Trabajo Fin de Grado

Grado en Ingeniería de las Tecnologías de Telecomunicación

Integración de OpenHAB y Home Connect en una aplicación Android para el control domótico de los electrodomésticos y la estimación de su consumo energético

Autor: Belén María Lozano Jurado

Tutor: María Teresa Ariza Gómez

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

Sevilla, 2023

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

Integración de OpenHAB y Home Connect en una aplicación Android para el control domótico de los electrodomésticos y la estimación de su consumo energético

Autor:

Belén María Lozano Jurado

Doctora:

María Teresa Ariza Gómez

Departamento de Ingeniería Telemática

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Grado: Integración de OpenHAB y Home Connect en una aplicación Android para el control domótico de los electrodomésticos y la estimación de su consumo energético

Autor: Belén María Lozano Jurado

Tutor: 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:

Sevilla, 2023

El Secretario del Tribunal

A mi familia y a mis amigos, por su apoyo incondicional.

Agradecimientos

Sin duda, el instante más significativo y emotivo en la creación de esta memoria se encuentra en este apartado. Quisiera expresar mi sincero agradecimiento a todas las personas que han contribuido de manera significativa en la realización de este proyecto. Su apoyo, orientación y colaboración han sido fundamentales para el éxito de esta iniciativa.

En primer lugar, quiero agradecer de corazón a mi tutora María Teresa Ariza Gómez su dedicación inquebrantable, su orientación experta y sus valiosos consejos a lo largo de todo este proyecto. Su compromiso ha sido fundamental para la realización exitosa de este proyecto.

No puedo dejar de expresar mi profunda gratitud hacia mis padres, hermano y abuela, así como a toda mi familia, cuyo apoyo incondicional ha sido mi roca durante todo este proceso. Sus palabras de aliento, comprensión y amor han sido mi fuente de motivación constante. Agradezco profundamente su paciencia y comprensión en los momentos en que mi atención estaba centrada en este proyecto. Su respaldo ha sido esencial en esta travesía.

Mi más sincero agradecimiento a mi compañero de vida, Alejandro, por ser mi punto de apoyo día tras día. Gracias por estar siempre ahí, escucharme, ayudarme y motivarme incluso en los momentos más complicados.

Gracias a mi querido amigo Juanjo, por ser uno de mis apoyos más importantes a lo largo de esta etapa universitaria. Te agradezco de corazón todos tus consejos, explicaciones y tu disposición para ayudar en todo momento. Gracias a mi amiga Carmen, por inspirarme y aconsejarme en el diseño gráfico de mi aplicación móvil. Y a todos mis amigos, gracias por vuestro apoyo y confianza.

Extiendo mi gratitud a todos mis profesores y compañeros por haberme ayudado durante la carrera.

Gracias a todos mis seres queridos que ya no están entre nosotros, porque sé que desde allá donde estéis me habéis mandado toda vuestra fuerza y apoyo.

Belén María Lozano Jurado

Sevilla, 2023

Resumen

En la actualidad, estamos inmersos en una era de avances tecnológicos que han transformado profundamente la forma en que interactuamos con nuestro entorno. El concepto de hogar inteligente ha ganado popularidad rápidamente, ofreciendo soluciones innovadoras para hacer nuestras vidas más cómodas y eficientes. El creciente uso de dispositivos móviles y la expansión de la conectividad a Internet han brindado nuevas oportunidades para controlar y gestionar nuestros electrodomésticos de manera remota.

En este contexto, el presente Trabajo de Fin de Grado se centra en el desarrollo de una aplicación para dispositivos Android que utiliza la API REST de OpenHAB para realizar peticiones a un sistema de automatización del hogar. Esta API, a su vez, redirige las peticiones hacia Home Connect, una plataforma que permite la comunicación con electrodomésticos compatibles con esta tecnología. Además, se ha implementado un servicio REST personalizado para llevar un seguimiento del consumo de energía de cada electrodoméstico y mostrar dicha información en la misma aplicación.

El objetivo principal de este proyecto es, por tanto, proporcionar a los usuarios una forma fácil y centralizada de controlar sus electrodomésticos que dispongan de la tecnología de Home Connect y monitorear su consumo de energía desde sus dispositivos móviles. Esto se ha logrado gracias a la integración entre OpenHAB y Home Connect.

Abstract

Currently, we are immersed in an era of technological advances that have profoundly transformed the way we interact with our environment. The smart home concept has rapidly gained popularity, offering innovative solutions to make our lives more comfortable and efficient. The increasing use of mobile devices and the expansion of internet connectivity have provided new opportunities to remotely control and manage our appliances.

In this context, this Final Degree Project focuses on the development of an Android application that utilizes OpenHAB's REST API to make requests to a home automation system. This API, in turn, redirects the requests to Home Connect, a platform that allows communication with appliances compatible with this technology. Additionally, a custom REST service has been implemented to track the energy consumption of each appliance and display that information in the same application.

The main goal of this project is, therefore, to provide users with an easy and centralized way to control their home appliances equipped with Home Connect technology and monitor their energy consumption from their mobile devices. This has been achieved through the integration between OpenHAB and Home Connect.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación y acrónimos

REST	Transferencia de Estado Representacional
API	Interfaz de Programación de Aplicaciones
SQL	Lenguaje de Consulta Estructurado
HD	Alta Definición
SSD	Unidad de Estado Sólido
XML	Lenguaje de Marcado Extensible
RAM	Memoria de Acceso Aleatorio
HTTP	Protocolo de Transferencia de Hipertexto
APT	Herramienta Avanzada de Gestión de Paquetes
JVM	Máquina Virtual de Java
TCP	Protocolo de Control de Transmisión
IP	Protocolo de Internet
URL	Localizador Uniforme de Recursos
MVC	Modelo Vista Controlador
BBDD	Base de Datos
IoT	Internet de las Cosas

1 INTRODUCCIÓN

El éxito es la suma de pequeños esfuerzos repetidos día tras día.

- Robert Collier-

1.1 Motivación

El presente proyecto surge de la creciente importancia de la automatización del hogar y la necesidad de promover la eficiencia energética en nuestra sociedad actual. Conscientes de la relevancia de este tema, este Trabajo de Fin de Grado se enfoca en el desarrollo de una aplicación moderna y práctica para dispositivos Android, que aproveche la interoperabilidad y flexibilidad de OpenHAB y la innovadora tecnología de Home Connect.

El objetivo principal de este proyecto es proporcionar a los usuarios una solución integral y práctica para controlar sus dispositivos del hogar a distancia y monitorear el consumo de los mismos desde la comodidad de sus dispositivos móviles. Esta aplicación no solo mejorará la calidad de vida de los usuarios al ofrecerles un control centralizado de sus electrodomésticos, sino que también promoverá la eficiencia energética al permitir un seguimiento detallado del consumo de energía de cada dispositivo, permitiendo al usuario optimizar su consumo y reducir costos. Este último aspecto adquiere una relevancia aún mayor en el contexto actual de inflación que supone gastos elevados al consumidor.

Además, esta investigación representa una oportunidad única para explorar, aprender y aplicar tecnologías de vanguardia, como el desarrollo de servicios REST y la integración de plataformas como OpenHAB y Home Connect. A nivel académico, este proyecto desempeñará un papel fundamental en mi crecimiento personal, permitiéndome adquirir y mejorar mis habilidades en áreas clave como la automatización del hogar, el diseño de aplicaciones móviles y la gestión del consumo energético.

Al abordar esta temática, se espera contribuir al conocimiento existente y ofrecer soluciones prácticas que beneficien tanto a los usuarios finales como al medio ambiente. Este proyecto representa una oportunidad para fusionar la innovación tecnológica con la preocupación por la sostenibilidad, estableciendo un precedente para futuras investigaciones y mejoras en el campo de la automatización del hogar.

1.2 Objetivos

En esta sección, se presentan los objetivos que se han establecido para el desarrollo de este proyecto:

- Realizar una investigación profunda sobre las tecnologías de Home Connect y OpenHAB, comprendiendo sus características, capacidades y requisitos de integración.
- Desarrollar una aplicación para dispositivos Android que permita a los usuarios controlar y monitorear sus electrodomésticos de forma remota.
- Implementar un servicio REST que permita llevar un seguimiento del consumo de energía de cada electrodoméstico y mostrar dicha información en la aplicación.

1.3 Antecedentes

Este proyecto se enmarca en una temática similar a la desarrollada por José de Lózar Alameda en su Trabajo de Fin de Grado titulado "Aplicación domótica en Android con OpenHAB para el control de los dispositivos del hogar" [1], bajo la tutela de María Teresa Ariza Gómez. Ambos proyectos se centran en el desarrollo de una aplicación Android que utiliza la API REST de OpenHAB para el control de dispositivos del hogar, aunque presentan diferencias clave.

A diferencia del proyecto de José de Lózar, el presente proyecto se centra en la implementación de una aplicación que permita el control y monitoreo de electrodomésticos compatibles con la tecnología Home Connect y en la creación de un servicio REST de consumo. Por otro lado, José de Lózar utilizó el protocolo MQTT para el control y monitoreo de sensores de temperatura y diodos led, y además desarrolló un servicio REST de autenticación de credenciales. Estas diferencias permiten ampliar las funcionalidades y enfoques en el ámbito de la automatización del hogar, ofreciendo nuevas posibilidades para el control de dispositivos y el seguimiento del consumo energético.

Al reconocer los antecedentes y trabajos previos relacionados con la temática, mi proyecto busca aportar avances y complementar las investigaciones anteriores, aprovechando las tecnologías actuales y explorando nuevas posibilidades para mejorar la automatización del hogar y la eficiencia energética.

1.4 Descripción de la solución

En este apartado, se presenta una descripción detallada de la solución desarrollada en este proyecto. La solución tiene como objetivo principal proporcionar a los usuarios un control centralizado y sencillo de sus dispositivos del hogar, así como un seguimiento detallado de su consumo de energía. A través de la integración de tecnologías como OpenHAB, Home Connect y el desarrollo de una aplicación para dispositivos Android, se ha creado una solución completa y versátil que aprovecha las ventajas de la automatización del hogar y la conectividad móvil. A lo largo de esta sección, se abordarán los aspectos clave de la solución, incluyendo su arquitectura y funcionalidades principales.

1.4.1 Objetivos Específicos

Los objetivos específicos de este proyecto son los siguientes:

- **Creación de una aplicación Android moderna e intuitiva:**

  Se desea que la aplicación móvil desarrollada tenga un diseño atractivo y disponga de retroalimentación para el usuario. Se hará uso de notificaciones y se mostrarán mensajes por pantalla para que el usuario sepa en todo momento el estado de sus acciones y pueda abortar alguna ejecución en caso de error. Además, se usarán *widgets* [1] que dotarán a la aplicación de interactividad con el usuario.

- **Instalación y configuración del servidor domótico de OpenHAB:**

  Se realizará la instalación del servidor de OpenHAB para establecer la base de la automatización del hogar. Esto incluye la configuración del *binding* [2] de Home Connect para la conexión de dispositivos compatibles con esta tecnología.

- **Conexión de la aplicación móvil con el servidor domótico de OpenHAB:**

  Se establecerá la comunicación entre la aplicación móvil y el servidor de OpenHAB para permitir el control remoto de los dispositivos del hogar. Esto incluye el envío de comandos y la recepción de actualizaciones de estado.

[1] Elementos visuales y funcionales que se utilizan para construir la interfaz de usuario, como botones, campos de texto, listas desplegables, casillas de verificación, barras de progreso, entre otros.
[2] Es un componente o extensión que permite la comunicación entre el sistema OpenHAB y dispositivos, servicios o tecnologías específicas.

- **Implementación de un servicio REST para el consumo energético de los electrodomésticos:**

  Se desarrollará un servicio REST personalizado para llevar un seguimiento del consumo energético de cada electrodoméstico. Este servicio permitirá registrar y consultar los datos de consumo de manera eficiente.

- **Despliegue y pruebas del servidor REST para el consumo:**

  Se llevará a cabo el despliegue del servidor REST y se realizarán pruebas exhaustivas para garantizar su correcto funcionamiento y rendimiento.

- **Conexión de la aplicación móvil con el servidor REST de consumo:**

  Se establecerá la comunicación entre la aplicación móvil y el servidor REST para obtener los datos de consumo energético de los electrodomésticos. La aplicación mostrará la información de manera clara y accesible para el usuario a través de diagramas de barras.

- **Almacenamiento y consulta del consumo energético de cada electrodoméstico:**

  Se implementará un sistema de almacenamiento de datos para registrar y almacenar el consumo energético de cada electrodoméstico. Además, se proporcionarán funcionalidades de consulta para que el usuario pueda acceder a su historial de consumo y realizar análisis de datos.

1.4.2 Funcionalidades

Las principales funcionalidades de este proyecto son las siguientes:

- **Control remoto de electrodomésticos:**

  La aplicación permitirá a los usuarios controlar de forma remota sus electrodomésticos compatibles con Home Connect. Concretamente, se podrá:

  - Encender/apagar el electrodoméstico.
  - Seleccionar el programa a ejecutar.
  - Arrancar/abortar la ejecución del programa.
  - Seleccionar algunas características como la temperatura o velocidad de lavado.
  - Recibir notificación cuando concluya el programa actual.

- **Monitoreo del consumo energético:**

  Los usuarios podrán ver información detallada sobre el consumo de energía de sus dispositivos y realizar un seguimiento de su uso.

- **Añadir, editar y eliminar nuevas habitaciones:**

  Se podrán incorporar nuevas habitaciones a la casa virtual, así como editarlas o eliminarlas de la base de datos, y se mostrarán de forma dinámica en el menú de habitaciones.

- **Listar los electrodomésticos disponibles en el servidor:**

  En la pantalla de cada habitación de la casa, se listan desde el servidor de OpenHAB los electrodomésticos que ya estén configurados para su uso.

1.4.3 Esquema de la arquitectura

En la figura 1-1, se presenta un esquema con la estructura general del proyecto, con el fin de facilitar la comprensión del mismo al lector. Podemos distinguir 4 elementos fundamentales:

- **Localhost** (Marco verde): formado por todos los componentes que se ejecutan en local en el portátil.
  a. **Emulador:** en este componente se ejecuta la aplicación Android desarrollada y se realizan las peticiones HTTP a los servidores de OpenHAB y de consumo. El emulador simula un dispositivo móvil en el cual se puede probar y ejecutar la aplicación. Destacar en este punto que para poder realizar peticiones a los servidores alojados en la dirección IP 127.0.0.1 (localhost), el emulador

utiliza la dirección de red especial 10.0.2.2. [2]

b. **Servidor de OpenHAB:** es una parte fundamental de la arquitectura, ya que se encarga de gestionar la comunicación con los electrodomésticos y proporcionar una interfaz para su control. Aquí se configuran los dispositivos y se definen las reglas y lógica de automatización.

c. **Servidor de consumo:** este componente se encarga de gestionar las solicitudes y respuestas relacionadas con el consumo energético de los electrodomésticos. Aquí se implementa el servicio REST que permite obtener datos de consumo y realizar acciones de control.

d. **Base de datos:** la base de datos es utilizada para almacenar de manera persistente la información relacionada con el consumo y el tiempo de uso de los electrodomésticos. Permite realizar consultas y análisis de los datos históricos.

- **Nube Home Connect** (Marco azul): este elemento representa la plataforma en la nube proporcionada por Home Connect [3]. Aquí se encuentran los servicios y APIs que permiten la comunicación con los electrodomésticos compatibles. Se compone de 2 dominios clave:

  a. **API Server Home Connect (api.home-connect.com):** este servidor actúa como intermediario entre el servidor de OpenHAB y los electrodomésticos físicos, permitiendo su control remoto y la obtención de información relevante.

  b. **API Simulator Home Connect (simulator.home-connect.com):** este servidor actúa como intermediario entre el servidor de OpenHAB y los electrodomésticos simulados, permitiendo su control remoto y la obtención de información relevante.

- **Electrodomésticos físicos:** estos son los dispositivos reales presentes en el hogar, como lavadoras, frigoríficos, lavavajillas, entre otros, que son compatibles con la tecnología de Home Connect. Estos electrodomésticos se conectan a la red local y se comunican con el API Server a través del dominio "api.home-connect.com".

- **Simulador web de electrodomésticos:** esta herramienta de simulación proporcionada por Home Connect permite emular y simular el comportamiento de los electrodomésticos. Facilita el desarrollo y las pruebas sin necesidad de tener los dispositivos físicos. El simulador web de electrodomésticos se arranca desde un navegador y se comunica con el servidor API Simulator a través del dominio "simulator.home-connect.com".

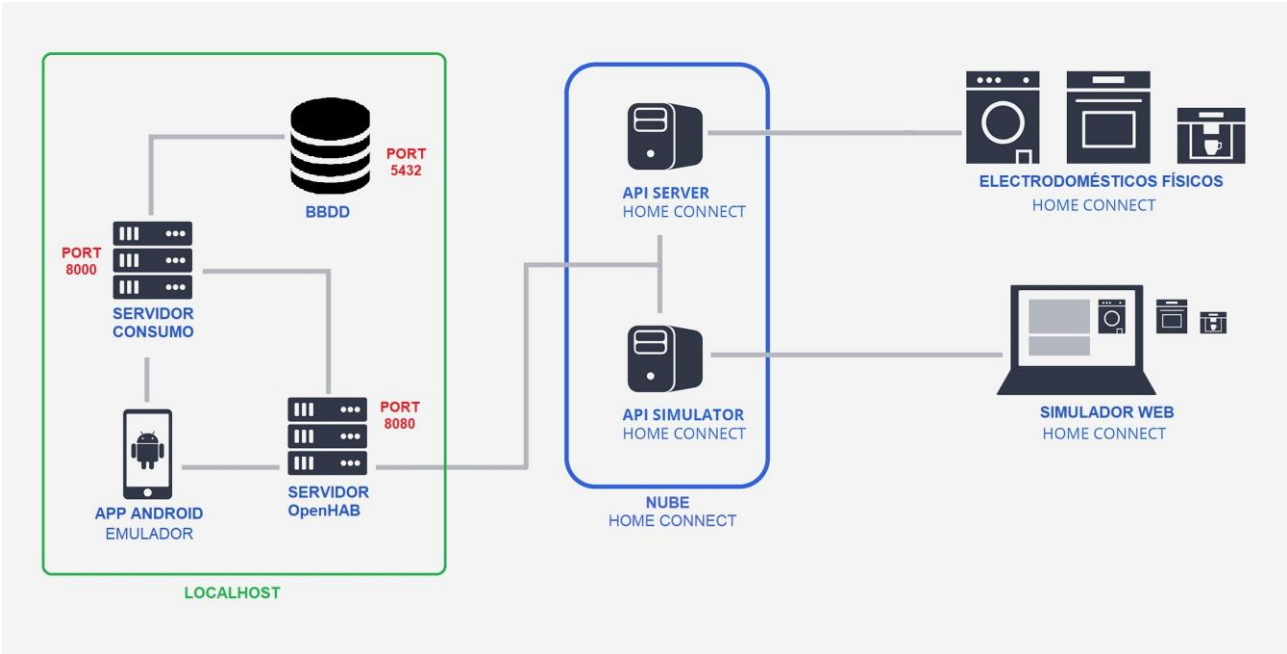

Figura 1-1. Arquitectura

1.5 Estructura de la memoria

En esta sección, se indica un breve resumen de lo que consta cada apartado clave de la presente memoria:

1. **Introducción:** se presentará la motivación detrás del proyecto, destacando la importancia de la automatización del hogar y la eficiencia energética en la sociedad actual. Se describirán los objetivos generales y específicos del proyecto, así como los antecedentes relevantes, las funcionalidades y un esquema general de la arquitectura para facilitar su comprensión al lector.
2. **Recursos utilizados:** este apartado incluirá un desglose y una breve explicación de los recursos hardware y las principales tecnologías software que se han usado en el proyecto.
3. **Estado del arte:** se realizará un análisis exhaustivo del estado del arte de las dos plataformas clave del proyecto, como son Home Connect y OpenHAB. Se investigará en detalle su definición, así como los procesos de configuración y uso. Además, se enumerarán las ventajas de integrar OpenHAB y Home Connect.
4. **Desarrollo del proyecto:** en este apartado, se indicará detalladamente el proceso seguido para la elaboración del proyecto. Se compondrá de 3 bloques claves:
    - **Aplicación móvil:** aquí se detallarán los aspectos clave de la aplicación móvil desarrollada. Se describirá la interfaz de usuario, las funcionalidades implementadas y su integración con los servidores REST de OpenHAB y consumo.
    - **Servicio REST de consumo:** se explicarán las características fundamentales del servicio REST implementado para el seguimiento del consumo energético. Se describirá la estructura del servicio, la comunicación con la base de datos y la seguridad adoptada en la implementación del servicio.
    - **Estimación consumo energético:** se abordará la utilización de las reglas de OpenHAB para llevar a cabo este cálculo de consumo energético. Se explicará cómo se realizará la división del consumo en tres tramos horarios: punta, llano y valle, permitiendo a los usuarios tener una comprensión clara y precisa de su patrón de consumo a lo largo del día.
5. **Conclusiones y líneas futuras:** en la última sección, se presentarán las conclusiones obtenidas a partir del desarrollo del proyecto. Se resaltarán los logros alcanzados, las lecciones aprendidas y las posibles mejoras o expansiones futuras.

2 RECURSOS UTILIZADOS

En este proyecto, se han utilizado diversos recursos hardware y software para lograr su correcta implementación.

2.1 Recursos Hardware

2.1.1 Portátil

Para llevar a cabo la implementación del proyecto, la realización de las pruebas, así como la redacción del presente escrito, se ha usado el portátil Dell Inspiron 15 5510 que podemos ver en la figura 2-1.

Figura 2-1. Portátil Dell Inspiron 15 5510

Sus principales características son las siguientes [4]:

- Pantalla de 15.6 pulgadas con resolución Full HD (1920 x 1080 píxeles).
- Procesador Intel Core i7 de undécima generación.
- Memoria RAM de 16 GB.
- Memoria SSD de almacenamiento de 1TB.
- Gráfica Intel iRIS X^e.

2.2 Recursos Software

2.2.1 Android Studio

Android Studio es un entorno de desarrollo integrado (IDE) ampliamente utilizado para crear aplicaciones móviles para dispositivos Android. Proporciona herramientas intuitivas y potentes que facilitan la escritura de código, la depuración, la creación de interfaces de usuario y la gestión de proyectos, lo que permite a los desarrolladores crear aplicaciones de alta calidad y optimizadas para la plataforma Android. [5]

Figura 2-2. Android Studio

2.2.2 Emulador del Google Pixel 4

Android Studio ofrece diversos emuladores y el elegido para este proyecto ha sido el Google Pixel 4 API 31. Este emulador replica las características hardware y software del dispositivo real, y permite probar y depurar las aplicaciones desarrolladas en el entorno de Android Studio de manera fácil e intuitiva.

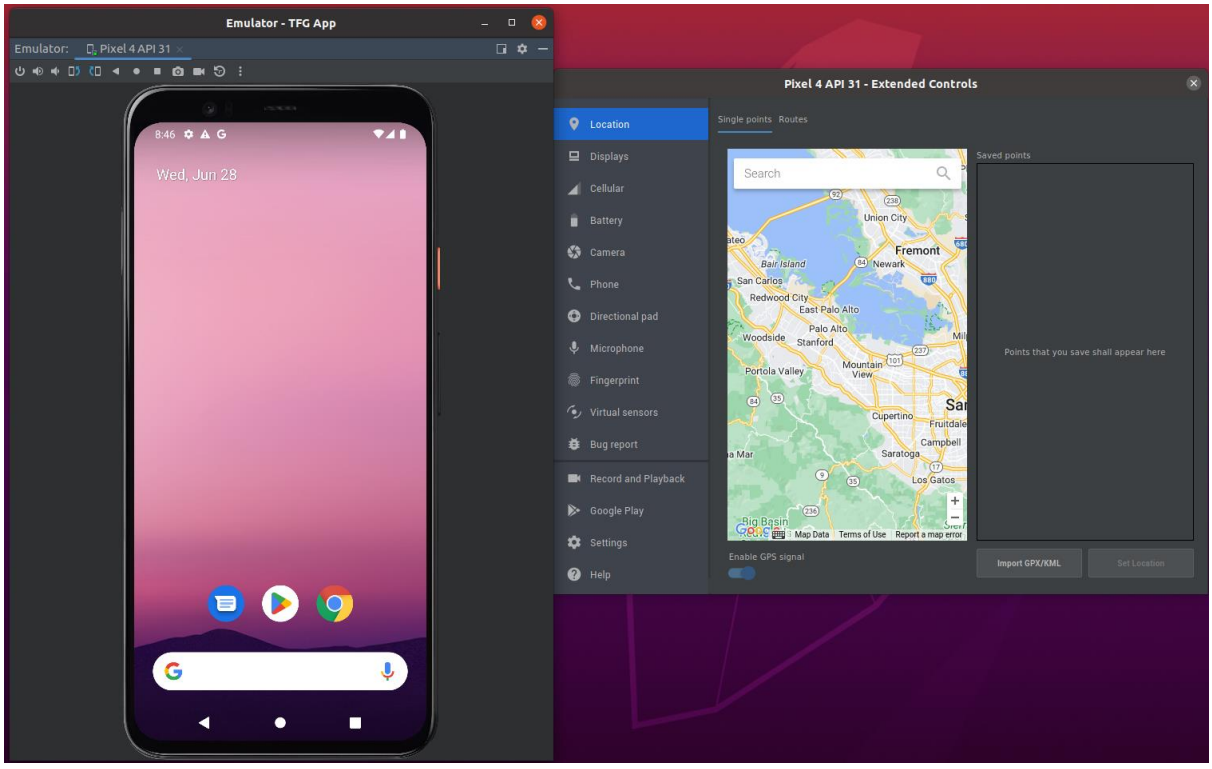

Figura 2-3. Emulador Pixel 4

2.2.3 OpenHAB

OpenHAB es un sistema de automatización del hogar de código abierto escrito en Java. [6]

Figura 2-4. Logo OpenHAB

2.2.4 Binding de Home Connect

Es un componente que permite la integración y comunicación entre el sistema de automatización del hogar OpenHAB y los dispositivos conectados a través de la plataforma Home Connect, permitiendo controlar y monitorear los electrodomésticos de manera centralizada. [7]

Figura 2-5. Logo Home Connect

2.2.5 Simulador de electrodomésticos de Home Connect

Para poder probar la comunicación entre la aplicación móvil y los electrodomésticos que soportan la tecnología Home Connect, se ha hecho uso del simulador de electrodomésticos que proporciona Home Connect. Concretamente, dispone de una cafetera, un frigorífico, una lavadora, una secadora, un lavavajillas y un horno, como podemos ver en la figura 2-6. [8]

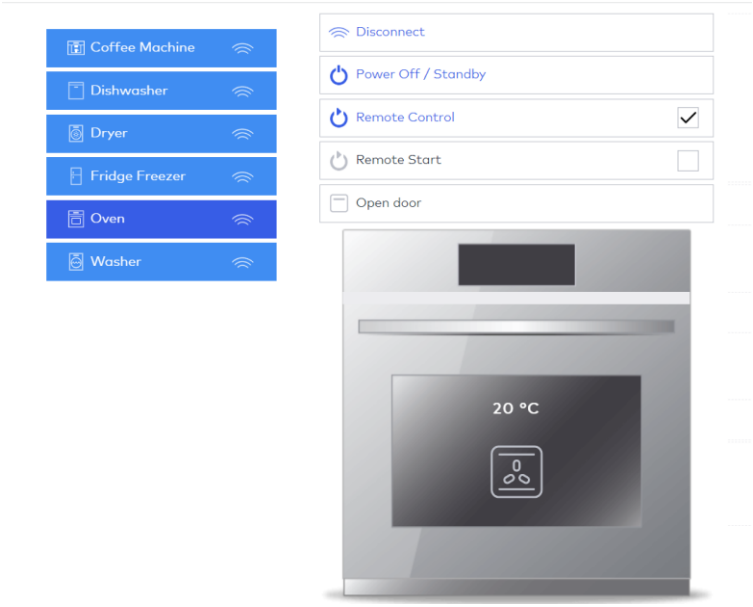

Figura 2-6. Simulador de electrodomésticos Home Connect

2.2.6 Ubuntu

El sistema operativo elegido para el desarrollo de la aplicación móvil, la ejecución en local del servidor personalizado de OpenHAB y el despliegue del servidor REST para el cálculo del consumo de los electrodomésticos, ha sido Ubuntu 20.04.

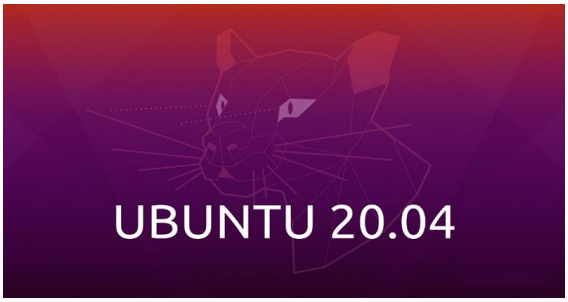

Figura 2-7. Ubuntu 20.04

2.2.7 Mozilla Firefox

Este ha sido el navegador elegido para poder visualizar el simulador de Home Connect y para poder acceder al servidor de OpenHAB y configurarlo.

