



**IMPLEMENTACIÓN DE UN MANEJADOR DE RECURSOS  
CON INTELIGENCIA AMBIENTAL PARA UNA VIVIENDA  
DEL SOLAR DECATHLON LATIN AMERICA 2019**

JUAN DAVID RAMIREZ VILLEGAS

UNIVERSIDAD DEL VALLE  
FACULTAD DE INGENIERÍA  
ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA  
PROGRAMA ACADÉMICO DE INGENIERÍA ELECTRÓNICA  
6 de Agosto de 2019



**IMPLEMENTACIÓN DE UN MANEJADOR DE RECURSOS  
CON INTELIGENCIA AMBIENTAL PARA UNA VIVIENDA  
DEL SOLAR DECATHLON LATIN AMERICA 2019**

JUAN DAVID RAMIREZ VILLEGAS

FABIO GERMÁN GUERRERO, M.Sc  
EDINSON FRANCO MEJÍA, Dr. Ing

UNIVERSIDAD DEL VALLE  
FACULTAD DE INGENIERÍA  
ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA  
PROGRAMA ACADÉMICO DE INGENIERÍA ELECTRÓNICA  
6 de Agosto de 2019

# Índice

<b>1</b>	<b>introduccion</b>	<b>4</b>
1.1	Resumen . . . . .	4
1.2	Contexto introductorio . . . . .	5
1.3	Planteamiento del problema . . . . .	5
1.4	Justificación . . . . .	7
1.5	Objetivos . . . . .	8
1.5.1	Objetivo general . . . . .	8
1.5.2	Objetivos especificos . . . . .	8
1.6	Lineamientos de desarrollo . . . . .	9
1.7	Diseño Generalizado . . . . .	10
<b>2</b>	<b>Marco Teorico</b>	<b>13</b>
2.1	Arquitectura. . . . .	13
2.2	Programacion orientada a objetos . . . . .	13
2.3	Servidor . . . . .	14
2.4	TCP y TLS . . . . .	14
2.5	JWT . . . . .	14
2.6	Mqtt . . . . .	14
2.7	Http . . . . .	14
2.8	Serverless . . . . .	14
2.9	Backend . . . . .	14
2.10	Aplicación web. . . . .	15
2.11	Aplicación de escritorio. . . . .	15
2.12	Frontend . . . . .	15
2.13	Dispositivo . . . . .	15
<b>3</b>	<b>Api de animación</b>	<b>17</b>
3.1	Investigacion y conceptualización . . . . .	17
3.2	Api de java . . . . .	18
3.3	Api de Python y Qt . . . . .	23
<b>4</b>	<b>House Manager</b>	<b>25</b>
4.1	Requerimientos Funcionales . . . . .	26
4.2	Funciones adicionales . . . . .	27
4.3	Algoritmos Utilizados . . . . .	29
4.3.1	Scheduler Handler . . . . .	30
4.3.2	Remote Mode Handler . . . . .	30
4.3.3	Mail Notification Handler . . . . .	30
4.3.4	Data Report Handler . . . . .	30
4.3.5	Indicator Handler . . . . .	30

4.4	Serverless Backend . . . . .	31
4.5	Requerimientos Funcionales . . . . .	31
4.6	Cloud functions . . . . .	31
4.7	Function 1 . . . . .	32
4.8	Function 2 . . . . .	32
4.9	Function 3 . . . . .	32
<b>5</b>	<b>User app</b>	<b>33</b>
5.1	Requerimientos Funcionales . . . . .	33
5.2	Funciones adicionales . . . . .	33
5.3	Algoritmos Utilizados . . . . .	41
5.3.1	Scheduler Handler . . . . .	41
5.3.2	Remote Mode Handler . . . . .	41

## Índice de figuras

Figura 1	Diagrama generalizado de las estructuras de software . . .	12
Figura 2	Conceptualización inicial de la api en sitio . . . . .	18
Figura 3	Estadísticas de los lenguajes de programacion con más demanda . . . . .	19
Figura 4	El diseño de estados del boton interactivo . . . . .	20
Figura 5	El diseño de los estados del boton interactivo desplegable	20
Figura 6	Movimiento según la ubicación de la función horizontal .	21
Figura 7	Primer Aproximación del diseño de la interfáz en sitio .	21
Figura 8	Ajuste de diseño de la interfáz en sitio . . . . .	22
Figura 9	Version final del concepto de diseño . . . . .	22
Figura 10	Botón de tamaño flexible genérico de la api de qt . . . . .	23
Figura 11	Boton de tamaño fijo usando imagenes y slider . . . . .	24
Figura 12	Visualizacion de la escalabilidad del sistema con una app no desarrollada por completo . . . . .	26
Figura 13	Escena del home del House Manager con barras de pro- greso para la indicacion dinamica . . . . .	26
Figura 14	Escena de control del House Managern con sus calendarios	28
Figura 15	Escena de medición del House Manager, con su respectivo panel para guardar los datos . . . . .	29
Figura 16	Escena de configuracion del dispositivo House Manager .	29
Figura 17	Escena para el inicio de sesion del House Manager . . . .	30
Figura 18	El menu desplegable con las aplicaciones actualmente desplegadas . . . . .	34
Figura 19	Escena de bienvenida 1 . . . . .	35
Figura 20	Escena de logg in . . . . .	36
Figura 21	Escena para a visualizacion del consumo historico . . . .	37
Figura 22	Escena para la medicion en tiempo real de las variables .	38
Figura 23	Escena para el control de los circuitos de la vivienda . . .	39
Figura 24	Escena de configuraciones, actualmente no disponible . .	40

## Índice de cuadros

Tabla 1	Requerimientos Funcionales del sistema . . . . .	10
---------	--	----

# 1. introduccion

## 1.1. Resumen

En este documento se explicara el proceso realizado para implementar una arquitectura software para la comunicacion de diferentes dispositivos bajo el paradigma de internet de las cosas. El sistema consta principalmente de 3 productos: el producto del backend, una aplicacion movil y una aplicación de escritorio. La aplicacin movil se encargo de monitorear y controlar todos los dispositivos conectados al sistema'. El aplicativo en sitio se encargo de manejar los sensores y actuadores de manera local en caso de perdida de conexion, de manera remota en presencia de una red wifi y otras funciones adicionales. Por ultimo, el servicio del backend se encargo de comunicar ambos productos de frontend y almacenar la informacion de los usuarios.

El producto del backend se implemento en el servicio de Google Cloud Services y utilizó los siguientes elementos para su funcionamiento: Un "Storage version 3 system"conectado a la red privada de Google para el almacenamiento de archivos, Una base de datos relacional MySQL para el almacenamiento de datos de monitoreo y control de los dispositivos, Una lista de cloud functions.<sup>a</sup>activadas por solicitudes http y por ellas mismas que se comunican directamente con la base de datos y finalmente, un broker nativo de Google Cloud llamado Cloud IOT"que recibe las conexiones de mqtt de los dispositivos del sistema.

La aplicación movil desarrollada constó de principalmente de 3 escenas: la escena de control, utilizada para el manejo de los actuadores disponibles desde el aplicativo en sitio, la escena de medicion, encargada de presentarle al usuario las mediciones adquiridas por el aplicativo en sitio en tiempo real y la escena principal, donde se puede observar un indicador del porcentaje de las variables mas relevantes respecto a un punto de referencia. Por ejemplo, el porcentaje de consumo de agua con respecto al promedio de consumo de agua en en un mes de un grupo familiar.

Por otra parte, el aplicativo en sitio se desarrollo usando python y se diseño para manejar un flujo de datos mas grandes y la posibilidad de funcionar desconectado de la red. La aplicacion le permitio al usuario programar rutinas de control de los circuitos a partir de horarios, visualizar los datos medidos, cambiar parametros de adquisicion como el tamaño del bufer y la frecuencia de muestreo, entre otros.

Finalmente la arquitectura se diseño bajo la posibilidad de soportar hasta 4000 dispositivos de alto trafico de datos entre los cuales se encuentran computado-

res, celulares y cualquier dispositivo capaz de ejecutar rutinas en python o nodejs.

## 1.2. Contexto introductorio

No es para nadie una sorpresa que durante este siglo, las nuevas generaciones que nacen rodeadas de la tecnología, están cambiando las costumbres y el paradigma actual de lo que significa una vivienda, con una sociedad cada vez más relacionada con la tecnología, nuevos desafíos de diseño e implementación se plantean, en el sentido de cumplir las expectativas de estas nuevas generaciones.

Básicamente el más importante beneficio utilizar la tecnología en viviendas comerciales es el proveer de servicios y facilidades a personas discapacitadas y ancianos [3], por otra parte, tenemos beneficios de monitorización y la capacidad de añadir herramientas que permitan la integración con redes eléctricas inteligentes. La popularidad de sistemas inteligentes ha venido incrementando debido al confort adicional, y herramientas de seguridad que pueden recibir los usuarios.

Por lo tanto, hay ciertos factores que deben ser tenidos en cuenta a la hora de diseñar un sistema de control y monitoreo de la vivienda, En este documento se explicara el proceso de diseño e implementacion de un software para el manejo inteligente de una vivienda del Solar Decathlon Latinoamerica utilizando diferentes tecnologias de frontend y backend

La arquitectura desarrollada se puede describir en 3 productos:

1. Un servicio de backend serverles basado en, un broker de mqtt, un servicio de almacenamiento de datos tipo S3, una base de datos relacional tipo mysql e interfaces http basadas en cloud functions.
2. Un programa encargado de administrar y controlar todas las rutinas de comunicacion, reporte y almacenamiento en el sitio de control con su respectiva interfaz de usuario.
3. Una aplicacion celular capaz de monitorear y controlar los datos generados localmente y los actuadores de la vivienda del Solar Decathlon.

