NUEVAS REALIDADES PARA LA EDUCACIÓN EN INGENIERÍA: CURRÍCULO, TECNOLOGÍA, MEDIO AMBIENTE Y DESARROLLO
13 - 16 DE SEPTIEMBRE 2022 CARTAGENA DE INDIAS, COLOMBIA
Encuentro Internacional de Educación en Ingeniería ACOFI

Sistema domótico centralizado. Fase control inalámbrico de cargas eléctricas

Juan Sebastián Hernández González, Rafael Martínez Bermúdez, Santiago Padilla Trujillo, Erika Sarria Navarro

Institución Universitaria Antonio José Camacho
Cali, Colombia

Resumen

En esta investigación se implementó un sistema domótico con una unidad central de procesamiento. El diseño del sistema permite que nuevas fases sean agregadas de tal forma que se pueda seleccionar en cada hogar que será instalado. La plataforma seleccionada fue la Rasberry Pi Zero W, cuyas características y precio la convierten en una opción ideal para implementaciones embebidas otorgándoles una gran capacidad de cálculo y la oportunidad del manejo de software libre La funcionalidad correspondiente a este proyecto, permite el control de encendido/apagado de módulos inalámbricos ubicados en las conexiones a la red eléctrica de ciertos dispositivos eléctricos del hogar de manera remota para, por ejemplo, que se pueda anticipar el encendido de ciertos electrodomésticos que permiten climatizar el ambiente de la casa o prender y apagar luces para simular la presencia de personas en los hogares.

La unidad central envía localmente, a través de conexión Wifi actualizaciones del estado encendido/apagado de dichos módulos de control de cargas. Dichas actualizaciones son recibidas desde la plataforma Firebase, donde a través de los servicios Realtime Database y Hosting se ha realizado la conexión para que los usuarios (habitantes del hogar donde se establece el sistema domótico) puedan acceder a la funcionalidad, a través de una aplicación en el celular. Cuando un usuario activa o desactiva una carga en la aplicación, se actualiza la base de datos en tiempo real de Firebase, y se propaga dicha actualización a la unidad central de procesamiento, y a los distintos usuarios que tengan activa la aplicación, llegando finalmente al módulo que debe ser encendido o apagado.

Se utilizó la herramienta de generación de prototipos Figma para el diseño de la interfaz de usuario remota, la cual se desarrolló usando JavaScript utilizando las librerías: React para definir la

arquitectura de la aplicación y Redux que permite manejar el estado de la aplicación y emitir actualizaciones ante acciones. Como resultado final obtenemos un sistema pensado desde su inicio de manera escalable que se convierte en el punto de partida de un sistema domótico central que abarque más características necesarias en los hogares.

Palabras clave: domótica; aplicación web con react; raspberry

Abstract

In this research, a home automation system with a central processing unit was implemented. The design of the system allows new phases to be added in such a way that they can be selected in each home that will be installed. The selected platform was the Rasberry Pi Zero W, whose characteristics and price determined it to be an ideal option for embedded implementations, granting them a great calculation capacity and the opportunity to manage free software. The functionality corresponding to this project allows remote control of the on/off of wireless modules located in the connections to the electrical network of certain electrical devices in the home, for example, so that the switching on of certain household appliances that allow climatize the environment of the house or turn lights on and off to simulate the presence of people in homes.

The central unit sends locally, via Wi-Fi connection, updates on the on/off status of said load control modules. These updates are received from the Firebase platform, where the connection has been made through the Realtime Database and Hosting services so that the users (inhabitants of the home where the home automation system is established) can access the functionality, through an application on the cell phone. When a user activates or deactivates a load in the application, the Firebase database is updated in real time, and this update is propagated to the central processing unit, and to the different users that have the application active, finally reaching the module which should be turned on or off.

Introduced Figma prototyping tool for remote UI design, which was developed using JavaScript using the libraries: React for defines the architecture of the application and Redux that allows managing the state of the application and issuing updates on actions.
As a final result, we obtain a system designed from the beginning in a scalable way that becomes the starting point of a central home automation system that encompasses the necessary characteristics in homes.