Figura 2-8. Mozilla Firefox

2.2.8 Spring

Spring es un entorno de desarrollo de aplicaciones Java que proporciona un conjunto de herramientas y funcionalidades para construir, entre otras cosas, servicios RESTful de manera eficiente y escalable. [9]

Figura 2-9. Spring

2.2.9 Maven

Maven es una herramienta de gestión de proyectos que facilita la construcción, gestión de dependencias y compilación de aplicaciones Java de manera eficiente. Está basado en un formato XML. [10]

Figura 2-10. Maven

2.2.10 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto y robusto que ofrece un alto nivel de funcionalidad y capacidad de adaptación para el almacenamiento y manipulación de datos. [11]

Figura 2-11. PostgreSQL

2.2.11 Figma

Figma es una herramienta de diseño colaborativo basada en la nube que permite a los equipos crear, prototipar y compartir diseños de interfaces de usuario de manera eficiente. [12]

Figura 2-12. Figma

2.2.12 RESTClient

Es una herramienta que permite realizar peticiones HTTP y probar servicios REST directamente desde el navegador, facilitando su desarrollo y depuración. [13]

Figura 2-13. RESTClient

2.2.13 Visual Studio Code

Visual Studio Code, a menudo llamado VS Code, es un editor de código gratuito y ligero desarrollado por Microsoft. Está diseñado para diversos lenguajes de programación y ofrece características como resaltado de sintaxis, autocompletado de código inteligente, depuración, integración de control de versiones y una amplia gama de extensiones. [14]

Figura 2-14. Visual Studio Code

2.2.14 GitHub

GitHub es una plataforma web que permite a los desarrolladores gestionar, respaldar y colaborar en proyectos de software a través de sistemas de control de versiones. Además de su utilidad para proyectos colaborativos, también es valioso para su uso de forma individual, brindándoles un espacio para mantener un historial de versiones, explorar y respaldar su código de manera eficiente. [15]

Figura 2-15. GitHub

2.2.15 Wireshark

Wireshark es una herramienta de análisis de red de código abierto que permite capturar y examinar el tráfico de datos en una red. Proporciona información detallada sobre los paquetes de datos, permitiendo a los administradores de red y desarrolladores diagnosticar problemas, analizar protocolos y supervisar la actividad de la red en tiempo real. [16]

Figura 2-16. Wireshark

2.2.16 Spring Security

Spring Security es un marco de seguridad altamente personalizable para aplicaciones Java. Proporciona herramientas para autenticar y autorizar usuarios, proteger recursos y gestionar sesiones. Con características como autenticación de usuarios, control de acceso y protección contra amenazas de seguridad, Spring Security permite a los desarrolladores fortalecer la seguridad de sus aplicaciones de manera efectiva. [17]

Figura 2-17. Spring Security

3 ESTADO DEL ARTE

En este capítulo, se indicará la definición, la configuración y otras características de las tecnologías Home Connect y OpenHab. Además, se expondrán las ventajas de integrar OpenHAB y Home Connect.

3.1 Home Connect

Figura 3-1. Logo Home Connect

3.1.1 Definición de Home Connect

Home Connect es una tecnología que permite conectar y controlar electrodomésticos inteligentes de forma remota a través de una aplicación móvil o un asistente de voz. Con Home Connect, se puede acceder a funciones específicas de cada aparato, recibir notificaciones, consultar su estado, obtener consejos y recetas personalizadas, entre otras funcionalidades. Las marcas de electrodomésticos que están asociadas a Home Connect son: Bosch, Siemens, Gaggenau, Neff y Balay.

Por otro lado, Home Connect ofrece su propia aplicación móvil para Android y iOS y, además, lo más interesante es que ofrece una API REST que permite a los desarrolladores crear aplicaciones y servicios innovadores que se integren con los electrodomésticos conectados. La API REST de Home Connect se basa en los principios de arquitectura RESTful y utiliza el formato JSON para el intercambio de datos.

Para facilitar el uso de la API, Home Connect proporciona un portal del desarrollador donde se puede encontrar documentación, ejemplos de código, guías de inicio rápido y un simulador de electrodomésticos. El portal del desarrollador también permite gestionar las credenciales de acceso, las suscripciones y el soporte técnico.

3.1.2 Registro y configuración de Home Connect

En esta sección, exploraremos detalladamente los pasos requeridos para configurar nuestra cuenta en la web del desarrollador de Home Connect, permitiéndonos aprovechar al máximo esta tecnología y llevar a cabo un control eficiente de nuestros electrodomésticos.

1. Lo primero que tendremos que hacer es registrarnos en el portal del desarrollador [18] y rellenar nuestros datos personales como se puede observar en la figura 3-2.

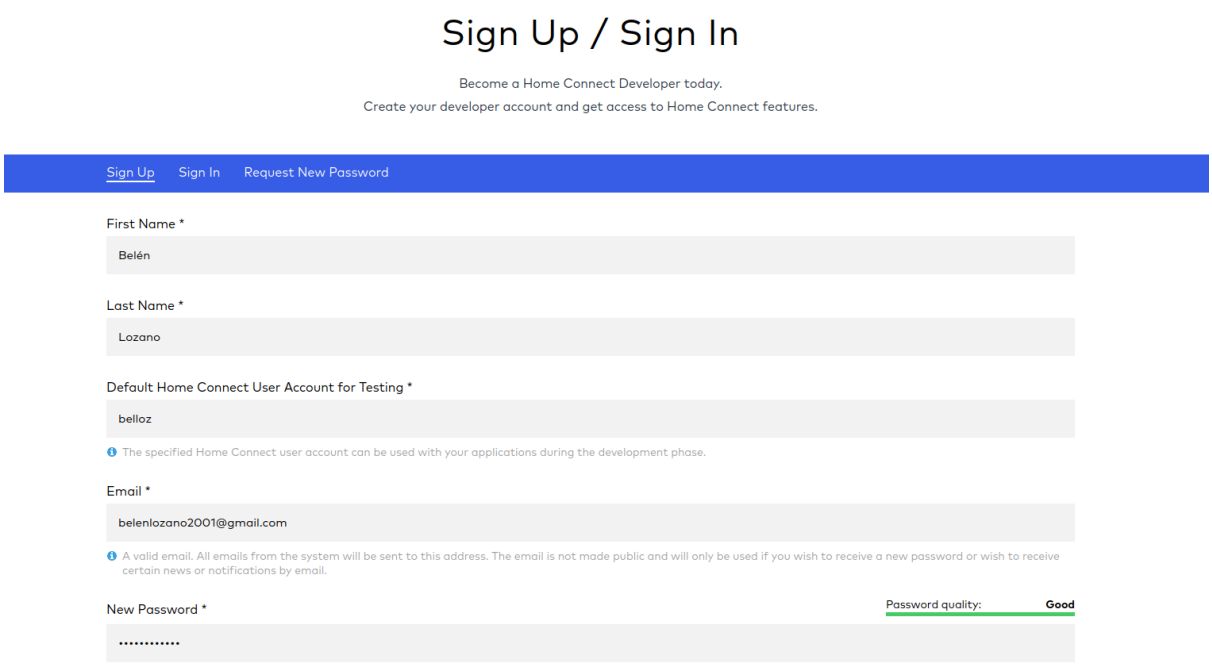

Figura 3-2. Registro usuario Home Connect

2. Una vez registrada nuestra cuenta, nos dirigimos a la pestaña Applications [19] y registramos una nueva aplicación. En la figura 3-3, podemos observar cómo habría que configurar los diferentes campos para que todo funcione de forma óptima. Es de especial importancia que en el apartado "Home Connect User Account for Testing" indiquemos el mismo usuario que hemos configurado en el paso 1 y, en "Redirect URI", hay que indicar la IP y puerto donde se encuentre corriendo el servidor de OpenHAB seguido, por ejemplo, de "/homeconnect".

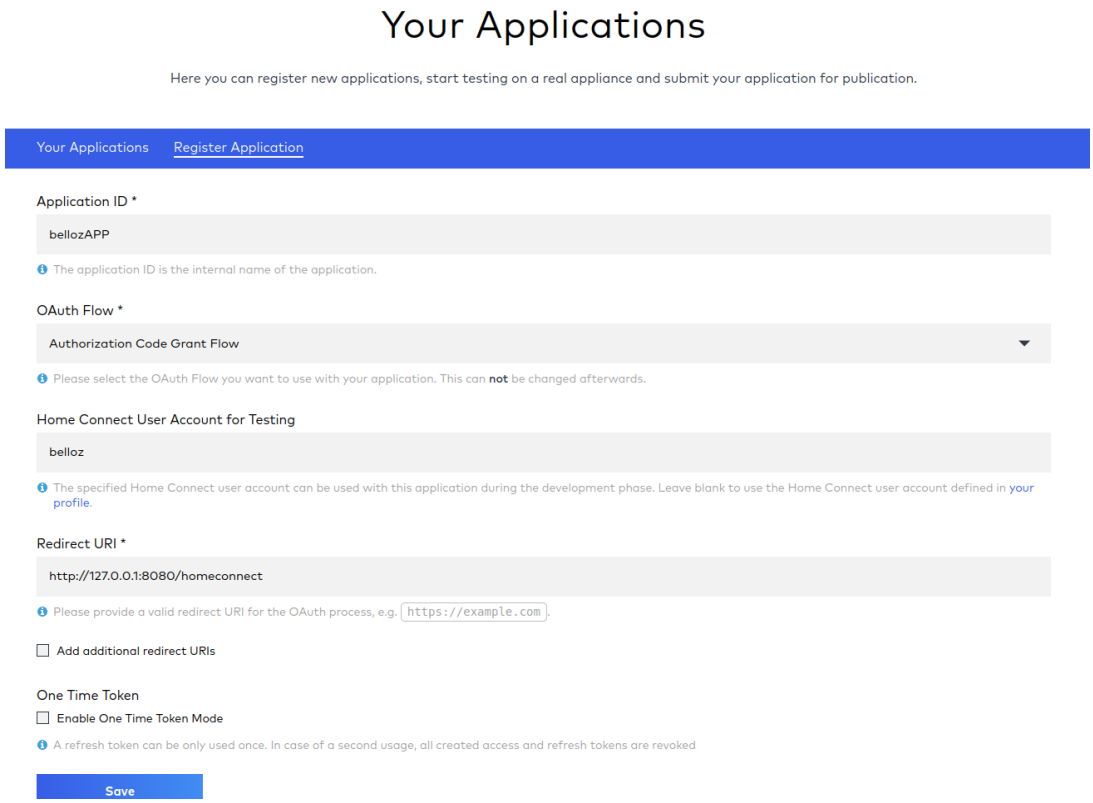

Figura 3-3. Registro app Home Connect

3. Una vez registrada la aplicación, si todo lo hemos realizado correctamente, nos debería salir algo similar a la figura 3-4. A la izquierda, tenemos un ID de cliente y dos enlaces hacia el cliente API Web que ofrece Home Connect, para familiarizarnos con los *endpoints*[3] de su API REST. Y, a la derecha, tenemos los datos de la aplicación que hemos creado en el paso 2.

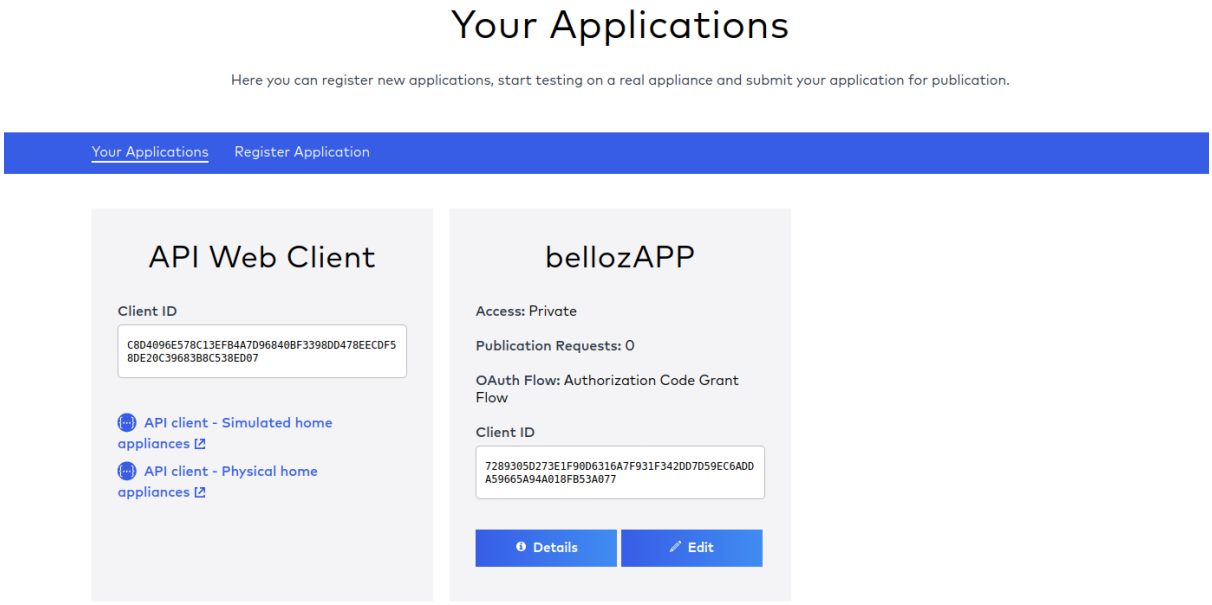

Figura 3-4. Menú aplicaciones Home Connect

4. Si hacemos clic en "Details" podremos ver la clave secreta del cliente y el resto de los datos que hemos rellenado en el registro de la aplicación. Estos datos serán de crucial importancia para asociar nuestra aplicación Home Connect con OpenHAB. En la figura 3-5, tenemos los datos de nuestra aplicación Home Connect titulada "bellozAPP".

[3] URLs específicas de un servicio web o API a las cuales se puede acceder para realizar solicitudes y obtener respuestas de datos o realizar acciones específicas.

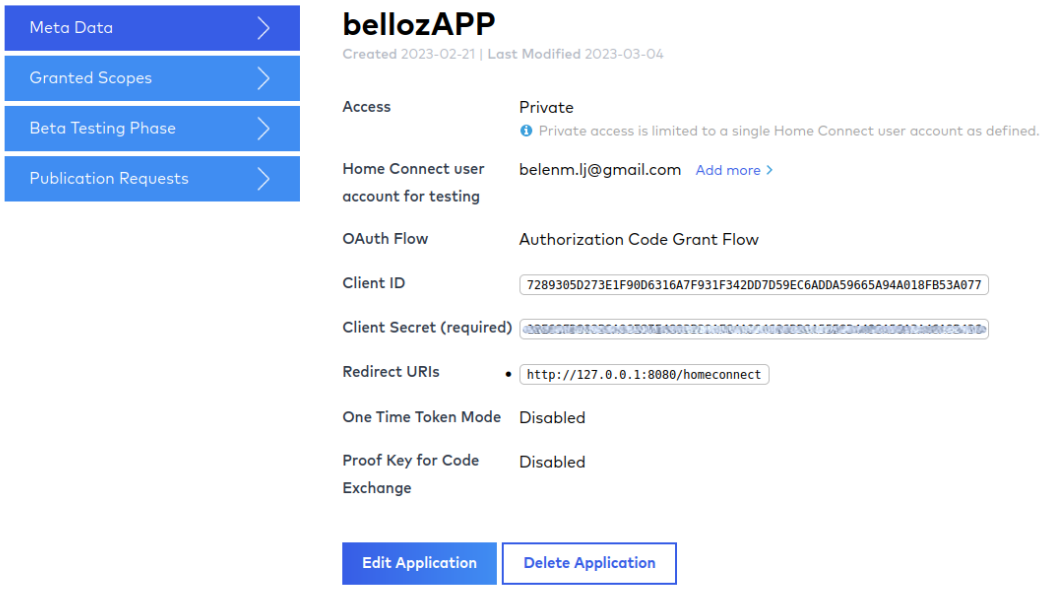

Figura 3-5. Datos bellozAPP Home Connect

Llegados a este punto, ya disponemos de todos los datos esenciales para establecer una vinculación exitosa entre Home Connect y OpenHAB.

3.1.3 Simulador de Home Connect

Este apartado se enfocará en explorar detalladamente el simulador de Home Connect, que es una herramienta clave en el proceso de desarrollo y prueba de este proyecto.

Concretamente, el simulador que nos proporciona Home Connect dispone de 6 dispositivos (cafetera, lavavajillas, secadora, frigorífico/congelador, horno y lavadora) que simulan el funcionamiento de los electrodomésticos reales. Esta herramienta, facilita la verificación del comportamiento de la aplicación en diferentes escenarios y configuraciones, eliminando la necesidad de interactuar con electrodomésticos físicos durante las etapas de prueba tempranas. Esto agiliza el proceso de desarrollo y permite identificar posibles problemas antes de la implementación final.

Además, en nuestro caso, dado que no disponemos de ningún electrodoméstico físico donde realizar las pruebas y su precio es muy elevado, esta herramienta toma mucha mayor relevancia dado que sin su existencia hubiera sido imposible verificar el correcto funcionamiento del proyecto.

A continuación, mostramos la figura 3-6 donde se puede observar el simulador de la cafetera y las opciones de las que dispone.

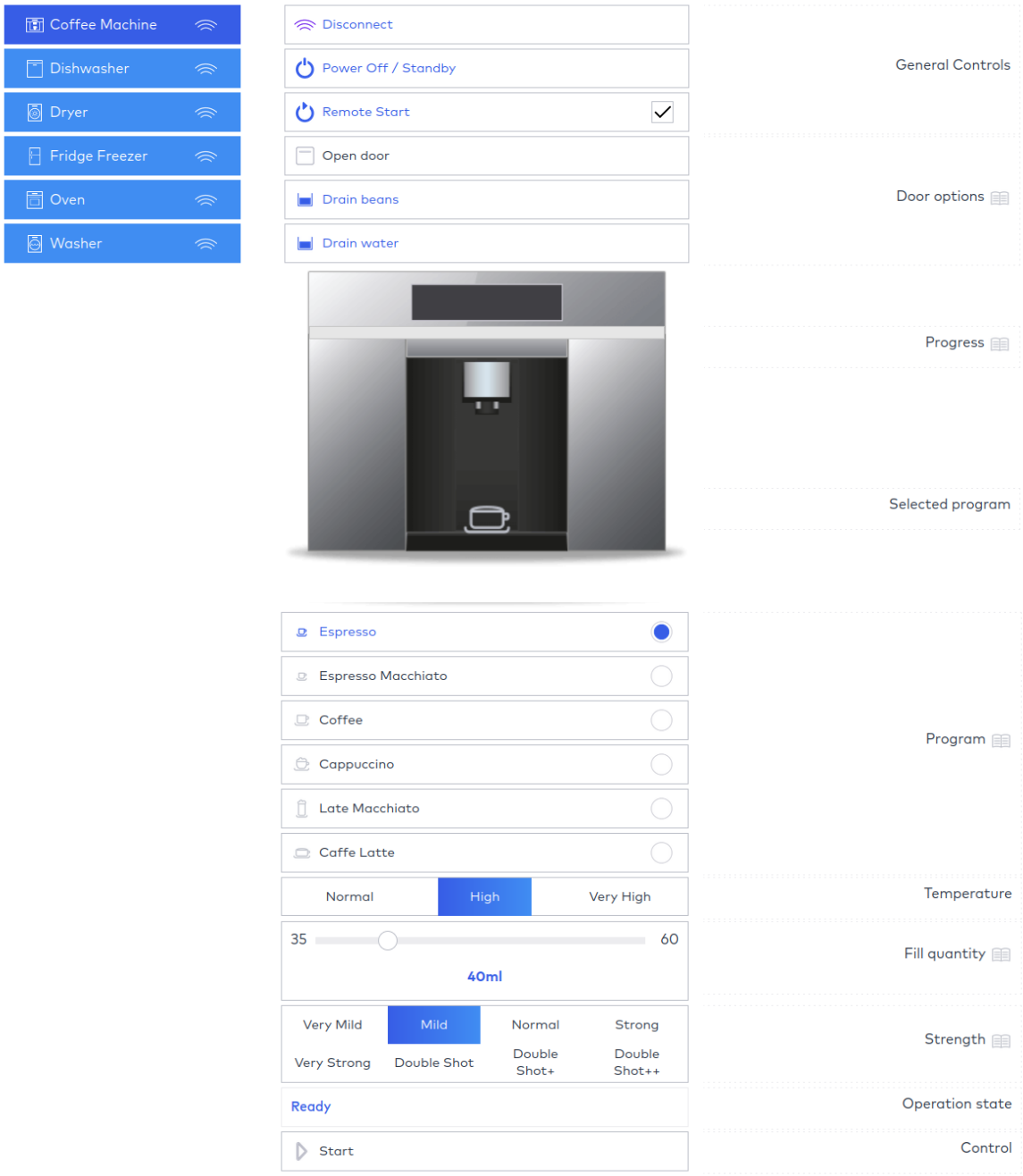

Figura 3-6. Simulador cafetera

Esta interfaz nos permitirá, por tanto, poder hacer comprobaciones de que todo está funcionando como deseamos una vez que hayamos implementado las funcionalidades de los electrodomésticos en la aplicación móvil. Por ejemplo, si desde la app hacemos clic en "Iniciar programa", en el simulador debería aparecer una barra de progreso indicando que se está haciendo correctamente el café seleccionado. De lo contrario, sería un indicativo claro de que está ocurriendo un problema de comunicación entre las APIs de OpenHAB y Home Connect o sencillamente no se está haciendo bien la petición desde la aplicación móvil a la API REST de OpenHAB.

3.1.4 API REST de Home Connect

La API REST de Home Connect [20], que engloba tanto el dominio de electrodomésticos físicos (api.home-connect.com) como el simulador (simulator.home-connect.com), se erige como un pilar fundamental en la interacción fluida entre los dispositivos y aplicaciones. Esta sección se sumerge en la exploración de la API REST de Home Connect, indagando en sus características, funcionalidades y su papel central en la comunicación y control de los electrodomésticos compatibles con esta tecnología. Tanto los electrodomésticos físicos como el simulador interactúan a través de esta API para brindar una experiencia coherente y eficiente en la gestión y monitoreo de los dispositivos en un entorno domótico.

Dado que en nuestro caso es OpenHAB el encargado de hacer las peticiones con su *binding* de Home Connect a esta API, se explicarán los aspectos claves de esta documentación, pero sin entrar en mucho detalle.

3.1.4.1 Quick Start

Home Connect ofrece un pequeño tutorial para familiarizarnos rápidamente con su API [21]. En este tutorial podremos llevar a cabo la autorización, monitorización y control de algunas características de los electrodomésticos, tanto físicos como simulados, para obtener unas nociones básicas del manejo de la API.

3.1.4.2 General

En el apartado general, la documentación nos indica algunos consejos para realizar de manera eficiente las peticiones a algunos endpoints. Además, en este apartado se desglosan los diferentes errores que puede devolver la API REST. Por ejemplo, el error 405 indicaría que el método indicado no está disponible para ese recurso.

3.1.4.3 Authorization

El apartado autorización de la API de Home Connect se refiere al proceso que permite a una aplicación de terceros acceder a los datos y controlar los electrodomésticos de Home Connect. Para ello, la aplicación debe solicitar y obtener el consentimiento del usuario final, que es el propietario o el usuario autorizado de los electrodomésticos. La API de Home Connect utiliza el protocolo OAuth 2.0 para gestionar la autorización, que es un estándar ampliamente utilizado y seguro para proteger los recursos de los usuarios.

3.1.4.4 Programs and Options

Este apartado, por su parte, incluye una descripción general de los programas y opciones que se pueden controlar a través de la API de Home Connect para cada tipo de electrodoméstico, como lavadoras, secadoras, lavavajillas, hornos, frigoríficos, etc. También, dispone de una tabla con los identificadores y los valores posibles de cada programa y opción, así como las restricciones y dependencias entre ellos.

3.1.4.5 States

En el apartado estados, nos explican cómo consultar los valores que representan el estado actual de un electrodoméstico, como la temperatura, el tiempo restante, el nivel de agua, etc. Cada estado tiene un identificador, un valor y una unidad.

3.1.4.6 Settings

En el apartado de ajustes, nos indican cómo consultar y modificar los valores que representan las preferencias o configuraciones de un electrodoméstico, como la potencia, el brillo, el sonido, el modo eco, etc. Cada ajuste tiene un identificador, un valor y una unidad.

3.1.4.7 Events

Los eventos, son los mensajes que se envían desde los electrodomésticos al cliente cuando ocurre algún cambio en el estado, el programa, las opciones o los *settings* de los mismos. Cada evento tiene un tipo, un identificador y un valor.

3.1.4.8 Commands

Por último, los comandos son las acciones que se envían desde el cliente a los electrodomésticos para controlarlos de forma remota. Cada comando tiene un identificador y un valor. Los comandos se pueden enviar mediante peticiones HTTP con el método PUT.

3.2 OpenHAB

Figura 3-7. Logo de OpenHAB

3.2.1 Definición de OpenHAB

OpenHAB es una plataforma de automatización del hogar de código abierto diseñada para integrar y controlar diversos dispositivos y tecnologías en un sistema cohesivo. Su arquitectura modular permite la comunicación entre diferentes sistemas y protocolos, brindando una solución unificada para la automatización y monitoreo del hogar.

Los elementos fundamentales de OpenHAB son:

- **Bindings** (Enlaces): son módulos de software que permiten la comunicación entre OpenHAB y los dispositivos a través de diferentes protocolos y sistemas. Cada *binding* está diseñado para un tipo específico de dispositivo o tecnología.
- **Things** (Dispositivos): representan los dispositivos físicos o virtuales que se integran en el sistema, como electrodomésticos, sensores, luces, termostatos, etc. Cada *thing* está asociado con un *binding* que define cómo interactuar con el dispositivo a través de su protocolo específico.
- **Items** (Elementos): son tipos básicos de datos que representan características específicas de los dispositivos (*things*). Los *items* pueden ser interruptores, selectores, cadenas de texto o cualquier otro tipo de información que se desee controlar o monitorear.
- **Channels** (Canales): los canales son enlaces lógicos entre *things* e *items*. Representan las diferentes funcionalidades que un dispositivo puede tener. Un *thing* puede tener varios canales, y cada uno se asocia a un *item* específico.
- **Rules** (Reglas): las reglas permiten definir acciones automáticas basadas en eventos. Pueden ser desencadenadas por cambios en los *items* o por eventos del sistema. Las reglas son esenciales para la automatización avanzada y la toma de decisiones en el sistema.

En conjunto, estos elementos permiten la creación de un sistema flexible y personalizado de automatización del hogar, donde se pueden definir reglas y acciones para controlar y monitorear dispositivos de diferentes tipos y marcas, todo a través de una interfaz unificada.

3.2.2 Configuración de OpenHAB

En el contexto del presente trabajo, la configuración de OpenHAB desempeña un papel crucial para garantizar un funcionamiento fluido y eficiente de la plataforma de automatización del hogar. Este apartado se adentrará en tres aspectos esenciales: en primer lugar, se explorará la configuración general que se realiza una vez que el servidor OpenHAB ha sido exitosamente iniciado en un entorno Ubuntu. Posteriormente, se abordará la configuración específica del *binding* de Home Connect, que permite la interacción y control de dispositivos electrodomésticos a través de la tecnología Home Connect. Por último, se mostrará la configuración relativa a los *things*, *items* y *channels* asociados al *binding* de Home Connect. Todos estos aspectos son fundamentales para establecer una base sólida en la que se construirá la integración y la gestión eficaz de los dispositivos domóticos.

3.2.2.1 Configuración general OpenHAB

Una vez arrancado correctamente el servidor y obtenida la respuesta exitosa de arranque (ver Anexo A), basta con acceder a la interfaz web de OpenHAB para poder configurar nuestros dispositivos. Para ello, ingresamos en un navegador web la siguiente URL:

http://127.0.0.1:8080

1. Si es la primera vez que accedemos a la misma, lo primero que nos aparecerá es una pantalla de registro para que creemos nuestra cuenta de administrador como podemos ver en la figura 3-8.

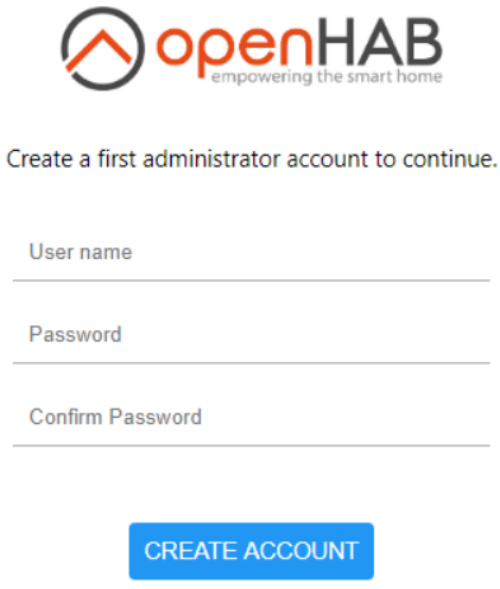

Figura 3-8. Registro OpenHAB

2. Seguidamente, nos aparecerá una ventana consultándonos nuestros datos de localización geográfica. Haremos clic en *Skip Setup* (saltar).

3. A continuación, aparece una ventana para instalar los complementos que queramos. Haremos clic en *Install Add-ons Later* (instalar más tarde).

4. Después, ya nos aparecerá el menú de inicio de la interfaz web de OpenHAB. Aquí hacemos clic a la pestaña con el candado que nos indica que iniciemos sesión como administrador para poder acceder a los ajustes, como podemos observar en la figura 3-9.

