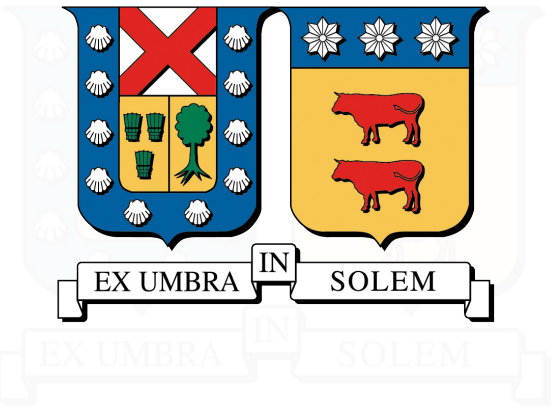
2022-08

iHEMS: A Web-Based Service for Monitoring Smart Homes

Yuste Lucero, Sebastián Ignacio

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAISO - CHILE



iHEMS: A Web-Based Service for Monitoring Smart Homes

SEBASTIÁN IGNACIO YUSTE LUCERO

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL TELEMÁTICO

PROFESOR GUÍA	:	MOHAMED ABDELHAMID
PROFESOR CORREFERENTE	:	JOSÉ MANUEL MARTINEZ

Agosto 2022

RESUMEN EJECUTIVO

El consumo de energía eléctrica está aumentando de manera significativa a lo largo de todo el mundo, llegando a estimarse que en un par de años las distribuidoras de este preciado recurso, no sean capaces de dar abasto para solventar la necesidad de todos los usuarios.

Con el fin de poder concientizar a los usuarios de su consumo energético y el uso de energías renovables es que nace el desafío de desarrollar e implementar una plataforma IoT para el monitoreo de los electrodomésticos del hogar, la que permite a los usuarios de ella observar su consumo en tiempo real. Este trabajo de título se ve alimentado por datos provenientes de electrodomésticos reales, con ayuda de un intermediario entre la toma de corriente y el dispositivo objetivo. Estos intermediario son los Smart Plugs, los que envían continuamente información hacia un servidor en la nube para ser accedida por la plataforma y ser desplegada al usuario en distintos intervalos de tiempo, los que pueden ser seleccionados por él.

La idea de la plataforma propuesta es que los usuarios puedan tener su propia cuenta dentro de ella, donde pueden ver el consumo de sus dispositivos organizados por habitaciones. Además, cada usuario tiene la posibilidad de adquirir el número necesario de Smart Plugs y agregarlos o desagregarlos dentro de la plataforma por medio de un código asociado al dispositivo físico.

En este documento se explora el diseño de la arquitectura IoT del desafío, en la cual se divide la problemática por capas, las que implican la toma de datos, la comunicación de ellos, su almacenamiento y su aplicación dentro de la plataforma. Además, se muestra los bosquejos generados para la visualización de la información, y se finaliza con el resultado obtenido y su comparación con el resultado esperado.

keywords: HEMS (Home Energy Management System), ILM (Intrusive Load Monitoring), IoT platform, Web Application, SLN (Smart Load Node)

ABSTRACT

The consumption of electricity is increasing significantly throughout the world, and it is estimated that in a couple of years the distribution power company will not be able to meet the demand of all users.

In order to make users aware of their energy consumption and the use of renewable energy resources, the challenge of developing and implementing an IoT platform for monitoring household appliances, which allows users to observe their consumption in real time, was born. This thesis work is fed by data coming from real household appliances, with the help of an intermediary between the power outlet and the target device. These intermediaries are the Smart Plugs, which continuously send information to a server in the cloud to be accessed by the platform and displayed to the user at different time intervals, which can be selected by the user.

The idea of the proposed platform is that users can have their own account within it, where they can see the consumption of their devices inside the home. In addition, each user has the possibility to acquire the required number of Smart Plugs and add or remove them within the platform by means of a code associated to the physical device.

This work explores the design of the IoT architecture of the challenge, in which the problem is divided by layers, which involve data collection, data communication, storage and application within the platform. It also shows the sketches generated for the visualization of the information, and ends with the result obtained and its comparison with the expected result.

Glosario

- **API**: Conjunto de funciones y procedimientos que permiten acceder a las características o datos de otro sistema informático.
- **Backend**: Parte de un sistema informático a la que no accede directamente el usuario, normalmente responsable de almacenar y manipular datos.
- **Endpoint**: URL a tráves del que se accede a la API.
- **ESP8266**: Microchip Wi-Fi de bajo coste, con software de red TCP/IP incorporado, y capacidad de microcontrolador.
- **Firmware**: Clase específica de software informático que proporciona el control de bajo nivel para el hardware específico de un dispositivo.
- **Frontend**: Parte de un sistema informático con la que el usuario interactúa directamente.
- **Gateway**: Dispositivo utilizado para conectar dos redes diferentes, especialmente una conexión a Internet.
- **HEMS**: Plataforma tecnológica compuesta por hardware y software que permite al usuario supervisar el uso y la producción de energía y controlar y/o automatizar manualmente el uso de la energía en un hogar.
- **IoT**: Interconexión a través de Internet de dispositivos informáticos integrados en objetos cotidianos, lo que les permite enviar y recibir datos.
- **JSON**: Estándar de formato de archivos e intercambio de informacion
- **Machine Learning**: Uso y desarrollo de sistemas informáticos capaces de aprender y adaptarse sin seguir instrucciones explícitas, mediante el uso de algoritmos y modelos estadísticos para analizar y sacar conclusiones de los patrones de los datos.
- **Microgrids**: Pequeña red de usuarios de electricidad con una fuente de suministro local que suele estar conectada a una red nacional centralizada pero que es capaz de funcionar de forma independiente.

- **Node.js**: Entorno de ejecución de JavaScript de código abierto y multiplataforma.
- **Raspberry Pi**: Ordenador de bajo costo de tamaño compacto con la habilidad de interactuar con el mundo exterior por medio de modulos.
- **SLN**: Nodo que permite el monitoreo del consumo de electrodomésticos además de agregar una capa de inteligencia a los mismos sin la necesidad de modificarlos ni al tomacorriente,
- **Smart Grids**: Red de suministro de electricidad que utiliza la tecnología de las comunicaciones digitales para detectar y reaccionar a los cambios locales en el uso.
- **Smart Homes**: Casa equipada con iluminación, calefacción y dispositivos electrónicos que pueden controlarse a distancia mediante un smartphone o un ordenador.
- **Smart Plug**: Enchufe inteligente que va colocado entre los cables de alimentación de un dispositivo y las tomas de corriente.
- **Uptime**: Tiempo durante el cual una máquina, especialmente un ordenador, está en funcionamiento.
- **Telemetría**: Proceso de registro y transmisión de las lecturas de un instrumento, en este caso de los smart plugs.
- **TIC**: Tecnologías de la Información y la Comunicación son los recursos y herramientas que se utilizan para el proceso, administración y distribución de la información a través de elementos tecnológicos.

Índice de Contenidos

Índice de Tablas

Índice de Figuras

1 | Introducción

En la actualidad, el consumo de energía de la red residencial -incluidos los hogares y los edificios- aumenta continuamente.

Estamos en una era en que se necesita un cambio, porque como usuarios nos estamos volviendo más conscientes del uso eficiente de la energía, queriendo incluso llegar a producirla mediante paneles fotovoltaicos o turbinas eólicas, y ese cambio viene de la mano de las Smart Grids, el Internet of Things (IoT) y el Internet of Energy (IoE).

Se ha prestado gran atención a los sistemas de gestión de la energía en el hogar (HEMS por su sigla en ingles). Sin embargo, los electrodomésticos que ya se utilizan son cargas no inteligentes que no admiten ninguna instalación de comunicación y procesamiento de la información.

1.1. ¿Cómo son los HEMS actualmente?

Se pueden considerar dos enfoques principales para la comunicación entre los electrodomésticos y los HEMS:

- Hacer que los electrodomésticos sean inteligentes incorporando la comunicación y el procesamiento de la información.
- Integrar inteligencia en las tomas de corriente.

Para la primera solución, hay muchas cargas no inteligentes y resulta más costoso incrustar la inteligencia específica en los electrodomésticos para convertirlos en nodos de carga inteligentes (SLN por su sigla en ingles). En el caso de la segunda solución, para implantar la inteligencia (comunicación y procesamiento de la información) en las tomas

de corriente, es necesario realizar importantes modificaciones en la infraestructura del cableado eléctrico [1].

1.2. Equipo de desarrollo

Dentro del contexto de memorias multidisciplinarias se formó el equipo iHEMS, el cual esta compuesto por:

- Nicolás Calderón: Estudiante de Ingeniería Civil Telemática, encargado del desarrollo de la capa física y de comunicación del desafío.
- Sebastián Yuste: Estudiante de Ingeniería Civil Telemática, encargado del desarrollo de la capa middleware y aplicación del desafío.

1.3. ¿Cuál es nuestro desafío?

Como equipo iHEMS nos enfrentamos al desafío de desarrollar una plataforma para: recoger, almacenar y mostrar el consumo energético en un conjunto de hogares. Para ello se utilizó un modelo de monitoreo intrusivo de cargas, que implica la instalación de smart plugs, los que se ubican entre los electrodomésticos y la toma de corriente, con el fin de monitorear continuamente el consumo de energía de cada uno de los aparatos del hogar.

1.4. Acercamiento a la solución

Este desafío nace con el propósito de extender el trabajo realizado previamente por el equipo de EViG, en el que hacen una prueba de conceptos acerca de la toma y almacenamiento de mediciones de propiedades físicas de los dispositivos eléctricos del hogar (potencia, corriente, voltaje, entre otros). Su objetivo principal es desarrollar una plataforma de gestión de energía para la red integrada de vehículos eléctricos (EViG) y fuentes de energía renovable en el sistema de distribución de energía. Se pretende alcanzar los siguientes objetivos de investigación:

- Desarrollo de una plataforma de gestión energética inteligente basada en IoT para una casa inteligente con un vehículo eléctrico. La plataforma propuesta debe ser capaz de gestionar la carga del vehículo eléctrico mediante el control de los electrodomésticos inteligentes de la casa y los recursos de energía renovable para cumplir con las necesidades del consumidor sin sobrecargar el sistema de distribución de energía.

En este documento se exploró un nuevo enfoque, que tiene como objetivo presentar un funcionamiento eficiente para los electrodomésticos no inteligentes utilizando SLN en una red de área doméstica (HAN por su sigla en inglés). La solución SLN es una solución de bajo costo que no necesita ningún cambio en la infraestructura del cableado en el hogar -ya que va entre la toma de corriente y la carga- ni ningún cambio en los electrodomésticos en la fase de fabricación. Los nodos de carga inteligentes se distribuyen dentro del hogar para formar una HAN.

Se pueden utilizar diferentes tecnologías de comunicación inalámbrica como medio de comunicación.

Las principales aportaciones de la solución propuesta son:

- No es necesario modificar las infraestructuras eléctricas del hogar.
- No es necesario modificar los aparatos eléctricos del hogar.
- La solución puede soportar todo tipo de cargas no inteligentes.
- La solución proporciona diferentes medidas como la tensión, la corriente, la potencia y el factor de potencia.

El desafío comienza desde la toma de datos hasta su visualización y no se enfoca solamente en el desarrollo de una plataforma, debido a que el equipo espera que en el desarrollo de este se explore y compruebe que las tecnologías utilizadas hasta el momento son las mejores o si existen soluciones alternativas que permitan desarrollar un prototipo más sólido que logre mejores resultados para la solución general. Sin embargo en este documento solo se abarcan los contenidos relacionados al almacenamiento de los datos recibidos y la visualización de estos en la plataforma, tomando como base para su recolección el trabajo realizado por el otro integrante del grupo.

Una vez desarrollada la plataforma, se da el paso al equipo para futuras aplicaciones,

como la utilización de los datos recopilados para la detección de actividades del diario vivir analizando el uso de los dispositivos electrónicos mediante machine learning, o la compra y venta de energía entre pares en caso de un exceso de producción por parte de fuentes renovables mediante contratos inteligentes

1.5. Objetivo General

Diseño e implementación de una plataforma IoT para recoger, almacenar y mostrar tanto el consumo de energía de diferentes electrodomésticos, como también el consumo general de un conjunto de casas inteligentes en una comunidad, con el fin de concientizar sobre el consumo de energía a los usuarios.

1.6. Objetivos Específicos

- **Definición de requisitos y servicios:**

  Permite conocer las problemáticas del cliente y definir los objetivos a lograr durante el desarrollo total del proyecto planteado por la contraparte. Además, da paso a la discusión con la contraparte para que las funcionalidades propuestas y desarrolladas como solución sean acorde a lo que ellos requieren.

- **Definir una arquitectura de red para los sensores y medidores inteligentes:** Estructurar una arquitectura de red que conecte nodos, gateway y servidores de forma eficiente, con el fin de poder realizar mediciones de consumo eléctrico y almacenar la información obtenida para luego ser utilizada.

- **Levantar un servidor de almacenamiento en la nube:** Permite acceder a la información almacenada desde cualquier lugar, mediante el uso de una API.

- **Desarrollar plataforma para la visualización:** Dicha plataforma cuenta con la visualización del consumo de electricidad en tiempo real, con vistas del consumo individual de cada dispositivo para los dueños de casa, y vistas del consumo general de cada casa para un usuario administrador.

1.7. Estructura del documento

Este documento se estructura de la siguiente forma: en el Capítulo 2 se estudió los conceptos referentes a Smart Grid, IoT, IoE y la estructura y funcionamiento de los Smart Homes. En el Capítulo 3 se hizo una revisión de los trabajos previos referentes al uso de hardware, protocolos y aplicaciones que utilizaron en HEMS los autores de los diferentes documentos mencionados en él. En el Capítulo 4 se estructura el modelo de capas del desafío, y se explica los componentes pertenecientes a cada capa y la función que cumplen dentro del desarrollo. Además, se señala la arquitectura de componentes propuesta de la solución basados en lo estudiado en el Capítulo 3. En el Capítulo 5 se comparan y seleccionan las diferentes tecnologías que se utilizan en cada capa y se muestran los diseños preliminares de la plataforma. En el Capítulo 6 se presenta las configuraciones y desarrollo por capas de las tecnologías elegidas en los capítulos anteriores. En Capítulo 7 se muestra lo obtenido durante el desarrollo completo del desafío, además de analizar el cumplimiento de los requisitos establecidos por el cliente y la verificación de su funcionamiento. Por último, en el Capítulo 8 se genera un análisis del trabajo realizado junto con un plan de trabajos futuros para mejorar el funcionamiento de la plataforma.

2 | Marco Teórico

En el capitulo anterior se planteron los conceptos de Smart Grids, el Internet of Things (IoT) y el Internet of Energy (IoE). Ahora es cuando se ahonda un poco más en ellos y se explica su relevancia en el desarrollo de un HEMS.

2.1. ¿Qué son las Smart Grids?

Como se explica en [2], las Smart Grids son aquellas redes eléctricas que permiten la integración inteligente de las diferentes acciones de los usuario, con el fin de entregar energías sostenibles, económicas y seguras. Además, una Smart Grid busca entregar tecnologías innovadoras, junto con el monitoreo, control y comunicación inteligente, con el objetivo de:

- Reducir el impacto ambiental que produce el consumo eléctrico.
- Promover e informar a los consumidores de diferentes fuentes de suministro eléctrico.
- Entregar un servicio seguro y de confianza a los consumidores finales.
- Integrar de manera sencilla a la red eléctrica generadores de distintos tipos y baterías de diferentes tamaños.

¿Por qué Smart Grids?

Los Smart Grids permiten innovar en 4 aspectos significativos [2]:

1. Energías más limpias: plantas más eficientes, energías renovables no convencionales más accesibles.
2. Mejoras significativas en las tecnologías de almacenamiento.

3. Sistemas de comunicaciones inteligentes y autónomos.

4. Generación distribuida.

Esto lleva a tener hogares que se abastezcan de manera autónoma de electricidad, provenientes de fuentes de energía renovables con diferentes tipos de generadores -solares, eólicos, etc-, pudiendo incluso abastecer a otros consumidores o entregar la energía a las distribuidoras para disminuir los costos de consumo. También se podrá pensar en casas que cuenten con estaciones eléctricas que abastezcan a los autos eléctricos o incluso tener sistemas V2H (Vehicle to Home) en que los vehículos entreguen parte de la energía de sus baterías para alimentar a los dispositivos del hogar. Incluso se puede pensar en servicios que permitan al consumidor decidir cuándo, cuánto y a qué costo consumen la energía, logrando un mercado eléctrico elástico y que favorezca a todos sus integrantes.

2.2. Internet de las cosas (IoT)

Otro de los conceptos relevantes para la compresión de la solución propuestas es el del "Internet de las cosas", el que se explica en [3]. Con él se busca dar acceso a internet a los diferentes dispositivos que utilizamos en nuestro día a día como autos, electrodomésticos, sensores, entre otros. El fin de esto es lograr que los objetos intercambien datos por medio de internet para lograr automatizar algunas funciones de los dispositivos, y así liberar el requerimiento de una supervisión de personas.

El IoT es utilizado en diversos campos de trabajo debido a su versatilidad y fácil implementación dentro de la industria, permitiendo monitoreo y control de regadíos inteligentes, conocer la localización del ganado, detectar movimiento de personas dentro de un área, entre muchas otras cosas. Todo esto ha sido posible gracias al respaldo tecnológico que ha tenido el IoT. Algunas de las tecnologías relevantes son:

- **Acceso a tecnología de sensores de bajo costo y bajo consumo:** La industria de sensores cada vez hace sensores más asequibles y fiables, lo que favorece al desarrollo del IoT.
- **Conectividad:** Se ha creado una gran cantidad de protocolos para transferencia de datos entre cosas.

- **Plataformas de Cloud Computing:** La gran cantidad de proveedores de servicios en la nube permite escalabilidad en los proyectos de la industria, sin necesidad de controlarlo todo.
- **Machine Learning y Análisis:** El creciente campo del aprendizaje automático permite manipular la información recolectada por los objetos y realizar análisis sobre ella, ampliando los límites del IoT.
- **Inteligencia artificial conversacional:** Los avances en redes neuronales han llevado al procesamiento de lenguaje natural, generando asistentes personales digitales como Alexa, Cortana o Siri y se han vuelto atractivos, asequibles y fáciles de usar de manera doméstica.

Dentro de los campos a nivel industrial se encuentran:

- Fabricación Inteligente.
- Ciudades Inteligentes.
- Redes eléctricas inteligentes.
- Entre otros.

En el desafío se utilizó el IoT dentro del campo de las redes eléctricas inteligentes (Smart Grid) para conocer los valores de consumo de distintos electrodomésticos detectado por smart plugs, los que son enviados por telemetría hacia un nodo central para luego ser procesados y almacenados.

2.3. Internet de la Energía (IoE)

También es necesario conocer el concepto del internet de la energía (IoE), explicado en mayor profundidad en [4]. El internet de la energía busca que los consumidores sean conscientes del ciclo energético por medio de las TIC y el IoT, formando una cadena digital de energía basado en el cuidado de la misma.

Dentro del ciclo de la electricidad influyen diferentes actores y en diferentes procesos que se detallaran a continuación.

Los actores involucrados son: Consumidores/Productores, red eléctrica principal y las fuentes de energía renovable.

Respecto a las fuentes de energía renovables se busca incrementar el acceso a ellas. Con esta acción se podría:

- Generar soluciones de generación energética innovadoras y vanguardistas.
- Acelerar los procesos de generación de energía de manera local.
- Equilibrar la oferta y demanda de energía.
- Generar la integración de energías heterogéneas.

Todo esto debido a que se espera que la energía pueda fluir en múltiples direcciones: de la distribuidora al cliente y del cliente a la distribuidora o incluso a otro cliente.

Por otro lado, la distribuidora eléctrica procura mejorar la estabilidad de la red, lo que repercute en:

- Evitar cortes del suministro eléctrico.
- Disminuir tiempos de inactividad .
- Tener un mayor control de la calidad de la energía entregada por la red.

Esto ayuda a obtener un servicio continuo y con un flujo constante y sin fluctuaciones de energía hacia los distintos hogares.

Por último, la IoE enfocada en los productores y consumidores (desde ahora prosumers) se encarga de optimizar el uso y la gestión de los recursos distribuidos, los cuales representan todo tipo de energía generada a baja escala y localizada cercana a la fuente de consumo. Algunos ejemplos de fuentes de energías distribuidas son los paneles fotovoltaicos y las turbinas eólicas que se instalan en los hogares.

Dentro de la optimización y gestión de los recursos distribuidos se espera:

- Generar transacciones de energía peer-to-peer.
- Elaborar un sistema de comercio autónomo basado en microgrids.
- Controlar los recursos distribuidos de manera automática.

Con estos 3 actores se espera lograr un ciclo energético que contribuya a disminuir la carga que existe a la red eléctrica, y para eso las microgrids son la piedra angular del IoE que permitirán abastecer gran la demanda energética que los consumidores generan.

2.4. Monitoreo de carga

Al centrarse en el desarrollo de una plataforma de monitoreo de cargas es inevitable que surjan las preguntas: ¿Qué es el monitoreo de carga? y ¿Como se realiza?.

El monitoreo de cargas se encarga de detectar el consumo de los electrodomésticos en hogares inteligentes permitiendo que los usuarios puedan gestionar y manejar sus dispositivos para que funcionen de forma autónoma y óptima, siendo el primer paso para lograr la eficiencia energética y la concientización de los usuarios, como se explica en [5]. En este articulo también se responde la segunda pregunta: ¿Como se realiza?. Para ello se puede realizar monitoreo de carga mediante 2 métodos: Monitoreo de carga no intrusiva -NILM por su sigla en inglés- y monitoreo de carga intrusivo -ILM por su sigla en inglés-.

En NILM se utiliza un enfoque de monitoreo basado en software en donde se utiliza un único punto de detección de consumo energético. Esta solución ha sido ampliamente estudiada durante el último período de tiempo por ser una manera más atractiva de monitorear, debido a su bajo costo de implementación -ya que solo se monitorea en 1 punto-, pero a pesar de esto, el método NILM muestra resultados con menor precisión y una mayor dificultad de implementación en escenarios reales.

Por otro lado, ILM es un método basado en hardware donde se distribuyen los puntos de medición con el fin de obtener mayor fiabilidad de los datos y eficiencia. Esta alternativa es relativamente más cara que su contra parte, debido a que se necesita un mayor número de dispositivos para su implementación. Estos dispositivos son submedidores o nodos sensores que se instalan cerca de los electrodomésticos objetivos -generalmente entre el dispositivo y la toma de corriente-. Sin embargo, el ILM está tomando fuerza, debido a que permite el control del funcionamiento de dispositivos y electrodomésticos, además de medir su consumo, sin mencionar que permite obtener el perfil de consumo de los electrodomésticos

mediante el etiquetado de los datos.

Dado lo anterior, el equipo de trabajo ha decidido utilizar el método ILM para la toma de datos del consumo, y es el método que se utilizó a lo largo de todo el diseño y desarrollo de la solución propuesta.

2.5. Arquitectura de Smart Home

En general, la configuración del sistema del hogar inteligente se muestra en la Figura 2.1. La configuración del sistema puede clasificarse en diferentes categorías como: hogar autónomo, hogar conectado a la red, hogar conectado a la red con sistema híbrido de energía renovable (HRES por su sigla en ingles), hogar conectado a la red con HRES y sistema de almacenamiento de energía (ESS por su sigla en ingles).

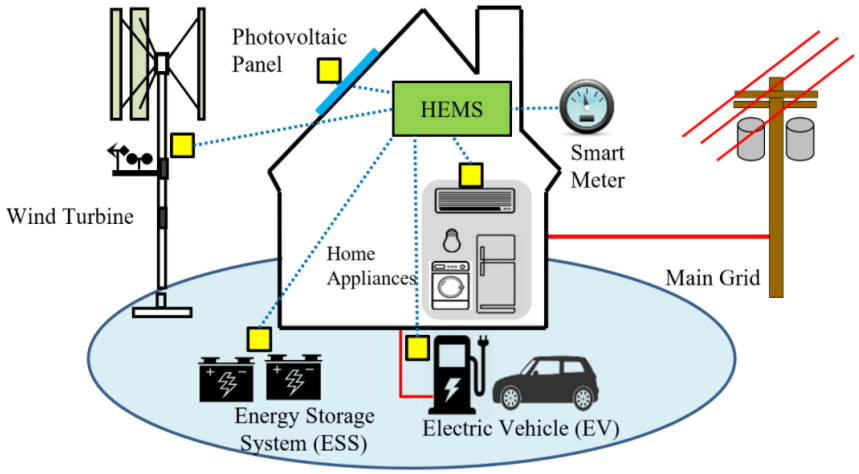


Figura 2.1: Diagrama esquematico de HEMS en el Smart House

Tipo 1: Hogar autónomo. Esta configuración se suele denominar fuera de la red. La mayoría de los hogares autónomos incluyen fuentes de energía renovable como la fotovoltaica y/o los aerogeneradores.

Tipo 2: Hogar conectado a la red. En esta configuración, el hogar está conectado a la red sin RES o ESS. El hogar pretende optimizar el consumo de energía en función de diferentes criterios como el tiempo de uso (TOU por su sigla en ingles), el precio en tiempo real y el confort del usuario.

Tipo 3: Hogar conectado a la red con HRES. En esta configuración, el hogar está conectado a la red con RES como la fotovoltaica y/o los aerogeneradores.

Tipo 4: Hogar conectado a la red con HRES y ESS. Esta configuración admite la conexión a la red y la integración de las RES con los ESS.

El desarrollo del desafío se centró en el **Tipo 4: Hogar conectado a la redc on HRES y ESS.**

2.6. Funciones de Smart Home

En la Figura 2.2 se muestran las principales funcionalidades de los Smart HEMS, que incluyen el monitoreo, el registro, el control, la gestión y la alarma [6]. Estas se resumen en Figura 2.2.

Figura 2.2: Funciones principales de los Smart HEMS

Entrando en más detalle sobre cada una de las funciones se tiene:

2.6.1. Monitoreo

El monitoreo ofrece un fácil acceso a la información en tiempo real sobre el consumo de energía y permite a los usuarios centrarse en el ahorro de electricidad. También puede proporcionar servicios de visualización de los modos de funcionamiento y el estado energético de cada electrodoméstico.

2.6.2. Registro

El registro consiste en recopilar y guardar la información de los datos sobre el uso de la electricidad de los electrodomésticos, las generaciones de los DER y el estado del almacenamiento de energía. Este servicio también contiene un análisis de la respuesta a la demanda para conocer los precios en tiempo real de la red eléctrica.

2.6.3. Control

Hay dos tipos de control: el control directo y el control remoto. El control directo se implementa tanto en el equipo como en el sistema de control; mientras que el control remoto significa que los clientes pueden acceder en línea para supervisar y controlar los patrones de uso de los dispositivos en el hogar a través de un ordenador personal o un teléfono inteligente desde el exterior.

2.6.4. Gestión

La gestión es la función más importante de HEMS para mejorar la optimización y la eficiencia del uso de la energía eléctrica en la casa inteligente. Abarca una gama de servicios que incluyen el servicio de gestión de sistemas de energía renovable, el servicio de gestión de almacenamiento de energía, el servicio de gestión de electrodomésticos y servicio de gestión de vehículos eléctricos y baterías.

2.6.5. Alarma

Genera una alarma y la envía al centro HEMS con información sobre la localización de fallos, por ejemplo, si se detecta alguna anomalía.

2.7. Diseño de Hogar

Para generar una estructura del hogar se diseñó un prototipo modular, en donde se establece la cantidad de smart plugs por habitación. Estos módulos se dividieron entre recintos habitables y no habitables como se muestra en la Figura 2.3 y Figura 2.4

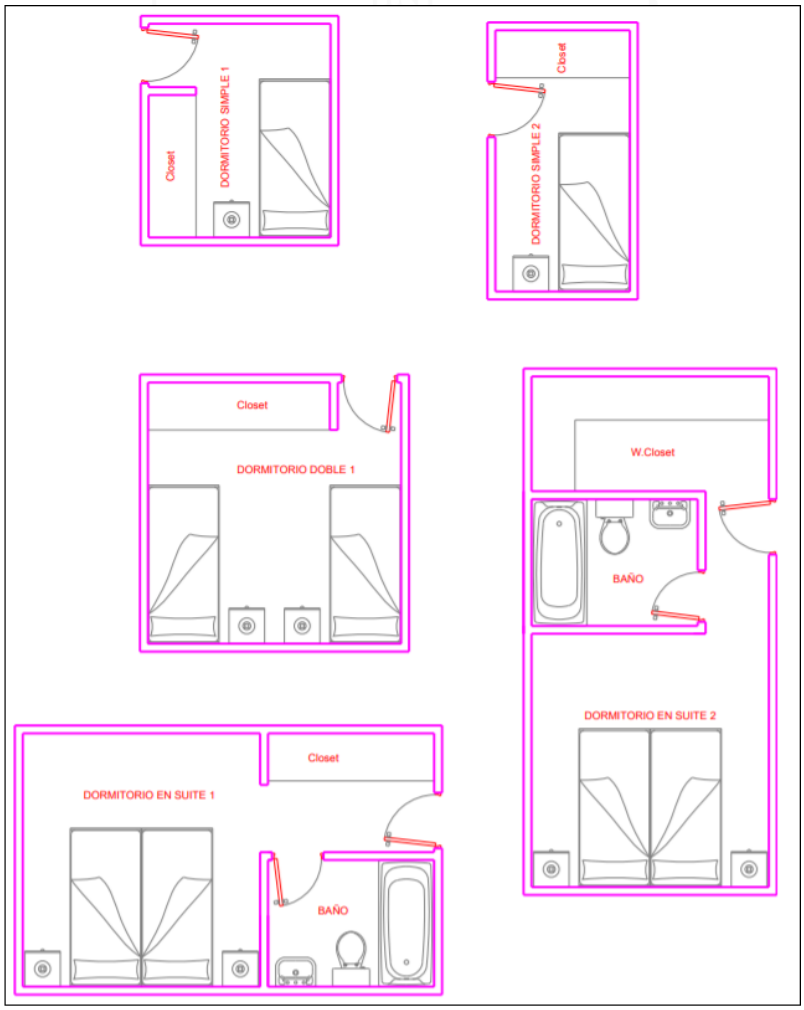


Figura 2.3: Diseño modular de recintos habitables.

