UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

Investigación para desarrollo de dispositivo modular de control de consumo eléctrico inalámbrico seguro y accesible para domótica e industria.

Agustín Parmisano Sabbione - Tomás Francisco Barbieri.

Director: Marrone Luis
Asesor Profesional: Pablo Ridolfi

Agradecimientos

A nuestra familia, a nuestros amigos y amigas,
y a nuestros compañeros y compañeras,
que nos acompañaron y aconsejaron en todo este trayecto de vida.

A nuestro director y a nuestro asesor profesional,
que nos guiaron en este proceso de investigación y desarrollo.

A todos los profesores y profesoras que nos ayudaron.

A todos los luchadores y luchadoras en la historia que buscaron una ciencia
transformadora y una sociedad más justa, libre y soberana.

Por una ciencia informática y una tecnología al servicio de nuestro pueblo.

Índice

1. Introducción

1.1 Motivación

La crisis energética a nivel internacional es visible y afecta a muchos sectores de la sociedad de distinta forma. Teniendo en cuenta que la industrialización e informatización crecen exponencialmente, el panorama es de un consumo excesivo de energía, el cual genera preocupación en cómo va a obtenerse dicha energía, pensando que las fuentes son finitas y las condiciones de cada uno de los países son distintas a la hora de pensar estrategias y soluciones en este tema.
La necesidad de investigar y desarrollar nuevas fuentes de obtención de energías es imperiosa para lograr tecnologías alternativas que solucionen de una vez por todas esta problemática, o aporten a dicha solución. Sin embargo, no podemos dejar de mirar la problemática desde el lado de la mejora continua en la administración de los recursos, disminuyendo el consumo excesivo y otorgándole al usuario la información de lo que está consumiendo para control y estudios futuros.
En este sentido, además de obtener una información muy valiosa del consumo total de un usuario, podríamos analizar en particular el funcionamiento de distintos dispositivos eléctricos o electrónicos, si estos funcionan de forma regular o tienen variaciones en el consumo.
Pensando en esta línea, se propone desarrollar un sistema de control de consumo eléctrico modular y adaptable a cualquier vivienda o establecimiento que facilite el sensado del consumo de cada equipo eléctrico o electrónico, visualizando en tiempo real dichos datos y guardandolos en una base de datos persistente, generando asi estadísticas para su estudio y comparación. Así mismo, el sistema le otorgaría al usuario la posibilidad de programar los dispositivos para que se apaguen o enciendan según una fecha y hora determinada o según la cantidad de electricidad consumida en un período de tiempo.

1.2 Objetivos

En este trabajo se plantea la investigación para el desarrollo del sistema antes mencionado con el objetivo de obtener un control diario y estadísticas periódicas del consumo eléctrico de un hogar, comercio, edificio público o industria, a través de una interfaz web y una aplicación móvil; utilizando software y hardware libres, sensores dedicados y cálculos estadísticos específicos para tal fin.
Se analizaron temas que refieren a las técnicas de diseño de Hardware (HW) modular, herramientas de sensado de consumo eléctrico, comunicación inalámbrica segura entre los dispositivos modulares y servidores, estadísticas de consumos e interfaces de usuario amigables para el control del sistema.
Para lograr el desarrollo de dicha herramienta lo desglosamos en las siguientes tareas:

- Análisis de los distintos tipos de placas de circuito impreso (PCB) según sus posibles prestaciones: WiFi (estándar 802.11), comunicación serial, cantidad de entradas

analógicas y/o digitales, capacidad de adaptación de sensores complejos y capacidad y velocidad de respuesta a órdenes en tiempo real.

- Análisis de los distintos tipos de sensores de consumo eléctrico. Combinaciones de diversas medidas de sensado, ley de ohm, medidores de corriente, medidores de potencia, invasivos y no invasivos, resistividad, tolerancia a picos altos de corriente.
- Análisis de conectividad inalámbrica con distintas plaquetas PCB, velocidad de respuesta, claridad de la recepción y trasmisión de los mensajes, tasa de baudios por comunicación serial, buffering de paquetes, implementación de una estrategia de comunicación segura y confiable para los dispositivos.
- Investigación de alternativas para el desarrollo de posibles servidores para la administración centralizada de los múltiples dispositivos, como soporte para la estructura de la interfaz gráfica para interactuar con los usuarios y con el fin de de almacenar los datos medidos para futura generación de estadísticas.
- Análisis de interfaces de usuario amigables, intuitivas y estables. Realizando una aplicación móvil y una interfaz web.
- Realizar el desarrollo con las herramientas seleccionadas.
- Realizar los distintos test a nivel lógico de servidor.

El sistema en su totalidad consta de módulos de hardware y del sistema de software propiamente dicho que obtiene los datos de los módulos, los almacena y los ofrece en distintas interfaces y aplicaciones. Llevará el nombre de Elektron y cada módulo de hardware se reconocerá como Unidad Elektron (o Elektron como abreviatura).

1.3 Problemática

La informática y las tecnologías son una solución cada vez más rentable para los problemas cotidianos de los humanos. Diversas actividades como la economía, la industria, la educación y la administración son atravesadas cada vez más por las tecnologías de la información. La mayor parte de la población mundial es beneficiada por estas innovadoras soluciones que transforman toda actividad en información útil para ser medida, estudiada y finalmente utilizada para mejorar las estrategias de cómo se gestionan las distintas actividades. Sin embargo, uno de los problemas más emblemáticos de estos rápidos avances tecnológicos, es la contaminación ambiental que estos producen, en materia de desechos electrónicos, y la cantidad de energía eléctrica que se consume para hacer funcionar los aparatos electrónicos, pieza vital de la informática y sus derivados.

Nuestra investigación apunta a contribuir a un mejor estudio del consumo energético diario, para pequeños y medianos consumidores, mediante el monitoreo de dicho consumo, almacenamiento de esa información persistente en el tiempo y su posterior control estratégico y estadístico. Se pretende también con esto aportar a mejorar las prestaciones de las matrices energéticas a nivel local y regional. En el siguiente apartado se describen algunos de los problemas que el consumo poco controlado de energía eléctrica está produciendo en nuestro entorno.

1.3.1 Cambio Climático.

A modo introductorio, para luego relacionarlo con el consumo de la energía eléctrica, es importante contar cómo comenzó el debate por la preocupación del cambio climático. Desde hace más de veinticinco años es considerado seriamente por organizaciones internacionales y países de todo el mundo este problema que afecta a todo el planeta. Por esta razón es que los principales países responsables de causar dicha problemática se reunieron en Brasil hace ya varios años, con el fin de generar un acuerdo para mitigar el problema, dicho acuerdo se tituló como Protocolo de Kioto.

El Protocolo de Kioto **[1]** sobre el cambio climático fue y es actualmente un acuerdo de orden internacional que tiene como meta la reducción de emisiones de gases de efecto invernadero. El proyecto se haya enmarcado dentro de la Convención Marco de las Naciones Unidas sobre Cambio Climático (CMNUCC). La CMNUCC tiene origen en la Cumbre de la Tierra de Río de Janeiro en 1992, y representa en sí mismo uno de los instrumentos jurídicos internacionales clave destinado a la lucha contra el calentamiento global. En el acuerdo se encuentra firmado el compromiso de los países industrializados de reducir sus emisiones de algunos gases de efecto invernadero, responsables del calentamiento global. El Protocolo de Kioto estudia y aplica a las emisiones de los seis gases de efecto invernadero (EI) más importantes: el dióxido de carbono (CO_2), el metano (CH_4), el óxido nitroso (N_2O), los hidrofluorocarbonos (HFC), los perfluorocarbonos (PFC) y el hexafluoruro de azufre (SF_6), entre los principales causantes del EI.

Hoy en día, alrededor de la mitad del dióxido de carbono liberado por la quema de combustibles fósiles permanece alojado en la atmósfera mientras que el resto es absorbido por la vegetación y los océanos. Recientes proyecciones del modelo climático resume en el reporte que durante el transcurso del siglo XXI, que la temperatura de la superficie global podrá aumentar de 0.3 a 1.7 °C para el escenario de emisiones más bajas y de 2.6 a 4.8 °C en un escenario de emisiones máximas. Estos hallazgos fueron reconocidos por las academias científicas nacionales de los países más industrializados y no están disputados por ninguna entidad científica de origen nacional o internacional de ningún país del mundo.

El cambio climático futuro y sus impactos se diferencian de región a región alrededor del mundo. A los efectos anticipados que incluyen aumento de la temperatura global, aumento del nivel del mar mundial, precipitaciones cambiantes, expansión de los desiertos en zonas subtropicales, se le suma el hecho comprobado que en los continentes el calor es mayor (y mucho mayor en los polos) lo que acelera el retroceso de los glaciares, derretimiento de permafrost y del hielo oceánico. Otros cambios que se producen por las temperaturas extremas propiciadas por el efecto invernadero pueden verse como olas de calor intensas, desarrollo de precipitaciones violentas que probablemente provoquen inundaciones, grandes tormentas de nieve, acidificación de los océanos, extinción de especies debido a los rápidos cambios de

temperatura, entre otros fenómenos. A esto se le suma el abandono de áreas habitadas cercanas a las costas por el incremento de los niveles del mar, inundaciones de ríos y afluentes.
Debido a que el sistema climático sufre de una gran "inercia" y ya que los gases de efecto invernadero permanecerán en la atmósfera por largo período de tiempo, muchos de estos efectos no van a existir solamente por décadas o siglos, si no que persistirán por miles de años **[2]**.

Entre las posibles respuestas sociales al calentamiento global se incluye la mitigación del mismo por reducción de emisiones de gases de efecto invernadero, adaptación a sus efectos, contrucción de sistemas resilientes a sus efectos, y una posible futura ingeniería climática. Muchos países son miembros de la Convención de Cambio Climático (United Nations Framework Convention on Climate Change -UNFCCC-), cuyo objetivo final es prevenir cambios climáticos peligrosos a nivel antropogénico, es decir, cambios climáticos peligrosos al ser humano. Los países miembros de la UNFCCC estuvieron de acuerdo en que es necesario hacer grandes recortes en las emisiones y que el calentamiento global debería limitarse bien por debajo de los 2.0 °C en relación en los niveles pre-industriales.

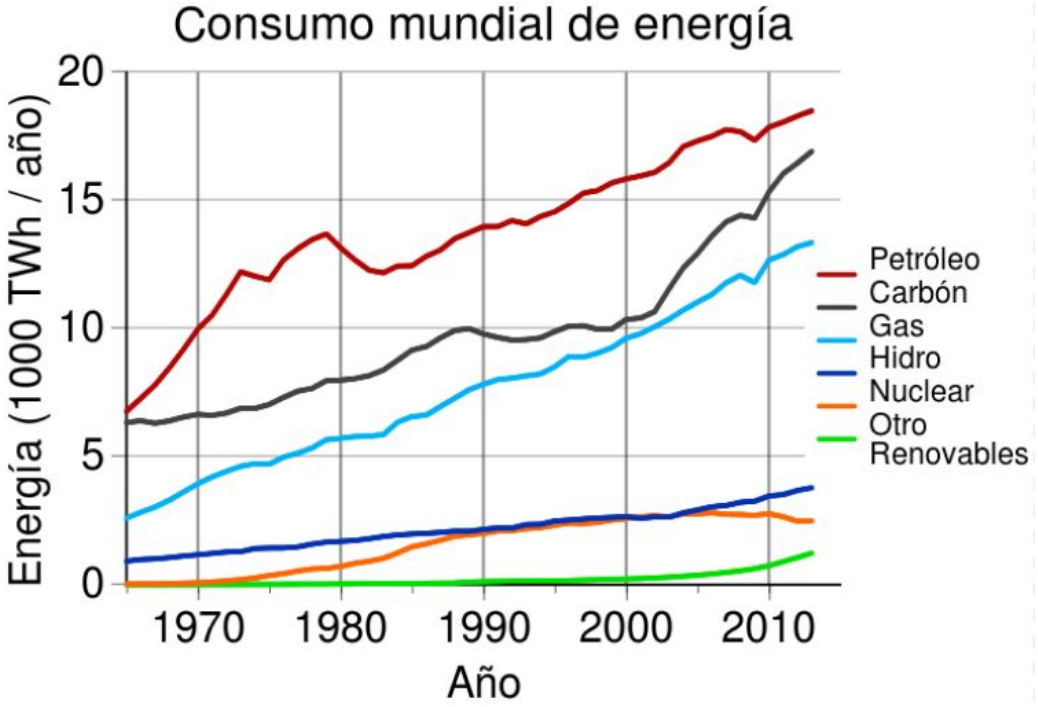


Figura 1.1: Consumo mundial de energía por fuente de generación.

Fuente: File:World energy consumption.svg

La alta demanda de energía por parte de los países desarrollados, son la principal causa del calentamiento global, debido a que sus emisiones contaminantes son las mayores del planeta. Esta demanda de energía hace que cada vez más se extraigan y consuman los recursos energéticos como el petróleo, el carbón y el gas natural dejando a las energías renovables y otras fuentes de generación de energía eléctrica, no emisoras dióxido de carbono, como segunda o tercera opción, fenómeno que puede visualizarse en la figura 1.1 **[3]**.

La dependencia en el uso de combustibles fósiles generada desde la Revolución Industrial para casi todas las actividades humanas, y sobre todo para las actividades económicas, hacen muy difícil el traspaso de estas fuentes de generación de energía eléctrica a fuentes renovables o no contaminantes. A esta desventaja económica debe sumarse la cantidad de capitales que deben invertirse por año (los cuales aumentan año tras año de manera exponencial) debido a arreglos u obras para soportar los desastres naturales causados en su mayoría por el cambio climático, mucha parte de las inversiones económicas relacionadas al cambio climático suelen ser para ayudar a sectores desfavorecidos de la población de los países donde ocurren estos desastres naturales. La lucha constante entre la desviación económica por negarse a invertir en energías no contaminantes y la posterior pérdida de capitales para subsanar los daños causados es una de las principales razones por las cuales no se hacen grandes cambios para mejorar el contexto global actual.

Una vez comprendida y visualizada como real y potencial la amenaza al hablar de cambio climático, nos es oportuno comenzar a pensar en la mitigación del mismo. Cuando nos referimos a la mitigación del cambio climático al mismo tiempo estamos apuntando al ahorro energético con el fin de reducir los efectos potenciales del calentamiento global **[4]**.
El consenso científico sobre el calentamiento global, junto con el principio de prevención y el temor de un abrupto cambio climático, conducen a nuevos esfuerzos para desarrollar tecnologías y ciencias con el fin de reducir los efectos del calentamiento global.
Una de las estrategias para abordar la problemática del calentamiento global existente o a corto plazo es fomentar una fuerte campaña tecnológica de ahorro energético, campaña que directamente obliga un cambio en las estrategias económicas con respecto a la producción de bienes y servicios que identifican las actividades principales de las sociedades de los países más desarrollados. La relación directa entre encarar mejores estrategias económicas y a la par ayudar al retroceso del cambio climático puede entenderse con claridad en el informe del economista británico Sir Stern Rebenson, en el cual luego de 700 páginas de análisis económico ambiental concluye afirmando que se necesita una inversión equivalente al 1% del PIB mundial para mitigar los efectos del cambio climático y que de no hacerse dicha inversión el mundo se expondría a una recesión que podría alcanzar el 20% del PIB global **[5]**.
El informe Stern Rebenson muestra diferentes maneras de contener el cambio climático, estas incluyen: reducir la demanda de bienes y servicios que producen altas emisiones, incrementar la eficiencia, incrementar el uso y desarrollo de tecnologías de bajo nivel de dióxido de carbono y reducir las emisiones de combustible.

1.3.2 Eficiencia en Consumo Eléctrico

La acción de ahorrar energía eléctrica haciendo un buen uso de la misma en todos los ámbitos posibles es claro que impacta en varios aspectos importantes relacionados con la actividad humana. La reducción del cambio climático y prevención de futuras crisis energéticas son algunos de los aspectos más importantes, sobre todo al tener en cuenta que es la principal necesidad para el desarrollo humano en los países industrializados o en proceso de industrialización. Cuando se habla del estudio del consumo de energía eléctrica es fundamental dividir el mismo por sector, como muestra la figura 1.2, ya que ayuda a visibilizar su utilización para futuras tomas de decisiones con respecto a su producción, distribución y ahorro **[6]**.

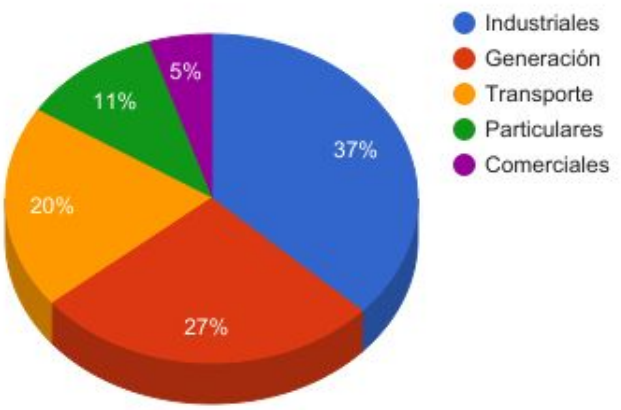


Figura 1.2: Uso de la energía promedio por sector en Países Desarrollados y en Vías de Desarrollo.
Fuente: Elaboración propia con datos de https://www.eia.gov/outlooks/ieo/pdf/0484(2016).pdf

Una de las soluciones a estos problemas de deficiente administración por parte de los usuarios finales, hogares, comercios, pequeñas, medianas y grandes empresas e industrias sería la posibilidad de tener un control absoluto del consumo eléctrico de sus actividades, obtener toda la información al respecto, horarios, consumos topes, consumos fijos, temporadas (entre otros datos) y lograr la sistematización y estadistización de dicha información para la toma de decisiones estratégicas por parte de los usuarios. Al lograr un control numérico real en las micro actividades de los usuarios tanto de hogares como en otros contextos donde se utilicen dispositivos electrónicos con fines económicos, es posible generar informes del verdadero costo de las distintas actividades y productos, y de esta forma llevar a cabo una investigación de cuánto es necesario de consumo eléctrico mínimo para una región o para brindar cierto servicio o producto con el fin de no desanimar a la economía y poder brindar los correspondientes subsidios en razón a las actividades llevadas a cabo y su consumo.

Llama la atención que a nivel de organismos de gobierno en nuestro país existen algunos que recomiendan utilizar ciertos aparatos eléctricos tantas horas por día máximo debido a sus consumo en kWh. Los productos de dichas recomendaciones no se categorizan ni por marca ni por etiqueta de eficiencia energética, resulta difícil sistematizar los consumos si no se categorizan ya que cada producto consumirá según su marca, modelo, antigüedad, cantidad de uso, condiciones externas climáticas y condiciones eléctricas del establecimiento, además de la cantidad de tiempo utilizado. El organismo gubernamental, que plantea éstas estrategias de otorgamiento de subsidios según los consumos de los habitantes de nuestro país, sólo hace referencia a la cantidad de horas que pueden utilizarse los aparatos eléctricos, sin mencionar las demás variables, para llegar a un consumo mínimo y de esta forma beneficiar al usuario con dichos subsidios o penalizarlo con altas tarifas.

Algunas posibles soluciones a estos problemas de información radican en directamente obtener la información del consumo eléctrico de cada aparato individual en tiempo real y hacer uso de dichos datos para generar estadísticas que ayuden al usuario final a visualizar cuáles son sus consumos de sus actividades e incluso darle la posibilidad de automatizar acciones para disminuir estos consumos, como apagados automáticos según hora, fecha o cantidad de consumo acumulado en determinado tiempo para cada dispositivo. Esta información de consumo eléctrico por dispositivo individual no es común ni fácil de obtener por lo que se

termina generalizando y englobando en grandes grupos de consumos energéticos (como se puede ver en la figura 1.3), problema que no permite un control directo y un entendimiento del consumo diario por parte de los usuarios.

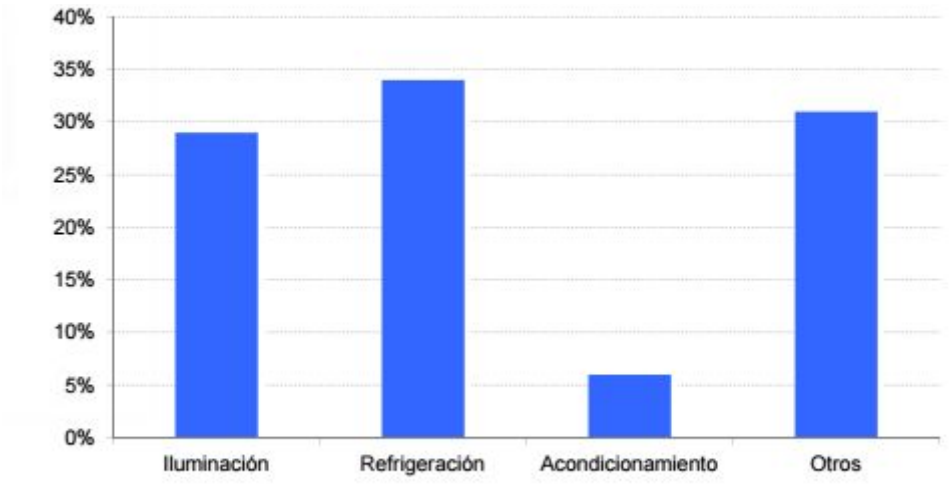


Figura 1.3: Uso de la electricidad (en porcentajes) en el sector residencial.
Fuente: Informe nacional de monitoreo de la eficiencia energética de la República Argentina, 2014 (CEPAL) https://drive.google.com/file/d/0B6NafLbaJ62zaktueDYzUGlublE/view

1.4 Solución y Desarrollo

1.4.1 Descripción general

La solución que planteamos para la problemática presentada es desarrollar un dispositivo que mida el consumo eléctrico mediante la conexión directa a los tomacorrientes regulares, permitiendo el sensado del consumo del dispositivo que se conecta al tomacorriente y realizando el envío de dichos datos a un servidor mediante WiFi. Luego toda esa información podrá ser consultada por el usuario tanto en la interfaz móvil como en la web.

Dicho dispositivo, junto al Backend y el Frontend alojados en el servidor antes mencionado, conformarán en su conjunto el sistema de control y monitoreo de consumo eléctrico que llamaremos Elektron. Los usuarios podrán utilizar el sistema mediante su interacción con una aplicación web o una aplicación móvil las cuales consumen los datos del sistema de un Backend, que estará permanentemente recibiendo datos de los distintos dispositivos y estará enviando órdenes a los mismos para su apagado o encendido. A continuación se muestra en la Figura 1.4 el modelo general del sistema para una primera aproximación al mismo:

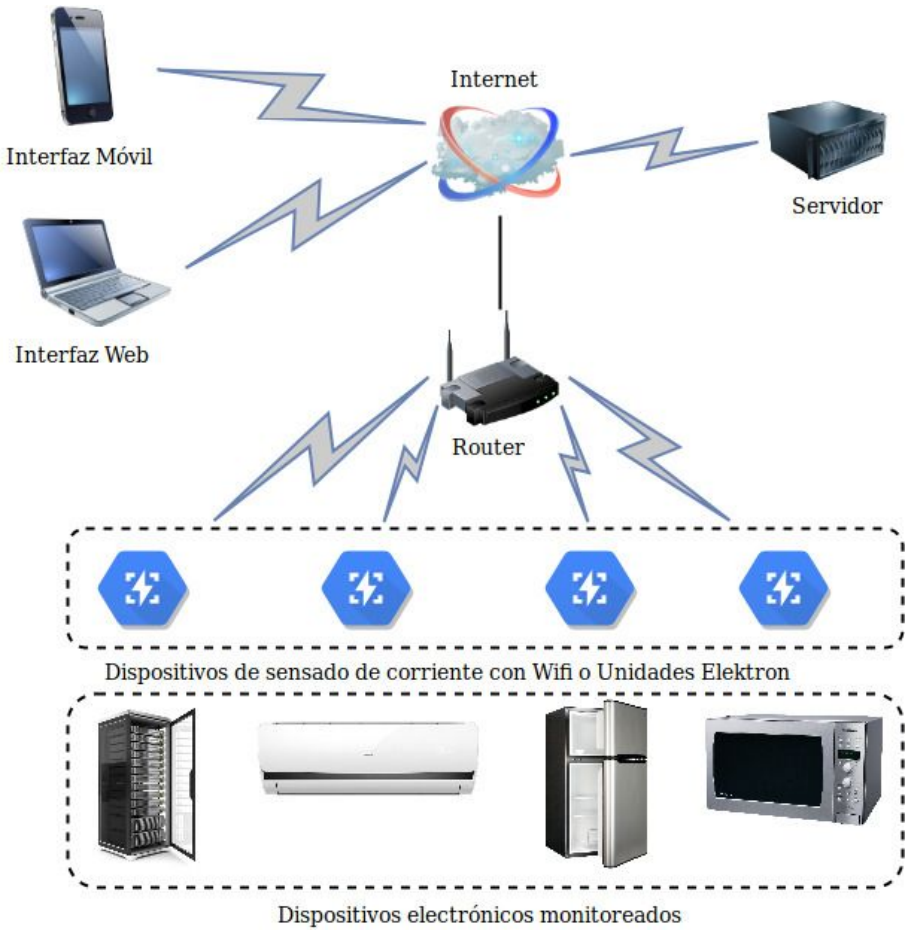


Figura 1.4: Gráfico del funcionamiento general del sistema. Fuente: Elaboración propia.

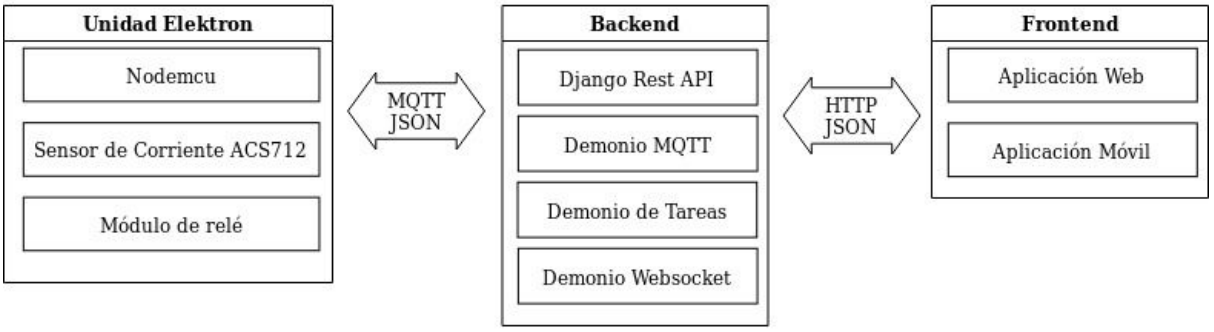


Figura 1.5: Modelo General del Sistema Elektron. Fuente: Elaboración propia.

La Figura 1.5 divide al sistema en tres partes principales que se irán detallando a lo largo del documento. A continuación se dará una primera descripción resumida para entender el modelo general del sistema.

Unidad Elektron

La Unidad Elektron es un conjunto de dispositivos de hardware que ensamblados cumplen la tarea de medir la corriente eléctrica de un toma corriente, procesar dichos datos y enviarlos como datos de consumo eléctrico por WiFi al servidor. También cumple la tarea de apagar o encender los dispositivos conectados según las órdenes recibidas desde el servidor. Los mensajes con los datos de las Unidades Elektron son enviados al servidor a través del protocolo MQTT (Message Queuing Telemetry Transport) y con formato JSON (JavaScript Object Notation), acción que se detallada en su respectiva sección más adelante.

Backend

El Backend está alojado en el servidor y es un conjunto de aplicaciones desarrolladas en lenguaje de programación Python las cuales cumplen la tarea de ser el core o núcleo del sistema: reciben y envían los datos, almacenan la información y realizan la lógica del sistema. También se describe en profundidad más adelante en su respectiva sección.

Frontend

El Frontend es un conjunto de dos aplicaciones, una web y una móvil, que también están alojadas en el servidor y tienen la función de presentar una interfaz usable y amigable a los usuarios para que interactúen de manera amigable y ágil con el sistema. Las peticiones de datos y órdenes que se efectúan hacia el Backend desde el Frontend utilizan el protocolo HTTP y sus respuestas respetan el formato JSON.

1.4.2 Medición de Consumo Eléctrico

Las tecnologías para la lectura automática de consumo han estado evolucionando mucho en los últimos años alrededor del mundo. Estas tecnologías son conocidas como AMR (Automatic Meter Reading, lectura de medición automática en español) y su función principal es la de obtener el consumo de forma automática, generar diagnósticos y con estos validar el estado de los servicios de medición de gas, agua o electricidad. Esta información generada se envía a una base de datos central para el cálculo de los impuestos y la detección y aviso de problemas. Esta tecnología fue y sigue siendo de mucha ayuda para los proveedores de servicios básicos, permitiéndoles ahorrar traslados inoportunos a cada domicilio para medir el consumo. Otra ventaja radica en que, con esta tecnología, es posible tener una estimación más certera del consumo, teniendo en tiempo real los datos y así calcular su costo de forma más precisa, en vez de basarse en datos pasados del consumo o predecir los mismos en base a un comportamiento estimado del consumo.

Entre las más sofisticadas podemos encontrar la lectura de consumo eléctrico a través de WiFi con varios ejemplos en el mercado actual, la mayoría privativos y cerrados.
Un ejemplo de la industria es WiSmart, que según sus proveedores y fabricantes, resulta ser una plataforma versátil que puede ser utilizada en muchas variedades de aplicaciones eléctricas hogareñas con el objetivo de proveer comunicación wireless TCP/IP entre sus dispositivos de medición.

Si bien es posible encontrar numerosos dispositivos y diversas tecnologías para la medición del consumo de energía eléctrica, la mayoría sólo puede medir el consumo total acumulado en tiempo real o el total de cierta franja temporal (el consumo total acumulado de un mes de todas las bocas de una vivienda o puesto comercial, por ejemplo). Son muy pocos los que pueden medir el consumo exacto de cada tomacorriente o cada dispositivo electrónico o eléctrico en funcionamiento dentro de un establecimiento.

Hasta la fecha se encuentran en el mercado variados dispositivos de control y programación de prendido y apagado de forma independiente de dispositivos eléctricos, que proveen al usuario una serie de aplicaciones móviles para controlar y programar la fecha, hora y duración del encendido y apagado de estos tomacorrientes inteligentes. Son conocidos en el mercado como Smart Plugs pero ninguna de las marcas más conocidas cuentan con la servicio de medición de consumo eléctrico.

Nuestra idea de proyecto está más orientada al uso sencillo y eficiente de los tomacorrientes, monitoreando el consumo, pudiendo ser hogares particulares, locales comerciales, oficinas o pequeñas y medianas industrias. A diferencia de los Smart Plugs que existen hoy en el mercado que están principalmente apuntados a lo estético y el confort del usuario, nuestro sistema intenta generar además un uso consciente y estratégico del recurso energético brindando la información de consumo de cada tomacorriente en tiempo real y facilitando la consulta de dichos datos en cualquier franja temporal.

Mediante la combinación de la medición eléctrica y la posibilidad de prender y apagar remotamente el tomacorriente es posible generar otro tipo de controles y automatizaciones, no sólo programando un dispositivo para que encienda y apague relacionándolo con el tiempo si no también en relación con el consumo acumulado en cierta franja horaria, permitiéndole al usuario controlar sus gastos energéticos, separando de forma individual cada dispositivo electrónico en uso y de esta manera reducir el impacto económico y ambiental.

Además un detalle importante de nuestra investigación es la idea de generar distintos tipos y niveles de información estadística, mediante la obtención y el almacenado de los datos en tiempo real de cada dispositivo conectado. Esto último ayudaría a los usuarios y administradores a llevar un control de sus consumos según la temporada, la hora del día y otros períodos temporales a elección, para futuras tomas de decisiones de consumo o incluso para el estudio de posibles reemplazos de equipos que demuestren no estar funcionando como deberían.

1.4.3 Sobre el cálculo del costo y contaminación en CO_2

Es importante que, al ser un sistema orientado a usuarios finales, el mismo pueda tener información útil y de fácil comprensión. Es por esto que desarrollamos módulo en las interfaces de usuario que enseñan el cálculo del costo acumulado por el consumo en kWh registrado y que contaminación en CO_2 representa dicho consumo.

