Universidad Autónoma de Madrid
Escuela politécnica superior

Grado en Ingeniería Informática

Trabajo Fin de Grado

1718_072_IC DISEÑO E IMPLEMENTACIÓN DE UN HUB DE CONTROL DOMÓTICO

Autor: Pallarés Jiménez, Ignacio
Tutor: Delgado Mohatar, Óscar
Ponente: Anguiano Rey, Eloy

JULIO 2018

1718_072_IC DISEÑO E IMPLEMENTACIÓN DE UN HUB DE CONTROL DOMÓTICO

Autor: Pallarés Jiménez, Ignacio
Tutor: Delgado Mohatar, Óscar
Ponente: Anguiano Rey, Eloy

Grupo de la EPS (opcional)
Dpto. de XXXXX
Escuela Politécnica Superior
Universidad Autónoma de Madrid
JULIO 2018

Resumen

Resumen

La domótica consiste en la automatización del hogar. Los sistemas domóticos, son aquellos capaces de domotizar una vivienda, y proporcionan servicios de comunicación, seguridad, eficiencia energética...etc. Sin duda alguna, la comunicación entre estos sistemas es algo esencial, existiendo redes cableadas e inalámbricas para ello, pudiendo ser controlados estos sistemas desde dentro y fuera del hogar.

BRIMO es un proyecto de código abierto para la gestión y el control de dispositivos domóticos en el hogar. Es una alternativa open source de bajo coste para todas aquellas personas que deseen domotizar su hogar de una manera barata y sencilla. No utiliza protocolos privados, y cualquiera que lo desee puede utilizar y modificar la aplicación a su gusto.

Además, cualquier persona puede crear sus dispositivos (sensores, actuadores o cámaras) de manera sencilla, siempre que éstos sigan los requisitos establecidos. Brimo nos ayuda, gracias a una interfaz web sencilla e intuitiva, a ordenar nuestros dispositivos, visualizar sus estados y mandar comandos a los dispositivos que los acepten.

La aplicación está pensada para ejecutarse en entornos ligeros, concretamente en una Raspberry Pi 3 (precio asqequible), pero también puede ser ejecutada en cualquier ordenador tras una simple configuración. Utiliza una arquitectura REST sobre el protocolo HTTP para comunicarse con los dispositivos, y una arquitectura MVC (Model View Controller) para la interacción con el usuario.

La función principal de Brimo es de "bridge", punto común entre el usuario y los dispositivos, se encarga de poner en contacto al usuario con los dispositivos.

Palabras Clave

Domótica, código abierto, REST, raspberry, HTTP, sensores, MVC, actuadores, bridge.

Abstract

Domotic consists of home automation. Domotic systems are those capable of automating a home, and provide communication services, security, energy efficiency ... etc. Communication between these systems is essential, existing wired and wireless networks for it, that allows us to controll them from inside and outside the home.

BRIMO is an open-source project for the management and control of domotic devices in the home.It is a low-cost open source alternative for all people who want to domotize their home in a cheap and simple way. It does not use private protocols, and anyone can use or modify the application on its preferences.

Besides, anyone is able to create its own devices (sensors, actuators or cameras) in a simple way following the application requirements. Brimo helps us, thanks to a simple and intuitive web interface, to arrange our devices, seeing their status and to send them commands.

The application is designed to be runned on lightweight devices, specifically into a Raspberry Pi 3 (low cost), but it could be also runned on any computer after a simple configuration. It uses REST architecture over HTTP protocol to communicate with devices and a MVC (Model View Controller) architecture for the interaction with users.

The main function of Brimo is to act as bridge, the common point between users and devices: Brimo is the responsible of the communication between them.

Key words

Domotic, open-source, low cost, REST, raspberry, HTTP, sensors, MVC, actuators.