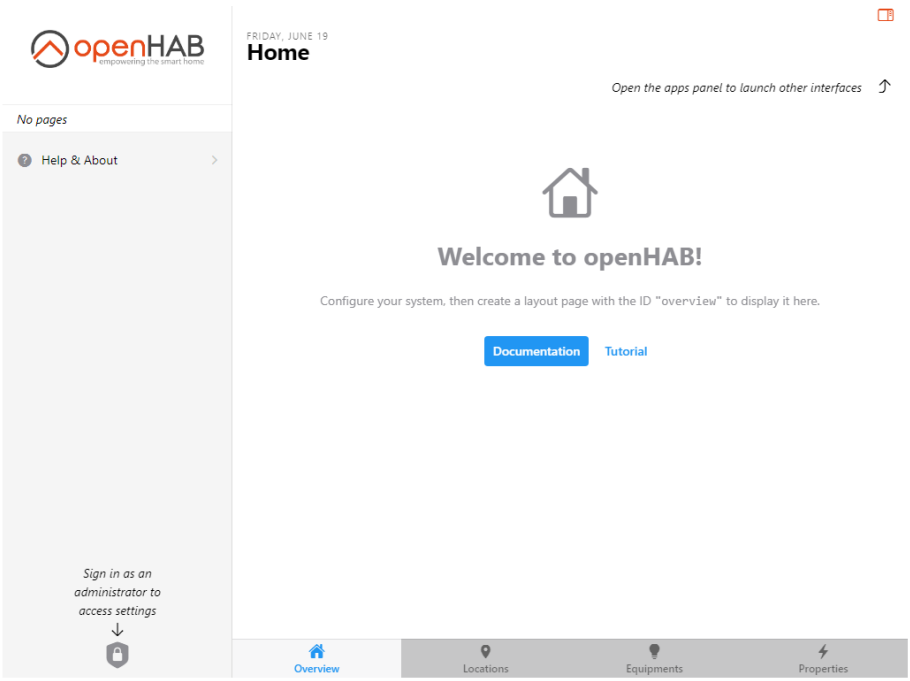

Figura 3-9. Menú de inicio OpenHAB

5. Iniciamos sesión con las credenciales que introdujimos en el paso 1 y ya nos saldrán los ajustes donde podremos configurar el servidor a nuestro gusto, como podemos ver en la figura 3-10.

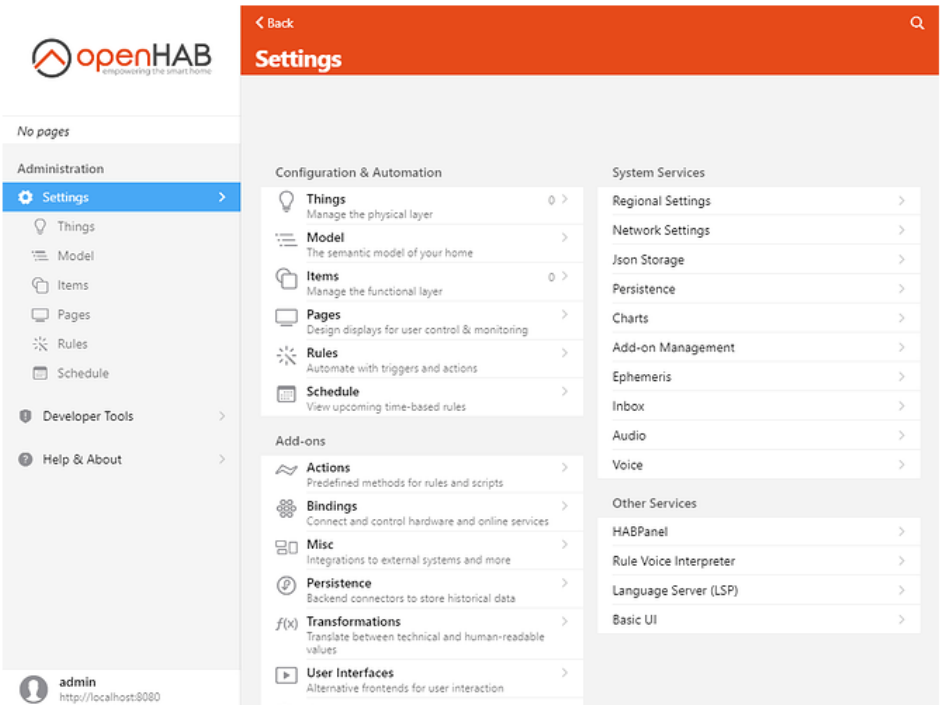

Figura 3-10. Ajustes OpenHAB

De esta manera, ya tendríamos configurado OpenHAB listo para añadir todos los elementos que queramos en nuestra casa virtual.

3.2.2.2 Configuración *binding* Home Connect

Lo primero que haremos para comenzar la configuración de nuestro servidor de OpenHAB será instalar el *binding* de Home Connect. Para ello, en los ajustes (*Settings*) navegamos hasta el apartado *Add-ons* y hacemos clic al apartado *Bindings.* Una vez ahí, buscamos *HomeConnect Binding* y hacemos clic en instalar (ver figura 3-11).

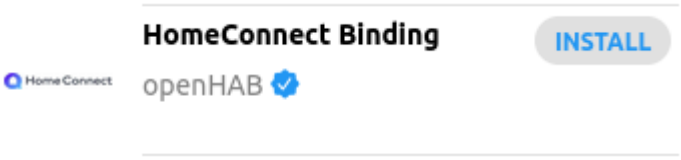

Figura 3-11. Binding HomeConnect

A continuación, procederemos a la configuración del *binding* siguiendo el manual de OpenHAB dedicado al *binding* de Home Connect [24]:

1. Nos vamos al apartado *Things* de la interfaz web de OpenHAB y le damos al botón de añadir (ver figura 3-12).

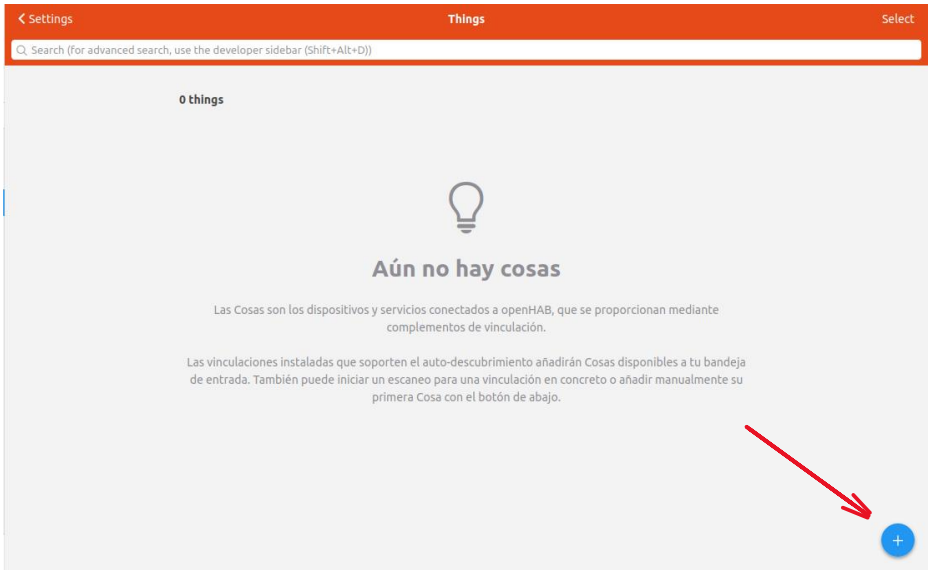

Figura 3-12. Añadir Thing

2. Seleccionamos el *binding* de Home Connect (ver figura 3-13).

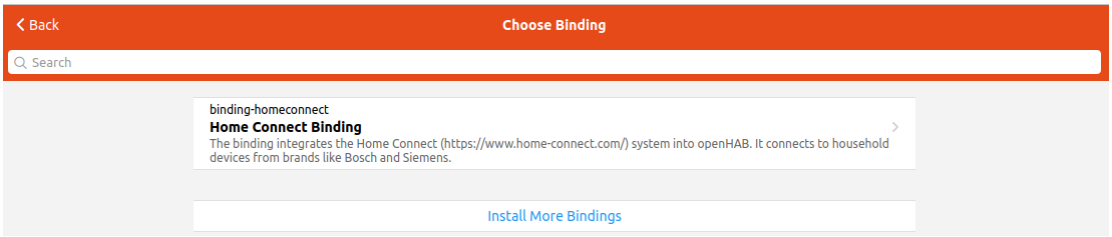

Figura 3-13. Elegir Binding

3. Hacemos clic en Home Connect API para configurar el *binding* (ver figura 3-14).

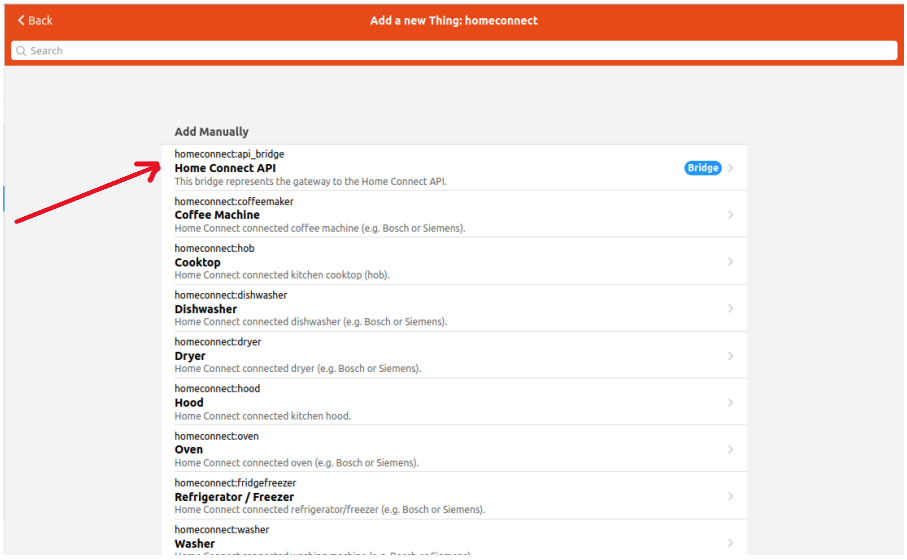

Figura 3-14. Elegir Home Connect API

4. Rellenamos los datos de *Client ID* y *Client Secret* con las credenciales proporcionadas por Home Connect para nuestra aplicación (figura 3-5, apartado 3.1.2) y luego hacemos clic en *Show advanced* para seleccionar la opción del simulador (ya que no tenemos dispositivos físicos que dispongan de la tecnología Home Connect para hacer nuestras pruebas). Por último, hacemos clic en el botón *Create Thing*. Podemos ver en la figura 3-15 cómo debe quedar esta configuración.

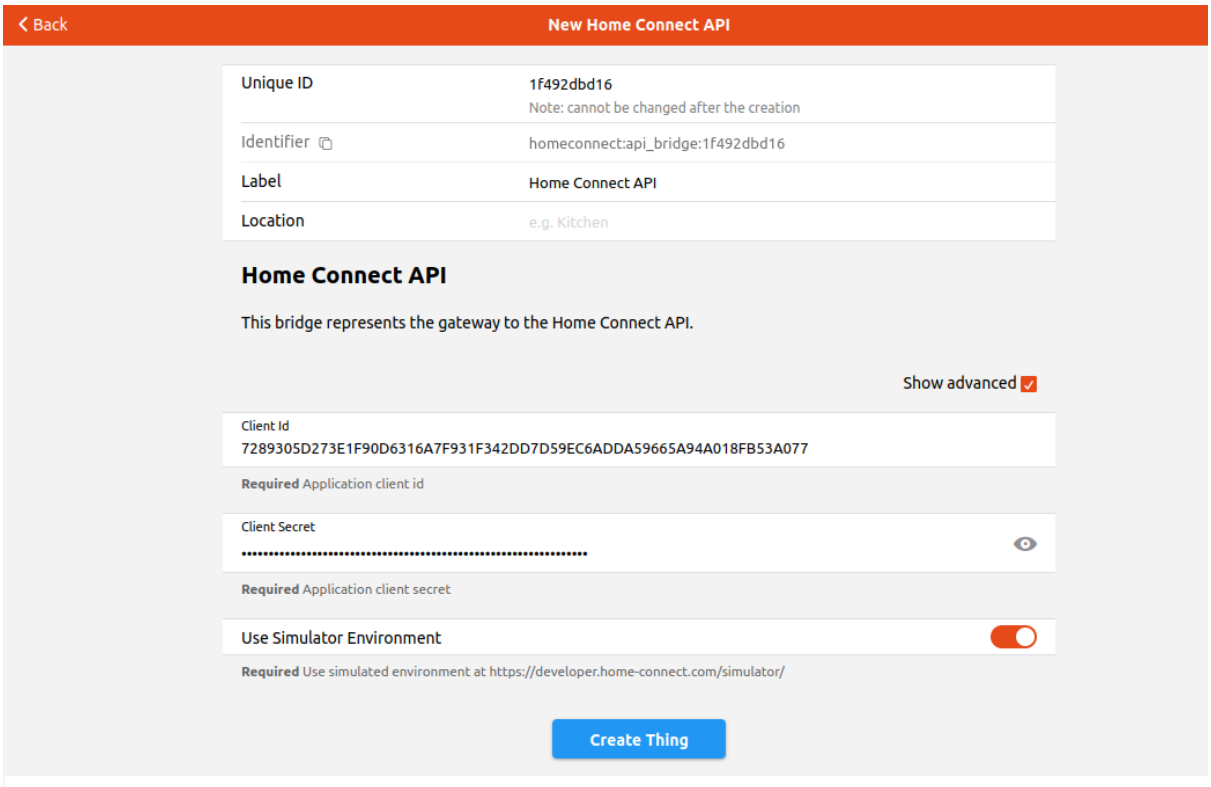

Figura 3-15. New Home Connect API

5. A continuación, ingresaremos en el navegador esta URL http://127.0.0.1:8080/homeconnect para obtener la autorización por parte de la API de Home Connect y hacemos clic en *Authorize Bridge* (figura 3-16).

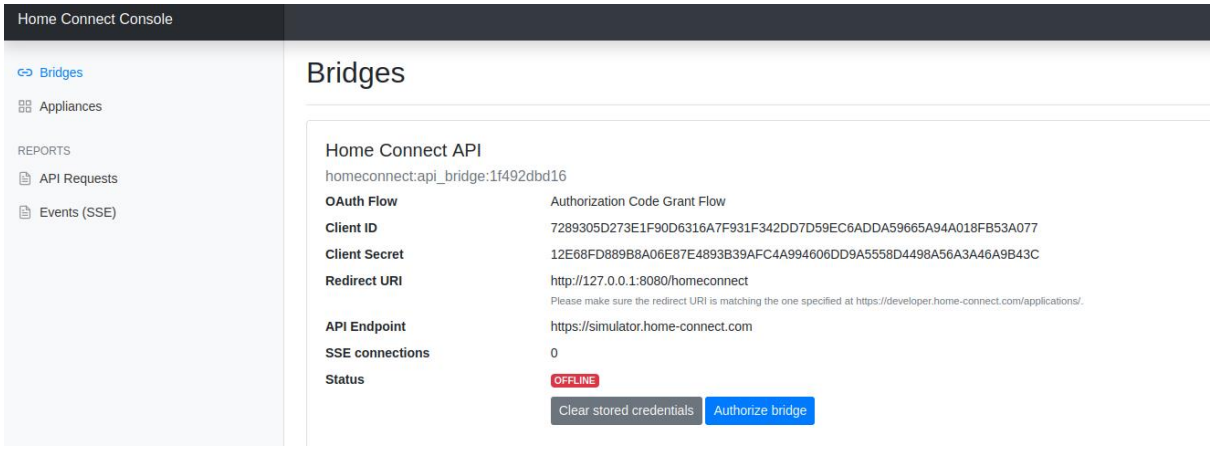

Figura 3-16. Home Connect Bridge

6. Seguidamente, nos redirecciona a una página de Home Connect donde debemos ingresar nuestras credenciales para obtener el token de autorización a la API REST de Home Connect (ver figura 3-17).

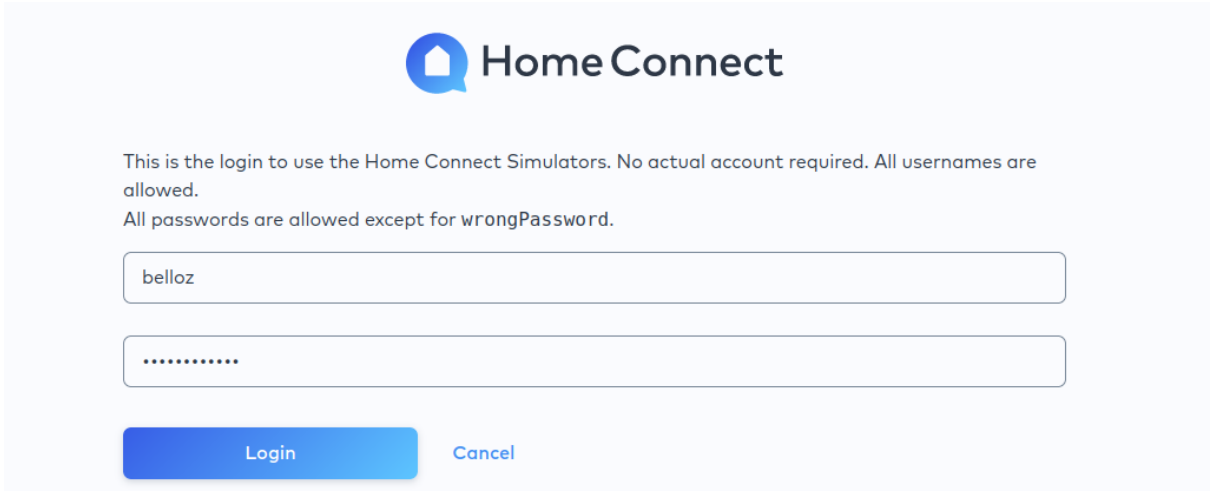

Figura 3-17. Ingresar credenciales Home Connect

7. Al hacer clic en *Login* nos redirecciona a otra página para aceptar un conjunto de permisos que determinan qué acciones y datos puedes acceder y manipular en los dispositivos. Los podemos observar en la figura 3-18.

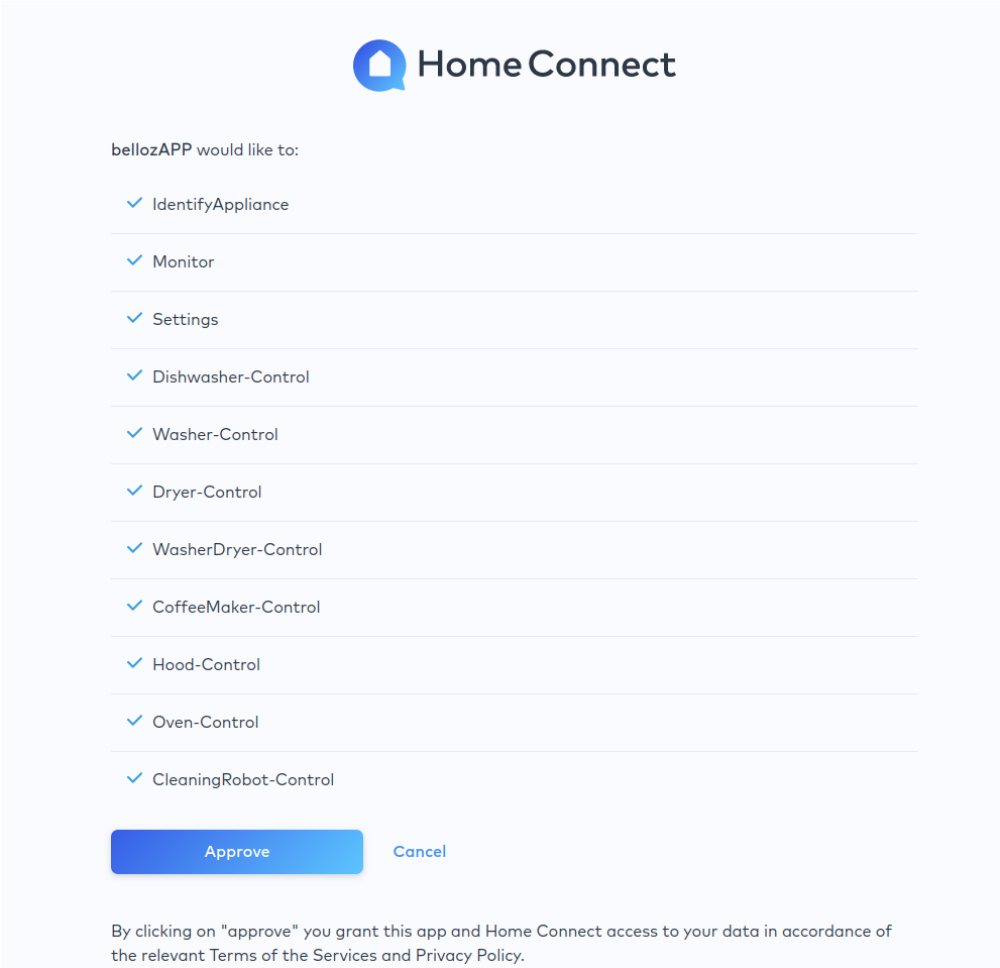

Figura 3-18. Scopes Home Connect

8. Si todo ha salido correctamente debería salir una pantalla similar a la de la figura 3-19.

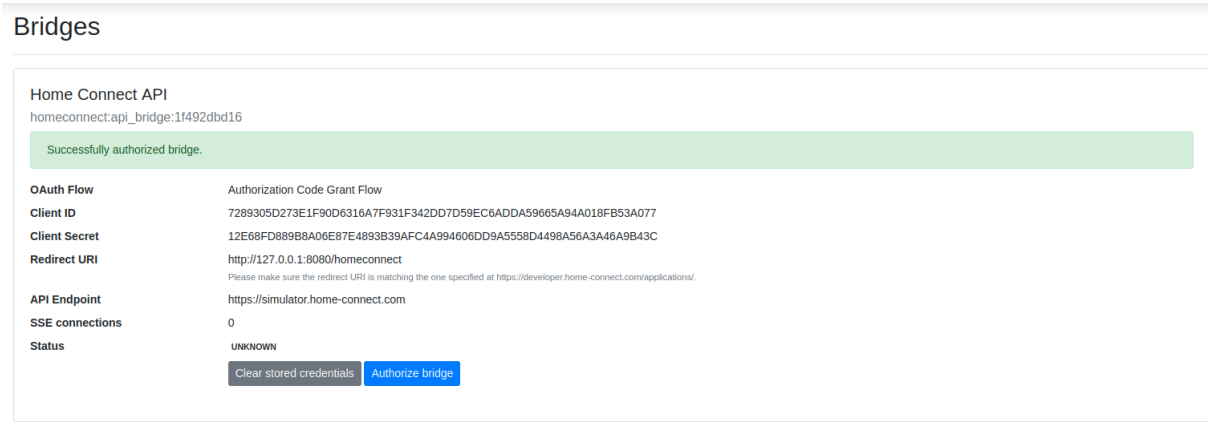

Figura 3-19. Autorización exitosa bridge

De esta manera, ya tendríamos correctamente configurado nuestro *binding* de Home Connect en OpenHAB.

3.2.2.3 Configuración de los electrodomésticos del binding de Home Connect

Una vez configurado nuestro binding de Home Connect, es el momento de vincular al mismo nuestros dispositivos del simulador para poder controlarlos correctamente desde nuestra aplicación móvil. Para ello, es necesario crear estos dispositivos (*things)* y, posteriormente, asociarlos mediante canales (*channels*) a los diferentes elementos (*items*) de los electrodomésticos.

3.2.2.3.1 Creación de Things del binding Home Connect

Para la creación de los dispositivos Home Connect en OpenHAB, seguiremos los siguientes pasos:

1. Hacemos clic en añadir un nuevo *Thing* de igual manera que en la figura 3-12 del apartado anterior. Seguidamente, hacemos clic en el *binding* de Home Connect y, a continuación, tenemos 2 opciones: meter manualmente la configuración de cada dispositivo o hacer clic en el botón *Scan* para que detecte los dispositivos compatibles que se encuentren conectados a la misma red que el servidor de OpenHAB.
2. Para hacerlo manualmente, hacemos clic en el tipo de dispositivo que queremos añadir (figura 3-20).

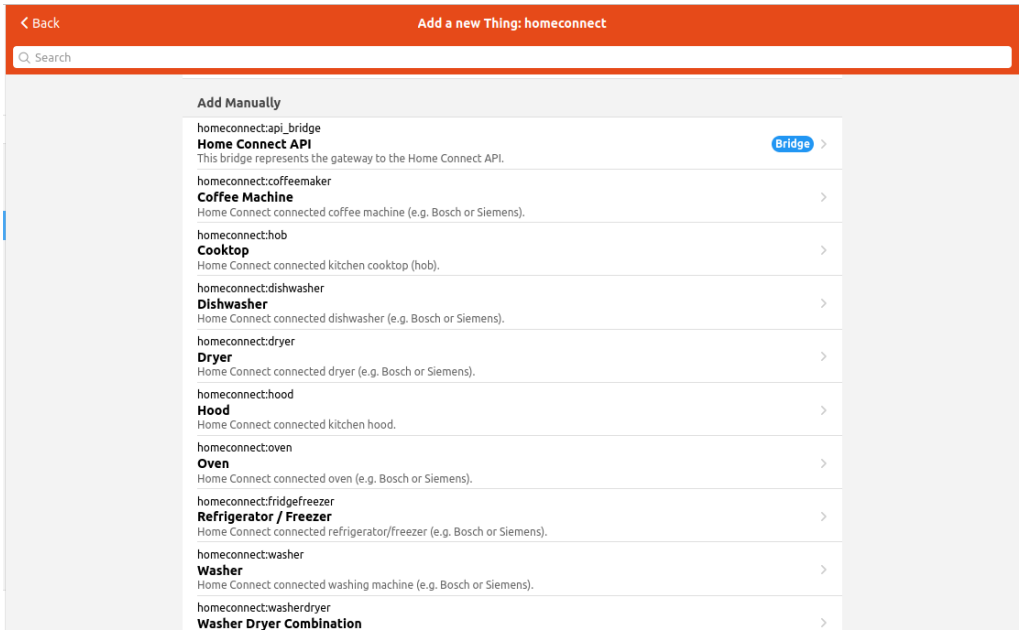

Figura 3-20. Añadir Thing manualmente

Por ejemplo, si seleccionamos la cafetera (Coffee Machine), habría que rellenar los datos que aparecen en la figura 3-21.

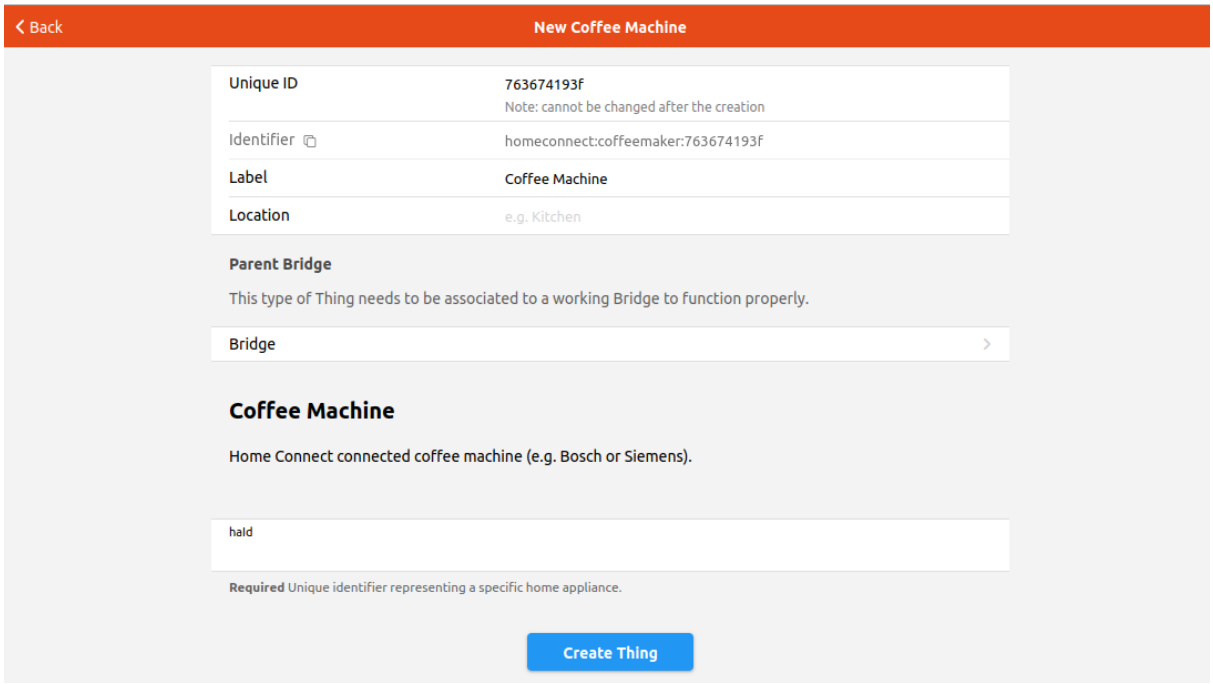

Figura 3-21. New Coffee Machine

3. En cambio, si hacemos clic en el botón *Scan* es mucho más sencillo, ya que se rellenan automáticamente estos datos (ver figura 3-22).

