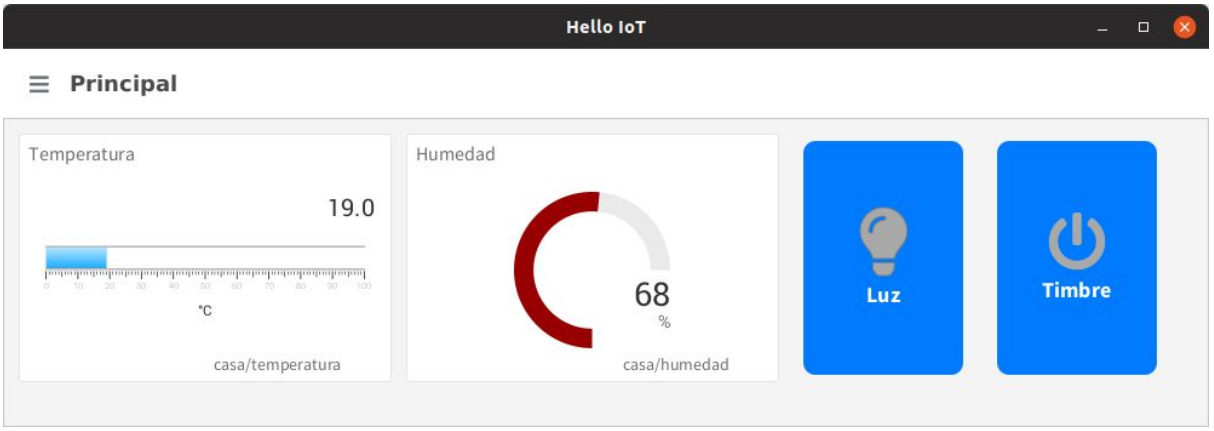
Docker

1. Crear otro contenedor con node red denominado 'mynodered1', este contenedor no debe mapear volúmenes y el puerto de acceso debe ser 2880. Comprobar funcionamiento de ambos y listar archivos de datos de nuevo container. Redactar un documento con los comandos y/o capturas de pantalla de la resolución.
2. Crear otro contenedor con mysql 5.7 denominado 'mysql57nuevo', luego, crear una tabla t1 con la columna c1 de tipo varchar(50) e insertar 3 filas 'a','b' y 'c' en el contenedor 'mysql57', luego, crear una tabla federada t1federada en 'mysql57nuevo', comprobar el funcionamiento. Redactar un documento con los comandos, archivos y/o capturas de pantalla de la resolución.
3. Crear una nueva base de datos en MongoDB llamada 'db1', crear una colección, dentro de db1 llamada 'productos', insertar al menos 5 productos, cada producto debe tener al menos los siguientes atributos: nombre, rubro, precio, enStock, vencimiento. Consulte todos los productos de la colección, busque productos por rango de precios, actualice los precios subiendo un 10% de un rubro en particular, elimine los productos vencidos. Redactar un documento con los comandos, archivos y/o capturas de pantalla de la resolución.
4. Agregar un nuevo usuario en mosquitto debian:debian, habilitar websockets y probar que funcione el nuevo protocolo con el nuevo usuario. *TIPs: observar los puertos que se mapean cuando se crea el contenedor. Con el archivo propuesto en el seminario se pierde el puerto mqtt, para habilitarlo anexar 'port 1883'*, además de la configuración de websockets. Un posible cliente: http://hivemq.com/demos/websocket-client

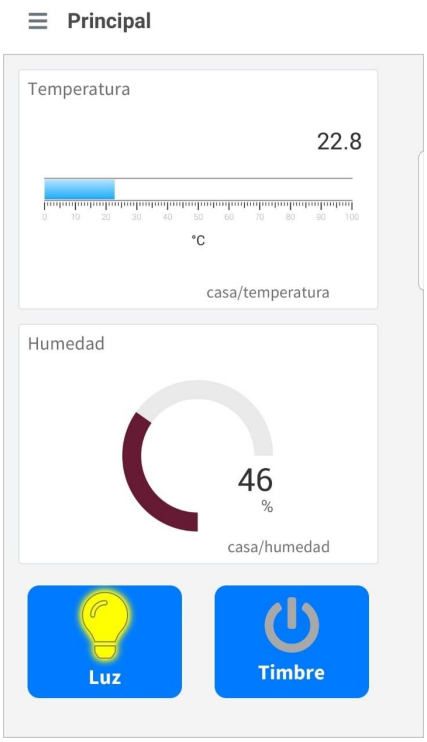
MQTT

1. Investigue la herramienta HelloIoT (https://github.com/adrianromero/helloiot) e implemente un dashboard como el siguiente:

Desktop



Mobile:



Teniendo en cuenta la siguiente estructura de tópicos:

Sensores (los sensores, solo informan cambios de estado):
`casa/th` → publica un json `{ "temperatura": 18.5, "humedad" : 56.8 }`
`casa/temperatura` → publica un número decimal con la temperatura
`casa/humedad` → publica un número decimal con la humedad
`casa/timbre/estado` → publica el estado del timbre representado por un número 1=encendido, 0=apagado
`casa/luz/estado` → publica el estado de la luz representado por un número 1=encendida, 0=apagada
`casa/servo/estado` → publica el estado del servo motor, valores entre 0 y 180

Actuadores:
`casa/timbre` → publicar 1 para encender el timbre (se mantiene encendido por 2 segundos), 0 para apagarlo
`casa/luz` → publicar 1 para encender y 0 para apagar
`casa/servo` → mueve el servomotor, valores válidos 0 a 180 grados

Punto de Oro: Administrar el servo motor

Node-RED

1. Crear un primer flow que contenga el nodo inject, change y dos nodos debug que deben salir de change, se debe inyectar timestamp cada 3 segundos y la primera inyección se debe hacer pasados 1 segundo. El nodo change debe crear el atributo

mensaje en **msg** con el valor "*Hola mundo*". Uno de los nodos debug debe mostrar **payload** y el otro **mensaje**. Debe adjuntar el archivo físico que contiene el flow.

2. En base al ejemplo HttpDeep (download), crear un servicio con la url 'download_v2' que permita descargar cualquier archivo que se encuentre en la carpeta /data enviando como parámetro de query 'filename', el tipo mime se debe calcular automáticamente en base al nombre del archivo.
   Por ejemplo: http://localhost:8080/download_v2?filename=vp.jpg, en este caso:
   Se buscará el archivo **/data/vp.jpg** y el tipo mime calculado será **image/jpeg**
   TIP: investigar el comando file de linux
3. REST: Crear una API REST que administre una lista de productos, cuyos atributos son: nombre, rubro, precio, enStock. Además de las acciones vistas, deberá crear una acción que retorne la lista de productos filtrada por rubro y/o rango de precios. El Storage que debe usar es MongoDB.
4. Dashboard: utilizando los componentes de Dashboard, cree una UI (interface gráfica) cliente de la API del punto anterior. En otras palabras, que se pueda administrar gráficamente el recurso expuesto en API, con opciones de: agregar, listar, modificar, eliminar, etc