Agradecimientos

Índice general

Índice de Figuras

Índice de Tablas

1 Introducción

1.1. Motivación del proyecto

Los sistemas domóticos por lo general utilizan una arquitectura centralizada: un controlador (bridge) es el encargado de enviar y recibir información de los dispositivos domóticos y las interfaces. Se utilizan sistemas centralizados debido a que abaratan mucho el coste de los dispositivos domóticos, así los dispositivos tienen poca electrónica y programación, y la responsabilidad principal reside en el bridge. Este enfoque tiene sentido cuando se trata de muchos dispositivos en un hogar, que es el caso ideal, si solo tuviésemos un sensor carecería de sentido tener un sensor y un bridge para manejarlo.

El problema principal que existe con los sistemas centralizados se encuentra en la **compatibilidad** entre dispositivos y bridges. Por lo que he observado [1], todavía falta mucha estandarización en el ámbito de la domótica: cada fabricante usa sus medios y protocolos haciendo incompatibles bridges y dispositivos. Además, estos dispositivos no suelen ser muy asequibles. Por lo tanto, nos encontramos ante la necesidad de comprar todos los dispositivos de una misma marca o tener muchos bridges, lo que nos obligaría a manejar cada dispositivo desde su correspondiente bridge.

La domótica puede hacernos la vida en el hogar mucho más sencilla, ayudándonos a ahorrar tiempo y dinero que podremos invertir en otras cosas. Los hogares todavía están muy poco automatizados, y mi principal motivación ha sido acercar la domótica a las personas y aprender acerca de ella. Gracias a nuestro sistema manejamos todos los dispositvos a través de un solo bridge de manera sencilla y eficaz.

1.2. Objetivos y enfoque

El objetivo último de nuestro proyecto es desarrollar un sistema que sea capaz de recibir y enviar información de dispositivos domóticos y sea capaz de interactuar con el cliente.

Los **requisitos** que debe cumplir nuestro sistema son:

- Ligero. Un bridge no debería necesitar demasiada capacidad de procesamiento y de memoria, y es necesario que no sea muy costoso, por lo tanto, la ligereza es requisito indispensable.
- Compatibilidad. Necesitamos que nuestro bridge no sea únicamente compatible con un tipo de sensor, o un modelo de cámara
- Interfaz sencilla y adaptable a cualquier dispositivo. Necesitamos que la interfaz de nuestro bridge sea compatible con cualquier dispositivo sin perder funcinalidad.
- Seguridad. La seguridad en la domótica es algo indispensable, confío en que el día de mañana incluso las cerraduras de nuestras casas serán automáticas, y no podemos dejar la responsabilidad de la seguridad de nuestra a casa a un sistema con vulnerabilidades de seguridad.
- Escalable. Nuestro sistema ha de ser escalable y debemos pensar en todo momento en ampliaciones y trabajos futuros. La domótica evoluciona a pasos agigantados y podríamos añadir funcionalidades a nuestro sistema practicamente a diario. No obstante, es necesario acotar firmemente los límites de nuestro proyecto para ceñirnos a las horas que corresponden a un TFG, aunque debemos tener muy en cuenta en todo momento trabajos futuros y ampliaciones. Además, debemos tener en cuenta la escalabilidad: domótica en un hospital, en una ciudad...etc.

1.3. Metodología y plan de trabajo

Otro ejemplo de imagen:

Figura 1.1: Ejemplo pie de figura 2

2

Estado del arte

2.1. Introducción

3

Diseño del sistema

En este capítulo se describirá el diseño del sistema desarrollado. En la sección 3.1 se detallará la arquitectura del sistema global. En el apartado 3.2 se profundizará en la arquitectura interna del HUB.

3.1. Necesidades del sistema

En esta sección se analizarán las necesidades de nuestro sistema y los dispositivos con los que nos comunicaremos.

3.1.1. Necesidades de los dispositivos

Antes de empezar a diseñar el sistema y elegir el protocolo que se utilizará y el medio físico por el que se comunicarán nuestros dispositivos es necesario analizar los dispositivos que podrán conectarse a nuestro HUB así como sus necesidades. Una vez determinados los requisitos del protocolo se estudiará el medio físico de comunicación.