Con esto en mente, el prototipo de hogar a desarrollar es una conjunción de habita-

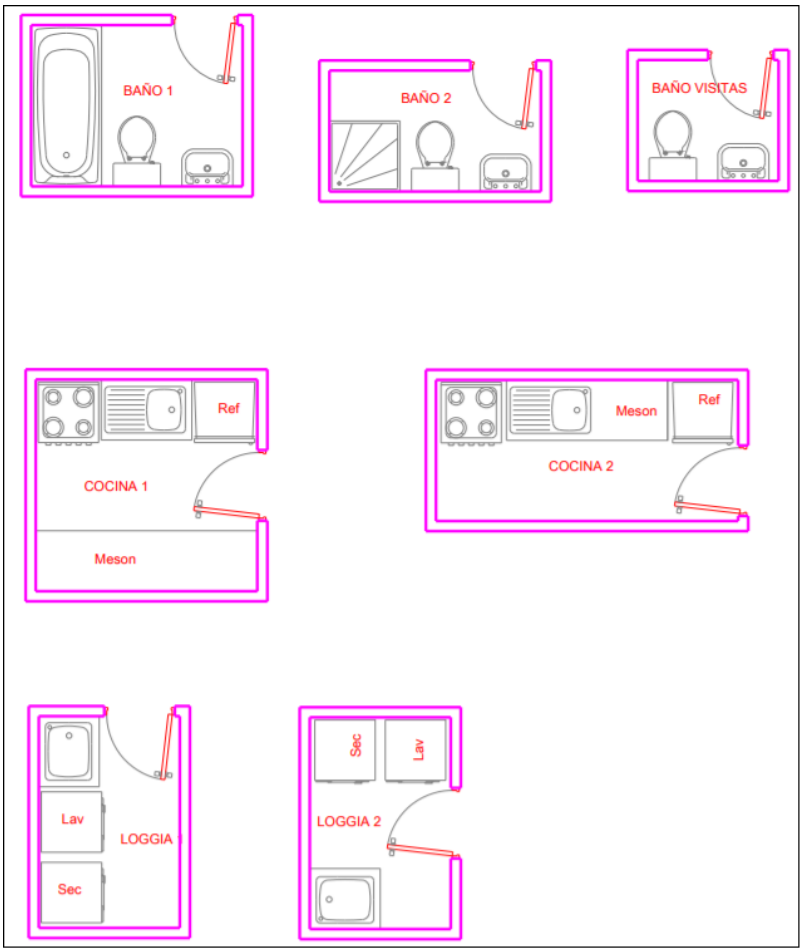


Figura 2.4: Diseño modular de recintos no habitables.

ciones las cuales tienen definida la cantidad de smart plugs a utilizar por recinto.

El diseño de estos recintos fueron hechos en base a las leyes chilenas de habitabilidad, las cuales están definidas en [7]. Sin embargo, la casa de testing cuenta con 5 smart plugs distribuidos en diferentes habitaciones, monitoreando los siguientes electrodomésticos objetivos: Refrigerador, TV Dormitorio, Calefactor, Lavadora, TV Living.

3 | Estado del arte

Una vez comprendidos los principales conceptos asociados al monitoreo energético de los electrodomésticos, se procede a mostrar el proceso investigativo realizado, con el fin de obtener referencias y extraer información realizada por diferentes autores para poder generar una solución que permita el monitoreo de energía en un hogar.

En [8; 9], los autores clasificaron el sistema de gestión de la energía (EMS) en cuatro grupos: sistema de gestión de la energía del hogar (HEMS), sistema de gestión de la energía del edificio (BEMS), sistema de gestión de la energía de la fábrica (FEMS), sistema de gestión de la energía de la comunidad (CEMS), como se muestra en la Figura 3.1.

El CEMS desempeñan un rol importante en toda la comunidad mediante la comunicación de otros sistemas dentro y fuera de ella. Además, este calcula el balance energético óptimo y -en caso de emergencia- controla el flujo de energía hacia las entidades inferiores para cubrir la demanda energética.

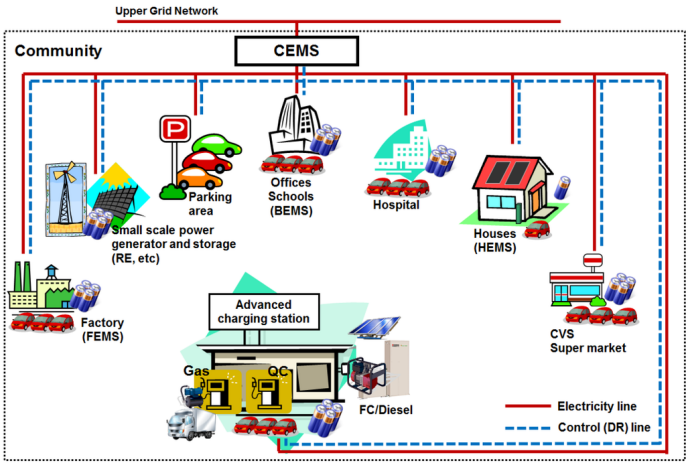


Figura 3.1: Diagrama esquemático CEMS [8]

La Figura 3.2 muestra la integración de los vehículos eléctricos como apoyo al CEMS a pequeña escala. El EMS controla la demanda y el suministro de electricidad a través de la información procedente de diferentes fuentes como: la carga, la información meteorológica, la empresa de servicios públicos y el sistema de información de vehículos (VIS).

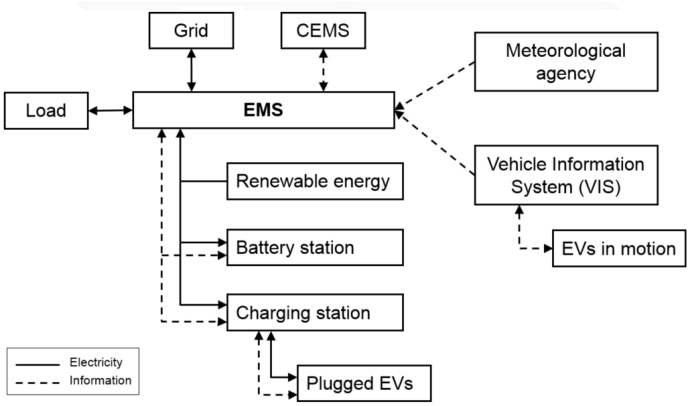


Figura 3.2: Diagrama de integración de los VE con los ME [8]

Como el vehículo eléctrico tiene una gran cantidad de almacenamiento de energía, puede hacer que el EMS sea más eficaz.

En [9], se propuso un control predictivo del modelo de la batería de almacenamiento del vehículo eléctrico, basado en un modelo autorregresivo y en la optimización del enjambre de partículas en un sistema de gestión de la energía doméstica. El objetivo principal era controlar el estado del almacenamiento del VE para minimizar el coste de la electricidad del HEMS y evaluar el beneficio de la compañía eléctrica. Se consideraron tres casos "sin PV y sin red eléctrica", "con PV y sin red eléctrica" y "con PV y con red eléctrica". El modelo a utilizar es el llamado "AR model" [10], en donde se controla el almacenamiento de energía de un VE, considerando viajes, ya que la función principal de un automóvil es transportar a los usuarios. En él se consideraron dos perfiles con diferentes acciones, incluyendo "perfil de desplazamiento: ir a la oficina, trabajar y volver a casa" y "perfil de ama de casa: llevar a un niño a la guardería, hacer la compra y buscar a un niño en la guardería".

La manera en que solucionan el problema de optimización es mediante el uso de PSO -Particle Swarm Optimization-, considerando 3 casos:

- VE sin panel fotovoltaico (PV) y sin red eléctrica

- VE con PV y sin red eléctrica
- VE con PV y con red eléctrica

El termino “con/sin red eléctrica” hace referencia a cuando el VE se puede cargar/-descargar en la oficina del usuario, y con esto obtener ganancias de la venta de energía a la empresa para la cual trabaja el dueño del VE.

Como resultado del problema de optimización se determinó que el mejor de los casos para reducir el costo asociado al consumo eléctrico del hogar es "VE con PV y red eléctrica".

En [11], al igual que el artículo anterior, propone un sistema combinado de gestión de la energía doméstica que incluye la tecnología Vehicle-To-Home (V2H) o Home Centralized Photovoltaic (HCPV). El enfoque propuesto busca controlar la demanda de energía en el hogar mediante la programación óptima de los aparatos de automatización, contemplando las diferentes condiciones climáticas que afectan la producción de energía de los PV.

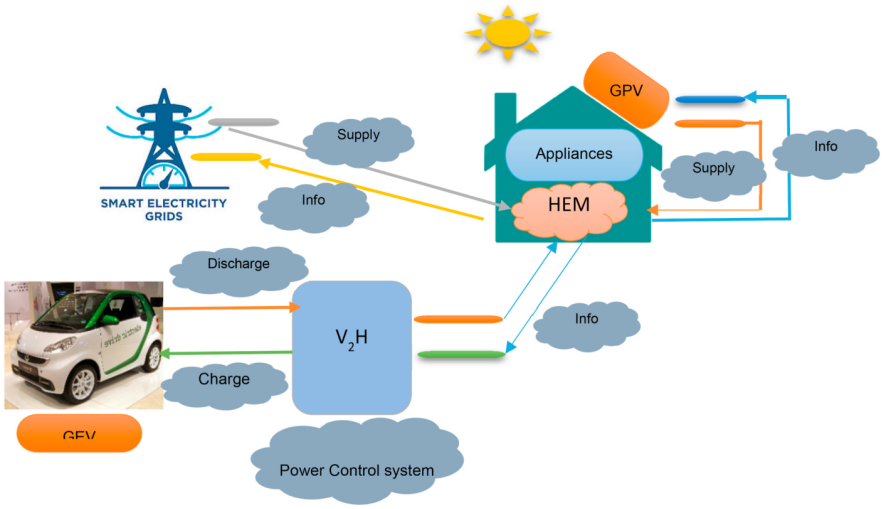


Figura 3.3: Diagrama de concepto V2H-HCPV [11]

Sin energía adicional de la red, el diseño propuesto de HCPV cubre la demanda de electricidad del hogar en tiempo soleado y nublado. Por otro lado, la combinación de PV y electricidad transmitida por V2H es suficiente para cubrir la demanda de carga doméstica con tiempo nublado. En este sentido, se diseña y discute un Sistema de Gestión de Energía Doméstica programado con varias restricciones para realizar operaciones correctas del sistema y satisfacer la demanda de carga.

El algoritmo propuesto intenta caracterizar y percibir las circunstancias de la fuente utilizando la demanda de energía.

Los resultados obtenidos demuestran que la tecnología de funcionamiento de H2V puede reducir eficazmente la demanda de energía bajo la volatilidad de la carga solar.

En [12] se investigó un algoritmo de control para el sistema de gestión de la energía en el hogar (HEMS) para supervisar y programar los aparatos eléctricos que se aplican en cualquier casa convencional.

El objetivo es reducir el consumo de electricidad y el costo de la misma en lugares con un modelo de precios basado en el tiempo de uso (Time Of Use). Para ello se realizó la implementación de hardware en una casa de prueba utilizando un sistema de control propio, Smart Plugs -diseñados y fabricados por los autores- y un paneles fotovoltaicos como fuente suplementaria de energía.

Los Smart Plugs utilizados son de 3 tipos, capaces de medir el consumo de electrodomésticos, la temperatura de aparatos térmicos, o el estado de carga y potencia de salida de las fuentes de energía.

En la arquitectura del sistema, el consumo global de los aparatos eléctricos que funcionan en la red está limitado por un algoritmo, y se mantiene por debajo de un determinado valor que se fija en función del consumo total del edificio y de la capacidad de la batería.

Este algoritmo propuesto se centra en la optimización del gasto monetario asociado al consumo eléctrico, dado que la electricidad en horas punta tiene un costo más elevado, por lo que se intenta gestionar el funcionamiento de los aparatos eléctricos para que funcionen con batería o con electricidad de la red dependiendo de la hora como se muestra en **??**.

Las baterías se recargan a través del controlador de suministro de los paneles solares durante el día y de la red durante la noche, cuando el precio de la electricidad es más barato. El algoritmo puede priorizar los electrodomésticos en función de su prioridad de uso -que se define de antemano- para optimizar el consumo de electricidad basado en el límite predefinido de la demanda.

La premisa incluía 14 electrodomésticos que funcionaron durante tres meses para

su verificación. Los resultados muestran que el sistema implementado con el algoritmo propuesto es capaz de reducir el precio de la electricidad significativamente, hasta un 15 % por día en comparación con el consumo habitual.

En [13] , los autores desarrollaron un prototipo para un sistema de comunicación inalámbrica de HEMS en una casa/edificio inteligente mediante la implementación de una interfaz web entre un gestor de energía y las cargas pesadas. El prototipo del sistema ha sido diseñado, construido y probado en el laboratorio. Además, se ha analizado el canal entre el consumidor y el gestor de energía mediante simulaciones con respecto al rendimiento, el retardo de extremo a extremo y el retardo de ida y vuelta. En la Figura 3.4 se muestra el diagrama esquemático del sistema de gestión de energía doméstica inalámbrica (WHEM).

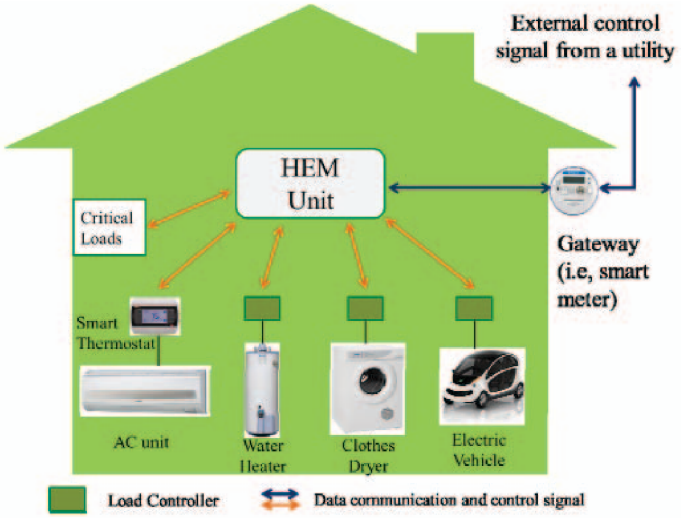


Figura 3.4: Vista previa del sistema WHEM [13]

Los autores dividieron su solución en 3 bloques: el módulo de control de la carga, que se encarga de gestionar y realizar las mediciones de consumo de los dispositivos; la unidad de comunicación, que facilita el intercambio de información; y el gateway, que es el enlace de comunicación entre la unidad de gestión doméstica y el gestor de energía en el lado de la red inteligente.

El sistema WHEM se compone de los 3 módulos explicados con anterioridad en donde cada uno sirve para lo siguiente:

- **Control de carga:** Proporciona monitorización, control y comunicación.

- **Comunicación:** Establece la comunicación entre los controladores de carga y la unidad HEM a través de diferentes tecnologías de comunicación como PLC, WiFi, Bluetooth y ZigBee
- **Gateway:** Permite la comunicación bidireccional entre el lado de la empresa de servicios públicos y el sistema HEM.

Para la implementación de WHEM, el módulo de control de carga consiste en:

- Un microcontrolador (ATmega329)
- Un módulo de RF (CC1101) -que opera a una frecuencia de 433 MHz-.
- Un relé DPDT de 10A

La unidad HEM se ha implementado con una Raspberry Pi, que se comunica con los sensores a través de un enlace de radio formando una topología en estrella, que se ha configurado como un servidor web para incluir los datos de la carga, así como enviar las señales de control ON/OFF para las cargas conectadas.

Para medir el rendimiento y la fiabilidad de la red, se simuló la red de comunicación utilizando el simulador OPNET. El modelo a simular consta de 5 nodos, un concentrador y un servidor y se consideraron diferentes métricas, como el rendimiento de la red y el retardo de extremo a extremo.

En [14] se propone un método eficiente de almacenamiento de datos en bases de datos NoSQL, con el fin de supervisar el estado de los dispositivos de un hogar en tiempo real. Además se desarrollo una aplicación para la gestión de energía doméstica (HEM), la cual se utilizó para evaluar el método propuesto.

Los autores consideran el uso de una base de datos NoSQL distribuida, destacando 3 características importantes, la consistencia de los datos, la disponibilidad, y la tolerancia a particiones. La idea de utilizar una base de datos distribuida es disminuir los tiempos de solicitud de grandes volúmenes de datos, recurriendo a varios nodos de la base al mismo tiempo. La aplicación para teléfonos inteligentes implementada incluye las siguientes tres características: recordatorio, visor de costes y analizador de costes, como se muestra en la Figura 3.5.

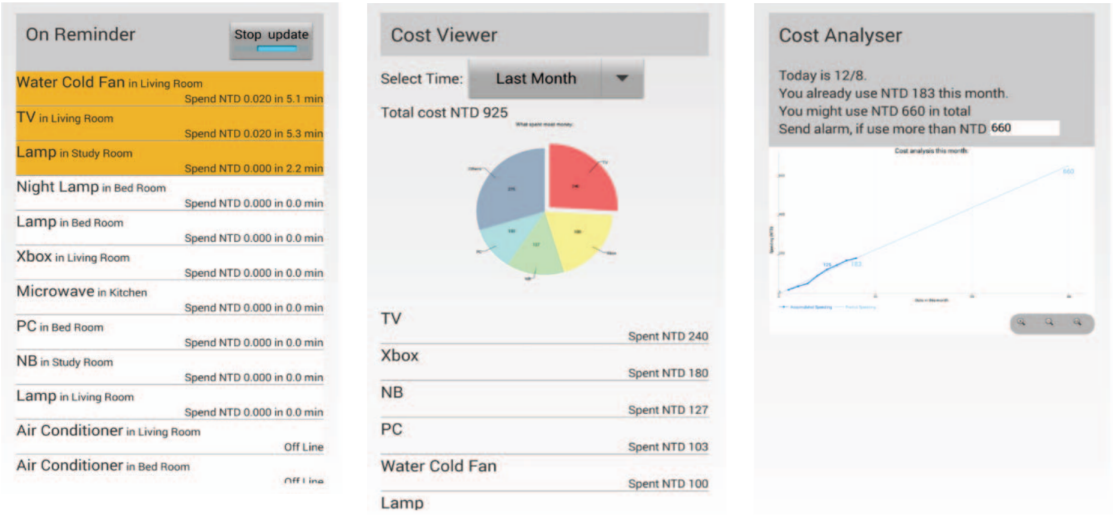


Figura 3.5: Funciones de aplicación móvil [14]

El problema que enfrentan es que al almacenar datos de series de tiempo -como las que se obtienen al realizar mediciones constantemente-, la información cercana en el tiempo suele almacenarse dentro de los mismos nodos, desperdiciando el provecho que se puede obtener de la arquitectura distribuida.

La solución implementada fue aplicar una función de hash a la llave de los registros para desagregarlos, recuperando las bondades de una base de datos distribuida.

La evaluación se hizo sobre un clúster HBase que construyeron los autores como dataset de pruebas. El método propuesto mostró resultados prometedores, entregando la información solicitada a más del 95 % de los usuarios en 0.5s mientras otros cientos o miles de aplicaciones solicitaban información al mismo tiempo, lo que permite satisfacer al menos cientos de aplicaciones móviles simultáneamente.

En [15] se analiza un sistema de monitoreo de energía en tiempo real, basado en IoT para el control y la supervisión dentro de una industria, donde se ha decidido establecer un sistema eficiente de monitoreo de la energía para analizar su utilización diaria, como paso preliminar de las actividades de conservación de esta. En este sentido, se consideran los sistemas de monitorización de energía basados en IoT. Muchos tipos de dispositivos IoT como Arduino, Raspberry Pi, Intel Edison, Mediatek linkit one, NVIDIA Jetson Nano, etc. están disponibles para realizar esta tarea.

En comparación con otros métodos de adquisición de datos, los sistemas basados en Raspberry Pi resultan ser la mejor opción para realizar la tarea en base a sus criterios técnicos, por su bajo coste y por ser una tecnología altamente fiable para la monitorización del consumo energético de la industria. Por lo tanto, se establece un sistema de monitorización de energía basado en Raspberry Pi con los contadores de energía existentes en la industria de los equipos de conmutación. Esta se utiliza con el entorno de desarrollo Node.js para recoger los datos (varios parámetros eléctricos) de los contadores de energía existentes en la industria y almacenarlos localmente en una base de datos InfluxDB, para acceder a ellos a través del ordenador portátil o del teléfono móvil utilizando Grafana.

El sistema de monitorización es útil para la industria para entender el patrón de energía del día a día, que es esencial para facilitar las medidas de conservación de energía para minimizar su consumo.

En [16] se estudia el impacto que tiene Telegraf - un software de recopilación de métricas-. Para ello realizaron varios experimentos para estudiar cómo afecta Telegraf al sistema base en un servidor de centro de datos y un nodo IoT.

Dentro de las pruebas realizadas se evaluaron distintas configuraciones de Telegraf evaluando diferentes escenarios, en donde se varió el orden de magnitud del intervalo medido -10ms, 100ms, 1s, 10s y 100s- y el número de métricas recogidas -1, 5 y 10 métricas-. Estas últimas están correlacionadas en la Tabla 3.1. Además, se utilizó InfluxDB 1.7.6 como el servicio de almacenamiento que recogió los datos de Telegraf, y reside en otra máquina en una red diferente del servidor o nodo IoT que tiene Telegraf instalado.

Para el escenario del servidor del centro de datos, utilizaron:

- Una instancia de Google Cloud Compute con CentOS7 (centos-7- v20190213). Esta tiene:
  - 1 vCPU (Intel Skylake o superior).
  - 3,75 GB de memoria.
  - 30 GB de disco de almacenamiento.

Para el escenario del nodo IoT, utilizaron:

Tabla 3.1: Métricas recolectadas en cada configuración [16]

Telegraf Inputs	Number of Metrics		
	1	*5*	*10*
inputs.cpu	X	X	X
inputs.disk		X	X
inputs.diskio		X	X
inputs.mem		X	X
inputs.net		X	
inputs.processes			X
inputs.swap			X
inputs.system			X
inputs.interrupts			X
inputs.kernel			X
inputs.kernel_vmstat			X

- Una Raspberry Pi 3 Modelo B con Ubuntu 19.10. Esta tiene:
  - 1 CPU de cuatro núcleos a 1,2GHz (ARM Cortex A53).
  - 1GB de memoria RAM.
  - 1 tarjeta de memoria MicroSD de 32GB de disco de almacenamiento.
  - 1 sensor de temperatura DS18B20 conectado a la Raspberry Pi.

Para cada uno de los casos se comparó el uso de Telegraf en el sistema, frente a un sistema de las mismas características sin Telegraf instalado.

Los resultados mostrados en la tabla II y III de [16] le permiten a los autores determinar que Telegraf es ligero y adecuado para servir como agente de monitorización en tiempo real en ambos escenarios. Sin embargo, para situaciones que requieren una monitorización de muy alta frecuencia, es decir, una recogida de métricas más rápida que 1s, este sistema puede ser demasiado intensivo en recursos para el despliegue de nodos IoT.

En [17], los autores propusieron un marco HEMS de software que es diferente de las soluciones existentes y no requiere ningún hardware adicional, lo que disminuye

significativamente el costo que implica la instalación de medidores en el hogar.

La solución propuesta ha sido diseñada e implementada en el sistema operativo T-Kernel, utilizado en las aplicaciones embebidas de los clientes. El consumo de energía se estima a partir de la CPU que se correlaciona con el porcentaje de inactividad.

En la Figura 3.6 se muestra una visión general de la arquitectura HEMS. En ella se puede observar que la capa “Software HEMS Aggregator” tiene un color diferente. Esto es debido a que los autores explican que es esta capa la más importante del sistema, ya que se encuentra continuamente recogiendo y agregando el consumo de los periféricos de los dispositivos. Por otro lado, la capa “CoAP” es la encargada de enviar la data a un servidor HEMS central. Este último se encarga de sumar la información de los dispositivos para luego desplegarla al usuario.

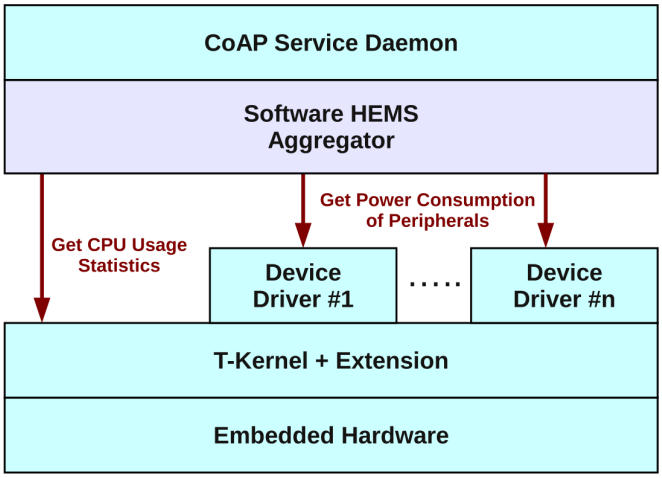


Figura 3.6: Framework de Software HEMS [17]

En [18], los autores desarrollaron y evaluaron un sistema de “experiencia de usuario como servicio” en su sistema de gestión de la energía doméstica.

Los estudios convencionales consideran la visualización del consumo de energía eléctrica mediante: tablas, gráficos y números. En ellos no se considera realizar acciones en beneficio de reducir el consumo de energía eléctrica de los consumidores. Tomando lo anterior en cuenta, el sistema desarrollado consiste en la gamificación y el ciclo Planificar-Hacer-Verificar-Actuar (PDCA, por su sigla en ingles), donde se utiliza el "perfil"del registro de vida en un juego, mientras que el ciclo PDCA permite una utilización continua

del juego por parte del usuario.

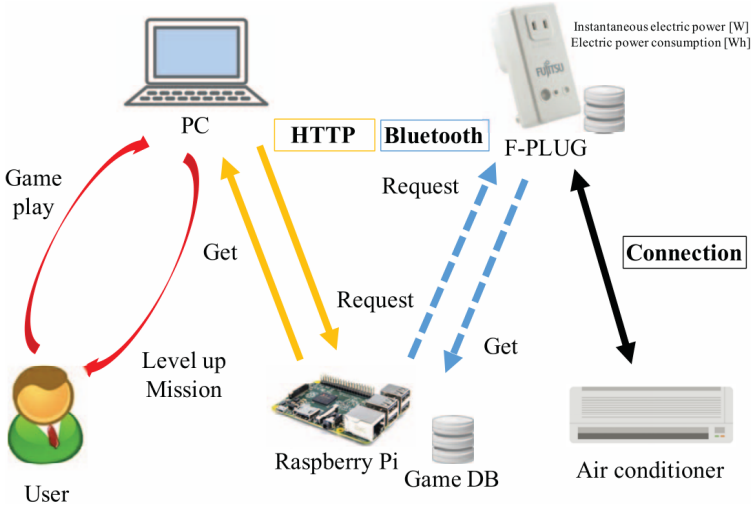


Figura 3.7: Diagrama del prototipo de sistema de gamificación [18]

En el sistema prototipo, el usuario se conecta a un servidor de juegos alojado en una Raspberry Pi a través de un PC utilizando HTTP. Se ha considerado un acondicionador de aire debido al alto consumo de energía, como se muestra en la Figura 3.7, donde se ha utilizado un Smart Plug para la adquisición del consumo de energía eléctrica. Por otro lado, La comunicación entre el Smart Plug y el servidor de juegos se realiza vía Bluetooth.

La Figura 3.8 muestra la aplicación web en la que el usuario es un personaje que se enfrenta a sus electrodomésticos. Al derrotar a los electrodomésticos, los usuarios ganan experiencia e ítems del juego al reducir el consumo de energía durante ciertas franjas horarias. Esta solución es bastante innovadora y permite a los usuarios conocer mejor el consumo asociado a sus electrodomésticos, volviéndolos conscientes de la demanda energética generada periódicamente, y así generar un plan de acción para reducir este consumo en caso de estar superando las cifras de su consumo medio mensual.

Dentro de las referencias del documento se analizó una aplicación creada por “Toshiba Feminity Club” [Figura 3.9] en la cual se observó un diseño basado en la personificación de un usuario en un personaje de la plataforma y la administración de los dispositivos del hogar por medio de una plataforma atractiva y personalizable, pudiendo modificar los ambientes del hogar y el estado de los electrodomésticos, los cuales esta separados por habitaciones.

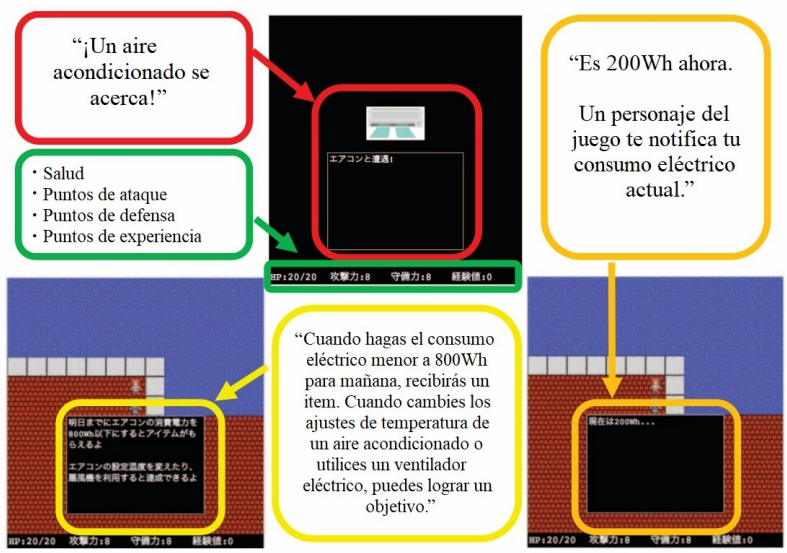


Figura 3.8: Aplicación Web de Gamificación [18]

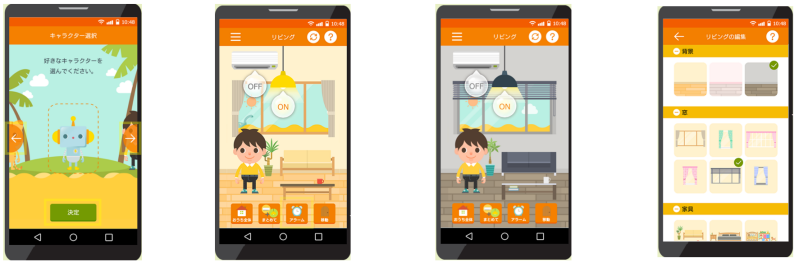


Figura 3.9: Aplicación móvil Toshiba Feminity Club [19]

Resumen

Con toda la información recopilada se logra confeccionar la tabla de resumen mostrada en Tabla 3.2 destacando las capas trabajadas por cada uno de los autores. Además, se observa cuales son las capas que se abarca en la solución presentada en el documento.