En el caso del costo acumulado, para este trabajo se decidió simplificar el cálculo en base a un valor fijo de precio por kWh consumido, que es de ~$1,77 por kWh. Este valor se obtuvo de

cuadros tarifarios que ofrece Edelap (Empresa distribuidora de Energía de La Plata) en su sitio web[1], pensando en un usuario estándar que consume habitualmente entre 150 kWh y 325 kWh por mes.

Por otro lado, para el caso de la contaminación en CO_2, se decidió tomar un cálculo que se encuentra en el portal web del ministerio de energía y minería de la Nación[2]. En dicha fuente se pueden hallar los cálculos del factor de emisión de CO_2 en base al consumo en kWh. El factor utilizado en este trabajo fue de ~0,5 tCO_2/MWh.
Es importante destacar que dicha contaminación depende de la matriz energética del país o región en la que se esté monitoreando el consumo, ya que es en el momento de la producción de la energía eléctrica (y en base a qué tipo de recursos) que se produce dicha liberación de CO_2. Por ejemplo para el caso de la Unión Europea[3], el factor de emisión es de 0,35 mientras que para la Argentina es de aproximadamente 0,5.

Para un trabajo futuro se pensó la posibilidad que el usuario pueda configurar los costos y la emisión de CO_2 según los proveedores de energía, datos gubernamentales y/o que dichos parámetros sean obtenidos automáticamente por región o país.

[1] http://www.edelap.com.ar/wp-content/uploads/2018/01/tarifarioA4_1_2_18.pdf
[2] http://datos.minem.gob.ar/dataset/calculo-del-factor-de-emision-de-co2-de-la-red-argentina-de-energia-electrica
[3] https://www.camarazaragoza.com/wp-content/uploads/2012/10/calculoemisiones.xls

1.5 Estructura de la Tesis

La tesis consta de cinco capítulos. A continuación se realiza una breve descripción de cada uno de estos.

En el capítulo 1 (en el que nos encontramos), se realiza una introducción al proyecto de investigación, cuáles son los objetivos y la problemática atacada.

En el capítulo 2 se desarrolla el Estado del Arte donde se relata la investigación llevada a cabo sobre tecnologías relacionadas al monitoreo y control de artefactos electrónicos y se describe el estudio realizado sobre la corriente alterna y sensores de corriente.

El capítulo 3 está destinado al desarrollo realizado como parte de la investigación llevada a cabo. En este capítulo se describen las herramientas utilizadas para el desarrollo integral del sistema en su totalidad.

En el capítulo 4 se describen las diferentes pruebas realizadas en puntos claves del sistema con el fin de visualizar los posibles problemas y analizarlos.

En el capítulo 5 se detallan las conclusiones obtenidas con respecto al ahorro de energía y sobre los problemas analizados a raíz de las pruebas realizadas. Al final del capítulo se presenta el trabajo a futuro.

2. Estado del Arte

El objetivo de este capítulo es mostrar una parte del proceso de investigación que implicó observar algunos casos exitosos de investigaciones y proyectos similares que se encuentran en funcionamiento y también la revisión de muchos conceptos relacionados al estudio de la electrónica y electricidad que escapan a nuestra disciplina, pero que eran fundamentales para alcanzar los objetivos de este trabajo.

La búsqueda de herramientas relacionadas a nuestro proyecto, es decir, que pertenezcan al mismo campo de investigación y desarrollo aplicado al campo del consumo de energía, fue fundamental para entender el contexto en el que se encuentra esta área y que aportes nuevos o diferentes podríamos agregar.

Retomando con el uso concreto de tecnologías en el campo energético, uno de los ejemplos más impactantes a nivel comercial y por su relevancia regional y gubernamental es el de Smart Grid. El departamento de energía de Estados Unidos presenta su proyecto Smart Grid donde señala que una de sus metas principales es lograr un sistema de viviendas inteligente llamado Smart Home en el cual se ofrecen una serie de soluciones en forma de tomacorrientes inteligentes y sistemas de control de consumo eléctrico hogareño. Toda la información se encuentra directamente conectada y centralizada a la red de energía inteligente o Smart Grid. De esta forma el departamento de energía puede proveer de tecnología y servicios de control de consumo eléctrico hogareño a la población desde las empresas privadas vinculadas con el estado [4].

Aunque en el proyecto estadounidense no se detalla con claridad cómo se llevan a cabo las diferentes aplicaciones y soluciones tecnológicas sirve como ejemplo de las posibilidades que esta combinación de conceptos y tecnologías pueden otorgar.
Un ejemplo más terrenal para entender cómo implementar algunas soluciones aplicadas a los recursos energéticos utilizando sistemas inteligentes, y para dar una vista más contundente de un caso exitoso es el proyecto abierto Open Energy Monitor[5]. Dicho proyecto trata de una serie de módulos de código libre para desarrollar herramientas de monitoreo de energía que ayudan a los usuarios a relacionar el uso de la electricidad, los sistemas eléctricos y el reto de la implementación de energía sostenible. El proyecto se basa principalmente en lograr un espacio de aprendizaje técnico y reflexivo del uso de la energía, a través de un portal web muy completo con guías casi cien por ciento libres desde el hardware, el software y el armado de estructuras de bajo costo. Su objetivo principal es brindar información para el desarrollo de herramientas de monitoreo de consumo, producción y ahorro energético, concientizando el buen uso la misma, y, como segundo objetivo, discutir los modelos sustentables de las matrices productivas de los diversos modelos económicos.

[4] En este proyecto, muchos de los electrodomésticos de la casa inteligente estarán conectados en red, permitiendo operarlos a través de su EMS (Energy Management System). https://www.smartgrid.gov/the_smart_grid/smart_home.htm

[5] https://openenergymonitor.org/

El proyecto OpenEnergyMonitor en una primera instancia se divide en cuatro ejes principales: el uso de energía hogareña, la producción de energía solar, la carga de baterías para automóviles eléctricos y el monitoreo del uso de energía eléctrica en acondicionamiento del hogar. En su primer eje, el uso de energía, el proyecto se enfoca en brindar soluciones basadas en software libre y hardware (en su mayoría libre) para el sencillo monitoreo de la electricidad, temperatura y humedad, así como también permite ver los datos históricos y en tiempo real. Su segundo eje relacionado a la producción de energía solar permite a los usuarios explorar la producción de energía por parte de paneles fotovoltaicos, comparar consumos, calcular el uso interno y optimizar las demandas eléctricas. El tercer eje del proyecto está abocado a la carga de baterías para automóviles eléctricos y permite la conectividad WiFi del sistema de carga, ajustar la tarifa de carga, obtener registros de todos los datos y brinda una API abierta para el libre uso y configuración por parte de los usuarios. Por último en su eje de monitoreo del uso de energía eléctrica en acondicionamiento del hogar, el proyecto brinda las herramientas para el monitoreo de electricidad y temperatura de los sistemas de acondicionamiento hogareños, permite hacer un seguimiento de la performance de estos sistemas al mismo tiempo que generar diagnósticos para su consulta remota.

De los cuatro ejes mencionados anteriormente desarrollados por el proyecto Open Energy Monitor, el que se encuentra más cercano a nuestra investigación es el primero, uso de energía hogareña. Este eje se centra en brindar al usuario un sistema de monitoreo simple para entender el consumo de energía. Permite visualizar y explorar la energía eléctrica consumida en tiempo real diariamente en kWh. El proyecto deja explícito todo el hardware y el software necesario junto a las instrucciones de ensamblado y configuración para que el usuario pueda implementar y poner a punto su sistema de control de consumo eléctrico hogareño. De todas formas, hay dos diferencias claves que encontramos en este proyecto y nuestra idea:

1. La demanda de hardware de nuestro sistema es mucho menor a la del sistema propuesto por OpenEnergyMonitor. OpenEnergyMonitor demanda a los usuarios adquirir una serie de unidades de hardware muy específico sólo gestionado por su organización. Aunque en las especificaciones aclaran que la mayoría de sus unidades de hardware están basadas en tecnologías como Arduino o Raspberry Pi, las variaciones internas al mismo no están aclaradas y hace al sistema poco escalable por parte de desarrolladores o usuarios especializados. En este sentido nuestro proyecto de investigación es más abierto y económico. Basándose en módulos desacoplados, permite a los futuros desarrolladores y usuarios armar sus propias configuraciones y combinaciones de componentes para el uso que prefieran.

2. El proyecto de monitoreo de consumo de energía eléctrica hogareña planteado por OpenEnergyMonitor está pensado para hacer un monitoreo general del hogar, ubicando sus sensores en la caja de corriente central del hogar. Nuestra investigación se enfoca en generar un módulo desacoplado que pueda utilizarse para medir el consumo de cualquier aparato eléctrico gracias a que es acoplable como un adaptador común entre cualquier tomacorriente y cualquier dispositivo eléctrico. Esto permite a los usuarios tener un registro del consumo en tiempo real y estadístico de los aparatos eléctricos que deseen cuando deseen.

2.1 Automatización y Control Doméstico Inteligente: Domótica

La domótica es la automatización de componentes eléctricos y electrónicos de una casa (la cual llamaremos casa inteligente o Smart Home en inglés) generando el control y automatización de iluminación, calefacción (como termostatos inteligentes), ventilación, aire acondicionado y seguridad. También se encuentran electrodomésticos como lavadoras, secadoras, hornos, refrigeradores y/o congeladores, entre otros. Normalmente la conectividad WiFi o Bluetooth es utilizada para la interconexión entre los diversos subsistemas, centralizando el control para supervisión y acción concreta.

Los aparatos domésticos (y no domésticos también) que son controlados remotamente a través de alguna red como Internet o utilizando Bluetooth entran dentro de otro campo vinculado a la domótica llamado Internet de las Cosas (IoT - Internet of things). Además de ser controlados a la distancia (inclusive en otras partes del mundo), realizan una conexión especial con las funciones específicas de cada aparato, incluyendo la utilización de sensores que obtienen datos del mundo físico y los digitalizan. Todo este monitoreo se realiza generalmente con interfaces de usuario presentes en el lugar (montados en la pared), o a través sistemas almacenados en la nube, accesibles desde los teléfonos móvil, tablets o formatos web.

Mientras la industria productiva avanza rápidamente, se han podido asentar muy pocos estándares aceptados mundialmente sobre la domótica e Internet de las cosas. Entre los protocolos de comunicaciones más populares para los productos Smart Home se incluyen X10, Ethernet, RS-485, 6LoWPAN, Bluetooth LE (BLE), ZigBee y Z-Wave, y otros protocolos propietarios que son incompatibles entre sí. Los fabricantes a menudo impiden implementaciones independientes mediante la retención de documentación y por litigio.

Un dato económico interesante para observar es que el mercado de la automatización doméstica tuvo un valor de US $ 5,77 millones en 2013, y se prevé que alcanzará un valor de mercado de US $ 12,81 millones para el año 2020 **[7]**, lo que indica que a pesar de ser un campo nuevo de investigación y producción, avanza rápidamente.

2.2 Sensado de Corriente

Este apartado introduce brevemente ciertos temas fundamentales para comprender el cálculo del consumo eléctrico: el conocimiento de los conceptos de energía eléctrica, las distintas cargas eléctricas, los tipos de aparatos domésticos y la manera en la que se puede obtener la información de su consumo.

Con el fin de estimar el consumo de cada uno de los dispositivos eléctricos que pueden encontrarse en un domicilio particular es preciso primero conocer cómo se definen los diferentes tipos de cargas eléctricas en una red de corriente alterna o AC. No todos los dispositivos eléctricos interaccionan con el sistema eléctrico de la misma forma, sus consumos dependen de

la impedancia que presentan a la red eléctrica. Es por esto que primero es necesario comprender cómo se debe medir cada tipo de carga.

Cargas Resistivas:

Las lámparas incandescentes, planchas, calentadores eléctricos, cocinas eléctricas, entre otros dispositivos presentan a la red eléctrica cargas puramente resistivas, lo que quiere decir que su consumo de corriente es prácticamente igual a la tensión dividida por su resistencia (ley de Ohm). Una carga puramente resistiva genera una onda de corriente en fase con la tensión y es similar a la siguiente:

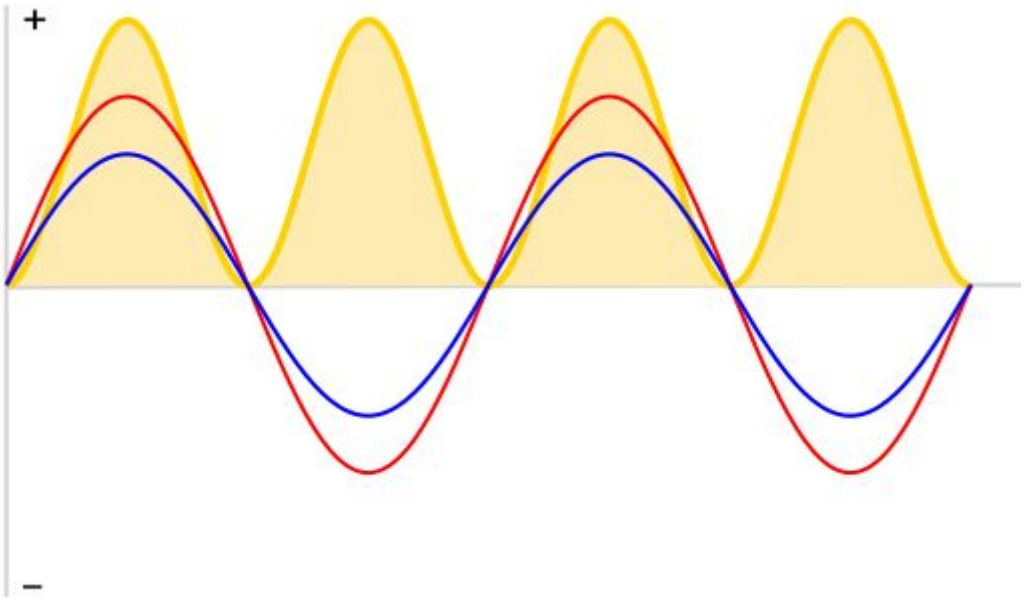


Figura 2.1: Relaciones de tensión y corriente en una carga puramente resistiva. En amarillo la potencia p(t), en rojo el voltaje v(t) y en azul la corriente i(t). Fuente: https://learn.openenergymonitor.org/electricity-monitoring/ac-power-theory/introduction.

En la figura 2.1 la línea amarilla es la potencia que es igual al producto de la tensión y la corriente en todo momento. Notar que la potencia es siempre positiva. En este caso, se utiliza la convención de que la dirección positiva es la energía que fluye hacia la carga (alimenta el dispositivo conectado).

Cargas Reactivas:

En los electrodomésticos tales como lavaplatos, heladeras, lavarropas, aires acondicionados y taladros entre otros, las ondas de tensión y corriente no están en fase. Esto ocurre porque los mismos cuentan con componentes inductivos (por ejemplo, motores). Una carga parcialmente inductiva genera una onda de corriente desfasada con respecto a la onda de tensión, de forma similar a la siguiente (Figura 2.2):

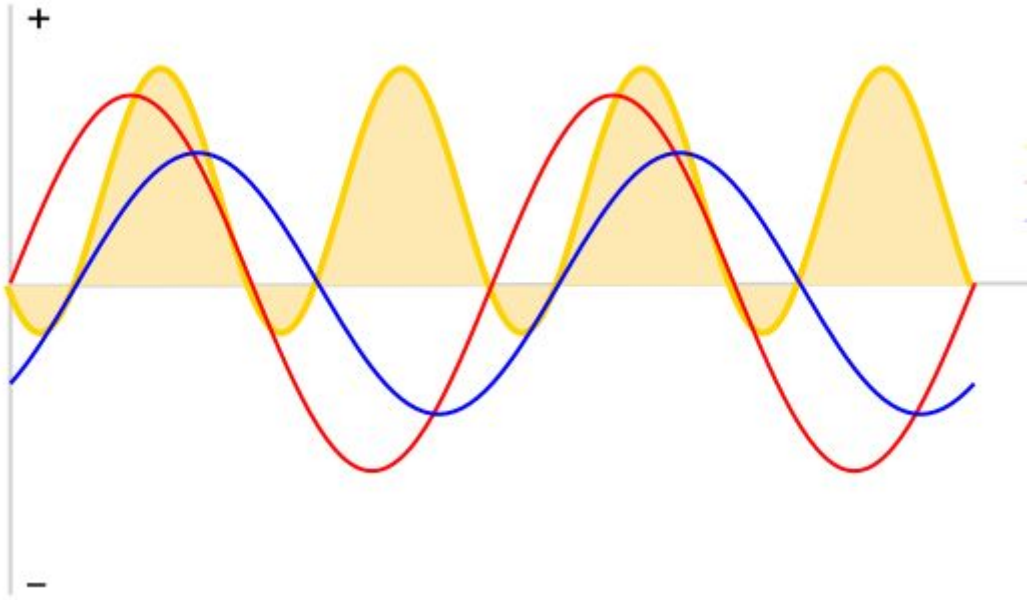


Figura 2.2: Relaciones de tensión y corriente en una carga parcialmente inductiva. En amarillo la potencia p(t), el rojo el voltaje v(t) y en azul la corriente i(t). Fuente: https://learn.openenergymonitor.org/electricity-monitoring/ac-power-theory/introduction.

Notar que en la figura 2.2 la línea amarilla que corresponde a la potencia se vuelve negativa por un período de tiempo, la parte positiva fluye hacia la carga que alimenta al dispositivo y la parte negativa fluye de vuelta hacia el sistema eléctrico principal **[8]**.

Otra cuestión importante a considerar es que las formas de onda de voltaje y corriente se han separado, se desfasan. Este desfase o desplazamiento, existente entre las onda de voltaje y corriente, se indica en términos fasoriales con el ángulo entre los fasores tensión y corriente. El coseno de este ángulo se denomina coseno de Phi o factor de potencia. Para obtener el coseno de Phi es preciso medir la tensión y la corriente simultáneamente. En este proyecto se utiliza un sensor de corriente, pero no se mide la tensión. Por lo tanto, no es posible calcular el coseno de Phi ni la amplitud de la tensión, aunque es posible estimarla con cierto error. Se podrá efectuar la incorporación del sensor extra como trabajo a futuro pero basta con tener la capacidad de medir solo la corriente para demostrar que es posible realizar un sistema que permita monitorear dichas variables en tiempo real y almacenarlas para consultas o toma de decisiones futuras que es el propósito de nuestra investigación.

Existen distintos tipos de sensores que permiten obtener una medición del consumo de corriente que se está efectuando en un circuito determinado y se dividen en dos grandes grupos: sensores invasivos y no invasivos. La principal diferencia se encuentra en que los sensores invasivos se interponen en alguna parte del circuito para realizar la medición y en el caso de los no invasivos todo se realiza desde proximidades del circuito pero sin interrumpirlo. En algunos casos se utiliza el efecto Hall para obtener dicho dato. Lo que varía es la exactitud de la información que obtiene, teniendo más exactitud el invasivo.

2.3 Sensor Invasivo ACS712

Para nuestro trabajo utilizamos el sensor ACS712 de Allegro™, que es un sensor de corriente por efecto Hall. Provee un solución económica y además precisa para medir corriente AC o DC (Corriente continua), ya sea en ambientes industriales o comerciales. Este sensor funciona transformando un campo magnético generado por la circulación de una corriente eléctrica por un alambre de cobre interno en el sensor, convirtiendo este campo en un voltaje variable. Esto significa que a mayor cantidad de corriente, mayor voltaje se tendrá en la salida del sensor.

Este sensor es ofrecido por el fabricante en 3 modelos distintos: ACS712ELCTR-05B-T que mide hasta 5A, el ACS712ELCTR-20B-T que mide hasta 20A y el ACS712ELCTR-30B-T que mide hasta 30A. Todos los modelos entregan un rango de voltaje en su pin de salida que varía entre 0 y 5V dándonos una mejor exactitud en el modelo de 5A que en el de 30A. En este punto cabe aclarar que aunque el sensor pueda entregar de 0 a 5 V en nuestro proyecto utilizamos la plaqueta controladora NodeMCU, que alimenta al sensor con 3.3 V, tema que se profundizará más adelante en el apartado 2.4.
El ACS712 (figura 2.3) al ser modular nos facilita su conexión, el mismo trae una bornera para conectar la línea que desea medirse y 3 pines, dos para conectar la alimentación (positivo y negativo) y un pin para la salida analógica de donde obtendremos los datos del sensado.

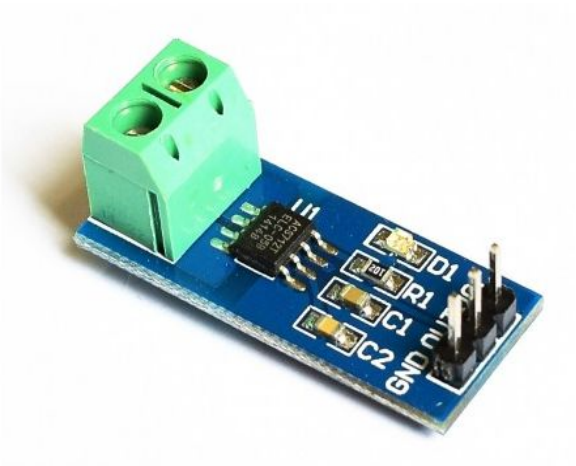


Figura 2.3: Sensor de efecto Hall ACS712 de Allegro Microsystems. Fuente: Allegro Microsystems ACS712.

Características del sensor de efecto Hall ACS712[6]:

- Bajo ruido en señal analógica.
- Ancho de banda del dispositivo de 80 kHz.
- Error total de salida 1.5% a T_A=25°C.
- Tamaño pequeño (low-profile SOIC8).

[6].http://www.allegromicro.com/en/Products/Current-Sensor-ICs/Zero-To-Fifty-Amp-Integrated-Conductor-Sensor-ICs/ACS712.aspx

- Resistencia del conductor interno de 1.2 mΩ.
- Operación de alimentación única de 5.0 V.
- Sensibilidad de salida de 66 a 185 mV/A
- Voltaje de salida proporcional a corrientes AC o DC.
- Calibrado de fábrica para mejor exactitud.
- Salida de voltaje offset extremadamente estable.

Como mencionamos anteriormente, el rango de corriente que se puede medir con este sensor y su sensibilidad varían dependiendo del modelo. En la siguiente figura (Figura 2.4) mostramos los tres modelos:

Modelo	**Rango**	**Sensibilidad**
ACS712ELCTR-05B-T	-5 a 5 A	185 mV/A
ACS712ELCTR-20B-T	-20 a 20 A	100 mV/A
ACS712ELCTR-30B-T	-30 a 30 A	66 mV/A

Figura 2.4: Relación entre Rango de Intensidad en Amperios y Sensibilidad en mV/A según recomienda el fabricante.

El sensor entrega un valor de 2.5 VDC para una corriente de 0A y a partir de allí se incrementa proporcionalmente de acuerdo a la sensibilidad, teniendo una relación lineal entre la salida de voltaje y la corriente.

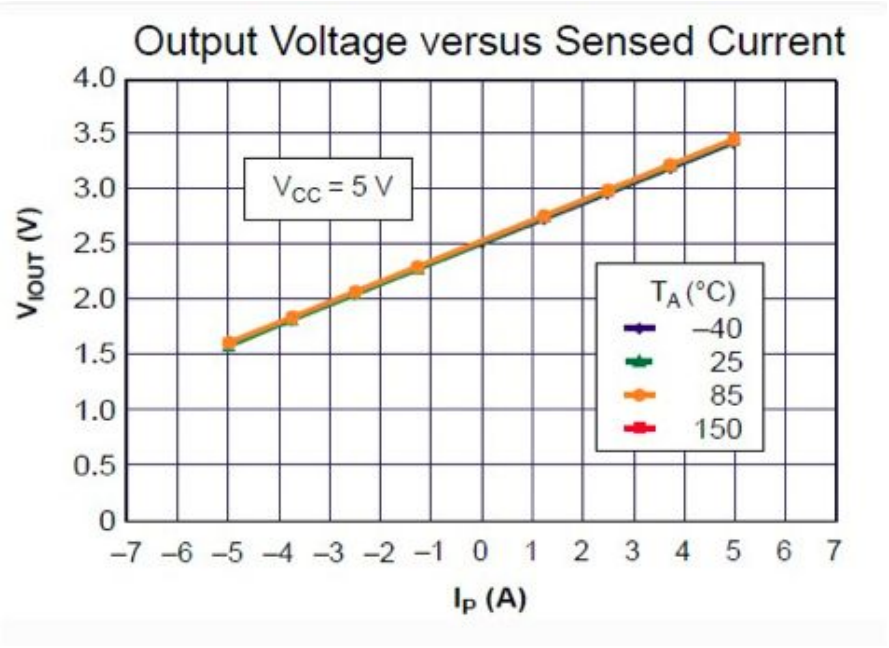


Figura 2.5: Relación lineal entre el Voltaje obtenido de salida del sensor y la Corriente sensada por el mismo. Fuente: Allegro Microsystems ACS712.

Existe entonces una relación lineal entre el Voltaje de salida del sensor y la Corriente, donde la pendiente es la sensibilidad y la intersección en el eje Y es 2.5 voltios como puede verse en la figura anterior (Figura 2.5). La ecuación de la recta será la siguiente

$$V = m\,I + 2.5$$

Donde la pendiente es m y equivale a la Sensibilidad, e **I** es la corriente.
Despejando tendremos la ecuación para hallar la corriente a partir de la lectura del sensor:

$$I = V - 2.5 \,/\, Sensibilidad$$

2.4 Ordenador de placa reducida: NodeMCU

En nuestro proyecto de investigación decidimos utilizar la placa ordenadora reducida (Single Board Computer o SBC) NodeMCU. Se decidió utilizar NodeMCU por su gran comunidad, por ser de diseño abierto (Open Source), porque cuenta con múltiples librerías para la interacción con una gran familia de sensores y módulos (entre los que se encuentra el sensor ACS712) y, por sobre todo, porque está basado en el dispositivo ESP8266, un microcontrolador WiFi de bajo costo que cuenta con compatibilidad total con el protocolo TCP/IP.

A Continuación, mencionamos las características del ESP8266:

- CPU RISC de 32-bit: Tensilica Xtensa LX106 a un reloj de 80 MHz
- RAM de instrucción de 64 KB, RAM de datos de 96 KB
- Capacidad de memoria externa flash QSPI - 512 KB a 4 MB (puede soportar hasta 16 MB)
- IEEE 802.11 b/g/n WiFi
    - Soporte de autenticación WEP y WPA/WPA2
- 16 pines GPIO (Entradas/Salidas de propósito general)
- SPI, I²C,
- 1 conversor ADC de 10-bit (pin A0).

A modo ilustrativo, se muestra en la siguiente figura (2.6) el esquema de pines de una placa NodeMCU.

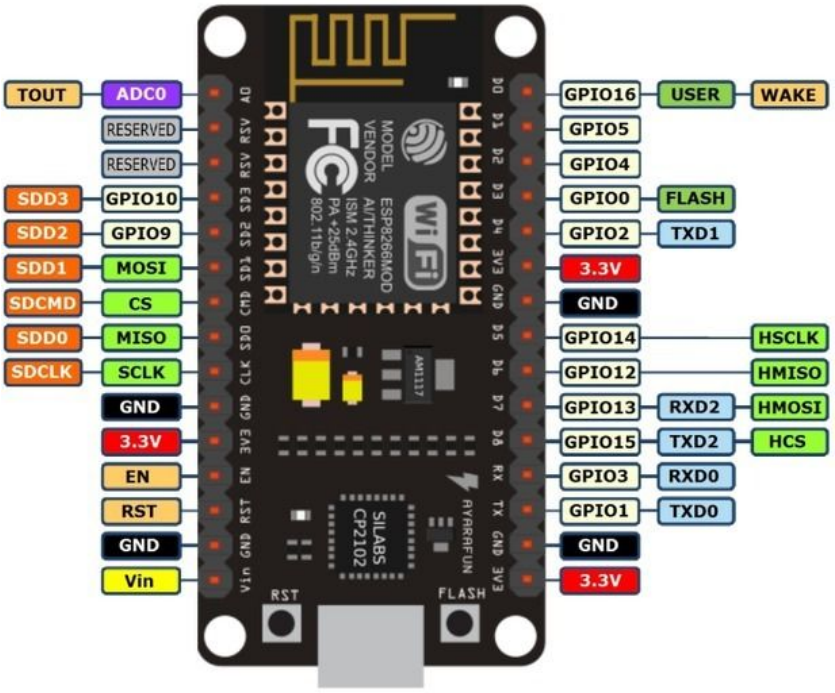


Figura 2.6.Esquema de pines de NodeMCU basado en microcontrolador ESP8266. Fuente: https://iotbytes.wordpress.com/NodeMCU-pinout/.

2.5 Proceso de sensado con la placa NodeMCU y el sensor ACS712

Para poder obtener los datos analógicos provenientes del sensor ACS712 con la placa NodeMCU es preciso conectar el pin de salida de datos del sensor a un pin de entrada de datos analógicos de la placa.

2.5.1 Tomando muestras del sensor ACS712 con NodeMCU

El microcontrolador utilizado por el NodeMCU, el ESP8266, posee 1 entrada analógica. Los valores aceptados de entrada son de entre 0 y 3.3 volts. Éstos valores de entrada se mapean con valores enteros de medición dentro del código del NodeMCU los cuales varían entre 0 y 1023. Esto sostiene una resolución de medición de 3.3 volts / 1024 unidades o, 0,0032 volts (3,2mV) por unidad y esta resolución de entrada puede ser modificada vía software. En el caso que nos ocupa, dado que la plaqueta puede alimentar al sensor con 3.3 V como máximo, se utilizan sólo 1,32 V en total (debido a la relación lineal vista en la figura 2.5). Cabe destacar que estos valores de entrada, al traducirse a digital, generan una pérdida que podría solucionarse con un circuito amplificador externo que escapa del tema específico de nuestra investigación.

La entrada analógica del ESP8266 (representada en la figura anterior -2.6- como ADC-0 -Conversor Analógico Digital; Analog-to-Digital Converter-) tiene un sistema que convierte los valores analógicos en valores digitales, con el propósito de facilitar su procesamiento.

En el caso de la línea de alimentación de la red eléctrica principal, sabemos que la forma de onda de la tensión es senoidal o sinusoidal. Para calcular la potencia es necesario conocer la forma de la onda de la corriente. Suponiendo que la corriente es senoidal pura es posible obtener un resultado aproximado del consumo eléctrico aplicando una fórmula básica. De no ser la onda de la corriente de forma senoidal pura es posible que se generen distorsiones en la misma.

En nuestro trabajo queremos enfatizar en los aspectos relacionados al sistema informático integral, las funcionalidades especiales como la visualización en tiempo real y el resguardo histórico de datos, elementos que explayaremos en profundidad durante todo el capítulo 3. La información que tenemos como insumo es el consumo eléctrico de cargas puramente resistivas, pero como la complejidad de este tema escapa a los temas vistos durante la carrera, no ahondaremos en los detalles eléctricos teóricos.

Los valores que brinda el ACS712 son muy variables al medir corriente alterna. Para asegurarnos obtener los valores picos tanto máximos como mínimos de cada ciclo de la onda senoidal, es necesario obtener muchas muestras en un período de tiempo lo suficientemente largo (mínimamente lo que tarda la onda en realizar un ciclo completo). Sabiendo que la corriente alterna ofrecida por los proveedores tiene una frecuencia de 50Hz, y que el NodeMCU cuenta con un procesador con un reloj de 80Mhz, podemos concluir que será lo suficientemente rápido (siempre que tome muestras consecutivas con poca o ninguna interrupción) para obtener las muestras necesarias.