## 1.3. Planteamiento del problema

Hoy en día las tecnologías orientadas internet y la mayoría de los objetos utilizados en la cotidianidad, en general, están cada vez más relacionadas entre

sí; el pasado año se vio un incremento de 2.79 billones de nuevos dispositivos conectados a la red de redes (entre los cuales no solo se encuentran celulares computadores y tablets). Nuevos dispositivos como aires acondicionados, televisores y electrodomésticos en general son lanzados al mercado con la capacidad de comunicarse con el usuario a través de aplicaciones web o dispositivos móviles día tras día. Lo anterior se debe al aumento en los desarrollos de hardware y software para sistemas con la capacidad de acceso a la red; es decir, con la capacidad de acceder a comunicaciones inalámbricas, servidores locales, servicios en la nube, entre otras propuestas.

Teniendo en cuenta lo anterior, muchos sistemas basados en diseños embebidos y/o tarjetas de desarrollo CPU se han propuesto como forma de integración y aplicación de sistemas de domótica en el hogar. La mayoría de estos proyectos o aplicativos son pensados para automatizar tareas específicas o agregarle una componente de comunicación remota al hogar. Si bien estos son puntos atractivos para un consumidor, el concepto de vivienda inteligente se puede expandir por más terrenos, tan diversos como el que se presentará en este documento.

Un concepto tan abierto y ventajoso como el de la vivienda inteligente o el internet de las cosas puede ser utilizado en el marco del concurso Solar Decatlón Latín América, el cual plantea el desarrollo de un proyecto de vivienda residencial amigable con el medio ambiente, donde el diseño arquitectónico ganador se decide en base a 10 indicadores cada uno con 100 puntos calificados de manera cualitativa y cuantitativa por los jueces.

Para la porción de la puntuación que es calificada de manera cuantitativa los jueces utilizan medidores que, dependiendo del indicador, puede corresponder para monitorear o medir el consumo eléctrico, temperatura, humedad, picos de consumo (3 kilo watts) donde se establece un valor de mínimo y máximo para un rango de puntos desde cero hasta la máxima calificación.

Considerando lo anterior, se plantea realizar la mejor aproximación tecnológica para cumplir de manera más acercada los indicadores de: eficiencia energética, innovación, balance eléctrico energético y condiciones de confort, puesto que son aquellos que pueden ser solucionados gestionados y monitoreados por tecnologías actuales. Basándose en lo expuesto anteriormente se plantea la siguiente pregunta al problema de ingeniería:

¿Cómo diseñar e implementar un sistema software que permita la gestión de cargas eléctricas y el monitoreo de variables físicas de manera local y remota para el concurso del Solar Decatlón latín América?.



## 1.4. Justificación

Respecto a la vivienda inteligente o la domótica en el hogar, se puede decir que la literatura es amplia y que existen diferentes propuestas comerciales de desarrollo como lo son: Calaos, Domoticz, Home Assistant, OpenHAB, OpenMotics, donde cada marca propone su marco de trabajo, y diferentes plataformas de desarrollo tanto software como hardware. Una desventaja de estas propuestas es su interfaz hardware, en la mayoría de los casos el hardware adicional corresponde a dispositivos desarrollados y mantenidos por ellos mismos, lo que significa que limita la escalabilidad de las

Las anteriores tecnologías son propuestas donde un centro de mando controla dispositivos externos, pero, por otra parte, si pensamos en cada módulo hardware con la capacidad de conectarse a la red de redes de manera independiente (Iot) como lo hace la propuesta realizada en el proyecto del “Smart switch” [?]; se tiene como ventaja la independencia sobre las características físicas del hogar o de la capacitación técnica del instalador del sistema, pero como desventaja, al momento de escalar el problema a un mayor número de circuitos se pueden presentar problemas de saturación de la red inalámbrica (wifi) y sería más costoso debido al hardware adicional para cada nuevo circuito que se desee controlar.

Teniendo en cuenta las comparaciones anteriores, se podría pensar que un sistema que englobe varios sub sistemas podría ser el adecuado desde el punto de vista de la escalabilidad, pero esa no es la única razón; Hoy en día los métodos de generación alternativa y energía renovable se encuentran en crecimiento, por lo que nace la necesidad de interconectar redes eléctricas inteligentes, es decir, una red con un componente de generación y consumo tanto como AC y DC. Este tipo de topología eléctrica se le conoce como nanogrid, y se ampliará la teoría respecto a este concepto en futuras secciones del documento. Esta topología eléctrica añade como variable la generación eléctrica junto con una nueva forma física de energía (corriente directa).

Inclusive, se puede observar que artículos académicos se han dedicado a analizar este concepto en el contexto Colombiano, por ejemplo un estudio sobre microgrids mencionó la importancia y necesidad de implementar viviendas inteligentes como se ve a continuación: “desde el contexto de seguridad eléctrica, equidad social y mitigación del impacto ambiental en Colombia, el sistema energético debe afrontar los nuevos retos requeridos para satisfacer la demanda. Desde un punto de vista técnico, es necesario dotar la red tradicional con las características de una red inteligente ”.

Pero ¿acaso una red inteligente implica una casa inteligente? La respuesta es; parcialmente sí. Considerando que hoy en día toda vivienda realiza un monitoreo o medición de al menos 2 variables críticas; el consumo eléctrico, y el consumo de agua, ambos son de interés para la compañía prestadora de servicios públicos en la vivienda.

Resaltando la importancia de las anteriores variables físicas y teniendo en cuenta el modelo de nanogrid al cual podría llegar a ser una vivienda; la gestión de información adicional a la necesaria para cumplir la domótica resalta a la vista. En adición, estos componentes de generación y consumo eléctrico son la base para hablar sobre el impacto ambiental de las costumbres internas de los habitantes de un hogar, que resulta de vital interés para la optimización de los 4 ítems de calificación en el solar decathlon.

Finalizando, una propuesta integral de domótica y gestión inteligente de información como la que se presenta en este documento facilita la verificación y el cumplimiento de los ítems anteriormente mencionados, contemplando que el sistema quedara diseñado de tal manera que sea fácil la escalabilidad para aspectos como; seguridad, aseguramiento contra accidentes (incendio), viviendas de diferente tamaño, interfaz de voz y video.

## **1.5. Objetivos**

### **1.5.1. Objetivo general**


Implementar un sistema software para una vivienda del Solar Decathlon capaz de gestionar el control y la medición de variables físicas de manera local y remota.

### **1.5.2. Objetivos específicos**

- Diseñar una API que permita implementar interfaces gráficas, haciendo uso de imágenes personalizadas, para lograr una interacción amigable con el usuario.
- Implementar una aplicación software que funcione en el sitio y le permita al usuario gestionar la información de interés y controlar las cargas eléctricas.
- Implementar un aplicativo móvil que le permita al usuario tener la experiencia de interactuar con una vivienda inteligente.

## 1.6. Lineamientos de desarrollo

Debido a la naturaleza del proyecto, parte de los aspectos definidos al momento de establecer el anteproyecto fueron unas necesidades funcionales definidas en el siguiente documento:

	<b>Universidad del Valle</b> <b>–Implementación de un gestor de recursos para el hogar con inteligencia ambiental. —</b>	<b>Rev:</b> <b>003</b>
<b>Título:</b> <b>ESPECIFICACIÓN DE REQUERIMIENTOS FUNCIONALES</b>	<b>Documento :</b> <b>ERF-000</b>	<b>Página :</b> <b>1 de 1</b>

REVISIÓN HISTÓRICA			
Rev.	Descripción del Cambio	Autor	Fecha
001	Construcción del documento	Juan David Ramirez Villegas	18/03/2017
002	Correcciones	Juan David Ramírez Villegas	15/11/2017
003	Revisión	Juan David Ramírez Villegas	18/12/2017

Ref #	Funciones	Categoría
1.0	USER APP (application móvil)	O
1.1	El usuario debe poder acceder a la información medida en tiempo real.	E
1.2	El usuario debe poder acceder a los datos históricos recopilados en el mes y la relación de sus gastos con ellos.	O
1.3	El usuario debe poder cambiar los parámetros de configuración escogidos por defecto para la vivienda del Solar Decathlon.	E
1.4	El usuario debe recibir recomendaciones para mejorar su entorno y su huella hídrica cada cierto tiempo.	E
1.5	La aplicación notificará al usuario cuando la factura de energía o agua esté vencida.	O

1.6	El usuario debe ser capaz de ver un índice del impacto ambiental de su estilo de vida basado en datos cualitativos y cuantitativos	<b>E</b>
1.7	El sistema debe ser capaz de notificar las pérdidas eléctricas, y por consiguiente económicas, de un mal uso de los horarios establecidos por defecto.	<b>O</b>
2.0	HOUSE MANAGER (Aplicación en sitio)	<b>E</b>
2.1	El sistema debe mostrar gráficamente la cantidad de agua y potencia consumida durante el día contrastándolas con un valor máximo recomendado.	<b>E</b>
2.2	Por defecto, el uso de las cargas eléctricas más significativas debe programarse durante los picos de generación en la casa. Estos horarios podrán ser modificados bajo una advertencia de uso no eficiente.	<b>E</b>
2.3	El usuario debe poder acceder a la información medida en tiempo real.	<b>E</b>
2.4	El sistema debe poder comunicarse con una base de datos que represente todas las variables y el estado de los circuitos de la vivienda.	<b>E</b>

Cuadro 1: Requerimientos Funcionales del sistema

Donde la columna de la categoría indica si el requerimiento indicado es uno esencial (E) u opcional (O) para el desarrollo del sistema. Este documento será de gran importancia puesto que definio los lineamientos de desarrollo de todas las aplicaciones.

## 1.7. Diseño Generalizado

El sistema diseñado se diseño con las mejores condiciones de; modularidad, costo, tamaño y escalabilidad. Para cada uno de estas características se ecogieron tecnologías y se desarrollaron funciones claves.

Desde la perspectiva de la modularidad, se desarrollaron aplicaciones de frontend capaces de integrarse con 2 tipos de protocolos de comunicacion: mqtt y https, ademas, para acceder a los servicios de la nube se implemento

el servicio de backend en la plataforma "Google Cloud Plataform" puesto que permite utilizar de la mejor manera todos los servicios de google: su asistente de voz, su interfaz de inteligencia artificial, sus servicios de autenticacion, su red privada de fibra optica que es de las mas rapidas y mayor cobertura, entre otros.

