Desarrollo de aplicaciones en la nube.
Segundo parcial

Ricardo López Fócil
A01327311

Marzo 2017

1 Introduccion

A lo largo de este escrito se detallará el proyecto que realicé como examen de segundo parcial. El proyecto consiste en la creación de 4 microservicios, los cuales son administrados por Docker.

El sistema consiste en la administración de dispositivos de internet de la cosas. A lo largo de las siguientes secciones se detallan estos puntos.

En los servicios se incluyen un servicio que provee una interfaz web, uno que provee una API pública, uno que provee acceso autenticado y un último que manejará la base de datos en Mongo.

2 Estructura general

La estructura general de la aplicación es de la siguiente manera:

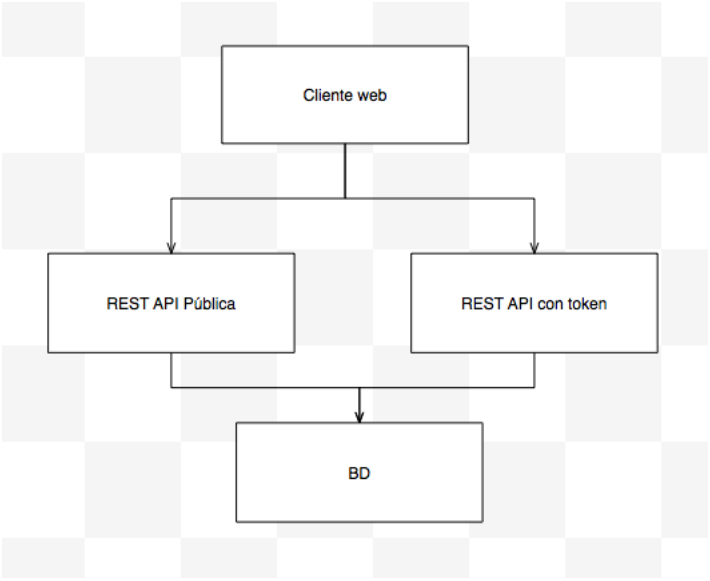

Figure 1: Arquitectura básica

Lo más importante a notar aquí es que la aplicación web nunca accede directamente a la base de datos, si no que usa las API, tanto pública como privada para realizar sus funciones.

3 Página Web

La página web permite a los clientes está dividida en tres partes. Cada una de ellas utiliza llamadas de AJAX GET y POST para obtener los datos de las dos API.

La primera de ellas es el index, donde se puede observar la información general del proyecto. Además, los usuarios pueden ver en esta sección los dispositivos que se encuentran registrados en el sistema. Cada uno de ellos está identificado por el ícono de Linux.

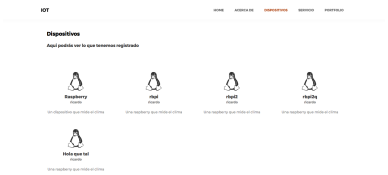

Figure 2: Página principal

La segunda de ellas utiliza la parte pública de la aplicación. En este sitio los usuarios verán los sensores que posee ese dispositivo. Al seleccionar uno de ellos, se cargan los últimos valores de este sensor y se genera estadística básica sobre ese sensor. La interfaz luce de la siguiente manera:

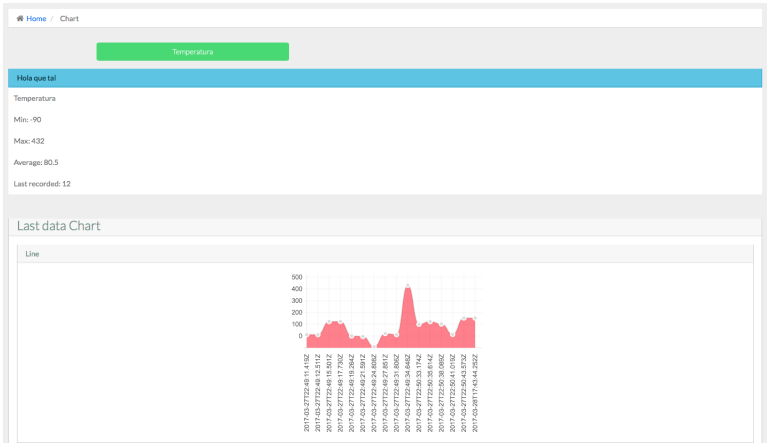

Figure 3: Estadística

La última parte de la interfaz web sirve para usuarios que tengan cuenta en la API. Aquí los usuarios pueden crear nuevos dispositivos así como editar y eliminar sus dispositivos existentes. Cabe aclarar que aunque se provee esta interfaz, el usuario podría usar directamente la API privada. Toda la interfaz descrita anteriormente funciona mediante llamadas POST y GET de JQuery.

