Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Aplicación domótica en Android con OpenHAB para el control de los dispositivos del hogar

Autor: José de Lózar Alameda
Tutora: María Teresa Ariza Gómez

Dpto. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Aplicación domótica en Android con OpenHAB para el control de los dispositivos del hogar

Autor:
José de Lózar Alameda

Doctora:
María Teresa Ariza Gómez

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Trabajo Fin de Grado: Aplicación domótica en Android con OpenHAB para el control de los dispositivos del hogar

Autor: José de Lózar Alameda

Tutora: María Teresa Ariza Gómez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Sevilla, 2020

"La información es poder"
Francis Bacon

Agradecimientos

Me es difícil escribir estas palabras debido a todo lo que significan, sin querer dejarme a nadie. Durante la realización de este Trabajo Fin de Grado han acontecido momentos extraordinarios que se alejan de todo lo que cualquier persona pudiera pensar. Por ello me es difícil y me emociona a su vez escribir estas palabras para darle las gracias a todos aquellos que han estado durante todo este tiempo.

En primer lugar, agradecer a la tutora de este Trabajo Fin de Grado, María Teresa Ariza Gómez, toda la paciencia, el tiempo, los consejos, las reuniones, las presentaciones y el empeño que ha dedicado para ayudarme en la realización. Gracias por dejarme realizar aquella idea que se me pasó un día por la cabeza y que después de muchas reuniones cogió forma y sentido. A su vez agradecer a todos los profesores que he tenido durante la carrera que me han hecho madurar e ilusionarme aún más, si se puede, por el mundo de las Telecomunicaciones y todo lo que abarca.

Me gustaría agradecer, desde lo más profundo de mi ser, a mi familia que siempre ha estado junto a mí. Sin ellos no hubiera empezado esta carrera ni llegado hasta donde estoy ahora. A mi madre su eterna paciencia y comprensión. A mi padre las constantes charlas motivacionales y sus ganas de vida. A mi hermana su gran locura y capacidad de volver feliz a los demás. Y a Consuelo, por sacar día tras día lo mejor de mí, por hacerme ver la vida desde distintos puntos de vista, por hacerme madurar a pasos agigantados, por ser la luz de mi camino y hacer de este el camino que quiero y el más bonito de los que se podría elegir. Gracias por ser mi cruz en el mapa.

Gracias a mis amigos.

Gracias a los que ya se fueron, que siempre estarán.

José de Lózar Alameda

Sevilla, 2020

Resumen

Conforme pasan los años, la sociedad siente la necesidad de estar conectada a Internet. La posibilidad de poder interconectar todo lo que nos rodea es un hecho y cada vez más las nuevas tecnologías se adaptan a dicha necesidad. La domotización del hogar permite el control de cualquier elemento electrónico conectado a la red, desde persianas hasta lavadoras.

El objetivo de este Trabajo Fin de Grado es analizar y utilizar la plataforma de código abierto cuya principal funcionalidad es la automatización del hogar que es OpenHAB [1], así como el desarrollo de una aplicación móvil en Android que sirva de interfaz y conexión con cada hogar, e implementar un servicio en la nube que sirva para la autenticación de los usuarios en la aplicación. OpenHAB utiliza tecnología REST para consultar y publicar los diferentes estados que pueden estar en una casa [2].

Abstract

As the years go by, society feels the need to be connected to the Internet. The possibility of being able to interconnect everything around us is a fact and more and more new technologies are adapting to this need. Home automation allows the control of any electronic element connected to the network, from blinds to washing machines.

The objective of this Final Degree Project is to analyze and use the open source platform whose main functionality is home automation which is OpenHAB, as well as the development of a mobile application on Android that serves as an interface and connection with each home, and implement a cloud service to login to application users. OpenHAB uses REST technology to query and publish the different states that can be in a house.

Índice

ÍNDICE DE FIGURAS

Notación y acrónimos

AJAX	Asynchronous JavaScript And XML
API	Interfaz de Programación de Aplicaciones
AWS	Amazon Web Services
BLE	Bluetooth Low Energy
CPU	Central Processing Unit
DNS	Domain Name System
EC2	Elastic Compute Cloud
GPIO	General Purpose Input/Output
GPS	Global Positioning System
GPU	Graphics Processing Unit
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IBM	International Business Machines Corporation
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IP	Internet Protocol
I2C	Inter-Integrated Circuit
JDK	Java Development Kit
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
MQTT	Message Queue Telemetry Transport
M2M	Machine to Machine
NAT	Network Address Translation
OS	Operating System
OSI	Open System Interconnection

POM	Project Object Model
QoS	Quality of Service
RAM	Random Access Memory
REST	Representational State Transfer
RGB	Red Green Blue
RH	Relative Humidity
SD	Secure Digital
SDRAM	Synchronous Dynamic Random-Access Memory
SPI	Serial Peripheral Interface
SQL	Structured Query Language
SSH	Secure SHell
STUN	Session Traversal Utilities for NAT
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UI	User Interface
URL	Uniform Resource Locator
USB	Universal Serial Bus
UTF-8	8-bit Unicode Transformation Format
V	Voltio
Vcc	Tensión de colector
Vdd	Tensión de drenador
VIN	Tensión de entrada
XML	eXtensible Markup Language
ºC	Grados Centígrados

1 INTRODUCCIÓN

1.1 Motivación del trabajo

El sector del Internet De Las Cosas comenzó su auge en 1999 y hoy en día sigue en pleno desarrollo sin vislumbrar ningún final aparente. Esta constante evolución ha sido gracias a las nuevas tecnologías capaces de soportar dicho progreso. La interconexión de las cosas cotidianas hace más fáciles las tareas y el control absoluto del hogar.

Las aplicaciones para los dispositivos que están conectados a internet son infinitas pero se pueden subdividir en tres ramas diferentes según su utilidad: la rama de consumo, la rama empresarial y la rama de infraestructura. Observando la primera de las ramas, se puede decir que es el ámbito para la que se crean más dispositivos IoT. Entretenimiento, automóviles, hogar y salud. El principal problema de este tipo de dispositivos es que al existir tanta cantidad, la calidad es muy variada llegando a encontrar dispositivos que pueden llegar a suponer un agujero de seguridad en los hogares. En cuanto a la segunda de las ramas, el IoT empresarial, los dispositivos se destinan a la recolección de información útil de los consumidores. Sirven para hacer estudios de mercado y hacer un seguimiento de los hábitos de los consumidores. El IoT industrial se basa en el control de operaciones y el seguimiento de pautas de control en procesos industriales, como puede ser el control de una cadena embotelladora o una fábrica de automóviles o la recolección de productos en la agricultura.

La gran cantidad de dispositivos en el mercado hace que la interconexión de estos mismos pueda llegar a ser una ardua tarea. OpenHAB nace en 2010 para ofrecer una solución y proponer un software de automatización del hogar de código abierto y escrito en Java. Con este software se pueden implementar las tecnologías más utilizadas, pudiendo así interconectar casi cualquier dispositivo IoT. Propone además una implementación sencilla y la capacidad de desarrollar cualquiera de las tres ramas del mundo del IoT, aunque principalmente se haya basado en la rama del consumo.

1.2 Objetivos y enfoque

El presente Trabajo Fin de Grado tiene marcados los siguientes objetivos:

- Análisis de OpenHAB.
- Instalación de OpenHAB en Raspberry Pi [3].
- Montaje de escenario en hogar con sensores Arduino [4].
- Instalación y uso de servidor y cliente del protocolo STUN [5].
- Creación de servicio REST en Amazon Web Services para el registro y autenticación de los usuarios [6].

1.2.1 Objetivos específicos

- Lectura de información dada por sensores: La lectura de datos se hará sobre el estado actual de los diferentes sensores Arduino que se encuentran incorporados dentro del sistema. Se utilizará el sensor de temperatura y humedad DHT11 [7], varios leds como simulación de bombillas en el sistema domótico y un relé con bombilla (dos RELAY 5Vdc) [8]. Para dicha tarea se utilizará el protocolo MQTT [9], suscribiéndose al topic indicado para el sensor o pidiendo la información al sensor.

- Envío de órdenes a sensores: Las órdenes consisten en el envío de nuevos estados a los sensores que deben cambiar de estado o para consultar el estado de los sensores. Para ello se utilizará MQTT, publicando el nuevo estado en el topic correspondiente.
- Almacenamiento y consulta de credenciales: Uso de la base de datos PostgreSQL [10], localizada en el servidor Amazon Web Services, que almacena el nombre de usuario, contraseñas, IP destino del servidor asociado al usuario y el puerto de dicho servidor.
- Creación de aplicación Android domótica: Aplicación para acceso a servidor del hogar de forma local o remota. Acceso según credenciales, creación dinámica de objetos y consulta de estado de dichos objetos a través de la API REST incorporada en OpenHAB.
- Creación de servicio de autenticación: Servicio REST, que hace uso del almacenamiento y consulta de credenciales. Comprueba que el usuario y contraseña dados en la petición de consulta exista en el sistema, devolviendo como respuesta la IP destino del servidor local asociado al usuario y el puerto al que debe conectarse.
- Instalación y uso de servidor STUN [11]: Recibe petición para saber la IP pública del router que enmascara la IP privada del servidor local. Devuelve dicha IP pública. El puerto en este caso debe ser configurado por el usuario.
- Implementación de servicio REST [12]: El servicio se implementa a través de SPRING [13]. El cual permite el registro y la autenticación del usuario, para el acceso al servicio domótico.
- Instalación y uso de servicio domótico: Para automatizar el hogar se utiliza la plataforma OpenHAB.
- Instalación y uso de módulos Arduino: Se utilizan leds, relés y sensores de temperatura conectadas a placas NodeMCU [14], las cuales actuarán como esclavos.
- Instalación y uso de Raspberry Pi: es el servidor central. Actúa como maestro, gestionando las placas NodeMCU.

1.2.2 Funcionalidades de cara al cliente o al usuario final

- Fácil instalación de servicio local.
- Registro en el servicio.
- Acceso seguro a la aplicación.
- Lista de sensores conectados.
- Gestión e interacción de sensores.
- Consulta la distribución de sensores.
- Consulta del número de Items creados en el sistema domótico.
- Consulta del número de estancias implementadas en el sistema domótico.
- Consulta del estado de la temperatura en tiempo real.
- Consulta del estado de bombillas en tiempo real.
- Cambiar el estado de las bombillas implementadas en tiempo real.
- Cambiar el estado aumentando o disminuyendo el botón slider.

1.2.3 Esquema de la arquitectura

En el esquema de la figura 1-2-3-1 se observan los cinco principales componentes del Trabajo Fin de Grado, que son: el servidor web en AWS [15], la aplicación móvil de Android, la Raspberry Pi, la NodeMCU y el sensor Arduino. También en el esquema se puede observar los subcomponentes de estos. Mencionar especialmente el sistema MQTT instalado en OpenHAB que se puede ver un esquema en la figura 1-2-3-2. La comunicación entre el servidor web, la aplicación móvil y la Raspberry Pi se harán utilizando el protocolo HTTP mientras que la comunicación entre la Raspberry Pi y la NodeMCU será utilizando el protocolo MQTT.

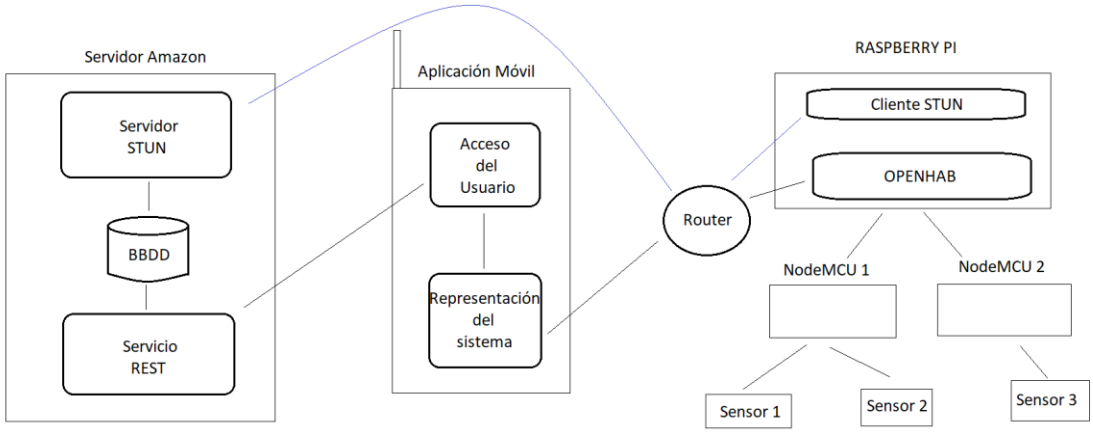
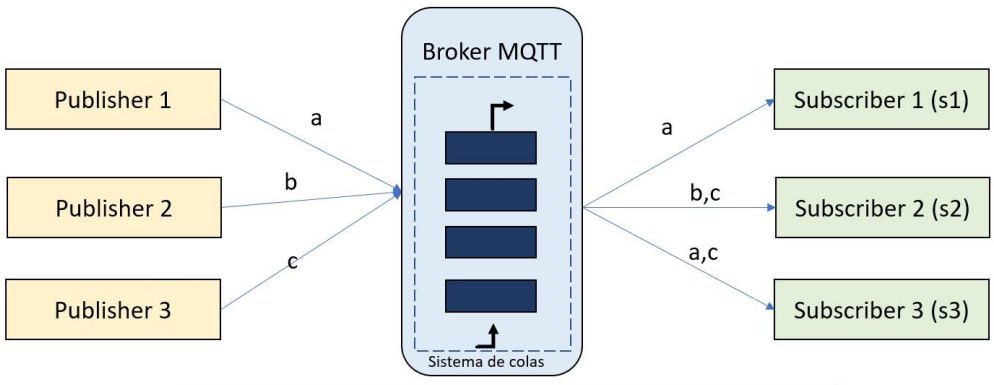


Figura 1-1: Esquema de la arquitectura del Trabajo Fin de Grado.



Topic Name	Subscriber
a	s1,s3
b	s2
c	s2,s3

Figura 1-2: Esquema del modelo MQTT.

1.3 Estructura de la memoria

Esta memoria comenzará exponiendo los recursos hardware y software empleados para la realización del Trabajo Fin de Grado. Avanzará con un análisis profundo de la herramienta OpenHAB que proporciona la capacidad de interconexión de los dispositivos y el análisis del protocolo MQTT de comunicación para tener conciencia de las diferentes características con ambos y las extensas capacidades que poseen pero que en algún caso no se ha incorporado en el desarrollo de este Trabajo Fin de Grado, para que el lector sea consciente de hasta donde pueden abarcar estas tecnologías. Finalmente se expondrá la solución propuesta y cómo se ha construido el escenario, así como un apartado de futuras líneas de desarrollo y comentarios sobre el Trabajo Fin de Grado.

1.4 Justificación del trabajo

1.4.1 Limitaciones y problemas

En la búsqueda de una solución factible han surgido las siguientes limitaciones y problemas:

- La necesidad de cifrar las comunicaciones ante posibles ataques.
- El uso de un certificado digital autofirmado en vez de un certificado digital verificado por una entidad certificadora.
- El cambio de IP del router del hogar con cada reinicio de este.
- La implantación de sensores a los dispositivos sin conexión que forman parte de la red eléctrica del hogar.
- La necesidad de un sistema intermedio entre la aplicación móvil y el servidor OpenHAB instalado en la Raspberry Pi.

1.4.2 Soluciones propuestas

Como soluciones a las limitaciones y problemas se han propuesto conforme al orden del apartado anterior:

- El uso de un certificado digital autofirmado para el cifrado de la conexión entre aplicación móvil y servicio de registro y autenticación.
- Seguir adelante con el certificado autofirmado, ignorando cualquier aviso que pueda mostrar un navegador web.
- El uso del protocolo STUN para conocer la IP del router, donde se encuentra el servidor OpenHAB, que debe ser actualizada si este sufre un reinicio.
- Se ha propuesto como medida para la realización del Trabajo Fin de Grado instalar los sensores en una maqueta de una casa para no hacer cambios en un hogar real.
- El registro en la herramienta EC2 de Amazon Web Service durante un año de forma gratuita para la instalación del servicio de registro y autenticación y el servidor STUN.

2 RECURSOS UTILIZADOS

La imaginación es más importante que el conocimiento. El conocimiento es limitado. La imaginación rodea al mundo.

Albert Einstein

A continuación se exponen los recursos utilizados para llevar a cabo la implementación del sistema domótico. Se comenzará describiendo los principales recursos hardware definiendo brevemente cada uno, pasando posteriormente a la descripción de los recursos software.

2.1 Recursos Hardware

2.1.1 Raspberry Pi Model 3b+

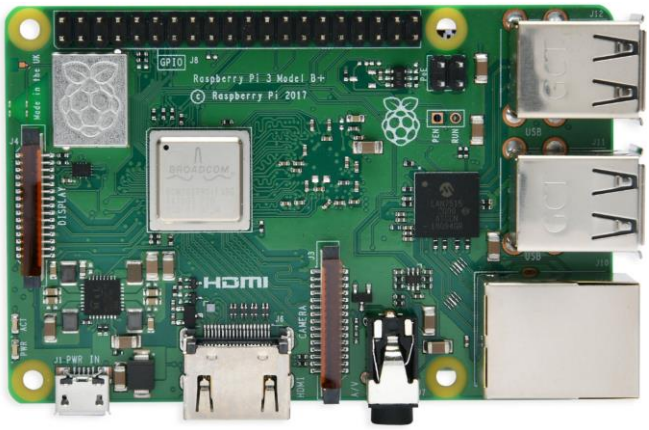


Figura 2-1: Raspberry Pi Modelo 3b+.

La Raspberry Pi Modelo 3b+ es el modelo más avanzado de la familia de las Raspberry Pi. Esta placa es como un ordenador de bajo coste y tamaño reducido. Actuará como servidor local en el hogar.

Las características de la Raspberry Pi Modelo 3b+ son:

- CPU + GPU: Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz.
- RAM: 1GB LPDDR2 SDRAM.
- Wi-Fi: 2.4GHz y 5GHz IEEE 802.11.b/g/n/ac.
- Bluetooth: 4.2 BLE.
- Ethernet: Gigabit Ethernet sobre USB 2.0 (300Mbps).
- GPIO de 40 pines.

- HDMI
- 4 puertos USB 2.0.
- Alimentación Micro USB.
- Power-over-Ethernet.
- Entrada/salida de auriculares/vídeo compuesto.

2.1.2 NodeMCU v3

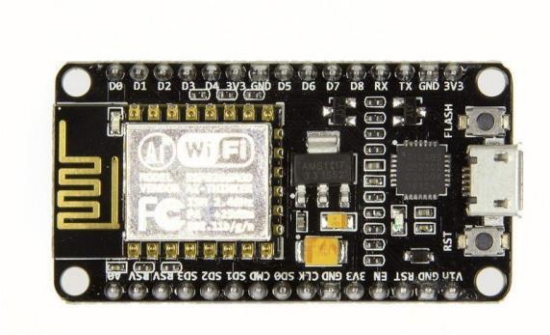


Figura 2-2: Placa NodeMCU v3.

La placa NodeMCU es una placa de desarrollo basada en el ESP-12E, la cual facilita la conexión entre dispositivos. Esta placa de desarrollo es totalmente abierta, a nivel de software y hardware. Actuará como recolector de información de los sensores y le enviará la información a la Raspberry Pi.

Las características de la NodeMCU v3 son:

- Procesador: ESP8266 @ 80MHz (3.3V) (ESP-12E).
- 4MB de memoria FLASH (32 MBit).
- WiFi 802.11 b/g/n.
- Regulador 3.3V integrado (500mA).
- Conversor USB-Serial CH340G / CH340G.
- Función Auto-reset.
- 9 pines GPIO con I2C y SPI.
- 1 entrada analógica (1.0V max).
- 4 agujeros de montaje (3mm).
- Pulsador de RESET.
- Entrada alimentación externa VIN (20V máx.).

2.1.3 Sensor DHT11

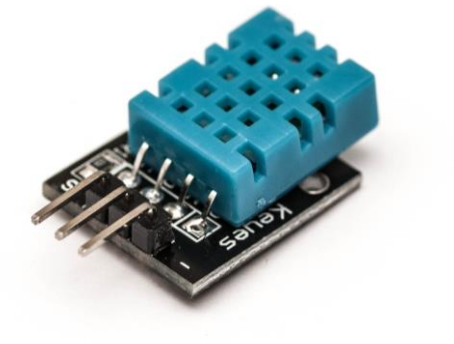


Figura 2-3: Sensor de temperatura DHT11.

El DHT11 es un sensor de temperatura y humedad de bajo coste. Muestra los datos a través de una señal digital mediante el pin de datos. Consta de tres patas o pines, la tierra, el pin de datos y la entrada de tensión. Gracias a la librería de Arduino DHT.h de Adafruit, el uso del sensor de temperatura y humedad es accesible.

Las características del sensor de temperatura DHT11 son:

- Alimentación: 3Vdc ≤ Vcc ≤ 5Vdc.
- Rango de medición de temperatura: 0 a 50 °C.
- Precisión de medición de temperatura: ±2.0 °C .
- Resolución Temperatura: 0.1°C.
- Rango de medición de humedad: 20% a 90% RH.
- Precisión de medición de humedad: 4% RH.
- Resolución Humedad: 1% RH.
- Tiempo de medición: 1 segundo.

2.1.4 Led

Figura 2-4: Dispositivo led.

Un diodo emisor de luz está formado por un diodo de unión p-n el cual emite luz cuando está activado. Al aplicar una tensión adecuada en los terminales, los electrones se recombinan con los huecos en la región de la unión p-n, liberando energía en forma de fotones [17].

2.1.5 Modulo Relé Arduino

Figura 2-5: Módulo de 2 relés.

El módulo relé Arduino es un dispositivo electromagnético que al ser estimulado por una corriente débil, actúa como interruptor abriendo o cerrando un circuito en el que se disipa una potencia mayor que en el circuito que estimula el dispositivo.

Las características de un módulo relé Arduino son:

- Cada relé de 5v requiere una corriente de 20mA.
- Activación mediante señal de 5V que puede controlar directamente el microcontrolador.
- Tolerancia: 250v AC / 30v DC 10A.

2.1.6 Teléfono Android

Figura 2-6: Teléfono móvil Huawei P30.

El dispositivo móvil que se usará para el alojamiento de la aplicación Android que actuará como cliente.

2.1.7 Portátil

Figura 2-7: Portátil Acer Aspire VX 15.

Portátil usado para el desarrollo y las pruebas del Trabajo Fin de Grado.

2.1.8 Tarjeta MicroSD

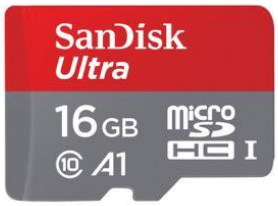


Figura 2-8: Tarjeta MicroSD Clase 10.

Tarjeta de capacidad media y gran calidad, la cual aloja el sistema operativo de la Raspberry Pi.

2.1.9 Punto de Acceso

Figura 2-9: Punto de acceso TP-Link.

Este punto de acceso proporciona la red local donde se alojará el sistema domótico, además de servir como puerta de enlace a Internet.

2.1.10 Cable RJ45

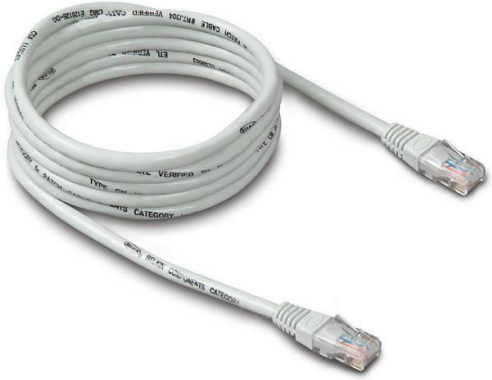


Figura 2-10: Cable Ethernet RJ45.