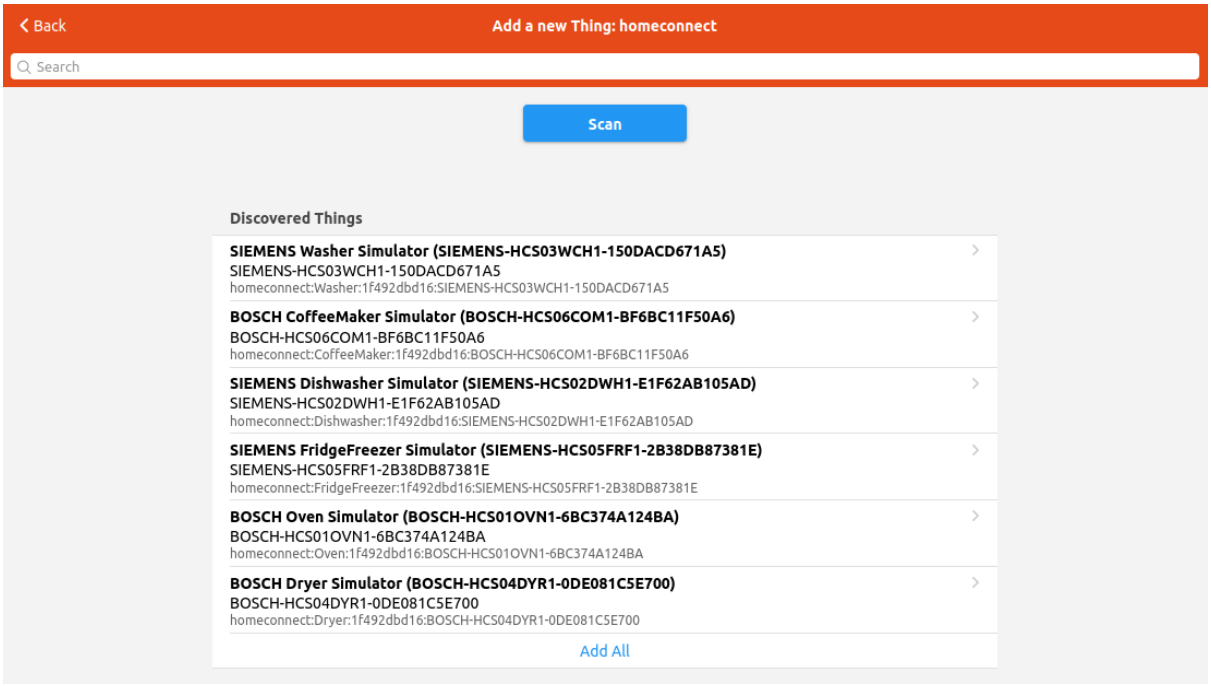

Figura 3-22. Scan new things

Hacemos clic en *Add All* y ya se nos añaden automáticamente todos los dispositivos conectados a nuestra red.

4. A continuación, un paso opcional es modificar el nombre de estos dispositivos para que sea más legible y además añadir la zona de la casa en la que se encuentran, rellenando para ello el campo *Location* (ver figura 3-23).

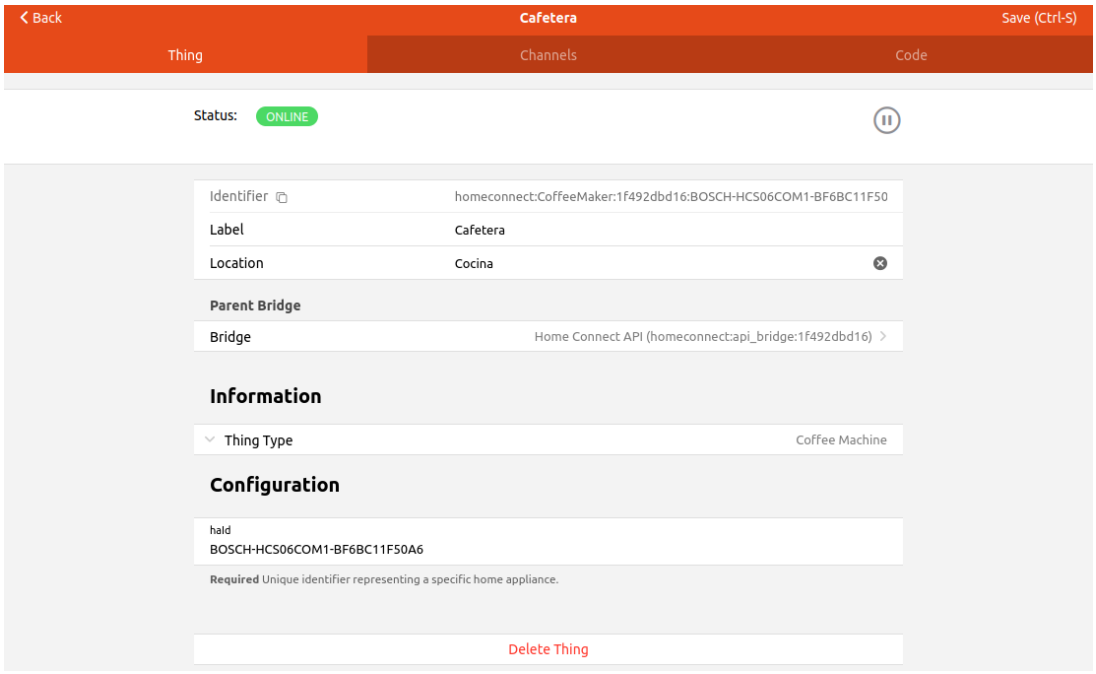

Figura 3-23. Configuración cafetera

5. De esta manera, ya tendríamos nuestros 6 dispositivos disponibles en el simulador de Home Connect vinculados a OpenHAB, como podemos comprobar en la figura 3-24.

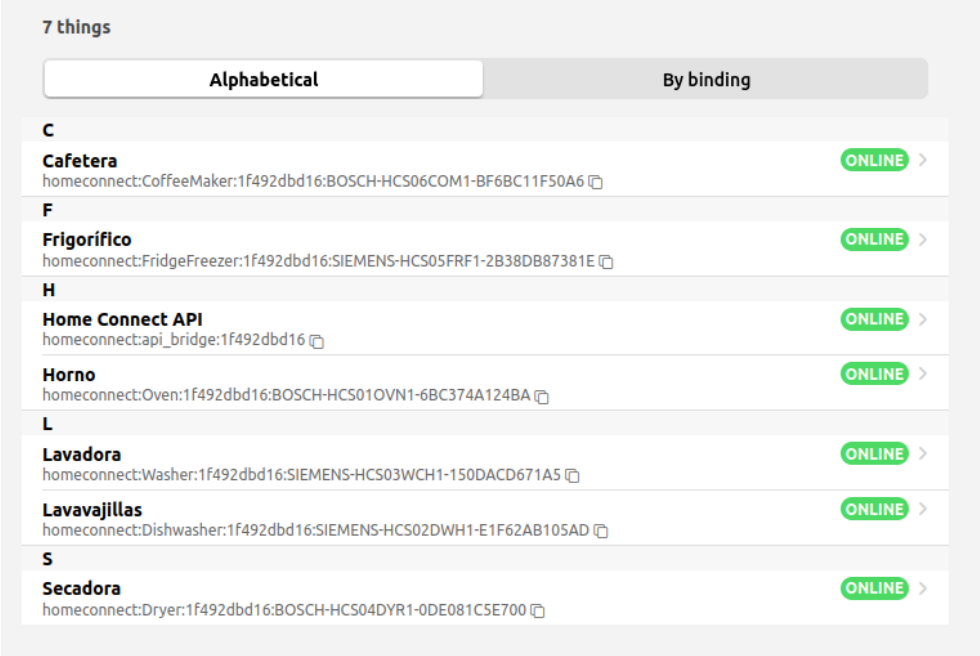

Figura 3-24. Things Home Connect

3.2.2.3.2 Creación de Items y vinculación de Channels del binding Home Connect

Una vez que se han establecido los 6 dispositivos del simulador de Home Connect en el entorno de OpenHAB, el siguiente paso clave es la creación de elementos (*items*) que actuarán como intermediarios para gestionar la interacción entre la aplicación móvil y los dispositivos. En este apartado, exploraremos el proceso de configuración de elementos y la forma en que se establece la conexión funcional entre los dispositivos (*things*) y los elementos (*items*) a través de canales (*channels*), facilitando así un control coherente y eficiente de los electrodomésticos mediante la plataforma OpenHAB.

Para ello, seguiremos los siguientes pasos:

1. Para añadir los *items* correspondientes a cada electrodoméstico, nos vamos a la lista de *things* y hacemos clic en el electrodoméstico al que queramos añadirle sus *items*. Seguidamente, le damos a la pestaña *Channels* donde veremos un listado con todos los canales disponibles. En la figura 3-25 podemos observar los *channels* correspondientes a la cafetera.

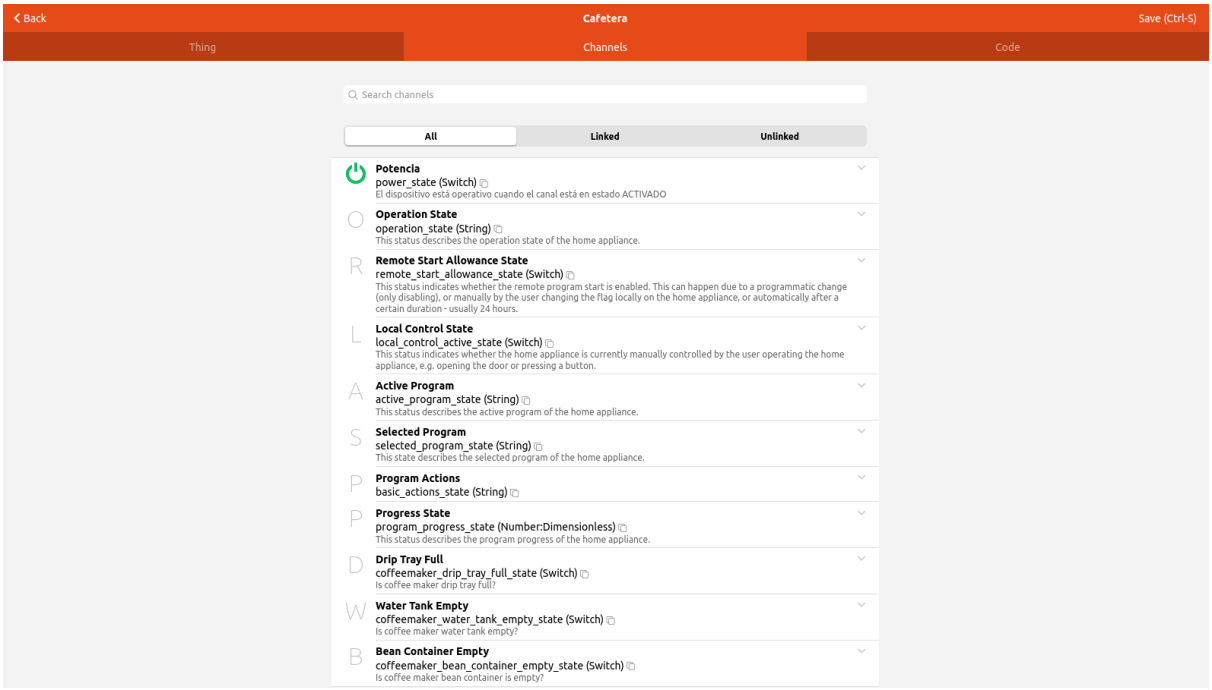

Figura 3-25. Channels cafetera

2. Hacemos clic por ejemplo en el primer canal "Potencia" y vemos que sale una pestaña donde pone "*Add Link to Item…"* (ver figura 3-26).

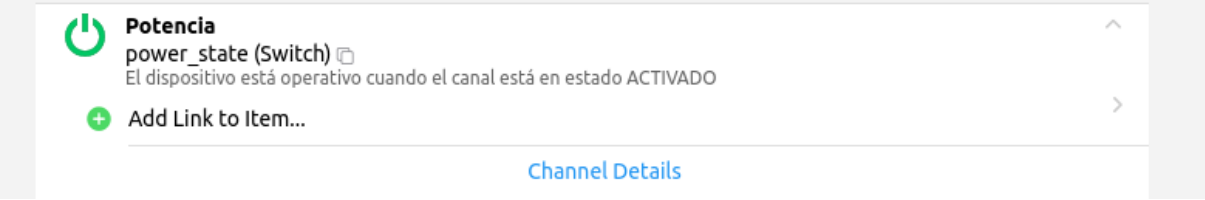

Figura 3-26. Channel potencia cafetera

3. Le damos a "*Add Link to Item…"* y nos permite 2 opciones: usar un *Item* existente o crear un nuevo *Item*, y vincularlo al canal. Elegimos la segunda opción, dejamos los valores por defecto y hacemos clic en el botón *Link*. Así ya tendríamos creado el nuevo *item* y vinculado a su canal correspondiente (ver figura 3-27).

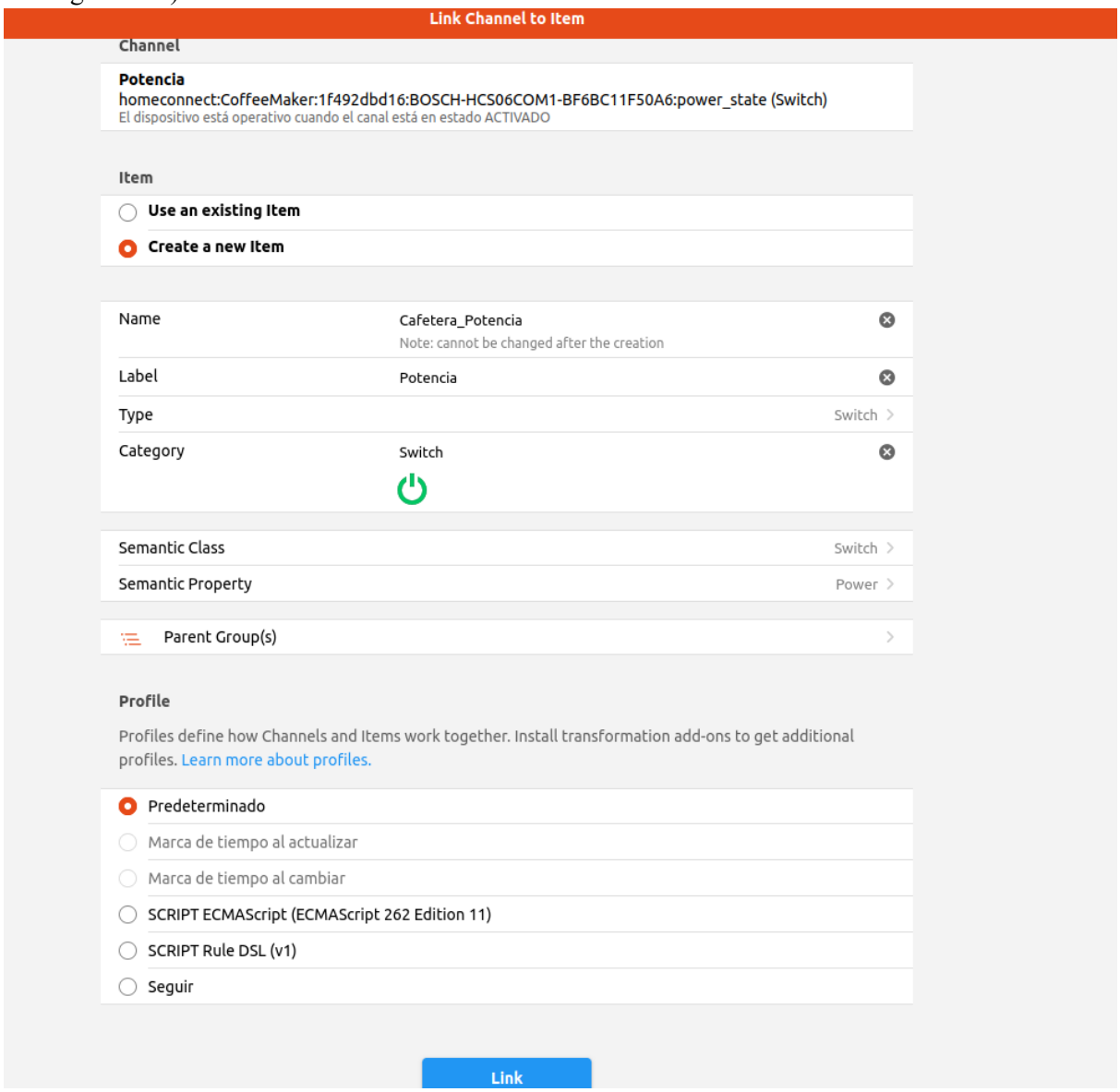

Figura 3-27. New item Potencia

4. De la misma manera, repetimos este procedimiento con el resto de los canales e *items* que queramos configurar en cada electrodoméstico.

A continuación, en la tabla 1 podremos ver los diferentes *Items* asociados a sus *Things* necesarios en nuestro proyecto:

Thing	Item	Descripción
Cafetera	Cafetera_Potencia	Estado actual de potencia de la cafetera (ON/OFF).
	Cafetera_Selected_Program	Programa seleccionado en la cafetera (Espresso, Espresso Macchiato, Coffee, Cappuccino, Late Macchiato, Caffe Latte).
	Cafetera_Program_Actions	Permite iniciar o parar el programa seleccionado en la cafetera.
	Cafetera_Operation_State	Estado de operación de la cafetera (Inactive, Ready, Run, Finished, Aborting) .
Lavavajillas	Lavavajillas_Potencia	Estado actual de potencia del lavavajillas (ON/OFF).
	Lavavajillas_Selected_Program	Programa seleccionado en el lavavajillas (Auto 35-45, Auto 45-65, Auto 65-75, Eco 50, Quick 45).
	Lavavajillas_Program_Actions	Permite iniciar o parar el programa seleccionado en el lavavajillas.
	Lavavajillas_Remaining_Program_Time	Tiempo restante para que finalice el programa seleccionado en el lavavajillas.
	Lavavajillas_Operation_State	Estado de operación del lavavajillas (Inactive, Ready, Run, Finished, Aborting).
Secadora	Secadora_Selected_Program	Material de la ropa a secar (Cotton, Synthetic, Mix).
	Secadora_Drying_Target	Tipo de secado (Iron dry, Cupboard dry, Cupboard dry plus).
	Secadora_Program_Actions	Permite iniciar o parar el programa seleccionado en la secadora.
	Secadora_Remaining_Program_Time	Tiempo restante para que finalice el programa seleccionado en la secadora.
	Secadora_Operation_State	Estado de operación de la secadora (Inactive, Ready, Run, Finished, Aborting).
Frigorífico/congelador	Frigorifico_Refrigerator_Temperature	Selector de temperatura del frigorífico (entre 2 y 8 ºC).
	Frigorifico_Refrigerator_Super_Mode	Permite activar la opción de super cooling del frigorífico.
	Frigorifico_Freezer_temperature	Selector de temperatura del congelador (entre -24 y -16 ºC).

	Frigorifico_Freezer_Super_Mode	Permite activar la opción de super freezing del congelador.
Horno	Horno_Potencia	Estado actual de potencia del horno (ON/OFF).
	Horno_Selected_Program	Programa seleccionado en el horno (Top/Bottom, Circulation Air, Pre Heat, Pizza Mode).
	Horno_Setpoint_Temperature	Selector de temperatura del horno (entre 30 y 250 ºC).
	Oven_Selected_Duration	Selector de duración del programa de horneado (mín. 1min, máx. 23h:59min).
	Horno_Program_Actions	Permite iniciar o parar el programa seleccionado en el horno.
	Horno_Remaining_Program_Time	Tiempo restante para que finalice el programa seleccionado en el horno.
	Horno_Operation_State	Estado de operación del horno (Inactive, Ready, Run, Finished, Aborting).
Lavadora	Lavadora_Selected_Program	Programa seleccionado en la lavadora (Cotton, Easy Care, Mix, Silk, Wool).
	Lavadora_Washing_Program_Temperature	Selector de temperatura del agua (entre Cold y 90 ºC).
	Lavadora_Spin_Speed	Velocidad de centrifugado (entre Off y 1600 rpm).
	Lavadora_Program_Actions	Permite iniciar o parar el programa seleccionado en la lavadora.
	Lavadora_Remaining_Program_Time	Tiempo restante para que finalice el programa seleccionado en la lavadora.
	Lavadora_Operation_State	Estado de operación de la lavadora (Inactive, Ready, Run, Finished, Aborting).

Tabla 1. Items de cada electrodoméstico

3.2.3 API Explorer de OpenHAB

Dentro del entorno de OpenHAB, el API Explorer se erige como una herramienta esencial para explorar y comprender la funcionalidad subyacente de la plataforma. En la figura 3-28, podemos ver los endpoints disponibles en la API REST de OpenHAB. Nos serán de especial interés /items, /things y /rules. Estos endpoints en particular captan nuestro interés debido a su papel fundamental en la gestión y control de elementos, dispositivos y reglas en el sistema de automatización del hogar proporcionado por OpenHAB (ver tabla 2).

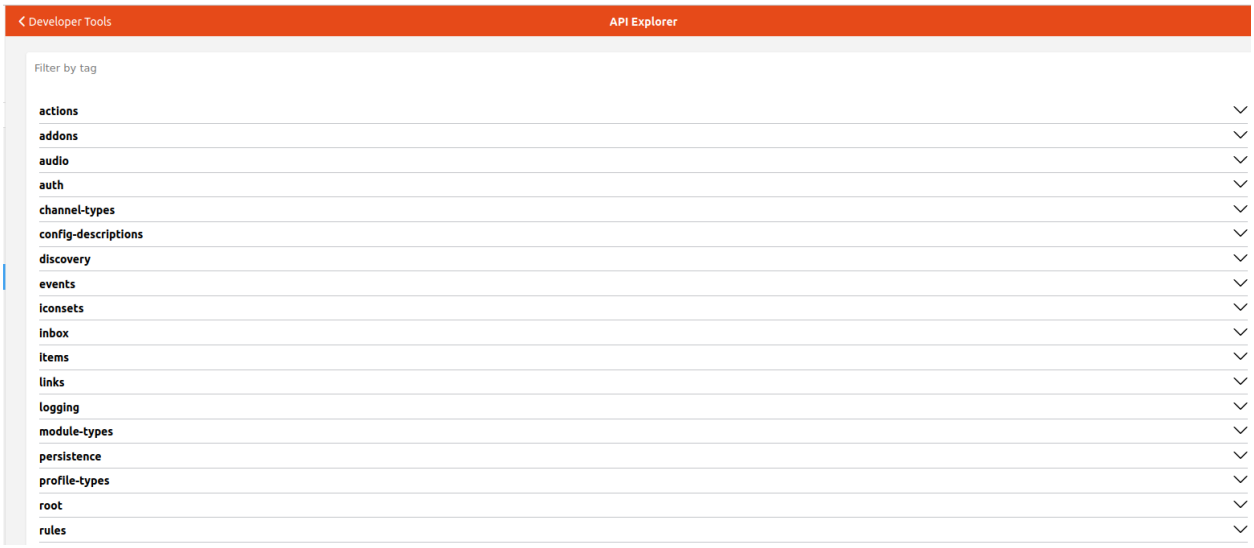

Figura 3-28. Endpoints API REST OpenHAB

Endpoint	Método HTTP	Descripción
/rest/items/{item_name}	GET	Permite obtener información detallada del *item* cuyo nombre sea {item_name}. Esta información puede ser el estado actual, una propiedad, etc.
	POST	Permite actualizar el estado o la propiedad del *item* cuyo nombre sea {item_name}. La información actualizada del *item* debe enviarse en el cuerpo de la petición.
/rest/things	GET	Permite obtener una lista con todos los dispositivos registrados en el sistema de OpenHAB.
/rest/rules/{rule_uid}	PUT	Permite actualizar la regla con el identificador único {rule_uid}. La información actualizada de la regla debe enviarse en el cuerpo de la petición.

Tabla 2. Endpoints utilizados en OpenHAB

A continuación, podremos observar un ejemplo de uso de la API Explorer de OpenHAB. Haremos una petición GET a http://127.0.0.1:8080/rest/items/Cafetera_Selected_Program (ver figura 3-29).

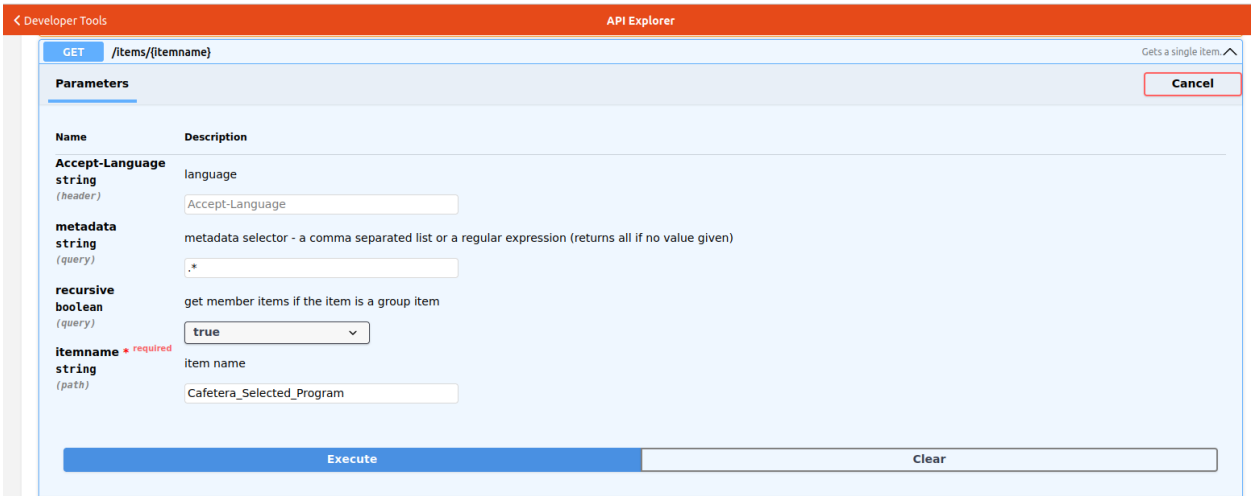

Figura 3-29. Petición GET API Explorer OpenHAB

La respuesta que obtenemos del servidor la podemos ver en la figura 3-30. Obtenemos una respuesta exitosa con código 200 OK y en el cuerpo de la respuesta podemos ver un JSON con los diferentes programas de la cafetera, entre otras cosas.

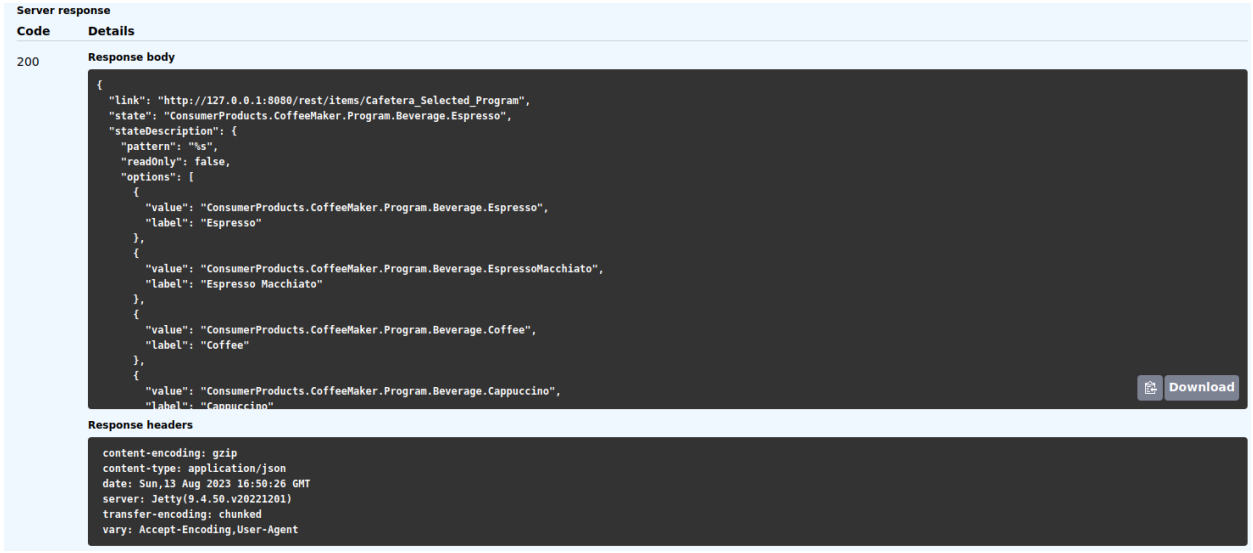

Figura 3-30. Respuesta del servidor a la petición GET

3.3 Ventajas de integrar OpenHAB y Home Connect

Llegados a este punto, podemos preguntarnos: ¿qué sentido tiene usar OpenHAB junto al *binding* de Home Connect en vez de usar exclusivamente la API REST que ofrece Home Connect? En este apartado, trataremos de explicar los motivos que nos han llevado a tomar esta decisión.