***Keywords**: domotics; web app with react; raspberry*

1. Introducción

Este proyecto nació desde el semillero de investigación SELECT, en medio de la pandemia de covid-19, con reuniones virtuales desde los hogares de docentes y estudiantes, y mirando hacia las necesidades que rodeaban el día a día de la nueva normalidad. Con esto en mente se identificaron problemas en la distribución de los elementos eléctricos dentro del hogar ya que se cuenta con un

número limitado de interruptores y tomacorrientes. Por ejemplo, las habitaciones básicas de la mayoría de hogares solo cuentan con un interruptor que normalmente se encuentra al lado de la puerta, con el fin de poder encender la luz al momento de entrar a la habitación, pero no se pensó en el instante que el usuario se encontrara en su cama y quisiera apagar o encender la luz. Del mismo modo, comúnmente solo se encuentra un tomacorriente por habitación contrario a lo que recomienda el RETIE en su artículo 28.1 sección b sobre la necesidad de por lo menos dos tomacorrientes dobles (Ministerio de Minas y Energía, 2013) y esto produce que se deban conectar extensiones o regletas alterando el ambiente de orden y comodidad. Todos estos problemas se dispararon estando con muchas personas trabajando y estudiando desde su residencia y pasando una mayor parte de su día interactuando mucho más, con los dispositivos eléctricos del hogar.

Unido a esto se cuenta con conexiones de internet en muchos de los hogares y el semillero SELECT quería aprovechar todos estos factores para construir una plataforma que a la par que solucionara la cuestión inicial planteada anteriormente, pudiera escalar a muchas otras aplicaciones ya no solo de activación y desactivación de cargas, sino incluso de monitoreo de alarmas en el hogar, alimentación de mascotas y demás posibilidades aún por explorar.

Los trabajos más representativos consultados incialmente fueron “Sistemas de monitoreo energético y control domótico basado en tecnología ‘internet de las cosas” (Escobar and Villazón, 2018), “Dispositivo de monitoreo centralizado y escalable para casas inteligentes” de (Rojas, 2020) y “Diseño e implementación de una aplicación domótica para el monitoreo y el control de cargas eléctricas residenciales” (Ortiz and Campoverde, 2016).

2. Desarrollo del prototipo

El diseño del prototipo se realizó en base al diagrama de bloques planteado en la Figura 1.

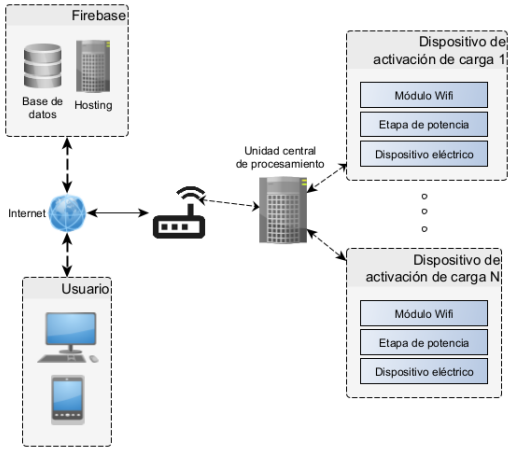


Figura 1. Diagrama de bloques del sistema domótico.

Los dispositivos de activación de carga permitirán la activación/desactivación de luces y/o equipos eléctricos conectados a la red eléctrica del hogar. Cada uno de estos dispositivos de carga cuentan con una etapa de potencia alimentada desde los 110Vac a través de un conversor AC/DC que

entrega el voltaje necesario para alimentar un módulo wifi ESP-01 que activa y desactiva la bobina de un relé a través del pin GPIO2. Es importante resaltar que se necesitan tanto 5V para la bobina, como 3.3V para la alimentación de los módulos wifi.

La unidad central de procesamiento es la parte más importante dentro del proyecto, es la encargada de recibir las solicitudes del usuario (que las realiza desde la aplicación, pero modifican una base de datos en la nube), desde internet a través de una conexión Wi-Fi dentro del hogar. Se encarga también de enviar los datos a los dispositivos de activación de carga para controlar los dispositivos eléctricos dentro de la vivienda que se encuentren conectados al sistema. Finalmente, también se encarga de mostrar al usuario permanentemente, los cambios que se hicieron. Está basada en una Raspberry Pi Zero W, que sirve de puente entre los dispositivos de carga y los usuarios. En la Raspberry se ejecuta un servidor MQTT, el cual recibe mensajes desde un programa en Python que le indica que dispositivos de carga deben modificar su estado. Los dispositivos de carga actúan como clientes MQTT a la espera de un cambio en el estado lógico de HIGH a LOW o viceversa, lo cual significa una acción deseada de parte del usuario que conlleva al encendido o apagado del dispositivo eléctrico específico.