Cable de red Ethernet necesario para que la Raspberry Pi siempre tenga una conexión óptima.

2.1.11 Protoboard

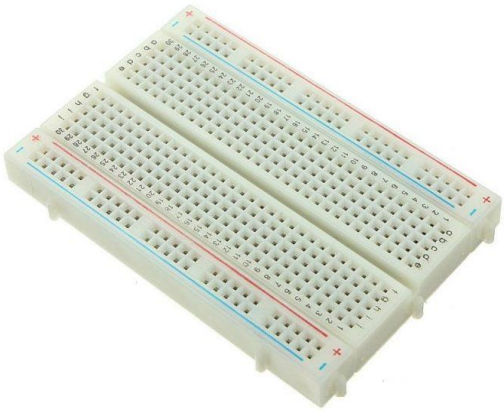


Figura 2-11: Protoboard.

La protoboard sirve en este trabajo para el montaje de los sensores y placas del escenario.

2.1.12 Cables Puente

Figura 2-12: Cables Puente Macho-Macho.

Cables de conexión entre placas y sensores.

2.1.13 Fuentes de alimentación

2.1.13.1 Fuente de alimentación Raspberry Pi

Figura 2-13: Fuente de alimentación Micro USB.

Alimentación de 5V que se le proporciona a la Raspberry Pi.

2.1.13.2 Fuente de alimentación Punto de Acceso

Figura 2-14: Fuente de alimentación punto de acceso.

Alimentación de que se le proporciona al punto de acceso.

2.1.13.3 Fuente de alimentación NodeMCU

Fuente de alimentación compatible con la placa NodeMCU.

- Módulo alimentación Arduino

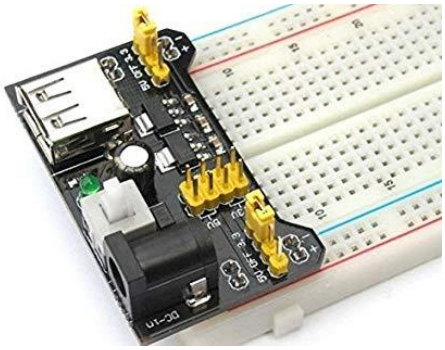


Figura 2-15: Módulo de alimentación Arduino.

- Pila
- USB
- Fuente de alimentación

- Fuente alimentación Micro USB

2.2 Recursos Software

2.2.1 Arduino IDE

Figura 2-16: Arduino IDE.

Software de Arduino de código abierto, cuya finalidad es escribir código y subirlo a la placa. Escrito en java. Se ejecuta en Windows, Mac OS X y Linux. Se puede utilizar con cualquier placa Arduino [18].

2.2.2 Win32DiskImager

Figura 2-17: Win32DiskImager.

Win32DiskImager está diseñado para escribir una imagen en un dispositivo extraíble como una tarjeta SD [19]. Sirve para montar la imagen del sistema operativo en una tarjeta SD.

2.2.3 SDcardFormatter

Figura 2-18: SDcardFormatter.

Programa de formateo de dispositivos extraíbles [20].

2.2.4 Putty

Figura 2-19: Putty.

Software de código abierto que funciona como cliente SSH y telnet, el cual está disponible para Windows y plataformas Unix [21]. Se utiliza para abrir una sesión con el servidor en AWS y la Raspberry Pi.

2.2.5 GNU/Linux

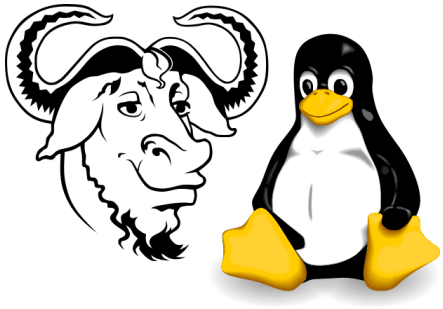


Figura 2-20: GNU/Linux

Sistema operativo multiplataforma, multiusuario y multitarea [22]. La distribución Ubuntu ha sido utilizada para la fase de desarrollo del trabajo y su puesta en marcha en el servidor en la nube [23].

2.2.5.1 Raspbian

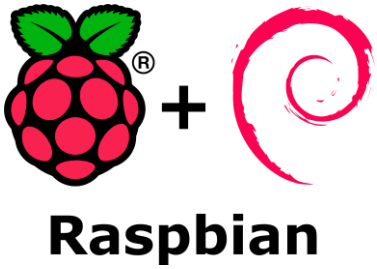


Figura 2-21: Sistema Operativo Raspbian.

Sistema operativo oficial compatible con la Raspberry Pi [24].

2.2.5.2 OpenHABian

OpenHABian es un sistema Linux con los paquetes recomendados de OpenHAB ya incorporados [25].

2.2.5.3 Ubuntu/Centos/Debian

Distribuciones Linux que se usarán en este Trabajo Fin de Grado como sistema operativo en el que se desarrollará el mismo.

2.2.6 MQTT

Figura 2-22: MQTT.

MQTT son las siglas de Message Queing Telemetry Transport. Es un protocolo de comunicación M2M. Está basado en la pila TCP/IP. Basado en el patrón publicador/suscriptor, en el que un cliente publica un tema llamado topic y otro cliente interesado en recibir los mensajes de este tema se suscribe a él para poder recibirlos. Todos los mensajes llegan a un nodo central llamado "Broker" que distribuirá los mensajes a cada destinatario.

2.2.6.1 MQTT server

Software de código abierto que implementa el nodo central o Broker para el funcionamiento del sistema.

2.2.6.2 MQTT client

Software de código abierto que implementa el cliente del sistema.

2.2.6.3 MQTT.fx

Figura 2-23: MQTT.fx.

Aplicación que actúa como cliente y facilita la publicación de mensajes y la suscripción a temas [26]. Se utiliza para probar las configuraciones de MQTT en este trabajo.

2.2.7 Google Chrome

Figura 2-24: Google Chrome.

Navegador web desarrollado por Google. Usado en este Trabajo Fin de Grado para usarlo como cliente y acceder al sistema domótico OpenHAB.

2.2.8 Android Studio

Figura 2-25: Android Studio.

Entorno de desarrollo oficial para Android [27]. Basado en IntelliJ IDEA de JetBrains. Disponible para Windows, Mac OS y Linux. En este entorno se ha desarrollado la aplicación Android del sistema domótico [28].

2.2.9 PostMan

Figura 2-26: PostMan.

Software que permite el envío de peticiones HTTP REST sin necesidad de desarrollar un cliente [29].

2.2.10 Spring

Figura 2-27: Spring.

Plataforma Java de código abierto. Spring sirve para crear código de alto rendimiento y reutilizable.

2.2.10.1 Maven

Figura 2-28: Maven.

Maven es una herramienta software para la gestión y construcción de proyectos Java basado en XML [30].

2.2.10.2 POM

Figura 2-29: POM.

POM s la unidad fundamental de trabajo en Maven [31]. Contiene los valores predeterminados para la mayoría de los proyectos. Es un archivo XML que contiene información sobre el proyecto y los detalles de configuración utilizados por Maven para construir el proyecto.

2.2.10.3 Spring Security

Figura 2-30: Spring Security.

Spring Security es un framework de autenticación y control de acceso potente y personalizable [32]. Usado para asegurar aplicaciones Spring. En este caso usado para incorporar el certificado autofirmado.

2.2.11 OpenHAB

Figura 2-31: OpenHAB.

OpenHAB es una plataforma de código abierto cuya funcionalidad es la automatización del hogar.

2.2.12 PostgreSQL

Figura 2-32: PostgreSQL.

Sistema de gestión de base de datos relacional orientado a objetos y de código abierto. Almacena las credenciales de usuario del servidor.

2.2.13 STUN Server

Figura 2-33: STUN Server.

Software que funciona como servidor. Implementa el protocolo de red STUN. Es un protocolo de tipo cliente /servidor que permite a clientes con NAT encontrar su dirección IP pública, el tipo de NAT en el que se encuentra y el puerto de Internet asociado con el puerto local a través de NAT. Sirve para configurar una conexión entre dos hosts que se encuentran tras enrutadores NAT.

2.2.14 AWS

Figura 2-34: AWS.

Proveedor de servicios en la nube. Contendrá el servicio REST del Trabajo Fin de Grado.

2.2.15 Swagger

Figura 2-35: Swagger.

Swagger es un framework para documentar APIs REST desde fuentes diferentes [33]. Se puede describir, producir, consumir y visualizar APIs.

3 ESTADO DEL ARTE

En este capítulo se profundizará en la definición, configuración, instalación y uso de OpenHAB y MQTT.

3.1 OpenHAB

Figura 3-1: Logo OpenHAB.

Plataforma de código abierto cuya funcionalidad es la automatización del hogar. OpenHAB tiene la capacidad de incorporar dispositivos y sistemas diferentes en una interfaz sencilla. Esta plataforma además proporciona la inclusión de reglas de automatización las cuales son las fortalezas que componen OpenHAB y hacen que sea una de las herramientas más versátiles en este campo.

Para entender OpenHAB se necesitan entender los componentes básicos del sistema:

- *Binding*: es el componente software que se instala en el sistema que establece la comunicación entre el dispositivo físico y el *Thing*.
- *Things*: representación del dispositivo en el sistema OpenHAB.
- *Items*: representación de la información sobre los dispositivos en el sistema OpenHAB.
- *Channels*: enlace entre un *Thing* y un *Item*. Define como se puede comunicar un *Thing* con un *Item*.
- *Rules*: acciones automáticas.
- *Sitemap*: interfaz del usuario que permite la representación de la información y las interacciones con los dispositivos.

3.1.1 Configuración de OpenHAB

Como se ha definido antes, un Thing representa la entidad física que debe ser gestionada por el sistema. Los things son conectados a OpenHAB a través de bindings. A la hora de añadir un nuevo Thing, primero se debe identificar el Binding que añadirá el tipo del Thing. Un ejemplo sería añadir un elemento Bluetooth, en este caso habría que instalar previamente el Binding Bluetooth.

Los Channels representan las diferentes funciones que puede tener un Thing. Por ejemplo, un sensor de temperatura (DHT11) se compondría de las funciones de temperatura y humedad. Funcionan como vínculo entre los things y los ítems, y sirven como el enlace entre la capa física y la capa de aplicación.

Cada Thing va a proporcionar uno o varios Channels a través de los cuales se conseguirá acceder a la funcionalidad de cada Thing. Y estos a su vez se vinculan a ítems. Los ítems controlan los things y consumen su información. Una vez que todo esté correctamente vinculado, estará disponible para las interfaces de usuario y para crear reglas sobre ellos.

3.1.1.1 Things

En la siguiente tabla se muestran los diferentes estados que puede presentar un Thing para su identificación y clasificación.

ESTADO	DESCRIPCIÓN ESTADO	DETALLE ESTADO	DESCRIPCIÓN DETALLE ESTADO
UNINITIALIZED	Estado inicial, cuando se agrega o se inicia. Si el proceso de inicialización falla o el enlace no está disponible. Los comandos que se envían a los Channels no se procesarán.	NONE	Sin detalle de estado.
		HANDLER_MISSING_ERROR	Enlace no disponible o no iniciado.
		HANDLER_REGISTERING_ERROR	El controlador falló en la fase de registro del servicio.
		HANDLER_INITIALIZING_ERROR	El controlador falló al inicializarse.
		HANDLER_CONFIGURATION_PENDING	El controlador está registrado pero faltan parámetros de configuración para su inicialización.
		BRIDGE_UNINTIALIZED	El puente asociado no está inicializado.
		DISABLED	Deshabilitado.
INITIALIZING	Estado mientras se inicializa. Los comandos que se envían a los Channels no se procesarán.	NONE	Sin detalle de estado.
UNKNOWN	Inicializado pero no se puede determinar si está online u offline.	NONE	Sin detalle de estado.

ONLINE	El dispositivo funciona correctamente y puede procesar comandos.	NONE	Sin detalle de estado.
		CONFIGURATION_PENDING	El Thing está esperando a transferir información de configuración a un dispositivo. El Binding debe asegurarse que se acepta la configuración.
OFFLINE	El dispositivo no funciona correctamente y es posible que no se puedan procesar comandos.	NONE	Sin detalle de estado.
		COMMUNICATION_ERROR	Error al comunicarse con el dispositivo.
		CONFIGURATION_ERROR	Un problema en la configuración impide la comunicación con el dispositivo.
		BRIDGE_OFFLINE	Puente fuera de línea.
		FIRMWARE_UPDATING	El Thing se está actualizando.
		DUTY_CYCLE	El Thing está bloqueado para su uso posterior.
		GONE	El Thing se ha eliminado del puente o la red a la que pertenece y no se encuentra disponible. El usuario puede ahora eliminar el Thing del sistema.
REMOVING	El dispositivo debe eliminarse, pero el Binding no confirmó la eliminación. Los comandos no se pueden procesar.	NONE	Sin detalle de estado.

REMOVED	El dispositivo se eliminó.	NONE	Sin detalle de estado.

Tabla 3-1: Estados de un Thing.

3.1.1.2 Items

En la siguiente tabla se clasifican los diferentes tipos de Items.

NOMBRE ITEM	DESCRIPCIÓN	TIPOS DE COMANDO
Color	Información color (RGB)	OnOff, IncreaseDecrease, Percent, HSB
Contact	Estado de un item.	OpenClosed
DateTime	Guarda la fecha y la hora.	-
Dimmer	Valor porcentual para atenuadores.	OnOff, IncreaseDecrease, Percent
Group	Colección de items	-
Image	Datos binarios de una imagen.	-
Location	Guarda coordenadas GPS.	Point
Number	Valores en formato numérico.	Decimal
Number:<dimension>	Información sobre la dimensión del ítem.	Quantity
Player	Controlador de reproductores.	PlayPause, NextPrevious, RewindFastforward
Rollershutter	Persianas.	UpDown, StopMove, Percent
String	Texto.	String
Switch	Encendido/Apagado. Típicamente usado para luces.	OnOff

Tabla 3-2: Tipos de Items.

3.1.1.3 Sitemaps

Son los utilizados para seleccionar los elementos para componer una presentación al usuario de cara a la configuración de las interfaces de usuario. Los Sitemaps son archivos de texto con extensión ".sitemap".

Los conceptos presentes en la sintaxis de un archivo ".sitemap" se describen como:

- Elementos: componen la interfaz de usuario. Además presentan información y permiten la interacción con los dispositivos.
- Parámetros: se puede configurar un conjunto de parámetros que personalizan un elemento.
- Bloques: encapsulación de elementos y permite anidar uno detrás de otro.
- Dependencias: OpenHAB proporciona parámetros para el comportamiento dinámico, de tal manera que admite estas dependencias y hace posible las interacciones dependientes entre los diferentes elementos del sistema.

3.1.2 Guía de instalación

A continuación, se procederá a explicar los requisitos y pasos necesarios para la instalación.

OpenHAB está escrito en Java y se puede instalar en Linux, Windows, MacOS, OpenHABian, Raspberry Pi, PINE A64, Ambian, Docker, Synology DiskStation y QNAP NAS. En esta guía de instalación se seguirán los pasos para su instalación en una Raspberry Pi Modelo 3B+.

Para instalar en una Raspberry Pi se pueden utilizar dos métodos. El primero es usar una imagen OpenHABian. Mientras que la otra alternativa sería utilizar una imagen de Raspbian (Debian para Raspberry Pi) e instalar OpenHAB siguiendo el método de instalación para Linux.

Como requisito general para ambas opciones es necesario tener instalado la versión Java 8 de la JVM. Para comprobar la versión instalada en una distribución Linux se utiliza el siguiente comando:

```
java -version
```

3.1.2.1 OpenHABian

Es un sistema Linux con los paquetes recomendados de OpenHAB ya incorporados.

Las principales características de OpenHABian son:

- La configuración no necesita de una pantalla o teclado y puede realizarse a través de Wi-Fi o Ethernet.
- Contiene el paquete OpenHAB 2 instalado.
- Contiene Zulu Embedded OpenJDK Java 8 instalado.
- Posee herramienta de configuración de OpenHABian.
- Provisto del sistema de logs OpenHAB Log Viewer.
- Uso compartido de archivos Samba.
- Paquetes útiles Linux preinstalados (vim,mc, htop,…).
- Pantalla de información de inicio de sesión.
- Experiencia personalizada de Shell Bash.
- Configuración de vim personalizada para la sintaxis de OpenHAB.
- Configuración de nano personalizada para la sintaxis de OpenHAB.
- Es específico de Raspberry Pi en cualquiera de sus versiones.

- Preparación:

Se seguirán los siguientes pasos para la instalación de OpenHABian:

1. Descargue la última versión estable OpenHABian de la web oficial de OpenHAB[1].
2. Monte la imagen en la tarjeta SD.
3. Inserte la tarjeta, conecte el cable Ethernet y encienda la Raspberry Pi.
4. A continuación, OpenHABian se configurará automáticamente.
5. El dispositivo estará disponible bajo el nombre DNS local o bajo su IP.
6. Conéctese al panel de control de OpenHAB: http://openHAB:8080
7. Conéctese a los recursos compartidos de la red Samba con nombre de usuario OpenHABian y contraseña OpenHABian.
8. Conéctese al OpenHAB Log Viewer: http://openHAB:9001

Y por fin está listo para su uso.

3.1.2.2 Instalación en Raspbian

Para la instalación de OpenHAB en RaspBian es necesario tener la imagen de Raspbian montada en la tarjeta SD de la Raspberry Pi. En primer lugar es necesario ejecutar todos los pasos siguientes en un terminal o a través de una conexión SSH.

Existen 3 formas de instalar OpenHAB 2:

1. El proyecto OpenHABian con la herramienta de configuración.
2. Con un paquete de repositorio.
3. Desde un archivo de forma manual.

En este Trabajo Fin de Grado se instalará OpenHAB a través de un paquete de repositorio, al ser la opción más recomendada por los desarrolladores OpenHAB. Como Raspbian es un sistema basado en apt, se procederá con este.

En primer lugar se almacena la clave del repositorio.

```
wget -qO - 'https://bintray.com/user/downloadSubjectPublicKey?username=openHAB' | sudo apt-key add –
```

Y se instala el módulo que permitirá a apt utilizar el protocolo HTTPS.

```
sudo apt-get install apt-transport-https
```

Se añade a la lista de fuentes del sistema el repositorio estable de OpenHAB 2. Se debe utilizar el repositorio estable, al ser el repositorio más fiable.

```
echo 'deb https://dl.bintray.com/openHAB/apt-repo2 stable main' | sudo tee /etc/apt/sources.list.d/openHAB2.list
```

Se comprueba si existe alguna actualización.

```
sudo apt-get update
```

Y se instala OpenHAB 2.

```
sudo apt-get install openHAB2
```

Finalmente se completa la instalación con la descarga e instalación de los complementos de OpenHAB.

```
sudo apt-get install openHAB2-addons
```

- Primer inicio en Raspbian:

Si todo ha ido bien, se puede iniciar OpenHAB y hacer que se inicie automáticamente cuando el sistema sea encendido gracias a los siguientes comandos:

```
sudo /etc/init.d/openHAB2 start
sudo /etc/init.d/openHAB2 status
sudo update-rc.d openHAB2 defaults
```

Después de 15 minutos, OpenHAB estará listo. Sólo tardará este tiempo la primera vez que se inicie. Entonces será posible acceder al panel de control del sistema como se observa en la figura 3-2, donde openHAB-device hace alusión a la IP del servidor:

http://openHAB-device:8080

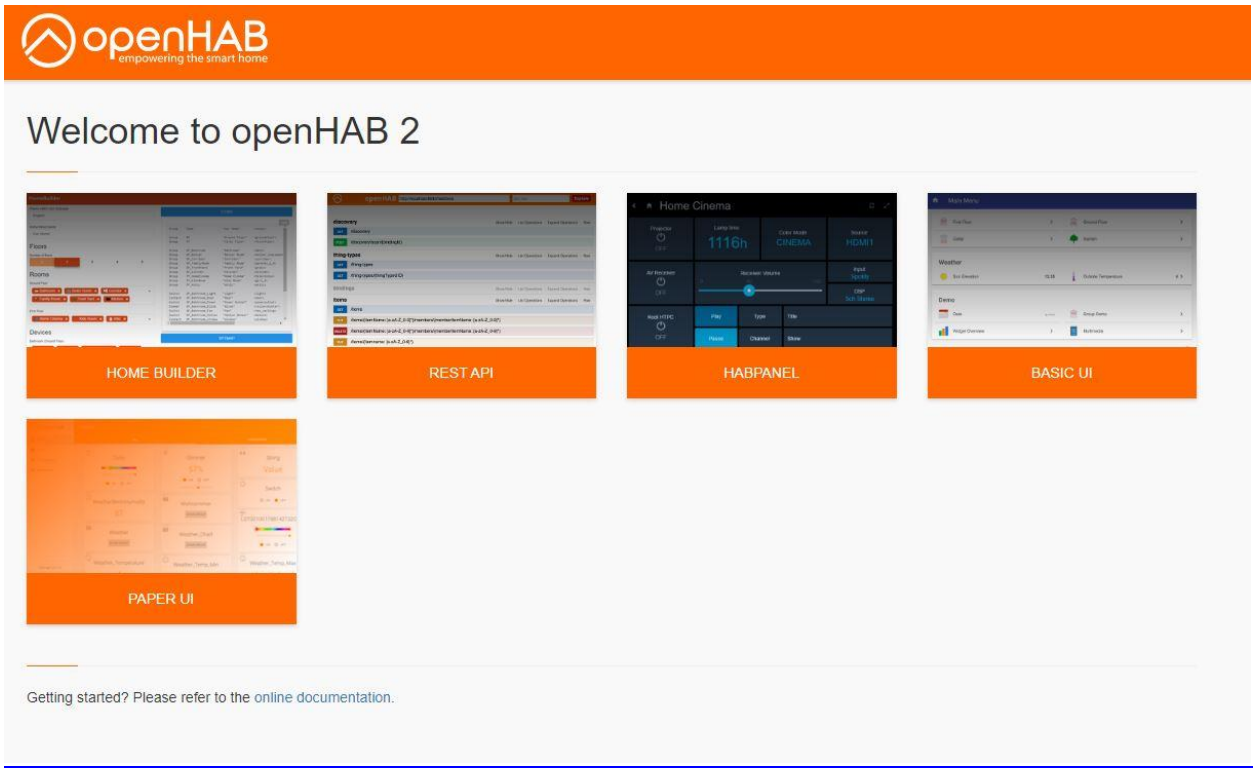


Figura 3-2: Interfaces de OpenHAB.

Comandos de control del sistema:

- Estado:

```
sudo /etc/init.d/openHAB2 status
```

- Encendido:

```
sudo /etc/init.d/openHAB2 start
```

- Reinicio:

```
sudo /etc/init.d/openHAB2 restart
```

- Parada:

```
sudo /etc/init.d/openHAB2 stop
```

Y para desinstalar:

```
sudo apt-get purge openHAB2*
sudo rm /etc/apt/sources.list.d/openHAB2.list
```

Localización de los archivos al instalar el repositorio:

Aplicación OpenHAB	/usr/share/openHAB2
Complementos	/usr/share/openHAB2/addons
Configuración del sitio	/etc/openHAB2
Archivos de registro	/var/log/openHAB2
Datos de usuario (base de datos)	/var/lib/openHAB2
Copias de seguridad	/var/lib/openHAB2/backups
Configuración del servicio	/etc/default/openHAB2

Tabla 3-3: Localización de archivos en el sistema.