Tabla 3.2: Resumen de estado del arte

Referencia	Capa Física	Capa de Comunicación	Capa de Middleware	Capa de Aplicación
[8]	✓	x	x	✓
[9]	✓	x	x	✓
[10]	✓	x	x	✓
[11]	✓	x	x	✓
[12]	✓	x	x	✓
[13]	✓	✓	x	✓

[14]	x	x	✓	x
[15]	✓	x	✓	✓
[16]	x	x	✓	x
[17]	x	x	x	✓
[18]	x	x	x	✓
iHEMS	✓	✓	✓	✓

4 | Arquitectura IoT de la Plataforma

Las mediciones del consumo es la capa más baja del sistema y la base de toda la arquitectura del sistema. Pero, si es la base de todo, ¿Cuales son el resto de capas?.

Para responder a esa pregunta es necesario visualizar nuevamente el articulo [5, Figura 2], en donde se visualiza una arquitectura para el monitoreo intrusivo de cargas. A partir de este diagrama se generó una arquitectura compacta a utilizar en este documento, con el fin de simplificar el modelo y lograr una mayor comprensión del mismo. [Figura 4.1]

4.1. Capa Física

En esta capa se produce la recolección de los datos del consumo de los electrodomésticos que se enviarán con posterioridad a las capas superiores. Contiene tanto los dispositivos físicos (electrodomésticos, vehículos eléctricos y calefacción), como los dispositivos de percepción, encargados de la adquisición de los datos.

Dentro de los dispositivos de percepción de la arquitectura se encuentran los smart plugs y la smart lighting. Los smart plugs son la base de la propuesta de solución, ya que estos dispositivos se encargan de detectar y empaquetar el consumo de los electrodomésticos objetivos. Con ellos es posible medir características dependientes del tiempo y/o basadas en la frecuencia. Las características típicas dependientes del tiempo son la potencia activa (P), la potencia reactiva (Q), la tensión (V), la corriente (I) y las trayectorias V-I. A partir de estas características, se pueden obtener otras, como la potencia compleja y la potencia aparente. Sin embargo, estos dispositivos tienen un factor critico que viene dado por las limitaciones tecnológicas, y es el tiempo de muestreo [5]. Este tiempo es variable y dependiente del

Figura 4.1: Diagrama de arquitectura para visualización de consumo

modelo especifico de smart plug a utilizar.

4.2. Capa de Comunicación

En esta capa se produce la integración y comunicación entre dispositivos. Aquí es donde se definen las tecnologías de comunicación y los protocolos a utilizar. La elección de ellos depende netamente del objetivo al cual se apunta.

Como se observa en [20] y [5], los servidores donde se procesan y almacenan los datos se encuentran fuera de las Home Area Network (HAN) y en el extremo de la red, por lo que las tecnologías de mayor utilización en este tipo de arquitecturas son las tecnologías inalámbricas de corto alcance como Bluetooth, Wi-Fi o Zigbee, además de integrar un nodo central o gateway que es el intermediario entre la HAN y la WAN -lugar donde se

encuentran los servidores-.

4.3. Capa de Middleware

Esta capa es la encargada de recopilar los datos provenientes de capas anteriores con el fin interactuar entre los dispositivos IoT y la aplicación. Los requisitos de esta capa son muy elevados, contemplando: la abstracción de la arquitectura, la gestión de las funciones, el diseño de la programación y la implementación. La capa de middleware no sólo oculta las complejidades de las capas inferiores, sino que también proporciona servicios de datos para que las capas superiores cumplan fácilmente sus objetivos funcionales. Por lo tanto, esta capa es la más difícil de diseñar, ya que es independiente de la aplicación y se centra más en los datos. Además, debe tener en cuenta los requisitos funcionales, como la gestión de datos, el almacenamiento de datos, el procesamiento de datos (análisis de big data, análisis de datos en tiempo real, análisis de datos profundos con IA), y los requisitos no funcionales, como la escalabilidad, la seguridad, la disponibilidad, la fiabilidad, la entrega de datos en tiempo real, la privacidad, entre otros [21].

Algunos de los servicios estudiados para el desarrollo de esta capa en la solución son: DigitalOcean, AWS, Google Cloud. Estas plataformas nos permiten almacenar los datos provenientes de los smart plugs y le proveen al equipo la posibilidad de escalar los servicios en un futuro.

4.4. Capa de Aplicación

La capa de aplicación es aquella que se enfoca en los servicios ofrecidos a los usuarios. Es por esto que dentro de esta capa se pueden agregar diferentes tipos de soluciones relacionadas con el monitoreo de carga intrusivo (ILM).

Dentro de las aplicaciones más utilizadas para ILM esta la comprensión del consumo energético a nivel local y global, la evaluación y simulación de entornos NILM, el reconocimiento de la actividad humana y la localización de aparatos [5]. Los detalles relativos a cada una de estas aplicaciones se pueden encontrar en [22]. Este proyecto se centra en la comprensión del consumo energético a nivel local, buscando generar conciencia del uso de

la energía en los usuarios, y a la vez generar un conjunto de datos que puedan alimentar un algoritmo de machine learning para la detección de electrodomésticos, y con ello reconocer las actividades del día a día de las personas. Cabe mencionar que los datos a desplegar por esta capa son los obtenidos por los submedidores asociados a sus electrodomésticos objetivos, por lo que los datos obtenidos deben ser etiquetados de alguna manera.

4.5. Requisitos del Sistema

Antes de comenzar a describir las tecnologías que se utilizaron para la resolución del problema, es necesario conocer los requisitos descritos por el cliente para obtener una plataforma con las características necesarias para el funcionamiento de su prototipo, además del perfil de usuarios esperados que utilizarán el sistema.

4.5.1. Requisitos Funcionales

Los requisitos funcionales definen el comportamiento del sistema, es decir, describen lo que este debe hacer. Esto desde el punto de vista de interpretar lo que espera el usuario objetivo de la plataforma. Con esto en mente los requisitos funcionales definidos son los siguientes:

RF1 El smart plug debe ser capaz de medir la energía que consumen los electrodomésticos.

RF2 El smart plug debe enviar la información de cada medición al gateway.

RF3 El gateway debe proveer conectividad a los dispositivos.

RF4 El gateway proporciona conectividad segura por medio de autenticación y posibilidad de configuración remota.

RF5 El gateway envía los datos a un servicio en la nube con los metadatos necesarios para otorgar temporalidad a la información (fecha y hora de medición).

RF6 Los paquetes de información son recibidos, procesados, almacenados y estandarizados en un servicio en la nube.

RF7 Se disponibiliza endpoints de una API para acceder a los datos desde la nube.

RF8 La plataforma permite el ingreso de usuarios mediante autenticación.

RF9 La plataforma permite la creación de cuentas para dueños de casa y administradores.

RF10 La plataforma accede a los datos en la nube correspondientes al tipo de privilegios para su posterior visualización.

RF11 La plataforma permite al usuario dueño de casa agregar o desagregar dispositivos a la red para su monitoreo.

RF12 La plataforma permite al usuario dueño de casa ordenar los dispositivos a través de habitaciones.

RF13 La plataforma permite al usuario dueño de casa navegar por los dispositivos agregados en ella para visualizar sus mediciones.

RF14 La plataforma permite al usuario dueño de casa modificar y eliminar dispositivos y habitaciones.

RF15 La plataforma notifica al usuario dueño de casa a los usuarios cuando se agrega, desagrega o se desvincula un dispositivo electrónico.

RF16 La plataforma permite al administrador de una comunidad agregar o desagregar hogares para su monitoreo.

RF17 La plataforma permite al administrador de una comunidad visualizar el consumo general de energía de cada hogar.

RF18 La plataforma permite extracción de datos en formato csv.

4.5.2. Requisitos no Funcionales

Los requisitos no funcionales del sistema definen atributos de calidad por los cuales el usuario juzga la operación del sistema. Para esto se definieron los siguientes:

RNF1 Los dispositivos electrónicos deben estar conectados constantemente a los smart plugs.

RNF2 La disponibilidad de la red entregada por el gateway debe tener un uptime del 100 %.

RNF3 El gateway debe tener conectividad a internet todo el tiempo.

RNF4 El gateway debe tener comunicación con la nube todo el tiempo.

RNF5 La plataforma debe ser amigable, responsiva, auto informativa e intuitiva.

RNF5 La plataforma presenta los datos de forma clara e intuitiva.

4.5.3. Requisitos de Interfaces

Una vez se conocen los requisitos funcionales y no funcionales, se procede a conocer los requisitos de interfaces, en los cuales se explica como espera el usuario que responda el sistema frente a diferentes eventos.

La Tabla 4.1 muestra la lista de eventos externos a los que el sistema responde. La primera columna indica el nombre del evento; la segunda es la descripción del mismo. El "iniciador" es el componente externo al sistema que inicia el evento. Los parámetros son los datos asociados al evento. La respuesta es el nombre de una respuesta, cuya descripción está en la Tabla 4.2.

Tabla 4.1: Eventos externos.

Evento	Descripción	Iniciador	Parámetros	Respuesta
Medición de parámetros	El smart plug detecta que el dispositivo ha comenzado a ser utilizado y comienza la medición de sus parámetros.	Detección del uso de un dispositivo	Dispositivo electrónico encendido	Paquete con propiedades físicas del dispositivo
Envío de información	Una vez se captura los datos proveniente del dispositivo, el smart plug envía el paquete al gateway	Medición de parámetros completada	Paquete con propiedades físicas del dispositivo	Transmisión de paquete a gateway
Recepción de paquete	Al recibir el paquete del smart plug, el gateway almacena los datos que luego serán enviada a un servidor en la nube	Recepción de paquete proveniente de smart plug	Paquete con propiedades físicas del dispositivo recepcionado	Almacenamiento del paquete.

Almacenamiento en la nube	Una vez el paquete es almacenado de manera local, este se procesa y estandariza para ser redireccionado y almacenado en un servicio en la nube	Estandarización y procesamiento del paquete almacenado localmente	Paquete estandarizado y procesado	Redirección y almacenamiento en la nube.
Acceso a datos en la nube	Se disponibiliza endpoints a los cuales se puede acceder para extraer los datos que luego será utilizada por la plataforma	Acceso a endpoint de la API	Consulta de información	Paquete con información solicitada
Iniciar Sesión	Cuando se accede a la plataforma con un usuario previamente registrado se muestra una vista correspondiente a los privilegios del tipo de usuario	Iniciar sesión en plataforma	Usuario y contraseña	Visualización de dashboard principal
Registrarse	Cuando un usuario desea utilizar la plataforma, pero no tiene una cuenta, este debe registrarse para obtener sus credenciales que le permitan ingresar a la plataforma.	Usuario registrando su cuenta en la plataforma	Datos de usuario requeridos	Visualización de dashboard principal
Visualización de la información	Cuando se recibe el paquete con la información solicitada, este es desplegado en una plataforma de visualización -ordenado por habitaciones y dispositivos para dueños del hogar, y por casas para administradores de una comunidad - para entregarle significado a la información.	Recepción de paquete con información solicitada	Paquete con información solicitada	Visualización de la información
Modificación de parámetros	Si el usuario desea cambiar algún parámetro de los dispositivos puede hacerlo mediante la plataforma	Selección de cambio de parámetros en la plataforma	Valor de los parámetros nuevos	Notificación de cambio realizado
Alerta de nuevo dispositivo	Se alerta al usuario en caso de que haya agregado un nuevo dispositivo a la plataforma.	Conexión de un dispositivo	Paquete con información de un nuevo dispositivo	Notificación de nuevo dispositivo
Alerta de desvinculación	Cuando el usuario desvincula un dispositivo se envía una alerta informativa al usuario	Desvinculación de un dispositivo	Paquete con información de desvinculación	Notificación de desvinculación.

La Tabla 4.2 muestra de manera detallada las respuestas del sistema frente a eventos externos.

Tabla 4.2: Respuestas del sistema.

Respuesta	Descripción	Parámetros
Paquete con propiedades físicas del dispositivo	Contiene la información relevante extraída por los smart plugs	JSON con información de potencia, corriente, voltaje entre otros parámetros físicos de los dispositivos.
Transmisión de paquete a gateway	Envía el paquete desde el smart plug al gateway para su posterior almacenamiento	JSON con información de potencia, corriente, voltaje entre otros parámetros físicos de los dispositivos.
Almacenamiento del paquete.	Guarda localmente la información obtenida del paquete JSON	JSON con información de potencia, corriente, voltaje entre otros parámetros físicos de los dispositivos.
Redirección y almacenamiento en la nube.	Se estandariza y procesa el paquete JSON para ser almacenado en la nube	JSON estandarizado y parametrizado
Paquete con información solicitada	Paquete formato JSON entregado por la API	JSON con información solicitada
Visualización del dashbord principal	Vista que despliega la información del usuario que inició sesión	JSON con información solicitada
Visualización de la información	Formato gráfico para la data obtenida	JSON con información solicitada
Notificación de cambio realizado	Alerta de modificación de parámetros	Alerta emergente con mensaje de notificación
Notificación de nuevo dispositivo	Alerta de dispositivo conectado	Alerta emergente con mensaje de notificación
Notificación de desvinculación	Alerta de desvinculación de dispositivos	Alerta emergente con mensaje de notificación

4.6. Propuesta de Solución

En esta sección se toma la información descrita anteriormente para seleccionar las diferentes tecnologías que darán vida a la solución desarrollada. Para esto se debe partir analizando el estado del arte y sus aportes al proyecto.

4.6.1. ¿Como afecta lo estudiado en el estado del arte en nuestra solución?

De los documentos estudiados se pudo extraer información y puntos de vista de otras soluciones similares que llevaron a la exploración de diferentes alternativas de resolución de problemas.

Del articulo [18], se observa que la experiencia del usuario y la manera de desplegar la información puede favorecer al momento de utilizar la plataforma y lograr que esta tenga coherencia con el mundo real. Es por esto que se decide dividir el despliegue de la información referente a los dispositivos del hogar en habitaciones, las cuales son creadas y configuradas por el usuario para simular los espacios de su casa.

En [13] se destaca la arquitectura utilizada y el uso de la Raspberry Pi como dispositivo central de administración.

En [14] se da una mirada a las distintas opciones disponibles de bases de datos para aplicaciones de IoT.

De [17] y [9] se rescataron funciones que competen al proyecto, pero que por tiempo quedaron fuera del desarrollo. Estas funciones son la predicción del consumo energético para establecer un modelo de comercio en donde el usuario pueda saber cuando le conviene vender el excedente de energía generado en su hogar, utilizando incluso el almacenamiento energético de su VE y sin perder la posibilidad de realizar sus labores diarias por la venta de sus recursos. Además, en [11] se estudia la diversidad de cambios que pueden afectar a la producción de energía frente a diferentes situaciones climáticas.

Por último, el stack de tecnologías usadas por [15] para la implementación de su sistema resulta un punto base de comparación contra las elecciones para este proyecto.

4.6.2. Arquitectura del Sistema

Con todo lo mencionado anteriormente y sumado al Capítulo 4 se realizó en la Figura 4.2 la arquitectura propuesta para comprender de manera gráfica los componentes de la solución propuesta. Es necesario mencionar que el diagrama presentado esta dividido por las capas descritas con anterioridad.

El sistema se compone de 3 entidades principales: el usuario, que espera visualizar gráficos de consumo; el servicio en la nube, que recibe los datos telemétricos de los dispositivos del hogar, los almacena en una base de datos y disponibiliza mediante una API; y las residencias, que cuentan con medidores y electrodomésticos conectados directamente a un Smart Plug, enviando la información a un nodo central llamado Gateway. Este último tiene la función de reenviar la información recibida al servicio en la nube. Además se representan dos perspectivas, la inferior es la del administrador de una comunidad, y la superior la de un dueño de casa, siendo esta un zoom detallado de la primera. Ambas perspectivas pasan por el mismo middleware, donde son separadas para mostrar a cada tipo de usuario la información pertinente.

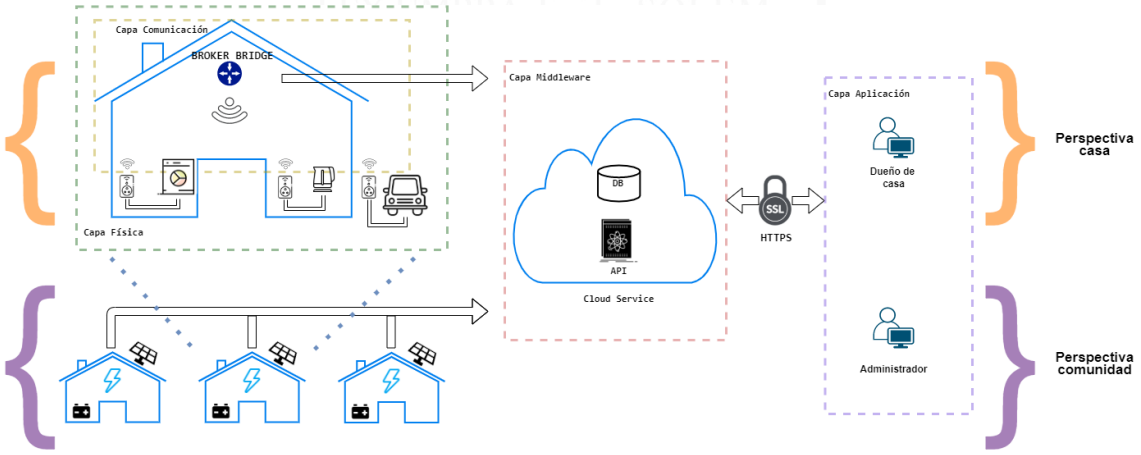


Figura 4.2: Arquitectura propuesta

5 | Software

Esta sección abarca dos capas de la arquitectura de la plataforma, la capa de middleware y la de aplicación, donde se explican las consideraciones hechas para cada una de ellas. Específicamente se da a conocer las tecnologías y frameworks seleccionados para la implementación de la plataforma.

5.1. Capa de Middleware

En esta sección se realizan las comparativas de los componentes relacionados con esta capa, los que son: Hosting, Bases de datos y Backend.

5.1.1. Hosting

El hosting esta asociado a la elección de servidores en la nube donde la plataforma se desarrolló y desplegó para ser accedida desde cualquier parte del mundo y en cualquier momento.

Dentro del mercado existen muchos proveedores de servicio en la nube, tales como: Microsoft Azure, Oracle Cloud, IBM, entre muchos otros.

Para realizar la elección, se comparó 2 proveedores: Amazon Web Service (AWS) y DigitalOcean.

Para comenzar, DigitalOcean se concentra en tres puntos clave de venta para destacar:

1. Simplicidad

2. Precio

3. Servidores virtuales de alto rendimiento

Y se centran en ofrecer a los desarrolladores una forma fácil y rápida de crear instancias Linux asequibles que denominan droplets. DigitalOcean es compatible con la mayoría de las distribuciones modernas de Linux: Ubuntu, Fedora, Debian y CentOS. Es sencillo configurar varias aplicaciones en sus droplets, por ejemplo, Ruby on Rails, LAMP, Ghost, Docker o stack.

Los precios de DigitalOcean son los más asequibles entre todos los proveedores de nube. Los precios comienzan en 0,007$/hora o 5$/mes y ofrecen una fácil transición entre las tarifas por hora y las mensuales. Su paquete más popular, llamado droplet, cuesta 0,015 dólares/hora o 10 dólares/mes y proporciona un núcleo de procesador, 1GB de memoria, 30GB de disco SSD y 2TB de transferencia.

Por otro lado AWS, de Amazon, es el líder del mercado con diferencia; se estima que Amazon tiene tanto poder de computación como los siguientes 11 rivales de la lista juntos. Además de poseer los mayores centros de datos del mundo ubicados estratégicamente en 9 regiones del planeta.

Amazon también tiene una sólida base de datos de documentación de ayuda, pero con el gran tamaño de la oferta de servicios disponible en el portal, la mayoría de los clientes necesitan contactar con el soporte en algún momento. Disponen de un enorme equipo de soporte que puede atenderlo, pero el soporte no está incluido en todos los paquetes. Los gastos de asistencia técnica pueden suponer hasta un 10 % de su gasto mensual, lo que puede suponer una cantidad importante para las organizaciones más grandes [23].

Basados en lo anterior, la elección realizada fue DigitalOcean, por ser amigable con desarrolladores que desean prototipar de manera rápida y sencilla, a la vez que el precio es accesible para máquinas de buena calidad, y con la posibilidad de mejorar las características de la maquina virtual con un par de clics.

5.1.2. Base de datos

Para realizar la elección de base de datos, se investigó aquellas más utilizadas en el ámbito del internet de las cosas. Basándose en [24] se realizó la comparación de algunas de las opciones más populares, destacándose en la Tabla 5.1 alguna de sus características más importantes.

Tabla 5.1: Comparación de Bases de datos.

	MongoDB	**CrateDB**	**InfluxDB**
Tipo	Orientada a documentos	SQL	Orientada a series de tiempo
Lenguaje	C++, JavaScript, Python	Java	Go
Características	No estructurado. Propósito general.	Distribuido. Millones de datos por segundo. Esquema dinámico.	Llave-valor. Indexación de series. Interpolación de puntos faltantes. Submuestreo automático.

Además, en [25] se llevó a cabo un estudio del desempeño de 4 bases de datos distintas, MySQL, MongoDB, MonetDB e InfluxDB, comparando sus tiempos de respuesta frente a consultas realizadas a datos obtenidos de las lecturas de medidores inteligentes. Los resultados que obtuvieron para distintos tipos de consultas son los mostrados en la Figura 5.1.

A partir de esto, dada la especialidad de InfluxDB para series de tiempo, correspondientes con el tipo de dato recolectado por los smart plugs, y su eficiente operación para grandes cantidades de datos, es que se elige como la base de datos seleccionada para el almacenamiento de los datos de consumo a través del tiempo.

Sin embargo, hay otro tipo de información no relacionada al tiempo que también debe manejarse . Estos son los datos de los usuarios, para los que se selecciona MongoDB como la base de datos utilizada, ya que gracias a su flexibilidad, permite realizar un prototipado rápido y adaptable a las cambiantes necesidades del proyecto.

Query Number	Result	Runtime (s)
MongoDB 1	84965 row(s) returned	0.65
MongoDB 2	781 row(s) returned	57.26
MongoDB 3	1 row(s) returned	0.04
MongoDB 4	697409 row(s) returned	0.88
MongoDB 5	33856 row(s) returned	854.75
MongoDB 6	33856 row(s) returned	874.18
MongoDB 7	0 row(s) returned	1054.73
InfluxDB 1	84965 row(s) returned	1.38
InfluxDB 2	781 row(s) returned	13.55
InfluxDB 3	1 row(s) returned	10.85
InfluxDB 4	697409 row(s) returned	0.08
InfluxDB 5	33856 row(s) returned	15.37
InfluxDB 6	33856 row(s) returned	10.37
InfluxDB 7	0 row(s) returned	58.88
MonetDB 1	84965 row(s) returned	8.54
MonetDB 2	781 row(s) returned	60.46
MonetDB 3	1 row(s) returned	57.19
MonetDB 4	697409 row(s) returned	1.88
MonetDB 5	33856 row(s) returned	78.64
MonetDB 6	33856 row(s) returned	76.63
MonetDB 7	0 row(s) returned	256.15
MonetDB 8	101444 row(s) returned	265.65
MonetDB 9	3 row(s) returned	9054.64
MySQL 1	84965 row(s) returned	134.70
MySQL 2	781 row(s) returned	121.69
MySQL 3	1 row(s) returned	117.76
MySQL 4	697409 row(s) returned	121.81
MySQL 5	33856 row(s) returned	148.98
MySQL 6	33856 row(s) returned	158.33
MySQL 7	0 row(s) returned	451.45
MySQL 8	101444 row(s) returned	473.36
MySQL 9	3 row(s) returned	18184.06

Figura 5.1: Resultados y tiempo de ejecución de consultas

5.1.3. Backend

La elección del backend se hizo en base a conocimientos existentes, ya que previamente se desarrolló con NodeJS y Express. Estas herramientas permiten el desarrollo rápido de prototipos de aplicaciones web y móviles.

NodeJS es un entorno de desarrollo basado en Javascript y Express es un framework de NodeJS que permite configuración de middelwares para responder a peticiones HTTP; define una tabla de enrutamiento que se utiliza para realizar diferentes acciones basadas en el método HTTP y la URL; y permite renderizar dinámicamente páginas HTML basadas en

el paso de argumentos a las plantillas. Por lo que utilizar NodeJS y Express permite resolver de manera sencilla problemas asociados a peticiones HTTP y creación de endpoints para acceder a la información almacenada en las bases de datos.

5.2. Capa de Aplicación

5.2.1. Framework frontend

Finalmente, el framework utilizado para la implementación de la plataforma de visualización, se eligió basado en los 3 más predominantes en el mercado en la actualidad, estos son: React, Angular y Vue, todos ellos implementados en Javascript y basados en componentes. En Tabla 5.2 se mencionan algunas de sus características importantes para la toma de decisión.

Tabla 5.2: Comparación de frameworks para frontend.

	React	**Angular**	**Vue**
Dificultad de aprendizaje	Media	Alta	Baja
Documentación	Baja	Buena	Buena
Comunidad	Grande	Grande	Pequeña
Robustez	Respaldado por Facebook	Respaldado por Google	No tiene un fuerte respaldo
Enlace de datos	Unidireccional	Bidireccional	Bidireccional
Especialidad	Proyectos de mediana escala	Proyectos complejos de gran escala	Integración progresiva en proyectos

Se consideró también una encuesta realizada a desarrolladores sobre sus experiencias y preferencias en cuanto a estos frameworks [26], con los resultados obtenidos mostrados en la Figura 5.2. De aquí se puede observar que React da buenas señales de una numerosa comunidad satisfecha con su uso, Vue tiene cierto reconocimiento pero no tanto uso, y Angular tiene una gran cantidad de usuarios en comparación a los de anteriores, que no lo volverían a usar.

Tomando en cuenta la envergadura del proyecto, considerando que se trata de una plataforma no tan grande -implementada desde cero- sin una extensa experiencia previa ninguno de estos frameworks, y basado en la comunidad existente para la resolución de

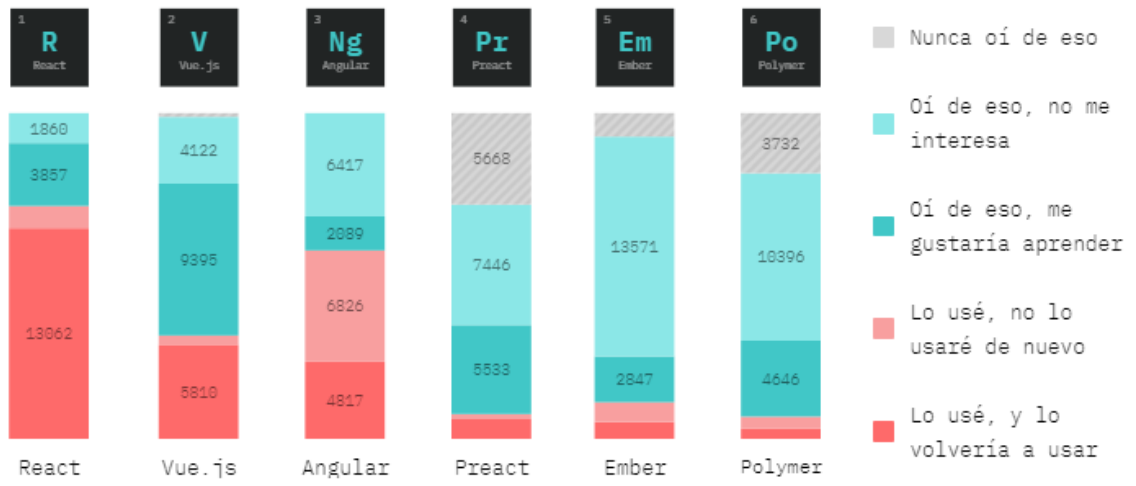


Figura 5.2: Encuesta de frameworks Frontend

conflictos, es que se selecciona React como el framework para la implementación. Además, React es una biblioteca de javascript al igual que las tecnologías que se utilizan en el backend, lo que permite desarrollar tanto backend como frontend con solo un lenguaje de programación.

5.2.2. Diseño de plataforma

Para comenzar se describen los modelos de navegación de cada tipo de usuario al interactuar con las diferentes interfaces de la plataforma. En la Figura 5.3 se observa el flujo que realiza un usuario dueño de hogar, mientras que la Figura 5.4 presenta el flujo de un administrador

Basado en estos modelos, se diseñó con la herramienta Figma los Mockup -o bosquejos- de las interfaces, con el fin de validar las funcionalidades esperadas con el cliente y ser usadas como referencia para la implementación

Mockup

La primera pantalla con la que se encuentra un usuario, independiente de su rol, es un formulario de inicio de sesión, en donde accederá al sitio ingresando su correo y contraseña en caso de ya contar con una cuenta Figura 5.5.