3. Plataforma Firebase

En la plataforma Firebase se configuran dos servicios principales que se usan en el sistema, el primero es el servicio de Realtime Database, o base de datos en tiempo real que permite almacenar información en el servicio y que los distintos dispositivos que se encuentran conectados reciban una actualización apenas los datos cambian. Se escogió esta plataforma porque su capa gratuita provee la mayor capacidad de almacenamiento con respecto a las otras opciones gratuitas, y la escritura y lectura de los datos es más rápida debido a que su gestor de bases de datos es de tipo no relacional. El otro servicio de Firebase que se utiliza es el Hosting o alojamiento de la aplicación Web. El Hosting que provee Firebase permite servir una aplicación Web de forma gratuita y que los usuarios que cuenten con una conexión a internet puedan acceder a través de una conexión con seguridad SSL (Secure Sockets Layer) (Google Developers, 2022).

Cuando un usuario realiza un cambio en la aplicación, dicho cambio se ve reflejado en la Realtime Database y, tanto los demás usuarios que se encuentren en la aplicación como la unidad central de procesamiento, reciben la correspondiente actualización de los datos. En la Raspberry de la unidad central, el programa en Python se conecta como administrador de la base de datos (usando la Firebase admin SDK) y puede hacer consultas como cuando se usa mysql, modificar los datos o establecer listener para escuchar cambios en tiempo real. Con esta actualización la unidad central envía las correspondientes instrucciones a los distintos módulos con el objetivo de que se enciendan o se apaguen, según haya indicado el usuario. Realtime Database utiliza una estructura de pares clave-valor para definir los elementos que la componen, la clave debe ser una cadena única y el valor puede ser de diferentes tipos, cadenas, booleanos, números, otros objetos con pares clave-valor, etc. Dentro del código esta estructura se define dentro del archivo databse.rules.json.

En la Figura 2 se define cual es la estructura que van a seguir los datos dentro de la base de datos. La base de datos tiene una clave principal llamada systems en donde se definen los distintos system,

numerados del 1 al n según el número de unidades centrales de procesamiento conectados a sus distintos dispositivos de control de cargas. Esto permite realmente que el sistema sea escalable.

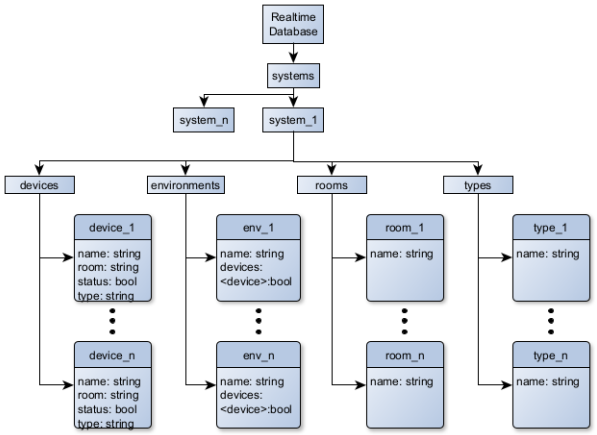


Figura 2. Estructura de datos de la Realtime Database

Cada uno de los sistemas se compone de 4 elementos: devices, environments, rooms y types.

- El grupo devices contiene todos los dispositivos conectados a dicho sistema, la clave es el identificador del dispositivo y cada dispositivo cuenta con un nombre (name), un estatus (status) que se refiere a si está encendido o no, la habitación a la que pertenece (room) y el tipo del dispositivo (type). De esta manera, room y type indican claves correspondientes a dichos elementos dentro de sus respectivos grupos.
- En el grupo environments, cada uno de los ambientes tiene su clave, su nombre y una lista de los dispositivos con el estado al que debe cambiar cada uno cuando se active el ambiente.
- En el grupo rooms y en el grupo types cada una de las habitaciones y de los tipos, respectivamente, cuentan con su identificador y su nombre.

