PROYECTO FINAL DE INGENIERÍA

SISTEMA DE TELEMETRÍA BASADO EN TECNOLOGÍA IoT, APLICADO A REDES AÉREAS DE MEDIA TENSIÓN

Moreira, Matías – LU 1037692

Ingeniería en telecomunicaciones

Pintos, Agustín – LU 1066131

Ingeniería electromecánica

Tutor:

Abad, Fernando, Universidad Argentina de la Empresa

2019

UNIVERSIDAD ARGENTINA DE LA EMPRESA

FACULTAD DE INGENIERÍA Y CIENCIAS EXACTAS

RESUMEN

El proyecto consiste en el diseño de un sistema de telemetría integral basado en tecnología IoT que permitirá obtener y procesar variables críticas en las redes de distribución de media tensión, con el objetivo de generar alarmas que posibiliten la detección de fallas en tiempo real. Tener visibilidad de dichas fallas, permitirá a las empresas distribuidoras de energía eléctrica no sólo actuar de manera más rápida sobre la solución del problema, sino también tener un registro histórico de las fallas ocurridas y generar medidas de mantenimiento preventivo. Esto impacta de manera directa en la disponibilidad del servicio ofrecido.

El presente documento se encuentra estructurado en ocho capítulos principales. En el primer capítulo se detallan los objetivos y el alcance del proyecto. En el segundo, se da contexto a la situación tecnológica actual y los métodos de detección de fallas existentes. Continuando con el tercer capítulo, en el mismo se definen las herramientas, metodologías y tecnologías utilizadas que dan sustento al desarrollo del sistema propuesto. Posteriormente, se procede con el capítulo cuatro en el cual se describe la construcción del sistema de telemetría en cuestión. En el capítulo cinco, se lleva a cabo el análisis económico para validar la viabilidad del proyecto. Y, por último, en el capítulo seis se exponen las conclusiones a las cuales hemos arribado. En los capítulos siete y ocho se incluyen los anexos y la bibliografía respectivamente.

ABSTRACT

The project consists of designing a telemetry system based on IoT technology which will allow obtaining and processing critical variables in medium voltage distribution networks. The purpose of implementing this telemetry system is to generate alarms that allow fault detection in real time. Having visibility of these faults will allow electricity distribution companies not only to act more quickly on the solution of the problem, but also to have a historical record of the faults that have occurred and generate preventive maintenance measures. This directly impacts the availability of the service offered.

This document is structured in eight main chapters. The first chapter details the objectives and scope of the project. In the second, context is given to the current technological situation and existing fault detection methods. Then in the third chapter, the tools, methodologies and technologies used that support the development of the proposed system are defined. Subsequently, we proceed with chapter four in which the construction of the telemetry system in question is described. In chapter five, the economic analysis is carried out to validate the viability of the project. And, finally, in chapter six the conclusions to which we have arrived are exposed. Chapters seven and eight include the annexes and the bibliography, respectively.

Índice

Capítulo 1: Introducción

1.1 Objetivo

Diseño, desarrollo y evaluación, de un sistema de telemetría basado en tecnología IoT, aplicado a redes aéreas de media tensión.

Se plantean como objetivos secundarios:

- Análisis de la problemática tecnológica actual, estado del arte y relevamiento del estado de situación de la industria frente a la utilización de IoT como solución.

- Diseño y desarrollo del prototipo del sistema de telemetría, teniendo en cuenta tanto el prototipado de una terminal inalámbrica como el de la aplicación web.

- Análisis de viabilidad económica del proyecto.

1.2 Alcance

Se plantea como alcance del proyecto el desarrollo de un sistema de telemetría integral capaz de medir, procesar, transmitir e informar el valor de las variables relevantes en redes aéreas de transmisión de energía en media tensión.

1.3 Componentes básicos del sistema de telemetría

En general, todo sistema de telemetría actual se puede dividir en tres grandes partes, los endpoints o dispositivos finales (encargados de obtener la información), la red de transporte (encargada de establecer la comunicación) y finalmente el sistema informático donde se almacenan, procesan y visualizan las variables. En el caso particular del presente proyecto, se compone de la siguiente manera:

-Terminal inalámbrica

Son dispositivos electrónicos autónomos, basados en microcontroladores, capaces de recolectar y acondicionar los valores obtenidos a través de los distintos periféricos y transmitir dichos datos de manera inalámbrica. Los módulos periféricos, o sensores, que obtendrán los datos son tres: un sensor de corriente diseñado específicamente para dicha tarea, un acelerómetro y un módulo GPS.

-Red de comunicaciones

La red de transporte de datos está basada en la utilización de la tecnología de comunicación inalámbrica, la cual es utilizada específicamente para dispositivos de bajo consumo y largo alcance.

-Aplicación Web:

Está formada por una base de datos en donde se encontrarán almacenados los valores de las variables obtenidas y una página web (que interactúa con dicha base de datos) en donde el usuario podrá visualizar tanto los datos como las alarmas configuradas.

1.4 Beneficios de la implementación del sistema de telemetría

Una vez que el sistema se encuentre en funcionamiento permitirá actualizar procesos que hoy en día se realizan en forma manual, como por ejemplo, encontrar el tramo donde se produjo un corte de energía a lo largo de toda la línea de media tensión. Actualmente, en esos casos, el método utilizado para identificar el tramo afectado consta de encender luces testigo en el lugar del inconveniente, con lo cual, el proceso requiere que una cuadrilla verifique visualmente cada uno de las luces testigos a lo largo de grandes distancias. Este proceso es claramente ineficiente y es un problema de la actualidad que se puede resolver, o al menos mejorar implementando dicho sistema de telemetría.

Capítulo 2: Estado del arte

En la presente sección se detalla la situación actual del mercado eléctrico argentino y las redes aéreas de distribución eléctrica con el fin de dar a conocer el contexto de la problemática hallada a la cual se busca brindar solución.

2.1 Mercado eléctrico argentino

En el Mercado Eléctrico Mayorista (MEM) convergen la oferta y la demanda de energía eléctrica. En el mismo se distinguen los agentes que prestan servicio de aquellos agentes que reciben dicho servicio. Dentro del primer grupo se encuentran los generadores, los transportistas y los distribuidores mientras que el segundo grupo lo integran los grandes, medianos y pequeños usuarios.

- Generadores

Su función es la de producir la energía eléctrica. Su actividad es de interés general y está respaldada por la ley N° 24.065, considerando que la misma constituye una actividad productiva conformada por múltiples operadores los cuales compiten entre sí.

Los generadores inyectan la energía generada en el sistema de Transporte y/o Distribución, y reciben una remuneración por dicha energía. En Argentina, el 63% de las centrales generadoras están accionadas mediante combustibles fósiles, mientras que un 27% de las mismas corresponde a centrales hidroeléctricas y el 10% restante está conformado por centrales atómicas y de energías sustentables.

- Transportistas

Son los encargados de transportar la energía eléctrica desde el punto en el cual es entregada por los generadores hasta los centros de consumo conformados por los Distribuidores y los Grandes Usuarios. En Argentina, el transporte de energía eléctrica es un servicio público reconocido en la ley N° 24.065, conformando un monopolio natural, de forma tal que tanto sus precios como la calidad de servicio se encuentran regulados por el Estado.

Ninguno de los demás agentes del mercado podrá ser propietario de una empresa transportista. Sin embargo, el Poder Ejecutivo puede autorizar a un Generador, Distribuidor o Gran Usuario a construir una red de transporte para cubrir sus propias necesidades.

- Distribuidores

Son los encargados de distribuir la energía eléctrica entre los usuarios finales. Su actividad es reconocida como servicio público y monopolio natural en la ley N° 24.065, de forma tal que, al igual que con los transportistas, sus precios y la calidad de servicio están regulados por el Estado.

Los Distribuidores están obligados a abastecer toda la demanda existente por parte de los usuarios finales, no pudiendo alegar abastecimiento insuficiente.

- Grandes usuarios

Son aquellos que establecen contratos de forma independiente y negocian directamente con el proveedor de servicio el precio de la energía eléctrica suministrada.

Según la energía y la potencia que demanden, los Grandes Usuarios se clasifican en:

- ❖ Grandes usuarios mayores (GUMA)
- ❖ Grandes usuarios menores (GUME)
- ❖ Grandes usuarios particulares (GUPA)

Los Grandes Usuarios pueden firmar tres tipos de contratos: Contratos de Potencia, Contratos de Abastecimiento y Contratos de Energía. CAMMESA debe tener conocimiento de dichos contratos y se encargará de administrarlos.

Dentro de los Grandes Usuarios se incluyen también los autoproductores, los cuales pueden ser:

- Autogeneradores: son aquellos actores cuya actividad principal es la producción de bienes y/o servicios, y, a su vez, producen energía eléctrica como actividad

secundaria. En caso de vender los excedentes de energía, se comportan como generadores y en caso de comprar energía se comportan como un GUMA.

- Cogeneradores: son aquellos que generan energía térmica, como el vapor, para uso industrial y a su vez producen energía eléctrica. Sus excedentes se venden al Mercado Eléctrico comportándose como Generadores.

2.1.1 Comercialización del Mercado Eléctrico Mayorista

Dentro del Mercado Eléctrico Mayorista, existen distintas formas de comercializar la energía de forma tal que se regulan tres tipos de mercados:

- Mercado SPOT: los precios varían de forma horaria dependiendo de la oferta y de la demanda de energía eléctrica que se tenga a cada instante. Es decir, dependen de la demanda de energía por parte de los consumidores finales y de la disponibilidad de generadores existente para satisfacer dicha demanda. El ingreso de los generadores a la red se hace en base a los precios de venta de cada uno de ellos, de forma tal que inyectarán energía a la red aquellos generadores con un precio de venta más económico hasta cubrir la demanda, dejando sin operar aquellos generadores más costosos siempre que la demanda no aumente. La salida de los generadores se realiza en base al mismo criterio, es decir, cuando la demanda disminuye, el generador más costoso es el primero en cesar su operación.

- Mercado ESTACIONAL: se definen dos periodos semestrales los cuales tienen como fecha de inicio el primer día del mes de mayo y del mes de noviembre. Para cada periodo se define un precio para la energía eléctrica el cual dependerá de lo que se espera que cueste la misma durante los seis meses. Los Distribuidores compran la energía al precio fijado y las diferencias respecto al mercado SPOT se tienen en consideración al siguiente período semestral. Los periodos están definidos en función de la generación hidráulica y en cómo afectan los diferentes periodos del año a este tipo de generación.

- Mercado a TÉRMINO: corresponde al mercado integrado por los contratos establecidos entre un Generador y un Distribuidor o un Gran Usuario. A partir del mismo, se determinan condiciones de pago y entrega de la energía y, a su vez, se establecen los plazos dentro de los cuales tiene vigencia dicho contrato y las penalizaciones por incumplimiento de este.

2.1.2 Instituciones del mercado eléctrico

En Argentina, existen numerosas instituciones que forman parte del mercado eléctrico mayorista y se encargan tanto de su administración como de su regulación.

2.1.2.1 Secretaría de energía

Las funciones de la Secretaría de Energía fueron definidas en el Decreto N° 27 del día 27 de mayo del año 2003, dentro de las cuales se destacan: definir los precios estacionales y los ajustes trimestrales, controlar aquellos entes encargados de la administración de los servicios públicos privatizados o concesionados, supervisar la relación entre los diferentes actores del mercado eléctrico, evaluar el impacto ambiental de las distintas etapas involucradas en la generación, transporte y distribución de la energía eléctrica y promover programas de desarrollo de fuentes renovables.

2.1.2.2 Compañía Administradora del Mercado Eléctrico Mayorista S.A

Para administrar de forma idónea el Mercado Eléctrico Mayorista, se constituyó en el año 1992 de acuerdo con lo establecido en el Artículo 35 de la ley N° 24.065 una sociedad anónima sin fines de lucro denominada Compañía Administradora del Mercado Eléctrico Mayorista S.A. (CAMMESA).

Dicha compañía es una empresa privada con fines públicos y tiene como objetivos *"realizar el Despacho Técnico y Económico del Sistema Argentino de Interconexión, maximizar la seguridad del sistema y calidad del suministro, minimizar los precios mayoristas del Mercado Spot, plantear las necesidades de potencia para la satisfacción de la demanda, optimizar la aplicación de las normas que a los efectos del funcionamiento del Mercado Eléctrico dicte la Secretaría de Energía y supervisar el funcionamiento del Mercado a Término y administrar el despacho técnico de los contratos del MEM"* (Fuente: Secretaría de Energía, Presidencia de la Nación).

El 80% de la compañía pertenece en partes iguales por los Generadores, Transportistas, Distribuidores y Grandes usuarios mientras que el 20% restante es propiedad del Estado Nacional, quien preside el directorio de esta y tiene la capacidad de veto en las decisiones.

2.1.2.3 Ente Nacional Regulador de la Electricidad (ENRE)

De acuerdo con lo establecido en el Artículo 54 de la Ley 24.065, se le da origen al Ente Nacional Regulador de la Electricidad. El ENRE es un organismo encargado de regular la actividad eléctrica para cumplir con los objetivos del Estado Nacional y controlar que los Generadores, Transportistas y Distribuidores obedezcan lo establecido en el Marco Regulatorio y en los Contratos de Concesión.

Las funciones principales del ENRE son:

- Garantizar el cumplimiento de los derechos de los usuarios finales.
- Asignar tarifas justas y razonables, promoviendo la competitividad en el sector.
- Promover la inversión privada en la generación, el transporte y la distribución de la energía eléctrica.
- Promover el uso generalizado de las líneas de transporte y distribución, otorgando libre acceso a las mismas.

2.1.2.4 Sistema Argentino de Interconexión (SADI)

El Sistema Argentino de Interconexión es el nombre utilizado para denominar a la red eléctrica Argentina que interconecta los Generadores, Transportistas, Distribuidores y Usuarios transportando la energía eléctrica generada a lo largo y ancho del país.

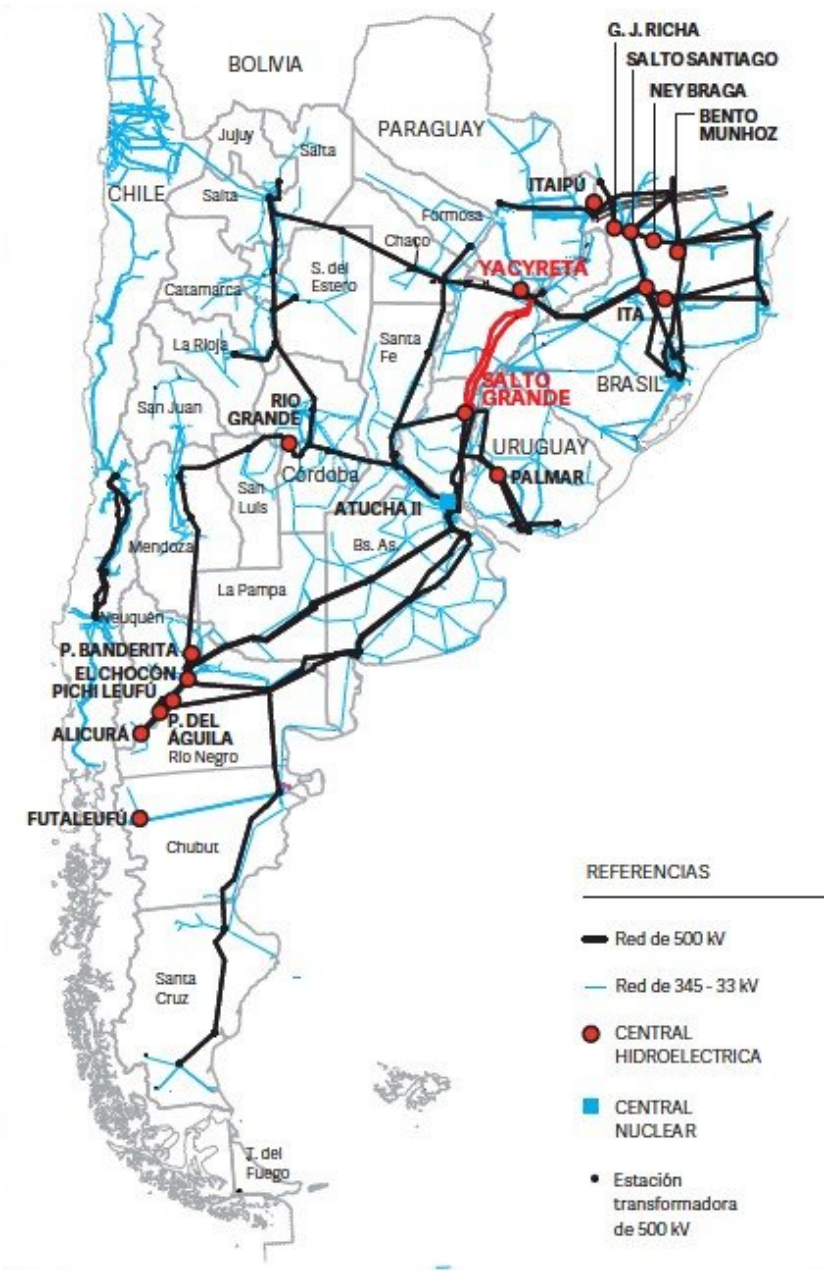

Imagen I: Sistema Argentino de Interconexión

Como se mencionó anteriormente, el Sistema Argentino de Interconexión (SADI) se encuentra administrado y regulado por el Ente Nacional Regulador de la Electricidad (ENRE) y por la Compañía Administradora del Mercado Mayorista Eléctrico Sociedad Anónima (CAMMESA).

El SADI participa en toda la cadena de suministro eléctrico, desde la generación hasta el punto de consumo por los usuarios. Dicha cadena puede ser dividida en cuatro etapas:

- Etapa de generación: es la etapa que involucra a los Generadores y consiste en la producción de la energía eléctrica que se inyecta en la red de transporte. En Argentina, la generación eléctrica ronda los 20 kV, aunque pueden encontrarse ciertas plantas generadoras que producen energía eléctrica alcanzando los 27 kV.
- Etapa de transporte: en esta etapa, se lleva a cabo el transporte de la energía eléctrica desde las plantas generadoras hasta las subestaciones de distribución. En todo conductor a través del cual circula corriente, se producen pérdidas por calor debido a los choques producidos entre los electrones en movimiento y las partículas del material del conductor. Esto se conoce como pérdidas por efecto Joule y, en consecuencia, la energía disponible a la salida del conductor será inferior a la que se inyectó en su entrada. Con el objetivo de reducir dichas pérdidas, el transporte de energía eléctrica en Argentina se realiza en alta tensión lo cual corresponde a aproximadamente 500 kV(aunque puede encontrarse zonas de transporte en 220 kV y 330 kV), reduciendo así los valores de corriente transportada. La tensión es elevada a través de estaciones elevadoras que tienen como entrada la energía producida por las plantas generadoras. Previo a la entrada en las subestaciones de distribución, se rebaja la tensión a 132 kV.
- Etapa de distribución: la misma corresponde al reparto de la energía eléctrica desde las subestaciones de distribución hasta los clientes industriales o residenciales. La distribución se realiza en media tensión, siendo esta reducida a 13,2 kV o 33 kV en las estaciones de distribución.
- Etapa de consumo: esta etapa hace referencia al consumo de la energía eléctrica por parte de los clientes finales. Dependiendo del tipo de usuario, la energía eléctrica puede ser entregada en media tensión o reducida a baja tensión (380/220 V) en centros de transformación.

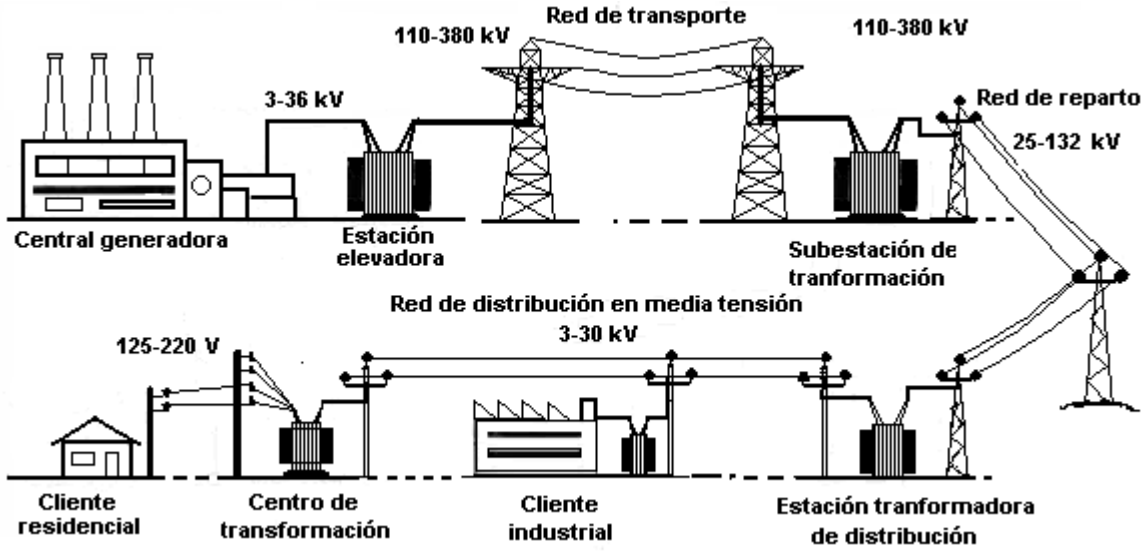

Imagen II: Cadena de suministro eléctrico

2.2 Redes eléctricas de distribución en media tensión

Las líneas eléctricas se encargan del transporte de la energía entre dos puntos siendo uno de los principales elementos dentro de una red eléctrica. Como se mencionó anteriormente, las líneas eléctricas de distribución en media tensión en Argentina operan con valores que van desde 13,2 hasta 33 kV.

Las redes de distribución en media tensión pueden clasificarse en dos categorías principales:

Redes subterráneas: consisten en conductores aislados enterrados y se implementan principalmente en centros urbanos, zonas industriales de alta densidad o en aquellos casos en los cuales las distancias a recorrer sean cortas. Si bien el costo de un sistema subterráneo puede ser entre 5 a 10 veces superior al de un sistema aéreo, la vida útil del mismo es el doble alcanzando los 50 años. A su vez, las redes subterráneas son menos propensas a sufrir averías o accidentes, aunque la localización de las fallas suele ser menos sencilla que en las redes aéreas. En Argentina, se ubica como ejemplo la Ciudad Autónoma de Buenos Aires la cual cuenta en la mayor parte de su territorio con redes de distribución subterráneas.

Redes aéreas: consisten en conductores ubicados entre postes verticales. En Argentina, este sistema se utiliza en centros urbanos, zonas rurales, zonas industriales o en aquellos casos en los cuales las distancias a recorrer sean elevadas. La vida útil estimada de estos sistemas de distribución es de 25 años, sin embargo, los costos de estos son considerablemente inferiores a los costos de los sistemas subterráneos, lo cual constituye uno de los motivos principales de su elección. Sin embargo, al encontrarse expuestos las probabilidades de sufrir averías o fallas son mayores, aunque su localización y reparación suele ser más sencilla.

2.2.1 Topologías de redes aéreas en media tensión

Existen dos grandes grupos en los que se pueden clasificar las redes eléctricas aéreas en media tensión: red radial y red en anillo. Ambos proveen ventajas y cuentan con desventajas que los hacen óptimos para diferentes aplicaciones.

Los sistemas radiales consisten en un centro de carga que alimenta un número determinado de cargas a través de un determinado camino conformado por un único conductor. Cada una de las cargas se encuentra conectada al centro de carga a través de un único cable. Este tipo de sistemas provee un control concentrado desde el centro de alimentación. Es usual en sistemas de distribución domiciliaria y circuitos de iluminación urbanos.

Los sistemas en anillo son más complejos que los radiales, pero brindan mayor seguridad de suministro. Las cargas pueden ser alimentadas desde dos extremos distintos de la red. Durante el servicio, las cargas reciben suministro de un único extremo y, en caso de existir una falla en el camino de uno de los conductores, dicha falla es aislada y las cargas reciben la energía a través del extremo secundario.

También es posible encontrar una topología mixta que surge de una combinación entre redes radiales y redes en anillo.

2.2.2 Fallas en redes de distribución aéreas en media tensión

Se define como falla a cualquier perturbación que impide el correcto funcionamiento del sistema de distribución. Las líneas de distribución aéreas están expuestas a diferentes condiciones que pueden ocasionar fallas, tales como corrosión de los materiales, descargas atmosféricas conocidas como rayos, rotura de los materiales, vientos intensos, entre otras. En los sistemas de distribución aéreos de media tensión pueden existir fallas paralelas o fallas serie.

2.2.2.1 Falla paralela

Consiste en la circulación de corriente entre dos o más fases o entre la(s) fase(s) y tierra. Esto genera un aumento brusco de la intensidad de corriente ya que la misma estará limitada por las impedancias de la fuente, de la línea, de la propia falla y, en caso de falla fase-tierra, por la puesta a tierra del sistema.

Dentro de las fallas paralelas pueden darse: fallas línea - tierra, fallas línea - línea, fallas líneas – línea - tierra o fallas trifásicas.

Fallas línea – tierra

Consiste en el flujo de corriente entre una fase y tierra. Existen tres tipos de fallas línea – tierra, también llamadas monofásicas a tierra. Estos son: Falla fase R a tierra, falla fase S a tierra y falla fase T a tierra.

Fallas línea - línea

Consiste en un cortocircuito entre dos de las tres fases existentes. Por lo que los tres tipos de fallas línea – línea son: falla fase R y fase S, falla fase R y T y falla fase S y T.

Fallas línea – línea – tierra

Son aquellas fallas en las que participan dos de las tres fases existentes en el sistema y tierra. Los tres tipos de fallas que pueden darse son: falla fase R y fase S a tierra, falla fase R y fase T a tierra y falla fase S y fase T a tierra.

Fallas trifásicas

En las fallas trifásicas participan las tres fases R, S y T en forma simultánea. Se denominan fallas simétricas en aquellos casos en los que la resistencia de falla sea igual en las tres fases mencionadas.

2.2.2.2 Falla serie

Las fallas serie son aquellas producidas por la interrupción de una o más fases, generando que las impedancias de las fases difieran entre sí. Si bien no genera flujos de corriente de intensidades elevadas, esto provoca la interrupción del servicio en una fase a los usuarios finales. Dentro de las fallas serie pueden encontrarse: falla de una fase abierta o falla de dos fases abiertas.

Fallas Una Fase Abierta

Son aquellas fallas en las cuales una de las tres fases se interrumpe. Los casos posibles son: fase R abierta, fase S abierta y fase T abierta.

Falla Dos Fases Abiertas

Corresponde a aquellas fallas en las cuales dos de las tres fases existentes en un sistema de distribución de media tensión se interrumpen de forma simultánea. Los casos posibles son: fase R y fase S abiertas, fase R y fase T abiertas y fase S y fase T abiertas.

2.2.2.3 Métodos de localización de fallas

La localización de fallas constituye un aspecto fundamental en los sistemas de distribución para garantizar una determinada calidad del servicio. Existen diferentes métodos de localización de fallas que se pueden clasificar en dos grandes grupos:

- Inspección visual: es el método más rudimentario para la detección de fallas. Se basa en identificar el punto en el cual se produjo la falla recorriendo la línea. Puede ser un recorrido de línea completa o seccionar la misma. Es un método lento utilizado en los inicios de la industria eléctrica.
- Por medición desde una referencia fija: son algoritmos que analizan una determinada información de la línea teniendo en cuenta una referencia fija con valores conocidos y permiten delimitar la zona en la cual se produjo la falla, pero no identificar la posición exacta de la misma. Dentro de este grupo se incluyen: el método basado en el cálculo de impedancia utilizado por su simplicidad y bajo costo; y el método basado en onda viajera que se centra en medir el tiempo que toma un frente de onda en propagarse desde el punto de medición en el extremo de la línea hasta el punto de discontinuidad provocado por la falla.

  Frente a los métodos actuales de localización de fallas en redes aéreas de media tensión, se propone la elaboración de un dispositivo que no solo permita detectar la existencia de una falla y generar una alerta, sino también que otorgue la ubicación exacta de la misma con el objetivo de reducir los tiempos de localización de estas.

Capítulo 3: Definición de herramientas, metodologías y tecnologías

3.1 Herramientas de desarrollo

Una vez definida la solución tecnológica final en base a la investigación previamente descrita, se procede a definir los entornos y herramientas de desarrollo apropiados para cada etapa del proyecto.

En este sentido, podemos dividir el proyecto en tres partes: Firmware, Hardware y Software.

3.1.1 Hardware

3.1.1.1 Entorno de desarrollo integrado (IDE)

Se decidió utilizar la herramienta MPLAB X como IDE (Integrated Development Enviroment), esta elección se desprende de la marca utilizada en el microcontrolador (Microchip). MPLAB X ofrece una interfaz sencilla para el usuario, consta de un editor modular que permite seleccionar los distintos microcontroladores soportados, además de permitir la grabación de estos circuitos integrados directamente, a través del programador.

Microchip ofrece, en forma gratuita, la posibilidad de utilizar esta herramienta tanto en Windows, como en sistemas operativos basados en UNIX o LINUX, lo cual lo hace un entorno mucho más flexible que se ajusta a las necesidades del desarrollador.

Otra ventaja importante del entorno es que soporta el lenguaje “C”, el cual es un lenguaje de bajo nivel, flexible y potente. El lenguaje C reemplaza al lenguaje Assembler en casi todos los desarrollos de sistemas embebidos actuales, debido básicamente a que ofrece una performance similar, pero con mayor facilidad de interpretación y entendimiento del código, lo cual acelera los tiempos de desarrollo.

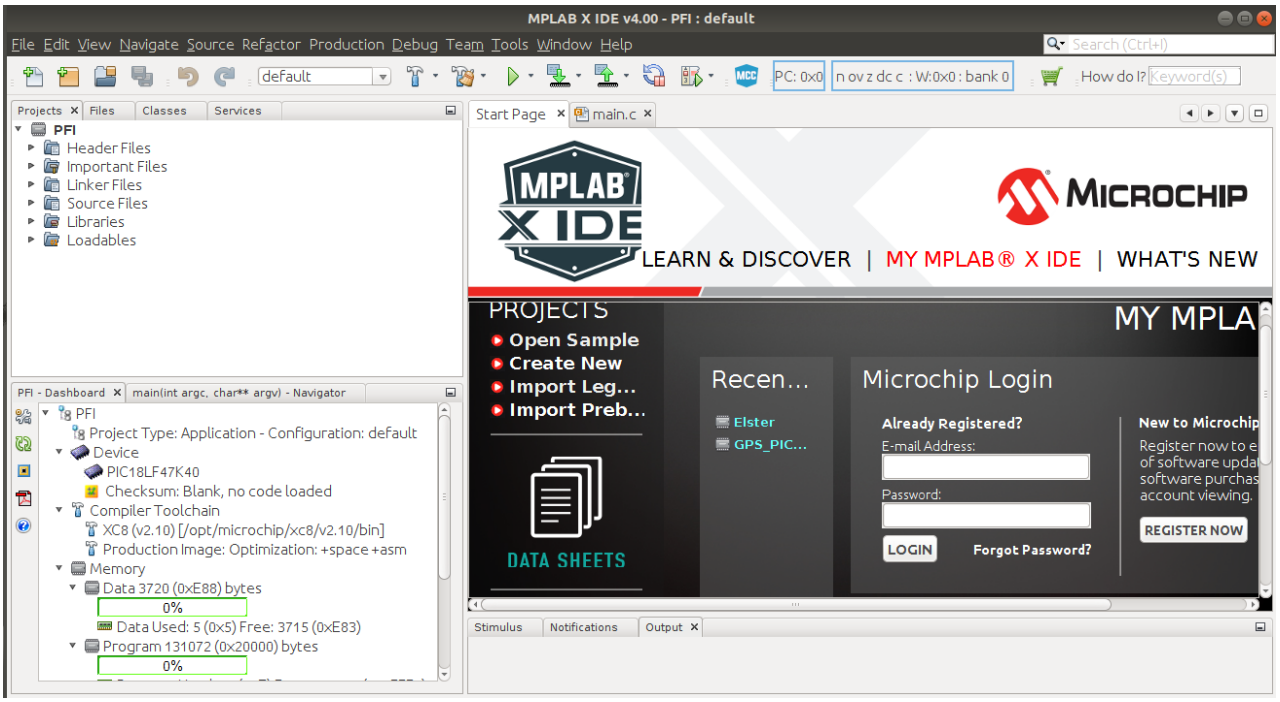

Imagen III: Pantalla principal del entorno de desarrollo integrado MPLAB X.

Dentro de MPLAB X, podemos encontrar un plug-in muy potente de desarrollo conocido como MCC (Microchip Code Configurator). Esta herramienta permite generar el código en lenguaje C correspondiente a la configuración de bits realizada dentro de la pestaña MCC. Su principal ventaja es que admite una configuración simple y visual tanto de cosas sencillas como definir un pin como entrada hasta incluso módulos tan complejos como el ADC, con tan solo algunos clicks.

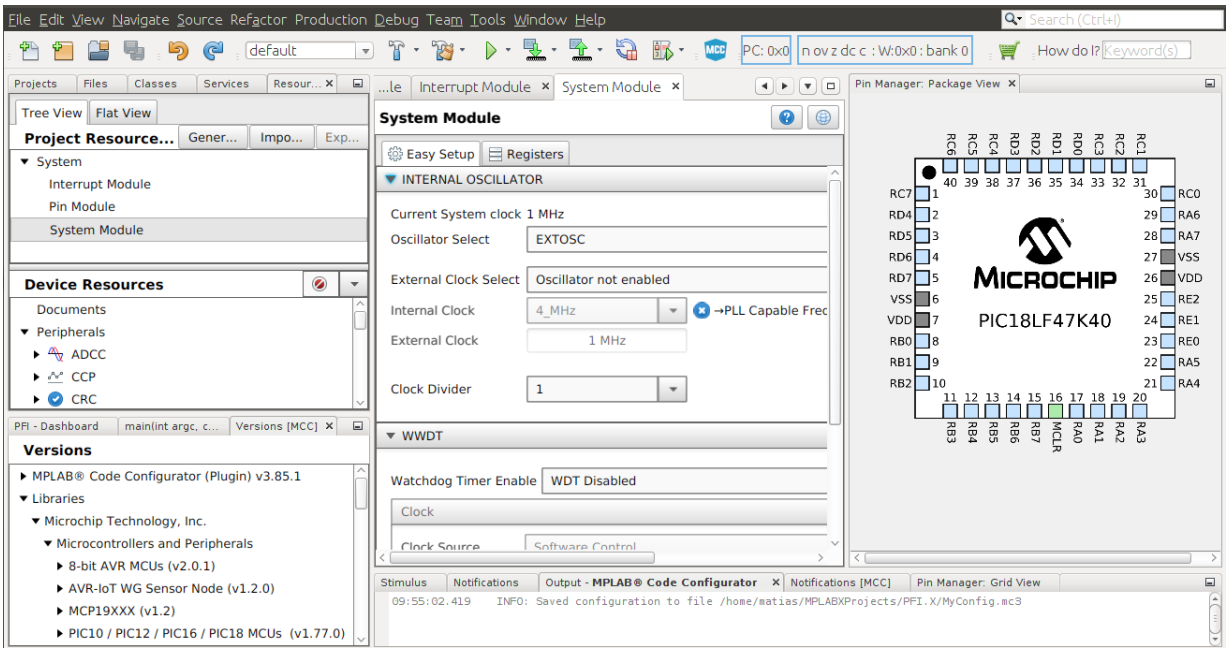

Imagen IV: Pantalla principal del módulo plug-in MCC.

Cabe aclarar que el código generado por MCC, es simplemente un esqueleto del cual se puede partir para luego ajustarlo según la necesidad en cuestión, pero es de gran utilidad para acortar los tiempos de desarrollo.

3.1.1.2 Programador

El programador de microcontroladores elegido es el ICD3 o "In Circuit Debugger 3", el mismo es propio de la marca Microchip y es el recomendado para programar microcontroladores PIC, debido a que ofrece la capacidad de grabar el programa en el microcontrolador, y además permite realizar procesos de DEBUG sobre el circuito en cuestión en tiempo real, lo que permite agilizar aún más los tiempos de desarrollo.

3.1.1.3 Simuladores y software de diseño PCB

Antes de llevar a cabo un circuito en forma física, es esencial realizar simulaciones previas en programas apropiados para lograr anticipar resultados y acotar el margen de error. Luego, una

vez las simulaciones nos otorguen los resultados esperados, se procede a materializar el circuito.

En última instancia, una vez el circuito funciona acorde a lo esperado en forma física, se procede a diseñar un PCB o placa de circuito impreso final.

Las herramientas utilizadas para dicho proceso, se pueden dividir en dos partes, herramientas de simulación y herramientas de diseño de Circuitos PCB.

3.1.1.3.1 Simulador de circuitos electrónicos

Como herramienta de simulación de circuitos electrónicos se utiliza el programa Multisim, desarrollado por la empresa National Instruments, ya que cuenta con una interfaz amigable, simple de usar y con resultados lo suficientemente precisos.

3.1.1.3.2 Software de diseño PCB

Como software de diseño de circuitos impresos, se optó por Altium Designer. Siendo uno de los software de diseño PCB más utilizados en el mercado.

3.1.2 Software

3.1.2.1 Entorno de desarrollo integrado (IDE)

Se utiliza Visual Studio Code como entorno de desarrollo integrado ya que es una herramienta de uso gratuito que ofrece soporte para la depuración, refactorización de código e integración con GIT, sistema de control de versiones, entre otros beneficios para el desarrollador.

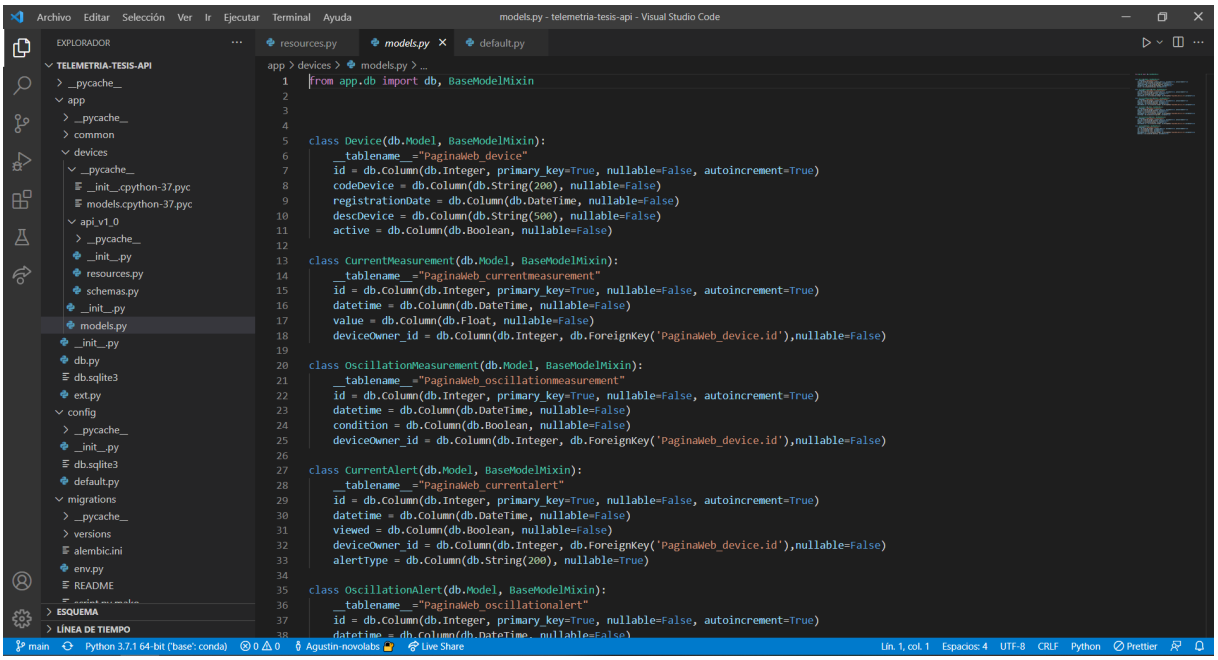

Imagen V: Visual Studio Code IDE

Microsoft ofrece la posibilidad de utilizar esta herramienta tanto en Windows, como en sistemas operativos basados en UNIX o LINUX y macOS, lo cual lo hace un entorno mucho más flexible que se ajusta a las necesidades del desarrollador.