De lo contrario podrá dirigirse a la pantalla de registro Figura 5.6. En ella se ve un

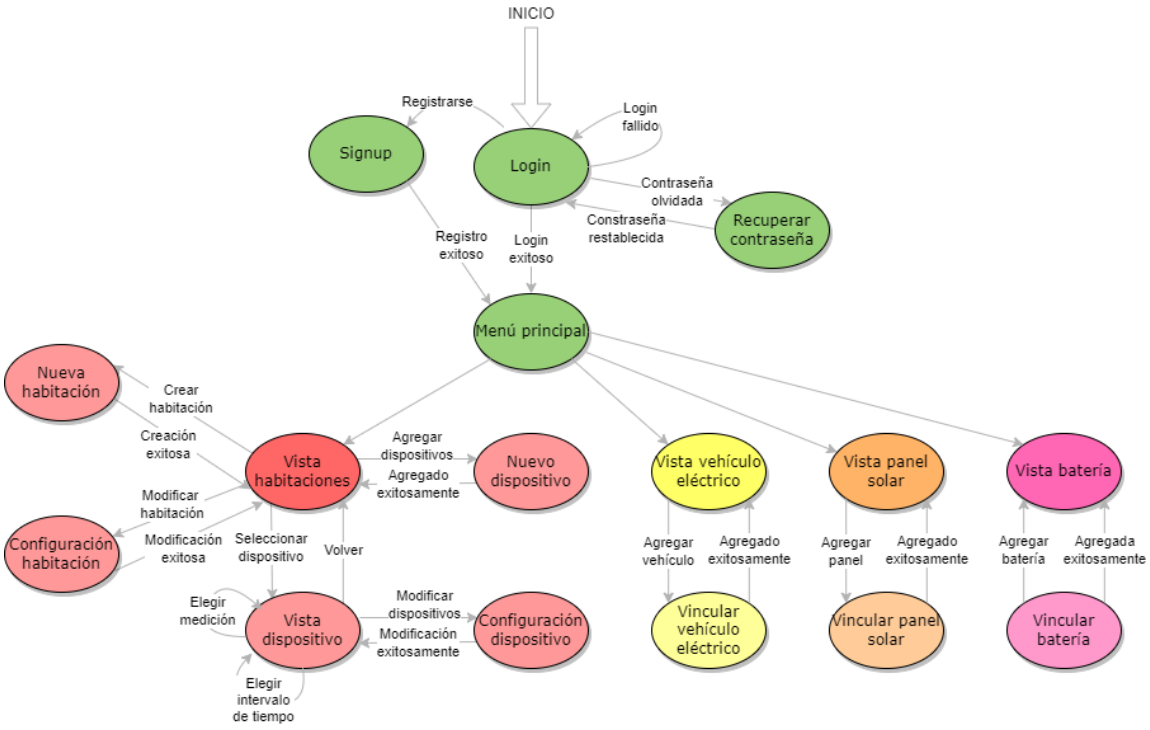


Figura 5.3: Modelo de navegación de interfaces para usuario.

formulario donde debe ingresar su nombre, apellido, correo, contraseña y especificar su rol para crear su cuenta. Luego de esto el usuario ingresa directamente a la plataforma.

Una vez iniciada sesión, dependiendo del tipo de usuario, la pantalla mostrada será distinta. Para el caso de un usuario dueño de casa corresponde a la mostrada en la Figura 5.7, donde se visualiza el consumo general del hogar, además del estado del vehículo eléctrico u otras fuentes de energía en caso de poseer. Desde está página también se puede acceder a distintas vistas a través de los enlaces presentes en la barra de navegación.

La segunda vista a la que se puede acceder es la de dispositivos que se observa en Figura 5.8, ahí el usuario encontrará una lista de todos los dispositivos que posea, ordenados por habitaciones, teniendo la opción de moverse entre ellas, y agregar nuevas habitaciones y dispositivos. Además, el usuario puede seleccionar un dispositivo de la lista si desea ver sus datos telemétricos en detalle, lo que lo lleva a la vista del dispositivo en específico Figura 5.9.

En ella se puede: ver un gráfico de las distintas métricas recolectadas, seleccionar la métrica que desea visualizar, y modificar el rango de tiempo a mostrar. Además, puede

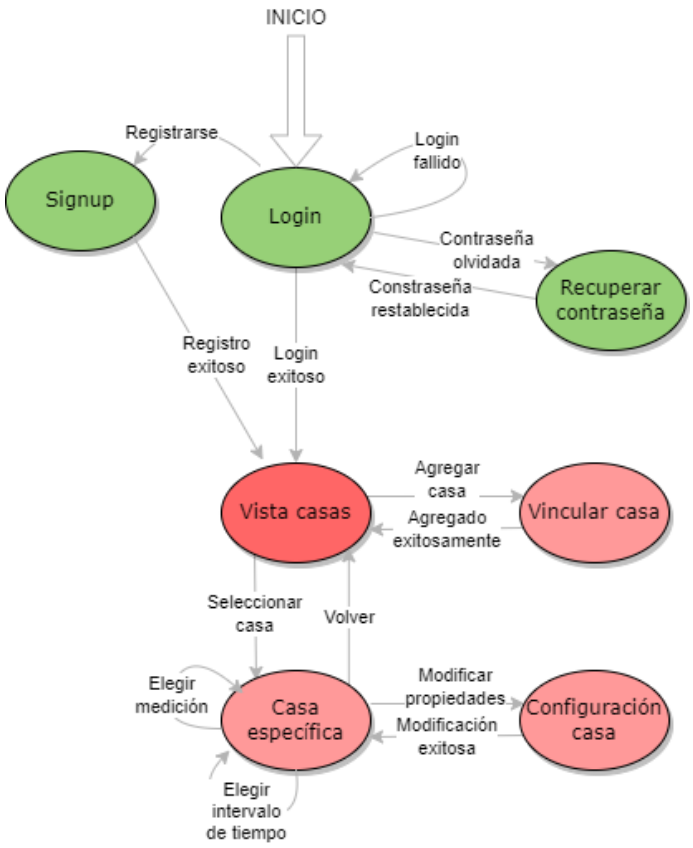


Figura 5.4: Modelo de navegación de interfaces para administrador.

volver a la pagina anterior para seleccionar otro dispositivo.

En el caso de un usuario administrador de una comunidad, la vista que encontrará al iniciar sesión es la de la Figura 5.10, donde se muestra una lista con los usuarios u hogares bajo su administración.

Al seleccionar alguna de estas casas, es llevado a la vista de consumo general de esta, mostrada en la Figura 5.11, similar a lo que vería el dueño de esta casa al acceder desde su cuenta.

Figura 5.5: Diseño login.

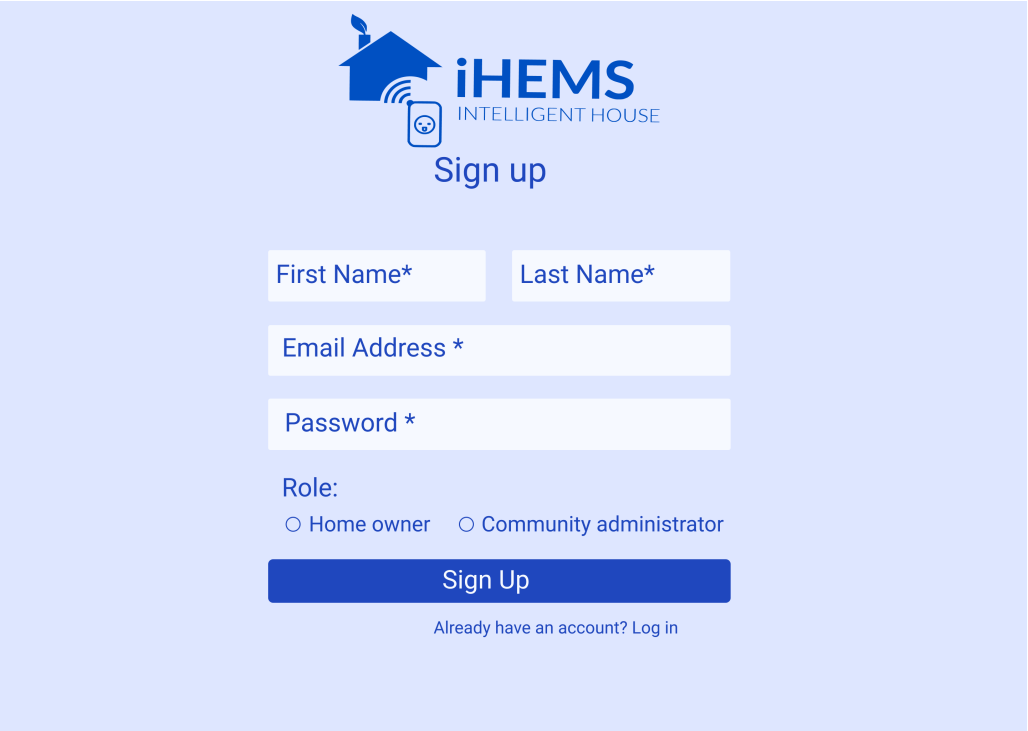


Figura 5.6: Diseño sign up

Figura 5.7: Diseño vista inicio para usuario.

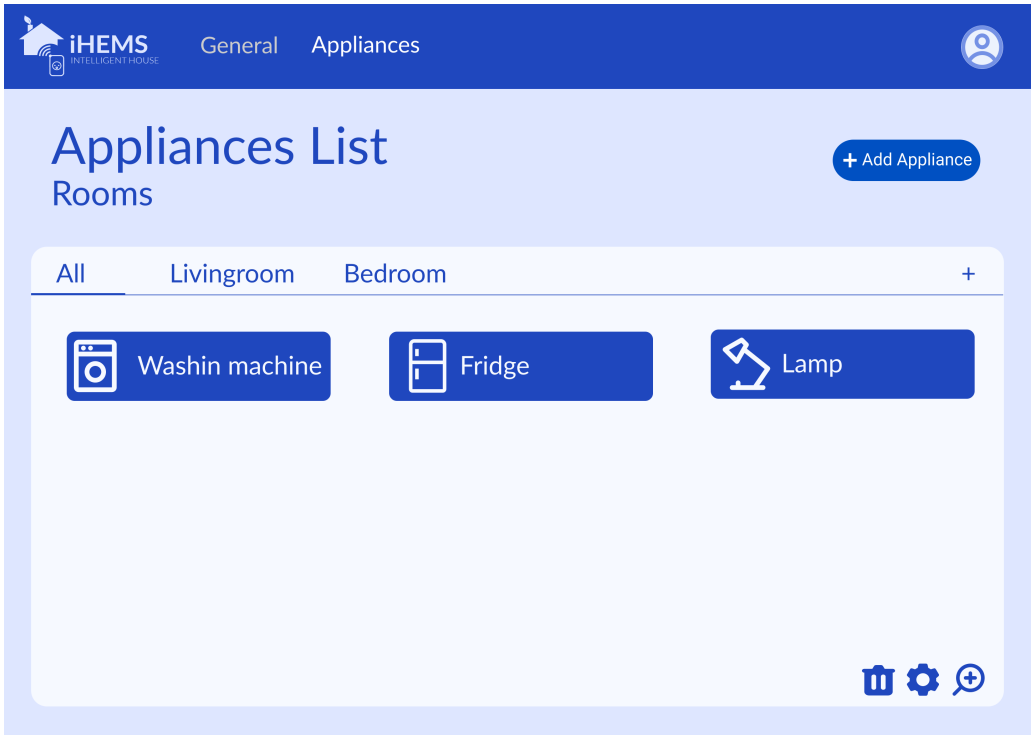


Figura 5.8: Diseño vista habitaciones.

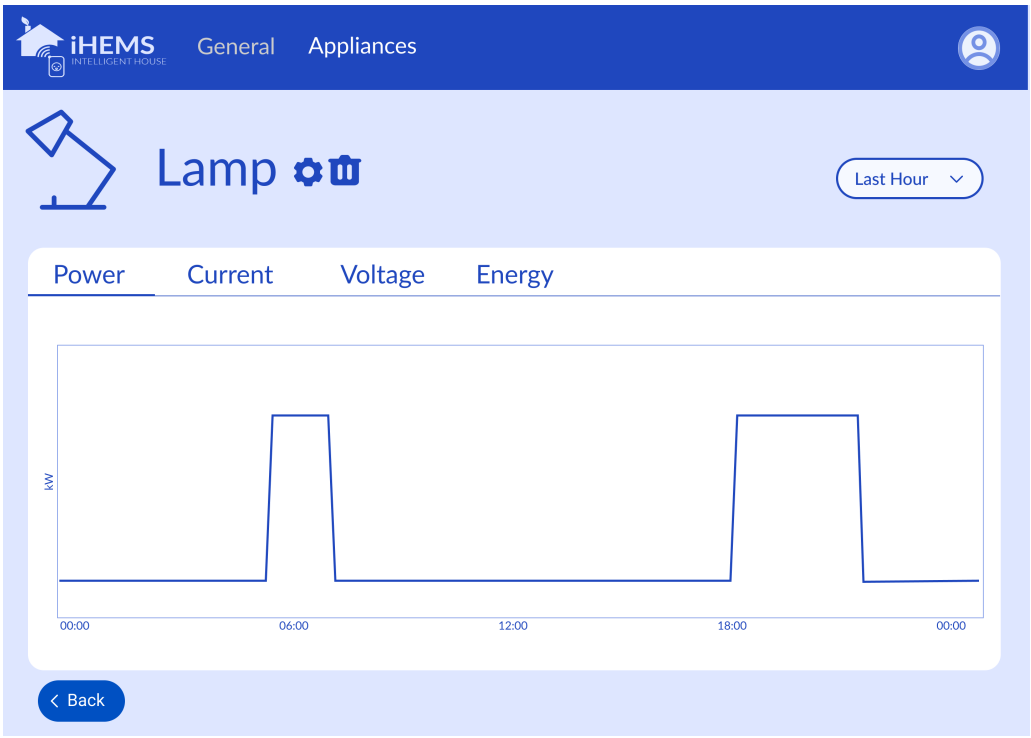


Figura 5.9: Diseño vista dispositivos.

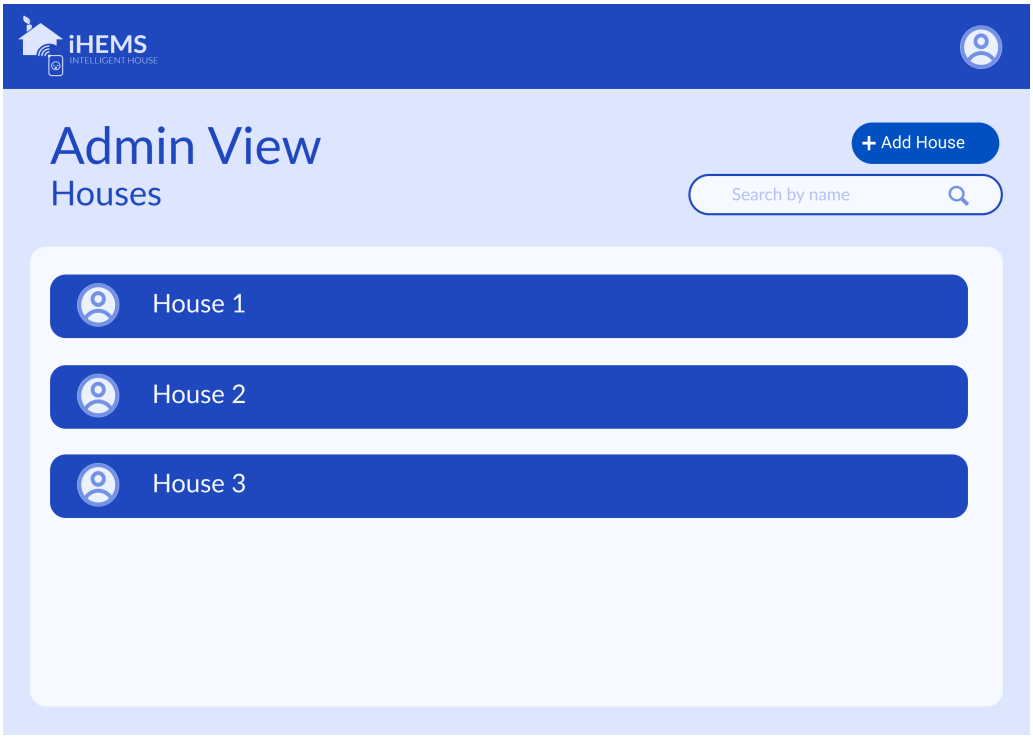


Figura 5.10: Diseño vista administrador

Figura 5.11: Diseño vista casa administrador

6 | Implementación

6.1. Arquitectura del Sistema

Con todo lo mencionado en Capítulo 5 se actualizó el diagrama de arquitectura para incluir las tecnologías seleccionadas para la implementación de las capas de middleware y aplicación, como se muestra en la Figura 6.1. Además, se incluyen las elecciones para las capas física y de comunicación, cuyo estudio comparativo fue realizado por otro miembro del equipo en su trabajo de memoria correspondiente.

En la siguiente lista se muestra una recopilación en detalle de todas las tecnologías utilizadas en el prototipo de solución y su función dentro de él.

- **Smart Plug:** Sonoff Pow R2
- **Gateway:** Raspberry Pi 3B+
- **Front-end:** React + Giraffe
- **Back-end:**
  - **API REST:** NodeJS + Express
  - **Base de datos telemétricos:** InfluxDB + Telegraf
  - **Base de datos usuarios:** MongoDB
- **Cifrado:** SSL

Con el objetivo de comprender la funcionalidad de cada componente del sistema se dividieron las tecnologías en módulos como se observa en Figura 6.2, los que de detallan en la Tabla 6.1

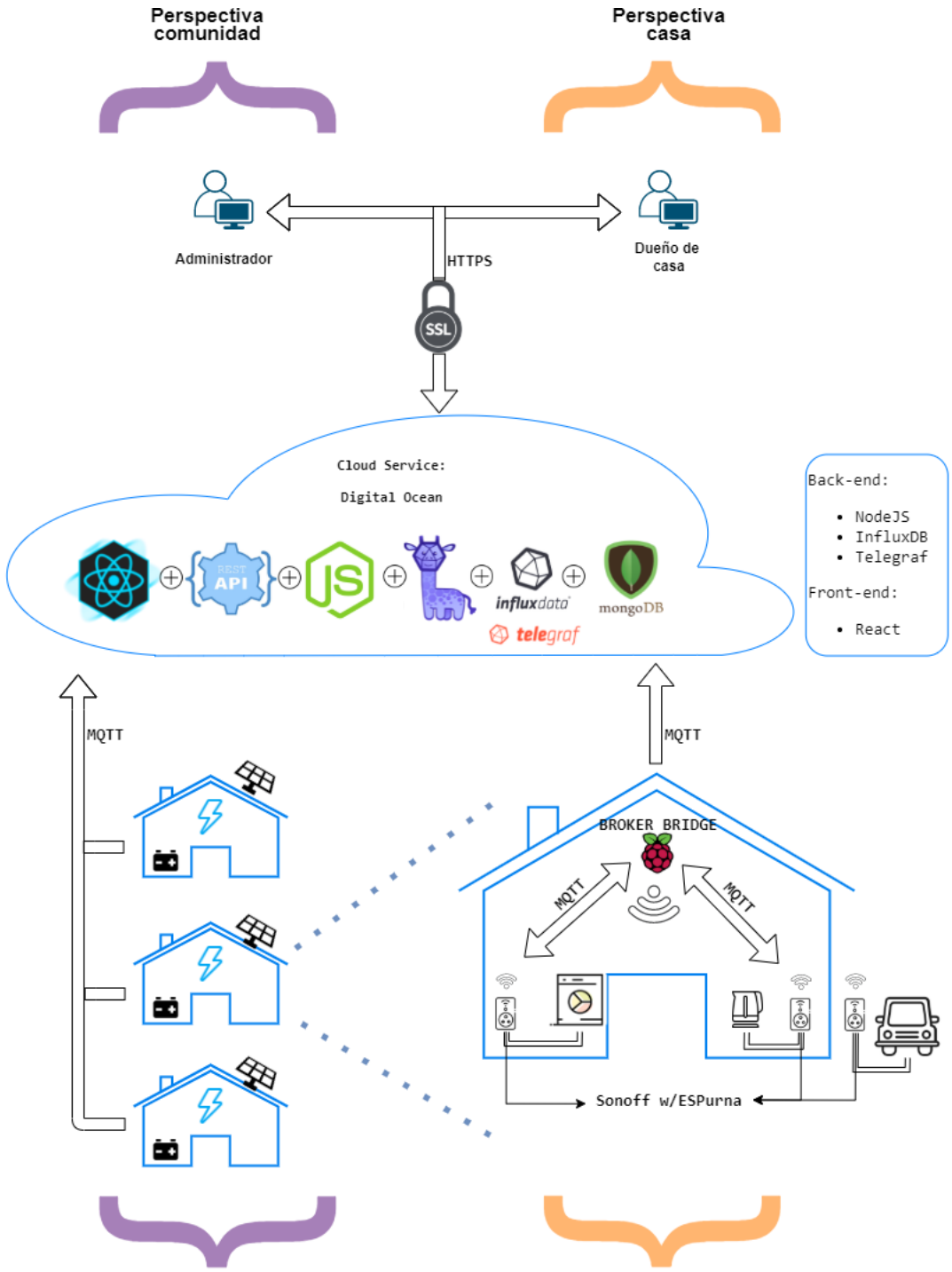


Figura 6.1: Arquitectura de componentes propuesta.

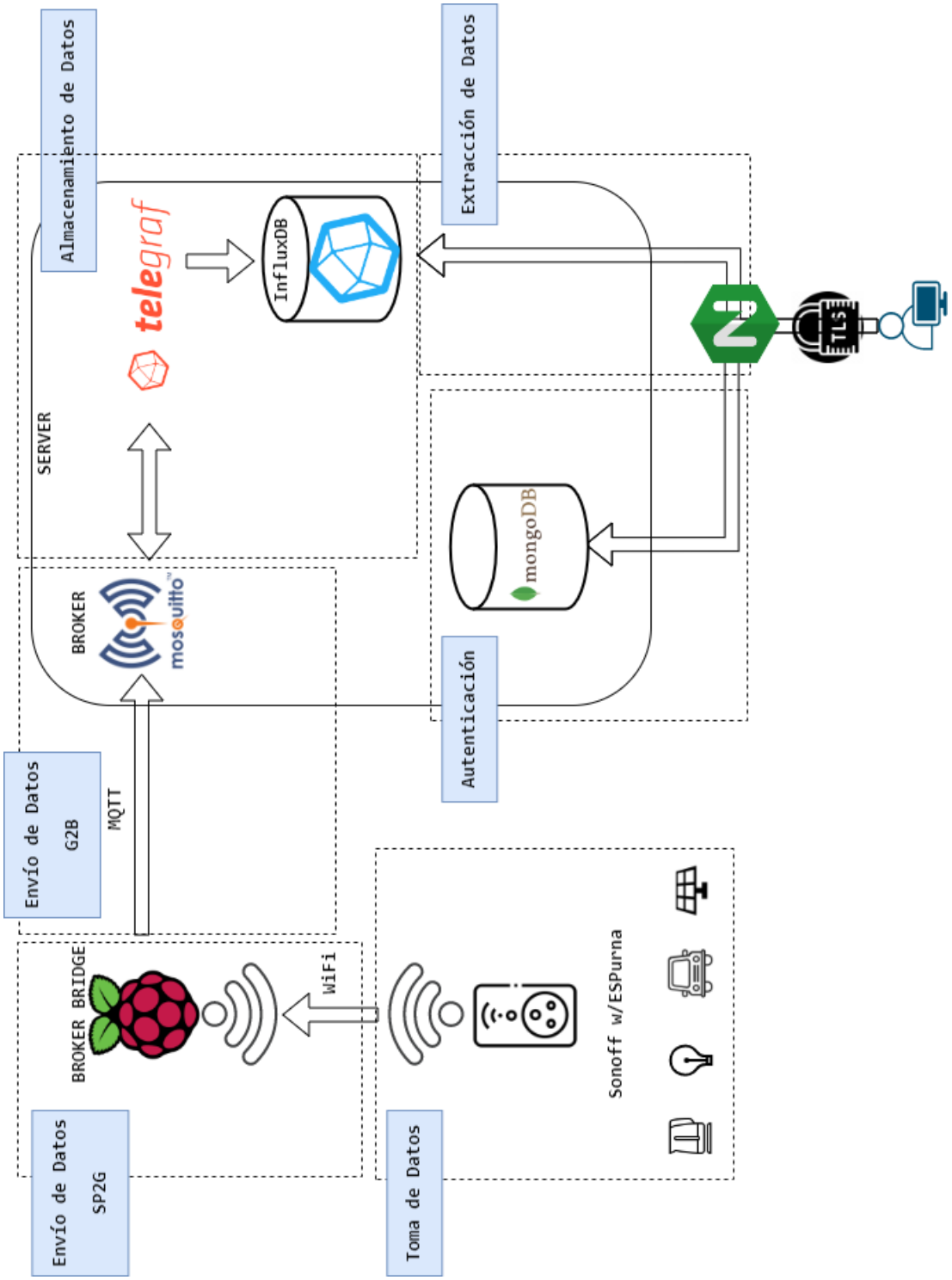


Figura 6.2: Diagrama de Flujo Módulos.

Tabla 6.1: Descripción de Módulos.

Módulo	Función
1.-Toma de datos	En este módulo se realiza la toma de datos de los electrodomésticos y se envían al gateway.
2.-Envío de datos Smart Plug a Gateway (SP2G)	Una vez los datos son recopilados, se encapsulan en un paquete para ser enviados al gateway. Este último es el encargado de otorgar conectividad segura para su configuración.
3.-Envío de datos Gateway a Broker (G2B)	Cuando el gateway recibe datos de los smart plugs se reenvían automáticamente a un servidor alojado en la nube.
4.-Almacenamiento de datos	El servidor almacena los datos en una base de datos cuando son recibidos.
5.-Autenticación	El módulo de autenticación esta encargado de almacenar las credenciales de los usuarios de la plataforma y, además, de entregarle un token a los usuarios que sirve para acceder a los datos. Para esto se disponibiliza una API que permite realizar el proceso de autenticación
6.-Extracción de datos	La misma API descrita anteriormente tiene la funcionalidad de extracción de la información desde la base de datos. Para ello se utiliza el token obtenido en el modulo de autenticación, con el fin de verificar que cada usuario acceda solo a los datos que le corresponden, para el despliegue visual de ellos en la web.

6.2. Implementación Backend

El backend se divide en 2 secciones: Bases de datos y API. Estas se detallan a continuación:

6.2.1. Bases de datos

La implementación de las bases de datos a su vez tienen 2 secciones: InfluxDB y MongoDB. La primera se enfoca en almacenar los datos telemétricos y la segunda almacenar las credenciales de los usuarios.

6.2.1.1. InfluxDB

Lo primero es añadir el repositorio de Influx a apt con las siguientes líneas de comando:

1. `wget -qO- https://repos.influxdata.com/influxdb.key | sudo apt-key add -`

2. `source /etc/os-release`

3. `echo "deb https://repos.influxdata.com/debian $(lsb_release -cs) stable"`

`| sudo tee /etc/apt/sources.list.d/influxdb.list`

Luego actualizar con los nuevos repositorios e instalar:

`sudo apt update && sudo apt install -y influxdb`

Para finalizar, correr el servicio y se configura para ejecutarse al iniciar el sistema operativo:

1. `sudo systemctl unmask influxdb.service`
2. `sudo systemctl start influxdb`
3. `sudo systemctl enable influxdb.service`

Con esto se instaló InfluxDB. Luego fue necesario instalar y configurar Telegraf para que los datos recibidos por el broker MQTT del servidor sean almacenados de manera automática en InfluxDB. Para esto se instaló Telegraf con el comando:

`sudo apt install telegraf`

Esto se vio simplificado debido a la actualización de los repositorios al instalar InfluxDB.

Una vez instalado, se procede a configurar el archivo `telegraf.conf` alojado en `/etc/telegraf/` de la siguiente manera: Buscar apartado descrito y modificar lo que se muestra a continuación:

```
[[inputs.mqtt_consumer]]
    servers = ["tcp://127.0.0.1:1883"]
    topics = [
     "/sonoff/data",
   ]
   qos = 0
   persistent_session = false
   client_id = ""
   name_override="mqtt_consumer_json_test"
   tag_keys=["host","mac","ip"]
   json_time_key="time"
```

```
json_time_format="2006-01-02 15:04:05"
data_format = "json"
```

Listing 6.1: telegraf.conf

Esto permite a Telegraf mantenerse constantemente monitoreando el tópico `/sonoff/data` al que los smart plugs envían los paquetes de MQTT. Además, por funcionamiento de telegraf, solo almacena los datos numéricos de manera automática en InfluxDB, por lo que es necesario definir los `tag_keys` .

El apartado `json_time_key` y `json_time_format` son para: utilizar la fecha y hora del paquete -que es la hora exacta de la toma de datos realizada por el smart plug-; y el formato con el que se almacena la fecha y hora recogida. Por último `data_format` permite almacenar la información en formato JSON.

Con todo esto se generó el siguiente formato de datos que se almacenan en InfluxDB Figura 6.3, donde el **Measurement** actúa como contenedor de **timestamp**, **fields** y **tags**. Comparado con SQL, un measurement es conceptualmente similar a una tabla.

Entrando más a fondo en los conceptos, el timestamp es la fecha y hora en que se realizó la medición.

Los fields se dividen en field key, que es el nombre que asocia la medición -corriente, voltaje, etc-; y field value, que es el valor asociado a la medición. Las field key son de formato texto y las field values tienen valores numericos.

Por último los tags. Estos también se dividen en key y value, pero en este caso ambos son de tipo texto. En este caso `SmartPlugID` sería el identificador de los smart plug y las mediciones serían las MAC de cada uno de los smart plugs.