Con el objetivo de conectar la información que se encuentra en la Realtime Database de Firebase, se crea un programa en Python utilizando la librería de administrador de Firebase para este lenguaje de programación. Este programa se encarga de recibir la información de la base de datos inicialmente y luego de cada actualización, con la información recibida se envían las señales correspondientes al servidor MQTT (ver Figura 3).

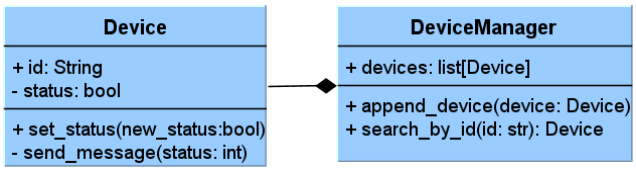


Figura 3. Estructura del programa en Python dentro de la Unidad Central de Procesamiento.

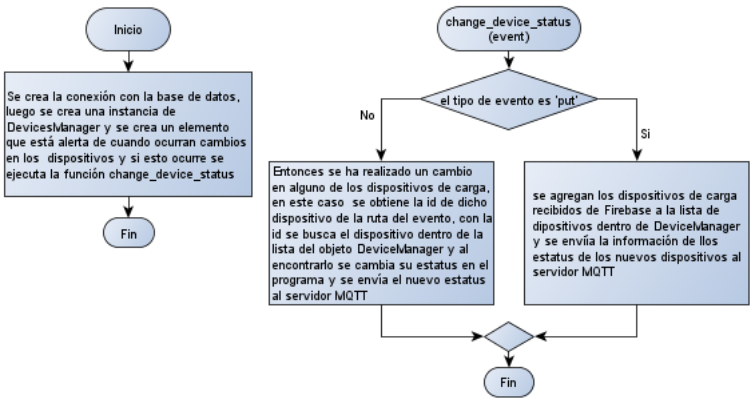


Figura 4. Diagrama de flujo de la ejecución del programa de Python.

En el programa se definen dos clases

- Device: representa un dispositivo de carga dentro del sistema, a este dispositivo se le asigna una id y un estatus. Esta clase cuenta con dos métodos los cuales son set_status, que permite cambiar el estatus del dispositivo, y send_message, que permite enviar el estatus del dispositivo al servidor MQTT para que el dispositivo de carga sea capaz de actualizar su estado al indicado.
- DeviceManager: se encarga de almacenar en una lista los dispositivos de carga actuales del sistema, gestionar cuando se agreguen nuevos dispositivos al sistema y cuenta también con un método que permite buscar un dispositivo de carga específico dentro de la lista usando la id de dicho dispositivo.

Luego crea el punto de entrada del programa y la función que se encarga de realizar el cambio del estado de un dispositivo dentro de una función cuando se reciba la información correspondiente de la base de datos de Firebase, esto se puede observar en la Figura 4. Al iniciar el programa se realiza la conexión con la base de datos de Firebase y luego se utiliza esta conexión para asignar un elemento que ejecute la función change_devices_status cada vez que ocurra un evento con algún cambio en uno de los dispositivos de carga.

4. Interfaz de usuario remota

Para la interfaz de usuario remota se deseaban utilizar tecnologías que permitieran al usuario interactuar con el sistema de forma remota, así se eligió usar tecnologías Web para la interfaz, con el objetivo de agilizar su desarrollo y permitir que los usuarios pudieran acceder desde distintos dispositivos, como lo pueden ser un teléfono inteligente, una computadora, una Tablet, etc. JavaScript es el lenguaje de programación que se utiliza para el desarrollo de las interfaces de usuario en las aplicaciones Web. Es posible construir un sinfín de aplicaciones usando únicamente JavaScript, HTML y CSS, pero existen herramientas construidas con JavaScript que permiten que el desarrollo de estas aplicaciones sea más ágil y sencillo. Algunas de las herramientas consultadas fueron: Angular, React, Svelte y Vue (Castro, 2021). Finalmente se decide usar React debido a que su flexibilidad permite construir una interfaz con la librería y usar otras herramientas más acordes

que permitan la aplicación de otras funcionalidades como la integración con Firebase. Además, la aplicación contará con constante interacción con el usuario por lo que otras herramientas como Svelte no representan una buena opción.