Con el objetivo de estudiar las formas de ondas periódicas, como la onda senoidal antes mencionada, es preciso conocer el concepto de valor eficaz. El valor eficaz es el valor cuadrático medio o RMS por sus siglas en inglés (Root Mean Square) de una magnitud eléctrica. Este valor se puede obtener mediante una serie de operaciones aplicadas a los datos entregados por el sensor ACS712 durante un período de tiempo. En nuestro proyecto de investigación realizamos dichas operaciones con código similar a C (el cual luego se carga al NodeMCU y se encuentra expresado más abajo) y se pueden resumir en los siguientes pasos:

1. Leer los valores del sensor en 50 tandas de 0.02 segundos de duración cada una, tomando una muestra cada 20 µs.
2. Encontrar los promedios máximos y mínimos de las tensiones entregadas por el sensor pico a pico de la onda tanto en su parte positiva como negativa.
3. Afectar la diferencia de los promedios de las tensiones para transformar la tensión sensada a corriente según los valores de conversión del ADC del NodeMCU (que transforma valores de 0 a 3.3 v en 0 a 1024 valores digitales).
4. Dividir el valor calculado por dos para encontrar el valor medio del mismo (ya que obtenemos 2 Vpico debido a la suma de los valores absolutos de los picos negativo y positivo).
5. Multiplicar el valor medio del pico obtenido por 0.707, lo que equivale al valor cuadrático medio (correspondiente a una señal senoidal ideal) obteniendo la tensión eficaz o voltaje RMS.

En base a lo analizado anteriormente se decidió tomar los valores máximos y mínimos de 50 ciclos de onda, almacenarlos en dos arreglos (un arreglo de 50 valores máximos y otro arreglo de 50 valores mínimos) y luego promediar los valores de dichos arreglos con el objetivo de tener resultados con el menor error posible. Cada una de las 50 iteraciones a su vez toma muestras durante 0.02 segundos (que es lo que tarda un ciclo completo de onda sinusoidal) y guarda el valor pico máximo y mínimo obtenido durante ese tiempo para luego agregarlo al arreglo correspondiente.

Cabe destacar que con la velocidad de procesamiento del NodeMCU (80MHz) una instrucción tarda en realizarse 12.5 nanosegundos. Para leer el valor del pin anlógico y actualizar los picos máximos y mínimos utilizamos 6 instrucciones, entonces, una muestra es obtenida del sensor cada 75 nanosegundos obteniendo durante el período de 0.02 segundos unas 26666.66 muestras. Como esa cantidad de muestras es excesiva para el estudio de la onda que se debe realizar, se decidió agregar una instrucción más que detenga el muestreo durante 20 us, dándonos un total de 1000 muestras (aproximadas) con un intervalo exacto de tiempo entre cada muestra.

Una vez calculado el voltaje RMS, falta multiplicar el resultado por el factor de escala (sensibilidad) correspondiente a la versión del ACS712 que estemos utilizando (de 5, 10 o 30 A, datos detallados en la figura 2.4) para obtener el valor de RMS de la corriente que se está midiendo. En nuestro proyecto se decidió utilizar la versión de 30A lo que corresponde a un valor de sensibilidad de 0.66.

A continuación se detalla el código utilizado para la medición y la obtención de los valores analizados anteriormente. Cabe aclarar que el código a continuación es el que se utiliza solo para la medición de corriente y la obtención del valor de potencia aparente a partir de las mediciones ya que el código en su totalidad cargado en el NodeMCU incluye también las funciones de configuración de la red y las funciones de envío de datos y recepción de comandos a y desde el servidor:

```
float func_read_current_sensor() {
  const int sensorIn = A0; //pin ADC del NodeMCU.
  int mVperAmp = 66; //Sensibilidad del sensor de 30A según
fabricante.
  double voltage = 0;
  double VRMS = 0;
  double AmpsRMS = 0;
  float inputV = 220.0; //tensión de la red, podría ser enviado
por parámetro para achicar el error.
  unsigned int pF = 100; //factor de potencia con coseno de Phi en
1.
  float WH = 0;
  voltage = getVPP(); //se obtiene la corriente media.
  VRMS = voltage * 0.707; //se multiplica la corriente eficaz
obtenida por valor cuadrático medio.
  AmpsRMS = (VRMS * 1000) / mVperAmp; //se multiplica el voltaje
```

```
RMS por mil para pasar de milivolts a volts y se divide por la
sensibilidad del sensor.

  WH = (inputV * AmpsRMS) * (pF / 100.0); //se multiplica la
corriente eficaz por la tensión y el resultado por el factor de
potencia (1 en nuestro caso) para obtener la potencia aparente.

  return (WH); //se retorna la potencia aparente para ser enviada
al sistema.
}
float getVPP()
{
  float result;
  int readValue;
  int maximos[50];
  int minimos[50];
  float resultMax = 0;
  float resultMin = 0;
  Metro sensor_metro = Metro(0.02); //timer de 2 microsegundos.

  for  (int i = 0; i < 50; ++i) { //se toman 50 ondas.

      int maxValue = 0;
      int minValue = 1024;
      uint32_t start_time = millis(); //milisegundos actuales.

      if(sensor_metro.check()) {
           while((millis()-start_time) < 20) //muestreo durante
0.02 segundos.
           {

                readValue = analogRead(C_SENSOR1); //se lee del
sensor.
                if (readValue > maxValue)
                {
                     maxValue = readValue; //pico máximo de la
onda actual.
                }
                if (readValue < minValue)
                 {
                     minValue = readValue; //pico mínimo
(negativo) de la onda actual.
                 }
           }
      }
      maximos[i] = maxValue; //se guardan los picos en sus
arreglos.
      minimos[i] = minValue;
```

```
  }

   //se promedia cada pico.
   int promMax = 0;
   int promMin = 0;
   for (int i = 0; i < 50; ++i) {
      promMax = promMax + maximos[i];
      promMin = promMin + minimos[i];
   }

   resultMax = promMax / 50;
   resultMin = promMin / 50;

   //se restan los promedios de los picos, los convierto a valores
digitales según el voltaje del NodeMCU y los divido por 2 para
tener su media (los picos mínimos son negativos por lo que se
suman a los positivos en la resta).
   result = ((((resultMax - resultMin) * 3.3)/1024.0) / 2.0);

   return result;
}
```

3. Desarrollo

En este capítulo se detalla cada etapa de la investigación y del desarrollo del sistema, tanto de la idea general, como de cada sección que compone al sistema completo en sí mismo. Empezando por la arquitectura general del sistema, siguiendo con el servidor, las estrategias y protocolos de comunicación con los dispositivos, el modelo de datos, las tareas automatizadas y las estadísticas. Por último se realiza una explicación de las interfaces gráficas web y móvil junto sus características en usabilidad y las herramientas que brindan a los usuarios.

3.1 Arquitectura y Funcionamiento General del Sistema.

En esta sección se hará un análisis sobre la arquitectura general del sistema dividiéndola en diferentes capas o módulos desacoplados que interactúan entre sí para llevar a cabo todas las tareas que competen al sistema.

En una primera aproximación, la idea general detrás de Elektron es la de un sistema que mediante una serie de módulos desacoplados provea tanto a usuarios como desarrolladores una plataforma para gestionar, controlar y monitorear el consumo eléctrico de cualquier aparato eléctrico con capacidad de conectarse a un tomacorriente común. El sistema consta de un dispositivo de sensado -tipo adaptador- que mide la corriente exigida por el aparato electrónico a monitorear, un servidor que centraliza los datos de todos los dispositivos de medición y los almacena para futuras consultas y dos interfaces, una web y otra móvil, amigables al usuario para la sencilla visualización de los datos (tanto en tiempo real como históricos) y control del sistema en general.

Para tener una mejor visualización, mantenerlo ordenado y fácil de desarrollar, testear y extender cada una de las partes del sistema, se tomó la decisión de dividir al mismo en módulos desacoplados, que se comuniquen entre sí mediante diversos canales. A continuación se hará una introducción general al sistema visto desde el usuario, para luego profundizar en los siguiente capítulos de este informe cada uno de los módulos.

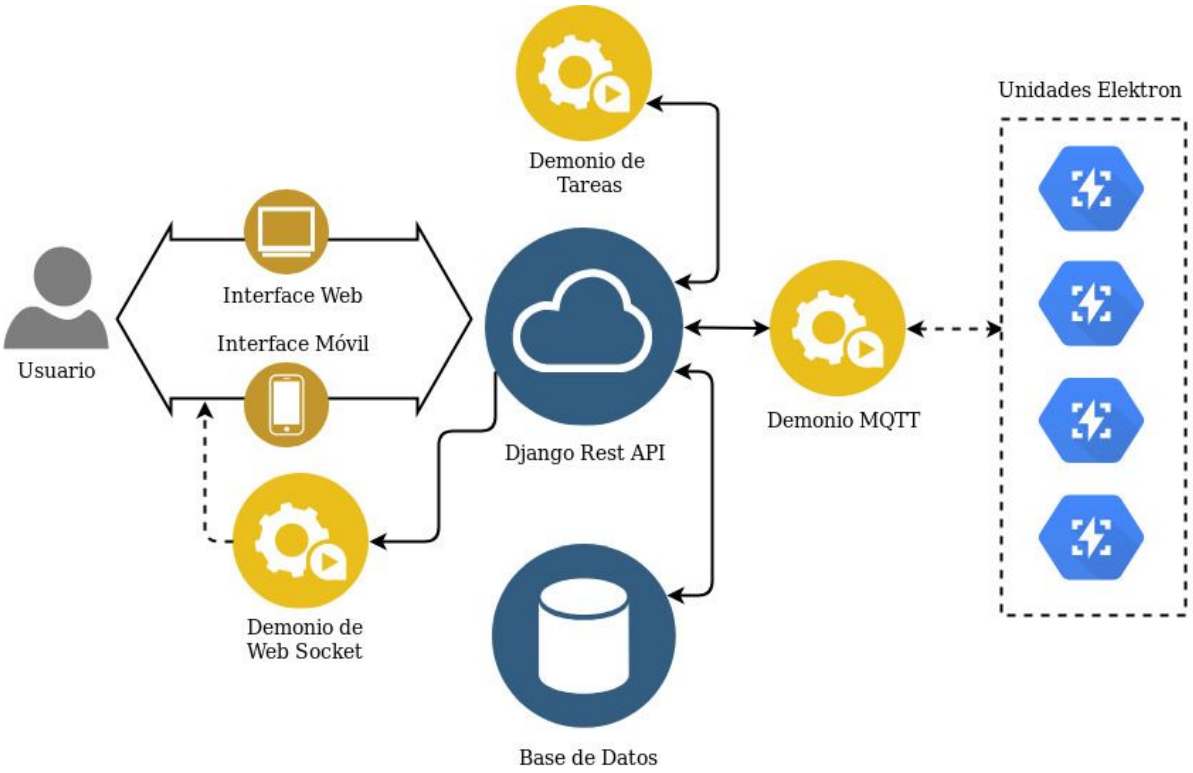


Figura 3.1: Modelo del Sistema General con Demonios. Fuente: Elaboración propia.

Para entender mejor el funcionamiento del sistema lo más sencillo es verlo desde la perspectiva del usuario final, que deberá tener en su poder un dispositivo de medición Unidad Elektron o Elektron (seguir la lectura referenciando la figura 3.1).

El usuario primero deberá conectar la Unidad Elektron (la cual se le provee ensamblada y funcional) en cualquier tomacorriente hembra de su casa, oficina, comercio o donde desee, el cual se encenderá automáticamente y ejecutará su código interno que se explica paso a paso más adelante. El usuario luego podrá conectar el dispositivo a medir al tomacorriente hembra que la Unidad Elektron provee y posteriormente a una configuración de conexión inicial se iniciará el sensado del dispositivo o electrodoméstico conectado.
Es importante indicar en este punto que se precisa del módulo servidor del sistema Elektron instalado en algún equipo remoto (por ejemplo un servidor privado virtual o VPS por sus siglas en inglés con los permisos y firewalls correctamente configurados para comunicación externa) o un servidor local (algún ordenador conectado a la red local). También cabe destacar que es preciso contar con alguna red local (WiFi) con acceso a Internet en caso de tener un servidor remoto o una red local (WiFi) con o sin acceso a Internet en caso de tener el servidor local o que no tenga la posibilidad de tener ese tipo de conexión externa a Internet. Tanto en redes locales con acceso o sin acceso a Internet, el sistema funciona de la misma manera, con la diferencia que con una red sin acceso a Internet sólo será posible obtener datos y enviar órdenes con dispositivos conectados a dicha red local mientras que con una red con acceso a un servidor remoto será posible manejar el sistema desde cualquier sitio (en ambos casos es posible utilizar tanto la aplicación móvil como la interfaz web).

Una vez conectada la Unidad Elektron a un tomacorriente (teniendo una red local WiFi configurada con o sin acceso a internet) tratará de conectarse a la red local con las credenciales

que estén almacenadas en su memoria interna (proceso que se explica con profundidad de detalles en su correspondiente apartado más adelante). Para el caso de que no tenga las credenciales correctas o se haya cambiado de contraseña de la red, la Unidad Elektron se configurará automáticamente poniéndose en modo punto de acceso para que el usuario se conecte a la misma y configure las credenciales de la red (SSID y password) y la IP o url del servidor.
Luego de que la Unidad Elektron logra acceder a la red local WiFi comienza a enviar datos del sensado del aparato que esté conectado al servidor con el Sistema Elektron principal instalado.
El usuario podrá entonces logearse al sistema principal de Elektron a través de la interfaz móvil utilizando su dispositivo móvil o a través de la interfaz web utilizando cualquier navegador en cualquier ordenador (cabe destacar la diferencia de la conexión al sistema tanto si es remoto como si es local, la cual sólo es la url o IP del servidor alojando al sistema y la red de conexión del usuario).

Luego de haber ingresado al sistema, el usuario podrá visualizar todas las Unidades Elektron que estén conectadas a la red eléctrica y con configuración de conectividad para esa ip o url del servidor donde ingreśo al sistema (cada una de las interfaces de usuario gráficas tanto web como móviles serán descritas detalladamente con imágenes en sus respectivos capítulos más adelante). Las unidades Elektron listadas en la pantalla inicial de las interfaces web y móvil se hallarán en estado deshabilitado (si acaban de ser conectadas y nunca se las habilitó), lo que inhabilita la captura y el envío de datos por parte de las mismas.
El usuario puede habilitar dichos dispositivos para comenzar a recibir los datos de medición de las aparatos eléctricos conectados a la Unidades Elektron. El sistema provee a lo usuarios monitorear de esta manera los aparatos eléctricos que desee así como también apagarlos o encenderlos, crear alertas, tareas automáticas y generar estadísticas por intervalos de tiempo configurables.

Una vez comprendido el funcionamiento del sistema en general pasaremos a describir brevemente cada módulo y las tecnologías utilizadas para su funcionamiento.

3.1.1 Comunicación entre los dispositivos y el servidor:

Para la comunicación entre los dispositivos de medición y el servidor elegimos utilizar un protocolo de mensajería liviano para comunicaciones máquina a máquina o M2M (machine to machine). El mismo es conocido con el nombre de MQTT . Es un protocolo de mensajería tipo publicadores / suscriptores, extremadamente simple y liviano, diseñado para dispositivos con restricciones de hardware que precisen interactuar en redes de bajo ancho de banda, alta latencia o poco confiables. Los principios de diseño son minimizar el uso del ancho de banda de la red y los requisitos de recursos del dispositivo al tiempo que se intenta garantizar la fiabilidad y cierto grado de seguridad de la entrega. Estos principios también hacen que el protocolo sea ideal para el mundo emergente de "máquina a máquina" o "Internet de las cosas" de dispositivos conectados, y para aplicaciones móviles donde el ancho de banda y la potencia de la batería son de primordial importancia.

3.1.2 Módulo de Hardware, Dispositivo: Unidad Elektron:

Cada Unidad Elektron consta de un objeto complejo compuesto por componentes de electrónica, microelectrónica y electricidad. Para esta sección mencionaremos los componentes que forman parte de cada Unidad Elektron:

- Una plaqueta NodeMCU basado en ESP8266 12-e o ESP32 de ESPRESSIF.
- Un sensor de corriente AD/DC invasivo ACS712 de Allegro Microsystems.
- Un módulo relé de una unidad Arduino compatible.
- Una fuente de alimentación simple de 5V 1A con cable micro usb compatible con la plaqueta NodeMCU.
- Un tomacorriente macho.
- Un tomacorriente hembra.

En la siguiente imagen (figura 3.2) se muestran los elementos esenciales de la unidad elektron: el NodeMCU, el sensor ACS712 y el Relé.

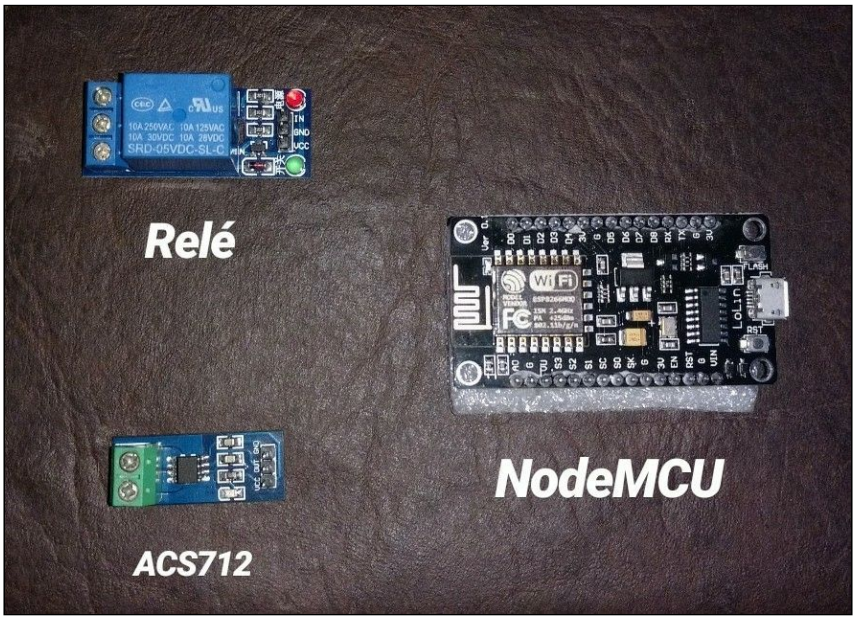


Figura 3.2: Foto del NodeMCU, ACS712 y Relé. Fuente: Elaboración propia.

El funcionamiento, descrito brevemente, comienza en el sensor de corriente conectado al NodeMCU, que mide el consumo del dispositivo electrónico conectado a la Unidad Elektron y envía dichos datos al pin análogico (A0) del NodeMCU. Este último envía cada una determinada cantidad de tiempo el dato del sensado a través de WiFi utilizando el protocolo liviano de mensajería por colas MQTT al servidor, que espera el dato con un demonio dedicado a MQTT (se detalla más adelante) para luego fusionarse con el resto del sistema. Los datos medidos constan de la IP del dispositivo (asignada por el router al conectarse por primera vez), la mac (única por dispositivo), el nombre o etiqueta del mismo, el estado (encendido o apagado) y los datos del sensado en Watts.

La librería utilizada en los dispositivos para que se comuniquen con el servidor mediante el protocolo MQTT lleva el nombre de PubSubclient con licencia MIT **[9]**. Se optó por esta librería ya que es una de las pocas desarrolladas para ser incluídas en dispositivos basados en microcontroladores como es el NodeMCU 1.0.

3.1.3 Backend

En esta sección se describe en profundidad todos y cada uno de los aspectos que integran al servidor y todos sus módulos (demonios) que interaccionan para el correcto y completo funcionamiento del sistema en todas sus capas.

3.1.3.1 Módulo Demonio MQTT (Dispositivos - Servidor)

Es el módulo de demonio que recibe los datos desde los dispositivos a través de sus canales MQTT o tópicos. MQTT utiliza la estrategia de comunicación de publicadores y subscriptores y para lograr ésto se utilizan los tópicos. Un tópico en MQTT no es más que un canal o una cola donde se almacenarán los mensajes publicados a dicho tópico y se consumirán en el instante de que son publicados por los clientes que estén suscritos a ese tópico. La mecánica es la siguiente: cuando un dispositivo, en nuestro caso podría ser cualquier Unidad Elektron o el Servidor Elektron publican un mensaje en un tópico, inmediatamente los dispositivos suscritos a dicho tópico recibirán ese mensaje. En nuestro sistema se cuenta con un tópico de comunicación Unidad Elektron al Servidor y múltiples tópicos (uno para cada Unidad Elektron conectada y dada de alta en el Servidor Elektron distinguibles unívocamente por la mac de cada dispositivo). En el primer tópico o el tópico de datos se encolan todos los mensajes de sensado que cada Unidad Elektron publique e inmediatamente se recibirán en el demonio de MTT que luego procederá como se explica más adelante. En el segundo tópico o tópico de órdenes (uno por cada Unidad Elektron) se publicarán las órdenes que el usuario o el servidor mediante tareas automáticas le enviarán a cada dispositivo unívoco para que las ejecute (prender o apagar).
El demonio MQTT luego de que recibe un mensaje por su tópico de datos se encarga de consultar a una API desarrollada con el framework de Python Django si el dispositivo que envía los datos ya existe en el sistema (a través de su dirección mac). Si el dispositivo no existe, lo da de alta deshabilitado. Los dispositivos siempre se dan de alta por primera vez en modo deshabilitado y es tarea del usuario habilitarlos en la lista de dispositivos de la pantalla principal de dispositivos de cualquiera de la interfaces del sistema. Si los dispositivos no están habilitados por algún usuario el demonio MQTT no acepta los datos, sólo toma datos de los dispositivos habilitados. Ésto fue pensado de ésta manera para tener un mejor control de los dispositivos conectados por los usuarios y por una cuestión de seguridad (para evitar denegaciones de servicio o inundaciones de datos malintencionadas, por dar algunos ejemplos).
En el caso en que el dispositivo esté habilitado, el demonio crea un objeto de tipo dato (con la fecha que lo recibió) en la base de datos del sistema, utilizando la API Rest Django por cada dato que envía cada Unidad Elektron. El demonio también se encarga de enviarle órdenes o datos a los dispositivos o Unidades Elektron (utilizando el tópico de órdenes) en el caso que el

usuario apague o encienda un dispositivo o se ejecute alguna tarea programada de apagado o encendido.
A su vez, el demonio de MQTT, se encarga de enviar los datos de los dispositivos habilitados a otro demonio del servidor, llamado demonio de Websocket, que se explicará en la siguiente sección.
Este módulo está escrito en el lenguaje de programación Python y utiliza la librería Paho MQTT que es descripta más adelante.
En conclusión, este módulo recibe datos por un canal MQTT y los persiste en la base de datos alojada en el servidor mediante mensajes POST a un url de la API que brinda dicho servidor. Al estar separado del sistema central (servidor que controla la base de datos y el resto de los módulos) podría reemplazarse este módulo con otros que utilicen distintos protocolos de mensajería como COAP o WiFiDirect, como también cambiar la API conectada a la Base de Datos, pudiendo utilizar Symfony, Ruby o el lenguaje que nos convenga según nuestras necesidades o posibilidades.

3.1.3.2 Módulo Demonio Websocket (Servidor - Interfaces Gráficas)

Este demonio, así como el anterior, se ejecuta por separado del resto de los módulos del sistema y se comunica con estos mediante pasajes de mensajes en colas.
El demonio de Websocket, utilizando la tecnología de comunicación Websocket, se encarga de enviar los datos obtenidos desde el demonio MQTT mencionado anteriormente a todas las instancias de las interfaces web y móviles que estén con los puertos Websockets abiertos y conectados al servidor local o remoto.
Las instancias de las interfaces gráficas (tanto las web como las móviles) abrirán su puerto de comunicación vía Websocket cuando se hallen en la pantalla propia de algún dispositivo habilitado recibiendo datos o cuando se hallen en la pantalla principal de monitoreo de datos en tiempo real de todos los dispositivos. Dichas pantallas mencionadas anteriormente obtienen los datos vía Websocket y los traducen en gráficos que cambian en tiempo real enseñando las variaciones en el consumo eléctrico para cada dispositivo habilitado de manera amigable al usuario. Si las pantallas que enseñan los datos en tiempo real no están instanciadas en las interfaces gráficas, los puertos Websocket del demonio se quedan esperando conexiones de dichos clientes pero continúan recibiendo datos del demonio MQTT. Este módulo, como el descrito anteriormente, puede desacoplarse del resto del sistema pudiendo ser intercambiado o removido totalmente en el caso de no ser necesario el envío de datos en tiempo real. Además, como se mencionó con respecto al módulo anterior, es posible cambiar desde dónde se reciben los datos que serán enviados por Websocket en caso de precisar otra tecnología de comunicación entre dispositivos que no sea MQTT.
Este módulo está escrito en el lenguaje de programación Python y utiliza la librería Tornado que es descripta más adelante.

3.1.3.3 Módulo Servidor API Rest Django (Servidor - Base de Datos)

Este módulo puede llegar a considerarse como el principal visto del punto de vista de las arquitecturas de aplicaciones web ya que es el encargado de obtener y persistir los objetos de datos provenientes del demonio MQTT o de la interfaces gráficas (mediante los métodos HTTP POST y GET) en la base de datos a través de consultas a la misma. Siempre que el demonio MQTT reciba los datos de un dispositivo este será validado con el servidor API Rest Django con el método GET y luego realizará el envío de los datos de los dispositivos habilitados con el método POST para su almacenado. El servidor API Rest Django persistirá tanto los dispositivos nuevos como los datos de todos aquellos que estén habilitados al recibirse con el método POST desde el demonio MQTT. También tiene la responsabilidad de brindar a las interfaces web una serie de urls para la consulta de dispositivos, datos, tareas y otros objetos para la generación de estadísticas, gráficos y el envío de órdenes o altas de tareas automáticas. Por último el servidor API Rest Django interacciona activamente con el módulo demonio de Tareas Automáticas que se explicará a continuación.

3.1.3.4 Módulo Demonio de Tareas Automáticas

Este módulo tiene la función de manejar las tareas automáticas creadas por los usuarios del sistema. Las tareas pueden ser de dos tipos diferentes, tareas por valor de dato (DataTask) y tareas por fecha y hora (DateTimeTask). Las tareas tienen asociados estados, funciones y dispositivos. Los estados de una tarea pueden ser dos, lista o terminada (ready y done); Las funciones que ejecutarán las tareas pueden ser dos: encender y apagar (turn on y shutdown). Los dispositivos asociados a las tareas pueden ser cualquiera de los dispositivos conectados y dados de alta en el sistema.
Por otro lado, el manejador de tareas (TaskHandler) es un demonio que se ejecuta en segundo plano en el servidor verificando los estados de las tareas cada segundo. Consulta al servidor Django por las tareas en estado Ready (tareas listas a ser manejadas) y las inserta en una cola de tipo objeto Queue de Python. Luego las tareas son desencoladas para ser manejadas, el manejador trata de manera diferente cada tipo de tarea.
Las tareas de tipo DataTask (tareas según valor o tareas por valor) deben ejecutar su función cuando el manejador detecte que algún dato de su dispositivo asociado iguala o supera o sea menor al valor asociado a la DataTask. Para esto el manejador consulta el último dato que haya recibido el servidor (a través del Demonio MQTT) por parte del dispositivo asociado a la tarea.
Para las tareas de tipo DateTimeTask el manejador debe tener en cuenta el atributo de tipo datetime que la tarea contiene ya que este atributo contendrá la fecha y la hora (minuto, hora, día, mes y año) en la cual se deberá ejecutar la función asociada a la tarea. El demonio manejador de tareas consulta la fecha actual y la compara con la fecha de la DateTimeTask, si la fecha actual es igual o mayor a la fecha a comparar, ejecuta la función correspondiente con el dispositivo asociado. Ambos tipos de tareas pueden configurarse para repetir su funcionamiento la cantidad de veces que disponga el usuario que las crea o edita. Cada vez que se cumple la condición para que se complete la tarea restará un ciclo de repetición y se actualizará la fecha de última ejecución para avisar al demonio de tareas que dicha tarea ya se ejecutó recientemente.

Cabe destacar que esta ventaja modular ayuda mucho a la escalabilidad del sistema y la apertura del mismo en cuenta a la posibilidad de que otros programadores o investigadores puedan reutilizar cada módulo por separado o cambiar módulos según su conveniencia.

3.1.3.5 Python y Django

El sistema, como ya se mencionó en varias ocasiones en este documento, está compuesto por una serie de módulos desacoplados, que interactuando entre sí mediante el pasaje de mensajes con diversas estrategias y protocolos, llevan a cabo múltiples tareas que terminan, en su conjunto, cumplieando los objetivos para los cuales se pensaron y desarrollaron. Todos estos módulos, y prácticamente todo el sistema (con excepción de las interfaces web y móvil), están desarrollados en el lenguaje de programación Python. Se optó por Python por las características que se describen a continuación:

- Lenguaje Interpretado: al momento de desarrollar representa una ventaja enorme, ya que las pruebas pueden realizarse inmediatamente después de que los cambios fueron persistidos. Si bien el hecho de que sea un lenguaje interpretado no ayuda demasiado en lo relacionado a la performance, existe en Python la posibilidad de realizar módulos en algún lenguaje compilado (C++ por ejemplo), los cuales pueden ser importados y utilizados desde Python.

- Gran Número de Librerías: algo que siempre caracterizó a Python es el hecho de que posee un espectro muy interesante de librerías para muchas actividades. Desde ORMs para el modelado de bases de datos siguiendo una orientación a objetos, pasando por Frameworks para el desarrollo de interfaces web y utilización de JSON, hasta módulos para el manejo de protocolos como HTTP, de forma muy simple pero no por eso menos potente. De las librerías más importantes que provee Python para el desarrollo de nuestro proyecto podemos nombrar:

    - Tornado: Tornado es un conjunto de librerías desarrolladas en el lenguaje de programación Python que en su conjunto forman un framework web para gestionar conexiones de red asincronas. Gracias a la utilización de entradas y salidas no bloqueantes, Tornado nos permitió escalar a grandes números de conexiones de consultas largas de Websocket, entre otro tipo de conexiones. En nuestro desarrollo era imprescindible mantener una conexión abierta de Websocket por cada cliente o interfaz que esté observando los datos graficados en tiempo real con el fin de generar una retroalimentación realista que permita al usuario comprender y analizar los datos en cada instante de tiempo.

    - Paho MQTT: es un proyecto desarrollado por IBM en el que se implementan y desarrollan una serie de librerías y clientes para la extensión y el uso del protocolo MQTT, fundamental para nuestro desarrollo. El proyecto Paho además implementa librerías para ser utilizadas en varios lenguajes de programación como JAVA y Python entre sus principales desarrollos.

- HTTPie: HTTPie se compone de un solo comando HTTP diseñado para la depuración simple y la interacción con servidores HTTP, APIs Restful y servicios web.

Del lenguaje Python además logramos aprovechar la implementación de un Framework conocido como Django para el core principal del sistema.

Django es un framework web desarrollado en y para el lenguaje de programación Python que simplifica a los desarrolladores la construcción de sistemas de web robustos de manera rápida, limpia y escalable. Es de código abierto y gratuito permite la interacción con una diversa familia de librerías y utilidades desarrolladas en Python.

Django, como la mayoría de los Frameworks web, respeta el patrón de diseño conocido como Modelo–vista–controlador con la leve diferencia de que en Django se gesta una estrategia que los creadores llaman Modelo Vista Template donde el Modelo cumple la misma funcionalidad que el Modelo de cualquier otro Framework (la parte de representación de los datos), la Vista cumple la función de Controlador o de cómo se obtendrán y se filtraran dichos datos, también conocido como lógica; y por último la parte del Template que cumple la función de Vistas o de renderización de los datos como en otros Frameworks.
Como se expresó anteriormente la meta fundamental de Django es facilitar la creación de sitios web complejos.

3.1.3.6 API Rest y Modelo Detallado

El sistema general y los módulos que interaccionan cumplen con las características de una API ya que de esta manera se vuelve natural el uso desacoplado de cada módulo. El servidor Django, intenta respetar con todas las características de una estrategia de tipo Rest para el despliegue de sus servicios. En esta sección se desarrollará brevemente el concepto de una API y qué es una estrategia Rest, para luego explicar dichas características relacionadas a nuestro sistema.

Una interfaz de programación de aplicaciones (API) es un conjunto de definiciones de programas y tareas ofrecidas como biblioteca para ser usados otros software de capa de aplicación.
En términos generales, se trata de un conjunto de métodos de comunicación claramente definidos entre varios componentes de software. Una buena API facilita el desarrollo de un programa informático proporcionando todos los bloques de construcción, que luego son agrupados por el programador. Una API puede ser para un sistema basado en la web, sistema operativo, sistema de base de datos, hardware o biblioteca de software. Una especificación de API puede tomar muchas formas, pero a menudo incluye especificaciones para rutinas, estructuras de datos, clases de objetos, variables o llamadas remotas. POSIX, la API de Microsoft Windows, la biblioteca de plantillas estándar de C ++ y las API de Java son ejemplos de diferentes formas de API.

