> dataValues = new ArrayList<~>();
    SimpleDateFormat simpleDateFormat = new SimpleDateFormat( pattern: "HH:mm:ss");
    private Thread MeasureThread;

    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
```

Continuación de la figura 63.



```
46      private static final String ARG_PARAM1 = "name";
47      private static final String ARG_PARAM2 = "topic";
48
49      // TODO: Rename and change types of parameters
50      private String mName;
51      private String mTopic;
52
53      public ViewEnergyConsumption() {
54          // Required empty public constructor
55      }
56
57      /**
58       * Use this factory method to create a new instance of
59       * this fragment using the provided parameters.
60       *
61       * @param name Parameter 1.
62       * @param topic Parameter 2.
63       * @return A new instance of fragment ViewEnergyConsumption.
64       */
65      // TODO: Rename and change types and number of parameters
66  @   public static ViewEnergyConsumption newInstance(String name, String topic) {
67          ViewEnergyConsumption fragment = new ViewEnergyConsumption();
68          Bundle args = new Bundle();
69          args.putString(ARG_PARAM1, "name");
70          args.putString(ARG_PARAM2, "topic");
71          fragment.setArguments(args);
72          return fragment;
73      }
74
75      @Override
76      public void onCreate(Bundle savedInstanceState) {
77          super.onCreate(savedInstanceState);
78          if (getArguments() != null) {
79              mName = getArguments().getString(ARG_PARAM1);
80              mTopic = getArguments().getString(ARG_PARAM2);
81          }
82          getParentFragmentManager().setFragmentResultListener( requestKey: "voltage",
83                  lifecycleOwner: this, new FragmentResultListener() {
84              @Override
85              public void onFragmentResult(@NonNull String requestKey, @NonNull Bundle result) {
86                  String voltage = result.getString( key: "voltage");
87                  double porcent = Double.parseDouble(voltage)/260*70;
88                  textViewVoltage.setText(voltage);
89                  progressBarVoltage.setProgress((int) porcent);
90              }
```

Continuación de la figura 63.

```
91          });
92          getParentFragmentManager().setFragmentResultListener( requestKey: "current",
93                  lifecycleOwner: this, new FragmentResultListener() {
94              @Override
95              public void onFragmentResult(@NonNull String requestKey, @NonNull Bundle result) {
96                  String current = result.getString( key: "current");
97                  double porcent=1;
98                  if (Double.parseDouble(current)<=15){
99                      porcent = Double.parseDouble(current)/15*70;
100                 } else if (Double.parseDouble(current)>15){
101                     porcent = 70;
102                 }
103                 if (porcent>0 && porcent<1){
104                     porcent = 1;
105                 }
106                 textViewCurrent.setText(current);
107                 progressBarCurrent.setProgress((int) porcent);
108             }
109
110         });
111         getParentFragmentManager().setFragmentResultListener( requestKey: "power",
112                 lifecycleOwner: this, new FragmentResultListener() {
113             @Override
114             public void onFragmentResult(@NonNull String requestKey, @NonNull Bundle result) {
115                 String power = result.getString( key: "power");
116                 double porcent=1;
117                 if (Double.parseDouble(power)<=1800){
118                     porcent = Double.parseDouble(power)/1800*70;
119                 } else if (Double.parseDouble(power)>1800){
120                     porcent = 70;
121                 }
122                 if (porcent>0 && porcent<1){
123                     porcent = 1;
124                 }
125                 textViewPower.setText(power);
126                 progressBarPower.setProgress((int) porcent);
127             }
128         });
129         getParentFragmentManager().setFragmentResultListener( requestKey: "energy",
130                 lifecycleOwner: this, new FragmentResultListener() {
131             @Override
132             public void onFragmentResult(@NonNull String requestKey, @NonNull Bundle result) {
133                 String energy = result.getString( key: "energy");
134                 textViewEnergy.setText(energy + " KWh");
135                 Float energyf = Float.parseFloat(energy);
```

Continuación de la figura 63.

```
136                 addDataGraph(energyf);
137             }
138         });
139         getParentFragmentManager().setFragmentResultListener( requestKey: "charge",
140                 lifecycleOwner: this, new FragmentResultListener() {
141             @Override
142             public void onFragmentResult(@NonNull String requestKey, @NonNull Bundle result) {
143                 String charge = result.getString( key: "charge");
144                 textViewCharge.setText(charge);
145             }
146         });
147     }
148
149     @Override
150     public View onCreateView(LayoutInflater inflater, ViewGroup container,
151                              Bundle savedInstanceState) {
152         // Inflate the layout for this fragment
153         return inflater.inflate(R.layout.fragment_view_energy_consumption, container,
154                 attachToRoot: false);
155     }
156
157     @Override
158     public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
159         super.onViewCreated(view, savedInstanceState);
160
161         name = view.findViewById(R.id.textViewName);
162         textViewVoltage = view.findViewById(R.id.textViewVolt);
163         textViewCurrent = view.findViewById(R.id.textViewCurrent);
164         textViewPower = view.findViewById(R.id.textViewPower);
165         textViewEnergy = view.findViewById(R.id.textViewEnergy);
166         textViewCharge = view.findViewById(R.id.textViewCharge);
167         progressBarVoltage = view.findViewById(R.id.circular_pb_volt);
168         progressBarCurrent = view.findViewById(R.id.circular_pb_curr);
169         progressBarPower = view.findViewById(R.id.circular_pb_power);
170         mpLineChart=(LineChart) view.findViewById(R.id.linechart);
171
172         name.setText(mName);
173         graphStart();
174
175         MeasureThread = new NewThread();
176         MeasureThread.setName("req measurements");
177         MeasureThread.start();
178     }
179
180     @Override
```

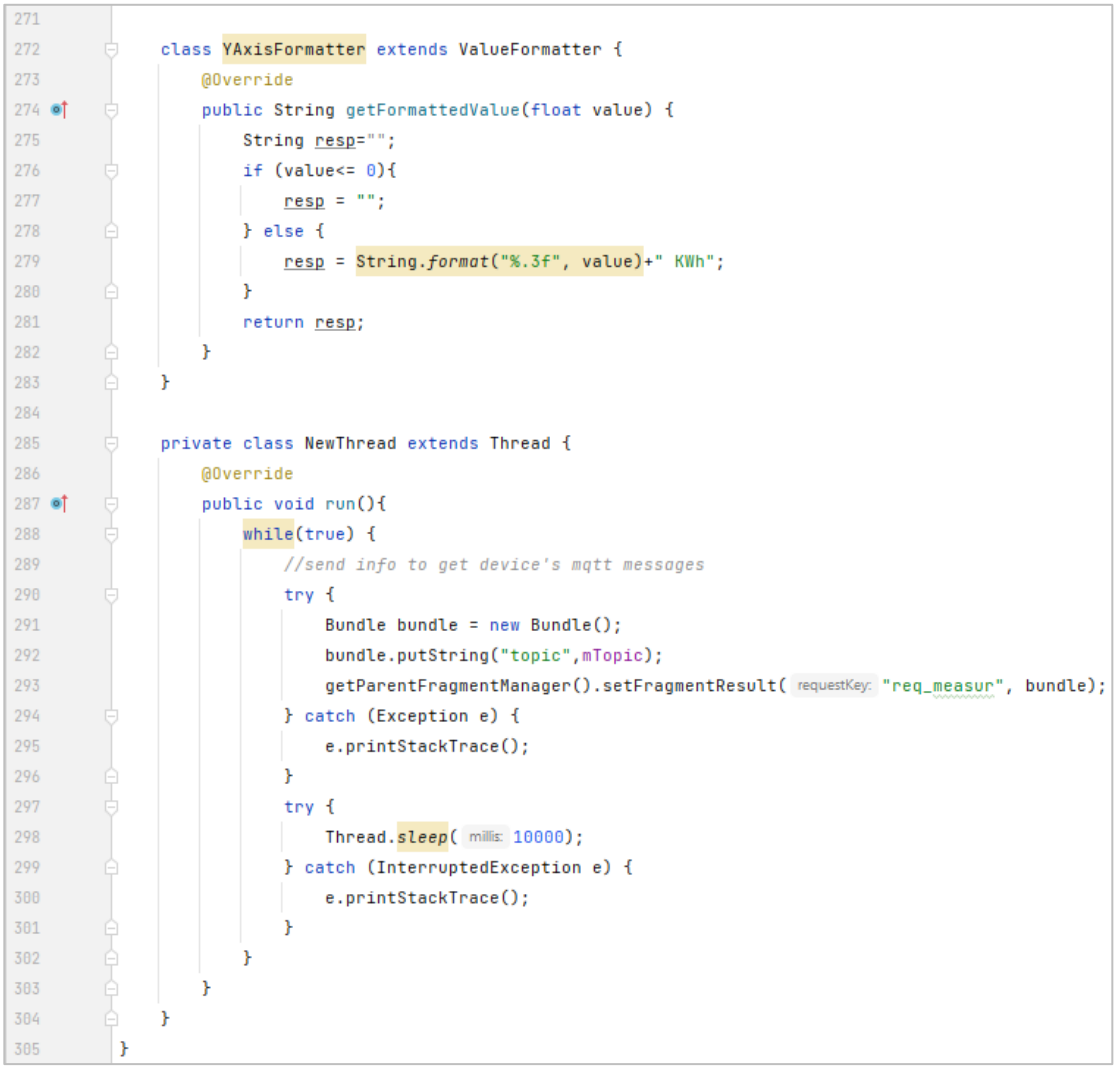
Continuación de la figura 63.

```
181     public void onStop() {
182         super.onStop();
183         MeasureThread.interrupt();
184         progressBarVoltage.setProgress(0);
185         progressBarCurrent.setProgress(0);
186         progressBarPower.setProgress(0);
187         dataValues.clear();
188         Bundle loginBundle = new Bundle();
189         loginBundle.putString("unsuscribe","");
190         getParentFragmentManager().setFragmentResult( requestKey: "unsuscribe",loginBundle);
191     }
192
193     public void graphStart(){
194         //custom graph
195         mpLineChart.setBackgroundColor(Color.parseColor( colorString: "#00FFFFFF"));
196         mpLineChart.setNoDataText("No Data");
197         mpLineChart.setNoDataTextColor(Color.WHITE);
198         mpLineChart.getAxisLeft().setDrawGridLines(false);
199         mpLineChart.getAxisRight().setDrawGridLines(false);
200         mpLineChart.getXAxis().setDrawGridLines(false);
201         mpLineChart.getAxisLeft().setTextColor(Color.WHITE);
202         mpLineChart.getAxisRight().setTextColor(Color.parseColor( colorString: "#00FFFFFF"));
203         mpLineChart.getXAxis().setTextColor(Color.WHITE);
204         mpLineChart.getXAxis().setDrawAxisLine(false);
205         mpLineChart.getAxisRight().setDrawAxisLine(false);
206         mpLineChart.getAxisLeft().setDrawAxisLine(false);
207         mpLineChart.getXAxis().setDrawLabels(true);
208         mpLineChart.getLegend().setTextColor(Color.WHITE);
209         mpLineChart.getLegend().setVerticalAlignment(Legend.LegendVerticalAlignment.TOP);
210         mpLineChart.getLegend().setHorizontalAlignment(Legend.LegendHorizontalAlignment.CENTER);
211         mpLineChart.getLegend().setDrawInside(false);
212         //graph description
213         Description description = new Description();
214         description.setText("");
215         mpLineChart.setDescription(description);
216         //graph axis formatter
217         mpLineChart.getXAxis().setValueFormatter(new XAxisFormatter());
218         mpLineChart.getAxisLeft().setValueFormatter(new YAxisFormatter());
219         //mpLineChart.getAxisLeft().setStartAtZero(true);
220     }
221
222     public void addDataGraph(Float value){
223         LineDataSet lineDataSet1 = new LineDataSet(dataValues, label: "Energy");
224         ArrayList<ILineDataSet> dataSets = new ArrayList<>();
225         dataSets.add(lineDataSet1);
```

Continuación de la figura 63.

```
226
227         String NowTime = simpleDateFormat.format(new Date());
228         NowTime = NowTime.replace( target: ":", replacement: "");
229         dataValues.add(new Entry(Float.parseFloat(NowTime),value));
230
231         LineData data = new LineData(dataSets);
232         mpLineChart.setData(data);
233         mpLineChart.invalidate();
234
235         //custom line
236         lineDataSet1.setLineWidth(3);
237         //lineDataSet1.setColor(Color.parseColor("#E91E61")); 7A0BC0
238         lineDataSet1.setColor(Color.parseColor( colorString: "#E91E61"));
239         lineDataSet1.setDrawCircles(false);
240         lineDataSet1.setDrawValues(false);
241         lineDataSet1.setValueTextColor(Color.WHITE);
242
243         LineData newdata = new LineData(dataSets);
244         mpLineChart.setData(newdata);
245         mpLineChart.invalidate();
246     }
247
248     class XAxisFormatter extends ValueFormatter {
249
250         @Override
251 @       public String getAxisLabel(float value, AxisBase axis) {
252             axis.setLabelCount( count: 2, force: true);
253             String time="";
254             if (Float.toString(value).charAt(5)=='.'){
255                 time = "0"+Float.toString(value).charAt(0)+":"+
256                         Float.toString(value).charAt(1)+""+
257                         Float.toString(value).charAt(2)+":"+
258                         Float.toString(value).charAt(3)+""+
259                         Float.toString(value).charAt(4);
260             }else {
261                 time = Float.toString(value).charAt(0) + "" +
262                         Float.toString(value).charAt(1) + ":" +
263                         Float.toString(value).charAt(2) + "" +
264                         Float.toString(value).charAt(3) + ":" +
265                         Float.toString(value).charAt(4) + "" +
266                         Float.toString(value).charAt(5);
267             }
268             return time;
269         }
270     }
```

Continuación de la figura 63.



```
271
272     class YAxisFormatter extends ValueFormatter {
273         @Override
274         public String getFormattedValue(float value) {
275             String resp="";
276             if (value<= 0){
277                 resp = "";
278             } else {
279                 resp = String.format("%.3f", value)+" KWh";
280             }
281             return resp;
282         }
283     }
284
285     private class NewThread extends Thread {
286         @Override
287         public void run(){
288             while(true) {
289                 //send info to get device's mqtt messages
290                 try {
291                     Bundle bundle = new Bundle();
292                     bundle.putString("topic",mTopic);
293                     getParentFragmentManager().setFragmentResult( requestKey: "req_measur", bundle);
294                 } catch (Exception e) {
295                     e.printStackTrace();
296                 }
297                 try {
298                     Thread.sleep( millis: 10000);
299                 } catch (InterruptedException e) {
300                     e.printStackTrace();
301                 }
302             }
303         }
304     }
305 }
```

Fuente: elaboración propia, realizado con captura de pantalla.

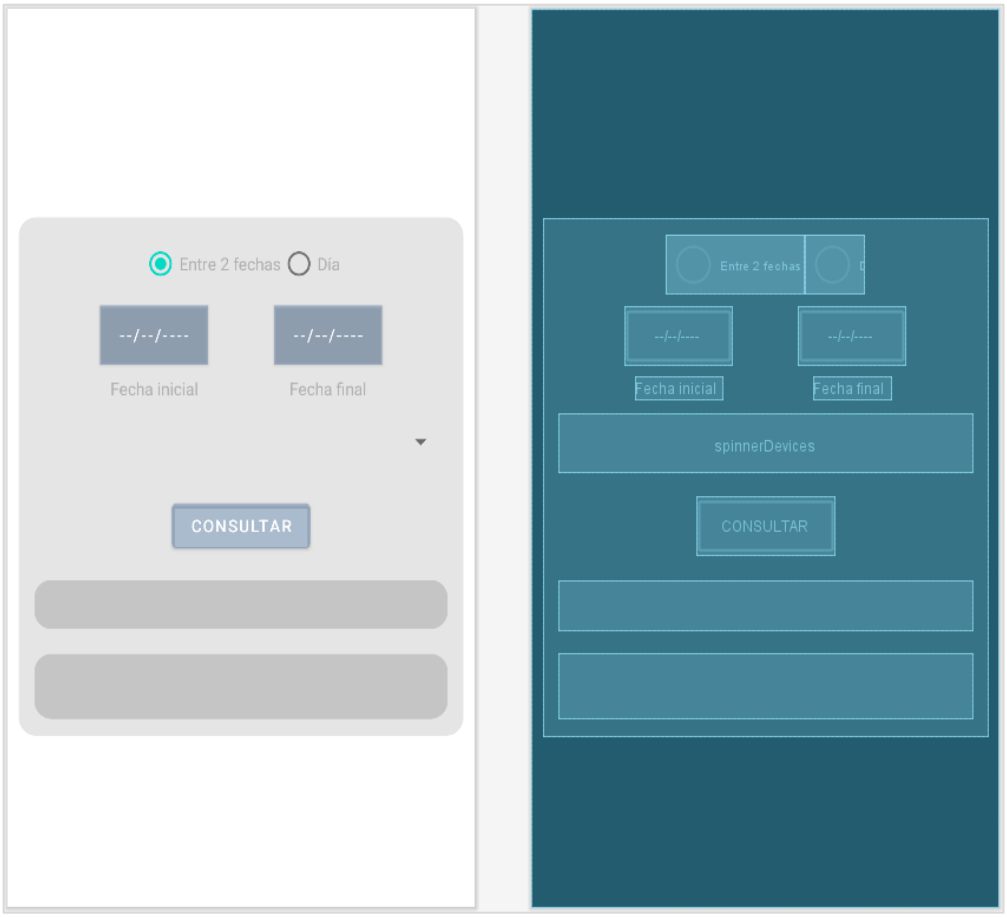
5.2.2.7. *Fragment* ConsumptionHistory

Por medio de este *fragment* se realizan las consultas en el historial de consumo siempre y cuando se haya establecido conexión con el servidor MQTT ya que esta función se manda a llamar desde el *fragment* ViewDecives, este

recurso hace que se envíen mensajes MQTT con los datos a consultar y una vez los mensajes llegan al servidor remoto este interpreta la información y de ser válida realiza una consulta a la base de datos devolviendo la respuesta por medio de un mensaje MQTT, luego esa respuesta es mostrada al usuario.

La manera en la que un dispositivo Android realiza las consultas SQL a la base de datos es de una manera indirecta, y la consulta se solicita remotamente (por medio de MQTT), y se ejecuta localmente con los criterios solicitados.

Figura 64. **Diseño y código XML *fragment* ConsumptionHistory (App Android)**



Continuación de la figura 64.

fragment_consumption_history.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/ConstraintLayoutHistory"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ConsumptionHistory">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"
        android:layout_marginTop="30dp"
        android:layout_marginEnd="10dp"
        android:background="@drawable/round_corners"
        android:backgroundTint="#80000000"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <RadioGroup
            android:id="@+id/radioGroup2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:orientation="horizontal"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent">

            <RadioButton
                android:id="@+id/radioButtonDates"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:checked="true"
                android:orientation="horizontal"
                android:text="Entre 2 fechas"
```

Continuación de la figura 64.

```
                android:textAlignment="center"
                android:textColor="#B1B1B1" />

            <RadioButton
                android:id="@+id/radioButtonDay"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Día"
                android:textColor="#B1B1B1" />

        </RadioGroup>

        <Button
            android:id="@+id/btnDateFinal"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="10dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="10dp"
            android:background="#F2000000"
            android:text="--/--/----"
            android:visibility="visible"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toEndOf="@+id/btnDateIni"
            app:layout_constraintTop_toBottomOf="@+id/radioGroup2" />

        <Spinner
            android:id="@+id/spinnerDevices"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="10dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="10dp"
            android:minHeight="48dp"
            android:popupBackground="@color/colorBackgroundSpinner"
            android:textAlignment="center"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.442"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textView20"
            tools:ignore="SpeakableTextPresentCheck" />
```

Continuación de la figura 64.

```
    <Button
        android:id="@+id/btnConsult"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:text="Consultar"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/spinnerDevices" />

    <Button
        android:id="@+id/btnDateIni"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"
        android:layout_marginTop="10dp"
        android:layout_marginRight="10dp"
        android:background="#F2000000"
        android:text="--/--/----"
        app:layout_constraintEnd_toStartOf="@+id/btnDateFinal"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/radioGroup2" />

    <TextView
        android:id="@+id/textView20"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:text="Fecha inicial"
        android:textColor="#B1B1B1"
        app:layout_constraintEnd_toEndOf="@+id/btnDateIni"
        app:layout_constraintStart_toStartOf="@+id/btnDateIni"
        app:layout_constraintTop_toBottomOf="@+id/btnDateIni" />

    <TextView
        android:id="@+id/textDateFinal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:text="Fecha final"
        android:textColor="#B1B1B1"
        android:visibility="visible"
```

Continuación de la figura 64.

```
            app:layout_constraintEnd_toEndOf="@+id/btnDateFinal"
            app:layout_constraintStart_toStartOf="@+id/btnDateFinal"
            app:layout_constraintTop_toBottomOf="@+id/btnDateFinal" />

        <TextView
            android:id="@+id/textViewHistEnergy"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="10dp"
            android:layout_marginTop="20dp"
            android:layout_marginEnd="10dp"
            android:background="@drawable/round_corners"
            android:backgroundTint="#B3000000"
            android:textAlignment="center"
            android:textColor="@color/white"
            android:textSize="24sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/btnConsult" />

        <TextView
            android:id="@+id/textViewHistCharge"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="10dp"
            android:layout_marginTop="20dp"
            android:layout_marginEnd="10dp"
            android:layout_marginBottom="10dp"
            android:background="@drawable/round_corners"
            android:backgroundTint="#B3000000"
            android:textAlignment="center"
            android:textColor="@color/white"
            android:textSize="34sp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textViewHistEnergy" />
    </androidx.constraintlayout.widget.ConstraintLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Fuente: elaboración propia, realizado con captura de pantalla.

Figura 65. **Código Java *fragment* ConsumptionHistory (App Android)**

ConsumptionHistory.java

```
1  package com.example.monitorenergtico;
2
3  import android.app.AlertDialog;
4  import android.app.DatePickerDialog;
5  import android.content.Context;
6  import android.content.SharedPreferences;
7  import android.os.Bundle;
8  import androidx.annotation.NonNull;
9  import androidx.annotation.Nullable;
10 import androidx.fragment.app.Fragment;
11 import androidx.fragment.app.FragmentResultListener;
12 import android.view.LayoutInflater;
13 import android.view.View;
14 import android.view.ViewGroup;
15 import android.widget.AdapterView;
16 import android.widget.ArrayAdapter;
17 import android.widget.Button;
18 import android.widget.RadioButton;
19 import android.widget.Spinner;
20 import android.widget.TextView;
21 import android.widget.Toast;
22 import java.net.NetworkInterface;
23 import java.net.SocketException;
24 import java.text.ParseException;
25 import java.text.SimpleDateFormat;
26 import java.util.ArrayList;
27 import java.util.Calendar;
28 import java.util.Collections;
29 import java.util.Date;
30 import java.util.List;
31 import java.util.Set;
32
33 public class ConsumptionHistory extends Fragment {
34
35     DatePickerDialog datePickerDialog;
36     SimpleDateFormat simpleDateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd");
37     Date dateIni, dateFinal;
38     Button btnDateIni, btnDateFinal, btnConsult;
39     RadioButton radioButtonDates, radioButtonDay;
40     TextView textViewDateFinal, textViewHistEnergy, textViewHistCharge;
41     Spinner spinnerDevices;
42
43     Boolean isDateIni;
44     String device, nameDevice;
45
```

Continuación de la figura 65.

```
46      public ConsumptionHistory() {
47          // Required empty public constructor
48      }
49
50      @Override
51      public void onCreate(Bundle savedInstanceState) {
52          super.onCreate(savedInstanceState);
53
54          getParentFragmentManager().setFragmentResultListener( requestKey: "history",
55                  lifecycleOwner: this, new FragmentResultListener() {
56              @Override
57              public void onFragmentResult(@NonNull String requestKey,
58                                           @NonNull Bundle result) {
59                  textViewHistEnergy.setText(result.getString( key: "energy"));
60                  textViewHistCharge.setText(result.getString( key: "charge"));
61              }
62          });
63      }
64
65      @Override
66      public View onCreateView(LayoutInflater inflater, ViewGroup container,
67                               Bundle savedInstanceState) {
68          // Inflate the layout for this fragment
69          return inflater.inflate(R.layout.fragment_consumption_history, container,
70                  attachToRoot: false);
71      }
72
73      @Override
74      public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
75          super.onViewCreated(view, savedInstanceState);
76          btnDateIni = view.findViewById(R.id.btnDateIni);
77          btnDateFinal = view.findViewById(R.id.btnDateFinal);
78          btnConsult = view.findViewById(R.id.btnConsult);
79          radioButtonDates = view.findViewById(R.id.radioButtonDates);
80          radioButtonDay = view.findViewById(R.id.radioButtonDay);
81          textViewDateFinal = view.findViewById(R.id.textDateFinal);
82          textViewHistEnergy = view.findViewById(R.id.textViewHistEnergy);
83          textViewHistCharge = view.findViewById(R.id.textViewHistCharge);
84          spinnerDevices = view.findViewById(R.id.spinnerDevices);
85          initDatePicker();
86          initSpinner();
87
88          btnDateIni.setOnClickListener(view1 ->{
89              isDateIni=true;
90              datePickerDialog.show();
```

Continuación de la figura 65.

```
91          });
92
93          btnDateFinal.setOnClickListener(view1 ->{
94              isDateIni=false;
95              datePickerDialog.show();
96          });
97
98          radioButtonDates.setOnClickListener(view1 -> {
99              btnDateFinal.setVisibility(View.VISIBLE);
100             textViewDateFinal.setVisibility(View.VISIBLE);
101         });
102
103         radioButtonDay.setOnClickListener(view1 -> {
104             btnDateFinal.setVisibility(View.GONE);
105             textViewDateFinal.setVisibility(View.GONE);
106         });
107
108         btnConsult.setOnClickListener(view1 -> {
109
110             if ((radioButtonDates.isChecked() && (dateIni==null || dateFinal==null))
111                 || (radioButtonDay.isChecked() && dateIni==null)) {
112                 Toast.makeText(getActivity(), text: "Debe llenar todos los campos",
113                         Toast.LENGTH_SHORT).show();
114             } else{
115                 if (radioButtonDates.isChecked()){
116                     if (dateIni.before(dateFinal)){
117                         consult( isDay: false);
118                     } else {
119                         Toast.makeText(getActivity(),  text: "La fecha inicial debe ser " +
120                                 "mayor a la final", Toast.LENGTH_LONG).show();
121                     }
122                 } else if (radioButtonDay.isChecked()){
123                     consult( isDay: true);
124                 }
125             }
126         });
127     }
128
129     @Override
130     public void onStop() {
131         super.onStop();
132         dateIni=null;
133         dateFinal=null;
134         btnDateIni.setText("--/--/--");
135         btnDateFinal.setText("--/--/--");
```

Continuación de la figura 65.