Los principales dispositivos domóticos que hemos encontrado son: sensores de temperatura, sensores de humedad, sensores de luz, sensores de movimiento, medidores de distancia, sensores de humo, sensores magnéticos, cámaras, bombillas, enchufes, termostatos, motores, aires acondicionados, interruptores y altavoces.
Estos dispositivos pueden ser divididos en dos grupos: sensores y actuadores.
Los sensores solamente envían determinada información a nuestro HUB (comunicación unidireccional), mientras que los actuadores reciben mensajes con determinados comandos a parte de enviar información del estado en el que se encuentran (comunicación bidireccional).

Agrupación de los dispositivos encontrados:

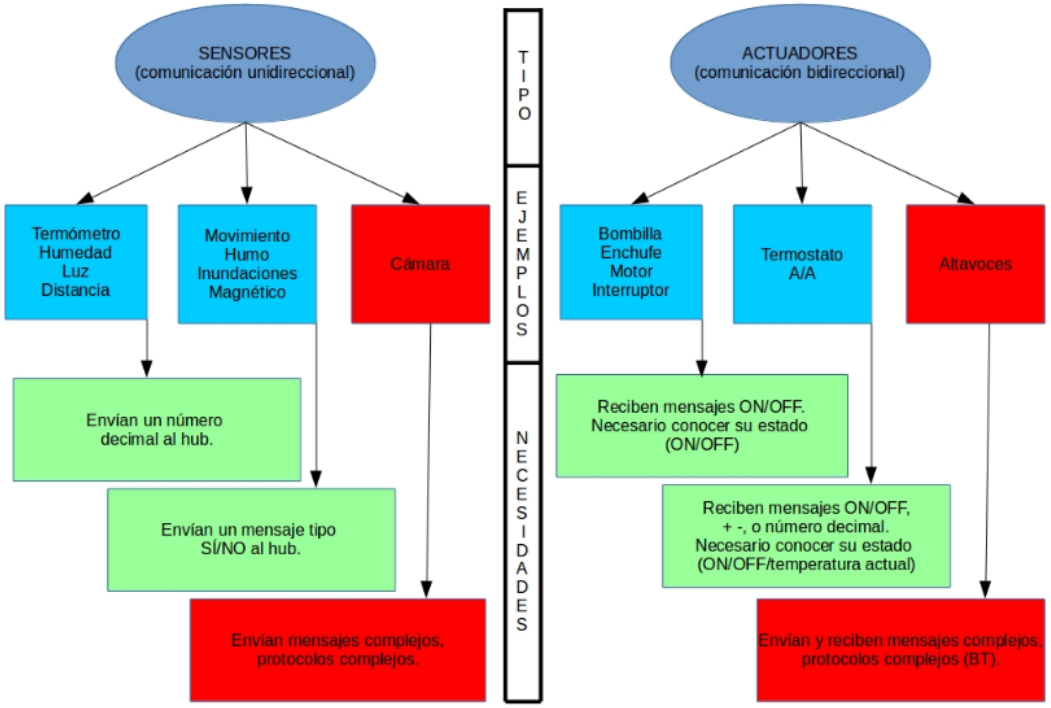

Figura 3.1: Tipos de dispositivos

Para la definición del protocolo dividiremos los dispositivos en tres tipos:

- Tipo 1 (sensores): el hub sólo recibe información de los sensores. El hub no necesita saber qué tipo de información recoge (número decimal, SÍ/NO...etc), simplemente la actualiza y la muestra al usuario.
- Tipo 2 (actuadores): estos dispositivos envían información al HUB y son capaces de recibir comandos tipo ON/OFF, +/-, número decimal...etc.
- Tipo 3: cámaras IP. Este sensor recibirá un tratamiento especial debido a la necesidad de una comunicación constante y rápida.