La transferencia de estados representativos (Rest por sus siglas en inglés Representative State Transfer) o los servicios web Restful, es una forma de proporcionar interoperabilidad entre sistemas informáticos en Internet. Los servicios web compatibles con Rest permiten a los sistemas solicitantes acceder y manipular representaciones textuales de recursos web utilizando un conjunto uniforme y predefinido de operaciones sin estado. Existen otras formas de servicios web, que exponen sus propios conjuntos arbitrarios de operaciones como WSDL y SOAP **[10]**.
Los "recursos web" se definieron por primera vez en la World Wide Web como documentos o archivos identificados por sus URL (Dirección de Recursos Uniformes) , pero hoy en día tienen una definición mucho más genérica y abstracta que abarca todo lo que puede ser identificado, nombrado, dirigido o manejado en cualquier manera alguna, en la web. En un servicio Web Restful, las solicitudes realizadas en el URI (Identificadores de Recursos Uniformes) de un recurso generarán una respuesta que puede estar en XML, HTML, JSON o algún otro formato definido.

La respuesta puede confirmar que se ha hecho alguna alteración en el recurso almacenado y que puede proporcionar enlaces de hipertexto a otros recursos o colecciones de recursos relacionados. Usando HTTP, como es más común, el tipo de operaciones disponibles incluyen las predefinidas por los verbos HTTP GET, POST, PUT, DELETE y así sucesivamente. Mediante el uso de un protocolo sin estado y operaciones estándar, los sistemas Rest buscan un rendimiento rápido, fiabilidad y la capacidad de crecimiento, al volver a utilizar componentes que se pueden gestionar y actualizar sin afectar al sistema en su conjunto, incluso mientras se está ejecutando. El término transferencia de estados representativos fue introducido y definido en 2000 por Roy Fielding en su tesis doctoral. Fielding utilizó Rest para diseñar HTTP 1.1 y URI. El término pretende evocar una imagen de cómo se comporta una aplicación web bien diseñada: es una red de recursos web (una máquina de estado virtual) donde el usuario avanza a través de la aplicación seleccionando vínculos como / user / juan, y operaciones tales como GET o DELETE (transiciones de estado), dando lugar a que el siguiente recurso (que representa el siguiente estado de la aplicación) se transfiera al usuario para su uso.

Si bien en el desarrollo de nuestra aplicación central desarrollada en Django no responde en su totalidad a una API Restful como se describió anteriormente, comparte muchas de las cualidades, características y estrategias de una.
Nuestra aplicación de DJANGO corre en el puerto 8000 de cualquier servidor (remoto o local) y queda a la espera de peticiones del protocolo HTTP con los métodos POST o GET (en una API Rest o Restful se implementan casi todos los métodos del protocolo HTTP en estados representacionales equivalentes, en nuestra solución no es así) para la consulta, creación, modificación o baja de los objetos modelados.
Los objetos modelados en la última versión de nuestro desarrollo se describen a continuación, además de representarlo en un diagrama de clases (figura 3.3):

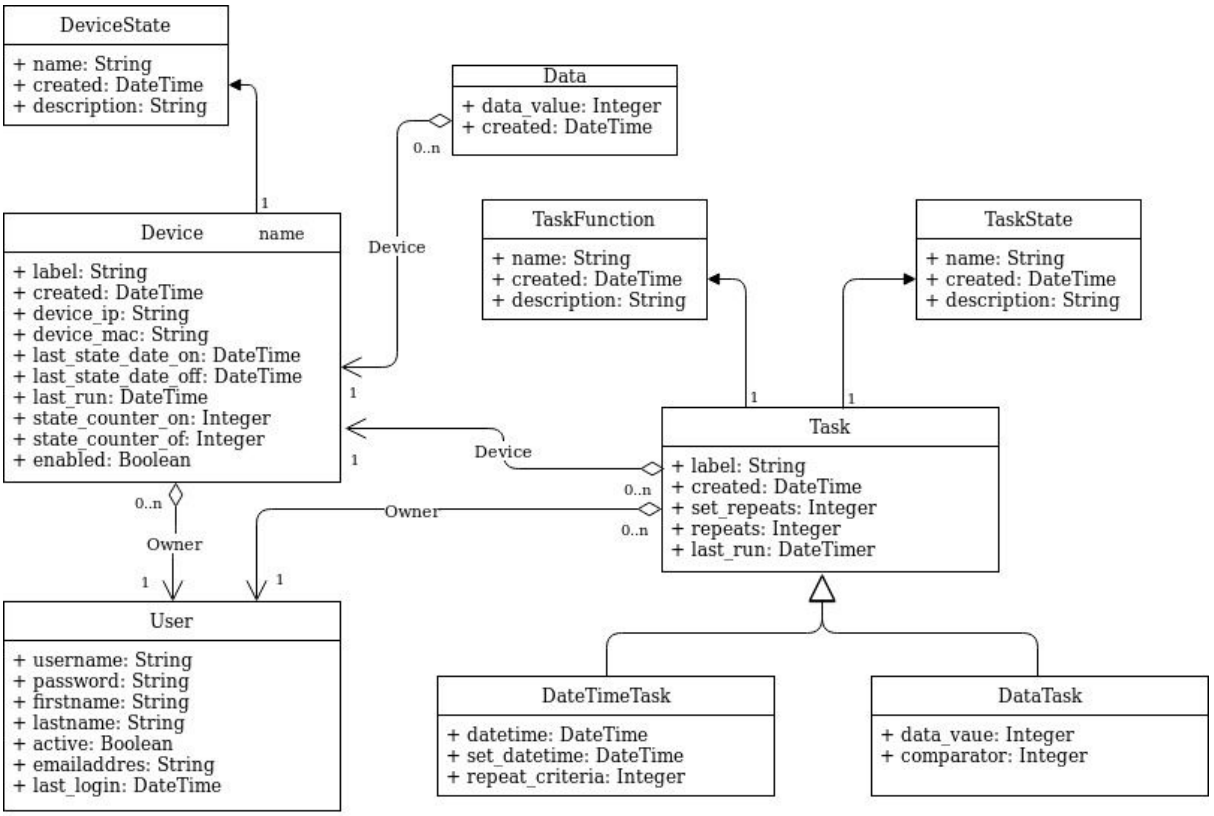


Figura 3.3: Diagrama de Clases UML. Fuente: Elaboración propia.

- Device (dispositivo): en el modelo de datos, la clase Device pretende ser una representación lógica de los dispositivos o Unidades Elektron descritos en capítulos anteriores. Esta clase contiene las características lógicas de los dispositivos físicos necesarias para la interacción y manipulación de los mismos, dichos atributos son:

    - Device_ip (dirección IP del dispositivo): Representa la dirección IP de cada dispositivo. Esta dirección IP es obtenida desde el dispositivo la primera vez que envía un mensaje al demonio MQTT, el cual consulta a Django si existe preguntando por su mac y en caso de no existir, lo crea mediante un mensaje POST con los datos que el dispositivo envía de de sí mismo, entre los cuales uno de los más importantes es su IP. Si bien la IP del dispositivo es importante para poder conectarse a él directamente de ser necesaria alguna configuración manual, no es necesaria para recibir mensajes ni enviar órdenes a estos ya que el protocolo usado de mensajería (MQTT) se basa en suscriptores y escritores, tanto los dispositivos como el demonio de MQTT se suscriben a distintos tópicos unívocos para cada uno (basándose en su mac) gestionados por el broker MQTT alojado en el servidor y de esta forma reciben las actualizaciones de datos y órdenes.
    - Device_mac (dirección mac del dispositivo): representa la dirección de hardware del dispositivo. Como la dirección IP y los demás datos del dispositivo, es recibida por el servidor en cada paquete de datos que es enviado desde el demonio MQTT cuando este la recibe desde cada dispositivo. La dirección mac es importante ya que al ser única por dispositivo es utilizada por el servidor y el demonio MQTT para la distinción unívoca de cada dispositivo con el objetivo de

detectar dispositivos nuevos o saber de qué dispositivo es cada dato y poder enviar señales o funciones a cada uno.

- Created (fecha de creación del dispositivo): la fecha de creación en el modelo de datos lógico de la clase Device no es más que un registro interno de cuándo fue dado de alta el mismo por parte del demonio MQTT (que sería la primera vez que el dispositivo se comunicó con el servidor). Este puede ser un dato útil teniendo en cuenta que gracias a él se podrá ver cuándo el dispositivo fue conectado sin necesidad de habilitarlo para recibir datos del mismo.
- Label (etiqueta o nombre): el label de la clase dispositivo sirve para que los usuarios puedan asignarles un nombre representativo para ellos según el uso que se le esté dando (el aparato electrónico que se esté monitoreando, por ejemplo, heladera, pava eléctrica o impresora 3d). El label por defecto que tendrán los dispositivos es Elektron más los últimos cuatro caracteres de su mac con el objetivo de identificarlos unívocamente. Aunque siempre es recomendable que el usuario asigne un nombre representativo.
- Enabled (habilitado): este atributo de la clase Device es uno de los más importantes. Es el indicador que va a permitir a los dispositivos ser escuchados y tenidos sus datos en cuenta por parte del demonio MQTT. Cuando un dispositivo se conecta por primera vez al sistema, es decir, cuando le envía un primer mensaje al demonio MQTT, este último verifica que el dispositivo exista en el sistema (comparando su mac con los dispositivos ya dados de alta) y en caso de no existir, lo crea con el flag de enabled deshabilitado (en False). Los usuarios pueden buscar los dispositivos en la lista de dispositivos en cualquiera de las interfaces gráficas (web o móvil) y cambiar el estado del dispositivo de deshabilitado a habilitado y vice versa. Cada vez que un dispositivo envía un dato al demonio MQTT este primero verifica su mac y luego verifica que esté habilitado. Si el dispositivo no está habilitado (enabled en False) el demonio MQTT ignora los datos que este envía, en cambio, si el dispositivo está habilitado (enabled en True) el demonio MQTT crea un objeto data (que se verá más adelante) por cada dato que reciba utilizando el método del protocolo HTTP POST del servidor Django. En resumen, este flag enabled permite que se almacenen datos de los dispositivos que estén habilitados por el usuario.
- Owner (dueño): el atributo owner de la clase Device contiene una clave foránea a la clase primitiva de Django auth.User del modelo de clases de autenticación que el framework provee y que no se describe en este documento para no profundizar demasiado en el framework y no perder el foco de la investigación.
- Devicestate (estado del dispositivo): este atributo de la clase dispositivo es en sí mismo otra clase (clave foránea a la clase Devicestate) que se describe a continuación:

    - Name: la clase Devicestate (estado del dispositivo) es sencilla en su composición, cuenta con un atributo name (nombre) que sirve para que los usuarios identifiquen de manera más amigable los estados de los dispositivos que podrían indicar cualquier estado de cualquier dispositivo, sea prendido, apagado, desconectado o algún estado de error por defecto, entre otras posibilidades que podrían agregarse como

estados de dispositivos en alguna posible versión más genérica o evolucionada del sistema.

        - Description (descripción): el atributo descripción ayuda a los usuarios a agregar un poco más de información al estado del dispositivo, aunque los estados encendido o apagado no tienen mucha descripción extra para agregar, un estado como el de desconectado o algún error, podría necesitar de una descripción extra sobre lo que está sucediendo con el dispositivo, por qué se encuentra en ese estado y cuáles son las posibilidades desde ese punto.

- Data (Dato): la Siguiente clase del modelo a analizar es Data. Las instancias de la clase data serán los datos creados por el demonio MQTT en caso de que el dispositivo que se esté comunicando, y enviando dichos datos, esté registrado (mac conocido de la clase Devices o nuevo pero válido) y habilitado (flag enabled de la clase Devices) entonces los datos serán dados de alta por el demonio MQTT en el servidor Django con el método HTTP POST, como se mencionó en repetidas ocasiones. La clase data debe representar de la mejor y desacoplada manera el dato enviado por el dispositivo (sea cual sea este dato), que en el caso de esta investigación será el consumo eléctrico aproximado generado por el aparato electrónico conectado a la unidad Elektron (el dispositivo que envía el dato).

  Cabe destacar que las instancias de objeto Dato creadas y almacenadas en el sistema no pueden ser editadas, creadas, modificadas o eliminadas por los usuarios ya que son datos reales enviados por los dispositivos, sin embargo podría existir la opción de configurar el sistema para que dichos datos de alguna fecha lejana sean movidos a algún almacenamiento extra para mejorar la estabilidad de la base de datos y alivianar la carga del servidor.

  Los atributos de la clase Data se describen a continuación:

    - Data_value (valor del dato): el atributo data_value no es más que el valor mismo del dato que el dispositivo envía, por ejemplo, algún dato de algún sensor que esté conectado al mismo como podría ser un sensor de corriente, humedad o temperatura. En nuestra investigación los dispositivos envían el valor del consumo eléctrico provocado por el o los aparatos electrónicos conectados al mismo. Las instancias de los objetos Data son almacenados en la base de datos para futuras consultas, con el atributo data_value es posible graficar y obtener estadísticas de los datos enviados por los dispositivos, lo cual en nuestro caso significa producir estadísticas y gráficos de los consumos eléctricos producidos por diversos aparatos. El valor de los datos es una parte clave para el demonio gestionador de tareas automáticas (de tipo por valor) que se explicó anteriormente.
    - Date (fecha): el atributo fecha de la clase Data es la fecha en que se envía el dato al demonio MQTT desde cada dispositivo. Cada dato tendrá una fecha completamente diferente debido a su naturaleza y formato. Las fechas en los datos le sirven a los usuarios del sistema para ordenarlos, generar distintas estadísticas, gráficos y sacar conclusiones. La fecha en los datos es clave para el demonio gestionador de tareas automáticas (por fecha del dato) que se explicará más adelante en su correspondiente capítulo.

    - Device (dispositivo): el atributo Device de la clase Data es un identificador o clave foránea a la instancia de clase Device (descripta anteriormente) correspondiente para dicho dato, en otras palabras, el atributo device indica de qué dispositivo corresponde el dato.

- Task (tarea): la clase Task es, tal vez, hasta el momento, la más compleja del sistema en su totalidad. La clase Task o Tarea es la encargada de brindar al usuario la posibilidad de crear tareas automaticas y planificadas basándose en el valor o fecha de los datos que son almacenados en la base de datos a partir de ser enviados en tiempo real por los dispositivos habilitados en el momento. Las tareas se dividen en dos subgrupos importantes (habiendo una subclase por cada sub tarea que hereda de la clase Task, las cuales se explicarán más adelante). Una subclase de Task, la DataTask se accionará dependiendo del valor de los datos y la otra subclase, la DateTimeTask se accionará dependiendo de una fecha y hora configurable por el usuario. Las tareas al accionarse harán dos cosas, primero, ejecutarán alguna función (para lo cual existe otra clase TaskFunction que se explicará en breve), segundo cambiarán de estado (para lo cual existe otra clase llamada TaskState que se explicará a continuación). Los atributos de la clase Task son los siguientes:

    - Created (fecha de creación): la fecha de creación de las tareas no es más que una fecha que se genera automáticamente cuando el usuario crea la tarea. Este atributo, aunque simple, es muy importante al momento de llevar a cabo las tareas ya que el demonio encargado de manejarlas necesita conocer la fecha de creación de la tarea para compararla con la fecha de emisión y recepción de los datos desde los dispositivos con el fin de validar si debe tener en cuenta esos datos o no al momento de evaluar su ejecución.
    - Label (etiqueta o nombre): tal cual como en la clase Device, explicada anteriormente, el atributo label sirve para que los usuarios puedan crear Tareas con algún nombre que sea descriptivo de la acción o tarea a realizarse.
    - Description (descripción): al igual que en la clase Device, el atributo description sirve para que los usuarios tengan la posibilidad de escribir algo de texto que sirva para entender las funciones, criterios, estados y posibilidades que pueden tener relación con las Tareas creadas.
    - TaskState (estado de la tarea): el atributo TaskState es una clave foránea a una instancia de la clase TaskState. Las tareas (Tasks) desde que son creadas hasta que terminan el trabajo que les fue asignado, pueden pasar por varios estados. Estos estados indican si la tarea está lista para ser ejecutada, si se está ejecutando o si se ejecutó. En el desarrollo que tenemos realizado hasta el momento las tareas pueden tener dos estados, ready o done (lista o terminada). Como se acaba de mencionar, cuando una tarea está en estado ready está lista para ser ejecutada por el demonio de tareas. Cuando una tarea cumple con los parámetros de valor o fecha para ser ejecutada suceden dos cosas, primero se ejecuta la acción asignada a la misma, luego se cambia su estado de lista a terminada (de ready a done). Las tareas en estado done, si bien son listadas en las interfaces de usuario son ignoradas por el demonio gestionador de tareas.

    - TaskFunction: el atributo TaskFunction (Función de la Tarea) de la clase Task corresponde a una clave foránea que apunta a una instancia de la clase TaskFunction . Dicha clase representa una función a ser ejecutada por el demonio manejador de tareas cuando corresponda manejar la tarea. Las funciones que existen como instancias de esta clase (hasta el punto de desarrollo que nuestra investigación alcanzó) son turnon y shutdown (prender y apagar). Estas funciones hacen exactamente lo que indican sus nombres, prenden o apagan el dispositivo asociado a la tarea. Esto ocurre de esta manera porque en nuestra investigación tan sólo tenemos un relevador como actuador en nuestras plaquetas pero el desarrollo tanto del servidor como del sistema de tareas y funciones está pensado para que pueda crearse y llevarse a cabo cualquier función, sea regar, sensar una variable, encender un motor, prender una bomba, etc.
    - Device: el atributo Device (Dispositivo) de la clase Task corresponde a una clave foránea que apunta a una instancia de la clase Device (descripta anteriormente). Los dispositivos asociados a las tareas se verán afectados por las funciones de la tareas cuando corresponda ejecutarlas.
    - Owner: el atributo owner (dueño) de la clase Task corresponde a una clave foránea que apunta a una instancia de a la clase auth. User propia del sistema de autenticación de Django. Este atributo Owner fue explicado anteriormente para la clase Device y cumple la misma funcionalidad en esta clase Task.

- DataTask (Tarea por Dato): la clase DataTask es una subclase de la clase Task recién descripta. Dicha subclase hereda todos los atributos de su clase padre y define un atributo nuevo llamado data_value el cual no es más que un simple valor almacenado en una cadena de caracteres para dar la posibilidad de que sea tanto un número como una palabra clave. Las tareas de tipo DataTask tienen la peculiaridad de activarse o ser ejecutadas por el demonio manejador de tareas cuando el valor de su dispositivo asociado alcanza o supera o esta por debajo del valor asignado en el atributo data_value de la instancia de DataTask correspondiente. Con este tipo de tareas es posible controlar el consumo de cierto aparato electrónico ya que si el valor del consumo en tiempo real del mismo supera el valor configurado en la tarea se le puede enviar la señal de apagado desde el demonio manejador de tareas. Cabe destacar que para mejorar la abstracción del sistema en general, los comparadores para la activación de éste tipo de tareas pueden ser menor, mayor e igual con el objetivo de brindar la posibilidad de ejecutar funciones con datos provenientes de otros tipos de sensores como ya se mencionó anteriormente.

- DateTimeTask (Tarea por Fecha y Hora): la clase DateTimeTask es otra subclase de la clase Task. Esta subclase hereda todos los atributos de su clase padre y define un atributo propio llamado datetime el cual es de tipo DateTime. Las tareas de tipo DateTimeTask se diferencian de las tareas de tipo DataTask ya que su criterio para ejecutar su función asociada corresponde a la fecha y hora almacenada en su atributo DateTime. Cuando el demonio manejador de tareas corrobora que la fecha actual (en año, mes, día, hora y minuto) es mayor o igual a la fecha de la DateTimeTask ejecuta su función asociada. Con este tipo de tareas es posible encender o apagar (o ejecutar

cualquier función) el dispositivo que deseemos en la fecha y hora que deseemos según nuestras necesidades.

- ElektronUser (Usuario Elektron): la clase ElektronUser reimplementa la clase auth.User de Django con la simple diferencia de tener sus propias urls para ejecutar las diferentes acciones con los métodos HTTP más comunes (POST y GET). No tiene caso detenerse mucho en esta clase ya que sería describir el framework Django y su sistema de autenticación y usuarios lo cual no es el punto de esta investigación.

Como puede notarse en alguna de sus clases como TaskState y TaskFunction, la idea, además de desarrollar un sistema de control y monitoreo de consumo eléctrico centralizado, es generar una plataforma o una especie de API desacoplada y modular la cual sirva para cualquier necesidad IoT que pueda presentarse pudiendo ser útil sean cuales sean los sensores, actuadores, la plaqueta y los datos a procesar.

3.1.3.7 Base de Datos: MySql e InfluxDB

Con respecto a la base de datos se eligió utilizar dos enfoques, uno para los datos simples o no muy escalables y otro esquema para los datos más significativos en relación a su incremento en cantidad. Cuando hablamos de Internet de las Cosas o de medición de variables del mundo físico en tiempo real con persistencia de dichos datos, hay que tener en cuenta que dependiendo del formato de almacenamiento de los datos y su posterior procesamiento, es posible que luego de un tiempo se tengan problemas por las grandes cantidades de estos datos.
Éstos problemas son muy comunes en sistemas de sensado y debido a su naturaleza e importancia, existen tecnologías y estrategias en materia de motores de bases de datos para tratar enormes cantidades de información.
En nuestro sistema se almacenan los usuarios y las tareas automatizadas, entre otros objetos, utilizando el motor de base de datos MySQL debido a su robustez y sencilla utilización mediante consultas desde Django.
Para los objetos como los dispositivos y los datos medidos utilizamos una tecnología de base de datos propia de la familia de Internet de las Cosas conocida como InfluxDB[7]. InfluxDB es una base de datos de código abierto basada en series de tiempo desarrollada por InfluxData. Está escrita en lenguaje Go[8] y optimizada para almacenamiento rápido y de alta disponibilidad y recuperación de datos de series de tiempo en campos tales como monitoreo, métricas, datos de sensores (Internet de las Cosas) y el interesante aporte del análisis en tiempo real. El poder de las consultas basadas en series de tiempo de InfluxDB nos permitió agrupar promedios de los datos obtenidos cada cinco segundos en series de una hora, algo de mucha importancia debido a la naturaleza de las mediciones eléctricas que se calculan como watt o kilowatt por hora, y también agrupando, promediando y obteniendo las sumatorias de dichos datos por día lo que nos permite mejorar nuestra área de estadísticas y datos históricos con unas pocas consultas a InfluxDB.

[7] https://www.influxdata.com/
[8] https://golang.org/

3.1.3.8 Seguridad

Cuando hablamos de seguridad hay que tener en cuenta dos grandes puntos, el primero es que ningún sistema informático es cien por ciento seguro (esté conectado o no a alguna red) y segundo, que la matriz de seguridad se divide en diversas capas y no siempre se puede alcanzar la seguridad de todas en un sistema complejo. Para tener una visión más ordenada de la seguridad en el sistema desarrollado vamos a dividirla en dos capas importantes: seguridad en el Backend y seguridad en los dispositivos de sensado inalámbricos (Unidades Elektron).

3.1.3.8.1 Seguridad en el Backend

Con respecto a la seguridad en el Backend hay que tener en cuenta antes de hacer un análisis que en este proyecto se utiliza una API Rest desarrollado mediante el framework web Django. Con esta tecnología se pueden analizar dos cuestiones principales, la seguridad que el framework mismo provee y la seguridad de la API Rest desarrollada con nuestras estrategias.
La seguridad de Django según sus fuentes oficiales recae sobre todo en sus componentes de Frontend, o como lo llaman en su modelo, los templates que conforman la interacción con los usuarios. En nuestro caso los templates se reemplazan mediante la comunicación de dos interfaces gráficas (la web en AngularJS y la móvil en IONIC, también con AngularJS) y nuestra API Rest.
En Django tan sólo escapa de la seguridad en interacción con los usuarios el protocolo de comunicación utilizado para las peticiones SSL/HTTPS, los cuales son utilizados automáticamente mediante la configuración de algunos archivos base del Framework, acción que se recomienda que se lleve a cabo luego de la finalización del desarrollo y puesta en producción del sistema por cuestiones de facilitación del testing para los desarrolladores.
El segundo aspecto importante de la seguridad del Backend es la API Rest. El sistema desarrollado cuenta con una API Rest que funciona de capa intermedia entre los datos en la base de datos y las distintas interfaces gráficas; las interfaces solicitan los datos a través de peticiones HTTP como POST y GET a rutas específicas de la API y esta responde con los datos correspondientes en formato JSON. En este contexto, el único problema o brecha de seguridad es tener el conocimiento de las rutas y las credenciales de autenticación para la petición de estos recursos. Dichas credenciales de autenticación son gestionadas por un administrador de autenticación que el framework Django provee y hasta la fecha no posee vulnerabilidades conocidas en su versión más reciente que es la que se utilizó para el desarrollo del sistema. Para la autenticación entre las interfaces y el servidor entonces se usó un JWT (Json Web Token), un estándar que se basa en JSON (formato que ya utilizamos) para establecer un token en los Frontend que sirva para la validación de los usuarios, protección de las rutas y generar la seguridad en esa comunicación. Para lograr el uso de JWT se necesito implementar dicho protocolo tanto en las interfaces como en el servidor.

3.1.3.8.2 Seguridad en las Unidades Elektron

En esta sección se detallan los aspectos más importantes de seguridad con respecto a los dispositivos de sensado inalámbrico (NodeMCU con microcontrolador ESP8266).

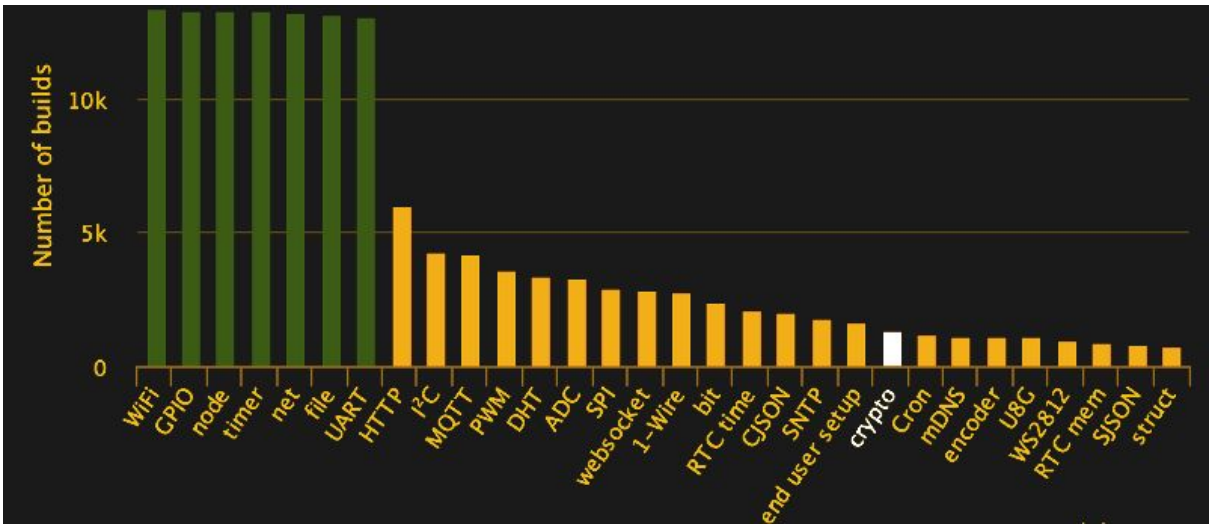


Figura 3.4: Imágenes de firmwares para NodeMCU (ESP8266) creadas durante Abril y Mayo del 2017. Fuente: Blog Hackaday.

El circuito integrado ESP8266 hace que Internet de las Cosas sea posible a bajo costo. Según el popular sitio de creación automática de firmwares a medida para NodeMCU (NodeMCU-build.com/stats.php) en donde se muestra una gráfica de los últimos 60 dias de uso (similar a la de la figura 3.4 más arriba), han habido 190753 compilaciones de firmware personalizadas para esta plataforma. De ellos, solo el 20% aproximadamente tiene soporte SSL y el 10% incluye el módulo de criptografía. Esto es un indicador de que la seguridad en el sector IoT no suele ser tenida en cuenta como su importancia lo exige.

En esa web (https://hackaday.com) un grupo de ingenieros en seguridad informática publica diariamente información sobre lo último en la temática, y entre esos tópicos, se explica la aplicación de las funciones de cifrado AES (Advanced Encryption Standard) y autorización con funciones hash para el protocolo MQTT usando el chip ESP8266 corriendo el firmware de NodeMCU.[9] El propósito del artículo es detallar paso a paso las distintas problemáticas y soluciones con respecto al desarrollo de un protocolo de seguridad más seguro para estas placas microcontroladoras. El resultado es un sistema que cifra y auténtica de extremo a extremo, evitando las escuchas a lo largo del camino y la suplantación de datos válidos, sin depender de SSL.