A su vez, Visual Studio Code tiene la posibilidad de instalar diversas extensiones que brindan soporte al desarrollador facilitando y acelerando el desarrollo.

3.2 Tecnologías de hardware

En este apartado, se definen las metodologías y tecnologías a utilizar en lo que respecta a los puntos críticos del desarrollo de la terminal inalámbrica brindando a su vez un marco teórico introductorio en cada uno de ellos:

- Tecnología de comunicación
- Alimentación y almacenamiento de energía
- Adquisición del valor de corriente en un conductor

3.2.1 Tecnología de comunicación

En esta sección, se analizarán las diferentes alternativas existentes en el mercado respecto a las tecnologías de comunicación aplicables al proyecto, teniendo en cuenta sus topologías, ventajas y desventajas. El objetivo es demostrar por qué LoRaWAN es la que mejor se ajusta, según nuestro criterio, a las necesidades del proyecto. Dichas necesidades se pueden resumir en largo alcance, bajo consumo y bajo costo.

Teniendo en cuenta estas tres características, es evidente que estamos hablando de la utilización de tecnologías LPWAN (Low Power Wide Area Networks).

Luego de realizar la investigación correspondiente, se muestra una lista de posibles tecnologías a utilizar, las cuales se describen a continuación:

- Tecnología Celular NB-IOT, CAT-M
- LoRaWAN
- SigFox

3.2.1.1 Tecnología Celular: NB-IoT, CAT-M

La utilización de tecnología celular es siempre una opción a evaluar cuando hablamos de comunicación inalámbrica, debido en parte a su gran masividad. Ahora bien, esta no es del todo aplicable a este proyecto en sus versiones más conocidas como pueden ser 2G, 3G y 4G, debido básicamente al gran consumo de los endpoints en este tipo de soluciones. Esto contradice una de las características esenciales del proyecto, que es el bajo consumo.

Si bien a medida que las generaciones fueron avanzando los consumos son cada vez menores (se debe tener en cuenta que 2G tiene picos de corriente de hasta 2A), siguen siendo excesivos

en nuestro caso en particular. Para resolver este inconveniente y expandir el uso de estas tecnologías fue que se desarrollaron, entre otras, LTE-M y NB-IoT.

En este caso, nos centraremos en Narrow Band IoT (NB-IoT) debido a que es, de las tecnologías celulares, la que más se ajusta a las necesidades del proyecto.

3.2.1.2 NB-IoT

NB-IoT es una tecnología de IoT de banda estrecha especificada en la versión 13 de 3GPP en junio de 2016. NB-IoT puede coexistir con GSM (sistema global para comunicaciones móviles) y LTE (evolución a largo plazo) en bandas de frecuencia con licencia (por ejemplo, 700MHz, 800 MHz y 900 MHz).

NB-IoT ocupa un ancho de banda de frecuencia de 200 KHz, que corresponde a un bloque de recursos en la transmisión GSM y LTE. Con esta selección de banda de frecuencia, los siguientes modos de operación son posibles:

Operación autónoma: un escenario posible es la utilización de las bandas de frecuencias GSM actualmente utilizadas.

Operación de la banda de guarda: utilizando los bloques de recursos no utilizados dentro de la banda de guarda de una portadora LTE.

Operación en banda: utilizando bloques de recursos dentro de una portadora LTE.

NB-IoT puede ser compatible con solo una actualización de software además de la infraestructura LTE existente. El protocolo de comunicación NB-IoT se basa en el protocolo LTE. De hecho, NB-IoT reduce las funcionalidades del protocolo LTE al mínimo y las mejora según sea necesario para las aplicaciones de IoT.

NB-IoT se optimizó para mensajes de datos pequeños y poco frecuentes y evita las funciones que no se requieren para el propósito de IoT, por ejemplo, mediciones para monitorear la

calidad del canal, la incorporación de portadoras y la conectividad dual. Por lo tanto, los dispositivos finales requieren solo una pequeña cantidad de batería, lo que los hace rentables. En consecuencia, la tecnología NB-IoT puede considerarse como una nueva interfaz aérea desde el punto de vista de la pila de protocolos, mientras se construye sobre la infraestructura LTE bien establecida. NB-IoT permite la conectividad de hasta 100000 dispositivos finales por celda con el potencial de aumentar la capacidad agregando más operadores NB-IoT. NB-IoT utiliza el acceso múltiple por división de frecuencia de portadora única (FDMA) en el enlace ascendente y el FDMA ortogonal (OFDMA) en el enlace descendente, y emplea la modulación por desplazamiento de fase en cuadratura (QPSK).

La velocidad de datos está limitada a 200 kbps para el enlace descendente y a 20 kbps para el enlace ascendente. El tamaño máximo de carga útil para cada mensaje es de 1600 bytes. La tecnología NB-IoT puede alcanzar 10 años de vida útil de la batería al transmitir 200 bytes por día (en promedio).

3.2.1.3 Sigfox

Sigfox es un operador de red LPWAN que ofrece una solución de conectividad de IoT de extremo a extremo basada en sus tecnologías patentadas. Sigfox despliega sus estaciones base patentadas equipadas con radios cognitivas definidas por software y las conecta a los servidores de back-end mediante una red basada en IP. Los endpoints conectados a estas estaciones base utilizan modulación por desplazamiento de fase binaria (BPSK) en una portadora de banda ISM sub-GHZ de banda ultra estrecha (100 Hz). Sigfox utiliza bandas ISM sin licencia, por ejemplo, 868 MHz en Europa, 915 MHz en América del Norte y 433 MHz en Asia. Al emplear la banda ultra estrecha, Sigfox usa el ancho de banda de frecuencia de manera eficiente y experimenta niveles de ruido muy bajos, lo que lleva a un consumo de energía muy bajo, una alta sensibilidad del receptor y un diseño de antena de bajo costo a expensas de un rendimiento máximo de solo 100 bps. Sigfox inicialmente sólo admitía la comunicación de enlace ascendente, pero luego evolucionó a tecnología bidireccional con una

asimetría de enlace significativa. La comunicación de enlace descendente, es decir, los datos de las estaciones base a los endpoints solo puede ocurrir después de una comunicación de enlace ascendente. El número de mensajes a través del enlace ascendente está limitado a 140 mensajes por día. La longitud máxima de carga útil para cada mensaje de enlace ascendente es de 12 bytes. Sin embargo, el número de mensajes a través del enlace descendente está limitado a cuatro mensajes por día, lo que significa que no se admite el reconocimiento de cada mensaje del enlace ascendente. La longitud máxima de carga útil para cada mensaje del enlace descendente es de ocho bytes. Sin el soporte adecuado de acuses de recibo, la fiabilidad de la comunicación de enlace ascendente se garantiza mediante la diversidad de tiempo y frecuencia, así como la duplicación de transmisión. Cada mensaje de los endpoints se transmite varias veces (tres de forma predeterminada) a través de diferentes canales de frecuencia. Para ello, en Europa, por ejemplo, la banda entre 868,180 MHz y 868,220 MHz se divide en 400 canales ortogonales de 100 Hz (entre ellos 40 canales están reservados y no se utilizan). Como las estaciones base pueden recibir mensajes simultáneamente en todos los canales, el dispositivo final puede elegir aleatoriamente un canal de frecuencia para transmitir sus mensajes. Esto simplifica el diseño del dispositivo final y reduce su costo.

3.2.1.4 LoRa

LoRa es una tecnología de capa física que modula las señales en la banda ISM sub-GHZ utilizando una técnica de espectro ensanchado patentada. Al igual que Sigfox, LoRa utiliza bandas ISM sin licencia, es decir, 868 MHz en Europa, 915 MHz en América del Norte y 433 MHz en Asia. La comunicación bidireccional se proporciona mediante la modulación de chirp de espectro ensanchado (CSS) que difunde una señal de banda estrecha sobre un ancho de banda de canal más amplio. La señal resultante tiene bajos niveles de ruido, lo que permite una alta resistencia a las interferencias, y es difícil de detectar o interferir. LoRa utiliza seis factores de dispersión (SF7 a SF12) para adaptar la tasa de datos y el rango de compensación. Un factor de expansión más alto permite un rango más largo a expensas de una velocidad de

datos más baja, y viceversa. La velocidad de datos de LoRa está entre 300 bps y 50 kbps, dependiendo del factor de expansión y el ancho de banda del canal. Además, los mensajes transmitidos usando diferentes factores de propagación pueden ser recibidos simultáneamente por las estaciones base. La máxima longitud de la carga útil para cada mensaje es de 243 bytes.

LoRa-Alliance estandarizó un protocolo de comunicación basado en LoRa llamado LoRaWAN (primera versión en 2015). Usando LoRaWAN, cada mensaje transmitido por un dispositivo final es recibido por todas las estaciones base en el rango. Al explotar esta recepción redundante, LoRaWAN mejora la proporción de mensajes recibidos con éxito. Sin embargo, lograr esta característica requiere varias estaciones base en el vecindario, lo que puede aumentar el costo de implementación de la red. Las recepciones duplicadas resultantes se filtran en el sistema backend (servidor de red) que también tiene la inteligencia necesaria para verificar la seguridad, enviar acuses de recibo al dispositivo final y enviar el mensaje al servidor de aplicaciones correspondiente. Además, LoRaWAN explota múltiples recepciones del mismo mensaje por diferentes estaciones base para localizar dispositivos finales. Para este propósito, se utiliza la técnica de localización basada en la diferencia de tiempo de llegada (TDOA) apoyada por una sincronización de tiempo muy precisa entre múltiples estaciones base. Además, múltiples recepciones del mismo mensaje en diferentes estaciones base evitan el traspaso en la red LoRaWAN (es decir, si un nodo es móvil o en movimiento, no se necesita traspaso entre las estaciones base). Además, LoRaWAN proporciona varias clases de dispositivos finales para abordar los diferentes requisitos de una amplia gama de aplicaciones de IoT, por ejemplo, requisitos de latencia.

Las diferentes clases de dispositivos existentes son:

- Dispositivos finales bidireccionales (clase A): dispositivos finales de clase A permiten comunicaciones bidireccionales donde la transmisión de enlace ascendente de cada dispositivo final es seguida por dos ventanas de recepción de enlace descendente corto. El lote de transmisiones programado por el dispositivo final se basa por sus propias necesidades de comunicación con una pequeña variación basada en un tiempo

aleatorio. Esta operación de clase A es el sistema de dispositivo final de menor potencia para aplicaciones que solo requieren una comunicación de enlace descendente breve después de que el dispositivo final haya enviado un mensaje de enlace ascendente. Las comunicaciones de enlace descendente en cualquier otro momento tendrán que esperar hasta el siguiente mensaje de enlace ascendente del dispositivo final.

- Dispositivos finales bidireccionales con lotes de recepción programados (clase B): además de las ventanas de recepción aleatoria de la clase A, los dispositivos de clase B abren ventanas de recepción adicionales en las horas programadas . Para abrir las ventanas de recepción a la hora programada, los dispositivos finales reciben una baliza sincronizada en el tiempo desde la estación base. Esto permite que el servidor de red sepa cuándo está escuchando el dispositivo final.

- Dispositivos finales bidireccionales con ranuras de recepción máximas (clase C): los dispositivos finales de clase C tienen ventanas de recepción abiertas casi continuamente y solo se cierran cuando transmiten a expensas de un consumo excesivo de energía.

LoRaWAN cuenta con dos mecanismos de activación de dispositivos finales posibles, ABP (Activación por personalización) u OTAA (Activación por aire).

Para entender la diferencia, primero se deben tener en cuenta algunas definiciones que se detallan a continuación.

El DevEUI es un ID único de 64 bits asignado a un dispositivo final por el fabricante. Este valor está vinculado al hardware y no se puede modificar.

A diferencia de DevEUI, que identifica un dispositivo final globalmente, el DevAddr de 32 bits identifica el dispositivo final dentro de la red actual y toda la comunicación después de unirse a la red se realiza con él. Un valor de DevAddr consta de NwkAddr (dirección del

dispositivo final dentro de la red) con el prefijo NwkID (identificador de red). Cabe destacar que el DevAddr puede no ser único: varios dispositivos pueden tener el mismo DevAddr.

El DevAddr y las claves de sesión se asignan a un dispositivo final durante un procedimiento llamado activación. Como se mencionó anteriormente, LoRaWAN admite dos modos de activación de un dispositivo final: ABP (activación por personalización) y OTAA (activación por aire).

- ABP: En la activación ABP, el DevAddr es fijo y las claves de sesión para una red preseleccionada están codificadas en el dispositivo final y permanecen iguales durante toda la vida útil de un dispositivo final. Al utilizar este modo, un dispositivo final omite el procedimiento de unión.

- OTAA: Los dispositivos finales OTAA cuentan con claves "raíz". En la activación de OTAA, un dispositivo final realiza un procedimiento de unión con una red LoRaWAN, durante el cual se asigna un DevAddr dinámico a un dispositivo final y se utilizan claves raíz para derivar claves de sesión. Por lo tanto, DevAddr y las claves de sesión cambian a medida que se establece cada nueva sesión.

3.2.1.5 Comparación entre tecnologías de comunicación

- Duración de la batería: En Sigfox, LoRa y NB-IoT, los dispositivos finales están en modo de suspensión la mayor parte del tiempo fuera de la operación, lo que reduce la cantidad de energía consumida, es decir, una vida útil prolongada de los dispositivos finales. Sin embargo, el dispositivo final NB-IoT consume energía adicional debido a la comunicación síncrona y el manejo de QoS, y sus modos de acceso OFDM / FDMA requieren más corriente pico. Este consumo de energía adicional reduce la vida útil del dispositivo final NB-IoT en comparación con Sigfox y LoRa Sin embargo, NB-IoT ofrece la ventaja de una baja latencia. A diferencia de Sigfox, LoRa proporciona la clase C para manejar también una baja latencia bidireccional a expensas de un mayor consumo de energía. Por lo tanto, para aplicaciones que son insensibles a la latencia y no tienen una gran cantidad de datos para enviar, Sigfox y Class-A LoRa son las mejores opciones. Para aplicaciones que requieren baja latencia, NB-IoT y clase C LoRa son las mejores opciones.

- Cobertura y alcance de la red: La principal ventaja de uso de Sigfox es que una sola estación base puede cubrir una ciudad entera (es decir, un alcance> 40 km). En Bélgica, un país con una superficie total de aproximadamente 30 500 km2, el despliegue de la red Sigfox cubre todo el país con solo siete estaciones base. Por el contrario, LoRa tiene un alcance menor (es decir, un alcance menor a 20 km) que solo requiere tres estaciones base para cubrir toda una ciudad como Barcelona. NB-IoT tiene las capacidades de alcance y cobertura más bajas (es decir, alcance <10 km). Se centra principalmente en la clase de dispositivos que se instalan en lugares alejados del alcance típico de las redes celulares (por ejemplo, en interiores, en interiores profundos). Además, la implementación de NB-IoT se limita a las estaciones base LTE. Por lo tanto, no es adecuado para regiones rurales o suburbanas que no se benefician de la cobertura LTE.

- **Costos:** Se deben considerar varios aspectos de costo, como el costo del espectro (licencia), el costo de red / implementación y el costo del dispositivo. La siguiente tabla muestra el costo de Sigfox, LoRa y NB-IoT:

	Costo del espectro	**Costo de despliegue**	**Costo del dispositivo**
Sigfox	Gratuito	>4000 euros / estación base	<2 euros
LoRa	Gratuito	>100 euros / Gateway > 1000 euros / estación base	<3 euros
NB-IoT	Pago	>15000 euros / estación base	>20 euros

Tabla I: Comparación entre costos de las tecnologías de comunicación

Es evidente que Sigfox y LoRa son más rentables en comparación con NB-IoT.

En resumen, Sigfox, LoRa y NB-IoT tienen sus respectivas ventajas en términos de diferentes factores de IoT, como se muestra en el siguiente gráfico:

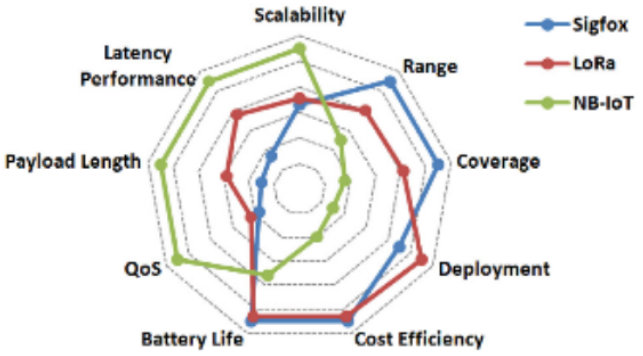

Imagen VI: Comparación entre las tecnologías de comunicación

A continuación se muestra una tabla comparativa a modo de resumen de las tres tecnologías en cuestión:

	Sigfox	**LoraWAN**	**NB-IoT**
Modulación	BPSK	CSS	QPSK
Frecuencia	Bandas no licenciadas ISM (868 MHz, 915 MHz y 433 MHz)	Bandas no licenciadas ISM (868 MHz, 915 MHz y 433 MHz)	Frecuencias LTE Licenciadas
Ancho de banda	100 Hz	250 Khz y 125 Khz	200 Khz
Max. Velocidad de transmisión	100 bps	50 Kbps	200 Kbps
Bidireccional	Limitado / Half-Duplex	Si / Half-Duplex	Si / Hlaf-Duplex
Cantidad de mensajes por día	140	Sin Límite	Sin límite

Longitud Max. De carga útil	12 Bytes	243 Bytes	1600 Bytes
Alcance	10 Km (Urbano), 40Km (rural)	5 Km (Urbano), 20 km (rural)	1 Km (Urbano), 10 Km (rural)
Inmunidad al ruido	Muy alta	Muy alta	baja
Autenticación / encriptación	No soportado	Si (AES 128)	Si (Encriptación LTE)
Velocidad de transmisión adaptativa	No soportado	Si	No
Localización	Si (RSSI)	Si (TDOA)	No
Estándar	Sigfox company / ETSI	LoRa-Alliance	3GPP

Tabla II: Comparación entre las tecnologías de comunicación

3.2.1.6 Conclusión

En base a lo detallado anteriormente, la tecnología de comunicación que utilizaremos para este proyecto es LoRaWAN.

La elección se basa principalmente en el buen alcance, baja potencia y la relación costo-beneficio que la tecnología ofrece.

3.2.2 Alimentación y almacenamiento de energía

En la presente sección se detallan los fundamentos físicos en los cuales se basa el mecanismo utilizado para generación y almacenamiento de energía eléctrica implementado en el dispositivo.

3.2.2.1 Energía solar

La energía solar forma parte de las energías renovables y es aprovechada mediante la captación de la radiación electromagnética producida por el sol. El sol genera energía a partir de reacciones nucleares de fusión producidas en su núcleo, de forma tal que la masa perdida se convierte en energía conforme a la ecuación de Einstein:

$$E = mc$$

Donde:

- m es la masa perdida.
- c es la velocidad de la luz en el vacío.

3.2.2.1.1 Energía solar fotovoltaica

Se han desarrollado numerosas tecnologías que permiten captar la energía solar y, a partir de ella, generar energía útil. Dentro de ellas se destaca la tecnología solar fotovoltaica, la cual se basa en transformar la radiación electromagnética del sol en energía eléctrica a través de las células fotovoltaicas.

Un sistema fotovoltaico se compone de:

- Generadores fotovoltaicos.
- Acumuladores de energía.
- Regulador de carga.

Generadores fotovoltaicos

Los generadores fotovoltaicos están conformados por un conjunto de células fotovoltaicas conectadas en serie, denominados paneles fotovoltaicos, con el objetivo de que la suma de tensiones generadas por cada una de ellas alcance el valor de tensión deseado para el sistema.

Las células fotovoltaicas son dispositivos semiconductores que tienen la capacidad de aumentar la densidad de electrones libres al recibir estímulos externos. La red cristalina del material semiconductor está dotada de impurezas aceptoras y donadoras. Las primeras, denominadas impurezas tipo p, son átomos que presentan en su última capa un electrón menos que los átomos que constituyen la red originando la aparición de posiciones vacías en la red, denominadas huecos. Mientras que las pertenecientes al segundo grupo se conocen como impurezas tipo n y disponen de un electrón extra en su última capa, de forma tal que cuentan con la capacidad de perder ese electrón fácilmente aumentando el número de electrones libres en el material. Dichas impurezas, separadas en dos zonas, dan origen a una unión *pn* la cual genera un campo eléctrico en el seno de la célula fotovoltaica con dirección de la región n hacia la región p, alejando los electrones libres de los huecos.

La energía de un fotón incidente sobre el material semiconductor es absorbida por los electrones ubicados en la banda de valencia lo que les otorga el gap de energía necesario para romper el enlace con el átomo al que pertenecen y pasar a la banda de conducción, quedando disponibles para desplazarse a través del material y dejando un hueco en el enlace en el que se encontraban, el cual se comporta como carga positiva. Los huecos ubicados en la región p son dirigidos hacia el extremo donde se encuentra el metal que forma parte del contacto del semiconductor con el circuito externo y extrae un electrón perteneciente al contacto. Por su parte, los electrones libres en la región n son dirigidos hacia la zona del contacto y cedidos al metal. Esto permite la generación de un flujo de corriente a través del circuito externo.

En la actualidad, los módulos fotovoltaicos tienen un rendimiento máximo aproximado del 25%, es decir, pueden aprovechar únicamente una cuarta parte de la energía solar que perciben. Los valores de tensión y, en mayor medida, los de corriente entregados dependen de la disponibilidad de luz solar por lo que es primordial disponer de un sistema de

almacenamiento de energía. Esto puede realizarse a través de baterías, tales como baterías abiertas de plomo ácido, o acumuladores de carga, como supercapacitores.

Acumuladores de energía

Los acumuladores le garantizan a la carga un suministro estable de energía, independientemente de la energía solar disponible en el momento del consumo. De esta forma, se satisface la demanda energética de la carga de forma prolongada en el tiempo, siempre que el acumulador conserve la carga suficiente. El tipo de acumulador a utilizar dependerá de la aplicación y de las características de carga y descarga durante su operación.

Los acumuladores de uso fotovoltaico deben contar con la capacidad para soportar numerosos ciclos de carga, de forma tal que es común el uso de baterías de plomo ácido, baterías electrolíticas, entre otras. Sin embargo, una alternativa viable en instalaciones fotovoltaicas es el uso de supercapacitores. Un supercapacitor es un condensador electroquímico con la capacidad de suministrar una densidad energética mucho mayor en comparación a los condensadores normales debido a que su capacitancia es miles de veces superior. Las ventajas de los supercapacitores respecto de las baterías son las siguientes:

- Ciclos de carga: Los supercapacitores soportan miles de ciclos sin reducir su rendimiento, mientras que las baterías ven reducido su rendimiento a un número menor de ciclos. De forma tal que, para prolongar la vida útil de un sistema fotovoltaico, los capacitores son la opción adecuada.
- Tiempos de carga y descarga: los tiempos de carga y descarga de las baterías dependen de reacciones químicas y, por lo tanto, suelen ser notablemente mayores a los de los supercapacitores.
- Características físicas: para una misma aplicación, las baterías son de mayor tamaño y peso que los supercapacitores.

Reguladores de carga

Los reguladores de carga son dispositivos electrónicos conectados entre los módulos fotovoltaicos y los acumuladores de energía, y controlan el estado de la carga de estos últimos. Las funciones principales de un regulador de carga son las siguientes:

- Limita la descarga de los acumuladores, evitando que sobrepase un valor mínimo de carga determinado.
- Regula el proceso de carga, evitando que se produzcan sobrecargas.
- Control de la carga existente en los acumuladores.

3.2.2.2 Conclusión

Dado que los dispositivos constituyen endpoints ubicados a la intemperie sin acceso a una fuente de alimentación tradicional, se decide utilizar la energía solar fotovoltaica como fuente de energía para su alimentación eléctrica.

3.2.3 Adquisición del valor de corriente en un conductor

En la presente sección se detallan los fundamentos físicos en los cuales se basa el mecanismo de lectura de corriente en un conductor implementado en el dispositivo.

3.2.3.1 Ley de Biot-Savart

La ley de Biot-Savart establece que la corriente que circule por un conductor que forma parte de un circuito cerrado producirá un campo magnético cuya magnitud en un punto P del espacio dependerá de la intensidad de corriente que lo origino, la distancia radial al conductor y la permeabilidad magnética del medio en el cual se genera dicho campo magnético.

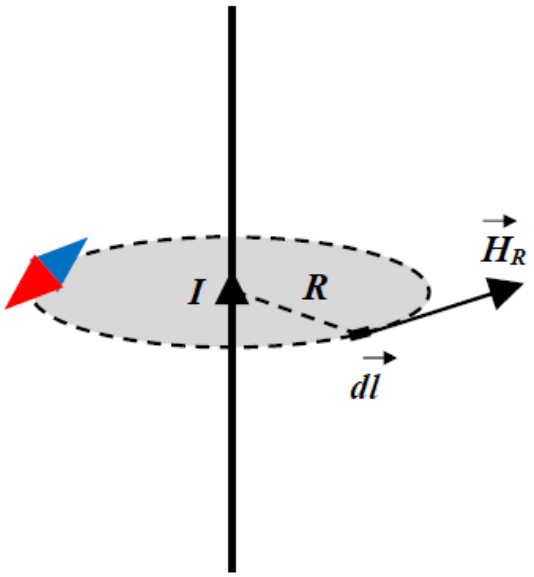

Imagen XV:campo magnético generado, Ley de Biot-Savart

La expresión que define esta ley es:

$$d\vec{B} = \frac{\mu}{4\pi} i \frac{\vec{dl} \times \vec{r}}{r^3}$$

Donde:

- $d\vec{B}$ es un diferencial de campo magnético generado en el punto P.
- $\boldsymbol{\mu}$ es la permeabilidad magnética del medio.
- i es la corriente que circula por el conductor.
- r es la distancia radial del conductor al punto en el cual se evalúa el campo magnético.
- $\vec{dl}$ es un diferencial de longitud del conductor en la dirección y sentido de la corriente.
- $\vec{r}$ es el vector posición que va del conductor al punto P.

Considerando un conductor rectilíneo donde el ángulo existente entre el vector $\vec{dl}$ y $\vec{r}$ es un ángulo recto, se calcula el módulo del campo magnético en el punto P como:

$$B = \frac{\mu}{2\pi r} i$$

3.2.3.2 Ley de inducción electromagnética

La ley de inducción electromagnética de Faraday establece que la tensión que se induce en un circuito cerrado es proporcional a la variación en el tiempo del flujo magnético que atraviesa una superficie cuyo borde es el circuito.

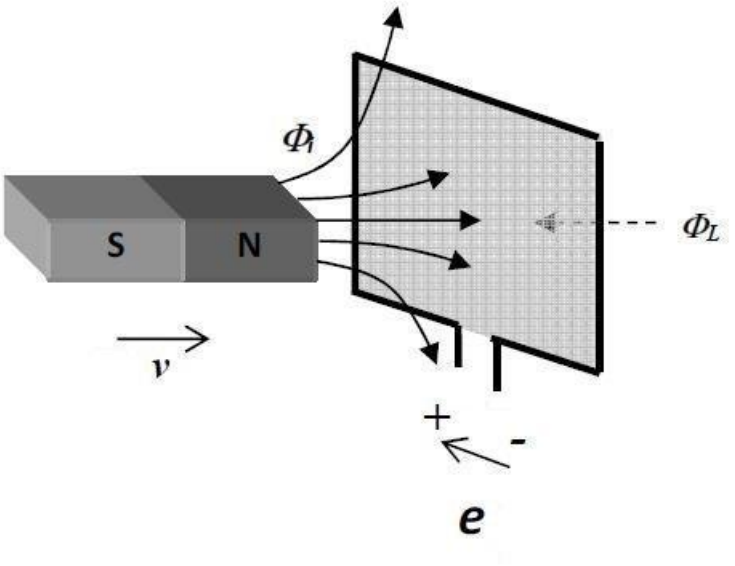

Imagen VII: ley de inducción electromagnética de Faraday

La ecuación que define esta ley es:

$$\oint_C \vec{E} \,.\, d\vec{l} = \quad -\frac{d}{dt}\int_S \vec{B} \,.\, d\vec{S}$$

Siendo:

- $\vec{E}$ el campo eléctrico inducido.
- $d\vec{l}$ es el diferencial de longitud del circuito que forma el contorno C.

- $\vec{B}$ es el campo magnético que atraviesa la superficie S, cuyo borde es C.

El signo negativo en el lado derecho de la expresión se debe a la ley de Lenz, la cual establece que la fuerza electromotriz se inducirá en el circuito cerrado de forma tal que su campo magnético se oponga a la variación del flujo que la produjo.

En caso de que la superficie se mantenga constante en el tiempo, se puede obtener a partir del teorema de Stokes la siguiente expresión perteneciente a las ecuaciones de Maxwell:

$$\nabla x \vec{E} = - \frac{\partial \vec{B}}{\partial t}$$

Para el caso de una bobina, la tensión inducida no sólo dependerá de la variación del campo magnético percibido por la bobina sino también dependerá del número de espiras que disponga la misma. Para el caso mencionado, la expresión anterior se transforma en la siguiente:

$$V_{ind} = - \frac{d\phi_{conc}}{dt}$$

Donde:

- V_{ind} es la tensión inducida entre los bornes de la bobina.
- ϕ_{conc} es el flujo magnético concatenado y variable en el tiempo percibido por la bobina.

El flujo magnético concatenado percibido por la bobina será la suma de los flujos magnéticos percibidos por cada una de las espiras que conforman la bobina. El flujo magnético percibido por una espira se calcula como:

$$\phi = \vec{B}\vec{S}$$

$$\phi = BScos(\theta)$$

Siendo:

- θ el ángulo existente entre la normal de la superficie y las líneas de campo magnético que la atraviesa.

  Para simplificar el análisis, puede considerarse que todas las espiras poseen la misma superficie y son atravesadas por las mismas líneas de flujo magnético. De esta forma, la tensión inducida en una bobina puede calcularse como:

$$V_{ind} = - NScos(\theta)\frac{dB}{dt}$$

  Donde:

- N es el número de espiras de la bobina.

3.2.3.3 Conclusión

A partir de lo expuesto previamente, se lleva a cabo la construcción de un sensor de corriente inductivo cuyo diseño se encuentra detallado en la sección 5.2.2.1 del presente documento.

3.3 Tecnologías de software

En la presente sección se describen las tecnologías, conceptos y metodologías utilizadas para el desarrollo de la aplicación.

3.3.1 Lenguajes de programación

En la presente sección, se describen los lenguajes de programación utilizados para el desarrollo de la aplicación.

3.3.1.1 HTML

HTML (HyperText Markup Language) es un lenguaje de hipertexto que permite la elaboración de documentos con características multimedia, tales como textos, imágenes y enlaces, entre otras. Estos documentos son interpretados y renderizados por un navegador web. Estos documentos están compuestos por diversas etiquetas definidas por el lenguaje las cuales le dan una estructura, organizando la información en diferentes bloques y visibilizando la misma a los usuarios.

Para el desarrollo del Front end de la aplicación se utiliza el lenguaje HTML en su versión 5, siendo la misma la última versión estable y OpenSource.

3.3.1.2 Hojas de estilo (CSS)

Las hojas de estilo CSS (Cascade Style Sheets) permiten definir la apariencia que tendrá un documento HTML, como por ejemplo, márgenes, tamaño y color de la fuente, imagen de fondo, entre otros. En cada hoja se definen reglas que, al ser llamados desde el documento, asignan las características establecidas al bloque correspondiente. Cada regla se compone de un conjunto clave-valor, siendo la clave el atributo a asignar mientras que el valor es la unidad o la propiedad a aplicar.

Para el desarrollo del Front end de la aplicación se utiliza CSS en su versión 3, siendo la misma la última versión estable y OpenSource.

3.3.1.3 JavaScript

JavaScript es un lenguaje de programación interpretado que es soportado por la gran mayoría de los navegadores web modernos. Se basa en un lenguaje de scripting orientado a objetos utilizado para desarrollar aplicaciones web, entre otros usos. Si bien puede utilizarse también del lado del servidor, dicho lenguaje corre del lado del cliente introduciendo mejoras en la interfaz de usuario ya que permite el desarrollo de páginas web dinámicas.

Se utiliza JavaScript en su versión ECMAScript 5.1 para el desarrollo del Front end de la aplicación siendo la misma la última versión estable y OpenSource.

3.3.1.4 Python

Python es un lenguaje de programación interpretado y multiplataforma de código abierto. Al ser un lenguaje multiparadigma, soporta diversos estilos de programación tales como la programación orientada a objetos y la programación funcional.

Una de las características principales de este lenguaje es su sencilla sintaxis lo cual permite acelerar los tiempos de desarrollo evitando que los mismos se vean afectados por el lenguaje.

Si bien Python tiene múltiples usos, puede ser utilizado para el desarrollo de aplicaciones web utilizando frameworks como Django.

Para el desarrollo de la aplicación se utiliza Python en su versión 3.7.1.

3.3.2 Almacenamiento de los datos

Dado que los datos a almacenar poseen una estructura predefinida, se ha decidido utilizar una base de datos relacional. Este tipo de base de datos se basa en el uso del modelo relacional el cual plantea el almacenamiento de la información en relaciones, conocidas como tablas, las cuales agrupan de forma lógica un conjunto de datos denominados tuplas o filas. Cada dato perteneciente a una tupla corresponde a un campo determinado conocido como columna.

Para manipular la información se utiliza un lenguaje relacional basado en el álgebra relacional, el cual permite definir la forma de realizar una consulta, y en el cálculo relacional, que indica lo que se pretende devolver, conocido como SQL (Structured Query Language). Dicho lenguaje es declarativo, de alto nivel y permite la definición, manipulación y control de los datos en bases de datos relacionales. Su alcance incluye todas las operaciones CRUD (Create, Read, Update, Delete) vinculadas a los datos.

Para el almacenamiento de los datos se seleccionó MySQL como sistema de gestión de base de datos relacional en su versión 8.0, siendo la misma la última versión estable.

3.3.3 API Rest

Una API Rest es una interfaz web que permite la interacción entre dos o más servicios dando la posibilidad de traficar información y manejar recursos entre ellos.

Para comprender la capacidad de manejo de recursos por parte de una API Rest, es importante definir los diferentes métodos HTTP que existen los cuales definen la acción que se debe realizar sobre un recurso determinado. Los mismos son:

- GET: utilizado para consultar información sobre un recurso determinado al servidor.
- POST: se utiliza para insertar un nuevo registro en la base de datos.
- PUT: permite actualizar un registro existente por completo.

- PATCH: similar al método anterior, con la diferencia de que solo actualiza un fragmento del registro.
- DELETE: utilizado para eliminar un registro en la base de datos del servidor.
- HEAD: se utiliza para obtener información sobre un registro determinado sin devolver dicho registro.

3.3.4 Frameworks de desarrollo

Un framework consiste en un conjunto de módulos que permiten el desarrollo de aplicaciones a partir de librerías o funcionalidades ya creadas. En la presente sección se describen los frameworks utilizados para el desarrollo de la aplicación web y la API Rest.

3.3.4.1 Django

Django es un framework de desarrollo web de alto nivel que utiliza Python como lenguaje de programación y provee un marco de trabajo completo y eficiente para desarrollar Aplicaciones Web de una gran complejidad. El motivo de su utilización se debe a que permite un desarrollo robusto y rápido de aplicaciones web haciendo foco en brindar seguridad y escalabilidad a las mismas.

Django propone un patrón de desarrollo conocido como modelo-vista-controlador (MVC), el cual está formado por tres niveles

- El Modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La Vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El Controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

  La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva

para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.).

El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación.

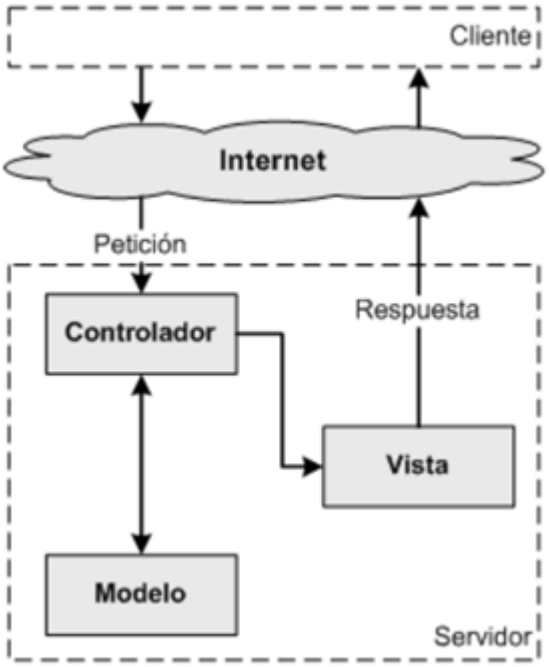

Imagen VIII: esquema de la arquitectura MVC

Para la construcción de la aplicación web que se detalla en la sección 5.4 se utiliza Django en su versión 2.2.

3.3.4.2 Flask

Flask es un microframework que utiliza Python como lenguaje y permite un desarrollo rápido y sencillo de servicios web, tales como una API Rest. Se define como “micro” ya que mantiene un núcleo simple y ligero pero extensible, permitiendo incorporar a demanda

aquellas bibliotecas necesarias para el desarrollo y funcionamiento de la aplicación en cuestión.

Estas cuestiones lo convierten en un microframework optimo para el desarrollo de la API Rest que se detalla en la sección 5.4.4, para la cual se utiliza Flask en su versión 2.0.

Capítulo 4: Sistema de telemetría

4.1 Descripción general

El sistema de telemetría desarrollado en este proyecto es una solución integral y escalable que permitirá a las empresas de transporte de energía eléctrica tener un mayor control sobre las variables críticas de las redes de media tensión. Dichas empresas no sólo tendrán acceso a las mediciones de consumo en tiempo real, sino que además el sistema es capaz de disparar alarmas en caso de que ocurran determinadas fallas o condiciones de operación riesgosas.

Los eventos que el sistema puede detectar fueron definidos a partir de la investigación previa realizada, los cuales son:

- Falla por cortocircuito
- Oscilación extrema de los cables

Los encargados de detectar dichos eventos son las “terminales inalámbricas”, que constan de la electrónica necesaria, para obtener, procesar y enviar la información obtenida. Estos dispositivos se colocan directamente sobre los cables de media tensión a través de un sistema de agarre especialmente diseñado.

La gran ventaja del sistema es que además de poder detectar este tipo de anomalías en tiempo real, las terminales inalámbricas cuentan con ubicación geográfica (GPS), lo cual permite identificar el tramo de la red donde ocurre el problema.

Para que todo el sistema pueda funcionar, es necesario que los dispositivos de medición sean capaces de reportar esta información hacia un servidor, esto se logra a través del uso de la tecnología LoRaWAN, la cual fue diseñada específicamente para dispositivos IoT de bajo consumo. Cabe destacar que el presente proyecto no incluye el diseño y despliegue de la red de comunicaciones.

Una vez que la información requerida llega al servidor, la misma es almacenada en una base de datos. Luego, una página web realizará consultas a dicha base de datos para poder mostrar la información al usuario final.

Esquema general de la solución.