a. Estructura de la Interfaz de la Aplicación

Para realizar el diseño de la interfaz de usuario remota se decidió usar la herramienta de generación de prototipos Figma. La aplicación se divide en 3 pestañas principales, dentro de las cuales se encuentra el funcionamiento principal de la interfaz, como el encendido y apagado de los dispositivos, la separación de los dispositivos por habitación y finalmente el uso de ambientes para controlar un grupo de dispositivos a la vez. Las pestañas principales de la aplicación se diseñan con el objetivo de permitir al usuario observar el estado actual de los distintos dispositivos de carga y poder encenderlos o apagarlos de manera individual o por habitaciones, o incluso habilitar configuraciones personalizadas por el usuario, dentro del sistema (ver Figura 5).

Cada una de las pestañas principales se concentra en cumplir un objetivo:

- Rooms: muestra al usuario todos los dispositivos de carga conectados al sistema agrupados según la habitación y permite que encienda o apague dichos dispositivos según desee, únicamente presionando el botón asociado con dicho dispositivo.
- Devices: muestra todos los dispositivos en un único grupo, permitiendo al usuario observar y manipular el estado de los diferentes dispositivos de manera conjunta.
- Environments: en esta pestaña el usuario es capaz de activar algún ambiente que haya configurado con anterioridad.

Las tres pestañas cuentan con elementos similares: el encabezado, con el nombre de la pestaña y un botón de opciones, una barra de navegación que permite al usuario viajar entre estas tres pestañas a través de tres botones. Adicional a las pestañas principales, cada pestaña cuenta con páginas secundarias que permiten añadir funcionalidad a la aplicación como agregar o eliminar dispositivos, modificar ambientes personalizados o poder obtener más información de estos elementos.

Figura 5. Diseño de las pestañas principales de la interfaz remota.

b. Definición de la Estructura de Carpetas de la Aplicación

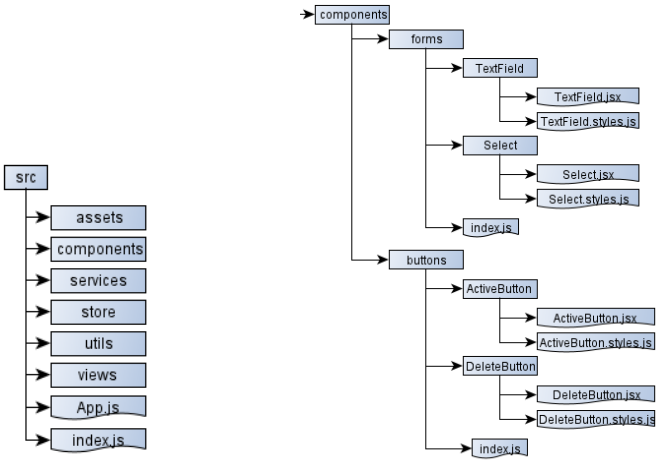
La estructura de carpetas se refiere a como se organizan los archivos de la aplicación de forma que facilite la lectura y la comprensión del código por parte de los desarrolladores, así como evitar muchos errores que provienen de no comprender correctamente el orden de la aplicación. React.js es una biblioteca de JavaScript únicamente centrada en controlar las vistas que permite a los desarrolladores definir su propia arquitectura de la aplicación, algo fundamental para aplicaciones que se espera que escalen, la arquitectura que tenga la aplicación es fundamental para el desarrollo de los contenidos Web.

Para definir la arquitectura de la interfaz remota se utilizó una arquitectura basada en (Rascia, 2021) la cual permite organizar de una forma ordenada los distintos recursos de la aplicación Web, mejorando su legibilidad, su escalabilidad y la facilidad para encontrar errores en el código o mejorar distintas características. Dentro de dicha arquitectura, todo el código fuente de la interfaz remota se encuentra dentro de la carpeta source o src, (ver Figura 6a). En de esta carpeta los archivos se separan según su función dentro de la aplicación. Por ejemplo, en la carpeta assets se almacenan todos los recursos multimedia que necesita la aplicación como lo pueden ser imágenes, iconos, sonidos, etc. Dentro de views se almacenan las distintas páginas que tendrá la aplicación y dentro de components se guardarán los componentes de React que puedan ser reutilizados dentro de la aplicación. Se decide que cada componente de React se almacene en su propia carpeta en donde se encuentran todos los archivos que dicho componente necesita como la definición del componente y los estilos visuales de este (ver Figura 6b).

c. Definición de la estructura del estado global con Redux.