3.1.3 Configuración inicial

La primera vez que se inicie OpenHAB, se debe seleccionar uno de los cuatro paquetes: Paquete estándar, paquete experto, paquete simple y paquete de demostración.

Paquete	Recomendado para	Interfaces incluidas
Estándar	Recomendado por OpenHAB para usuarios normales.	- Home Builder. - Paper UI. - Basic UI. - HABPanel.
Experto	Recomendado para usuarios de la versión OpenHAB 1.x.	- Paper UI. - Classic UI. - Basic UI. - HABPanel. - HABmin. - REST API interactiva.
Simple	Recomendado para contener únicamente componentes que permitan un proceso de instalación y configuración controlado totalmente por la interfaz de usuario.	- Paper UI. - Motor de reglas. - HABPanel.
Demostración	Recomendado para fines de demostración y pruebas.	- Paper UI. - Basic UI. - HABPanel. - Enlaces para Yahoo Weather, Belkin WeMo, etc. - Servicio de persistencia RRD4j para almacenamiento temporal. - Servicio de transformación MAP.

Tabla 3-4: Tipos de paquetes de instalación.

3.1.3.1 Tipos de interfaces

- Home Builder:

Home Builder es el contenedor de los ficheros para los ítems, Sitemaps y el panel HABPanel.

Tiene las siguientes características:

- Clasificación de los ítems dentro de cada habitación y crea grupos para ellos.
- Agregación iconos a los ítems.
- Agregación de etiquetas a los ítems.
- Alineación automática de los elementos verticalmente.
- Generación de archivo de Sitemap.
- Generación de paneles de HABPanel correspondiente a los ítems.

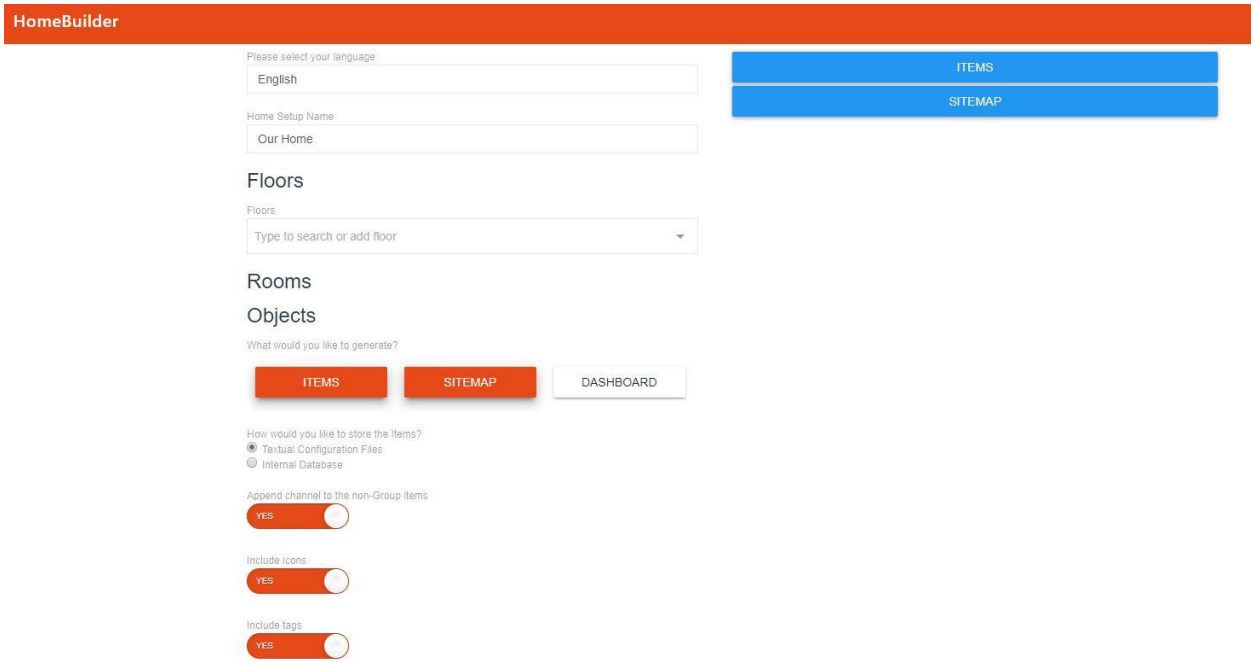


Figura 3-3: Interfaz HomeBuilder.

Uso:

- Nombre de la casa:

Una casa debe ser provista de un nombre que la identifique. La etiqueta para el ítem Home será el nombre del Sitemap. El ítem Home es el ítem raíz de la estructura de la casa, contendrá plantas y grupos de objetos.

- Pisos:

Se selecciona la cantidad de pisos que compone una casa. Cada uno de ellos tendrá su propio grupo. Los pisos contendrán habitaciones y estos a su vez objetos.

- Objetos:

Los objetos son los dispositivos que existen físicamente en una habitación, como es el caso de los sensores. Son representados en Home Builder como Items singulares. Cada objeto será añadido a la lista de ítems con su correspondiente etiqueta, icono, tipo y grupo.

Un ejemplo sería:

Switch Luz_Banio “Light” <light> (Banio,gLight) {channel=””}

Agrupación de objetos.

Group:Switch:OR(ON,OFF) gLight “Light” <light> (Home)

- Paper UI:

Paper UI es la interfaz de usuario que permite configurar la instancia de OpenHAB. Actualmente no funciona para una interfaz completa, pero tiene las siguientes características:

- La administración de complementos sirve para instalar o desinstalar complementos de OpenHAB.
- Función de descubrimiento: es la búsqueda de dispositivos y servicios que se encuentran en la red y su agregación a la configuración.
- Vinculación de elementos a los canales: posibilidad de vincular directamente los canales a los elementos.

Aún es necesario la definición de los Sitemap y los ítems, así como las configuraciones de persistencia y reglas de los archivos de configuración.

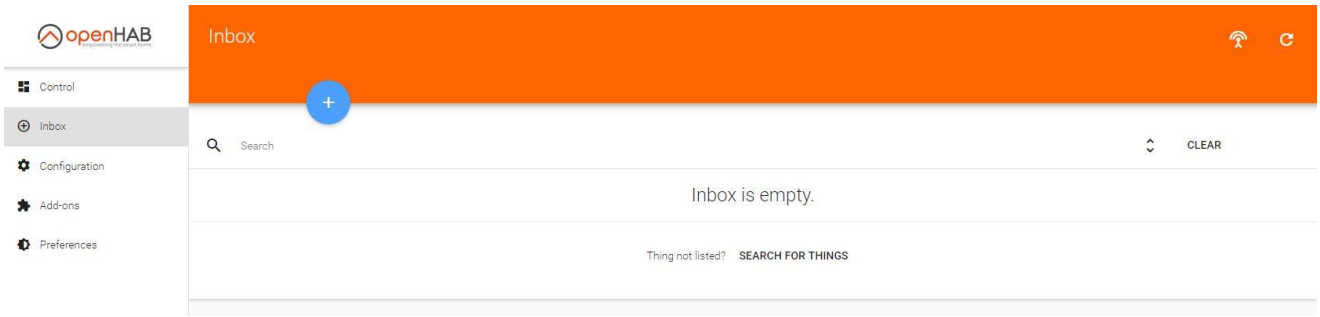


Figura 3-4: Interfaz Paper UI.

- HABmin:

HABmin es la interfaz que proporciona al usuario funciones de usuario y administrativas.

Características:

- Sensible: debe funcionar bien en todos los dispositivos. Aunque la dificultad de su uso es variable.
- Tiene una gran variedad de temas.
- Es trazable ya que posee gráficos modernos e históricos.
- Editor gráfico de reglas.
- Multilingüe.

- HABPanel:

HABPanel es la interfaz que permite la creación de paneles fáciles de usar para el usuario. Estos paneles se pueden diseñar de forma interactiva en lugar de utilizar los archivos de configuración.

Los conceptos de HABPanel son:

- Panel Registry: almacenamiento central de HABPanel en una instancia de OpenHAB dada, pudiendo contener diferentes configuraciones de panel.
- Configuración de panel: contenedor de panel junto con su configuración y la definición de widgets personalizados.

- Panel: Conjunto de paneles que se representan a los usuarios finales, que también pueden cambiar entre ellos mediante el menú.
- Tablero: compuesto por widgets discretos colocados en la superficie del tablero en el momento del diseño.

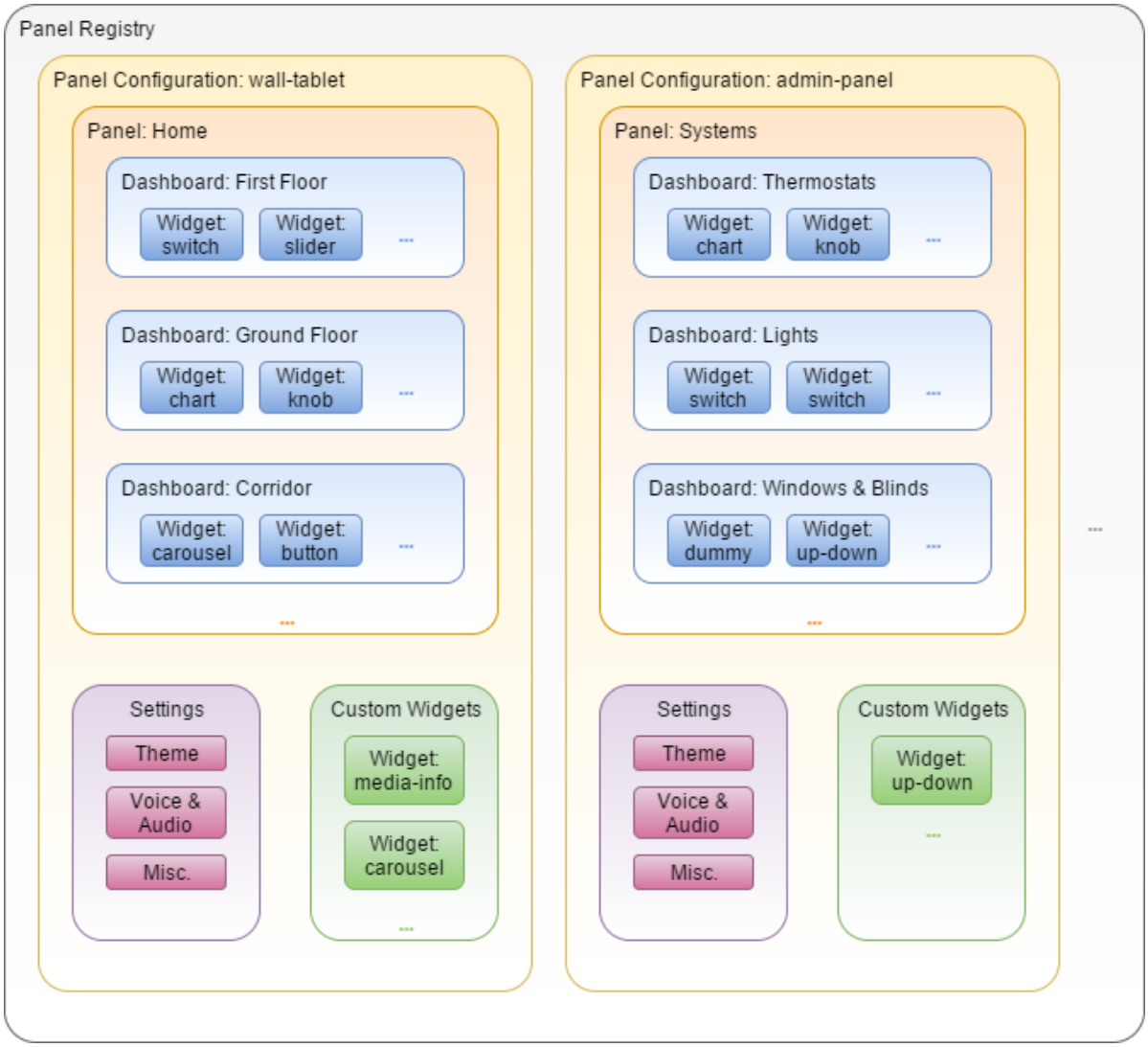


Figura 3-5: Estructura de la interfaz HABPanel.

Persistencia de datos:

Cuando se ejecuta por primera vez HABPanel en un nuevo navegador o dispositivo, se muestra un tutorial para comenzar desde cero o cambiar la configuración de panel. Al principio, al no tener una configuración inicial, se ejecutará en modo almacenamiento local. Dicho almacenamiento se mantendrá únicamente en el almacenamiento local del navegador y no se mantendrá nada en el servidor. Al establecer una configuración de panel que esté activa, este cambio actualizará la configuración de panel del servidor, permitiendo así compartir dicha configuración con el resto de los dispositivos y navegadores.

Los cambios en el modo de almacenamiento se realizan en la página de configuración desde el menú principal o el panel lateral, donde se presenta una lista de configuraciones de panel en la sección *Configuración de almacenamiento actual.* Recordando que si solo aparece la opción *Almacenamiento local* se debe a que no se ha guardado una primera configuración de panel.

Utilizando Paper UI se puede acceder a las diferentes variables de configuración de servicio que usa HABPanel para almacenar sus datos en el servidor OpenHAB. Además se pueden definir las siguientes propiedades:

- panelsRegistry: contiene todo el registro en formato JSON.
- lockEditing: al habilitarse, todas las instancias de HABPanel ocultan sus funciones de edición. Esta función es aconsejable para que no se permita a los usuarios finales modificarlos fácilmente.
- initialPanelConfig: si no existe una configuración local y no está configurada esta propiedad, se mostrará el tutorial hasta que cambie la configuración del panel. Esta propiedad permite omitir el tutorial y cambiar directamente la configuración.

Principales características de la interfaz:

- Menú principal: Página de inicio de HABPanel.
- Cajón lateral:
    - Encabezado.
    - Lista de paneles para acceso rápido.
    - Pie de página con fecha y hora actuales.
- Diseñador del tablero: constructor del tablero donde se pueden posicionar, configurar, redimensionar y agregar widgets.
- Ejecución de paneles: los widgets pueden interactuar y los eventos enviados por el servidor se reciben al actualizar los estados de los diferentes elementos, por lo que el HABPanel se actualiza automáticamente.

Características y configuraciones adicionales:

- Nombre de panel
- Tema
- Imagen de fondo
- Hoja de estilo adicional
- Imagen de encabezado
- Ocultar el pie de página
- Ocultar botones de la barra de herramientas
- Mostrar un reloj en el encabezado
- Formato del reloj del encabezado
- Prevenir el desplazamiento
- Administrar widgets
- Voz
- Lectura en voz alta de nuevo estado
- Botón de voz flotante en la parte inferior de la pantalla
- Actualización de nuevo estado

Widgets:

- Widgets estándar:
  - Duumy: posee el estado del widget, una etiqueta y un icono opcional.
  - Switch: informa de su estado y puede cambiarlo entre ON y OFF.
  - Label: muestra un texto y tiene opciones de color y fuente.
  - Button: al hacer clic o tocar el botón se realiza la acción asociada al botón.
  - Slider: barra deslizante que representa el estado de elementos numéricos y actualiza los valores dentro de un rango predefinido.
  - Knob: similar al "slider" pero de forma rotativa. Se pueden alterar tanto su apariencia como su comportamiento.
  - Selection: muestra el estado actual de un elemento, abre un menú de opciones configuradas para enviar comandos a este elemento.
  - Color picker: selección de color.
  - Image: muestra una imagen y puede actualizarla a intervalos regulares.
  - Frame: muestra una página web externa en un HTML.
  - Clock: muestra un reloj analógico o digital. También puede mostrar la fecha actual.
  - Chart: el uso de los servicios de persistencia de OpenHAB para trazar gráficas. También puede mostrar imágenes de gráficos generados por el servidor.
  - Timeline: presenta cambios de estado a lo largo del tiempo.
  - Template: permite que una plantilla HTML AngularJS configurada por el usuario sea renderizada y alojada.
- Custom widgets: plantilla AngularJS para crear un widget personalizado, el cual se puede reutilizar y compartir.

➢ Basic UI:

Basic UI es la interfaz de usuario básica, la cual está diseñada en Material Design Lite de Google.

Posee las siguientes características:

- Diseño adaptable según el tamaño de la pantalla.
- Navegación AJAX
- Actualización en vivo.

Posee un diseño que muestra todos los elementos correspondientes. Se pueden crear Sitemaps y acceder a ellos de dos formas diferentes:

1. Configuration / Services / Basic UI / Configure, y establecer el nombre del Sitemap determinado.
2. Pasar el parámetro "sitemap" a la URL para acceder:

   http://hostname:8080/basicui/app?sitemap=sitemapname

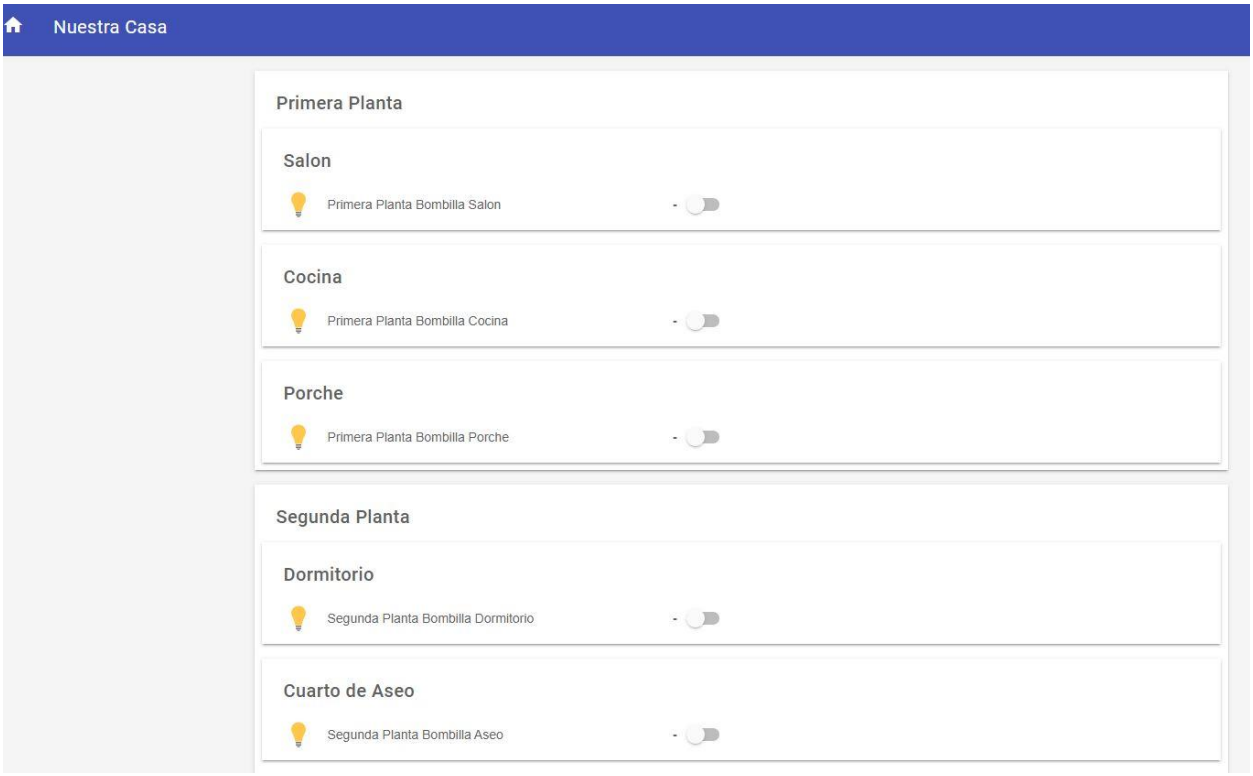


Figura 3-6: Interfaz Basic UI.

➢ Classic UI:

Interfaz de usuario clásica original de OpenHAB 1. Es la interfaz más estable y la más utilizada, aunque su aspecto no presenta una estética moderna. El acceso a los sitemaps es el mismo que el explicado en Basic UI.

http://hostname:8080/classicui/app?sitemap=sitemapname

- Motor de reglas:

Permite editar las reglas de forma gráfica e interactuar con los scripts JSR223(JavaScript, Jython, etc.)

- Instalación:

Add-ons / Misc / Rule Engine (Experimental) pulsar en instalar. Al refrescar el navegador, aparecerá Rules en el menú principal.

Una regla consta de un nombre, etiquetas y una descripción como información básica. Aunque los componentes principales son módulos. Estos módulos pueden instanciarse en cada regla y son:

- 'Trigger': especifican los eventos que activan la ejecución de una regla.
- 'Condition': actúa como filtro para la ejecución de reglas. Se ejecutará sólo si se satisfacen todas las condiciones.
- 'Action': realizan operaciones en OpenHAB. Si se especifica más de una regla se ejecutan de forma secuencial. También la salida de la acción anterior se puede usar como entrada de la siguiente acción.

Cada módulo se crea a través de una plantilla llamada Module type, el cual especifica los parámetros para la plantilla. Existen algunos tipos de módulos genéricos que usan configuraciones predefinidas.

La plantilla Module type está formada por:

- uid: id único.
- label: etiqueta de texto.
- description: texto de información.
- configDescriptions: lista de metadatos de las propiedades de configuración.
    - name
    - type: “text”, “integer”, “decimal”, “boolean”.
    - label: etiqueta.
    - description: información.
    - required: bandera que indica si la propiedad de configuración es opcional. Valor por defecto: false.
    - defaultValue: valor por defecto de la propiedad de configuración cuando no está especificado en la regla.
- inputs variables: lista de metadatos de objetos de entrada.
    - name
    - type: nombre de la clase java. (“java.lang.Integer”).
    - label: etiqueta.
    - description: información.
    - defaultValue: : valor por defecto de la propiedad de configuración cuando no está especificado en la regla.
    - tags: especifica cómo debe ser considerado un valor dado. Por ejemplo la fecha.
- outputs variables: lista de metadatos de objetos de salida.
    - name
    - type:
    - label: etiqueta.
    - description: información.
    - defaultValue: : valor por defecto de la propiedad de configuración cuando no está especificado en la regla.
    - reference: el tipo del valor de la propiedad se especifica al parámetro de entrada o configuración.
    - tags: especifica cómo debe ser considerado un valor dado. Por ejemplo de la fecha.

Los tipos admitidos en los objetos entrada/salida pueden ser cualquier cadena y se necesita la siguiente validación:

- Si el tipo de entrada y el tipo de salida son iguales a la cadena, la conexión es válida.
- Si el tipo de entrada es “*” y el tipo de salida es cualquiera, la conexión es válida.
- Si el tipo de entrada y el tipo de salida son cadenas que representan nombres completos que se pueden cargar y el tipo de entrada es asignable desde el tipo de salida, la conexión es válida.

Los tipos del objeto Configuration son:

- TEXT: valor de texto UTF-8.
- INTEGER: valor entero con signo en el rango de Integer#MIN_VALUE, Integer#MAX_VALUE.
- DECIMAL: valor flotante, en el rango de Float#MIN_VALUE, Float#MAX_VALUE.
- BOOLEAN: booleano: verdadero o falso.

- API REST:

Se puede acceder a la mayoría de los datos de OpenHAB desde otros programas gracias a la API REST que proporciona OpenHAB. En concreto, permite el acceso a los datos como 'things'. 'items' y 'bindings', así como invocar acciones que pueden cambiar el estado de los things o influir en el comportamiento de otros elementos de OpenHAB. El acceso a través de Internet a la API REST es posible pero representa un problema de seguridad.