En primera instancia, la idea del proyecto era realizar las peticiones directamente a la API REST de Home Connect, sin embargo, a lo largo del desarrollo del proyecto, nos percatamos de que la integración de OpenHAB y Home Connect nos ofrecía una serie de ventajas que se detallan a continuación:

1. **Integración centralizada:** OpenHAB actúa como centro de control para la automatización del hogar, permitiendo la integración de diferentes sistemas y dispositivos. Con el bridge de Home Connect en OpenHAB, puedes agregar y controlar fácilmente tus electrodomésticos compatibles con Home Connect junto con otros dispositivos en un solo lugar.
2. **Interoperabilidad**: OpenHAB proporciona una capa de abstracción que permite la comunicación con una amplia gama de dispositivos y plataformas. Esto es altamente beneficioso, ya que permite agregar al servidor y a la aplicación móvil cualquier dispositivo compatible con la plataforma de OpenHAB. Por el contrario, la API de Home Connect solo permite el control y monitoreo de los dispositivos compatibles con la tecnología Home Connect.
3. **Flexibilidad y personalización:** OpenHAB ofrece flexibilidad en términos de configuración y personalización. Puedes adaptar la funcionalidad y la lógica de control según tus necesidades específicas, lo que te permite crear reglas y automatizaciones personalizadas. Esto te brinda un mayor grado de control sobre tus dispositivos y la posibilidad de crear experiencias personalizadas para los usuarios de tu aplicación. Por ejemplo, gracias a las reglas de OpenHAB es fácilmente estimable el consumo energético de los dispositivos.
4. **Persistencia de datos:** OpenHAB tiene la capacidad de integrarse fácilmente con otras herramientas y tecnologías que permiten la persistencia de datos.

En definitiva, la decisión de integrar OpenHAB junto al *binding* de Home Connect en lugar de utilizar exclusivamente la API REST de Home Connect, se ha revelado como una elección sólida y estratégica. Esta combinación ha demostrado ser beneficiosa en varios frentes, proporcionando una serie de ventajas clave para el desarrollo de este proyecto.

4 DESARROLLO DEL PROYECTO

En esta sección clave de nuestro proyecto, entraremos en el núcleo del desarrollo al abordar tres aspectos esenciales: la creación de la aplicación Android, la implementación del servicio REST de consumo y la configuración de reglas en OpenHAB para estimar el consumo de los electrodomésticos Home Connect. Cada uno de estos elementos representa un pilar fundamental en la integración y funcionalidad de nuestra solución. Exploraremos en detalle el proceso de desarrollo de la aplicación móvil, destacando su diseño intuitivo y su capacidad para interactuar con los dispositivos. Además, nos adentraremos en la implementación del servicio REST, que permite el seguimiento preciso del consumo energético de los electrodomésticos. Por último, exploraremos cómo la configuración de reglas en OpenHAB juega un papel crucial en la estimación precisa del consumo, proporcionando una experiencia de automatización completa y eficiente.

4.1 Aplicación Android IntelliHome

En este apartado, nos sumergiremos en la esencia de nuestro proyecto al explorar en profundidad la creación de la aplicación IntelliHome que actúa como el portal central para el control y monitoreo de los electrodomésticos Home Connect. Desde la conceptualización hasta la implementación, examinaremos cómo esta aplicación se convierte en un medio eficiente y accesible para los usuarios finales, permitiéndoles interactuar de manera intuitiva con sus electrodomésticos desde la comodidad de sus dispositivos móviles. En esta sección, exploraremos en profundidad los tres pilares que sustentan la aplicación: el diseño intuitivo de la interfaz de usuario, la integración de la API REST de OpenHAB y la conexión de la aplicación con el servidor REST de consumo. Cada uno de estos aspectos desempeña un papel crucial en la creación de una experiencia de usuario cohesiva y enriquecedora.

El código de la aplicación IntelliHome se puede descargar en el siguiente repositorio de GitHub:

https://github.com/bellozjur/IntelliHome.git

4.1.1 Diseño de la Interfaz de Usuario

La interfaz de usuario es el puente que conecta la funcionalidad de una aplicación con la experiencia del usuario. En este apartado, exploraremos en detalle cómo se ha diseñado y estructurado la interfaz de la aplicación Android para brindar una experiencia intuitiva y atractiva. Desde la disposición de los elementos hasta la elección de los colores y las tipografías, cada aspecto del diseño ha sido cuidadosamente considerado para asegurar que los usuarios puedan interactuar con facilidad y eficiencia con los dispositivos Home Connect y su consumo energético.

Para llevar a cabo un correcto diseño de la IU, se han seguido los siguientes pasos:

1. **Diseño de las pantallas en Figma:** en primer lugar, hemos utilizado la herramienta de diseño gráfico Figma para diseñar las pantallas principales de nuestra aplicación. En la figura 4-1, podemos ver el diseño inicial realizado en Figma de la parte de la aplicación encargada de controlar la cafetera.
2. **Conversión del diseño a XML:** en segundo lugar, hemos procedido a convertir los diseños realizados en Figma a código XML. Para ello, nos hemos ayudado de la propuesta de código que nos proporciona Figma para cada elemento, y seguidamente, hemos adaptado el código a nuestro gusto hasta obtener los resultados que queríamos.
3. **Implementación de la navegación y flujo de usuario:** en tercer lugar, hemos dado funcionalidad a los distintos botones y vistas del diseño para que el usuario pudiera moverse entre las diversas pantallas de forma fluida e intuitiva. Para ello, hemos utilizado el atributo *onClick* e implementado los métodos correspondientes para que nos llevaran a las actividades convenientes.

4. **Implementación de las interacciones y retroalimentación con el usuario:** por último, se ha llevado a cabo la implementación de los detalles que hacen mucho más visual, atractiva y cómoda a nuestra aplicación. Se ha hecho uso de algunos widgets como *PopupMenu* para lanzar desplegables con las diferentes opciones que puede seleccionar el usuario o cuadros de diálogo para interactuar con el usuario y solicitar que se rellenen ciertos datos o actuar a modo de aviso, entre otras muchas funcionalidades.

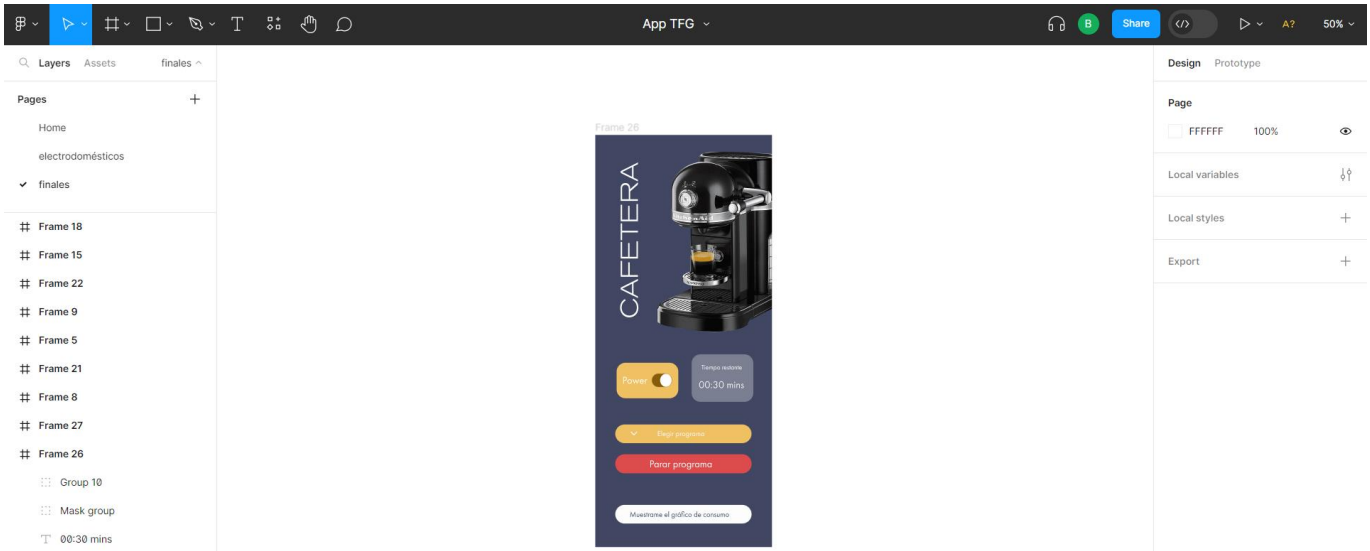

Figura 4-1. Diseño cafetera Figma

A continuación, procederemos a presentar y detallar las distintas **pantallas** que conforman la aplicación móvil:

4.1.1.1 Pantalla Login

Cada vez que el usuario inicie la aplicación IntelliHome esta será la pantalla de bienvenida, donde se invitará al usuario a introducir sus credenciales para poder acceder a su casa virtual, como podemos ver en la figura 4-2.

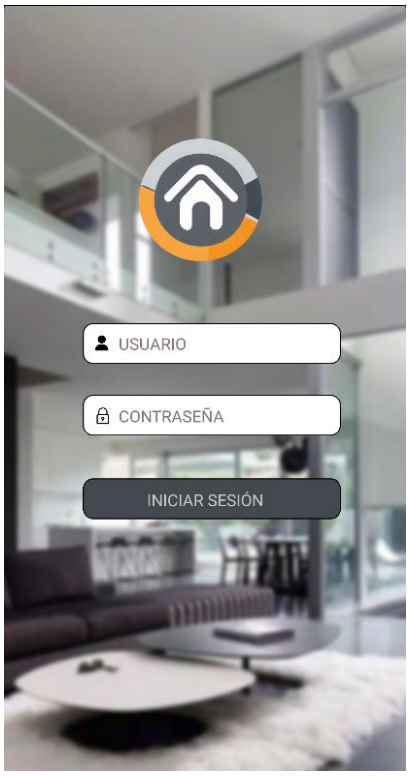

Figura 4-2. Pantalla Login

Cuando el usuario haga clic en INICIAR SESIÓN tras haber rellenado sus datos, se enviará una petición POST al endpoint /login del servidor de consumo, donde se comprobará si las credenciales son correctas o incorrectas y se le dará acceso o no en consecuencia al usuario.

4.1.1.2 Pantalla Habitaciones

Si el usuario ha introducido de forma correcta las credenciales, se cargará la pantalla “Habitaciones”, donde se listarán automáticamente todas las habitaciones disponibles en la tabla “habitaciones” de la base de datos SQLite de la aplicación (ver figura 4-3). Para lograr esto, es fundamental que el usuario haya previamente creado al menos una habitación siguiendo las indicaciones proporcionadas en la funcionalidad “Añadir nueva habitación”, explicada unas líneas más abajo. La aplicación utiliza una consulta para recuperar todas las habitaciones almacenadas en la base de datos y las presenta en la pantalla de forma ordenada. Cada habitación se muestra con su nombre y la imagen seleccionada por el usuario como icono. Esto proporciona al usuario una vista rápida y organizada de todas las habitaciones en su hogar.

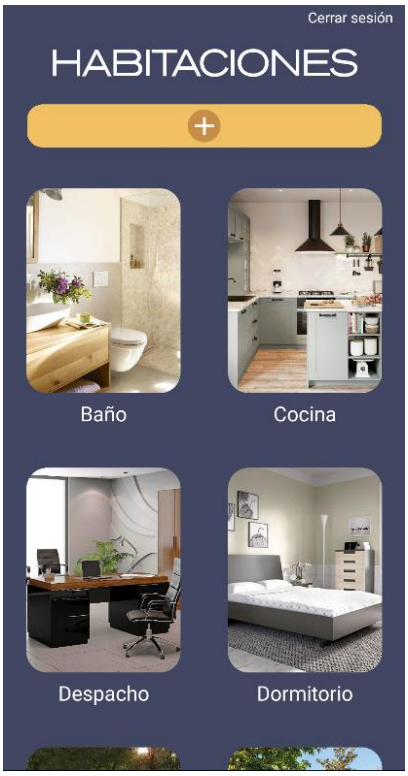

Figura 4-3. Pantalla Habitaciones

Esta pantalla tiene diversas funcionalidades:

1. **Cerrar sesión:** en la esquina superior derecha, al hacer clic en el texto "Cerrar sesión", seremos redirigidos a la pantalla de *Login*, al mismo tiempo que todos los datos asociados a la sesión que se acaba de cerrar serán eliminados.

2. **Añadir nueva habitación:** si hacemos clic en el botón amarillo con el símbolo “+”, nos saldrá un cuadro de diálogo como podemos ver en la figura 4-4. Este cuadro nos pedirá que insertemos un nombre a la nueva habitación así como que seleccionemos de la galería una imagen para ponerle de icono a esta nueva habitación. Cuando hagamos clic en AÑADIR, se insertará esta nueva habitación en la base de datos y ya aparecerá listada entre las demás habitaciones.

   Nota: es importante destacar que el nombre asignado a la habitación coincida con el atributo "Location" de los dispositivos en OpenHAB. Esto permitirá que los electrodomésticos ubicados en dicha habitación se muestren listados en la pantalla correspondiente a la misma.

Figura 4-4. Añadir habitación

3. **Editar habitación**: si mantenemos pulsada una de las habitaciones que aparecen listadas, nos aparecerá un menú con dos opciones como podemos ver en la figura 4-5. Si hacemos clic en "Editar" nos aparecerá un cuadro de dialogo similar al de "Añadir habitación" y podremos cambiarle el nombre y la imagen a la habitación que hemos pulsado como se puede apreciar en la figura 4-6. Cuando hagamos clic en ACEPTAR, se modificará esta habitación en la base de datos y ya aparecerá el cambio listado entre las demás habitaciones.

4. **Eliminar habitación**: si mantenemos pulsada una de las habitaciones que aparecen listadas, nos aparecerá un menú con dos opciones como podemos ver en la figura 4-5. Si hacemos clic en "Eliminar" nos aparecerá un cuadro de como se puede apreciar en la figura 4-7, preguntándonos si estamos seguros de querer eliminar la habitación que hemos pulsado. Cuando hagamos clic en ACEPTAR, se eliminará esta habitación de la base de datos y ya no aparecerá en el listado de las habitaciones.

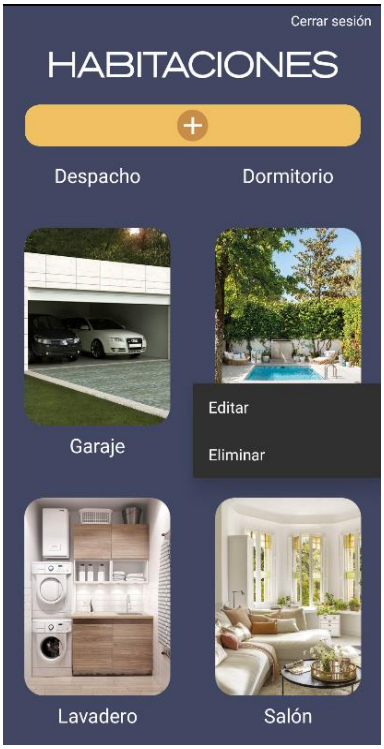

Figura 4-5. Editar o eliminar habitación

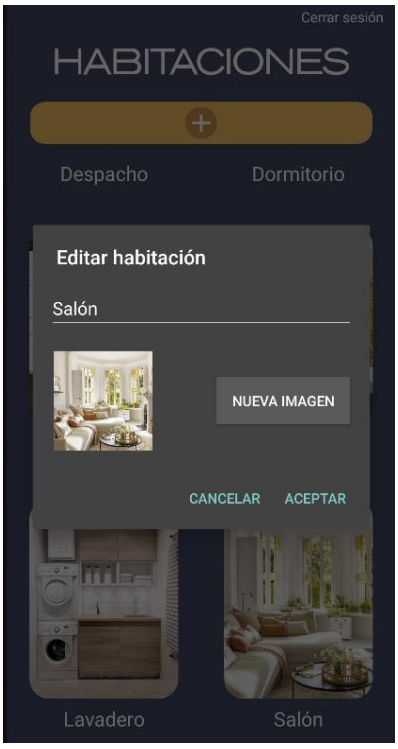

Figura 4-6. Editar habitación

Figura 4-7. Eliminar habitación

4.1.1.3 Pantalla Electrodomésticos: Cocina

Cuando hacemos clic en "Cocina" en la pantalla "Habitaciones", nos redirigimos a la pantalla "Electrodomésticos". Al iniciar esta actividad, se hace una petición GET al endpoint /things del servidor de OpenHAB de manera que se cargan en esta pantalla los electrodomésticos cuyo atributo Location tenga el valor *Cocina*. En nuestro caso, los electrodomésticos que están en esta ubicación son los que podemos ver en la figura

Figura 4-8. Pantalla Cocina

4-8.

4.1.1.3.1 Pantalla Cafetera

Si hacemos clic a la "Cafetera" en la pantalla "Electrodomésticos" (Cocina), seremos redirigidos a la pantalla "Cafetera" (ver figura 4-9). En esta pantalla tenemos diversas funcionalidades (entre paréntesis el elemento XML utilizado):

- Interruptor *Power* para activar o desactivar la potencia de la cafetera (View).
- Panel informativo con el estado de operación actual de la cafetera (TextView).
- Menú desplegable para elegir el programa de la cafetera (PopupMenu).
- Botón para iniciar o parar el programa seleccionado en la cafetera (View).
- Botón para navegar hacia el gráfico de consumo energético de la cafetera (View).

Cada una de estas funcionalidades, están asociadas a una o varias peticiones al servidor REST de OpenHAB o al de consumo, que serán detalladas en apartados posteriores.

Figura 4-9. Pantalla Cafetera

4.1.1.3.2 Pantalla Frigorífico

Si hacemos clic al "Frigorífico" en la pantalla "Electrodomésticos" (Cocina), seremos redirigidos a la pantalla "Frigorífico" (ver figura 4-10). En esta pantalla tenemos diversas funcionalidades (entre paréntesis el elemento XML utilizado):

- Selector de temperatura del frigorífico (NumberPicker).
- Interruptor *Super Cooling* para activar o desactivar el modo super cooling del frigorífico (Switch).
- Selector de temperatura del congelador (NumberPicker).
- Interruptor *Super Freezing* para activar o desactivar el modo super freezing del congelador (Switch).
- Botón para navegar hacia el gráfico de consumo energético del frigorífico/congelador (View).

Cada una de estas funcionalidades, están asociadas a una o varias peticiones al servidor REST de OpenHAB o al de consumo, que serán detalladas en apartados posteriores.

Figura 4-10. Pantalla Frigorífico

4.1.1.3.3 Pantalla Horno

Si hacemos clic al "Horno" en la pantalla "Electrodomésticos" (Cocina), seremos redirigidos a la pantalla "Horno" (ver figura 4-11). En esta pantalla tenemos diversas funcionalidades (entre paréntesis el elemento XML utilizado):

- Interruptor *Power* para activar o desactivar la potencia del horno (View).
- Panel informativo con el tiempo restante aproximado para concluir el programa actual (TextView).
- Menú desplegable para elegir el programa del horno (PopupMenu).
- Selector de temperatura del horno (NumberPicker).
- Selector para elegir la duración del programa de horneado (TimePicker).
- Botón para iniciar o parar el programa seleccionado en el horno (View).
- Botón para navegar hacia el gráfico de consumo energético del horno (View).

Cada una de estas funcionalidades, están asociadas a una o varias peticiones al servidor REST de OpenHAB o al de consumo, que serán detalladas en apartados posteriores.

Figura 4-11. Pantalla Horno

4.1.1.3.4 Pantalla Lavavajillas

Si hacemos clic al "Lavavajillas" en la pantalla "Electrodomésticos" (Cocina), seremos redirigidos a la pantalla "Lavavajillas" (ver figura 4-12). En esta pantalla tenemos diversas funcionalidades (entre paréntesis el elemento XML utilizado):

- Interruptor *Power* para activar o desactivar la potencia del lavavajillas (View).
- Panel informativo con el tiempo restante aproximado para concluir el programa actual (TextView).
- Menú desplegable para elegir el programa del lavavajillas (PopupMenu).
- Botón para iniciar o parar el programa seleccionado en el lavavajillas (View).
- Botón para navegar hacia el gráfico de consumo energético del lavavajillas (View).

Cada una de estas funcionalidades, están asociadas a una o varias peticiones al servidor REST de OpenHAB o al de consumo, que serán detalladas en apartados posteriores.

Figura 4-12. Pantalla Lavavajillas

4.1.1.4 Pantalla Electrodomésticos: Lavadero

Cuando hacemos clic en "Lavadero" en la pantalla "Habitaciones", nos redirigimos a la pantalla "Electrodomésticos". Al iniciar esta actividad, se hace una petición GET al endpoint /things del servidor de OpenHAB de manera que se cargan en esta pantalla los electrodomésticos cuyo atributo Location tenga el valor *Lavadero*. En nuestro caso, los electrodomésticos que están en esta ubicación son los que podemos ver en la figura 4-13.

Figura 4-13. Pantalla Lavadero

4.1.1.4.1 Pantalla Lavadora

Si hacemos clic a la "Lavadora" en la pantalla "Electrodomésticos" (Lavadero), seremos redirigidos a la pantalla "Lavadora" (ver figura 4-14). En esta pantalla tenemos diversas funcionalidades (entre paréntesis el elemento XML utilizado):

- Panel informativo con el tiempo restante aproximado para concluir el programa actual (TextView).
- Menú desplegable para elegir el programa de lavado (PopupMenu).
- Menú desplegable para elegir la temperatura del agua de la lavadora (PopupMenu).
- Menú desplegable para elegir la velocidad de centrifugado de la lavadora (PopupMenu).
- Botón para iniciar o parar el programa seleccionado en la lavadora (View).
- Botón para navegar hacia el gráfico de consumo energético de la lavadora (View).

Cada una de estas funcionalidades, están asociadas a una o varias peticiones al servidor REST de OpenHAB o al de consumo, que serán detalladas en apartados posteriores.

Figura 4-14. Pantalla Lavadora

4.1.1.4.2 Pantalla Secadora

Si hacemos clic a la "Secadora" en la pantalla "Electrodomésticos" (Lavadero), seremos redirigidos a la pantalla "Secadora" (ver figura 4-15). En esta pantalla tenemos diversas funcionalidades (entre paréntesis el elemento XML utilizado):

- Panel informativo con el tiempo restante aproximado para concluir el programa actual (TextView).
- Menú desplegable para elegir el material de la ropa (PopupMenu).
- Menú desplegable para elegir el programa de secado (PopupMenu).
- Botón para iniciar o parar el programa seleccionado en la secadora (View).
- Botón para navegar hacia el gráfico de consumo energético de la secadora (View).

Cada una de estas funcionalidades, están asociadas a una o varias peticiones al servidor REST de OpenHAB o al de consumo, que serán detalladas en apartados posteriores.

Figura 4-15. Pantalla Secadora

Nota: las demás pantallas de habitaciones que figuran en el menú "Habitaciones" no se presentarán en este documento, ya que no disponemos de electrodomésticos en esas ubicaciones específicas. Al hacer clic en estas pantallas, solo se mostrará el título de la habitación correspondiente, sin que aparezca ningún electrodoméstico listado.

4.1.1.5 Pantalla Consumo

Por último, a la pantalla "Consumo" podemos acceder cuando pulsamos sobre el botón de color blanco "Mostrar gráfico de consumo" que aparece al final de cada pantalla de un electrodoméstico. Esta pantalla nos muestra un gráfico de barras con el consumo energético diario dividido en 3 colores correspondiente a las 3 franjas horarias de consumo:

- Consumo punta (color rojo).
- Consumo llano (color azul).
- Consumo valle (color verde).

Como podemos observar en la figura 4-16, en la zona superior de esta pantalla aparece una sección de filtrado para poder obtener el consumo del electrodoméstico en el intervalo de días que introduzcamos. Si hacemos clic en cualquiera de los iconos del calendario (desde o hasta), saldrá una ventana (DatePicker) con un calendario donde podemos elegir un día concreto. Una vez seleccionado los días "desde" y "hasta", basta con hacer clic en GRAFICAR para obtener el consumo del intervalo seleccionado. Un ejemplo de filtrado puede verse en la figura 4-17.

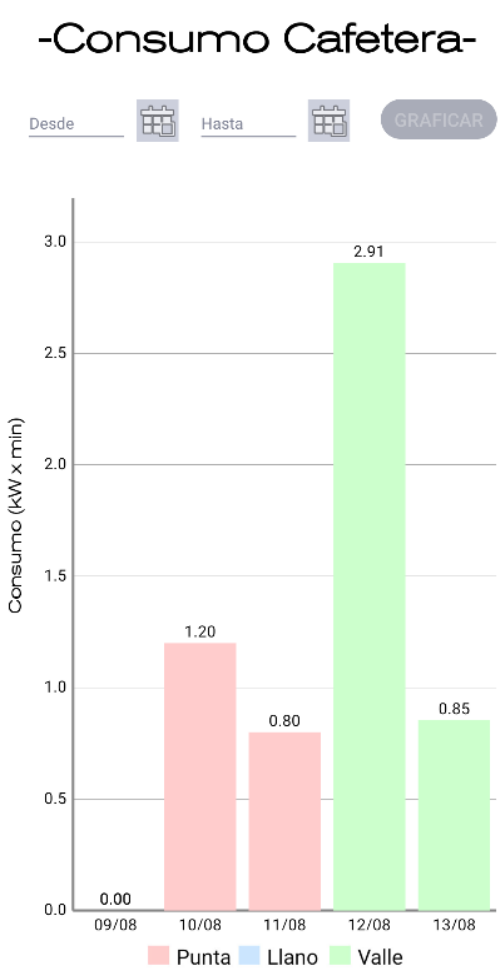

Figura 4-16. Pantalla Consumo

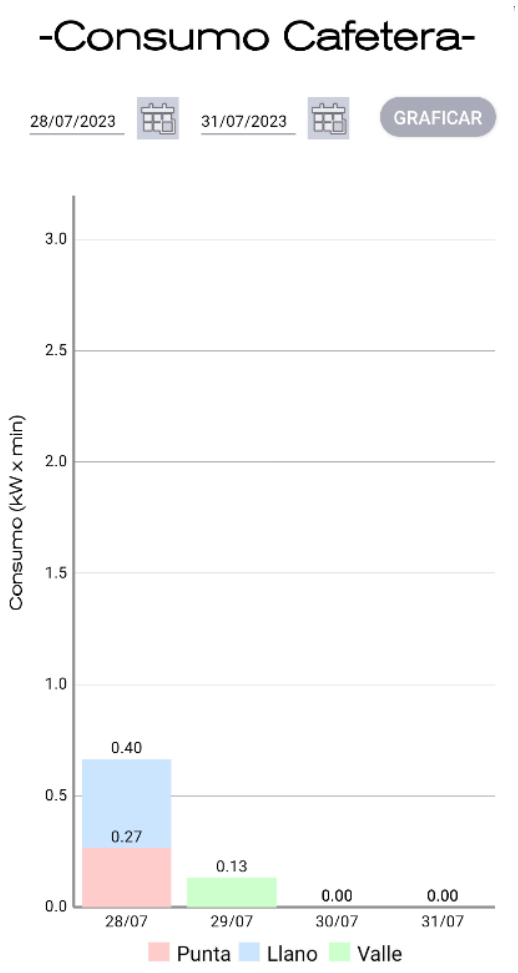

Figura 4-17. Consumo filtrado

4.1.2 Conexión de la aplicación móvil con el servidor de OpenHAB

En este apartado, se abordará la integración fundamental de la API REST de OpenHAB en la aplicación Android. Se explorarán los procesos de comunicación entre la aplicación y el servidor de OpenHAB, detallando cómo se establecen las solicitudes y respuestas a través de la API.

4.1.2.1 Conceptos transversales

La integración de la API REST se ha visto fuertemente influenciada por el patrón MVC. Se ha diseñado una clase clara y sencilla de usar, *OpenHabRestHandler*, la cual se encarga de delegar las peticiones HTTP a la clase *HttpHandler* y de gestionar las respuestas.

A continuación, para aclarar este concepto, se muestra un fragmento de código con un método denominado *startProgramCoffeeMaker* dentro de la clase *OpenHabRestHandler*, que implementa la interfaz *IOpenHabRestHandler*. Este método se encarga de iniciar un programa en la cafetera mediante una solicitud POST a través de la clase *HttpHandler*.

```
@Override
public void startProgramCoffeeMaker(HandlerCallback callback) {

   httpHandler.postRequest(CAFETERA_PROGRAM_ACTION_URL, "start", new HttpCallBack() {
       @Override
       public void onSuccess(Object obj) throws JSONException {
          Log.d("OpenHabRestHandler", "Programa de la cafetera iniciado correctamente.");
          callback.onSuccess(obj);
       }
       @Override
       public void onError(VolleyError error) {
          Log.d("OpenHabRestHandler", "Error al iniciar el programa de la cafetera: " +
error.getMessage());
          callback.onError();
       }
    });
}
```

Las peticiones HTTP las efectúa, por tanto, la clase HttpHandler, la cual hace peticiones GET, POST y PUT al servidor de OpenHAB según se le indique. Para conseguirlo se ha usado la librería Volley.

Tanto las peticiones de la Vista al Controlador como las del Controlador a la API REST de OpenHAB utilizan el patrón callback, con callbacks distintos para adaptarse a las necesidades específicas de cada componente.

4.1.2.2 Comunicación entre la aplicación, OpenHAB y Home Connect

Para clarificar la comunicación entre la aplicación móvil (cliente) y los servidores de OpenHAB y Home Connect, podemos observar el diagrama de secuencia de la figura 4-18.

En el ejemplo concreto del diagrama de secuencia, el usuario quiere seleccionar el programa “Cappuccino” de la cafetera y, posteriormente, hace clic en “Iniciar programa”.