En cuanto al costo, se utilizaron tecnologias de backend totalmente escalables tipo serverless para ajustar los gastos a unicamente los necesarios, teniendo en cuenta que las pruebas a gran escala estuvieron lejos del alcance de este proyecto, aún asi se desarrollo toda la arquitectura pensando en la necesidad de optimizar en la mayor medida los gastos. Teniendo en cuenta que no se hicieron pruebas a gran escala los servicios de backend fueron posibles gracias a las cuotas gratuitas del mayorista utilizado para desplegar el servicio (Google)

En cuanto al requerimiento de escalabilidad, la decision de diseñar serverless juega el papel mas importante: Los servicios serverless le permitiran a la aplicacion permanecer funcional sin hacer ningun cambio inclusive si se tienen 4000 usuarios.

Para comprender mejor el producto, se puede observar en la figura 1 un diagrama generalizado de los componentes de software del sistema, donde principalmente los desarrollos hacen parte de 3 grandes bloques: el bloque de la nube, el bloque del dispositivo embebido (Raspberry), y el bloque del dispositivo movil (Android).

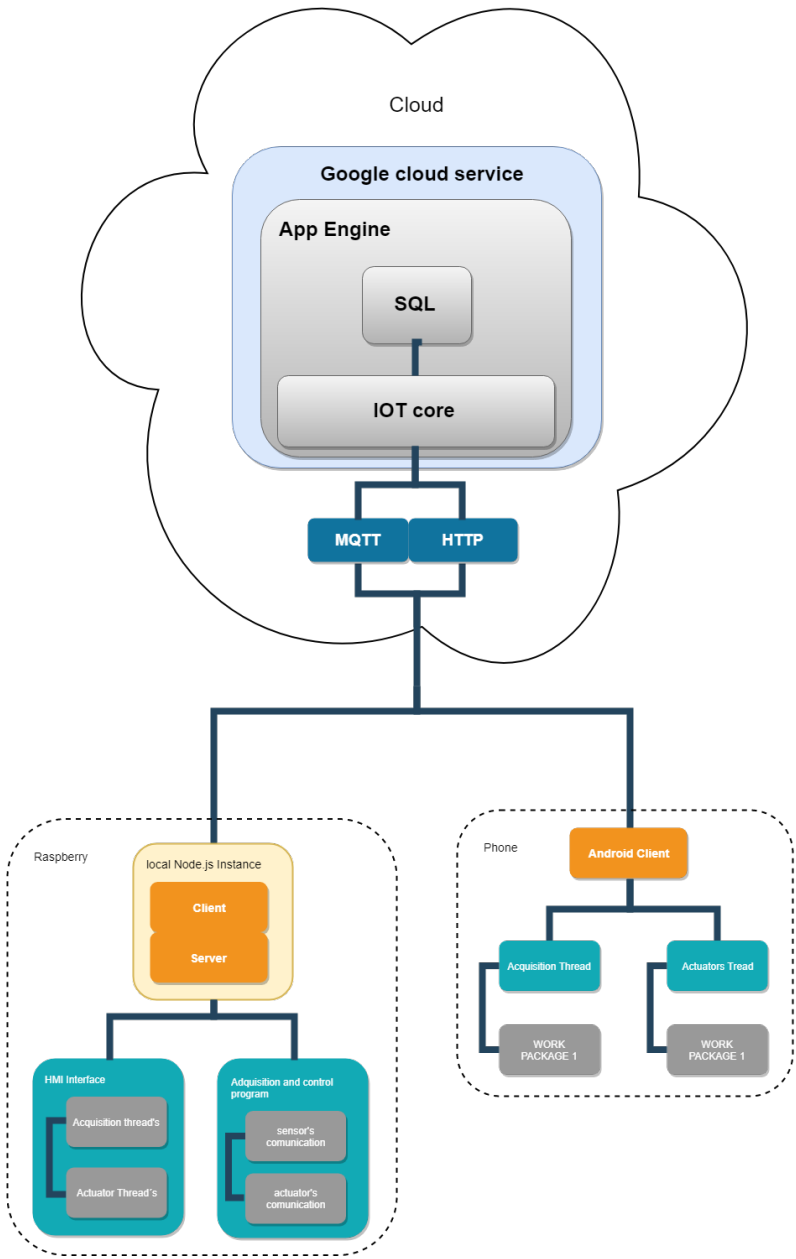


Figura 1: Diagrama generalizado de las estructuras de software

## 2. Marco Teorico

Para la optima comprension del proyecto de ingenieria explicado en este documento es necesario el entendimiento de diferentes conceptos, tecnologias o teorias. Estos conceptos seran explicados de manera breve en el transcurso de este capitulo,

### 2.1. Arquitectura.

Se define como un conjunto de componentes de software que operan en de manera relacionada y utilizan interfaces entre ellos. Los componentes de software utilizan un conjunto de herramientas de programación como APIs, Bibliotecas, Frameworks, para su óptimo diseño. Como aclaración las herramientas mencionadas anteriormente se definen como:

- **API (Application Programming Interface):** Es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
- **Framework:** En el marco de desarrollo de software, un Framework (entorno de trabajo) corresponde a la estructura modular que sirve de base para la organización y desarrollo de software. Normalmente, pueden incluir soporte para lenguajes de programación, librerías, entre otras.
- **Biblioteca:** En programación, una biblioteca o librería es un conjunto de implementaciones funcionales, diseñadas en un lenguaje de programación específico, la cual ofrece una interfaz para la funcionalidad que se invoca.

La arquitectura de software que se implementó en este proyecto consta de 3 bloques importantes; un servicio en la nube, una aplicacion celular y una aplicacion de escritorio. Estos a su vez utilizan herramientas, como las anteriormente descritas, para comunicar o relacionar elementos internamente.

### 2.2. Programacion orientada a objetos

\*\*\*\* describir muy bien la iteraccion shild parent en este paradigma de desarrollo puesto que se menciona mucho en el documento

## 2.3. Servidor

Un módulo hardware y software con amplia capacidad de almacenamiento y procesamiento. Desde el punto de vista del software, los servidores son programas de computador que atienden las peticiones de otros programas llamados clientes. Dentro de estos llamados los principales servicios de un servidor son: compartir datos, información y recursos de hardware. Desde el punto de vista de hardware toda la información entrante de los dispositivos (clientes) es recibida a través de una interfaz de red y reconocida para su posterior procesamiento y almacenamiento. Desde una perspectiva mas simplista un servidor es un computador con especificaciones de hardware fijas como: ram, cpu, memoria intena, etc, enfocado unicamente en ofrecer los servicios anteriormente mencionados.

## 2.4. TCP y TLS

\*\* hablar del metodo de comunicacion sencillo de tcp (asegurar la recepcion) y su version encriptada Tls

## 2.5. JWT

\*\*\* explicar el funcionamiento del estandar de los json web token y los diferentes metodos de encriptacion.

## 2.6. Mqtt

\*\*\*\*\*Definir de manegra generalizada el protocolo mqtt

## 2.7. Http

\*\*\*\*\*Definir de manegra generalizada el protocolo http

## 2.8. Serverless

\*\*hablar del concepto de serverless teniendo en cuenta que ya se explico el conepto tradicional de servidor

## 2.9. Backend

\*\*\*\*misma historia para el backend dado que ya se hablo del servidor en su generalidad



## 2.10. Aplicación web.

Una aplicación web es un programa que se codifica en un lenguaje interpretable por los navegadores web, esto le permite a la aplicación ser independiente del sistema operativo y depender de un interpretador web (navegador). Dentro de los lenguajes utilizados como desarrollo para aplicaciones web se tienen: html, javascript, php, asp, python, ruby, etc.

## 2.11. Aplicación de escritorio.

Una aplicación de escritorio es cualquier software que pueda ser instalado en un computador o sistema de cómputo, y que permita ejecutar ciertas rutinas. En este sentido una aplicación web puede ser una aplicación de escritorio. Los lenguajes de programación de aplicaciones de escritorio son un conjunto más amplio, los que más predominan son aquellos basados en máquinas virtuales o entornos de ejecución puesto que permiten realizar desarrollos independientes del sistema operativo. Dentro de los más conocidos están los lenguajes: Java, Python, C++, Golang, etc.

Dentro de los elementos materiales del sistema se encuentran los dispositivos que van a encargarse de la capa física durante el proceso de comunicación y adquisición de datos, a continuación se definirán los componentes más importantes de estos componentes.

## 2.12. Frontend

\*\* describir el concepto de frontend teniendo en cuenta los dos conceptos anteriormente mencionados

## 2.13. Dispositivo

Un dispositivo es un objeto hardware que puede ser visto como un sistema embebido con capacidades de procesamiento y comunicación a diferentes redes. En el dispositivo hardware reside el software que sirve como el manejador de las magnitudes físicas, voltajes, o sensores que están conectadas a él. Este componente normalmente posee las herramientas para el envío de información a servidores locales o remotos, en adición, dentro de los componentes que integran un dispositivo se tienen los siguientes:

- **Sistema de archivos:** Este componente de software es el encargado de almacenar los datos del sistema ya sea localmente, en un servidor de manera remota. Además de lo anterior, este módulo sirve para el almacena-

miento de la información necesaria para el funcionamiento del sistema operativo.

- **Colector Cliente:** Este componente es el encargado de gestionar toda la información de los sensores que será enviada a manera de flujos de datos utilizando un protocolo de comunicación, esta capa de software añade los campos de metadata (control y gestión) para la efectiva comunicación.
- **Capa de Sensores:** Este software es el encargado de proveer todos los drivers lógicos para la conexión de los elementos físicos conectados al dispositivo.

Debido al tiempo estipulado para el proyecto el dispositivo y los métodos de adquisición no serán de alta importancia en el desarrollo de la arquitectura y se asumirá que todo desarrollo de los dispositivos debe tener solucionada la comunicación con el servidor.