Cabe mencionar que InfluxDB almacena los datos en un WAL, que es un caché temporal de archivos almacenados recientemente con el fin de comprimirlos de mejor manera para luego guardarlos permanentemente en una TSM. Una TSM es el formato de almacenamiento diseñado para influxDB que permite una mayor compresión en los datos. Este último esta compuesto por Shards que contienen datos reales codificados y comprimidos.

Measurement: Sonoff							
Timestamp	Current	Voltage	Power	Reactive	Apparent	Energy	SmartPlugID
2015-08-18T00:00:00Z	0.132	230	19	24	31	0.028	98:F4:AB:E2:D7:24
2015-08-18T00:06:00Z	7.626	218	1660	182	1689	0.218	98:F4:AB:E2:E8:B8
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Timestamp Fields Tags

Figura 6.3: Formato de base de datos InfluxDB

6.2.1.2. MongoDB

Para poder configurar las credenciales de los usuarios se utilizó la librería Mongoose y se implementó el Schema de los usuarios como se muestra a continuación en los Listing 6.2 y Listing 6.3:

```
UserSchema ={
    "firstName":String,
    "lastName":String,
    "email":String,
    "password":String,
    "UserID":String,
    "smartPlugs":[{
        "smartPlugID": String,
        "smartPlugRoom": String,
        "smartPlugName": String
    }],
    "rooms":[{
        "roomID":mongoose.Types.ObjectId,
        "roomName":String,
    }],
    "electricMeterID": String,
    "electricVehicleID": String,
    "solarPanelID": String,
    "batteryID": String
}
```

Listing 6.2: User Schema

```
AdminSchema = {
    "firstName": String,
```

```
    "lastName": String,
    "email": String,
    "password": String,
    "UserID": String,
    "renters": [{
        "firstName": String,
        "lastName": String,
        "UserID": String
    }]
}
```

Listing 6.3: Admin Schema

6.2.2. API

Para la implementación de la API fue necesario definir los endpoints con su funcionalidad, los cuales se detallan en la Tabla 6.2. En ella se describen los endpoints a los que se hacen las consultas, junto con su operación CRUD y los parámetros que debe recibir. La implementación de estos endpoints fue realizada de manera eficiente gracias a Express, con la ayuda de Mongoose y el cliente de Influx para los accesos a los datos de los usuarios y sus dispositivos.

El funcionamiento detallado de las rutas definidas es el siguiente:

- **/auth/**: Recibe el email y contraseña del usuario, que en caso de ser correctas, entrega un Json Web Token (JWT) para identificar al usuario y permitir el acceso a sus datos. Este token es una cadena de caracteres que contiene información sobre la identidad del usuario y el tiempo de expiración, cifradas mediantes una llave secreta, de modo que las consultas futuras deben proveer este token para validar la identidad, y cualquier alteración al token original resultará inválida sin la llave correcta.
- **/auth/new**: Recibe el nombre, apellido, email, contraseña y el tipo de un nuevo usuario para crear su cuenta. Luego de verificar que este usuario no exista previamente, se guarda en la base de datos y retorna un JWT para su ingreso al sistema.
- **/auth/renew**: Esta ruta tiene un doble propósito, verifica que el token entregado sea válido para manejar los ingresos automáticos a la plataforma, además de generar uno

Tabla 6.2: Rutas API

Ruta	Método	Headers	Body

/auth/	POST	Content-Type: application/json	{
			email: String,
			password: String
			}

/auth/new	POST	Content-Type: application/json	{
			firstName: String,
			lastName: String,
			email: String,
			password: String,
			userType: String
			}

/auth/renew	GET	x-token: token	{}
/crud/appliances	GET	x-token: token	{}

/crud/new_appliance	POST	Content-Type: application/json
		x-token: token
			{
			smartPlugID: String,
			smartPlugRoom: String,
			smartPlugName: String
			}

/crud/modify_appliance/:smartPlugID	PUT	Content-Type: application/json
		x-token: token
			{
			smartPlugRoom: String,
			smartPlugName: String
			}

/crud/delete_appliance/:smartPlugID	DELETE	Content-Type: application/json x-token: token	{}
/crud/rooms	GET	x-token: token	{}

/crud/new_room	POST	Content-Type: application/json
		x-token: token
			{
			roomName: String
			}

/crud/modify_room/:roomID	PUT	Content-Type: application/json
		x-token: token
			{
			roomName: String
			}

/crud/delete_room/:roomID	DELETE	Content-Type: application/json x-token: token	{}
/crud/get_sources/:userID	GET	x-token: token	{}

/crud/new_source/:sourceType	PUT	Content-Type: application/json
		x-token: token
			{
			sourceID: String
			}

/crud/houses	GET	x-token: token	{}

/crud/new_house	POST	Content-Type: application/json
		x-token: token
			{
			houseID: String,
			houseName: String
			}

/crud/modify_house/:houseID	PUT	Content-Type: application/json
		x-token: token
			{
			houseName: String
			}

/crud/delete_house/:smartPlugID	DELETE	Content-Type: application/json x-token: token	{}
/crud/meter/:houseID	GET	x-token: token	{}
/measurements/Hour/:smartPlugID	GET	x-token: token	{}
/measurements/Today/:smartPlugID	GET	x-token: token	{}
/measurements/Week/:smartPlugID	GET	x-token: token	{}
/measurements/Month/:smartPlugID	GET	x-token: token	{}
/measurements/All/:smartPlugID	GET	x-token: token	{}

nueva una vez que el primero haya expirado.

- **/crud/appliances**: Retorna una lista de los dispositivos registrados por el usuario identificado en el JWT, con sus correspondientes nombres y habitación a la que pertenecen.
- **/crud/new_appliance**: Registra un nuevo dispositivo en la base de datos para el usuario identificado en el JWT a partir de los datos entregados en el cuerpo de la consulta, y retorna si la operación se realizó correctamente.
- **/crud/modify_appliance/:smartPlugID**: Realiza una modificación al dispositivo correspondiente de acuerdo a los datos entregados en el cuerpo, y retorna si la operación se realizó correctamente.
- **/crud/delete_appliance/:smartPlugID**: Elimina el dispositivo correspondiente de la base de datos y retorna si la operación se realizó correctamente.
- **/crud/rooms**: Retorna una lista de las habitaciones registrados por el usuario identificado en el JWT.
- **/crud/new_room**: Registra una nueva habitación en la base de datos para el usuario identificado en el JWT, a partir de los datos entregados en el cuerpo de la consulta, y retorna si la operación se realizó correctamente además del ID de la habitación creada.
- **/crud/modify_room/:roomID**: Realiza una modificación a la habitación correspondiente de acuerdo a los datos entregados en el cuerpo, y retorna si la operación se realizó correctamente.
- **/crud/delete_room/:roomID**: Elimina la habitación correspondiente de la base de datos y retorna si la operación se realizó correctamente.
- **/crud/get_sources/:UserID**: Retorna el identificador asociado a cada una de las fuentes de energía presentes en el hogar del usuario especificado en los parámetros, entre las que se encuentran el medidor eléctrico, los paneles solares y la batería, para su posterior acceso a las lecturas realizadas. Este endpoint es accesible tanto para los propios dueños de casa, como para el administrador de comunidad.
- **/crud/new_source/:sourceType**: Modifica el campo correspondiente al tipo de fuente

del usuario identificado en el JWT. Dado que se trata de una modificación a un campo existente en el esquema de usuario, este endpoint funciona para agregar por primera vez una fuente o para modificar una ya existente.

- **/crud/houses**: Retorna una lista con los identificadores de usuarios para los que el administrador de comunidad identificado en el JWT tiene acceso.
- **/crud/new_house/:houseID**: Registra una nueva casa en la base de datos para el administrador de comunidad identificado en el JWT a partir de los datos entregados en el cuerpo de la consulta y retorna se si la operación se realizó exitosamente.
- **/crud/modify_house/:houseID**: Realiza una modificación a la casa correspondiente de acuerdo a los datos entregados en el cuerpo, y retorna si la operación se realizó exitosamente.
- **/crud/delete_house/:houseID**: Elimina la casa correspondiente de la base de datos y retorna si la operación se realizó exitosamente.
- **/measurements/Hour/:smartPlugID**: Retorna las mediciones realizadas por el dispositivo solicitado durante la última hora.
- **/measurements/Today/:smartPlugID**: Retorna las mediciones realizadas por el dispositivo solicitado durante las últimas 24 horas.
- **/measurements/Week/:smartPlugID**: Retorna un resumen por día de las mediciones realizadas por el dispositivo solicitado durante la última semana.
- **/measurements/Month/:smartPlugID**: Retorna un resumen por día de las mediciones realizadas por dispositivo solicitado durante el último mes.
- **/measurements/All/:smartPlugID**: Retorna todas las mediciones realizadas por dispositivo solicitado.

6.3. Implementación Frontend

A continuación se explica el funcionamiento de cada una de las vistas implementadas del sistema, y su interacción con la API descrita anteriormente.

6.3.1. Login

La vista de login cuenta con un formulario de ingreso con campos para el email y la contraseña del usuario, una vez estos son ingresados, la plataforma realiza una consulta de tipo POST a la API en la URL *https://api.ihems.cl/auth*, insertando los datos en el cuerpo de la solicitud. En caso de ser las credenciales correctas, se recibe y almacena el token de autenticación para futuras consultas, y dependiendo del tipo de usuario identificado en la respuesta por parte del servidor, se redirecciona a la vista de habitaciones en caso de ser un usuario dueño de casa, o a la vista de hogares en caso de ser un administrador. En caso contrario se recibe un mensaje de error que se despliega al usuario en pantalla mediante una alerta.

Además cuenta con un enlace directo que redirecciona a la vista de registro.

Cabe destacar que si el usuario ha iniciado sesión previamente y cuenta con un token guardado en su navegador, la plataforma primero comprueba la validez de este incluyéndolo en una consulta GET a la ruta *https://api.ihems.cl/auth/renew*, redireccionándolo directamente a la vista correspondiente. Este flujo puede es representado en la Figura 6.4.

6.3.2. Sign up

En la vista de registro se encuentra un formulario que requiere el nombre, apellido, correo, contraseña y el tipo del usuario que desea registrarse, y envía estos datos en el cuerpo de una solicitud POST a la ruta *https://api.ihems.cl/auth/new*. Si obtiene una respuesta correcta, recibe y almacena el token de acceso, de modo que se inicia sesión inmediatamente redirigiendo a la vista de habitaciones. Si la respuesta es errónea se recibe un mensaje de error que se muestra mediante una alerta.

Además cuenta con un enlace directo que redirecciona a la vista de inicio de sesión.

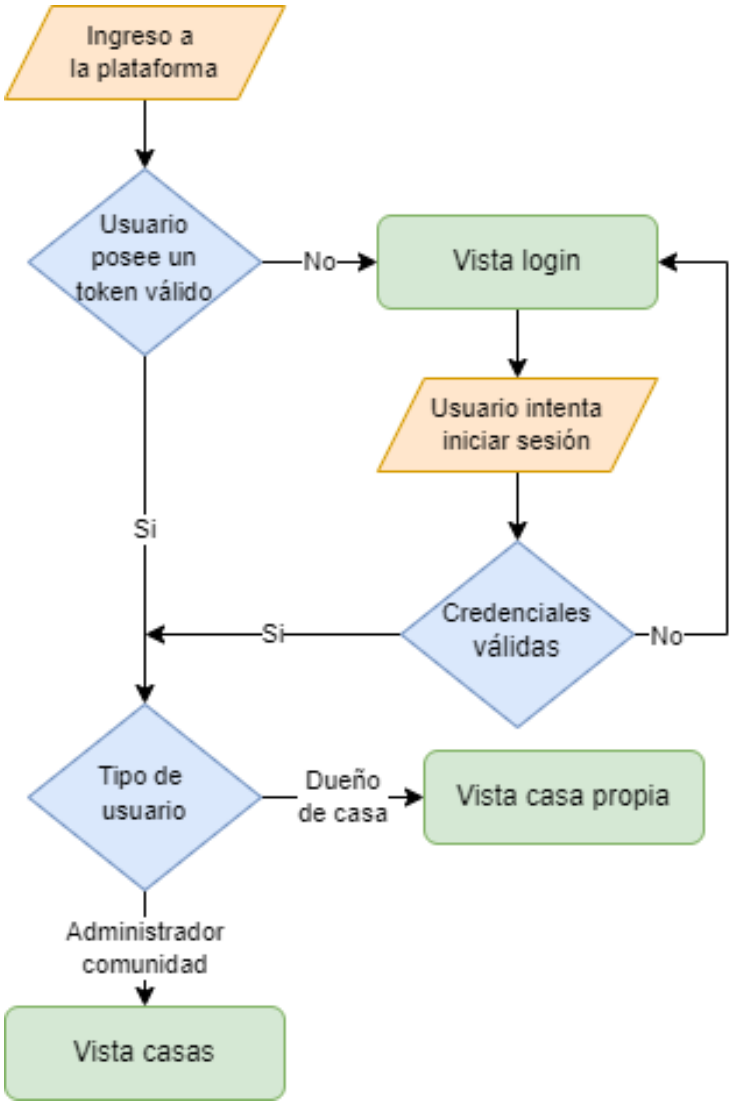


Figura 6.4: Flujo login.

6.3.3. Habitaciones

La vista de habitaciones es la principal de la aplicación web y maneja 3 variables principales:

- **rooms**: corresponde a una lista de las habitaciones que el usuario actual posee registradas. Estas son obtenidas al consultar la API con una petición GET a la ruta *https://api.ihems.cl/crud/rooms* entregando como parámetro el token de acceso.
- **currentRoom**: un objeto con el ID y el nombre de la habitación por la cual se está filtrando actualmente, ambos inicializados al comienzo de la sesión. Por defecto es

"All".

- **smartplugs**: corresponde a una lista de los dispositivos registrados por el usuario actual. Estos son obtenidas al consultar la API con una petición GET a la ruta *https://api.ihems.cl/crud/appliances* entregándole como parámetro el token de acceso.

A partir del estado de estas variables se muestra un panel con la lista de habitaciones y la lista de dispositivos filtrada de acuerdo a la habitación actualmente seleccionada.

Además cuenta con las siguientes funcionalidades:

- **Selección de habitación**: cuando el usuario escoge una habitación, esta se refleja en la variable de estado currentRoom, y se actualiza correspondientemente la lista de dispositivos mostrados.
- **Añadir habitación**: al hacer clic sobre el icono "+" junto a las pestañas de habitaciones, se abre una ventana emergente solicitando el nombre de la nueva habitación. Este se envía en el cuerpo de una solicitud POST con su respectivo token a la ruta *https://api.ihems.cl/crud/new_room*. En caso de una respuesta exitosa, se actualiza la lista rooms con la habitación recién creada, y en cualquier caso se notifica el estado de la operación mediante una alerta.
- **Modificar habitación**: al hacer clic sobre el icono de tuerca con una habitación seleccionada, se abre una ventana emergente para modificar su nombre. Este se envía en el cuerpo de una solicitud PUT con su respectivo token a la ruta *https://api.ihems.cl/crud/modify_room*. En caso de una respuesta exitosa, se actualiza la lista rooms con los cambios de la habitación, y en cualquier caso se notifica el estado de la operación mediante una alerta.
- **Eliminar habitación**: al hacer clic sobre el icono de papelera con una habitación seleccionada, se despliega una ventana emergente que solicita la confirmación del usuario, en caso de confirmar, se envía solicitud DELETE con su respectivo token e ID de habitación a la ruta *https://api.ihems.cl/crud/delete_room*. En caso de una respuesta exitosa se actualiza la lista rooms excluyendo la habitación eliminada y se establece el valor de currentRoom a "All". En cualquier caso se notifica el estado de la operación mediante una alerta. Cabe destacar que cualquier dispositivo perteneciente

previamente a esa habitación deberá ser asignado a una nueva de forma manual por el usuario.

- **Seleccionar dispositivo**: al hacer clic sobre uno de los dispositivos de la lista, se redirige al usuario a la vista del dispositivo correspondiente. Esta vista es explicada en detalle en Subsección 6.3.4.
- **Añadir dispositivo**: al hacer clic sobre el botón "+ add appliance" se abre una ventana emergente con un formulario para ingresar el ID adjunto en el dispositivo físico, el nombre que se le desea dar y la habitación a la que pertenece. Estos datos son enviados en el cuerpo de una consulta POST junto con el token de acceso a la ruta *https://api.ihems.cl/crud/new_appliance*, si se obtiene una respuesta exitosa se actualiza la lista appliances con el dispositivo recién agregado, y en cualquier caso se notifica el estado de la operación mediante una alerta.
- **Cerrar sesión**: cuando se haga clic sobre el botón de logout el token de autenticación es eliminado de la memoria del navegador, volviendo a la vista de inicio de sesión.

6.3.4. Dispositivo

Esta es la vista encargada de la visualización de las mediciones capturadas por el dispositivo especificado como parámetro en la URL de acuerdo a los ajustes dados por el usuario. Para esto posee 3 variables de estado principales:

- **currentMeasurement**: representa la medición actual que el usuario está visualizando, cuyo valor puede encontrarse entre "power", "current", "voltage", "energy", "apparent" o "reactive", siendo el primero de estos el predeterminado.
- **timeFrame**: representa el intervalo de tiempo en el que el usuario desea ver las mediciones, y su valor puede ser "Hour", "Today", "Week", "Month" o "All", siendo "Hour" el valor por defecto.
- **data**: es un arreglo que almacena los valores de todas las mediciones realizadas por el smart plug en el intervalo de tiempo especificado por timeFrame. Estos datos son mostrados en un gráfico de línea en el que el eje x representa el tiempo y el eje y la magnitud de la medida correspondiente. La información es obtenida inicial-

mente mediante una consulta GET a la ruta *https://api.ihems.cl/measurements/Hour* con el ID del dispositivo y el token de autenticación. Posteriormente se actualiza de manera automática consultando cada 10 segundos a la ruta correspondiente de *https://api.ihems.cl/measurements/* dependiendo de la variable currentMeasurement.

Las funcionalidades presentes en esta vista son las siguientes:

- **Seleccionar métrica**: cuando el usuario escoge una métrica, esta se refleja en la variable de estado currentMeasurement, y se actualizan los datos mostrados en el gráfico de acuerdo a esta.
- **Cambiar intervalo de tiempo**: a través de un menú desplegable se pueden elegir distintos intervalos de tiempo. Una vez seleccionado se actualiza la variable timeFrame y se dispara una consulta a la ruta de *https://api.ihems.cl/measurements/* correspondiente para actualizar el gráfico.
- **Modificar dispositivo**: al presionar sobre el icono de tuerca se abre una ventana emergente con un formulario para modificar el nombre o habitación a la que pertenece el dispositivo, para luego ser enviada en el cuerpo de una consulta PUT a la ruta *https://api.ihems.cl/crud/modify_appliance* con su respectivo ID y token de autenticación del usuario. Si se recibe una respuesta exitosa se actualizan las características del dispositivo, y en cualquier caso se notifica el resultado de la operación con una alerta.
- **Eliminar dispositivo**: al presionar sobre el icono de papelera, se despliega una ventana emergente que solicita la confirmación del usuario, en caso de confirmar, se envía una consulta DELETE a la ruta *https://api.ihems.cl/crud/delete_appliance* con el ID del dispositivo y el token. En caso de una respuesta satisfactoria se regresa a la vista de habitaciones actualizando la lista de dispositivos. En cualquier caso se notifica el resultado mediante una alerta.
- **Volver a vista de habitaciones**: al hacer clic sobre el botón “back” se redirige al usuario a la vista de habitaciones.
- **Cerrar sesión**: este proceso es idéntico al realizado desde la vista de habitaciones.

6.3.5. Vista general

En esta vista se visualizan los datos relacionados a las distintas fuentes asociadas al hogar del usuario, para esto cuenta con las variables de estado **currentMeasurement**, **timeFrame** y **data**, análogas a las de la vista de dispositivo, además de la variable **sources**, un objeto con los identificadores de las fuentes asociadas, obtenido al realizar una consulta **GET** a la ruta *https://api.ihems.cl/get_sources/* al cargar la página.

Cuenta también con las funcionalidades análogas de **'Seleccionar métrica'** y **'Cambiar intervalo de tiempo'**, además de la opción de agregar o modificar las fuentes del hogar, realizando una consulta de tipo PUT a la ruta *https://api.ihems.cl/crud/new_source/*, especificando el tipo de fuente y su identificador.

6.3.6. Vista administrador

Esta es la vista principal para el usuario administrador, que contiene la lista de casas bajo su administración en una única variable de estado **houses**.

Para organizar estas casas se cuenta con las siguientes funcionalidades:

- **Seleccionar casa**: al hacer click sobre una de las casas de la lista, el usuario es redirigido a la vista de casa, detallada en la siguiente subsección.
- **Añadir casa**: al hacer clic sobre el botón "+ add house" se abre una ventana emergente con un formulario para ingresar el ID de la casa a agregar y el nombre que se le desea dar. Estos datos son enviados en el cuerpo de una consulta POST junto con el token de acceso a la ruta *https://api.ihems.cl/crud/new_house*, si se obtiene una respuesta exitosa se actualiza la lista con la casa recién agregada, y en cualquier caso se notifica el estado de la operación mediante una alerta.
- **Cerrar sesión**: este proceso es idéntico al de las vistas mencionadas anteriormente.

6.3.7. Casa

Esta vista se encarga de la visualización del consumo y generación de energía de la casa especificada como parámetro en la URL. Para esto, al igual que las vistas

general y de dispositivo, cuenta con las variables de estado relacionadas a los gráficos textbfcurrentMeasurement, **timeFrame** y **data**, y las funcionalidades de **'Seleccionar métrica'** y **'Cambiar intervalo de tiempo'**.

6.4. Emulación Smart home

Con el propósito de crear un entorno de pruebas compacto y de bajo costo para poder validar el funcionamiento de la plataforma, sin la necesidad de adquirir aparatos costosos como paneles solares, se tomó como alternativa el uso de microcontroladores NodeMCU para emular el comportamiento de un smartplug, ya que son capaces de conectarse a la red wifi mediante un chip ESP8266 del mismo tipo que utilizan los enchufes reales para mandar las mediciones.

Para mantener una coherencia en la veracidad entre los datos enviados por los dispositivos reales y los emulados, se utilizaron datos provenientes del dataset Dataport de Pecan Street [27]. Este dataset contiene mediciones del consumo y generación de energía de cientos de hogares a lo largo de varios años con distintas resoluciones de tiempo. Entre los datos existentes, se seleccionó una casa con las mediciones necesarias para la implementación en este proyecto, teniendo datos de lavadora, calefactor, microondas, refrigerador, horno eléctrico, medidor general, panel solar y vehículo eléctrico. Entre estos datos se buscó un día en particular donde cada uno de los aparatos presentara un funcionamiento relevante, los cuales fueron procesados y guardados en cada nodeMCU en un formato de archivos delimitados por coma. Específicamente, se seleccionaron los datos del hogar 1240 durante el día 05/05/2019, los cuales se encuentran graficados en la Figura 6.5. Dadas las limitaciones de memoria y almacenamiento de estos microcontroladores, se usaron los datos con resolución de un minuto, y separando el día en 8 periodos de 3 horas cada uno, almacenandolos en sus propios archivos.

Una vez teniendo los datos reales, se programaron los nodeMCU para poder acceder a estos y enviar los mensajes al gatewey de la misma forma en que lo haría un dispositivo real. Para esto, en primera instancia al iniciar su funcionamiento se conecta a la misma red wifi en la que se encuentra el gateway con las credenciales correspondientes, se sincroniza con la hora actual a través de un servidor NTP, y abre el archivo con los datos correspondientes

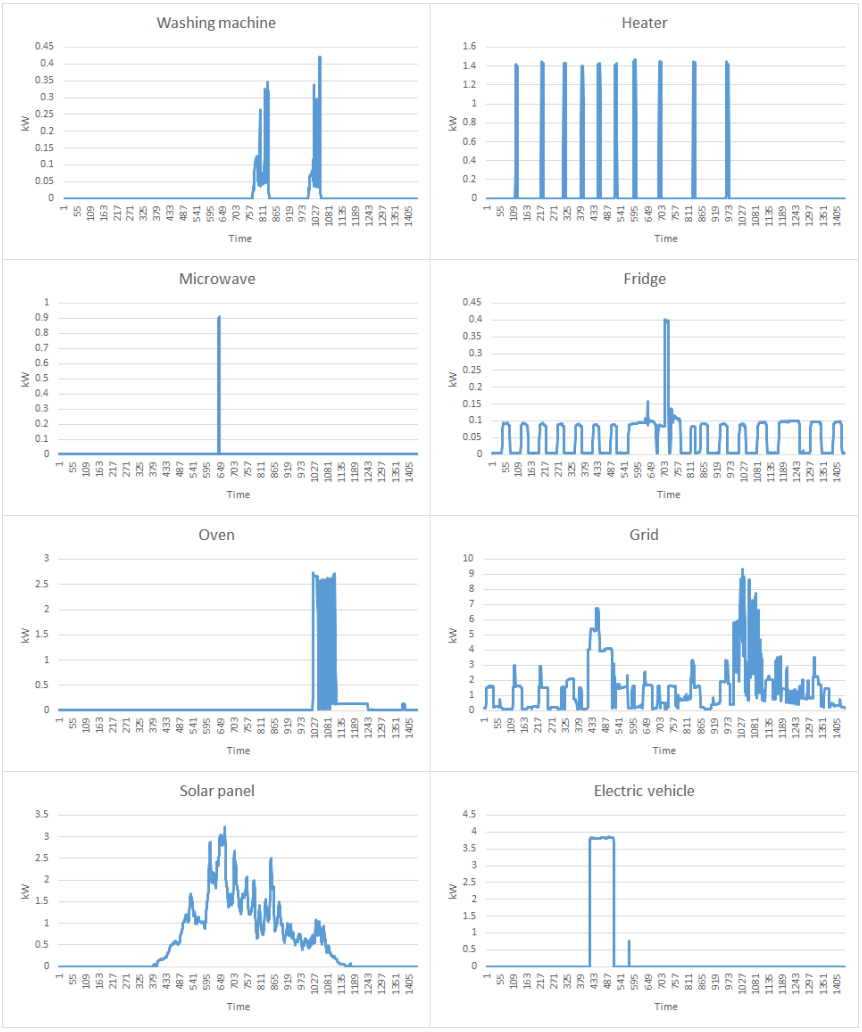


Figura 6.5: Datos Dataport.

a ese período de tiempo. Dentro de su bucle de funcionamiento, comprueba que aún se encuentra leyendo el archivo correspondiente a la hora actual, de lo contrario abre el correcto, busca en este archivo la última medición registrada previa a la hora actual, y en caso de no haberlo enviado aún, lo envía a través de un mensaje MQTT, para posteriormente repetir el bucle. Este comportamiento puede verse resumido en el diagrama de la Figura 6.6, además, el código utilizado puede encontrarse en el Anexo Listing 1.

De esta forma se tienen distintos casos de prueba para el comportamiento del sistema, representados en la Figura 6.7. En primer lugar, se tiene la perspectiva de hogar, en esta se comprueba el correcto funcionamiento de la plataforma con electrodomésticos reales encontrados en el hogar donde se desarrolló la parte relacionada al hardware de la solución. Estos electrodomésticos incluían un calefactor, una lavadora, un refrigerador y dos

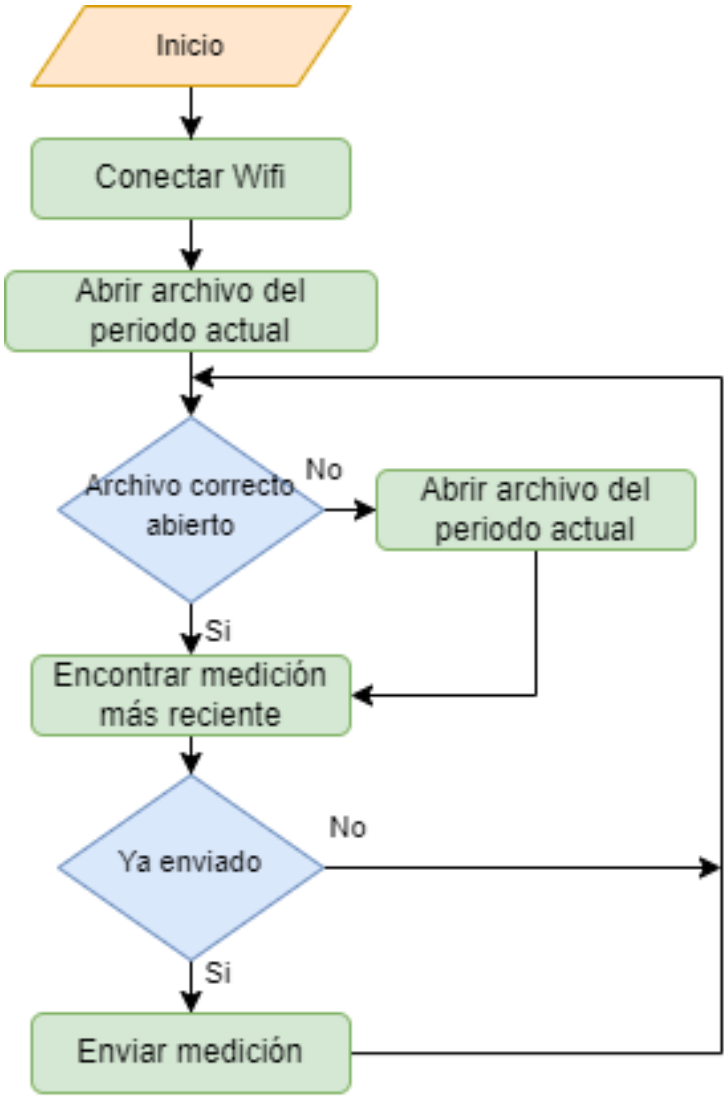


Figura 6.6: Funcionamiento NodeMCU.

televisiones. Posteriormente se llevó a cabo el caso de prueba emulando aparatos similares, como calefactores, lavadoras o microondas. El banco de pruebas para este escenario luce como el de la Figura 6.8. En segundo lugar, una vez que se comprobó que el comportamiento de la plataforma es el mismo para dispositivos reales y emulados, es posible utilizar la emulación para probar el funcionamiento desde la perspectiva de comunidad, y con aparatos de difícil acceso, como paneles solares o vehículos eléctricos. En este caso la existencia de distintos hogares fue emulada mediante una segunda raspberry configurada como gateway de manera similar a la primera, pero conectadas a redes locales distintas sin contacto entre ellas. Los resultados obtenidos de estas pruebas son discutidos en mayor detalle en el capítulo siguiente.