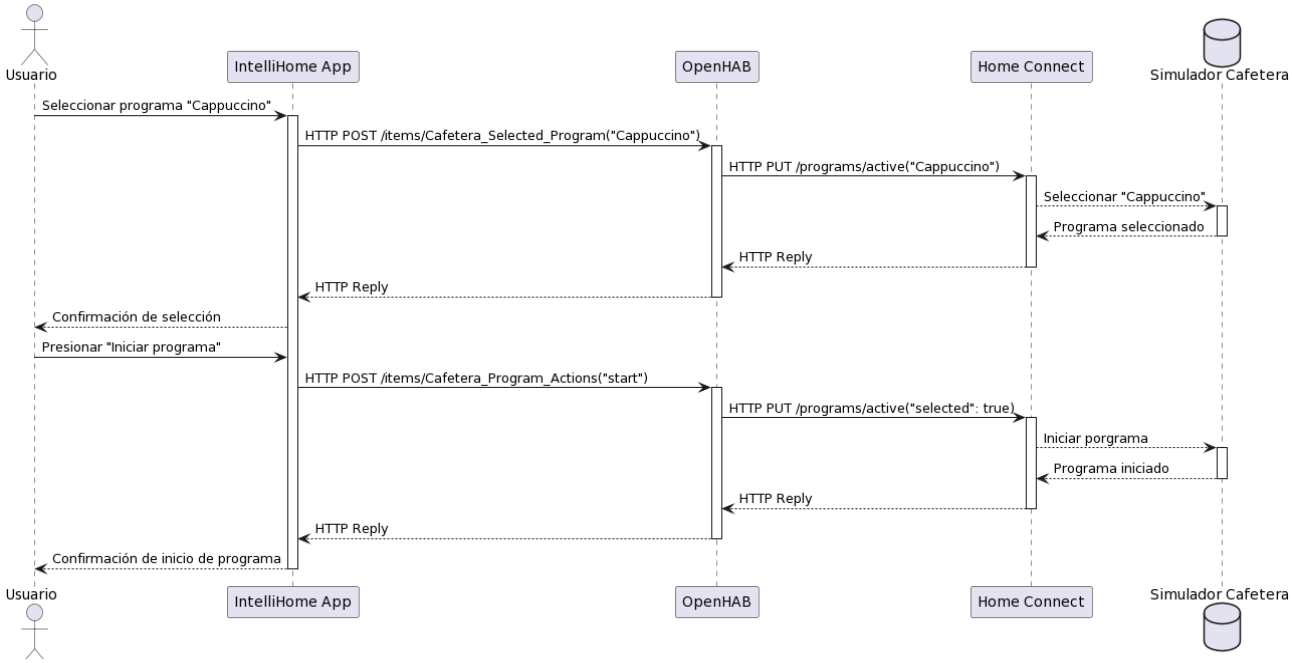

Figura 4-18. Diagrama de secuencia comunicación APP, OpenHAB y Home Connect

4.1.3 Conexión de la aplicación móvil con el servidor REST de consumo

En esta sección, se aborda el proceso de interacción entre la aplicación Android y el servidor REST desarrollado para el seguimiento del consumo energético de los electrodomésticos. Aquí se detalla cómo se establecen las consultas al servidor para obtener información actualizada sobre el consumo de energía de los dispositivos conectados. Este apartado es fundamental para comprender cómo se obtiene y presenta al usuario la información sobre el consumo de sus electrodomésticos.

4.1.3.1 Conceptos transversales

De forma análoga a la integración de la API REST de OpenHAB en nuestra aplicación, en este caso se ha diseñado otra clase, ConsumoRestHandler, la cual se encarga de delegar las peticiones HTTP a la clase HttpHandler y de gestionar las respuestas.

Las peticiones HTTP las efectúa, al igual que en el caso de OpenHAB, la clase HttpHandler, la cual hace peticiones GET al servidor de consumo usando la librería Volley.

Tanto las peticiones de la Vista al Controlador como las del Controlador al servidor REST de consumo utilizan el patrón callback, con callbacks distintos para adaptarse a las necesidades específicas de cada componente.

4.1.3.2 Comunicación entre la aplicación, el servidor de consumo y la base de datos

Para clarificar la comunicación entre la aplicación móvil (cliente), el servidor de consumo y la base de datos, podemos observar el diagrama de secuencia de la figura 4-19.

Concretamente, el usuario quiere obtener el gráfico de barras con el consumo energético de la cafetera. Esto desencadena 3 peticiones GET al servidor de consumo, el cual consulta la base de datos en consecuencia y devuelve los datos de consumo solicitados a la aplicación móvil, la cual imprime estos datos en forma de gráfica haciendo uso de la librería MPAndroidChart.

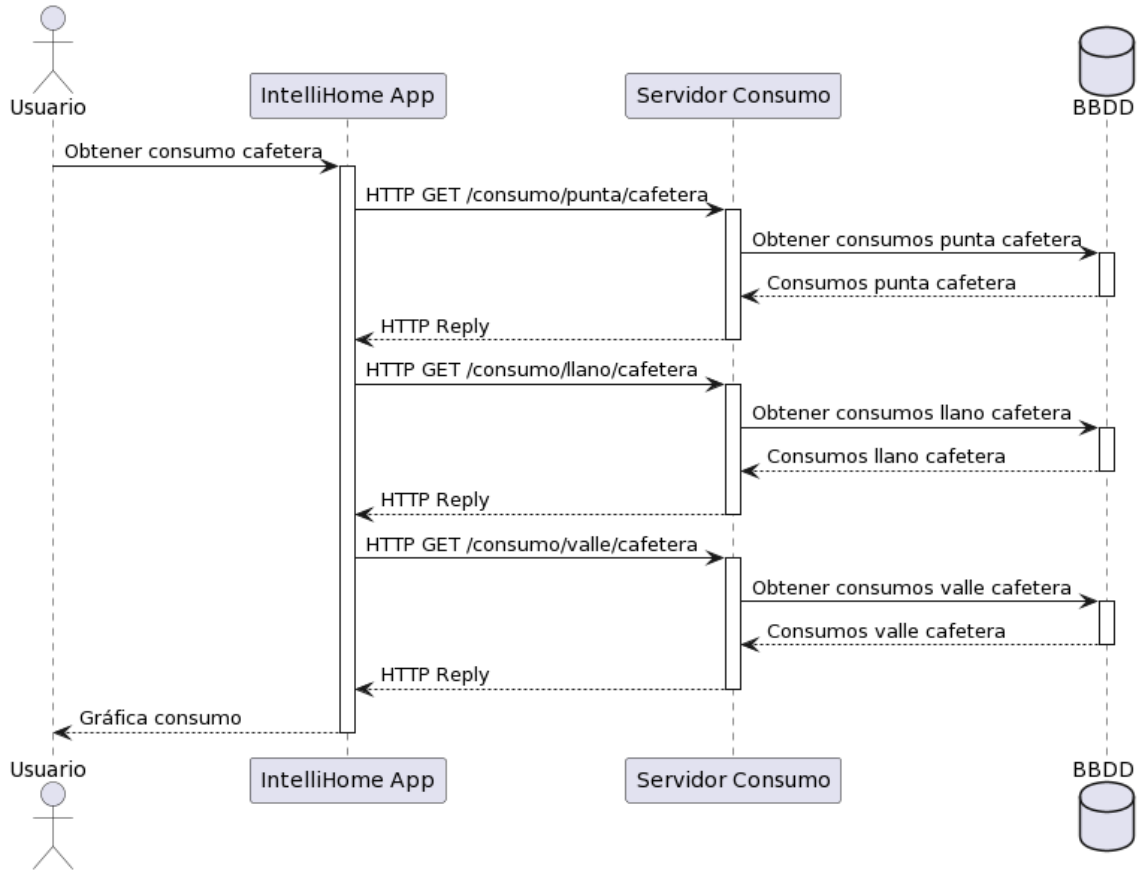

Figura 4-19. Diagrama de secuencia comunicación APP, servidor de consumo y BBDD

4.2 Servicio REST de consumo

Esta sección es otro de los grandes pilares de este proyecto. Se abordará en detalle las características esenciales del servicio que se ha implementado para supervisar el consumo de energía de los electrodomésticos. Aquí, se proporcionará una descripción exhaustiva de la estructura integral de este servicio, así como la forma en que interactúa con la base de datos subyacente. Además, se abordará la cuestión de la seguridad adoptada en la implementación, detallando las medidas tomadas para garantizar la integridad y confidencialidad de los datos en tránsito. Este apartado no solo ofrecerá una visión profunda del funcionamiento técnico del servicio REST de consumo, sino que también subrayará su importancia en el contexto más amplio de la aplicación, al brindar a los usuarios la capacidad de monitorear y optimizar su consumo energético de manera efectiva.

El código del servicio REST de consumo se puede descargar en el siguiente repositorio de GitHub:

https://github.com/bellozjur/serv-consumo-tfg.git

4.2.1 Servidor REST de consumo

Este subapartado se centra en explorar en detalle la configuración y estructura del servidor, incluyendo la integración con Spring Framework, la definición de endpoints y la creación de clases controladoras dedicadas. Además, se examinará cómo los endpoints permiten la interacción eficiente con la base de datos para la obtención y almacenamiento de información vital.

4.2.1.1 Dependencias

Las dependencias son elementos esenciales que permiten que el proyecto funcione correctamente. En el archivo POM (Project Object Model), se han incluido las siguientes dependencias relevantes:

- **PostgreSQL**: la dependencia de PostgreSQL se ha agregado para la integración con la base de datos.
- **Spring Security**: la inclusión de la dependencia de Spring Security permite la implementación de medidas de seguridad en la aplicación.

A continuación, se muestra un fragmento del archivo POM que refleja estas dependencias:

```
<!-- PostgreSQL -->
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.5.4</version>
</dependency>

<!-- Spring Security -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

4.2.1.2 Configuración del servidor

Para que la aplicación pueda ser accedida y utilizada, es necesario configurar la dirección y el puerto donde el servidor estará escuchando. En este proyecto, se ha configurado el servidor para que escuche en la dirección IP y el puerto específico:

- **Dirección IP:** localhost (127.0.0.1).
- **Puerto:** 8000 (indicado en el archivo application.properties).

Esta configuración permite que la aplicación sea accesible en la dirección http://localhost:8000.

4.2.1.3 Definición de Endpoints

En este apartado, se abordará una parte fundamental del desarrollo de la aplicación: la definición y descripción de los endpoints de la API REST implementada. Los endpoints son puntos de acceso a través de los cuales los usuarios y las aplicaciones pueden interactuar con el servidor. Cada endpoint representa una acción específica que la API puede llevar a cabo, permitiendo la comunicación y el intercambio de información de manera estructurada.

Los diferentes endpoints del servicio REST de consumo pueden verse en la tabla 3 que se muestra a continuación:

Endpoint	Método HTTP	Descripción
/consumo	POST	Permite crear un nuevo consumo en la tabla consumo de la base de datos.
/consumoTramo	POST	Permite crear un nuevo consumo en la tabla consumo_tramo de la base de datos.
/consumo/punta/{electrodomestico}	GET	Permite obtener una lista con todos los consumos punta del electrodoméstico indicado.
/consumo/llano/{electrodomestico}	GET	Permite obtener una lista con todos los consumos llano del electrodoméstico indicado.
/consumo/valle/{electrodomestico}	GET	Permite obtener una lista con todos los consumos valle del electrodoméstico indicado.
/login	POST	Permite autenticar a los usuarios en el sistema.

Tabla 3. Endpoints del servicio Consumo

Para crear un nuevo consumo o consumoTramo en necesario indicar la cabecera *"Content-Type: application/json"* y en el cuerpo del mensaje hay que enviar, por ejemplo:

- **POST a http://localhost:8000/consumo:**

Cuerpo:

{"electrodomestico": "Cafetera","fecha_inicio": "2023-07-26","hora_inicio": "23:59:40","fecha_fin": "2023-07-27","hora_fin": "00:00:10","consumo_punta": 0,"consumo_llano": 0.533,"consumo_valle": 0.266}

- **POST a http://localhost:8000/consumoTramo:**

Cuerpo:

{"electrodomestico": "Cafetera","fecha":"2023-07-26","consumo_punta": 0,"consumo_llano": 0.533, "consumo_valle": 0}

En cambio, para realizar una petición de consumo en horario punta:

- **GET a http://localhost:8000/consumo/punta/Cafetera.**

Un ejemplo de respuesta que podríamos obtener por parte del servidor a esta consulta sería:


```
[{"electrodomestico":"Cafetera","fecha":"2023-06-28",
"consumoTotalDiario":3.2}, {"electrodomestico":"Cafetera","fecha":"2023-06-30",
"consumoTotalDiario":2.4},
{"electrodomestico": "Cafetera","fecha":"2023-07-03", "consumoTotalDiario":37.25}]
```


4.2.1.4 Clases controladoras

Dentro del marco del desarrollo del servicio REST de consumo, se han implementado clases controladoras para gestionar las diversas funcionalidades de manera organizada y eficiente. Estas clases actúan como intermediarias entre las solicitudes realizadas por la aplicación móvil o el servidor de OpenHAB y la lógica subyacente.

4.2.1.4.1 Consumo Controller

ConsumoController.java es una clase fundamental que se encarga de gestionar las solicitudes relacionadas con el consumo energético de los electrodomésticos. A través de una cuidadosa configuración de endpoints y métodos, este controlador permite que la aplicación móvil envíe solicitudes para obtener información detallada sobre el consumo almacenada en la base de datos o, en su defecto, permite al servidor OpenHAB ingresar nuevos registros de consumo en la base de datos.

Esta clase implementa 5 métodos que podemos ver a continuación en la tabla 4:

Métodos	Endpoint	Descripción
creaConsumo	/consumo	Permite crear un nuevo consumo en la tabla consumo de la base de datos. Este método recibe y devuelve un objeto de tipo Consumo.
creaConsumoTramo	/consumoTramo	Permite crear un nuevo consumo (dividido) en la tabla consumo_tramo de la base de datos. Es decir, si el consumo ocupa varios tramos horarios, se crea un consumo para cada horario (ver ejemplo en la siguiente página). Este método recibe y devuelve un objeto de tipo ConsumoTramo.
getConsumoPunta	/consumo/punta/{electrodomestico}	Permite obtener una lista con todos los consumos punta del electrodoméstico indicado. Este método recibe el nombre del electrodoméstico y devuelve una lista de objetos de tipo ConsumoTotalDiario.

getConsumoLlano	/consumo/llano/{electrodomestico}	Permite obtener una lista con todos los consumos llano del electrodoméstico indicado. Este método recibe el nombre del electrodoméstico y devuelve una lista de objetos de tipo ConsumoTotalDiario.
getConsumoValle	/consumo/valle/{electrodomestico}	Permite obtener una lista con todos los consumos valle del electrodoméstico indicado. Este método recibe el nombre del electrodoméstico y devuelve una lista de objetos de tipo ConsumoTotalDiario.

Tabla 4. Métodos Consumo Controller

Ejemplo de división de un registro de consumo en varios consumos por tramo:

- **Consumo:**

{"electrodomestico": "Cafetera","fecha_inicio": "2023-07-26","hora_inicio": "23:59:40","fecha_fin": "2023-07-27","hora_fin": "00:00:10","consumo_punta": 0,"consumo_llano": 0.533,"consumo_valle": 0.266}

Para este registro de consumo que tendríamos almacenado en la tabla "consumo" de la base de datos, en la tabla "consumo_tramo" se vería dividido en 2 consumos: uno para registrar los 0.533 kWmin de horario llano y otro para el 0.266 kWmin de horario valle. Esto se ha implementado de esta manera para poder tener un cálculo preciso en el caso de aquellos consumos que puedan ocupar distintos días, es decir, que comience un día X el consumo y finalice un día Y, como el presente ejemplo. De esta manera podremos determinar el consumo exacto que se ha realizado cada día.

- **Consumos por tramo:**
  - Punta:

{"electrodomestico": "Cafetera","fecha":"2023-07-26","consumo_punta": 0,"consumo_llano": 0.533,"consumo_valle": 0}

  - Llano:

{"electrodomestico": "Cafetera","fecha":"2023-07-27","consumo_punta": 0,"consumo_llano": 0,"consumo_valle": 0.266}

Las clases Consumo, ConsumoTramo y ConsumoTotalDiario tienen los siguientes constructores:

```
//Consumo: consumo original (sin dividir por tipo de consumo según su tramo horario)
public Consumo(int id, String electrodomestico, LocalDate fecha_inicio, LocalTime hora_inicio, LocalDate fecha_fin, LocalTime hora_fin, double consumo_punta, double consumo_llano, double consumo_valle) {...}

//ConsumoTramo: consumo dividido en tramos (sólo un tipo de consumo por registro)
public ConsumoTramo(int id, String electrodomestico, LocalDate fecha, double consumo_punta, double consumo_llano, double consumo_valle) {...}
```

```
//ConsumoTotalDiario: suma de todos los consumos registrados en un día concreto,
por tramos (lo que se imprime en el gráfico de barras)
public ConsumoTotalDiario(String electrodomestico, LocalDate fecha, double
consumo_total_diario) {...}
```

4.2.1.4.2 Login Controller

La clase **LoginController.java** desempeña un papel fundamental en la seguridad y autenticación de la aplicación. Encargada de manejar las solicitudes de inicio de sesión (POST al endpoint /login), este controlador verifica las credenciales proporcionadas por el usuario y responde con una cadena de texto.

```
@Autowired
private UserRepository userRepository;

@Autowired
private BCryptPasswordEncoder bCryptPasswordEncoder;

@PostMapping("/login")
public ResponseEntity<String> login(@RequestBody Usuario usuario) {
    Usuario user = null;

    try {
        user = userRepository.findByUsername(usuario.getUsuario());
    } catch (SQLException e) {
        e.printStackTrace();
    }

    if (user == null) {
        System.out.println("Usuario no encontrado");
        return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Usuario no encontrado");
    }

    if (bCryptPasswordEncoder.matches(usuario.getContrasenia(), user.getContrasenia())) {
        // Autenticación exitosa
        System.out.println("Inicio de sesión exitoso");
        return ResponseEntity.ok("Inicio de sesión exitoso");
    } else {
        // Contraseña incorrecta
        System.out.println("Contraseña incorrecta");
        return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Contraseña incorrecta");
    }
}
```

Este código se entenderá mejor con el diagrama de secuencia del apartado 4.2.2.4.

La clase Usuario tiene el siguiente constructor:

```
public Usuario(String usuario, String contrasenia) {
        this.usuario = usuario;
        this.contrasenia = contrasenia;
}
```

4.2.2 Spring Security

La protección de datos y recursos es una prioridad en el diseño de cualquier aplicación. Este subapartado se enfoca en cómo se ha implementado la seguridad utilizando Spring Security para el servidor REST de Consumo. Se detallarán los procedimientos de autenticación y autorización que aseguran que los endpoints solo sean accesibles por usuarios autorizados. Mediante este enfoque, se garantiza que los datos sensibles estén protegidos y solo puedan ser accedidos por aquellos que hayan iniciado sesión correctamente.

Una vez añadida la dependencia indicada en el apartado 4.2.1.1 necesaria para implementar Spring Security en nuestro servidor, se procede a la creación de las clases *SecurityConfig*, *UserDetailsServiceImpl* y *UserRepository* explicadas a continuación:

4.2.2.1 SecurityConfig

Esta clase representa una configuración de seguridad, donde se establecen las reglas de acceso y autenticación para las rutas y recursos protegidos. Concretamente:

- Se inyecta la interfaz *UserDetailsService* para obtener los detalles del usuario.
- Se define un bean *BCryptPasswordEncoder* que proporciona un algoritmo de hash[4] para cifrar contraseñas.
- Se crea un filtro de seguridad (*SecurityFilterChain*) para controlar el acceso a las diferentes rutas.
- En *authorizeHttpRequests*, se establece que la ruta "/login" es accesible sin autenticación, mientras que otras rutas requieren autenticación.
- Se configura la autenticación básica mediante *httpBasic(),* es decir, se requiere usuario y contraseña.
- Se define un *DaoAuthenticationProvider* que utiliza el *UserDetailsService* y el cifrador de contraseñas.
- Se configura un *AuthenticationManager* para gestionar la autenticación.

```
@Configuration
public class SecurityConfig {

   @Autowired
      private UserDetailsService userDetailsService;

   @Bean
   public BCryptPasswordEncoder passwordEncoder() {
      return new BCryptPasswordEncoder();
   }

   @Bean
   public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
      http
      .authorizeHttpRequests((authorize) -> authorize
         .requestMatchers("/login").permitAll()
         .anyRequest().authenticated()
      )
      .httpBasic()
         .and()
      .csrf(AbstractHttpConfigurer::disable);
```

[4] Un hash es una salida única y fija de un algoritmo de hash que transforma datos en una cadena de caracteres, esencial para la verificación y la seguridad de datos, como contraseñas o verificación de integridad.

```
        return http.build();
    }

    @Bean
    public DaoAuthenticationProvider authenticationProvider() {
        DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
        authProvider.setUserDetailsService(userDetailsService);
        authProvider.setPasswordEncoder(passwordEncoder());
        return authProvider;
    }

    @Bean
    public AuthenticationManager authenticationManager(AuthenticationConfiguration authConfig)
throws Exception {
        return authConfig.getAuthenticationManager();
    }
}
```

4.2.2.2 UserDetailsServiceImpl

Esta clase llamada *UserDetailsServiceImpl* es una implementación de *UserDetailsService*, una interfaz en Spring Security que maneja la carga y validación de detalles de usuarios. Utiliza un repositorio de usuarios *UserRepository* para buscar y cargar información de usuario desde la base de datos. Luego, construye un objeto *UserDetails* para la autenticación y autorización, incluyendo el nombre de usuario, contraseña y roles asociados.

```
@Service
public class UserDetailsServiceImpl implements UserDetailsService {

    @Autowired
    private UserRepository userRepository;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        // Implementa la carga de usuarios desde la base de datos
        Usuario usuario = null;
        try {
            usuario = userRepository.findByUsername(username);
        } catch (SQLException e) {
            throw new UsernameNotFoundException("Error al buscar usuario en la base de datos", e);
        }
        if (usuario == null) {
            throw new UsernameNotFoundException("Usuario no encontrado: " + username);
        }
        return User.builder()
            .username(usuario.getUsuario())
            .password(usuario.getContrasenia())
            .roles("USER")
            .build();
    }
}
```

4.2.2.3 UserRepository

Esta clase actúa como un repositorio para acceder y recuperar información de usuario desde nuestra base de datos. Utiliza una consulta SQL para buscar un usuario por su nombre de usuario en la tabla *usuarios*. Si se encuentra una coincidencia, crea un objeto Usuario con los datos recuperados y lo devuelve. Esta clase está etiquetada con la anotación @Repository, indicando que es un componente de acceso a datos y permitiendo que Spring lo administre como un *bean* dentro del contexto de la aplicación.

```
@Repository
public class UserRepository {

    //Devuelve el Usuario indicado por el nombre de usuario
    public Usuario findByUsername(String username) throws SQLException {

        final String QUERY_USUARIO = "SELECT * FROM usuarios WHERE usuario = ?";

        Connection conn = null;
        ResultSet rs = null;
        PreparedStatement st = null;

        Usuario u = null;

        conn = Conexion.conecta();
        st = conn.prepareStatement(QUERY_USUARIO);
        st.setString(1, username);

        rs = st.executeQuery();

        if (rs.next()) {
            u = new Usuario(rs.getString("usuario"), rs.getString("contrasenia"));
        }