A pesar de que existen plataformas más potentes que pueden soportar sencillamente SSL (por ejemplo Raspberry Pi, Orange Pi, Friendly ARM) en esta investigación se hizo hincapié y se dió mayor importancia a un hardware de menor costo y más accesible para combinarlo con el muy utilizado y adecuado protocolo de IoT, MQTT. AES es algo que podría ser implementado en cualquier AVR[10] de ser necesario.
Como se mencionó en capítulos anteriores, nuestros dispositivos de sensado inalámbrico utilizan el protocolo de comunicación MQTT. Este protocolo no tiene funciones de seguridad integradas

[9] https://hackaday.com/2017/06/20/practical-iot-cryptography-on-the-espressif-esp8266/
[10] Los **AVR** son una familia de microcontroladores RISC del fabricante estadounidense Atmel. https://es.wikipedia.org/wiki/AVR

más allá de la autenticación con nombre de usuario y contraseña, por lo que es común el cifrado y la autenticación a través de una red con SSL.
Sin embargo, SSL es bastante exigente en términos de procesamiento para el ESP8266 ya que cuando se habilitada esta funcionalidad desde el firmware le queda (a nuestro dispositivo) mucha menos memoria para el resto de las funciones.
Como alternativa más ligera, se decidió cifrar solo la carga de datos que se envían, y contar con una identificación de sesión y una función hash para la autenticación.
Una forma muy sencilla de hacerlo es usar Lua y el módulo de NodeMCU Crypto, que incluye compatibilidad con el algoritmo AES en modo CBC[11], así como la función hash HMAC[12].
Usar el cifrado AES correctamente requiere de tres partes con el fin de producir texto cifrado: un mensaje a ser cifrado, una clave y un vector de inicialización (IV o Initialization Vector en inglés). Los mensajes y las claves son conceptos sencillos, pero vale la pena analizar el vector de inicialización.
Cuando se codifica un mensaje en AES con una clave estática, siempre producirá el mismo resultado. Por ejemplo, el mensaje "usernamepassword" cifrado con la clave "1234567890ABCDEF" podría producir un resultado como "E40D86C04D723AFF". Si se ejecuta nuevamente el cifrado con la misma clave y el mismo mensaje, se obtendrá el mismo resultado. Esto permite varios tipos de ataques comunes, especialmente ataques con análisis de patrones y ataques de repetición[13].
En un ataque de análisis de patrones, se utiliza el conocimiento que una determinada porción de datos siempre producirá el mismo texto cifrado para adivinar cuál es el propósito del contenido de los diferentes mensajes sin conocer realmente la clave secreta. Por ejemplo, si se envía el mensaje "E40D86C04D723AFF" antes de todas las demás transacciones, se puede llegar a la conclusión rápidamente que se trata de un inicio de sesión. En resumen, si el sistema de inicio de sesión es simplista, enviar ese paquete (un ataque de repetición) podría ser suficiente para identificarse como un usuario autorizado.
Los vectores de inicialización (IVs) hacen el análisis de patrones más difícil. Un IV es una información enviada junto con la clave lo cual modifica el resultado del texto cifrado final. Como su nombre lo indica, inicializa el estado del algoritmo de cifrado antes de que ingresen los datos. Es necesario que cada IV sea diferente para cada mensaje enviado de modo que los datos repetidos se cifren diferente, y en algunos sistemas de cifrado (como AES-CBC) se requiere que sea impredecible; una forma práctica para lograr esto es aleatorizar el IV cada vez. Los IV no tienen que mantenerse en secreto, pero es una práctica común ofuscarlos de alguna manera.
Si bien esto brinda cierta protección contra el análisis de patrones, no ayuda con los ataques de repetición. Por ejemplo, la retransmisión de un conjunto dado de datos cifrados aún duplicará el resultado. Para evitar eso, necesitamos autenticar al remitente. Es posible utilizar una identificación de sesión pública, generada de manera pseudoaleatoria para cada mensaje. Esta

[11] "*En criptografía, un modo de operación de cifrado de bloques (CBC en inglés) es un algoritmo que utiliza un cifrado de bloques para proporcionar un servicio de información como confidencialidad o autenticidad*". https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation
[12] "*En criptografía, un HMAC (a veces desabreviado como código de autenticación de mensaje hash con clave o código de autenticación de mensaje basado en hash) es un tipo específico de código de autenticación de mensaje (MAC) que incluye una función hash criptográfica y una clave criptográfica secreta*". https://en.wikipedia.org/wiki/HMAC
[13] "*Un ataque de repetición (también conocido como ataque de reproducción) es una forma de ataque de red en la que una transmisión de datos válida se repite o retrasa de manera maliciosa o fraudulenta*". https://en.wikipedia.org/wiki/Replay_attack

identificación de sesión puede ser generada por el dispositivo receptor al publicar un tópico MQTT.
Como en nuestra investigación no utilizamos Lua fue necesario investigar alguna librería compatible con Arduino (en lenguaje Wiring) para lograr un cifrado AES-CBC o similar.
Una de las librerías mejor desarrolladas hasta el momento es la librería Arduino-crypto la cual se utilizó para el cifrado de datos en las Unidades Elektron.
Con el objetivo de hacer pruebas con el cifrado AES se implementó dicho cifrado desde unos bots (que simulan el comportamiento de las Unidades Elektron). Dichos bots cifran entonces los datos antes de enviarlos al demonio de MQTT, el cual descifra el AES utilizando la librería Crypto de Python, para luego almacenar los datos y enviarlos por Websocket descifrados.
Debido a que AES precisa de una clave de cifrado secreta, que en nuestro caso está publicada junto al código, fue necesario agregarle un salt para mejorar su seguridad. Un salt no es más que un grupo de caracteres extra que se agregan al principio o final de la clave original con el objetivo dificultar el descubrimiento de la misma. En nuestro sistema el salt que se agrega a la clave de cifrado original actuará como clave privada compartida (técnica utilizada en el cifrado simétrico), pero para realizar esto es preciso que el salt sea el mismo en cada Unidad Elektron (donde se realiza el cifrado) y en el servidor (donde se realiza el descifrado). Aquí es importante recordarle al lector que al configurar por primera vez las Unidades Elektron es necesario indicarles el SSID y la Password de la red a la cual deben conectarse para enviar los datos al servidor remoto. A su vez, los usuarios deberán configurar en el sistema (también por única vez) la clave de descifrado, la cual deberá ser la password correspondiente a la red. De esta manera se puede asegurar que tanto las Unidades Elektron como el servidor compartirán parte de la clave (el salt) para el cifrado y descifrado de los datos del sensado.

3.2 Frontend: La GUI

La interfaz de usuario es una parte fundamental del sistema que venimos explicando en este trabajo. Es fundamental en el sentido que es un sistema orientado a usuarios finales que no tienen que tener necesariamente un conocimiento profundo de la electrónica y/o informática. También porque desde nuestro punto de vista, entendemos que el usuario que vaya a utilizar el sistema no solamente es una parte clave del mismo sino que en base a su conocimiento y decisiones es como tiene que pensarse un sistema.

El dominio del problema tiene un nivel de complejidad que consideramos alto y que no necesariamente todas las personas comprenden o necesitan comprender. En particular porque es un sistema que maneja varios niveles de información, yendo desde la parte física del hardware (los sensores midiendo datos de la realidad), pasando por un procesamiento y persistencia de los datos en un servidor hasta el contacto final con el sistema a través de sus interfaces. Como la interacción final es únicamente a través de estas interfaces nuestro foco fue en simplificarlo lo máximo posible sin dejar de mostrar la funcionalidad necesaria y prestando importancia al feedback entre el sistema y el usuario (entendiéndose como feedback a la constante visualización de los estados del sistema, como que se realizaron correctamente las operaciones, o presentación de errores, problemas de conexión, etc).

En ese marco es que se desarrollaron dos interfaces de usuario, una para navegadores web y otra para sistema operativo android. Fueron pensadas desde la visión del usuario final, aplicando conceptos de "user experience", que luego comentaremos, y poniendo el foco principalmente en la usabilidad de cada una de las plataformas. En ese sentido desde ambas es posible acceder a la misma funcionalidad, pero con disposición de la información distinta, adaptada a los distintos tamaños de pantallas, pero también manteniendo una línea general de diseño, contenido, conceptos manejados, colores y adaptación del usuario al uso del sistema.

Más adelante en este capítulo ahondaremos en varias aristas que corresponden al análisis de la usabilidad de las interfaces, qué tecnologías se utilizaron, por qué se eligieron esas y no otras, qué ventajas y desventajas se fueron encontrando; cómo se solucionaron algunos inconvenientes ó cuáles son las funcionalidades finales para los usuarios usuario, entre varias cosas más.

3.2.1 El desafío de dos interfaces únicas para un mismo sistema

Antes de comenzar a explicar específicamente la funcionalidad final ofrecida para el usuario y cómo es la interacción con el sistema, nos parece importante hacer un breve repaso sobre cómo fue el proceso del diseño de las interfaces y el uso de diversas tecnologías en nuestro Frontend, que no es la centralidad del problema pero si un paso importante de analizar y explicar.

La decisión de avanzar con la implementación de una doble interfaz fue una cuestión importante, claramente elevaba el nivel de complejidad de la solución, pero la enriquecía en valores claves.

Actualmente el desarrollo web es una de las herramientas más utilizadas, a pesar del gran avance del desarrollo en móviles, sigue predominando mucho el formato web. También hace tiempo que se viene trabajando el concepto de web responsive, que realmente fue y sigue siendo una solución extraordinaria para la adaptación de páginas web a diversos tamaños de pantalla y dispositivos. Pero esta forma de realización, es decir, una interfaz web con un buen trabajo responsive y la posibilidad de usarse desde el celular, no nos convencia del todo. En particular porque nuestro análisis fue desde la usabilidad primero y de la funcionalidad luego. Veíamos que las aplicaciones moviles nativas cumplian con una cuota mucho mayor de calidad (como complemento a la interfaz web) o también en algunos casos hasta reemplazar la web, según el uso, usuario que predomina o el modelo de negocios apuntado, etc.

El diseño de una aplicación móvil requería no solamente un costo mayor de tamaño del sistema sino también de aprendizaje de nuevas tecnologías. En el balance creímos oportuno avanzar con esta solución doble. Como aspectos positivos podemos encontrar claramente una mejor experiencia de usuario desde una aplicación nativa: utiliza las herramientas brindadas desde el teléfono directamente y no a través del navegador, tiene también memoria interna propia y permite tener en ese marco una mejor performance y eficiencia de los recursos.

Hay varias cosas más que el formato móvil nos permitía volar con imaginación y creatividad: tener una experiencia del usuario muy distinta a la web, crear una agilidad de uso y temporalidad de respuesta distinta, como también generar otros contextos en los que se pueda

usar el sistema, es decir, no solamente sentados en un escritorio con una computadora, sino tener la aplicación en todo momento. Cada vez teníamos más ítems a favor para tomar la decisión de avanzar en la aplicación móvil; nos pareció que había muchas herramientas que nos brindaba la natividad de la aplicación. La independencia del navegador hace que se puedan utilizar otros elementos más fácil, como lo son la cámara, el gps, las notificaciones, funciones específicas del teléfono, comunicación entre aplicaciones y muchas cosas más. Si bien en su mayoría no fueron utilizadas en este trabajo, si están planteadas como trabajo futuro en la mejora de estas interfaces. Es muy importante que al comenzar un sistema, más allá que se avance en algunas funcionalidades nomas, la arquitectura debe poder soportar cambios y actualizaciones.

Por último, además de estos elementos que vamos resaltando, el segundo paso fue encontrar las tecnologías que nos permitieron tener una gran reutilización de funcionalidad y diseño, para ser consecuentes en el desarrollo modular que nos propusimos. En el siguiente apartado hablaremos en detalle de las tecnologías.

3.2.2 Una GUI dinámica con Angular JS, HTML5 y Ionic

Desde nuestro punto de vista y como también ocurrió en otras partes del sistema, entendemos como muy importante el momento de definición de las tecnologías con las que se va a trabajar, y con las que se definió en este Frontend en particular, ya que dichas tecnologías están involucradas desde el minuto cero en lo que corresponde a la elicitación de requerimientos. Cuando empezamos a pensar las tecnologías a utilizar, nos preguntamos e investigamos por qué convenía una por sobre otra, cuales eran las mejores para la arquitectura de software que estábamos pensando, cuales lograrían una mejor performance y estabilidad, entre otras cosas.

Como mencionamos en capítulos anteriores, la arquitectura del sistema que planificamos es la de una interfaz de usuario (Frontend) separado del servidor, con la lógica de presentación de datos y de funcionalidades independiente. Esto nos permitió tener dos interfaces distintas pero con una lógica única del mismo servidor, evitando repetir mucha funcionalidad y lograr que sea más escalable, debido a que estos dos espacios funcionan de forma autónoma. Este análisis arquitectónico comenzó con la búsqueda de alguna tecnología que pueda adaptarse y acoplarse a estas necesidades, pero que también pueda ser modular y escalable.

Muchas tecnologías hoy en día cumplen con estos requisitos, no solo en diferentes lenguajes sino en diferentes formatos y diferentes proyectos open source. Es por eso que pusimos en la balanza un elemento clave a la hora de decidir las tecnologías: que sean lo más estables posibles, que veamos en su historia de desarrollo y avance una estabilidad que nos permita hacer una aplicación que dure en el tiempo sin necesidad de andar actualizando por modificaciones externas. Otro de los requerimientos fundamentales que influyó mucho era el de utilizar tecnologías libres, de código abierto, ya que nuestro proyecto se enmarca en ese objetivo, de que sea libre, entonces también lo tenían que ser sus sub-partes.

Nos quedaba por resolver entonces, pensando en estos parámetros, qué tecnologías eran las más apropiadas para construir una aplicación móvil para android y otra web, al mismo tiempo, reutilizando la mayor cantidad de elementos, recursos y tiempo.

El desarrollo web desde hace mucho tiempo que viene creciendo y mejorando sus tecnologías, teniendo como una de sus mejores arquitecturas la MVC (modelo vista controlador), tanto que se ha diseñado desde hace un tiempo ese mismo esquema para el Frontend independiente del servidor. También muchos de los frameworks de desarrollo móvil utilizan este modelo basado en HTML, css y javascript como motor principal de la funcionalidad y aspectos visuales **[11] [12]**. A diferencia de la web, los frameworks para móviles poseen funcionalidad propia para luego convertir la aplicación en nativa de los diversos sistemas operativos móviles: android, IOs, windows phone, entre otros. Esto nos habilitaba una opción muy interesante para elegir alguna de estos frameworks, que aunque no compartan el 100% de las cosas, si compartieran lo fundamental del desarrollo, la lógica para pensar la solución y la funcionalidad.

Dentro de esos framework de desarrollo móvil optamos por IONIC, que es uno de los más utilizados hoy en día **[13]**. Es una herramienta open source y gratuita con la cual se pueden combinar varias tecnologías, entre ellas HTML, css y javascript, para el desarrollo de aplicaciones híbridas. Tiene un funcionamiento y performance muy buena en la aplicación final construida, lo que es muy importante para las aplicaciones móviles.

Además de garantizar todo lo anteriormente mencionado, este framework tiene una herramienta muy valiosa, que es la de un diseño visual de los componentes de las aplicaciones muy simple, atractivo y funcional, parte fundamental de cualquier sistema orientado a usuarios finales.
IONIC tiene predefinidos muchísimos componentes visuales y templates para que no solamente las aplicaciones sean realmente funcionales sino que además se vean muy elegantes.

Dentro de este poderoso framework, el motor de la lógica y del comportamiento esta de la mano de Angular. En la versión 1 de IONIC, se usa Angular JS (que es la utilizada en este trabajo) y en la versión 2 (y en adelante) de IONIC se utiliza la última versión y formato framework de Angular (Angular a secas, sin el JS).

Angular JS **[14] [15]** es una librería de javascript que permite tener la lógica separada del servidor generando modularmente toda la funcionalidad, reutilizando elementos, código y lógica. Si bien hay muchísimas librerías o frameworks de javascript que cumplen con estas condiciones, esta es una de las más importantes, en cuanto a la estabilidad principalmente. Una de las cualidades principales es la de tener un comportamiento reactivo de la parte visual, respondiendo a los cambios del modelo de datos de forma automática, sin tener que recargar la página web o la aplicación. Esto nos permite tener aplicaciones muy dinámicas y con un gran feedback al usuario de todo lo que va ocurriendo en el sistema.

Dentro de este relato y antes de continuar con el análisis de cada funcionalidad en particular, queríamos dedicarle un párrafo al complejo tema de las versiones de las tecnologías, complejo en el sentido del exponencial avance en cantidad y calidad de herramientas de desarrollo en relación a la mantenibilidad y sostenibilidad de los sistemas. Las variables versiones de las diversas tecnologías (y nuevas que van apareciendo también) hacen que se vuelva muy importante el momento de definir la arquitectura, los alcances a corto plazo y la proyección a largo plazo de los sistemas nuevos. Las versiones avanzan de manera exponencial muchas

veces y esos cambios no siempre son simples correcciones de errores o bugs, sino que muchas veces cambian la lógica de funcionamiento por completo, como es el caso entre la versión 1 de angular (Angular JS) y la versión 2 (Angular framework). Entre estas dos no solo cambia el formato de creación de funcionalidad sino que en la versión 2 incluye por defecto más tecnologías como la son Typescript, Sass y más elementos. Otro caso de los que vimos es la evolución de IONIC a lo largo de este breve tiempo, comenzando hace 4 años en su primer versión, hasta la versión actual 3.5.3 **[16].**

3.2.3 Control remoto, programación por fecha y consumo.

Es importante volver a resaltar, como hemos venido haciendo en las diversas partes del documento, que este sistema fue y es pensado para usuarios finales sin necesidad de tener conocimiento de informática, hardware y/o software, por lo tanto la interfaz final no solamente debía ser lo más intuitiva posible sino que funcional. Es por eso que cada una de las funciones disponibles desde este lugar fueron pensadas y armadas en torno a esto.

Aquí explicaremos, brevemente al comienzo y luego con más detalles las funcionalidades más importantes dentro del sistema. Que variantes hemos encontrado dentro de los dos formatos de interfaz (web y móvil) y qué decisiones tomamos en cada caso; muchas de las decisiones de cómo disponer la información y las funciones dependen de los tamaños de pantalla, pero siempre con el objetivo final que en cada una de esas cumpla con lo necesario.

En primera instancia, podemos decir que desde las interfaces de usuario se dividen en las siguientes secciones: inicio, componentes, monitor, tareas, estadísticas y datos históricos. Además el sistema maneja una autenticación de usuarios, tanto para iniciar la sesión como para cerrarla. Inicialmente el sistema empieza esperando el ingreso del usuario a la sesión, para luego ingresar a la sección principal de inicio. A modo explicativo se irán agregando una captura de pantalla de cada una de las secciones.

A continuación de muestran las páginas de login de usuario tanto de la web (figura 3.5) como de la aplicación móvil (figura 3.6). Además, más abajo se muestran las capturas de las páginas de inicio de la web (figura 3.7) y de la aplicación móvil (figura 3.8).

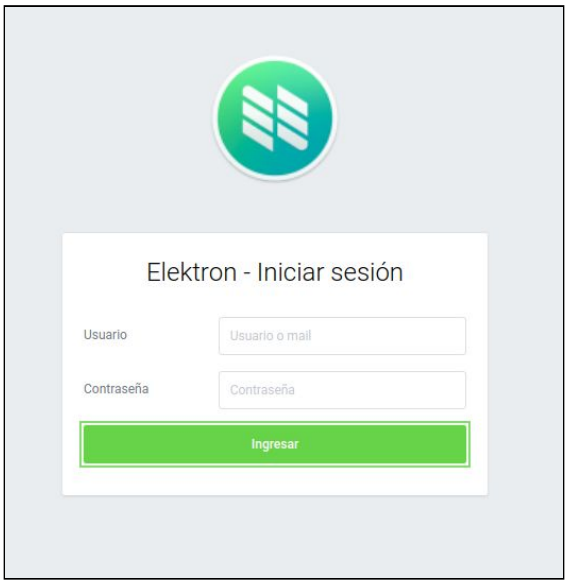


Figura 3.5 - Login de usuario web

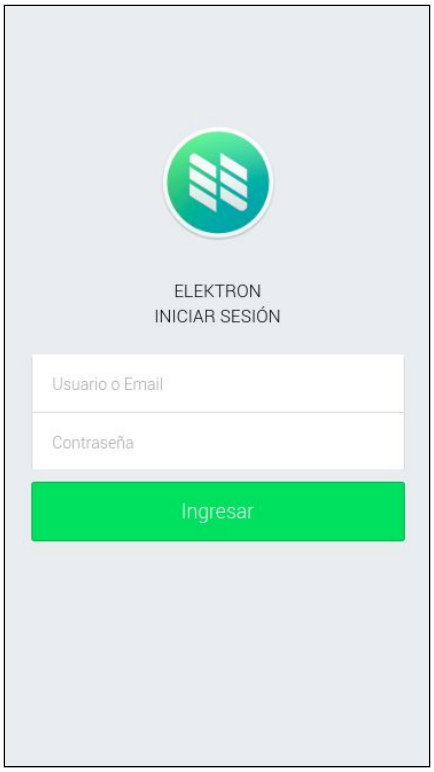


Figura 3.6 - Login de usuario movil

Figura 3.7 - Inicio del usuario web

Figura 3.8 - Inicio del usuario móvil

Dentro de la sección de inicio el usuario tiene un gráfico que representa en tiempo real el consumo eléctrico de alguno de los componentes que estén conectados y con posibilidades de enviar datos (se permitirá elegir cuál de esos componentes vamos a estar visualizando en el momento) así como también ver el estado actual de la conexión en tiempo real. Hay también un botón que permite reestablecer la conexión si se perdió por algún motivo. Volviendo al gráfico, se trata de un gráfico lineal que dispone en el tiempo como fue y es el consumo del componente seleccionado, marcando de forma vertical el consumo en Watts que tuvo y de forma horizontal el momento de la medición. Cuando la medición en tiempo real está activada, se verá un gráfico reactivo que se irá moviendo de derecha a izquierda incorporando las nuevas mediciones desde la derecha.

Además, la sección de inicio cuenta con una serie de bloques de información útil, con la idea de sintetizar cierta información de interés del usuario, como puede ser la cantidad de componentes conectados y desconectados, las tareas programadas y que cantidad se ha consumido, junto a su precio estimado y contaminación en CO_2. Estos últimos dos valores, que fueron explicados en la sección de solución y desarrollo (Capítulo 1.4) brindan una información adicional, que si bien hoy no tienen una precisión tan exacta, creemos que son de gran importancia para el usuario para el uso responsable de energía.

Para destacar también, y como mencionamos antes sobre las diferentes pantallas, la información es la misma ,pero la disposición es distinta. En el caso de la web se puede aprovechar más el espacio a lo ancho, pudiéndose ver el menú en todo momento, cosa que no pasa en el móvil, que para ver el menú hay que apretar en el botón de la izquierda arriba con forma de 3 rayas horizontales.

En la sección de componentes de detalla justamente un listado de todos los componentes ingresados y registrados en el sistema, cuál es su estado actual y que información relevante tienen (nombre, IP y MAC). La versión web se muestra en la figura 3.9.

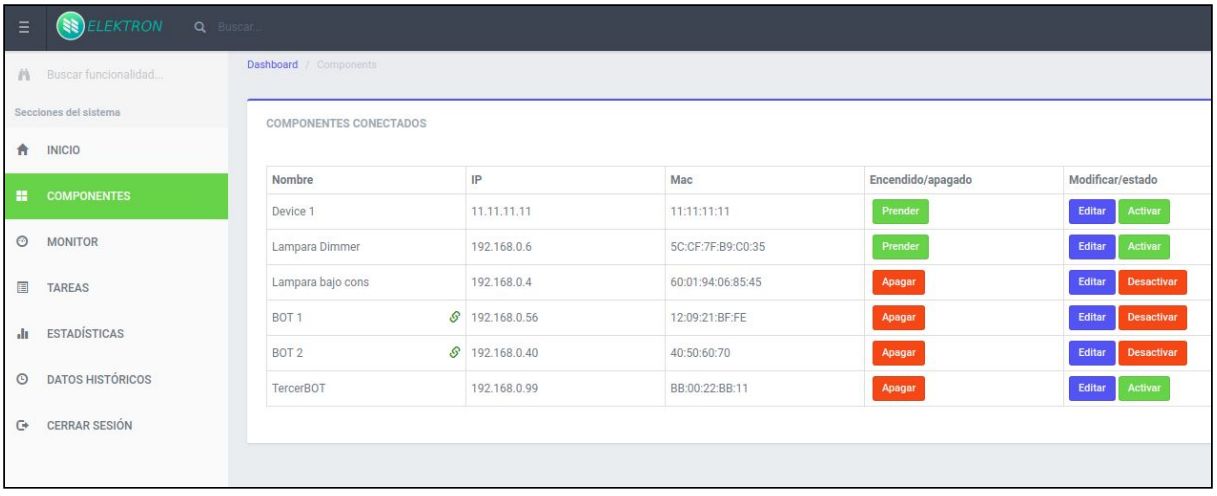


Figura 3.9 - Componentes para usuario web

Dentro de las acciones que pueden realizarse aquí es apagar o encender algún componente, activar o desactivar y también modificar el nombre de los componentes.

En la sección de componentes para la versión móvil (figura 3.10) la realizamos en dos partes, para no saturar de información y poder explayar más el estado de cada componente. En la primera interfaz se muestra un listado general de componentes con su estado actual y luego podemos ver en detalle cada uno, su último consumo, estado interno, etc.

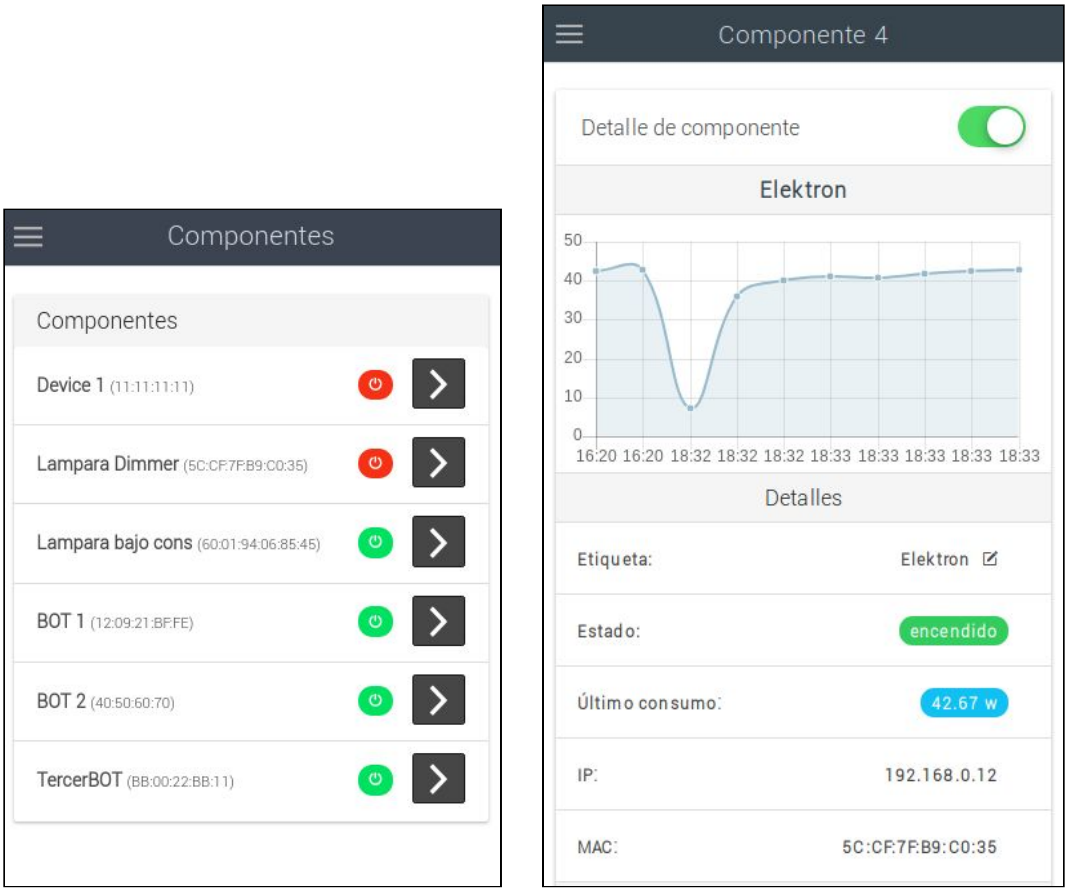


Figura 3.10 - Componentes para usuario móvil

Luego tenemos una sección llamada Monitor, que su objetivo principal es mostrar el consumo en tiempo real de varios componentes en simultáneo. La importancia de tener una sección en donde se pueda ver el consumo de más de un componente a la vez radicó en la posibilidad de comparar dichos consumo en una misma pantalla. Muchos de los análisis que se pueden hacer sobre el comportamiento de consumo de los componentes son comparando consumos con otros de igual condicion, para ver si estan fallando o no, si fueron bien configurados o simplemente para tener una observación de ese estilo por parte del usuario.

Por ejemplo, si estuviéramos midiendo tres heladeras en simultáneo, de marcas distintas pero misma categoría (entendiéndose categoría como el nivel de consumo expresado por los fabricantes) veríamos si son equivalentes, o si tienen diferencias o si su comportamiento es estable o varía mucho.

En este caso la interfaz del gráfico es más simple que en el inicio, en el caso de la web (figura 3.11) muestra los últimos diez datos, y aparece un gráfico lineal por cada componente en condiciones de enviar datos en tiempo real. Además se agrega el estado actual de la conexión, si los componentes han enviado datos o no y también retomar la conexión si esta falla por algún motivo. El diseño móvil se muestra en la figura 3.12

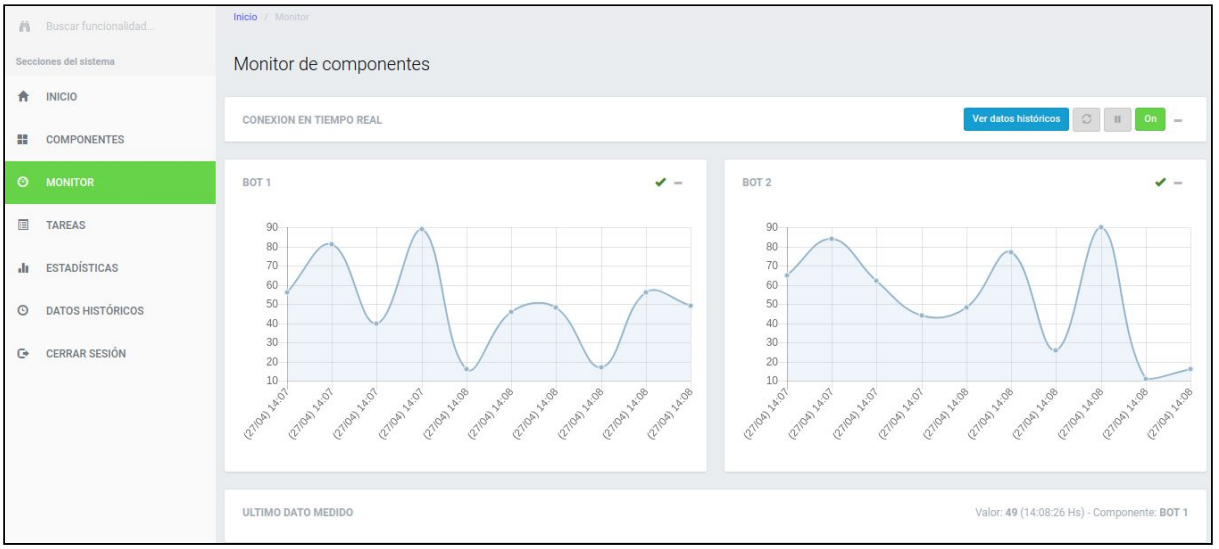


Figura 3.11 - Monitor para usuario web

Figura 3.12 - Monitor para usuario móvil

Una de las funcionalidades que brinda el sistema además de poder accionar sobre cada componente de forma manual (a través de las interfaces) es la de agregar tareas vinculadas a ellos mismos. Estas tareas para este trabajo se redujeron a 2: apagar y encender. Esto fue pensado en base a la idea de la automatización de las actividades diarias, tanto en base a un cronograma temporal con las tareas relacionadas a una fecha y hora, como también las basadas en el funcionamiento de los componentes en base al consumo.

En la sección de tareas (figura 3.13), entonces, tenemos listados los dos posibles tipos de tarea que podemos configurar: por consumo o por fecha/hora. De ambos listados se lista la información importante de las tareas como a qué dispositivo está asociado, acciones y formato de repetición. Además de los datos listados de cada uno están habilitadas algunas funciones relacionadas, como agregar tareas, editar las preexistentes (agregar y editar en figura 3.14) y borrar las que no queremos más.

Figura 3.13 - Tareas para usuario web

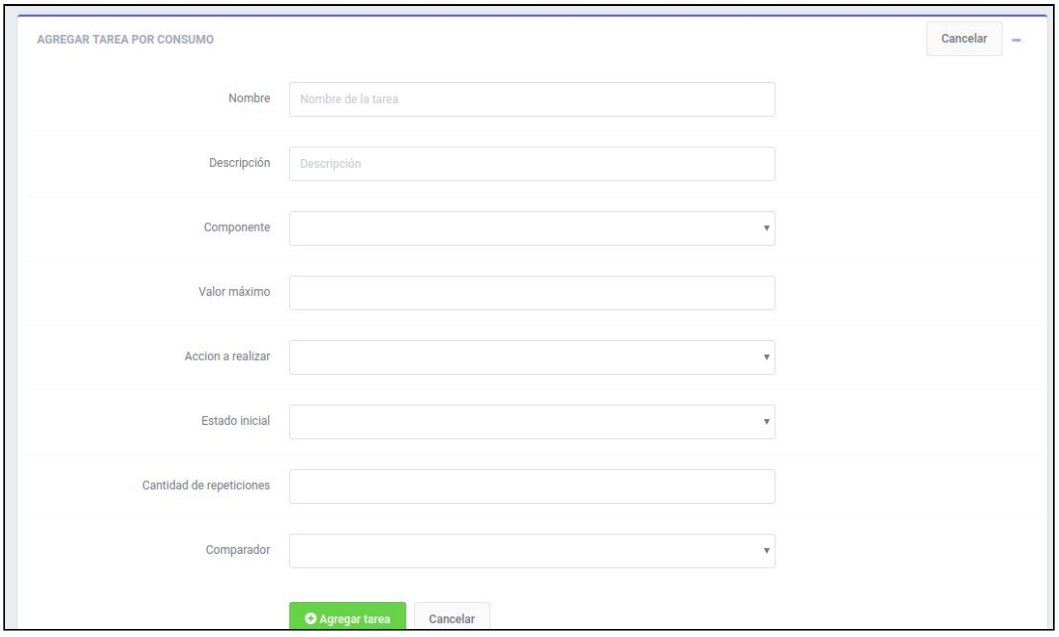


Figura 3.14 - Agregar/editar tareas para usuario web

De las tareas por consumo al crearse se asigna un valor numérico determinado en Watts y un comparativo para realizar la acción, que sea mayor, menor o igual. En caso de que se cumpla la condición se ejecuta la tarea.