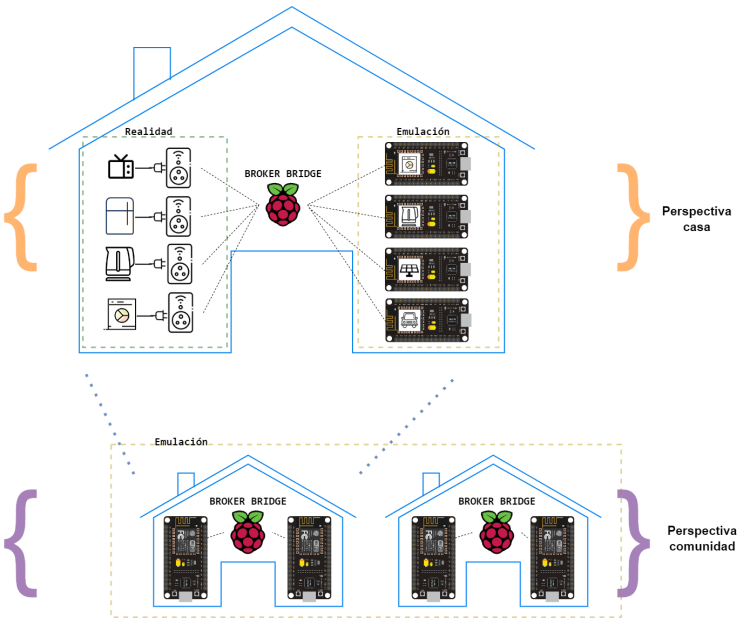


Figura 6.7: Diagrama casos de prueba.

Figura 6.8: Banco de pruebas emulación hogar.

7 | Resultados

En esta sección se muestran todos los resultados obtenidos del desarrollo del desafío. Estos se observan siguiendo el flujo de los datos -desde los smart plugs hasta su visualización-.

7.1. Firmware

Con la instalación del firmware en los smart plugs se pudo visualizar el consumo detectado de los electrodomésticos objetivos por medio de un servidor web alojado en los mismos Figura 7.1.

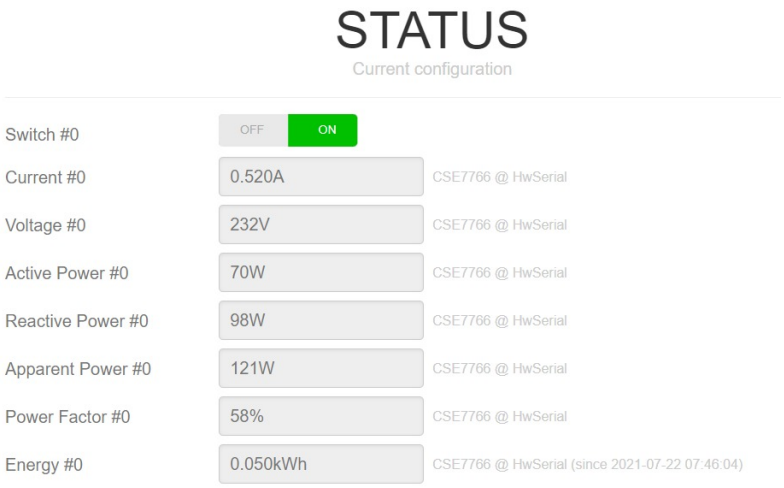


Figura 7.1: Consumo de un dipositivo en ESPurna

7.2. Comunicación

Se logró configurar la red Wi-Fi y el broker MQTT en los Smart Plugs de manera gráfica dentro de ESPurna para obtener la conectividad con el gateway, como se muestra en la Figura 7.2 .

Cuando la conectividad de los smart plugs con el gateway se estableció, se pudo observar los mensajes enviados por el protocolo de comunicación MQTT, tal como se muesta en la Figura 7.3.

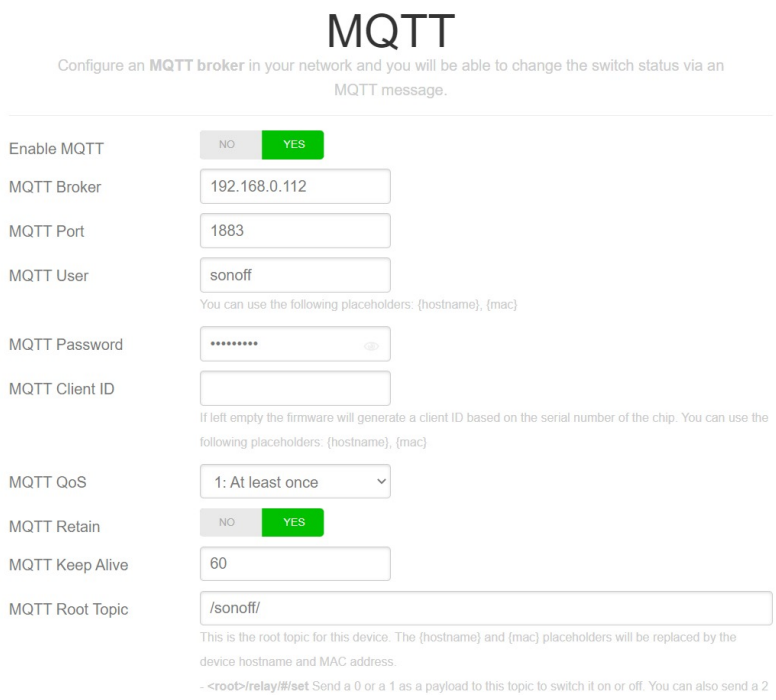


Figura 7.2: Configuración MQTT en ESPurna

7.3. Bases de datos

El envío de información al servidor por medio de MQTT se vio reflejado en el almacenamiento de la información en la base de datos InfluxDB como se muestra en la Figura 7.4.

Por otro lado la información de registro y login, junto con los dispositivos asociados a cada usuario se almaceno en una base de datos MongoDB, la que se detalla en la siguiente

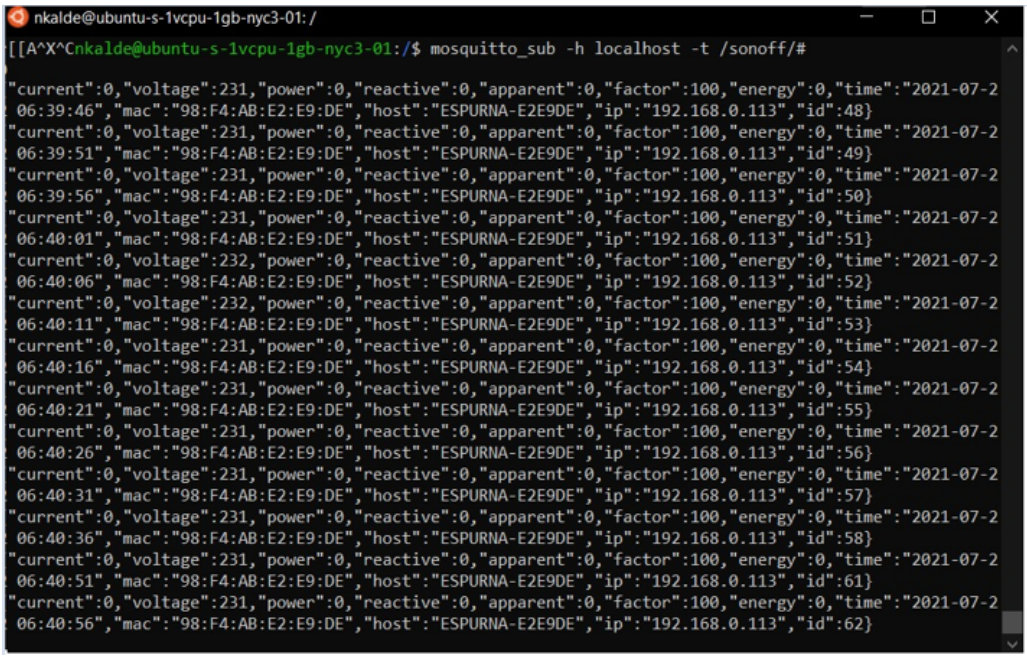


Figura 7.3: Topico sonoff en Mosquitto

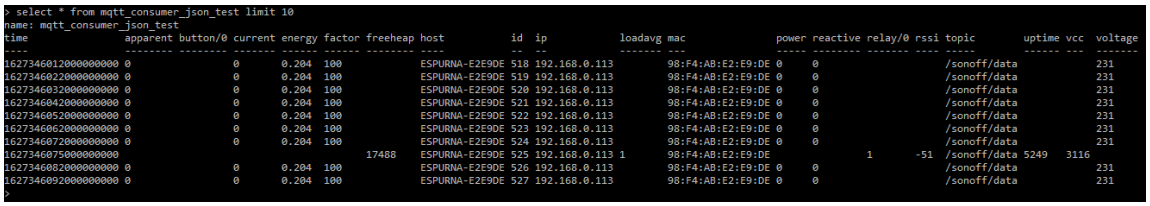


Figura 7.4: Base de datos InfluxDB

sección.

7.4. API

Para acceder a la información almacenada tanto en InfluxDB como en MongoDB se generó una API escrita en Javascript usando el framework de Express, cuyo código fuente se encuentra en https://gitlab.com/syustel/ihems/-/tree/backend/api y que fue testeada mediante Postman. Esto último consistió en evaluar las respuestas recibidas al realizar requerimientos CRUD -GET, POST, PUT, DELETE- a la url *https://api.ihems.cl/*, los que se observan en las siguientes imágenes -cabe mencionar que la explicación de el resultado esperado de cada endpoint esta descrito en Subsección 6.2.2-:

En Figura 7.5 se ve que al realizar la petición de inicio de sesión con correo y

contraseña correctos, se obtiene una respuesta exitosa con el ID del usuario, el token para su autenticación en la plataforma, y el tipo de usuario al que corresponde, pudiendo ser un usuario normal o un administrador, mientras que si las credenciales no son válidas, se obtiene un mensaje de error.

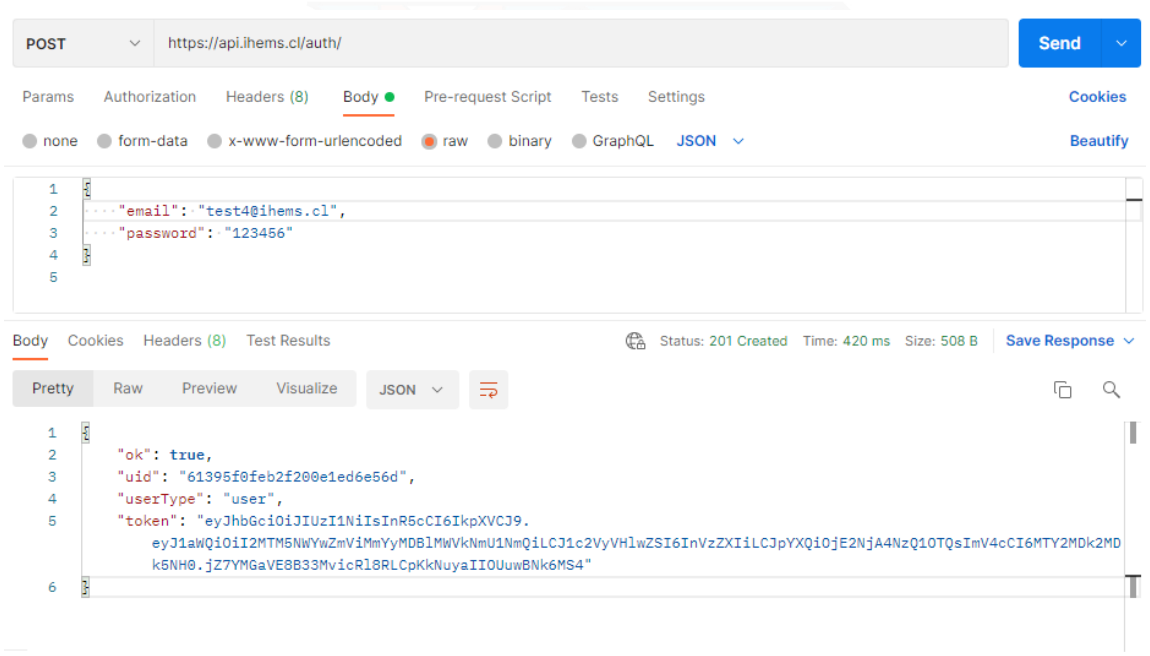


Figura 7.5: Petición POST *https://api.ihems.cl/auth/* correcta

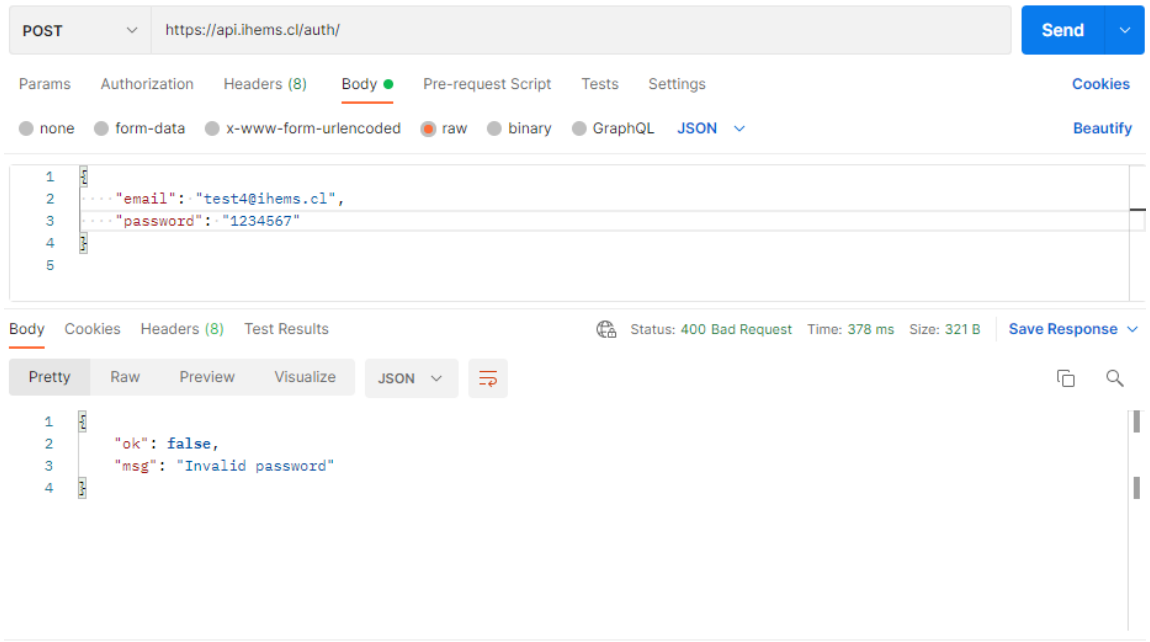


Figura 7.6: Petición POST *https://api.ihems.cl/auth/* incorrecta

Similar al caso anterior, al crear un nuevo usuario en la Figura 7.7 se entrega el ID del usuario y su token de autenticación cuando la operación se realiza correctamente.

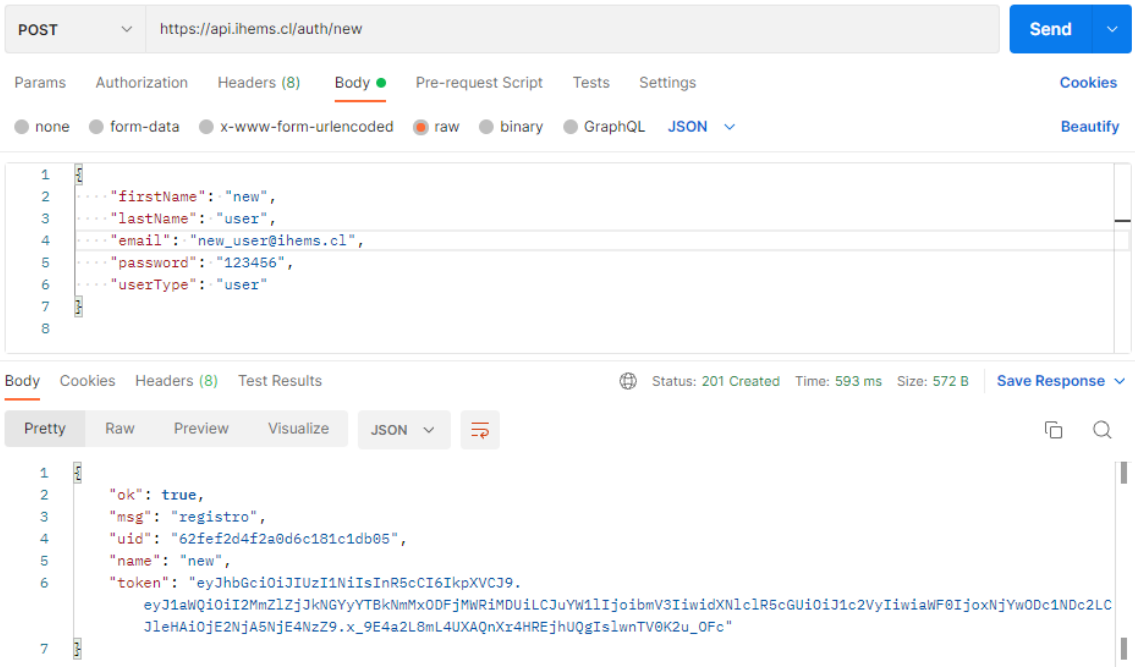


Figura 7.7: Petición POST *https://api.ihems.cl/auth/new*

En Figura 7.8, al renovar el token, cuando este es válido se retorna un mensaje de éxito junto a un nuevo token, por otro lado si el token es inválido el resultado obtenido es un mensaje de error como se ve en Figura 7.9.

La petición de habitaciones puede observarse en la Figura 7.10, donde se recibe una lista con los ID y nombre de las habitaciones del usuario, y la creación de una de estas se encuentra en la Figura 7.11.

Igualmente, la petición de dispositivos se encuentra en la Figura 7.12 mostrando la lista con su nombre, ID y habitación a la que pertenecen, y la creación de uno nuevo se encuentra en Figura 7.13.

En la Figura 7.14 se muestra la petición que retorna las distintas fuentes vinculadas para un usuario especifico, mientras que la Figura 7.15 muestra la petición encargada de registrar una de estas.

En el caso de los endpoints utilizados por lo administradores de comunidad, en la Figura 7.16 se encuentra el resultado de la petición de los hogares bajo la supervisión del

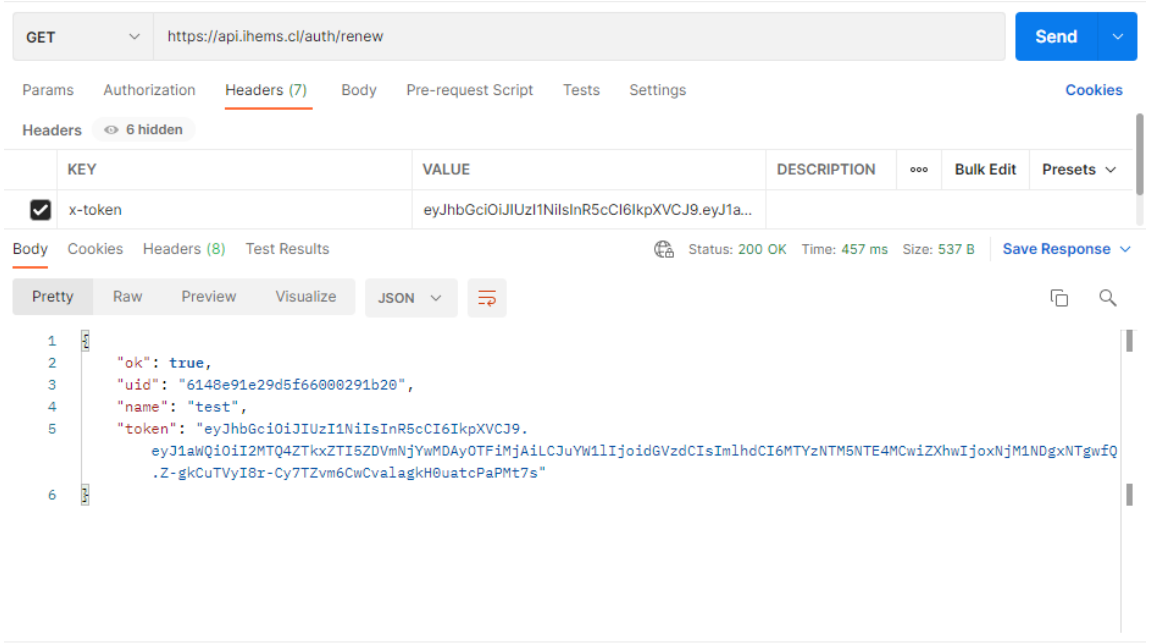


Figura 7.8: Petición GET *https://api.ihems.cl/auth/renew* token válido

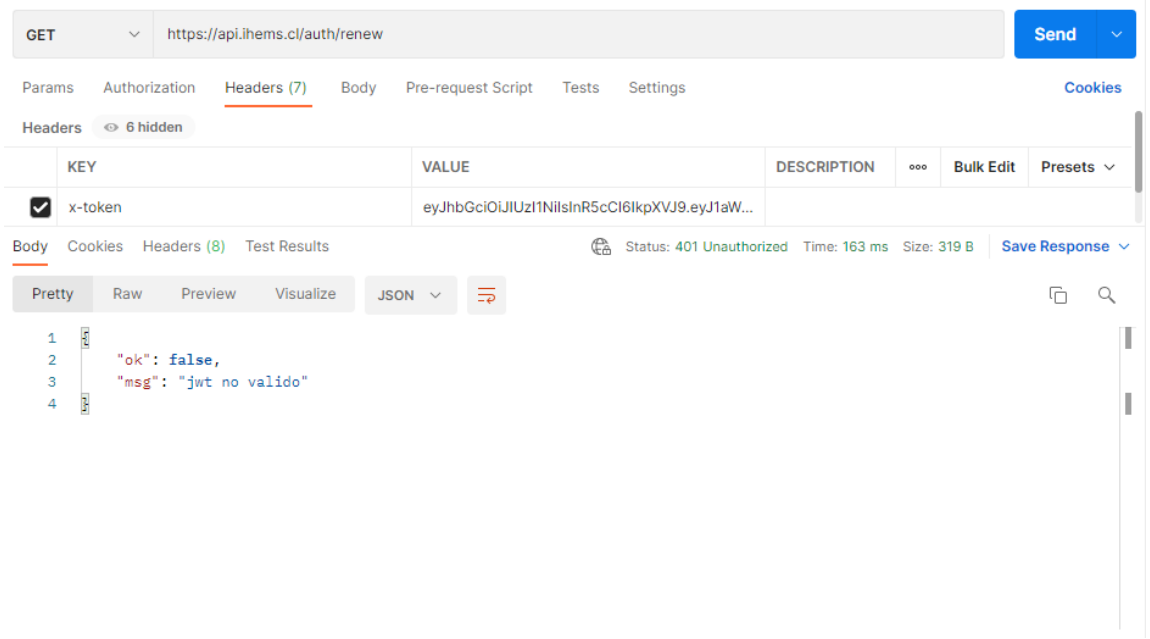


Figura 7.9: Petición GET *https://api.ihems.cl/auth/renew* token inválido

administrador. El registro de estos hogares en la base de datos se muestra en la petición de la Figura 7.17.

Finalmente, la solicitud de las mediciones se ve en Figura 7.18, siendo el resultado similar para los distintos intervalos de tiempo.

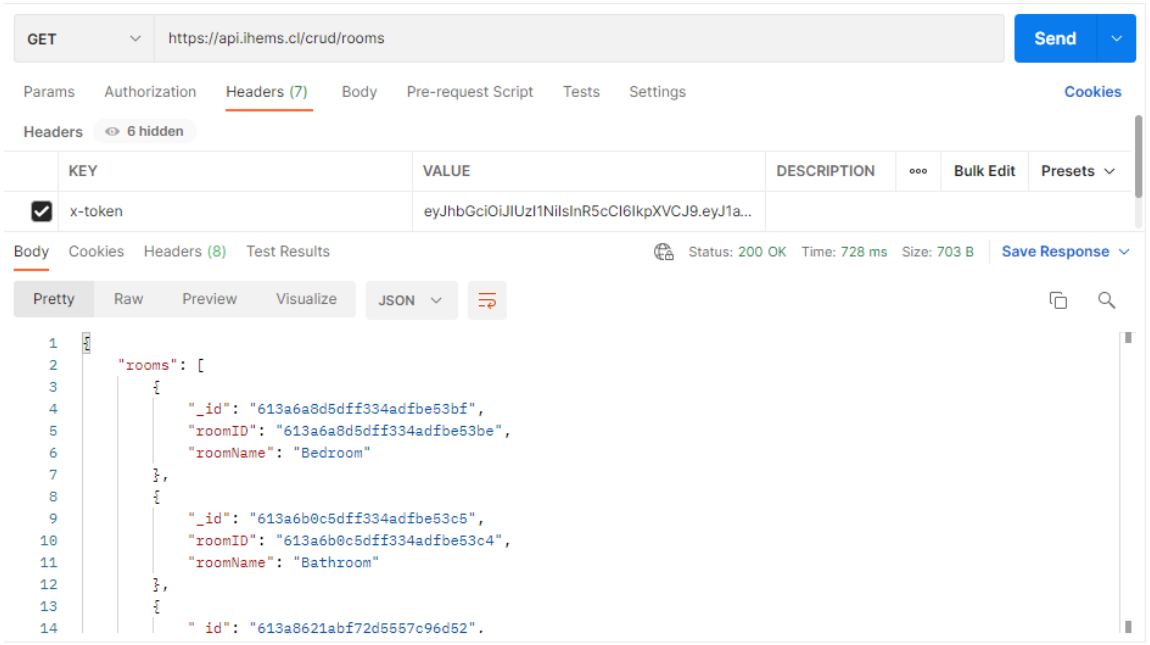


Figura 7.10: Petición GET *https://api.ihems.cl/crud/rooms*

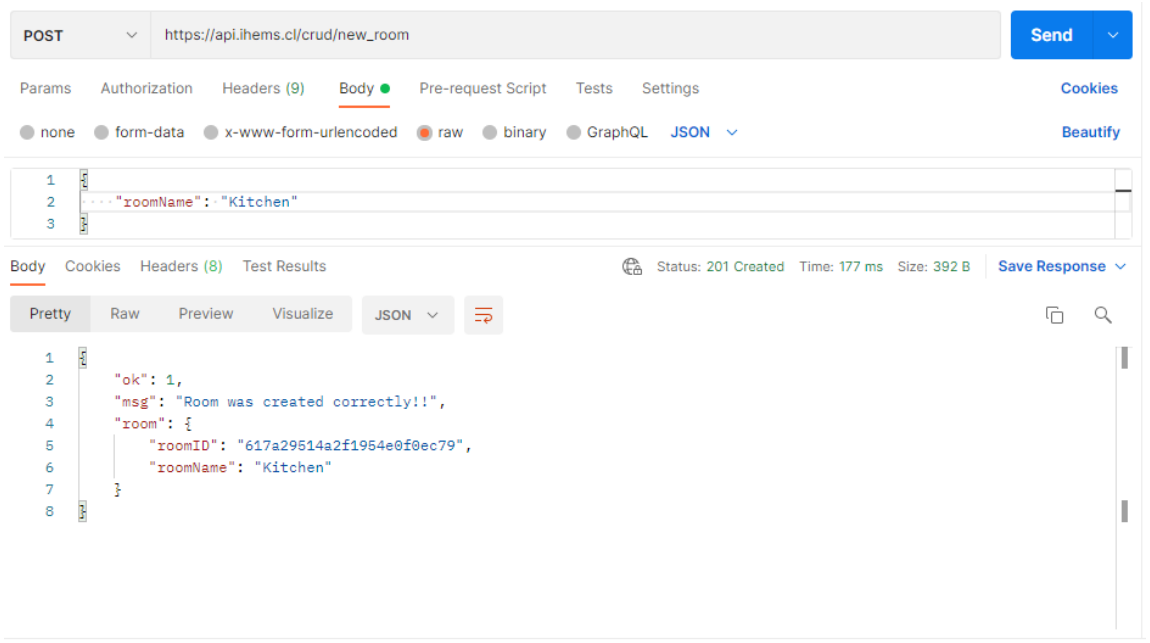


Figura 7.11: Petición POST *https://api.ihems.cl/crud/new_room*

7.5. Interfaces

En esta sección se analiza el resultado obtenido de la implementación de los mockups de la plataforma usando el framework de React. El código fuente de este puede ser

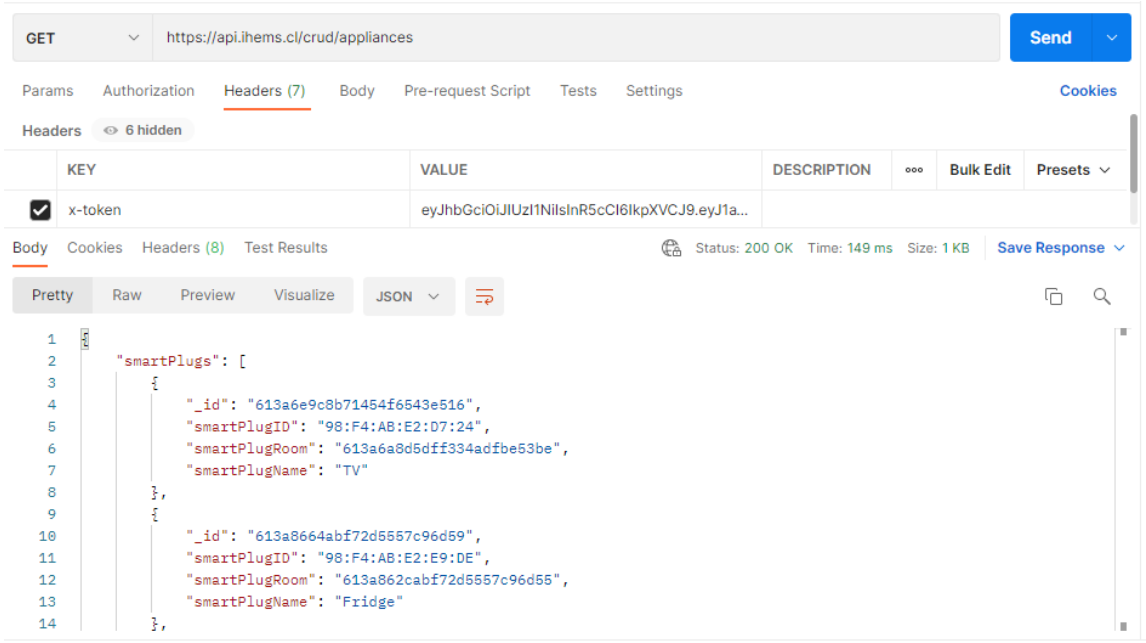


Figura 7.12: Petición GET *https://api.ihems.cl/crud/appliances*

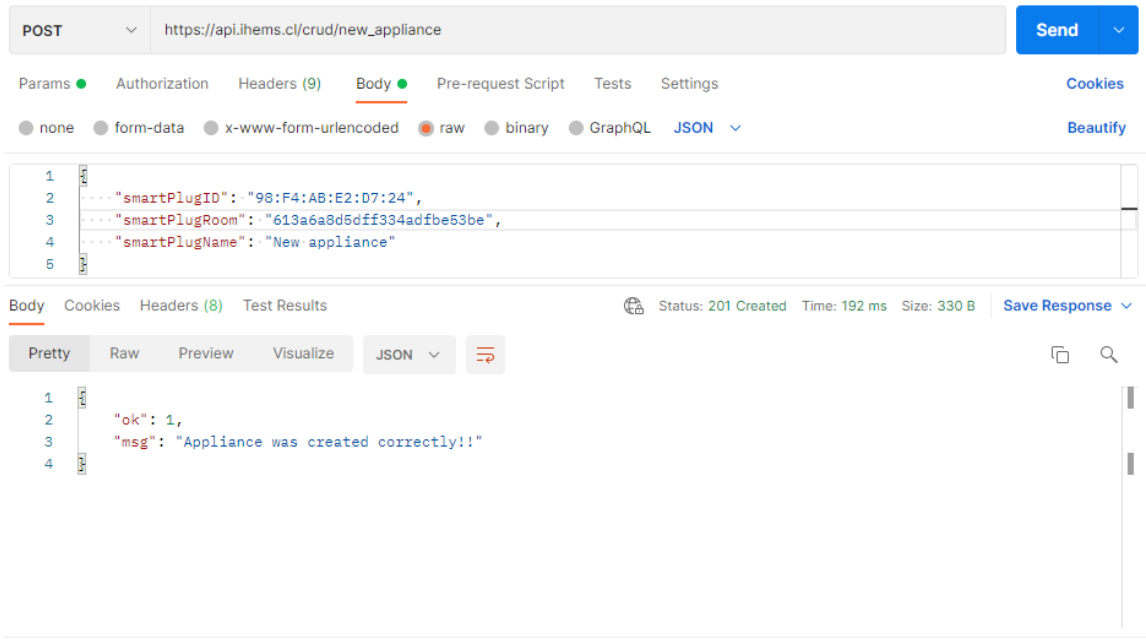


Figura 7.13: Petición POST *https://api.ihems.cl/crud/new_appliance*

encontrado en https://gitlab.com/syustel/ihems/-/tree/frontend/frontend.