        return u;
    }
}
```

4.2.2.4 Diagrama de secuencia inicio de sesión

Para aclarar el código descrito en el apartado 4.2.1.4.2 Login Controller, se presenta a continuación un diagrama de secuencia con la comunicación detallada entre las diferentes clases que intervienen en el proceso de inicio de sesión. El usuario rellena las credenciales y hace clic en INICIAR SESIÓN en la pantalla *Login*. Seguidamente, el servidor a través de las diferentes clases de las que se compone hace una serie de comprobaciones que podemos ver al detalle en la figura 4-20.

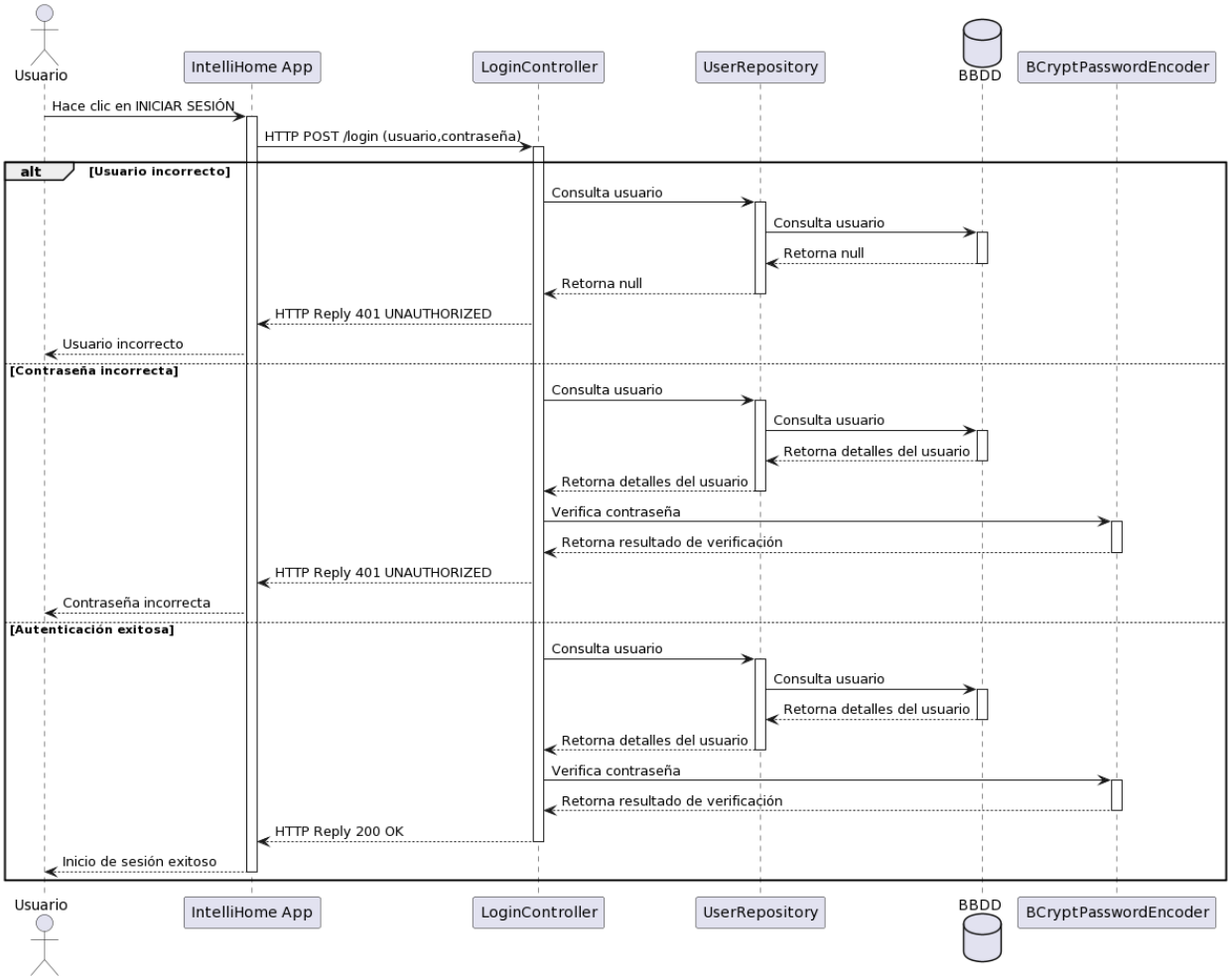

Figura 4-20. Diagrama de secuencia inicio de sesión

4.2.2.5 Diagrama de secuencia consulta consumo

Por otro lado, para ver la comunicación entre las distintas clases de configuración de Spring Security, se presenta otro diagrama de secuencia que refleja una petición del usuario el cual desea obtener la gráfica de consumo de la cafetera. Para ello, el servidor debe confirmar que el usuario está autorizado para realizar esa consulta y seguidamente, tras ser autorizado, el servidor le proporcionará los datos que solicita (ver figura 4-21).

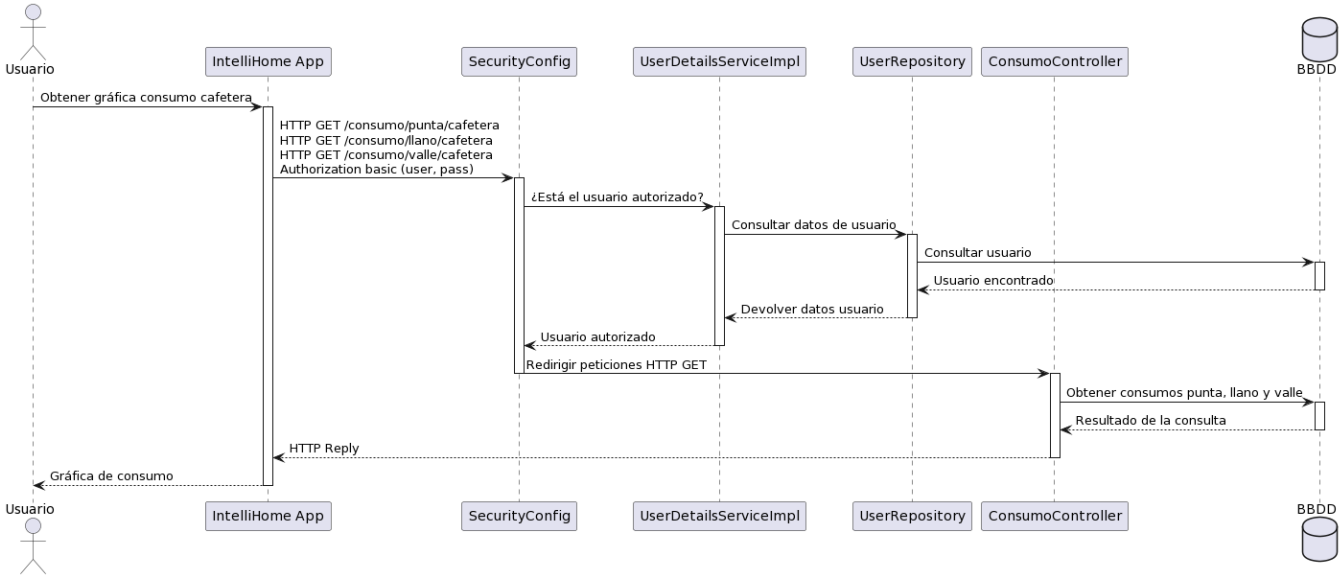

Figura 4-21. Diagrama de secuencia consulta consumo

4.2.2.6 Ejemplo de uso de Spring Security

Por último, se va a proporcionar un ejemplo de uso de Spring Security para verificar su correcta configuración.

Como hemos mencionado en apartados anteriores, al **endpoint /login** todo el mundo puede mandarle peticiones sin restricción.

- Si mandamos las credenciales correctas de un usuario, recibimos 200 OK por parte del servidor (ver figura 4-22).
- En cambio, si enviamos las credenciales incorrectas, recibimos 401 UNAUTHORIZED (ver figura 4-23).

Sin embargo, para el **resto de endpoints** del servidor, se necesita que las peticiones estén autorizadas para poder brindar una respuesta válida.

- Si intentamos hacer una petición GET al **endpoint /consumo/punta/cafetera** nos pedirá que nos autentiquemos como podemos observar en la figura 4-24.
- Al introducir las credenciales válidas y hacer clic en “Iniciar sesión”, ya nos permitirá obtener una respuesta por parte del servidor (ver figura 4-25).

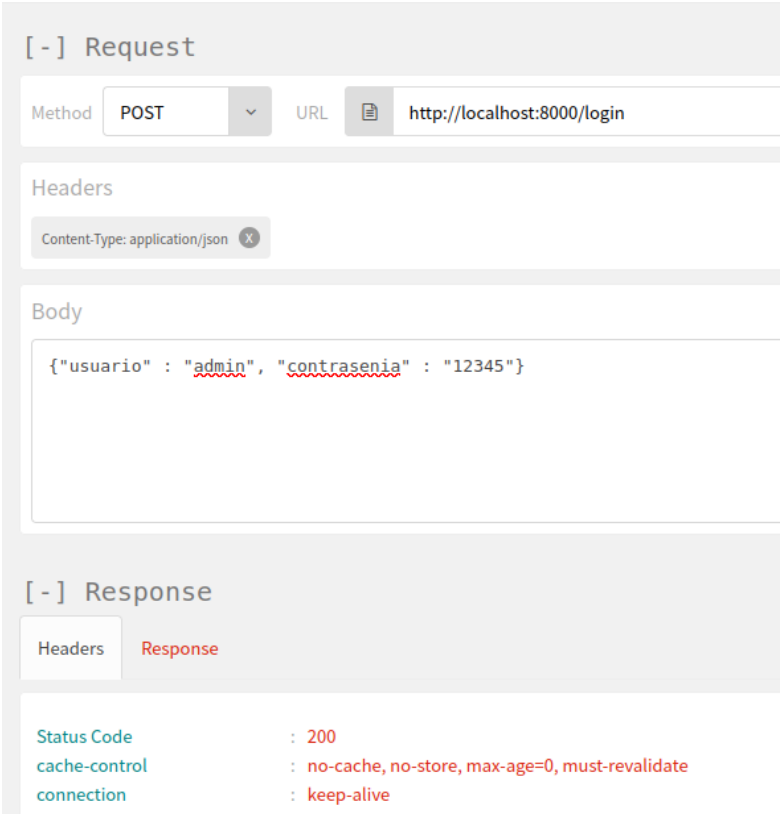

Figura 4-22. Petición /login correcta

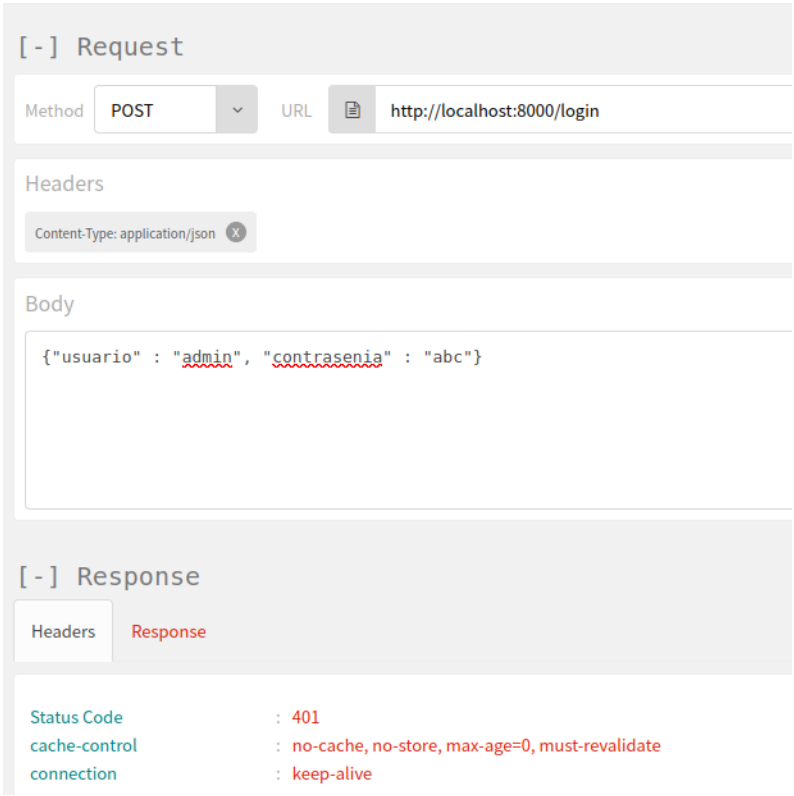

Figura 4-23. Petición /login incorrecta

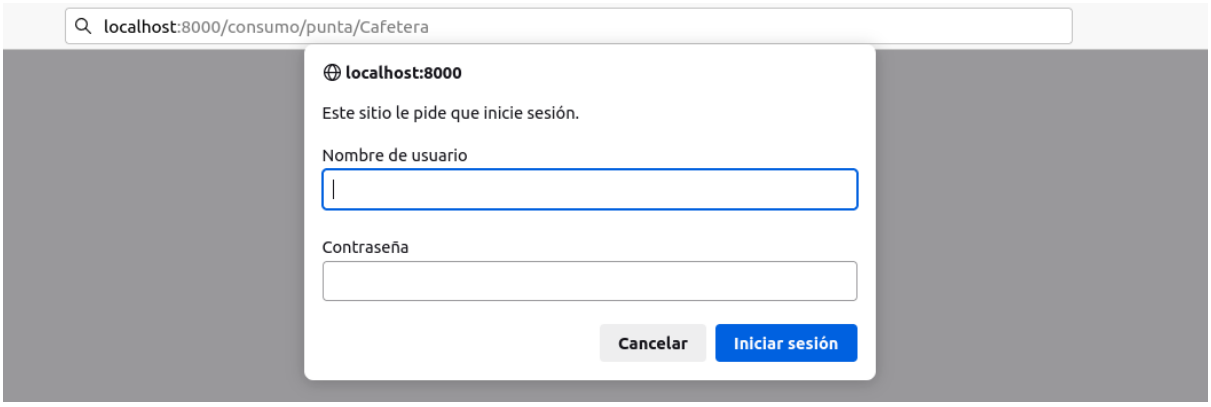

Figura 4-24. Petición de credenciales

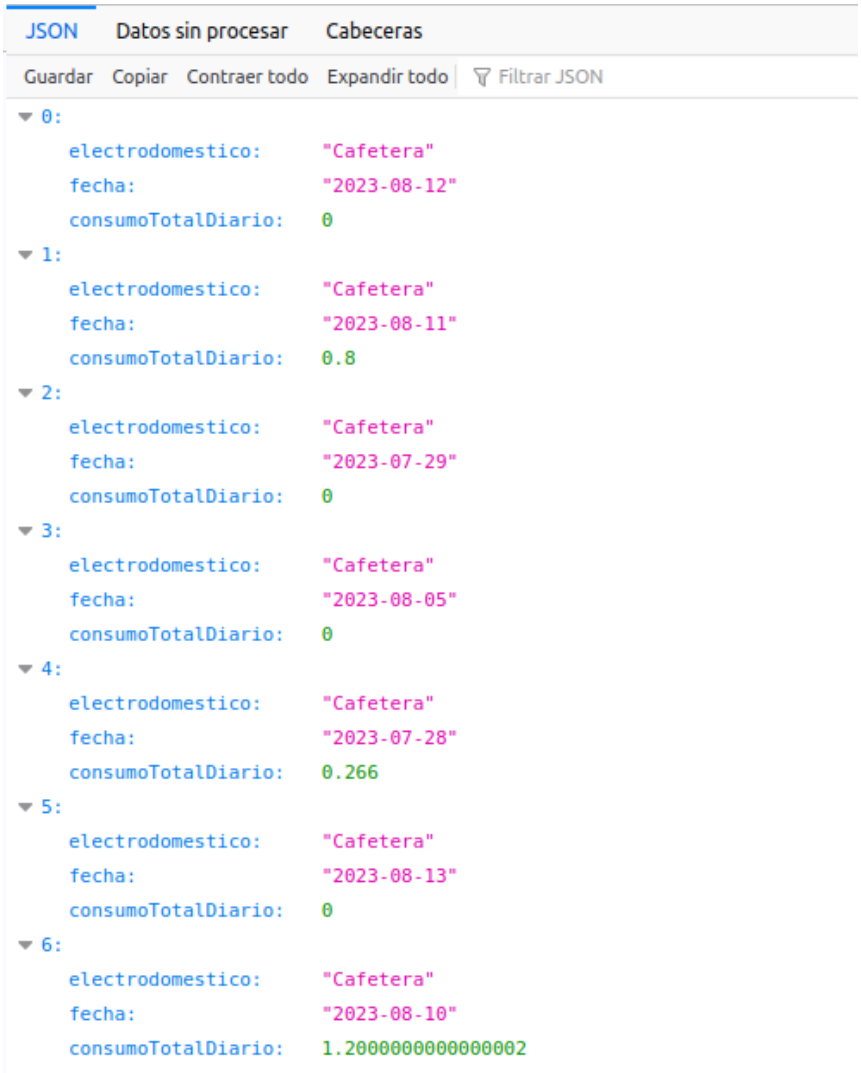

Figura 4-25. Respuesta correcta servidor

4.2.3 Base de datos

En este apartado, se desglosará la implementación de la base de datos, que juega un papel fundamental en la conservación de registros de inicio de sesión y los datos de consumo de los electrodomésticos. A través de una estructura sólida, se logra capturar y mantener información relevante, permitiendo un análisis detallado del comportamiento energético en el hogar.

4.2.3.1 Tabla consumo

La tabla "consumo", como ya hemos mencionado en apartados anteriores, se utiliza para almacenar información relacionada con el consumo de energía de los electrodomésticos. Cada fila en la tabla representa un registro de consumo y contiene detalles como el ID del registro, el nombre del electrodoméstico, las fechas y horas de inicio y fin del consumo, y los valores de consumo en diferentes tramos horarios (punta, llano y valle) en unidades de kilovatios por minuto (el motivo de usar esta atípica unidad de consumo se debe a que los tiempos de simulación de los electrodomésticos son muy pequeños). La tabla se crea utilizando un conjunto de columnas con diferentes tipos de datos, como enteros, cadenas y flotantes, para representar los diversos atributos del consumo. El código SQL de esta tabla sería:

```
DROP TABLE IF EXISTS consumo;
CREATE TABLE consumo (
  id SERIAL PRIMARY KEY,
  electrodomestico VARCHAR(255),
  fecha_inicio DATE,
  hora_inicio TIME,
  fecha_fin DATE,
  hora_fin TIME,
  consumo_punta FLOAT, --kW min
  consumo_llano FLOAT, --kW min
  consumo_valle FLOAT --kW min
);
```

Por otro lado, un ejemplo de inserción puede ser el siguiente:

```
INSERT INTO consumo (electrodomestico, fecha_inicio, hora_inicio, fecha_fin, hora_fin,
consumo_punta, consumo_llano, consumo_valle)
VALUES ('Cafetera', '2023-07-28', '17:59:55', '2023-07-28', '18:00:10', 0.266, 0.133, 0),
       ('Cafetera', '2023-07-28', '23:59:50', '2023-07-29', '00:00:05', 0, 0.266, 0.133);
```

4.2.3.2 Tabla consumo_tramo

La tabla "consumo_tramo" está destinada a almacenar los registros de consumo de energía de los electrodomésticos, pero en este caso, se agrupa por día y no por intervalos de tiempo específicos. La tabla contiene columnas como ID, nombre del electrodoméstico, fecha del consumo y los valores de consumo en diferentes tramos horarios (punta, llano y valle) expresados en kilovatios por minuto (el motivo de usar esta atípica unidad de consumo se debe a que los tiempos de simulación de los electrodomésticos son muy pequeños). La utilidad de esta tabla es analizar el consumo diario de los electrodomésticos en función de los diferentes tramos horarios, y según está programada su inserción (se explicará en el apartado 4.3 del documento), por cada fila solo puede haber un tipo de consumo que sea distinto de 0. El código SQL de esta tabla sería:

```
DROP TABLE IF EXISTS consumo_tramo;
CREATE TABLE consumo_tramo (
  id SERIAL PRIMARY KEY,
  electrodomestico VARCHAR(255),
```

```
  fecha DATE,
  consumo_punta FLOAT, --kW min
  consumo_llano FLOAT, --kW min
  consumo_valle FLOAT --kW min
);
```

Por otro lado, un ejemplo de inserción puede ser el siguiente:

```
INSERT INTO consumo_tramo (electrodomestico, fecha, consumo_punta, consumo_llano,
consumo_valle)
VALUES ('Cafetera', '2023-07-28', 0, 0.133, 0),
      ('Cafetera', '2023-07-28', 0.266, 0, 0),
      ('Cafetera', '2023-07-28', 0, 0.266, 0),
      ('Cafetera', '2023-07-29', 0, 0, 0.133);
```

4.2.3.3 Tabla usuarios

La tabla “usuarios” es fundamental para la implementación del sistema de autenticación y seguridad, ya que almacena las credenciales de los usuarios de manera segura y permite el acceso controlado a la aplicación y a los recursos protegidos. Cada entrada en esta tabla corresponde a un usuario, y se caracteriza por tres campos principales. El primero, "id", es una clave única generada automáticamente para cada usuario, lo que garantiza la unicidad de cada registro. El segundo campo, "usuario", almacena el nombre del usuario y es definido como un valor único, asegurando que no haya duplicados. Finalmente, el tercer campo, "contrasenia", guarda la contraseña del usuario, la cual es encriptada y almacenada en formato de texto cifrado. El código SQL de esta tabla sería:

```
DROP TABLE IF EXISTS usuarios;
CREATE TABLE usuarios (
  id SERIAL PRIMARY KEY,
  usuario VARCHAR(50) NOT NULL UNIQUE,
  contrasenia VARCHAR(100) NOT NULL
);
```

Por otro lado, un ejemplo de inserción puede ser el siguiente:

```
INSERT INTO usuarios (usuario, contrasenia)
VALUES ('admin', '$2a$10$LpPk0W6FfppYF08gdXMWf.nb/WL.Mhz4CUgtIkB2KU0krEPdt.lNa'),
      ('bellozjur', '$2a$10$mAB550KPze0kfiYJWCTLReSxKoVF/wqt6FW8bSJqe5V6jXSpBbmpO');
```

4.2.3.4 Clases DAO

En la estructura del proyecto, las clases DAO (Data Access Object) desempeñan un papel vital al interactuar con la base de datos. Estas clases actúan como intermediarios entre la lógica de negocio de la aplicación y la capa de persistencia de datos. Proporcionan métodos para realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) en la base de datos, asegurando un manejo eficiente y seguro de los datos (en nuestro caso sólo se ha necesitado implementar métodos de creación y lectura).

4.2.3.4.1 DAO de Consumo

El DAO de Consumo se encarga de gestionar la interacción con la tabla "consumo" en la base de datos. A través de su interfaz, se definen métodos para crear y leer registros de consumo. La función **creaConsumo()** permite agregar nuevas entradas a la tabla de consumo, mientras que **leeConsumo()** recupera información específica de un registro de consumo según su ID. Su interfaz es la siguiente:

```
public interface IConsumoDAO {
    public Consumo creaConsumo(Consumo consumo) throws SQLException;
    public Consumo leeConsumo(int id) throws SQLException;
}
```

4.2.3.4.2 DAO de ConsumoTramo

Por otro lado, el DAO de ConsumoTramo administra la manipulación de la tabla "consumo_tramo". En su interfaz, se han definido varios métodos esenciales: **creaConsumoTramo()** posibilita la inserción de nuevos registros de consumo por tramo horario. Además, los métodos **obtieneConsumoPunta(), obtieneConsumoLlano()** y **obtieneConsumoValle()** permiten obtener los registros de consumo específicos para los diferentes tramos horarios: punta, llano y valle. Finalmente, **leeConsumoTramo()** recupera un registro de consumo tramo mediante su ID. Su interfaz es la siguiente:

```
public interface IConsumoTramoDAO {
    public ConsumoTramo creaConsumoTramo(ConsumoTramo ConsumoTramo) throws SQLException;
    public List<ConsumoTotalDiario> obtieneConsumoPunta(String electrodomestico) throws SQLException;
    public List<ConsumoTotalDiario> obtieneConsumoLlano(String electrodomestico) throws SQLException;
    public List<ConsumoTotalDiario> obtieneConsumoValle(String electrodomestico) throws SQLException;
    public ConsumoTramo leeConsumoTramo(int id) throws SQLException;
}
```

4.3 Estimación del consumo energético

En este apartado, exploraremos una faceta fundamental de nuestro proyecto: la estimación del consumo energético de los electrodomésticos basada en la segmentación en tramos horarios de punta, llano y valle. Abordaremos detalladamente la concepción y configuración de las reglas en el entorno OpenHAB, que desempeñan un papel crucial en la obtención de una aproximación precisa del consumo. Además, examinaremos en profundidad cómo estas reglas permiten realizar cálculos estimativos y cómo han sido diseñadas para reflejar de manera eficaz la distribución de consumo energético a lo largo del día.

4.3.1 Tramos de consumo energético

Ante el auge de la tarificación del consumo energético por discriminación horaria, se ha decidido en este proyecto dividir el consumo en los tres tramos horarios fundamentales: punta (el más caro), llano (intermedio) y valle (el más barato). Esta segmentación permite capturar de manera más precisa la variabilidad del consumo eléctrico a lo largo del día, considerando los periodos de mayor y menor demanda.

Además de la segmentación en los tramos de consumo, es fundamental tener en cuenta los horarios en los que se aplican estas tarifas diferenciadas. En este contexto, los horarios de la luz se establecen de la siguiente manera [25]:

- **Periodo punta**: este es el tramo más costoso y se aplica en los horarios de 10:00 a 14:00 y de 18:00 a 22:00 horas, de lunes a viernes no festivos. Durante este periodo, el costo de la electricidad es más alto debido a la mayor demanda energética.
- **Periodo llano**: este tramo representa una tarifa intermedia y está vigente desde las 08:00 a las 10:00, de 14:00 a 18:00 y de 22:00 a 24:00 horas, en días laborables no festivos. Es un rango que abarca tanto momentos de demanda moderada como de mayor utilización de energía.
- **Periodo valle:** es la tarifa más asequible, el periodo valle se extiende desde la medianoche, las 00:00 horas, hasta las 08:00 horas todos los días y las 24 horas los fines de semana y festivos. Esto incentiva el uso de electrodomésticos en los momentos de menor demanda eléctrica.

Estos tramos de consumo y sus horarios asociados no solo son elementos fundamentales para calcular de manera precisa el costo de uso de los electrodomésticos, sino que también brindan la oportunidad de implementar estrategias de ahorro energético alentando a los usuarios a realizar actividades de mayor consumo en los momentos de tarifas más económicas. En el siguiente apartado, exploraremos cómo se han configurado y gestionado estas tarifas en el contexto de OpenHAB.

4.3.2 Creación y configuración de reglas de OpenHAB

Antes de sumergirnos en los detalles del código que nos permite realizar estimaciones precisas del consumo eléctrico, es esencial establecer un marco contextual que garantice un entendimiento completo de este proceso. Las reglas en OpenHAB son una herramienta clave en la automatización y el control del hogar inteligente. Permiten definir acciones a realizar en función de ciertas condiciones predefinidas. En este apartado, exploraremos cómo hemos utilizado estas reglas para estimar el consumo eléctrico en los tramos de tarificación horaria, brindando una visión más amplia de cómo la tecnología de OpenHAB puede ser aprovechada para optimizar el uso de electrodomésticos y ahorrar energía.

A continuación, se van a indicar los pasos procedentes para la creación y configuración correcta de reglas en OpenHAB:

1. Descargamos el complemento *JavaScript Scripting* para poder codificar las reglas en JavaScript y estas poder ser procesadas por el motor de JavaScript. Para ello, nos vamos a la interfaz web de OpenHAB y hacemos clic en: Configuración -> Add-ons -> Automation -> Languages & Technologies, y ahí buscamos e instalamos *JavaScript Scripting*.
2. Seguidamente, nos vamos al apartado Rules y le damos a crear una nueva regla (ver figura 4-26).

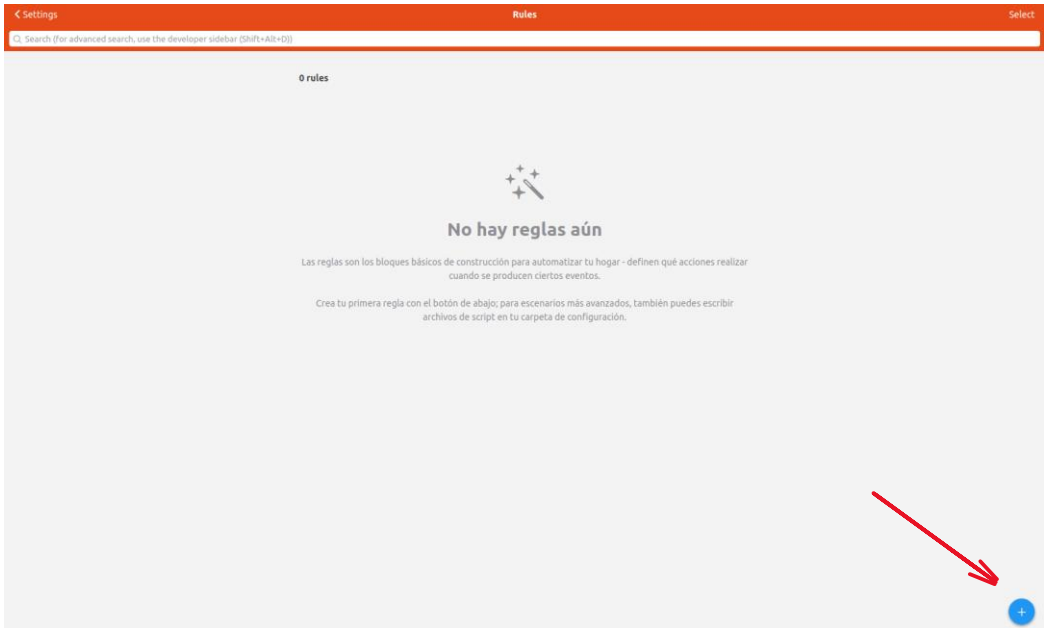

Figura 4-26. Crear nueva rule OpenHAB

3. En el siguiente paso, debemos rellenar los apartados *Name* y *Description*. Seguidamente, en el apartado *When* añadimos un *trigger* (disparador) para que se ejecute la regla cuando ocurra ese evento. Concretamente, este ejemplo vamos a configurarlo para que cuando el *item* "Cafetera_Operation_State" pase del estado *Ready* al estado *Run* (inicio de un programa de la cafetera), se ejecute un script en JavaScript (*Then*). En la figura 4-27 podemos ver cómo debería quedarse esta configuración.

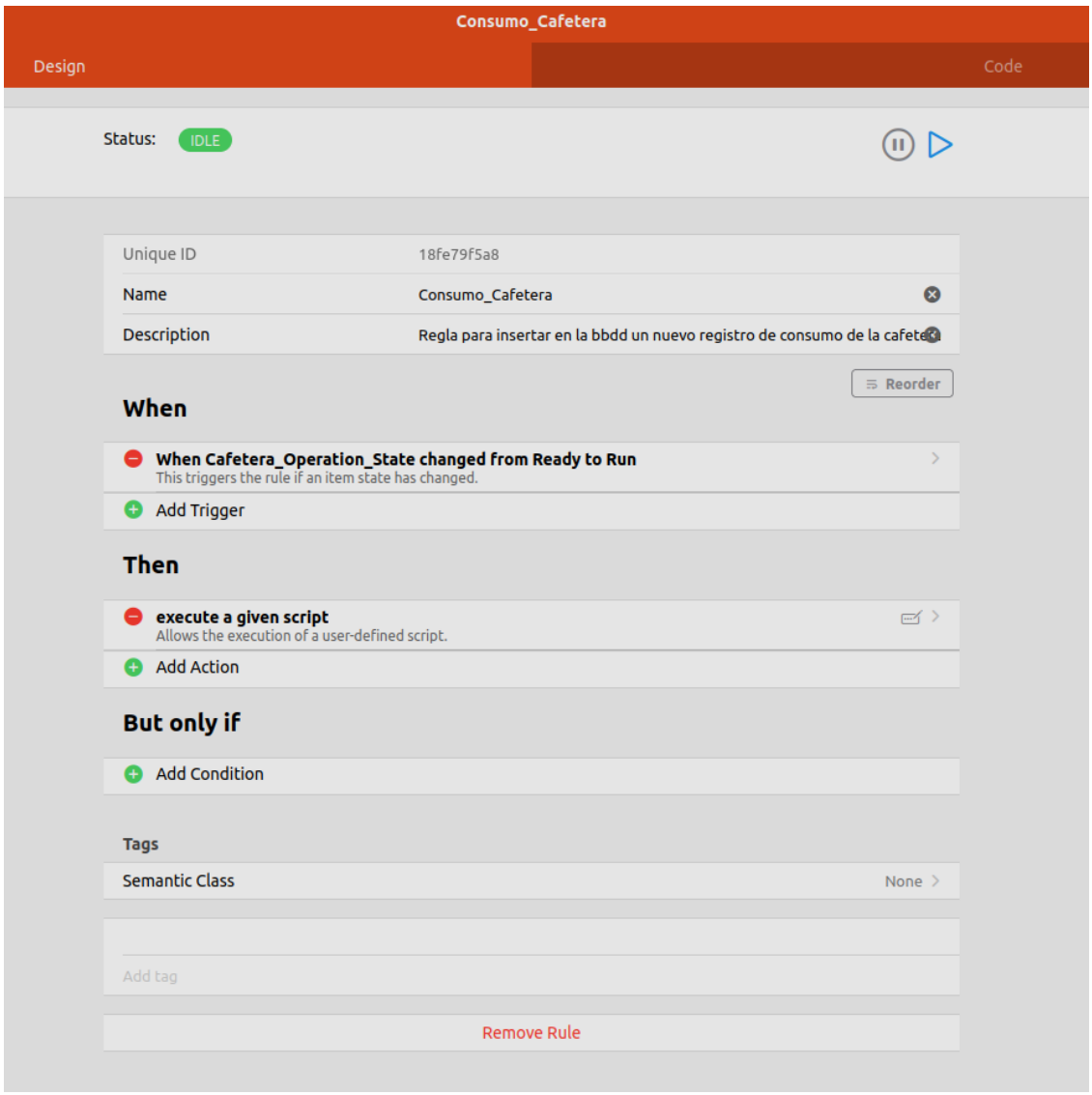

Figura 4-27. Configuración rule OpenHAB

4. Por último, si hacemos clic en la pestaña de edición que aparece en el extremo derecho del apartado *Then:* "execute a given script", se nos abrirá un script vacío donde tendremos que introducir el código que queremos que se ejecute cada vez que ocurra el evento que hemos configurado en el paso anterior.

El código JavaScript que se ha creado para cada uno de los electrodomésticos de Home Connect sigue una estructura similar. En el siguiente repositorio de GitHub se pueden descargar los 6 scripts correspondientes a los 6 electrodomésticos de los que disponemos:

https://github.com/bellozjur/reglas_openhab.git

A continuación, se van a comentar las características fundamentales del código usado para el cálculo estimado del consumo eléctrico de la cafetera.

En el primer fragmento de código, podemos identificar lo siguiente:

- Se crea un objeto llamado *holidays* que mapea fechas específicas (mes, día) con sus nombres de festividades correspondientes. Esto permitirá identificar si una fecha determinada es un día festivo.
- Se definen dos arrays: *TRAMO_L* para tramos en días laborables y *TRAMO_NL* para tramos en días no laborables (sábados, domingos y festivos). Cada elemento en los arrays contiene una tupla con dos elementos: la hora y el tipo de tramo horario ('P' para punta, 'L' para llano y 'V' para valle).
- Se define la potencia de la cafetera en vatios (W), lo que será esencial para calcular el consumo energético en cada tramo.
- Se importa la clase Java necesaria para realizar peticiones HTTP en OpenHAB. Luego, se definen credenciales de autenticación básica (nombre de usuario y contraseña) y se codifican en base64 para ser utilizadas en las cabeceras de las peticiones HTTP.
- Se crea un mapa de cabeceras HTTP que contiene la cabecera "Authorization" con las credenciales básicas codificadas. Esto es crucial para autenticar las solicitudes HTTP hacia el servidor.