```
136         radioButtonDates.setChecked(true);
137         btnDateFinal.setVisibility(View.VISIBLE);
138         textViewDateFinal.setVisibility(View.VISIBLE);
139     }
140
141     private void consult(boolean isDay){
142         Bundle bundle = new Bundle();
143         bundle.putString("device",device);
144         bundle.putString("name",nameDevice);
145         bundle.putString("dateIni",simpleDateFormat.format(dateIni));
146         if (isDay){
147             bundle.putString("type","req-data-day");
148         }else{
149             bundle.putString("type","req-data");
150             bundle.putString("dateFinal",simpleDateFormat.format(dateFinal));
151         }
152         getParentFragmentManager().setFragmentResult( requestKey: "consult",bundle);
153     }
154
155     private void initSpinner() {
156         //get devices
157         SharedPreferences preferences= getActivity().getSharedPreferences(
158                 name: "devices", Context.MODE_PRIVATE);
159         ArrayList arrayListDevices = new ArrayList();
160         Set<String> setArray=null;
161         try {
162             setArray = preferences.getStringSet( s: "data",  set: null);
163             arrayListDevices = new ArrayList<String>(setArray);
164         }catch (Exception e){
165
166         }
167         ArrayList arrayListShowDevi = new ArrayList();
168         ArrayList arrayListDevi = new ArrayList();
169         for(Object item:arrayListDevices){
170             String[] nameDevi = item.toString().split( regex: "\n");
171             arrayListShowDevi.add(nameDevi[0]);
172             arrayListDevi.add(nameDevi[1]);
173         }
174         device = (String) arrayListDevi.get(0);
175         nameDevice = (String) arrayListShowDevi.get(0);
176         //Fill payday spinner
177         ArrayAdapter<CharSequence> adapterSpi = new ArrayAdapter(getActivity(),
178                 R.layout.spinner_item,arrayListShowDevi);
179         adapterSpi.setDropDownViewResource(R.layout.spinner_item);
180         spinnerDevices.setAdapter(adapterSpi);
```

Continuación de la figura 65.

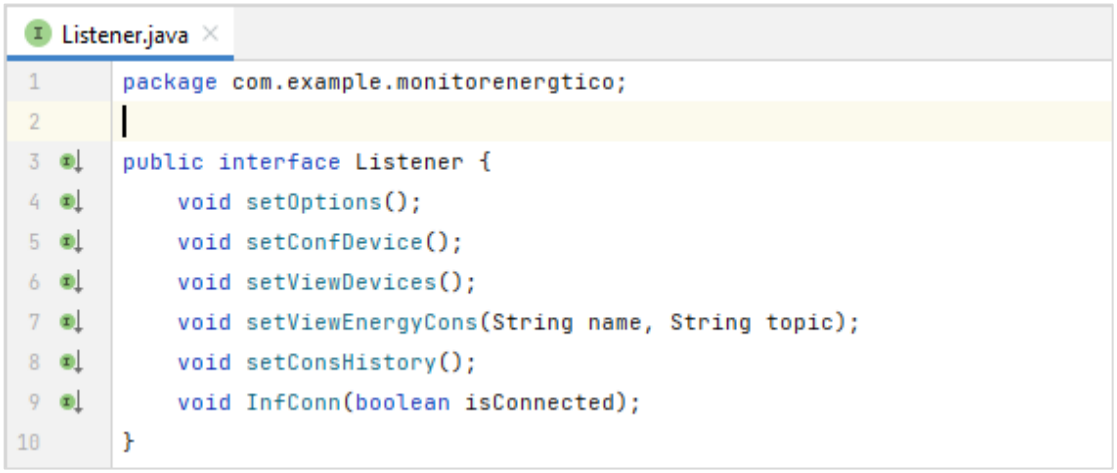
```
        spinnerDevices.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> adapterView, View view,
                                       int i, long l) {
                device = (String) arrayListDevi.get(i);
                nameDevice = (String) adapterView.getItemAtPosition(i);
            }
            @Override
            public void onNothingSelected(AdapterView<?> adapterView) { }
        });
    }

    private void initDatePicker() {
        DatePickerDialog.OnDateSetListener dateSetListener = (datePicker, year,
                                                              month, day) -> {
            String date = (day+"/"+(month+1)+"/"+year);
            if (isDateIni) { //Button Date Init was presed
                btnDateIni.setText(date);
                try {
                    dateIni = simpleDateFormat.parse( source: year+"-"+(month+1)+"-"+day);
                } catch (ParseException e) {
                    e.printStackTrace();
                }
            }else{ //Button Date Final was presed
                btnDateFinal.setText(date);
                try {
                    dateFinal = simpleDateFormat.parse( source: year+"-"+(month+1)+"-"+day);
                } catch (ParseException e) {
                    e.printStackTrace();
                }
            }
        };

        Calendar calendar = Calendar.getInstance();
        int year = calendar.get(Calendar.YEAR);
        int mounth = calendar.get(Calendar.MONTH);
        int day = calendar.get(Calendar.DAY_OF_MONTH);
        int style = AlertDialog.THEME_HOLO_DARK;
        datePickerDialog = new DatePickerDialog(getActivity(),style,dateSetListener,
                year,mounth,day);
    }
}
```

Fuente: elaboración propia, realizado con captura de pantalla.

5.2.2.8. Interfaz Java Listener

Los métodos de esta interfaz son implementados en la clase MainActivity (*fragment* menú principal), y sirven para sustituir los *fragmens* del contenedor principal, los métodos se llaman desde las clases que forman parte de los diferentes menús de la aplicación y dependiendo de la opción elegida por el usuario se llama al método definido para mostrar el *fragment* correspondiente a dicha opción.

Figura 66. **Código Java interfaz Listener**



Fuente: elaboración propia, realizado con captura de pantalla.

5.2.2.9. Clase Java SecureData

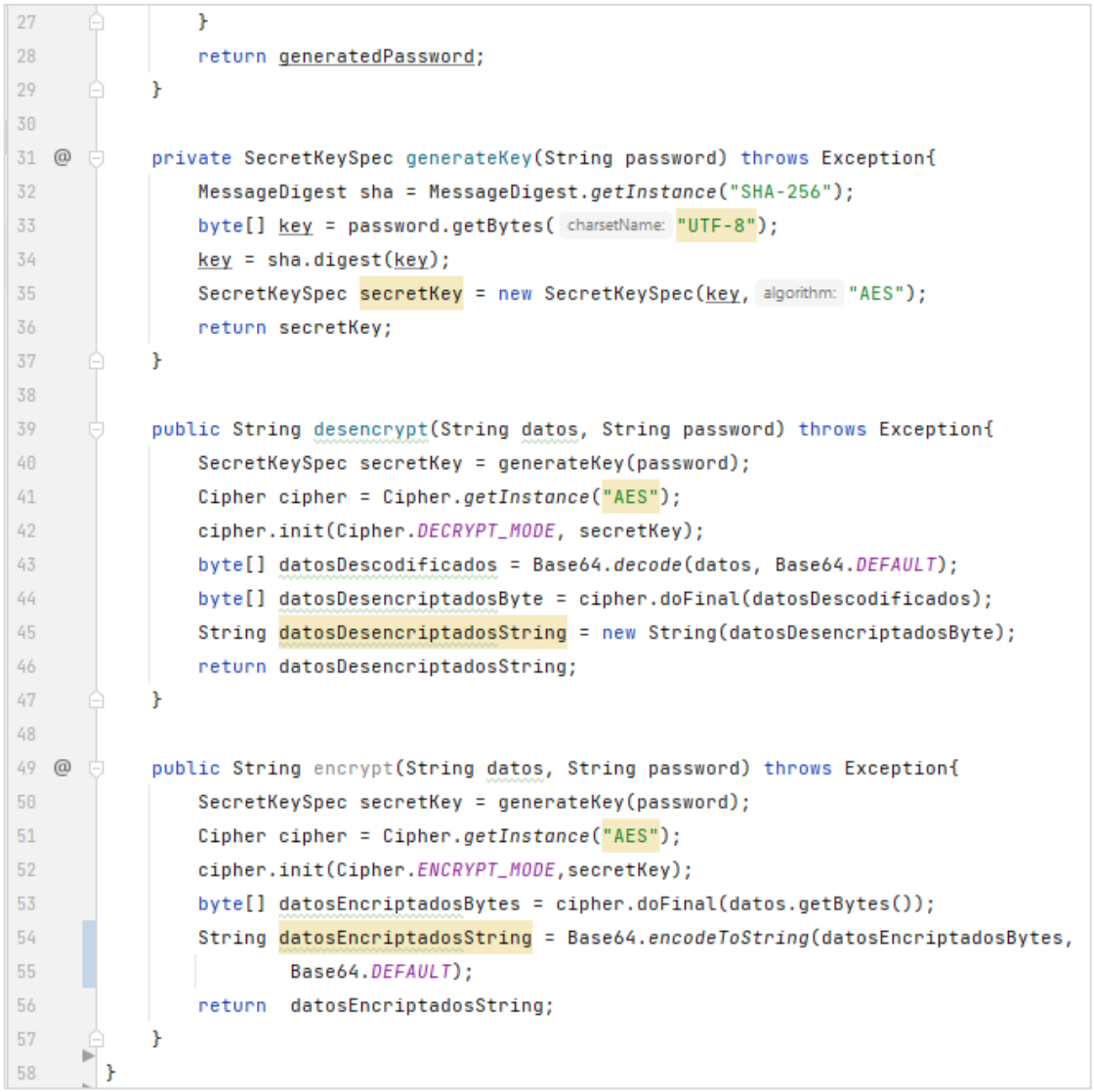
Esta clase es utilizada para cifrar y descifrar información sensible como lo son las credenciales de conexión, al momento de tratar de establecer conexión MQTT por medio de esta clase se lee un archivo cifrado que contiene las credenciales de conexión para ser desencriptadas y así poder obtener su información en texto plano con el fin de autenticar el acceso al servidor MQTT. Los métodos de esta clase implementan los algoritmos AES y SHA 512 para realizar las funciones de criptografía requeridas.

Figura 67. **Código Java clase SecureData**

SecureData.java

```
package com.example.monitorenergtico;

import android.util.Base64;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class SecureData {

    public String get_SHA_512(String passwordToHash){

        String generatedPassword = null;
        String salt="Qw0JzV7Gbq*BkP&rz$SJv^O6uv1oZEZp2nYLQXN#qTb11FFQ3w";
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-512");
            md.update(salt.getBytes());
            byte[] bytes = md.digest(passwordToHash.getBytes());
            StringBuilder sb = new StringBuilder();
            for(int i=0; i< bytes.length ;i++){
                sb.append(Integer.toString( t (bytes[i]&0xff)+0x100, radix 16).substring(1));
            }
            generatedPassword = sb.toString();
        }
        catch (NoSuchAlgorithmException e){
            e.printStackTrace();
        }
```

Continuación de la figura 67.



```
        }
        return generatedPassword;
    }

    private SecretKeySpec generateKey(String password) throws Exception{
        MessageDigest sha = MessageDigest.getInstance("SHA-256");
        byte[] key = password.getBytes( charsetName: "UTF-8");
        key = sha.digest(key);
        SecretKeySpec secretKey = new SecretKeySpec(key, algorithm: "AES");
        return secretKey;
    }

    public String desencrypt(String datos, String password) throws Exception{
        SecretKeySpec secretKey = generateKey(password);
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.DECRYPT_MODE, secretKey);
        byte[] datosDescodificados = Base64.decode(datos, Base64.DEFAULT);
        byte[] datosDesencriptadosByte = cipher.doFinal(datosDescodificados);
        String datosDesencriptadosString = new String(datosDesencriptadosByte);
        return datosDesencriptadosString;
    }