3.1.2. Necesidades del sistema

Una vez analizadas las necesidades de los dispositivos podemos analizar las necesidades de nuestro sistema. Para el desarrollo de nuestro sistema necesitaremos una arquitectura que nos permita:

- Comunicación bidireccional entre los dispositivos y el hub: es necesario que el hub conozca información de los dispositivos, registre dispositivos y gestione dispositivos, así como también es necesario que los dispositivos puedan recibir comandos provenientes del hub. El hub debe permitir aceptar dispositivos con diferentes comandos, y no ceñirse sólo a un número cerrado de comandos (ON/OFF, +/-,...).
- Comunicación entre el hub y la interfaz: será necesario que la información de los dispositivos y el estado de los mismos sea accesible a través de la interfaz de usuario. Además el usuario

debe ser capaz de gestionar los dispositivos y enviar comandos a los actuadores a través de la interfaz.

- Seguridad en la comunicación: es imprescindible que toda comunicación se realice de manera segura, de tal manera que nadie pueda modificar o acceder a nuestra información.
- Escalabilidad: aunque durante la realización de nuestro proyecto nos centraremos únicamente en la comunicación mediante protocolo HTTPS, es necesario diseñar un sistema escalable que el día de mañana pueda funcionar con diferentes protocolos y dispositivos.

3.2. Protocolos

En esta sección se describirá el protocolo que se utilizará en las comunicaciones entre los dispositivos y el HUB.

3.2.1. Protocolo HTTPS

El protocolo elegido para la comunicación entre dispositivos, hub e interfaz es el protocolo HTTPS (Hypertext Transfer Protocol Secure).
Este protocolo nos da la posibilidad de implementar una API REST consumible por parte de los dispositivos y por parte de la interfaz, sin necesidad de utilizar diferentes protocolos para los diferentes canales.
Además, de estar muy estandarizadas a día de hoy, las APIs REST nos dan la capacidad de separar entre cliente y servidor y de ser capaces de utilizar diferentes lenguajes y tecnologías para cada lado. Es decir, podemos tener un servidor escrito en Express.js (javascript), una interfaz gráfica utilizando Angular5 (TypeScript), y unos actuadores/sensores que utilicen CherryPy (Python).
Además, utilizar HTTPS nos ofrece la posibilidad de crear un canal de comunicación cifrado, de manera que la información que circula en dicho canal no pueda ser descifrada por ningún intermediario ni se pueda sufrir un ataque Man-In-The-Middle*.

3.3. Arquitectura del sistema

En esta sección se describirá el diseño y la arquitectura del sistema de manera global, incluyendo dispositivos actuadores, dispositivos sensores y el propio hub.

3.3.1. Arquitectura del sistema

Teniendo en cuenta las necesidades de nuestro sistema realizaremos una arquitectura similar a las arquitecturas de microservicios, en la que el hub y los actuadores serán los hosts de un servidor REST y serán capaces de recibir y procesar peticiones.
El hub recibirá peticiones de parte de la interfaz de usuario y de los dispositivos, y lanzará peticiones a los actuadores. Para ello se establecerán dos APIS publicadas por el HUB y consumibles por la interfaz de usuario y los dispositivos:

- Interface API: será consumida por la interfaz de usuario. Se encargará de enviar la información de los dispositivos al usuario: número de dispositivos, localización, etc... Además, permitirá al usuario gestionar dispositivos y enviarles comandos.

- Sensors API: será consumida por actuadores y sensores por igual. Permitirá a los dispositivos darse de alta en el sistema y actualizar periódicamente su información.

Los actuadores, además de lanzar peticiones al hub para informar de su estado, deberán ser capaces de recibir peticiones del hub con diferentes comandos. Para ello se establecerá otra API que todos los dispositivos deberán seguir, para que así el HUB consuma la misma API en los diferentes dispositivos. Será denominada en adelante como Actuators API.
Estableceremos y explicaremos estas APIs en los capítulos siguientes.
Todas las peticiones deben ser securizadas, y debemos asegurar que ningún intruso pueda acceder y/o modificar la información de nuestro sistema.