Es posible acceder a la documentación de la API REST a través del panel de control de OpenHAB, una vez que sea instalada como una interfaz de usuario personalizada. Mostrará todos los comandos disponibles.

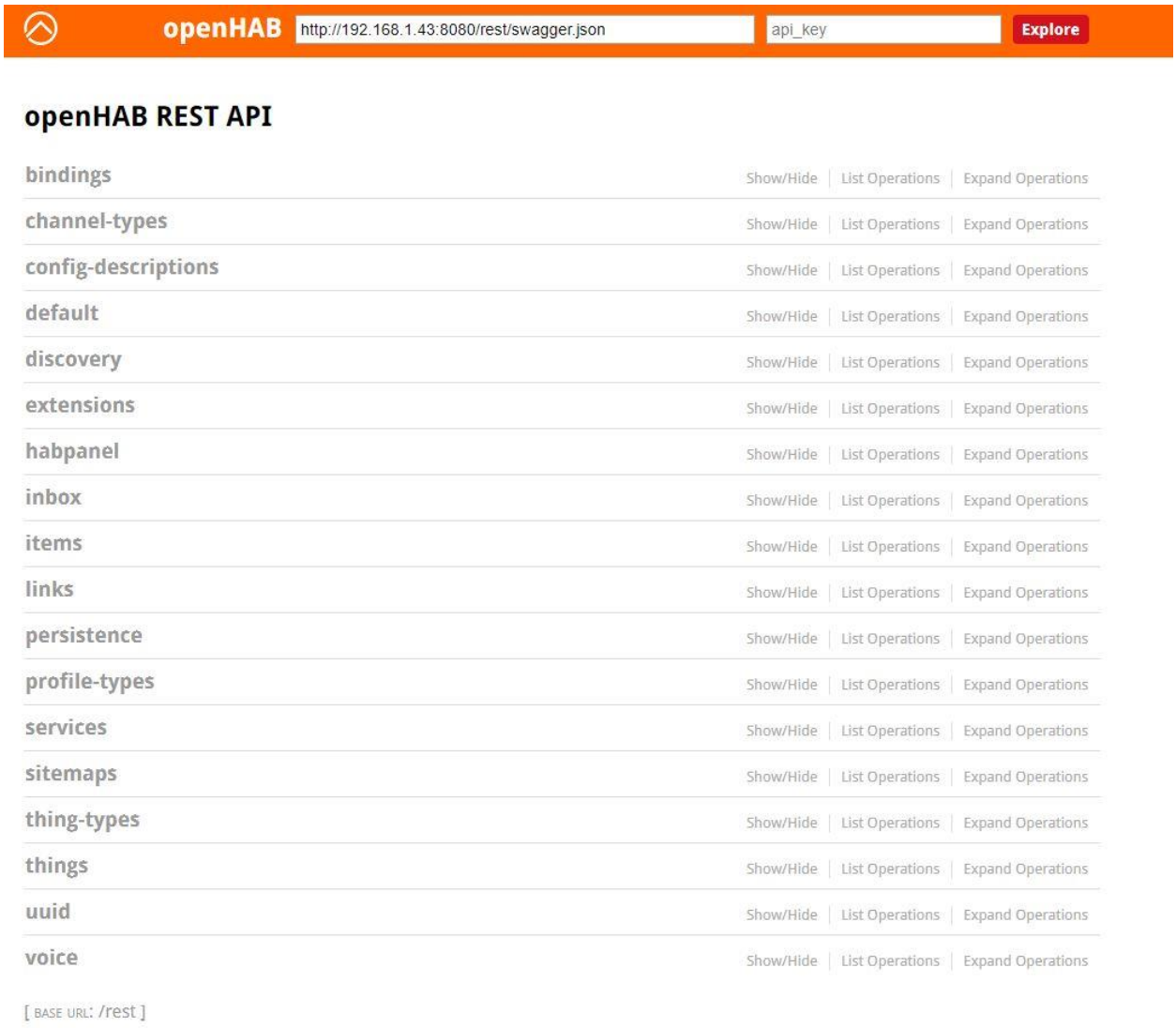


Figura 3-7: Interfaz API REST.

3.2 MQTT

Figura 3-8: Logo MQTT.

Message Queue Telemetry Transport es un protocolo de transporte de mensajería basado en el modelo cliente-servidor publicador-subscriptor. Su fácil implementación, su diseño y que sea abierto hace que sea ideal para su uso en entornos M2M y contextos IoT. Este protocolo funciona sobre TCP/IP del modelo OSI. Fue creado en 1999 por dos ingenieros de IBM y ARCON, con el fin de interconectar sensores con un consumo muy bajo.

Entre sus características se encuentran:

- Uso del patrón publicador-subscriptor.
- Transporte del mensaje con el desconocimiento del contenido de este.
- La calidad de servicio consta de tres tipos: Como mucho una vez, al menos una vez, exactamente una vez.
- Sobrecarga de red e intercambio de mensajes minimizados para reducir el tráfico de red.
- Si se produce una desconexión anormal, existe un mecanismo para notificar a las partes interesadas cuando se produce.

3.2.1 Terminología

Concepto	Definición y descripción
Conexión de red	Construcción dada por el protocolo de transporte que utiliza MQTT. Conecta el cliente al servidor y proporciona los medios para enviar una secuencia ordenada sin pérdidas en ambas direcciones.
Mensaje de aplicación	Son los datos transportados por el protocolo MQTT a través de la red para la aplicación. Cuando un mensaje de aplicación es transportado por MQTT, contiene datos de carga útil, una determinada calidad de servicio, unas propiedades y un topic.

Cliente	Es el programa usado por MQTT. - abre la conexión de red con el servidor. - publica mensajes de aplicación para otro cliente interesado en recibirlos. - se suscribe para recibir los mensajes de aplicación que está interesado en recibir. - cancela la suscripción para no recibir los mensajes de aplicación. - cerrar la conexión de red con el servidor.
Servidor	Es el programa intermediario entre los clientes publicadores y los clientes suscriptores. - acepta la conexión de red desde los clientes. - acepta los mensajes de aplicación publicados por los clientes. - procesa peticiones de suscripción y de cancelación de suscripción de los clientes. - reenvía los mensajes de aplicación a los clientes suscritos. - cierra la conexión de red desde los clientes.
Sesión	Interacción entre el cliente y un servidor, la cual se define por un estado.
Suscripción	Asociada a una sesión. Se compone de un topic y una calidad de servicio.
Suscripción compartida	Suscripción asociada a más de una sesión.
Suscripción de comodines	Suscripción a un topic que coincide con varios topics a la vez.
Nombre de topic	La etiqueta asociada a un mensaje de aplicación la cual coincide con la suscripción en el servidor.
Filtro de topic	Expresión contenida en la suscripción para indicar el interés en uno o varios topics.
Control de paquete MQTT	Un paquete de información que es enviado por la red.
Paquete malformado	Paquete de control que no se puede analizar de acuerdo con la especificación.
Error de protocolo	Un error es detectado al analizar el paquete y encontrar que contiene datos no permitidos por el protocolo o es inconsistente con el estado del cliente o el servidor.
Will mensaje	Un mensaje de aplicación que es publicado por el servidor después del cierre de la conexión de red cuando no se ha cerrado con normalidad.
Código Unicode no permitido	El conjunto de códigos de control Unicode.

Tabla 3-5: Terminología del sistema MQTT.

3.2.2 Paquete de control MQTT

La estructura del paquete de control de MQTT está formado por tres campos:

Cabecera fija (presente en todos los paquetes de control MQTT)
Cabecera variable (presente en algunos paquetes de control MQTT)
Carga de paquete (presente en algunos paquetes de control MQTT)

Tabla 3-6: Estructura del paquete de control.

3.2.2.1 Cabecera fija de un paquete de control MQTT

La cabecera fija está formada por siete bits y su estructura es la siguiente:

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	Tipo de paquete de control MQTT				Bandera específica para cada tipo de paquete de control MQTT			
2	Longitud restante							

Tabla 3-7: Estructura de cabecera fija de un paquete de control.

- Tipo de paquetes de control MQTT:

Nombre	**Valor**	**Dirección del flujo**	**Descripción**
Reservado	0	Prohibido	Reservado
CONNECT	1	Cliente->Servidor	Petición de conexión
CONNACK	2	Servidor->Cliente	Confirmación conexión
PUBLISH	3	Cliente->Servidor O Servidor->Cliente	Mensaje publicado
PUBACK	4	Cliente->Servidor O Servidor->Cliente	Confirmación de mensaje publicado
PUBREC	5	Cliente->Servidor O Servidor->Cliente	Mensaje publicado recibido
PUBREL	6	Cliente->Servidor O Servidor->Cliente	Confirmación mensaje publicado recibido

PUBCOMP	7	Cliente->Servidor O Servidor->Cliente	Publicación completa
SUBSCRIBE	8	Cliente->Servidor	Petición de suscripción
SUBACK	9	Servidor->Cliente	Confirmación de petición de suscripción
UNSUBSCRIBE	10	Cliente->Servidor	Petición de cancelación de suscripción
UNSUBACK	11	Servidor->Cliente	Confirmación de petición de cancelación de suscripción
PINGREQ	12	Cliente->Servidor	Petición PING
PINGRESP	13	Servidor->Cliente	Respuesta PING
DISCONNECT	14	Cliente->Servidor O Servidor->Cliente	Notificación de desconexión
AUTH	15	Cliente->Servidor O Servidor->Cliente	Intercambio de autenticación

Tabla 3-8: Tipos de paquetes de control de MQTT.

- Bits bandera:

Paquetes de control MQTT	**Bandera de cabecera fija**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
CONNECT	Reservado	0	0	0	0
CONNACK	Reservado	0	0	0	0
PUBLISH	Usado en MQTT v5.0	DUP	QoS		RETAIN
PUBACK	Reservado	0	0	0	0
PUBREC	Reservado	0	0	0	0
PUBREL	Reservado	0	0	1	0
PUBCOMP	Reservado	0	0	0	0
SUBSCRIBE	Reservado	0	0	1	0

SUBACK	Reservado	0	0	0	0
UNSUBSCRIBE	Reservado	0	0	1	0
UNSUBACK	Reservado	0	0	0	0
PINGREQ	Reservado	0	0	0	0
PINGRESP	Reservado	0	0	0	0
DISCONNECT	Reservado	0	0	0	0
AUTH	Reservado	0	0	0	0

Tabla 3-9: Bits bandera según paquetes de control de MQTT.

DUP: entrega duplicada de un paquete PUBLISH.

QoS: Calidad de servicio de un paquete PUBLISH.

- QoS 0: entrega como mucho una vez.
- QoS 1: entrega al menos una vez.
- QoS 2: entrega exactamente una vez.

RETAIN: Bandera de un mensaje retenido PUBLISH.

- Longitud restante:

La longitud restante representa el número de bytes restantes dentro del paquete de control, incluidos los datos de la cabecera variable y la carga útil. No incluye los bytes utilizados para codificar la longitud restante. El tamaño de paquete es el número total de bytes en un paquete de control MQTT, que es igual a la longitud de la cabecera fija más la longitud restante.

3.2.2.2 Cabecera variable

Algunos paquetes de control MQTT contienen una cabecera variable. Es común que esté compuesto por el campo de identificador de paquete y un campo de propiedades.

Los paquetes que requieren un identificador de paquete están incluidos en la siguiente tabla.

Paquete de control MQTT	Campo de identificador de paquete
CONNECT	NO
CONNACK	NO
PUBLISH	SI (si QoS > 0)
PUBACK	SI
PUBREC	SI
PUBREL	SI
PUBCOMP	SI

SUBSCRIBE	SI
SUBACK	SI
UNSUBSCRIBE	SI
UNSUBACK	SI
PINGREQ	NO
PINGRESP	NO
DISCONNECT	NO
AUTH	NO

Tabla 3-10: Campo identificador de paquete del paquete de control.

- Propiedades:

El campo propiedades es el último campo en la cabecera variable de los paquetes CONNECT, CONNACK, PUBLISH, PUBACK, PUBREC, PUBREL, PUBCOMP, SUBSCRIBE, SUBACK, UNSUBSCRIBE, UNSUBACK, DISCONNECT y AUTH. Las propiedades se componen de un campo de longitud y a continuación se exponen las propiedades.

Identificador		**Nombre (uso)**	**Tipo**	**Paquetes / Propiedades Will**
Dec	**Hex**			
1	0x01	Indicador de formato de la carga útil	Byte	PUBLISH, Propiedades Will
2	0x02	Intervalo de expiración del mensaje	Entero de 4 bytes	PUBLISH, Propiedades Will
3	0x03	Tipo de contenido	Cadena codificada UTF-8	PUBLISH, Propiedades Will
8	0x08	Topic de respuesta	Cadena codificada UTF-8	PUBLISH, Propiedades Will
9	0x09	Datos de correlación	Datos binarios	PUBLISH, Propiedades Will
11	0x0B	Identificador de suscripción	Entero de byte variable	PUBLISH, SUBSCRIBE
17	0x11	Intervalo de expiración de sesión	Entero de 4 bytes	CONNECT, CONNACK, DISCONNECT

18	0x12	Identificador de cliente asignado	Cadena codificada UTF-8	CONNACK
19	0x13	Servidor persistente	Entero de 2 bytes	CONNACK
21	0x15	Método de autenticación	Cadena codificada UTF-8	CONNECT, CONNACK, AUTH
22	0x16	Datos de autenticación	Datos binarios	CONNECT, CONNACK, AUTH
23	0x17	Solicitar información de respuesta	Byte	CONNECT
24	0x18	Intervalo de retraso Will	Entero de 4 bytes	Propiedades Will
25	0x19	Solicitar información de respuesta	Byte	CONNECT
26	0x1A	Información de respuesta	Cadena codificada UTF-8	CONNACK
28	0x1C	Referencia de servidor	Cadena codificada UTF-8	CONNACK, DISCONNECT
31	0x1F	Cadena de razón	Cadena codificada UTF-8	CONNACK, PUBACK, PUBREC, PUBREL, PUBCOMP, SUBACK, UNSUBACK, DISCONNECT, AUTH
33	0x21	Recibir el máximo	Entero de 2 bytes	CONNECT, CONNACK
34	0x22	Máximo alias topic	Entero de 2 bytes	CONNECT, CONNACK
35	0x23	Alias topic	Entero de 2 bytes	PUBLISH
36	0x24	Máxima QoS	Byte	CONNACK
37	0x25	Conservar	Byte	CONNACK
38	0x26	Propiedades de usuario	Cadena codificada UTF-8	CONNECT, CONNACK, PUBLISH, Propiedades Will, PUBACK, PUBREC, PUBREL, PUBCOMP, SUBSCRIBE, SUBACK,

				UNSUBSCRIBE, UNSUBACK, DISCONNECT, AUTH
39	0x27	Máximo tamaño de paquete	Entero de 4 bytes	CONNECT, CONNACK
40	0x28	Suscripción comodina disponible	Byte	CONNACK
41	0x29	Identificador de suscripción disponible	Byte	CONNACK
42	0x2A	Suscripción compartida disponible	Byte	CONNACK

Tabla 3-11: Campo de propiedades de la cabecera variable.

3.2.2.3 Carga útil

La carga útil se refiere a la parte final de un paquete y solo algunos paquetes de control MQTT lo contienen.

Paquete de control MQTT	Carga útil
CONNECT	Requerido
CONNACK	Vacío
PUBLISH	Opcional
PUBACK	Vacío
PUBREC	Vacío
PUBREL	Vacío
PUBCOMP	Vacío
SUBSCRIBE	Requerido
SUBACK	Requerido
UNSUBSCRIBE	Requerido
UNSUBACK	Requerido
PINGREQ	Vacío
PINGRESP	Vacío
DISCONNECT	Vacío
AUTH	Vacío

Tabla 3-12: Carga útil según el paquete.

3.2.3 Tipos de paquete

En este apartado se especificarán los distintos tipos de paquete de MQTT.

3.2.3.1 CONNECT

Después de establecer la conexión de red por un cliente al servidor, el primer paquete enviado del cliente al servidor debe ser el paquete CONNECT.

Un cliente puede enviar solo un CONNECT más aparte del primero. El servidor debe procesar el segundo CONNECT enviado desde un cliente como un error de protocolo y cerrar la conexión de red.

La carga útil contiene uno o más campos codificados. Especifican un único identificador de cliente, un topic, un nombre de usuario y una contraseña. El cliente puede omitir alguno de estos campos, excepto el identificador de cliente.

3.2.3.2 CONNACK

CONNACK es el paquete enviado por el servidor a un cliente en respuesta de un paquete CONNECT. El servidor debe enviar un CNNACK con el código razón 0x00 antes de enviar algún paquete distinto a AUTH. El servidor no debe enviar más de un CONNACK en una conexión de red.

Si el cliente no recibe el paquete CONNACK durante una razonable cantidad de tiempo, el cliente debería cerrar la conexión de red. La razonable cantidad de tiempo depende del tipo de aplicación.

3.2.3.3 PUBLISH

PUBLISH es el paquete enviado desde un cliente al servidor o desde el servidor al cliente para transportar un mensaje de aplicación.

3.2.3.4 PUBACK

PUBACK es el paquete respuesta al paquete PUBLISH con QoS 1.

3.2.3.5 PUBREC

PUBREC es la respuesta a un paquete PUBLISH con QoS 2. Es el segundo paquete del intercambio de protocolos con QoS 2.

3.2.3.6 PUBREL

PUBREL es el paquete de respuesta a un paquete PUBREC. Es el tercer paquete del intercambio de protocolos con QoS 2.

3.2.3.7 PUBCOMP

PUBCOMP es la respuesta al paquete PUBREL. Es el cuarto y último paquete del intercambio de protocolos con QoS 2.

3.2.3.8 SUBSCRIBE

SUBSCRIBE es el paquete enviado desde el cliente al servidor para crear una o más suscripciones. Cada suscripción registra el interés de un cliente en uno más topics. El servidor envía el paquete PUBLISH al cliente para reenviar mensajes de aplicación que se publicaron en los topics que coinciden con estas suscripciones. El paquete SUBSCRIBE también especifica la máxima calidad de servicio con la que el servidor puede enviar un mensaje de aplicación al cliente.

3.2.3.9 SUBACK

SUBACK es el enviado al servidor por el cliente para confirmar la recepción y el procesamiento del paquete SUBSCRIBE.

El paquete SUBACK contiene una lista de códigos de razón, los cuales especifican el máximo nivel de QoS que es garantizado o el error encontrado para cada suscripción que son respondidas por el paquete SUBSCRIBE.

3.2.3.10 UNSUBSCRIBE

UNSUBSCRIBE es el enviado por el cliente al servidor para cancelar una suscripción a un topic.

3.2.3.11 UNSUBACK

El paquete es enviado por el servidor al cliente para confirmar la recepción de un paquete UNSUBSCRIBE.

3.2.3.12 PINREQ

PINGREQ es el paquete enviado por el cliente al servidor y puede ser usado para:

- Indicar al servidor que el cliente está activo ante la ausencia de algún otro paquete de control MQTT.
- Solicitar al servidor que responda para confirmar que está activo.
- Ejercitar la red para indicar que la conexión de red está activa.

3.2.3.13 PINGRESP

PINGRESP es el paquete enviado por el servidor al cliente en respuesta del paquete PINGREQ. Indica que el servidor sigue activo.

3.2.3.14 DISCONNECT

DISCONNECT es el paquete final de los paquetes de control MQTT. Enviado del cliente al servidor. Indica la razón porque la conexión de red está siendo cerrada. El cliente o el servidor debe mandar el paquete DISCONNECT antes de cerrar la conexión de red. Si la conexión de red es cerrada sin que el cliente primero envíe un paquete DISCONNECT con código de razón 0x00 y la conexión tiene un mensaje Will, el mensaje Will es publicado.

El servidor no debe enviar un DISCONNECT hasta que haya enviado un paquete CONNACK con código de razón inferior a 0x80.

3.2.3.15 AUTH

El paquete AUTH es enviado del cliente al servidor o viceversa como parte de un intercambio de autenticación extendido, como una autenticación de desafío/respuesta. Es un error de protocolo para el cliente o servidor que se envíe un paquete AUTH si el paquete CONNECT no contiene el mismo método de autenticación.

4 DESCRIPCIÓN DE LA SOLUCIÓN DESARROLLADA

La innovación es lo que distingue a un líder de un seguidor

Steve Jobs

La solución que se ha llevado a cabo consta de cuatro partes o componentes bien diferenciados. La placa NodeMCU la cual posee los sensores del hogar y que actúa como receptor de los cambios de estado y publicador de los estados actuales de los sensores. La placa Raspberry Pi donde se ha implementado OpenHAB y el cliente del protocolo STUN. El servidor en la nube de Amazon Web Services en el que está alojado el servicio de autenticación de credenciales y el servidor del protocolo STUN. Y en último lugar la aplicación móvil que funcionará como interfaz de cara al sistema para la interacción con los diferentes sensores del hogar.

El orden ha sido especificado así en esta memoria para la continuación de un hilo conductor basado en la necesidad de implementar todos y cada uno de los componentes. En el apartado final de este capítulo se explica cómo se realiza la comunicación entre los componentes.

4.1 Primer paso: Instalación de medios inteligentes en el hogar

¿Cuál es la finalidad de este Trabajo Fin de Grado? La finalidad es poder domotizar un hogar de forma económica y efectiva. Para ello una de las mejores maneras para domotizar una casa es agregar dispositivos a los elementos que ya se poseen para que se pueda interactuar con ellos desde donde se quiera.

Se opta entonces por una placa que pueda enviar y recibir información y que sea económica, además entre sus funcionalidades debe tener tecnología Wi-Fi. Por eso se utiliza la placa NodeMCU. Una placa a la que se puede conectar cualquier sensor Arduino, realizando así cualquier medición del entorno o poder controlarlos.

4.1.1 Placa NodeMCU

La placa NodeMCU debe conectarse a una protoboard en medio del canal de forma que se tenga acceso a todos los pines. A través de la entrada Micro-USB se subirá el código del programa que ejecutará la placa. En primer lugar se configura para que se conecte a la red del hogar y sea accesible. En segundo lugar se hace uso del protocolo MQTT para la comunicación con la placa y poder así gestionar los sensores conectados a ella. Los sensores utilizados en este ejemplo son leds excepto un relé instalado en el Salón que es una muestra real del dispositivo a instalar en cualquier enchufe o interruptor del hogar.