    public String encrypt(String datos, String password) throws Exception{
        SecretKeySpec secretKey = generateKey(password);
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.ENCRYPT_MODE,secretKey);
        byte[] datosEncriptadosBytes = cipher.doFinal(datos.getBytes());
        String datosEncriptadosString = Base64.encodeToString(datosEncriptadosBytes,
                Base64.DEFAULT);
        return  datosEncriptadosString;
    }
}
```

Fuente: elaboración propia, realizado con captura de pantalla.

6. SERVIDOR REMOTO

Un servidor es básicamente un equipo de cómputo destinado a recibir y resolver peticiones. Cuando un servidor no forma parte de la misma red local que el usuario, pero aun así es alcanzable utilizando un enrutador conectado a la red de Internet, entonces se dice que el servidor es remoto, esto abre la posibilidad de que el servidor pueda encontrarse en un lugar geográficamente distinto al usuario.

Con el desarrollo de las tecnologías de la información nace la computación en la nube que es un servicio en donde los recursos se encuentran distribuidos horizontalmente y localizados en un centro de datos, este servicio brinda la capacidad de tener un espacio virtual en el centro de datos con la finalidad de usar los recursos de esa porción de poder computacional para propósitos puntuales como pueden ser: almacenamiento de datos, alojamiento web, soporte de aplicaciones web, ciencia de datos, entre otros. La contratación de un servicio como este ahorra al usuario la problemática de tener la infraestructura y el conocimiento para montar un servidor equipado con un sistema óptimo que le permita manejar las peticiones de la mejor forma posible. Algunos de los proveedores que ofrecen computación en la nube son:

- Amazon AWS: perteneciente a Amazon cuenta con más de 100 productos de paga basados en computación en la nube, algunos de estos productos cuentan con pruebas gratuitas que permiten conocer de una mejor forma el servicio para luego evaluar si se adapta a la necesidad o bien es necesaria otra solución, AWS también cuenta con escalabilidad lo que permite aumentar los recursos computacionales contratados.

- Google Cloud: Plataforma lanzada el 7 de abril de 2008. Entre sus más de 100 productos los más destacados son Compute Engine, Cloud SQL, Dataflow, Cloud Storage, entre otros. Al ser un servicio perteneciente a una empresa de renombre es muy popular en el medio, así como también por su innovación y desarrollo tecnológico.

- DigitalOcean: es un proveedor de origen estadunidense que cuenta con variedad de productos y servicios entre los más notables se encuentra los *Droplets* que ofrecen máquinas virtuales escalables ofreciendo una dirección IP pública. Cuenta con centros de datos en ciudades como: New York, San Francisco, Ámsterdam, Singapur, Frankfurt, Toronto y Bangalore.

- Alibaba Cloud: de origen asiático y al igual que sus competidores también cuenta con más de 100 productos disponibles para soluciones individuales y empresariales, Ofrece pruebas gratuitas y descuentos a nuevos usuarios.

- Microsoft Azure: perteneciente al gigante de la computación Microsoft, es una de las principales plataformas usadas en el desarrollo de aplicaciones IoT debido a la cantidad de Appis que implementa en sus servicios, además de contar con algunos servicios gratuitos durante determinado tiempo y de por vida.

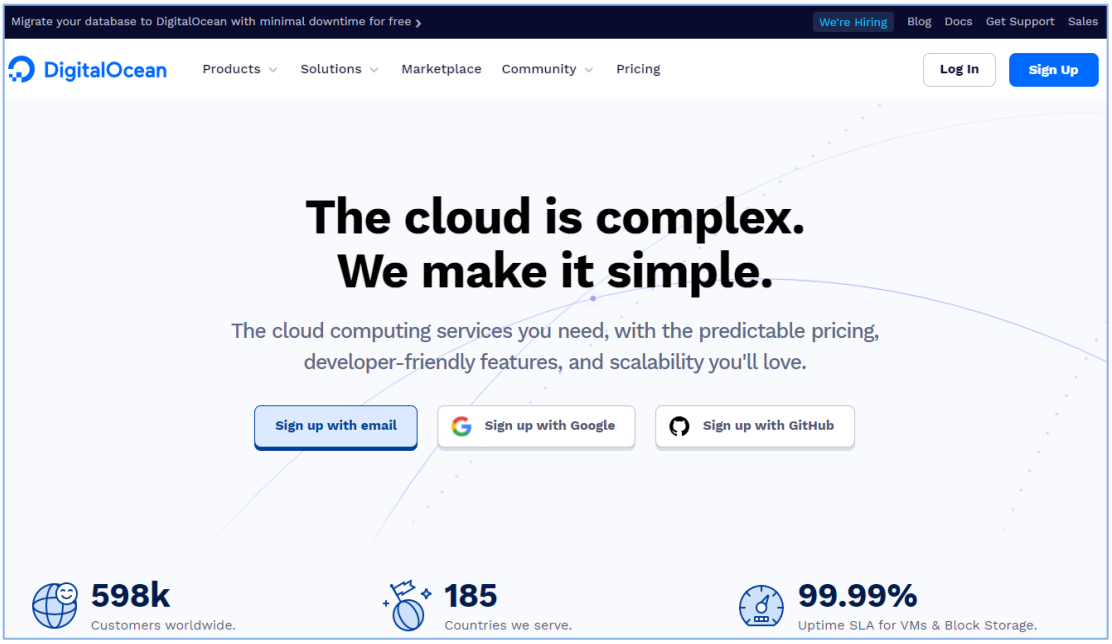
6.1. Creación de máquina virtual en DigitalOcean

Para manejar los servicios de la aplicación presentada en este trabajo de graduación se ha decidido optar por los recursos de DigitalOcean para adquirir una máquina virtual la cual se configuró de la siguiente manera:

Para contratar uno de los productos que ofrece DigitalOcean primero se debe ir al sitio web (figura 68), y crear una cuenta o iniciar sesión en caso ya se tenga una, también dispone de otras opciones de inicio de sesión como lo son por medio de un correo electrónico válido, una cuenta Google o GitHub.

El sitio web de DigitalOcean (https://www.digitalocean.com), cuenta con tutoriales totalmente gratuitos donde se puede encontrar información variada de programación, bases de datos, instalación de paquetes, administración de los distintos productos disponibles, entre otros.

Figura 68. **Sitio web DigitalOcean**

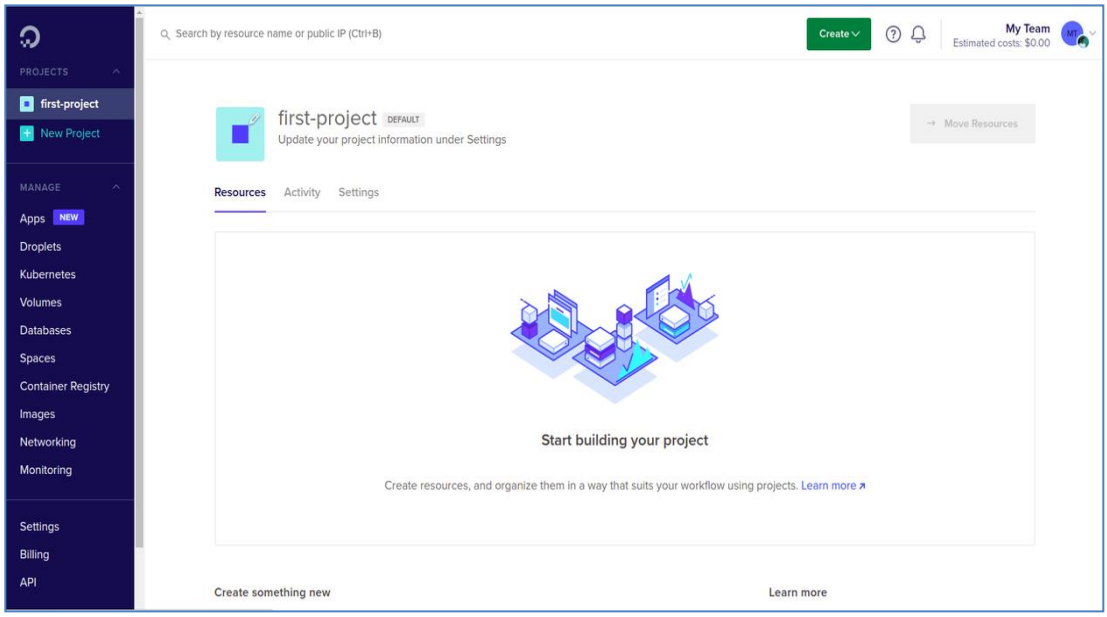


Fuente: elaboración propia, realizado con captura de pantalla.

Cuando ya se haya iniciado la sesión el sitio web presentará una página como se muestra en la figura 69 donde se dispone de un panel con distintas

opciones que permiten crear y administrar nuevos proyectos, recursos, direcciones IP flotantes, configuración DNS, cortafuegos, entre otras.

Figura 69. **Página mostrada después de iniciar sesión en DigitalOcean**

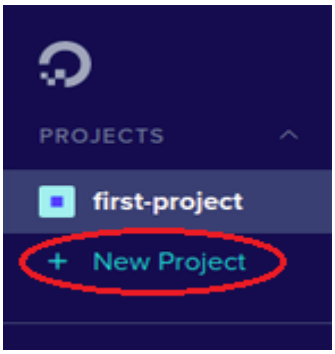


Fuente: elaboración propia, realizado con captura de pantalla.

6.1.1. Creación de nuevo proyecto en DigitalOcean

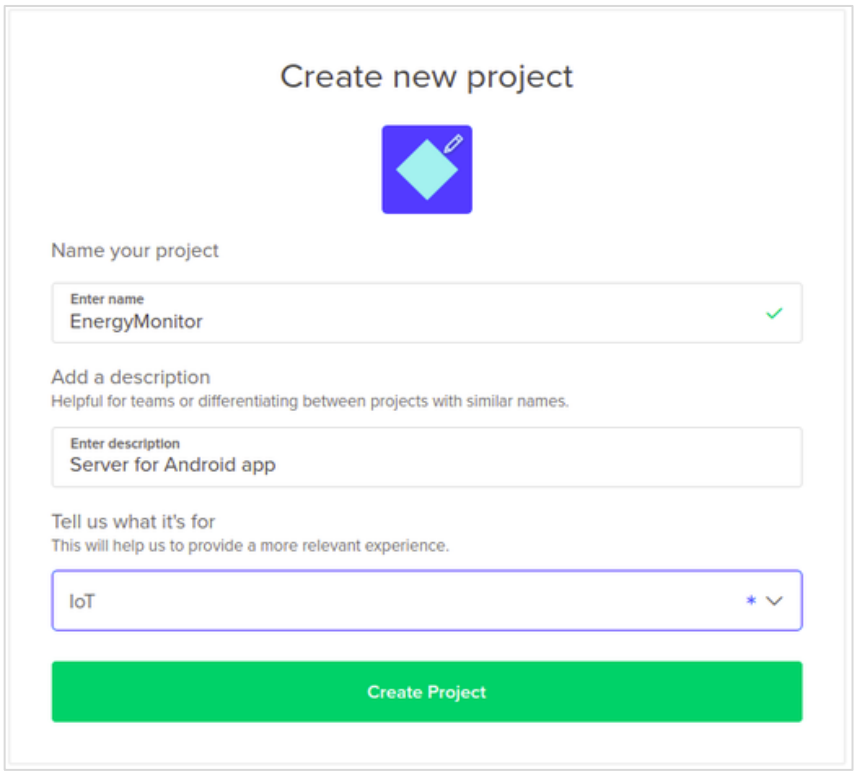
Para organizar de mejor manera los recursos destinados a la aplicación se puede hacer uso de los proyectos, que en DigitalOcean son un espacio destinado a agrupar recursos que comparten un mismo propósito, para crear un nuevo proyecto se debe de hacer clic en el botón con el texto *New Project,* el cual se encuentra en la parte superior izquierda de la página.

Figura 70. **Botón crear nuevo proyecto DigitalOcean**



Fuente: elaboración propia, realizado con captura de pantalla.

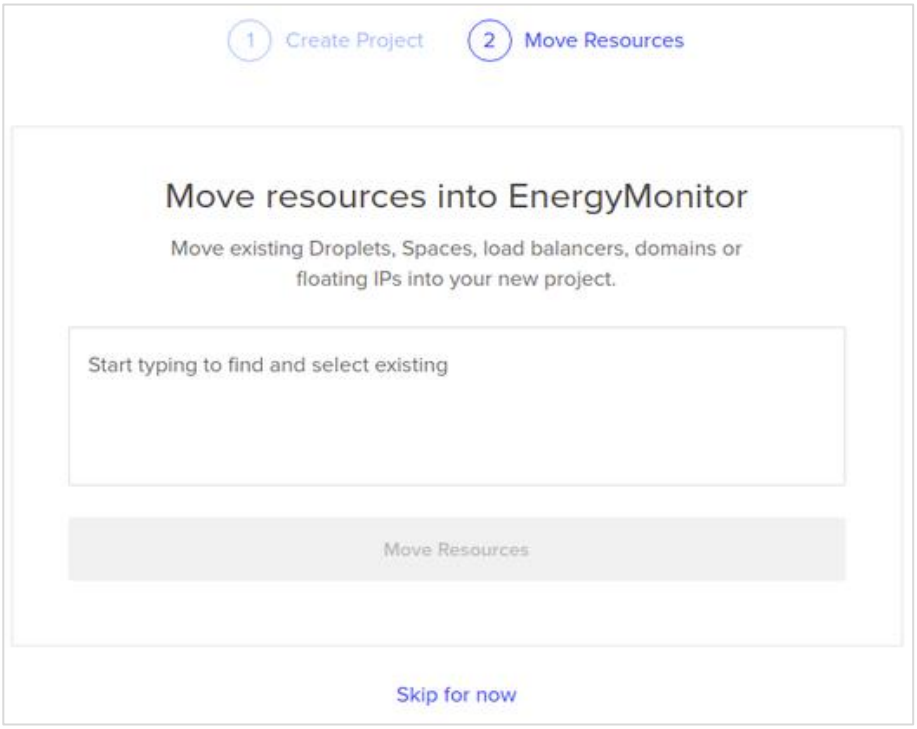
Figura 71. **Nuevo proyecto DigitalOcean**



Fuente: elaboración propia, realizado con captura de pantalla.

Después de hacer clic en el botón *New Project,* se mostrará una ventana (figura 71), en donde se encuentra un formulario, el cual tiene 3 campos que son: nombre, descripción y aplicación, para este proyecto se asignará el nombre: EnergyMonitor, la descripción: *Server for Android app*, y la aplicación: IoT. Después de haber rellenado los campos del formulario se procede a hacer clic en el botón *Create Project,* y hará que el sitio muestre el apartado llamado *Move Resources* (figura 72), que está destinado a asignarle recursos previamente creados al nuevo proyecto.

Figura 72. **Agregar recursos a nuevo proyecto DigitalOcean**



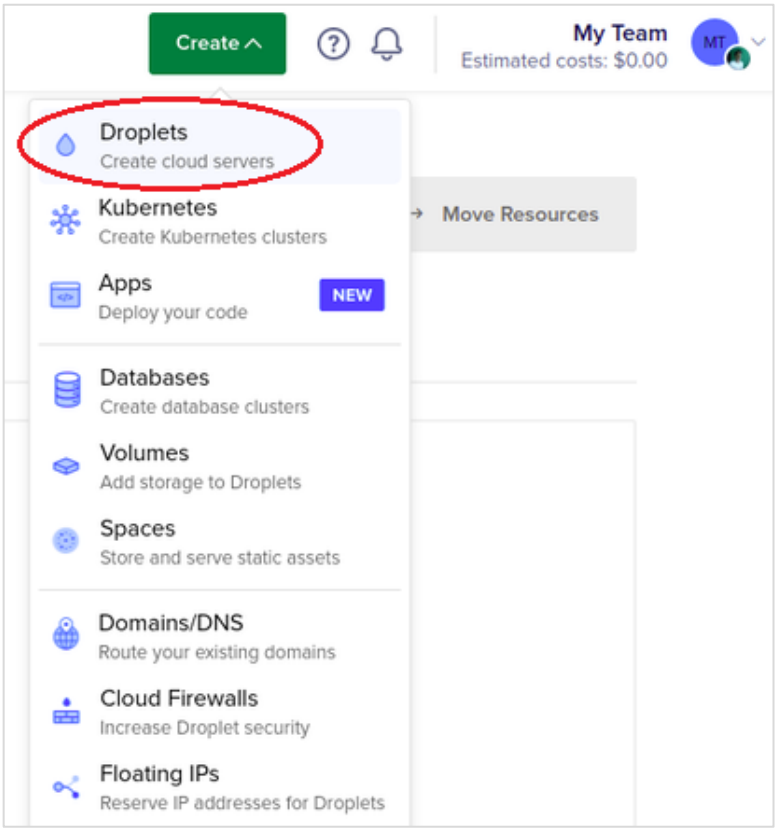
Fuente: elaboración propia, realizado con captura de pantalla.

Suponiendo que aún no se cuenta con los recursos previamente creados se hace clic en el botón *Skip for now.*

6.1.2. Creación de recurso *Droplet* en DigitalOcean

Para crear un recurso se debe hacer clic en el botón *Create,* (figura 73), ubicado en la parte superior derecha, se elige el recurso llamado *Droplets* con descripción *Create cloud servers*, que básicamente es un producto que ofrece una máquina virtual en la nube con varias opciones de configuración y la posibilidad de poder escalar los recursos en caso la aplicación tenga un crecimiento que requiera de un poder computacional mayor para resolver todas las peticiones realizadas por los usuarios.

Figura 73. **Crear recursos DigitalOcean**



Fuente: elaboración propia, realizado con captura de pantalla.

Luego de hacer clic en el botón para crear *Droplets* se mostrará una página de configuración para la máquina virtual que se está a punto de crear.

6.1.2.1. Elección de imagen

En el apartado *choose an imagen,* (figura 74), se elige la imagen que realizará la instalación del sistema operativo de la máquina virtual, se puede cargar una imagen personalizada o usar las imágenes que actualmente (enero 2022), se encuentran disponibles, estas son:

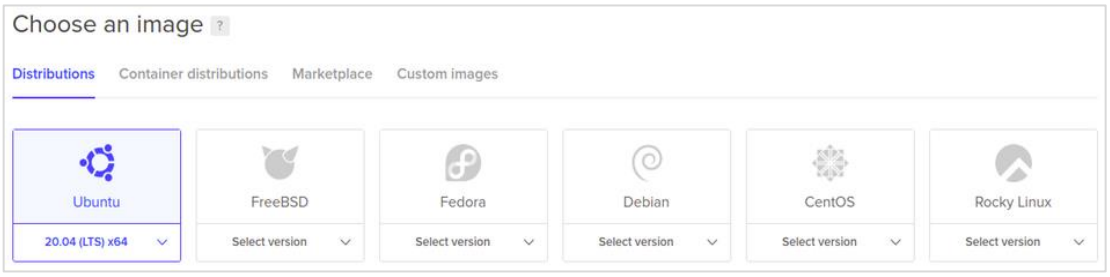
- Ubuntu
    - 21.10 x64
    - 20.04 (LTS) x64
    - 18.04 (LTS) x64
- FreeBSD
    - 12.2 zfs x64
    - 12.2 ufs x64
- Fedora
    - 35 x64
    - 34 x64
- Debian
    - 11 x64

- 10 x64

- CentOS
  - 8 Stream x64
  - 7 x64

- Rocky Linux
  - 8.5 x64
  - 8.4 x64

Figura 74. **Selección de imagen para *droplets* DigitalOcean**



Fuente: elaboración propia, realizado con captura de pantalla.

Para la creación del servidor de este proyecto se ha decidido optar por Ubuntu 20.04 (TLS) x64.

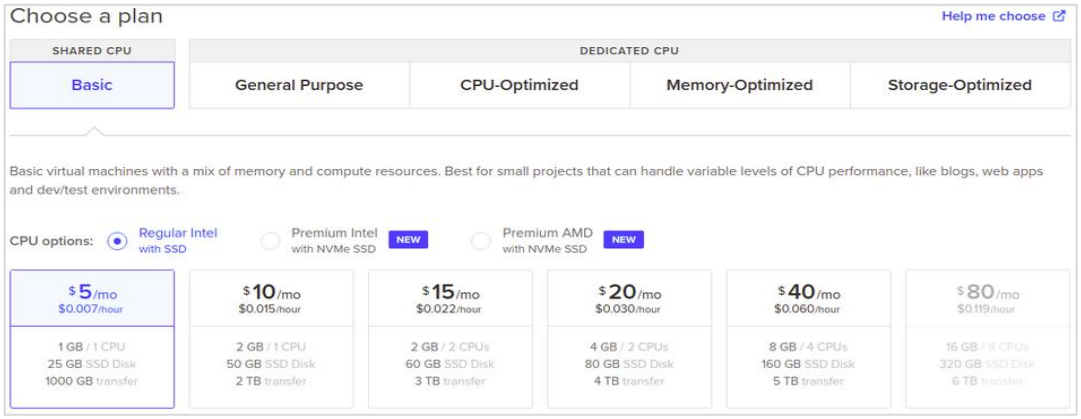
6.1.2.2. Elección de plan

En el apartado *choose a plan,* (figura 75), se configura el hardware que tendrá disponible la máquina virtual y se encuentra organizado según el objetivo

de la aplicación, cada grupo cuenta con diferentes configuraciones. La forma en la que se organiza por aplicación es la siguiente:

- Básica (CPU compartido)
    - Procesador regular Intel con SSD
    - Procesador premium Intel con NVMe SSD
    - Procesador premium AMD con NVMe SSD
- Propósito general (CPU dedicado)
    - Opción de multiplicar al doble el tamaño del disco SSD.
- CPU Optimizado (CPU dedicado)
    - Opción de multiplicar al doble el tamaño del disco SSD.
- Memoria Optimizada (CPU dedicado)
    - Opción de multiplicar al triple o séxtuple el tamaño del disco SSD
- Almacenamiento optimizado (CPU dedicado)
    - Opción de multiplicar al 1.5 el tamaño del disco SSD

Figura 75. **Elección de plan *droplets* DigitalOcean**



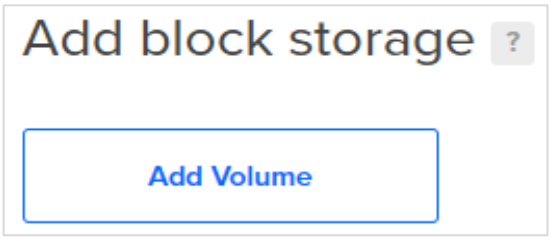
Fuente: elaboración propia, realizado con captura de pantalla.

El plan elegido para esta aplicación es: básico, procesador regular Intel con SSD, $5 por mes.

6.1.2.3. Agregar almacenamiento extra

El siguiente apartado de configuración es llamado *add block storage,* (figura 76), que consiste en agregar almacenamiento extra. Para esta aplicación no se optará por ese recurso.

Figura 76. **Agregar almacenamiento extra *droplets* DigitalOcean**

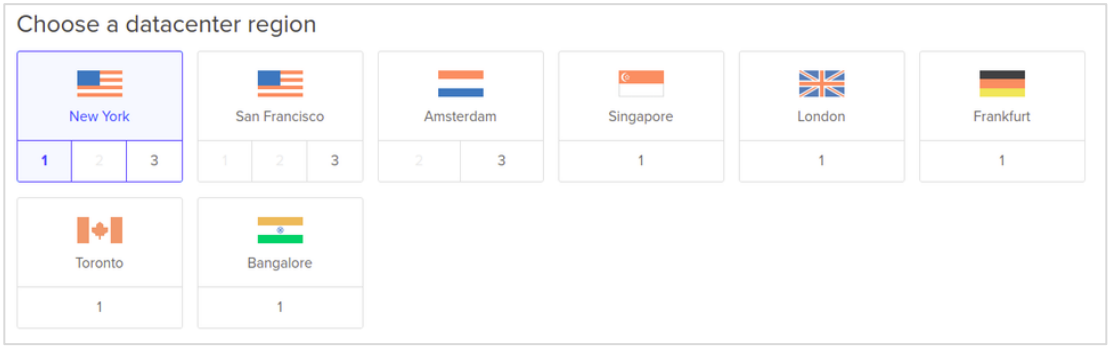


Fuente: elaboración propia, realizado con captura de pantalla.

6.1.2.4. Elección centro de datos

Para seleccionar el lugar geográfico donde se ubicará el servidor es necesario elegir la ciudad de ubicación del centro de datos de interés en el apartado *choose a datacenter region*, la figura 77 muestra las ciudades disponibles y los espacios que son elegibles para reservar.

Figura 77. **Región centro de datos Droplets DigitalOcean**



Fuente: elaboración propia, realizado con captura de pantalla.

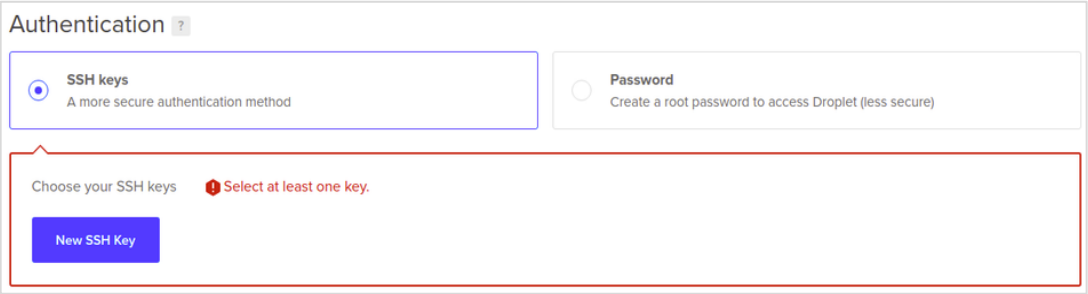
Para la máquina virtual de esta aplicación se ha elegido el centro de datos 1 de New York.

6.1.2.5. Tipo de autenticación

Para establecer las configuraciones de acceso remoto al sistema operativo de la máquina virtual se tiene el apartado *authentication,* (figura 78), que brinda las opciones de establecer una contraseña para validar los inicios de sesión o usar claves SSH opción recomendada por ser la más segura, aunque esta opción lleva más trabajo ya que si aún no se tiene creada una clave es necesario crearla e importarla, para importar claves se hace clic en el botón *New SSH Key* y este

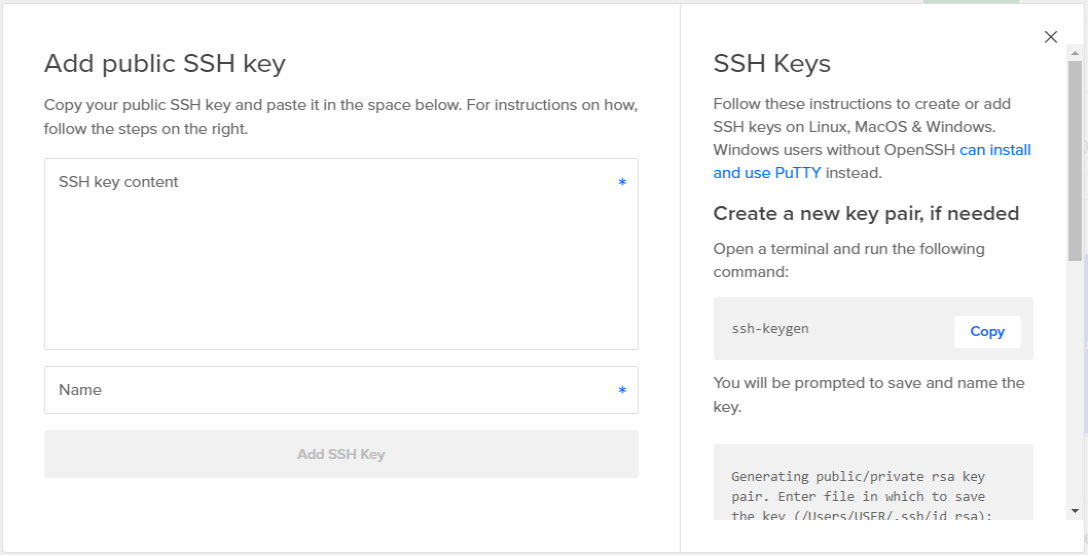
abrirá una ventana (figura 79), en la que se encuentra un apartado para copiar la clave y asignarle un nombre, también en la misma ventana de lado derecho se encuentra un instructivo de cómo crear pares de claves SSH en entornos Linux, MacOS y Windows.

Figura 78. **Autenticación *droplets* DigitalOcean**



Fuente: elaboración propia, realizado con captura de pantalla.

Figura 79. **Agregar clave SSH *droplets* DigitalOcean**



Fuente: elaboración propia, realizado con captura de pantalla.

6.1.2.6. Generación de par de claves SSH en Ubuntu

El proceso para genera un par de claves SSH en el sistema operativo Ubuntu y luego importarlas al apartado de autenticación de DigitalOcean.

- Paso 1: abrir una terminal escribiendo la combinación de teclas Ctrl + Alt + t, y escribir el comando ssh-keygen.

- Paso 2: asignar un nombre al par de claves SSH.

- Paso 3: establecer una contraseña para el uso de la clave privada, este paso es opcional y si no se desea establecer una contraseña bastará con dejar en blanco el campo y pulsar la tecla *Enter*.

    - Paso 3.1: si se ha establecido una contraseña para el uso de la clave privada por medio de la línea de comandos se solicitará volver a ingresar la contraseña, si las contraseñas son diferentes no se procederá a generar las claves hasta que estas coincidan.

- Paso 4: por medio de la línea de comandos se le informará al usuario que el par de claves ha sido generado exitosamente.

Tabla III. **Comando para generar par de calves SSH Ubuntu**

Entrada	Salida
ssh-keygen	Se solicita al usuario un nombre para el par de claves: *Generating public/private rsa key pair.* *Enter file in which to save the key (/root/.ssh/id_rsa):*
	Se solicita un pase para usar la clave (si se deja en blanco no se colocará ninguno). *Enter passphrase (empty for no passphrase)*
	Si se ingresó una clave se solicita la confirmación de esta: *Enter the same passphrase againg:*
	Si todo esta correcto se procede a generar el par de claves.

Fuente: elaboración propia.

Figura 80. **Generación de par de claves SSH en Ubuntu**

Fuente: elaboración propia, realizado con captura de pantalla.

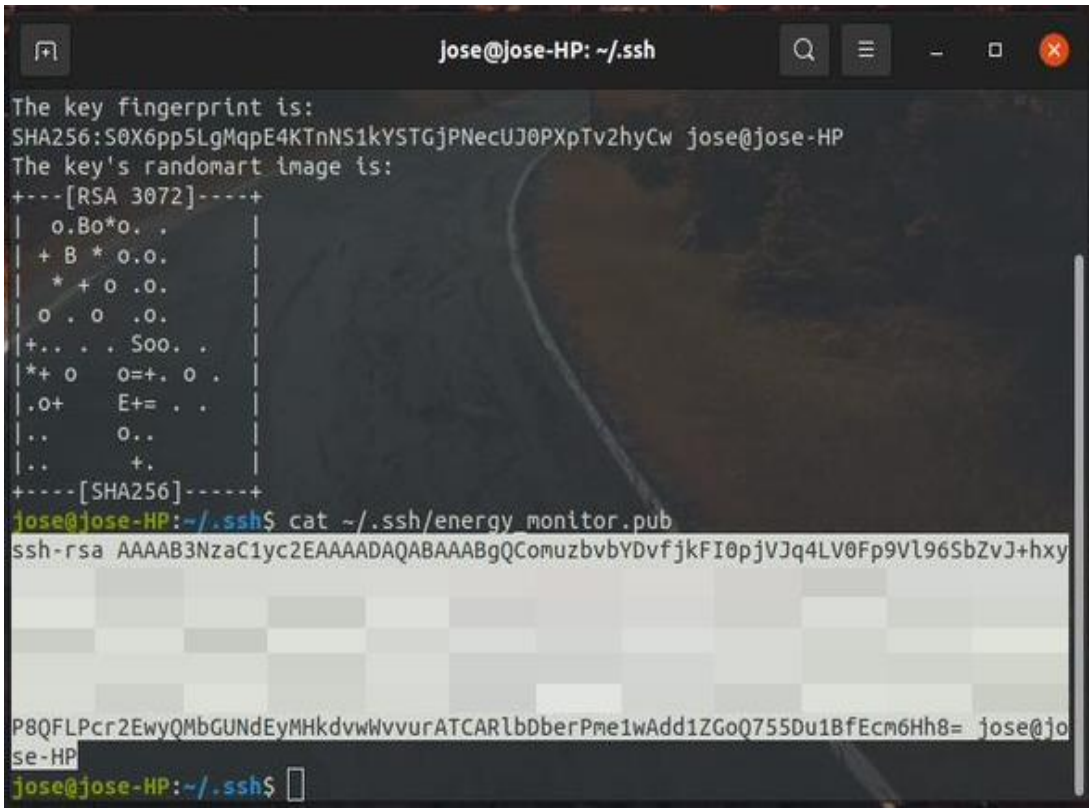
- Paso 5: mostrar la clave pública por medio de la terminal y el comando cat, después seleccionar y copiar el texto arrojado por la salida del comando.

Tabla IV. **Comando mostrar clave pública SSH con comando cat**

Entrada	Salida
cat ~/.shh/nombre_clave.pub	texto de clave pública

Fuente: elaboración propia, realizado con captura de pantalla.

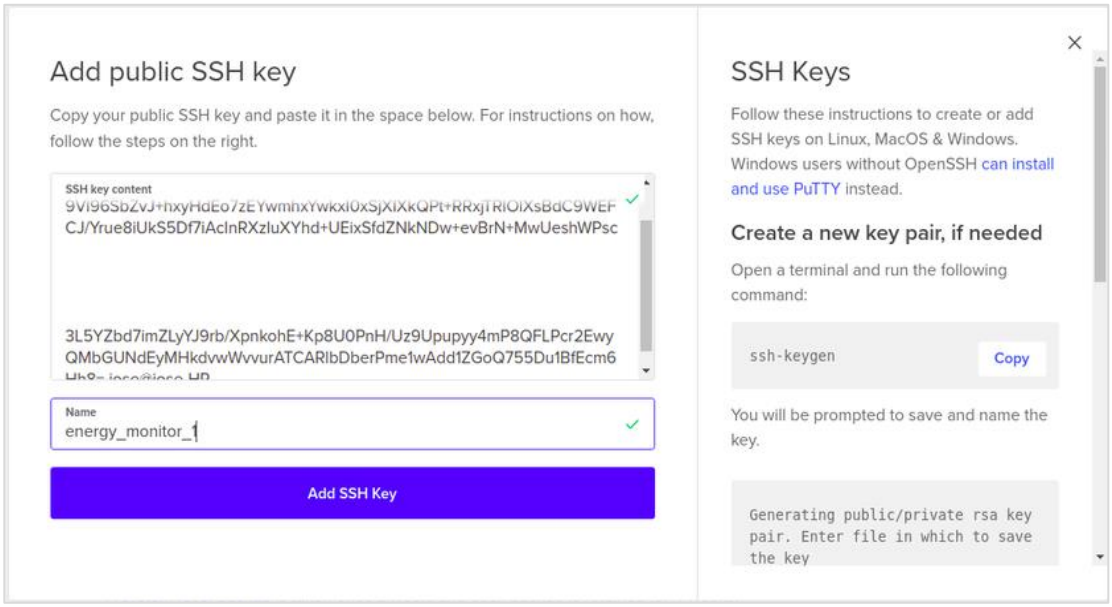
Figura 81. **Mostrar clave pública SSH en terminal Ubuntu**



Fuente: elaboración propia, realizado con captura de pantalla.

- Paso 6: pegar el texto contenido en la clave pública en el campo *SSH key content*, de la ventana web *add public SS key,* luego en el campo Name, de la misma ventana web asignar un nombre identificador de clave, cabe recalcar que esta clave no se limita únicamente a funcionar como autenticación de un solo recurso y por el contrario puede ser usada como medio de autenticación de otros recursos independiente del proyecto donde estén asignados.

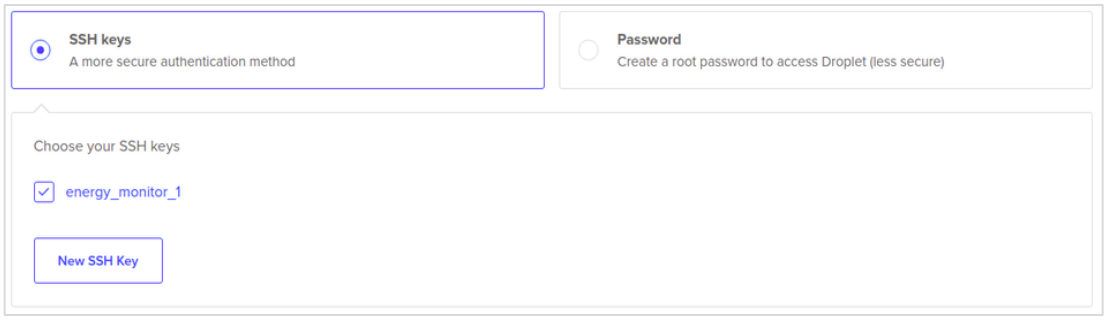
Figura 82. **Clave SSH agregada a DigitalOcean**



Fuente: elaboración propia, realizado con captura de pantalla.

- Paso 7: guardar la clave haciendo clic en el botón Add SSH Key, si todas las entradas son correctas la clave SSH estará disponible para ser usada como medio de autenticación para los inicios de sesión tal como lo muestra la figura 83.

Figura 83. **Selección de calve SSH para autenticación Droplets DititalOcean**



Fuente: elaboración propia, realizado con captura de pantalla.

6.1.2.7. Opciones adicionales

La siguiente opción de configuración es la mostrada en el apartado *select additional options,* en donde es posible agregar las siguientes opciones adicionales:

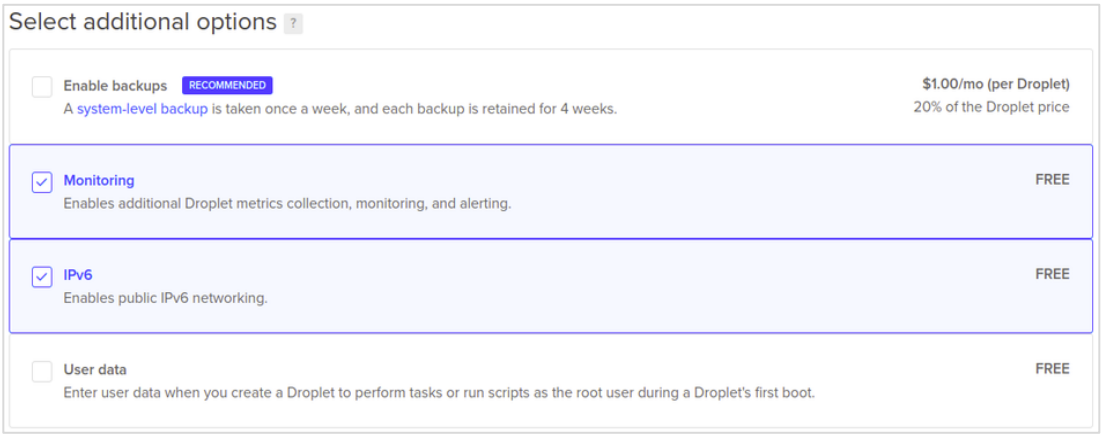
- *Enable backups*: esta es una opción de paga y sirve para realizar copias de seguridad una vez cada semana.

- *Monitoring*: opción gratuita que perímete obtener métricas, monitoreo y alertas.

- IPv6: habilita una red IPv6 de manera gratuita

- *User data*: Ingresa información para ejecutar tareas o *scripts* durante el primer arranque.

Figura 84. **Opciones adicionales Droplets DigitalOcean**



Fuente: elaboración propia, realizado con captura de pantalla.

6.1.2.8. Finalización y creación