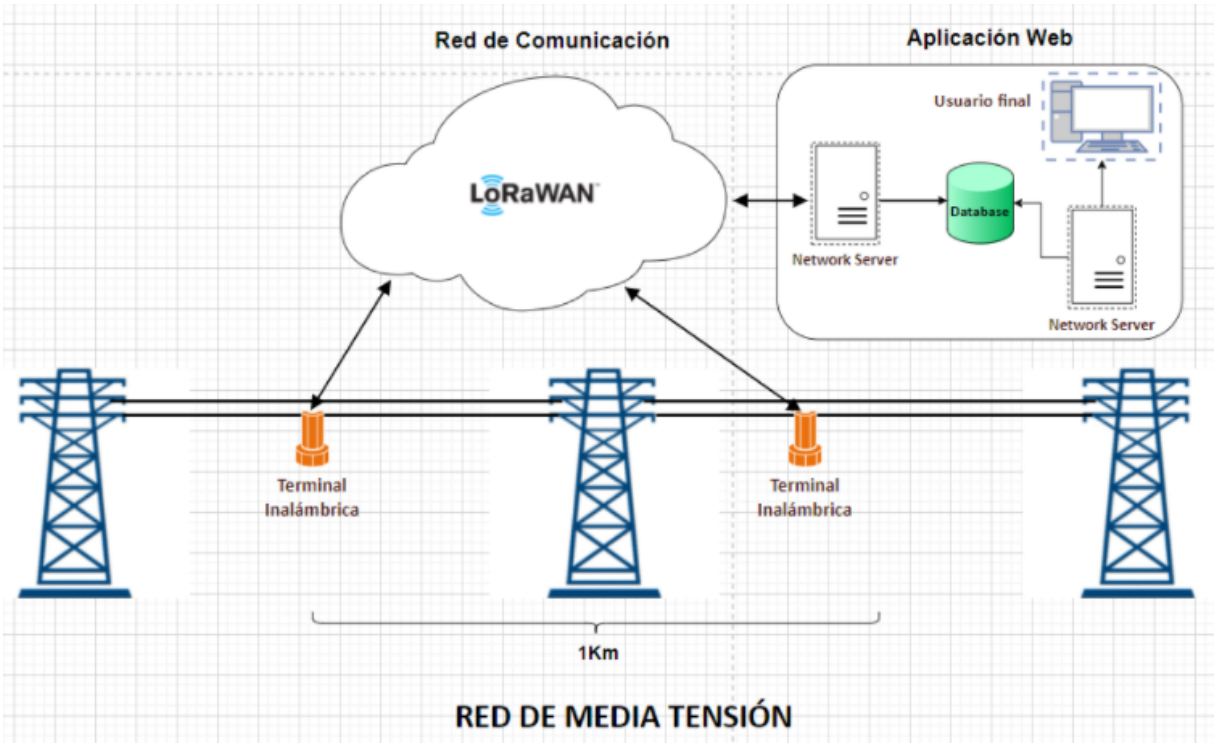

Imagen IX: Esquema general de la solución

Como se puede apreciar en el diagrama y se mencionó en la introducción, la solución consta de tres partes fundamentales: la red de comunicación, las terminales inalámbricas y la aplicación web. A continuación se procede a detallar el desarrollo de las terminales, de la aplicación web y, además, la construcción del prototipo.

4.2 Terminales inalámbricas

Para mayor facilidad de entendimiento, las terminales inalámbricas las podemos subdividir a su vez en las siguientes partes fundamentales:

- Alimentación y Almacenamiento de energía
- Adquisición
- Comunicación
- Procesamiento
- Gabinete

A continuación se puede ver un diagrama general de la terminal inalámbrica:

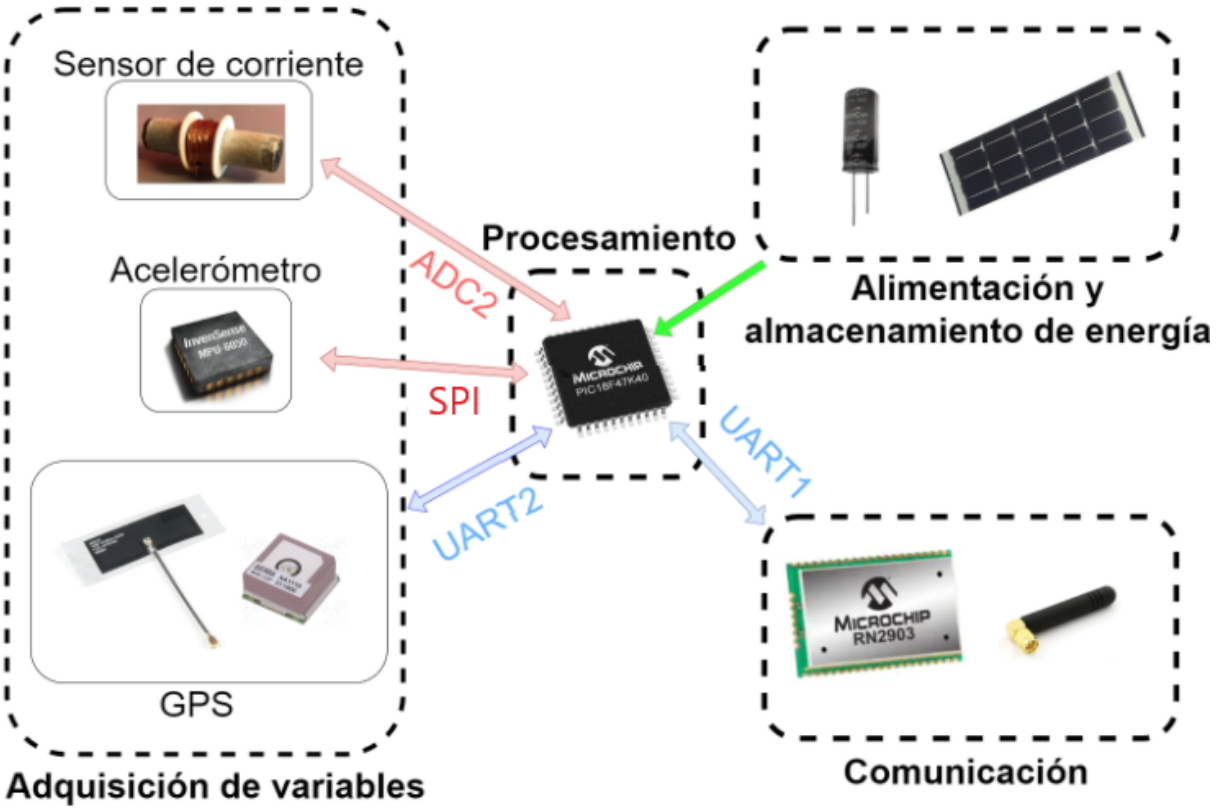

Imagen X: Esquema general de la terminal inalámbrica

4.2.1 Alimentación y almacenamiento de energía

Tal como se menciona en la sección 3.2.2 del presente documento, los dispositivos utilizarán la energía solar como fuente de energía. En la esta sección se detalla el diseño y la construcción del sistema de alimentación y almacenamiento de energía.

4.2.1.1 Diseño del circuito de alimentación

En función de los módulos considerados para el dispositivo, el mismo debe ser alimentado con 2,7 V y el consumo aproximado es de 700 μA mientras se encuentra en reposo y 53 mA durante el estado activo. De forma tal que el sistema de alimentación debe ser capaz de brindarle suministro eléctrico que cubra las condiciones mencionadas.

Para el diseño del sistema de alimentación se tienen en cuenta las siguientes consideraciones:

- El dispositivo no dispondrá de luz solar durante diez horas diarias, lo que equivale a 36.000 segundos, de forma tal que la carga total de los supercapacitores debe ser capaz de alimentarlo durante ese periodo.
- Se utilizará un regulador de tensión conectado entre los paneles y los supercapacitores que permita cargar estos últimos a un valor máximo de tensión de 2,7V.
- La tensión de alimentación mínima admitida es de 2,4 V.
- El dispositivo se encontrará en estado “activo” cuatro veces por hora por un lapso de 15 segundos por vez. Esto equivale a 600 segundos en un lapso de 10 horas, de forma tal que los 35.400 segundos restantes el dispositivo se encuentra en estado de “reposo”. En consecuencia, se considera despreciable el tiempo de estado “activo”.
- Los paneles solares generan una corriente de carga máxima de 40 mA.
- Se tendrá en cuenta un margen de seguridad del 20% para la capacitancia teórica calculada.
- Los supercapacitores deben adquirir el 95% de su carga en un lapso menor a las tres horas.

- Se considera que el dispositivo corresponde a una carga completamente resistiva cuya magnitud se calcula como:

$$R = \frac{V}{I}$$

$$R_{reposo} = \frac{2{,}7\ V}{700x10^{-6}\ A} = 3.857\ \Omega$$

4.2.1.1.1 Cálculo de acumuladores

Proceso de descarga

La selección del supercapacitor dependerá de la capacitancia requerida para la aplicación. El proceso de descarga de un condensador responde a la ecuación:

$$V_c(t) = V_0 e^{-\frac{t}{RC}}$$

Siendo:

- $V_c(t)$ la tensión entre los bornes del supercapacitor al instante t.
- V_0 el valor de tensión del supercapacitor cargado.
- R la resistencia del circuito.
- C la capacidad del supercapacitor.
- t el instante medido en segundos.

En consecuencia, la capacitancia necesaria se calcula como:

$$C = -\frac{t}{Rln(\frac{V_c(t)}{V_0})}$$

Como se mencionó previamente, se pretende que para el instante t=36.000s, la tensión entre los bornes del supercapacitor alcance el valor mínimo admitido(2,4V) y se considera un

margen de seguridad del 20%.Teniendo en cuenta dichas condiciones de operación, la capacitancia necesaria para alimentar el dispositivo es:

$$C =- \ 1,2 \frac{36.000\ s}{3.857\ \Omega \ln ln \left(\frac{2,4\ V}{2,7\ V}\right)} = \ 95,1\ F$$

Proceso de carga

La corriente que alimenta a los supercapacitores durante el proceso de carga será brindada por los paneles solares, de forma tal que en primer lugar se calcula la resistencia necesaria en el circuito de carga para limitar la corriente. La corriente de carga de un condensador responde a la ecuación:

$$I(t) = \frac{V_0}{R}(e^{-\frac{t}{RC}})$$

Siendo:

- V_0 la tensión de la fuente de alimentación.
- R la resistencia del circuito.
- C la capacidad del supercapacitor.
- t el instante de carga medido en segundos

La corriente máxima se dará en el instante t=0 y, según se estableció como condición de diseño, debe tener un valor de 40 mA. En consecuencia, la resistencia mínima del circuito de carga se calcula como:

$$R_{min} = \frac{V_0}{I(0)}\left(e^{-\frac{0}{RC}}\right) = \frac{2,7V}{0,04A} = \ 67,5\Omega$$

A efectos prácticos, se utilizará una resistencia de 68Ω lo cual corresponde a un valor normalizado de resistencias comerciales.

Se estableció como condición de diseño el tiempo de carga necesario para cada supercapacitor. Para ello, es necesario analizar el proceso de carga de estos responde a la siguiente ecuación:

$$V_c(t) = V_0(1 - e^{-\frac{t}{RC}})$$

Siendo:

- $V_c(t)$ la tensión entre los bornes del supercapacitor al instante t.
- V_0 la tensión de la fuente de alimentación.
- R la resistencia del circuito.
- C la capacidad del supercapacitor.
- t el instante medido en segundos.

En consecuencia, el tiempo de carga se calcula como:

$$t = - RCln(1 - \frac{V_c(t)}{V_0})$$

Como se mencionó previamente, es condición de diseño que para un instante t menor a las tres horas(10.800s) el capacitor adquiera el 95% de la carga máxima(2,7V). En función de las consideraciones mencionadas y de la capacitancia necesaria, el tiempo de carga de cada supercapacitor será:

$$t = - \ 68\Omega \ 50F \ln ln\left(1 - \frac{0{,}95 \ 2{,}7V}{2{,}7V}\right) = 10.185 \ segundos = 2{,}83 \ horas$$

En función de la capacitancia necesaria y de los valores de capacitancia nominales disponibles en el mercado, se concluyó que la mejor opción es la instalación de dos supercapacitores de 50 F. La conexión de estos se realiza en paralelo de forma tal que sus capacidades se suman, alcanzando los 100 F totales. El código del modelo elegido es DGH506Q2R7.

Imagen XI: Supercapacitor DGH506Q2R7

4.2.1.1.2 Selección de paneles solares

La selección del panel solar dependerá de los valores de tensión y corriente que se necesitan para cumplir con las condiciones de diseño y, a su vez, del rendimiento de cada uno de ellos. Para el desarrollo del prototipo se seleccionaron dos modelos de paneles solares que cumplen con las condiciones de diseño, pero poseen características diferentes y se evaluó su comportamiento para determinar cuál es el adecuado para la aplicación en cuestión.

4.2.1.2 Pruebas del sistema de alimentación

El objetivo de las pruebas es evaluar empíricamente los tiempos de carga y descarga de los supercapacitores. El proceso de carga se realizó a través de dos módulos solares diferentes con el objetivo de analizar las diferencias entre ambos y determinar cuál es el óptimo para la aplicación. Ambos sistemas alimentaron dos supercapacitores de las características mencionadas en el diseño bajo las mismas condiciones ambientales realizándose las pruebas en simultáneo. El proceso de descarga se realiza a través de una carga resistiva de valor nominal 3,9kΩ. Los modelos de paneles solares utilizados son:

- MP3-37: módulo fotovoltaico flexible. En condiciones óptimas de operación entrega 3V – 50mA en condiciones óptimas ambientales. Se utilizaron dos módulos conectados en serie para alcanzar la tensión necesaria.

Imagen XII: módulo fotovoltaico flexible MP3-37

- AM-5907 CAR: módulo fotovoltaico rígido. En condiciones óptimas de operación entrega 7,7V – 45,7mA en condiciones óptimas ambientales. Se utilizó un módulo único.

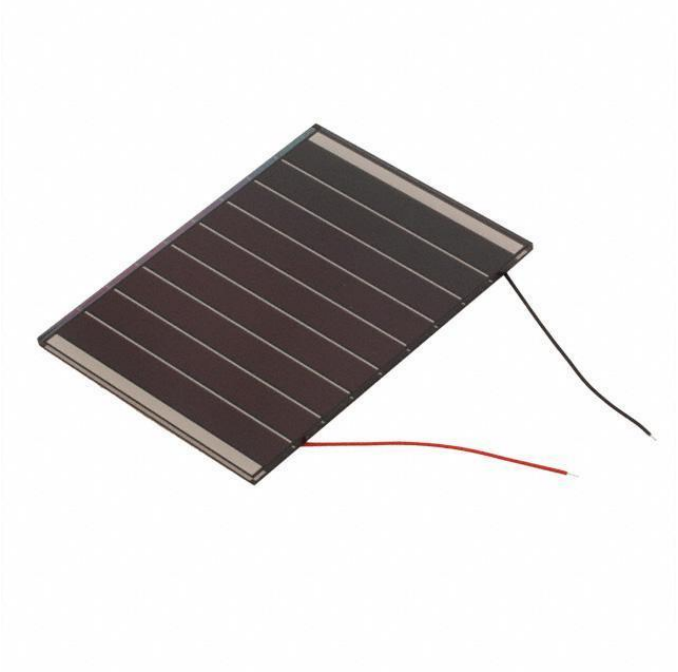

Imagen XIII: módulo fotovoltaico rigido AM-5907 CAR

Ambos modelos entregan una tensión mayor a la admitida por los supercapacitores, de forma tal que se intercala en el circuito un regulador de tensión entre los paneles solares y los supercapacitores que permite que estos últimos reciban los 2,7V requeridos. El modelo de regulador de tensión elegido es LM2931-N.

Imagen XIV: regulador de tensión LM2931-N

El circuito de conexión responde al siguiente esquema:

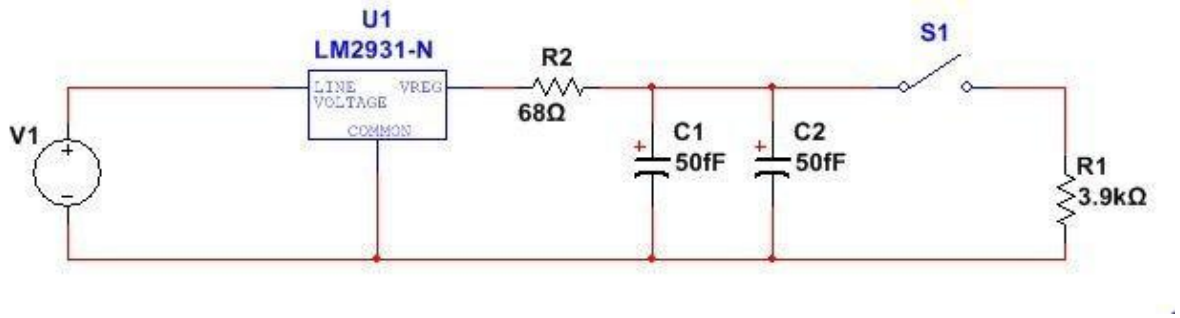

Imagen XV: esquema del circuito de conexión

A continuación, se adjuntan imágenes de los circuitos construidos con ambos módulos fotovoltaicos.

Circuito de conexión con el módulo MP3-37

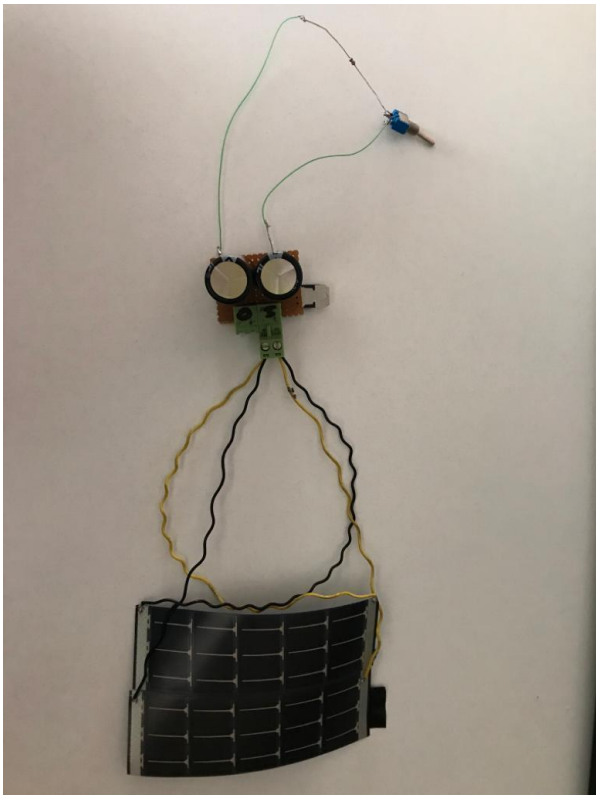

Imagen XVI: circuito de conexión con módulo MP3-37

Circuito de conexión con el módulo AM-5907 CAR

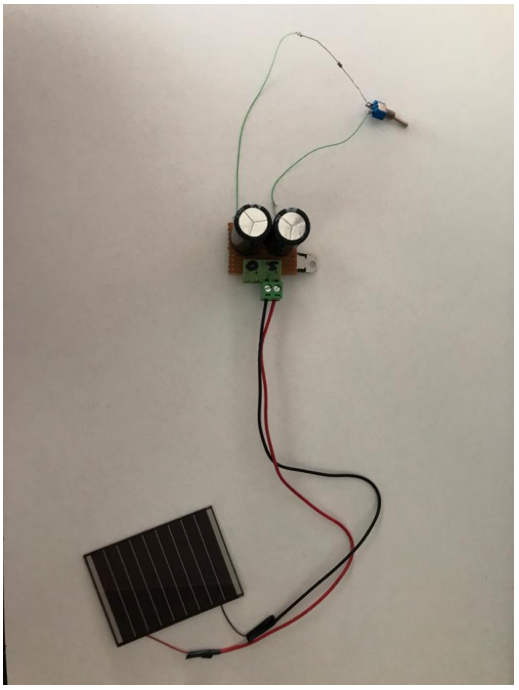

Imagen XVII: circuito de conexión con módulo AM-5907 CAR

4.2.1.2.1 Resultados obtenidos

Durante las pruebas, se midió la caída de tensión en la carga cada una hora minutos durante un lapso de 48 horas. Los valores obtenidos fueron los siguientes:

Fecha y hora	Panel rigido(mV)	Paneles flexibles(mV)
19/7/2019 13:02	2665	2686
19/7/2019 14:03	2713	2686
19/7/2019 15:04	2713	2686

19/7/2019 16:05	2713	2686
19/7/2019 17:06	2706	2679
19/7/2019 18:07	2692	2658
19/7/2019 19:07	2672	2638
19/7/2019 20:08	2658	2612
19/7/2019 21:09	2645	2599
19/7/2019 22:10	2632	2580
19/7/2019 23:11	2618	2561
20/7/2019 00:12	2605	2542
20/7/2019 01:13	2599	2524
20/7/2019 02:14	2586	2512
20/7/2019 03:15	2573	2494
20/7/2019 04:16	2567	2482

20/7/2019 05:17	2555	2470
20/7/2019 06:19	2542	2459
20/7/2019 07:20	2530	2441
20/7/2019 08:06	2524	2436
20/7/2019 09:07	2536	2482
20/7/2019 10:08	2580	2592
20/7/2019 11:09	2672	2686
20/7/2019 12:09	2706	2686
20/7/2019 13:10	2713	2686
20/7/2019 14:11	2713	2686
20/7/2019 15:12	2706	2686
20/7/2019 16:14	2706	2679
20/7/2019 17:00	2699	2672

20/7/2019 18:02	2686	2658
20/7/2019 19:03	2672	2632
20/7/2019 20:04	2658	2612
20/7/2019 21:05	2652	2592
20/7/2019 22:05	2638	2580
20/7/2019 23:06	2632	2561
21/7/2019 00:07	2618	2548
21/7/2019 01:08	2612	2530
21/7/2019 02:09	2599	2524
21/7/2019 03:10	2592	2500
21/7/2019 04:11	2580	2488
21/7/2019 05:12	2567	2470
21/7/2019 06:13	2561	2459

21/7/2019 07:14	2548	2447
21/7/2019 08:00	2548	2430
21/7/2019 09:01	2555	2447
21/7/2019 10:03	2599	2506
21/7/2019 11:03	2679	2665
21/7/2019 12:04	2713	2686
21/7/2019 13:06	2713	2686

Tabla III: resultados de pruebas realizadas con ambos módulos solares

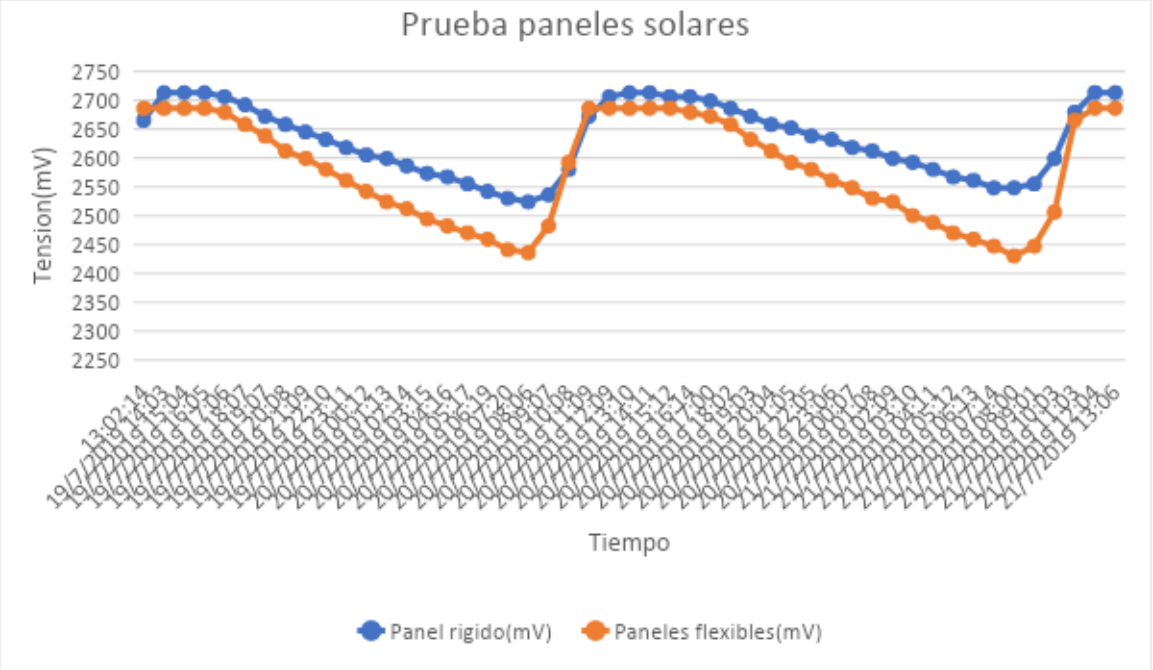

Imagen XVIII: gráfico de resultados de pruebas realizadas con ambos módulos solares

En primer lugar, es necesario aclarar que los valores de tensión en ambos circuitos difieren por dos razones principales:

a) Las cargas utilizadas no son perfectamente iguales por lo que el consumo varía y en consecuencia la variación en la tensión entre los bornes de salida de cada módulo de supercapacitores varía de instante a instante.

b) Los paneles solares se re-conectan a su respectivo módulo de supercapacitores en el mismo instante, punto en el cual existe una diferencia en los valores de tensión los bornes salida de los módulos por lo mencionado en el apartado anterior.

Sin embargo, estas diferencias en los valores de tensión hacen más clara la conclusión: los paneles flexibles logran mejorar el tiempo de carga de los supercapacitores. A partir del análisis de los datos puede observarse que los mismos logran alcanzar el 95% de la tensión máxima (2,7V) entre los bornes de salida de los supercapacitores en un lapso medio de 2,5 horas partiendo de un valor de tensión inicial de aproximadamente 90 mV inferior respecto al

panel rígido. Por su parte, estos últimos alcanzan el objetivo en un tiempo medio de 3 horas considerando que es menor la diferencia entre los valores de tensión inicial y final.

Debido a estas conclusiones, se seleccionó como panel solar para el prototipo el modelo MP3-37. Con el objetivo de establecer un margen de seguridad, se considera que el rendimiento del panel puede disminuir hasta un 50% en condiciones climáticas desfavorables. De forma tal que se instalarán dos módulos MP3-37 conectados en serie para obtener la tensión de alimentación necesaria para el dispositivo aún es las condiciones mencionadas.

4.2.2 Adquisición de variables

En la presente sección se detallan los mecanismos utilizados por el dispositivo para la obtención de las variables de interés mencionadas previamente.

4.2.2.1 Sensor de corriente inductivo

Con el objetivo de conocer la intensidad de corriente existente en el conductor de una fase dada, se desarrolló un sensor inductivo no invasivo. El principio de funcionamiento de este consiste en entregar a la salida un determinado valor de tensión proporcional a la intensidad de corriente existente en el conductor basándose en la ley de Biot-Savart y la ley de induccion electromagnética detalladas en las secciones 3.2.3.1 y 3.2.3.2 respectivamente del presente documento.

4.2.2.1.3 Diseño del sensor

Con el objetivo de simplificar el diseño del sensor, se tienen en cuenta las siguientes suposiciones:

a. La corriente a sensar que circula por el conductor responde a una curva senoidal definida por la siguiente expresión:

$$i = I_p sen(2\pi ft)$$

Donde:

- I_p es el valor pico de intensidad de corriente.
- f es la frecuencia de la señal.
- t es la variable de tiempo.

b. La frecuencia de la señal a medir es 50 Hz.

c. El conductor por el cual circula la corriente a sensar corresponde a un conductor rectilíneo y la normal de la superficie de cada espira son paralelas a dicho conductor.

d. El ángulo existente entre la normal de la superficie de las espiras de la bobina y las líneas de campo magnético es 0° o 180° dependiendo del semiciclo positivo y negativo de la corriente. Es decir, ambas curvas son paralelas en la superficie de cada una de las espiras.

e. El campo magnético total percibido por cada espira será el campo calculado para el centro de esta y será homogéneo para todas las espiras que conformen la bobina, considerando que las mismas se encuentran superpuestas en la misma posición y son el contorno de una superficie de valor constante e igual para todas ellas.

f. El medio en el cual se genera el campo magnético es el vacío por lo que la permeabilidad magnética utilizada será:

$$\mu = 4\pi\, x\, 10^{-7} \frac{N}{A^2}$$

A partir de la ley de Biot-Savart y la ley de inducción magnética de Faraday presentadas anteriormente en conjunto con las suposiciones consideradas, se concluye que la expresión

que permite calcular la tensión inducida en una bobina por una corriente que circula en un conductor rectilíneo responde a la siguiente expresión:

$$V_{ind} = - NS\frac{d(\frac{\mu}{4\pi r} I_p sen(2\pi ft))}{dt}$$

$$V_{ind} = - \frac{NS\mu f I_p}{2\,r} cos(2\pi ft)$$

De forma tal que la tensión inducida en la bobina será una señal sinusoidal desfasada 90° respecto de la señal que circula en el conductor principal.

Se utilizará como referencia la tensión interna del microcontrolador que tiene un valor nominal de 2,024V. Si bien el microcontrolador es alimentado con una tensión de 2,7 V y la misma podría ser utilizada como referencia, se desestimó esta opción debido a que el valor de la tensión aportado por la fuente podrá variar dependiendo de la carga existente en los supercapacitores, de forma tal que las mediciones tomadas dejarían de ser válidas. A su vez, la entrada digital del microcontrolador es capaz de leer tensiones positivas por lo que se rectifica la señal de salida del sensor. En consecuencia, se establecen las siguientes condiciones de diseño:

1. Para que el microcontrolador reciba una señal continua positiva, se coloca en la salida de la bobina un diodo que actúa como rectificador de media onda de la señal sinusoidal inducida. Se considera que el mismo tiene una caída de tensión de 0,7V.

2. La corriente máxima a leer será de 200 A. de forma tal que, al tratarse de una señal sinusoidal, el valor pico de esta será:

$$I_p = \sqrt{2}\, I_{RMS}$$

$$I_p = \sqrt{2}\, 200\, A = 282,85\, A$$

3. Se define 1,8V como el valor de tensión pico admisible en la entrada digital del microcontrolador, de forma tal que se tenga un margen suficiente para poder leer

condiciones de falla en el conductor principal. De forma tal que, sumado a la caída de tensión en el diodo, la tensión máxima inducida en la bobina debe ser:

$$V_{ind-pico} = 2,5V$$

4. Las espiras que conformen la bobina serán el contorno de una superficie circular que tendrá un diámetro de 2 cm, de forma tal que el valor de la superficie será:

$$S = \frac{\pi d^2}{4}$$

$$S = \frac{\pi(0,02m)^2}{4} = 0,00031416m^2$$

5. El centro de la superficie de las espiras se ubicará a una distancia radial *r* de 0,01 m del conductor principal.

A partir de las consideraciones y condiciones mencionadas, se calcula el número de espiras necesarias para cumplir con los objetivos de diseño:

$$N = 2\frac{V_{ind-pico}\, r}{S\mu f I_p}$$

$$N = 2\frac{2,5V\ 0,01\ m}{0,00031416\ m^2\ 4\pi\ x\ 10^{-7}\frac{N}{A^2}\ 50\ Hz\ 282,5\ A} = 8.967\ espiras$$

4.2.2.1.4 Construcción del sensor

Debido a que no existe circulación de corriente en la bobina, sino que solo existe inducción de tensión, no existe restricción para el valor de la sección del alambre que conforma la bobina, con lo cual, el mismo fue seleccionado para ocupar el menor espacio físico posible. De forma

tal que para la construcción del sensor se utilizó alambre de cobre esmaltado de 0,2 mm de diámetro.

La bobina fue construida utilizando una máquina bobinadora semiautomática utilizando un carrete plástico de $2\,cm$ de diámetro diseñado a través del software Autodesk Inventor Professional 2019 e impreso en 3D.

4.2.3.1.5 Pruebas del sensor

Se llevaron a cabo dos pruebas para evaluar el comportamiento del sensor construido. Los elementos utilizados para la prueba fueron:

- Generador de corriente alterna 250A-50Hz

Imagen XIX: generador de corriente alterna 250A-50Hz

- Sensor inductivo de corriente construido

Imagen XX: sensor inductivo de corriente construido

- Osciloscopio Hantek DSO5072p

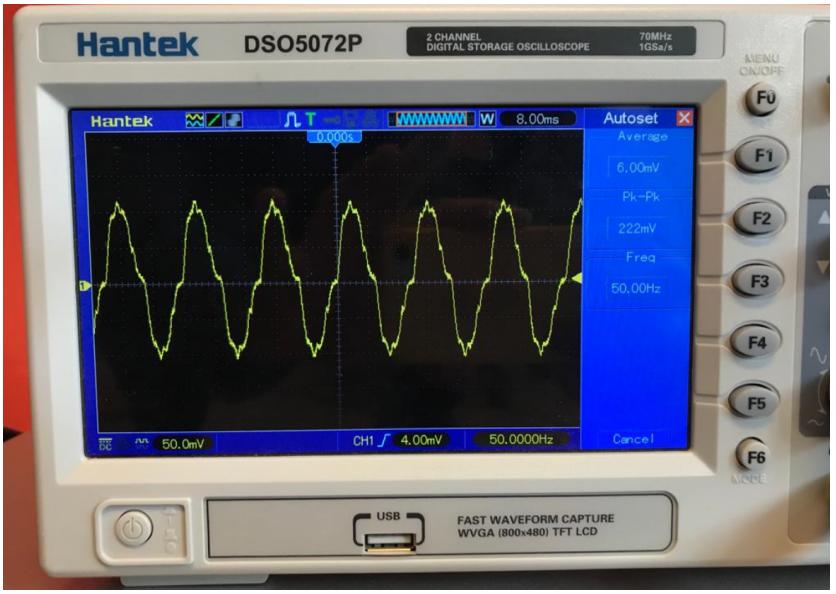

Imagen XXI: Osciloscopio Hantek DSO5072p

La primera de las pruebas consistió en generar un flujo de corriente alterna de 50Hz de frecuencia a partir del generador de corriente y aumentar su intensidad en intervalos de 10A hasta alcanzar los 200A de valor eficaz. La segunda prueba realizada consiste en reducir la intensidad de corriente en intervalos de 10A partiendo de los 200A de valor eficaz. El sensor

fue colocado sobre el cable quedando el centro de las espiras a 1cm del conductor y se midió con el osciloscopio los valores de tensión pico inducidos entre los bornes de la bobina.

Imagen XXII: conexión para pruebas de medición de corriente de sensor inductivo

4.2.2.1.6 Resultados obtenidos

Los resultados obtenidos fueron los siguientes:

Intensidad de corriente(A)	Prueba 1(V)	Prueba 2(V)
10	0,1325	0,234
20	0,22	0,397
30	0,41	0,565
40	0,524	0,741
50	0,72	0,94
60	0,87	1,047
70	0,94	1,129
80	1,052	1,25
90	1,215	1,42
100	1,298	1,598
110	1,48	1,78
120	1,63	1,814
130	1,71	1,975
140	1,869	2,126
150	2,1	2,217
160	2,12	2,325
170	2,265	2,45
180	2,45	2,51
190	2,5175	2,574
200	2,65	2,65

Tabla IV: resultados de pruebas de medición de corriente de sensor inductivo

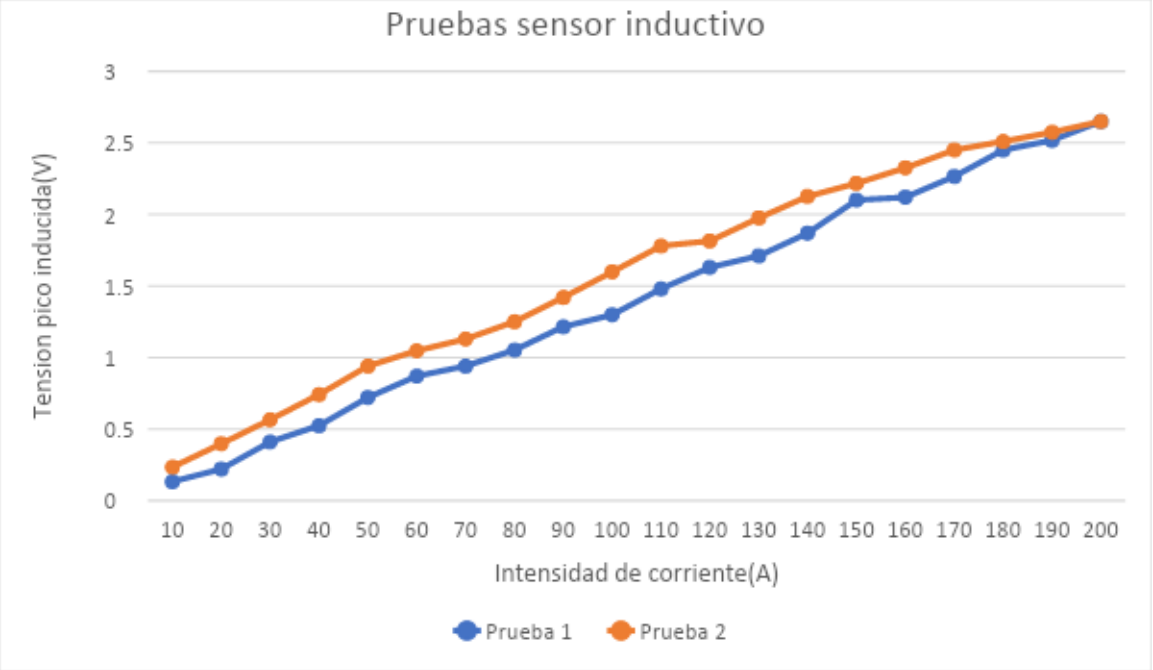

Imagen XXIII: gráfico de resultados de pruebas de medición de corriente de sensor inductivo

A partir de las pruebas realizadas puede concluirse que el sensor construido tiene un comportamiento prácticamente lineal que depende de los valores de intensidad de corriente que se tengan en el conductor.

4.2.2.2 Acelerómetro

Con el objetivo de generar una alarma en el sistema cuando existan vientos fuertes que puedan afectar al cableado eléctrico aéreo, es necesario realizar mediciones periódicas con algún sensor capaz de detectar movimiento u oscilaciones en el cable. En este caso, luego de una investigación previa, concluimos que la manera más económica y con un consumo acorde a la aplicación es utilizar un acelerómetro, en particular el ADXL362BCCZ-RL.

Esta decisión se basa principalmente en cuatro características principales de este componente.

- Consumo de corriente extremadamente bajo (menor a 2uA)
- Comunicación a través de interfaz serie SPI

- Modo de activación por movimiento
- Tensión de alimentación desde 1.6 v hasta 3.5 v

4.2.2.2.1 Conexión SPI con el acelerómetro

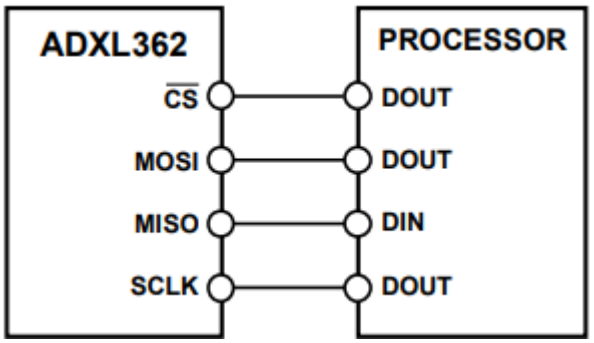

Imagen XXIV: esquema de conexión SPI microcontrolador-acelerómetro

En la figura anterior se puede apreciar la correcta conexión SPI de 4 cables entre el microcontrolador y el acelerómetro. Siendo:

CS: Chip Select (Activo bajo)

MOSI: Master output Slave input

MISO: Master input Slave output

SCLK: Serial clock

El puerto SPI utiliza una estructura multibyte en la que el primer byte es un comando. El conjunto de comandos del ADXL362 es:

0x0A: registro de escritura

0x0B: registro de lectura

0x0D: leer FIFO

La estructura de comando para el registro de lectura y el registro de escritura comandos es el siguiente:

</ CS abajo> <byte de comando (0x0A o 0x0B)> <dirección byte> <byte de datos> <bytes de datos adicionales para multibyte>…</ CS Alto>

4.2.2.2.2 Modo Wake-up

El modo Wake-up es ideal para la detección simple de presencia o ausencia de movimiento con un consumo de energía extremadamente bajo (270 nA a una tensión de alimentación de 2,0 V).