Redux es una librería que provee un estado global estructurado y ordenado para una aplicación, lo que permite a todos los elementos dentro de la aplicación web acceder a la información contenida en el estado de una forma sencilla y estandarizada que se comporte de forma consistente, además de ser fácil de probar (Abramov et al, 2022).

Las reglas establecidas en los estados son conocidas con el nombre de reducer. En este caso se separan los reducers según el tipo de elemento al que pertenecen, es decir, se crea un reducer para las habitaciones, otro para los dispositivos, etc. También se encargan de agregar nuevos elementos en el estado, editar o eliminar los existentes. En la Figura 7 se presenta el reducer que altera la información de los dispositivos de carga dentro del estado global. Fue nombrado como devices y principalmente se encarga de añadir los dispositivos de carga al estado, eliminarlos y editar su información.



a) b)

Figura 6. a) Estructura de carpetas del código fuente de la interfaz remota b) Ejemplo de la estructura dentro de la carpeta componentes

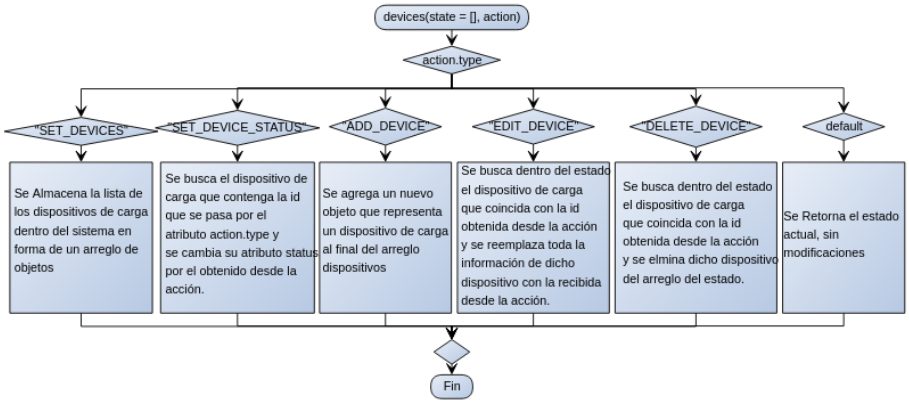


Figura 7. Reducer devices

Los demás reducers tienen un comportamiento similar, relacionado con las posibilidades que tiene el usuario dentro de la aplicación. Se espera que el usuario sea capaz de visualizar, añadir, editar y eliminar la información de las habitaciones, los dispositivos de carga y los ambientes.

d. Definición de los componentes de React en la aplicación

Para configurar el acceso a los distintos componentes de la aplicación de manera remota, se utiliza la librería React, la cual permite configurar la navegación entre los distintos componentes de una página o aplicación Web, ofreciendo al usuario tenga un acceso más cómodo a este contenido.

A partir del diseño de las páginas principales se identifican los posibles componentes de React que compondrán la interfaz como se observa en la Figura 8. Aquí se identifican 11 componentes, donde tres pertenecen a la vista de dichas páginas (RoomsMain, DevicesMain y Environmnets-Main).

Dentro de estos componentes principales se debe cargar la información para que se visualicen los datos necesarios dentro de la página. Al iniciar la aplicación todos los datos se cargan de la base

de datos en Firebase y se almacenan dentro del estado global. Luego cada uno de estos componentes toma la información del estado global y lo dibuja en pantalla. Por ejemplo, el componente DevicesCard toma la información de una habitación y dibuja el nombre de la habitación y un componente ActivateButton para que el usuario sea capaz de encender o apagar todos los dispositivos de carga dentro de dicha habitación al mismo tiempo; luego utiliza el componente Group para dibujar los dispositivos de carga dentro de dicha habitación separándolos por el tipo de dispositivo.

Figura 8. Identificación de componentes de las páginas principales.