Desde el punto de vista del desarrollo de diseño gráfico es necesario conocer los conceptos que facilitan la implementación de una apropiada interfaz HMI. Dentro del marco del diseño, los conceptos de mayor importancia para este proyecto son:

### 3. Api de animación

#### 3.1. Investigacion y conceptualización

Después de imaginar y delimitar y el problema a tratar, cómo primera instancia, se comenzó con la selección del tipo de entorno de trabajo que sería utilizado para el desarrollo del aplicativo ejecutable de manera local en la vivienda del solar decathlon, durante el transcurso del documento este programa será llamado "House Manager".

Para seleccionar el entorno de trabajo se realizó una averiguación de los posibles entornos de desarrollo que permitieran implementar programación orientada a objetos en dispositivos embebidos, dentro de los entornos de desarrollo más conocidos se observó; python, Java, processing, c++, entre otros. Todos los anteriores con comunidades muy enfocada a su flexibilidad y constante mejoramiento.

Después se compararon a grandes rasgos los módulos de interfaz gráfica que ofrecían los entornos de programación mas conocidos y utilizados en la actualidad, como se puede observar a continuación:

- En el caso de c++ se observó que el entorno de trabajo mas apropiado para desarrollar aplicativos gráficos era .net, la desventaja de este aplicativo era su estrecha relación con el lenguaje de programación de bajo nivel c#, por ende el manejo de memoria del programa incrementaba en gran medida los costos de desarrollo.
- Para Python se observó que los entornos de desarrollo de interfaz gráfica nativos tenían apariencias precarias similar a la nativa de windows 98, en adición, los frameworks mas estéticos tenían un soporte independiente al del lenguaje de programación como tal, por otra parte el lenguaje de programación como tal era multiplataforma y de muy alto nivel, lo que le permitiría al programador desarrollar algoritmos de mayor funcionalidad con menos líneas de código.
- En el caso de Java se observó que el entorno de desarrollo para interfaces gráficas era nativo del lenguaje de programación y permitía una mayor flexibilidad a la hora de desarrollar esquemas complejos con la excepción que se debía codificar detalladamente el comportamiento del objeto, lo que significaba mas tiempo de trabajo para el programador y archivos con mas líneas de código.

Como etapa de conceptualización, se realizó una primera versión del diseño del aplicativo principal junto con sus respectivas escenas. Fig 2. Los elemen-

tos utilizados en esta primera etapa de diseño se consistieron principalmente de: botones, caja de opciones, contenedores, y paneles animados. Para esta etapa de conceptualización no se tenía una paleta de colores escogida pero si se buscaba tener en cuenta tonalidades verdes siendo congruentes con la temática amigable con el medio ambiente que rodea al Solar Decathlon, también, se persiguió el poder seleccionar a partir de la pantalla principal diferentes "sub aplicativos" que podrían ser diseñados de manera independiente.

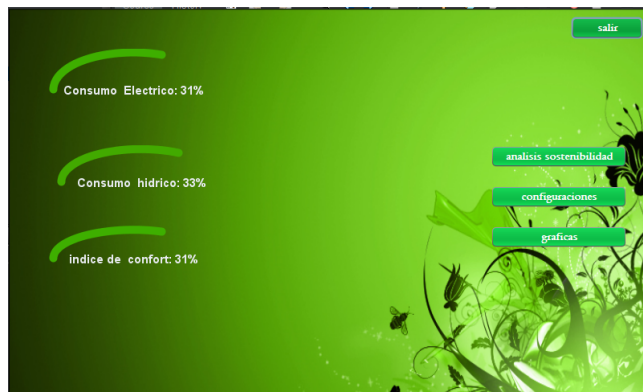


Figura 2: Conceptualización inicial de la api en sitio

Para hacer realidad el diseño de esta primera etapa de conceptualización se utilizó como lenguaje de programación el lenguaje de programación Java puesto que representaba en su mayor parte el lenguaje más utilizado en la Universidad del Valle durante el momento.

En adición, Java es un lenguaje de programación de muy alta demanda, de hecho, el de mayor ofertas de trabajo registradas durante el 2017 en el portal de trabajo internacional indeed.com, seguido de javascript, c# y python, tal como se observa en la figura 3.

### 3.2. Api de java

Todos los objetos implementados para esta api fueron desarrollados estrictamente con java puro y sin librerías externas con el objetivo de obtener un código limpio, totalmente multiplataforma y que heredara paquetes únicamente de los módulos swing nativos de java. Como primera instancia se desarrolló el componente funcional de la API, es decir la parte programática, y finalmente se escribió el javadoc, el cual es un documento integrado que describe como se deben usar los objetos

Normalmente para java es necesario construir y empaquetar el código que contenga la interacción de manera manual con java puro. Por ejemplo, en javascript

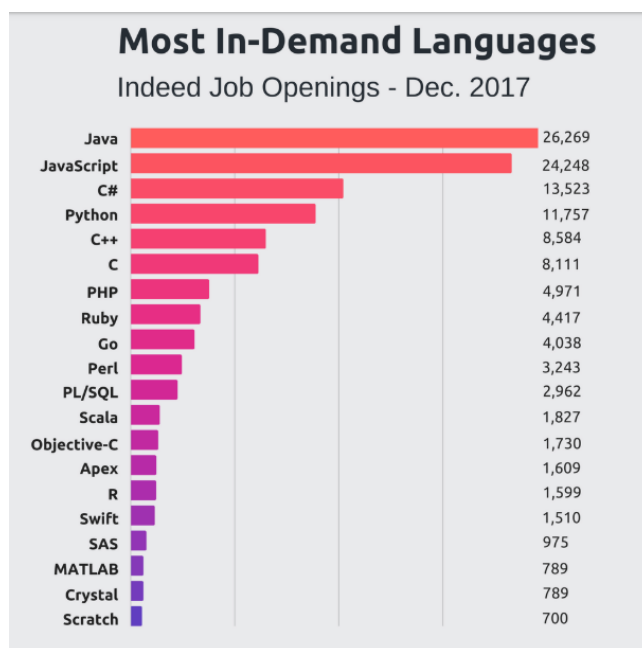


Figura 3: Estadísticas de los lenguajes de programación con más demanda

es posible enlazar funciones directamente sin acceder a paquetes adicionales como los `.event listeners` de java, puesto que todo es considerado un objeto; desde un número hasta una función.

El primer objeto desarrollado en la API de java fue el botón personalizado, usando este objeto contenido en la API se puede crear un botón de cualquier tamaño con una, dos o tres imágenes para los estados; "normal", "presionado" y "encima". Un ejemplo de la utilización de la API se puede ver en la figura 4. Este objeto funciona ajustando automáticamente su tamaño basado en la resolución de las imágenes, esto para complementar con una metodología de diseño: primero el dibujo, luego el programa.

El segundo objeto gráfico desarrollado fue un botón expandible o "caja de opciones", que permitiera generar un botón desplegable con la posibilidad de modificar texto de manera superficial basado en dos o tres imágenes para la parte superior, inferior y media respectivamente. Una implementación utilizada en la aplicación se puede observar en la figura 5. Cada una de las imágenes del botón desplegable debe tener sus 3 imágenes de estado: normal, presionado y encima para interactuar de manera dinámica con el mouse.

Como tercer objeto gráfico, se desarrolló un objeto de animación que podía mover un objeto tipo swing (esta es la clase madre de la mayoría de los objetos gráficos en java) en 4 posibles direcciones; las dos horizontales y las dos verticales, tal como se observa en la figura 6. La función permite modificar la ubi-

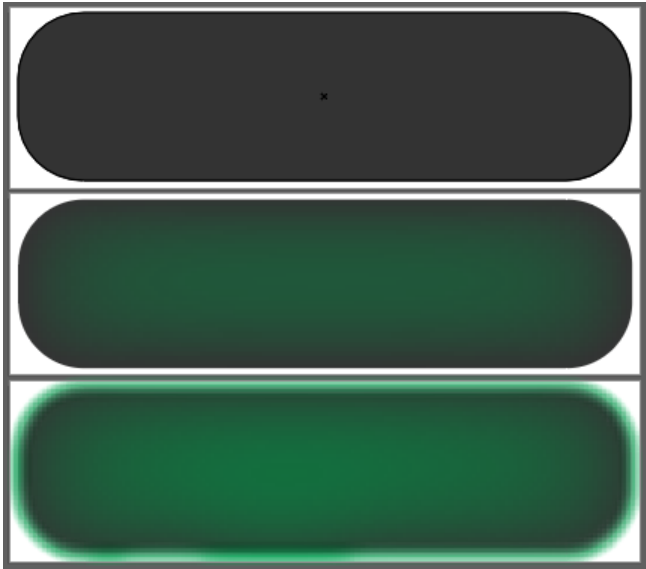


Figura 4: El diseño de estados del boton interactivo

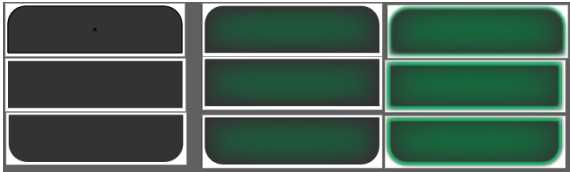


Figura 5: El diseño de los estados del boton interactivo desplegable

cacion actual del objeto moviendolo de manera linear con una rata de refresco dada (actualizacion visual) y una velocidad constante, desde una cordenada X o Y dependiendo de en cual eje coordenado se desee mover el componente.



Figura 6: Movimiento según la ubicación de la función horizontal

El desarrollo de la api fue de importancia para la construccion de la primera version del aplicativo en sitio, la primera version se puede observar en la figura 7, debido a la recursiva implementacion grafica se obtuvo un programa que sobrecargaba un sistema de computo completo (portatil dell con procesador i5 segunda generacion, 4gb de ram DDR3, y un disco duro HDD). Por lo tanto se busco cambiar el entorno de desarrollo a uno con mas flexibilidad a la hora del diseño grafico, con capacidad de funcionar en sistemas de más bajo computo tipo Single Computer Boards como la Raspberry o la Beaglebone. Para probar el desempeño de la api, se realizaron pruebas con los elementos ya diseñados en la api de Java y se observo que resultaba una carga para el cpu realizar las animaciones, principalmente, cuando se deseaba mover más de 5 objetos Jswing.