Este modo de funcionamiento reduce el consumo de corriente a un nivel muy bajo midiendo la aceleración sólo unas seis veces por segundo para determinar si hay movimiento. Si se detecta movimiento, el acelerómetro puede responder de forma autónoma de las siguientes formas:

1) Cambiar al modo de medición a “ancho de banda completo”
2) Enviar una interrupción a un microcontrolador
3) Despertar los circuitos aguas abajo, según la configuración

Resulta particularmente interesante para este proyecto utilizar el modo wake-up de manera que el sistema quede en un bajo consumo (en estado de reposo) hasta que el umbral de movimiento sea superado y se dé aviso al microcontrolador a través de una interrupción (2).

Para detectar actividad, el ADXL362 compara el valor absoluto de los datos de aceleración de 12 bits (con signo) con los de 11 bits (sin signo) del valor THRESH_ACT.

El valor THRESH_ACT, se refiere a un valor sin signo de 11 bits que comprende el registro THRESH_ACT_L (que contiene sus ocho LSB) y el registro THRESH_ACT_H (que contiene sus tres MSB). El valor en g depende del ajuste del rango de medición seleccionado.

THRESH_ACT [g] = THRESH_ACT / Sensibilidad

Imagen XXV: registros asociados al modo wake-up

4.2.2.2.3 Mapa de registros de funciones INT1 / INT2

Los registros INT1 e INT2 configuran los pines de interrupción INT1 / INT2 respectivamente. Los bits [B6: B0] seleccionan qué funciones generan una interrupción en el pin. Si su bit correspondiente se establece en 1, la función genera una interrupción en el pin INT. El Bit B7 configura si el pin opera en activo alto (B7 bajo) o modo activo bajo (B7 alto).

Bits	Bit Name	Settings	Description	Reset	Access
7	INT_LOW		1 = INT1 pin is active low.	0x0	RW
6	AWAKE		1 = maps the awake status to INT1 pin.	0x0	RW
5	INACT		1 = maps the inactivity status to INT1 pin.	0x0	RW
4	ACT		1 = maps the activity status to INT1 pin.	0x0	RW
3	FIFO_OVERRUN		1 = maps the FIFO overrun status to INT1 pin.	0x0	RW
2	FIFO_WATERMARK		1 = maps the FIFO watermark status to INT1 pin.	0x0	RW
1	FIFO_READY		1 = maps the FIFO ready status to INT1 pin.	0x0	RW
0	DATA_READY		1 = maps the data ready status to INT1 pin.	0x0	RW

Tabla V: Bits del registro INTMAP1

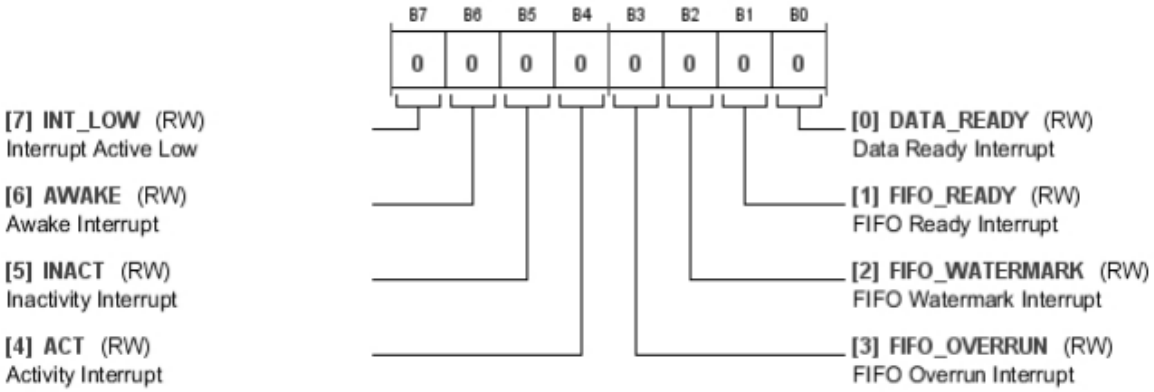

Imagen XXVI: Descripción registro INMAP1

Table 16. Bit Descriptions for INTMAP2

Bits	Bit Name	Settings	Description	Reset	Access
7	INT_LOW		1 = INT2 pin is active low.	0x0	RW
6	AWAKE		1 = maps the awake status to INT2 pin.	0x0	RW
5	INACT		1 = maps the inactivity status to INT2 pin.	0x0	RW
4	ACT		1 = maps the activity status to INT2 pin.	0x0	RW
3	FIFO_OVERRUN		1 = maps the FIFO overrun status to INT2 pin.	0x0	RW
2	FIFO_WATERMARK		1 = maps the FIFO watermark status to INT2 pin.	0x0	RW
1	FIFO_READY		1 = maps the FIFO ready status to INT2 pin.	0x0	RW
0	DATA_READY		1 = maps the data ready status to INT2 pin.	0x0	RW

Tabla VI: Bits del registro INTMAP2

En base a lo descrito anteriormente, se debe reservar en el microcontrolador un pin de propósito general (GPIO) para conectar la salida de la interrupción del acelerómetro. De esta manera, una vez configurado el acelerómetro y conectado correctamente con el microcontrolador, el mismo recibirá una señal de interrupción cada vez que el umbral seteado de movimiento sea superado y así interpretar esta señal como indicador de que un viento suficientemente fuerte está generando oscilaciones peligrosas en los cables. Cabe destacar que, para establecer el umbral de movimiento con un mayor grado de precisión se deben realizar pruebas empíricas.

4.2.2.3 Módulo GPS

El módulo GPS elegido para este proyecto es el modelo XA1110 de la marca SIERRA WIRELESS. El mismo se ajusta perfectamente a las necesidades, a continuación se enumeran algunas de sus características principales:

- Sensibilidad: -165 dbm
- Bajo consumo de corriente (3.3V): GPS+GLONASS: 23mA/ 26mA /29mA (min / típico / max)
- Trackeo: 24mA / 28mA /32mA (min / típico/ max)
- Interfaces de comunicación: I2C/ SPI/ UART (configurables)
- Antena interna y externa
- 33 canales de trackeo GPS +GLONASS
- 99 canales de adquisición GPS +GLONASS

Se utilizará la interfaz serie UART disponible para realizar la comunicación con el microcontrolador. Dicha comunicación se basa en comandos del tipo AT. Los comandos se pueden encontrar en la documentación del XA1110, dicha documentación se encuentra en la sección “Anexo” del presente documento.

Cabe destacar que el formato seleccionado para el envío de la posición geográfica es el NMEA, este es un protocolo que define los requerimientos de datos y tiempo de transmisión en el formato serial. Define también la norma para que cada equipo sea emisor y pueda ser escuchado por muchos receptores.

4.2.3 Comunicación

Como se describió anteriormente, una vez que las distintas variables son adquiridas y procesadas por la terminal, el siguiente paso es enviar esa información de manera inalámbrica.

Esto se realiza a través de un módulo de comunicación que se encarga de modular y transmitir los datos.

4.2.3.1 Módulo de comunicación RN2903

El módulo RN2903 de la marca Microchip es un transceptor de baja potencia y largo alcance que funciona en la banda de 915 MHz. El mismo soporta modulaciones FSK, GFSK y LoRa.

La elección de este transceptor se debe principalmente a los siguientes elementos:

- Bajo consumo
- Bajo costo
- Comunicación a través del puerto serie (UART)

El RN2903 es ideal para aplicaciones de sensorizado a distancia y es utilizado para innumerables aplicaciones IoT. Esto se debe a que utilizando un microcontrolador externo económico (solo hace falta que cuente con interfaz UART para enviar comandos) ya es posible controlar el módulo y lograr enviar información al aire (utilizando las modulaciones antes mencionadas).

Para más información sobre el módulo de comunicaciones, referirse a la hoja de datos del mismo, presente como anexo de este documento.

Como se explica en la sección 5.2 del presente documento, la tecnología de comunicación elegida es LoRaWAN. Con lo cual, el módulo es configurado de manera tal que el mismo transmita utilizando modulación LoRa con activación por personalización (ABP).

4.2.4 Lógica de procesamiento

El microcontrolador utilizado para este proyecto de desarrollo es el PIC18LF47k40, a continuación se listan sus características más importantes respecto a los requerimientos específicos del proyecto:

- Bajo costo

- Bajo consumo, ya que cuenta con tecnología XLP (Extreme Low Power)
- Cuenta con un módulo ADCC (ADC con computación)
- Puede funcionar con tensiones de alimentación menores a 3.3v
- Gran disponibilidad de puertos GPIO
- Soporta dos canales de comunicación UART/EUSART simultáneos
- Cuenta con interfaz SPI / I2C

Nota: Para más detalles acerca del dispositivo, al final de este documento se puede encontrar la hoja de datos correspondiente.

El funcionamiento del código es el siguiente:

El dispositivo se encuentra la mayor parte del tiempo dentro de un loop while(1) {} en estado de bajo consumo, utilizando la función Sleep(). El mismo se puede "despertar" por tres diferentes motivos:

- Existe un cortocircuito: Esto ocurre cuando se supera el umbral de tensión configurado en el módulo ADCC en el puerto al que se encuentra conectado el sensor de corriente, lo cual dispara una rutina de interrupción.
- Existe una oscilación del cable: Este evento se da cuando el acelerómetro genera una señal de interrupción hacia un pin GPIO del microcontrolador.
- Por tiempo: El microcontrolador debe tener una sincronización de tiempo para decidir cuándo corresponde transmitir datos o no.

En los casos que ocurran eventos tales como un cortocircuito o una oscilación extrema del cable, el microcontrolador inmediatamente sale de su estado de bajo consumo y comienza el proceso de transmisión, para enviar la alarma al servidor. El proceso de transmisión se basa en

una comunicación asíncrona serie (UART) con el módulo de LoRa RN2903, utilizando los puertos RC6 para transmitir y RC7 para recibir.

4.3 Prototipo de la terminal inalámbrica

El prototipo realizado cuenta con la mayoría de las funcionalidades de la terminal inalámbrica y es una primera versión del producto final. Dentro de dichas funcionalidades encontramos:

- Detección de baja o alta tensión:
- Ubicación GPS
- Transmisión de la información a través de LoRaWAN

4.3.1 Esquemático del circuito

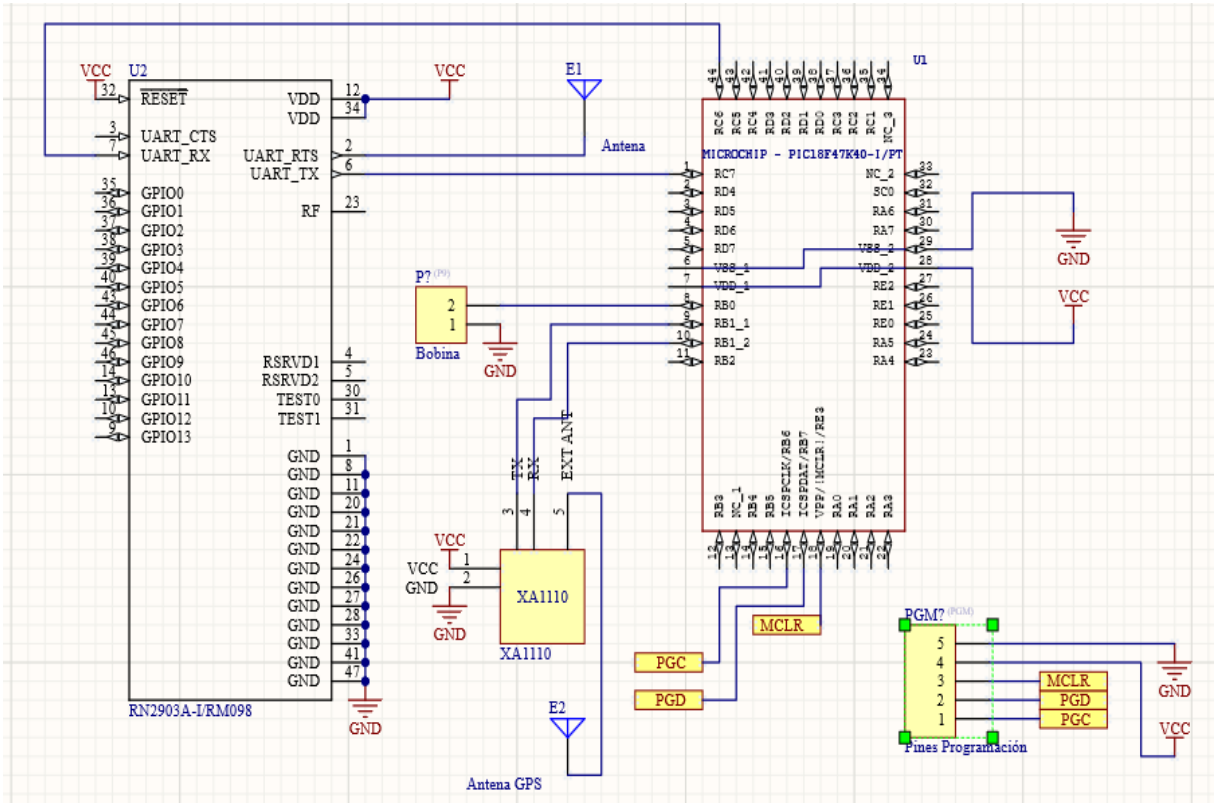

Imagen XVII: esquemático del circuito

4.3.2 Fotografías del prototipo

4.3.2.1 PCB (Lado A)

Imagen XXVIII: PCB lado A

4.3.2.2 PCB (Lado B)

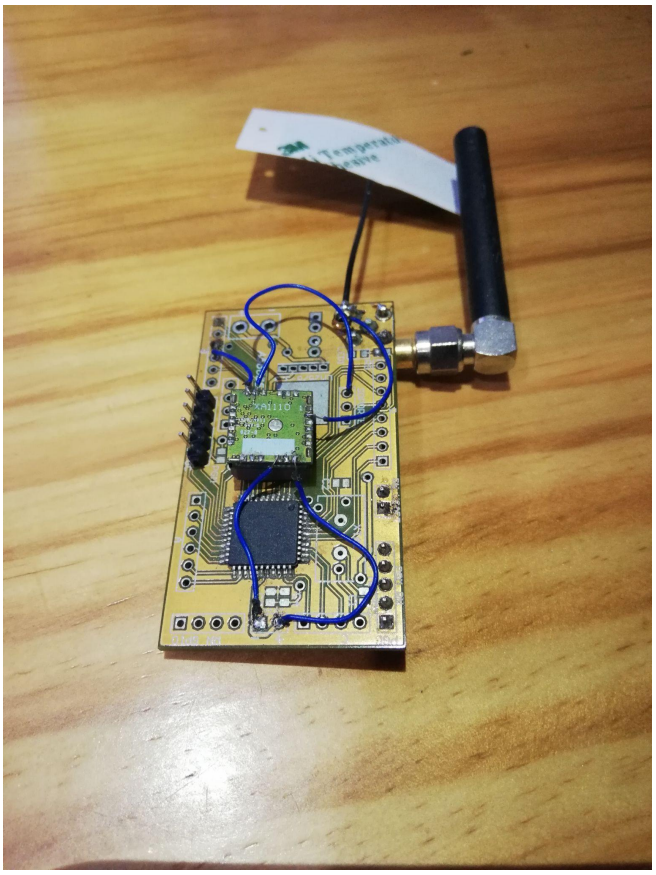

Imagen XXIX: PCB lado B

4.3.3 Gabinete 3D

Para el diseño del gabinete del dispositivo se utilizó el software Autodesk Inventor Professional 2019, software de diseño mecánico y CAD 3D. Se elaboraron cuatro piezas totales que, al ser ensambladas, constituyen el gabinete. Las piezas serán construidas por impresión 3D. A continuación, se describe cada una de ellas:

4.3.3.1 Pieza N°1

La pieza N°1 cuenta con un soporte para el carrete en el cual se bobino el sensor inductivo de corriente y, a su vez, cuenta con un espacio para montar el circuito impreso del dispositivo. Los orificios permiten el paso de los extremos de la bobina para conectarlos a la placa.

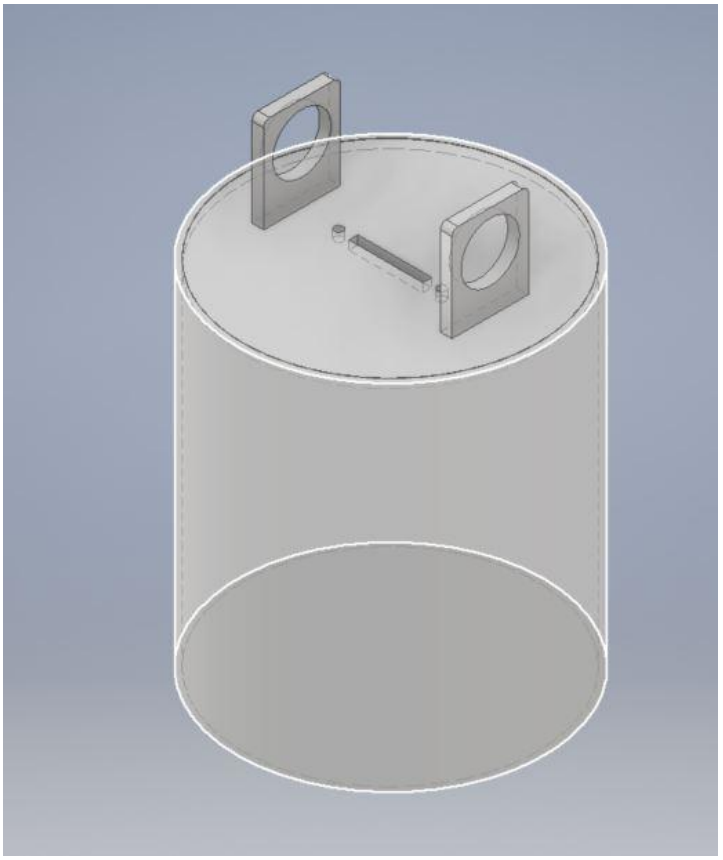

Imagen XXX: gabinete pieza 1

4.3.3.2 Pieza N°2

La pieza N°2 es la que se encontrará en contacto con el conductor de corriente en media tensión. En la parte frontal dispone de dos espacios diseñados para colocar los ganchos que permitirán mantener el dispositivo “agarrado” al conductor. En la parte posterior cuenta con el espacio adecuado para colocar el soporte del sensor inductivo de corriente.

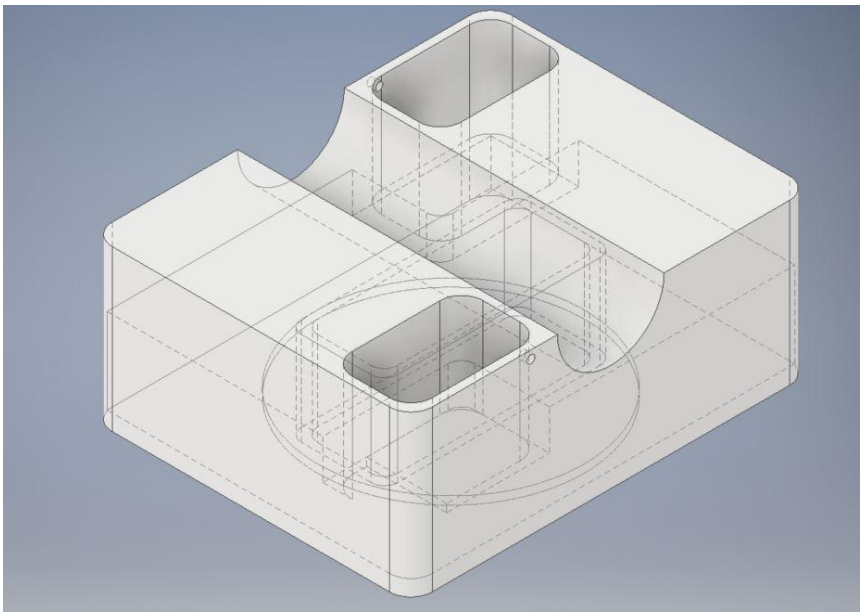

Imagen XXXI: gabinete pieza 2 vista superior

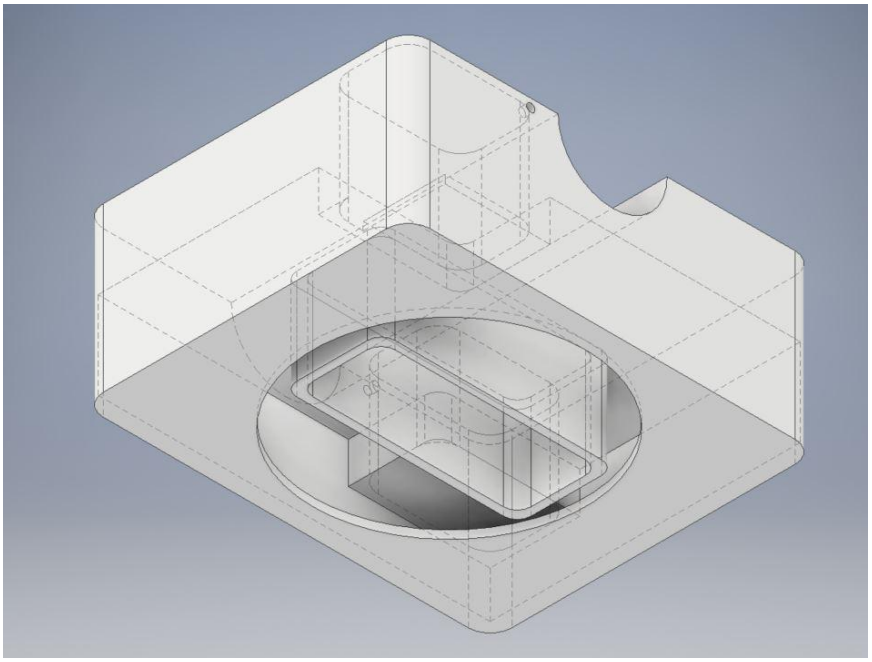

Imagen XXXI: gabinete pieza 2 vista inferior

4.3.3.3 Pieza N°3

La pieza N°3 es la parte inferior del gabinete. Cuenta con orificios que permiten el drenaje de agua generada por posible condensación del aire húmedo dentro del dispositivo, reduciendo la probabilidad de generar fallas internas. Además, posee un soporte para facilitar el montaje sobre la línea.

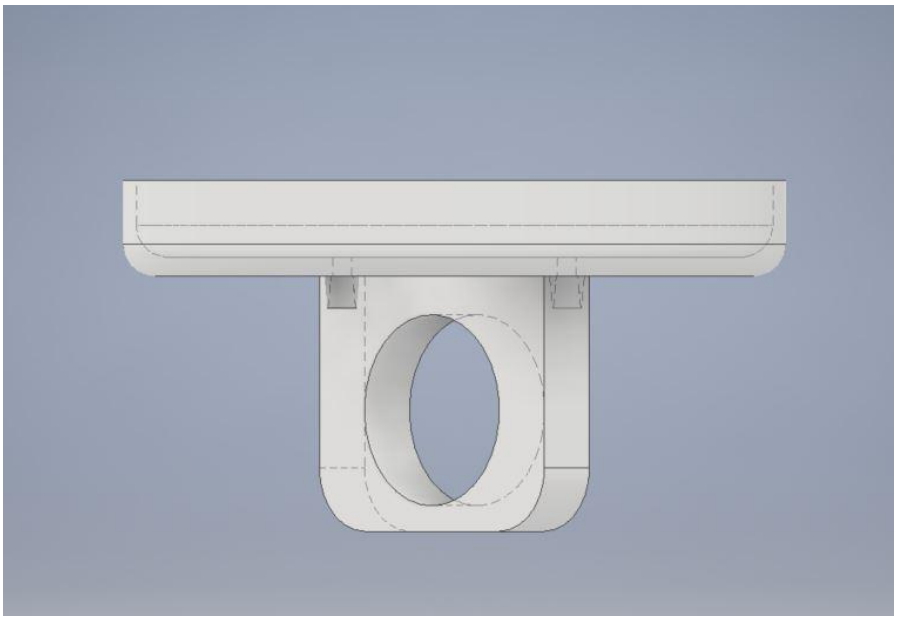

Imagen XXXIII: gabinete pieza 3

4.3.3.4 Pieza N°4

La pieza N°4 corresponde a los ganchos que permiten el agarre del dispositivo al conductor eléctrico. De esta pieza se requieren dos unidades.

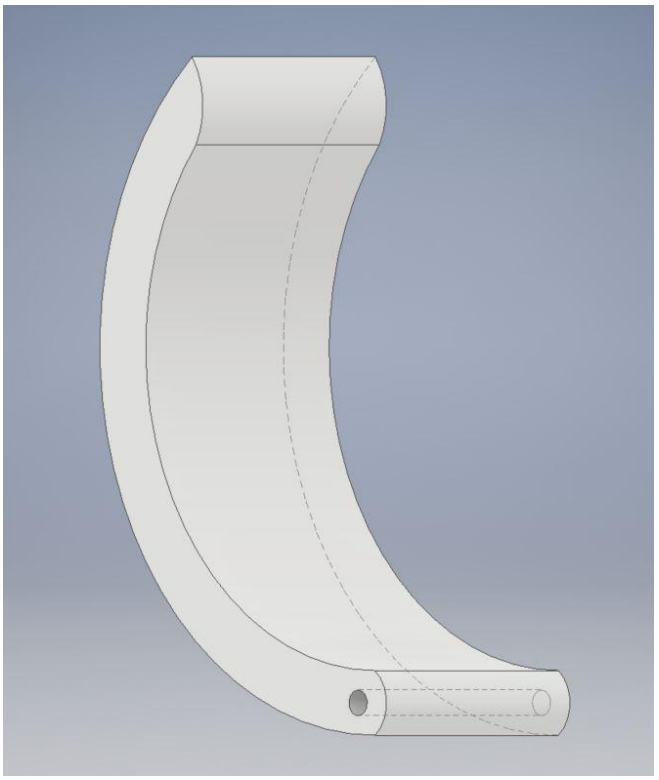

Imagen XXXIV: gabinete pieza 4

Se adjuntan imágenes del ensamblaje de las piezas dando origen al gabinete final del dispositivo.

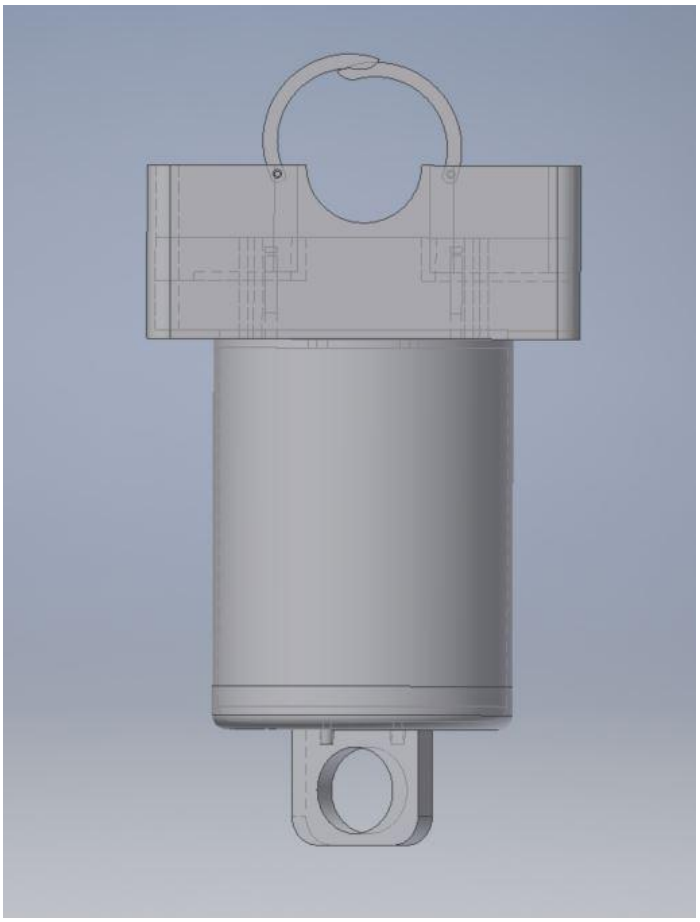

Imagen XXXV: gabinete ensamblaje final vista lateral

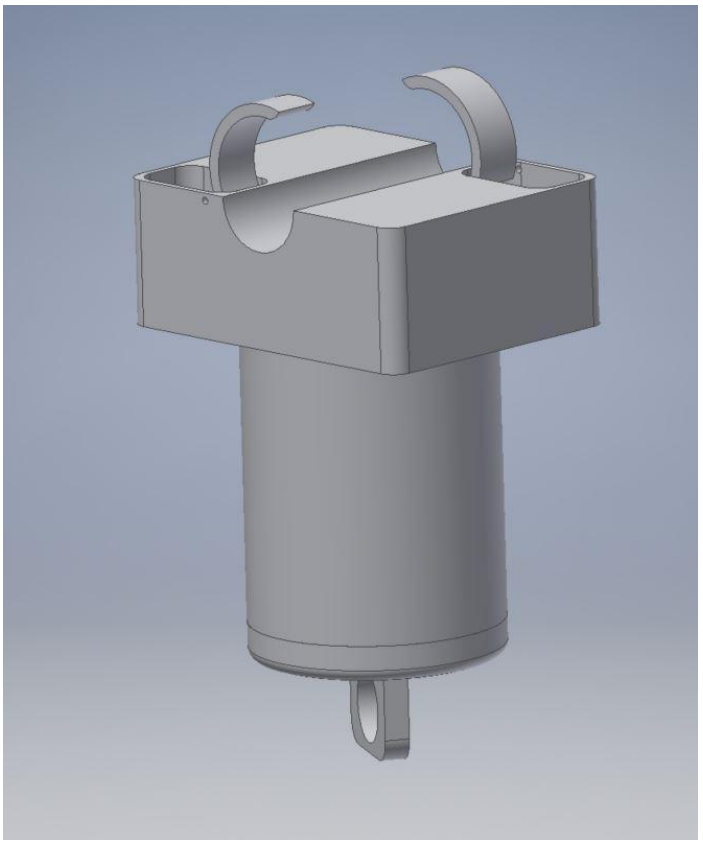

Imagen XXXVI: gabinete ensamblaje final vista lateral superior

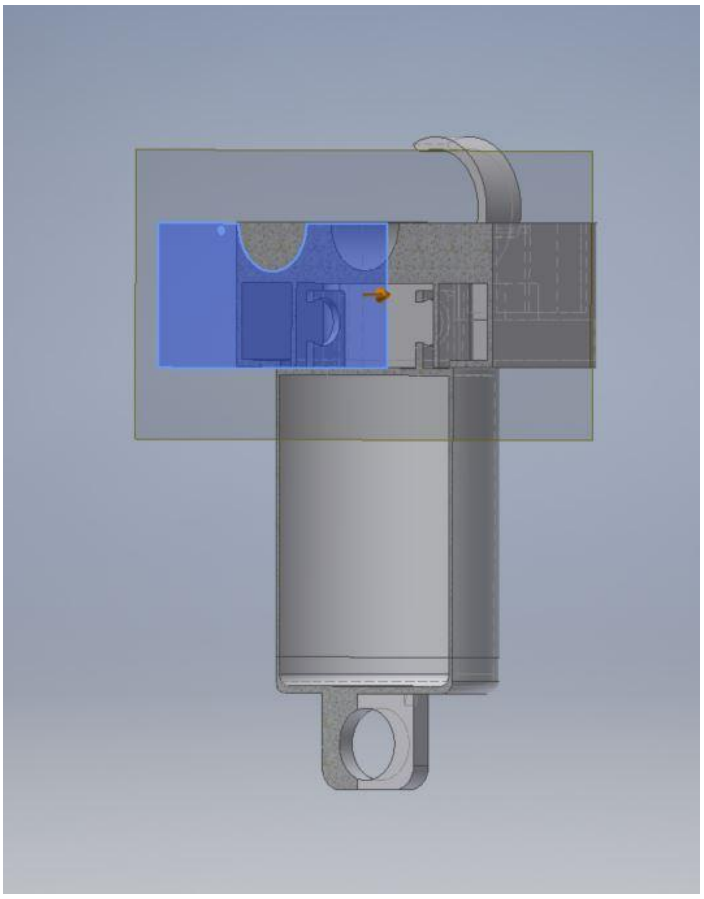

Imagen XXXVII: gabinete ensamblaje final vista lateral en corte

4.4 Aplicación web y API Rest

Con el objetivo de brindar una solución llave en mano, se diseñó y desarrolló una API Rest que permite recuperar y almacenar los datos enviados por los dispositivos y una aplicación web que permite a los usuarios visualizar dichos datos. En la presente sección se detallan los aspectos técnicos y funcionales de las mismas.

4.4.1 Requerimientos del sistema

En función del dispositivo desarrollado, se plantearon los siguientes requerimientos para el sistema:

- Geolocalización del dispositivo: la aplicación debe mostrar la información de la geolocalización del dispositivo.
- Mediciones de corriente: la aplicación debe mostrar los valores de corriente medidos por el dispositivo.
- Mediciones de oscilación: la aplicación debe mostrar los valores de oscilación medidos por el dispositivo.
- Alertas de alto consumo: la aplicación debe mostrar las alertas generadas por el dispositivo por alto consumo.
- Alertas de bajo consumo: la aplicación debe mostrar las alertas generadas por el dispositivo por bajo consumo.
- Alertas de oscilaciones: la aplicación debe mostrar las alertas generadas por el dispositivo por oscilaciones elevadas.

Como se mencionó previamente, se diseñó y desarrolló una aplicación web con el fin de mostrar la información transmitida por el dispositivo al usuario final.

4.4.2 Clases del sistema

La programación orientada a objetos es un modelo de programación que utiliza objetos para representar entidades del negocio que se desea modelar. Cada objeto dispone de atributos, características propias de este, y comportamientos, que representan aquellas funciones que puede realizar.

Con el objeto de satisfacer los requerimientos planteados, se construyó el diagrama clases que modela los diferentes objetos y controladores que forman parte del sistema. Este diseño se utiliza en la construcción de la aplicación a través del framework utilizado. En la siguiente imagen, se presenta dicho diagrama.

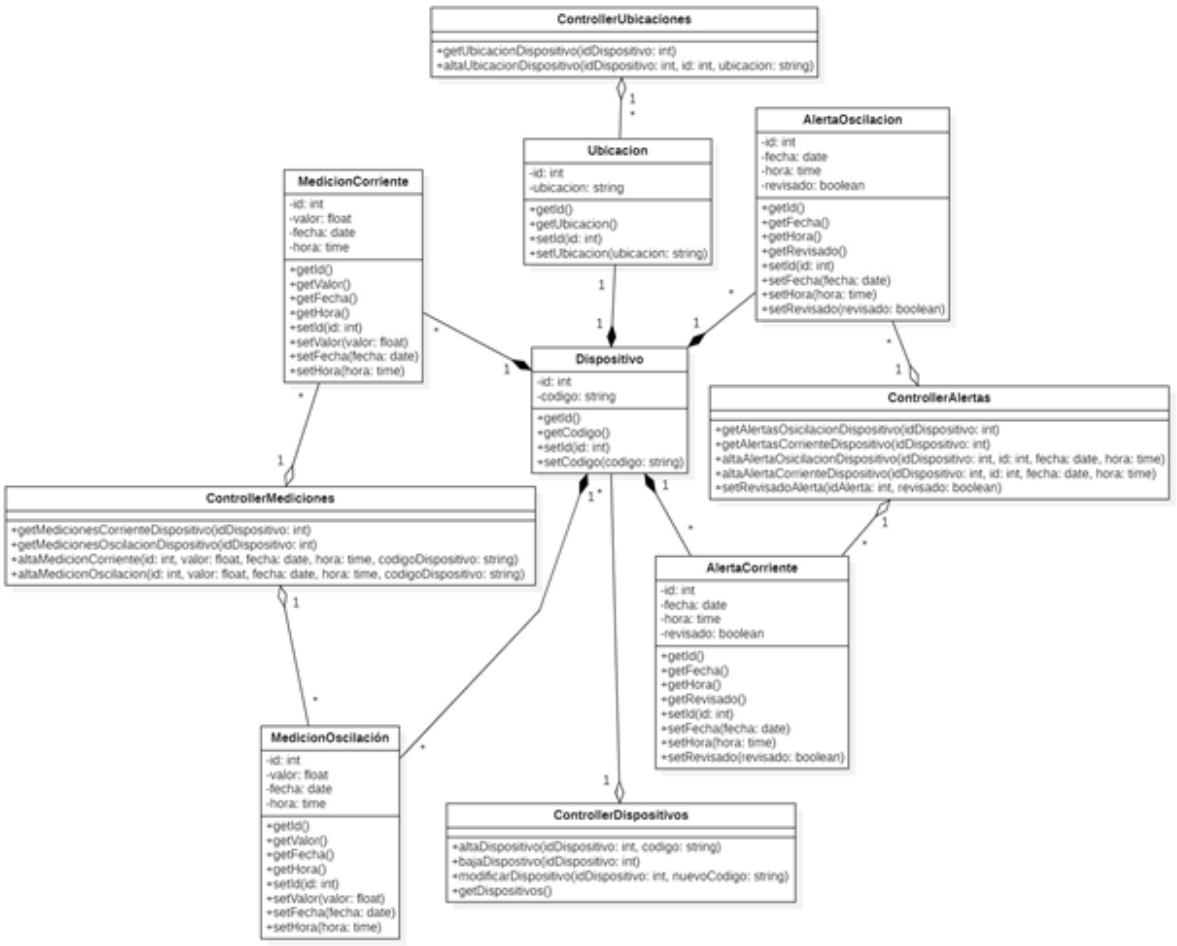

Imagen XXXVIII: diagrama de clases del sistema

A continuación, se detallan cada una de las clases incluidas en el diagrama teniendo en cuenta sus atributos y métodos:

- Dispositivo: la clase permitirá instanciar un dispositivo. La misma cuenta con los siguientes atributos:
    - Id: número de identificación único para cada dispositivo.
    - Código: cadena de caracteres única para cada dispositivo.

  Por otro lado, la clase mencionada cuenta con los siguientes métodos:

    - getId(): permite recuperar el id del objeto instanciado.

- getCodigo(): permite recuperar el código del objeto instanciado.
- setId(): permite establecer el id de un objeto instanciado.
- setCodigo(): permite establecer el código de un objeto instanciado.

- Ubicación: la clase permitirá instanciar la ubicación de un dispositivo. Cabe destacar que la misma cuenta con una relación de composición y una multiplicidad 1:1 con la clase Dispositivo. Esta clase cuenta con los siguientes atributos:
  - Id: número de identificación único para cada ubicación.
  - Código: cadena de caracteres que representa la ubicación del dispositivo.

  Por otro lado, la clase mencionada cuenta con los siguientes métodos:

  - getId(): permite recuperar el id del objeto instanciado.
  - getUbicacion(): permite recuperar el valor de la ubicación del objeto instanciado.
  - setId(): permite establecer el id de un objeto instanciado.
  - setCodigo(): permite establecer el valor de la ubicación de un objeto instanciado.
- MedicionCorriente: la clase permitirá instanciar una medición de corriente correspondiente a un dispositivo. Cabe destacar que la misma cuenta con una relación de composición y una multiplicidad *:1 con la clase Dispositivo ya que un dispositivo puede contar con varias mediciones, pero cada medición corresponde a un único dispositivo. Esta clase cuenta con los siguientes atributos:
  - Id: número de identificación único para cada medición de corriente.
  - Valor: valor flotante correspondiente a la medición de corriente.
  - Fecha: valor de la fecha en la cual se realizó la medición.
  - Time: hora en la cual se tomó la medición.