Dentro del Group se dibuja nuevamente un componente ActivateButton y luego se dibuja un componente Item por cada uno de los dispositivos dentro de dicho grupo, es decir dispositivos con el mismo tipo dentro de la misma habitación. Dentro de cada componente Item se dibuja un icono, el nombre del dispositivo de carga dentro del sistema y un componente ActivateButton, este botón en este caso se encarga de, cuando se presione, enviar la información del cambio a la base de datos de Firebase, el componente Item crea una comunicación en tiempo real con la base datos y cuando el estatus cambia esta información se actualiza en el componente y en el estado global.

5. Conclusiones

Durante la investigación se demostró que las redes Wi-Fi del hogar se pueden aprovechar para realizar telemetría y domótica, utilizando protocolos de envío de datos como MQTT, hacia y desde un dispositivo central sin perder información en el envío de la misma. Luego para poder realizar el control desde una ubicación remota, es necesario que el hogar cuente con una conexión a internet, para enviar y recibir información de Firebase.

Es necesario definir correctamente la estructura en la base de datos de Firebase para que la información tenga consistencia y evitar errores a la hora de leer la información existente o de almacenar nueva información, ya que se busca controlar múltiples dispositivos de activación de cargas usando una unidad central y una misma aplicación web. Para aplicaciones futuras y basados en la construcción realizada, se conectarán múltiples unidades centrales ubicadas en distintos hogares.

En cuanto a la aplicación, para mantener de forma adecuada el estado global de la aplicación Web usando Redux, es necesario definir las reglas que seguirá cada reducer almacenando la información de manera adecuada y manteniendo la integridad de los demás datos en el estado. Para poder visualizar los cambios realizados en la base de datos de Firebase en React, es necesario utilizar herramientas como useEffect para que, al recibir una actualización de los estados en la Realtime Database, se modifique la interfaz.

6. Referencias

Artículos de revistas

- Escobar, E., and Villazón, A. (2018). Sistema de monitoreo energético y control domótico basado en tecnología "Internet de las cosas". Investigación & Desarrollo, Vol. 18, No.1, pp. 103-116.

Libros

- Ministerio de Minas y Energía. (2013). Reglamento Técnico de Instalaciones Eléctricas (RETIE). Bogotá, Colombia. Ministerio de Minas y Energía.
- Ortiz, O. O., & Campoverde, P. D. (2016). Diseño e implementación de una aplicación domótica para el monitoreo y el control de cargas eléctricas residenciales. Cuenca, Ecuador. Universidad Politécnica Salesiana.
- Rojas, S. (2020). Dispositivo de monitoreo centralizado y escalable para casas inteligente. Cali, Colombia. Universidad San Buenaventura Cali.

Fuentes electrónicas

- Abramov, D. and the Redux documentation authors. (2022, febrero). Redux Fundamentals, Part 1: Redux Overview. Consultado el 3 de marzo de 2022 en https://redux.js.org/tutorials/fundamentals/part-1-overview
- Castro, J. D. (2021, septiembre). React vs. Angular vs. Vue vs. Svelte | ¿Cómo elegir tu próxima herramienta frontend? ¿Cuál es mejor? Platzi. Consultado el 10 de diciembre de 2021 en https://platzi.com/blog/react-angular-vue-svelte/
- Google Developers. (2022, enero). Firebase Hosting. Consultado el 5 de febrero de 2022 en https://firebase.google.com/docs/hosting
- Rascia, T. (2021, junio). React Architecture: How to Structure and Organize a React Application. Consultado el 10 de enero de 2022 en https://www.taniarascia.com/react-architecture-directory-structure/

Sobre los autores

- **Juan Sebastián Hernández González**: Estudiante de Ingeniería Electrónica de Institución Universitaria Antonio José Camacho. jshernandezgonzalez@estudiante.uniajc.edu.co
- **Rafael Martínez Bermúdez**: Estudiante de Ingeniería Electrónica de Institución Universitaria Antonio José Camacho. rafaelmartinez@estudiante.uniajc.edu.co

- **Santiago Padilla Trujillo**: Estudiante de Ingeniería Electrónica de Institución Universitaria Antonio José Camacho. spadilla@estudiante.uniajc.edu.co
- **Erika Sarria Navarro**: Ingeniería Electrónica, Magister en Ingeniería con Énfasis en Ingeniería Electrónica de Universidad del Valle. Docente Tiempo Completo de Institución Universitaria Antonio José Camacho. esarrian@admon.uniajc.edu.co

---