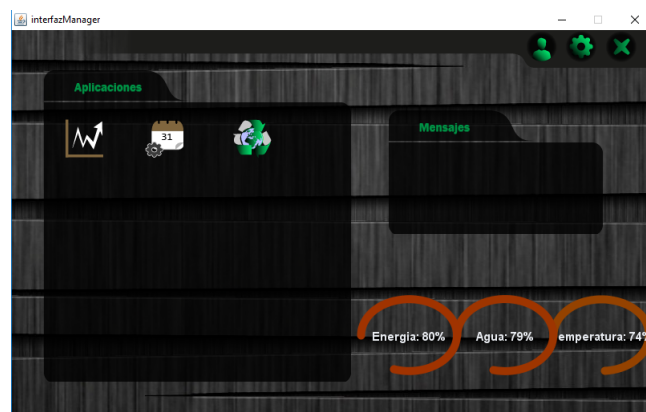


Figura 7: Primer Aproximación del diseño de la interfáz en sitio

Despues de haber programado los elementos necesarios para generar la estructura basica del aplicativo se replanteo el diseño con una estructura más moderna y parecidas a las tendencias en diseño movil tal como se observa en la figura 8 con el objetivo de que se percibiera que ambos aplicativos dearrollados en este sistema podian ofrecer los mismos servicios. Ademas, generar una experiencia de usuario similar en ambas plataformas



Figura 8: Ajuste de diseño de la interfáz en sitio

En un tercer nivel de conceptualizacion se re diseño la interfaz grafica estandarizando y simplificando los componentes que esta contenia; se selecciono la paleta de colores y agregaron algunos nuevos objetos al diseño generalizado como se puede observar en la figura vease en la figura 9. Para la correcta seleccion de la paleta de colores se busco generar una simetria triangular en el espacio de colores HSV, usando como referencia un color sacado del logo oficial de la Çhameleon house”(nombre oficial de la casa diseñada para el solar decathlon).

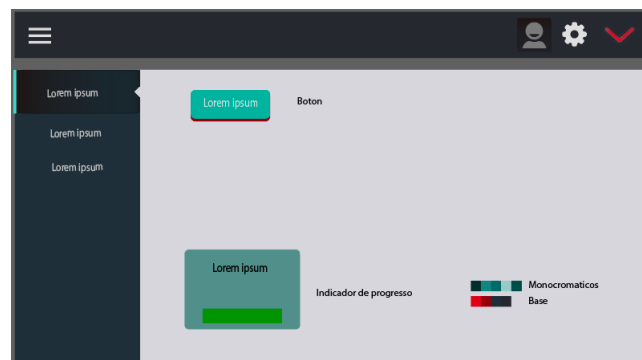


Figura 9: Version final del concepto de diseño

Debido a la estructura de Java, codificar todos los nuevos objetos resultaba una tarea de mayor trabajo y con baja recompensa, es decir, mas lineas de codigo por objeto; como solución se tradujo lo que se tenia desarrollado hasta el momento de la api a un lenguaje de programacion de mas alto nivel (Python) y se utilizo un framework exclusivamente para el desarrollo de la interfaz grafica, debido a que se trataba de un modulo completamente estrito en C le permitia a la aplicacion tener un mejor rendimiento grafico, con la desventaja de dificultar el efecto multiplataforma de python, para mantener el efecto, el framework deberia soportar por completo la plataforma sobre la cual se estu-



viera corriendo (Windows, macós, linux, ubuntu, etc).

### 3.3. Api de Python y Qt

El framework utilizado para desarrollar los modulos graficos se llama Qt y para poder utilizarlo es necesario instalar unas dependencias C especificas y la lipreria respectiva para Python. Una vez Qt esta correctamente configurado el diseño estatico del aplicativo se realiza a partir de un "lenguaje de marcado"(markup language) autogenerado llamado Qml que define las propiedades graficas estáticas de los objetos a utilizar, el Qml viene de la mano con un lenguaje tipo "hoja de estilo"(style sheet), a partir del cual se le adjudican propiedades interactivas y estilos que en conjunto son comprendidos por python y su libreria Pyside. Para esta etapa de diseño se utilizó una paleta de colores complementarios para los objetos importantes y tonalidades oscuras de grises azulados para los fondos. Todos dentro del espectro de colores de la marca de la casa del Solar Decathlon Latinoamerica de la Universidad del Valle

Teniendo en cuenta el nuevo framework se utilizo la herramienta de diseño de Qt desing (oficial del framework) para codificar de manera automatica las propiedades estaticas de la interfaz grafica. Tambien, se codificaron manualmente las propiedades dinamicas en un archivo de Qss que luego fue cargado por la libreria Pyside para ser procesado como un objeto Python.

Primero, se codifico un boton de la misma manera como se codifico el codigo de la API para Java; con este opjeto de python se puede crear un boton de cualquier tamaño con una, dos o tres imagenes para los estados; "normal", "presionadoz .<sup>en</sup>cima". Un ejemplo de la utlizacion de la api se puede ver en la figura 10.



Figura 10: Botón de tamaño flexible genérico de la api de qt

segundo se codifico un boton generico con las mismas funcionalidades del boton anterior, pero en lugar de funcionar basado en imagenes funcionaba con un color dado y las interacciones "normal", "presionadoz .<sup>en</sup>cima"son calcuadas atenuando y resaltando el color para generar un efecto de sombra e iluminacion. Para el aplicativo se utilizo el color mostrado en la figura 11.

Tercero, se desarrollo un expandible o "caja de opciones", personalizada, para ello se modifiko el objeto qss Qoptionbar que permite generar un boton des-

plegable con la posibilidad de modificar texto de manera superficial basado en un color específico.

Cuarto, se programo una barra deslizable para facilitar el proceso de ingresar datos numericos a la aplicacion desde una interfa tactil. Nuevamente se usaron los colores de la paleta de colores seleccionada y se opto por la solucion de diseño mas sencilla posible tal como se puede ver en la imagen 11

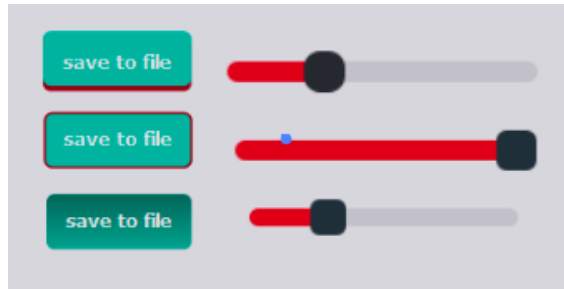


Figura 11: Boton de tamaño fijo usando imagenes y slider

Finalmente se personalizaron los colores de todos los espacios y elementos secundarios utilizados, como: la grafica, la barra de menu, los botones especiales del menu, etc.

## 4. House Manager

Una vez descritos todos los componentes gráficos necesarios para realizar la aplicacion se desarrollaron todos los objetos necesarios para implementar los requerimientos descritos en la hoja requerimientos funcionales”del formato RUP 1. La aplicacion fue desarrollada utilizando el paradigma MVC, para ello, se organizo el proyecto en 3 carpetas principalmente; view, control y model, donde todos los componentes y scripts son gestionados desde un archivo principal. Debido a la naturaleza grafica del proyecto y las limitaciones tecnicas del hardware raspberry, se utilizo Python 2.7 para garantizar la compatibilidad de la mayor cantidad de sistemas operativos y librerias. Para el desarrollo de la aplicacion fueron utilizadas las siguientes librerias:

- paho-mqtt: Esta libreria se utilizó para establecer la comunicación con el broker de mqtt nativo de Google cloud iot.
- numpy, scipy: Estas listrerias fueron utilizadas para facilitar el procesamiento de vectores y mejorar el tiempo de procesamiento de los mismos.
- pyqtgraph: Fue utilizada para gestionar y renderizar las gráficas con un valor optimo de fotogramas por segundo
- pyqtgraph: Fue utilizada para gestionar y renderizar las gráficas con un valor optimo de fotogramas por segundo
- : http.client: Esta libreria fue utilizada para establecer una comunicacion basada en solicitudes http con un .endpoint”del backend
- : pyjwt cryptography: Con esta libreria se desarrolló la etapa de encriptacion de datos adicional basada en Java Web Tokens.

Dentro de la carpeta del proyecto se pueden encontrar scripts que fueron importantes para el despliegue de la aplicacion. Por ejemplo; en el directorio ”./src/installing/”se encuentran los ejecutables para la instalacion de la aplicacion en windows y debian; en el directorio ”./src/test/”se encuentran las rutinas de testing utilizadas para algunas rutinas y objetos utilizados en el programa.

Pensando en la escalabilidad del sistema, se definio un espacio para las .aplicaciones.ªdicionales donde se pudieran incluir nuevas rutinas de adquisicion y control para dispositivos nuevos 12. Como prueba de concepto se utilizo una rutina serial para adquirir el valor de khw acumulado en un contador electrico bifasico Inelca. Respecto a esta consideracion de diseño se hablara mas en proximos apartados

En adicicion, se diagramo la experiencia de usuario de tal forma que el control y la medicion de los dispositivos conectados a la tarjeta tuvieran el mismo



Figura 12: Visualizacion de la escalabilidad del sistema con una app no desarrollada por completo

costo de interacción”, es decir, que ambos estuvieran a la misma cantidad de clicks desde cualquier ventana del sistema. Por ende, todas las aplicaciones adicionadas en versiones futuras le seran mas faciles de asimilar al usuario.