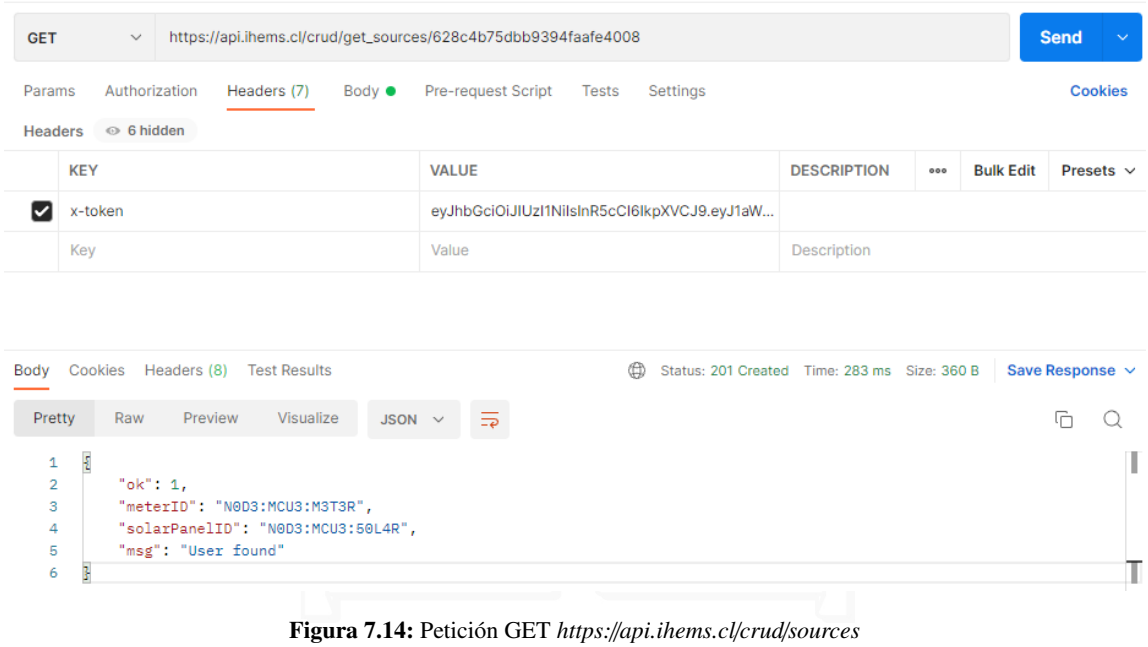


Figura 7.14: Petición GET *https://api.ihems.cl/crud/sources*

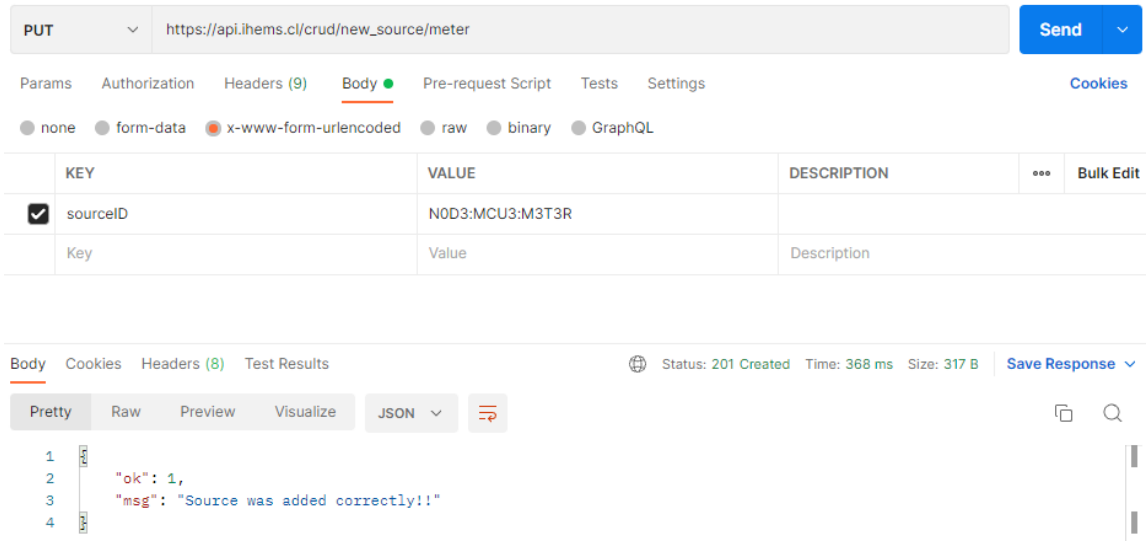


Figura 7.15: Petición PUT *https://api.ihems.cl/crud/new_source*

7.5.1. Login

La vista de inicio de sesión mostrada en la Figura 7.19 contiene todos los elementos esperados del diseño inicial, el formulario para la credenciales del usuario, acompañado del logo del proyecto y una imagen representativa de la solución, solo que con algunas diferencias en las dimensiones (al igual que en el resultado de las demás vistas). Cabe destacar que si bien se encuentran presentes visualmente las opciones de “recordar al

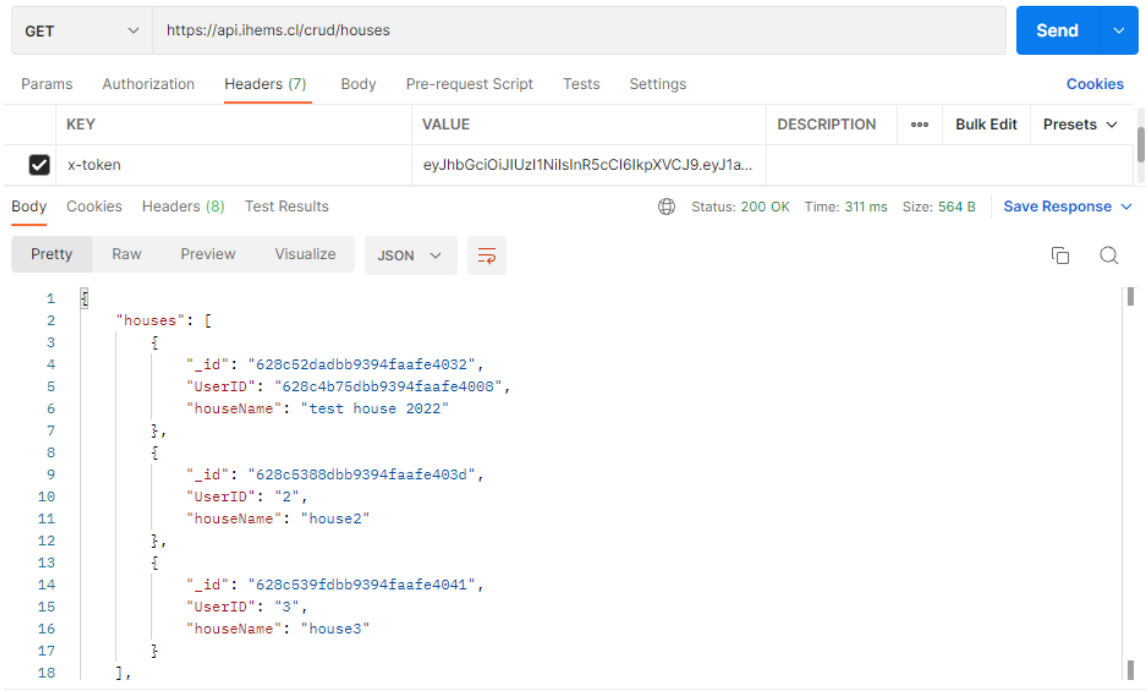


Figura 7.16: Petición GET *https://api.ihems.cl/crud/houses*

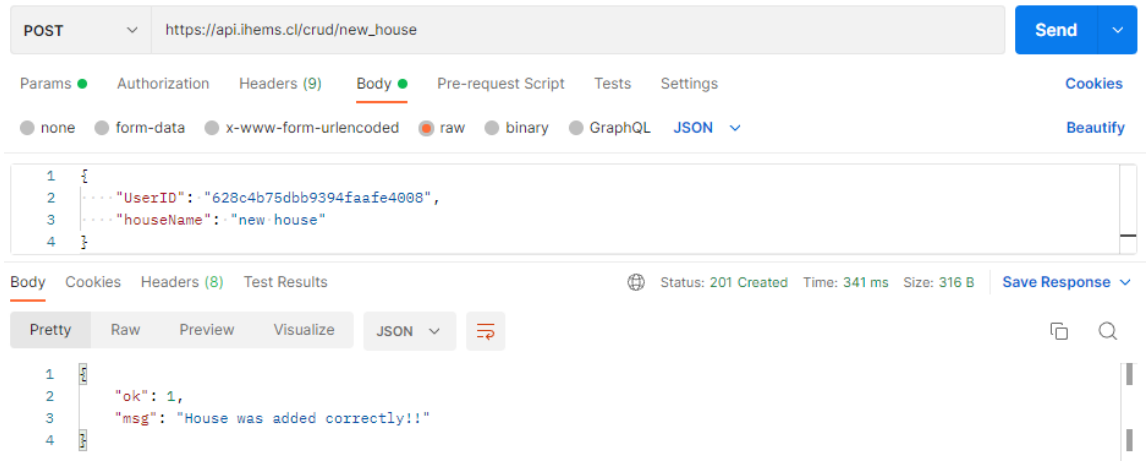


Figura 7.17: Petición POST *https://api.ihems.cl/crud/new_house*

usuario" y de "recuperar contraseña", estas no son funcionales hasta el momento.

7.5.2. Sign Up

La vista de registro, como se ve en la Figura 7.20 también cuenta con todos los elementos esperados de su diseño.

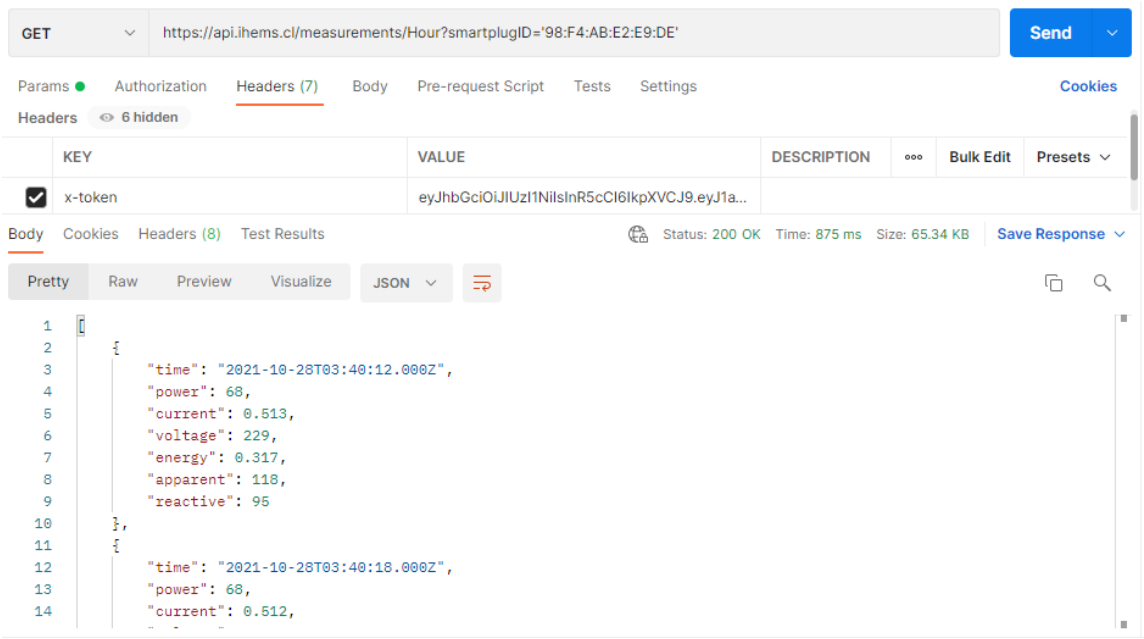


Figura 7.18: Petición GET *https://api.ihems.cl/measurements/Hour*

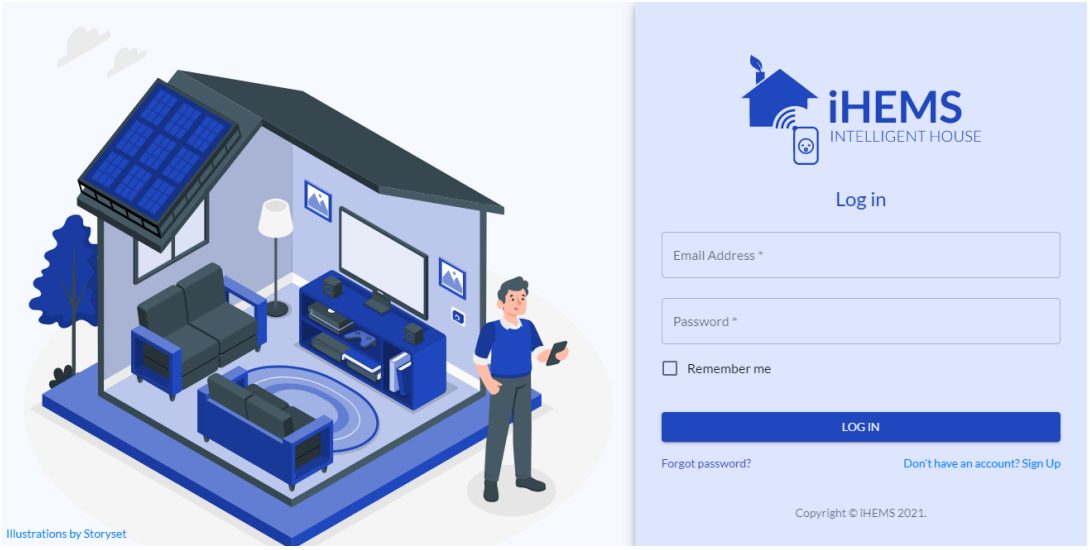


Figura 7.19: Vista login

7.5.3. Habitaciones

En el caso de un usuario dueño de casa, al iniciar sesión con una cuenta nueva, este aun no posee ninguna habitación ni dispositivo vinculado, como se ve en Figura 7.21. En la Figura 7.22 y Figura 7.23 se ven los menús para registrar habitaciones y dispositivos respectivamente. Una vez realizada la configuración inicial por el usuario la vista queda de la forma de la Figura 7.24. Además, en las Figura 7.25 y Figura 7.26 se muestran los menús

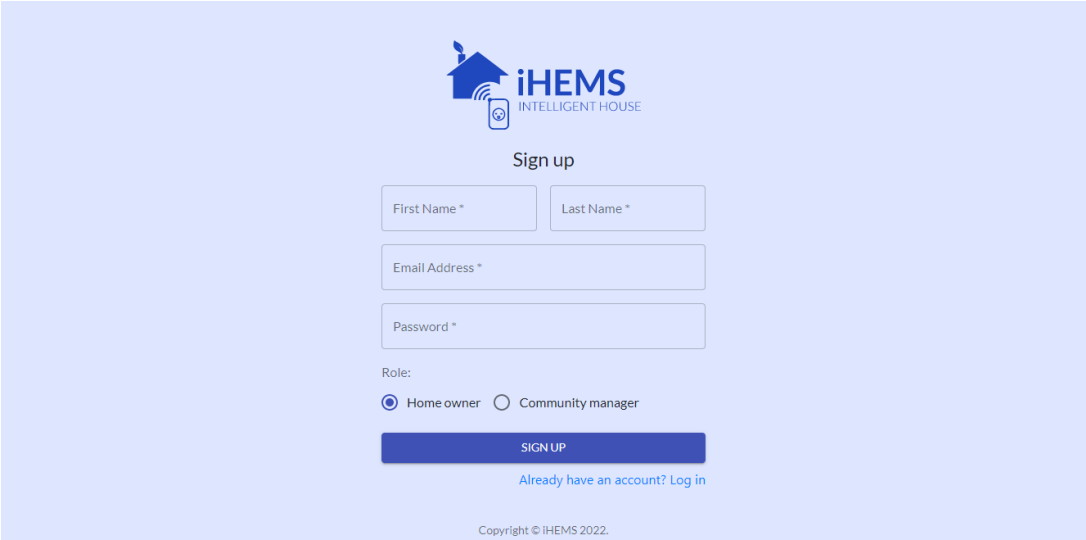


Figura 7.20: Vista signup

de modificación y confirmación de eliminación de habitaciones respectivamente.

Una diferencia importante de este resultado con respecto al diseño planteado es la forma en la que se despliegan los dispositivos, siendo una lista en vertical en lugar de en una cuadricula. Además, en la barra de navegación se encuentra solamente un botón para cerrar sesión, y no un menú del perfil de usuario.

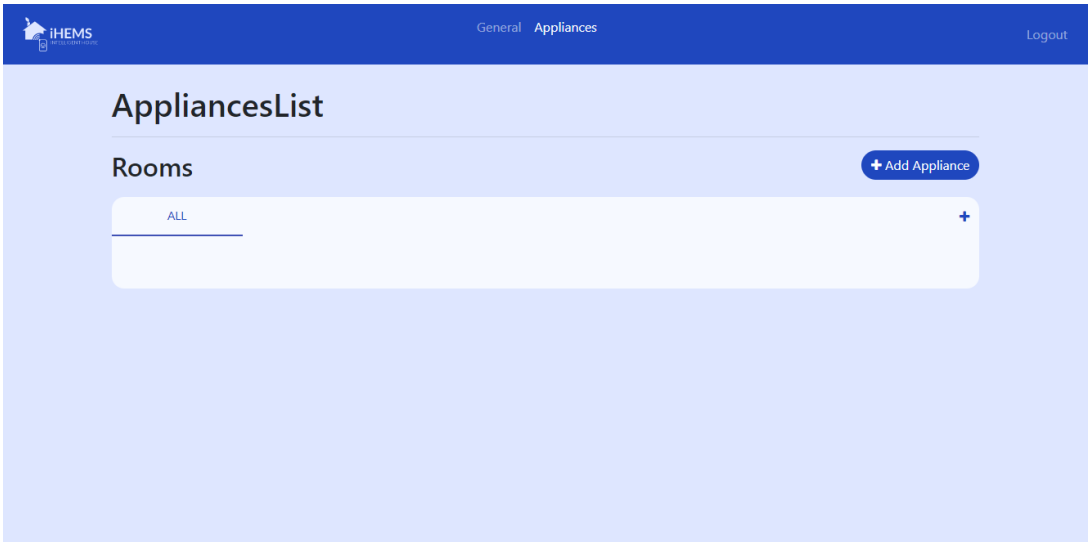


Figura 7.21: Vista habitaciones nuevo usuario

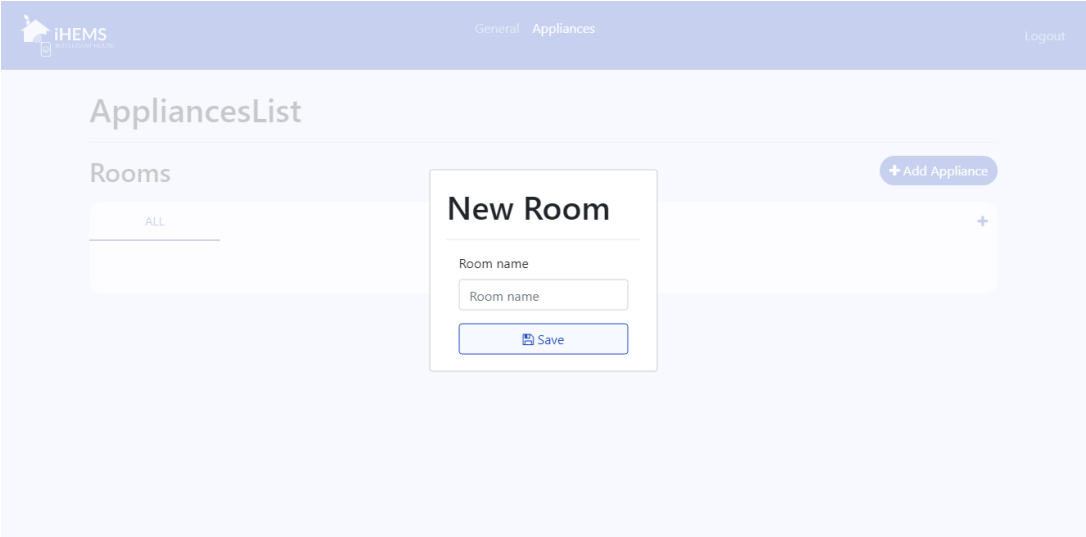


Figura 7.22: Menú agregar habitación

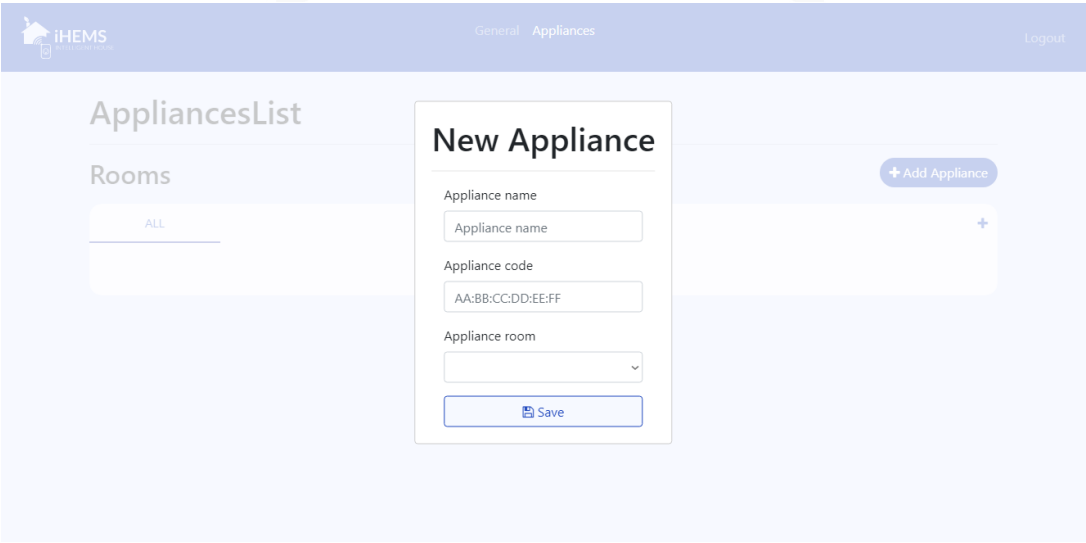


Figura 7.23: Menú agregar dispositivo

7.5.4. Dispositivo

En la Figura 7.27 se ve el resultado de la vista de dispositivo, cuya principal diferencia con el diseño propuesto es la falta del icono para identificar el electrodoméstico al que está conectado. Además, en la Figura 7.28 puede verse el gráfico resultante de uno de los dispositivos emulados, comprobándose el funcionamiento de la plataforma para ambos casos de prueba. También se puede ver en Figura 7.29 y Figura 7.30 los menús para modificar y confirmar la eliminación de un dispositivo respectivamente.

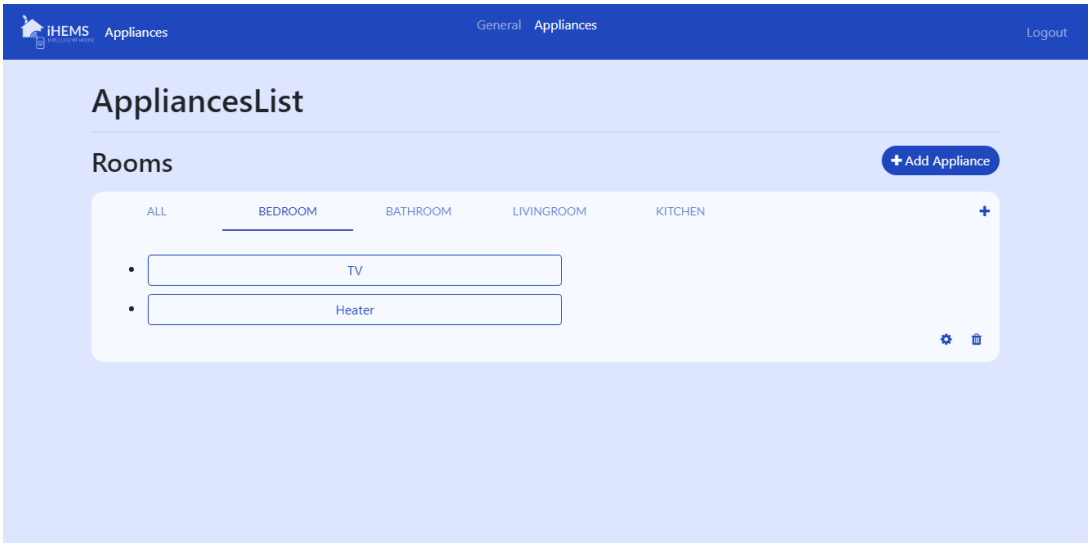


Figura 7.24: Vista habitaciones

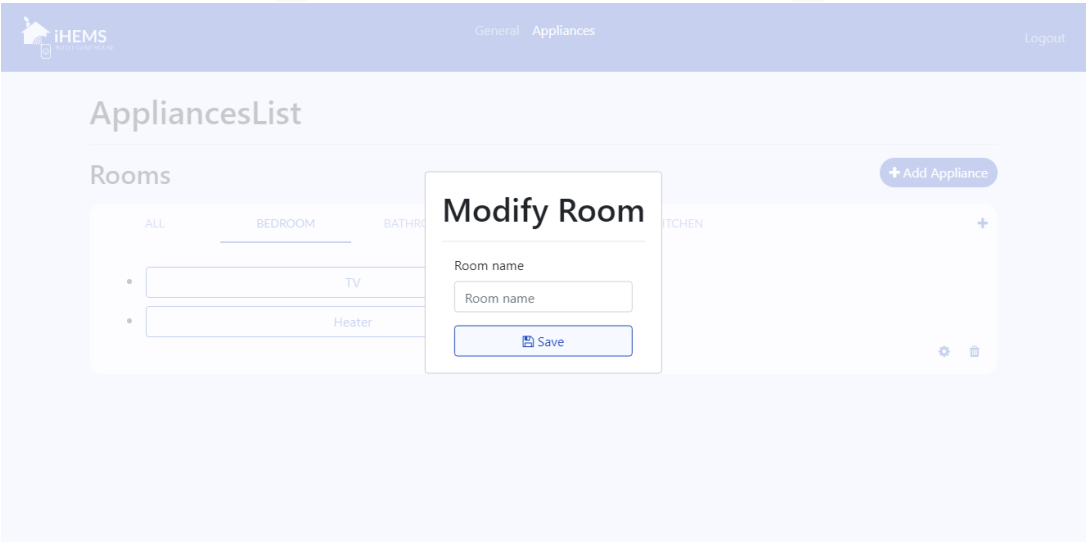


Figura 7.25: Menú modificar habitación

7.5.5. Vista general

Dado que es posible probar el funcionamiento de la plataforma con los dispositivos emulados, la vista general, como se ve en la Figura 7.31, muestra las gráficas de las fuentes con datos provenientes de los dispositivos emulados, en este caso del consumo total de un hogar y la generación por parte de un panel solar.