  Por otro lado, la clase mencionada cuenta con los siguientes métodos:

  - getId(): permite recuperar el id del objeto instanciado.
  - getValor(): permite recuperar el valor de la medición instanciada.
  - getFecha(): permite recuperar el valor de la fecha del objeto instanciado.

- getTime(): permite recuperar el valor de la hora del objeto instanciado.
- setId(): permite establecer el id de un objeto instanciado.
- setValor(): permite establecer el valor de la medición instanciada.
- setFecha(): permite establecer el valor de la fecha del objeto instanciado.
- setTime(): permite establecer el valor de la hora del objeto instanciado.

- MedicionOscilacion: la clase permitirá instanciar una medición de oscilación correspondiente a un dispositivo. Cabe destacar que la misma cuenta con una relación de composición y una multiplicidad *:1 con la clase Dispositivo ya que un dispositivo puede contar con varias mediciones, pero cada medición corresponde a un único dispositivo. Esta clase cuenta con los siguientes atributos:
  - Id: número de identificación único para cada medición de oscilación.
  - Valor: valor flotante correspondiente a la medición de oscilación.
  - Fecha: valor de la fecha en la cual se realizó la medición.
  - Time: hora en la cual se tomó la medición.

  Por otro lado, la clase mencionada cuenta con los siguientes métodos:

  - getId(): permite recuperar el id del objeto instanciado.
  - getValor(): permite recuperar el valor de la medición instanciada.
  - getFecha(): permite recuperar el valor de la fecha del objeto instanciado.
  - getTime(): permite recuperar el valor de la hora del objeto instanciado.
  - setId(): permite establecer el id de un objeto instanciado.
  - setValor(): permite establecer el valor de la medición instanciada.
  - setFecha(): permite establecer el valor de la fecha del objeto instanciado.
  - setTime(): permite establecer el valor de la hora del objeto instanciado.
- AlertaCorriente: la clase permitirá instanciar una alerta de corriente correspondiente a un dispositivo. Cabe destacar que la misma cuenta con una relación de composición y una multiplicidad *:1 con la clase Dispositivo ya que un dispositivo puede contar con varias alertas, pero cada alerta corresponde a un único dispositivo. Esta clase cuenta con los siguientes atributos:

- Id: número de identificación único para cada alerta de corriente.
- Fecha: valor de la fecha en la cual se generó la alerta.
- Time: hora en la cual se generó la alerta.
- Revisado: atributo booleano que establece si la alerta fue revisada o no por el usuario de la aplicación.

Por otro lado, la clase mencionada cuenta con los siguientes métodos:

- getId(): permite recuperar el id del objeto instanciado.
- getFecha(): permite recuperar el valor de la fecha del objeto instanciado.
- getTime(): permite recuperar el valor de la hora del objeto instanciado.
- getRevisado(): permite recuperar el valor del atributo 'revisado' del objeto instanciado.
- setId(): permite establecer el id de un objeto instanciado.
- setFecha(): permite establecer el valor de la fecha del objeto instanciado.
- setTime(): permite establecer el valor de la hora del objeto instanciado.
- setRevisado(): permite establecer el valor del atributo 'revisado' del objeto instanciado.

- AlertaOscilacion: la clase permitirá instanciar una alerta de oscilación correspondiente a un dispositivo. Cabe destacar que la misma cuenta con una relación de composición y una multiplicidad *:1 con la clase Dispositivo ya que un dispositivo puede contar con varias alertas, pero cada alerta corresponde a un único dispositivo. Esta clase cuenta con los siguientes atributos:
  - Id: número de identificación único para cada alerta de oscilación.
  - Fecha: valor de la fecha en la cual se generó la alerta.
  - Time: hora en la cual se generó la alerta.
  - Revisado: atributo booleano que establece si la alerta fue revisada o no por el usuario de la aplicación.

  Por otro lado, la clase mencionada cuenta con los siguientes métodos:

  - getId(): permite recuperar el id del objeto instanciado.

- getFecha(): permite recuperar el valor de la fecha del objeto instanciado.
- getTime(): permite recuperar el valor de la hora del objeto instanciado.
- getRevisado(): permite recuperar el valor del atributo 'revisado' del objeto instanciado.
- setId(): permite establecer el id de un objeto instanciado.
- setFecha(): permite establecer el valor de la fecha del objeto instanciado.
- setTime(): permite establecer el valor de la hora del objeto instanciado.
- setRevisado(): permite establecer el valor del atributo 'revisado' del objeto instanciado.

Las clases descritas previamente son aquellas que forman parte del modelo de datos. Por otro lado, el sistema contará con diferentes controladores que servirán de enlace entre las vistas y los modelos, con el fin de responder los diferentes requerimientos del sistema. Por este motivo, los controladores no cuentan con atributos propios, sino que disponen de métodos que permiten interactuar con las clases pertenecientes al modelo. A continuación, se detallan los diferentes controladores.

- ControllerDispositivos: es el controlador le da al sistema la capacidad de interactuar con los diferentes dispositivos. Es importante destacar que dicho controlador posee una relación de agregación y una multiplicidad 1:* con la clase 'Dispositivo', ya que él mismo maneja todos los dispositivos del sistema. Los métodos que posee son:
  - altaDispositivo(): método que recibe como parámetro el id y el código del nuevo dispositivo y genera el alta en el sistema.
  - bajaDispositivo(): método que recibe como parámetro el id del dispositivo y genera la baja en el sistema.
  - modificarDispositivo(): método que recibe como parámetro el id del dispositivo a modificar y el nuevo valor del código asignado generando la modificación en el sistema.
  - getDispositivos(): método que devuelve en una lista todos los dispositivos del sistema.

- ControllerUbicaciones: es el controlador le da al sistema la capacidad de interactuar con las ubicaciones correspondientes a los diferentes dispositivos. Es importante destacar que dicho controlador posee una relación de agregación y una multiplicidad 1:* con la clase ‘Ubicación’, ya que el mismo maneja todas las ubicaciones registradas en el sistema. Los métodos que posee son:
    - altaUbicacion(): método que recibe como parámetro el id del dispositivo al cual corresponde, el id y el valor de la ubicación y genera el alta en el sistema.
    - getUbicacionDispositivo(): método que recibe como parámetro el id del dispositivo y devuelve la ubicación de este.

- ControllerMediciones: es el controlador le da al sistema la capacidad de interactuar con las diferentes mediciones, tanto de corriente como de oscilación. Es importante destacar que dicho controlador posee una relación de agregación y una multiplicidad 1:* con las clases ‘MedicionCorriente’ y ‘MedicionOscilacion’, ya que el mismo maneja todas las mediciones del sistema. Los métodos que posee son:
    - altaMedicionCorriente(): método que recibe como parámetro el id, el valor, la fecha y la hora de una nueva medición de corriente y genera el alta en el sistema.
    - altaMedicionOscilacion(): método que recibe como parámetro el id, el valor, la fecha y la hora de una nueva medición de oscilación y genera el alta en el sistema.
    - getMedicionesCorriente(): método que devuelve una lista con todas las mediciones de corriente realizadas por un dispositivo y registradas en el sistema.
    - getMedicionesOscilacion(): método que devuelve una lista con todas las mediciones de oscilación realizadas por un dispositivo y registradas en el sistema.

- ControllerAlertas: es el controlador le da al sistema la capacidad de interactuar con las diferentes alertas, tanto de corriente como de oscilación. Es importante destacar que dicho controlador posee una relación de agregación y una multiplicidad 1:* con las clases 'AlertaCorriente' y 'AlertaOscilacion', ya que el mismo maneja todas las alertas del sistema. Los métodos que posee son:
    - altaAlertaCorriente(): método que recibe como parámetro el id, la fecha, la hora y el valor revisado de una nueva alerta de corriente y genera el alta en el sistema.
    - altaAlertaOscilacion(): método que recibe como parámetro el id, la fecha, la hora y el valor revisado de una nueva alerta de oscilación y genera el alta en el sistema.
    - getAlertasCorriente(): método que devuelve una lista con todas las alertas de corriente realizadas por un dispositivo y registradas en el sistema.
    - getAlertasOscilacion(): método que devuelve una lista con todas las alertas de oscilación realizadas por un dispositivo y registradas en el sistema.
    - setRevisadoAlerta(): método que recibe como parámetro el id y el nuevo valor del atributo 'revisado' de una alerta y lo modifica en el sistema.

4.4.3 Estructura de datos

Tal como se mencionó en la sección 4.3.3 del presente documento, se utiliza MySQL como base de datos para efectuar la persistencia de estos. En esta sección se detalla la estructura definida para el almacenamiento de los datos necesarios.

4.4.3.1 Diagrama entidad – relación

El diagrama entidad-relación permite modelar en primera instancia los datos que maneja el sistema con el fin de obtener una primera aproximación al modelado de la base de datos requerida. En dicho diagrama se exponen los siguientes conceptos:

- Entidad: representa un objeto o concepto del mundo real con existencia propia que lo diferencia de los demás.
- Atributo: hace referencia a una característica que posee una entidad. Cada entidad tiene sus propios atributos.
- Relación: corresponde a los vínculos existentes entre las entidades.
- Cardinalidad: dada una relación que vincula dos entidades, la cardinalidad se refiere a la cantidad de entidades de un tipo que se relacionan con una entidad dada de otro tipo. Los tipos de cardinalidades son:
    - Uno a uno (1:1): un registro de la entidad A se relaciona con un único registro de la entidad B.
    - Uno a muchos (1:N): un registro de la entidad A se relaciona con muchos registros de la entidad B.
    - Muchos a uno (N:1) muchos registros de la entidad A se relacionan con un único registro de la entidad B.
    - Muchos a muchos (N:N): muchos registros de la entidad A se relacionan con muchos registros de la entidad B.

Con el objeto de modelar los datos de la aplicación, se elaboró el diagrama entidad-relación en el cual se exponen las diferentes entidades que forman parte del sistema, sus atributos y sus relaciones.

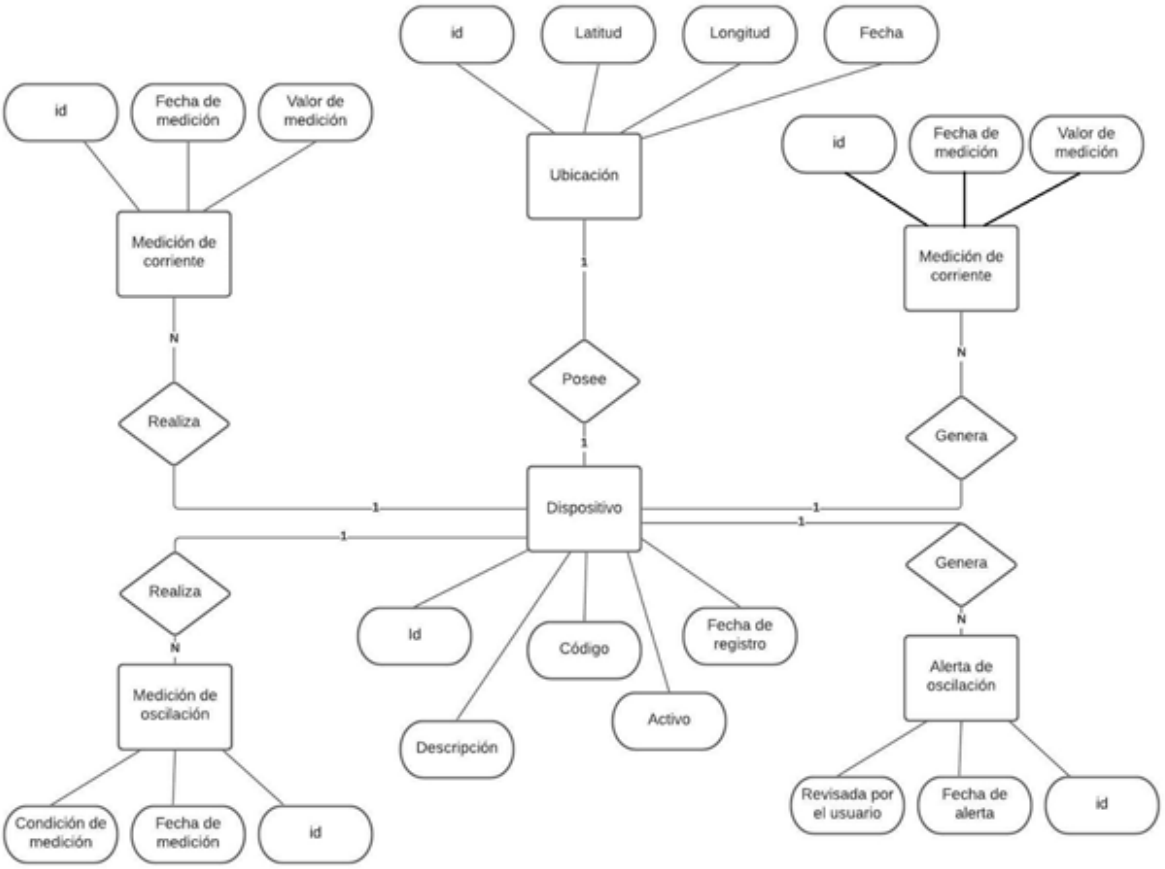

Imagen XXXIX: diagrama entidad-relación del sistema

4.4.3.2 Modelo lógico

A partir del diagrama entidad-relación, se generó el modelo lógico el cual define la construcción de la base de datos que utiliza el sistema. El paso de un modelo conceptual a un modelo lógico se realiza efectuando los siguientes pasos:

1. Convertir entidades en tablas.
2. Convertir relaciones en claves externas.
3. Convertir atributos en columnas.
4. Normalización.

En conclusión, el modelo lógico utilizado se encuentra representado por el siguiente diagrama.

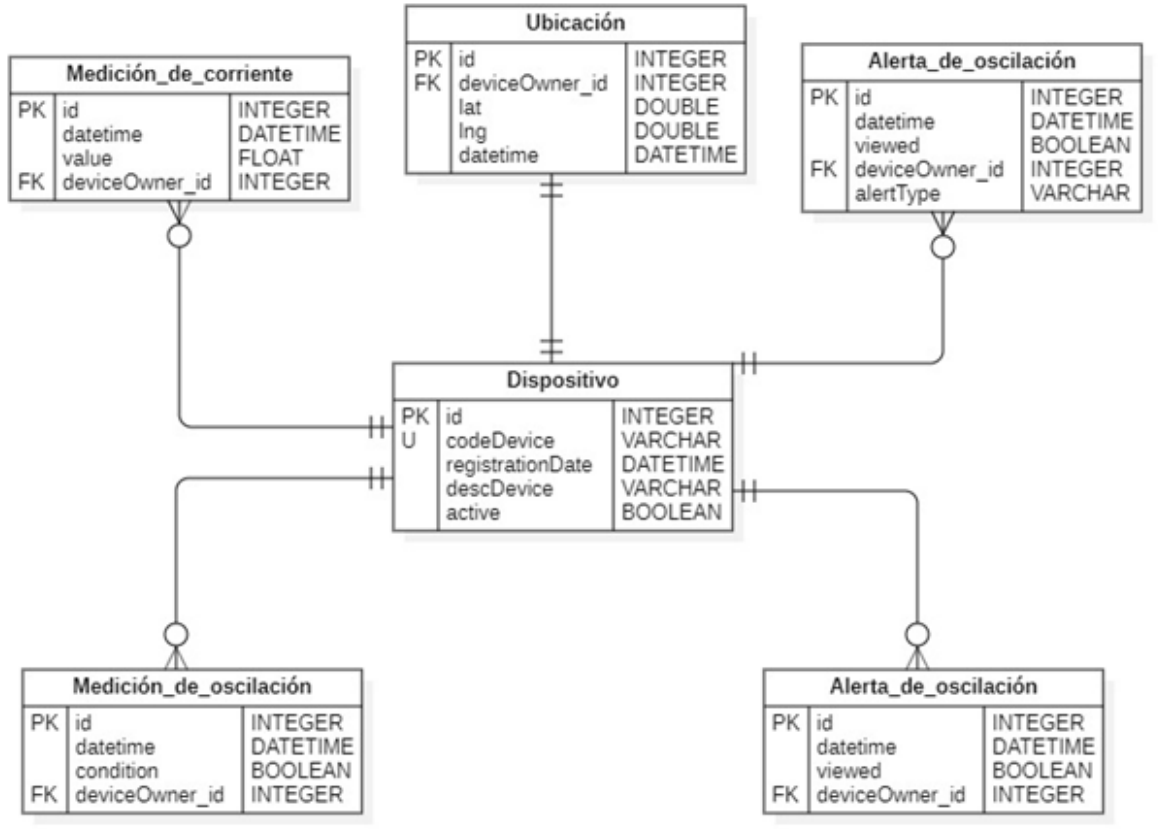

Imagen XL: esquema lógico del sistema

4.4.4 API Rest para el almacenamiento de datos

Como se mencionó previamente, el desarrollo del presente proyecto no incluye el montaje de una red de LoRaWAN, por lo que se utiliza aquella provista por un proveedor externo. Los servicios brindados por dicho proveedor consisten en la captación de los mensajes transmitidos por el dispositivo y su almacenamiento en una base de datos externa al sistema. El acceso a dichos datos se realiza a través de una aplicación web provista por el proveedor la cual, a su vez, puede ser configurada para retransmitir los mensajes recibidos a una dirección de ip y puerto determinado a través de internet.

Para independizar el sistema desarrollado de la red de LoRaWAN brindada por el proveedor externo, se desarrolló una API Rest que se encuentra funcionando en un servidor externo y se encarga de recibir y almacenar los datos enviados por la aplicación del proveedor de red. Es importante destacar que dicha API Rest consiste en un servicio para la comunicación externa del sistema con la aplicación que dispone el proveedor de red y la única acción que puede realizar es la de generar nuevos registros en la base de datos. De esta forma, los datos correspondientes al dispositivo quedan almacenados en un servidor propio y no se requiere el uso de una aplicación desarrollada por terceros para acceder a los mismos. Por otro lado, en caso de que el usuario final despliegue su propia red LoRaWAN, los Gateway de esta deberán enviar los datos a almacenar a la dirección de ip y puerto correspondientes al servidor en el cual se encuentra corriendo el servicio el cual se encargará de su procesamiento y almacenamiento.

En el caso de la API desarrollada, la misma cuenta con un único endpoint disponible del tipo POST. Dicho endpoint espera recibir en el body de la petición información sobre el dispositivo que envía el mensaje y el tipo de mensaje recibido. Dicho body debe responder a la siguiente estructura en formato JSON:

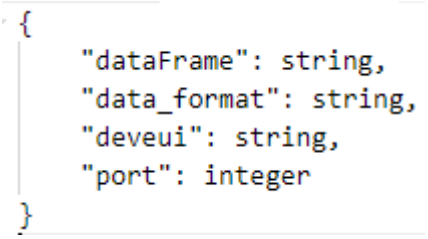

Imagen XLI: estructura del body requerido por el endpoint

- El campo “deveui” contiene el código de dispositivo que ha enviado el mensaje. Es importante destacar que el servicio únicamente procesara y almacenara el mensaje en caso de que dicho código haya sido registrado en el sistema previamente. En caso contrario, se descarta el mensaje.
- El campo “port” indica el puerto en el cual el dispositivo envía la transmisión. El valor del puerto permite distinguir el tipo de mensaje se ha enviado. En caso de que el valor

del puerto sea 1, se considera que el dato enviado por el dispositivo corresponde a su ubicación. Por otro lado, en caso de que el valor del puerto sea 2, se considera que el mensaje enviado por el dispositivo corresponde a una medición de corriente. Por último, si el valor del puerto es 3, el mensaje corresponde a una alerta generada por el dispositivo.

- El campo “data_format” indica el sistema de codificación utilizado en el campo “dataFrame”.
- El campo “dataFrame” indica el valor del dato enviado por el dispositivo. En caso de que dicho mensaje tenga como valor de puerto 1, el valor del campo “dataFrame” estará formado por los valores de latitud y longitud correspondientes a la ubicación del dispositivo obtenidos a través del módulo GPS. En caso de que el mensaje tenga 2 como valor de puerto, el valor del campo “dataFrame” corresponderá al valor de la medición de corriente efectuada por el dispositivo. Por último, si el valor del puerto es 3, el valor del campo “dataFrame” variará entre las siguientes opciones:
    - “A”: indica que el mensaje enviado corresponde a una alerta de alto consumo de corriente.
    - “B”: indica que el mensaje enviado corresponde a una alerta de bajo consumo de corriente.
    - “C”: indica que el mensaje enviado corresponde a una alerta de oscilación.

La API Rest fue desarrollada utilizando Fask como framework y presenta la siguiente estructura de archivos y directorios:

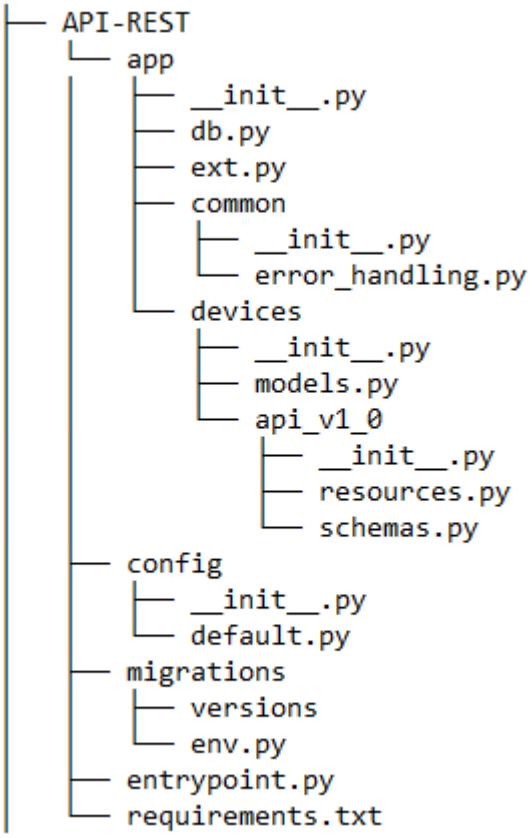

Imagen XLII: estructura de archivos y directorios de la API Rest desarrollada

A continuación, se detallan las funcionalidades de aquellos archivos de vital importancia para el funcionamiento de esta.

- db.py (/API-REST/app/): se definen las operaciones de guardado y lecturas necesarias para la base de datos.
- error_handling.py (/API-REST/app/common/): se definen las funciones de gestión de errores del servicio.
- models.py (/API-REST/app/devices/): se crean los modelos con los que trabaja el servicio. Es importante destacar que estos modelos coinciden con los creados para la aplicación web ya que utilizan la misma base de datos y, en consecuencia, las mismas entidades.

- resources.py (/API-REST/app/devices/api_v1_0/): se definen los endpoints que tiene la API Rest y las funciones que debe realizar cuando se recibe una petición en uno de ellos. En este caso, se tiene un único endpoint del tipo POST como se explicó previamente.
- schemas.py (/API-REST/app/devices/api_v1_0/): se define la estructura del mensaje JSON que se espera recibir .
- default.py (/API-REST/config/): se establecen los parámetros de conexión a la base de datos.
- env.py (/API-REST/migrations/): se definen las funciones de migración de modelos a la base de datos.
- entrypoint.py (/API-REST /): se define los parámetros de ejecución del servicio.
- requeriments.txt (/API-REST /): se definen las librerías utilizadas y versiones de estas.

4.4.5 Aplicación web

Como se mencionó previamente, la aplicación web le permite a los usuarios dar de alta dispositivos y visualizar mediciones y alertas de corriente y oscilación, como así también la ubicación de cada uno de dichos dispositivos. Tal como se detalla en la sección 4.3.5.1 del presente documento, el framework utilizado para el desarrollo de dicha aplicación es Django.

4.4.5.1 Estructura de la aplicación

En la presente sección se detalla la estructura de organización utilizada para la aplicación y cuál es la función de los archivos de vital importancia para su funcionamiento.

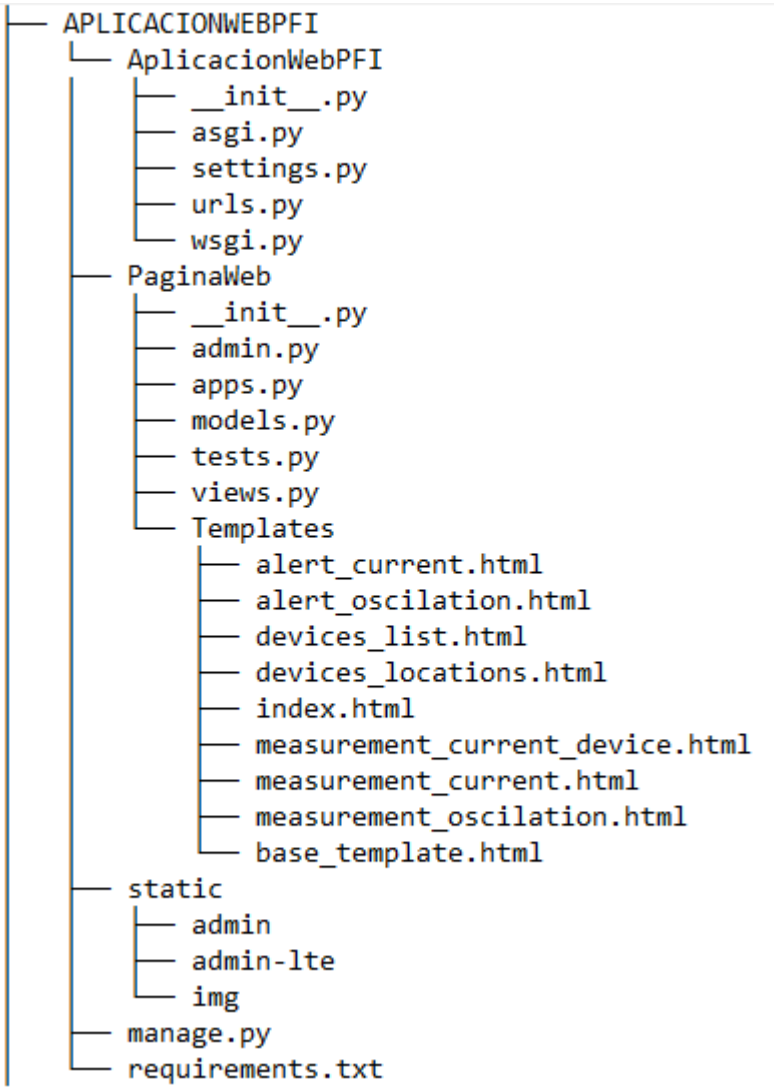

Imagen XLIII: estructura de archivos y directorios de la aplicación web desarrollada

A continuación, se describe el contenido y la funcionalidad de cada uno de los archivos existentes que son requeridos para el funcionamiento de la aplicación:

- settings.py (/AplicacionWebPFI/settings.py): en él se definen los siguientes parámetros de configuración necesarios para la aplicación:
    - SECRET_KEY: corresponde a la clave utilizada para elaborar los hashes necesarios para aquellos datos sensibles que deben ser encriptados por la aplicación.

    - ALLOWED_HOSTS: define los nombres de dominio habilitados para correr la aplicación.
    - INSTALLED_APPS: mapea las aplicaciones instaladas dentro del proyecto.
    - TEMPLATES: define, entre otros parámetros, la ubicación de los archivos HTML que corresponden a los templates de la aplicación.
    - DATABASES: establece los parámetros de conexión a la base de datos a utilizar (nombre, usuario, contraseña, host y puerto).
    - STATIC_URL: define la ubicación del directorio de los archivos estáticos de la aplicación.
- urls.py (/AplicacionWebPFI/settings.py): en el mismo se definen las rutas que posee la aplicación. Dichas rutas tienen definido un template determinado y un controlador que le comunica los datos necesarios al mismo.
- models.py (/PaginaWeb/models.py): en el mismo se definen los modelos con los que trabaja la aplicación. Cabe destacar que dichos modelos coinciden con los presentados en el modelo lógico de la misma ya que el framework genera automáticamente las tablas correspondientes en la base de datos.
- views.py (/PaginaWeb/views.py): en él se definen los controladores de la aplicación. Los mismos serán quienes interactúen entre los modelos y las vistas para darle funcionamiento a la aplicación.
- Templates (/PaginaWeb/Templates/): en este directorio se ubican todos los templates que forman parte de la aplicación y que serán visualizados por el usuario.

4.4.5.2 Uso de la aplicación

En la presente sección se detallan las diferentes pantallas con las que cuenta la aplicación y las funcionalidades que las mismas ofrecen para el usuario final.

Independientemente de la ruta a la que se acceda, el sistema mantiene dos características. Por un lado, en la esquina superior izquierda existe un menú de notificaciones donde el usuario puede observar de forma rápida e intuitiva cuantas alertas se han recibido y requieren de

atención. Por otro lado, en la esquina superior derecha se dispone de un botón que abre un menú lateral desplegable, dándole acceso al usuario a todas las secciones de la aplicación.

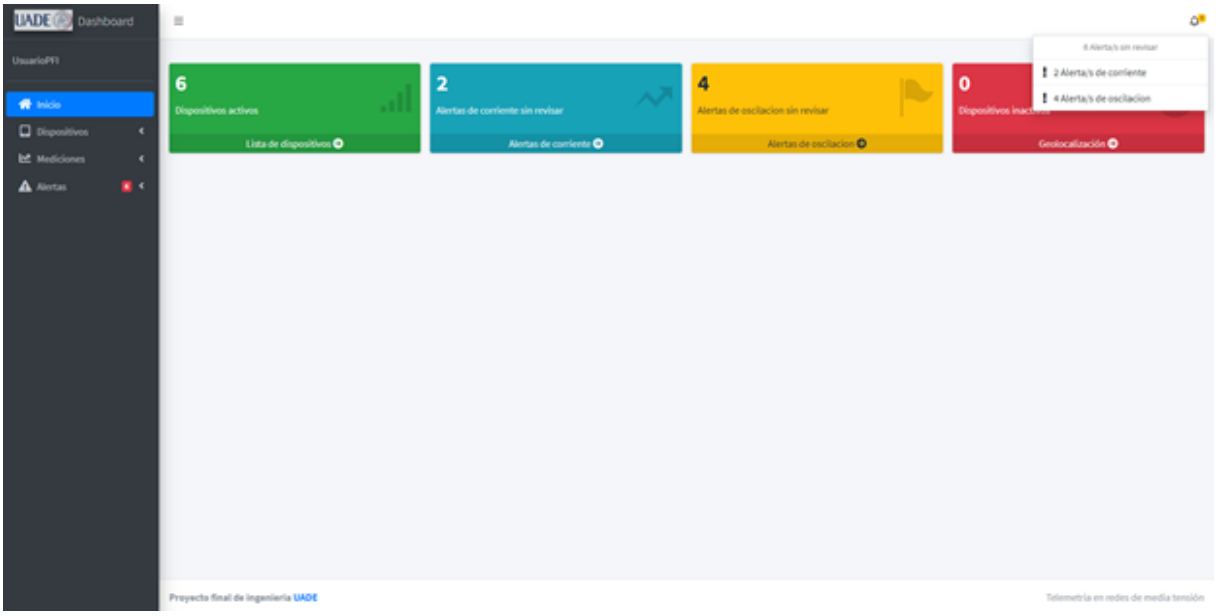

Imagen XLIV: pantalla de inicio de la aplicación web

Con respecto a la pantalla principal, el sistema presenta una vista de inicio accediendo a la ruta /index. En la misma puede visualizarse un resumen de la cantidad de dispositivos activos, alertas tanto de corriente como de oscilación que se encuentran pendientes de revisión y la cantidad de dispositivos inactivos, es decir, aquellos que fueron registrados en el sistema pero que aún no se recibieron datos.

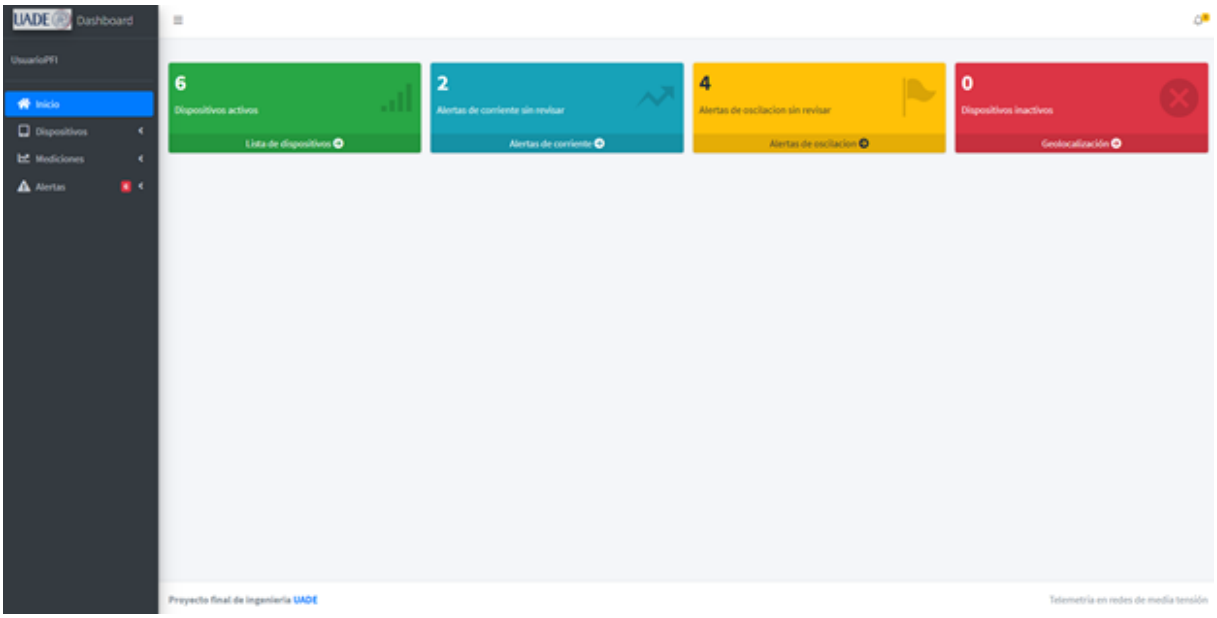

Imagen VL: pantalla de inicio de la aplicación web

Desde el menú desplegable, puede accederse a la lista de dispositivos cargados por el usuario en el sistema haciendo clic en Dispositivos/Lista de dispositivos, lo cual redirige al usuario a la ruta /devices_list. Desde esta pantalla, el usuario puede observar una lista que incluye todos los dispositivos que ha registrado. A su vez, dicha pantalla dispone de un ABM de dispositivos a través del cual el usuario puede dar de alta un nuevo dispositivo, eliminar o modificar un dispositivo existente .

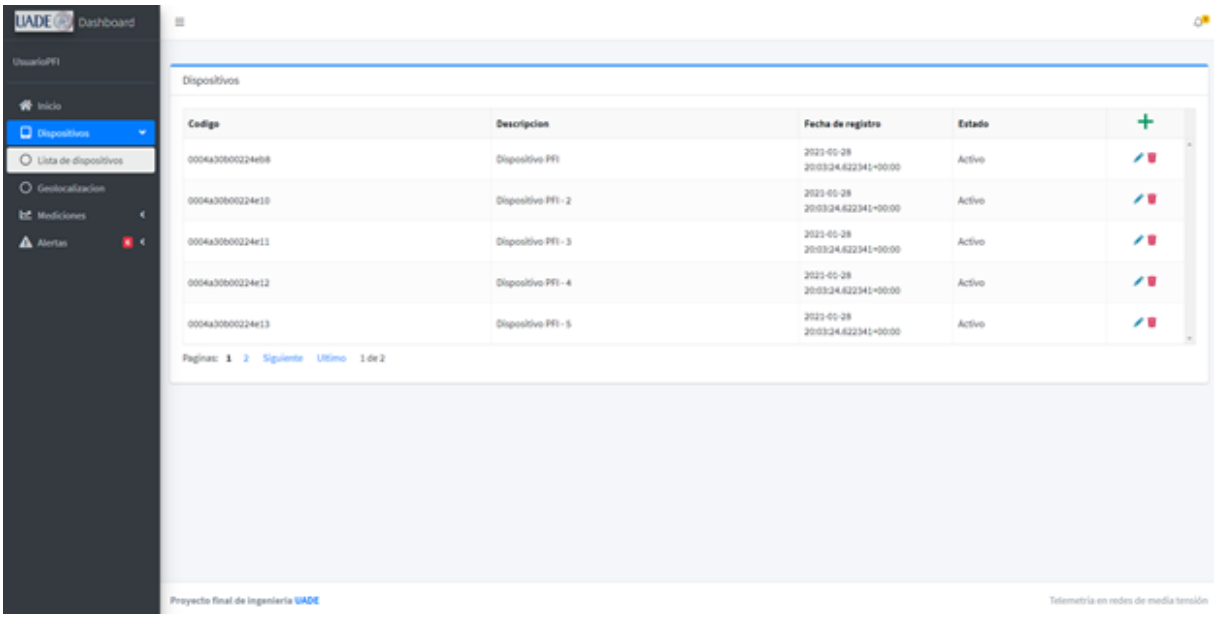

Imagen VLI: pantalla de listado de dispositivos de la aplicación web

Dentro del menú Dispositivos, el usuario puede acceder a la pantalla de geolocalización de estos. En la misma, ubicada en la ruta /devices_locations, se muestra un mapa con la ubicación de los dispositivos la cual queda es enviada por cada uno de ellos haciendo uso del módulo GPS integrado. Dependiendo del estado de cada dispositivo, su ubicación se mostrará con pines de diferentes colores.

- Verde: son aquellos dispositivos que no tienen alertas activas sin revisar.
- Azul: aquellos dispositivos que poseen alertas de corriente sin revisar.
- Amarillo: aquellos dispositivos que poseen alertas de oscilación sin revisar.
- Naranja: aquellos dispositivos que poseen alertas de corriente y de oscilación sin revisar.
- Violeta: aquellos dispositivos que fueron registrados en el sistema, pero no se han recibido transmisiones durante un periodo de tiempo superior a un día.

En caso de existir dispositivos activos que aún no han transmitido su ubicación, los mismos se informarán en el panel inferior derecho.

Cabe destacar que para informar la posición de cada dispositivo se utiliza la API de Google Maps la cual es de acceso gratuito y puede ser utilizada a través de JavaScript.

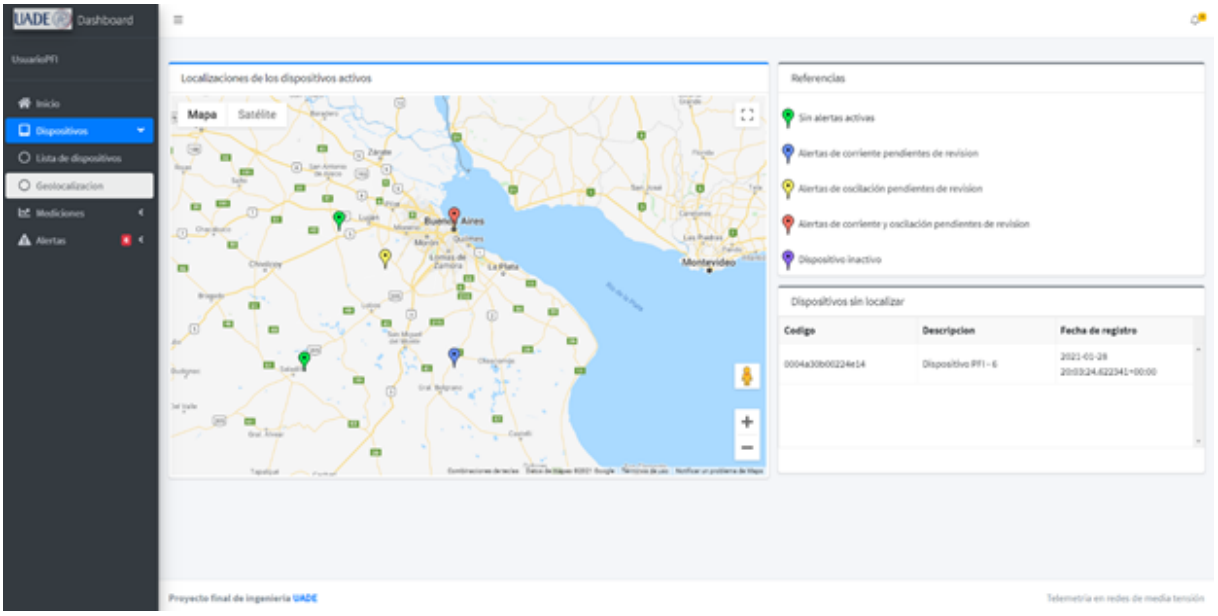

Imagen VLII: pantalla de geolocalización de dispositivos de la aplicación web

Desde el menú Mediciones, el usuario puede conocer las mediciones realizadas por los dispositivos registrados en el sistema.