Para el caso de las tareas por fecha y hora, se elige un día y una hora determinada que al cumplirse se ejecutará la tarea. Para ambos casos se puede configurar un número de repeticiones que indicaría la cantidad de veces que podrá ejecutarse manteniendo "viva" la tarea hasta que quede sin efecto.

Para el caso de la aplicación móvil se muestran las figuras 3.15 para los listados y la figura 3.16 para agregar o editar tareas.

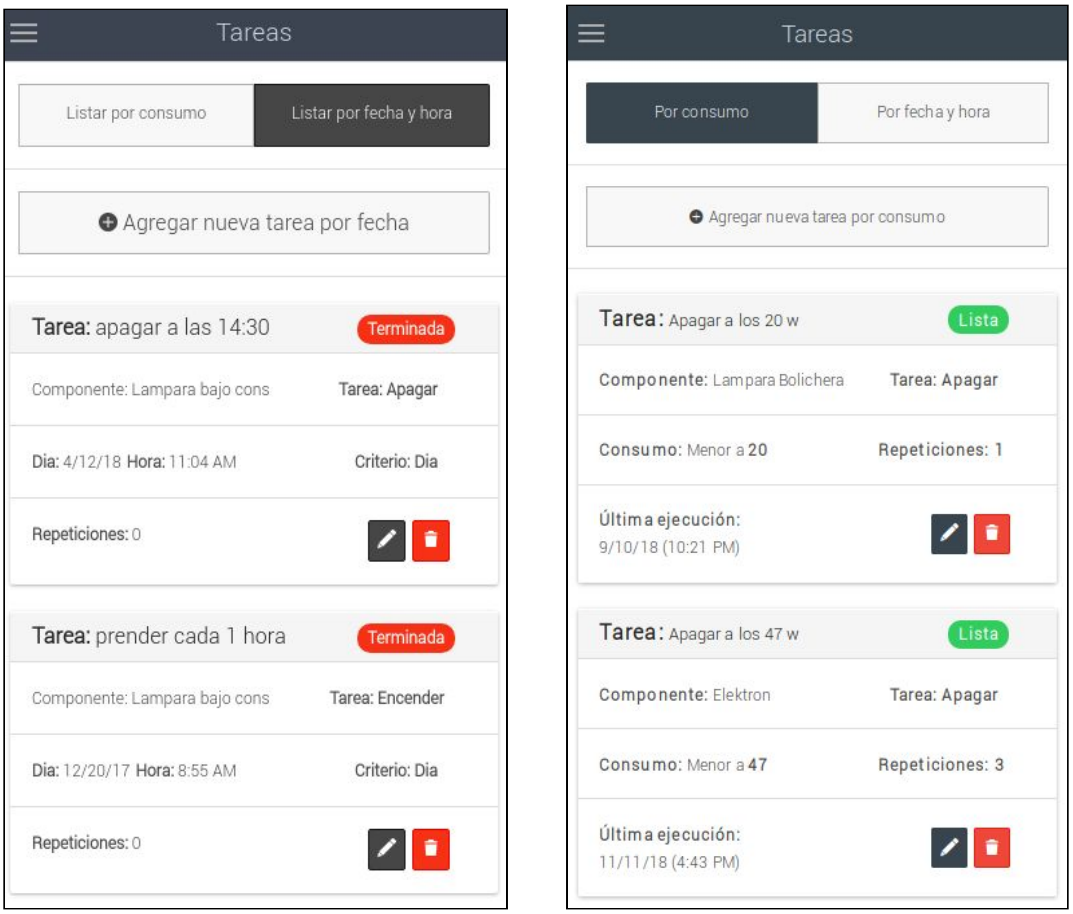


Figura 3.15 - Tareas para usuario móvil

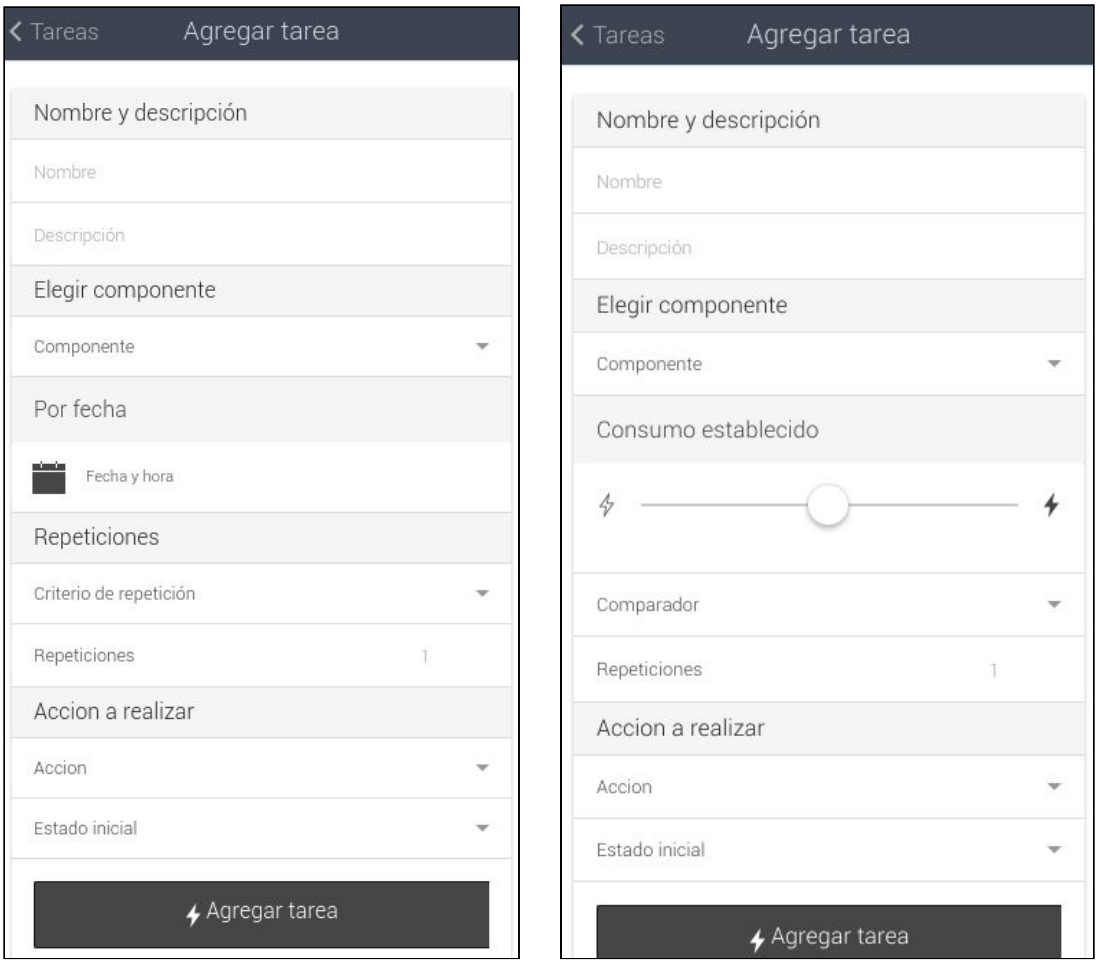


Figura 3.16 - Agregar/editar tareas para usuario móvil.

3.2.4 Menú de estadísticas

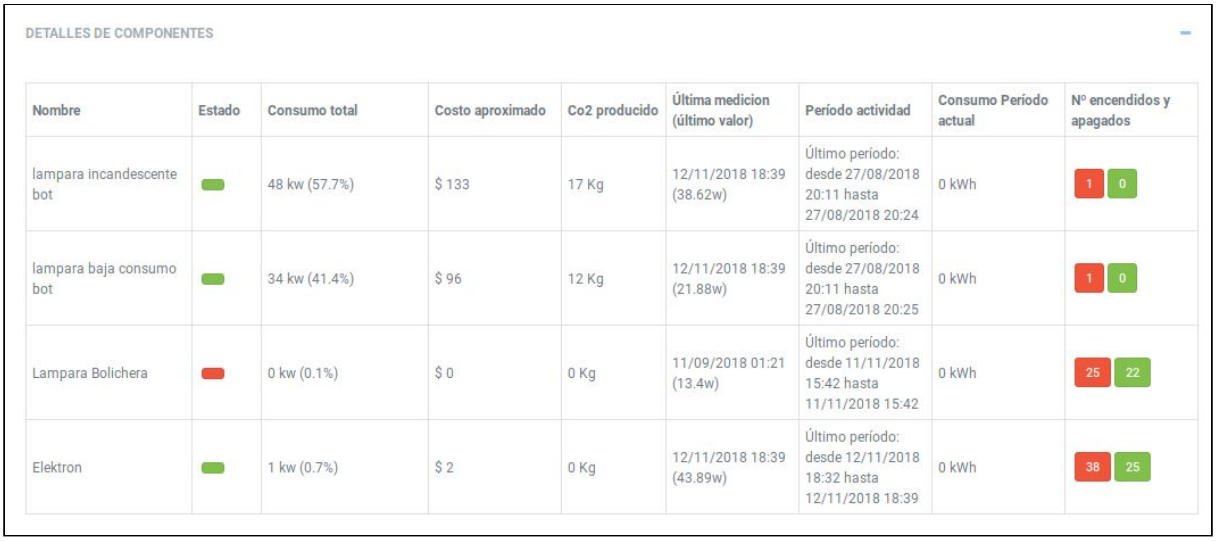
¿Por qué la sección de estadísticas tiene un módulo aparte? En principio porque nos parece una de las partes más importantes del sistema en cuanto al manejo de la información, pero no del manejo de la información per se, sino del procesamiento de la misma, de lograr un resumen, una nueva forma de analizar ese gran volumen de datos que guardamos de cada uno de los componentes sensados.

Al usuario final seguramente le interese ver los datos concretos medidos y hacer una supervisión específica sobre un componente o simplemente tener un seguimiento minucioso de los datos cada corto plazo, pero también lo importante es poder sintetizar toda la información

en datos más concretos, resumidos y que nos permiten tomar decisiones en base al funcionamiento del sistema de forma global.

Tomamos la palabra estadística como eso, una forma sintetizada de información, de presentación en varios formatos, varias formas de verlo. Para mostrar que es importante generar esta información sintetizada lo aclararemos con un ejemplo concreto: un componente con normal funcionamiento guarda un dato de consumo eléctrico cada 5 segundos, lo que significa 12 datos por minuto, 720 datos por hora, 17280 datos por día, y más de 500.000 datos por mes. Es una cantidad de datos enorme que es difícil de mostrar, de visualizar, de entender y analizar. Para observar detenidamente cada uno de estos datos armamos una sección de datos históricos (que se explica en el siguiente apartado), pero nos interesaba esos 500 mil datos puedan ser transformados en otra medida más clara y sencilla de entender por el usuario.

En ese sentido, se pensó un listado resumido de componentes (figuras 3.17 para web y 3.18 para móvil), con una serie de datos estadísticos asociados a cada uno: estado actual, consumo total del componente en relación al total del sistema, costo aproximado y nivel de contaminación en CO_2 asociado a ese consumo, cuál fue la última medición, cuánto duró el último periodo de actividad y su consumo específico. Por último se muestran la cantidad de veces que fue apagado y encendido a través el sistema.


DETALLES DE COMPONENTES

Nombre	Estado	Consumo total	Costo aproximado	Co2 producido	Última medicion (último valor)	Período actividad	Consumo Período actual	Nº encendidos y apagados
lampara incandescente bot		48 kw (57.7%)	$ 133	17 Kg	12/11/2018 18:39 (38.62w)	Último período: desde 27/08/2018 20:11 hasta 27/08/2018 20:24	0 kWh	1 0
lampara baja consumo bot		34 kw (41.4%)	$ 96	12 Kg	12/11/2018 18:39 (21.88w)	Último período: desde 27/08/2018 20:11 hasta 27/08/2018 20:25	0 kWh	1 0
Lampara Bolichera		0 kw (0.1%)	$ 0	0 Kg	11/09/2018 01:21 (13.4w)	Último período: desde 11/11/2018 15:42 hasta 11/11/2018 15:42	0 kWh	25 22
Elektron		1 kw (0.7%)	$ 2	0 Kg	12/11/2018 18:39 (43.89w)	Último período: desde 12/11/2018 18:32 hasta 12/11/2018 18:39	0 kWh	38 25

Figura 3.17 - Estadísticas para usuario web

Figura 3.18 - Listado de estadísticas para usuario móvil

Además de este listado, se pensó en otras dos visualizaciones distintas de la información, en primer lugar un gráfico de torta que muestra qué porcentaje sobre el consumo total representa cada uno de los componentes, con distintos colores y de forma intuitiva para que sea reconocido rápidamente.

Y por otro lado, se hizo otro gráfico, que realiza una comparación de consumo entre dos componentes. Puede el consumo agruparse por dia o por hora. El objetivo es poder comparar los consumos para sacar nuevamente conclusiones que sean necesarias o útiles.

Esto se representa en las figuras 3.19 para web y 3.20 para móvil.

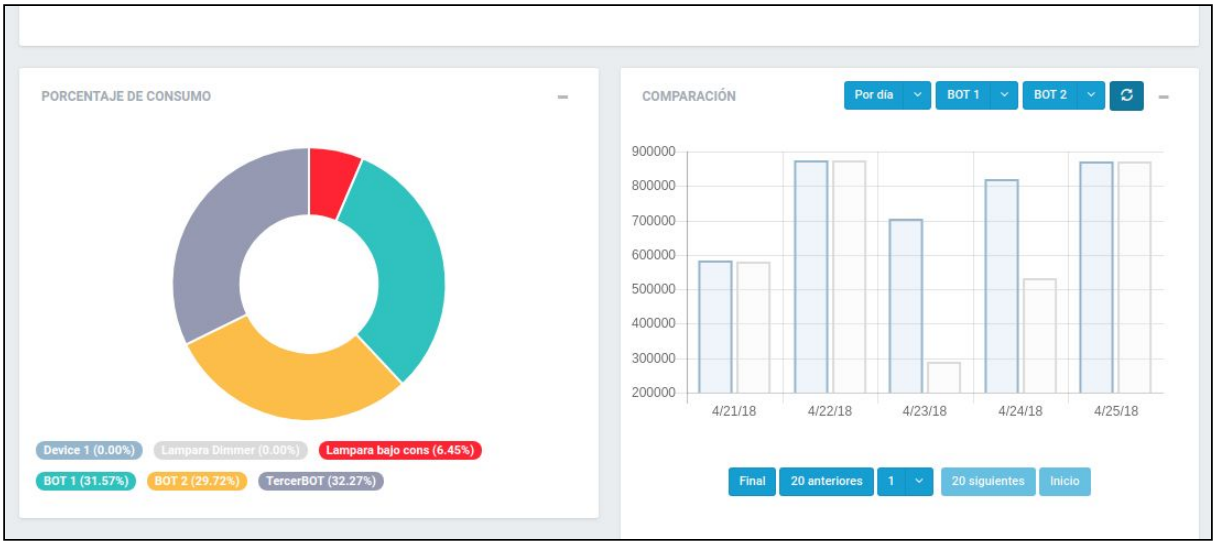


Figura 3.19 - Gráfico de estadísticas para usuario web

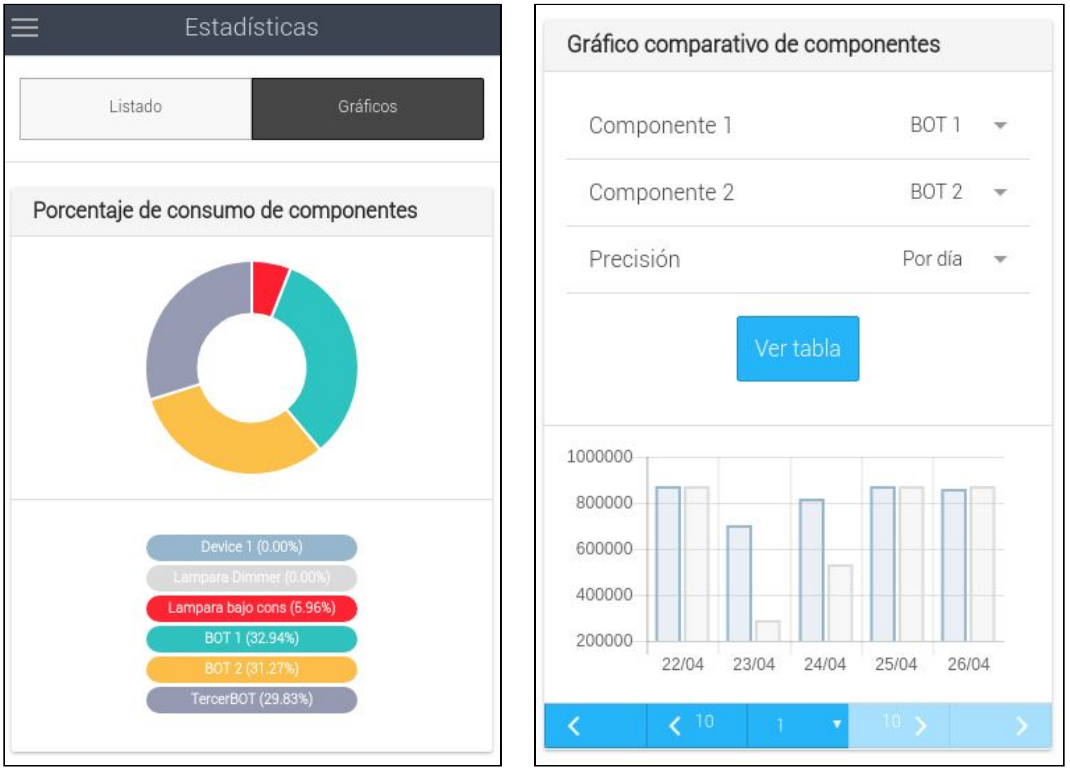


Figura 3.20 - Gráficos de estadísticas para usuario móvil.

3.2.5 Visor de datos históricos

Como decíamos, la medición del consumo genera muchísimos datos, y también explicado en la sección de estadísticas, es importantísimo generar un resumen de toda esa información, pero también creemos que tiene que estar la opción de poder verla más detallada o extendida. En esta sección se verá esto último.

Alineado a esto se pensó en una pantalla exclusivamente para poder ver el consumo a lo largo del tiempo y desglosado por cada uno de los componentes separados. Entonces para poder ver esos datos lo que tenemos que configurar es un componente determinado, un periodo de tiempo y la precisión con la que vamos a divisar estos datos. La precisión, al igual que en la parte del gráfico comparativo de estadísticas, será acumulado por dia o por hora. Este formulario para configurar estos datos aparece al comienzo de la sección (figuras 3.21 y 3.22).

Figura 3.21 - Configurar datos históricos para usuario web

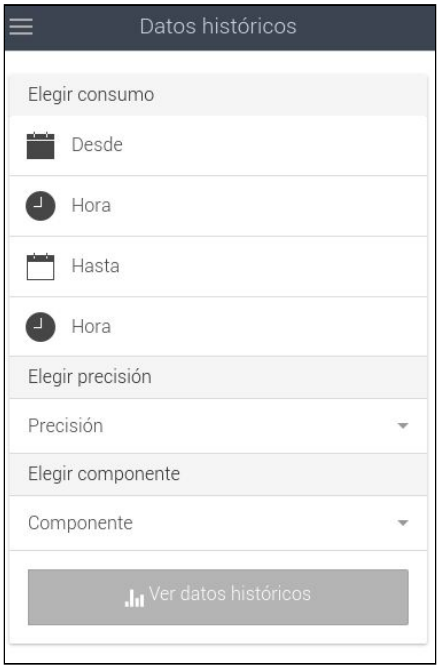


Figura 3.22 - Configurar datos históricos para usuario móvil

Una vez seleccionado el periodo y accionado el botón de ver datos, se incorporarán a la interfaz: un gráfico lineal con la información de consumo en orden cronológico, paginada cada 20 datos en el caso de la web y cada 10 datos en el caso del móvil, y un resumen de información del período (figuras 3.23 y 3.24). El paginado nos permite visualizar toda la información por bloques, permitiendo traer de a poco, yendo para adelante o para atrás en los diversos bloques o dirigirnos a una determinada página. Adicionalmente, en la misma sección se puede ver los datos del período seleccionado, cantidad de páginas y consumo total.

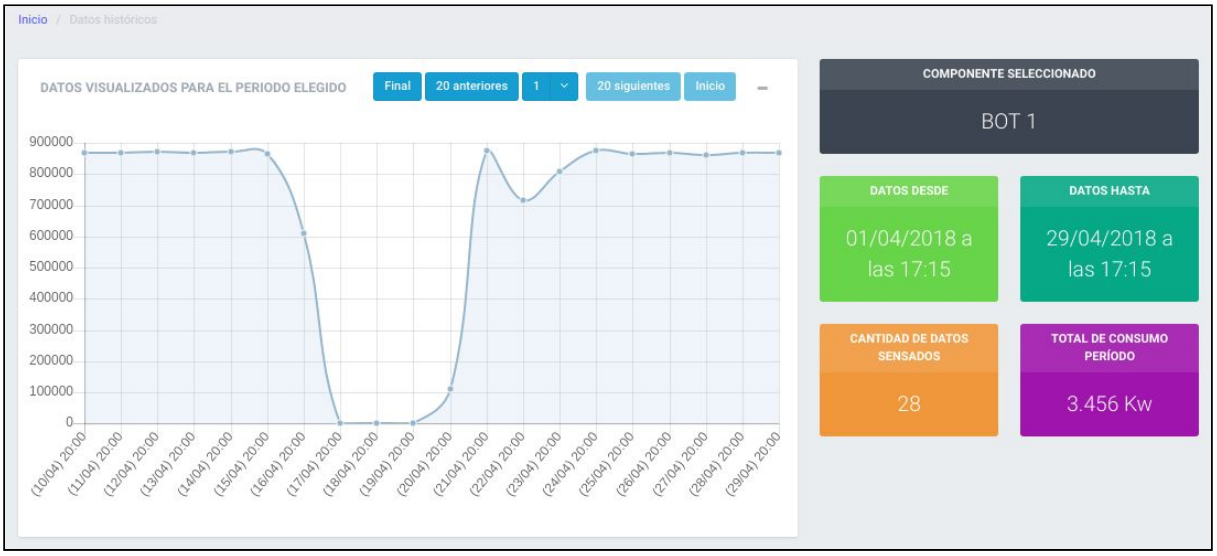


Figura 3.23 - Datos históricos para usuario web.

Figura 3.24 - Datos históricos para usuario móvil.

3.2.6 Manejo de sesión de usuario con el servidor

La seguridad e integridad del sistema es tema fundamental para que funcione correctamente, que no tenga imprevistos y que sea de total dominio del usuario. Para proteger las funcionalidades del sistema se decidió que exista la autenticación por parte del usuario con una clave y un usuario, limitando a que solamente él puede realizar las operaciones, cada una de las operaciones será protegida.

En los sistemas que involucran el manejo de Internet en las cosas domésticas (heladera, lavarropas, etc) la seguridad es fundamental. Se hizo un fuerte hincapié en esto ya que al estar los componentes conectados a Internet y teniendo una IP que podría ser accedida desde cualquier parte del mundo, hay que tomar recaudos necesarios a nivel de seguridad. Además, el identificador único permite que cada usuario trabaje solamente con sus datos, mediciones y componentes conectados al sistema y no pueda acceder a la información o tareas de otro usuario.

La interfaz del login de usuario es sencilla, solicitando el usuario ó email y la contraseña para poder ingresar. En la sección 3.2.3 están las interfaces correspondientes al inicio de sesión representadas por las figuras 3.4 y 3.5.

El objetivo de esta parte no es entorpecer el uso del sistema para los usuarios, sino establecer mínimos recaudos de seguridad. Para la facilidad de uso del sistema se pensaron otros elementos que se mencionan más adelante sobre la usabilidad con interfaces sencillas y el mantenimiento de la sesión hasta que el usuario fuerce la salida mediante la opción “cerrar sesión”. Esto facilita que tenga una interacción más fluida, pudiendo acceder a todas la funciones habiendo iniciado sesión la primera vez.

Técnicamente, la sesión se maneja tanto en las distintas interfaces (web y móvil) como en el servidor. Del lado del cliente se controla con Angular (javascript) ya que el Frontend está desacoplado del servidor y es importante que maneje su propia validación de sesión. En este lugar es donde primeramente se controla los datos ingresados por el usuario, para luego pasar a la validación en el servidor (donde están las cuentas y usuarios registrados). Es importante que tanto las interfaces de usuario como el servidor manejen la misma información de sesiones activas como de integridad de usuarios, sino tendríamos un problema grave. El servidor recibe los datos del usuario y los valida en su base de datos, para luego devolver la autenticación en caso de ser los datos correctos. SI no son correctos devuelve un error y no ingresa a la funcionalidad.

Para toda la funcionalidad siguiente, no solo la verifica siempre en el Frontend de angular sino en cada una de las peticiones que llegan al servidor, recibiendo un token en esa petición que representa la sesión iniciada previamente.

A través del siguiente diagrama (figura 3.25) se muestra concretamente como se realizaria el inicio de sesión y validación pertinente del usuario.

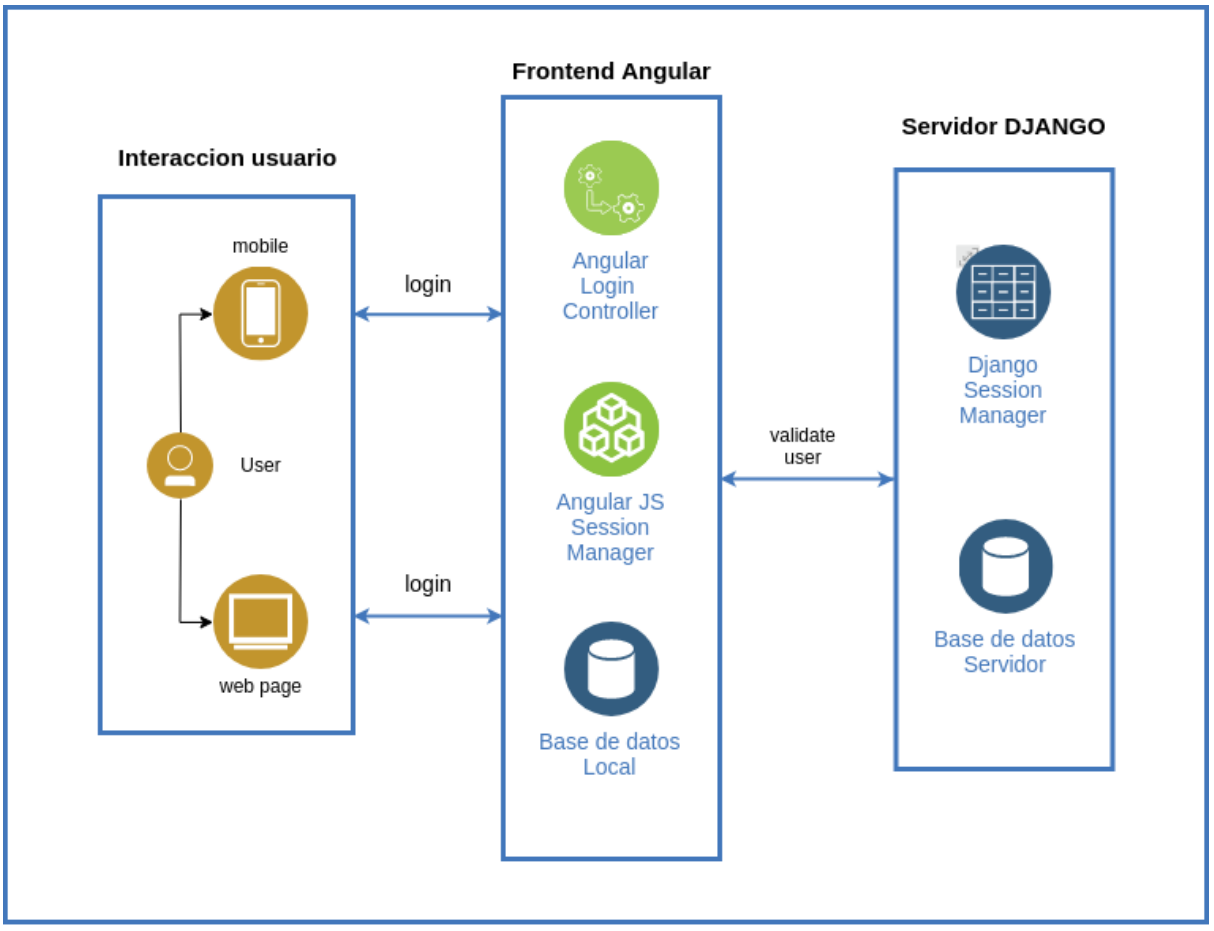


Figura 3.25 - Diagrama de inicio de sesión.

3.2.7 Conexión con el servidor y manejo de errores

La conexión REST con el servidor nos permite tener peticiones muy simples para traer la información: a través de diversas urls y pasándole los parámetros correspondientes, resolvemos todas las actividades y funcionalidades. Esto desacopla mucho al servidor de procesamiento, pero tienen que resolverse en el Frontend, y el lugar es en los controladores de angular JS.

Cada una de las secciones de la interfaz tiene su propio controlador, encargado de administrar la lógica de esa sección. En todos los casos realiza mayores o menores solicitudes de información al servidor, para luego disponerla visualmente dentro de la vista. Además, en algunos casos envía pedidos de modificación o de agregar datos al servidor como por ejemplo el caso de agregar, editar o borrar componentes.

Otro tema importante es la consistencia de la información, es decir, que tanto en la vista del sistema como en el servidor esté dispuesta la misma información y no se contradiga, sino podría generar un colapso en el funcionamiento. De esta forma es que se pensaron también los controladores, cada vez que desde la interfaz se modifica alguna información, se envían dichos cambios al servidor y se modifica la vista actual, permitiendo mantener la consistencia.

Además, cabe mencionar que la complejidad de la arquitectura REST también nos permite la posibilidad de de que esta utilizada por muchas interfaces y/o sistemas sin que estos se bloqueen o dependen necesariamente del servidor. Se podría estar utilizando el sistema en varias interfaces y aun así funcionará correctamente.

Por último y no por eso menos importante, hablaremos de la conexión en tiempo real. Es una de las características principales del sistema la posibilidad de mostrar el consumo que se va midiendo en cada uno de los componentes en el momento que ocurre o tal vez unos segundos después. En la sección de Backend se habló de los demonios de websocket, herramienta utilizada justamente para realizar esta conexión en tiempo real. Desde el Frontend nuestro objetivo era poder capturar cada uno de esos datos que nos iban llegando para poder mostrarlos en los gráficos. Para esto se combinaron las funciones de recepción de mensajes con las de actualización de los gráficos reactivos de Charts.js. **[17]**

En nuestro caso teníamos dos lugares donde disponer esta información en tiempo real: en el inicio y en el monitor. En ambos habían elementos comunes, como la creación de la conexión en tiempo real (se crea siempre desde el Frontend y nos colocamos como "clientes" de los demonios websocket) y la recepción de los mensajes. Pero la diferencia radica en la modificación de la vista en ambos casos: en el caso del inicio, solamente tenemos un gráfico, entonces de los paquetes entrantes filtramos solamente los del componente seleccionado en el gráfico; y para el caso del monitor, al tener un gráfico por componente en condiciones de enviar información, al recibir un mensaje, a través de la mac del dato buscará el gráfico a modificar.

El trabajo con la herramienta de websocket permitió que que esta funcionalidad sea real, que pueda resolverse, sin embargo tuvo varias complicaciones en el camino del desarrollo, cuestión que pasaremos a explicar más adelante en el apartado de problemáticas en el Frontend.

Pasando en limpio, las libertades que nos brinda esta arquitectura de tener desacoplados los módulos, también nos da la responsabilidad de que esas secciones tienen que acoplarse de una forma correcta, con buen manejo de errores y consistencia de datos. De todas formas, las tecnologías utilizadas cuentan con herramientas para hacer estas combinaciones, logrando el funcionamiento fluido buscado.