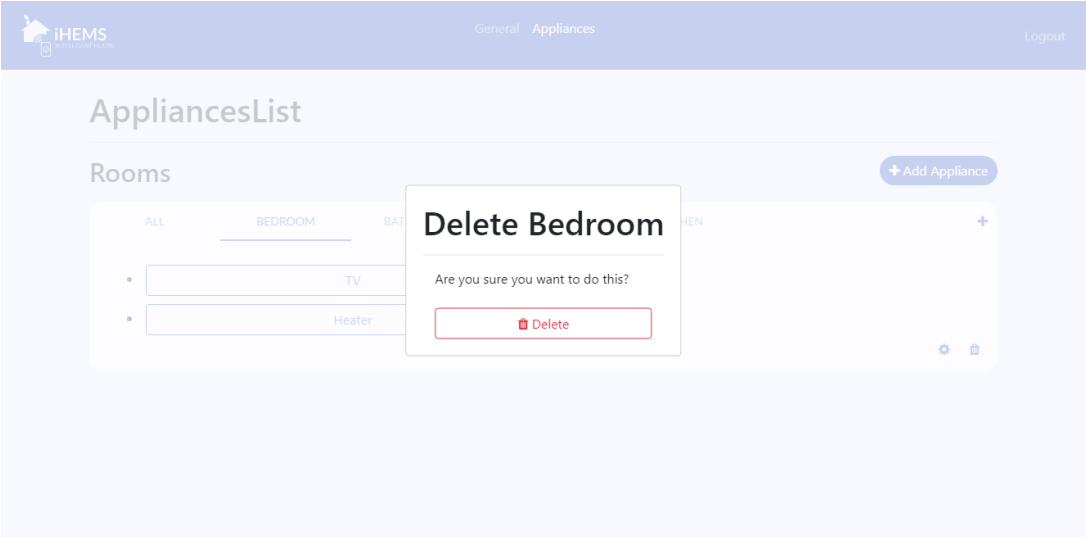


Figura 7.26: Menú eliminar habitación

Figura 7.27: Vista dispositivo

7.5.6. Vista administrador

En el caso de un usuario administrador de la comunidad, al registrarse por primera vez no cuenta con los hogares asociados a su comunidad, como se ven en la Figura 7.32. Una vez asociados los hogares a su cuenta, esta queda como se muestra en la Figura 7.33.

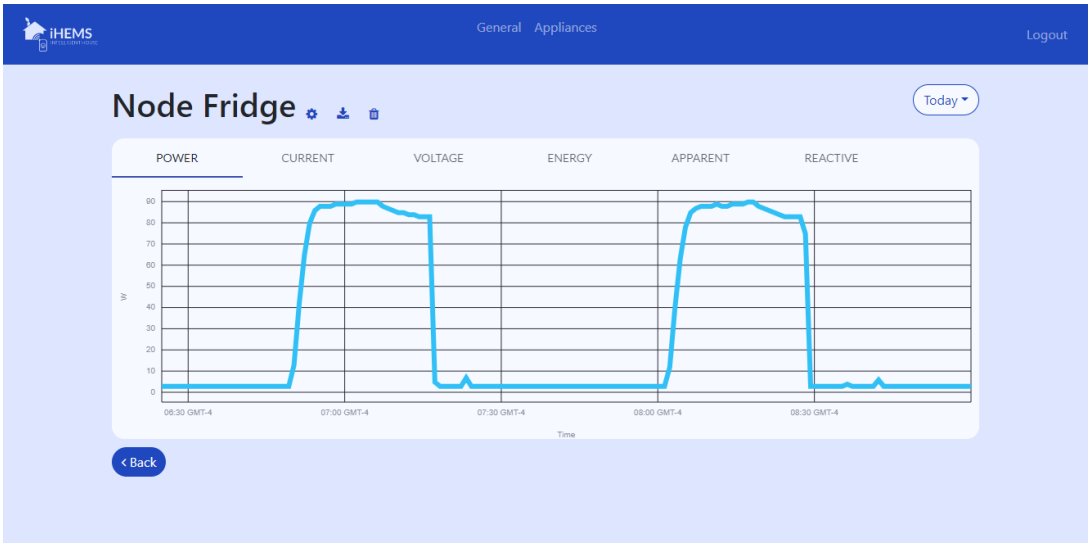


Figura 7.28: Vista dispositivo emulado

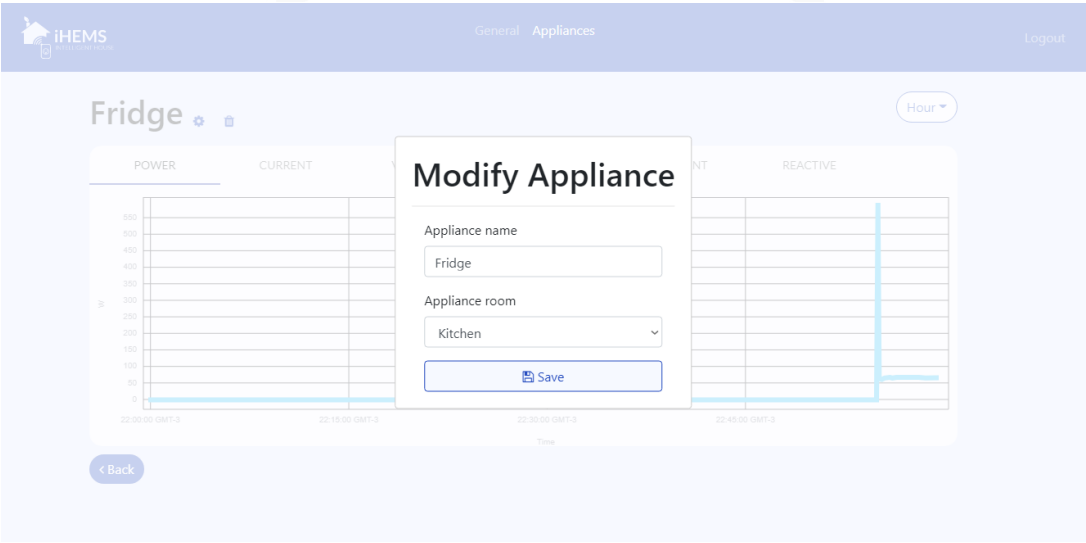


Figura 7.29: Menú modificar dispositivo

7.5.7. Vista casa

Finalmente, un usuario administrador es capaz de visualizar el consumo y generación de cada casa especifica bajo su administración, de manera similar a como lo haría el usuario dueño de esa casa, un ejemplo de esto puede verse en la Figura 7.34, donde nuevamente se destaca el uso de dispositivos emulados para obtener los datos a graficar.

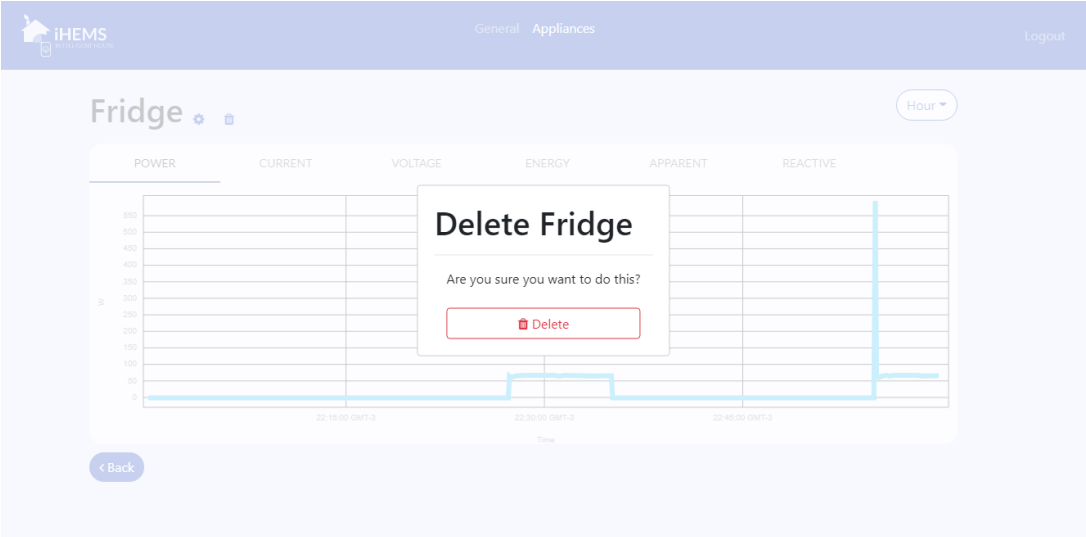


Figura 7.30: Menú eliminar dispositivo

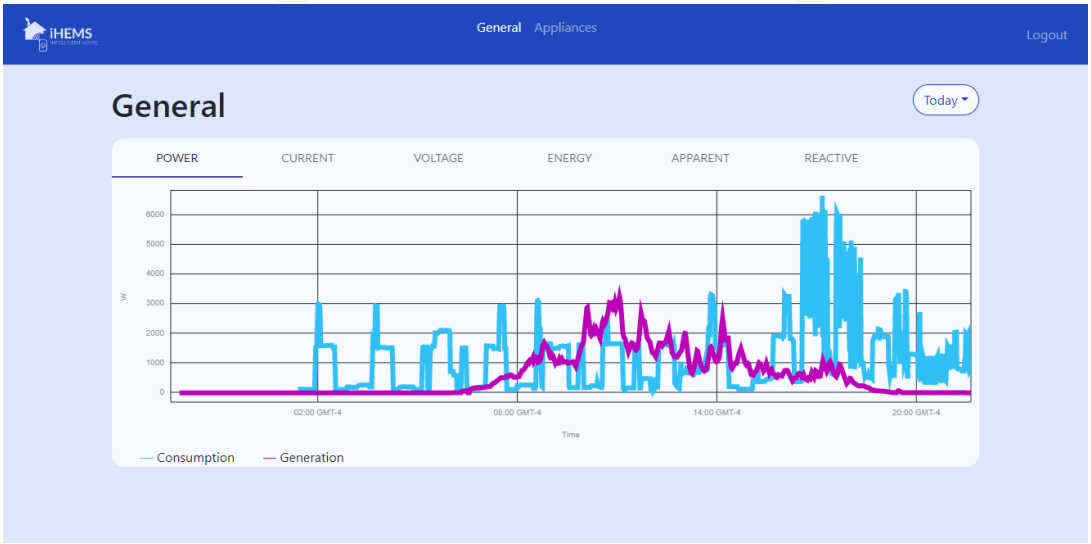


Figura 7.31: Vista general

7.6. Verificación y Validación

Los requerimientos definidos en el Capítulo 4 y los constantes cambios que estos han sufrido, han sido motivados y validados gracias a reuniones cada semana con el cliente, quien ha dejado claro las necesidades del proyecto y sus prioridades. El avance en el cumplimiento de estos requerimientos puede verse en la siguiente lista asociada a su respectiva validación:

- RF1: Listo

Figura 7.32: Vista casas nuevo usuario

Figura 7.33: Vista casas

- La capacidad de medir el consumo eléctrico se observa en la Figura 7.1, donde se muestra propiedades físicas tales como: potencia, corriente, voltaje, entre otros.

- RF2: Listo
- RF3: Listo
  - Como se observa en la Figura 7.3, el smart plug tiene conectividad al gateway y a su vez envía la información a él. Esto valida el requisito funcional 2 y 3.
- RF4: Listo

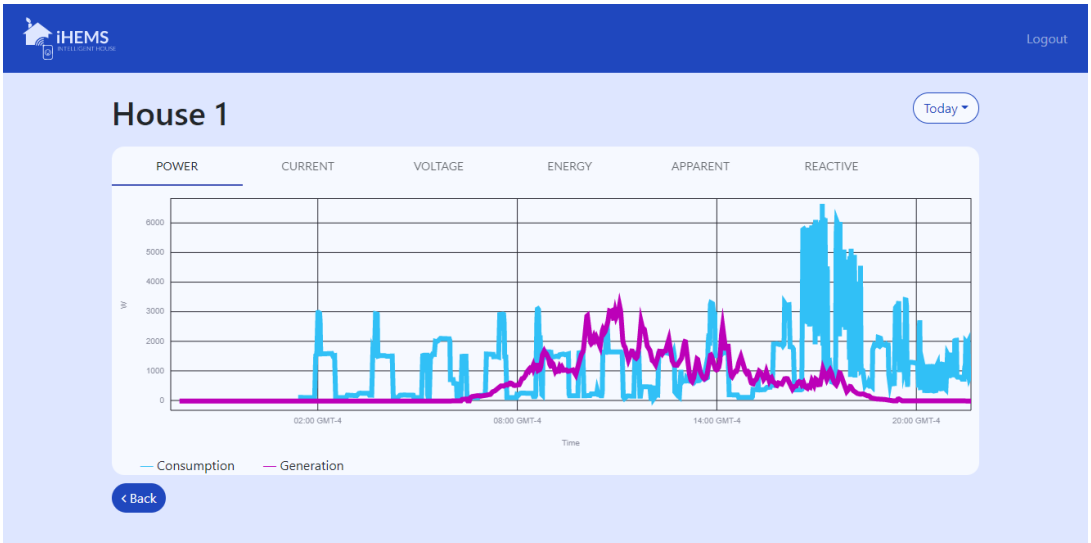


Figura 7.34: Vista casa

- En la Figura 7.2 se muestra que el smart plug se conecta al gateway mediante autenticación de usuario y contraseña.

- RF5: Listo
- RF6: Listo
  - Estos requisitos 5 y 6 se comprueban con la Figura 7.4, ya que se observa dentro de la base de datos los datos almacenados directamente por Telegraf en InfluxDB
- RF7: Posibilidad de mejoras
- RF8: Listo
- RF9: Listo
- RF10: Listo
- RF11: Listo
- RF12: Listo
- RF13: Listo
- RF14: Listo
- RF15: Listo
  - Los requisitos funcionales del 7 al 15 se pueden verificar en la Sección 7.4 y

Sección 7.5 donde se muestran capturas de pantalla de Postman que demuestran el funcionamiento de la API. También se muestran capturas de la plataforma.

- RF16: En desarrollo

8 | Conclusión

En este documento se presentó el desarrollo de una plataforma de monitoreo energético mediante un modelo de monitoreo de cargas intrusivo. Como se observó en el Capítulo 7, se dio cumplimiento a los requerimientos de la empresa, generando un prototipo de red y aplicación funcional, logrando recoger, transmitir, almacenar y desplegar datos provenientes de electrodomésticos utilizando los smart plug Sonoff Pow R2. Respecto a la plataforma, esta cumple con las demandas de la empresa, permitiendo que un usuario: se autentique en la aplicación con el uso de credenciales; pueda crear una cuenta para acceder a la plataforma en caso de no poseer una; pueda acceder a sus datos de consumo eléctrico ordenado por habitación y dispositivo, además de agregar, modificar o eliminar cualquiera de las habitaciones y dispositivos mencionados anteriormente; y por último la plataforma notifica al usuario en cada acción que realiza al interactuar con ella, con el fin de generar una experiencia agradable para el usuario al utilizarla.

Trabajos Futuros

Si bien el trabajo realizado cumple con las expectativas de la empresa, se propone realizar las siguientes mejoras:

- Instalación de OpenWRT a Gateway, con el objetivo de facilitar al usuario la tarea de dar conectividad al Gateway.
- Generar un tutorial de primer ingreso, para que el usuario reconozca la función de los botones y sepa como agregar sus smart plugs para visualizar el monitoreo.
- Configurar el encendido y apagado de los electrodomésticos a través de la plataforma.
- Implementación completa de la vista de administrador para poder extraer la informa-

ción del consumo de los usuarios y poder entrenar modelos de machine learning para uso de la empresa.

- Asociación de smart plugs con código QR, para facilitar la conectividad entre los dispositivos y la plataforma.

Estas mejoras quedan como trabajos futuros a aquellas personas que continúen con el trabajo realizado en este documento y tienen el objetivo de mejorar la calidad de la solución entregada.

Bibliografía

[1] Shashank Singh, Amit Roy, and M.P. Selvan. Smart load node for nonsmart load under smart grid paradigm: A new home energy management system. *IEEE Consumer Electronics Magazine*, 8(2):22–27, 2019. 1.1

[2] Cristóbal Escobar y Juan Francisco Garcés. El mundo y los smart grids: ¿que tan cerca estamos? *Pontificia Universidad Catolica de Chile*, pages 1–31, 2009. 2.1

[3] What is the internet of ihings (iot)? https://www.oracle.com/internet-of-things/what-is-iot/. 2.2

[4] Ying Wu, Yanpeng Wu, Josep M Guerrero, Juan C Vasquez, Emilio J Palacios-Garcia, and Jiao Li. Convergence and interoperability for the energy internet: From ubiquitous connection to distributed automation. *IEEE Industrial Electronics Magazine*, 14(4):91–105, 2020. 2.3

[5] Patricia Franco, José Manuel Martínez, Young-Chon Kim, and Mohamed A Ahmed. Iot based approach for load monitoring and activity recognition in smart homes. *IEEE Access*, 9:45325–45339, 2021. 2.4, 4, 4.1, 4.2, 4.4

[6] Bin Zhou, Wentao Li, Ka Wing Chan, Yijia Cao, Yonghong Kuang, Xi Liu, and Xiong Wang. Smart home energy management systems: Concept, configurations, and scheduling strategies. *Renewable and Sustainable Energy Reviews*, 61:30–40, 2016. 2.6

[7] Cuadro normativo y tabla de espacios y usos mínimos para el mobiliario., 2017. 2.7

[8] Muhammad Aziz, Takuya Oda, Takashi Mitani, Yoko Watanabe, and Takao Kashiwagi.

Utilization of electric vehicles and their used batteries for peak-load shifting. *Energies*, 8(5):3720–3738, 2015. 3, 3.1, 3.2, 3.2

[9] Yuto Yoshimura, Tomoaki Kondo, Michihiro Kawanishi, Tatsuo Narikiyo, and Akinori Sato. Model predictive control of ev storage battery with hems based on particle swarm optimization. In *2015 IEEE Innovative Smart Grid Technologies-Asia (ISGT ASIA)*, pages 1–5. IEEE, 2015. 3, 3, 3.2, 4.6.1

[10] T Yamaguchi, M Sumiya, S Inagaki, T Suzuki, A Ito, M Fujita, and J Kanamori. Model predictive control of car storage battery in hems considered car traveling. In *The SICE Annual Conference 2013*, pages 1352–1358. IEEE, 2013. 3, 3.2

[11] Sami Abdullah Ben Slama. Design and implementation of home energy management system using vehicle to home (h2v) approach. *Journal of Cleaner Production*, page 127792, 2021. 3, 3.3, 3.2, 4.6.1

[12] Mohammad Shakeri, Mohsen Shayestegan, SM Salim Reza, Iskandar Yahya, Badariah Bais, Md Akhtaruzzaman, Kamaruzzaman Sopian, and Nowshad Amin. Implementation of a novel home energy management system (hems) architecture with solar photovoltaic system as supplementary source. *Renewable energy*, 125:108–120, 2018. 3, 3.2

[13] MA Abouelela and MM Abouelela. Wireless communication role in home energy management system (hems). In *2015 23rd Telecommunications Forum Telfor (TELFOR)*, pages 204–207. IEEE, 2015. 3, 3.4, 3.2, 4.6.1

[14] Wei-Chen Chen, Ya-Hung Chen, Chao-Lin Wu, and Li-Chen Fu. An efficient data storage method of nosql database for hem mobile applications in iot. In *2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom)*, pages 336–339. IEEE, 2014. 3, 3.5, 3.2, 4.6.1

[15] Mani Dheeraj Mudaliar and N Sivakumar. Iot based real time energy monitoring system using raspberry pi. *Internet of Things*, 12:100292, 2020. 3, 3.2, 4.6.1

[16] Prapaporn Rattanatamrong, Yoottana Boonpalit, Siwakorn Suwanjinda, Ayuth Mangmeesap, Ken Subraties, Vahid Daneshmand, Shava Smallen, and Jason Haga.

Overhead study of telegraf as a real-time monitoring agent. In *2020 17th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 42–46. IEEE, 2020. 3, 3.1, 3, 3.2

[17] Takeshi Yashiro, Shinsuke Kobayashi, Noboru Koshizuka, and Ken Sakamura. A software hems framework for consumer electronics. In *2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE)*, pages 229–230. IEEE, 2013. 3, 3.6, 3.2, 4.6.1

[18] Kazuki Arima, Masayuki Kaneko, Takashi Murakami, Masao Isshiki, and Hiroshi Sugimura. User experience for hems based on gamification and life log. In *2016 IEEE 5th Global Conference on Consumer Electronics*, pages 1–2. IEEE, 2016. 3, 3.7, 3.8, 3.2, 4.6.1

[19] Toshiba feminity club. https://www.tlt.co.jp/tlt/products/hems/about/uchiconne.htm. 3.9

[20] G Dileep. A survey on smart grid technologies and applications. *Renewable Energy*, 146:2589–2625, 2020. 4.2

[21] Ying Wu, Yanpeng Wu, Josep M Guerrero, and Juan C Vasquez. Digitalization and decentralization driving transactive energy internet: Key technologies and infrastructures. *International Journal of Electrical Power & Energy Systems*, 126:106593, 2021. 4.3

[22] Antonio Ridi, Christophe Gisler, and Jean Hennebert. A survey on intrusive load monitoring for appliance recognition. In *2014 22nd International Conference on Pattern Recognition*, pages 3702–3707, 2014. 4.4

[23] Aws vs digitalocean. https://hackernoon.com/aws-vs-digitalocean-which-cloud-server-is-better-1386499a6664. 5.1.1

[24] Open source databases that work best for iot. https://www.opensourceforu.com/2018/05/open-source-databases-that-work-best-for-iot/. 5.1.2

[25] Jam Jahanzeb Khan Behan, Meesum Ali, Ali Inam, and Muhammad Talha Khan. Comparative analysis of rdbms and nosql databases. *DBKDA 2020*, page 38. 5.1

[26] Front-end frameworks - overview. https://2018.stateofjs.com/front-end-frameworks/overview/. 5.2

[27] Dataport - pecan street. https://www.pecanstreet.org/dataport/. 6.4

ANEXO

```
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include <PubSubClient.h>
#include <TimeLib.h>
#include <CSV_Parser.h>
#include <LittleFS.h>

// WiFi
const char* ssid = "Fe[U+FFFD]a";
const char* password = "209056089";

// NTP Servers:
static const char ntpServerName[] = "us.pool.ntp.org";

// MQTT Broker
const char* mqtt_server = "192.168.1.170";
const char *topic = "/sonoff/data";
const char *mqtt_username = "sonoff";
const char *mqtt_password = "ihems2021";
const int mqtt_port = 1883;
const char *mac = "N0D3:MCU3:R4FR1G3R4T0R";

WiFiClient espClient;
PubSubClient client(espClient);

WiFiUDP Udp;
unsigned int localPort = 8888;  // local port to listen for UDP packets

// Global variables
unsigned long lastMsg = 0;
```

```
#define MSG_BUFFER_SIZE (250)
char msg[MSG_BUFFER_SIZE];
char csv_char[2600];
int32_t *_time;
int32_t *power;
int current_measurement = 0;
int last_loaded_file = 0;
int last_sec_send = 0;

void setup() {
  Serial.begin(115200);

  // WiFi setup
  setup_wifi();

  Serial.println("Starting UDP");
  Udp.begin(localPort);
  Serial.print("Local port: ");
  Serial.println(Udp.localPort());
  Serial.println("waiting for sync");

  // NTP setup
  setSyncProvider(getNtpTime);
  setSyncInterval(300);

  // MQTT setup
  client.setServer(mqtt_server, mqtt_port);

  // LittleFS setup
  LittleFS.begin();

  // Load initial file
  int hour_4 = (hour()+20)%24;
  int file_id = hour_4/3;   // File name
  String file_name = "data";
  file_name.concat(file_id);
  file_name.concat(".csv");
  last_loaded_file = file_id;
```

```
  String csv_str = load_from_file(file_name); // Read content
  csv_str.toCharArray(csv_char, 2700);  // Convert to char array
}

void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  unsigned long _now = millis();
  if (_now - lastMsg > 30000) {
    lastMsg = _now;
    int hour_4 = (hour()+20)%24;

    // Check current loaded data
    if (last_loaded_file != hour_4/3) {
      int file_id = hour_4/3;   // File name
      String file_name = "data";
      file_name.concat(file_id);
      file_name.concat(".csv");
      last_loaded_file = file_id;

      String csv_str = load_from_file(file_name); // Read content
      csv_str.toCharArray(csv_char, 2700);  // Convert to char array
    }

    CSV_Parser cp(csv_char,"LL"); // Parse
    _time = (int32_t*)cp["time"];
    power = (int32_t*)cp["power"];

    int current_sec = hour_4*3600 + minute()*60 + second();

    for (int t = 0; t < cp.getRowsCount(); t++) {   // Find newest measurement
      if (current_sec > _time[t]) {
        current_measurement = t;
      }
    }
```

```
    if (_time[current_measurement] != last_sec_send) {  // Not send yet
      last_sec_send = _time[current_measurement];
      // Send new measurement
      snprintf(msg, MSG_BUFFER_SIZE,

    "{\"current\":0,\"voltage\":231,\"power\":%ld,\"reactive\":0,\"apparent\":0,\"factor\":100,\"ene
    %02d:%02d:%02d\",\"mac\":\"%s\",\"host\":\"Node-MCU\",\"ip\":\"192.168.1.180\",\"id\":638}",

    power[current_measurement], year(), month(), day(), hour(), minute(), second(), mac);
      Serial.print("Publish message: ");
      Serial.println(msg);
      client.publish(topic, msg);
    }
  }
}

/*-------- WiFi code ----------*/

void setup_wifi() {

  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());

  Serial.println("");
  Serial.println("WiFi connected");
```

```
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

/*-------- MQTT code ----------*/

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Create a random client ID
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    // Attempt to connect
    if (client.connect(clientId.c_str(), mqtt_username, mqtt_password)) {
      Serial.println("connected");
      // Once connected, publish an announcement...
      //client.publish("/sonoff/test", "primer mensaje");
      // ... and resubscribe
      client.subscribe("/sonoff/nodesub");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}

/*-------- NTP code ----------*/

const int NTP_PACKET_SIZE = 48; // NTP time is in the first 48 bytes of message
byte packetBuffer[NTP_PACKET_SIZE]; //buffer to hold incoming & outgoing packets

time_t getNtpTime()
{
  IPAddress ntpServerIP; // NTP server's ip address
```

```
  while (Udp.parsePacket() > 0) ; // discard any previously received packets
  Serial.println("Transmit NTP Request");
  // get a random server from the pool
  WiFi.hostByName(ntpServerName, ntpServerIP);
  Serial.print(ntpServerName);
  Serial.print(": ");
  Serial.println(ntpServerIP);
  sendNTPpacket(ntpServerIP);
  uint32_t beginWait = millis();
  while (millis() - beginWait < 1500) {
    int size = Udp.parsePacket();
    if (size >= NTP_PACKET_SIZE) {
      Serial.println("Receive NTP Response");
      Udp.read(packetBuffer, NTP_PACKET_SIZE);  // read packet into the buffer
      unsigned long secsSince1900;
      // convert four bytes starting at location 40 to a long integer
      secsSince1900 =  (unsigned long)packetBuffer[40] << 24;
      secsSince1900 |= (unsigned long)packetBuffer[41] << 16;
      secsSince1900 |= (unsigned long)packetBuffer[42] << 8;
      secsSince1900 |= (unsigned long)packetBuffer[43];
      return secsSince1900 - 2208988800UL;
    }
  }
  Serial.println("No NTP Response :-(");
  return 0; // return 0 if unable to get the time
}

// send an NTP request to the time server at the given address
void sendNTPpacket(IPAddress &address)
{
  // set all bytes in the buffer to 0
  memset(packetBuffer, 0, NTP_PACKET_SIZE);
  // Initialize values needed to form NTP request
  // (see URL above for details on the packets)
  packetBuffer[0] = 0b11100011;   // LI, Version, Mode
  packetBuffer[1] = 0;     // Stratum, or type of clock
  packetBuffer[2] = 6;     // Polling Interval
  packetBuffer[3] = 0xEC;  // Peer Clock Precision
  // 8 bytes of zero for Root Delay & Root Dispersion
```

```
  packetBuffer[12] = 49;
  packetBuffer[13] = 0x4E;
  packetBuffer[14] = 49;
  packetBuffer[15] = 52;
  // all NTP fields have been given values, now
  // you can send a packet requesting a timestamp:
  Udp.beginPacket(address, 123); //NTP requests are to port 123
  Udp.write(packetBuffer, NTP_PACKET_SIZE);
  Udp.endPacket();
}

/*-------- LittleFS code ----------*/

String load_from_file(String file_name) {
  String result = "";

  File this_file = LittleFS.open(file_name, "r");
  if (!this_file) { // failed to open the file, retrn empty result
    return result;
  }
  while (this_file.available()) {
      result += (char)this_file.read();
  }

  this_file.close();
  return result;
}
```

Listing 1: Código envío de datos NodeMCU