En primer lugar, en la pantalla de Mediciones de corriente, ubicada en la ruta /measurement_current, se muestra una lista donde se brinda información del dispositivo que ha realizado la medición, la fecha, la hora y el valor medido.

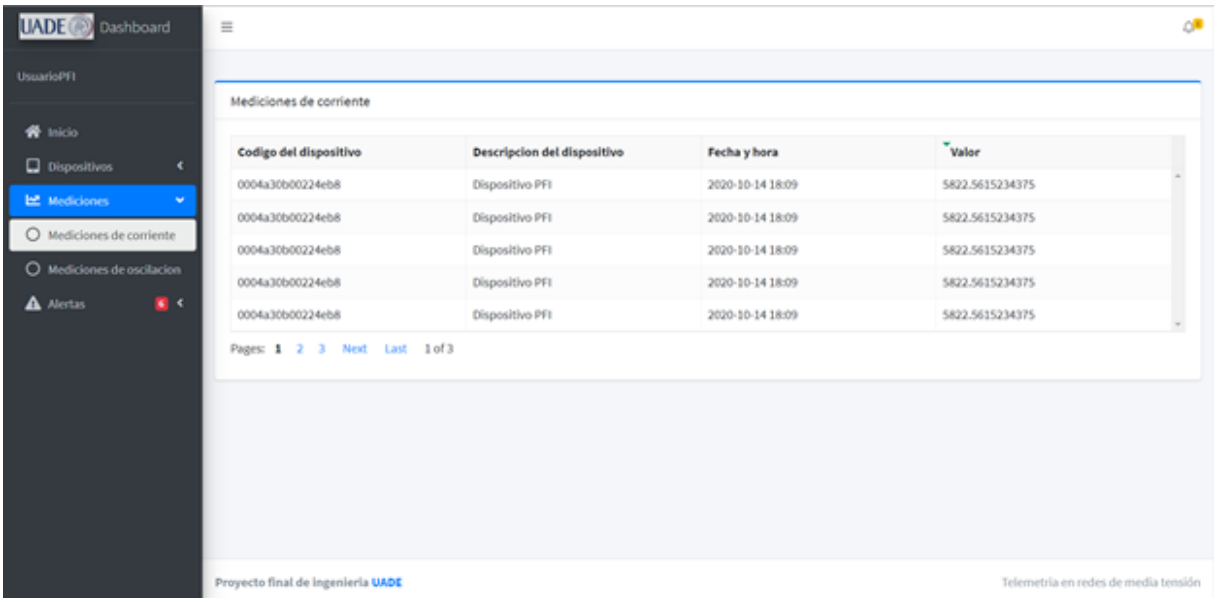

Imagen IIL: pantalla de mediciones de corriente de todos los dispositivos de la aplicación web

Para brindar una visión más organizada de estas mediciones, el usuario puede seleccionar el dispositivo que desee y el sistema mostrará gráficamente el registro de las últimas mediciones realizadas por el dispositivo seleccionado.

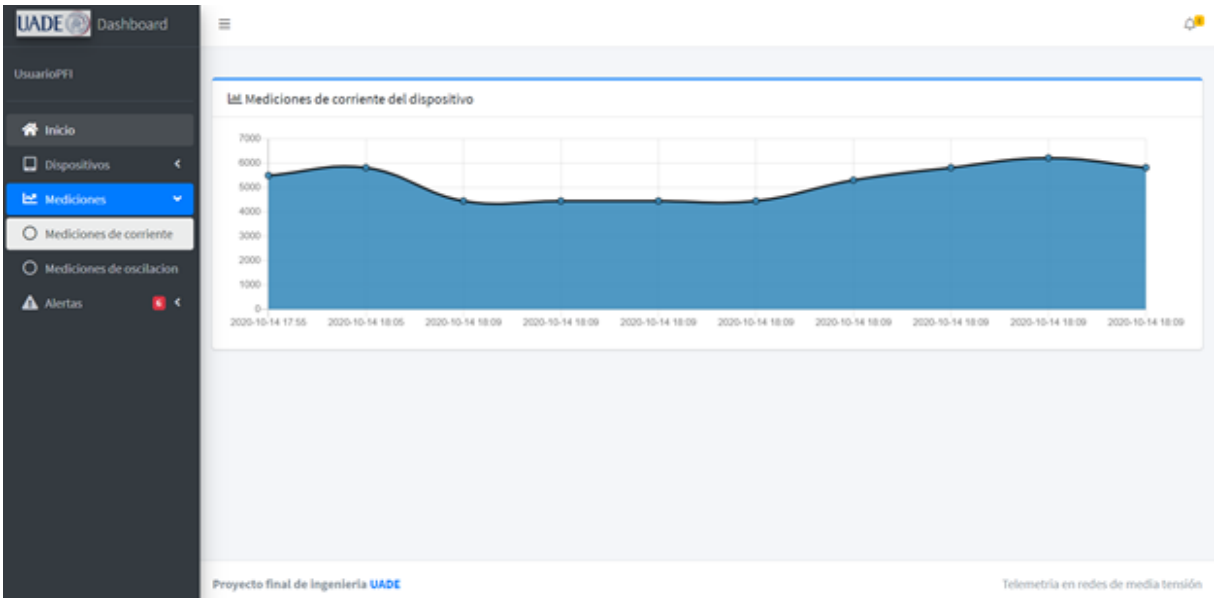

Imagen IL: pantalla de mediciones de corriente de un dispositivo de la aplicación web

Por otro lado, accediendo al menú "Mediciones de oscilación" el usuario será redirigido a la ruta /measurement_oscillation/ donde podrá observar el estado de oscilación del dispositivo. En caso de que no exista una oscilación elevada, el estado de este se mostrará como "Ok".

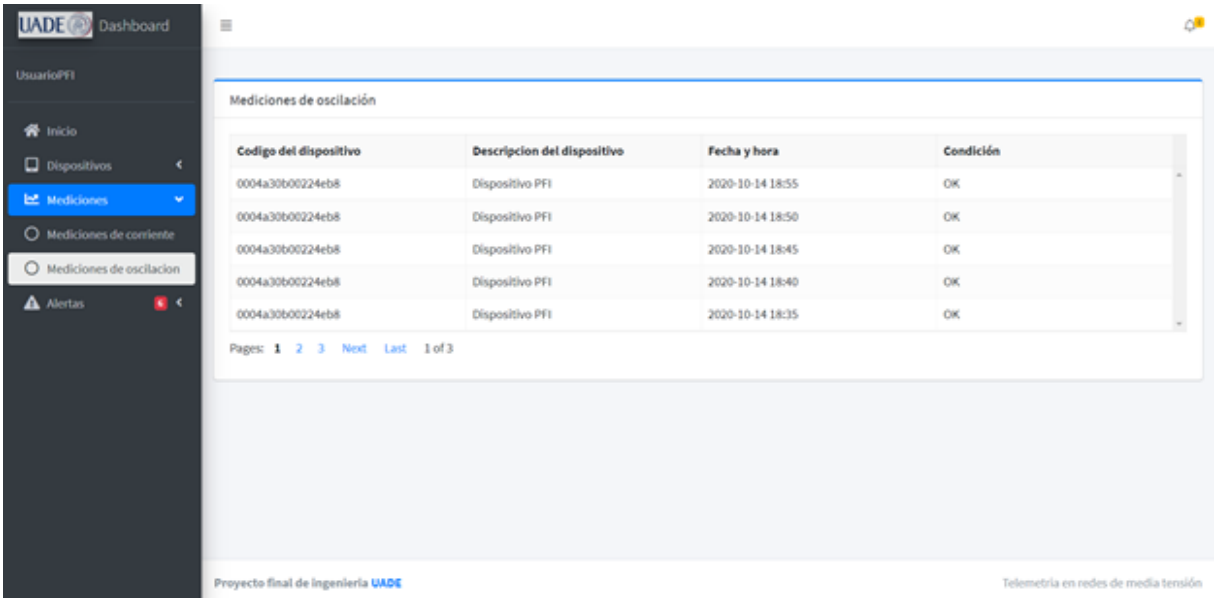

Imagen L: pantalla de mediciones de oscilación de todos los dispositivos de la aplicación web

Desde el menú Alertas, el usuario puede acceder tanto a la lista de alertas de corriente como a las de oscilación generadas por cada dispositivo registrado en el sistema.

Con respecto a las alertas de corriente, las mismas se muestran en la ruta /alert_current y le brindan información al usuario sobre el dispositivo que generó dicha alerta, fecha y hora en que se produjo la misma y el tipo de alerta la cual puede variar entre alto y bajo consumo. A su vez, el usuario puede marcar una alerta determinada como revisada, lo cual eliminará la notificación de esta.

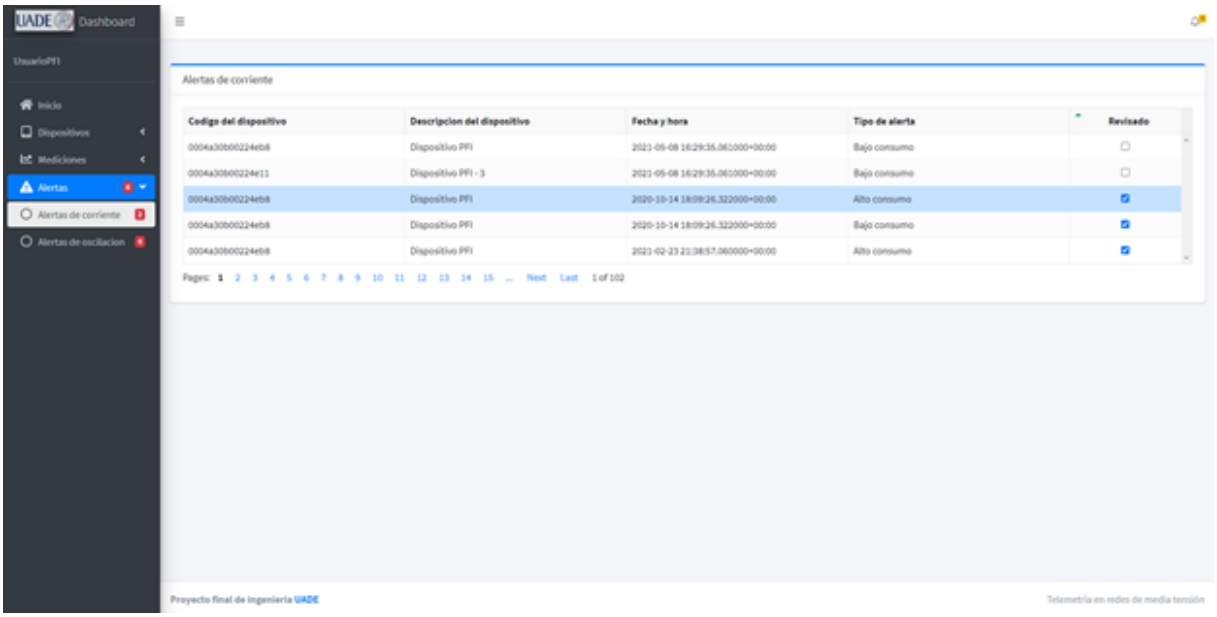

Imagen LI: pantalla de alertas de corriente generadas por todos los dispositivos de la aplicación web

Por último, la lista de alertas de oscilación puede visualizarse accediendo al menú correspondiente el cual redirige al usuario a la ruta /alert_oscillation/. En esta vista se brinda información sobre el dispositivo que generó la alerta, fecha y hora en que fue

registrada y, al igual que en las alertas de corriente, el estado de revisión de esta.

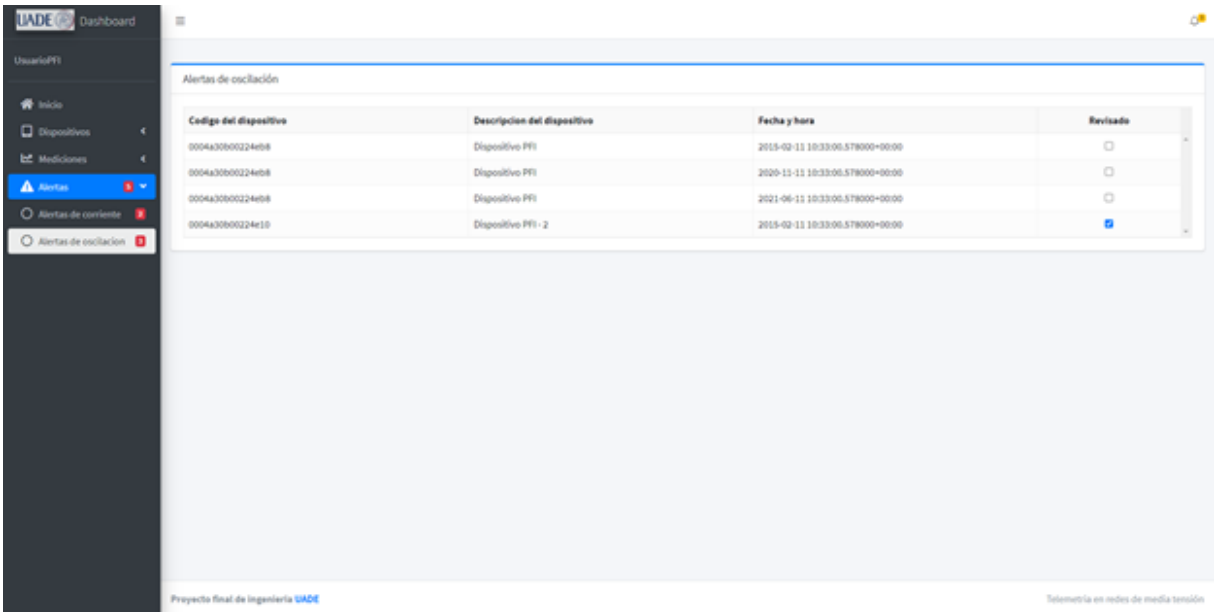

Imagen LII: pantalla de alertas de corriente generadas por todos los dispositivos de la aplicación web

4.4.6 Despliegue de la aplicación web y API Rest

Para llevar a cabo el despliegue del software desarrollado, tanto la aplicación web como la API Rest, se ha utilizado Heroku, una plataforma como servicio (PaaS por sus siglas en inglés) que permite desplegar aplicaciones en la nube desarrolladas en múltiples lenguajes de programación.

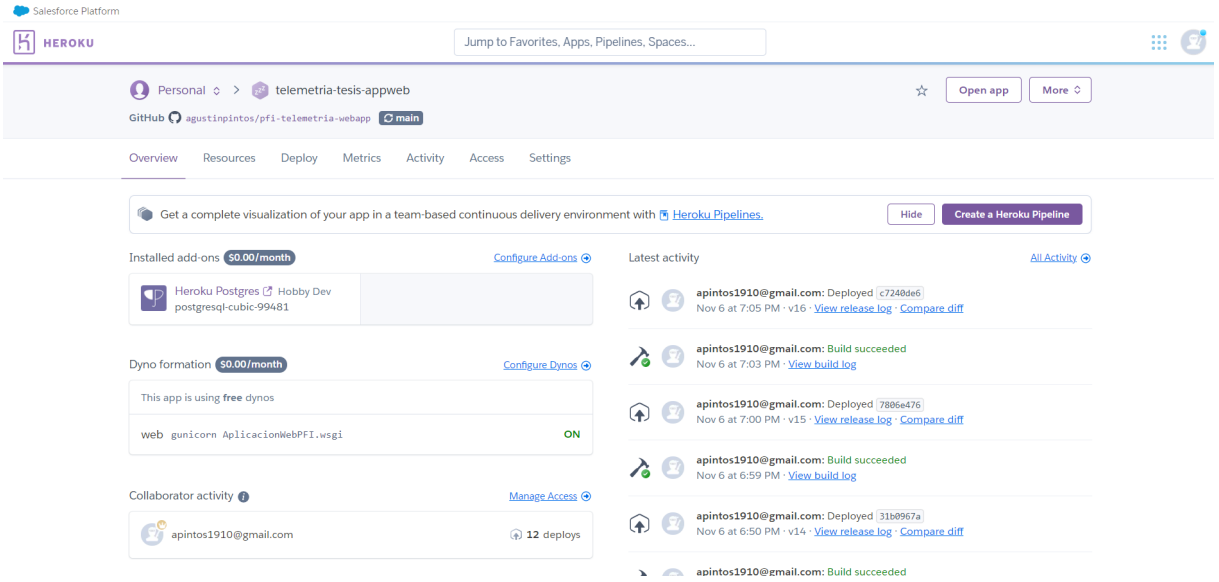

Imagen LII: Heroku como PaaS

La característica principal de las PaaS es que permiten abstraer al desarrollador de toda la infraestructura necesaria para el despliegue de una aplicación ya que la misma se crea de forma automática teniendo en cuenta el stack tecnológico utilizado.

Por otro lado, si bien existen múltiples PaaS en el mercado, la elección de heroku se centra en que brinda sus servicios de forma gratuita para un determinado nivel de consumo. Dicho consumo está relacionado con las horas mensuales que se encuentra en funcionamiento el contenedor que aloja la aplicación desplegada y, a su vez, cada contenedor es desactivado cuando no está siendo accedido por ningún cliente lo cual permite que solo se contemple el tiempo de uso del contenedor cuando existe un cliente que hace uso de la aplicación. Esta característica lo hace óptimo para el hosteo de aplicaciones en fase de desarrollo ya que reduce ampliamente los costos.

Capítulo 5: Marco económico - Financiero

En este capítulo, referido a los aspectos económicos que afectan a nuestro proyecto, tendremos en cuenta distintos factores que influyen directamente en la logística y producción de las terminales inalámbricas y del software desarrollado sobre las cuales verificaremos la posibilidad de que el proyecto sea viable o no. Para ello, realizaremos los siguientes estudios:

- Análisis FODA.
- Estimaciones de ventas
- Estudio de flujo de fondos.
- Estudio de la VAN, la TIR, el ROI y el Payback (Tiempo de retorno).

Luego de realizar todos los estudios mencionados, obtendremos algunas conclusiones importantes a partir de los indicadores calculados para poder definir si el proyecto es viable o no.

5.1 FODA

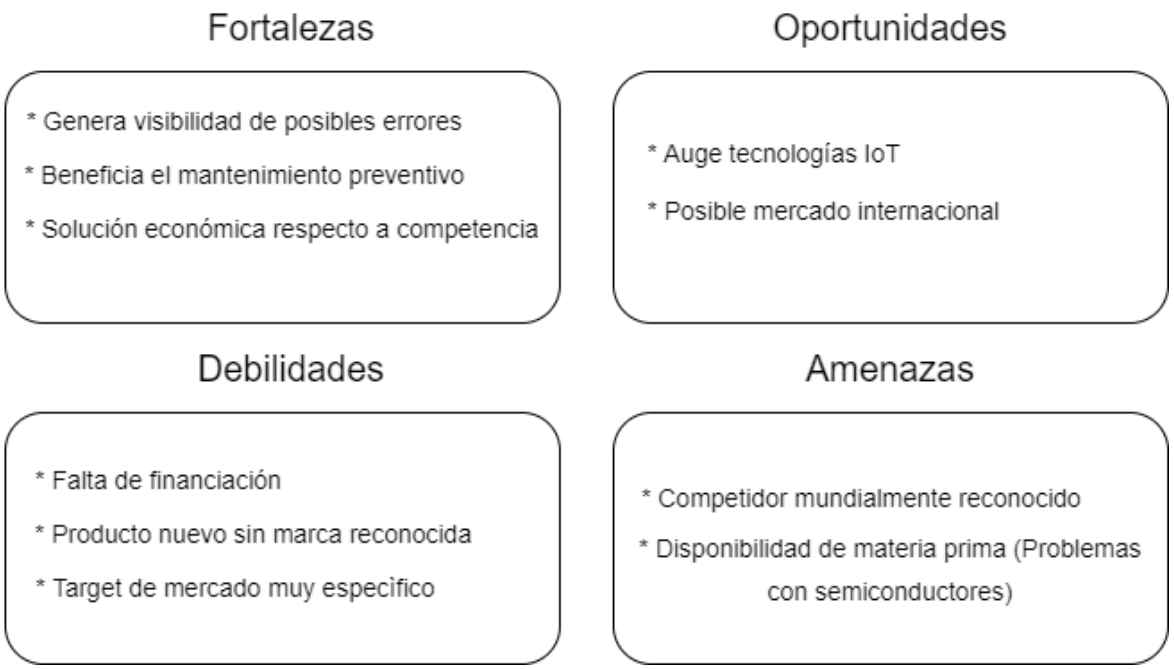

Imagen LIV: FODA

5.2 Estimaciones de ventas

5.2.1 Consideraciones previas

- Se define el tramo de tendido eléctrico que une las provincias de Rio Negro con Santa Cruz como el lugar donde potencialmente se podría implementar el sistema de telemetría, con el único fin de analizar la viabilidad del proyecto. Esta elección se basa en el estudio del mercado descrito en la sección 2 del presente documento y además en que en este tramo en particular (debido a los fuertes vientos presentes en el sur de Argentina) se generan grandes oscilaciones en los tendidos eléctricos, provocando cortes inesperados. Dado que una de las funcionalidades principales del sistema de telemetría es detectar dichas oscilaciones, la implementación en este tramo es idónea. El largo de dicho tramo es aproximadamente de unos 1600 km.
- No se tienen en cuenta aspectos de mantenimiento, implementación y operación de las terminales inalámbricas.
- No se tienen en cuenta aspectos de mantenimiento, implementación y operación de la red de comunicaciones LoRaWAN.
- La red debe estar implementada previamente por el cliente final o en su defecto, contratar el servicio a un carrier que provea cobertura en la zona.
- El objetivo es únicamente comercializar las terminales inalámbricas y el software de monitoreo.
- La distancia entre cada terminal inalámbrica a lo largo del tendido eléctrico es de 1 km.

5.2.2 Estimación de cantidad de unidades

En base a lo descrito anteriormente podemos estimar una venta de al menos 1600 dispositivos, que surgen de:

1600 km * 1 Dispositivo/Km = 1600 Dispositivos

5.3 Costos

En la presente sección se detallan los costos fijos y variables que se te tendrían considerando las estimaciones de ventas mencionadas previamente para poder evaluar posteriormente la viabilidad del proyecto. Cabe destacar que todos los montos están expresados en dólares estadounidenses.

5.3.1 Costos fijos, de mano de obra y patente

En la presente sección se detallan los costos fijos, de mano de obra y de patente estimados en una periodicidad mensual. Los mismos son utilizados en la sección siguiente para determinar la inversión inicial y el costo por unidad de cada terminal inalámbrica.

Costos generales fijos	$
Energia	23,00
Gas	9,00
Impuestos inmobiliarios	7,00
Limpieza	0,00
Agua	4,00
Internet/Cable	20,80
Alquiler	0,00
Telefonia	7,00
Total	**70,80**

Imagen LV: Costos fijos por mes

Mano de obra directa	$
Salarios	1.980,00
Capacitaciones	0,00
Comedor en planta	0,00
Mantenimiento de software	160,00
Total	**2.140,00**

Imagen LVI: Costos de mano de obra por mes

Intangible	$
Patente	60,00

Imagen LVII: Costos de patente del producto

5.3.2 Costos por materia prima

A continuación se detallan los costos de la materia prima necesaria para producir las 1600 unidades estimadas previamente. Dichos costos se utilizan posteriormente para el cálculo del costo unitario de cada terminal.

Materia prima	Unidades	Unidad+er (5%)	$ Unitario	$ Sub-Total
Capacitores	3.250,00	3.412,50	0,03	92,14
Resistencias	6.500,00	6.825,00	0,01	36,17
XA1110	500,00	525,00	11,00	5.775,00
RN2903	500,00	525,00	10,00	5.250,00
PIC18LF47K40	500,00	525,00	1,5	787,50
Antena GPS plana	500,00	525,00	4,00	2.100,00
Cobre 2,2 mm bobina Kg	500,00	525,00	10,00	5.250,00
Ferrite	500,00	525,00	4,00	2.100,00
Protoboards	10,00	10,50	2,30	24,15
Plaquetas multipropósito	10,00	10,50	0,41	4,25
ADXL362BCCZ-RL	500,00	525,00	4,00	2.100,00
Plasticos Pla 1 Kg	500,00	525,00	7,43	3.902,50
Tira 5 pines macho	500,00	525,00	0,01	5,25
Armado	500,00	525,00	2,00	1.050,00
Impresión PCB	500,00	525,00	5,00	2.625,00
Panel solar flexible MP3-37	500,00	525,00	2,00	1.050,00
Supercapacitor 50F	1.000,00	1.050,00	3,00	3.150,00
Conector SMA antena LoRa	500,00	525,00	5,00	2.625,00
Antena LoRa	500,00	525,00	6,00	3.150,00
LM2931-N	500,00	525,00	1,20	630,00
			77,68	**27.426,96**

Imagen LIIX: Costos de materia prima

5.3.3 Costos por maquinaria e instrumentos

En la presente sección se detallan los costos de la maquinaria necesaria para llevar a cabo la construcción de las terminales inalámbricas. Al igual que los costos presentados previamente, estos se utilizan posteriormente para el cálculo del costo unitario de cada terminal.

Maquinaria	Unidades	$ Unitario	$ Sub-Total
Estaciones de soldado	2	24,50	49,00
Osciloscopios	2	198,99	397,98
Impresora	1	50,00	50,00
Fuentes de laboratorio dobles	2	50,00	100,00
Multimetros	4	21,00	84,00
Impresora 3d	2	100,00	200,00
Soporte con lupa	2	12,00	24,00
Generador de corriente 20A	1	250,00	250,00
Computadora	2	300,00	600,00
Hub USB	2	5,00	10,00
Programador ICD3	2	20,00	40,00
Estanteria 5 niveles	2	18,40	36,80
Gavetero	1	20,00	20,00
Mueble y silla de oficina	2	100,00	200,00
		Total	**2.061,78**

Imagen LIX: Costos de maquinaria e instrumentos

5.3.4 Inversión inicial

A continuación se presenta la inversión inicial necesaria teniendo en cuenta la depreciación a lo largo de los 5 períodos estimados para el proyecto y, en segundo lugar, el cálculo del costo por unidad estimado para cada terminal inalámbrica.

Inversion Inicial			Periodo	Depreciacion x periodo					
Concepto	Valor del bien (en $)	Años Vida Util Depreciacion	-1	0	1	2	3	4	5
Maquinarias de laboratorio	1.554,98	10			155	155	155	155	155
Estanterias	36,80	10			4	4	4	4	4
Intangibles (Patente)	60,00	10			6	6	6	6	6
Elementos de Impresión	700,00	5			140	140	140	140	140
Bienes muebles (Mesas de trabajo y oficina	120,00	10			12	12	12	12	12
Total de Depreciacion					317	317	317	317	317
Capital de trabajo									
Materias primas	27.426,96								
Insumos generales									
TOTAL INVERSION INICIAL	29.898,74								

Imagen LX: Inversión inicial

		Valores x periodo promedio				
Costos x unidad	Costo inicial (Periodo 0)	1	2	3	4	5
Costo de materias primas	54,85	55,95	57,070	58,78	60,55	62,97
Costo de mano de obra	51,36	52,39	53,435	55,04	56,69	58,96
CIF	3,14	3,20	3,266	3,36	3,46	3,57
	109,35	**111,54**	**113,77**	**117,18**	**120,70**	**125,49**

Imagen LXI: Costo unitario de cada terminal

5.3 Estudio de Flujo de fondos

En la presente sección se lleva a cabo un análisis en relación al flujo de fondos del proyecto para determinar si el mismo es rentable o no. Estará basado en puntos importantes como el análisis mercado, ya desarrollado anteriormente, donde podemos estimar la cantidad de unidades que serán requeridas. En función de la zona considerada para efectuar la instalación de los equipos, calculamos que producir 1600 equipos es un número acorde.

Otros factores que se tienen en cuenta a la hora de realizar el flujo de fondos son: el costo de la materia prima, maquinarias y mano de obra necesaria, los gastos de servicios de la planta establecida y la tasa de interés que nos ofrecerá el banco para realizar la inversión inicial del proyecto. Es importante destacar que todos los costos están calculados en dólares.

En la siguiente imagen, se muestra el flujo de fondos del proyecto.

Flujo de fondos en dolares							
	Periodos						
Concepto	-1	0	1	2	3	4	5
Unidades estimadas de venta			150,00	200,00	250,00	400,00	600,00
Precio estimado de venta			150,00	180,00	200,00	210,00	230,00
Ingresos			22.500,00	36.000,00	50.000,00	84.000,00	138.000,00
Costos Variables			8.228,09	11.190,20	14.267,51	23.512,85	36.327,35
Costos Fijos			8.174,88	11.117,84	14.175,24	23.360,80	36.092,43
Utilidad bruta			**6.097,03**	**13.691,96**	**21.557,25**	**37.126,35**	**65.580,21**
Gastos de Comercializacion			-6.604,50	-8.589,36	-8.593,00	-9.263,52	-9.278,64
Gastos de Administracion			-7.200,00	-9.360,00	-13.104,00	-18.345,60	-25.683,84
Depreciacion total			-317,18	-317,18	-317,18	-317,18	-317,18
Amortizacion intereses			2.780,93	2.271,50	1.708,54	1.086,42	398,93
Utilidad antes de impuestos			**-5.243,71**	**-2.303,07**	**1.251,61**	**10.286,48**	**30.699,49**
Impuestos			-1.835,30	-806,08	438,06	3.600,27	10.744,82
Utilidad Neta			**-3.408,41**	**-1.497,00**	**813,55**	**6.686,21**	**19.954,67**
Depreciacion total			317,18	317,18	317,18	317,18	317,18
Total invercion inicial	29.898,74						
Devolucion de capital prestamo			4.848	5.358	5.920	6.543	7.230
Flujo de caja		-29.899	**1.756,85**	**4.177,69**	**7.051,20**	**13.545,98**	**27.501,93**

Tabla VII: flujo de fondos del proyecto

En función de lo comentado previamente, podemos establecer que el costo por unidad producida es de U$d 77,68. La inversión inicial requerida es de U$d 29.899 teniendo en cuenta 5 periodos de un año cada uno para la realización del proyecto.

En consecuencia, el préstamo bancario necesario para llevar a cabo el proyecto será de U$d 29.899 con una TNA del 10% a un plazo de 5 años (datos obtenidos del Banco Francés), considerando una frecuencia de pago mensual.

5.4 Análisis de los indicadores financieros

Para determinar si un proyecto es factible o no desde el punto de vista financiero, es importante analizar los siguientes indicadores: VAN, ROI, TIR y PAYBACK. En la siguiente tabla se presentan los valores obtenidos a partir del flujo de fondos desarrollado.

Inversion Inicial	29.899,00					
Periodo	5 años					
TASA	10%					
Calculo VAN	1.597	3.453	5.298	9.252	17.077	**6.777**
TIR	-94%	-60%	-28%	-4%	16%	
Payback	4,24856245	4 años y 3 meses				
ROI	22,67%					

Tabla VIII: indicadores financieros del proyecto

En primer lugar, al observar que el valor de la VAN es superior a cero, podemos establecer que el proyecto generaría ganancias que, traídas al momento actual, serían de U$d 6.777.

En segundo lugar, podemos establecer que el periodo de retorno de la inversión (payback) es de aproximadamente 4 años y 3 meses, por lo que consideramos que es un proyecto factible con un periodo de recupero aceptable.

En tercer lugar, la TIR es de 16% que es superior a la TNA ofrecida por el Banco Francés de un 10%.

Por último, el ROI es de 22,67%, lo cual representa un porcentaje mayor a cualquier otra inversión pasiva como un plazo fijo en dólares que otorga una tasa anual de un 3% aproximadamente. Es por ello que consideramos que llevar a cabo el proyecto nos daría beneficios superiores a otras alternativas de inversión.

En conclusión, todos los indicadores financieros son positivos por lo cual consideramos que el proyecto es factible y puede realizarse.

Capítulo 6: Conclusiones

Finalmente, en base a lo detallado anteriormente, podemos concluir que esta es una solución tecnológica innovadora que resuelve la problemática encontrada en el rubro en cuestión y genera beneficios tangibles al cliente final, es decir, las empresas distribuidoras de energía eléctrica en media tensión.

Los principales beneficios de implementar esta solución son:

- Mejora en métodos de detección de fallas: Esto se ve claramente reflejado al comparar la telemetría en tiempo real que implica el sistema propuesto con el método manual/visual que hoy en día las empresas de distribución eléctrica utilizan para identificar un corte en un tramo particular de la línea. El sistema de telemetría propuesto facilita esta tarea debido a la información geográfica enviada por los endpoints. Es decir, por ejemplo, que si un determinado endpoint genera una alarma de baja tensión, el cliente puede rápidamente ver en donde está ocurriendo dicho evento. De esta manera, se puede ir a solucionar el problema directamente al lugar en cuestión, sin perder tiempo recorriendo la línea en forma física en busca de dicha falla. Esto genera un ahorro económico al cliente, debido a que se optimiza el uso de los recursos y disminuyen los tiempos de solución de problemas, aumentando en consecuencia, la disponibilidad de la red.

- Plataforma unificada de visualización de eventos y alarmas: A partir de la aplicación web desarrollada, el cliente es capaz de tener un control del estado de la red de distribución de manera sencilla y centralizada. Teniendo acceso a información en tiempo real de cada endpoint, incluyendo datos de posición geográfica, alarmas por los diferentes motivos planteados, etc. Todo en un mismo sitio y accesible a través de internet.

- Escalabilidad: El sistema fue desarrollado teniendo en cuenta un posible aumento de dispositivos instalados en otras zonas de interés. De forma tal que, en caso de que el cliente encuentre necesario obtener información de las métricas planteadas en otros puntos de su red de distribución, solo será necesario la construcción e instalación de nuevos dispositivos los cuales se integrarán al sistema en funcionamiento una vez dados de alta en la aplicación web. Por otro lado, una vez implementada esta solución, el cliente tendrá desplegada, o en su defecto, tendrá acceso a la red LoRaWAN ofrecida por un proveedor. Esto implica que si en un futuro el cliente decide que es necesario medir una nueva variable que impacta de alguna manera en su negocio, simplemente hay que instalar un endpoint con la capacidad de medir dicha variable y transmitirla a través de la ya implementada red de comunicaciones.

- Métricas: Mientras más tiempo esté en funcionamiento el sistema, la información recolectada por los endpoints será cada vez mayor y el usuario contará con información histórica de gran utilidad a la hora de generar procesos de mejora sobre su red. A su vez, impacta directamente en la definición de tareas de mantenimiento preventivas inteligentes, que hagan que baje la probabilidad de ocurrencia de fallas inesperadas en la red.