3.2.8 Amigable al Usuario

3.5.8.1 - Fundamentos

A lo largo del documento y al comienzo de este capítulo se mencionó la importancia de las interfaces de usuario, pensadas desde la visión de un usuario final, que probablemente no tenga muchos conocimientos técnicos y que además se le facilite lo máximo posible el uso de la herramienta. En esta sección se detallarán cada una de las decisiones que surgieron desde el lugar de la usabilidad, analizando teóricamente los aspectos importantes y también concretamente cómo se implementan en las diferentes interfaces.

El objetivo de este proyecto a nivel interfaz no fue buscar o atraer usuarios de la web o del mercado de aplicaciones móviles, sino todo lo contrario, los usuarios serán los que utilicen el sistema integral. Esto determinó varios elementos que se detallaran a continuación.

A nivel más teórico, y luego volcado en lo específico de las diversas interfaces, se utilizaron principios de usabilidad planteados por los autores Steve Krug **[18]** y Jakob Nielsen **[19]**, así como también entraron en análisis algunos conceptos más.

El usuario de estas aplicaciones fue lo primero que analizamos. En primer lugar se puede decir que el conocimiento técnico del usuario podría ser variado ya que el sistema está enfocado en parte en la domótica y cualquier persona que quisiera utilizar el sistema debería estar capacitada para hacerlo. Luego hablaremos sobre esto.

En segundo lugar, el usuario vendrá desde un lugar más endógeno, esto quiere decir que las interfaces son parte de un sistema más grande interconectado. No serán interfaces que estarán expuestas a la web en su totalidad, sino que la mayoría de las funciones estarán bajo una autenticación de usuario y no de acceso libre como lo es un diario o una página de una librería. Esto determina varias vertientes de cómo se construirá la aplicación y sus interfaces: en principio el objetivo no va a ser atraer mediante buenas prácticas nuevos usuarios sino que con esas buenas prácticas sea más fácil de usar el sistema.

Por último, en este primer acercamiento al usuario del sistema, podemos decir que la confianza en la aplicación es algo crucial. No en vano se menciona esto ya que toda el área de domótica está en contacto con elementos físicos de la realidad que implican una responsabilidad de uso y dudas sobre cómo se comporta la tecnología. Es por esto que la confianza fue un factor clave y como uno de los contactos directos que tiene el usuario con el sistema es a través de la aplicación, era muy necesario que la interfaz sea consistente con lo que se ve y lo que se muestra, que tenga un buen feedback de errores, que haya un constante reflejo de las actividades y fundamentalmente la asistencia y simplificación en las acciones a realizar.

De los dos autores trabajados, Krug definitivamente es uno de los que simplifica más la experiencia de los usuarios o la teoría de la usabilidad. La simplifica no en el sentido de no hablar cosas importantes, sino todo lo contrario, de hablar en términos claros y concretos de cómo crear sitios web (y naturalmente puede ser expandido a aplicaciones móviles) fáciles de usar y aprender, que generen la confianza necesaria para que vuelvan a interactuar los mismos usuarios y sean de una naturaleza minimalista y con la información justa y necesaria.

Entonces, siguiendo por esta línea, fue fundamental que las aplicaciones tengan un funcionamiento adecuado, que en todo momento el usuario sepa lo que está pasando y si hay algún inconveniente, se entere también. Esto hará naturalmente que no se frustre al usarlas y haya un vínculo de confianza fundamental.

La simpleza en la disposición de la información así como también la rapidez para resolver las actividades hicieron que su uso sea mucho más ameno. La simpleza se expresa en disponer en cada una de las interfaces lo que tenga que tener y nada más, y que los rótulos que tenga cada una de las secciones sean bien representativos.

La navegación del sistema la pensamos concisa y fácil de ubicar, teniendo en todo momento referencia a donde estamos parados dentro del sitio, identificados los retornos a la página principal o a las navegaciones anteriores anteriores.

Un elemento interesante tenido en cuenta, pero que no se ha aplicado en este trabajo, pero se utiliza cuando hay muchas funcionalidades para usar es el de una opción de "buscador" dentro del sistema. Permite refrescar a un usuario confundido donde se podrá encontrar cierto elemento o función y es sumamente valioso, pero en este caso al no ser tantos los elementos que componen la interfaz, no fue tan necesario. En sitios donde no solo abunda la información, las secciones y el dinamismo de dicha información la búsqueda se volvería casi obligatoria.

3.2.8.2 - Sobre las tecnologías y la usabilidad

En el caso de la librería Angular JS (usada en ambas interfaces) tiene algunos temas interesantes a analizar a nivel usabilidad. Lo primero y más importante es que la disposición de la información es muy dinámica y reactiva, es decir, sin tener que recargar una página es posible actualizar la información que haya variado en el servidor y/o agregar nuevos elementos html a la página. Esto parece algo muy cotidiano de las páginas web actuales, pero que no deja de requerir especial atención y análisis: uno de los elementos fundamentales de Elektron fue la disposición en tiempo real de los datos que se vayan sensando de consumo eléctrico en gráficos en la interfaz. Esto hubiese sido posible si la tecnología utilizada no acompañaba estos requerimientos, en particular utilizando el protocolo Websocket se logró esto de forma satisfactoria y eficiente.

Otro de los puntos claves a analizar de la tecnología Angular JS es que nos permitió páginas muy dinámicas, no solo para completar sus funciones, sino para mostrar advertencias, errores, mensajes del servidor o simplemente realizar varias acciones en simultáneo. Por ejemplo, en una de las interfaces podría estar mostrando en tiempo real el consumo de energía y permitir la edición de los elementos que se están midiendo.

Una de las tecnologías fundamentales en este desarrollo fue el framework Frontend Bootstrap ya que nos permitió generar diseños no sólo agradables a la vista sino con una capacidad enorme de adaptarse a distintos tamaños de pantalla bajo un mismo diseño. Tablas, gráficos e imágenes que se adaptan al tamaño de la pantalla y bloques de contenido dentro de la página que se acomodan dependiendo del mediaquery que estén aplicando.

Para el diseño móvil, el framework IONIC cuenta con muchas herramientas aplicables a la usabilidad. En principio está conformado por una serie de componentes pre-armados para la utilización directa en las aplicaciones. Esto nos ayudó muchísimo para que el diseño sea uniforme y simple. Prioriza mucho el armado de apps minimalistas, con colores estables y bien combinados entre color de letra, fondo y bloques de contenido, con orientación al feedback del usuario con notificaciones, manejo de spinners (la ruedita que gira demostrando que esta cargando algo) y más cosas.

Por último sobre el framework ionic, tiene como base del proyecto una estructura fija para el menú y las distintas secciones que configuremos. Esto ayudó mucho a que el usuario se ubique fácilmente dentro del sistema y pueda salir o regresar rápidamente.

3.2.9 - Análisis detallado sobre los conceptos de usabilidad en ambas interfaces

Consistencia en las interfaces

Uno de los elementos fundamentales que tuvimos en cuenta fue la consistencia en la disposición de los elementos y de las secciones de las aplicaciones mobile y web. En particular la disposición del menú (figuras 3.26 y 3.27), las secciones que existen, colores y símbolos utilizados y el orden en el que aparecen.

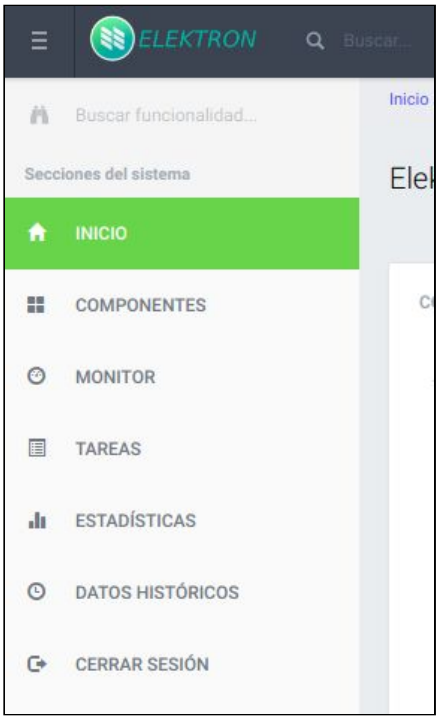


Figura 3.26 - Menú para usuario web

Figura 3.27 - Menú para usuario móvil

Colores de íconos demostrativos

Los colores de los iconos o botones los pensamos consistentes con la función que realizan. En general los colores verdes se utilizan para aceptar o agregar y los botones rojos para borrar o cancelar. En nuestro caso, en la app móvil se usó el icono en color verde para indicar que está encendido y rojo para cuando esta apagado; para el caso de la web, los botones encender y apagar de colores verde y rojo (figuras 3.28 y 3.29).

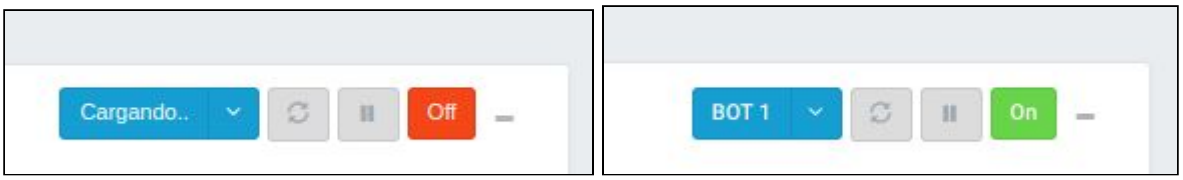


Figura 3.28 - Botones de conexión para usuario web.

Figura 3.29 - Botones de conexión para usuario móvil.

Iconos consistentes

Además se propuso la utilización de iconos consistentes revitalizando el significado para el cual fueron creados, lo cual es fundamental para el entendimiento del sitio por parte de los usuarios. Además, ayuda para el reconocimiento de las secciones. Como en varios de los casos explicados, también la consistencia de los iconos es entre las dos interfaces (figura 3.30).

Figura 3.30 - Iconos para usuario móvil (izquierda) y web (derecha)

Otro caso interesante para hablar es sobre los iconos es en las pantallas donde se muestra la conexión en tiempo real (Monitor e Inicio). Se puede observar en las figuras 3.11 y 3.12 para el caso del monitor y en las figuras 3.7 y 3.8 del inicio. En este caso se utilizaron varios iconos para trabajar la usabilidad del usuario. Por un lado se añadió la posibilidad de pausar y despausar la conexión, con iconos demostrativos. Y por otro lado, para recuperar la conexión websocket en caso que falle, el icono de recargar es demostrativo también.

Icono del menú desplegable estándar

Es importante el reconocimiento de algunos iconos fundamentales, aunque parezca que ya es común verlos en la mayoría de las aplicaciones o sitios, como el de la configuración con la engranaje y el del menú con las tres rayas (también llamado icono “hamburguesa”). Además de la posición que debe ser ubicado (generalmente a la izquierda arriba), se respetó el estándar de que el menú desplegable se utilice con el icono correspondiente (figura 3.31).

Figura 3.31 - Icono de menú desplegable para usuario móvil (izq) y web (der)

Prevención de errores en formularios

Es importante tratar de reducir todo lo que se pueda el margen de error que pueda cometer el usuario. En el caso de los formularios, se usó un selector de fechas en vez de ingresarla a mano, se usaron campos con opciones predefinidas y se armaron avisos de detalles con algún mensaje especial. Además, en los formularios para guardar o agregar una tarea, verifica si faltan datos, detalle que mostraremos en las siguientes imágenes.

La prevención de errores limitando las posibilidades y margen de ingresar datos no válidos o incompletos al sistema es una herramienta fundamental desde varios puntos de vista. En principio uno de los objetivos es que los datos que lleguen al servidor en este caso sean siempre válidos (exceptuando casos donde se vulneren en el camino de la petición http en la red). Otro elemento clave es que el usuario tiene menos dificultades al ingresar o elegir entre determinados valores, permitiendo que el aprendizaje en el sistema y la memoria que tenga que tener sean simplificados en gran medida. En última instancia, también coopera en explicar correctamente cuales son los pasos necesarios para resolver las funcionalidades del sistema, es decir, para poder crear una tarea nueva o buscar los datos históricos de un componente.

En las siguientes imágenes (Figura 3.32 a Figura 3.34) observaremos algunos de los elementos para evitar errores por parte del usuario, como fue la decisión de no habilitar el botón de ver datos hasta completar todos los campos, y ayuda para completar los datos complejos, en este caso la fecha y hora.

Nos parece importante aclarar que en los casos de los seleccionadores de fecha y hora aparecen algunas palabras en inglés. No es lo más apropiado a nivel usabilidad mezclar idiomas, pero en este caso la tecnología fue un limitante para poder configurar estos valores en idioma español.

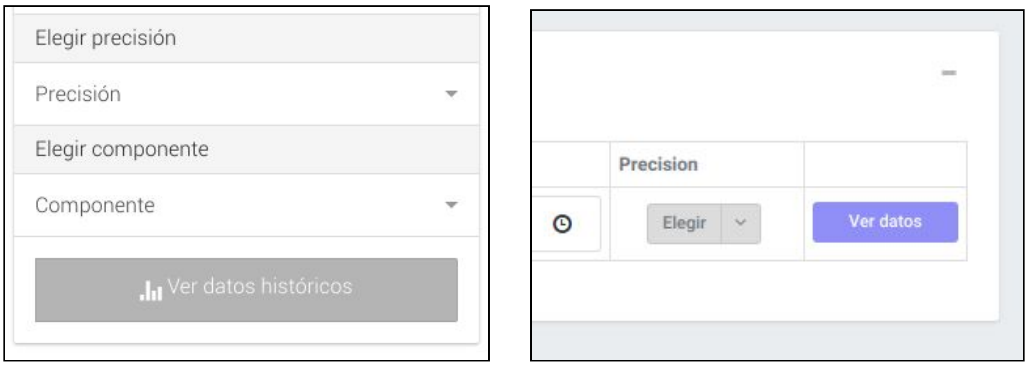


Figura 3.32 - Botón bloqueado para usuario móvil (izq) y web (der)

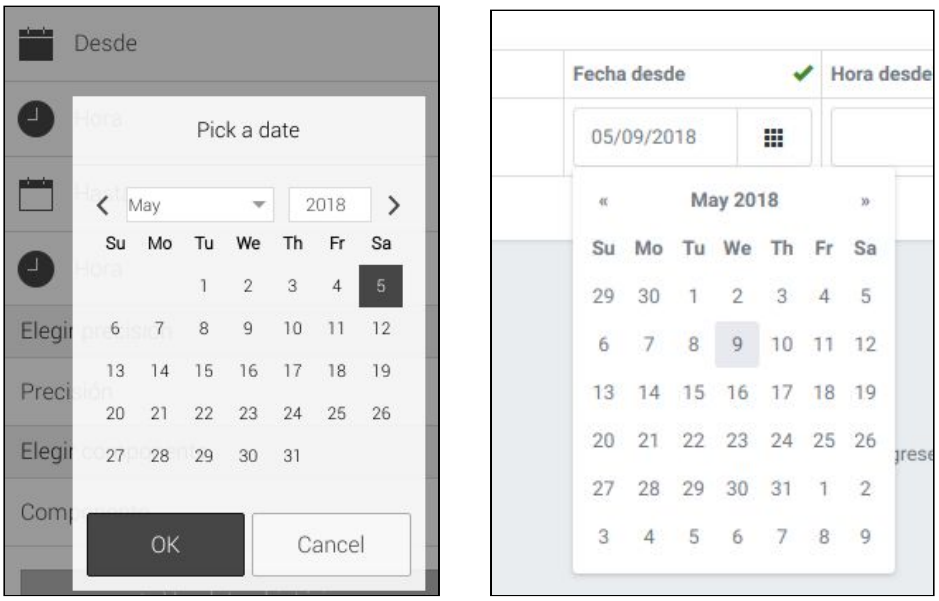


Figura 3.33 - Seleccionador de fecha para usuario móvil (izq) y web (der)

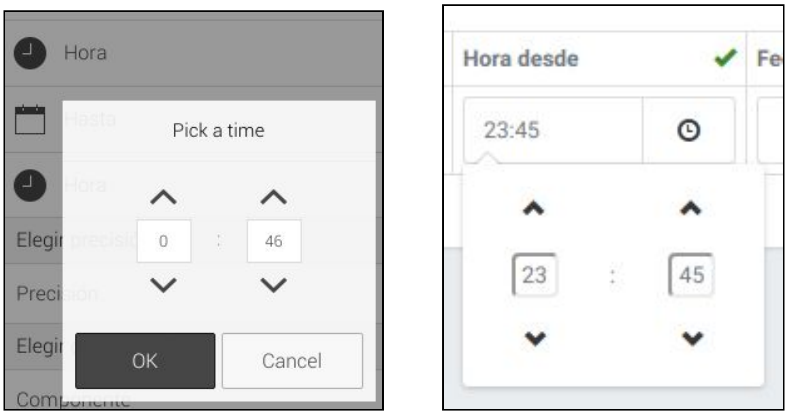


Figura 3.34 - Seleccionador de hora para usuario móvil (izq) y web (der)

Comportamiento reactivo y dinámico de la página

Esto puede ser analizado desde varios aspectos: en primer lugar, la transmisión en tiempo real de los datos de consumo necesitaba que la aplicación se muestre “viva”, es decir, que se vaya modificando la página a medida que aparezcan datos nuevos. En ese sentido se pensó también la funcionalidad de parar y activar la conexión.

Para no repetir la incorporación de imágenes muy similares a anteriores, esto se puede observar en las figuras 3.7 y 3.8 para el inicio y las figuras 3.11 y 3.12 para el monitor. Estos son los lugares donde se observa el dinamismo del tiempo real de los datos.

En segundo lugar, otro elemento para resaltar el dinamismo de la página fue agregar los últimos datos que llegan a través de websocket a un cuadro en las pantallas donde exista dicha conexión y permita visualizar siempre el último dato que llegó .

Figura 3.35 - Gráfico reactivo y últimos datos para usuario web.

Por tercer y último lugar, la espera del usuario mientras se carga una página, se realiza alguna modificación o simplemente tiene que esperar algún cambio es importante que sepa que está pasando algo, que se esta resolviendo ese problema. Esos momentos se decidió simular la espera con una de las herramientas más utilizadas hoy en día son los spinners (rueditas que van girando infinitamente). Estos spinners se cargan cuando comienza una operación que puede demorar unos segundos y cuando finaliza desaparecen (figuras 3.36 y 3.37).

Figura 3.36 - Icono loading para usuario web.

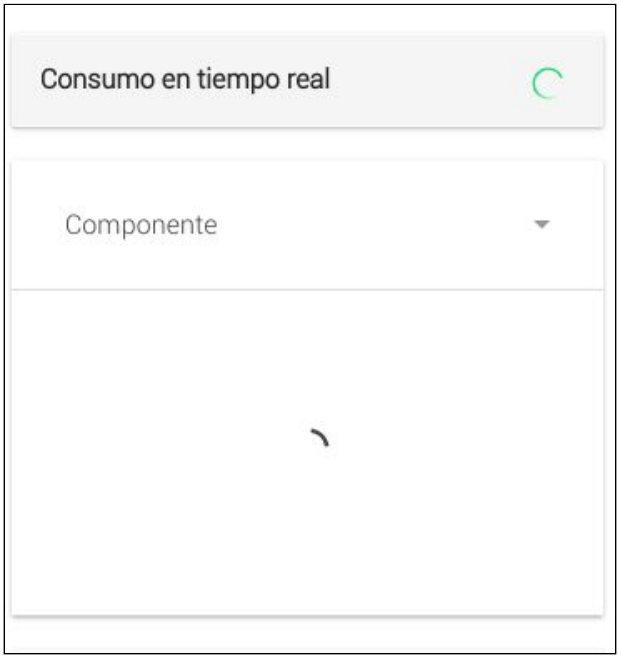


Figura 3.37 - Icono loading para usuario móvil.

Información adicional explicativa.

En este caso se analizó en los gráficos cómo aclarar la información que se dispone. El usuario tiene que tener la oportunidad de solicitar información de algo que no conoce del todo o quiere saber más detalles. Esto se desarrolló tanto para el caso de los gráficos de consumo en tiempo real y por hora/dia como el caso del gráfico de tortas, donde se representa el porcentaje de consumo de cada componente con respecto al total (figuras 3.38 y 3.39).

Figura 3.38 - Detalles adicionales en gráficos para usuario web.

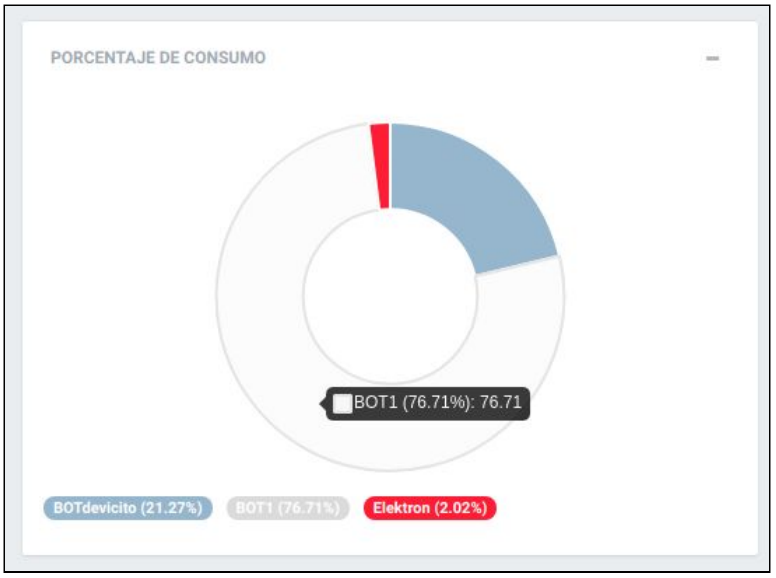


Figura 3.39 - Detalles adicionales en gráficos para usuario web.

Diseño minimalista y con la información justa y necesaria.

Es importante que las interfaces no estén sobrecargadas de información y que sean lo más simples posibles, sin dejar de lado la completitud de las diversas funcionalidades que provee el sistema. En este caso en el Inicio de la aplicación, se diseñó un gráfico representando la transmisión en vivo de datos y un resumen de los datos datos del sistema. Esta sencillez se ve reflejada en todas las interfaces, donde se agregó la información justa y necesaria para cumplir con cada una de las secciones..

Referencia a la navegación.

En todo momento el usuario tiene que saber dónde está parado dentro de la aplicación, qué ruta lo llevó hasta ahí, qué camino recorrió. Por lo tanto, se agregó esa referencia clara en cada una de las pantallas (figuras 3.40 y 3.41).

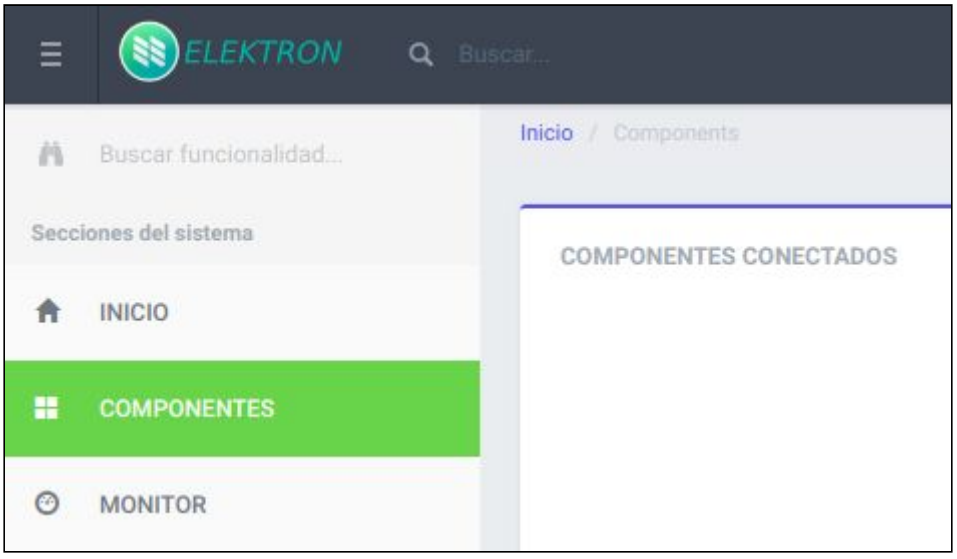


Figura 3.40 - Contexto de navegación para usuario web.

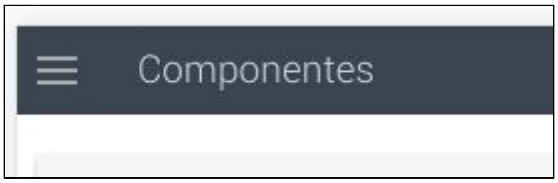


Figura 3.41 - Contexto de navegación para usuario móvil.

Manejo y aviso de errores.

El usuario debe tener un feedback de los errores que comete o que ocurren en el sistema. Esto es fundamental para la confianza en cada una de las funciones que pueda realizar. En este caso se avisó cuando confunde la contraseña, también en los casos donde se pierde la señal o conexión con algún aparato, entre otros casos (figuras 3.42 a 3.45).

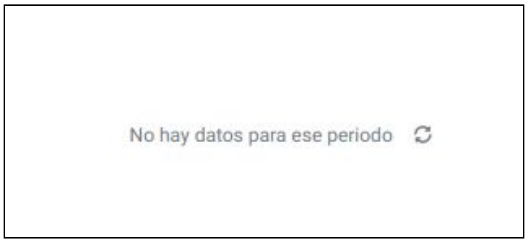


Figura 3.42 - Mensaje de aviso cuando no hay datos.

Figura 3.43 - Notificación de aviso cuando no hay datos.

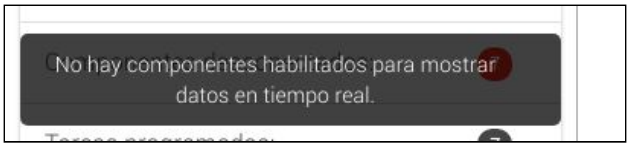


Figura 3.44 - Notificación de aviso cuando no hay componentes para mostrar información en tiempo real.

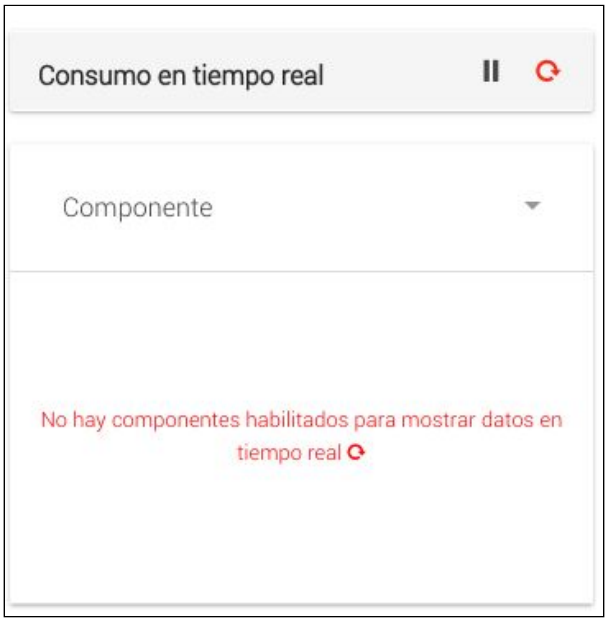


Figura 3.45 - Mensaje de aviso que no hay componentes para mostrar datos en tiempo real.

3.3 Problemas y Soluciones.

3.3.1 Interbloqueos

Como se mencionó anteriormente, los dispositivos obtienen datos por milésima de segundo del sensor de corriente, realizan el cálculo de la potencia y cada cinco segundos envían el dato de la energía acumulada al servidor. Al mismo tiempo los dispositivos deben estar escuchando los mensajes que pueden provenir del servidor a través de MQTT para ordenar el encendido o apagado del dispositivo sensado.
El problema que surge en este punto es que al hacer esta espera de 5 segundos, MQTT se bloquea y no recibe mensajes por parte del servidor. Por esta razón se pensó añadir una pequeña espera (delay) de un segundo pero sin bloquear a MQTT. Luego de algunas pruebas sacamos la conclusión de que si le agregamos un delay de 100 milisegundos no se bloquea MQTT y el NodeMCU recibe mensajes para prenderse y apagarse mientras sensa pero lo medido se altera y envía datos erróneos que modifican considerablemente la toma de datos de consumo eléctrico.

3.3.2 Solución a los Interbloqueos

Este comportamiento de los dispositivos recién mencionado genera un estado de bloqueo dentro de la iteración que sensa y envía los datos acumulados no permitiendo al dispositivo escuchar a su canal de suscripción MQTT para obtener los datos u órdenes enviadas desde el servidor. Lo primero que se pensó fué realizar una pequeña espera de 100 milisegundos utilizando la función *delay*. *E*sta espera lograba mejorar la respuesta al servidor por parte del dispositivo pero generaba una variación en el comportamiento del sensor de corriente que hacía variar drásticamente los datos de sensado. En una segunda aproximación a la solución se optó por investigar el comportamiento del sensor y la recepción de datos con esperas no bloqueantes y de esta manera lograr una especie de paralelismo teniendo dos “hilos” de ejecución, uno que se encargue de la medición y el envío de datos al servidor y el segundo destinado a la recepción de órdenes por parte del servidor y ejecución de dichas órdenes (encendido o apagado del módulo relé).

Para lograr el paralelismo de funciones entre la medición de corriente constante (milisegundos), el envío de datos (MQTT publish) cada 5 segundos y la recepción de órdenes provenientes del servidor (MQTT subscribe callback), se optó por utilizar la librería de Arduino llamada Metro. Metro nos brinda la posibilidad de sincronizar nuestras funciones mediante la configuración de un tiempo (en milisegundos por defecto) en el que nuestras funciones estarán listas para ejecutarse. Poniendo un tiempo de 100 milisegundos a la recepción de datos por parte del servidor y un tiempo de 1000 milisegundos para el sensado y envío de datos logramos no bloquear nuestras funciones para hacer el sensado, envío y recepción de datos en paralelo

haciendo que la función de recibir órdenes sea 10 veces más prioritaria (en tiempo de ejecución) que la de medición y envío de datos.

3.3.3 Conectividad al router borde.

Teniendo en cuenta que el dispositivo de hardware elegido para esta investigación (con el fin de conectar un sensor de corriente y enviar los datos pre calculados al servidor mediante WiFi) es el NodeMCU basado en el controlador ESP8266 v12-E nos encontramos con una primer problemática, conectar el NodeMCU a una red WiFi local de manera genérica. Explicando un poco más en profundidad el problema, el NodeMCU tiene la capacidad de conectarse como cliente a una red WiFi indicando tanto el SSID y la contraseña de dicha red, sin embargo, como nuestro sistema no será de instalación fija, si no que, debe garantizar la posibilidad de ser instalado en cualquier red local o remota para ser utilizado para cualquier propósito, entonces es preciso brindar la posibilidad de que un administrador o usuario común sin conocimientos previos logre configurar cada dispositivo para que se conecte a su red local de preferencia. Para esto es preciso generar una solución mediante algún tipo de interfaz sencilla que permita a un usuario no calificado ingresar y colocar las credenciales de la red (SSID, y password) a la que desea conectar sus dispositivos para que tengan conectividad remota a través del router. Dicha solución será explicada y detallada a continuación.

3.3.4 Solución a Conectividad al router borde: Comunicación y Configuración WiFi

La problemática recién descripta radica en la necesidad de una interfaz que sirva al usuario para configurar la conexión WiFi del dispositivo de sensado a la red local. Para lograr dicha conexión es necesario que el usuario ingrese el ssid y la contraseña de la red. A su vez, con el fin de que los dispositivos envíen los datos al servidor es preciso que conozcan la dirección IP del mismo.
Para lograr que el usuario tenga la posibilidad de ingresar el ssid, password e IP del servidor remoto (o local) se desarrolló una interfaz web alojada en la memoria interna (EPROM) del NodeMCU con código html simple. Cuando un dispositivo es conectado y se enciende lee su EPROM, si contiene algún ssid, password e IP intenta conectarse a la red local y luego al servidor, de no lograr alguna de las conexiones o no tener guardados dichos datos pasa a otro modo, el modo Access Point. En el modo Access Point el NodeMCU brinda una red propia para que el usuario se conecte, al conectarse y navegar a una IP específica que provee el dispositivo se presentan las redes escaneadas por el dispositivo (sus ssids y su potencia de señal) y tres entradas de texto, una para ingresar el ssid de la red que queremos que se conecte el dispositivo, la segunda para la contraseña de dicha red y la tercera para ingresar la IP o url del servidor a donde se enviarán los datos. Luego de ingresados los datos el dispositivo intenta conectarse a la red y al servidor dejando el modo Access Point y entrando en modo de sensado y envío de datos. De esta manera logramos brindar a cualquier tipo de usuario la posibilidad de configurar los dispositivos para cualquier red que desee y cualquier servidor.