#### 4.1. Requerimientos Funcionales

Durante la etapa de conceptualizacion del proyecto de ingeniería que este documento describe se utilizó una herramienta de planeación llamada RUP; con un documento llamado “Requerimientos funcionales”, dentro de los requerimientos implementados descritos en ese documento estan:

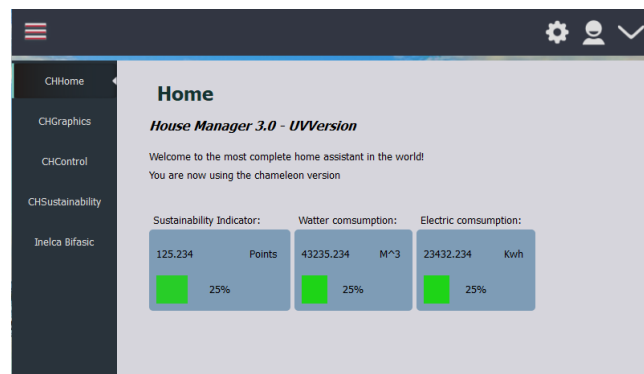


Figura 13: Escena del home del House Manager con barras de progreso para la indicacion dinamica

1. *“El sistema debe mostrar graficamente la cantidad de agua y potencia consumida durante el dia contrastandolas con un valor maximo recomendado”*: Para el desarrollo de esta funcionalidad se utilizó un componente

grafico tipo barra de progreso donde su 100 porciento tenia como referencia la cantidad de agua recomendada para el numero de personas presentes en la vivienda<sup>13</sup>. El numero de personas presentes en la vivienda fue un dato ingresado manualmente por configuracion inicial (sobre el sistema de configuracion se hablara en los siguientes apartados de este capitulo)

2. *"Por defecto el uso de las cargas electricas más significativas debe programarse surante los picos de generacion en la casa, estos horarios podran ser modificados bajo una advertencia de uso no eficiente"*: En miras de implementar esta funcionalidad se utilizo un elemento grafico tipo calendario para configurar de manera individual las ventanas de activación de las cargas electricas en la vivienda<sup>14</sup>. También, se consideró la opcion de activacion o desactivacion inmediata de los circuitos; para ello se, implementó una rutina flexible que permitio ignorar las cargas electricas que el usuario habia modificado manualmente (Esta rutina se nombro "Scheduler Handler"), lo anterior unicamente durante 24 horas para no afectar el objetivo de sostenibilidad de la vivienda. En adicon, se utilizo la conexion mqtt para accionar remotamente cualquiera de los circuitos localmente gestionados, en caso de recibir un mensaje de control para los circuitos el sistema tambien suspendiera por 24 horas la accion del "Schedule Handler".
3. *: .<sup>El</sup> sistema debe poder comunicarse con una base de datos que represente todas las variables y el estado de los circuitos de la vivienda"*: Para la implementacion de este requireimiento se creo un <sup>.endpoint.en</sup> en el servicio de backend con la capacidad de hacer consultas de historicos a la base de datos. La solución tiene este comportamiento puesto que la aplicacion del cliente finl no debe tener ningun tipo de credencial para el acceso a la informacion directmente, con una capa de procesamiento se pueden integrar este tipo de solicitudes con el mismo sistema de encriptacion del broker de mqtt (para más informacion de los endpoints y la rutina de encriptacion vease el capitulo "Backend").

## 4.2. Funciones adicionales

En adición a los requerimientos funcionales, se añadieron funciones a la conceptualización de diseño original para mejorar el funcionamiento de la plataforma y la experiencia de usuario. Dentro de estas funciones podemos encontrar la configuracion del sistema, la posibilidad de guardar localmente para no perder la informacion adquirida, etc. Para comprender mejor estas funciones adicionales, seran descritas a continuacion:

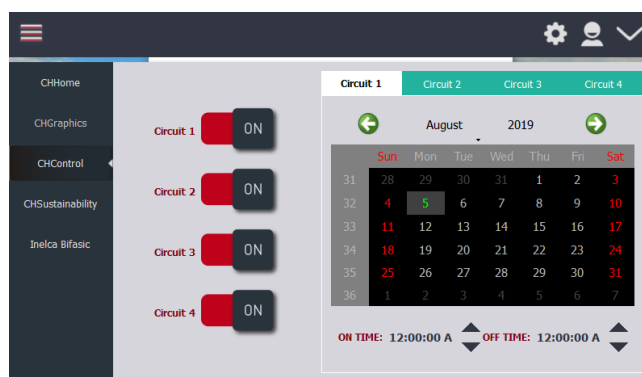


Figura 14: Escena de control del House Managern con sus calendarios

1. **Handler remoto:** Esta característica le permite al sistema cambiar de manera automática entre el modo remoto y el modo local para no perder información en ningún momento de la adquisición. En caso de la caída repentina de la red es capaz de cambiar al modo local sin la pérdida de ningún dato. También, es capaz de entender cuando el usuario desea que el dispositivo funcione permanentemente de manera local.
2. **Sistema de almacenamiento local:** Con esta característica el House Manager puede almacenar todos los datos adquiridos en una hoja de excel generando archivos organizados por días. También es capaz de almacenar en un archivo diferente los datos presentes en la gráfica en cuestión 15.
3. **Ajustes de configuración** Esta es una característica que incluye una ventana adicional donde se configuran los aspectos concernientes al sistema en general 16. Los aspectos configurables del sistema son: La frecuencia en segundos con la que se reportan los datos al servidor, el costo del kilo watt hora, el costo del metro cubico de agua, el día de notificación del correo (ver función "Notificación automática"), el modo de funcionamiento (remoto o local) y el nombre de la ubicación (este nombre es opcional y sirve para administrar los diferentes dispositivos).
4. **Notificación Automática:** Para el correcto desarrollo de esta función se compró el dominio [www.alfagenos.com](http://www.alfagenos.com) como remitente de las notificaciones enviadas por correo, esta notificación tendrá lugar dependiendo de la configuración establecida y notificará el gasto en pesos del consumo eléctrico y de agua hasta la fecha establecida en la configuración.
5. **Autenticación usando google:** Para desarrollar esta funcionalidad se utilizó la api de autenticación de google y se creó una escena aparte para la visualización del usuario 17.

6. **Doble nivel de encriptacion:** Debido a las exigencias de los servidores de google se implementaron 2 capas de seguridad de encriptacion para la comunicacion de la aplicacion con el backend. La primera capa es de tipo TLS y opera para cualquier producto de google, inclusive las operaciones realizadas por los navegadores. La segunda se implemento usando encriptacion RS256.<sup>a</sup> travez de JWT.

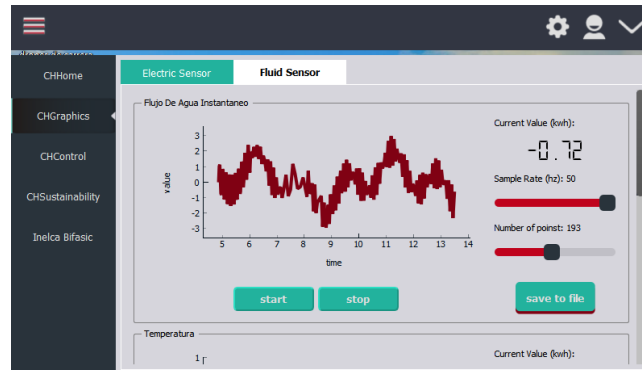


Figura 15: Escena de medición del House Manager, con su respectivo panel para guardar los datos

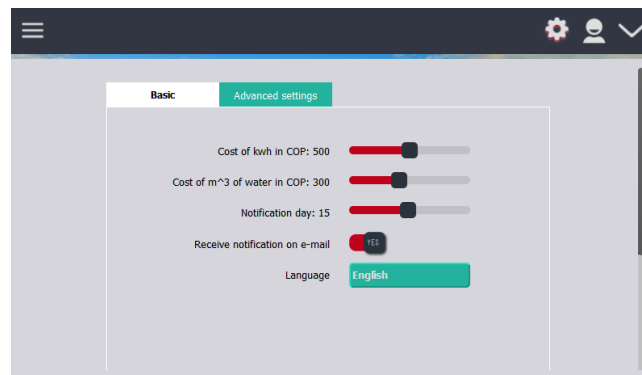


Figura 16: Escena de configuracion del dispositivo House Manager

### 4.3. Algoritmos Utilizados

\*\*\*\* preambulo pra hablar de los algoritmos utilizados en esta aplicacion y como representan threads en python. Mencionar la estructura de su funcion (reciben como informacion un puntero a su pariente mayor que en la mayoria de los casos es uno de los 3 elementos: Modelo, Vista o Controlador, que ocurren en segundo plano, es decir que no tienen un comportamiento muy determinista en el tiempo de ejecucion, etc.

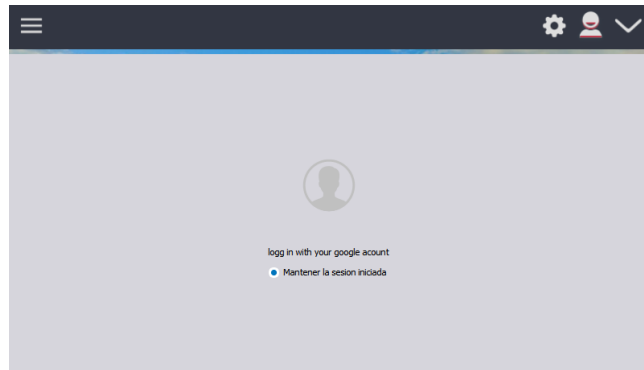


Figura 17: Escena para el inicio de sesion del House Manager

#### **4.3.1. Scheduler Handler**

\*\*\*incluir diagrama del algoritmo y explicarlo

#### **4.3.2. Remote Mode Handler**

\*\*\*incluir diagrama del algoritmo y explicarlo

#### **4.3.3. Mail Notification Handler**

\*\*\*incluir diagrama del algoritmo y explicarlo

#### **4.3.4. Data Report Handler**

\*\*\*incluir diagrama del algoritmo y explicarlo

#### **4.3.5. Indicator Handler**

\*\*\*incluir diagrama del algoritmo y explicarlo



#### 4.4. Serverless Backend

\*\*\*\* describir un poco el funcionamiento de google cloud y su implementacion con los componentes que necesita. Describir las librerias utilizadas para la aplicacion movil

- googleapis explicacion
- jsontokenweb explicacion

\*\*\* mencionar los archivos que son importantes para el proyecto

#### 4.5. Requerimientos Funcionales

\*\*\* hablar sobre como este sistema sirve para desarrollar indirectamente los componentes funcionales de ambos sistemas