Capítulo 7: Anexos

7.1 Anexo 1

7.1.1 Hojas de datos

RN2903: https://ww1.microchip.com/downloads/en/DeviceDoc/50002390E.pdf

XA1110: https://www.sierrawireless.com/iot-solutions/products/xa1110/

PIC18LF47K40:https://ww1.microchip.com/downloads/en/DeviceDoc/PIC18LF27_47K40-Data-Sheet-40001844E.pdf

MP3-37: https://www.powerfilmsolar.com/products/mp3-37

AM-5907 CAR: https://ar.mouser.com/datasheet/2/315/EP120B-775610.pdf

LM2931-N:https://www.ti.com/lit/ds/symlink/lm2931-n.pdf?ts=1639465961146&ref_url=https%253A%252F%252Fwww.google.com%252F

ADXL362:https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL362.pdf

7.2 Anexo 2

7.2.1 Código del firmware del prototipo

```
#include "mcc_generated_files/mcc.h"
/* Defines */
//hweui: 0004A30B00224EB8
//devaddr: E0000010"
#define GPS_PORT 1
#define DEFAULT_PORT 2
#define ALARM_PORT 3
#define GPS_CHANNEL 1
#define RN2903_CHANNEL 2

/* Funciones */
void Registro_Lora (void);
void TX_Lora(int,int,char);
void select_uart_channel(int);
void parse(void);

/* Variables Globales */
int lower_threshold_interrupt_flag=0;
int upper_threshold_interrupt_flag=0;
int adc_result;
int program_minutes=0;
int send_frequency=2;

char UTC_Time[] = "hhmmss.sss";
char Status[] = "X";              // A = dato válido, V= dato no válido
```

```
char Latitude[] = "3437.1170";          //formato: ddmm.mmmm
char N_S_Indicator[] = "S";             //N = Norte, S = Sur
char Longitude[] = "5822.5611";
char E_W_Indicator[] = "W";             //E = Este, W = Oeste
int cont=0;
char letra;
int i=0;

void main(void)
{
        unsigned long sleep_msec=0;

        SYSTEM_Initialize();// Initialize the device
        printf("***** COMIENZO DEL PROGRAMA *****\r\n");
        __delay_ms(1000);
        select_uart_channel(GPS_CHANNEL);
        parse();
        __delay_ms(3000);
        select_uart_channel(RN2903_CHANNEL);
        Registro_Lora();
        TX_Lora(GPS_PORT,1,'A');
        INTERRUPT_GlobalInterruptEnable();        // Enable the Global Interrupts
        INTERRUPT_PeripheralInterruptEnable();   // Enable the Peripheral Interrupts
        ADCC_StartConversion(Bobina);
        __delay_ms(1000);

        while (1)
        {
```

```
if(upper_threshold_interrupt_flag == 1 || lower_threshold_interrupt_flag == 1)
{
if(upper_threshold_interrupt_flag == 1)
{
        adc_result=ADCC_GetConversionResult();
        printf("***** ALERTA POR SOBRETENSION *****\r\n\r\n");
        RCSTAbits.SPEN=0; //Break Condition
        __delay_us(226);
        RCSTAbits.SPEN=1;
        printf("%c\r\n",0x55);//Send character 0x55 to autobaud detection
        __delay_ms(500);
        printf("Valor de tensión: %iv\r\n",adc_result);
        upper_threshold_interrupt_flag = 0;
        TX_Lora(ALARM_PORT,adc_result,'A');
}

if(lower_threshold_interrupt_flag == 1)
{
        adc_result=ADCC_GetConversionResult();
        printf("***** ALERTA POR BAJA TENSION*****\r\n\r\n");
        RCSTAbits.SPEN=0;
        __delay_us(226);
        RCSTAbits.SPEN=1;
        printf("%c\r\n",0x55);
        __delay_ms(500);
        printf("Valor de tensión: %iv\r\n",adc_result);
        lower_threshold_interrupt_flag=0;
        TX_Lora(ALARM_PORT,adc_result,'B');
```

```
        }
        }
        else
        {
        if(program_minutes % send_frequency == 0)
        {
                printf("***** ENVIO NORMAL *****\r\n\r\n");
                adc_result = ADCC_GetConversionResult();
                TX_Lora(DEFAULT_PORT,adc_result,'A');
                program_minutes++;
        }
        else
                program_minutes++;

        }

        ADCC_StartConversion(Bobina);
        sleep_msec = send_frequency * 90000;
        __delay_ms(2000);
        printf("sys sleep %lu\r\n",sleep_msec);
        __delay_ms(500);
        //printf("***** SLEEP MODE *****\r\n\r\n");
        SLEEP();
        __delay_ms(500);

        }
}
```

```
void Registro_Lora (void)
{
        printf("%s\r\n","mac set devaddr E0000010");

        if (RCSTAbits.OERR == 1)
        {
        RCSTA1bits.CREN = 0;
        RCSTA1bits.CREN = 1;
        }

        while (getch() != 'o' && getch() !='k')
        {
        }

printf("%s\r\n","mac set appskey 3C8F262739BFE3B7BC0826991AD0504D");

        if (RCSTAbits.OERR == 1)
        {
        RCSTA1bits.CREN = 0;
        RCSTA1bits.CREN = 1;
        }

        while (getch() != 'o' && getch() !='k')
        {
        }

printf("%s\r\n","mac set nwkskey 2B7E151628AED2A6ABF7158809CF4F3C");
```

```
        if (RCSTAbits.OERR == 1)
        {
        RCSTA1bits.CREN = 0;
        RCSTA1bits.CREN = 1;
        }

        while (getch() != 'o' && getch() !='k')
        {
        }
        __delay_ms(1000);
}

void TX_Lora(int port,int value,char aux)
{
        int i=0;
        int lat_port_N = 20;
        int lat_port_S = 21;
        int lon_port_E = 30;
        int lon_port_W = 31;

        float Latitude_f = 0;
        float Longitude_f = 0;

        printf("%s\r\n","mac join abp");
        CLRWDT();
        if (RCSTAbits.OERR == 1)
        {
        RCSTA1bits.CREN = 0;
```

```
RCSTA1bits.CREN = 1;
}

while( getch() != 'd')
{
}

if(port == ALARM_PORT)
{
if(aux == 'A')
{
printf("%s%i %c\r\n", "mac tx cnf ",port,aux);

if (RCSTAbits.OERR == 1)
{
RCSTA1bits.CREN = 0;
RCSTA1bits.CREN = 1;
}

while(getch()!='_')
{
}
}

if(aux == 'B')
{
printf("%s%i %c\r\n", "mac tx cnf ",port,aux);
```

```
if (RCSTAbits.OERR == 1)
{
RCSTA1bits.CREN = 0;
RCSTA1bits.CREN = 1;
}

while(getch()!='_')
{
}
}

}

if(port == DEFAULT_PORT)
{
printf("%s%i %X\r\n", "mac tx cnf ",port, value);

if (RCSTAbits.OERR == 1)
{
RCSTA1bits.CREN = 0;
RCSTA1bits.CREN = 1;
}

while(getch()!='_')
{
}

}
```

```
if(port == GPS_PORT)
{
Latitude_f=atof(Latitude);
Longitude_f=atof(Longitude);

if(N_S_Indicator[0] == 'N')
{
printf("%s%i %010lX\r\n", "mac tx cnf ",lat_port_N, Latitude_f);
if (RCSTAbits.OERR == 1)
{
RCSTA1bits.CREN = 0;
RCSTA1bits.CREN = 1;
}

while(getch()!='_')
{
}
}
else
{
printf("%s%i %010lX\r\n", "mac tx cnf ",lat_port_S, Latitude_f);
if (RCSTAbits.OERR == 1)
{
RCSTA1bits.CREN = 0;
RCSTA1bits.CREN = 1;
}
```

```
while(getch()!='_')
{
}
}

if(E_W_Indicator[0] == 'E')
{
printf("%s%i %010lX\r\n", "mac tx cnf ",lon_port_E, Longitude_f);
if (RCSTAbits.OERR == 1)
{
RCSTA1bits.CREN = 0;
RCSTA1bits.CREN = 1;
}

while(getch()!='_')
{
}
}

else
{
printf("%s%i %010lX\r\n", "mac tx cnf ",lon_port_W, Longitude_f);
if (RCSTAbits.OERR == 1)
{
RCSTA1bits.CREN = 0;
RCSTA1bits.CREN = 1;
}
```

```
        while(getch()!='_')
        {
        }
        }

        }
}

void select_uart_channel(int channel)
{
        if(channel == GPS_CHANNEL)
        {
        RCSTA1bits.SPEN = 0;
        RCSTA2bits.SPEN = 1;
        }

        if(channel == RN2903_CHANNEL)
        {
        RCSTA1bits.SPEN = 1;
        RCSTA2bits.SPEN = 0;
        }
}

void parse()
{
        while(cont<10){

        while(getch()!='N'){}
```

```
while(getch()!='R'){}
while(getch()!='M'){}
while(getch()!='C'){}
while(getch()!=','){}

while((letra = getch()) !=','){
        UTC_Time[i]=letra;
                i++;
}
i=0;
while((letra = getch()) !=','){
        Status[i]=letra;
                i++;
}
i=0;
while((letra = getch()) !=','){
        Latitude[i]=letra;
                i++;
}
i=0;
while((letra = getch()) !=','){
        N_S_Indicator[i]=letra;
                i++;
}
i=0;
while((letra = getch()) !=','){
        Longitude[i]=letra;
                i++;
```

```
        }
        i=0;
        while((letra = getch()) !=','){
                E_W_Indicator[i]=letra;
                        i++;
        }
        i=0;
       if(Status [0] == 'A'){
       cont++;
       }
       }
}
/**
 End of File
*/
```

7.3 Anexo 3

7.3.1 Código de la aplicación web

A continuación, se incluye el código de los archivos que han sido autogenerados por el framework y posteriormente modificados para el desarrollo de la aplicación web mencionada en el documento principal. Cabe destacar que en el archivo /requirements.txt se incluyen las librerías utilizadas y su correspondiente versión.

- **/AplicacionWebPFI/settings.py**

  import os

  # Build paths inside the project like this: os.path.join(BASE_DIR, ...)

  BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

  # Quick-start development settings - unsuitable for production

  # See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/

  # SECURITY WARNING: keep the secret key used in production secret!

  SECRET_KEY = '(8+a8)av3gwuhq+ku0jpc!@5k6f$!6t)8g1&hd5pr%htc@$%+u'

  # SECURITY WARNING: don't run with debug turnedcannot import name 'python_2_unicode_compatible' from 'django.utils.encoding'

  DEBUG = True

  ALLOWED_HOSTS = ['localhost','192.168.0.112','127.0.0.1','telemetria-tesis-appweb.herokuapp.com']

  # Application definition

  INSTALLED_APPS = [

```
        'adminlte3',
        'adminlte3_theme',
        'django.contrib.admin',
        'django.contrib.auth',
        'django.contrib.contenttypes',
        'django.contrib.sessions',
    'django.contrib.messages',
        'django.contrib.staticfiles',
        'PaginaWeb',



]


MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
        'django.middleware.clickjacking.XFrameOptionsMiddleware',
   'whitenoise.middleware.WhiteNoiseMiddleware',
]
```

```
ROOT_URLCONF = 'AplicacionWebPFI.urls'


TEMPLATES = [
        {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        #'DIRS':
['/home/agustin/Escritorio/PFI/ProyectoDjango/AplicacionWebPFI/PaginaWeb/Templ
ates'],
        'DIRS': [os.path.join(BASE_DIR, "PaginaWeb/Templates")],
        'APP_DIRS': True,
        'OPTIONS': {
        'context_processors': [
            'django.template.context_processors.debug',
            'django.template.context_processors.request',
            'django.contrib.auth.context_processors.auth',
            'django.contrib.messages.context_processors.messages',
        ],
        },
        },
]


WSGI_APPLICATION = 'AplicacionWebPFI.wsgi.application'



# Database
```

```
# https://docs.djangoproject.com/en/3.0/ref/settings/#databases


# DATABASES = {
#         'default': {
#         'ENGINE': 'django.db.backends.sqlite3',
#         'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
#         }
# }


DATABASES = {
        'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'pfi_telemetria',
        'USER': 'agustin',
        'PASSWORD': 'agustin123',
        'HOST': '66.97.34.41',
        'PORT': '3306',
        }
}


# DATABASES = {
#         'default': dj_database_url.config(
#         default=config('DATABASE_URL')
#         )
```

```
# }


# Password validation

# https://docs.djangoproject.com/en/3.0/ref/settings/#auth-password-validators


AUTH_PASSWORD_VALIDATORS = [

        {

        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',

        },

        {

        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',

        },

        {

        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',

        },

        {

        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',

        },

]



# Internationalization
```

```
# https://docs.djangoproject.com/en/3.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True



# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.0/howto/static-files/

STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'PaginaWeb')



STATICFILES_DIRS = (
        os.path.join(BASE_DIR, 'static'),
)
```

```
STATICFILES_STORAGE = 'whitenoise.storage.CompressedManifestStaticFilesStorage'
```

- **/AplicacionWebPFI/urls.py**

```
from django.contrib import admin
from django.urls import path
from PaginaWeb import views
from django.conf.urls.static import static
from django.conf import settings

urlpatterns = [
    path('admin/', admin.site.urls),
    path('ver_dispositivos/', views.ver_dispositivos),
    path('registrar_dispositivo/', views.registro_dispositivo),
    path('nuevo_dispositivo/', views.nuevo_dispositivo),

    path('index/', views.index),
    path('devices_list/', views.devices_list),
  path('devices_list/new/<codeDevice>&<descDevice>', views.devices_new),
  path('devices_list/edit/<deviceId>&<codeDevice>&<descDevice>', views.devices_edit),
  path('devices_list/delete/<codeDevice>', views.devices_delete),
    path('devices_locations/', views.devices_locations),
    path('measurement_current/', views.measurement_current),
  path('measurement_current/<codeDevice>', views.measurement_current_device),
```

```
        path('measurement_oscillation/', views.measurement_oscillation),
        path('alert_current/', views.alert_current),
    path('alert_current/<alertId>&<viewed>', views.alert_current_edit),

        path('alert_oscillation/', views.alert_oscillation),
    path('alert_oscillation/<alertId>&<viewed>', views.alert_oscillation_edit),

]+ static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

- **/PaginaWeb/Templates/alert_current.html**

```
{% extends 'base_template.html' %}
{%load static%}
{%block alert_menu%}menu-open{%endblock%}
{%block alert%}active{%endblock%}
{%block alert_current%}active{%endblock%}
{% block title%}Alertas de corriente{%endblock%}

{% block content %}

<!-- Main content -->

<div class="card card-primary card-outline">
        <div class="card-header">
        <h3 class="card-title">Alertas de corriente.</h3>
```

```
        </div>
        <!-- /.card-header -->
        <div class="card-body">
        <div id="jsGrid"></div>
        </div>
        <!-- /.card-body -->
  </div>



  <!-- Control Sidebar -->
  <aside class="control-sidebar control-sidebar-dark">
        <!-- Control sidebar content goes here -->
  </aside>
  <!-- /.control-sidebar -->
<!-- jQuery -->
<script src="{%static 'admin-lte/plugins/jquery/jquery.min.js' %}"></script>
<!-- jsGrid -->
<link type="text/css" rel="stylesheet" href="{%static
'admin-lte/plugins/jsgrid/jsgrid.min.css'%}" />
<link type="text/css" rel="stylesheet" href="{%static
'admin-lte/plugins/jsgrid/jsgrid-theme.min.css'%}" />


<script type="text/javascript" src="{%static
'admin-lte/plugins/jsgrid/jsgrid.min.js'%}"></script>
<script></script>
```

```
<!--Recupero los dispositivos y los paso a variable JS-->

<script>var currentAlertDicList=[];</script>

{% for alert in currentAlertList %}

<script>

var alertId="{{alert.alertId | safe}}";

var codeDevice="{{alert.codeDevice | safe}}";

var descDevice="{{alert.descDevice | safe}}";

var registrationDate="{{alert.registrationDate | safe}}";

var active="{{alert.active | safe}}";

var alertDatetime="{{alert.alertDatetime | safe}}";

var viewed="{{alert.viewed | safe}}";

var alertType="{{alert.alertType | safe}}";

var
alertDic={"alertId":alertId,"codeDevice":codeDevice,"description":descDevice,"registrationD
ate":registrationDate,"active":active,"alertDatetime":alertDatetime,"viewed":viewed,"alertTyp
e":alertType};

currentAlertDicList.push(alertDic);

console.log(currentAlertDicList);

</script>

{%endfor%}

<!--TABLA-->


<script>
```

```
$("#jsGrid").jsGrid({

width: "100%",

height: "auto",


sorting: true,


paging: true,

pageSize: 5,


noDataContent: "No se han registrado alertas",


data: currentAlertDicList,


fields: [

{ name: "codeDevice", title:"Codigo del dispositivo", type: "text", editing: false, width: 200},

{ name: "description", title:"Descripcion del dispositivo", type: "text", editing: false, width: 200},

{ name: "alertDatetime", title:"Fecha y hora", type: "text", editing: false, width: 200 },

{ name: "alertType", title:"Tipo de alerta", type: "text", editing: false, width: 200 },

//checkbox conf

{ name: "viewed", title: "Revisado",align: "center",

        itemTemplate: function(viewed, item) {

        return $("<input>").attr("type", "checkbox")

        .attr("checked", function() {
```

```
if (viewed === "False") {
item.uncheck = true;
item.check = false;
$(this).prop("unchecked", true);
} else {

item.checked = true;
item.uncheck = false;
$(this).prop("checked", true);
}
})
.on("click", function() {
if (item.uncheck === true) {
item.checked = true;
item.uncheck =  false;
console.log("cambio estado a revisado");
$(this).prop("checked", true);
window.location="/alert_current/"+item["alertId"]+"&"+"True";

} else {
item.uncheck = true;
item.checked = false;
console.log("cambio estado a no revisado");
$(this).prop("unchecked", true);
```

```
                window.location="/alert_current/"+item["alertId"]+"&"+"False";


                }
                });
                }
        },
        ]
        });
</script>

{% endblock %}
```

- **/PaginaWeb/Templates/alert_oscilation.html**

```
{% extends 'base_template.html' %}
{%load static%}
{%block alert_menu%}menu-open{%endblock%}
{%block alert%}active{%endblock%}
{%block alert_oscillation%}active{%endblock%}
{% block title%}Alertas de oscilación{%endblock%}

{% block content %}

<!-- Main content -->
```

```
<div class="card card-primary card-outline">
        <div class="card-header">
        <h3 class="card-title">Alertas de oscilación</h3>
        </div>
        <!-- /.card-header -->
        <div class="card-body">
        <div id="jsGrid"></div>
        </div>
        <!-- /.card-body -->
 </div>


 <!-- Control Sidebar -->
 <aside class="control-sidebar control-sidebar-dark">
        <!-- Control sidebar content goes here -->
 </aside>
 <!-- /.control-sidebar -->
<!-- jQuery -->
<script src="{%static 'admin-lte/plugins/jquery/jquery.min.js' %}"></script>
<!-- jsGrid -->
<link type="text/css" rel="stylesheet" href="{%static
'admin-lte/plugins/jsgrid/jsgrid.min.css'%}" />
<link type="text/css" rel="stylesheet" href="{%static
'admin-lte/plugins/jsgrid/jsgrid-theme.min.css'%}" />
```

```
<script type="text/javascript" src="{%static 'admin-lte/plugins/jsgrid/jsgrid.min.js'%}"></script>

<script></script>

<!--Recupero los dispositivos y los paso a variable JS-->

<script>var oscillationAlertDicList=[];</script>

{% for alert in oscillationAlertList %}

<script>

var alertId="{{alert.alertId | safe}}";

var codeDevice="{{alert.codeDevice | safe}}";

var descDevice="{{alert.descDevice | safe}}";

var registrationDate="{{alert.registrationDate | safe}}";

var active="{{alert.active | safe}}";

var alertDatetime="{{alert.alertDatetime | safe}}";

var viewed="{{alert.viewed | safe}}";

var alertDic={"alertId":alertId,"codeDevice":codeDevice,"description":descDevice,"registrationDate":registrationDate,"active":active,"alertDatetime":alertDatetime,"viewed":viewed};

oscillationAlertDicList.push(alertDic);

console.log(oscillationAlertDicList);

</script>

{%endfor%}

<!--TABLA-->

 <script>

        $("#jsGrid").jsGrid({

        width: "100%",
```

```
height: "auto",

sorting: true,

paging: true,
pageSize: 5,

noDataContent: "No se han registrado alertas",

data: oscillationAlertDicList,

fields: [
{ name: "codeDevice", title:"Codigo del dispositivo", type: "text", editing: false, width: 200},
{ name: "description", title:"Descripcion del dispositivo", type: "text", editing: false, width: 200},
{ name: "alertDatetime", title:"Fecha y hora", type: "text", editing: false, width: 200 },
//checkbox conf
{ name: "viewed", title: "Revisado",align: "center",
itemTemplate: function(viewed, item) {
return $("<input>").attr("type", "checkbox")
.attr("checked", function() {
if (viewed === "False") {
item.uncheck = true;
item.check = false;
```

```
$(this).prop("unchecked", true);
} else {

item.checked = true;
item.uncheck = false;
$(this).prop("checked", true);
}
})
.on("click", function() {
if (item.uncheck === true) {
        item.checked = true;
item.uncheck =  false;
console.log("cambio estado a revisado");
$(this).prop("checked", true);
window.location="/alert_oscillation/"+item["alertId"]+"&"+"True";

} else {
item.uncheck = true;
item.checked = false;
console.log("cambio estado a no revisado");
$(this).prop("unchecked", true);
window.location="/alert_oscillation/"+item["alertId"]+"&"+"False";

}
```

```
            });
            }
        },
        ]
        });
</script>

{% endblock %}
```

- **/PaginaWeb/Templates/alert_oscilation.html**

```
{% load static %}
<!DOCTYPE html>
<!--
This is a starter template page. Use this page to start your new project from
scratch. This page gets rid of all links and provides the needed markup only.
-->
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta http-equiv="x-ua-compatible" content="ie=edge">

  <title>{% block title%}{%endblock%}</title>
```

```
  <!-- Font Awesome Icons -->

    <link rel="stylesheet" href="{%static 'admin-lte/plugins/fontawesome-free/css/all.min.css' %}">

  <!-- Theme style -->

  <link rel="stylesheet" href="{%static 'admin-lte/dist/css/adminlte.min.css' %}">

  <!-- Google Font: Source Sans Pro -->

    <link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700" rel="stylesheet">

</head>

<body class="hold-transition sidebar-mini">

<div class="wrapper">


  <!-- Navbar -->

  <nav class="main-header navbar navbar-expand navbar-white navbar-light">

        <!-- Left navbar links -->

        <ul class="navbar-nav">

        <li class="nav-item">

          <a class="nav-link" data-widget="pushmenu" href="#" role="button"><i class="fas fa-bars"></i></a>

        </li>

        </ul>

        <!-- SEARCH FORM

        <form class="form-inline ml-3">

        <div class="input-group input-group-sm">

        <input class="form-control form-control-navbar" type="search" placeholder="Search" aria-label="Search">
```

```
        <div class="input-group-append">
        <button class="btn btn-navbar" type="submit">
        <i class="fas fa-search"></i>
        </button>
        </div>
        </div>
        </form>
-->
        <!-- Right navbar links -->
        <ul class="navbar-nav ml-auto">

        <!-- Notifications Dropdown Menu -->
        <li class="nav-item dropdown">
        <a class="nav-link" data-toggle="dropdown" href="#">
        <i class="far fa-bell"></i>
        <span class="badge badge-warning navbar-badge">{%ifequal quantityTotAlert 0 %}{{null}}{% else %}{{quantityTotAlert}}{%endifequal%}</span>
          </a>
        <div class="dropdown-menu dropdown-menu-lg dropdown-menu-right">
        <span class="dropdown-header">{{quantityTotAlert}} Alerta/s sin revisar</span>
        <div class="dropdown-divider"></div>
        <a href="/alert_current" class="dropdown-item">
        <i class="fas fa-exclamation mr-2"></i> {{quantityCurAlert}} Alerta/s de corriente
```

```
        </a>
        <div class="dropdown-divider"></div>
        <a href="/alert_oscillation" class="dropdown-item">
        <i class="fas fa-exclamation mr-2"></i> {{quantityOscAlert}} Alerta/s de oscilacion

        </a>

        </div>
        </li>

        </ul>
 </nav>
 <!-- /.navbar -->

 <!-- Main Sidebar Container -->
 <aside class="main-sidebar sidebar-dark-primary elevation-4">
        <!-- Brand Logo -->
        <a href="/index" class="brand-link">
        <img   src="https://www.infotyl.com.ar/wp-content/uploads/2009/04/prensa_uade.jpg"
alt="UADE Logo" class="brand-image elevation-3"
        style="opacity: .8">
        <!-- LOGO UADE - FALTA -->
        <span class="brand-text font-weight-light">Dashboard</span>
        </a>
```

```
<!-- Sidebar -->
<div class="sidebar">
<!-- Sidebar user panel (optional) -->
<div class="user-panel mt-3 pb-3 mb-3 d-flex">

<div class="info">
<a href="#" class="d-block">UsuarioPFI</a>
</div>
</div>

<!-- Sidebar Menu -->
<nav class="mt-2">
<ul class="nav nav-pills nav-sidebar flex-column" data-widget="treeview" role="menu" data-accordion="false">
<!-- Add icons to the links using the .nav-icon class
with font-awesome or any other icon font library -->
<li class="nav-item">
<a href="/index" class="nav-link {%block index%}{%endblock%}">
<i class="nav-icon fas fa-home"></i>
<p>Inicio</p>
</a>
    </li>
<li class="nav-item has-treeview {%block devices_menu%}{%endblock%}">
```

```
<a href="#" class="nav-link {%block devices%}{%endblock%}">
<i class="nav-icon fas fa-tablet-alt"></i>
<p>
    Dispositivos
    <i class="right fas fa-angle-left"></i>
</p>
</a>
<ul class="nav nav-treeview">
<li class="nav-item">
    <a href="/devices_list" class="nav-link {%block devices_list%}{%endblock%}">
    <i class="far fa-circle nav-icon"></i>
    <p>Lista de dispositivos</p>
    </a>
</li>
<li class="nav-item">
    <a href="/devices_locations" class="nav-link {%block devices_locations%}{%endblock%}">
    <i class="far fa-circle nav-icon"></i>
    <p>Geolocalizacion</p>
    </a>
</li>
</ul>
</li>
<li class="nav-item has-treeview {%block measurement_menu%}{%endblock%}">
```

```
<a href="#" class="nav-link {%block measurement%}{%endblock%}">
<i class="nav-icon fas fa-chart-line"></i>
<p>
    Mediciones
    <i class="right fas fa-angle-left"></i>
</p>
</a>
<ul class="nav nav-treeview">
<li class="nav-item">
    <a href="/measurement_current" class="nav-link {%block measurement_current%}{%endblock%}">
    <i class="far fa-circle nav-icon"></i>
    <p>Mediciones de corriente</p>
    </a>
</li>
<li class="nav-item">
    <a href="/measurement_oscillation" class="nav-link {%block measurement_oscillation%}{%endblock%}">
    <i class="far fa-circle nav-icon"></i>
    <p>Mediciones de oscilacion</p>
    </a>
</li>
</ul>
</li>
<li class="nav-item has-treeview {%block alert_menu%}{%endblock%}">
```

```
<a href="#" class="nav-link {%block alert%}{%endblock%}">

<i class="nav-icon fas fa-exclamation-triangle"></i>

<p>

    Alertas

    <i class="right fas fa-angle-left"></i>

    <span class="right badge badge-danger">{%ifequal quantityTotAlert 0 %}{{null}}{% else %}{{quantityTotAlert}}{%endifequal%}</span>

</p>

</a>

<ul class="nav nav-treeview">

<li class="nav-item">

    <a href="/alert_current" class="nav-link {%block alert_current%}{%endblock%}">

    <i class="far fa-circle nav-icon"></i>

    <p>Alertas de corriente<span class="right badge badge-danger">{%ifequal quantityCurAlert 0 %}{{null}}{% else %}{{quantityCurAlert}}{%endifequal%}</span></p>

    </a>

</li>

<li class="nav-item">

    <a href="/alert_oscillation" class="nav-link {%block alert_oscillation%}{%endblock%}">

    <i class="far fa-circle nav-icon"></i>

    <p>Alertas de oscilacion<span class="right badge badge-danger">{%ifequal quantityOscAlert 0 %}{{null}}{% else %}{{quantityOscAlert}}{%endifequal%}</span></p>

    </a>
```

```
        </li>
        </ul>
        </li>

        </ul>
        </nav>
        <!-- /.sidebar-menu -->
        </div>
        <!-- /.sidebar -->
</aside>

<!-- Content Wrapper. Contains page content -->
<div class="content-wrapper">
<!-- Content Header (Page header) -->
<div class="content-header">
        <div class="container-fluid">
        <div class="row mb-2">
    <div class="col-sm-6">
        <h1 class="m-0 text-dark"></h1>
        </div><!-- /.col -->

        </div><!-- /.row -->
        </div><!-- /.container-fluid -->
</div>
```

```
  <!-- /.content-header -->
  <div class="content">
        <div class="container-fluid">
  {% block content %}


  {% endblock %}
  </div>
  </div>
</div>
  <!-- Control Sidebar -->
  <aside class="control-sidebar control-sidebar-dark">
        <!-- Control sidebar content goes here -->
        <div class="p-3">
        <h5>Title</h5>
        <p>Sidebar content</p>
        </div>
  </aside>
  <!-- /.control-sidebar -->


  <!-- Main Footer -->
  <footer class="main-footer">
        <!-- To the right -->
        <div class="float-right d-none d-sm-inline">
        Telemetria en redes de media tensión
```

```
        </div>
        <!-- Default to the left -->
        <strong>Proyecto final de ingenieria <a href="https://www.uade.edu.ar">UADE</a></strong>
  </footer>
</div>
<!-- ./wrapper -->

<!-- REQUIRED SCRIPTS -->

<!-- jQuery -->
<script src="{%static 'admin-lte/plugins/jquery/jquery.min.js' %}"></script>
<!-- Bootstrap 4 -->
<script src="{%static 'admin-lte/plugins/bootstrap/js/bootstrap.bundle.min.js' %}"></script>
<!-- AdminLTE App -->
<script src="{%static 'admin-lte/dist/js/adminlte.min.js' %}"></script>
</body>
</html>
```

- **/PaginaWeb/Templates/devices_list.html**

```
{% extends 'base_template.html' %}
{%load static%}
<!--le asigno las clases activas a los menu de la pagina en la que estoy-->
{%block devices_menu%}menu-open{%endblock%}
```

```
{%block devices%}active{%endblock%}
{%block devices_list%}active{%endblock%}
{% block title%}Lista de dispositivos{%endblock%}
{% block content %}
<div class="card card-primary card-outline">
        <div class="card-header">
        <h3 class="card-title">Dispositivos</h3>
        </div>
        <!-- /.card-header -->
        <div class="card-body">
        <div id="jsGrid"></div>
        </div>
        <!-- /.card-body -->
 </div>


 <!-- Control Sidebar -->
 <aside class="control-sidebar control-sidebar-dark">
        <!-- Control sidebar content goes here -->
 </aside>
 <!-- /.control-sidebar -->
```

```
<!-- jQuery -->
<script src="{%static 'admin-lte/plugins/jquery/jquery.min.js' %}"></script>
<!-- jsGrid -->
<link type="text/css" rel="stylesheet" href="{%static 'admin-lte/plugins/jsgrid/jsgrid.min.css'%}" />
<link type="text/css" rel="stylesheet" href="{%static 'admin-lte/plugins/jsgrid/jsgrid-theme.min.css'%}" />

<script type="text/javascript" src="{%static 'admin-lte/plugins/jsgrid/jsgrid.min.js'%}"></script>
<script></script>
<!--Recupero los dispositivos y los paso a variable JS-->
<script>var devicesDicList=[];</script>
{% for dev in devicesList %}
 <script>
        var deviceId="{{dev.id | safe}}";
        var codeDevice="{{dev.codeDevice | safe}}";
        var desciption="{{dev.descDevice | safe}}";
        var registrationDate="{{dev.registrationDate | safe}}";
        var active="{{dev.active | safe}}";
        var devDic={"deviceId":deviceId,"codeDevice":codeDevice,"description":desciption,"registrationDate":registrationDate,"active":active};
        devicesDicList.push(devDic);
        console.log(devicesDicList);
```

```
 </script>
{%endfor%}
<!--TABLA-->


<script>
        //lenguaje de la tabla
        (function(jsGrid) {


jsGrid.locales.es = {
        grid: {
        noDataContent: "No encontrado",
        deleteConfirm: "¿Está seguro?",
        pagerFormat: "Paginas: {first} {prev} {pages} {next} {last}    {pageIndex} de {pageCount}",
        pagePrevText: "Anterior",
        pageNextText: "Siguiente",
        pageFirstText: "Primero",
        pageLastText: "Ultimo",
        loadMessage: "Por favor, espere...",
        invalidMessage: "¡Datos no válidos!"
        },


        loadIndicator: {
        message: "Cargando..."
```

```
},

fields: {
control: {
searchModeButtonTooltip: "Cambiar a búsqueda",
insertModeButtonTooltip: "Cambiar a inserción",
editButtonTooltip: "Editar",
deleteButtonTooltip: "Borrar",
searchButtonTooltip: "Buscar",
clearFilterButtonTooltip: "Borrar filtro",
insertButtonTooltip: "Insertar",
updateButtonTooltip: "Actualizar",
cancelEditButtonTooltip: "Cancelar edición"
}
},

validators: {
required: { message: "Campo requerido" },
rangeLength: { message: "La longitud del valor está fuera del intervalo definido" },
minLength: { message: "La longitud del valor es demasiado corta" },
maxLength: { message: "La longitud del valor es demasiado larga" },
pattern: { message: "El valor no se ajusta al patrón definido" },
range: { message: "Valor fuera del rango definido" },
min: { message: "Valor demasiado bajo" },
```

```
        max: { message: "Valor demasiado alto" }
        }
};

}(jsGrid, jQuery));
        //Implementa lenguaje español
        jsGrid.locale("es");

        $("#jsGrid").jsGrid({
        width: "100%",
        height: "auto",

        inserting: true,
        editing: true,
        sorting: true,
        paging: true,
        pageSize: 5,
        autoload: true,


        //Data de la tabla
        data: devicesDicList,
        //Funciones: nuevo, editar, borrar
        controller: {
```

```
                insertItem: function (item) {
                for (dev in devicesDicList){
                if(devicesDicList[dev]["codeDevice"]==item["codeDevice"]){
                        alert("El codigo ya esta registrado");
                        return(window.location="/devices_list/");
                }
                 }
                alert("Dispositivo registrado")

return(window.location="/devices_list/new/"+item["codeDevice"]+"&"+item["description"]);
                },
                updateItem: function (item) {
            for (dev in devicesDicList){
                console.log(devicesDicList[dev]["codeDevice"]);
                if(devicesDicList[dev]["codeDevice"]==item["codeDevice"]){
                        alert("El codigo ya esta registrado");
                        return(window.location="/devices_list/");
                }
               }

return(window.location="/devices_list/edit/"+item["deviceId"]+"&"+item["codeDevice"]+"&
"+item["description"]);
                },
                deleteItem: function (item) {
                alert("Dispositivo eliminado")
```

```
        return(window.location="/devices_list/delete/"+item["codeDevice"]);
            }
    },
    fields: [
    { name: "codeDevice", title:"Codigo", type: "text", width: 200, validate: "required" },
    { name: "description", title:"Descripcion", type: "text", width: 200,validate: "required",filtering: false},
    { name: "registrationDate", title:"Fecha de registro", type: "text", width: 100, editing: false,inserting: false,filtering: false },
    { name: "active", title:"Estado", type: "text", width: 100, editing: false,inserting: false,filtering: false,
    itemTemplate: function(active, item) {
        if(active=="True"){
        return("Activo");
        }
        else{
      return("Inactivo");
        }
    }},
    { type: "control", modeSwitchButton: true, editButton: true}

    ]
    });
```

```
</script>


{% endblock %}
```

- **/PaginaWeb/Templates/devices_locations.html**

```
{% extends 'base_template.html' %}
{%load static%}
<!--le asigno las clases activas a los menu de la pagina en la que estoy-->
{%block devices_menu%}menu-open{%endblock%}
{%block devices%}active{%endblock%}
{%block devices_locations%}active{%endblock%}
{% block title%}Geolocalizacion{%endblock%}
{% block content %}
<!---->
<!--MAPA-->
<style>

        /* Set the size of the div element that contains the map */
  #map {
        height: 100%;  /* The height is 400 pixels */
        width: 100%;  /* The width is the width of the web page */
        }
 </style>
 <!--pasamos la variable lista de dispositivos de python a una variable en JS-->
```

```
<script>var locationList=[];</script>
{% for loc in locationList %}
 <script>
        var codeDevice="{{loc.codeDevice | safe}}";
        var descDevice="{{loc.descDevice | safe}}";
        var active="{{loc.active | safe}}";
        var lat=parseFloat("{{loc.lat | safe}}");
        var lng=parseFloat("{{loc.lng | safe}}");
        var curAlertViewed= "{{loc.curAlertViewed | safe}}";
        var oscAlertViewed= "{{loc.oscAlertViewed | safe}}";
   locationList.push([codeDevice,lat,lng,curAlertViewed,oscAlertViewed,descDevice,active]);
        console.log(locationList);
 </script>
{%endfor%}
 <script>

// Initialize and add the map
        function initMap() {

//define el centro del mapa, si no existe una localizacion el centro es el congreso

        if (locationList.length!=0){
        var centro ={lat: locationList[0][1], lng: locationList[0][2]}
```

```
}

else{

var centro ={lat: -34.609701, lng: -58.392595}

}


var map = new google.maps.Map(document.getElementById('map'), {zoom: 12, center: centro});

// Crea las marcas en el mapa

for (var item in locationList){

var marker = new google.maps.Marker({title: locationList[item][0], position: {lat: locationList[item][1], lng: locationList[item][2]}, map: map});


if(locationList[item][3]=='True' && locationList[item][4]=='True' && locationList[item][6]=='True'){ //no hay alertas sin revisar, icono verde

marker.setIcon('http://maps.google.com/mapfiles/ms/icons/green-dot.png');


}

else if(locationList[item][3]=='False' && locationList[item][4]=='True' && locationList[item][6]=='True'){ //hay SOLO alertas de corriente sin revisar, icono azul

marker.setIcon('http://maps.google.com/mapfiles/ms/icons/blue-dot.png');

marker.setAnimation(google.maps.Animation.BOUNCE);


}

else if(locationList[item][3]=='True' && locationList[item][4]=='False' && locationList[item][6]=='True'){ //hay SOLO alertas de oscilacion sin revisar, icono amarillo

marker.setIcon('http://maps.google.com/mapfiles/ms/icons/yellow-dot.png');
```

```
marker.setAnimation(google.maps.Animation.BOUNCE);

 }

 else if(locationList[item][3]=='False' && locationList[item][4]=='False' && locationList[item][6]=='True'){//hay alertas tanto de corriente como de oscilacion sin revisar, icono rojo

marker.setIcon('http://maps.google.com/mapfiles/ms/icons/red-dot.png');

marker.setAnimation(google.maps.Animation.BOUNCE);

 }

 else{//el dispositivo no esta activo

 marker.setIcon('http://maps.google.com/mapfiles/ms/icons/purple-dot.png');

 marker.setAnimation(google.maps.Animation.BOUNCE);

 }

 }


 }

</script>


<!---->

<!--MAPA-->



 <!-- Main content -->


 <div class="row">
```

```
  <div class="col-lg-7 col-md-7 col-sm-12 col-xs-12">
<div class="card card-primary card-outline">
        <div class="card-header">
        <h3 class="card-title">Localizaciones de los dispositivos activos</h3>
        </div>
        <!-- /.card-header -->
        <div class="card-body p-0">
        <div class="d-md-flex">
         <div class="p-1 flex-fill" style="overflow: hidden">
        <!-- Map will be created here -->
        <div style="height: 600px; overflow: hidden">
        <div id="map">
                <script defer
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCpsj6hmHPi6YM7CvYvp-YMtQ
3hoVrgYnI&callback=initMap"></script>
        </div>
        </div>
        </div>

        </div><!-- /.d-md-flex -->
        </div>
        <!-- /.card-body -->
</div>
</div>
<div class="col-lg-5 col-md-5 col-sm-12 col-xs-12">
```

```
<div class="card card-secondary card-outline" style="height: 50%">

<div class="card-header">

<h3 class="card-title">Referencias</h3>

</div>

<div class="card-body p-0">

<br>

 <div class="info-box-content">

<span class="info-box-text"><img src="http://maps.google.com/mapfiles/ms/icons/green-dot.png">Sin alertas activas</span>

</div>

<br>

<div class="info-box-content">

<span class="info-box-text"><img src="http://maps.google.com/mapfiles/ms/icons/blue-dot.png">Alertas de corriente pendientes de revision</span>

</div>

<br>

<div class="info-box-content">

<span class="info-box-text"><img src="http://maps.google.com/mapfiles/ms/icons/yellow-dot.png">Alertas de oscilación pendientes de revision</span>

</div>

<br>

<div class="info-box-content">
```

```
                <span class="info-box-text"><img src="http://maps.google.com/mapfiles/ms/icons/red-dot.png">Alertas de corriente y oscilación pendientes de revision</span>

                </div>

                <br>

                <div class="info-box-content">

                <span class="info-box-text"><img src="http://maps.google.com/mapfiles/ms/icons/purple-dot.png">Dispositivo inactivo</span>

                </div>

                </div>

                 <br>

                </div>

                <div class="card card-secondary card-outline" style="height: 44%">

                <div class="card-header">

                <h3 class="card-title">Dispositivos sin localizar</h3>

                </div>

                <div class="card-body p-0"><div id="jsGrid"></div></div>


        </div>

        </div>

        </div>

<!-- jQuery -->

<script src="{%static 'admin-lte/plugins/jquery/jquery.min.js' %}"></script>

<!-- jsGrid -->

<link type="text/css" rel="stylesheet" href="{%static 'admin-lte/plugins/jsgrid/jsgrid.min.css'%}" />
```

```
<link type="text/css" rel="stylesheet" href="{%static 'admin-lte/plugins/jsgrid/jsgrid-theme.min.css'%}" />


<script type="text/javascript" src="{%static 'admin-lte/plugins/jsgrid/jsgrid.min.js'%}"></script>


<!--Recupero los dispositivos y los paso a variable JS-->

<script>var noLocationDicList=[];</script>

{% for loc in noLocationList %}

 <script>

        var codeDevice="{{loc.codeDevice | safe}}";

        var descDevice="{{loc.descDevice | safe}}";

        var registrationDate="{{loc.registrationDate | safe}}";

        var active="{{loc.active | safe}}";

        var devDic={"codeDevice":codeDevice,"description":descDevice,"registrationDate":registrationDate,"active":active};

        noLocationDicList.push(devDic);

        console.log(noLocationDicList);

 </script>

{%endfor%}

<!--TABLA-->


<script>

        $("#jsGrid").jsGrid({

        width: "100%",
```

```
        height: "100%",

        sorting: true,
        paging: true,
        pageSize: 2,

        noDataContent: "Todos los dipositivos se encuentran localizados",
        //Data de la tabla
        data: noLocationDicList,

        fields: [
        { name: "codeDevice", title:"Codigo", type: "text", width: 200},
        { name: "description", title:"Descripcion", type: "text", width: 200},
        { name: "registrationDate", title:"Fecha de registro", type: "text", width: 200 }
        ]
        });
</script>

{% endblock %}
```

- **/PaginaWeb/Templates/extends_template.html**

```
{% extends 'base_template.html' %}
{%block devices_menu%}menu-open{%endblock%}
{%block devices%}active{%endblock%}
```

```
{%block devices_locations%}active{%endblock%}
{% block title%}Geolocalizacion{%endblock%}

{% block content %}
{% endblock %}
```

- **/PaginaWeb/Templates/index.html**

```
{% extends 'base_template.html' %}
{%block index%}active{%endblock%}
{% block title%}Inicio{%endblock%}
{% block content %}
 <!-- Ionicons -->
                                        <link rel="stylesheet"
href="https://code.ionicframework.com/ionicons/2.0.1/css/ionicons.min.css">

      <!-- DIV CUADROS (FILA) -->
      <div class="row">

      <!-- CUADRO VERDE -->
      <div class="col-lg-3 col-md-3 col-sm-12 col-xs-12">
      <!-- small box -->
      <div class="small-box bg-success">
        <div class="inner">
         <h3>{{quantityDevicesActive}}<sup style="font-size: 20px"></sup></h3>
```