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

#define bsal 13     // Especifica el pin al que se conecta el relé (GPIO0)
#define ledpor 5
#define ledter 4
#define ledaseo 2
#define ledcoc 14
#define leddor 12

#define wifi_ssid "MOVISTAR_10APROB"
#define wifi_password "***********"

#define mqtt_server "192.168.1.43"
#define mqtt_port 1883

#define bsalon "PrimeraPlanta/BombillaSalon"    //Cambia el nombre del topic a tu gusto
#define bcocina "PrimeraPlanta/BombillaCocina"
#define bporche "PrimeraPlanta/BombillaPorche"

#define bterraza "SegundaPlanta/BombillaTerraza"
#define bdormitorio "SegundaPlanta/BombillaDormitorio"
#define baseo "SegundaPlanta/BombillaAseo"

WiFiClient espClient;
PubSubClient client;

void setup() {
    Serial.begin(115200);
    setup_wifi();
    client.setClient(espClient);
    client.setServer(mqtt_server, mqtt_port);
    client.setCallback(callback);
    pinMode(bsal, OUTPUT);
    digitalWrite(bsal, LOW);
    pinMode(ledpor, OUTPUT);
    digitalWrite(ledpor, LOW);
    pinMode(ledter, OUTPUT);
    digitalWrite(ledter, LOW);
    pinMode(ledaseo, OUTPUT);
    digitalWrite(ledaseo, LOW);
    pinMode(ledcoc, OUTPUT);
    digitalWrite(ledcoc, LOW);
    pinMode(leddor, OUTPUT);
    digitalWrite(leddor, LOW);
```

```
}

void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(wifi_ssid);

    WiFi.begin(wifi_ssid, wifi_password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

void reconnect() {
    // Bucle hasta conseguir la reconexión
    while (!client.connected()) {
        // Intento de conexión
        Serial.print("Attempting MQTT connection...");

        // Si utilizas usuario y contraseña, cambia la siguiente línea por esta
        // if (client.connect("ESP8266Client", mqtt_user, mqtt_password)) {
        if (client.connect("ESP8266Client")) {
            Serial.println("connected");
            client.subscribe(bsalon);
            client.subscribe(bcocina);
            client.subscribe(bporche);
            client.subscribe(bterraza);
            client.subscribe(bdormitorio);
            client.subscribe(baseo);
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");

            // Espera 5 segundos para un nuevo intento
            delay(5000);
        }
    }
}

bool checkBound(float newValue, float prevValue, float maxDiff) {
```

```
    return newValue < prevValue - maxDiff || newValue > prevValue + maxDiff;
}

void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Mensaje recibido [");
    Serial.print(topic);
    String str = topic;
    Serial.print(topic);
    Serial.print("] ");
    String(receivedChar) = "";
    for (int i = 0; i < length; i++) {
        receivedChar += (char)payload[i];
    }
   if (str == bsalon) {
    if (receivedChar == "ON"){
        digitalWrite(bsal, LOW);
        Serial.println(receivedChar);
    }
    if (receivedChar == "OFF"){
        digitalWrite(bsal, HIGH);
        Serial.println(receivedChar);
    }
  }
    // statements
  if(str == bcocina) {
  if (receivedChar == "ON"){
        digitalWrite(ledcoc, HIGH);
        Serial.println(receivedChar);
    }
    if (receivedChar == "OFF"){
        digitalWrite(ledcoc, LOW);
        Serial.println(receivedChar);
    }
  }
    // statements
  if (str == bporche) {
    if (receivedChar == "ON"){
        digitalWrite(ledpor, HIGH);
        Serial.println(receivedChar);
    }
    if (receivedChar == "OFF"){
        digitalWrite(ledpor, LOW);
        Serial.println(receivedChar);
    }
  }
  if (str == bdormitorio) {
    if (receivedChar == "ON"){
```

```
            digitalWrite(leddor, HIGH);
            Serial.println(receivedChar);
        }
        if (receivedChar == "OFF"){
            digitalWrite(leddor, LOW);
            Serial.println(receivedChar);
        }
      }
      if (str == baseo) {
        if (receivedChar == "ON"){
            digitalWrite(ledaseo, HIGH);
            Serial.println(receivedChar);
        }
        if (receivedChar == "OFF"){
            digitalWrite(ledaseo, LOW);
            Serial.println(receivedChar);
        }
      }
       if (str == bterraza) {
        if (receivedChar == "ON"){
            digitalWrite(ledter, HIGH);
            Serial.println(receivedChar);
        }
        if (receivedChar == "OFF"){
            digitalWrite(ledter, LOW);
            Serial.println(receivedChar);
        }
       }

}

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

En este código se especifican los pines donde están conectados los sensores, los topics MQTT donde se recibirán comandos o se publicarán los estados del sensor al que pertenece y la conexión inicial que se hace a la red local del hogar.

Los mensajes MQTT son enviados al bróker MQTT que será el encargado de organizarlos. Este bróker se instalará en el servidor central de nuestro hogar, más concretamente en el servidor OpenHAB instalado en la Raspberry Pi 3.

4.2 Segundo paso: Instalación de nodo central de recogida y envío de datos

4.2.1 Placa Raspberry Pi

En la preconfiguración de la Raspberry Pi se instalará Raspbian. Esta distribución Linux ayudará a la hora de implementar los diferentes componentes y librerías para que funcione todo el sistema.

La principal causa de usar una Raspberry Pi con una distribución Linux es poder instalar la plataforma OpenHAB para dotar a la casa de un centro de gestión, administración y representación de dispositivos domóticos. A su vez y de forma paralela solucionar el problema de cambio dinámico de IP pública del router del hogar, instalando un cliente STUN que averiguará que IP posee y enviando dicha información al servidor en la nube donde se guardan los datos asociados a la cuenta de servicio del usuario, la cual se actualizará.

a) OpenHAB

Para la instalación de OpenHAB se han seguido los pasos descritos en el capítulo 3 apartado 3.1.2.2 y el paquete elegido para la configuración inicial ha sido el paquete Experto. Obteniendo las siguientes interfaces:

- Paper UI.
- Classic UI.
- Basic UI.
- HABPanel.
- HABmin.
- REST API interactiva.

OpenHAB se utilizará para facilitar la comunicación con los dispositivos, que en este caso será una placa NodeMCU. Dicha comunicación será a través del protocolo MQTT. La placa NodeMCU implantará 6 dispositivos asociados a 6 dependencias de la maqueta que representa el hogar, que se debe tener en cuenta a la hora de la configuración de los elementos de OpenHAB.

Como el protocolo de comunicación con los dispositivos del hogar es MQTT se debe configurar OpenHAB para que sea capaz de utilizarlo, para ello se seguirán los siguientes pasos:

1. Accede a la página principal del servidor OpenHAB a través de un navegador usando la IP:

   http://IP_RASPBERRYPI:PUERTO

   (normalmente el puerto está preconfigurado en el 8080)

2. Accede a la interfaz Paper UI.
3. Accede a la pestaña Add-ons. Aquí se buscará los complementos que harán que OpenHAB pueda implementar y comunicarse con cualquier tecnología de la cual exista su complemento en el listado.
4. Busca en la lista de Bindings el complemento más reciente llamado MQTT Binding y se le da a instalar (figura 4-1). Hecho esto se habrá instalado el conector de OpenHAB asociado a MQTT, de tal forma que OpenHAB podrá comunicarse con dispositivos MQTT.

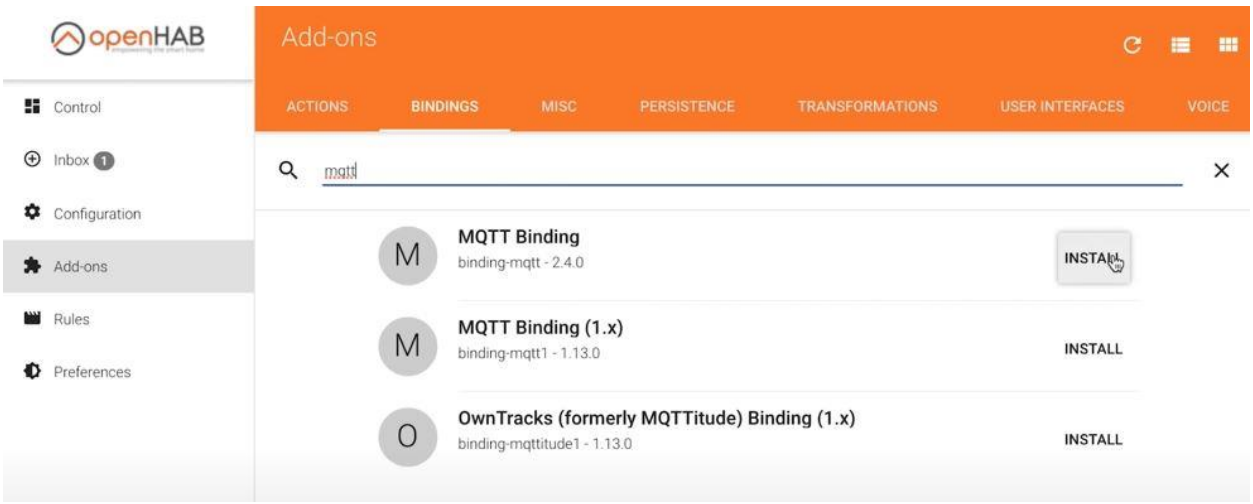


Figura 4-1: Instalación MQTT Binding.

5. Además en el servidor OpenHAB es necesario instalar, el nodo central que permite la gestión del protocolo MQTT. Este, aparte de poseer el sistema de colas de MQTT, puede funcionar como cliente. Para ello selecciona Add-ons – MISC.
6. Instala el complemento Embedded MQTT Broker. Una vez instalado se tendrán todas las funcionalidades activas gracias a OpenHAB. Tanto el cliente como el nodo central, el cual gestiona las publicaciones y suscripciones a los topics MQTT, serán capaz de publicar y recibir mensajes.
7. Se puede observar en la pestaña Inbox como aparece el elemento creado MQTT Broker.
8. A modo de prueba se puede usar el cliente MQTT.fx en cualquier ordenador y realizar pruebas para verificar que la instalación del protocolo MQTT ha sido correcta. Un ejemplo de la configuración de este cliente sería el de la figura 4-2.

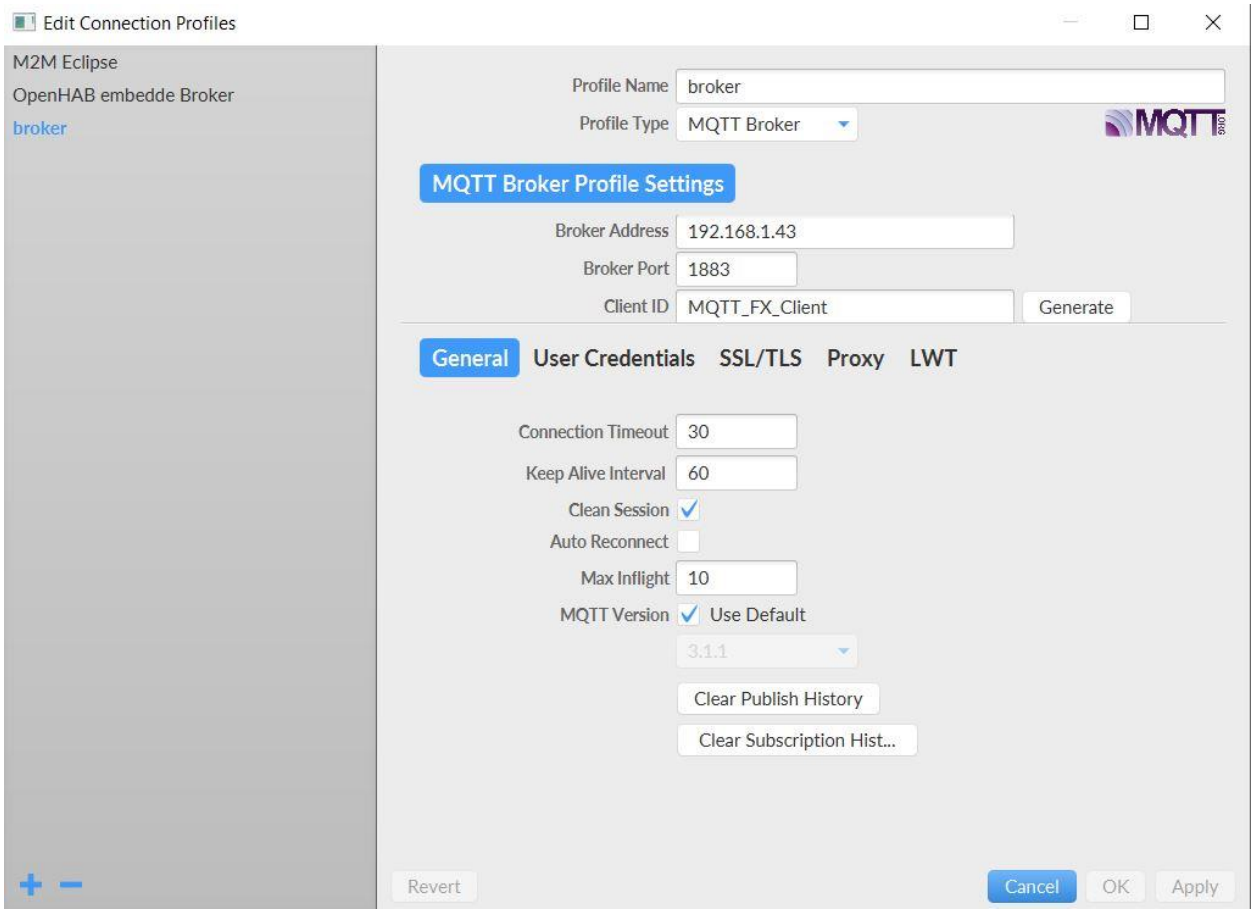


Figura 4-2: Configuración cliente MQTT.fx

Una vez instalada la tecnología ya se pueden instalar y configurar para su uso los dispositivos MQTT. En este apartado se explicará y configurará sólo un dispositivo MQTT siendo el mismo proceso el utilizado para el resto, siempre que la funcionalidad sea la misma. En el hogar se quieren implementar bombillas inteligentes en cada dependencia, por lo que son dispositivos con dos estados ON y OFF. Estos dispositivos en OpenHAB son definidos como Switch. Pues bien, a continuación se verá la implementación de una bombilla de una habitación del hogar en el servidor OpenHAB y su funcionamiento.

1. Accede a la interfaz PaperUI.
2. Accede a la pestaña Inbox.
3. Selecciona Search for Things y a continuación MQTT Binding.
4. Selecciona Add Manually.
5. Presiona sobre Generic MQTT Thing y aparecerá la interfaz de la figura 4-3.

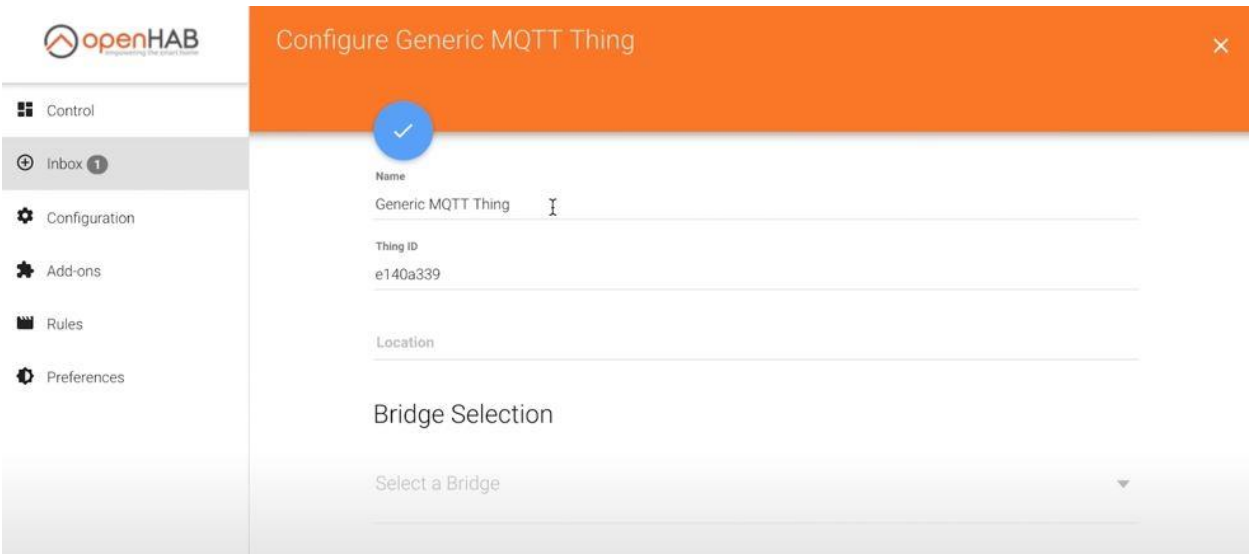


Figura 4-3: Interfaz Generic MQTT Thing de OpenHAB.

6. Añade un nombre y selecciona como Bridge el MQTT Broker. Pulsa sobre el check para finalizar. Así de simple es crear un Thing en OpenHAB.

La creación de este Thing es la indicación al sistema de que existe uno o varios dispositivos que funcionan con la tecnología MQTT y para configurarlos se deberá utilizar el Thing creado. Se puede asociar la creación de este Thing a la representación de la placa NodeMCU la cual alberga la bombilla. Ahora se asociarán los elementos con los que realmente se quiere interactuar en este Thing.

7. Una vez creado el Thing, se configuran los elementos asociados a este Thing que le dotarán de funcionalidad e interacción. Para ello accede a la pestaña Things y observa que el Thing ha sido creado.
8. Pulsa sobre el lápiz para entrar en el editor. Aparecerá la interfaz de la figura 4-4 y accediendo al Channel la figura 4-5.

Figura 4-4: Interfaz añadir Channel a un Thing en OpenHAB.

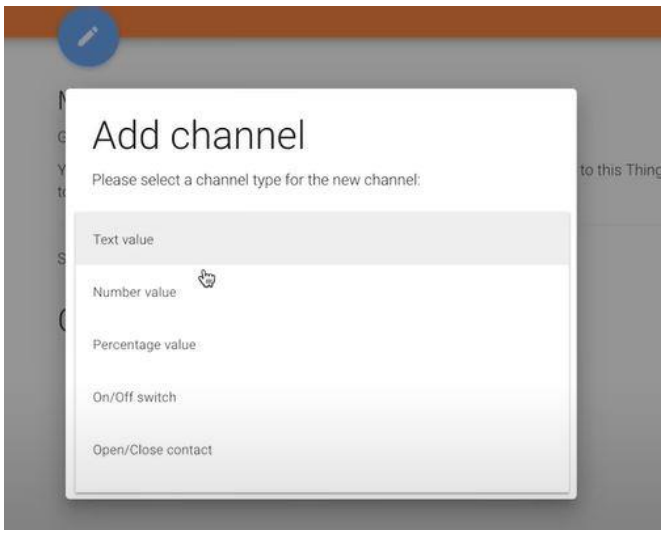


Figura 4-5: Tipos de Channel en OpenHAB.

9. Entonces se añadirá el Channel. Aquí se indica el tipo del Channel (Switch), un id para su distinción y una etiqueta. También el topic MQTT al que está asociado el elemento y la carga del paquete que se enviará al hacer un ON o un OFF (ON y OFF respectivamente).

El Channel actúa como conector entre el Thing (la tecnología que se quiere usar) y el Item (el dispositivo físico que se asocia a la tecnología). Puede haber varios Items asociados a un mismo Channel, por ejemplo una habitación que tiene dos bombillas y ambas quieren ser encendidas y apagadas a la vez, pues si ambas se asocian al mismo Channel ambas estarán suscritas al mismo topic.

10. Pulsa sobre el círculo que precede al canal y selecciona el item al que se quiere asociar el Channel. Como no existe ninguno de momento hay que pulsar sobre la opción create new item. La interfaz es la asociada a la figura 4-6.

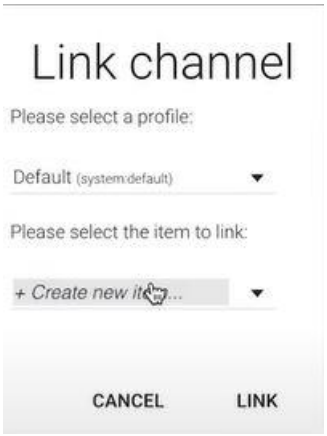


Figura 4-6: Interfaz Link Channel de OpenHAB.

11. Configura el nuevo item proporcionando un nombre, una etiqueta, una categoría y un tipo. Y se pulsa en Link.
12. Aparecerá entonces el circulo relleno. Esto quiere decir que existen Items asociados al Channel.
13. Reinicia OpenHAB para que los cambios se hagan efectivos.

```
sudo systemctl restart openHAB2.service
```

14. Pulsa sobre la pestaña Control dentro de la interfaz PaperUI y se observa los Items como funcionan y su estado actual.

El siguiente paso es crear la interfaz asociada a los Items para la interacción con el usuario.

15. En la Raspberry Pi accediendo a través de Putty. Accede al fichero /etc/openHAB2/sitemaps/ home.sitemap
16. Crear la estructura de la interfaz. En el caso de este ejemplo se indica la sintaxis como en la siguiente figura 4-7:

```
sitemap home label="Main Menu" {
    Frame label="Salón"{
                Switch item=LED1
                Switch item=LED2
        }
        Frame label="Exterior"{
                Text item=Temperature
        }
}
```

Figura 4-7: Interfaz sitemap del Trabajo Fin de Grado.

17. Accede a la interfaz BasicUI donde se tendrá la interfaz configurada con los elementos listos para su interacción y funcionamiento.

Estos pasos serán los mismos que se realizarán para el resto de los elementos. En el caso de este Trabajo Fin de Grado solo se dispone de una placa NodeMCU que contiene los 6 dispositivos MQTT que se alojan en diferentes partes de la casa. Entonces solo es necesario la creación de un Thing ya que todos utilizan el protocolo MQTT y se les asocia un canal diferente para cada dispositivo con el tipo Switch ya que todos son bombillas. Es interesante ver que a pesar de que en la instalación son elementos diferentes, todos funcionan como un Switch ON/OFF y es por ello por lo que se utiliza la misma configuración para implementarlos.

Por otro lado, a parte de la interfaz BasicUI, se pueden gestionar los diferentes elementos del sistema, es decir los que se hayan creado en él, a través de peticiones HTTP usando la API REST que proporciona OpenHAB. Esta funcionalidad será utilizada por la aplicación Android que se explica en el apartado 4.5.

b) Cliente STUN

Bien, ya se ha creado el servidor local y localmente se puede gestionar e interactuar con los dispositivos del hogar. Pero ¿qué ocurre si se quiere controlar los dispositivos desde una red diferente a la que pertenece el servidor local?. Pues ocurre que ante un acceso desde fuera de la red el router bloquea toda petición. Para ello es necesario crear una regla NAT que identifique una petición externa con IP router y puerto específico que se traduzca como la IP de la Raspberry Pi y puerto el asignado a OpenHAB. Eso solucionaría el problema del acceso desde fuera. Pero se necesitaría saber la IP externa del router para realizar las peticiones. Y esta IP cada vez que se reinicie el router cambiará. Por ello se instalará y usará el protocolo STUN.

En esta sección se analizará solo la parte del cliente, por lo que es necesario saber que el servidor simplemente devolverá la IP y el puerto desde donde se realiza la petición. En este caso devuelve la petición enmascarada por el router, de tal forma que lo que devuelve es la IP externa del router.

Cada vez que se pierda la conexión en la Raspberry Pi, así se identifica una caída de la red o un encendido del router, el cliente STUN simplemente lanzará una petición HTTP con los campos de usuario y contraseña al servidor STUN alojado en el servidor en la nube, el cual se encargará de interpretar la petición. La respuesta será un paquete HTTP Request con la IP externa del router y el puerto utilizado.

Para la instalación del cliente STUN se debe tener instalado los siguientes paquetes:

- g++: `sudo apt-get install g++`
- make: `sudo apt-get install make`
- Boost: `sudo apt-get install libboost-dev`
- OpenSSL: `sudo apt-get install libssl-dev`

Se puede descargar el cliente STUN del repositorio de GitHub:

https://github.com/jselbie/stunserver

Ejecuta el comando make en el directorio padre. Y ejecuta el fichero stuntestcode para comprobar que se ha instalado correctamente. A continuación se puede ejecutar el fichero stunclient seguido de la IP donde se aloja el servidor STUN, el puerto por defecto será el 3478 asociado al protocolo STUN, así el cliente realizará la petición para averiguar si existe un cambio de IP externa del router.

```
make
./stuntestcode
./stunclient IP
```

La interpretación del paquete en el servidor se explicará en el siguiente apartado 4.4.