Esquema de la arquitecura a seguir:

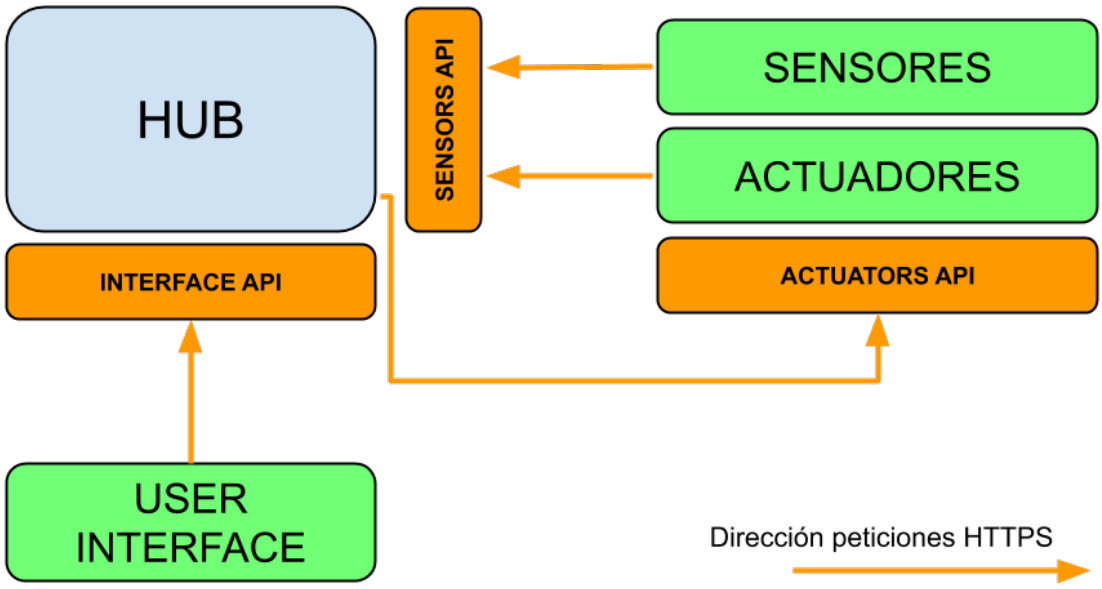

Figura 3.2: Tipos de dispositivos

3.3.2. Modelo de datos

En esta sección se describirán los modelos de datos a utilizar. Todos los datos residirán en el HUB, que será el encargado de orquestarlos, organizarlos y mantenerlos.

3.3.3. APIS

En esta sección se definirán y explicarán las diferentes APIS que se utilizarán en el sistema. Al tratarse de APIS REST, en la definición de cada método hay que informar: ruta del método, verbo (GET, POST, PUT, PATCH, DELETE), cuerpo de la petición (si existiera) y variables de la ruta (si existieran). A no ser que se indique lo contrario, el cuerpo de todas las peticiones debe estar en formato JSON, y debe ser informada la cabecera **Content-Type** con valor **application/json**.
Como se ha descrito anteriormente todas las peticiones deberán ir securizadas, para lo que se utilizarán tokens JWT. Es imprescindible que el token vaya en la cabecera **x-access-token** de cada petición, o de lo contrario la petición será denegada. La generación de tokens y la gestión

de usuarios es descrita por la API **login-api**.
Estas APIS funcionan como contratos entre publicador y consumidor, y es imprescindible que ambas partes consuman y publiquen de la manera acordada para que el sistema completo funcione. El cambio de uno de estos contratos debe ser indicado a todas las partes para que se tenga en cuenta en los desarrollos futuros.
Todos los métodos de estas APIs están enumerados y definidos en un proyecto de Postman desde el cual se pueden probar.