1. *.<sup>El</sup> usuario debe poder acceder a la informacion medida en tiempo real"*
2. *.<sup>el</sup> usuario debe poder acceder al historico del mes y la relacion de sus gastos con ellos"*
3. *.<sup>el</sup> usuario debe ser capaz de cambiar los parametros de configuraciones establecidos por defecto para la vivienda del solar decathlon"*
4. *"la aplicacion notificara al usuario cuando la factura este vencida"*
5. *.<sup>el</sup> usuario debe ser capaz de poder ver su impacto ambiental basado en datos cualitativos y cuantitativos "*
6. *.<sup>El</sup> sistema debe ser capaz de notificar las perdidas electricas y por consiguiente economicas de un mal uso de los horarios establecidos por defecto:"*

#### 4.6. Cloud functions

\*\*\*\*\* descripcion de cada una de las cloud functions utilizadas y las necesidades limitaciones y todas las características de las misas; websokets, lifetime period, ram memory 512 free, private network, free authentication, etc

1. **Handler remoto:**
2. **Sistema de almacenamiento local:**
3. **Ajustes de configuracion**

#### **4.7. Function 1**

\*\*\*\* Describir la funcion 1

#### **4.8. Function 2**

\*\*\*\* Describir la funcion 2

#### **4.9. Function 3**

\*\*\*\* Describir la funcion 3 ... y asi para todas las funciones

## 5. User app

\*\*\*\* describir un poco el funcionamiento de react native y su implementacion con componentes nativos. Describir las librerias utilizadas para la aplicacion movil

- fetch
- react-native-jwt

\*\*\* mencionar los archivos que son importantes para el proyecto

\*\*\* hablar e la escalabilidad teniendo en cuenta la organizacion de la app

### 5.1. Requerimientos Funcionales

\*\*\* introducir los requerimientos funcionales de la plataforma y su importancia

- a) *.El usuario debe poder acceder a la informacion medida en tiempo real”* description
- b) *.el usuario debe poder acceder al historico del mes y la relacion de sus gastos con ellos”* description
- c) *.el usuario debe ser capaz de cambiar los parametros de configuraciones establecidos por defecto para la vivienda del solar decathlon”* description
- d) *”la aplicacion notificara al usuario cuando la factura este vencida”* description
- e) *.el usuario debe ser capaz de poder ver su impacto ambiental basado en datos cualitativos y cuantitativos ”* description
- f) *.El sistema debe ser capaz de notificar las perdidas electricas y por consiguiente economicas de un mal uso de los horarios establecidos por defecto:”*

### 5.2. Funciones adicionales

\*\*\*\*\*Funciones adicionales

- a) **Handler remoto:** description
- b) **Sistema de almacenamiento local:** description
- c) **Ajustes de configuracion** description