4.3 Tercer paso: Instalación de servidor en la nube

El almacenamiento de credenciales, la autenticación, el registro de usuarios y el alojamiento del servidor STUN deben ser accesibles desde cualquier punto y en cualquier momento. Por ello la mejor manera de ofrecer este servicio es utilizar un servidor en la nube. Durante el desarrollo del Trabajo Fin de Grado se han utilizado dos tipos de proveedores de servidores en la nube: Clouding y Amazon Web Services (en adelante AWS, se muestra su interfaz en la figura 4-8). Ambos servicios son muy parecidos y económicos. Aunque la versatilidad que ofrece AWS y el plan gratuito de 12 meses hacen que sea la mejor opción para este Trabajo Fin de Grado. Básicamente se ha creado un servidor Ubuntu 18.04 en el que se ejecutan los diferentes servicios antes mencionados. En primer lugar se explicará el servidor STUN, luego el servicio REST de autenticación de credenciales y posteriormente el manejo de la base de datos. De la base de datos hay que especificar que está creada en Postgresql y almacena el usuario, la contraseña, la IP y el puerto.

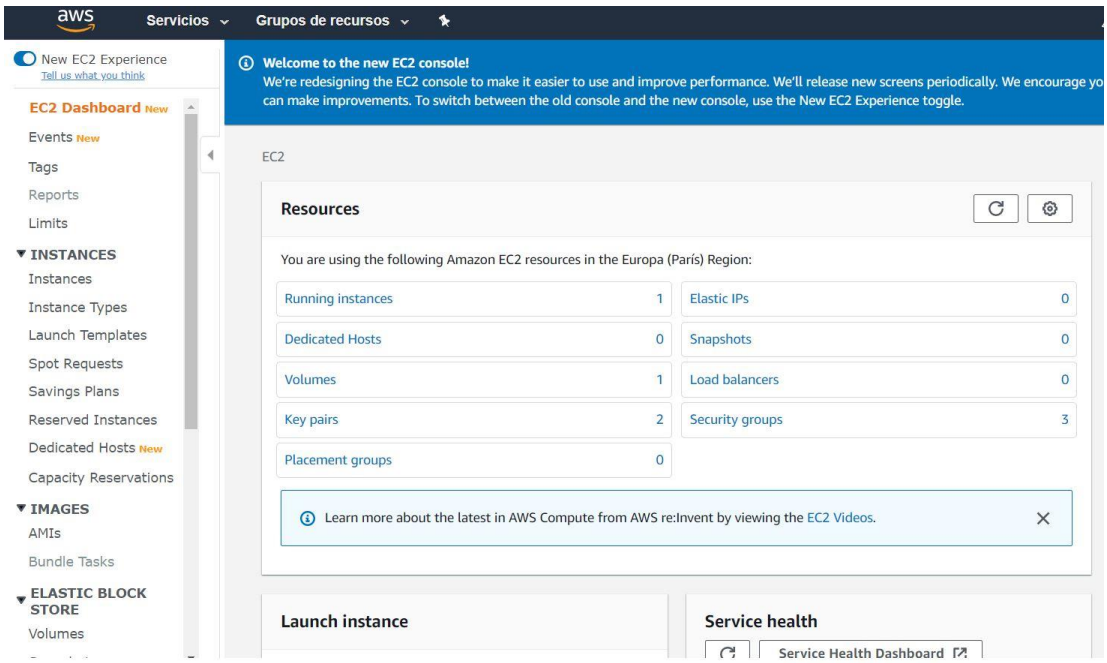


Figura 4-8: Interfaz AWS de Amazon.

a) Servidor STUN

El principal problema a la hora de establecer una comunicación externa es que se debe saber que IP externa tiene el router que da acceso al servidor local. Y como esta IP es dinámica, se utiliza el protocolo STUN para averiguar qué IP posee en ese momento. En el anterior apartado se ha explicado la parte del cliente y cómo envía las peticiones, en esta ocasión se verá cómo funciona el servidor, cómo se instala y cómo procesa los paquetes.

El funcionamiento básico del servidor STUN se basa en procesar los paquetes HTTP recibidos, analizar las cabeceras y extraer la IP y el puerto origen del paquete, esto coincidirá con la IP externa del router ya que al lanzar la petición desde una máquina de dentro de la red local, el router aplicará el enmascaramiento de su IP. Devolverá así una respuesta HTTP con la IP y el puerto que serán guardados en el cuerpo del paquete. El cliente recibirá el paquete viendo la IP externa y el puerto del router.

Como funcionalidades adicionales se ha añadido el análisis del cuerpo del paquete, donde se encontrará el usuario y la contraseña del usuario asociado a la IP. Si es la primera vez que el usuario ha accedido al sistema se creará una nueva entrada en la base de datos donde se guardan los usuarios del servicio. Si no, se comprueba si la IP es diferente a la guardada y si es así se actualiza dicha IP en la base de datos.

Para la instalación del servidor STUN básica se debe tener instalado los siguientes paquetes:

- g++: `sudo apt-get install g++`
- make: `sudo apt-get install make`
- Boost: `sudo apt-get install libboost-dev`
- OpenSSL: `sudo apt-get install libssl-dev`
-

Se puede descargar el servidor STUN del repositorio de GitHub:

https://github.com/jselbie/stunserver

Ejecuta el comando make en el directorio padre. Y ejecuta el fichero stuntestcode para comprobar que se ha instalado correctamente. A continuación se puede ejecutar el fichero stunserver que si no se le es indicado ningún argumento más, escucha por defecto las peticiones en el puerto 3478 y el servidor STUN comenzará a funcionar.

```
make
./stuntestcode
./stunserver
```

b) Servicio REST

El servicio REST que se implantará permite la autenticación de los usuarios en la aplicación móvil. De tal manera que esta permita un acceso seguro a los controles de los dispositivos del hogar. Para la creación del servicio se ha utilizado SPRING. El código del servicio se encuentra en la sección 1.2 del Anexo.

El servicio recibirá peticiones HTTP las cuales se clasificarán según su tipo y analizadas para obtener una respuesta. Los tipos de las peticiones y su respuesta respectivamente son:

PETICIÓN	RESPUESTA	FINALIDAD
GET	"Hola"	Probar si el servicio está operativo.
POST (email, contraseña)	IP y puerto. OK ERROR. Si algo ha ido mal.	Autenticación del usuario en el sistema.
PUT (email, contraseña, IP, puerto)	Integer 1. OK. Integer 0. ERROR.	Inserta un nuevo usuario en el sistema.

Tabla 4-1: Tipos de peticiones y respuestas del servicio REST.

A la hora de comprobar los datos de un usuario dado de alta en el sistema, se utiliza la herramienta JDBC de Java para la consulta de los datos en la base de datos PostgreSQL.

c) Base de Datos PostgreSQL

Las credenciales de usuario se almacenarán en una simple tabla donde los campos serán: usuario, contraseña, IP, puerto. Con clave primaria el usuario y la IP.

En las siguientes imágenes se observa un ejemplo de la base de datos PostgreSQL que se utiliza:

```
postgres=# \d
          List of relations
 Schema |   Name    | Type  |  Owner
--------+-----------+-------+----------
 public | respuesta | table | postgres
(1 row)
```

Figura 4-9: Base de datos “respuesta”.

```
postgres=# select * from respuesta;
     email      | passwd |       ip        | puerto
----------------+--------+-----------------+--------
 jose@gmail.com | 1234   | 192.168.141.138 | 3478
 josdeala       | 1234   | 88.8.104.104    | 8080
 jose           | 1234   | 88.8.104.104    | 54321
 usuario        | 1234   | 83.45.50.241    | 54321
 openhab        | 1234   | 88.8.104.122    | 54321
(5 rows)

postgres=#
```

Figura 4-10: Ejemplo de contenido de la tabla “respuesta” de la base de datos.

4.4 Cuarto paso: Aplicación Android

Finalmente se crea la aplicación Android para hacer uso del sistema. Esta tomará el papel de cliente en este Trabajo Fin de Grado y tendrá como finalidad mostrar al usuario todas las viviendas conectadas al sistema, las habitaciones y los sensores conectados al servidor central, pudiendo así interactuar con ellos.

En primer lugar se le dará la bienvenida al usuario con la interfaz de “Inicio de sesión” como se muestra en la siguiente ilustración.

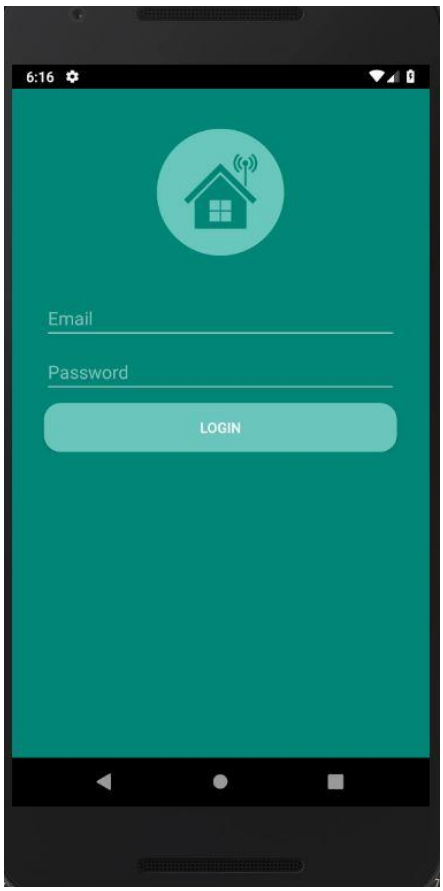


Figura 4-11: Interfaz aplicación móvil “Inicio de sesión”.

Esta interfaz constará de dos campos y un botón. El primero de ellos, es un campo de edición de texto en el que el usuario puede introducir su email para poder acceder al sistema, e introduciendo en el segundo campo la contraseña asociada a su cuenta. Por último, el usuario puede pulsar sobre el botón “LOGIN” para autenticarse en el sistema y poder así acceder a la aplicación.

Al pulsar el botón “LOGIN” la aplicación enviará una petición POST, la cual usará el protocolo HTTPS con las credenciales de usuario. Si algo ha salido mal, no se permitirá el acceso a los datos y esto ocurrirá hasta que se introduzcan unas credenciales correctas. En el caso de que haya salido bien, la aplicación recibirá la IP y el puerto al que deberá realizar la siguiente petición para obtener el número de viviendas asociadas al usuario. Para ello se realiza una petición GET al servidor local del usuario (OpenHAB). Una vez realizada dicha operación con éxito la aplicación muestra la interfaz “Viviendas”, como se observa a continuación.

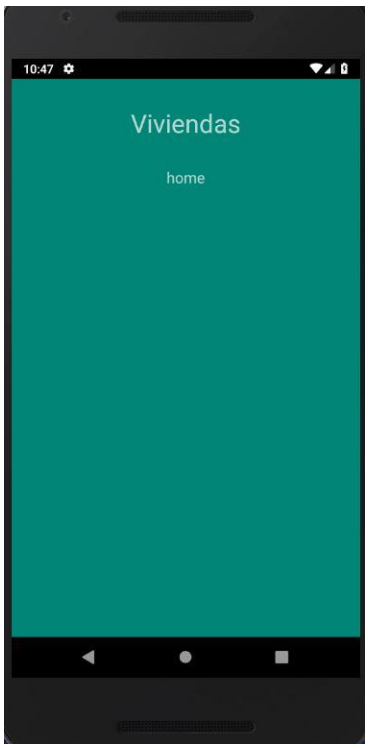


Figura 4-12: Interfaz aplicación móvil “Viviendas”.

En esta interfaz la aplicación muestra las viviendas asociadas al usuario. Pulsando sobre el nombre de la vivienda se puede acceder a todas las habitaciones y a todos los sensores de la vivienda. Para lograrlo se envía una petición GET que devolverá todos los elementos configurados en OpenHAB. De modo que se muestre así la interfaz “Habitaciones” como en la figura 4-13.

Figura 4-13: Interfaz aplicación móvil "Habitaciones".

En esta aplicación la interfaz depende de la vivienda, de las habitaciones y de los elementos conectados a la casa. Por consiguiente las interfaces "Viviendas" y "Habitaciones" crean los objetos dinámicamente, es decir, crearán sólo el número de objetos que devuelvan las peticiones REST y el tipo correspondiente, dependiendo únicamente de estas dos características. Los tipos de los objetos de la interfaz "Viviendas" son botones de tipo texto, esto es, un texto que puede ser pulsado. En cambio, los tipos de los objetos de la interfaz "Habitaciones" pueden ser de cuatro tipos diferentes: "Switch", "Slider", "Text" y "Setpoint".

- "Switch": Botón ON/OFF.
- "Slider": Barra de control ajustable de un mínimo a un máximo.
- "Text": Campo de texto con información
- "Setpoint": 2 botones de acción. (Ejemplo: Más o menos temperatura).

Si la cantidad de información presentada en cualquier interfaz supera los límites de la pantalla, simplemente deslizando hacia arriba o hacia abajo se puede observar esta por completo.

Gracias a todo esto, el control de todos los elementos de la casa se efectúa de manera eficiente y sencilla.

4.5 Descripción de la comunicación entre componentes

En este apartado se explicará un caso concreto de comunicación entre componentes, donde un usuario mandará una orden a un dispositivo ubicado en su hogar.

Se supone el inicio de la comunicación cuando un usuario accede a la aplicación móvil, introduce un nombre de usuario y una contraseña y pulsa sobre el botón de la interfaz "LOGIN". Esta acción va a desencadenar la primera comunicación existente y será entre el móvil y el servicio web de autenticación alojado en la nube. La aplicación móvil en este caso envía un paquete HTTP tipo POST con el usuario y la contraseña y será recibido por el servicio web que estará en el servidor AWS a la escucha de peticiones. El servicio al recibir la petición POST comprobará en la base de datos PostgreSQL la veracidad de los datos. Se supone que son correctos, así que el servicio tomará la IP y el puerto asociados al usuario y enviará ambos como respuesta a la petición inicial.

Aquí se inicia la segunda parte de la comunicación entre componentes. La aplicación recibe la IP del usuario y su puerto y tomará dichos valores para realizar una nueva petición en este caso de tipo GET y la enviará a la dirección que marcan los datos que se han recibido. Esta petición pasará por el router del hogar del usuario, que aplicando las reglas NAT necesarias reenviará a la placa Raspberry Pi donde se aloja el servidor OpenHAB. El servidor OpenHAB estará a la espera de recibir peticiones en su API REST. En este caso la petición GET está asociada con los sitemaps alojados en el sistema. La respuesta a esta petición es un JSON con el nombre del hogar y un link. Este link será donde la aplicación (al pulsar sobre el nombre del hogar) haga otra petición GET a la IP y puerto anteriores, esto provocará que el servidor OpenHAB mande como respuesta todas las estancias con todos los dispositivos asociados a cada una de las estancias. Entonces la aplicación será capaz de listar cada estancia con sus dispositivos con los estados de ellos (los que en ese momento OpenHAB tiene almacenados).

A continuación, un usuario pulsa sobre un botón ON/OFF cambiando el estado del botón de la bombilla del salón, por ejemplo de OFF a ON. Esto desencadena que la aplicación envíe una nueva petición POST con el nuevo estado (ON) y el nombre del dispositivo (bombillaSalon) al servidor OpenHAB. Cuando este lo recibe cambia el estado del controlador asociado a ese dispositivo, indicando que existe un cambio de estado. En este momento el servidor OpenHAB actuará como cliente MQTT. Lanzará una petición así mismo al topic que como ejemplo será /topic/bombillaSalon con el nuevo estado (ON). Al coincidir el bróker MQTT con el cliente será la misma máquina la que lance la petición, mirando en el sistema de colas que petición ha entrado, comprobando el destinatario que está suscrito a dicho topic y enviando la petición, en este caso a la placa NodeMCU que se encargará de apagar la bombilla del salón. El servidor OpenHAB contestará a la aplicación con el código estándar de respuesta 200 OK.

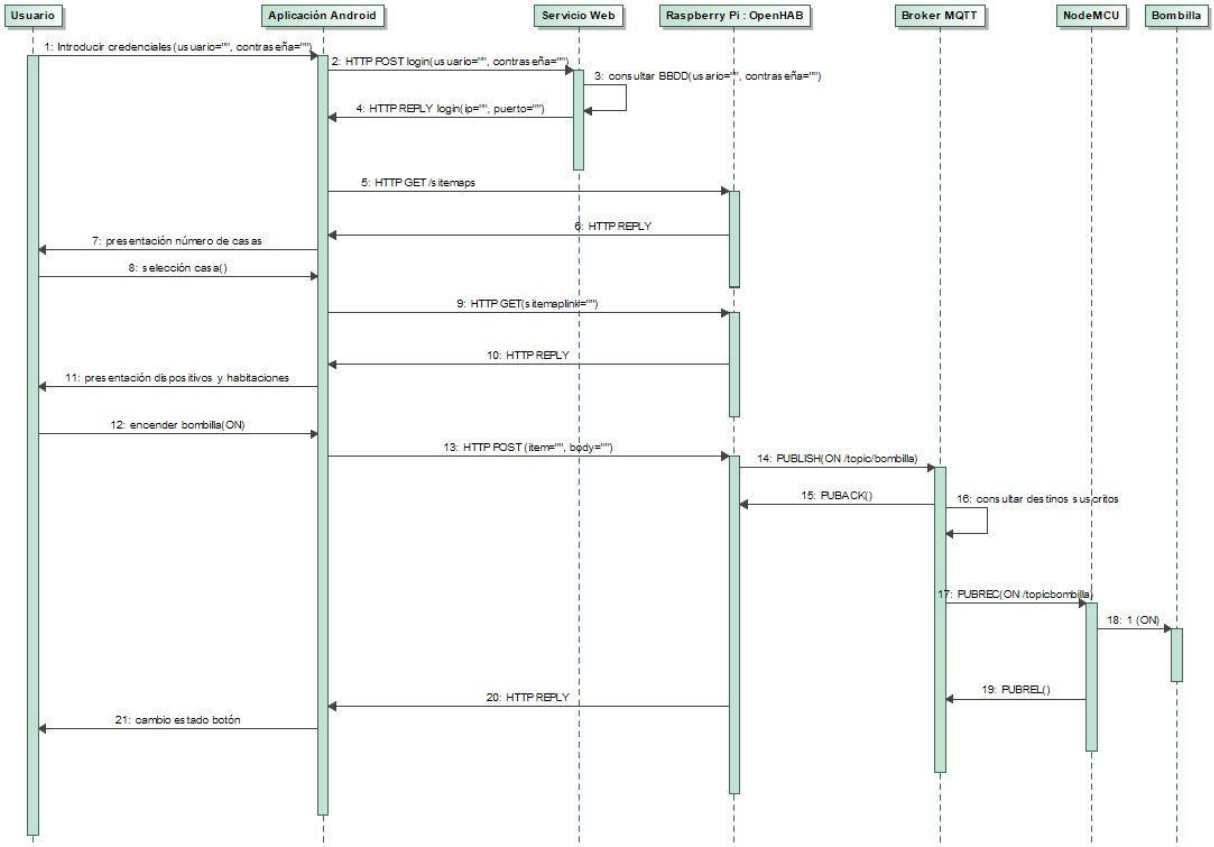


Figura 4-14: Diagrama de secuencia de la comunicación entre componentes.

5 FUTURAS AMPLIACIONES

Cáete siete veces, levántate ocho

Proverbio japonés

En este Trabajo Fin de Grado se ha explicado cómo realizar una solución domótica sencilla y completa, donde un usuario cualquiera puede gestionar los diferentes dispositivos inteligentes de su hogar e interactuar con ellos. Este proyecto puede considerarse una primera solución y por ende puede mejorarse y ampliarse en muchas de las ramas que lo componen como seguridad, funcionalidad o diseño. En este capítulo se expondrán las líneas de mejora que se han observado durante el desarrollo del trabajo.

5.1 Líneas de mejora

- Diseño:
  - El desarrollo de un lenguaje unificado para la creación de objetos en OpenHAB, definiendo así un lenguaje común referente a las divisiones que pueden encontrarse en un hogar. Por ejemplo: un hogar debe etiquetarse como “home”, una habitación como “room”, etc.
  - La incorporación de nuevos objetos en la aplicación móvil que ya existen en el sistema domótico OpenHAB como pueden ser: “Group”, “Selection”, “Colorpicker”, “WebView”, “MapView”, “Image”, “Video” o “Chart”.

    Al igual que se ha hecho en la aplicación móvil se pueden comparar estos objetos con algunos de los existentes en Android Studio. Entonces sólo se necesitaría identificar el tipo del objeto en concreto y asociarlo a un elemento en la interfaz de la aplicación, estableciendo además los campos que requiera.
  - Definir una actividad dentro de la aplicación móvil donde aparezca información propia de la aplicación, su desarrollador y su propietario.

    Esto resulta muy útil a la hora de reportar incidencias (‘troubleshooting’) y ponerse en contacto con los encargados de la aplicación, además de incorporar información sobre los términos y condiciones que conlleva el uso y el registro de la aplicación.
  - La capacidad de cambiar la aplicación móvil a otro idioma. Esto implicaría el acceso a un mercado más extenso. Simplemente habría que añadir los campos de texto de la aplicación a los ficheros de texto que se encuentran en la carpeta “string”, creando diferentes ficheros según los idiomas.

    Estos campos de texto poseen un identificador que estará en el código de la aplicación y será por el que se sustituya por el campo de texto al que haga referencia. Si se elige el idioma “español” se cargarán los identificadores asociados al fichero del idioma español, y así pasaría con cualquier otro.
  - Cambiar el diseño de la aplicación móvil a fuentes y colores comerciales, tras un posible estudio de marketing.
  - Creación de una interfaz web para el acceso de los usuarios al sistema desde un navegador web.

    Con esto se daría mayor operabilidad al usuario con el sistema. El uso del servicio REST permite adaptarse a cualquier tipo de cliente que realice este tipo de peticiones.

- Funcionalidad:
  - Interacción con asistentes de voz como Google Home, Alexa o Siri.
  - Registro a través de la aplicación móvil en el sistema.

    No solo se permite a la hora de instalar el servicio en un hogar, sino también desde cualquier lugar y de forma remota, sabiendo siempre de antemano la IP externa y el puerto asociados al router.

    La incorporación de un botón "REGISTER" asociado a una nueva actividad donde se introducen los campos y se realiza la petición PUT del protocolo HTTP.

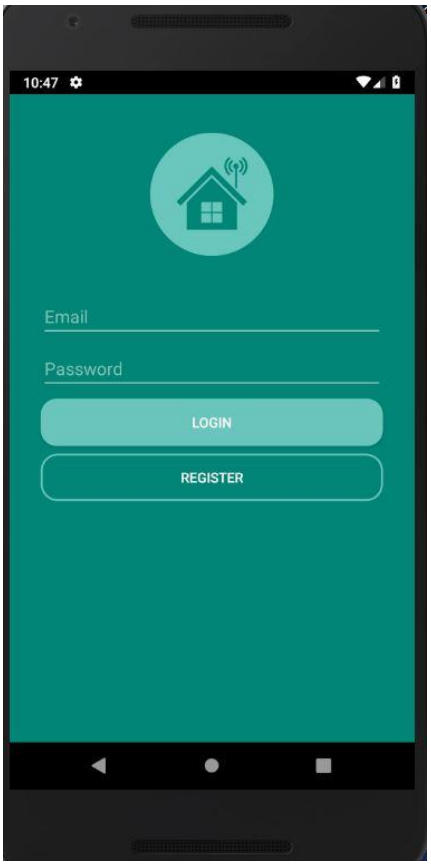


Figura 5-1: Interfaz de la aplicación móvil con funcionalidad de registro.