En el último apartado de configuración llamado *finalize and créate*, se presentan las siguientes opciones:

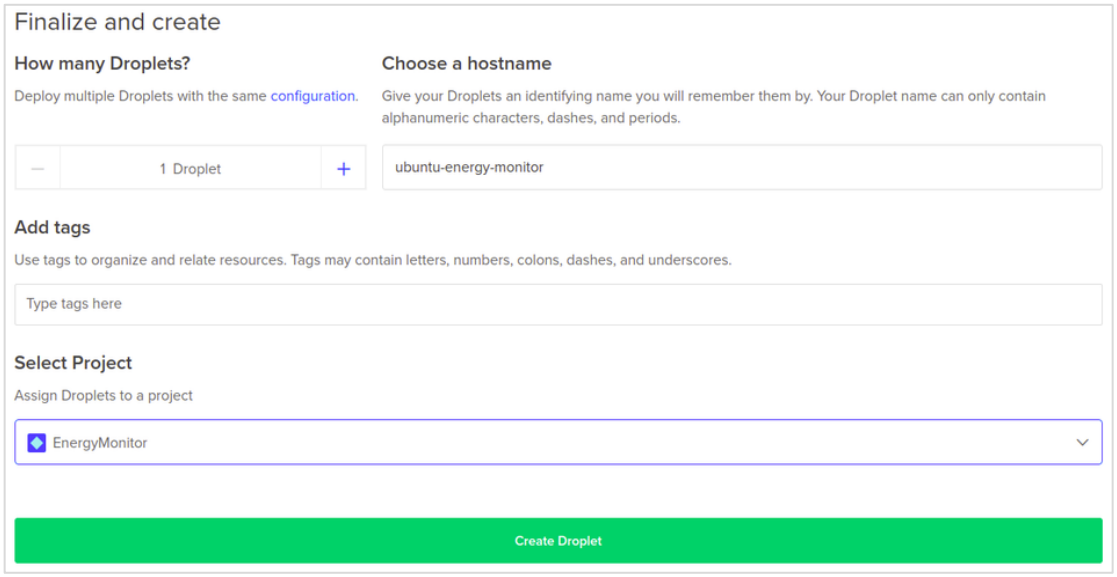
- *How many Droplets?* con esta opción se puede crear múltiples *droplets* con la misma configuración.

- *Choose a hostaneme*: todos los *droplets* necesitan tener un nombre que los identifique y es en este campo en donde se les asigna dicho nombre.

- *Add tags*: esta opción permite agregar etiquetas para organizar y relacionar recursos.

- *Select Project*: aquí se elige a que proyecto se va a asignar el *droplet* que se va a crear.

Figura 85. **Finalizar y crear Droplets DigitalOcean**



Fuente: elaboración propia, realizado con captura de pantalla.

Para este proyecto se usó un *droplet*, con el nombre Ubuntu-energy-monitor, sin etiquetas y asignado al proyecto EnergyMonitor.

Cuando ya se han configurado todos los parámetros del *droplet* se procede a crear lo haciendo clic en el botón *Create Droplet*. Si todo ha resultado como se deseaba, DigitalOcean procederá a crear el *droplet* con la máquina virtual e instalar el sistema operativo por medio de la imagen anteriormente seleccionada en el apartado de configuración *choose an imagen.*

Figura 86. **Creación en curso Droplets DigitalOcean**

Fuente: elaboración propia, realizado con captura de pantalla.

Tabla V. **Resumen configuración Droplet DigitalOcean**

Resumen creación de Droplet monitor energético
• Imagen: Ubuntu 20.04 (LTS) x64
• Plan Básico: ○ *Regular Intel with SSD* ○ $ 5 por mes ○ 1 GB / 1 CPU ○ 25 GB SSD ○ 1000 GB de transferencia
• Sin espacio adicional
• Región centro de datos: New York espacio 1
• Autenticación por medio de clave SSH
• Opciones adicionales: *monitoring* y IPv6
• 1 Droplet
• Nombre Droplet: ubuntu-energy-monitor
• Sin etiquetas
• Droplet asignado al proyecto EnergyMonitor

Fuente: elaboración propia.

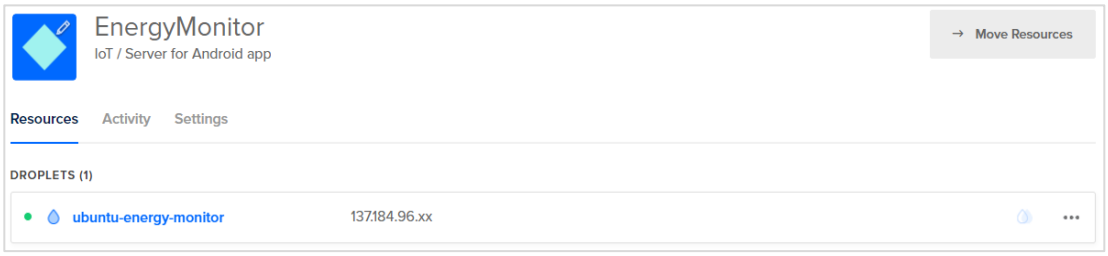
Una vez se haya terminado de crear el *droplet*, DigitalOcean mostrará un número de IPv4 para la máquina virtual, al hacer clic en dicho número, se mostrará un panel de administración para el *droplet*, en este panel se pueden encontrar las siguientes opciones:

- Graficas de rendimiento:
    - CPU
    - Carga
    - Memoria
    - Entrada/salida de disco
    - Uso disco
    - Ancho de banda
- Lanzar terminales como usuario *root* u otro
- Reiniciar contraseña de usuario *root*
- Apagar/encender la máquina virtual
- Habilitar IP flotante
- Redimensionar los recursos
- Aplicar Cortafuegos
- Actualización de kernel
- Consultar historial de eventos
- Destruir y reconstruir el *droplet*
- Agregar etiquetas
- Recuperación y reparación de archivos corruptos del sistema

Un proyecto puede tener más recursos asignados en función de la necesidad, pudiendo así asignar recursos de bases datos dedicadas, espacios

de almacenamiento, varios *droplets*, entre otros. Para esta aplicación solo se cuenta con un *droplet* como recurso.

Figura 87. **Finalización creación Droplets DigitalOcean**



Fuente: elaboración propia, realizado con captura de pantalla.

Figura 88. **Panel de administración Droplets DigitalOcean**

Fuente: elaboración propia, realizado con captura de pantalla.

6.2. Configuración inicial máquina virtual

Teniendo un *droplet* con un sistema operativo funcional se procede a hacer configuraciones iniciales con el fin de garantizar la seguridad y la disponibilidad de los servicios que se pretenden brindar por medio de actualización de parches de seguridad y la instalación de los recursos necesarios para el funcionamiento de la aplicación. (DigitalOcean, 2020)

6.2.1. Inicio de sesión usando usuario *root*

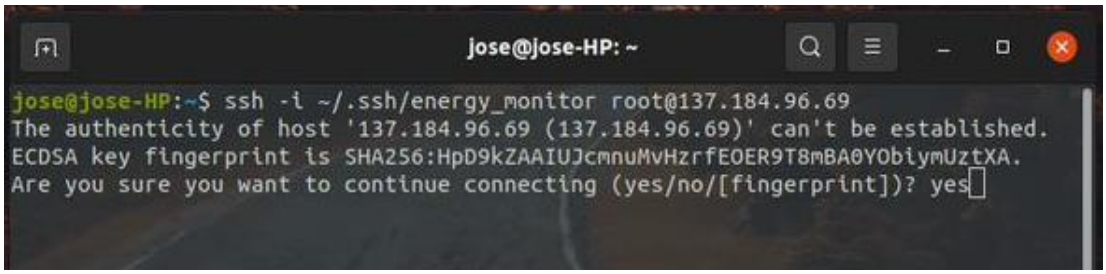
La primera vez que se accede por SHH no se establece la conexión y por medio de la línea de comandos se pregunta al usuario si desea continuar con la conexión, al momento de confirmar la continuación de la conexión se pregunta por la clave, de la clave privada en caso se haya asignado una clave al momento de su creación (DigitalOcean, 2018). Si todos los parámetros de conexión están correctos entonces se procederá a iniciar sesión en el servidor remoto (figura 90).

Tabla VI. **Comando inicio de sesión como *root* SSH usando calve privada (Ubuntu)**

Inicio de sesión SSH terminal Ubuntu
shh -i ~/.shh/nombre_clave root@ip

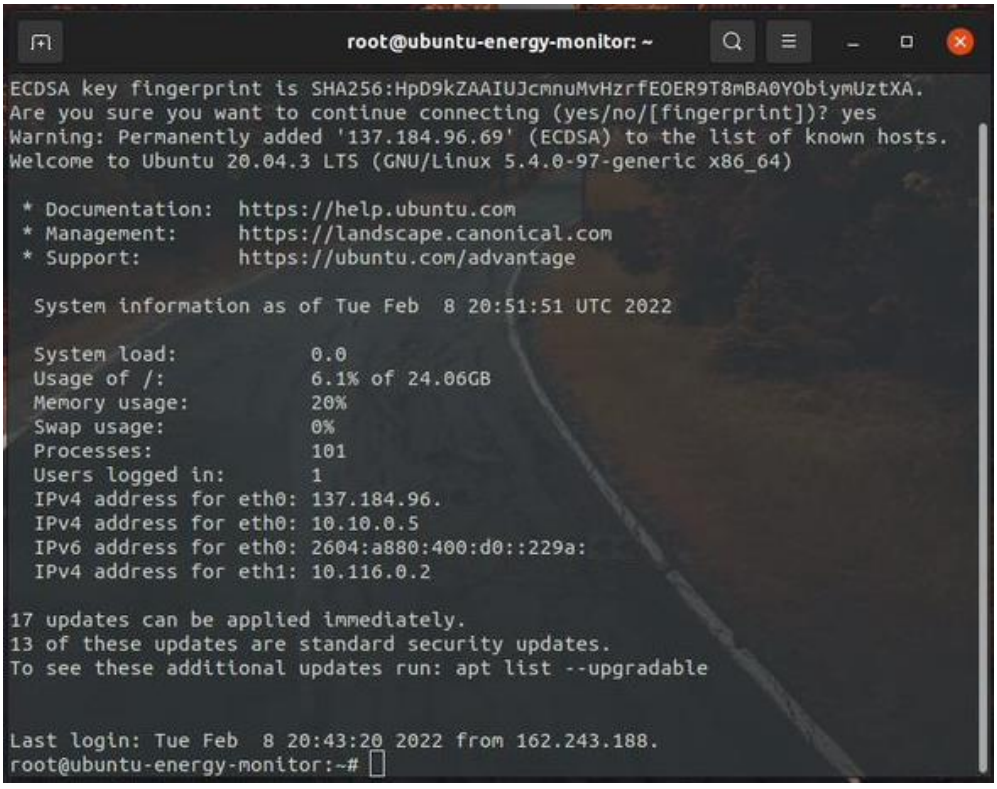
Fuente: elaboración propia.

Figura 89. **Inicio de sesión como *root* SSH usando calve privada en Ubuntu**



Fuente: elaboración propia, realizado con captura de pantalla.

Figura 90. **Sesión SSH iniciada con éxito en servidor remoto Ubuntu**



Fuente: elaboración propia, realizado con captura de pantalla.

6.2.2. Creación de usuario administrativo

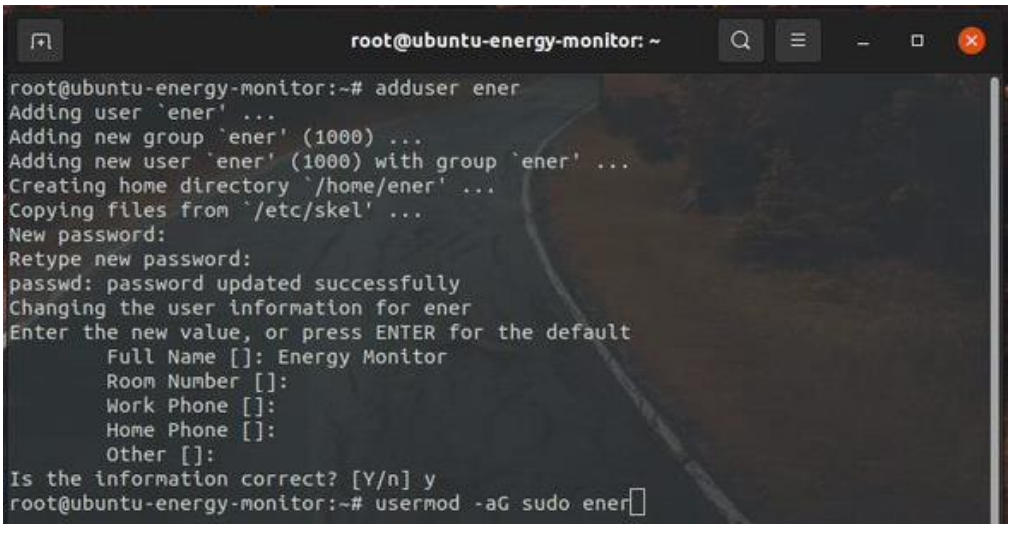
Para hacer más seguros los inicios de sesión es importante no realizarlos con una cuenta *root*. Después de haber iniciado sesión como usuario *root* crear un usuario nuevo usando el comando aduser seguido del nombre de usuario, por medio de la línea de comandos se preguntará por la contraseña y otros campos opcionales. Con el comando usermod -aG sudo, se otorgan los privilegios necesarios al nuevo usuario.

Tabla VII. **Comandos creación usuario con privilegios Ubuntu**

Creación de usuario	adduser nombre_usuario
Agregar usuario a grupo sudo	usermod -aG sudo nombre_usuario

Fuente: elaboración propia.

Figura 91. **Creación usuario con privilegios Ubuntu**



Fuente: elaboración propia, realizado con captura de pantalla.

Al usuario de Ubuntu destinado a ejecutar los servicios y acciones administrativas en el servidor se le asignó el nombre “ener”.

6.2.3. Configuración cortafuegos básico

El cortafuego permite a Ubuntu bloquear o permitir el tráfico entrante/saliente. El comando ufw, sirve para habilitar, configurar, crear y aplicar reglar para el cortafuegos que Ubuntu incorpora por defecto.

Tabla VIII. **Comandos de configuración básica cortafuegos Ubuntu**

Listar perfiles de aplicaciones registradas	ufw app list
Activar cortafuegos	ufw enable
Permitir conexión SSH	ufw allow OpenSSH

Fuente: elaboración propia.

Figura 92. **Configuración básica cortafuegos Ubuntu**

Fuente: elaboración propia, realizado con captura de pantalla.

6.3. Base de datos

La creación de una base de datos se puede ejecutar de diferentes maneras, las más comunes son por medio de la línea de comandos del gestor y usando interfaces gráficas de administración, es importante tener en cuenta primero la elección del gestor de base de datos puesto que por medio de este se ejecutan las tareas de administración como es la creación.

6.3.1. Instalación de MySQL en Ubuntu

El gestor de base de datos elegido para almacenar los datos de consumo de este monitor energético es MySQL bajo licencia Oracle, debido a que en la instalación de Ubuntu no se preinstala este gestor es necesario ejecutar la instalación por medio de una terminal. La instalación se lleva a cabo mediante los siguientes pasos:

- Paso 1: actualizar los repositorios APT del sistema operativo ingresando el comando sudo apt update a la terminal.

- Paso 2: instalar MySQL Server por medio del repositorio oficial usando el comando sudo apt install mysql-server en la terminal.

Tabla IX. **Comandos instalación MySQL Ubuntu**

sudo apt update
sudo apt install mysql-server

Fuente: elaboración propia.

Figura 93. **Instalación MySQL en Ubutun**

Fuente: elaboración propia, realizado con captura de pantalla.

6.3.2. Configuración MySQL

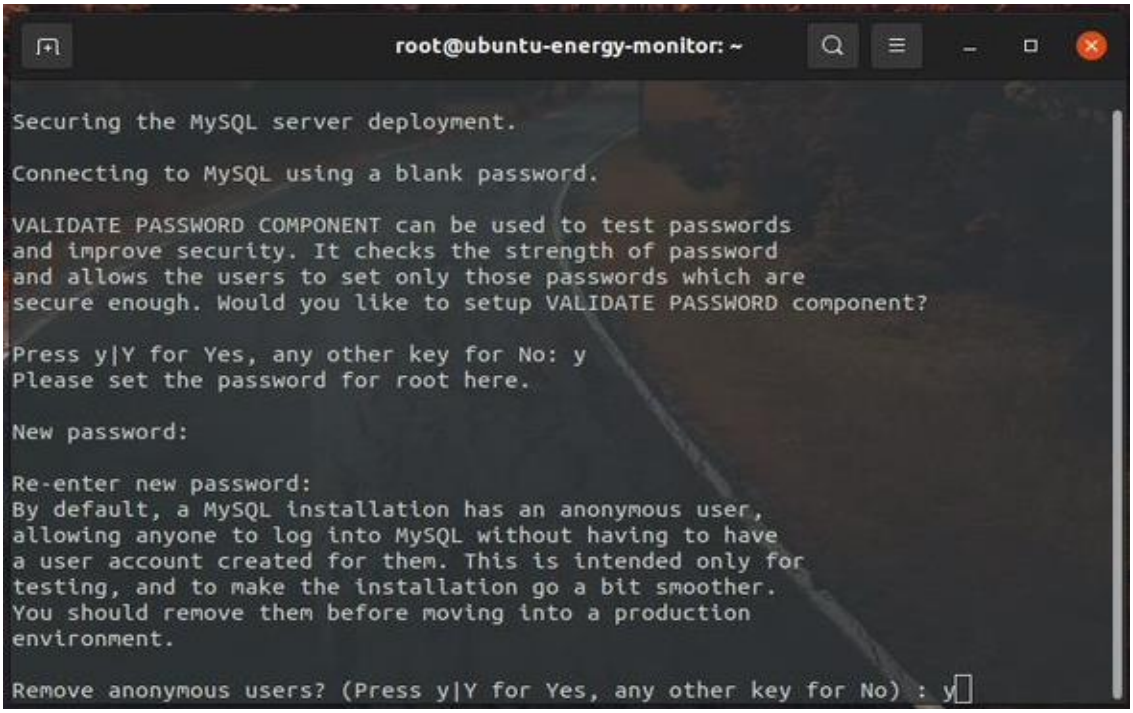
Para aplicar una configuración más segura se utiliza el comando sudo mysql_secure_installation, que tiene como tarea cambiar opciones inseguras que por defecto vienen aplicados en una instalación. Pasos configuración de seguridad:

- Paso 1: ingresar el comando sudo mysql_secure_installation en una terminal de Ubuntu.

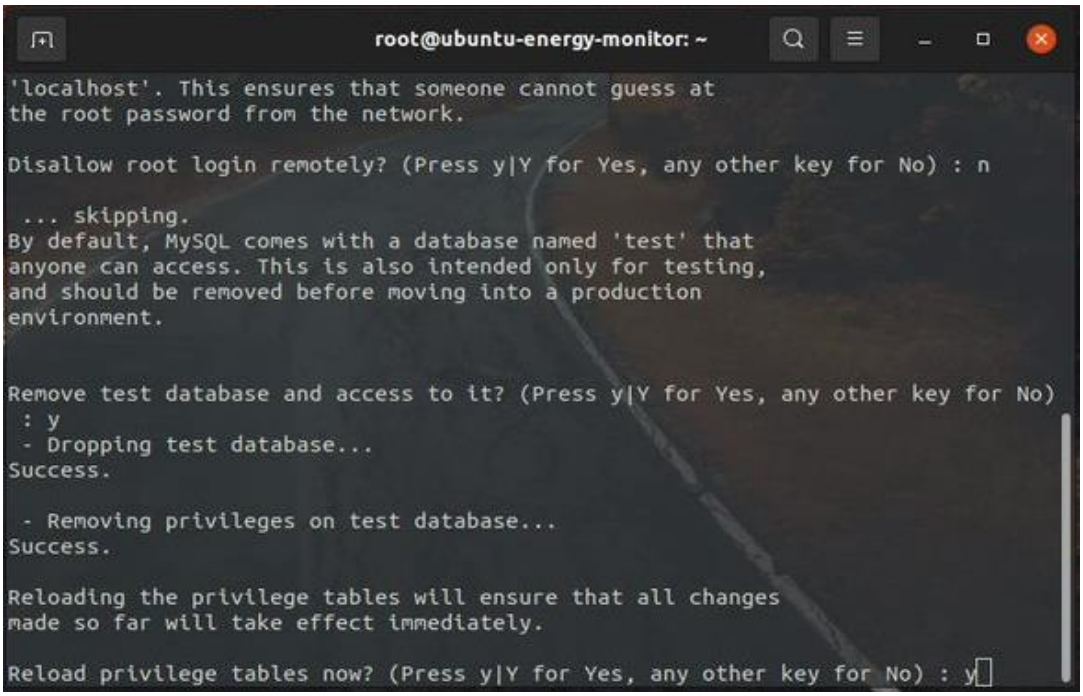
- Paso 2: elegir parámetro para configurar contraseña

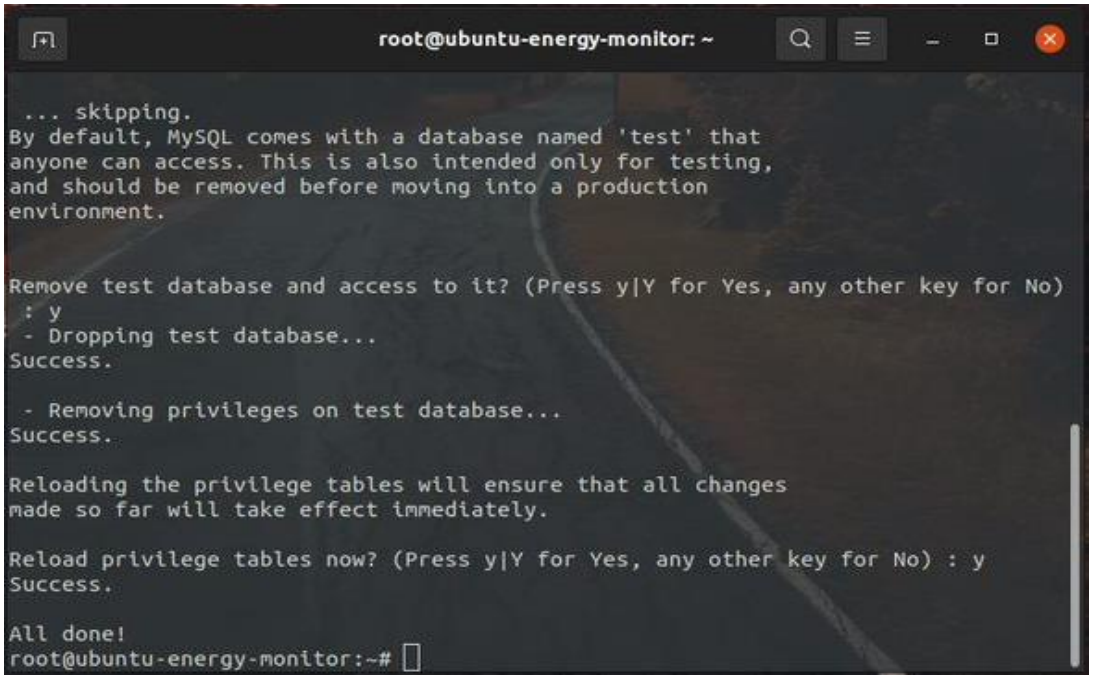
    - Paso 2.1: elegir nivel de validación de contraseña
    - Paso 2.3: ingresar una contraseña de validación
    - Paso 2.4: reingresar la contraseña de validación

Figura 94. **Configuración MySQL en Ubuntu**



Continuación de la figura 94.





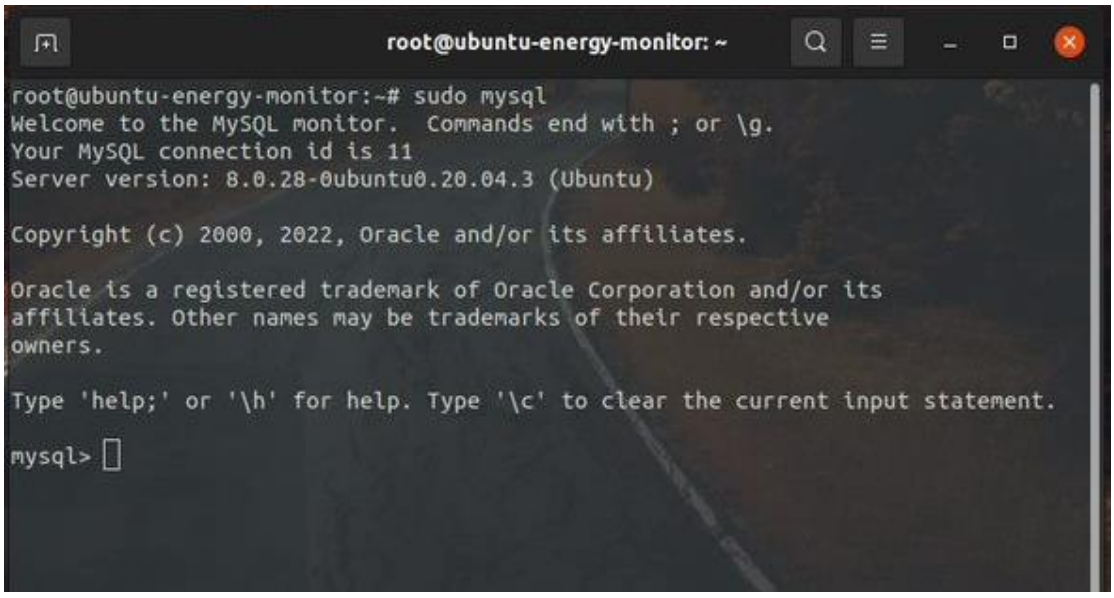
Fuente: elaboración propia, realizado con captura de pantalla.

6.3.3. Creación usuario MySQL

Por cuestiones de seguridad no se recomienda realizar las gestiones en base de datos por medio del usuario *root* que MySQL trae por defecto, lo recomendable es crear un nuevo usuario con los permisos necesarios para ejecutar las tareas que el servicio requiera. Pasos que seguir para crear un usuario en MySQL Ubuntu:

- Paso 1: abrir una terminal e ingresar el comando sudo mysql, si el usuario *root* se encuentra configurado para autenticarse por medio de auth_socket no se necesita una contraseña, en caso contrario es necesario ingresar una contraseña. Si la autenticación tuvo éxito se abrirá la ventana de consola de MySQL.

Figura 95. **Consola MySQL**



Fuente: elaboración propia, realizado con captura de pantalla.

- Paso 2: usar el comando CREATE USER especificando el nombre de usuario y contraseña.

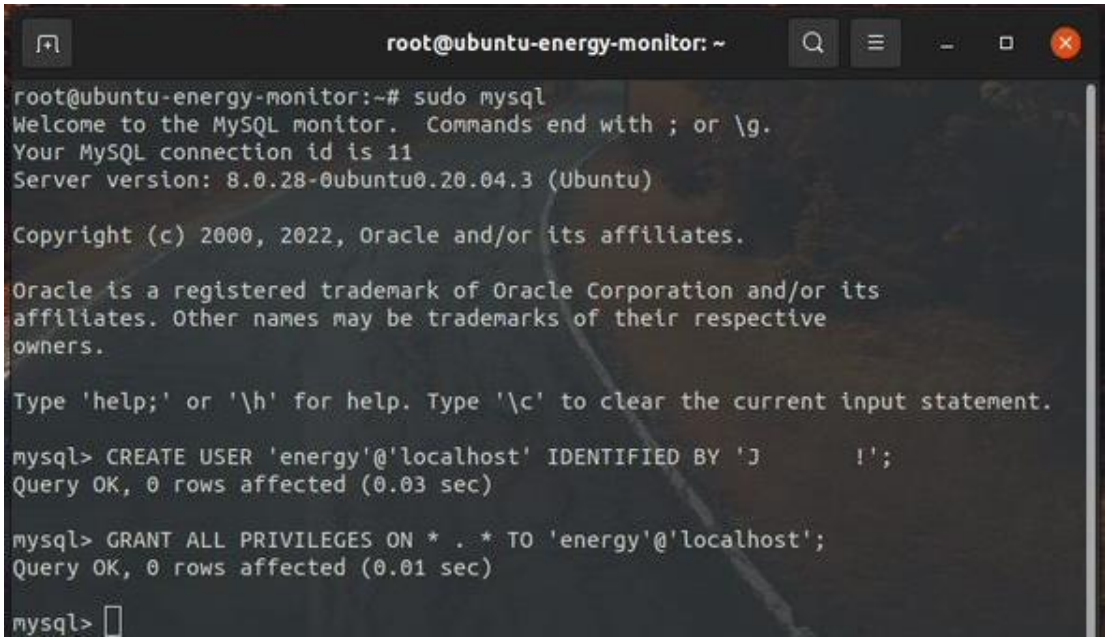
- Paso 3: agregar privilegios al nuevo usuario usando el comando GRANT ALL PRIVILEGES.

Tabla X. **Comandos crear usuario con privilegios MySQL**

CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'contraseña';
GRANT ALL PRIVILEGES ON *.* TO 'usuario'@'localhost' WITH GRANT OPTION;

Fuente: elaboración propia.

Figura 96. **Creación de usuario con privilegios MySQL**



Fuente: elaboración propia, realizado con captura de pantalla.

Tabla XI. **Comando inicio de sesión usando usuario no *root* MySQL (terminal Ubuntu)**

mysql -u usuario -p

Fuente: elaboración propia.

Al usuario de MySQL que servirá para administrar los datos de este proyecto se le asignó el nombre energy.

6.3.4. Creación base de datos

Para crear los comandos usados para crear una base de datos, primero se debe de acceder a la consola de MySQL con un usuario con los privilegios suficientes para ejecutar instrucciones CREATE.

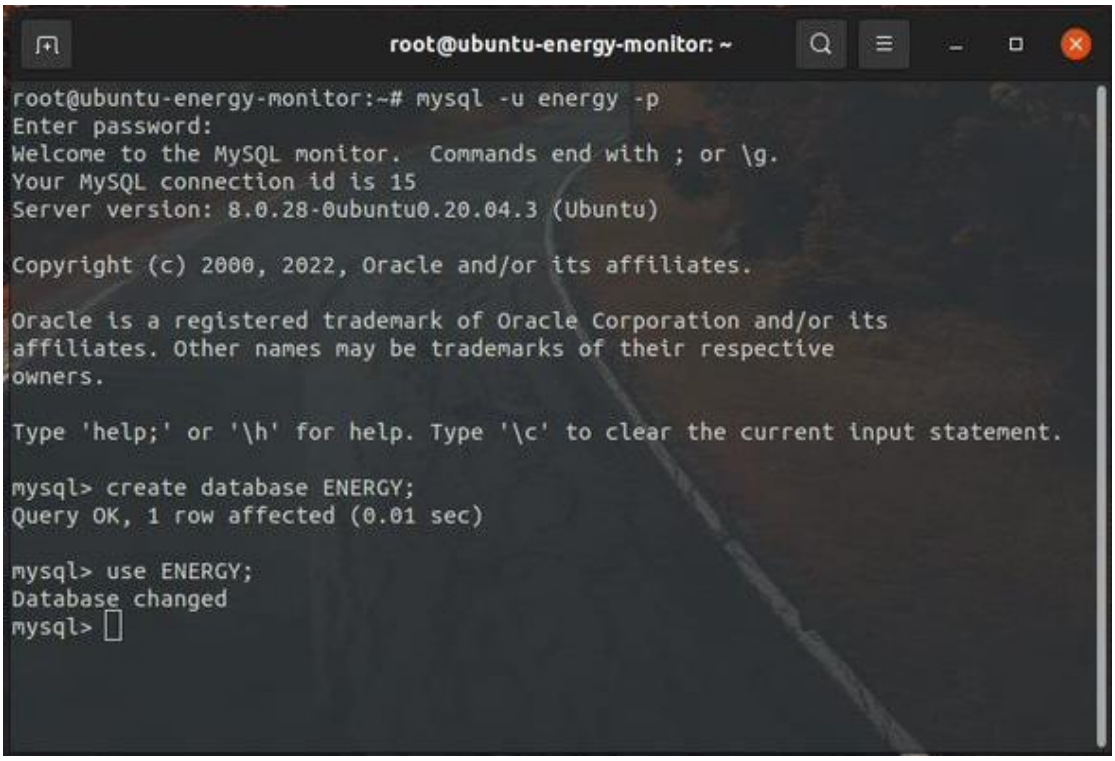
Tabla XII. **Comandos para crear y seleccionar base de datos MySQL**

CREATE DATABASE nombre_base_de_datos;
nombre_base_de_datos;

Fuente: elaboración propia.

Para este proyecto se le asignó el nombre ENERGY a la base de datos encargada de guardar toda la información de consumo relevante.

Figura 97. **Crear y seleccionar base de datos MySQL**



Fuente: elaboración propia, realizado con captura de pantalla.

6.3.4.1. Tablas en base de datos

La base de datos de este proyecto se compone de 2 tipos de tablas, estas son:

- Tabla de dispositivos registrados: en esta se encuentra registrada la información de todos los dispositivos de medición, los registros se crean a medida que se agregan nuevos dispositivos de medición al sistema. Los datos más relevantes que se almacenan en esta tabla son los siguientes:

- Nombre del dispositivo: cada dispositivo de medición tiene un único nombre que lo identifica y está conformado por el prefijo ENER-ESP32- seguido del número de creación (serie), del dispositivo. El número de creación es iniciado en 1 e incrementado a medida que se agregan nuevos dispositivos de medición al sistema.

- Nombre de la medición: este nombre puede repetirse entre dispositivos porque identifica a el dispositivo que está bajo monitoreo.

- Tarifa energética: este valor es la brindado por el proveedor de servicios y usualmente se encuentra en el recibo de cobro de energía eléctrica.

- Moneda: tipo de moneda que se utiliza para realizar el pago del servicio eléctrico.

- Día de pago: este dato sirve para reiniciar los contadores del dispositivo de medición, y es el día en el que se paga la energía consumida durante un periodo de tiempo que usualmente es un mes.

- Zona horaria: guarda la zona horaria del dispositivo de medición

- Tablas de consumo por dispositivo: cada dispositivo de medición posee una tabla en donde almacena los datos de consumo, el servidor está programado para solicitar a todos los dispositivos de medición información de consumo a cada 10 minutos, esto con el fin de guardar los datos de

consumo energético y ser usados para futuras consultas. Los campos más relevantes de esta tabla están conformados por los siguientes datos:

- Identificador de registro: este número es autoincrementado y generado automáticamente por el sistema y cumple el objetivo de identificar cada registro de medición, por lo que no puede repetirse.

- Nombre del dispositivo: nombre que posee el dispositivo de medición.

- Voltaje: almacena el voltaje registrado por la medición.

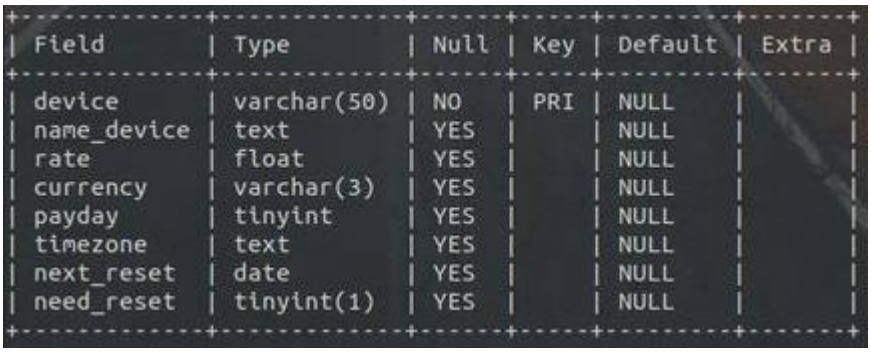
- Corriente: almacena la corriente registrada por la medición.

- Energía: este campo guarda la energía consumida por el dispositivo al cual se le está realizando el monitoreo.

- Cargo monetario: guarda el valor monetario que representa la energía consumida y es proporcional a la tarifa previamente configurada por el usuario.

- Fecha y hora: almacena la fecha y hora en la que se realizó la medición.

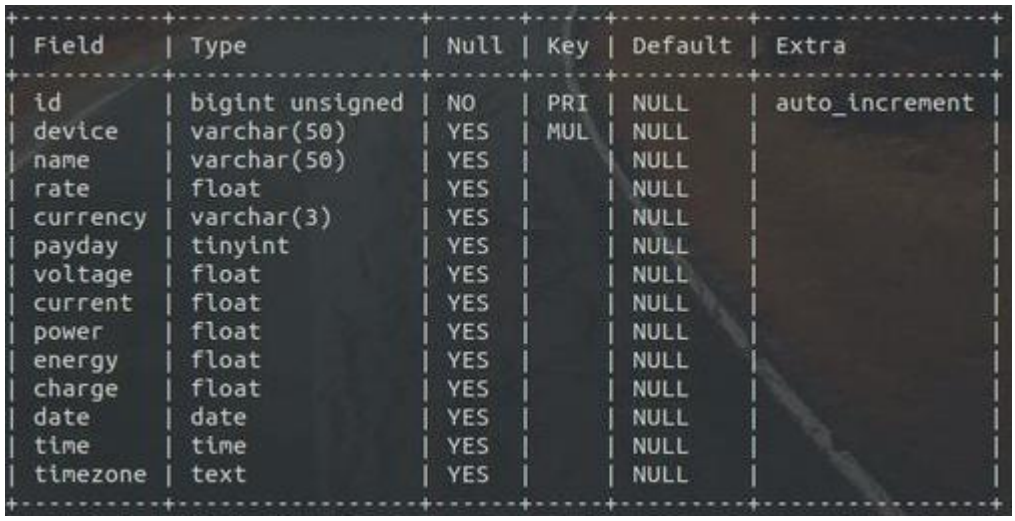
Tabla XIII. **Estructura de la tabla de dispositivos**



Field	Type	Null	Key	Default	Extra
device	varchar(50)	NO	PRI	NULL	
name_device	text	YES		NULL	
rate	float	YES		NULL	
currency	varchar(3)	YES		NULL	
payday	tinyint	YES		NULL	
timezone	text	YES		NULL	
next_reset	date	YES		NULL	
need_reset	tinyint(1)	YES		NULL	

Fuente: elaboración propia, realizado con captura de pantalla.

Tabla XIV. **Estructura de una tabla de consumo**



Field	Type	Null	Key	Default	Extra
id	bigint unsigned	NO	PRI	NULL	auto_increment
device	varchar(50)	YES	MUL	NULL	
name	varchar(50)	YES		NULL	
rate	float	YES		NULL	
currency	varchar(3)	YES		NULL	
payday	tinyint	YES		NULL	
voltage	float	YES		NULL	
current	float	YES		NULL	
power	float	YES		NULL	
energy	float	YES		NULL	
charge	float	YES		NULL	
date	date	YES		NULL	
time	time	YES		NULL	
timezone	text	YES		NULL	

Fuente: elaboración propia, realizado con captura de pantalla.

En este proyecto la tabla de dispositivos es llamada *devices*, y lleva como clave primaria el campo *device*. El nombre de las tablas de consumo está formado por el prefijo ENER_ESP32_ seguido del número de creación del

dispositivo, el número de identificación (campo llamado “id” en la estructura de la tabla), es asignado como clave primaria de las tablas de consumo.

Las tablas de consumo están relacionadas con la tabla de dispositivos por medio de una clave foránea (contenida en el campo *device* de las tablas de consumo), que hace referencia a la clave primaria de la tabla *devices*, esta relación es del tipo uno a muchos.

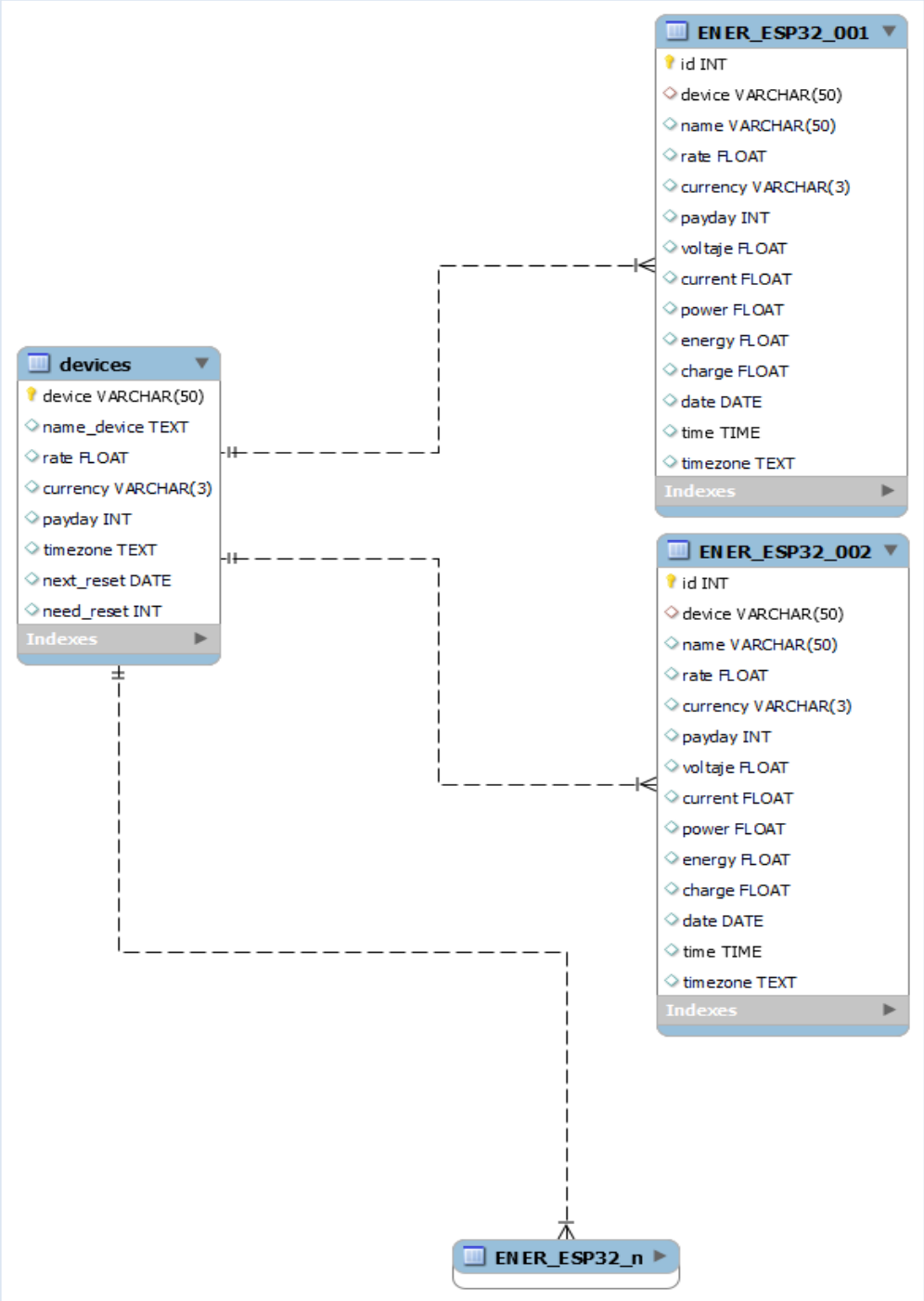
La figura 98 muestra un esquema para entender de una mejor manera como se compone la relación una a muchas en las tablas de la base de datos, es importante aclarar que dicha figura únicamente es usada con fines didácticos, por lo tanto, no es una representación exacta (aunque sí bastante aproximada), del esquema de tablas de la base de datos real desde que la tabla ENER_ESP32_n, no existe y solo es usada para representar el ultimo dispositivo de medición agregado al sistema.

Tabla XV. **Comandos MySQL usados para la creación de la tabla de dispositivos**