```
//Objeto clave, valor con los días festivos
var holidays = {
   "1,1": "Año Nuevo",
   "1,6": "Epifanía del Señor",
   "8,15": "Asunción de la Virgen",
   "10,12": "Fiesta Nacional de España",
   "11,1": "Todos los Santos",
   "12,6": "Día de la Constitución Española",
   "12,8": "Inmaculada Concepción",
};

//Arrays de tuplas de 2 elementos (hora, tramo)
var TRAMO_L = [[0, 'V'], [8, 'L'], [10, 'P'], [14, 'L'], [18, 'P'], [22, 'L']]; // Tramos laborables L-V (Punta, Valle, Llano)
var TRAMO_NL = [[0, 'V']]; //Tramos no laborables S-D y festivos (Valle)

//Potencia de la cafetera
var potencia_cafetera = 1600; // W

//HTTP
var http = Java.type('org.openhab.core.model.script.actions.HTTP');
var username = "admin";
var password = "12345";
var basicAuthCredentials = java.util.Base64.getEncoder().encodeToString((username + ":" + password).getBytes());

var headers = new java.util.HashMap();
headers.put("Authorization", "Basic " + basicAuthCredentials);
```

En el segundo fragmento de código, se pueden observar las diferentes funciones implementadas en nuestro código, las cuáles serán imprescindibles para determinar si es fin de semana, festivo y determinar el tramo horario en el que nos encontramos y el siguiente al actual.

```
//Función para saber si es fin de semana (calendario inglés)
function isWeekend(date) {
   const dayOfWeek = date.getDay();
   return (dayOfWeek === 6) || (dayOfWeek === 0);
}
//Función para saber si es festivo
function isPublicHoliday(date) {
   return !!holidays[date.getMonth()+1 + ',' + date.getDate()]; // !! convierte a booleano y si hay texto
devuelve true
 }
//Función para saber si es fin de semana o festivo
function isWeekendOrPublicHoliday(date) {
   return isPublicHoliday(date) || isWeekend(date);
}
//Función para saber en el tramo horario que estamos ahora
function obtenerTramoActual(tramos, date){
   var hora_actual = date.getHours();
   // Calcula el siguiente tramo teniendo en cuenta la hora actual
   for (var i = 0; i < tramos.length; i++) {
      if (hora_actual < tramos[i][0]) {
         return tramos[i-1];
      }
   }
   // Si no se encuentra un tramo es porque estamos en el último del día
   return tramos[tramos.length-1];
}
//Función para saber el siguiente tramo horario
function obtenerSiguienteTramo(tramos, date){
   var hora_actual = date.getHours();
   // Calcula el siguiente tramo teniendo en cuenta la hora actual
   for (var i = 0; i < tramos.length; i++) {
      if (hora_actual < tramos[i][0]) {
         return tramos[i];
      }
   }
   // Si no se encuentra un tramo, es que es el último del día. Hay que avisar con un null.
   return null;
}
//Función para obtener la fecha donde comienza el siguiente tramo
function obtenerFechaSigTramo(tramos, date){
   var siguiente_tramo = obtenerSiguienteTramo(tramos, date);
   if (siguiente_tramo === null) {
      // Si no se encuentra un tramo, es que es el último del día
      var previous_date = new Date(date.getFullYear(), date.getMonth(), date.getDate(), 0, 0, 0);
      // Añadimos 1 día
      return new Date(previous_date.getTime() + 24 * 60 * 60 * 1000); // en milisegundos
   }
```

```
    return new Date(date.getFullYear(), date.getMonth(), date.getDate(), siguiente_tramo[0], 0, 0);
}
//Función para rellenar las fechas y horas con un 0 delante para adecuarse al formato dd/mm/aaaa o
hh:mm:ss
function rellena(text){
    return text.toString().padStart(2, "0")
}
//Función para redondear resultados
function redondea(num, decimales){
    return Math.round((num + Number.EPSILON) * Math.pow(10, decimales)) / Math.pow(10,
decimales);
}
//Función para crear hora local
function createLocalDate() {
    return new Date(new Date().toLocaleString("en-US", {timeZone: "Europe/Madrid"}));
}
```

En tercer lugar, tiene lugar el cálculo de la fecha y hora de inicio y fin del consumo que estamos midiendo. Para ello, justo al iniciarse la ejecución del código se guarda la fecha y hora de inicio. Seguidamente, tiene lugar un bucle *while* que espera hasta que el estado del electrodoméstico cambie a "Ready", indicando que ha finalizado su funcionamiento. Es en este instante cuando se guardarán la fecha y hora de fin del consumo actual.

```
// Obtener fecha de inicio
var start_date = createLocalDate();
var hora_inicio =
rellena(start_date.getHours())+":"+rellena(start_date.getMinutes())+":"+rellena(start_date.getSeconds());
var fecha_inicio = start_date.getFullYear()+"-"+rellena((start_date.getMonth()+1))+"-
"+rellena(start_date.getDate());

console.log("Cafetera", "Fecha de inicio: " + fecha_inicio);
console.log("Cafetera", "Hora de inicio: " + hora_inicio);

var item = items.getItem("Cafetera_Operation_State");
console.log("Estado de operación de la cafetera: ", item.state);

// Esperar hasta que el programa finalice (pase de "Run" a "Ready")

var i = 0;
while (item.state.toString() != "Ready") {
    java.lang.Thread.sleep(1000); // Esperar 1 segundo antes de volver a verificar
    i++;
    console.log("Cafetera", "Iteración: " + i, "Estado de operación: " + item.state);
}

// Obtener fecha de finalización
var end_date = createLocalDate();
var hora_fin =
rellena(end_date.getHours())+":"+rellena(end_date.getMinutes())+":"+rellena(end_date.getSeconds());
var fecha_fin = end_date.getFullYear()+"-"+rellena((end_date.getMonth()+1))+"-
"+rellena(end_date.getDate());
```

```
console.log("Cafetera", "Fecha de fin: " + fecha_fin);
console.log("Cafetera", "Hora de fin: " + hora_fin);
```

A continuación, se encuentra el corazón de nuestro código, donde se realiza el cálculo del tiempo de uso y consumo en los distintos tramos horarios. Por un lado, almacenamos el tiempo de uso total en tres variables correspondientes a cada tramo *(t_uso_punta, t_uso_llano, t_uso_valle*). Estas variables se utilizarán para calcular el consumo total del electrodoméstico en cada periodo horario. Por otro lado, guardamos en otras tres variables el consumo parcial en cada uno de los tramos (*consumoPuntaParcial, consumoLlanoParcial, consumoValleParcia*l). Este valor representa el consumo generado en el tramo actual en evaluación (tiempo de uso del tramo actual, *diff*, multiplicado por la potencia de la cafetera, *potencia_cafetera*). Este cálculo parcial resulta esencial para el almacenamiento en la base de datos del consumo por tramos, permitiendo la diferenciación de consumos en caso de cambio de día. Posteriormente, se realiza una petición **POST** al endopoint **/consumoTramo** del servidor de consumo para registrar este consumo parcial en la tabla *consumo_tramo* de la base de datos, asegurando un registro preciso de los consumos en cada periodo.

```
// Calcular tiempo de uso
var t_uso_punta = 0;
var t_uso_valle = 0;
var t_uso_llano = 0;

var consumoPuntaParcial = 0;
var consumoValleParcial = 0;
var consumoLlanoParcial = 0;


var calc_start_date = start_date;
var calc_end_date;
var tramo_actual;
while (calc_start_date < end_date) {
   // Calcular qué hora está más cerca, si end_date o el siguiente cambio de tarifa
   if (isWeekendOrPublicHoliday(calc_start_date)) {
      tramo_actual = obtenerTramoActual(TRAMO_NL, calc_start_date);
      console.log("Cafetera", "Tramo actual: " + tramo_actual);
      var fecha_sig_tramo = obtenerFechaSigTramo(TRAMO_NL, calc_start_date);
      console.log("Cafetera", "Fecha siguiente tramo: " + fecha_sig_tramo);

   } else {
      tramo_actual = obtenerTramoActual(TRAMO_L, calc_start_date);
      console.log("Cafetera", "Tramo actual: " + tramo_actual);
      var fecha_sig_tramo = obtenerFechaSigTramo(TRAMO_L, calc_start_date);
      console.log("Cafetera", "Fecha siguiente tramo: " + fecha_sig_tramo);
   }

   console.log("Cafetera", "Calc_start_date: " + calc_start_date);
   var fecha_tramo_actual = calc_start_date.getFullYear()+"-
"+rellena((calc_start_date.getMonth()+1))+"-"+rellena(calc_start_date.getDate());
   console.log("Cafetera", "Fecha tramo actual: " + fecha_tramo_actual);

   calc_end_date = fecha_sig_tramo < end_date ? fecha_sig_tramo : end_date;
```

```
    console.log("Cafetera", "Calc_end_date: " + calc_end_date);
    var diff = calc_end_date.getTime() - calc_start_date.getTime(); // en milisegundos
    console.log("Cafetera", "diff: " + diff);

    //Reseteamos a 0 por los posibles cambios del anterior tramo
    consumoPuntaParcial = 0;
    consumoValleParcial = 0;
    consumoLlanoParcial = 0;

    switch (tramo_actual[1]) {
        case 'P':
            t_uso_punta += diff / 1000; //en seg
            console.log("Cafetera", "Tiempo de uso horario punta: " + t_uso_punta + " seg.");
            consumoPuntaParcial = redondea(((diff/1000) / 60000) * potencia_cafetera, 3); // en kW*min
            console.log("Cafetera", "Consumo punta parcial: " + consumoPuntaParcial.toString() + "
kW*min.");
            break;
        case 'V':
            t_uso_valle += diff / 1000; //en seg
            console.log("Cafetera", "Tiempo de uso horario valle: " + t_uso_valle + " seg.");
            consumoValleParcial = redondea(((diff/1000) / 60000) * potencia_cafetera, 3); // en kW*min
            console.log("Cafetera", "Consumo valle parcial: " + consumoValleParcial.toString() + "
kW*min.");
            break;
        case 'L':
            t_uso_llano += diff / 1000; //en seg
            console.log("Cafetera", "Tiempo de uso horario llano: " + t_uso_llano + " seg.");
            consumoLlanoParcial = redondea(((diff/1000) / 60000) * potencia_cafetera, 3); // en kW*min
            console.log("Cafetera", "Consumo llano parcial: " + consumoLlanoParcial.toString() + "
kW*min.");
            break;
    }

    //Hacer el POST a la dirección del servicio REST (Insertamos el consumo del tramo actual en la tabla
consumo_tramo de la bbdd)
    var url = "http://localhost:8000/consumoTramo";
    var json = JSON.stringify({
    electrodomestico: "Cafetera",
    fecha: fecha_tramo_actual,
    consumo_punta: consumoPuntaParcial,
    consumo_llano: consumoLlanoParcial,
    consumo_valle: consumoValleParcial
    });

    var request = http.sendHttpPostRequest(url, http.CONTENT_TYPE_JSON, json, headers, 5000);
    console.log("Cafetera", "Respuesta a la petición de consumo diario: " + request);


    calc_start_date = calc_end_date;
}
```

Finalmente, llegamos al cálculo de los consumos totales en cada tramo, medidos en kW*min. Esta elección de unidad poco convencional obedece a la naturaleza del simulador, que proporciona intervalos de tiempo extremadamente cortos. Por ejemplo, lo que en la vida real tomaría 2 horas, aquí se reduce a 1 minuto de simulación.

Para calcular el consumo, multiplicamos el tiempo acumulado de uso de cada tramo (expresado en segundos y convertido a minutos) por la potencia de la cafetera (dada en W y convertida a kW). Aunque esta estimación no es completamente fiel a la realidad, especialmente en electrodomésticos con programas largos, como lavavajillas o lavadoras, donde el consumo varía durante el ciclo, nuestro simulador carece de un vatímetro para medir el consumo instantáneo y proporcionar un valor preciso. Por tanto, optamos por calcular el consumo como:

$$consumo = tiempo_{uso} \times potencia_{electrodoméstico}$$

A continuación, al igual que con el registro del consumo parcial en la base de datos del caso anterior, realizamos un **POST** al servidor de consumo en el endpoint **/consumo** para almacenar el consumo completo en la tabla *consumo* de la base de datos. Aunque esta metodología no es idéntica a una medición precisa, nos brinda una estimación adecuada dentro de las limitaciones del simulador.

```
// Calcular consumo y redondear a 3 decimales
var consumoPunta = redondea((t_uso_punta / 60000) * potencia_cafetera, 3); // en kW*min
var consumoValle = redondea((t_uso_valle / 60000) * potencia_cafetera, 3); // en kW*min
var consumoLlano = redondea((t_uso_llano / 60000) * potencia_cafetera, 3); // en kW*min

console.log("Cafetera", "Consumo en punta: " + consumoPunta.toString() + " kW*min");
console.log("Cafetera", "Consumo en valle: " + consumoValle.toString() + " kW*min");
console.log("Cafetera", "Consumo en llano: " + consumoLlano.toString() + " kW*min");

// Hacer el POST a la dirección del servicio REST (insertamos consumo completo a la tabla consumo de la bbdd)
var url2 = "http://localhost:8000/consumo";
var json2 = JSON.stringify({
    electrodomestico: "Cafetera",
    fecha_inicio: fecha_inicio,
    hora_inicio: hora_inicio,
    fecha_fin: fecha_fin,
    hora_fin: hora_fin,
    consumo_punta: consumoPunta,
    consumo_llano: consumoLlano,
    consumo_valle: consumoValle
});
var request = http.sendHttpPostRequest(url2, http.CONTENT_TYPE_JSON, json2, headers, 5000);
console.log("Cafetera", "Respuesta del servidor: " + request);
```

Para depurar este código se ha hecho uso de la consola de OpenHAB, la cual en Ubuntu se ejecuta con el siguiente comando:

```
openhab-cli console
```

Al introducir este comando, nos pedirá una contraseña la cual por defecto es *habopen*. Seguidamente, para poder visualizar los logs que se vayan produciendo en tiempo real, bastaría con ejecutar en la consola de OpenHAB este comando:

```
log:tail
```

5 CONCLUSIONES Y LÍNEAS FUTURAS

Para concluir esta memoria, es crucial destacar los logros y avances que este proyecto ha alcanzado en el desarrollo de una aplicación Android, la creación de un servicio de consumo y la integración de tecnologías importantes en el sector IoT. Esta sección final no solo recopila las conclusiones personales y técnicas derivadas de este proceso, sino que también delineará las direcciones futuras que podrían llevar este proyecto a niveles aún más altos de innovación y eficacia.

5.1 Conclusiones personales

Mirando retrospectivamente, puedo observar que los propósitos que me propuse al comienzo de este proyecto han sido exitosamente cumplidos. Los resultados obtenidos no solo me llenan de satisfacción personal, sino que también refuerzan mis habilidades en el ámbito tecnológico.

A lo largo de este viaje, me encontré con desafíos y momentos de incertidumbre. Sin embargo, al contemplar el resultado final, aprecio profundamente cuánto he evolucionado durante este periodo. Las competencias que he adquirido, la confianza que he desarrollado y las oportunidades que ahora se vislumbran para mi crecimiento continuo son testimonio de este proceso. Aunque no contaba con experiencia previa en la creación de proyectos propios, he utilizado los fundamentos proporcionados por mi formación en Ingeniería y he seguido los consejos de mi tutora para alcanzar los objetivos propuestos.

Este proyecto no solamente marca una etapa en mi recorrido académico, sino que también confirma que la dedicación y la perseverancia pueden traducirse en logros tangibles y valiosos. Estoy emocionada por las lecciones aprendidas, la confianza que he ganado y las perspectivas emocionantes que esta experiencia ha abierto para mi futuro.

5.2 Conclusiones técnicas

Este proyecto ha culminado en la creación de una aplicación Android llamada "IntelliHome", que permite controlar los electrodomésticos Home Connect a través de la integración de OpenHAB. Además, se ha implementado un servicio REST que estima el consumo energético de los dispositivos conectados. La combinación de tecnologías Android, OpenHAB, Home Connect y Spring Boot, entre otras, ha resultado en una aplicación versátil y funcional, que permite a los usuarios controlar sus electrodomésticos y estimar su consumo de manera conveniente.

La integración con electrodomésticos Home Connect ha demostrado ser efectiva para brindar a los usuarios un control remoto y automatizado de sus dispositivos. La conexión con OpenHAB y la estimación del consumo basada en tramos horarios proporcionan un valor añadido al ayudar a los usuarios a gestionar su consumo de manera más eficiente.

Además, este proyecto ha reforzado mi habilidad para planificar y documentar un proyecto de manera estructurada. La creación de una memoria técnica completa, que incluye desde la arquitectura de la aplicación hasta la descripción de cada componente, me ha proporcionado una visión más profunda de la importancia de la planificación y la documentación en el desarrollo de software.

En general, este Trabajo Fin de Grado ha representado un aprendizaje significativo en el desarrollo de aplicaciones móviles y la implementación de servicios REST para la gestión energética. Ha permitido aplicar conceptos teóricos en un entorno práctico y ha proporcionado una base sólida para futuras exploraciones en el campo de la eficiencia energética y la automatización residencial.

5.3 Líneas futuras

El presente apartado destaca la mirada hacia el futuro de este proyecto, delineando posibles direcciones en las que podría evolucionar. Al analizar y anticipar nuevas oportunidades y desafíos, se exploran las áreas en las que se podría expandir o mejorar la aplicación, así como la adopción de nuevas tecnologías y enfoques para seguir enriqueciendo su funcionalidad y alcance.

5.3.1 Diseño

En cuanto a las mejoras del diseño, podemos destacar las siguientes:

- Creación e implementación de una pantalla Home que oriente al usuario hacia las principales funcionalidades de la aplicación móvil.
- Creación e implementación de una pantalla genérica Electrodomésticos que permita listar todos los *Things* de nuestro servidor OpenHAB.
- Creación e implementación de una barra de navegación inferior para que la experiencia del usuario sea más gratificante. Con esta barra el usuario podrá navegar entre las pantallas Home, Habitaciones y Electrodomésticos muy cómodamente (ver figura 5-1).

Figura 5-1. Barra de navegación

5.3.2 Funcionalidades

Dentro de las funcionalidades adicionales que podrían ser implementadas se encuentran:

- Añadir nuevas reglas de OpenHAB para poder programar el arranque de electrodomésticos según le convenga al usuario.
- Implementar notificaciones mediante la tecnología *Firebase Cloud Messaging* en conjunto con las reglas de OpenHAB, para así poder recibir notificaciones de finalización de programas o del estado de algún electrodoméstico incluso cuando la aplicación esté cerrada y el terminal bloqueado.
- Modificar el servicio REST de consumo para que permita distinguir el consumo por usuario.
- Añadir más electrodomésticos al servidor de OpenHAB aprovechando la amplia variedad de *bindings* que proporciona esta tecnología.

5.3.3 Seguridad

En el campo de la seguridad, se podría mejorar lo siguiente:

- Implementación de HTTPS, para cifrar las comunicaciones entre la aplicación y servidores, garantizando la confidencialidad y autenticidad de los datos. Requiere la adquisición de un certificado SSL/TLS, actualización del servidor y ajustes en la aplicación para mejorar la seguridad.

5.3.4 Infraestructura y despliegue

Para concluir, en infraestructura y despliegue algunos aspectos a mejorar podrían ser:

- Configuración de NAT para poder realizar peticiones a los servidores de OpenHAB y consumo desde redes externas.
- Alojar servidores de OpenHAB y consumo en una Raspberry Pi para garantizar su disponibilidad continua.
- Explorar la opción de incorporar vatímetros en los enchufes donde se conectan los electrodomésticos, permitiendo obtener mediciones de consumo precisas. Además, se deberá adaptar el código de las reglas de OpenHAB para almacenar la potencia en cada instante medida por estos vatímetros.

ANEXO A: INSTALACIÓN Y CONFIGURACIÓN DEL SERVIDOR DE OPENHAB EN UBUNTU

A) Instalación del servidor de OpenHAB

En este apartado, exploraremos en detalle el proceso de instalación de OpenHAB, un pilar central en la creación de un entorno domótico cohesivo y funcional. Si bien OpenHAB se puede instalar en varias plataformas (Windows, Linux, Docker, macOS, Raspberry Pi, etc) en este caso enfocaremos nuestra atención en la instalación en Linux, específicamente en el sistema operativo Ubuntu.

Antes de comenzar a instalar el servidor de OpenHAB, debemos asegurarnos de que nuestro dispositivo dispone de la versión Java 17 de la JVM. Para ello basta con poner en el terminal el comando:

```
java -version
```

Si no disponemos de la versión Java 17, podemos instalarla fácilmente siguiendo la documentación de OpenHAB. [22]

Hay que destacar que existen diferentes formas de instalar el servidor de OpenHAB en nuestro dispositivo. Nosotros seguiremos los pasos indicados en la documentación de OpenHAB [23] para los sistemas basados en APT como es el caso de Ubuntu.

1. Primero, añadimos la clave del repositorio de OpenHAB al gestor de paquetes (nota: /usr/share/keyrings podría ya existir):

```
curl -fsSL "https://openhab.jfrog.io/artifactory/api/gpg/key/public" | gpg --
dearmor > openhab.gpg
sudo mkdir /usr/share/keyrings
sudo mv openhab.gpg /usr/share/keyrings
sudo chmod u=rw,g=r,o=r /usr/share/keyrings/openhab.gpg
```

2. Después, podemos elegir entre la versión estable, de prueba o *snapshot*. En nuestro caso, elegiremos la versión estable:

```
echo 'deb [signed-by=/usr/share/keyrings/openhab.gpg]
https://openhab.jfrog.io/artifactory/openhab-linuxpkg stable main' | sudo tee
/etc/apt/sources.list.d/openhab.list
```

3. A continuación, actualizamos el índice de paquetes:

```
sudo apt-get update
```

4. Ahora, instalamos OpenHAB con el siguiente comando:

```
sudo apt-get install openhab
```

5. Por último, instalaremos el paquete de complementos:

```
sudo apt-get install openhab-addons
```

B) Arranque y parada del servidor de OpenHAB

Este apartado explorará los pasos necesarios para configurar y poner en marcha el servidor de OpenHAB en una distribución Ubuntu.

1. Para **arrancar el servidor**, si toda la instalación ha resultado exitosa basta con introducir en el terminal el siguiente comando:

```
sudo systemctl start openhab.service
```

2. Para **consultar el estado** del servicio, introducimos en la consola:

```
sudo systemctl status openhab.service
```

3. Si la puesta en marcha ha sido exitosa, tras poner el comando del paso 2 deberíamos recibir la respuesta que podemos observar en la figura Anexo 1.

```
● openhab.service - openHAB - empowering the smart home
     Loaded: loaded (/lib/systemd/system/openhab.service; disabled; vendor preset: enabled)
     Active: active (running) since Fri 2023-08-11 13:32:07 CEST; 24ms ago
       Docs: https://www.openhab.org/docs/
             https://community.openhab.org
   Main PID: 9518 (karaf)
      Tasks: 2 (limit: 18784)
     Memory: 724.0K
     CGroup: /system.slice/openhab.service
             ├─9518 /bin/sh /usr/share/openhab/runtime/bin/karaf daemon
             ├─9538 /bin/sh /usr/share/openhab/runtime/bin/karaf daemon
             ├─9539 /bin/sh /usr/share/openhab/runtime/bin/karaf daemon
             ├─9540 /bin/sh /usr/share/openhab/runtime/bin/karaf daemon
             └─9544

ago 11 13:32:07 bellozjur systemd[1]: Started openHAB - empowering the smart home.
```

Figura Anexo 1. Arranque exitoso servidor OpenHAB

De esta manera, ya tendríamos al servidor escuchando peticiones en el puerto TCP 8080 y la IP localhost (127.0.0.1).

4. Si deseamos poner por defecto que el servidor se **arranque automáticamente** cada vez que se inicie el dispositivo:

```
sudo systemctl daemon-reload
sudo systemctl enable openhab.service
```

5. Para **parar el servidor**, basta con introducir en el terminal:

```
sudo systemctl stop openhab.service
```

ANEXO B: INSTALACIÓN Y DESPLIEGUE DEL SERVIDOR REST DE CONSUMO EN UBUNTU

A) Instalación del servidor REST de Consumo

Antes de proceder con la instalación del servidor REST de Consumo, asegúrese de tener instalado Maven en su sistema. Puede descargarlo con el siguiente comando:

```
sudo apt install maven
```

Para comprobar si se ha instalado correctamente inserte en el terminal:

```
mvn -version
```

Una vez instalado Maven, siga estos pasos:

1. Descargue el archivo "serv-consumo-tfg.zip" del repositorio de GitHub indicado en el apartado 4.2 del presente documento y descomprímalo en una ubicación deseada en su sistema.
2. Abra una terminal y navegue hasta el directorio donde descomprimió el archivo utilizando el comando:

```
cd ruta/del/directorio
```

B) Inicialización de la base de datos

Desde el directorio del servidor, siga los siguientes pasos:

1. Navega al directorio “serv-consumo-tfg/scripts”:

```
cd scripts
```

2. Ejecuta el script dbsetup.sh, el cual comprueba si el gestor de bases de datos está instalado. Seguidamente, lo inicia y, por último, crea un usuario administrador y su base de datos.

```
./dbsetup.sh
```

Nota: Es importante que el script tenga los permisos adecuados de ejecución. Si obtienes un error de permisos, puedes otorgar los permisos de ejecución al script usando el siguiente comando:

```
chmod +x dbsetup.sh
```

C) Despliegue del servidor REST de Consumo

Una vez finalizada la instalación del servidor e inicialización de la base de datos, tiene lugar el despliegue del servidor. Para ello, basta con introducir en el terminal el siguiente comando desde el directorio del servidor:

```
mvn spring-boot:run
```

REFERENCIAS

[1] José de Lózar Alameda, 2020, «Aplicación domótica en Android con OpenHAB para el control de los dispositivos del hogar».[En línea]. Disponible en: https://idus.us.es/handle/11441/102851

[2] Android Studio, «Cómo configurar las redes de Android Emulator». [En línea]. Disponible en: https://developer.android.com/studio/run/emulator-networking?hl=es-419

[3] Home Connect, «System Architecture». [En línea]. Disponible en: https://developer.home-connect.com/how_it_works

[4] Dell, «Configuración y especificaciones». [En línea]. Disponible en: https://www.dell.com/support/manuals/es-es/inspiron-15-5510-laptop/inspiron-5510-setup-and-specifications/especificaciones-de-inspiron-15-5510?guid=guid-7c9f07ce-626e-44ca-be3a-a1fb036413f9&lang=es-mx

[5] Android Studio, «Plataforma». [En línea]. Disponible en: https://developer.android.com/about

[6] OpenHAB, «OpenHAB». [En línea]. Disponible en: https://www.openhab.org/

[7] OpenHAB, «Home Connect Binding». [En línea]. Disponible en: https://www.openhab.org/addons/bindings/homeconnect/

[8] Home Connect Developer Program, «Simulators». [En línea]. Disponible en: https://developer.home-connect.com/simulator (requiere registrarse en el portal de Home Connect Developer para poder acceder a los simuladores)

[9] Spring, «Spring | Home». [En línea]. Disponible en: https://spring.io/

[10] Maven, «Welcome to Apache Maven». [En línea]. Disponible en: https://maven.apache.org/

[11] PostgreSQL, «PostgreSQL». [En línea]. Disponible en: https://www.postgresql.org/

[12] Figma, «Figma». [En línea]. Disponible en: https://www.figma.com/

[13] RESTClient, «RESTClient, a debugger for RESTful web services». [En línea]. Disponible en: https://addons.mozilla.org/es/firefox/addon/restclient/

[14] Visual Studio Code, «Visual Studio Code». [En línea]. Disponible en: https://code.visualstudio.com/

[15] GitHub, «GitHub». [En línea]. Disponible en: https://github.com/

[16] Wireshark, «Wireshark». [En línea]. Disponible en: https://www.wireshark.org/

[17] Spring Security, «Spring Security». [En línea]. Disponible en: https://docs.spring.io/spring-security/reference/index.html

[18] Home Connect Developer Program, «Sign Up». [En línea]. Disponible en: https://developer.home-connect.com/

[19] Home Connect Developer Program, «Applications». [En línea]. Disponible en: https://developer.home-connect.com/applications

[20] Home Connect Developer Program, «API Docs». [En línea]. Disponible en: https://api-docs.home-connect.com

[21] Home Connect Developer Program, «Quick Start». [En línea]. Disponible en: https://api-docs.home-connect.com/quickstart/#the-home-connect-api

[22] OpenHAB, «Installation Overview». [En línea]. Disponible en: https://www.openhab.org/docs/installation/#prerequisites

[23] OpenHAB, «OpenHAB on Linux». [En línea]. Disponible en: https://www.openhab.org/docs/installation/linux.html

[24] OpenHAB, «Home Connect Binding». [En línea]. Disponible en: https://www.openhab.org/addons/bindings/homeconnect

[25] Endesa, «¿Cuáles son los horarios de la luz?». [En línea]. Disponible en: https://www.endesa.com/es/blog/blog-de-endesa/horarios-luz-valle-punta-llano