  - Sistema de sesiones. Un usuario sólo podrá tener una sesión activa en el móvil, si la sesión caduca será necesario autenticarse nuevamente.
  - Creación de usuarios invitados. Un usuario podrá dar de alta en su hogar a un usuario invitado el cual tendrá menos permisos y menos funcionalidades a la hora de interactuar con el hogar.

    OpenHAB permite la creación de roles y con la modificación de la base de datos del servidor en la nube para que admita una nueva característica "roles" se podría incorporar esta funcionalidad.
  - Recopilación de datos y su presentación en gráficas y hojas de cálculos (por ejemplo, para determinar el consumo).
  - Implementar campos de publicidad y monetización.
  - Sistema de alertas:

    El sistema puede enviar la información a la aplicación creando una notificación de alarma.

        - Alertas de consumo.

          El usuario puede introducir su cuota de consumo y poner límites, los cuales al sobrepasarse, el sistema envíe la información a la aplicación creando una notificación de alarma.

        - Alertas de movimiento.
        - Alertas de encendido.

    - Creación de reglas.

      OpenHAB proporciona la funcionalidad de crear reglas en el sistema, de tal manera que se podrían gestionar y modificar usando la API REST de OpenHAB. Esto permitiría al usuario una mayor interacción con el sistema, creando por ejemplo juego de luces, ambientes o alarmas.

- Seguridad:
    - Realizar peticiones HTTPS a servidor local.

      Actualmente el sistema sólo realiza peticiones HTTPS entre la aplicación y el servidor en la nube. Aplicando el mismo funcionamiento se puede crear una comunicación segura entre la aplicación móvil y el servidor local alojado en la Raspberry PI.

    - Creación de claves públicas y privadas por cada usuario.

      Cada usuario posee su propia clave pública y privada. La pública será enviada al servidor para poder realizar una comunicación segura bidireccional completa.

6 CONCLUSIONES

Gracias a la evolución y a las continuas mejoras en el mundo de Internet, la humanidad ha ido observando cómo su día a día se veía mejorado por el uso de las tecnologías emergentes, desde las tareas más sencillas y rutinarias, como en los trabajos más forzados. La posibilidad de interactuar con diversos dispositivos de cualquier clase de una forma sencilla, y prácticamente de forma automática, ha impulsado el mundo del IoT en esta era de compartición de datos.

Actualmente existen grandes potencias como Google o Amazon que proporcionan soluciones de cara a la interconexión y la agrupación de dispositivos inteligentes dentro de una red. ¿Cuál es el futuro en este campo?. ¿Cuál es el siguiente paso dentro del IoT y la domótica?.

La gran solución que se atisba en el horizonte más próximo son las Smart Cities donde todo está conectado, desde dispositivos hasta información. Es un hecho que los propios teléfonos móviles recogen todo tipo de información, y que esta es compartida para mejorar la experiencia del usuario. En una ciudad inteligente, gracias a compartir toda la información, se puede calcular y gestionar el consumo energético, reducir las emisiones de dióxido de carbono, mejorar el funcionamiento del transporte público, incluso evitar una pandemia mundial. El resultado u objetivos de este tipo de ciudades siempre estarán enfocados en generar riqueza y mejorar la sostenibilidad, y con ello el aumento del nivel de bienestar social.

Un ejemplo claro de hacia dónde vamos, se encuentra en los paneles de publicidad de tiendas donde se muestran los productos o bienes que pueden interesarnos especialmente a uno mismo. El teléfono, la pulsera inteligente, Alexa o una búsqueda en un ordenador recogen la información y posteriormente la comparten para que al andar por la calle se nos presente la publicidad más acorde a nuestros gustos.

Los Smart Buildings son un ejemplo a gran escala del hogar que se ha propuesto en este proyecto. Controlar la ventilación, la temperatura, las luces o la seguridad facilita a los propietarios y a los administradores la capacidad de mejorar el rendimiento y los recursos del edificio, así como controlar los gastos que este produce.

La diferencia entre el antes y el después es la interconexión de los dispositivos: sensores que captan datos y los comparten dentro de una misma red. Un edificio puede ser antiguo y en cambio estar dotado de sensores que lo conviertan en inteligente, como el claro ejemplo de infinidad de museos históricos que en la actualidad presentan sus obras de arte mediante realidad aumentada gracias a esos sensores que detectan el paso de los usuarios y sus intereses.

En este Trabajo Fin de Grado se ha puesto en marcha la solución en un hogar prototipo. Pero la realidad va mucho más allá de este ejemplo. Fábricas, grandes almacenes, empresas, edificios, parkings, hoteles son casos donde la tecnología y la domótica van a un nivel más allá.

Pronto todo estará interconectado. Desde las calles y sus farolas hasta la batería del coche. Surgen así muchas incógnitas que están por resolverse y que a día de hoy se siguen investigando. ¿Será seguro estar conectado las 24 horas del día en cualquier lugar en el que nos encontremos?¿Será la vida más sencilla gracias a tener acceso a toda la información?¿A qué coste?¿Qué datos estamos proporcionando y qué se están haciendo con ellos?

Tras haber profundizado en estas ideas, se ha puesto en práctica todo este contexto del mundo de la información en un entorno domótico que permita llevar estas incógnitas, entre otros muchos aspectos, a un proyecto tangible. A partir de una plataforma libre orientada a la domotización del hogar, se ha desarrollado un sistema que aúna un servidor local, un servicio web y una aplicación móvil. El servidor local sirve de nexo de unión entre los distintos dispositivos reuniendo la información necesaria y poniéndola en común. Mediante la aplicación, el usuario podrá ver reflejado en su pantalla el control que posee sobre todos los dispositivos de su hogar, sean de la tecnología que sean, y desde cualquier punto del mundo en tiempo real. De modo que, el usuario final pasará a ser un eslabón más de la red IoT.

REFERENCIAS

[1] openHAB, open Home Automation Bus. Documentation. https://www.openHAB.org/docs/

[2] openHAB, open Home Automation Bus. https://www.openHAB.org/

[3] Raspberry Pi, Raspberry Pi 3 + Datasheet. https://www.raspberrypi.org/documentation/hardware/computemodule/datasheets/rpi_DATA_CM3plus_1p0.pdf

[4] Arduino, Guide: Introduction. https://www.arduino.cc/en/Guide/Introduction

[5] IETF, STUN RFC. https://www.ietf.org/rfc/rfc5389.txt

[6] AWS, Amazon Web Services. https://aws.amazon.com/

[7] Mouser, DHT11 sensor datasheet. https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf

[8] Two RELAY Module. http://www.handsontec.com/dataspecs/2Ch-relay.pdf

[9] OASIS, MQTT Version 5.0. https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf

[10] PostgreSQL, PostgreSQL. https://www.postgresql.org/

[11] Jselbie, STUN Server and Client. https://github.com/jselbie/stunserver

[12] Spring, Spring Guides. https://spring.io/guides/

[13] Spring. https://spring.io/

[14] Handsontec, NodeMCU Datasheet. https://www.handsontec.com/pdf_learn/esp8266-V10.pdf

[15] AWS, Amazon EC2. https://aws.amazon.com/es/ec2/

[16]ESP-12E,components101. https://components101.com/sites/default/files/component_datasheet/ESP12E%20Datasheet.pdf

[17] Adafruit, Led. https://cdn-shop.adafruit.com/datasheets/WP7113SRD-D.pdf

[18] Arduino, Arduino IDE. https://www.arduino.cc/en/main/software

[19] SourceForge, Win32DiskImager. https://sourceforge.net/projects/win32diskimager/

[20] Uptodown, SDcardFormatter. https://sd-card-formatter.uptodown.com/windows

[21] Putty, Putty. https://www.putty.org/

[22] Free Software Foundation, GNU. https://www.gnu.org/

[23] Ubuntu, Ubuntu. https://ubuntu.com/

[24] Raspberrypi, Raspbian. https://www.raspberrypi.org/downloads/raspberry-pi-os/

[25] OpenHAB, OpenHABian. https://www.openHAB.org/docs/installation/openHABian.html

[26] Jens Deters, MQTT.fx. https://mqttfx.jensd.de/

[27] Android, Android Studio IDE. https://developer.android.com/studio

[28] Android, Android Studio Guide. https://developer.android.com/studio/intro

[29] Postman, Postman. https://www.postman.com/

[30] Apache, Maven. https://maven.apache.org/

[31] Apache Maven, POM.xml. https://maven.apache.org/guides/introduction/introduction-to-the-pom.html

[32] Spring, Spring Security. https://spring.io/projects/spring-security

[33] Swagger, Swagger. https://swagger.io/

[34] Azul, Zulu Embedded OpenJDK Java 8. https://www.azul.com/products/zulu-embedded/#:~:text=Deutsch%20(German)-,Zulu%20Embedded%E2%84%A2,Java%20for%20ISVs%20%26%20Embedded%20Systems&text=Azul%20is%20the%20world's%20largest,embedded%20systems%2C%20and%20the%20IoT.

ANEXO

1. Código de las implementaciones

1.1. Aplicación Android

- AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.dam.inicio">

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:name=".ColaVolley"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.NoActionBar"
        android:usesCleartextTraffic="true">
        <activity android:name=".Casa"></activity>
        <provider
            android:name=".MiContentProvider"
            android:authorities="com.contentprovider.elementos" />
        <activity android:name=".Habitacion" />

        <uses-library
            android:name="org.apache.http.legacy"
            android:required="false" />

        <activity android:name=".Datos" />
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

- MainActivity.java

```
package com.dam.inicio;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.content.Intent;
import android.widget.EditText;
import android.widget.TextView;

import com.android.volley.DefaultRetryPolicy;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.HurlStack;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.io.InputStream;

import java.net.HttpURLConnection;
import java.net.URL;

import java.security.KeyManagementException;
import java.security.KeyStore;
import java.security.KeyStoreException;
import java.security.NoSuchAlgorithmException;
import java.security.cert.Certificate;
import java.security.cert.CertificateException;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;

import java.util.HashMap;
import java.util.Map;

import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.TrustManager;
import javax.net.ssl.TrustManagerFactory;
import javax.net.ssl.X509TrustManager;

public class MainActivity extends AppCompatActivity {

    private EditText email;
    private EditText password;
    public static String ip, puerto;
    private int bandera = 0;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        email = (EditText) findViewById(R.id.editText3);
        password = (EditText) findViewById(R.id.editText4);
    }

    public void clickLogin(View view) {

        String url = "https://35.180.126.242:8443/login";
        HurlStack hurlStack = new HurlStack() {
            @Override
            protected HttpURLConnection createConnection(URL url) throws
IOException {
                HttpsURLConnection httpsURLConnection = (HttpsURLConnection)
super.createConnection(url);
                try {

httpsURLConnection.setSSLSocketFactory(getSSLSocketFactory());

httpsURLConnection.setHostnameVerifier(getHostnameVerifier());
                } catch (Exception e) {
                    e.printStackTrace();
                }
                return httpsURLConnection;
            }
        };

        final String em = email.getText().toString().trim();
        final String pass = password.getText().toString().trim();

        final StringRequest jsonObjectRequest = new
StringRequest(Request.Method.POST, url, new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                try {
                    Log.d("EUREKA",response.toString());
                    parseData(response);
                    if(bandera==1) {
                        Intent intent = new Intent(getApplicationContext(),
Datos.class);
                        intent.putExtra("ip",ip);
                        intent.putExtra("puerto",puerto);
                        startActivity(intent);
                    }
                } catch (Exception e) {
                    Log.d("ERROR1",e.toString());

                }
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Log.d("ERROR2",error.toString());
            }
        }){
            @Override
            protected Map<String,String> getParams(){
                Map<String,String> params = new HashMap<String, String>();
```

```
                params.put("email",em);
                params.put("password",pass);

                return params;
            }

        };


        jsonObjectRequest.setRetryPolicy(new DefaultRetryPolicy(
                1500,
                DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
                DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));

        final RequestQueue requestQueue = Volley.newRequestQueue(this,
hurlStack);

        requestQueue.add(jsonObjectRequest);


    }


    private HostnameVerifier getHostnameVerifier() {
        return new HostnameVerifier() {
            @Override
            public boolean verify(String hostname, SSLSession session) {
                return true;
            }
        };
    }


    private TrustManager[] getWrappedTrustManagers(TrustManager[]
trustManagers) {
        final X509TrustManager originalTrustManager = (X509TrustManager)
trustManagers[0];
        return new TrustManager[]{
                new X509TrustManager() {
                    public X509Certificate[] getAcceptedIssuers() {
                        return originalTrustManager.getAcceptedIssuers();
                    }

                    public void checkClientTrusted(X509Certificate[] certs,
String authType) {
                        try {
                            if (certs != null && certs.length > 0){
                                certs[0].checkValidity();
                            } else {

originalTrustManager.checkClientTrusted(certs, authType);
                            }
                        } catch (CertificateException e) {
                            Log.w("checkClientTrusted", e.toString());
                        }
                    }

                    public void checkServerTrusted(X509Certificate[] certs,
String authType) {
                        try {
                            if (certs != null && certs.length > 0){
                                certs[0].checkValidity();
```

```
                            } else {

originalTrustManager.checkServerTrusted(certs, authType);
                            }
                        } catch (CertificateException e) {
                            Log.w("checkServerTrusted", e.toString());
                        }
                    }
                }
        };
    }

    private SSLSocketFactory getSSLSocketFactory()
            throws CertificateException, KeyStoreException, IOException,
NoSuchAlgorithmException, KeyManagementException {
        CertificateFactory cf = CertificateFactory.getInstance("X.509");
        InputStream caInput =
getResources().openRawResource(R.raw.certificadohttps); // this cert file
stored in \app\src\main\res\raw folder path

        Certificate ca = cf.generateCertificate(caInput);
        caInput.close();

        KeyStore keyStore = KeyStore.getInstance("BKS");
        keyStore.load(null, null);
        keyStore.setCertificateEntry("ca", ca);

        String tmfAlgorithm = TrustManagerFactory.getDefaultAlgorithm();
        TrustManagerFactory tmf =
TrustManagerFactory.getInstance(tmfAlgorithm);
        tmf.init(keyStore);

        TrustManager[] wrappedTrustManagers =
getWrappedTrustManagers(tmf.getTrustManagers());

        SSLContext sslContext = SSLContext.getInstance("TLS");
        sslContext.init(null, wrappedTrustManagers, null);

        return sslContext.getSocketFactory();
    }

    public void parseData(String response) {

        try {
            JSONObject jsonObject = new JSONObject(response);
            if (jsonObject.getString("respuesta").equals("1")) {

                ip = jsonObject.getString("ip");
                puerto = jsonObject.getString("puerto");
                Log.d("ippuerto",ip+" "+puerto);
                bandera = 1;
            }
            else {
                bandera = 0;
            }


        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}
```

- Datos.java

```
package com.dam.inicio;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.widget.EditText;
import android.widget.TextView;

import com.android.volley.DefaultRetryPolicy;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;

public class Datos extends AppCompatActivity {


    private String ip;
    private String puerto;
    private String url;
    private String link;

    ArrayList<Botones> arrayList = new ArrayList<Botones>();

    ArrayList<String> arrayNombre = new ArrayList<>();
    ArrayList<String> arrayLink = new ArrayList<>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_datos);
        Intent intent = getIntent();
        ip = intent.getStringExtra("ip");
        puerto = intent.getStringExtra("puerto");

        url="http://"+ip+":"+puerto+"/rest/sitemaps";
        rest(url);

    }


    public void rest(String ur) {
        final StringRequest jsonObjectRequest = new
StringRequest(Request.Method.GET, ur, new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                try {
                    Log.d("EUREKA",response);
                    parseData(response);
                } catch (Exception e) {
```

```
                    Log.d("ERROR1",e.toString());
                }
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Log.d("ERROR2",error.toString());
            }
        });

        jsonObjectRequest.setRetryPolicy(new DefaultRetryPolicy(
                1500,
                DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
                DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
        ColaVolley.getInstance().getRequestQueue().add(jsonObjectRequest);
    }


    public void parseData(String response) {

        try {

            JSONArray jsonObject = new JSONArray(response);
            for (int i = 0; i<jsonObject.length();i++) {
                JSONObject tabla = jsonObject.getJSONObject(i);
                String name = tabla.getString("name");
                Log.d("name", name);
                if(!name.equals("_default")) {
                    link = tabla.getString("link");
                    Log.d("link", link);
                    arrayNombre.add(name);
                    arrayLink.add(link);
                }

            }
            Intent casa = new Intent(this,Casa.class);
            casa.putExtra("nombre",arrayNombre);
            casa.putExtra("link",arrayLink);
            casa.putExtra("ip",ip);
            casa.putExtra("puerto",puerto);
            startActivity(casa);
        } catch (JSONException e) {
            e.printStackTrace();
        }

    }


    public void onRestart() {
        Log.d("Ciclo_datos", "Función onRestart");
        super.onRestart();
        Intent intents = new Intent(this,MainActivity.class);
        startActivity(intents);
    }

}
```

- Casa.java

```
package com.dam.inicio;

import android.content.ContentResolver;
import android.content.ContentValues;
import android.content.Intent;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.TextView;


import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;

public class Casa extends AppCompatActivity {

    private static final Uri URI_CP =
Uri.parse("content://com.contentprovider.elementos/elementos");
    ArrayList<String> arrayNombre = new ArrayList<>();
    ArrayList<String> arrayLink = new ArrayList<>();
    String ip;
    String puerto;
    ContentResolver CR;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_casa);
        CR = getContentResolver();

        Intent intent = getIntent();
        arrayLink = intent.getStringArrayListExtra("link");
        arrayNombre = intent.getStringArrayListExtra("nombre");
        ip = intent.getStringExtra("ip");
        puerto = intent.getStringExtra("puerto");

        LinearLayout lineal = (LinearLayout) findViewById(R.id.linear);

        for(int i = 0; i<arrayNombre.size(); i++) {
            TextView textView = new TextView(this);
            textView.setId(i);
            textView.setText(arrayNombre.get(i));
            textView.setTextSize(18);
            textView.setGravity(Gravity.CENTER_HORIZONTAL);
```

```
            Log.d("id",Integer.toString(textView.getId()));

            textView.setOnClickListener(new View.OnClickListener()
            {
                public void onClick(View v)
                {
                    rest(arrayLink.get(v.getId()));
                }
            });
            lineal.addView(textView);
        }

    }

    @Override
    public void onStart() {
        Log.d("OnStart","ONSTART");
        super.onStart();
        CR.delete(URI_CP,null,null);
    }

    public void rest(String ur) {
        final StringRequest jsonObjectRequest = new
StringRequest(Request.Method.GET, ur, new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                try {
                    Log.d("EUREKA",response);
                    parseData(response);
                } catch (Exception e) {
                    Log.d("ERROR1",e.toString());
                }
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Log.d("ERROR2",error.toString());
            }
        });
        ColaVolley.getInstance().getRequestQueue().add(jsonObjectRequest);
    }

    public void parseData(String response) {

        try {
            int z = 1;
            JSONObject jsonObject = new JSONObject(response);

            JSONObject a = jsonObject.getJSONObject("homepage");
            String title = a.getString("title");
            JSONArray widgets = a.getJSONArray("widgets");

                for (int i = 0; i < widgets.length(); i++) {
                    JSONObject a1 = widgets.getJSONObject(i);
                    String piso = a1.getString("label");
                    Log.d("a.label", piso);
                    JSONArray widgets1 = a1.getJSONArray("widgets");
                    for (int t = 0; t < widgets1.length(); t++) {
                        JSONObject a2 = widgets1.getJSONObject(t);
                        String habitacion = a2.getString("label");
                        Log.d("a2.label",habitacion);
                        JSONArray widgets2 = a2.getJSONArray("widgets");
```

```
                        for (int t1 = 0; t1 < widgets2.length(); t1++) {
                            JSONObject aa = widgets2.getJSONObject(t1);
                            JSONObject item = aa.getJSONObject("item");
                            String name = item.getString("name");
                            String type = aa.getString("type");
                            CR.insert(URI_CP, setVALORES(z, name, type, piso,
title, habitacion));
                            Log.d("id", Integer.toString(z));
                            Log.d("name", name);
                            Log.d("tipo", type);
                            Log.d("piso", piso);
                            Log.d("titulo", title);
                            Log.d("habitacion", habitacion);

                            z = z + 1;
                        }
                    }
                }

            Intent intent1 = new Intent(getApplicationContext(),
Habitacion.class);
            intent1.putExtra("casa",title);
            intent1.putExtra("ip",ip);
            intent1.putExtra("puerto",puerto);
            startActivity(intent1);

        } catch (JSONException e) {
            e.printStackTrace();
        }
    }

    private ContentValues setVALORES(int id, String nombre,
                                     String tipo,
                                     String piso,
                                     String casa,
                                     String habitacion) {
        ContentValues valores = new ContentValues();
        if(id!=0)
            valores.put("_id", id);
        valores.put("nombre", nombre);
        valores.put("tipo", tipo);
        valores.put("piso", piso);
        valores.put("casa", casa);
        valores.put("habitacion", habitacion);
        return valores;
    }
}
```