```
CREATE TABLE 'devices' (
  'device' varchar(50) NOT NULL,
  'name_device' text,
  'rate' float DEFAULT NULL,
  'currency' varchar(3) DEFAULT NULL,
  'payday' tinyint DEFAULT NULL,
  'timezone' text,
  'next_reset' date DEFAULT NULL,
  'need_reset' tinyint(1) DEFAULT NULL,
  PRIMARY KEY (`device`)
)
```

Fuente: elaboración propia.

Figura 98. **Esquema de tablas**



Fuente: elaboración propia, realizado con MySQL Workbench.

6.4. *Broker* MQTT

El *broker* de mensajería MQTT elegido para este monitor energético es la versión de Eclipse Mosquitto que funciona bajo el protocolo MQTT 3.1/3.11. Se ha elegido por ser de fácil instalación, ligero y de código abierto. Por medio de mensajería MQTT se ejecutan tareas de administración, configuración y almacenamiento solicitas por los distintos usuarios (dispositivos de medición, dispositivos Android y usuarios de administración).

6.4.1. Instalación Eclipse Mosquitto Ubuntu

La instalación de Mosquitto se ejecuta en una terminal por medio de los comandos de instalación y de los repositorios APT oficiales de la aplicación. A continuación, se muestran los comandos a ingresar a la terminal para ejecutar la instalación:

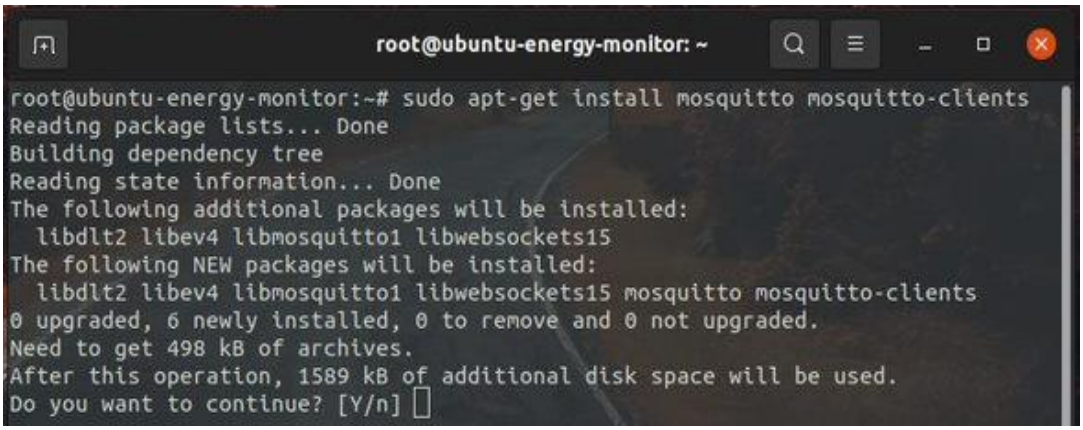
Tabla XVI. **Comandos de instalación Eclipse Mosquitto en Ubuntu**

sudo apt update
sudo apt-get install mosquitto mosquitto-clients

Fuente: elaboración propia.

Si se acaba de realizar la instalación de alguna aplicación por medio de una terminal y por ende ya se ingresó el comando sudo apt update, entonces no es necesario volver a escribirlo porque los repositorios ya fueron actualizados recientemente en la instalación anterior.

Figura 99. **Instalación Eclipse Mosquitto en Ubuntu**



Fuente: elaboración propia, realizado con captura de pantalla.

6.4.2. Configuración Eclipse Mosquitto

Una vez ya se haya finalizado la instalación de Mosquitto es necesario realizar algunas configuraciones de seguridad para evitar las vulnerabilidades del servidor MQTT.

6.4.2.1. Agregar nuevo usuario

Con el fin de autenticar los inicios de sesión en el servidor MQTT se crea un nuevo usuario asignando una contraseña segura. Los usuarios en Mosquitto se almacenan en el archivo llamado passwd, y para sobrescribirlo se cuenta con un comando que permite agregar nuevos usuarios.

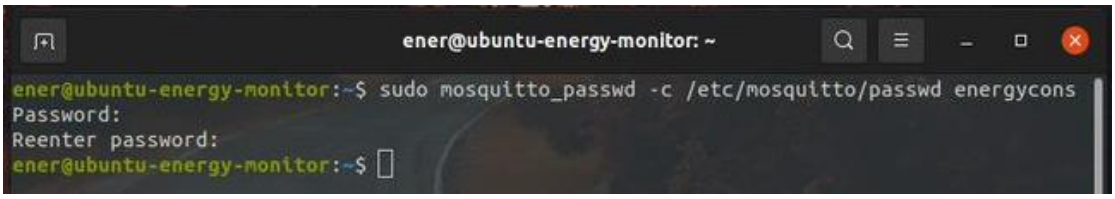
El usuario elegido para autenticar las conexiones al servidor (*broker*) MQTT de este monitor energéticos es energycons.

Tabla XVII. **Comando Mosquitto para agregar nuevo usuario en Ubuntu**

sudo mosquitto_passwd -c /etc/mosquitto/passwd nuevo_usuario

Fuente: elaboración propia.

Figura 100. **Nuevo usuario Mosquitto en Ubuntu**



Fuente: elaboración propia, realizado con captura de pantalla.

6.4.2.2. Especificar archivo de contraseñas

Cuando en la configuración de Mosquitto aún no se ha especificado el archivo de contraseñas la autenticación aún no se encuentra habilitada por lo que cualquier cliente con la dirección IP del servidor MQTT puede conectarse al servidor resultando en una mala práctica y dejando vulnerable la información contenida en los mensajes MQTT, esto se soluciona modificando el archivo de configuración por defecto o creándolo en caso no exista.

Tabla XVIII. **Comando abrir/crear archivo de configuración por defecto Mosquitto en Ubuntu**

sudo nano /etc/mosquitto/conf.d/default.conf

Fuente: elaboración propia.

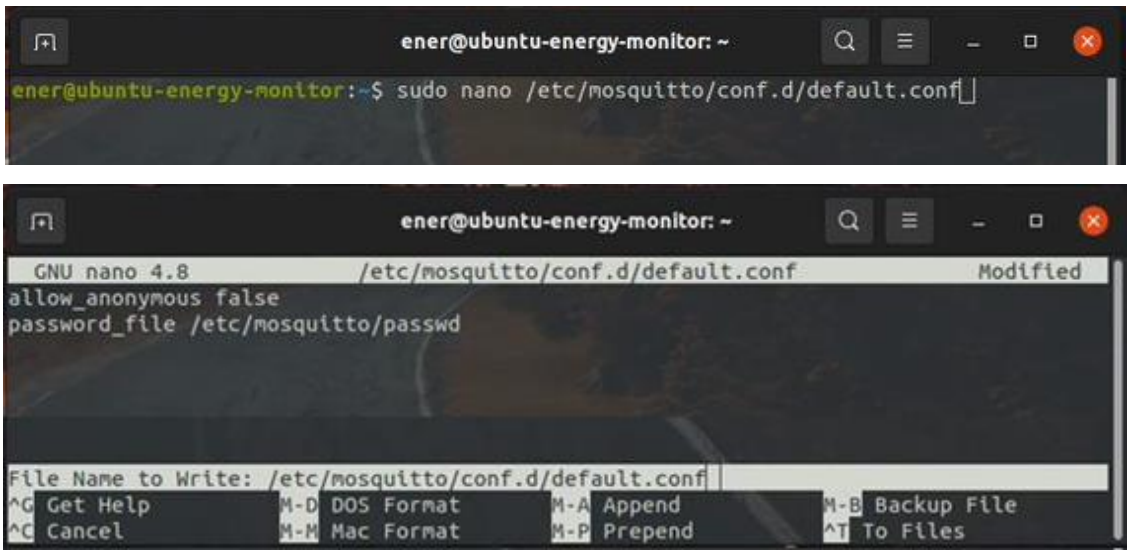
Después de ingresar el comando para abrir la configuración por defecto de Mosquitto en Ubuntu se abrirá una ventana del editor de texto nano, en donde se escribe el archivo de configuración inicial, es en este archivo en donde se rechazan las conexiones a usuarios desconocidos y se asigna un archivo de contraseñas para realizar las autenticaciones de conexión. Para ejecutar las acciones anteriormente mencionadas el archivo de configuración inicial debe de guardar el siguiente contenido:

Tabla XIX. **Contenido archivo de configuración por defecto (default.conf)**