```
            <p>Dispositivos activos</p>
        </div>
        <div class="icon">
            <i class="ion ion-connection-bars"></i>
        </div>
        <a href="/devices_list" class="small-box-footer">Lista de dispositivos <i class="fas fa-arrow-circle-right"></i></a>
        </div>
        </div>
        <!-- CUADRO VERDE -->
        <!-- CUADRO AZUL -->
        <div class="col-lg-3 col-md-3 col-sm-12 col-xs-12">
        <!-- small box -->
        <div class="small-box bg-info">
        <div class="inner">
          <h3>{{quantityCurAlert}}<sup style="font-size: 20px"></sup></h3>

            <p>Alertas de corriente sin revisar</p>
        </div>
        <div class="icon">
            <i class="ion ion-arrow-graph-up-right"></i>
        </div>
```

Alertas de corriente

<!-- CUADRO AZUL -->

<!-- CUADRO AMARILLO -->

<!-- small box -->

<h3>{{quantityOscAlert}}</h3>

Alertas de oscilacion sin revisar

Alertas de oscilacion

<!-- CUADRO AMARILLO -->

<!-- CUADRO ROJO -->

```
<div class="col-lg-3 col-md-3 col-sm-12 col-xs-12">
<!-- small box -->
<div class="small-box bg-danger">
<div class="inner">
   <h3>{{quantityDevicesInactive}}<sup style="font-size: 20px"></sup></h3>

        <p>Dispositivos inactivos</p>
</div>
<div class="icon">
        <i class="ion ion-close-circled"></i>
</div>
<a href="/devices_locations" class="small-box-footer">Geolocalización <i class="fas fa-arrow-circle-right"></i></a>
</div>
</div>
<!-- CUADRO ROJO -->

</div>
<div class="row" style="display:none;">
<div class="col-lg-6">
<div class="card">
<div class="card-body">
        <h5 class="card-title">Card title</h5>
```

```
<p class="card-text">
Some quick example text to build on the card title and make up the bulk of the card's
content.
</p>

<a href="#" class="card-link">Card link</a>
<a href="#" class="card-link">Another link</a>
</div>
</div>

<div class="card card-primary card-outline" style="display:none;">
<div class="card-body">
<h5 class="card-title">Card title</h5>

<p class="card-text">
Some quick example text to build on the card title and make up the bulk of the card's
content.
</p>
<a href="#" class="card-link">Card link</a>
<a href="#" class="card-link">Another link</a>
</div>
</div><!-- /.card -->
</div>
```

```
<!-- /.col-md-6 -->
<div class="col-lg-6" style="display:none;">
<div class="card">
<div class="card-header">
        <h5 class="m-0">Featured</h5>
    </div>
<div class="card-body">
        <h6 class="card-title">Special title treatment</h6>

        <p class="card-text">With supporting text below as a natural lead-in to additional content.</p>
        <a href="#" class="btn btn-primary">Go somewhere</a>
</div>
</div>

<div class="card card-primary card-outline" style="display:none;">
<div class="card-header">
        <h5 class="m-0">Featured</h5>
</div>
<div class="card-body">
        <h6 class="card-title">Special title treatment</h6>

        <p class="card-text">With supporting text below as a natural lead-in to additional content.</p>
        <a href="#" class="btn btn-primary">Go somewhere</a>
```

```
        </div>
        </div>
        </div>
        <!-- /.col-md-6 -->
        </div>
        <!-- /.row -->
{% endblock %}
```

- **/PaginaWeb/Templates/measurement_current_device.html**

```
{% extends 'base_template.html' %}
{%load static%}
{%block measurement_menu%}menu-open{%endblock%}
{%block measurement%}active{%endblock%}
{%block measurement_current%}active{%endblock%}
{% block title%}Mediciones de corriente del dispositivo {%endblock%}

{% block content %}
<div class="row">
  <div class="col-12">
        <!-- interactive chart -->
        <div class="card card-primary card-outline">
        <div class="card-header">
        <h3 class="card-title">
        <i class="far fa-chart-bar"></i>
```

```
            Mediciones de corriente del dispositivo
          </h3>
          </div>
          <div class="card-body">
          <canvas id="areaChart" style="min-height: 250px; height: 250px; max-height: 250px;
max-width: 100%;"></canvas>
          </div>
          <!-- /.card-body-->
          </div>
          <!-- /.card -->

  </div>
  <!-- /.col -->
</div>
<!-- /.row -->
<!-- Control Sidebar -->
<aside class="control-sidebar control-sidebar-dark">
<!-- Control sidebar content goes here -->
</aside>
<!-- /.control-sidebar -->

<!-- jQuery -->
<script src="{%static 'admin-lte/plugins/jquery/jquery.min.js'%}"></script>
```

```
<!-- Bootstrap 4 -->

<script src="{%static 'admin-lte/plugins/bootstrap/js/bootstrap.bundle.min.js'%}"></script>

<!-- FLOT RESIZE PLUGIN - allows the chart to redraw when the window is resized -->
<script src="{%static 'admin-lte/plugins/chart.js/Chart.min.js'%}"></script>
<!-- FLOT PIE PLUGIN - also used to draw donut charts -->
<script>var measurementDicList=[];</script>
<script>var measurementList=[];</script>
<script>var measurementDatetimeList=[];</script>
{% for measurement in measurementList %}
<script>
var measurementId="{{measurement.measurementId | safe}}";
var codeDevice="{{measurement.codeDevice | safe}}";
var descDevice="{{measurement.descDevice | safe}}";
var registrationDate="{{measurement.registrationDate | safe}}";
var active="{{measurement.active | safe}}";
var measurementDatetime="{{measurement.measurementDatetime | safe}}";
var measurementValue="{{measurement.measurementValue | safe}}";
var
measurementDic={"codeDevice":codeDevice,"description":descDevice,"registrationDate":reg
istrationDate,"active":active,"measurementDatetime":measurementDatetime,"measurementVa
lue":measurementValue};
//le elimino los segundos y todo lo que va atras
measurementDatetime=measurementDatetime.slice(0,measurementDatetime.lastIndexOf("+")
);
```

```
measurementDatetime=measurementDatetime.slice(0,measurementDatetime.lastIndexOf(":"))
;


measurementDicList.push(measurementDic);

measurementDatetimeList.push(measurementDatetime);

measurementList.push(measurementValue);

console.log(measurementList);

</script>

{%endfor%}


<script>

//--------------

        //- AREA CHART -

        //--------------


        // Get context with jQuery - using jQuery's .get() method.

        var areaChartCanvas = $('#areaChart').get(0).getContext('2d')


        var areaChartData = {

        labels  : measurementDatetimeList,

        datasets: [

        {

        label               : 'Medición: ',

        backgroundColor       : 'rgba(60,141,188,0.9)',
```

```
borderColor             : 'rgba(0,0,0,0.8)',
pointRadius             : 4,
pointColor      : '',
pointStrokeColor        : 'rgba(60,141,188,1)',
pointHighlightFill  : '#fff',
pointHighlightStroke: 'rgba(60,141,188,1)',
data                : measurementList
},


]
}

var areaChartOptions = {
maintainAspectRatio : false,
responsive : true,
legend: {
display: false
},
scales: {
xAxes: [{
gridLines : {
display : true,
},
```

```
        }],
        yAxes: [{
        gridLines : {
        display : true,
        },
        ticks: {
                beginAtZero: true
                }
        }]
        }
        }

        // This will get the first returned node in the jQuery collection.
        var areaChart               = new Chart(areaChartCanvas, {
        type: 'line',
        data: areaChartData,
        options: areaChartOptions
        })
</script>
{% endblock %}
```

- **/PaginaWeb/Templates/measurement_current.html**

```
{% extends 'base_template.html' %}
{%load static%}
```

```
{%block measurement_menu%}menu-open{%endblock%}

{%block measurement%}active{%endblock%}

{%block measurement_current%}active{%endblock%}

{% block title%}Mediciones de corriente{%endblock%}


{% block content %}

<!-- Main content -->


<div class="card card-primary card-outline">

  <div class="card-header">

        <h3 class="card-title">Mediciones de corriente</h3>

  </div>

  <!-- /.card-header -->

  <div class="card-body">

        <div id="jsGrid"></div>

  </div>

  <!-- /.card-body -->

</div>

<!-- jQuery -->

<script src="{%static 'admin-lte/plugins/jquery/jquery.min.js' %}"></script>

<!-- jsGrid -->

<link type="text/css" rel="stylesheet" href="{%static
'admin-lte/plugins/jsgrid/jsgrid.min.css'%}" />

<link type="text/css" rel="stylesheet" href="{%static
'admin-lte/plugins/jsgrid/jsgrid-theme.min.css'%}" />
```

```
<script type="text/javascript" src="{%static 'admin-lte/plugins/jsgrid/jsgrid.min.js'%}"></script>


<!--Recupero los dispositivos y los paso a variable JS-->

<script>var measurementDicList=[];</script>

{% for measurement in measurementList %}

<script>

var measurementId="{{measurement.measurementId | safe}}";

var codeDevice="{{measurement.codeDevice | safe}}";

var descDevice="{{measurement.descDevice | safe}}";

var registrationDate="{{measurement.registrationDate | safe}}";

var active="{{measurement.active | safe}}";

var measurementDatetime="{{measurement.measurementDatetime | safe}}";

var measurementValue="{{measurement.measurementValue | safe}}";

//le elimino los segundos y todo lo que va atras

measurementDatetime=measurementDatetime.slice(0,measurementDatetime.lastIndexOf("+")
);

measurementDatetime=measurementDatetime.slice(0,measurementDatetime.lastIndexOf(":"))
;


var
measurementDic={"codeDevice":codeDevice,"description":descDevice,"registrationDate":reg
istrationDate,"active":active,"measurementDatetime":measurementDatetime,"measurementVa
lue":measurementValue};

measurementDicList.push(measurementDic);

console.log(measurementDicList);
```

```
</script>

{%endfor%}

<!--TABLA-->

<script>

  $("#jsGrid").jsGrid({

        width: "100%",

        height: "auto",


        sorting: true,


        paging: true,

        pageSize: 5,

        noDataContent: "No se han registrado alertas",

        data: measurementDicList,

        rowClick: function(item) { window.location="/measurement_current/"+item['item']['codeDevice']; },

        fields: [

        { name: "codeDevice", title:"Codigo del dispositivo", type: "text", editing: false, width: 200},

        { name: "description", title:"Descripcion del dispositivo", type: "text", editing: false, width: 200},

        { name: "measurementDatetime", title:"Fecha y hora", type: "text", editing: false, width: 200,  },

        { name: "measurementValue", title:"Valor", type: "text", editing: false, width: 200, },
```

```
        ]
  });
</script>
{% endblock %}
```

- **/PaginaWeb/Templates/measurement_oscilation.html**

```
{% extends 'base_template.html' %}
{%load static%}
{%block measurement_menu%}menu-open{%endblock%}
{%block measurement%}active{%endblock%}
{%block measurement_oscillation%}active{%endblock%}
{% block title%}Mediciones de oscilación{%endblock%}

{% block content %}
<!-- Main content -->

<div class="card card-primary card-outline">
  <div class="card-header">
        <h3 class="card-title">Mediciones de oscilación</h3>
  </div>
  <!-- /.card-header -->
  <div class="card-body">
        <div id="jsGrid"></div>
  </div>
```

```
  <!-- /.card-body -->

</div>

<!-- jQuery -->

<script src="{%static 'admin-lte/plugins/jquery/jquery.min.js' %}"></script>

<!-- jsGrid -->

<link type="text/css" rel="stylesheet" href="{%static 'admin-lte/plugins/jsgrid/jsgrid.min.css'%}" />

<link type="text/css" rel="stylesheet" href="{%static 'admin-lte/plugins/jsgrid/jsgrid-theme.min.css'%}" />


<script type="text/javascript" src="{%static 'admin-lte/plugins/jsgrid/jsgrid.min.js'%}"></script>

<!--Recupero los dispositivos y los paso a variable JS-->

<script>var measurementDicList=[];</script>

{% for measurement in measurementList %}

<script>

var measurementId="{{measurement.measurementId | safe}}";

var codeDevice="{{measurement.codeDevice | safe}}";

var descDevice="{{measurement.descDevice | safe}}";

var registrationDate="{{measurement.registrationDate | safe}}";

var active="{{measurement.active | safe}}";

var measurementDatetime="{{measurement.measurementDatetime | safe}}";

var measurementCondition="{{measurement.measurementCondition | safe}}";

//le elimino los segundos y todo lo que va atras

measurementDatetime=measurementDatetime.slice(0,measurementDatetime.lastIndexOf("+")
);
```

```
measurementDatetime=measurementDatetime.slice(0,measurementDatetime.lastIndexOf(":"))
;

var
measurementDic={"codeDevice":codeDevice,"description":descDevice,"registrationDate":reg
istrationDate,"active":active,"measurementDatetime":measurementDatetime,"measurementCo
ndition":measurementCondition};

measurementDicList.push(measurementDic);

console.log(measurementDicList);

</script>

{%endfor%}

<!--TABLA-->

<script>

  $("#jsGrid").jsGrid({

        width: "100%",

        height: "auto",


        sorting: true,


        paging: true,

        pageSize: 5,


        noDataContent: "No se han registrado alertas",


        data: measurementDicList,
```

```
        fields: [

        { name: "codeDevice", title:"Codigo del dispositivo", type: "text", editing: false, width: 200},

        { name: "description", title:"Descripcion del dispositivo", type: "text", editing: false, width: 200},

        { name: "measurementDatetime", title:"Fecha y hora", type: "text", editing: false, width: 200 },

        { name: "measurementCondition", title:"Condición", type: "text", editing: false, width: 200,

        itemTemplate: function(viewed, item) {

        if(item.measurementCondition=="True"){

        return "OK";

        }

        else{

        return "Oscilación elevada";

        }

        }}

        ]

  });

</script>

{% endblock %}
```

- **/ PaginaWeb/apps.py**

```
from django.apps import AppConfig

class PaginawebConfig(AppConfig):

        name = 'PaginaWeb'
```

- **/PaginaWeb/models.py**

```
from django.db import models


class Device(models.Model):
        codeDevice = models.CharField(max_length=200)
        descDevice = models.CharField(max_length=500)
        registrationDate=models.DateTimeField()
        active=models.BooleanField(default=True)


class Location(models.Model):
        deviceOwner=models.ForeignKey(Device, on_delete=models.CASCADE)
        lat=models.FloatField(max_length=200)
        lng=models.FloatField(max_length=200)
        datetime=models.DateTimeField()


class CurrentMeasurement(models.Model):
        deviceOwner=models.ForeignKey(Device, on_delete=models.CASCADE)
        datetime=models.DateTimeField()
        value=models.FloatField()


class OscillationMeasurement(models.Model):
        deviceOwner=models.ForeignKey(Device, on_delete=models.CASCADE)
        datetime=models.DateTimeField()
```

```
    condition=models.BooleanField(default=True)#si es true esta OK, si es false esta oscilando mucho


class CurrentAlert(models.Model):
        deviceOwner=models.ForeignKey(Device, on_delete=models.CASCADE)
        datetime=models.DateTimeField()
        viewed=models.BooleanField(default=False)
        alertType= models.CharField(max_length=200)


class OscillationAlert(models.Model):
        deviceOwner=models.ForeignKey(Device, on_delete=models.CASCADE)
        datetime=models.DateTimeField()
        viewed=models.BooleanField(default=False)
```

- **/PaginaWeb/views.py**

```
from django.shortcuts import render


from django.http import HttpResponse
from datetime import datetime
from django.template import Template, Context
from django.template import loader
from PaginaWeb.models import Device,Location, CurrentAlert, OscillationAlert, CurrentMeasurement, OscillationMeasurement
import json
```

```
# Create your views here.

from PaginaWeb.models import Device


def ver_dispositivos(request):

        devices=Device.objects.all()

        devList=[]

        for dev in devices:


devList.append({'codeDevice':dev.codeDevice,'registrationDate':dev.registrationDate.strftime(
"%d-%b-%Y (%H:%M:%S.%f)")})#es una lista de diccionarios, necesito que sea diccionario
para poder manejar variables en la plantilla

        devDic={'devList':devList} # paso como parametro el diccionario, en la plantilla mis
variables seran las claves del diccionario, en este caso solo tenemos devList

        return render(request,'plantilla_dispositivos.html',devDic)


def registro_dispositivo(request):

        return render(request,'registro_dispositivo.html')


def nuevo_dispositivo(request):


newDevice=Device(codeDevice=request.GET['id_device'],descDevice=request.GET['desc'],re
gistrationDate=datetime.now())

        newDevice.save()

        return render(request,'registro_exitoso.html')

def index(request):

        basicInfoDic=basic_information()

    return render(request,'index.html',basicInfoDic)
```

```
def devices_list(request):
        basicInfoDic=basic_information()
        devicesList=Device.objects.all()
        devDic={'devicesList':devicesList}
        devDic.update(basicInfoDic)
        return render(request,'devices_list.html',devDic)


def devices_new(request,codeDevice,descDevice):

newDevice=Device(codeDevice=codeDevice,descDevice=descDevice,registrationDate=dateti
me.now())
        newDevice.save()
        return devices_list(request)


def devices_delete(request,codeDevice):
    Device.objects.filter(codeDevice=codeDevice).delete()
        return devices_list(request)


def devices_edit(request,deviceId,codeDevice,descDevice):

Device.objects.filter(id=deviceId).update(codeDevice=codeDevice,descDevice=descDevice)
        return devices_list(request)


def devices_locations(request):
```

#falta agregar logica de localizacion: cuando la ultima localizacion fue hace mas de x hs el atributo "active" pasaria a ser false. esto lo hace un proceso externo. pero si no tiene localizacion registrada en la tabla tambien deberiamos mostrarlo

basicInfoDic=basic_information()

locations=Location.objects.all()

locationList=[]

noLocationList=[]

devices=Device.objects.all()

#busca alertas que no hayan sido vistas

for location in locations:

curAlert=len(CurrentAlert.objects.filter(deviceOwner=location.deviceOwner,viewed=False))

if(curAlert>0):

curAlertViewed=False

else:

curAlertViewed=True

oscAlert=len(OscillationAlert.objects.filter(deviceOwner=location.deviceOwner,viewed=Fals e))

if(oscAlert>0):

oscAlertViewed=False

else:

oscAlertViewed=True

locationList.append({'codeDevice':location.deviceOwner.codeDevice,'descDevice':location.de

```
viceOwner.descDevice,'active':location.deviceOwner.active,'lat':location.lat,'lng':location.lng,
'curAlertViewed':curAlertViewed,'oscAlertViewed':oscAlertViewed})


        #selecciona los dispositivos registrados que no han sido localizados nunca

        for dev in devices:

        if(not Location.objects.filter(deviceOwner=dev).exists()):

                                                noLocationList.append({'codeDevice':
dev.codeDevice,'descDevice':dev.descDevice,'registrationDate':dev.registrationDate,'active':de
v.active})


        locDic={'locationList':locationList,'noLocationList': noLocationList}

        locDic.update(basicInfoDic)

        return render(request,'devices_locations.html',locDic)


def alert_current(request):

        basicInfoDic=basic_information()

        currentAlert=CurrentAlert.objects.all()

        currentAlertList=[]

        for curAlert in currentAlert:


currentAlertList.append({'alertId':curAlert.id,'codeDevice':curAlert.deviceOwner.codeDevice,
'descDevice':curAlert.deviceOwner.descDevice,'registrationDate':curAlert.deviceOwner.regist
rationDate,'active':curAlert.deviceOwner.active,'alertDatetime':curAlert.datetime,'viewed':cur
Alert.viewed,"alertType":curAlert.alertType})

    currentAlertDic={'currentAlertList':currentAlertList}

        currentAlertDic.update(basicInfoDic)

        return render(request,'alert_current.html',currentAlertDic)
```

```
def alert_current_edit(request,alertId,viewed):
    curAlert=CurrentAlert.objects.get(id=alertId)
        curAlert.viewed=viewed
        curAlert.save()
        return alert_current(request)


def alert_oscillation(request):
        basicInfoDic=basic_information()
    oscillationAlert=OscillationAlert.objects.all()
        oscillationAlertList=[]
        for oscAlert in oscillationAlert:

oscillationAlertList.append({'alertId':oscAlert.id,'codeDevice':oscAlert.deviceOwner.codeDev
ice,'descDevice':oscAlert.deviceOwner.descDevice,'registrationDate':oscAlert.deviceOwner.re
gistrationDate,'active':oscAlert.deviceOwner.active,'alertDatetime':oscAlert.datetime,'viewed':
oscAlert.viewed})
    oscillationAlertDic={'oscillationAlertList':oscillationAlertList}
        oscillationAlertDic.update(basicInfoDic)
        return render(request,'alert_oscillation.html',oscillationAlertDic)


def alert_oscillation_edit(request,alertId,viewed):
    oscAlert=OscillationAlert.objects.get(id=alertId)
        oscAlert.viewed=viewed
        oscAlert.save()
        return alert_oscillation(request)
```

```
def measurement_current(request):
        basicInfoDic=basic_information()
        currentMeasurement=CurrentMeasurement.objects.filter()
        measurementList=[]
        for measurement in currentMeasurement:

measurementList.append({'measurementId':measurement.id,'codeDevice':measurement.device
Owner.codeDevice,'descDevice':measurement.deviceOwner.descDevice,'registrationDate':mea
surement.deviceOwner.registrationDate,'active':measurement.deviceOwner.active,'measureme
ntDatetime':measurement.datetime,'measurementValue':measurement.value})
    currentMeasurementDic={'measurementList':measurementList}
        currentMeasurementDic.update(basicInfoDic)
        return render(request,'measurement_current.html',currentMeasurementDic)


def measurement_current_device(request,codeDevice):
        basicInfoDic=basic_information()
        dev=Device.objects.get(codeDevice=codeDevice)
        #recupera los ultimos 20 registros

currentMeasurement=CurrentMeasurement.objects.filter(deviceOwner=dev).order_by('dateti
me')[:20]
        measurementList=[]
        for measurement in currentMeasurement:

measurementList.append({'measurementId':measurement.id,'codeDevice':measurement.device
Owner.codeDevice,'descDevice':measurement.deviceOwner.descDevice,'registrationDate':mea
surement.deviceOwner.registrationDate,'active':measurement.deviceOwner.active,'measureme
ntDatetime':measurement.datetime,'measurementValue':measurement.value})
```

```
    currentMeasurementDic={'measurementList':measurementList}
        currentMeasurementDic.update(basicInfoDic)

        return      render(request,'measurement_current_device.html',currentMeasurementDic)

def measurement_oscillation(request):
        basicInfoDic=basic_information()
    oscillationMeasurement=OscillationMeasurement.objects.filter()
        measurementList=[]
    for measurement in oscillationMeasurement:

measurementList.append({'measurementId':measurement.id,'codeDevice':measurement.device
Owner.codeDevice,'descDevice':measurement.deviceOwner.descDevice,'registrationDate':mea
surement.deviceOwner.registrationDate,'active':measurement.deviceOwner.active,'measureme
ntDatetime':measurement.datetime,'measurementCondition':measurement.condition})
    oscillationMeasurementDic={'measurementList':measurementList}
    oscillationMeasurementDic.update(basicInfoDic)
    return render(request,'measurement_oscillation.html',oscillationMeasurementDic)

def basic_information():
        try:
        user="user"
        except Exception as ObjectDoesNotExist:
        #usuario no existe
        user=None
```

```
        quantityDevices=Device.objects.filter().count()
    quantityDevicesActive=Device.objects.filter(active=True).count()
    quantityDevicesInactive=Device.objects.filter(active=False).count()
    quantityCurAlert=CurrentAlert.objects.filter(viewed=False).count()
        quantityOscAlert=OscillationAlert.objects.filter(viewed=False).count()


    quantityTotAlert=quantityCurAlert+quantityOscAlert


        #quantityTotAlert=0

basicInfoDic={"user":user,"quantityDevices":quantityDevices,"quantityDevicesActive":quant
ityDevicesActive,"quantityDevicesInactive":quantityDevicesInactive,"quantityCurAlert":quan
tityCurAlert,"quantityOscAlert":quantityOscAlert,"quantityTotAlert":quantityTotAlert}


        return basicInfoDic
```

- **/manage.py**

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys



def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'AplicacionWebPFI.settings')
```

```
        try:
        from django.core.management import execute_from_command_line
        except ImportError as exc:
        raise ImportError(
        "Couldn't import Django. Are you sure it's installed and "
        "available on your PYTHONPATH environment variable? Did you "
        "forget to activate a virtual environment?"
        ) from exc
        execute_from_command_line(sys.argv)


if __name__ == '__main__':
        main()
```

- **/Procfile**

```
web: gunicorn AplicacionWebPFI.wsgi
release: python manage.py migrate
```

- **/requirements.txt**

```
aiohttp==3.7.4.post0
alembic==1.5.4
aniso8601==8.1.1
asgiref==3.4.1
async-timeout==3.0.1
```

attrs==21.2.0

backports.entry-points-selectable==1.1.0

certifi==2020.12.5

chardet==4.0.0

Click==7.0

dateparser==1.0.0

distlib==0.3.2

dj-database-url==0.5.0

Django==3.2.9

django-adminlte3==0.1.6

filelock==3.0.12

Flask==1.0.2

flask-marshmallow==0.14.0

Flask-Migrate==2.6.0

Flask-RESTful==0.3.8

Flask-SQLAlchemy==2.4.4

gunicorn==20.1.0

idna==2.10

importlib-metadata==4.6.1

itsdangerous==1.1.0

Jinja2==2.10

Mako==1.1.4

MarkupSafe==1.1.1

marshmallow==3.10.0

multidict==5.1.0

mysqlclient==2.0.3

numpy==1.20.3

pandas==1.2.4

platformdirs==2.0.2

pymodbus==2.4.0

PyMySQL==1.0.2

pyserial==3.5

python-binance==1.0.10

python-dateutil==2.8.1

python-decouple==3.5

python-editor==1.0.4

pytz==2021.1

regex==2021.4.4

requests==2.25.1

six==1.15.0

SQLAlchemy==1.3.20

sqlparse==0.4.2

typing-extensions==3.10.0.0

tzlocal==2.1

ujson==4.0.2

urllib3==1.26.5

virtualenv==20.5.0

websockets==9.1

Werkzeug==0.14.1

whitenoise==5.3.0

yarl==1.6.3

zipp==3.5.0

7.4 Anexo 4

7.4.1 Código de la API Rest

A continuación, se incluye el código de los archivos que han sido autogenerados por el framework y posteriormente modificados para el desarrollo de la API Rest mencionada en el documento principal. Cabe destacar que en el archivo /requirements.txt se incluyen las librerías utilizadas y su correspondiente versión.

- **/app/devices/api_v1_0/resources.py**

  from flask import request, Blueprint

  from flask_restful import Api, Resource

  from ..models import Device, CurrentMeasurement, OscillationMeasurement, CurrentAlert, OscillationAlert, Location

  import datetime

  import base64

  import struct

  apirest_mediciones = Blueprint('apirest_mediciones', __name__)

  api = Api(apirest_mediciones)

  class NuevaMedicionResource(Resource):

  def get(self):

  return('alguien llego')

  '''

```
    messages=[
    {"description":'Nueva medicion de corriente',"value":'cm'},
    {"description":'Nueva medicion de oscilacion',"value":'om'},
    {"description":'Nueva alerta de corriente',"value":'ca'},
    {"description":'Nueva alerta de oscilacion',"value":'oa'},
    {"description":'Nueva localizacion',"value":'l'}
    ]
    '''
    def post(self):
    data = request.get_json()
    deviceExists=False
  storedDevices=Device.get_all()
    print(data)
    for device in storedDevices:
    if device.codeDevice==data["deveui"]:
            deviceExists=True
            break

    if deviceExists==True:
    print("El dipositivo existe")
    print(device.codeDevice)
    date_time_str = data["timestamp"]
                        date_time_obj=datetime.datetime.strptime(date_time_str,
"%Y-%m-%dT%H:%M:%S.%fZ")
```

```
        message=data["dataFrame"]
        if message!='0a'and message!='0b' and message!='0c' and data["port"]!=3:
           message=str(message).replace(" ","")
                if len(message)<8:
              message="00000000"+message
                if len(message)>8:
              message=message[-8:]
                print(message)
           message=struct.unpack('!f', bytes.fromhex(message))[0]


        if data["port"]==2:

newMeassurementC=CurrentMeasurement(datetime=date_time_obj,value=struct.unpa
ck('!f', bytes.fromhex(message))[0],deviceOwner_id=device.id)
                newMeassurementC.save()
newMeassurementO=OscillationMeasurement(datetime=date_time_obj,condition=Tru
e,deviceOwner_id=device.id)
                newMeassurementO.save()
        elif data["port"]==20 or data["port"]==21 or data["port"]==30 or
data["port"]==31:
                b=str(message).split('.')
                      c=float(b[0])/100
                d=str(c).split('.')
                      if len(d[1])==2:
                      print(d[1])
                 elif len(d[1])<2 and int(d[1])!=0:
```

```
            d[1]=str(int(d[1])*10)
            print(int(d[1])*10)
                else:
                d[1]='00'
                print(d[1])
        degrees=float(d[0])
        minutes=float(d[1]+'.'+b[1])
        decimal=degrees+(minutes/60)
                coordinate=decimal
                print(coordinate)
                if data["port"]==20:
                lat=coordinate
                    locationExist=Location.simple_filter(deviceOwner_id=device.id)
            if(locationExist==[]):

newLocation=Location(datetime=date_time_obj,lat=lat,lng=0,deviceOwner_id=device
.id)
                newLocation.save()
                else:
                locationExist[0].datetime=date_time_obj
             locationExist[0].lat=lat
                locationExist[0].save()
                elif data["port"]==21:


            lat=(-1)*coordinate
```

```
            locationExist=Location.simple_filter(deviceOwner_id=device.id)


            if(locationExist==[]):

newLocation=Location(datetime=date_time_obj,lat=lat,lng=0,deviceOwner_id=device
.id)
                newLocation.save()
                else:
                locationExist[0].datetime=date_time_obj
                locationExist[0].lat=lat
                locationExist[0].save()
                elif data["port"]==30:
                lng=coordinate
            locationExist=Location.simple_filter(deviceOwner_id=device.id)
            if(locationExist==[]):

newLocation=Location(datetime=date_time_obj,lat=0,lng=lng,deviceOwner_id=devic
e.id)
                 newLocation.save()
                else:
                locationExist[0].datetime=date_time_obj
                locationExist[0].lng=lng
                locationExist[0].save()
                elif data["port"]==31:
            lng=(-1)*coordinate
            locationExist=Location.simple_filter(deviceOwner_id=device.id)
```

```
                if(locationExist==[]):

newLocation=Location(datetime=date_time_obj,lat=0,lng=lng,deviceOwner_id=devic
e.id)

                    newLocation.save()

                    else:

                    locationExist[0].datetime=date_time_obj

                locationExist[0].lng=lng

                    locationExist[0].save()

        elif message=='0a' and data["port"]==3:

            print("mensaje A")


newAlert=CurrentAlert(datetime=date_time_obj,viewed=False,deviceOwner_id=devic
e.id,alertType="Alto consumo")

                    print(newAlert)

                    newAlert.save()

        elif message=='0b' and data["port"]==3:


newAlert=CurrentAlert(datetime=date_time_obj,viewed=False,deviceOwner_id=devic
e.id,alertType="Bajo consumo")

                    newAlert.save()

        elif message=='0c' and data["port"]==3:


newAlert=OscillationAlert(datetime=date_time_obj,viewed=False,deviceOwner_id=d
evice.id)

                    newAlert.save()

        else:

                    print("El tipo de mensaje no se identifico")
```

```
        else:

        print("El dispositivo no fue registrado en la web")

api.add_resource(NuevaMedicionResource, '/api/rest/callback/payloads/ul', endpoint='nueva_medicion')
```

- **/app/devices/models.py**

```
from app.db import db, BaseModelMixin

class Device(db.Model, BaseModelMixin):

  __tablename__="PaginaWeb_device"

        id = db.Column(db.Integer, primary_key=True, nullable=False, autoincrement=True)

        codeDevice = db.Column(db.String(200), nullable=False)

        registrationDate = db.Column(db.DateTime, nullable=False)

        descDevice = db.Column(db.String(500), nullable=False)

        active = db.Column(db.Boolean, nullable=False)


class CurrentMeasurement(db.Model, BaseModelMixin):

  __tablename__="PaginaWeb_currentmeasurement"

        id = db.Column(db.Integer, primary_key=True, nullable=False, autoincrement=True)

        datetime = db.Column(db.DateTime, nullable=False)

        value = db.Column(db.Float, nullable=False)

        deviceOwner_id = db.Column(db.Integer, db.ForeignKey('PaginaWeb_device.id'),nullable=False)


class OscillationMeasurement(db.Model, BaseModelMixin):
```

```
    __tablename__="PaginaWeb_oscillationmeasurement"

    id = db.Column(db.Integer, primary_key=True, nullable=False, autoincrement=True)

    datetime = db.Column(db.DateTime, nullable=False)

    condition = db.Column(db.Boolean, nullable=False)

    deviceOwner_id = db.Column(db.Integer, db.ForeignKey('PaginaWeb_device.id'),nullable=False)


class CurrentAlert(db.Model, BaseModelMixin):

  __tablename__="PaginaWeb_currentalert"

    id = db.Column(db.Integer, primary_key=True, nullable=False, autoincrement=True)

    datetime = db.Column(db.DateTime, nullable=False)

    viewed = db.Column(db.Boolean, nullable=False)

    deviceOwner_id = db.Column(db.Integer, db.ForeignKey('PaginaWeb_device.id'),nullable=False)

    alertType = db.Column(db.String(200), nullable=True)


class OscillationAlert(db.Model, BaseModelMixin):

  __tablename__="PaginaWeb_oscillationalert"

    id = db.Column(db.Integer, primary_key=True, nullable=False, autoincrement=True)

    datetime = db.Column(db.DateTime, nullable=False)

    viewed = db.Column(db.Boolean, nullable=False)

    deviceOwner_id = db.Column(db.Integer, db.ForeignKey('PaginaWeb_device.id'),nullable=False)
```

```
class Location(db.Model, BaseModelMixin):

   __tablename__="PaginaWeb_location"

        id = db.Column(db.Integer, primary_key=True, nullable=False, autoincrement=True)

        lat = db.Column(db.Float, nullable=False)

        lng = db.Column(db.Float, nullable=False)

        deviceOwner_id = db.Column(db.Integer, db.ForeignKey('PaginaWeb_device.id'),nullable=False)

        datetime = db.Column(db.DateTime, nullable=False)
```

- **/app/db.py**

```
from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

class BaseModelMixin:

        def save(self):

        db.session.add(self)

        db.session.commit()

        def delete(self):

        db.session.delete(self)

        db.session.commit()

        @classmethod

        def get_all(cls):

        return cls.query.all()

        @classmethod

        def get_by_id(cls, id):
```

```
        return cls.query.get(id)

        @classmethod

        def simple_filter(cls, **kwargs):

        return cls.query.filter_by(**kwargs).all()
```

- **/app/__init__.py**

```
from flask import Flask, jsonify

from flask_restful import Api

from app.common.error_handling import ObjectNotFound, AppErrorBaseClass

from app.db import db

from app.devices.api_v1_0.resources import apirest_mediciones

from .ext import ma, migrate

import pymysql


pymysql.install_as_MySQLdb()


def create_app(settings_module):

        app = Flask(__name__)

   app.config.from_object(settings_module)

                                app.config['SQLALCHEMY_DATABASE_URI'] =
'mysql://agustin:agustin123@66.97.34.41:3306/pfi_telemetria'

        # Inicializa las extensiones

        db.init_app(app)

        ma.init_app(app)
```

```
        migrate.init_app(app, db)
    # Captura todos los errores 404
        Api(app, catch_all_404s=True)
        # Deshabilita el modo estricto de acabado de una URL con /
        app.url_map.strict_slashes = False
        # Registra los blueprints
    app.register_blueprint(apirest_mediciones)
        # Registra manejadores de errores personalizados
        register_error_handlers(app)
        return app
def register_error_handlers(app):
        @app.errorhandler(Exception)
        def handle_exception_error(e):
        return jsonify({'msg': 'Internal server error'}), 500
        @app.errorhandler(405)
        def handle_405_error(e):
        return jsonify({'msg': 'Method not allowed'}), 405
        @app.errorhandler(403)
        def handle_403_error(e):
        return jsonify({'msg': 'Forbidden error'}), 403
        @app.errorhandler(404)
    def handle_404_error(e):
        return jsonify({'msg': 'Not Found error'}), 404
    @app.errorhandler(AppErrorBaseClass)
```

```
    def handle_app_base_error(e):
    return jsonify({'msg': str(e)}), 500
@app.errorhandler(ObjectNotFound)
    def handle_object_not_found_error(e):
    return jsonify({'msg': str(e)}), 404
```

- **/config/default.py**

```
SECRET_KEY = '123447a47f563e90fe2db0f56b1b17be62378e31b7cfd3adc776c59ca4c75e2fc512c15f69bb38307d11d5d17a41a7936789'

PROPAGATE_EXCEPTIONS = True

# Database configuration

SQLALCHEMY_DATABASE_URI = 'mysql://pfi_telemetria:pfitelemetria@66.97.34.41:3306/pfi_telemetria'

SQLALCHEMY_TRACK_MODIFICATIONS = False

SHOW_SQLALCHEMY_LOG_MESSAGES = False

ERROR_404_HELP = False
```

- **/Procfile**

```
web: gunicorn entrypoint:app
```

- **/requirements.txt**

```
alembic==1.4.2

aniso8601==8.0.0

click==7.1.2

dj-database-url==0.5.0
```

Flask==1.1.2

flask-marshmallow==0.13.0

Flask-Migrate==2.5.3

Flask-RESTful==0.3.8

Flask-SQLAlchemy==2.4.4

gunicorn==20.0.4

itsdangerous==1.1.0

Jinja2==2.11.2

Mako==1.1.3

MarkupSafe==1.1.1

marshmallow==3.7.1

marshmallow-sqlalchemy==0.23.1

PyMySQL==1.0.2

python-dateutil==2.8.1

python-editor==1.0.4

pytz==2020.1

six==1.15.0

SQLAlchemy==1.3.19

Werkzeug==1.0.1

whitenoise==5.2.0

Capítulo 8: Bibliografía

http://www.energia.gob.ar/contenidos/verpagina.php?idpagina=3583

http://www.enre.gov.ar/web/web.nsf/EnreDef?openpage

https://aplic.cammesa.com/geosadi/?x=-4626656.180163297&y=-7068787.665372873&z=4&layers=5;10;1&labels=;;&transp=1;1;1&filter=

https://ri.itba.edu.ar/bitstream/handle/123456789/239/PROYECTO%20FINAL%20INGENIERIA%20INDUSTRIAL%20MERCADO%20ELECTRICO%20ARGENTINO.pdf?sequence=1&isAllowed=y

https://www.economia.gob.ar/peconomica/informe/notas_tecnicas/22%20NOTA%20TECNICA%20Nivel%20de%20Actividad%20%20inf%2070.pdf

https://www.academia.edu/32841756/Diagn%C3%B3stico_de_Fallas_en_Sistemas_El%C3%A9ctricos_de_Potencia

https://www.djangoproject.com/

https://flask.palletsprojects.com/en/2.0.x/

https://www.python.org/

https://developers.google.com/maps/documentation/javascript/overview

https://www.mysql.com/

https://developer.mozilla.org/es/docs/Web/HTML

https://developer.mozilla.org/es/docs/Web/JavaScript

https://developer.mozilla.org/es/docs/Web/CSS

https://www.heroku.com/

https://www.microchip.com/en-us/development-tools-tools-and-software/embedded-software-center/mplab-code-configurator

https://www.microchip.com/en-us/development-tools-tools-and-software/mplab-xc-compilers