- Habitacion.java

```
package com.dam.inicio;

import android.content.ContentResolver;
import android.content.Intent;
import android.database.Cursor;
import android.database.CursorIndexOutOfBoundsException;
import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.ScrollView;
import android.widget.SeekBar;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;

import com.android.volley.AuthFailureError;
import com.android.volley.DefaultRetryPolicy;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

public class Habitacion extends AppCompatActivity {
    private static final Uri URI_CP =
Uri.parse("content://com.contentprovider.elementos/elementos");
    String where = "";
    String nombre_casa = "";
    String ip;
    String puerto;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Intent intent = getIntent();
        nombre_casa = intent.getStringExtra("casa");
        ip = intent.getStringExtra("ip");
        puerto = intent.getStringExtra("puerto");
        ArrayList<Botones> datos = recuperarBotones();
        LinearLayout linear = new LinearLayout(this);
        linear.setOrientation(LinearLayout.VERTICAL);
```

```
linear.setBackgroundColor(getResources().getColor(R.color.colorPrimary));
        setContentView(linear);
        ScrollView scroll =new ScrollView(this);
        linear.addView(scroll);
        LinearLayout linearLayout = new LinearLayout(this);
        //setContentView(linearLayout);
        linearLayout.setOrientation(LinearLayout.VERTICAL);

linearLayout.setBackgroundColor(getResources().getColor(R.color.colorPrimary)
);
        scroll.addView(linearLayout);
        String antiguo_piso = "";
        String antigua_habitacion="";

        for(int i = 0; i<datos.size();i++) {

            Botones bot = datos.get(i);
            String nuevo_piso = bot.getPiso();

            String nueva_habitacion = bot.getHabitacion();

            if(!nuevo_piso.equals(antiguo_piso)) {
                TextView pis = new TextView(this);
                pis.setText(bot.getPiso());
                pis.setTextSize(22);
                pis.setPadding(8,32,12,0);
                linearLayout.addView(pis);
                antiguo_piso = nuevo_piso;
            }
            if(!nueva_habitacion.equals(antigua_habitacion)) {
                TextView hab = new TextView(this);
                hab.setText(bot.getHabitacion());
                hab.setTextSize(20);
                hab.setPadding(15,32,12,0);
                linearLayout.addView(hab);
                antigua_habitacion=nueva_habitacion;
            }

            switch (bot.getTipo()) {

                case "Switch":
                    final TextView tv = new TextView(this);
                    tv.setText(bot.getNombre());
                    tv.setTextSize(18);
                    tv.setPadding(52,8,12,8);
                    linearLayout.addView(tv);

                    final Switch sw = new Switch(this);
                    LinearLayout.LayoutParams rlp = new
LinearLayout.LayoutParams(

ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP_CONTENT);
                    rlp.setMargins(200,0,30,8);
                    sw.setLayoutParams(rlp);
                    sw.setText("OFF");

                    linearLayout.addView(sw);

                    sw.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
                        @Override
                        public void onCheckedChanged(CompoundButton
```

```
buttonView, boolean isChecked) {
                            String msg = isChecked ? "ON" : "OFF";
                            sw.setText(msg);
                            String nombre = tv.getText().toString();
                            post(msg,nombre);

                        }
                    });

                    break;
                case "Slider":
                    final TextView tv1 = new TextView(this);
                    tv1.setText(bot.getNombre());
                    tv1.setTextSize(18);
                    tv1.setPadding(52,8,12,8);
                    linearLayout.addView(tv1);
                    final TextView value = new TextView(this);
                    value.setText("50");
                    value.setPadding(52,8,12,0);
                    linearLayout.addView(value);
                    LinearLayout.LayoutParams llp = new
LinearLayout.LayoutParams(

ViewGroup.LayoutParams.FILL_PARENT,ViewGroup.LayoutParams.WRAP_CONTENT);
                    llp.setMargins(20,0,20,8);
                    SeekBar seek = new SeekBar(this);
                    seek.setProgress(50);
                    seek.setLayoutParams(llp);
                    linearLayout.addView(seek);



                    seek.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
                        String se = "";
                        @Override
                        public void onProgressChanged(SeekBar seekBar, int
progress, boolean fromUser) {
                            se = String.valueOf(progress);
                            value.setText(se);

                        }

                        @Override
                        public void onStartTrackingTouch(SeekBar seekBar) {
                        }

                        @Override
                        public void onStopTrackingTouch(SeekBar seekBar) {
                            post(se,tv1.getText().toString());
                        }
                    });
                    break;
                case "Text":
                    TextView tv2 = new TextView(this);
                    tv2.setText(bot.getNombre());
                    tv2.setTextSize(18);
                    tv2.setPadding(52,8,12,8);
                    linearLayout.addView(tv2);
                    TextView val = new TextView(this);
                    val.setText(bot.getNombre());
                    val.setTextSize(18);
```

```
                    val.setPadding(200,0,30,8);
                    linearLayout.addView(val);
                    break;
                case "Setpoint":
                    TextView tv3 = new TextView(this);
                    tv3.setText(bot.getNombre());
                    tv3.setTextSize(18);
                    tv3.setPadding(52,8,12,8);
                    linearLayout.addView(tv3);
                    LinearLayout linearLayout2 = new LinearLayout(this);
                    linearLayout2.setOrientation(LinearLayout.HORIZONTAL);
                    LinearLayout.LayoutParams slp = new
LinearLayout.LayoutParams(

ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP_CONTENT);

                    Button mas = new Button(this);
                    mas.setText("+");
                    mas.setTextSize(18);
                    slp.width = 120;
                    slp.height = 120;
                    slp.setMargins(20,8,8,8);

mas.setBackgroundColor(getResources().getColor(R.color.Celeste));
                    mas.setLayoutParams(slp);
                    linearLayout2.addView(mas);
                    final TextView va = new TextView(this);
                    va.setText("22");
                    va.setTextSize(18);
                    va.setPadding(20,8,8,8);
                    linearLayout2.addView(va);
                    Button menos = new Button(this);
                    menos.setText("-");
                    menos.setTextSize(18);
menos.setBackgroundColor(getResources().getColor(R.color.Celeste));
                    menos.setLayoutParams(slp);
                    linearLayout2.addView(menos);
                    linearLayout.addView(linearLayout2);
                    mas.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                            int a = Integer.valueOf(va.getText().toString());
                            a = a+1;
                            va.setText(Integer.toString(a));
                        }
                    });
                    menos.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                            int b = Integer.valueOf(va.getText().toString());
                            b = b-1;
                            va.setText(Integer.toString(b));
                        }
                    });
                    break;
                            }
        }
    }

    public ArrayList<Botones> recuperarBotones() {
        ContentResolver CR = getContentResolver();
        Cursor c;
```

```
        ArrayList<Botones> lista_botones = new ArrayList<Botones>();
        String[] valores_recuperar = {"_id", "nombre",
                "tipo","piso","casa","habitacion"};
        try {
            where = "casa='"+nombre_casa+"'";
            c = CR.query(URI_CP, valores_recuperar, where, null, null);
            where = "";
            c.moveToFirst();
            do {
                Botones botones = new Botones(c.getInt(0),
c.getString(1),c.getString(2),c.getString(3),c.getString(4),c.getString(5));
                lista_botones.add(botones);
            } while (c.moveToNext());
        } catch (CursorIndexOutOfBoundsException e) {
            Toast.makeText(getApplicationContext(), "No existen elementos
asociados a esta casa", Toast.LENGTH_LONG).show();
            finish();
        }
        return lista_botones;
    }

    public void post(final String mensaje, String nombre) {
        Log.d("mensaje",mensaje);

        String url = "http://"+ip+":"+puerto+"/rest/items/"+nombre;

        final StringRequest jsonObjectRequest = new
StringRequest(Request.Method.POST, url, new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                try {

                    Log.d("EUREKA",response.toString());

                } catch (Exception e) {

                    Log.d("ERROR1",e.toString());
                }
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Log.d("ERROR2",error.toString());
            }
        }){
            public byte[] getBody() throws AuthFailureError {
                String httpPostBody=mensaje;

                return httpPostBody.getBytes();
            }
            public String getBodyContentType()
            {
                return "text/plain; charset=utf-8";
            }
        };

        ColaVolley.getInstance().getRequestQueue().add(jsonObjectRequest);
    }
}
```

- Botones.java

```
package com.dam.inicio;

public class Botones {

    private String tipo;
    private String nombre;
    private String link;
    private int id;
    private int id_piso;
    private String piso;
    private String title;
    private String value;
    private String habitacion;

    public Botones(int id, String nombre, String tipo, String piso, String
title, String habitacion) {
        this.tipo = tipo;
        this.nombre = nombre;
        this.id = id;
        this.piso = piso;
        this.title = title;
        this.habitacion = habitacion;
    }

    public String getTipo() {
        return tipo;
    }

    public void setTipo(String tipo) {
        this.tipo = tipo;
    }

    public String getValue() {
        return value;
    }

    public void setValue(String value) {
        this.value = value;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getLink() {
        return link;
    }

    public void setLink(String link) {
        this.link = link;
    }

    public String getPiso() {
        return piso;
    }
```

```
    public void setTitle(String title) {
        this.title = title;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int setIdPiso() {
        return id_piso;
    }

    public void setIdPiso(int id_piso) {
        this.id_piso = id_piso;
    }

    public String getHabitacion() { return habitacion;}

    public void setHabitacion(String habitacion)
{this.habitacion=habitacion;}

}
```

- MiBaseDatos.java

```
package com.dam.inicio;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class MiBaseDatos extends SQLiteOpenHelper {
    private static final String TABLA_ELEMENTOS = "CREATE TABLE elementos " +
            "(_id INTEGER PRIMARY KEY, nombre TEXT," +
            " tipo TEXT," +
            " piso TEXT," +
            " casa TEXT," +
            "habitacion TEXT)";
    public MiBaseDatos(Context context) {
        super(context, "mibasedatos.db", null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(TABLA_ELEMENTOS);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
{
        db.execSQL("DROP TABLE IF EXISTS TABLA_ELEMENTOS");
        onCreate(db);
    }
    public void delete(SQLiteDatabase db) {
        db.execSQL("delete from elementos");
    }
}
```

- MiContentProvider.java

```
package com.dam.inicio;

import android.content.ContentProvider;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.util.Log;

public class MiContentProvider extends ContentProvider {

    private MiBaseDatos MBD;
    SQLiteDatabase SQLDB;
    private static final String NOMBRE_CP = "com.contentprovider.elementos";

    private static final int ELEMENTOS = 1;
    private static final int ELEMENTOS_ID = 2;
    private static final UriMatcher uriMatcher = new
UriMatcher(UriMatcher.NO_MATCH);
    static{
        uriMatcher.addURI(NOMBRE_CP, "elementos", ELEMENTOS);
        uriMatcher.addURI(NOMBRE_CP, "elementos/#", ELEMENTOS_ID);
    }

    @Override
    public String getType(Uri uri) {
        return null;
    }

    @Override
    public boolean onCreate() {
        MBD = new MiBaseDatos(getContext());
        return true;
    }

    @Override
    public Uri insert(Uri uri, ContentValues values) {
        long registro = 0;
        try {
            if (uriMatcher.match(uri) == ELEMENTOS) {
                SQLDB = MBD.getWritableDatabase();
                registro = SQLDB.insert("elementos", null, values);
            }
        } catch (IllegalArgumentException e) {
            Log.e("ERROR", "Argumento no admitido: " + e.toString());
        }

        // Comprobar si se inserto bien el registro
        if (registro > 0) {
            Log.e("INSERT", "Registro creado correctamente");
        } else {
            Log.e("Error", "Al insertar registro: " + registro);
        }

        return Uri.parse("elementos/" + registro);
    }

    @Override
```

```
    public int update(Uri uri, ContentValues values, String selection,
                      String[] selectionArgs) {
        String id = "";
        try {
            if (uriMatcher.match(uri) == ELEMENTOS_ID) {
                id = uri.getLastPathSegment();
                SQLDB = MBD.getWritableDatabase();
                SQLDB.update("elementos", values, "_id=" + id,
selectionArgs);
            }
        } catch (IllegalArgumentException e) {
            Log.e("ERROR", "Argumento no admitido: " + e.toString());
        }

        return Integer.parseInt(id);
    }

    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        int registro = 0;
        try {
            if (uriMatcher.match(uri) == ELEMENTOS) {
                SQLDB = MBD.getWritableDatabase();
                SQLDB.execSQL("DELETE FROM ELEMENTOS");
            }
        } catch (IllegalArgumentException e) {
            Log.e("ERROR", "Argumento no admitido: " + e.toString());
        }
        return registro;
    }

    @Override
    public Cursor query(Uri uri, String[] projection, String selection,
                        String[] selectionArgs, String sortOrder) {
        Cursor c = null;
        try {
            switch (uriMatcher.match(uri)) {
                case ELEMENTOS_ID:
                    String id = "_id=" + uri.getLastPathSegment();
                    SQLDB = MBD.getReadableDatabase();
                    c = SQLDB.query("elementos", projection, id,
selectionArgs,
                            null, null, null, sortOrder);
                    Log.d("ID","Elementos_id");
                    break;
                case ELEMENTOS:
                    SQLDB = MBD.getReadableDatabase();
                    c = SQLDB.query("elementos", projection, selection,
selectionArgs,
                            null, null, null, sortOrder);
                    Log.d("ID","Elementos");
                    break;
            }
        } catch (IllegalArgumentException e) {
            Log.e("ERROR", "Argumento no admitido: " + e.toString());
        }

        return c;
    }

}
```

- ColaVolley.java

```
package com.dam.inicio;

import android.app.Application;

import com.android.volley.RequestQueue;
import com.android.volley.toolbox.Volley;

//Clase que crea la cola Volley al iniciarse la aplicacion
public class ColaVolley extends Application {
    private static ColaVolley sInstance;
    private RequestQueue mRequestQueue;
    @Override
    public void onCreate() {
        super.onCreate();
        mRequestQueue = Volley.newRequestQueue(this);
        sInstance = this;
    }
    public synchronized static ColaVolley getInstance() {
        return sInstance;
    }
    public RequestQueue getRequestQueue() {
        return mRequestQueue;
    }
}
```

- activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorPrimary">

<android.support.constraint.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorPrimary"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imageView3"
        android:layout_width="140dp"
        android:layout_height="140dp"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="32dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/casa5" />
```

```
    <EditText
        android:id="@+id/editText3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginLeft="32dp"
        android:layout_marginTop="32dp"
        android:layout_marginEnd="32dp"
        android:layout_marginRight="32dp"
        android:ems="10"
        android:hint="Email"
        android:inputType="textPersonName"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView3"
        />

    <EditText
        android:id="@+id/editText4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginLeft="32dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="32dp"
        android:layout_marginRight="32dp"
        android:ems="10"
        android:hint="Password"
        android:inputType="textPassword"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editText3" />

      <Button
          android:id="@+id/button2"
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:layout_marginStart="32dp"
          android:layout_marginLeft="32dp"
          android:layout_marginTop="8dp"
          android:layout_marginEnd="32dp"
          android:layout_marginRight="32dp"
          android:text="Login"
          android:onClick="clickLogin"
          app:layout_constraintEnd_toEndOf="parent"
          app:layout_constraintStart_toStartOf="parent"
          app:layout_constraintTop_toBottomOf="@+id/editText4"
          android:background="@drawable/button_rounded"/>

</android.support.constraint.ConstraintLayout>
</ScrollView>
```

- activity_datos.xml

```
<?xml version="1.0" encoding="utf-8"?>

<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorPrimary"
    tools:context=".Datos">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="32dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginBottom="32dp"
        android:text="Conectando con el Servidor..."
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

- activity_casa.xml

```
<?xml version="1.0" encoding="utf-8"?>

<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="@color/colorPrimary"
tools:context=".Casa">

<LinearLayout
    android:id="@+id/linear"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="32dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
```

```
        android:layout_marginBottom="32dp"
        android:gravity="center_horizontal"
        android:text="Viviendas"
        android:textSize="30sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</LinearLayout>

</android.support.constraint.ConstraintLayout>
```

- activity_habitacion.xml

```
<?xml version="1.0" encoding="utf-8"?>

<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorPrimary"
    tools:context=".Habitacion">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="32dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginBottom="32dp"
        android:text="Habitación"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

1.2. Servicio Spring

- pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.or
g/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.springframework</groupId>
    <artifactId>gs-rest-service</artifactId>
    <version>0.1.0</version>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.5.RELEASE</version>
    </parent>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>


        <dependency>
            <groupId>com.jayway.jsonpath</groupId>
            <artifactId>json-path</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <version>9.3-1100-jdbc41</version>
        </dependency>
        <dependency>
            <groupId>io.springfox</groupId>
                <artifactId>springfox-swagger2</artifactId>
                <version>2.9.2</version>
        </dependency>
        <dependency>
```

```
                <groupId>io.springfox</groupId>
                <artifactId>springfox-swagger-ui</artifactId>
                <version>2.9.2</version>
        </dependency>
    </dependencies>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

- application.properties

```
server.port=8443
server.ssl.key-alias=https
server.ssl.key-store-type=JKS
server.ssl.key-password=password
server.ssl.key-store=classpath:alm-https.jks
```

- Appliaction.java

```
package service;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;


@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

- Respuesta.java

```
package service;

import java.util.Collections;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.bind.annotation.RequestMethod;

import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.builders.ResponseMessageBuilder;
import springfox.documentation.schema.ModelRef;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import io.swagger.annotations.*;

@ApiModel("Clase Respuesta")
public class Respuesta {

    @ApiModelProperty(value = "respuesta correcta", required=true)
    private int respuesta;

    @ApiModelProperty(value = "ip servidor openHAB", required=true)
    private String ip;

    @ApiModelProperty(value = "puerto servidor openHAB", required=true)
    private String puerto;


    public Respuesta(int respuesta, String ip, String puerto) {
        this.respuesta = respuesta;
        this.ip = ip;
        this.puerto = puerto;
    }

    public int getRespuesta() {
        return respuesta;
    }
```

```
    public void setRespuesta(int respuesta) {
        this.respuesta = respuesta;
    }

    public String getIp() {
        return ip;
    }

    public void setIp (String ip) {
        this.ip = ip;
    }

    public String getPuerto() {
        return puerto;
    }

    public void setPuerto(String puerto) {
        this.puerto = puerto;
    }
}
```

- RespuestaController.java

```
package service;

import java.util.concurrent.atomic.AtomicLong;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.context.annotation.Bean;
import org.springframework.web.bind.annotation.GetMapping;

import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.builders.ResponseMessageBuilder;
import springfox.documentation.schema.ModelRef;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

import java.io.*;
import java.rmi.*;
import java.rmi.server.*;
```

```
import java.util.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import io.swagger.annotations.*;


@RestController
@RequestMapping("/login")
@Api(value = "Microservicio Login", description = "Esta API nos permite dar acce
so al sistema a un usuario")
public class RespuestaController {

    @GetMapping
    public String hello() {
        return "Hola guapa";
    }

    @RequestMapping(method = RequestMethod.POST)
    @ApiOperation(value = "Login", notes = "Nos devuelve la ip y el puerto asoci
ado al servidor local del usuario" )
    public Respuesta post(@RequestParam(value="email",defaultValue="0") String e
mail, @RequestParam(value="password",defaultValue="0") String password) {
        Connection c = null;
    Statement st = null;
    int respuesta = 1;
    String ip = "";
    String puerto = "";
    try {
        System.out.println(email+" "+ password);
        Class.forName("org.postgresql.Driver");
            c = DriverManager.getConnection("jdbc:postgresql://localhost/dit","d
it","dit");

        st = c.createStatement();
        String query = "select * from respuesta where email = '"+email+"' and pa
sswd = '"+password+"'";
            ResultSet rs = st.executeQuery(query);
            System.out.println(rs.toString());
        if (rs.next() == false) {
                respuesta=0;
            }
```

```
            else {
                respuesta = 1;
                ip = rs.getString("ip");
                puerto = rs.getString("puerto");
            }
           rs.close();
           st.close();
           c.close();

    } catch (Exception e) {
        respuesta = 0;
        e.printStackTrace();
    }

    if (respuesta == 0) {
        return new Respuesta(respuesta,"ERROR","ERROR");
    }
    else {
        return new Respuesta(respuesta,ip,puerto);
    }
    }

    @RequestMapping(method = RequestMethod.PUT)
    @ApiOperation(value = "Inserta Nuevo Usuario", notes = "Nos devuelve si la i
nsercción ha sido correcta o no" )
    public int put(@RequestParam(value="email",defaultValue="0") String email, @
RequestParam(value="password",defaultValue="0") String password, @RequestParam(v
alue="ip",defaultValue="0") String ip, @RequestParam(value="puerto",defaultValue
="0") String puerto) {

        Connection c = null;
    Statement st = null;
    int bandera = 0;

        try {
            System.out.println(email+" "+ password+" "+ip+" "+puerto);
        Class.forName("org.postgresql.Driver");
            c = DriverManager.getConnection("jdbc:postgresql://localhost/dit","d
it","dit");

                st = c.createStatement();

                String query = "insert into respuesta values('"+email+"','"+pass
word+"','"+ip+"','"+puerto+"');";
                st.execute(query);
                st.close();
                c.close();
                bandera = 1;
```

```
        } catch (Exception e) {
                e.printStackTrace();
                bandera = 0;
            //log.info(e.getMessage());
                System.exit(0);
        }

        return bandera;

    }
}
```

- SwaggerConfiguration.java

```
package service;

import com.google.common.base.Predicate;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import springfox.documentation.builders.ApiInfoBuilder;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;
import springfox.documentation.spring.web.plugins.ApiSelectorBuilder;
import static springfox.documentation.builders.PathSelectors.regex;
import java.util.Collections;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import org.springframework.web.bind.annotation.RequestMethod;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.ResponseMessageBuilder;
import springfox.documentation.schema.ModelRef;
import springfox.documentation.service.Contact;
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import io.swagger.annotations.*;


@EnableSwagger2

@Configuration
```

```
public class SwaggerConfiguration {


    /**

     * Publish a bean to generate swagger2 endpoints

     * @return a swagger configuration bean

     */

    @Bean

    public Docket api() {

        return new Docket(DocumentationType.SWAGGER_2)



                .select()

                .paths(PathSelectors.ant("/login/*"))

                .apis(RequestHandlerSelectors.basePackage("service"))


                .build()
        .apiInfo(apiInfor());

                //.useDefaultResponseMessages(false);

    }

    private ApiInfo apiInfor() {
            return new ApiInfo(
            "API REST LOGIN",
            "Some custom description of API.",
            "API REST",
            "Terms of service",
            new Contact("José de Lózar Alameda", "TFG", "jose.lozar.alameda@gmai
l.com"),
            "License of API by US", "API license URL", Collections.emptyList());


    }

}
```

1.3. Cliente Spring

- pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.springframework</groupId>
    <artifactId>gs-consuming-rest</artifactId>
    <version>0.1.0</version>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.5.RELEASE</version>
    </parent>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
        </dependency>
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
        </dependency>
        <!-- we need httpclient here for client certificate handling -->
    <dependency>
        <groupId>org.apache.httpcomponents</groupId>
        <artifactId>httpclient</artifactId>
    </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

- Application.java

```
package service;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.client.RestTemplate;
import org.springframework.util.*;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

import java.util.HashMap;
import java.util.Map;

import org.springframework.http.HttpMethod;
import org.springframework.http.ResponseEntity;

import java.io.Console;

public class Application {

    private static final Logger log = LoggerFactory.getLogger(Application.class)
;
    private static String ip = "";
    private static String puerto = "";

    public static void main(String args[]) throws IOException {

    String email = "";
    String password1 = "";
    String password2 = "";

    String cad = "";


    RestTemplate restTemplate = new RestTemplate();
    MultiValueMap<String, Object> values = new LinkedMultiValueMap<String, Objec
t>();

    System.out.println("Introduzca email: ");
    Console terminal = System.console();
   email = System.console().readLine();
    System.out.println("Introduzca contraseña: ");
    char[] pw1 = terminal.readPassword();
```

```
    password1 = String.valueOf(pw1);
    System.out.println("Introduzca la contraseña de nuevo: ");
    char[] pw2 = terminal.readPassword();
    password2 = String.valueOf(pw2);
    cad = muestraContenido("/home/dit/Descargas/stunserver-master/a.txt");
    Parse(cad);
    System.out.println(ip+":"+puerto);
    if(password1.equals(password2)) {
        values.add("email", email);
        values.add("password", password1);
        values.add("ip", ip);
        values.add("puerto", puerto);
        restTemplate.put ( "https://www.josdeala.com:8443/login",values);
    } else {
        System.out.println("Ha introducido la contraseña erroneamente");
    }


    }

    public static String muestraContenido(String archivo) throws FileNotFoundExc
eption, IOException {

    String cadena;
      String result="";
      int bandera = 0;
      int flag = 0;
      FileReader f = new FileReader(archivo);
      BufferedReader b = new BufferedReader(f);
      while((cadena=b.readLine())!=null) {
        for(int i=0;i<cadena.length();i++) {
                if((cadena.charAt(i)==':' || cadena.charAt(i)=='.') && bandera =
= 1 && flag==0) {
                    result = result + cadena.charAt(i);
                }else{
                    if((Character.isDigit(cadena.charAt(i))) && flag ==0) {
                    result = result + cadena.charAt(i);
                        bandera = 1;
                    }
                    else {
                        if(bandera == 1)
                            flag =1;
                    }
                }
            }

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

```
    }

    public static void Parse(String c) {
        int band=0;
        for(int i=0;i<c.length();i++) {
            if(c.charAt(i)==':') {
                    band = 1;
            } else {
                if(band == 0) {
                    ip=ip+c.charAt(i);
                } else {
                    puerto=puerto+c.charAt(i);
                }
            }

        }
    }

}
```