```
allow_anonymous false
password_file /etc/mosquitto/passwd
```

Fuente: elaboración propia.

Figura 101. **Modificación de archivo de configuración por defecto**



Fuente: elaboración propia, realizado con captura de pantalla.

Para aplicar los cambios realizados en el archivo de configuración por defecto es necesario reiniciar Mosquitto. Para reiniciar el *broker* Mosquitto por medio de una terminal de Ubuntu se debe de ingresar el siguiente comando:

Tabla XX. **Comando reiniciar Mosquitto en Ubuntu**

```
sudo systemctl restart mosquitto
```

Fuente: elaboración propia.

6.4.2.3. Habilitar tráfico MQTT en cortafuegos Ubuntu

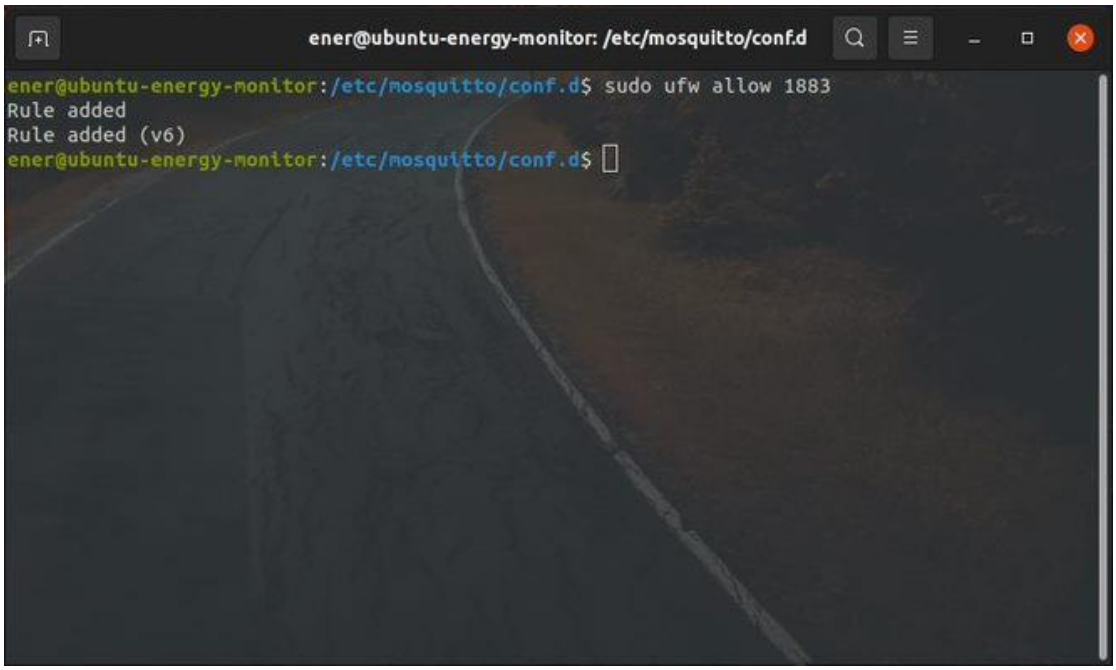
Si el cortafuegos de Ubuntu se encuentra habilitado y no cuenta con una regla que permita el tráfico en el puerto 1883 todas las solicitudes de conexión al servidor MQTT serán rechazadas lo que provocara una denegación de servicios, para solucionar esta problemática se deben de actualizar las reglas del cortafuegos ingresando el siguiente comando a través de una terminal en Ubuntu:

Tabla XXI. **Comando para permitir tráfico MQTT agregando regla al cortafuegos Ubuntu**

```
sudo ufw allow 1883
```

Fuente: elaboración propia.

Figura 102. **Permitir tráfico MQTT en cortafuegos Ubuntu**

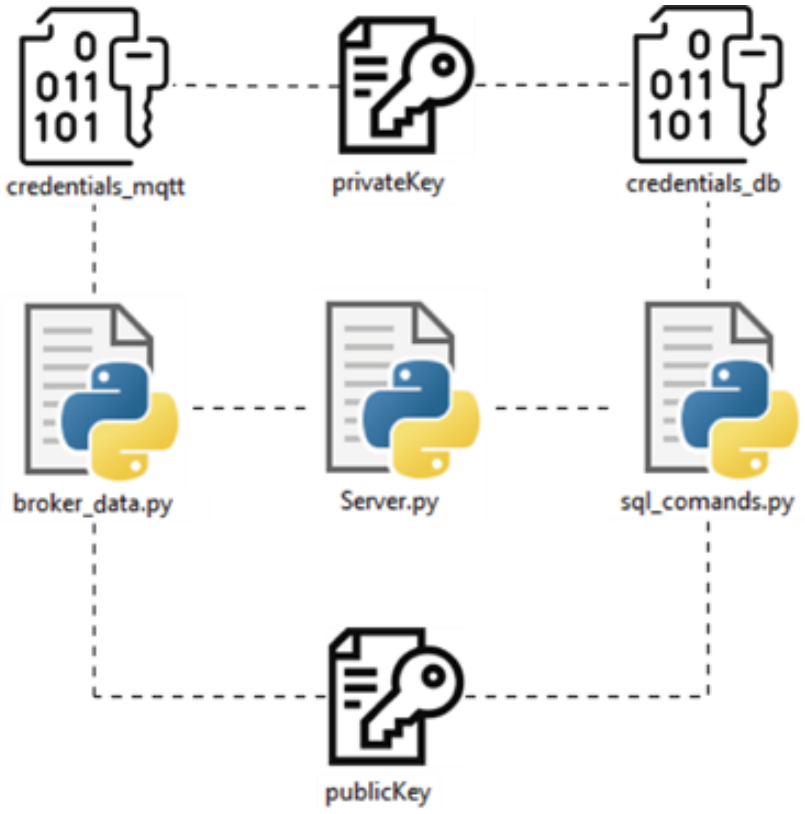


Fuente: elaboración propia, realizado con captura de pantalla.

6.5. Archivos del servidor

Los servicios de este monitor energético funcionan bajo el lenguaje de programación interpretado Python, la comunicación con el servidor se maneja utilizando mensajería MQTT por lo que el servidor solo recibe peticiones por medio del *topic* raíz energy. Para el manejo de las credenciales de base de datos MySQL y MQTT se usa un cifrado asimétrico de esta manera se evita tener las credenciales en texto plano en el código fuente o en un archivo.

Figura 103. **Archivos del servidor**



Fuente: elaboración propia, realizado con Adobe Illustrator.

- Criptografía asimétrica: es un proceso de cifrado que utiliza un par de llaves llamadas pública y privada. La llave pública se usa para cifrar el contenido de la información y no se podrá descifrar sin la clave privada, los usuarios comparten su clave pública con el propósito de recibir mensajes cifrados y en caso alguien intercepte el mensaje no pueda ser comprendido dado que la clave privada por ningún motivo debe ser compartida.

6.5.1. Archivos de credenciales

En el servidor se almacenan los archivos llamados credentials_db y credentials_mqtt, que son del tipo binario, previamente se cifró el contenido de credenciales en texto plano usando una clave pública RSA dado que su función es evitar que los datos sensibles (credenciales de acceso), sean visibles en caso el código fuente sea vulnerado.

En caso se deba cambiar las credenciales de acceso se deberán escribir en texto plano y luego usando el paquete de Python llamado Crypto se realiza el cifrado basado en el relleno de cifrado asimétrico óptimo OAEP en conjunto el cifrado por medio de una clave pública RSA.

6.5.2. Archivos de claves de criptografía

Estos archivos son usados para cifrar y descifrar las credenciales, en el servidor se encuentran 2 claves las cuales se identifican con los nombres de archivo privateKey y publicKey, estos archivos contienen en texto plano el contenido de dichas claves. El par de claves RSA (privateKey y publicKey), contienen una longitud de clave de 1024 bits y se generan usando el paquete de Python pycryptodome.

Figura 104. **Código Python para generar par de claves RSA**

```python
import Crypto
from Crypto.PublicKey import RSA
import binascii

random_generator = Crypto.Random.new().read

private_key = RSA.generate(1024, random_generator)
public_key = private_key.publickey()

private_key = private_key.exportKey(format='DER')
public_key = public_key.exportKey(format='DER')

private_key = binascii.hexlify(private_key).decode()
public_key = binascii.hexlify(public_key).decode()

prikey = open("privateKey.txt", "w")
prikey.write(private_key);
prikey.close()

pubkey = open("publicKey.txt", "w")
pubkey.write(public_key);
pubkey.close()
```

Fuente: elaboración propia, realizado con captura de pantalla.

6.5.3. Archivos Python

Estos archivos son los más importantes dado que algunos están en ejecución todo el tiempo a la escucha de peticiones por parte de los clientes, la lógica de este monitor está conformada por los siguientes archivos:

- server.py: es el archivo principal del servidor, su ejecución se da en tiempo real a la espera de peticiones a través de mensajes MQTT, las peticiones

se identifican por medio del *topic* MQTT del mensaje y por el encabezado de este, las tareas que lleva a cabo este archivo son las siguientes:

- Eliminar registros con un año de antigüedad: los registros de consumo de cada dispositivo de medición son eliminados de la base de datos cuando estos cumplen 1 año de antigüedad.

- Reiniciar contadores de consumo energético: cuando se llega el día de pago registrado por cada dispositivo de medición el servidor envía una instrucción para reiniciar el contador de consumo energético, por medio de la respuesta del dispositivo de medición el servidor es capaz de registra el reinicio del contador.

- Almacenar registros de consumo energético: por medio de una rutina de programación el servidor solicita a cada 10 minutos información de consumo energético a todos los dispositivos de medición, dicha información se solicita para ser guardada en la base de datos.

- Responder a solicitudes de historial de consumo: cuando un usuario con un dispositivo Android desea ver su historial de consumo este usa la función establecida por la aplicación Android la cual enviará una petición al servidor solicitando información del historial de consumo en determinada fecha y si las entradas de fecha son válidas el servidor responderá con la información que se encuentra almacenada en la base de datos y que cumpla con los criterios de búsqueda establecidos por el usuario.

- Actualizar la información almacenada de dispositivos de medición: la información de todos los dispositivos de medición se encuentra almacenada en la base de datos, cuando un usuario configura un dispositivo de medición este envía la información al servidor para posteriormente actualizar el registro con los datos de dicho dispositivo de medición.

- Agregar dispositivos de medición: esta opción es válida únicamente a nivel administrativo y ningún usuario puede hacer uso de ella, se usa para indicarle al servidor que hay un nuevo dispositivo de medición y como resultado el servidor creara una tabla para registrar el consumo y agregara el nuevo dispositivo a la tabla de dispositivos.

- Eliminar dispositivos de medición: esta opción al igual que la de agregar un dispositivo de medición es de uso administrativo y sirve para eliminar un dispositivo de medición del sistema, la acción de esta opción es eliminar la tabla de consumo del dispositivo y eliminar el registro de ese dispositivo en la tabla de dispositivos.

- sql_comands.py: este archivo contiene una clase destinada a administrar la base de datos, en los atributos de esta clase se descifra el archivo de credenciales usando la clave privada del servidor y se obtienen las credenciales de conexión a base de datos, y se declara el *socket* para conectar con la base de datos. Los métodos implementados por la clase son usados para eliminar, crear, actualizar y consultar registros en tablas, así como también eliminar y crear tablas.

- broker_data.py: para conectar con el *broker* MQTT el archivo llamado server.py, solicita las credenciales por medio del archivo broker_data.py, en consecuencia, se lee el archivo de credenciales credentials_mqtt, luego lo descifra usando la clave privada del servidor y devuelve los datos por medio de un array que almacena las credenciales que posteriormente serán usadas para establecer la conexión al broker MQTT.

Figura 105. **Código archivo server.py**

```
from posixpath import split
import paho.mqtt.client as mqtt
import logging
import time
import os
import threading
import sys
from datetime import datetime
from dateutil.relativedelta import relativedelta
from datetime import date
import pytz
from broker_data import *
from sql_comands import *

LOGFILE = 'server.log'

sql = db_manage()    #This object hand new users and logins

#Configuracion inicial de logging
logging.basicConfig(
    level = logging.INFO,
    format = '[%(levelname)s] (%(threadName)-10s) %(message)s'
    )


#verify reset date for all devices, delete old
#registers (registers of one year ago) and send mqtt msgs
def mng_data():
    sql_check = db_manage()    #This object make request to database
    old_date = datetime.now().date()
    while True:
        today = datetime.now()

        #send a mqtt message every 10 minutes to all devices in order
        #to save their current consumption
        if(today.minute%10==8 and today.second==0):
            client.publish("energy/devices", "save-consum", 0, False)

        #check if devices need reset every hour
        if(today.minute==0 and today.second==1):
            dates = sql_check.get_next_reset()   #get all next reset dates add devices
            for i in dates:
                if (i[1]!=None):           #check if date reset is empty
                    #get current date from time zone
                    try :
                        local_today = datetime.now(pytz.timezone(i[2]))
                        if (local_today.date()>i[1]):
                            #compares current date wiht next reset date
                            next_date=(str(i[1].year)+"-"+
                                       str(i[1].month+1)+"-"+
```

Continuación de la figura 105.

```
                                                    str(i[1].day))
                                #set a new next reset date
                                resp = sql.set_next_reset(i[0],next_date)
                                client.publish("energy/devices/"+i[0], "reset", 2, False)
                                log = (str(datetime.now())+" Update next date reset of " +
                                    i[0] + " return: " + resp)
                                logging.info(log)
                                save_log(log)
                            except:
                                log = str(datetime.now())+" Invalid zone time for " + i[0]
                                logging.info(log)
                                save_log(log)

            if (today.date()>old_date and today.second==2):
                old_date = today.date()
                delete_date = today - relativedelta(years=1)
                sql_check.delete_old_registers(delete_date)

            time.sleep(1)

#Callback que se ejecuta cuando nos conectamos al broker
def on_connect(client, userdata, flags, rc):
    log = str(datetime.now())+" Running..."
    logging.info(log)
    save_log(log)

#Callback que se ejecuta cuando llega un mensaje al topic suscrito
def on_message(client, userdata, msg):
    inf = msg.payload.decode().replace("'","").split('\n')

    if (str(msg.topic).split("/")[1]=="devices"):
        if (inf[0]=="req"):
            device = inf[1]
            #ask if device need energy reset
            if (sql.need_reset(device)):
                client.publish("energy/devices/"+device, "reset", 2, False)
                log = (str(datetime.now())+" insert data of " + inf[1] +
                " return: "+"device need reset and data wasn't added")
                logging.info(log)
                save_log(log)
            else :
                name = inf[2]
                rate = inf[3]
                currency = inf[4]
                payday = inf[5]
                timezone = inf[6]
                volt = inf[7]
                current = inf[8]
                power = inf[9]
                energy = inf[10]
```

Continuación de la figura 105.

```
                charge = inf[11]
                reg_date = datetime.now(pytz.timezone(timezone)).date()
                reg_time = datetime.now(pytz.timezone(timezone)).time()
                #insert a new measurement
                resp = sql.insert_data(device,name,rate,currency,payday,volt,current,
                                       power,energy,charge,reg_date,reg_time,timezone)
                log = str(datetime.now())+" insert data of " + inf[1] + " return: " + resp
                logging.info(log)
                save_log(log)
            if(inf[0]=="ok-reset"):
                #measure device indicates its energy counter was reestarted
                sql.reg_reset(str(msg.topic).split("/")[2],0)
                #regist new reset energy counter
            if(inf[0]=="req-data" or inf[0]=="req-data-day"):
                #measure device ask for energy consumption saved
                device = str(msg.topic).split("/")[2]
                name = inf[1]
                date_ini = date(int(inf[2].split("-")[0]),
                                int(inf[2].split("-")[1]),
                                int(inf[2].split("-")[2]))
                date_final = None
                if inf[0]=="req-data":
                    date_final = date(int(inf[3].split("-")[0]),
                                      int(inf[3].split("-")[1]),
                                      int(inf[3].split("-")[2]))

                resp = sql.histroy(device,name,date_ini,date_final)
                if date_final==None:
                        date_conf=str(date_ini)
                else:
                    date_conf=str(date_ini)+"-"+str(date_final)
                if resp != ():
                    cons = (str(resp[0])+" kWh"+'\n'+str(resp[1][0][0])+resp[1][0][1]
                            +str(resp[1][1][0])+resp[1][1][1])
                    client.publish("energy/devices/"+device, "res-data"+'\n'+date_conf
                    +'\n'+cons, 2, False)
                else:
                    client.publish("energy/devices/"+device, "res-data"+'\n'+date_conf
                    +"\n"+"0"+'\n'+"0", 2, False)
                log = str(datetime.now())+" "+device+" ask for energy consumption saved"
                logging.info(log)
                save_log(log)

        if (str(msg.topic)=="energy/update"):
            if (inf[0]=="req"):
                resp = sql.update(inf[1],inf[2],inf[3],inf[4],inf[5],inf[6])
                log = str(datetime.now())+" Update data of " + inf[1] + " return: " + resp
                if resp=="ok":
                    client.publish("energy/devices/"+inf[1], "ok-update", 2, False)
                logging.info(log)
```

Continuación de la figura 105.

```
151             save_log(log)
152
153         if (str(msg.topic)=="energy/control/add_device"):
154             resp = sql.add_device(inf[0])
155             if (resp=="ok"):
156                 client.subscribe(("energy/"+inf[0], 0))
157             log = str(datetime.now())+ "add device " + inf[0] + " return: " + resp
158             logging.info(log)
159             save_log(log)
160
161         if (str(msg.topic)=="energy/control/delete_device"):
162             resp = sql.delete_device(inf[0])
163             if (resp=="ok"):
164                 client.unsubscribe("energy/"+inf[0])
165             log = str(datetime.now())+ "delete device " + inf[0] + " return: " + resp
166             logging.info(log)
167             save_log(log)
168
169     def save_log(data):
170         logCommand = 'echo "' + data + '" >> ' + LOGFILE
171         os.system(logCommand)
172
173     #mqtt configuration
174     client = mqtt.Client(clean_session=True)
175     client.on_connect = on_connect
176     client.on_message = on_message
177     client.username_pw_set(cred_mqtt[2], cred_mqtt[3])
178     client.connect(host=cred_mqtt[0], port = int(cred_mqtt[1]))
179     cred_mqtt=""
180     qos = 0   #qos MQTT
181
182     #Subscripcion to MQTT topics
183     client.subscribe([("energy/update", qos),("energy/control/add_device", qos),
184         ("energy/control/delete_device", qos),("energy/devices/+",qos)])
185
186     #start thread to receive mqtt messages
187     client.loop_start()
188
189     #thread to update reset dates
190     thread_next_reset = threading.Thread(name = 'thread to update reset dates',
191                                          target = mng_data,
192                                          args = (),
193                                          daemon = True
194                                          )
195     thread_next_reset.start()
196
197     # avoid script ends.
198     try:
199         while True:
200             pass
```

Continuación de la figura 105.

```
201
202 except KeyboardInterrupt:
203     log = str(datetime.now())+" MQTT broker disconnecting"
204     logging.warning(log)
205     save_log(log)
206     if thread_next_reset.isAlive():
207         thread_next_reset._stop()
208
209 finally:
210     #close mqtt services
211     client.loop_stop()
212     client.disconnect()
213     log = str(datetime.now())+" Closing"
214     logging.info(log)
215     save_log(log)
216     sys.exit()
```

Fuente: elaboración propia, realizado con captura de pantalla.

Figura 106. **Código archivo sql_comands.py**

```
from locale import currency
import pymysql
import Crypto
import binascii
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
from datetime import datetime
from datetime import date
import pytz

class db_manage(object):

    def __init__(self):
        self.db = ""
        self.cursor = ""
        #------------------------------ Decrypt credentials --------------
        prikey = open("privateKey.txt", "r")
        private_key = prikey.read()
        prikey.close()

        private_key = RSA.importKey(binascii.unhexlify(private_key))

        encrypted_file = open("credentials_db.txt", "rb")
        encrypted_credentials = encrypted_file.read()
        encrypted_file.close()

        cipher = PKCS1_OAEP.new(private_key)
        self.credentials = (cipher.decrypt(encrypted_credentials)).decode()
        private_key=""
        self.credentials = self.credentials.split('\n')
        #-----------------------------------------------------------------

    def delete_old_registers(self,year_ago):
        self.start_conn()
        #get all tables
        sql = ("SHOW TABLES")
        self.cursor.execute(sql)
        info = self.cursor.fetchall()
        for tables in info:
            if tables[0]!="devices":
                sql = ("DELETE FROM " + tables[0].replace("-","_") +
                       " WHERE date <= '" + str(year_ago) + "'")
                self.cursor.execute(sql)
        self.close_conn()

    def histroy(self,device,name,date_ini,date_final=None):
        hisGTQ = self.request_data("GTQ",device,name,date_ini,date_final)
        hisUSD = self.request_data("USD",device,name,date_ini,date_final)
        if hisGTQ!=() and hisUSD!=():
            return (hisGTQ[0]+hisUSD[0],((hisGTQ[1]," GTQ + "),(hisUSD[1]," USD")))
```

Continuación de la figura 106.