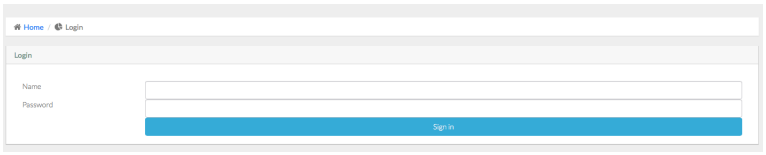

Figure 4: Login

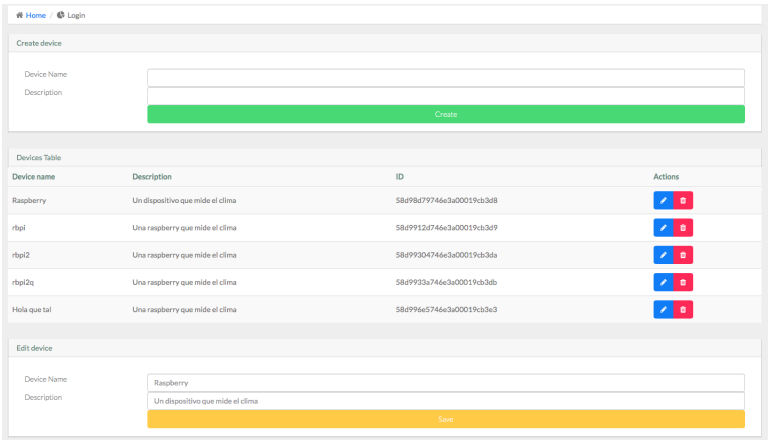

Figure 5: Main UI

4 REST API con token

La API consiste en publicar todo el contenido que se volvera disponible a traves de la API pública. Las acciones que se pueden realizar a traves de esta API son:

- Registrar usuario
- Login que genera un token
- Crear dispositivos
- Editar dispositivos
- Eliminar dispositivos

- Subir información de los dispositivos

Todas las peticiones exepto las primeras dos requieren el token generado por el Login. Este token se genera mediante la encriptación del id del usuario, una fecha de expiración y una llave secreta que nunca se revela al público. La API está en el puerto 8082.

5 REST API pública

La REST API pública contiene los elementos básicos necesarios para que los clientes consulten la información de los dispositivos. En esta API no se ejecutará ninguna acción del DML (no se modificará de ninguna manera la base de datos)

La API Pública permite las siguientes acciones:

- Ver los dispositivos incluyendo su id, nombre, descripción y dueño
- Ver las etiquetas de un dispositivo
- Consultar estadística básica de una etiqueta
- Ver los últimos N registros de una etiqueta

Todas estas acciones son consultadas por el portal web, sin embargo, un usuario puede consultar la API directamente en el puerto 8081.

6 Base de datos

La base de datos utiliza Mongo, con un modelo muy básico de tres colecciones. La primera son los usuarios, en donde unicamente se guarda el usuario y su contraseña. Posteriormente se tiene la colección de dispositivos, quienes tienen una relación con los usuarios. Por último, se tiene la información de los sensores, los cuales tienen un dispositivo, su etiqueta y su valor numérico. Por el momento la aplicación solo funciona con valores numéricos.

7 Montaje en Docker

Cada uno de los servicios usa un archivo de docker muy similar, donde solo varía el número de puerto que exponen. En estos archivos se usa una base con Debian e instala node en ellos. Tras esto se crea y carga el directorio /src y se copian los archivos fuentes en la máquina virtual.

Por último, se generan los paquetes mediante npm, se expone al exterior los puertos necesarios según el contenedor. Por último se pone a correr el servidor.

La base de datos únicamente utiliza la base de mongo y se corre de tal modo que los demas contenedores puedan encontrarla.

La conexión entre contenedores se realiza de manera muy simple mediante la creación de una red interna en Docker y se utiliza el DNS que ofrece docker por defecto.

Todos los contenedores se corren en modo detach y tienen un nombre que permite la interacción.

8 Instalación del sistema en un sistema

Para la instalación del sistema se creó un archivo de bash que ejecuta los comandos necesarios para que el entorno quede totalmente configurado. El script unicamente construye cada uno de los archivos en las carpetas e inicia los contenedores con su nombre. Entre el contenedor de mongo y los node se deja una pausa dado que si se inicia node sin base de datos la aplicación se da por terminada.

9 Conclusion

Este proyecto fue muy ambisioso pero al mismo tiempo gratificante. Durante el proceso aprendí a utilizar node, docker y reafirme mis conocimientos de mongo. También aprendi a utilizar LATEX.

El codigo del proyecto se encuentra en
https://github.com/rikyfocil/Nube-2p

La demostración se puede ver por ahora en la dirección:
http://54.202.85.225/