Sensors API

Esta API será consumida tanto por sensores como por actuadores, y les permitirá registrarse en el sistema y actualizar su información.

- **POST /brimo/sensors-api/devices**: se utilizará para el registro de dispositivos. En el cuerpo de la petición se informarán un nombre descriptivo (podrá ser modificado por el usuario más adelante) y frecuencia de actualización de la información [1]. Opcionalmente, en el caso de ser un actuador, el dispositivo informará de los comandos que es capaz de recibir. Estos comandos vendrán en forma de array y deben contener descripción y código de comando. Si el registro es correcto el HUB responderá con un 201 CREATED y un id de dispositivo. Este id será utilizado por el dispositivo más adelante para enviar información al HUB.

```
{
        "name": "new_devices",
        "freq": 12,
        "commands": [
                {
                        "command_descr": "ENCENDER",
                        "command_code": "ON"
                }
        ]
}
```

- **PUT /brimo/sensors-api/devices/{device-id}/info**: se utilizará para actualizar la información del dispositivo. El parámetro device-id indicará el id del dispositivo, proveniente del registro. Si la información se actualiza correctamente el HUB devolverá 200 OK.

```
{
        "info": "23 C"
}
```

Actuators API

Esta API será consumida por el HUB, y permitirá al HUB enviar comandos a los dispositivos.

[1] Si el dispositivo pasa más de los segundos informados sin actualizar información, entonces el HUB lo considera desconectado.

- **POST /brimo/actuators-api/commands?command_code=ON**: es el único método de la API. El HUB enviará esta petición para enviar comandos al dispositivo. El código de comando debe haber sido informado previamente en la fase de registro.

Interface API

Como hemos explicado anteriormente, esta API será consumida por la interfaz de usuario y permitirá al usuario obtener información de los dispositivos, gestionarlos y mandarles comandos:

- **GET /brimo/interface-api/devices**: esta petición nos devolverá información de todos los dispositivos dados de alta en el sistema. Al tratarse de una lista no se poblarán todos los campos del dispositivo, sólo los comunes: id del dispositivo, nombre, frecuencia, fecha de última actualización de la información, id de habitación y descripción de la habitación.

- **GET /brimo/interface-api/devices/{device-id}**: a diferencia de la petición anterior, se obtiene únicamente la información del dispositivo indicado con el parámetro device-id. Esta información es más completa, y además de la información de la petición anterior se obtiene la lista de comandos que acepta el dispositivo y su IP.

- **DELETE /brimo/interface-api/devices/{device-id}**: a través de esta petición el usuario podrá eliminar el dispositivo indicado. A partir de este momento el sistema denegará al dispositivo la comunicación con el mismo.

- **PATCH /brimo/interface-api/devices/{device-id}/?room-id=12&name=sensor-habitacion**: esta petición permitirá editar la habitación en la que se encuentra el dispositivo y su nombre. Ambos parámetros room-id y name son opcionales, aunque al menos uno debe estar presente.

- **DELETE /brimo/interface-api/devices/{device-id}**: a través de esta petición el usuario podrá eliminar el dispositivo indicado. A partir de este momento el sistema denegará al dispositivo la comunicación con el mismo.

- **PATCH /brimo/interface-api/devices/{device-id}/?room-id=12&name=sensor-habitacion**: esta petición permitirá editar la habitación en la que se encuentra el dispositivo y su nombre. Ambos parámetros room-id y name son opcionales, aunque al menos uno debe estar presente.

- **POST /brimo/interface-api/devices/{device-id}/commands?command-code=ON**: se utilizará para enviar comandos al dispositivo informado. La petición irá al HUB, que será el encargado de enviar otra solicitud al dispositivo correspondiente. Para ello, utilizará la IP del dispositivo.