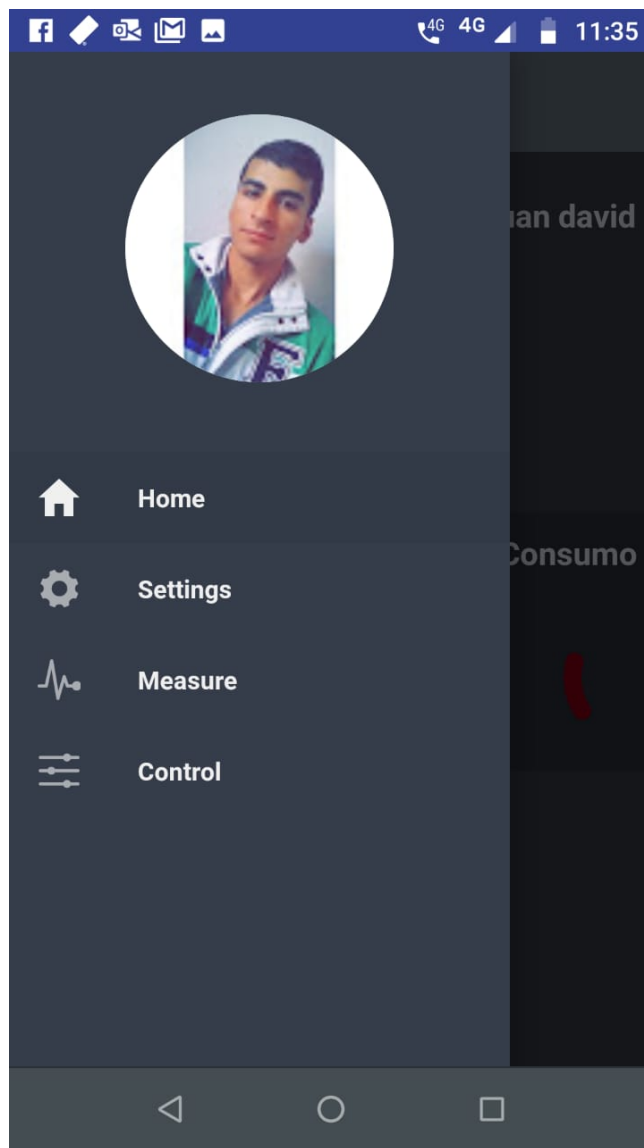


Figura 18: El menu desplegable con las aplicaciones actualmente desplegadas

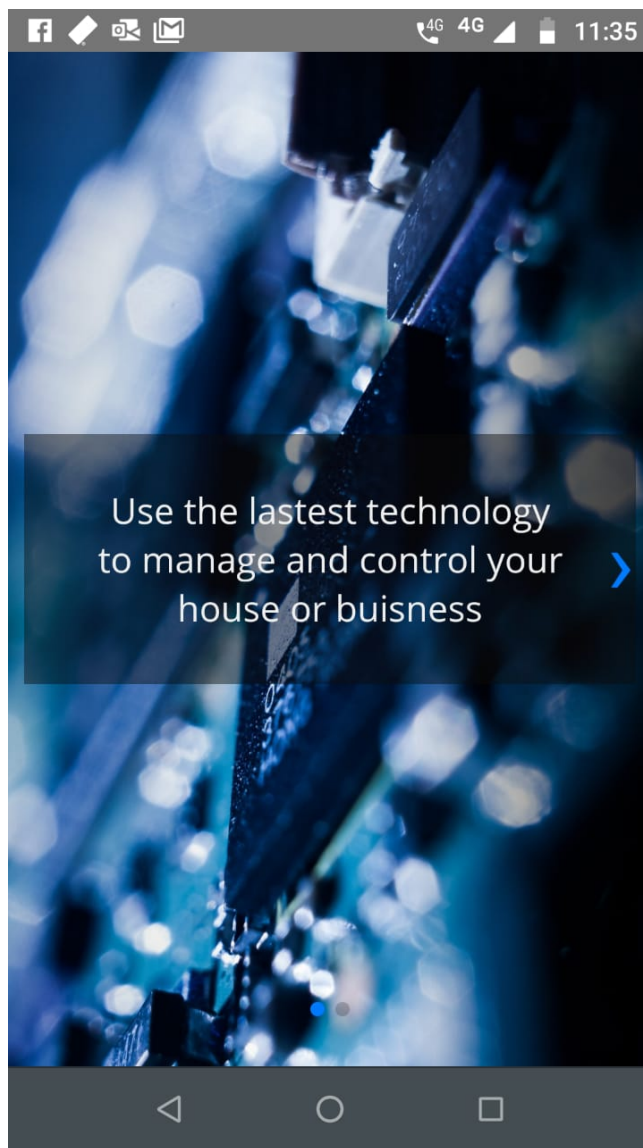


Figura 19: Escena de bienvenida 1

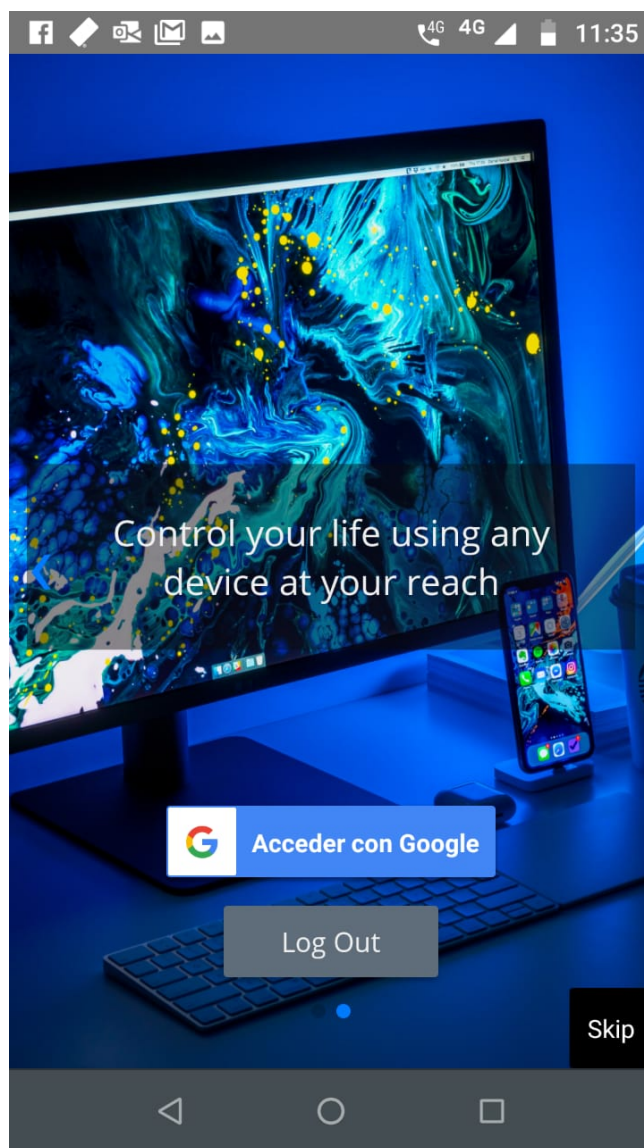


Figura 20: Escena de logg in

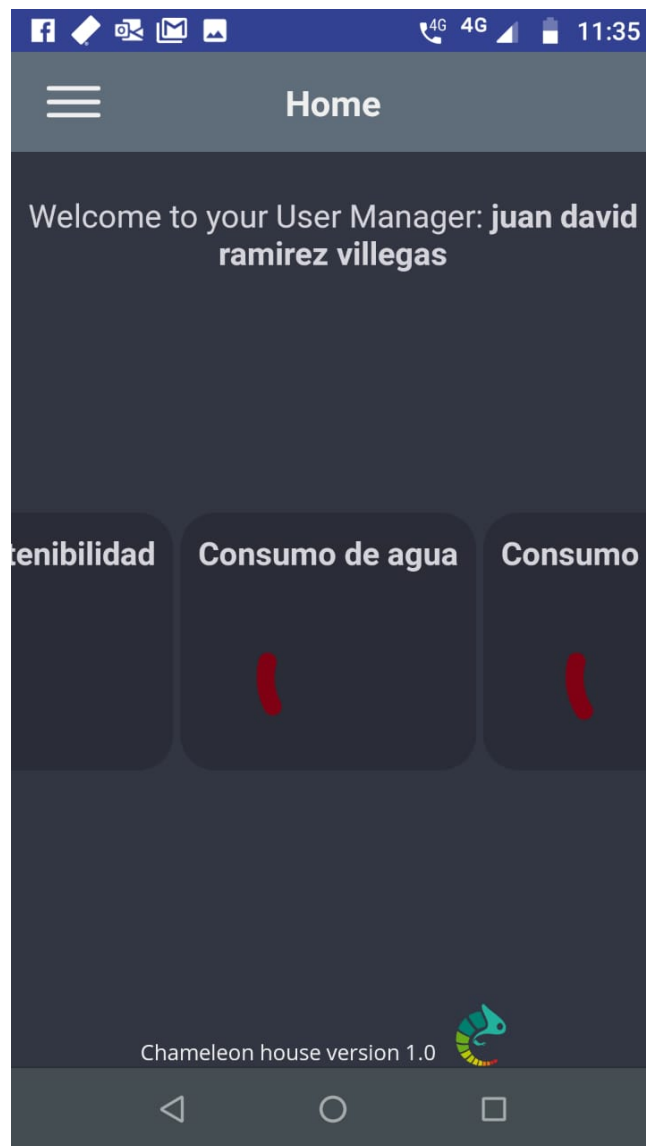


Figura 21: Escena para a visualizacion del consumo historico



Figura 22: Escena para la medicion en tiempo real de las variables



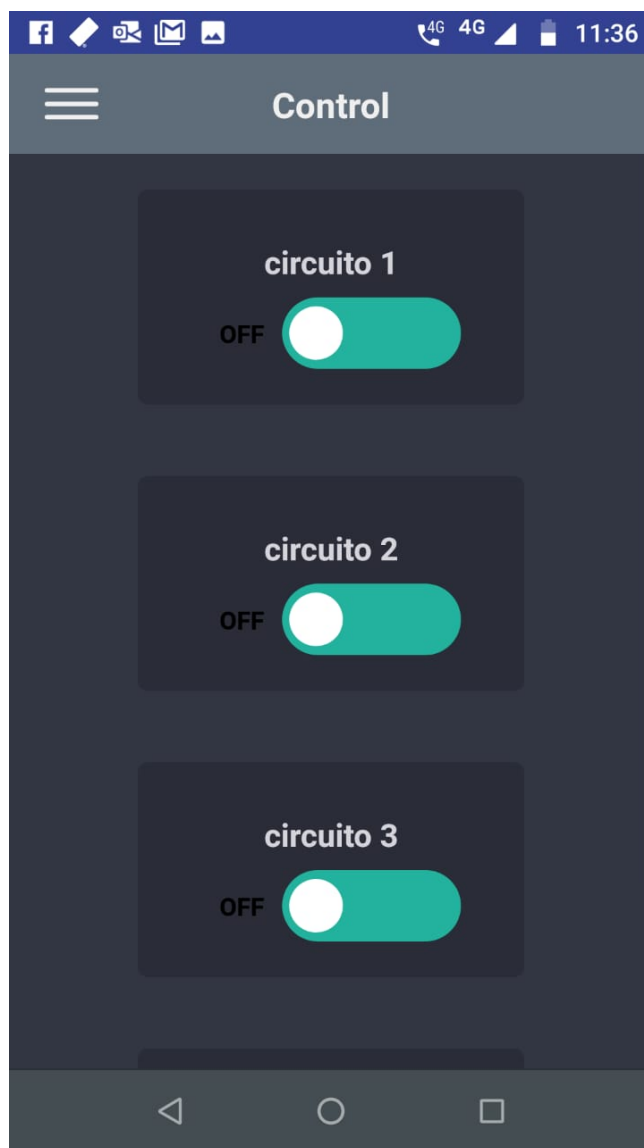


Figura 23: Escena para el control de los circuitos de la vivienda

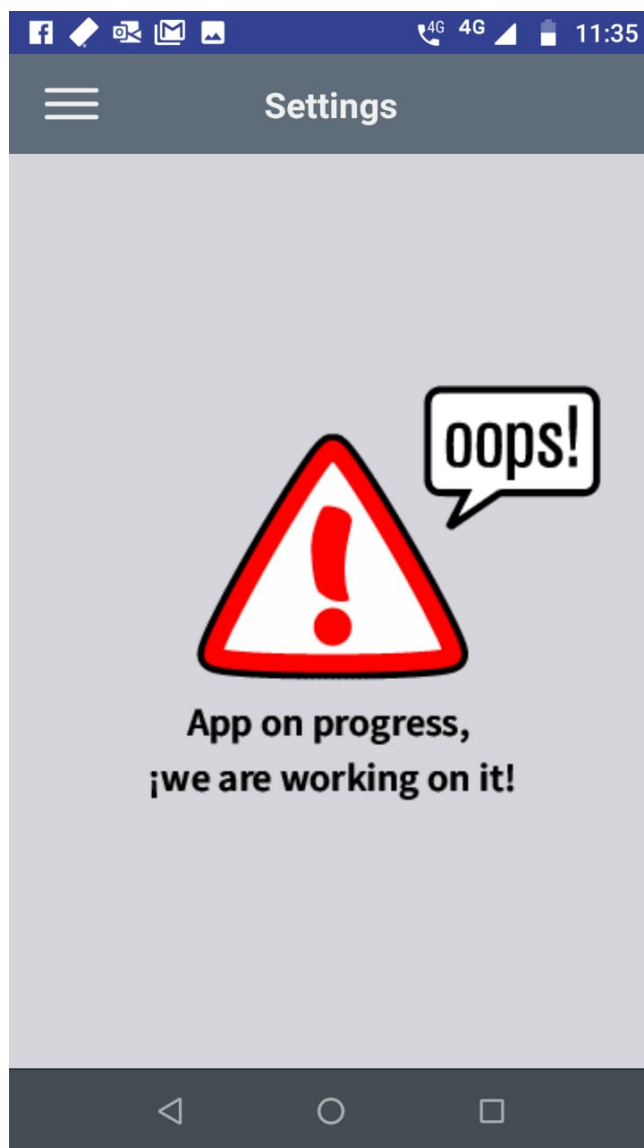


Figura 24: Escena de configuraciones, actualmente no disponible

### **5.3. Algoritmos Utilizados**

\*\*\*\* describir la naturaleza de un entorno de programacion orientado a relaciones como los algoritmos eran codificados en funcion de cambiar una propiedad del elemento

#### **5.3.1. Scheduler Handler**

\*\*\*incluir diagrama del algoritmo y explicarlo

#### **5.3.2. Remote Mode Handler**

\*\*\*incluir diagrama del algoritmo y explicarlo

## Referencias

- D. Rolando, S. Mora, A. A. Gómez, Y. Katherine, G. Ibáñez, J. Camilo, and R. Salcedo, “embebidos Raspberry Pi y Arduino para el manejo de un brazo robótico mediante una aplicación Android,” vol. 1, no. 2015, pp. 69–96, 2014.
- J. E. Giral Sala, R. Morales Caporal, E. Bonilla Huerta, J. J. Rodriguez Rivas, and J. D. J. Rangel Magdaleno, “A Smart Switch to Connect and Disconnect Electrical Devices at Home by Using Internet,” *IEEE Latin America Transactions*, vol. 14, no. 4, pp. 1575–1581, 2016.
- J. Cabrera, M. Mena, A. Parra, and E. Pinos, “Intelligent assistant to control home power network,” *2016 IEEE International Autumn Meeting on Power, Electronics and Computing, ROPEC 2016*, no. Ropec, 2017.
- A. Imteaj, T. Rahman, M. K. Hossain, M. S. Alam, and S. A. Rahat, “An IoT based Fire Alarming and Authentication System for Workhouse using Raspberry Pi 3,” *ECCE 2017 - International Conference on Electrical, Computer and Communication Engineering*, no. 0, pp. 899–904, 2017.
- R. Shete and S. Agrawal, “IoT Based Urban Climate Monitoring using Raspberry Pi,” *International Conference on Communication and Signal Processing, April 6-8, 2016, India IoT*, pp. 2008–2012, 2016.
- W. Mauricio, G. Ramírez, H. José, C. Flórez, E. G. Restrepo, A. Tatiana, and Z. Ortiz, “Redes inteligentes en el sistema eléctrico colombiano: Revisión de tema,” *Tecnura*, vol. 21, pp. 119–137, 2017. [Online]. Available: <http://revistas.udistrital.edu.co/ojs/index.php/Tecnura/article/view/12396/13024>
- A. Howedi and A. Jwaid, “Design and implementation prototype of a smart house system at low cost and multi-functional,” *FTC 2016 - Proceedings of Future Technologies Conference*, no. December, pp. 876–884, 2017.
- C. Herrera, J. Simon, C. E. Rodriguez, A. F. Jaramillo, A. Bernal, S. Ospina, and M. Luna, “Solar Decathlon Latin America and caribbean,” *Tech. Rep.* 1967, 2015.