```
51          if hisGTQ!=() and hisUSD==():
52              return (hisGTQ[0],((hisGTQ[1]," GTQ"),("","")))
53          if hisGTQ==() and hisUSD!=():
54              return (hisUSD[0],(("",""),(hisUSD[1]," USD")))
55          return ()
56
57      #get consumption history
58      def request_data(self, currency,device,name,date_ini,date_final=None):
59          self.start_conn()
60          # get the first consumption of init date
61          # in case a substraction is necesary
62          last_date = date_ini
63          sql = ("SELECT payday, DATE, energy, charge, id FROM "+device.replace("-","_")+
64              " WHERE (date='"+ str(last_date) +"') AND (currency="+'"'+currency+'")' +
65              'AND (name="'+name+'")')
66          self.cursor.execute(sql)
67          last_info = self.cursor.fetchall()
68
69          if date_final==None: #Consult only one day of energy data
70              sql = ("SELECT payday, DATE, energy, charge, id FROM "+device.replace("-","_")+
71                      " WHERE (date='"+ str(date_ini) +"') AND (currency="+'"'+currency+'")' +
72              'AND (name="'+name+'")')
73              self.cursor.execute(sql)
74              info = self.cursor.fetchall()
75              if info != ():
76                  if ((info[-1][0] == info[-1][1].day) or (info[-1][0] != info[0][0])):
77                      energy = info[-1][2]
78                      charge = info[-1][3]
79                  else:
80                      energy = info[-1][2]-last_info[0][2]
81                      charge = info[-1][3]-last_info[0][3]
82                  return (round(energy,3),round(charge,3))
83              return ()
84          else:                #Consult energy data between 2 dates
85              sql = ("SELECT payday, DATE, energy, charge, id FROM "+device.replace("-","_")+
86                      " WHERE (date BETWEEN '"+str(date_ini)+"' AND '"+str(date_final) +
87                      "') AND (currency="+'"'+currency+'")' + 'AND (name="'+name+'")')
88              self.cursor.execute(sql)
89              info = self.cursor.fetchall()
90
91              if info != ():
92                  prev_energy = 0
93                  prev_charge = 0
94                  if last_info != ():
95                      #if first consumption of init date has registers get first
96                      #consumption register in case a substraction is necesary
97                      prev_energy = last_info[0][2]
98                      prev_charge = last_info[0][3]
99                  add_reg=""       #indicate when a data need to be added to counters
100                 energy=0         #count to energy
```

Continuación de la figura 106.

```
                charge=0         #count to charge
                res=True         #indicate when a substraction is necesary

                for i in range(len(info)-2):
                    if i> 0:
                        if info[i-1][0]!=info[i][0]: #verify if payday changed
                            energy+=info[i-1][2]
                            charge+=info[i-1][3]
                            if res and date_ini.day!=(info[0][0]+1):
                                #a substraction it is necesary because date_ini is
                                # not equal to payday and a substraction has not been
                                # applied yet
                                energy -= prev_energy
                                charge -= prev_charge
                                res=False #indicates a substraction is not longer necesary
                        if info[i][0] == info[i][1].day:
                            #indicates payday it is equal to register's day
                            #and next time when they changed it will be necesary
                            #to add values of the las equal register
                            add_reg="equal"
                        else:
                            if add_reg=="equal":
                                #add the values of the last register of payday
                                add_reg=""
                                energy += info[i-1][2]
                                charge += info[i-1][3]
                                if res and date_ini.day!=(info[0][0]+1):
                                    #a substraction it is necesary because date_ini is
                                    # not equal to payday and a substraction has not been
                                    # applied yet
                                    energy -= prev_energy
                                    charge -= prev_charge
                                    res=False #indicates a substraction is not longer necesary
                #add values of last register to counters
                energy += info[len(info)-1][2]
                charge += info[len(info)-1][3]
                if res and date_ini.day!=(info[0][0]+1):
                    # a substraction it is necesary because date_ini is
                    # not equal to payday and a substraction has not been
                    # applied yet
                    energy -= prev_energy
                    charge -= prev_charge
                    self.close_conn()
                return (round(energy,3),round(charge,3))
            self.close_conn()
            return ()

    def insert_data(self,device,name,rate,currency,payday,volt,current,power,energy,
                    charge,date,time,timezone):
        self.start_conn()
```

Continuación de la figura 106.

```
151         sql = ("INSERT INTO " + device.replace("-","_") + " VALUES(NULL," +
152                                                 '"' + device + '"' + "," +
153                                                 '"' + name + '"' + "," +
154                                                 '"' + str(rate) + '"' + "," +
155                                                 '"' + currency + '"' + "," +
156                                                 '"' + str(payday) + '"' + "," +
157                                                 '"' + str(volt) + '"' + "," +
158                                                 '"' + str(current) + '"' + "," +
159                                                 '"' + str(power) + '"' + "," +
160                                                 '"' + str(energy) + '"' + "," +
161                                                 '"' + str(charge) + '"' + "," +
162                                                 '"' + str(date) + '"' + "," +
163                                                 '"' + str(time) + '"' + "," +
164                                                 '"' + timezone + '"' + ")")
165         try:
166             if(self.cursor.execute(sql)==1):
167                 self.close_conn()
168                 return "ok"
169         except:
170             self.close_conn()
171             return "error adding data"
172 
173     #ask if some measurement device need reset
174     def need_reset(self, device):
175         self.start_conn()
176         sql = "SELECT need_reset FROM devices WHERE device= "+'"'+device+'"'
177         try:
178             self.cursor.execute(sql)
179             resp = self.cursor.fetchall()
180             self.close_conn()
181             return resp[0][0]
182         except:
183             self.close_conn()
184             return 0
185 
186     def reg_reset(self,device,value):
187         self.start_conn()
188         sql = ("UPDATE devices SET need_reset="+str(value)+
189                                     " WHERE device=" + '"' + device + '"')
190         try:
191             self.cursor.execute(sql)
192             self.close_conn()
193             return "ok"
194         except:
195             self.close_conn()
196             return "error trying to update new reset"
197 
198     #get all measurement devices
199     def get_devices(self):
200         self.start_conn()
```

Continuación de la figura 106.

```
201         sql = "SELECT device FROM devices"
202         try:
203             self.cursor.execute(sql)
204             list = self.cursor.fetchall()
205             self.close_conn()
206             return list
207         except:
208             self.close_conn()
209             return ()
210 
211     def get_next_reset(self):
212         self.start_conn()
213         sql = "SELECT device, next_reset, timezone FROM devices"
214         try:
215             self.cursor.execute(sql)
216             list = self.cursor.fetchall()
217             self.close_conn()
218             return list
219         except:
220             self.close_conn()
221             return ()
222 
223     #update measurement device's next reset
224     def set_next_reset(self, device, next_reset):
225         self.start_conn()
226         sql = ("UPDATE devices SET next_reset=" + '"' + str(next_reset) + '"' +
227                                     ", need_reset=1 " +
228                                     "WHERE device=" + '"' + device + '"')
229         try:
230             self.cursor.execute(sql)
231             self.close_conn()
232             return "ok"
233         except:
234             self.close_conn()
235             return "error trying to update next date reset"
236 
237     #update measurement device's information
238     def update(self, device, name, rate, currency, payday, timezone):
239         self.start_conn()
240         today = datetime.now(pytz.timezone(timezone)).date()
241 
242         #determine payday
243         if (int(payday)>=today.day):
244             next_reset = str(today.year)+"-"+str(today.month)+"-"+payday
245         else:
246             next_reset = str(today.year)+"-"+str(int(today.month) + 1)+"-"+payday
247 
248         #verify if timezone changed
249         sql = 'SELECT timezone FROM devices WHERE device="'+device+'"'
250         try:
```

Continuación de la figura 106.

```
251             self.cursor.execute(sql)
252             old_timezone = self.cursor.fetchall()
253             #if timezone change all registers of day will be deleted
254             #to avoid date issues
255             if (old_timezone[0][0] != timezone):
256                 sql = ("DELETE FROM " + device.replace("-","_") + " WHERE date>=" +
257                        '"' + str(today) + '"')
258                 self.cursor.execute(sql)
259         except:
260             pass
261 
262         sql = ("UPDATE devices SET name_device=" + '"' + name + '"' +
263                                    ", rate=" + '"' + rate + '"' +
264                                    ", currency=" + '"' + currency + '"' +
265                                    ", payday=" + '"' + payday + '"' +
266                                    ", timezone=" + '"' + timezone + '"' +
267                                    ", next_reset=" + '"' + next_reset + '"' +
268                                  " WHERE device=" + '"' + device + '"')
269         try:
270             self.cursor.execute(sql)
271             self.close_conn()
272             return "ok"
273         except:
274             self.close_conn()
275             return "error"
276 
277     def add_device(self, device):
278         self.start_conn()
279         sql = ("INSERT INTO devices VALUES(" + '"' + device + '"' +
280                ",NULL,NULL,NULL,NULL,NULL,NULL,false)")
281         sql2 = ("CREATE TABLE " + device.replace("-","_") +
282                 "(id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,"+
283                 "device VARCHAR(50),"+
284                 "name VARCHAR(50),"+
285                 "rate float,"+
286                 "currency VARCHAR(3),"+
287                 "payday TINYINT,"+
288                 "voltage float,"+
289                 "current float,"+
290                 "power float,"+
291                 "energy float,"+
292                 "charge float,"+
293                 "date DATE," +
294                 "time TIME,"+
295                 "timezone TEXT,"+
296                 "foreign key (device) references devices(device))")
297         try:
298             if(self.cursor.execute(sql)==1):
299                 if(self.cursor.execute(sql2)==0):
300                     self.close_conn()
```

Continuación de la figura 106.

```
301                         return "ok"
302                     else:
303                         self.close_conn()
304                         return device + " could not create consumption table"
305                 else:
306                     self.close_conn()
307                     return device + " could not be added to devices table"
308             except:
309                 self.close_conn()
310                 return "error"
311 
312     def delete_device(self, device):
313         self.start_conn()
314         sql = "DELETE FROM devices WHERE device=" + '"' + device + '"'
315         sql2 = "DROP TABLE " + device.replace("-","_")
316         try:
317             if(self.cursor.execute(sql2)==0):
318                 if(self.cursor.execute(sql)==1):
319                     self.close_conn()
320                     return "ok"
321                 else:
322                     self.close_conn()
323                     return device + " register could not be deleted"
324             else:
325                 self.close_conn()
326                 return device + " table could not be deleted"
327         except:
328             self.close_conn()
329             return "error"
330 
331     #open connection to database
332     def start_conn(self):
333         self.db=pymysql.connect(host=self.credentials[0],
334             port=int(self.credentials[2]),
335             user=self.credentials[1],
336             passwd=self.credentials[3],
337             db=self.credentials[4],
338             charset='utf8') #connect database
339         self.cursor = self.db.cursor()      # Get cursor() method
340 
341     #close connection to database
342     def close_conn(self):
343         self.db.commit()
344         self.cursor.close()
345         self.db.close()
```

Fuente: elaboración propia, realizado con captura de pantalla.

Figura 107. **Código archivo broker_data.py**

```python
import Crypto
import binascii
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
#-------------------------------- Decrypt credentials ---------
prikey = open("privateKey.txt", "r")
private_key = prikey.read()
prikey.close()

private_key = RSA.importKey(binascii.unhexlify(private_key))

encrypted_file = open("credentials_mqtt.txt", "rb")
encrypted_credentials = encrypted_file.read()
encrypted_file.close()