- **GET /brimo/interface-api/devices/rooms**: devolverá la lista actual de habtiaciones registradas. Se devolverán en forma de array y en cada una de ellas vendrán informadas descripción e identificador.

- **POST /brimo/interface-api/devices/rooms**: se utilizará por el usuario para añadir habitaciones. Únicamente es necesario informar el nombre de la nueva habitación. Si el registro de la habitación es correcto, entonces el HUB devolverá 201 CREATED con el id de la nueva habitación.

Login API

Esta API será utilizada para la generación de los JWT, que necesariamente, deben ir informados en las cabeceras de cada petición. Además, gestionará los usuarios con acceso al sistema.

- **POST /brimo/login-api**: se deberán informar los campos usuario y contraseña para la correcta generación del token. En el caso de introducir credenciales inválidas el HUB devolverá 401 Unauthorized. En caso de éxito el HUB devolverá el token generado, que será válido para las siguientes dos horas.
- **POST /brimo/login-api/users**: servirá para añadir nuevos usuarios al sistema. Deberán informarse nombre de usuario y contraseña.
- **PUT /brimo/login-api/users**: servirá para modificar la contraseña y/o el nombre de usuario actuales. Deberán ir informados en el cuerpo de la petición al menos uno de los dos parámetros.

3.4. Arquitectura del hub

3.4.1. Módulos

Enrutador

Servicios

Repositorios

Middleware

4

Desarrollo del sistema

4.1. Hub

4.1.1. Tecnologías

4.1.2. Módulos

4.1.3. Configuración

4.1.4. Seguridad

4.2. Front end

4.2.1. Tecnologías

4.2.2. Componentes

4.2.3. Servicios

5

Experimentos Realizados y Resultados

5.1. Bases de datos y protocolo

5.2. Sistemas de referencia

5.3. Escenarios de pruebas

5.4. Experimentos del sistema completo

6

Conclusiones y trabajo futuro

Glosario de acrónimos

- **Sistema domótico**: Un sistema domótico es el conjunto de controladores y dispositvios que hacen posible la automatización del hogar.
- **Dispositivo domótico**: Dispositivo que nos ayuda a la automatización del hogar actuando o recopilando información. Ejemplos: sensores de temperatura, relés, cámaras...etc.
- **Bridge**: Dispositivo que nos ayuda a controlar y administar diferentes dispositivos domóticos. Los dispositivos se conectan al bridge, y el cliente interactua directamente a través de él.
- **REST**: Arquitectura software que se apoya en el protocolo HTTP. Se utiliza en arquitecturas cliente-servidor. El cliente tiene operaciones básicas y predefinidas: GET, POST, PUT, DELETE... Y el sevidor responde a las peticiones con su correspondiente código HTTP. Cada recurso del servidor es direccionable a través de su URI.
- **JSON**: JavaScript Object Notation: formato de texto sencillo para el intercambio de datos.
- **Angular5**: Framework de código abierto mantenido por Google para la creación y mantenimiento de Single Page Applications (SPA). Desarrollado en TypeScript.
- **SPA**: Del inglés Single Page Application: aplicación web que se ejecuta en una sola página, sin necesidad de refrescar el navegador, haciendo más fluida la navegación.
- **MVC**: Modelo Vista Controlador: arquitectura software que separa los datos (Modelo) de la interfaz de usuario (Vista), su comunicación y lógica se encuentra en el controlador.
- **Responsive**: Diseño web cuyo objetivo es adaptar la apariencia de la página web a diferentes dispositivos.
- **Man-In-The-Middle**: Tipo de ataque en el que el atacante es capaz de interceptar y/o modificar mensajes enviados entre dos partes. Comúnmente este ataque se realiza en redes donde se utilizan protocolos HTTP, el atacante es capaz de captar las peticiones y modificarlas.

Bibliografía

[1] Autor Apellidos. Titulo del artículo. *Revista de publicación*, pages 65–73, 2008.

A

Manual de utilización

B

Manual del programador