3.3.5 Visualización de datos en tiempo real: conectividad websocket

Como se comentó en partes anteriores del documento, una de las funcionalidades más importantes del sistema es la visualización en tiempo real del consumo de los aparatos electrónicos. En base a esta premisa se pensaron varias alternativas y tecnologías, tomando como la opción más viable la tecnología WebSocket, que nos permite establecer un canal de comunicación directo entre la interfaz y el servidor, que persista en el tiempo y a través de él lleguen los datos nuevos a representar en los gráficos correspondientes.

En esta sección de problemáticas queríamos mostrar un par de los inconvenientes que fuimos teniendo con esta tecnología y como los fuimos solucionando.

Yendo directamente a las situaciones que nos encontramos, las pasaremos a explicar a continuación:

En primer lugar, no en todas las secciones de nuestro sistema era necesaria la conexión en tiempo real, es más, solamente en dos secciones es necesaria: en el inicio y en el monitor. Cuando intentamos recorrer las otras funcionalidades del sistema esta conexión no debería entorpecer el funcionamiento, es decir, que siga abierta la conexión de datos en tiempo real cuando estamos configurando nuevas tareas o visualizando datos históricos.

Por otro lado, la tecnología de websocket lleva algunos años de desarrollo pero aun así tiene inestabilidades en el funcionamiento. Estas inconsistencias son provocadas por varios factores: la conexión a internet del dispositivo que está usando la interfaz, la arquitectura y potencia del servidor que aloja la información, problemas internos de la librería no soportados en las versiones actuales, entre otras cosas.

Por último, para poder trabajar con una función que sea en tiempo real era fundamental y casi obligatorio que la interfaz provea de un feedback y un manejo de errores a la altura de la situación. Si la conexión falla por algún motivo, si no han llegado datos por algún motivo o cual fuese el error, el usuario tiene que ser el primero en enterarse. Para esto es que se pensó también una resolución.

3.3.6 Solución a la conectividad websocket

Varias eran las opciones que se nos aparecían para resolver estos temas que planteamos en el punto anterior, pero tuvimos que tomar definiciones y aquí las expondremos.

En primer término, para que no haya interferencia entre la conexión websocket y las demás funcionalidades se decidió que la misma solamente funcione en las pantallas que la necesiten (y

no en segundo plano todo el tiempo). Cuando ingresemos a las secciones de inicio y de monitor, se establecerá una nueva conexión con el servidor, que servirá para esa sección y se cerrará cuando nos dirijamos hacia otra sección se cerrará en caso que esta se haya establecido.

Otra opción que en principio parecía mejor y podríamos haber usado pero la tecnología no lo permitía era reutilizar la conectividad: por única vez al comienzo del uso del sistema se establecería la conexión que sirva para toda la sesión y permanezca abierta en segundo plano. Esta opción no fue viable principalmente por dos motivos, primero la duración en el tiempo de la misma era relativo, podía caerse en cualquier momento y sería más difícil recuperarla si fallara, y por otro lado la funcionalidad de la conectividad en tiempo real para el inicio y para el monitor era distinta; en ambos casos llegaban datos nuevos para mostrarse, pero en las dos pantallas se resolvía de forma distinta y eso era muy complejo de configurar esa variante en la tecnología websocket adaptada a angular y javascript.

De todas formas, para este tema se terminó resolviendo tener una variable global del sistema que contenga dicha conexión y sobre ella se opere, creando y/o cerrándola, a modo de tener un objeto único (patrón de diseño Singleton) y no tener muchas instancias de la misma en tiempo de ejecución.

Por otro lado, se hizo hincapié en un fuerte análisis de errores posibles de la tecnología y de la vinculación con el servidor a través de la red en general. Con ello se implementó el feedback correspondiente al usuario, para que en todo momento sepa que está pasando en el sistema, porque llegan o no llegan datos, cuál fue el último y en qué estado está la conexión.

En las dos pantallas de inicio y monitor aparece el estado de la conexión en tiempo real, si está establecida o no con un botón (o ícono) verde en caso positivo o rojo negativo. Ese botón también sirve para regenerar la conexión en caso que sea necesaria (figuras 3.27 y 3.28).
Durante el momento que se intenta conectar o reconectar aparece un icono girando, que cuando se establece (o no) cambia nuevamente al botón del color. Esto permite mostrar al usuario que hay una operación en proceso y debe esperar. También desaparecen algunos botones para que no haya posibles errores de apretar funciones que se crucen.

Como conclusión del uso de la tecnología websocket podemos decir que tiene cierta inestabilidad en el uso, principalmente cuando se inician y cierran muchas conexiones en simultáneo, pero mediante capturas de errores, reconexión y buen feedback al usuario se puede utilizar correctamente.

4. Pruebas de Campo

4.1 Prueba de estrés de Conexión

Las pruebas de estrés de conexión que se realizan son principalmente de envío de datos desde cada unidad independientemente (estudiando la reacción individual de cada unidad a este flujo) y de la capacidad de respuesta de conexión del servidor al tener muchos dispositivos conectados. Estas pruebas se desarrollaron, se obtuvieron conclusiones y se aplicaron los debidos ajustes. Todos los procesos que se acaban de mencionar para las pruebas fueron de carácter cíclico y se describen a continuación.

Estrés de envío de datos en franjas temporales ajustables (punto de vista de los dispositivos):

Luego de una serie de pruebas se llegó a la conclusión de que si se envían datos cada un segundo desde los NodeMCUs (componente principal de cada Unidad Elektron) al servidor (tanto local como remoto) se puede llegar a reiniciar el dispositivo debido a la poca memoria principal que el mismo tiene. Vemos como solución aumentar el intervalo de envío de datos de 1 segundo a 5 teniendo que generar una variable acumulada del consumo eléctrico en esos 5 segundos dentro de la lógica del dispositivo. Esto podría sacarnos precisión de medición o sensación de realismo con respecto a la visualización de la situación en la medición eléctrica en tiempo real pero ayudaría a mejorar a bajar el estrés de conectividad entre el servidor y los dispositivos y el servidor y las interfaces gráficas. Además aumentar el intervalo de 1 segundo a 5 bajaría la cantidad de datos creados en la base de datos lo que aliviana la densidad de datos en el servidor y en la base de datos para su mantención a largo plazo, escalabilidad en cantidad de dispositivos en simultáneo por sistema y para la búsqueda y representación de datos para la generación de estadísticas posteriores.

Luego de cambiar la cantidad de tiempo a 5 segundos entre los envíos de los datos de sensado obtenemos un mejor rendimiento por parte de los módulos de hardware (se bloquean y reinician con mucha menor frecuencia, casi nula) y además logramos mitigar un poco la enorme densidad de datos almacenados en la base de datos del lado del servidor lo que también ayuda en las consultas para su obtención mejorando la performance del sistema en general.

4.2 Pruebas de performance

Como mencionamos previamente en el capítulo 2, los artefactos que este sistema puede medir son los que tengan cargas resistivas. Es por eso que elegimos para las pruebas una lamparita incandescente común, una plancha de ropa común, y un soldador de estaño común. El objetivo las pruebas fue verificar el correcto funcionamiento tanto del sensor como del relé, en mediciones de artefactos comunes que se encuentran en un hogar.

Los resultados arrojados fueron positivos ya que obtuvimos una buena respuesta tanto del sensor de corriente como del relé en todas las mediciones. Se observó durante un tiempo prolongado si se mantenía o no el flujo de datos por parte de los sensores al servidor y la capacidad de conexión y desconexión del relé a través de los numerosos ciclos.

A continuación se muestran capturas de pantalla de los resultados arrojados en el tiempo para cada dispositivo medido en varias pruebas con diversos consumos.

Lampara incandescente de 28 Watt

Para el caso de la lámpara común, se midió durante menos de 1 minuto (entre las 00:09 y las 00:10 hs) con un total de 20 muestras, y se arrojó un resultado entre los valores de ~41.5 y ~51.3 w, lo que nos indica que están dentro de los valores normales.

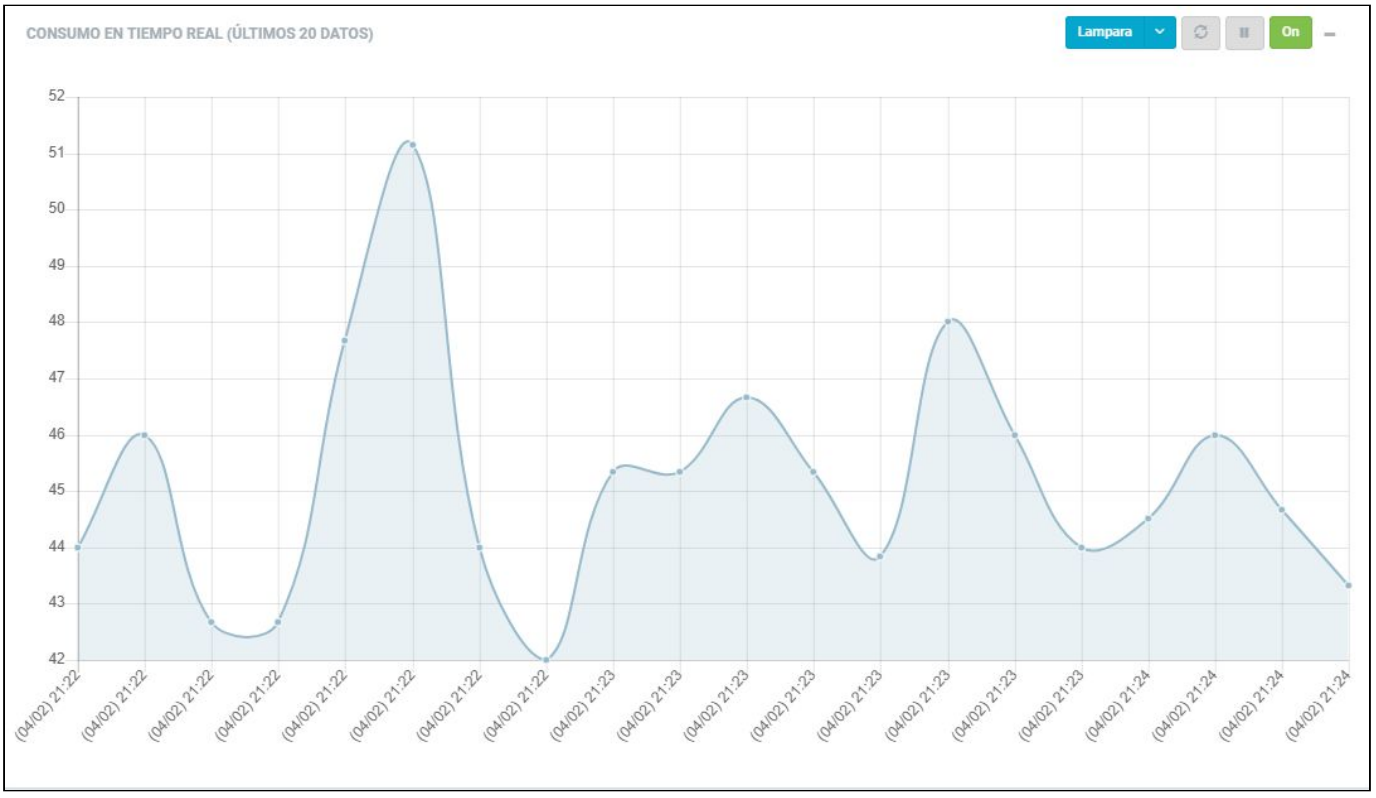


Figura 4.1 - Captura de pantalla del gráfico de sensado en tiempo real de una lámpara incandescente.

Soldador de Estaño de 30 Watt

En el caso de la medición de corriente del soldador de estaño también se registraron datos durante menos de un minuto y observamos una estabilidad aún mayor que la lamparita.
Los valores obtenidos están entre los ~41.5 y ~51 lo que nos dice que tiene un consumo muy estable.

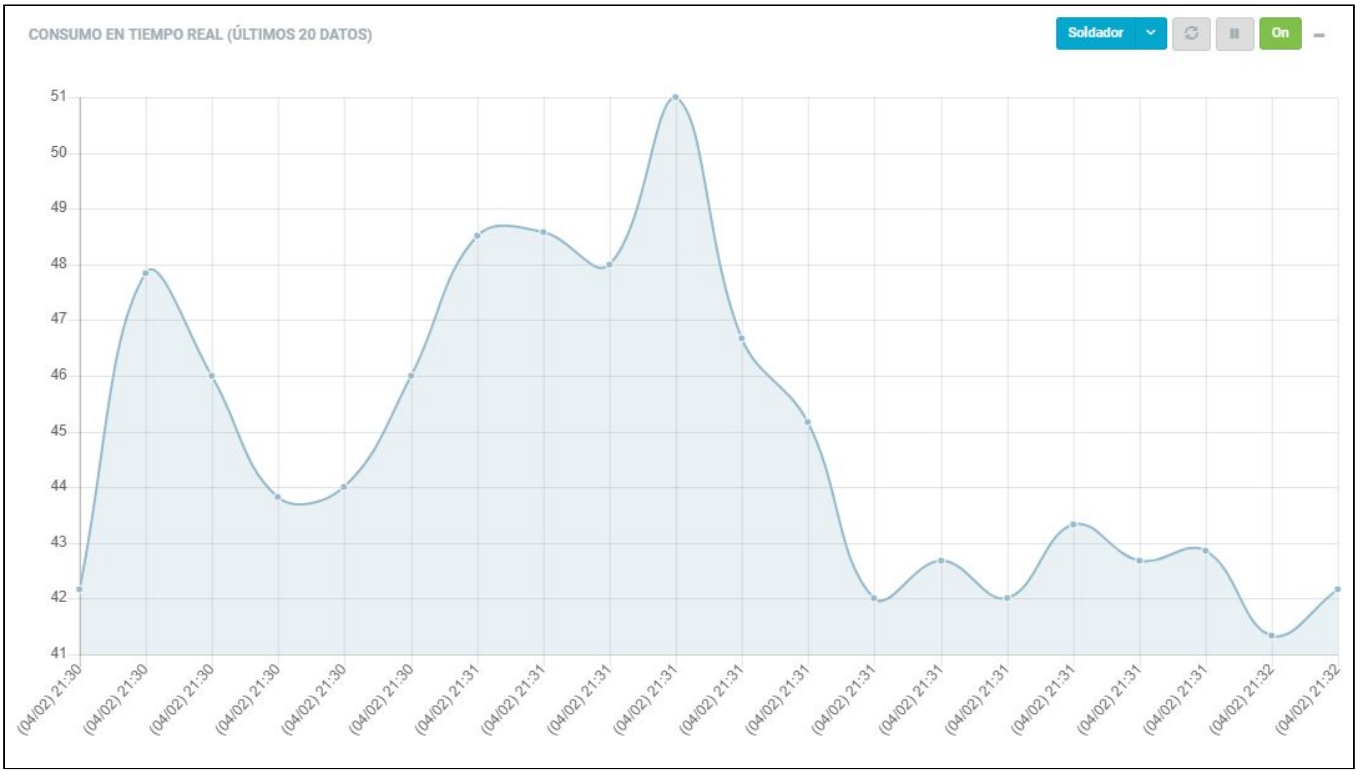


Figura 4.2 - Captura de pantalla del gráfico de sensado en tiempo real de un soldador de estaño.

Plancha (a máxima potencia):

Para el caso de la plancha común (a máxima potencia) se obtuvieron datos por casi tres minutos, comprendiendo el momento de calentamiento de la plancha, el post calentamiento bajando mucho el consumo y un nuevo calentamiento cuando se enfrió levemente.
En el calentamiento inicial, se observa un consumo que llega a los ~800w y se mantiene durante 30 segundos aproximadamente. Una vez llegado al tope de calentamiento, disminuye notablemente el consumo llegando alrededor de los ~30w, hasta volver a efectuar un calentamiento. En ese nuevo calentamiento, el pico alcanzado es menor al inicial, siendo de alrededor de ~650w.
A comparación de los otros dos artefactos, observamos que la plancha tiene un consumo muchísimo más alto que se efectúa en poco tiempo, mientras que los otros dos artefactos, consumen estable durante todo el periodo.

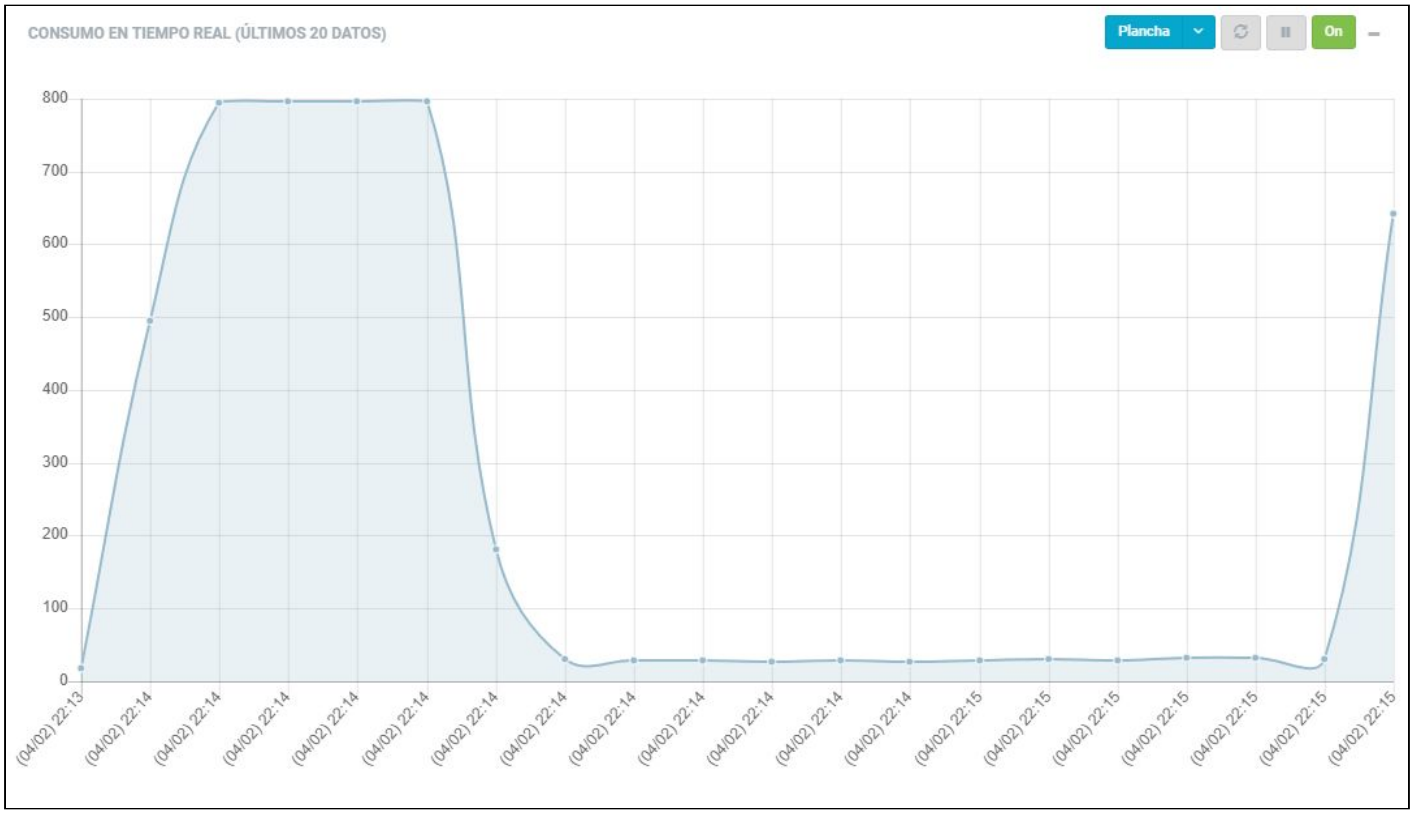


Figura 4.3 - Captura de pantalla del gráfico de sensado en tiempo real de una plancha común (a máxima potencia).

4.3 Prueba de estrés de Datos

Las pruebas de estrés de datos tienen como objetivo analizar si existe alguna problemática en alguna de las partes del sistema sea en el almacenamiento o graficado de los datos en el caso de tener múltiples dispositivos funcionando en una franja de tiempo considerable. Para realizar estas pruebas se desarrollaron una serie de bots o simuladores de dispositivos o unidades Elektron tan sólo con software. En total se pusieron en funcionamiento cuatro dispositivos simulados o bots durante cuarenta días los cuales enviaban datos cada cinco segundos generando un total de 691200 datos. En ningún momento surgió alguna problemática del sistema con respecto a esto más que la necesidad de paginar los datos al ser consumidos por las aplicaciones del Frontend, algo normal en la mayoría de los sistemas. Sin embargo no se descarta la posibilidad de mejorar el almacenamiento de datos comprimidos en patrones de comportamiento en los casos de no existan picos de medición que sirvan de muestra para estudios futuros con el objetivo de alivianar la cantidad de registros en la base de datos.

5 Conclusiones y Trabajo a Futuro

5.1 Conclusiones generales

Luego del trabajo de investigación y desarrollo llevado a cabo se lograron casi en su totalidad los objetivos que se plantearon al comienzo del mismo. A pesar algunas limitaciones (mencionadas en la sección de problemas y soluciones), se consiguió generar una herramienta útil para la medición, control y análisis del consumo de energía y comportamiento de los artefactos eléctricos y electrónicos.

Los datos obtenidos en tiempo real y almacenados a lo largo del tiempo permiten no solamente calcular el consumo de un ambiente o los consumos específicos de un aparato, sino también la forma en que estos consumen. Es decir, si mantienen una estabilidad, si tienen comportamientos erróneos o también hacer comparaciones entre varios aparatos iguales para corroborar si su consumo es equivalente. Como se detalla más adelante como trabajo a futuro, con la incorporación de los sensores adecuados se podrá completar el monitoreo de consumo de todo tipo de aparato, incluyendo aquellos con cargas reactivas, por ejemplo.

Cabe destacar la importancia de que este desarrollo puede serle útil tanto al usuario común para verificar sus consumos como a los usuarios industriales, con verificación de consumos más extensos o sobre determinadas maquinarias. También aporta información y herramientas para desarrolladores de artefactos eléctricos de bajo consumo o para investigadores especialistas en corriente alterna. Lo consideramos como aporte al ámbito académico en general, pero también al productivo, incluyendo el desarrollo industrial y el impacto en lo económico y social.

5.1.1 Sobre el Ahorro de Energía

Uno de los principales objetivos del sistema es proporcionar al usuario una herramiento para reducir los gastos de consumo eléctrico. Ésto es posible gracias a varios elementos.

En primer lugar desde el análisis de consumo, es decir, si a través del sistema observamos que un aparato consume mucha energía, intentaremos minimizar su uso o directamente no usarlo si es posible.

En segundo lugar, el sistema nos brinda la posibilidad de encender o apagar los artefactos mediante tareas programadas, esto permite que en momentos en que se puedan apagar aparatos se haga de forma automatizada. También en el caso de las tareas por consumo, si sobrepasa determinado consumo (por ejemplo un aire acondicionado) podría apagarse para evitar consumos excesivos.

Otro de los objetivos (y no por eso menos importante) era el de concientización del uso de energía y mejora del rendimiento de los recursos elećtricos con el objetivo de la disminuir

liberación de dióxido de carbono. Nuestra solución además de permitir el automatizado de tareas para mejorar los consumos de los aparatos eléctricos de manera inteligente muestra la cantidad de energía utilizada por cada dispositivo, intentando fomentar la concientización por parte de los usuarios y reducir consumos innecesarios.

5.1.2 Sobre el desarrollo

5.1.2.1 Servidor: Django Rest API

La utilización de Django -framework- para el desarrollo del núcleo del sistema web resultó muy eficiente, ya que el mismo se caracteriza por su capacidad de escalabilidad, y nos permitió satisfactoriamente la incorporación de otros servicios o módulos.

Django nos permitió además encapsular dentro de su estructura todas las diversas y complejas funcionalidades, ofreciendo de manera simple la comunicación hacia las aplicaciones web y móvil como también hacia las unidades elektron. Cada una de las operaciones son representadas mediante una URL en el sistema web y cada una de las respuestas es mediante un JSON que simplifica mucho la comunicación y la eficiencia de la red.

Hay que destacar que tanto la naturaleza de Django como el poder de Python permitió simplificar la codificación de las acciones complejas. Por ejemplo, permitir al Frontend determinar el offset y el limit de una lista retornada por una query, serializar cada objeto y retornarlo en formato JSON en una sóla línea de código.

Sin embargo, todavía no existen librerías funcionales y oficiales de Django para su integración con tareas automatizadas, devolución de datos por Websocket o transferencia de datos a dispositivos remotos por MQTT. Esto nos obligó a desarrollar los servicios (demonios) de MQTT, Websocket y de Tareas (Tasks) como otros “clientes” que consumían y creaban objetos en el sistema a través de los métodos de la API.

5.1.2.2 Servidor: Demonios

El sistema posee múltiples interacciones entre sus diversos subsistemas y con varios protocolos de comunicación (TCP, Websocket y MQTT) a la vez. Por esta razón fue necesario elegir entre dos posibilidades, buscar soluciones y estrategias que se puedan acoplar nativamente con Django o desarrollar las propias. Debido a lo mencionado en el apartado anterior se optó por el desarrollo propio de los demonios o servicios que se encargaran de gestionar las distintas comunicaciones y funcionalidades extra.

Esto nos permitió también que el sistema esté separado en módulos acoplables y desacoplables, pero como contracara negativa encontramos que no se puede garantizar la misma estabilidad que tienen las librerías ya probadas e integradas nativamente al framework.

5.1.2.3 Frontend

En cuanto a las interfaces de usuario logradas, se pueden concluir varias cosas: en primer lugar, el enfoque puesto en la usabilidad, es decir, en la visión de cómo el usuario interactúa con el sistema, nos brindó un valioso aporte de cómo deben mostrarse los datos, que información es más útil que otra, como simplificar cada una de las pantallas sin que pierda la esencia de lo que quiere comunicar, cómo ubicar un contenido sobre otro y principalmente que la interacción entre el humano y el sistema sea lo más fluida posible, con buen feedback y buen manejo de errores.

Por otro lado, pensar en una doble interfaz para dispositivos de distinta naturaleza (formato móvil y web) nos hizo investigar sobre la gran diversidad de tecnologías que hoy existen y cómo cada una se aplica de mejor forma a un dominio determinado. A su vez, qué cosas existen en común para poder reutilizar secciones y módulos dentro de cada una de ellas, como enfocar los componentes visuales a cada tipo de pantalla y como se dispone la información dependiendo de qué tipo de usuario interactua.

5.2 Trabajo a Futuro

5.2.1 Capa de abstracción para la comunicación

Uno de los puntos más importantes al hablar de múltiples dispositivos que se comunican entre sí o con un servidor central es entender que este concepto es mejor conocido como comunicación máquina a máquina o M2M como se mencionó en repetidas ocasiones. Cuando hablamos de protocolos de comunicación M2M debemos tener en cuenta que existen muchas opciones y que no siempre vamos a poder elegir con libertad el protocolo utilizado tanto en los dispositivos como en el servidor. Por ejemplo, es posible que en alguna implementación del sistema sea necesario enviar los datos de los dispositivos elektron utilizando el protocolo COAP o WiFiDirect por lo que el servidor debería poseer una capa de abstracción como endpoint transparente para la comunicación M2M para soportar tanto MQTT, como los dos protocolos que se acaban de mencionar.

5.2.2 Incorporación de sensor de tensión

Una de las problemáticas que se encontraron durante el desarrollo e investigación del proyecto fue el hecho de no poder medir cargas reactivas por falta del valor de la tensión. Como se detalló en el capítulo 2 para poder calcular el consumo eléctrico de cargas reactivas es preciso medir la tensión y la corriente simultáneamente, algo que no es posible con nuestro proyecto por contar solo con un sensor de corriente. Como trabajo a futuro planteamos la posibilidad de

incorporar un sensor de tensión para ese fin. Para que esto sea posible, es necesaria también la incorporación de una plaqueta controladora con más de un convertidor analógico digital, como por ejemplo, la placa ESP32. Esta placa cuenta con dos bloques de 8 pines con la capacidad de tomar datos anlógicos y un módulo Bluetooth que ayudaría en las configuraciones iniciales del sistema de sensado (la configuración para la conexión al punto de acceso de la red inalámbrica) [14].

La placa actual que utilizamos (NodeMCU basada en el microcontrolador ESP8266) cuenta con una sola entrada con conversor anlógico digital, lo que dificulta la lectura de dos sensores analógicos en simultáneo. Con esta placa solo sería posible si se logra multiplexar la lectura anlógico digital para leer los dos sensores concurrentemente.

En conclusión, incorporando la placa ESP32 junto con el sensor de tensión (cada sensor tendría un pin propio para comunicarse) y con algunos cambios en el código que ejecutan las placas, se podría medir el consumo de cargas reactivas y enviar dichos datos al mismo sistema que se desarrolló para ésta investigación sin tener que realizar cambios en el Backend ni en el Frontend.

[14] ESPRESSIF: https://www.espressif.com/en/products/hardware/esp32/overview

Índice de figuras

Bibliografía

[1] Cambio Climático .org - Protocolo de Kioto - http://www.cambioclimatico.org/tema/protocolo-de-kyoto

[2] Peter U. Clark et al.: Consequences of twenty-first-century policy for multi-millennial climate and sea-level change. Nature Climate Change 6, 2016, 360-369, doi:10.1038/NCLIMATE2923

[3] Cambio Climático .org Causas de los cambios en el clima http://www.cambioclimatico.org/tema/causas-del-cambio-climatico

[4] IPCC Glossary Working Group III, p. 818

[5] Francis Cairncross (30 de octubre de 2006). «" Time to get Stern on climate change"». *The First Post*. Archivado desde el original el 3 de noviembre de 2015.

[6] Grupo Intergubernamental de Expertos sobre el Cambio Climático - Sistemas de Energía http://www.ipcc.ch/pdf/assessment-report/ar5/wg3/ipcc_wg3_ar5_chapter7.pdf

[7] "Research and Markets: Global Home Automation and Control Market 2014-2020 - Lighting Control, Security & Access Control, HVAC Control Analysis of the $5.77 Billion Industry". *Reuters*. 2015-01-19. Archived from the original on 2016-05-05.

[8] An Introduction to AC Power - https://learn.openenergymonitor.org/electricity-monitoring/ac-power-theory/introduction

[9] Librería de MQTT PubSubClient - https://pubsubclient.knolleary.net/

[10] "Web Services Architecture". World Wide Web Consortium. 11 February 2004. 3.1.3 Relationship to the World Wide Web and Rest Architectures. Retrieved 29 September 2016.

[11] https://ionicframework.com/

[12] https://ionicframework.com/docs/v1/components/

[13] Post sobre el gran impacto de uso de la herramienta en el 2015: https://blog.ionicframework.com/how-2015-went-for-ionic/

[14] https://es.wikipedia.org/wiki/AngularJS

[15] Página oficial de la librería Angular JS: https://angularjs.org/

[16] Última versión de ionic disponible: https://github.com/ionic-team/ionic/releases/tag/v3.9.2

[17] Librería de gráficos en javascript adaptada a la librería Angular JS http://jtblin.github.io/angular-chart.js/

[18] Krug, Steve (Pearson Educación, 2006). “No me hagas pensar: una aproximación a la usabilidad en la web”. ISBN 8483222868, 9788483222867
https://books.google.com.ar/books/about/No_me_hagas_pensar.html?id=MlPtOwAACAAJ&redir_esc=y&hl=es-419

[19] Nielsen, Jacob (1993). “Usability Engineering”. ISBN 0-12-518406-9