cipher = PKCS1_OAEP.new(private_key)
cred_mqtt = (cipher.decrypt(encrypted_credentials)).decode()
private_key=""
cred_mqtt = cred_mqtt.split('\n')
#--------------------------------------------------------------
```

Fuente: elaboración propia, realizado con captura de pantalla.

6.5.4. Manejo de *Topics* MQTT

El archivo server.py, es el encargado de recibir y administrar los mensajes MQTT, por lo tanto, también es el encargado de realizar las suscripciones a los *topics* MQTT necesarios para el funcionamiento de este monitor energético. En el siguiente listado, se describe brevemente la función de cada topic MQTT:

- energy: cumple la función de *topic* raíz, lo que significa que todos los demás *topics* secundarios o *subtopics* se derivan de este.

    - control: este *subtopic* es de uso administrativo lo que significa que no es accesible por los usuarios.

- add_device: por medio de este *topic* se envía al servidor la instrucción de agregar un nuevo dispositivo de medición al sistema.

- delete_device: cuando se requiere eliminar un dispositivo de medición se envía la instrucción al servidor por medio de este *topic*.

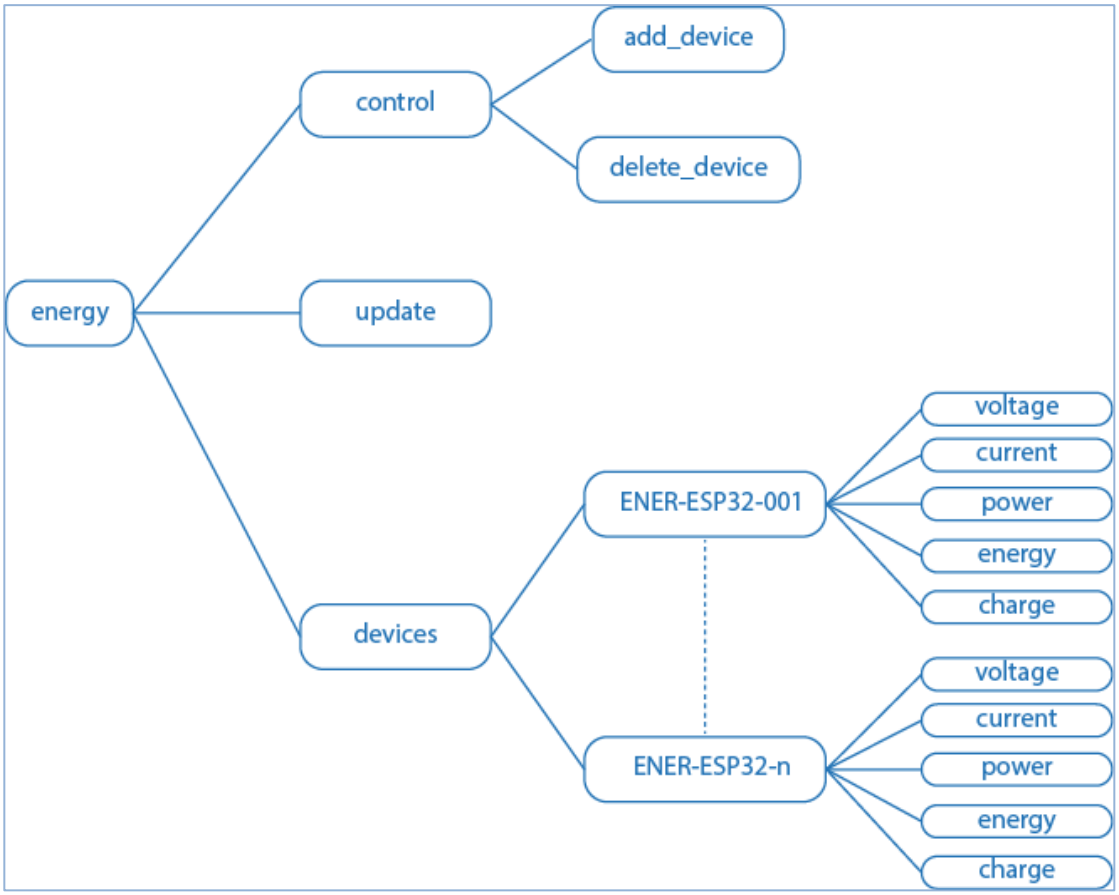
- update: por medio de este *topic* se manejan las actualizaciones en la configuración de los dispositivos de medición del sistema, cuando un usuario realiza configuraciones por medio de un dispositivo Android estas se transmiten al servidor utilizando este canal.

- *devices:* en este se agrupan los *topics* y *subtopics* pertenecientes a todos los dispositivos de medición.

  - ENER-ESP32-###: para guardar la información de consumo (solicitada a cada 10 minutos), en base de datos, consultar el historial y reiniciar el contador de consumo se usa este *topic*, por cada dispositivo de medición se tendrá un *topic* (y sus derivados), de este tipo. De este *topic* se derivan otros 5 en los cuales se transmiten las mediciones realizadas por el dispositivo de medición, además es importante aclarar que el archivo principal del servidor no interactúa con estos 5 *topics* ya que estos solo son utilizados para la visualización en tiempo real (en dispositivos Android), de las mediciones registrados por el dispositivo de medición. A continuación, se muestran los *subtopics* y el tipo de medición que comparten.

- ✓ voltaje: voltaje expresado en voltios
- ✓ current: corriente expresada en amperios
- ✓ power: potencia expresada en watts
- ✓ energy: energía expresada en kWh
- ✓ charge: valor monetario equivalente al consumo energético.

Figura 108. **Diagrama *topics* MQTT**



Fuente: elaboración propia, realizado con Adobe Illustrator.

7. MANUAL DE USUARIO

Por medio de este manual se describen e ilustran las instrucciones para la configuración y uso de este monitor energético, es importante seguir las instrucciones a manera de garantizar un funcionamiento óptimo en los diferentes dispositivos que conforman el sistema, para realizar las funciones de monitoreo es necesario contar con conexión a internet de lo contrario solo se tendrá disponible la opción de configuración.

7.1. Especificaciones energéticas

Los dispositivos eléctricos (clavija y enchufe), usados en el dispositivo de medición cumplen con la normativa NEMA 5-15, lo que significa que las especificaciones energéticas para este monitor dependen en gran parte de esta norma.

Tabla XXII. **Especificaciones energéticas**

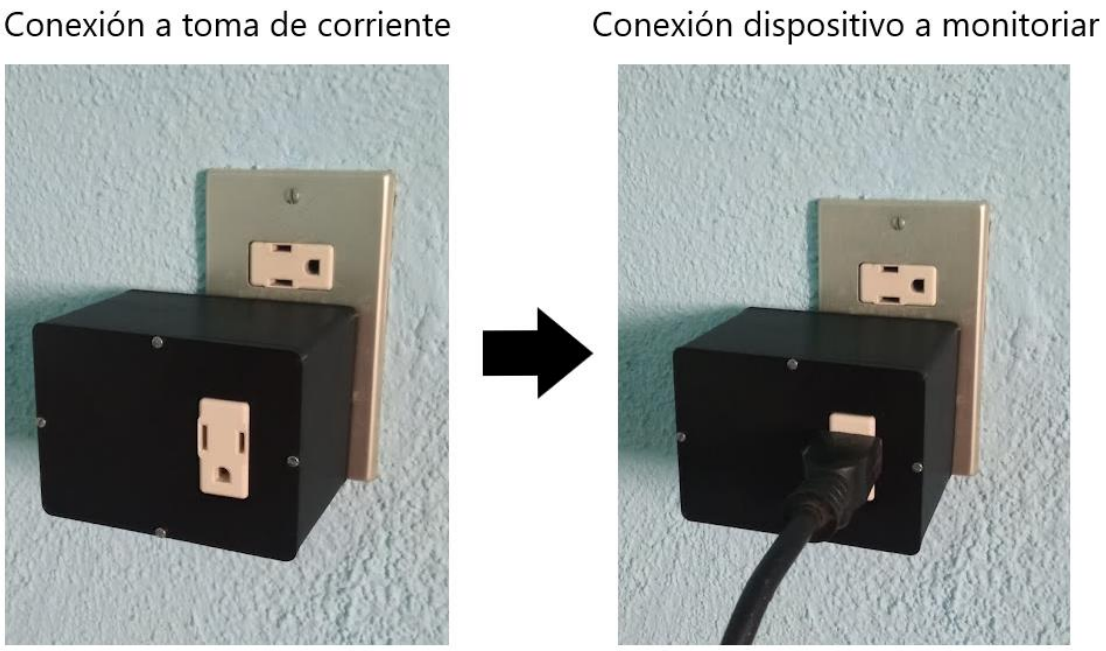
Especificaciones energéticas dispositivo de medición sin carga en la salida	
Voltaje	110 – 125 voltios
Corriente	0,03 amperios
Potencia	1,10 watts
Especificaciones energéticas salida dispositivo de medición	
Voltaje	110 – 125 voltios
Corriente máxima	15 amperios

Fuente: elaboración propia.

7.2. Conexión de dispositivo de medición

La forma correcta de realizar la conexión es primero conectando el dispositivo de medición a una toma de corriente, luego conectar a la salida el dispositivo al cual se le estará monitoreando. Desde el primer instante en el que el dispositivo de medición es energizado es capaz de registrar el consumo, aunque este no este configurado dado que el consumo es almacenado en memoria y consultado cuando sea necesario.

Figura 109. **Conexión de dispositivo de medición**



Fuente: elaboración propia, realizado con Paint3D.

7.3. Configuración inicial

Antes de poder visualizar y consultar el consumo energético es necesario realizar una configuración inicial para nuevos dispositivos de medición o para cuando se da el caso de realizar un nuevo ajuste en los parámetros energéticos que pueden cambiar con el transcurso del tiempo. Para realizar esta tarea se deberá contar con un dispositivo Android con bluetooth incorporado, los pasos a seguir son:

- Emparejar el dispositivo de medición por medio de bluetooth. Este procedimiento se realiza una única vez, en futuras configuraciones omitir este paso.

    - Abrir el menú de configuración del dispositivo Android
    - Buscar y entrar al apartado Bluetooth
    - Activar Bluetooth en caso se encuentre desactivado
    - Buscar en dispositivos disponibles ENER-ESP32-###
    - Vincular el dispositivo ENER-ESP32-###

- Abrir la aplicación dedicada y entrar al menú de configuración

    - En el menú principal elegir la opción Configurar un dispositivo de medición, es importante tener activada la función Bluetooth en Android para poder usar esta opción.

    - Se mostrará una lista con los dispositivos de medición emparejados previamente, seleccionar el dispositivo de interés.

- Conectar el dispositivo de medición a Internet

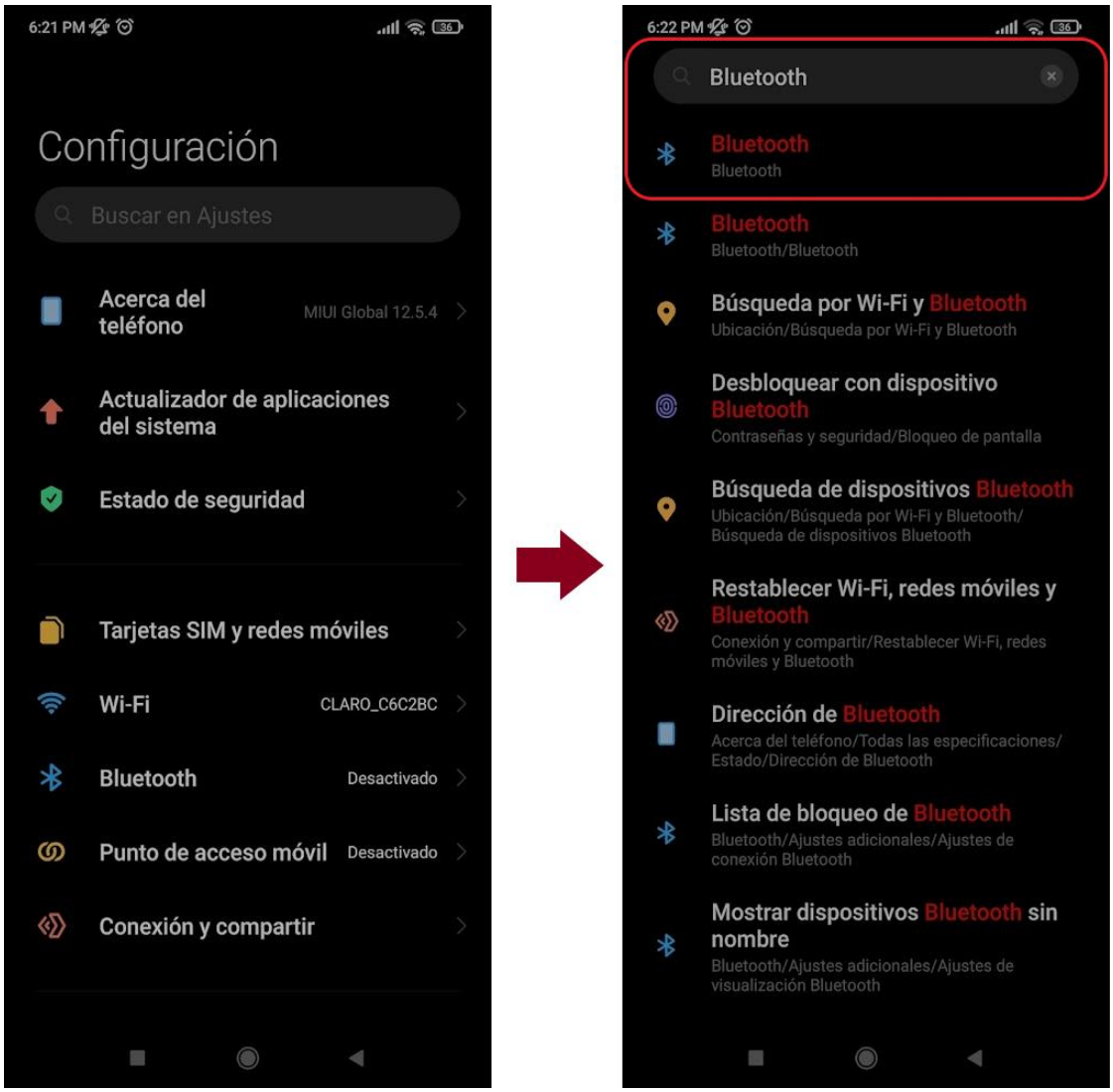
- En el apartado Credenciales WiFi del menú de configuración ingresar el nombre de la red y la contraseña.

- Pulsar el botón CONECTAR RED WIFI, si las credenciales son válidas cambiara el estado de la conexión.

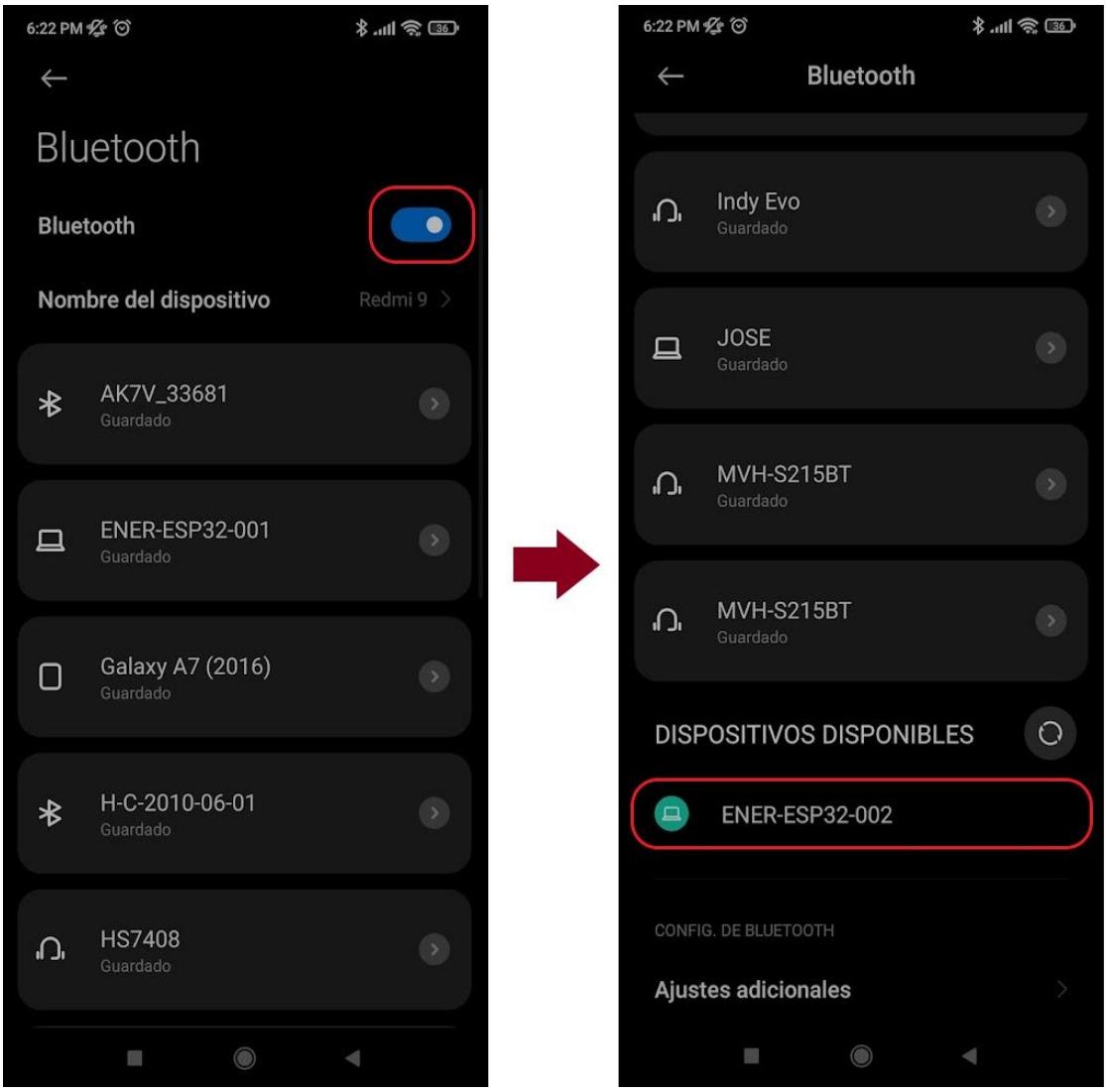
- Configurar parámetros energéticos

  - Tarifa energética: valor monetario por cada kWh consumido.
  - Moneda: moneda en la que se efectúa el pago.
  - Día de pago: día en el que se emite el recibo de cobro.
  - Nombre del dispositivo: corresponde al dispositivo sobre el cual se realizará el monitoreo.
  - Agregar dispositivo al visualizador: esta opción debe de estar habilitada para poder visualizar el consumo energético de lo contrario solo se podrá configurar.
  - Pulsar el botón GUARDAR CONFIGURACION, si todos los parámetros son correctos se mostrará un mensaje indicando el éxito en la operación.

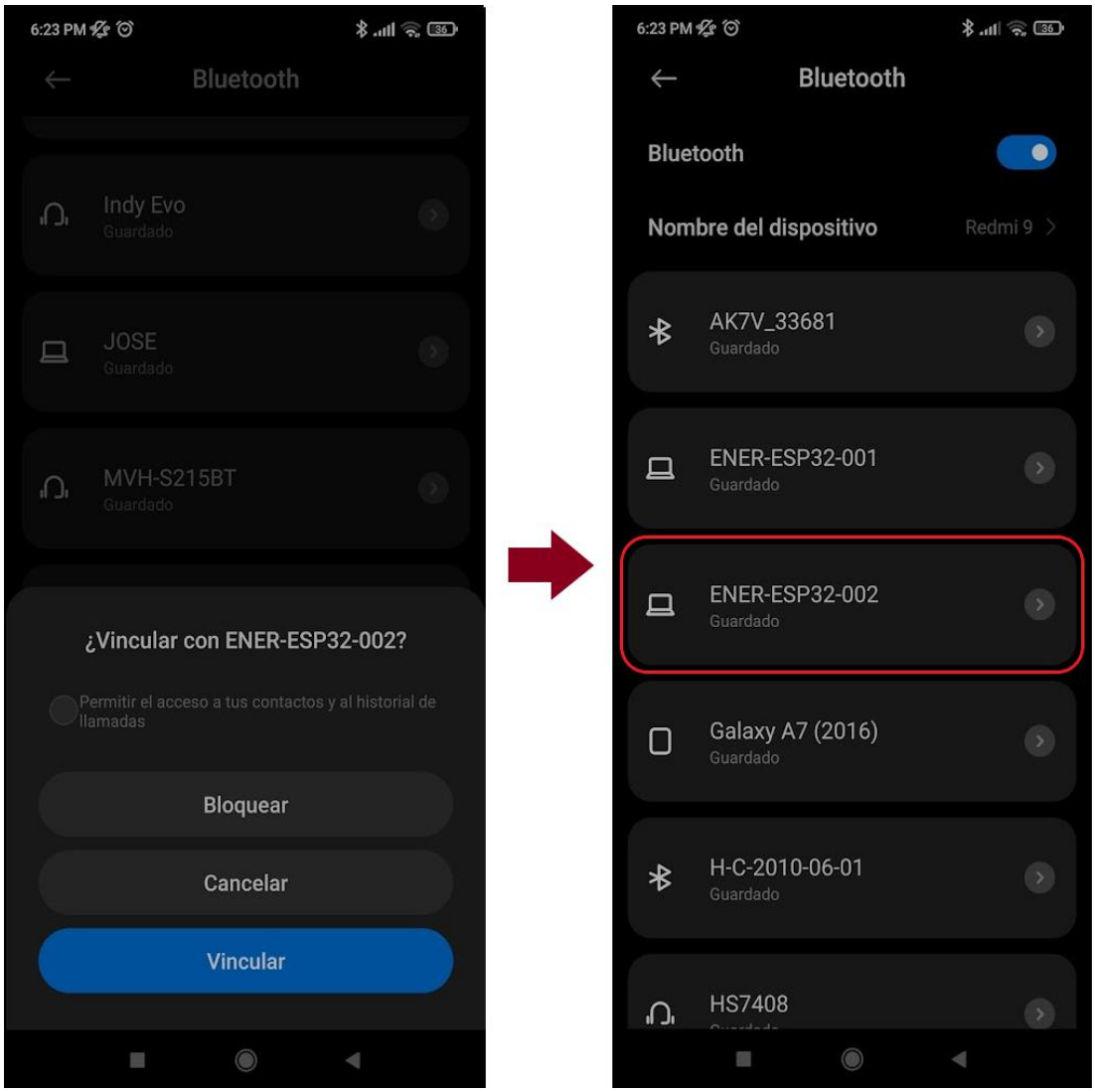
Figura 110. **Vincular un dispositivo de medición por medio de Bluetooth Android**



Continuación de la figura 110.

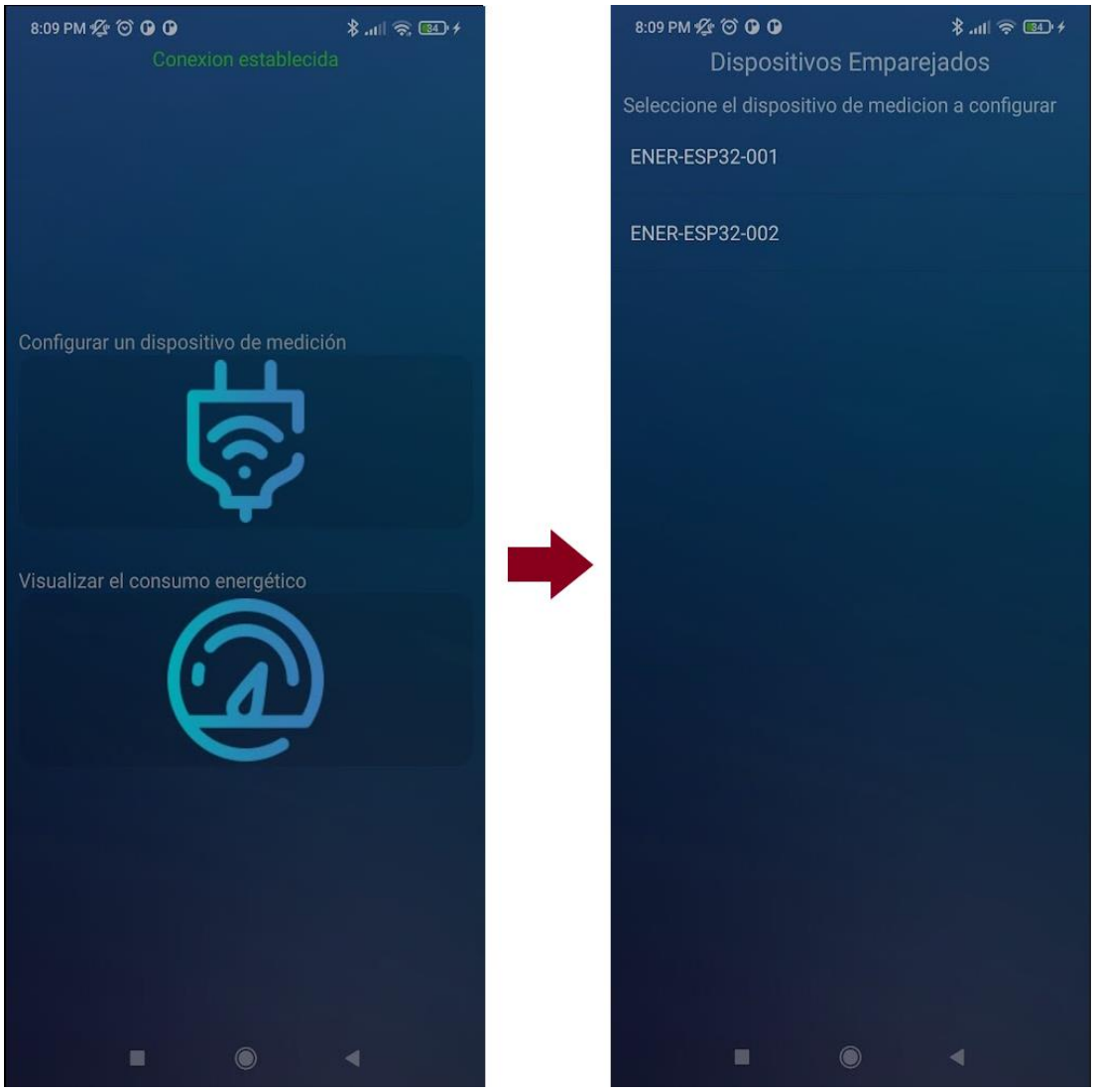


Continuación de la figura 110.

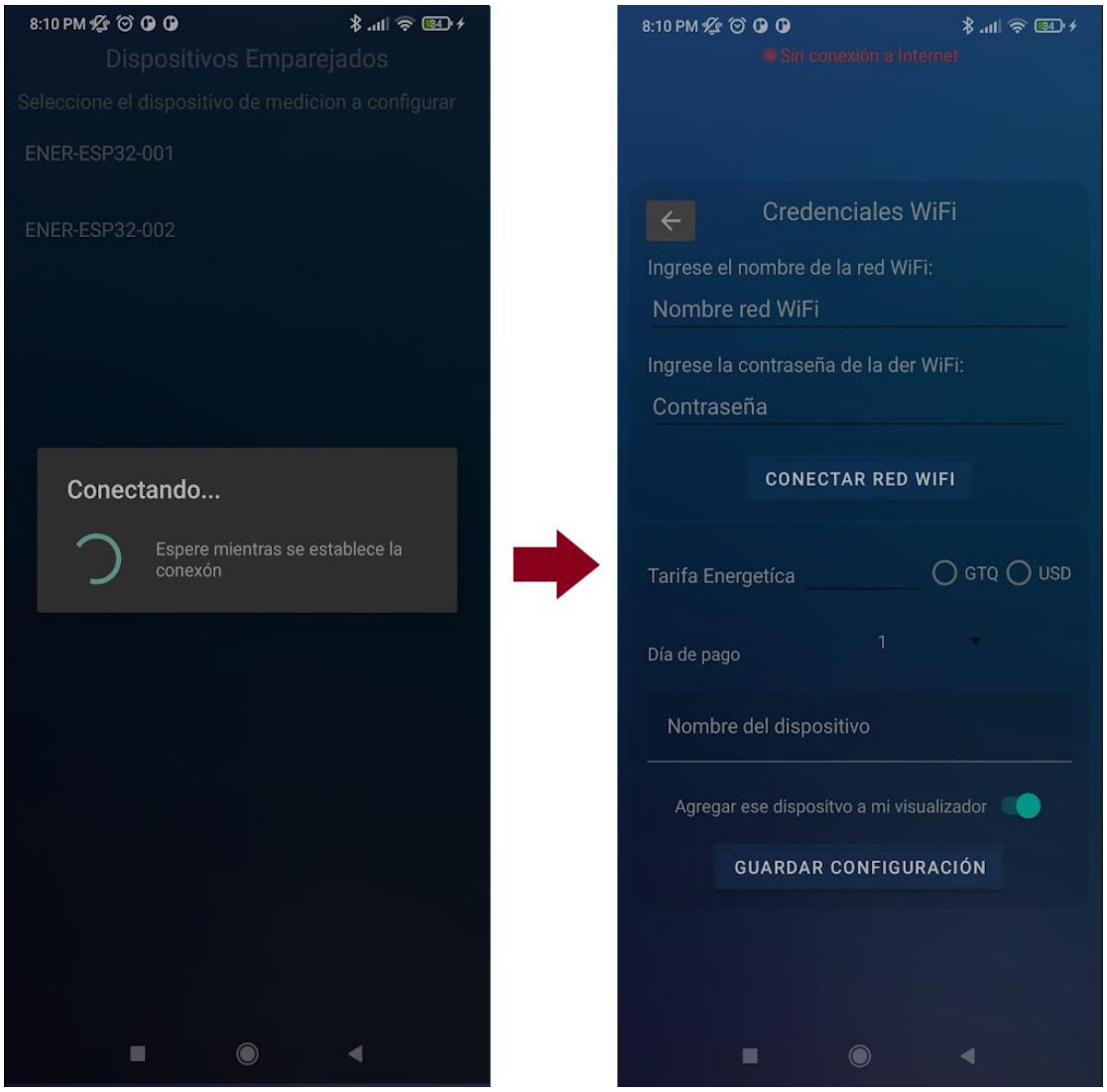


Fuente: elaboración propia, realizado con Paint3D.

Figura 111. **Abrir menú de configuración dispositivo de medición**

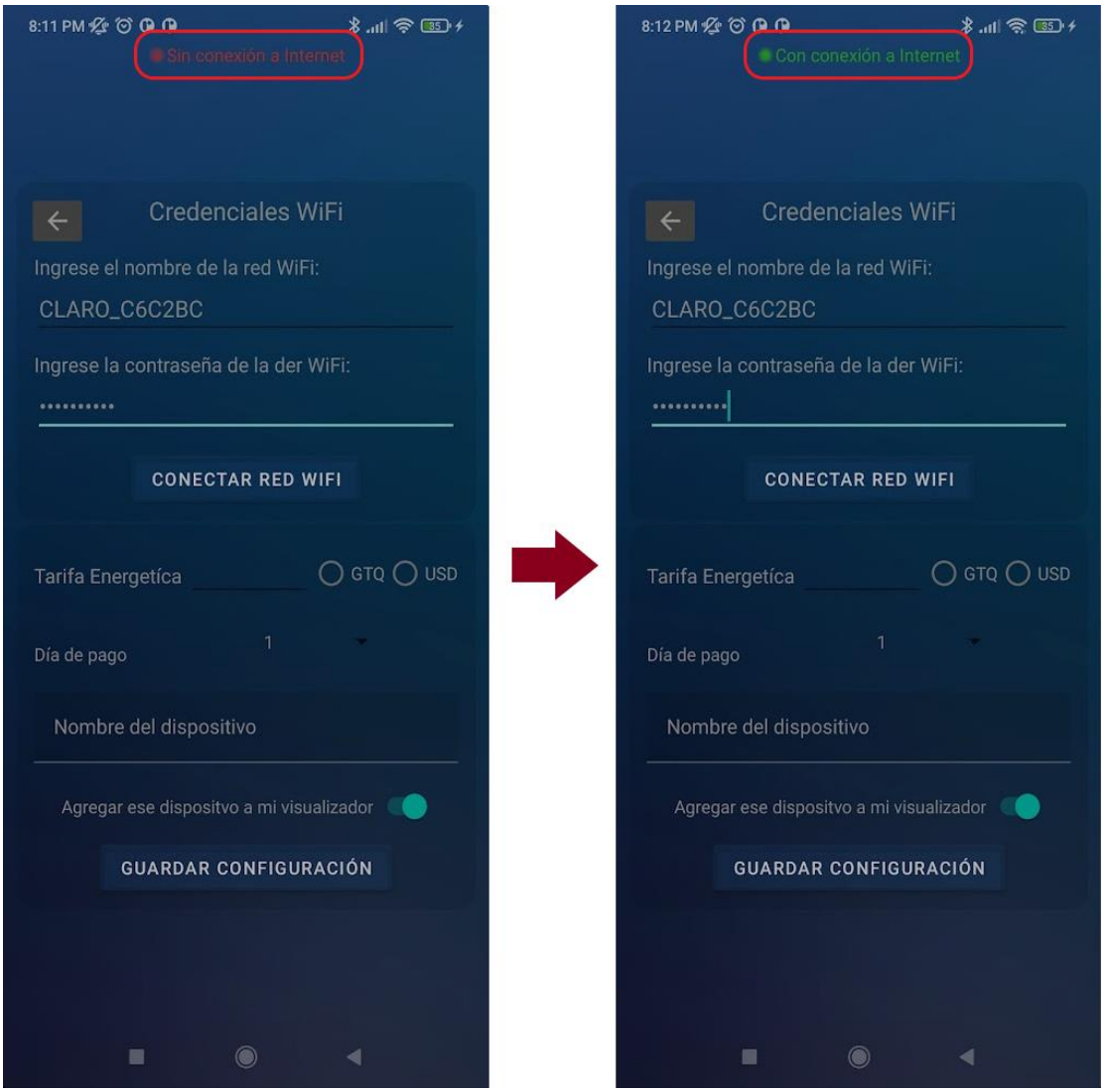


Continuación de la figura 111.



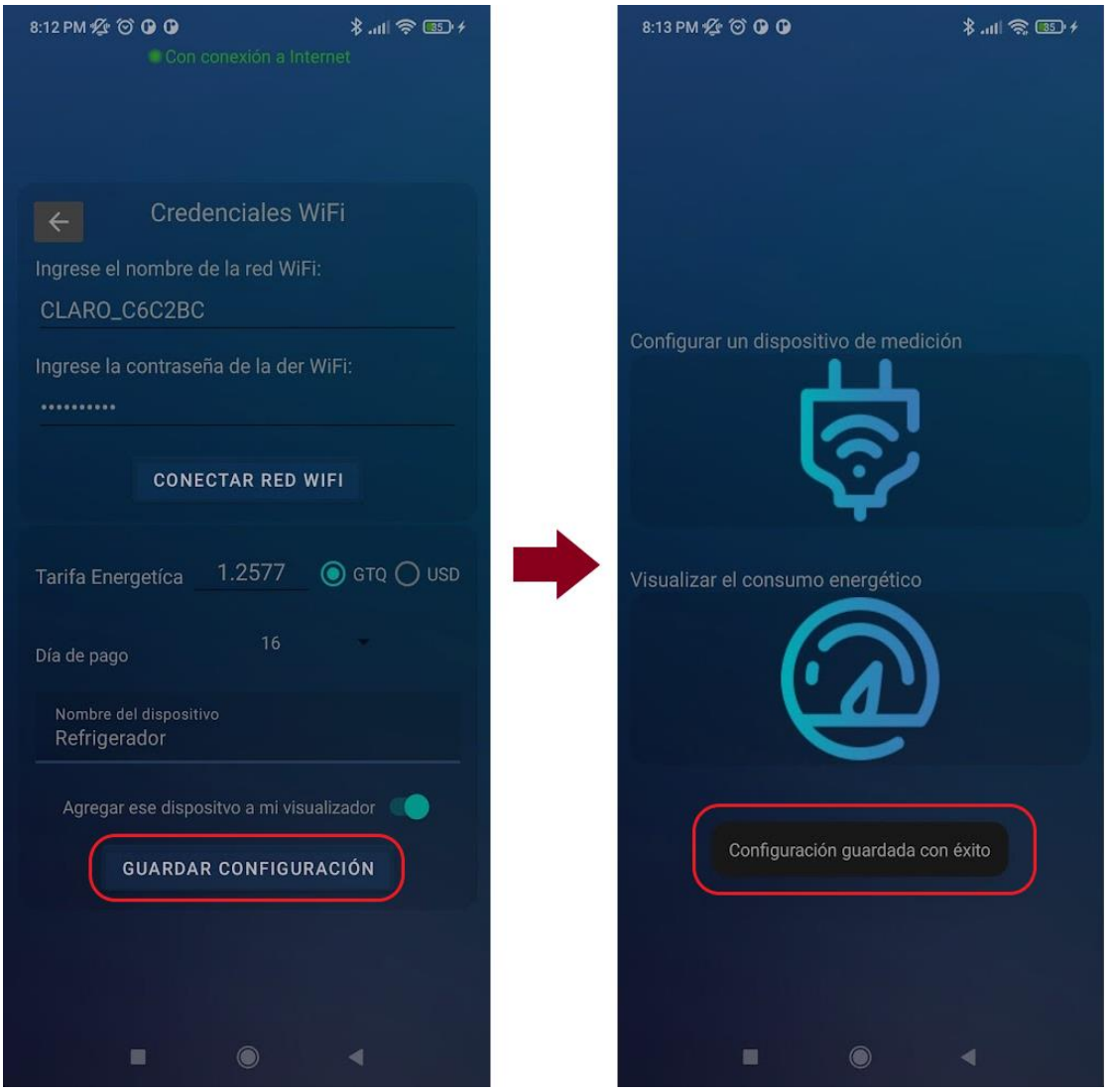
Fuente: elaboración propia, realizado con Paint3D.

Figura 112. **Conectar dispositivo de medición a Internet**



Fuente: elaboración propia, realizado con Paint3D.

Figura 113. **Configurar parámetros energéticos dispositivo de medición**

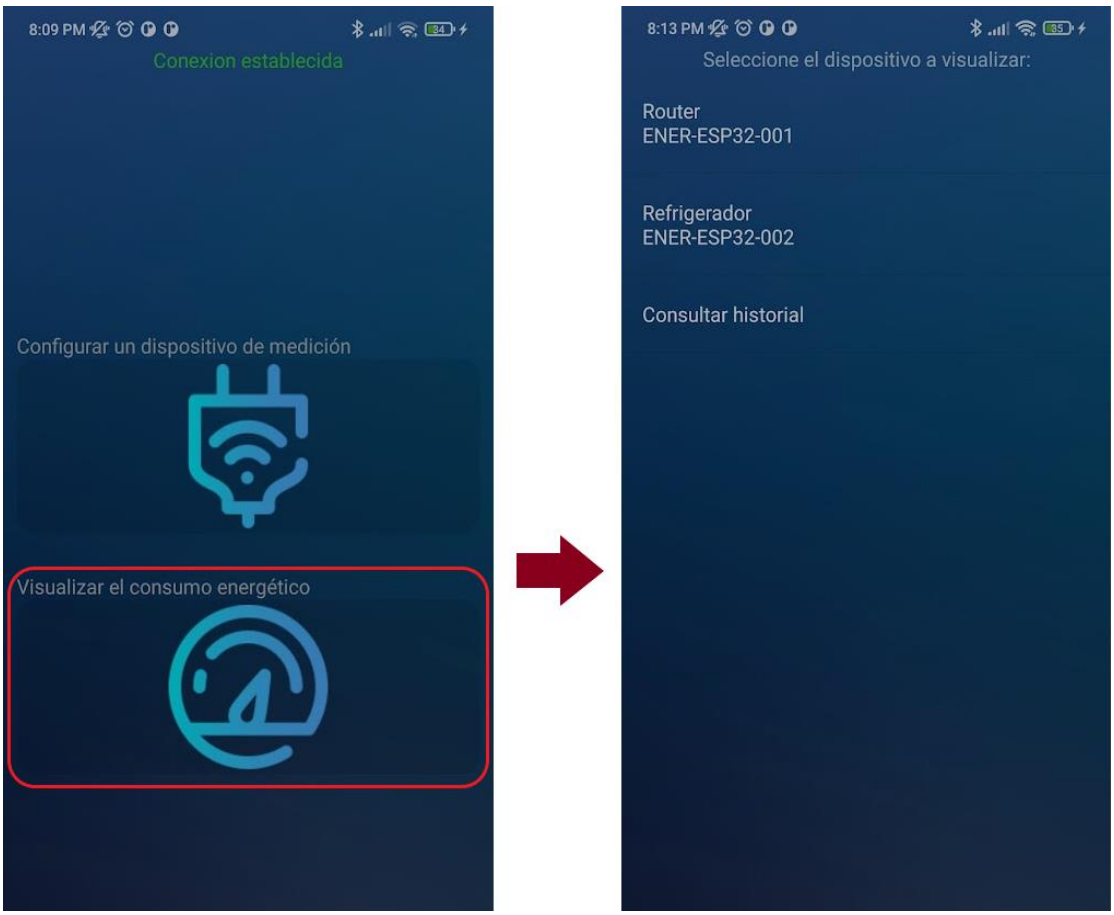


Fuente: elaboración propia, realizado con Paint3D.

7.4. Visualización de consumo

Esta función está disponible sólo si el dispositivo Android cuenta con conexión a Internet y si se cuentan con dispositivos de medición previamente agregados al visualizador, este menú cuenta con las opciones de: ver el consumo energético individual en tiempo real y consultar el consumo energético por día y entre dos fechas. Para acceder a este menú presionar la opción “visualizar el consumo energético” del menú principal.

Figura 114. **Acceder a menú de visualización**



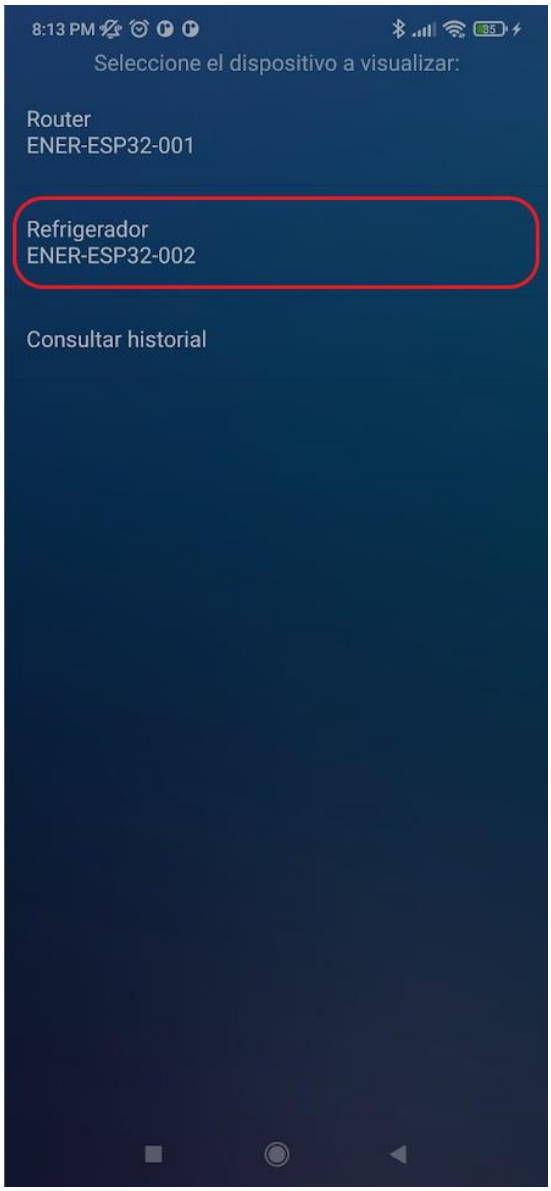
Fuente: elaboración propia, realizado con Paint3D.

7.4.1. Consumo en tiempo real

Muestra una visualización del consumo energético acompañado de valores de interés como lo son: voltaje, corriente y potencia, cuenta con una gráfica que traza el consumo energético a lo largo del tiempo, dicha gráfica permite observar el cambio en el consumo con respecto al tiempo, todos los valores mostrados en el visualizador se actualizan a cada segundo. Para acceder al visualizador primero se debe seleccionar el dispositivo de interés en consecuencia la visualización en tiempo real funciona de manera individual por cada dispositivo registrado.

En la gráfica de la figura 117 el tiempo inicial corresponde a la hora (formato 24 h), en la que se recibieron las primeras mediciones desde que se abrió el visualizador, el tiempo actual corresponde a la hora en la que se recibieron las últimas mediciones, la energía consumida y el valor monetario representan a los valores acumulados durante el periodo de cobro previamente definido mediante la configuración del día de pago.

Figura 115. **Visualizador de consumo**



Continuación de la figura 115.

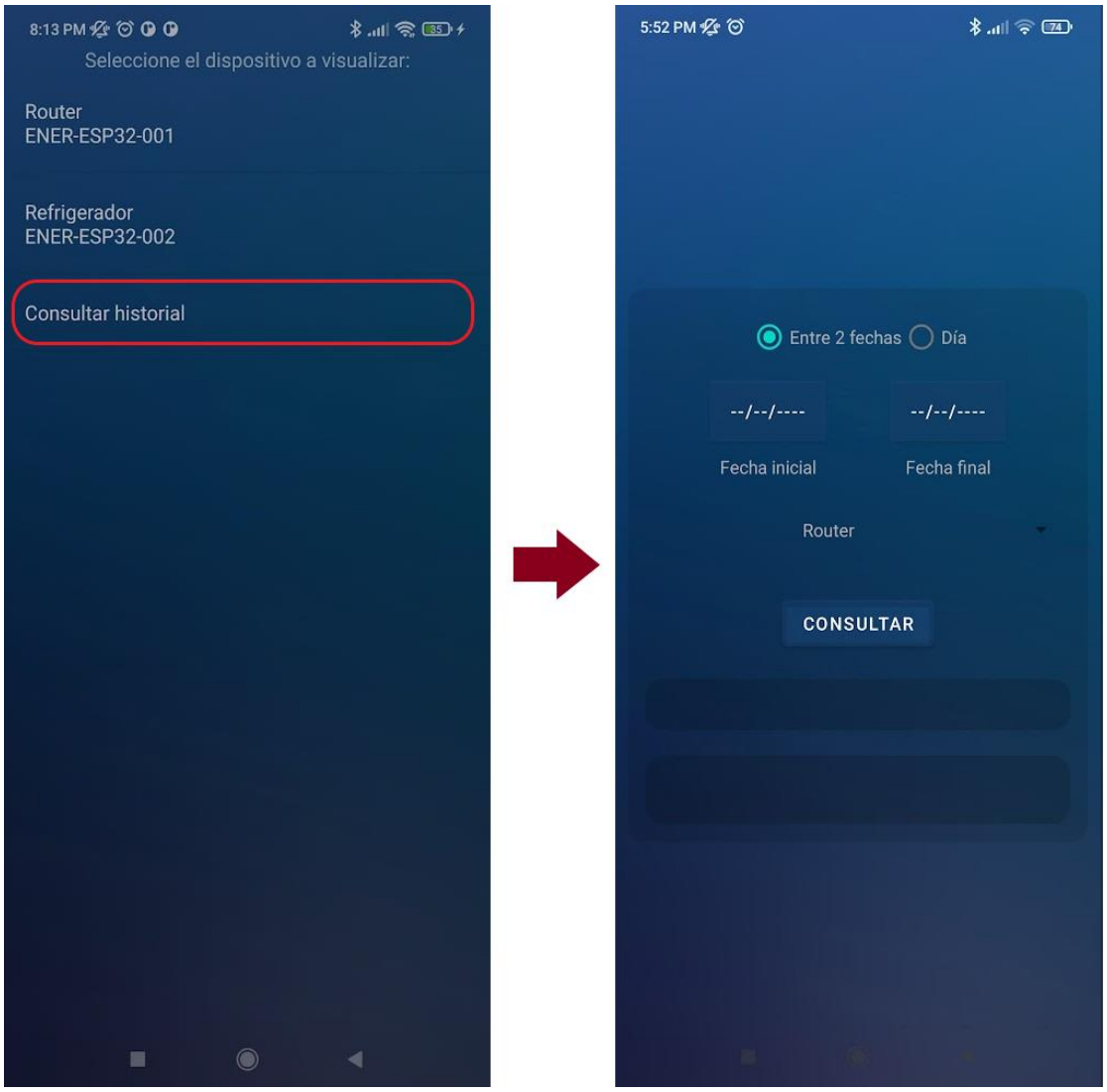
Fuente: elaboración propia, realizado con Paint3D.

7.4.2. Historial de consumo

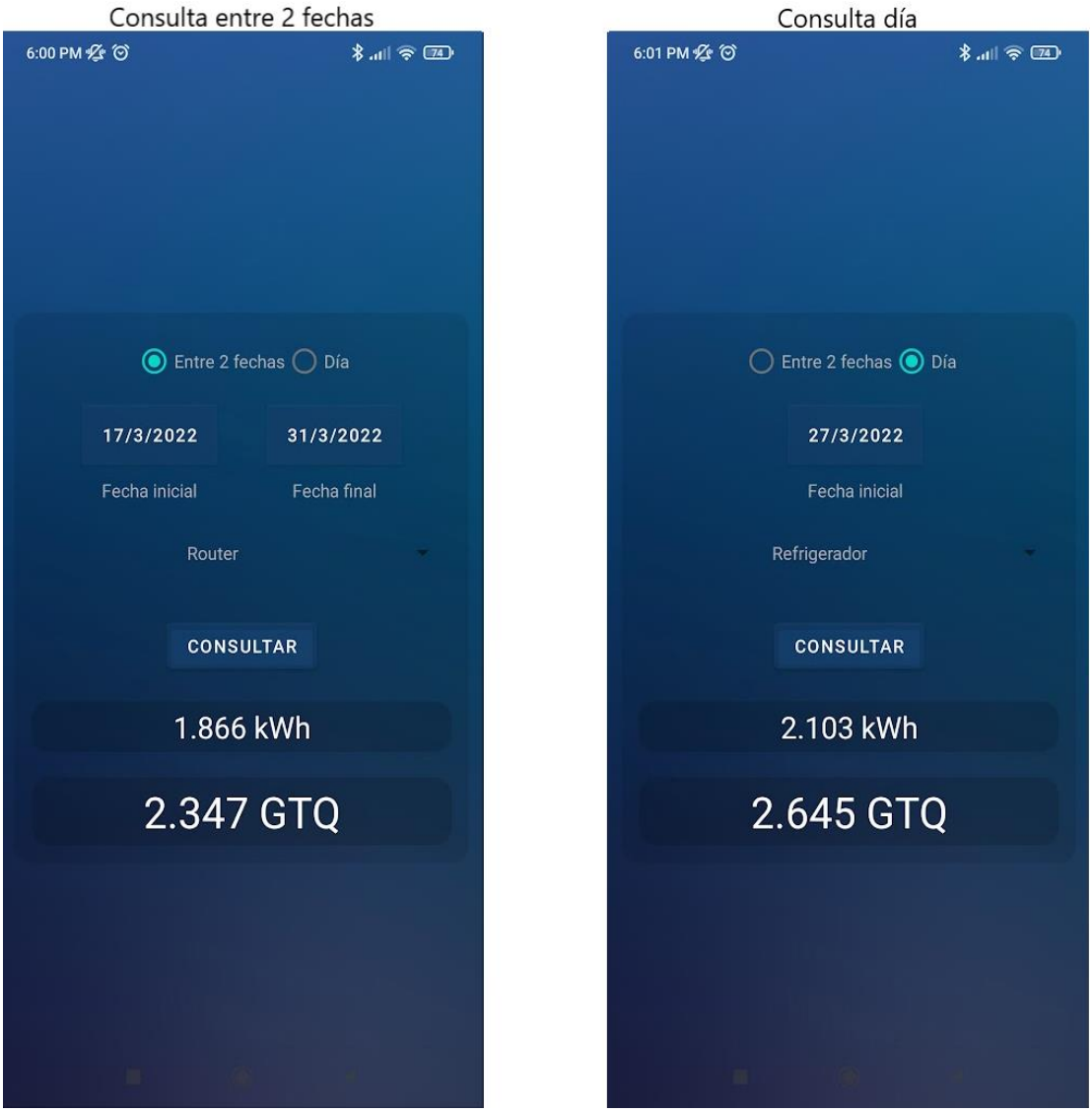
Para dispositivos de medición previamente configurados se tiene la opción de consultar el consumo energético en un determinado periodo de tiempo (entre días), o bien en un único día, lo cual es muy útil a la hora de querer obtener el gasto monetario que generan ciertos electrodomésticos o dispositivos de interés. Para acceder a esta función se debe seleccionar la opción Consultar historial, del menú de visualización. Pasos para realizar una consulta:

- Elegir el tipo de consulta
    - Entre 2 fechas
    - Día
- Elegir el dispositivo de interés
- Pulsar el botón CONSULTAR
- Si existen registros para la consulta solicitada se procede a mostrar el consumo (representado en kWh), y el monto monetario correspondiente.

Figura 116. **Consultar historial energético**



Continuación de la figura 116.



Fuente: elaboración propia, realizado con Paint3D.

7.5. Mantenimiento

El dispositivo de medición requiere de un mínimo mantenimiento para funcionar correctamente, y consiste en evitar exponer el dispositivo a altas temperaturas capaces de fundir los materiales con los que está fabricado, también se debe evitar botar el dispositivo ya que el impacto de la caída puede provocar la avería de los componentes electrónicos. A pesar de tomar todos los cuidados necesarios para manipular el dispositivo de medición, siempre cabe la posibilidad de un fallo debido a la avería temprana de sus componentes.

7.5.1. Errores de software

Se pueden producir errores en el monitor energético debidos a problemas con el software de los distintos procesos en ejecución, estos pueden presentarse por falta de depuración en el código de programación o simplemente por actualizaciones del sistema que pueden dejar obsoletas algunas funciones del monitor. Para solucionar algún problema presentado por el software del sistema es necesario informar sobre el mismo por medio del repositorio de GitHub.

7.5.1.1. Informar sobre un problema

Para poder informar sobre un problema de software es necesario tener una cuenta en GitHub, si no se tiene una, el sitio web (https://github.com), indica de manera clara como poder crear una cuenta nueva totalmente gratuita. Los errores de software pueden presentarse del lado servidor o bien del cliente, cuando se genera un error del lado del cliente este puede originarse en el dispositivo de medición o en la aplicación Android perteneciente al monitor energético. Pasos para informar sobre un problema:

- Acceder al repositorio por medio de la URL: https://github.com/jose-199/MonitorEnergetico.
- Abrir la sección denominada *issues*
- Hacer *click* en el botón *New issue*, si aún no se ha iniciado sesión el sitio lo solicitara.
- Se abrirá una ventana que solicitará los siguientes campos:

    - *Title*: en este campo se ingresa el título que tendrá el informe del error.

    - *Write*: aquí se describe el error encontrado o presentado a fin de ser solucionado. En este apartando también se permite adjuntar archivos que permitan describir de mejor forma el error.

- Enviar el informe haciendo clic en el botón *submit new issue.*

Figura 117. **Informar sobre un problema GitHub *(new issue*)**

Fuente: elaboración propia, realizado con captura de pantalla.

7.5.2. Posibles fallas en dispositivo de medición

Estas fallas son debidas a un mal funcionamiento de los sensores y componentes electrónicos del dispositivo, algunas de estas fallas pueden solucionarse con la reprogramación del microcontrolador, pero algunas otras requerían una sustitución de componentes y se recomienda sustituirlos por unos con las mismas especificaciones, de lo contrario las fallas pueden volver a reincidir en un corto periodo de tiempo o bien pueden provocar la avería de otros componentes.

7.5.2.1. Conexión Bluetooth

Esta conexión es utilizada para configurar el dispositivo de medición y si esta falla no se podrán realizar configuraciones iniciales o bien modificaciones de estas, las fallas más comunes son:

- El dispositivo Android no encuentra el dispositivo de medición para emparejar. Soluciones:
    - Verificar si la toma de corriente a la que se está conectando el dispositivo esta energizada, si no proceder a conectar a otra toma de corriente.
    - Usar otros dispositivos Android para tratar de emparejar el dispositivo de medición.
    - Si el microcontrolador cuenta con corriente eléctrica y además ningún dispositivo Android puede encontrarlo, probablemente el microcontrolador este dañado y necesite sustitución.

- El dispositivo de medición esta emparejado, pero no se puede establecer la conexión para configuración. Soluciones:

    - Verificar que el dispositivo Android tenga encendido el Bluetooth.
    - Verificar si la toma de corriente a la que se está conectando el dispositivo esta energizada, si no proceder a conectar a otra toma de corriente.

7.5.2.2. Conexión WiFi

La conexión WiFi en el dispositivo de medición es utilizada para conectarse a la red de internet por medio de un punto de acceso y de esta manera poder comunicarse con el servidor remoto con el propósito de consultar el consumo energético y su historial, por lo tanto, si esta conexión falla no se podrá visualizar ninguna información relacionada al consumo energético.

- Al ingresar el nombre de la red y contraseña, no se establece la conexión con el punto de acceso.

    - Verificar que el nombre y la contraseña de la red sean correctos
    - Verificar que el punto de acceso WiFi este habilitado

- Las credenciales (nombre de la red y contraseña) son correctas, pero aún no se tiene acceso a internet.

    - Asegurarse que el punto de acceso tenga conexión a internet, de lo contrario solo se realizara una conexión local.

7.5.2.3. Averías componentes electrónicos

Las averías en los componentes electrónicos pueden hacer que el dispositivo de medición no funcione correctamente o bien no funcione en su totalidad, es importante detectar el componente que está ocasionando la falla a fin de llevar a cabo su sustitución. A continuación, se presenta una guía de diagnóstico con el propósito de identificar los componentes en mal estado.

- Fuente conmutada: si este componente falla el dispositivo de medición no será capaz de funcionar y por ello el microcontrolador y el sensor energético no tendrán alimentación, indicadores de una fuente averiada:

    - El dispositivo de medición está conectado a un enchufe energizado y este no funciona.

- Microcontrolador: cuando este componente resulta averiado puede producir una falla parcial o total, indicadores de un microcontrolador en mal estado:

    - No es posible establecer una conexión Bluetooth o WiFi cuando el dispositivo Android si puede establecerla con otros dispositivos de medición.

    - El microcontrolador permanece apagado aun cuando la fuente de alimentación está en buen estado.

- Sensor energético (PZEM-004T): este componente es el encargado de registrar el consumo energético por medio de un transformador de

corriente y un medidor de voltaje alterno, las señales que indican un sensor en mal estado son las siguientes:

- No se registra ningún consumo energético
- El consumo energético dejo de aumentar cuando aún se demanda energía eléctrica.
- Los indicadores luminosos del sensor dejaron de funcionar.

Los componentes electrónicos por sustituir deben ser de las mismas características y especificaciones que los anteriores, para conocer el proceso de desensamble de piezas consultar el capítulo 4.5, donde se explica de forma gráfica la manera en que las piezas fueron armadas, de esta manera se podrá realizar el proceso inverso llevando así a la separación de las piezas para luego continuar con la sustitución de los componentes.

CONCLUSIONES

1. Los transformadores de corriente pueden ser usados para construir sensores con la capacidad de medir el consumo energético por medio de la inducción debida al cable conductor.

2. Teniendo el valor de consumo energético expresado en kWh se puede determinar el costo monetario equivalente a partir de la tarifa energética brindaba por la empresa encarga del suministro eléctrico.

3. Por medio de un microcontrolador compatible con la tecnología WiFi es posible diseñar e implementar dispositivos capaces de ser controlados de manera remota mediante una dirección IP.

4. Mediante el código mostrado (también compartido por medio de un repositorio en GitHub), en este trabajo de graduación se puede implementar de manera total o parcial el sistema utilizado por el monitor energético.

5. Los recursos pertenecientes a este trabajo de graduación pueden ser utilizados y modificados por otros desarrolladores a fin de adaptarlos a las necesidades de su propia aplicación.

RECOMENDACIONES

1. Ubicar el transformador de corriente de una manera tal que no pueda ser obstruido por cables ajenos a la medición u otros objetos que puedan causar interferencias en el funcionamiento.

2. Conocer en detalle los rubros de cobros aplicados a la factura eléctrica del sector con el fin de estimar un valor monetario más preciso en el consumo energético.

3. Implementar protocolos de seguridad de cifrado para establecer comunicaciones más seguras entre dispositivos de internet.

4. Utilizar el equipo de protección personal adecuado al realizar trabajos que involucren corriente eléctrica, esto con el propósito de resguardar la integridad física de las personas involucradas.

5. Emplear herramientas e instrumentos que cumplan con las especificaciones y estándares de calidad normados para la manipulación eléctrica.

REFERENCIAS

1. Android developers. (2021). *Descripción general de proyectos.* Recuperado de https://developer.android.com/studio/projects.

2. Android developers. (2021). *Introducción a Android Studio.* Recuperado de https://developer.android.com/studio/intro.

3. Android developers. (2022). *Descripción general del manifiesto de una app.* Recuperado de https://developer.android.com/guide/topics/manifest/manifest-intro.

4. Beginners Guide To The MQTT Protocol. (2022). *Steve's Internet Guide*. Recuperado de http://www.steves-internet-guide.com/mqtt/.

5. DigitalOcean. (2018). *How to connect to your Droplet with OpenSSH.* Recuperado de https://docs.digitalocean.com/products/droplets/how-to/connect-with-ssh/openssh/.

6. DigitalOcean. (2020). *Configuración inicial del servidor con Ubuntu 20.04.* Recuperado de https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-20-04-es.

7. Edminister, J. (1985). *Circuitos eléctricos*. México: McGraw-Hill.

8. Floyd, L. (2007). *Principios de circuitos eléctricos*. Naucalpan de Juárez, México: Pearson Educación.

9. Impresoras3D.com. (1 de enero, 2018). Guía definitiva sobre tipos de filamentos 3D [Mensaje en un blog]. Recuperado de https://www.impresoras3d.com/la-guia-definitiva-sobre-los-distintos-filamentos-para-impresoras-3d/.

10. Larry. (3 de junio, 2020). NEMA Connectors in Power Cords [Mensaje en un blog]. Recuperado de https://community.fs.com/blog/nema-connectors-in-power-cords.html.

11. Lozano, R. (13 de junio, 2021). Programar ESP32 con IDE Arduino [Mensaje en un blog]. Recuperado de https://www.taloselectronics.com/blogs/tutoriales/programar-esp32-con-ide-arduino.

12. Meirer, A. Kaufmann, M. (2019). *SQL & NoSQL Databases.* Alemania: Springer Vieweg.

13. Oppel, A. Sheldon, R. (2010). *Fundamentos de SQL*. Ciudad de México, México: McGraw-Hill.

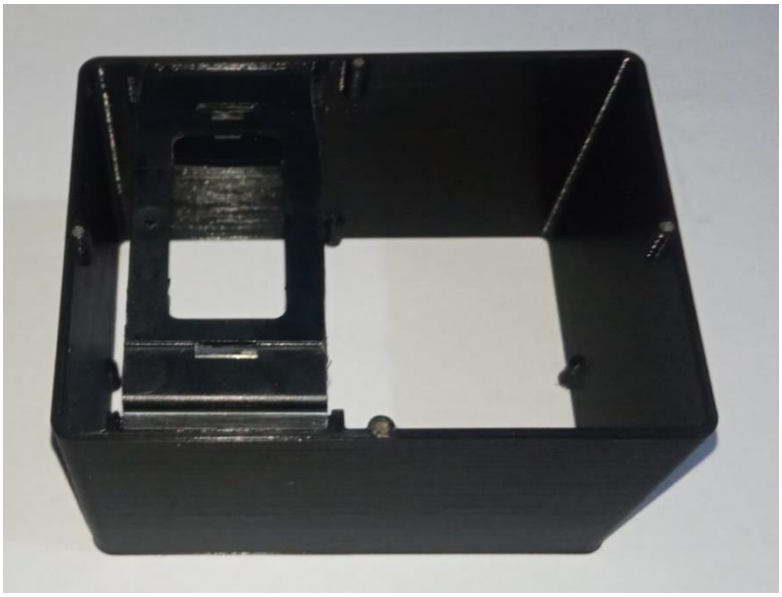
14. PlatformIO. (2014). *What is PlatformIO?.* Recuperado de https://docs.platformio.org/en/latest/what-is-platformio.html.

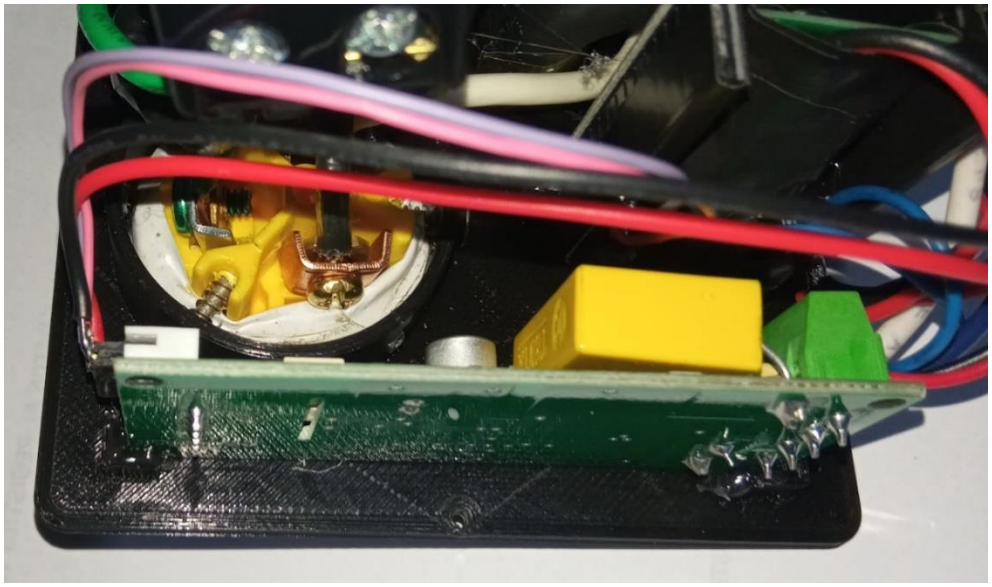
15. RedHat. (2019). *El concepto de IDE*. Recuperado de https://www.redhat.com/es/topics/middleware/what-is-ide.

16. Regidor, A. (22 de noviembre, 2016). *¿Qué es un archivo.STL?.* Recuperado de https://www.impresion3daily.es/que-es-un-archivo-stl/.

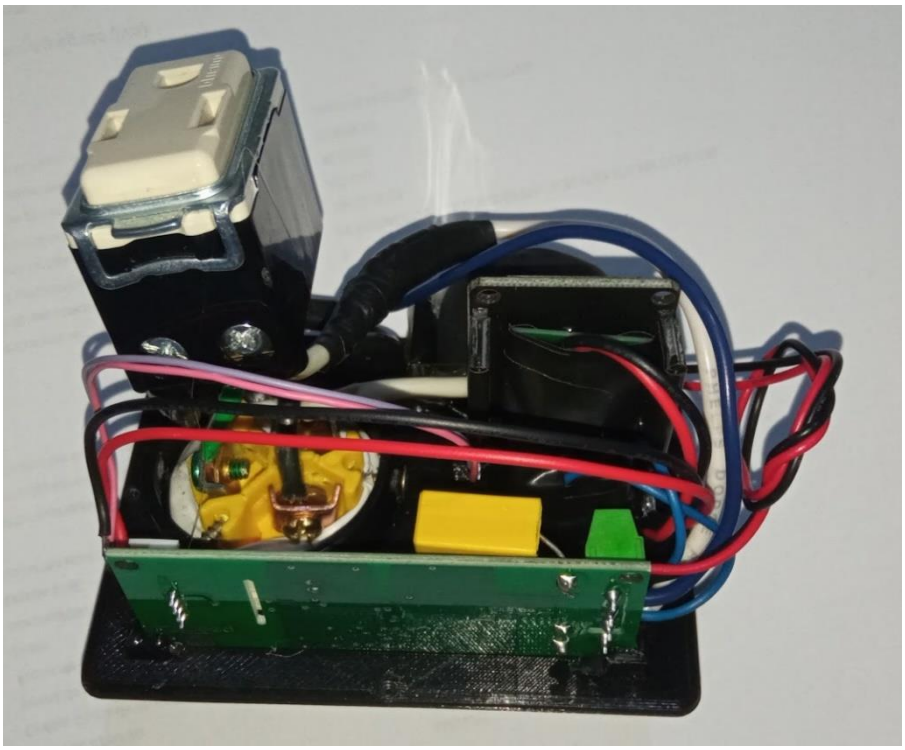
APÉNDICES

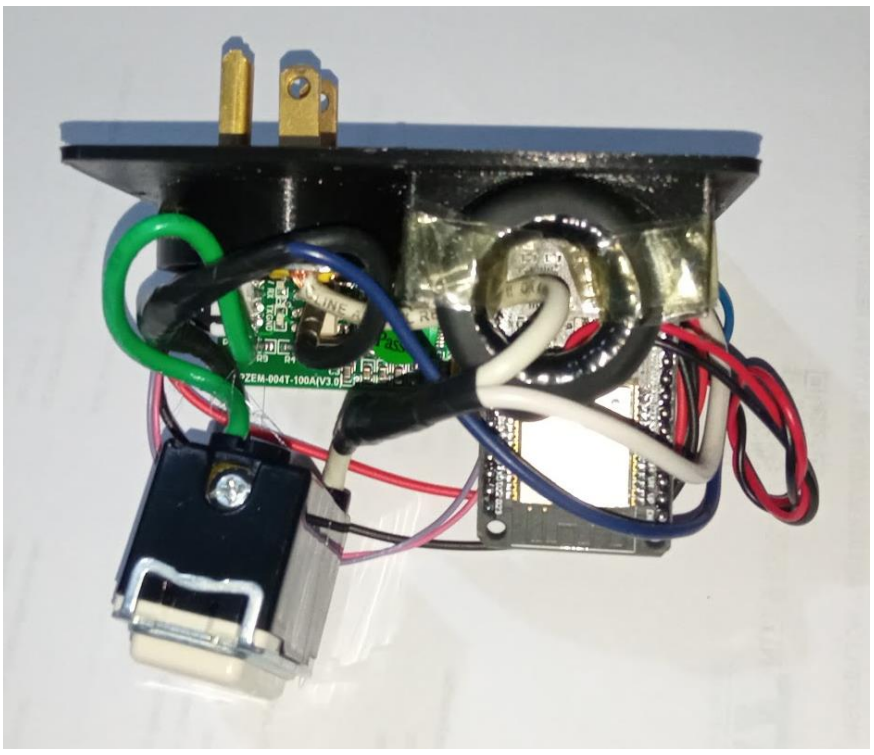
Apéndice 1. **Ensamble dispositivo de medición**



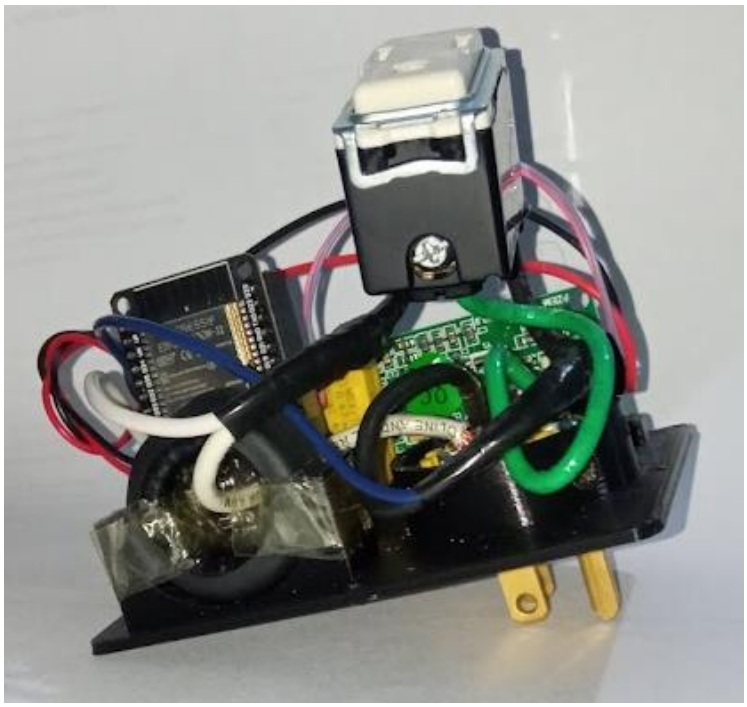


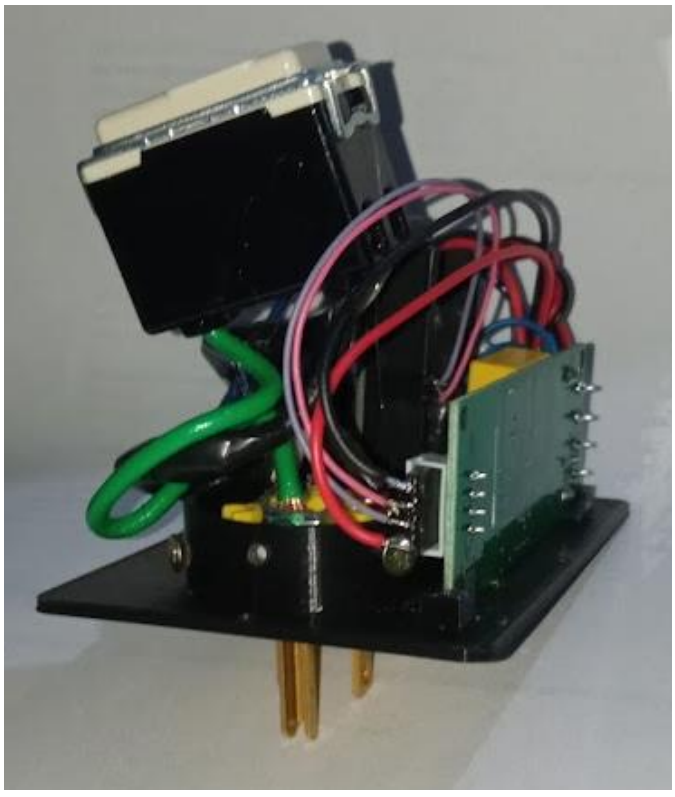
Continuación del apéndice 1.





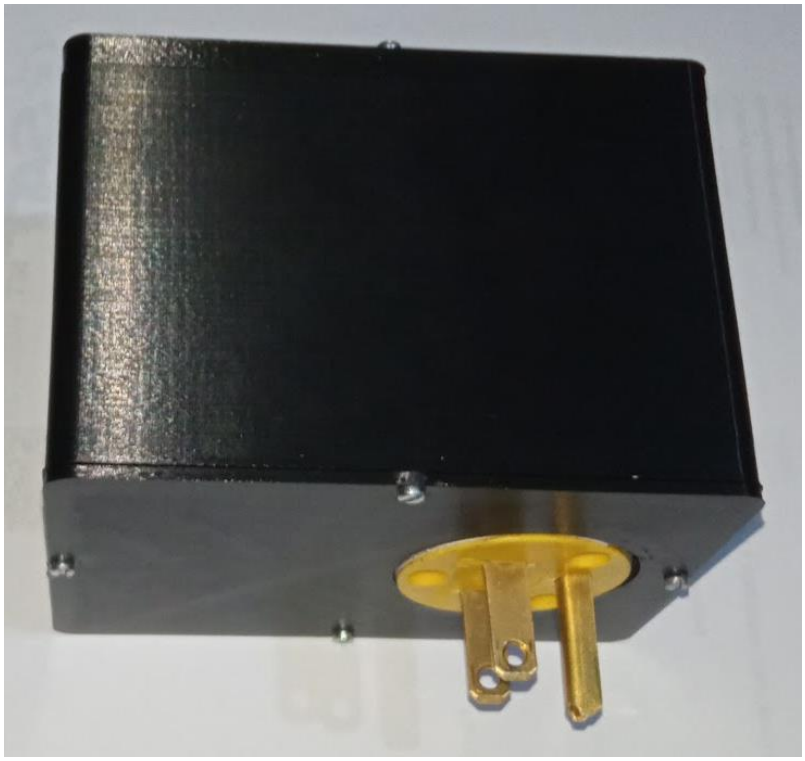
Continuación del apéndice 1.





Continuación del apéndice 1.

Continuación del apéndice 1.



Continuación del apéndice 1.

Fuente: elaboración propia.

Apéndice 2. **Consumo energético de un refrigerador (consulta de 23 días)**

Fuente: elaboración propia, realizado con captura de pantalla.

Apéndice 3. **Consumo energético de un *router* (consulta de 23 días)**

Fuente: elaboración propia, realizado con captura de pantalla.

Apéndice 4. **Visualización energética en tiempo real de un refrigerador**

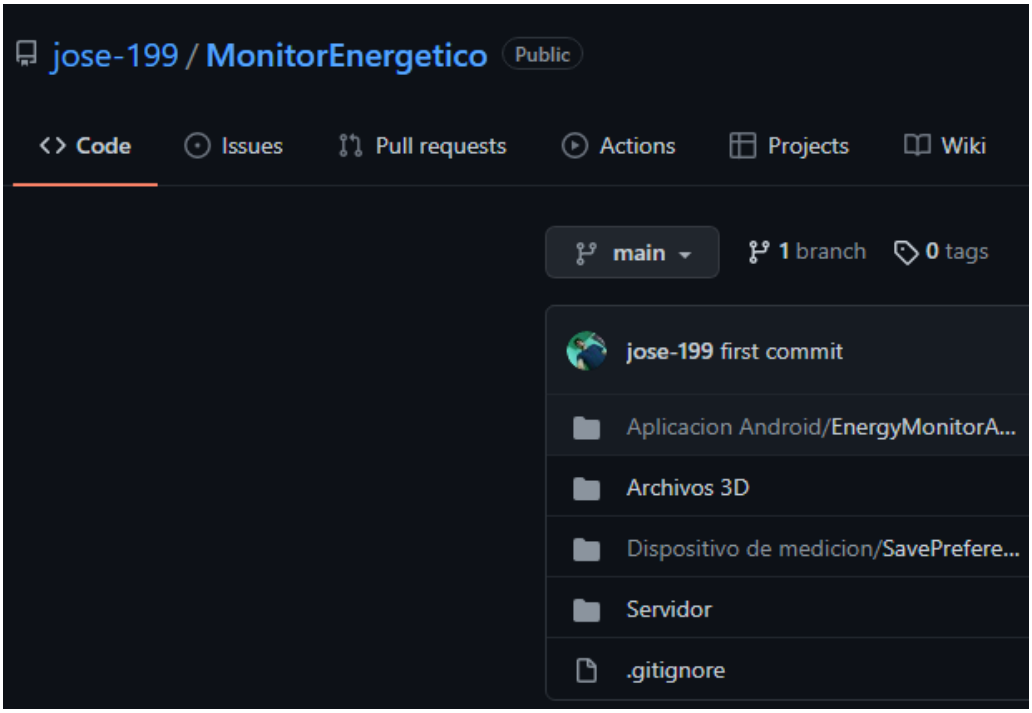
Fuente: elaboración propia, realizado con captura de pantalla.

Apéndice 5. **Visualización energética en tiempo real de un *router***

Fuente: elaboración propia, realizado con captura de pantalla.

ANEXO

Anexo 1. **Monitor energético**



Fuente: José Otzoy. (2022). *Monitor energético.* https://github.com/jose-199/MonitorEnergetico.