Materia: Desarrollo de aplicaciones para IoT

Trabajo Final: Sistema de riego automatizado
Estudiante: Raúl Emilio Romero

Descripción general del proyecto

Problemática
Se necesitará de recopilar la información proporcionada por un sensor que va a medir la humedad de la tierra mediante el uso de un tensiómetro el cual responde a cambios de tensión de humedad en el suelo y su funcionamiento se rige por la fuerza de succión del suelo. Consiste en un medidor de vacío y un tubo sellado con una capa de cerámica porosa.
La capa de cerámica simula movimiento del agua a través del suelo. Mientras más seco se encuentra el suelo, más alta será la lectura del tensiómetro. La interpretación de la lectura e un tensiómetro varía según el cultivo, el tipo de suelo y curva de humedad correlacionada. Sin embargo, se puede tomar de referencia que de 0 a 10 centibares (Cb) el suelo está saturado; de 10 a 30 Cb, el suelo está en CC; y, de 30 a 60 Cb, el suelo está seco y debe regarse de inmediato.

Se pide:
Crear una aplicación en ionic que permita:

- Dar un listado de dispositivos.
- Al entrar a algún dispositivo, brindar el último valor de medición por sensor en el gráfico
- Tener la opción dentro de la vista del dispositivo con sus medición, de poder abrir la electroválvula que le corresponde. (En el caso de que se abra o se cierre dicha electroválvula, se deberá insertar un registro en la tabla de Log_Riegos y por otro lado se necesitará realizar un insert sobre la tabla de mediciones para crear un nuevo registro con el nuevo valor solamente si se cierra la electroválvula).
- Tener otra opción que permita ver todas las mediciones de ese sensor como una tabla.
- Poder consultar el log de los riegos para una electroválvula.

Solución
El sistema de riego automatizado se resuelve a partir de plantear e implementar la siguiente arquitectura:

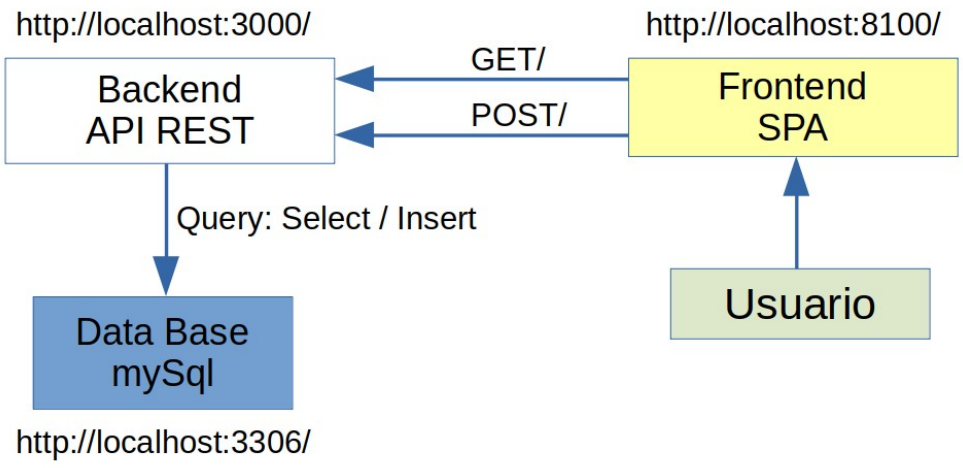


Figura 1: Arquitectura general del sistema

Como se observa en la Figura 1 la arquitectura del sistema consiste en tres subsistemas principales: el Frontend, el Backend y la base de datos.
De esta manera el usuario final del sistema interactúa con el sistema de riego automático por medio de un acceso desde internet al Frontend. El Frontend es el encargado de brindar al usuario toda la interactividad necesaria para comunicarse con el sistema de riego, es decir la interfaz de usuario. El Frontend se diseño e implemento en Ionic (basado en Angular) como una Simple Page Application (SPA), pensada esencialmente para dispositivos móviles.

Cada vez que el usuario solicita datos al sistema como, cantidad de dispositivos, mediciones por dispositivo o log de electroválvulas el Frontend por medio de peticiones http del tipo GET o POST se comunica con el Backend, quien es el encargado de interactuar directamente con el sistema (electroválvulas, sensores, etc.) y consultar el registro de mediciones y logs en la base de datos.

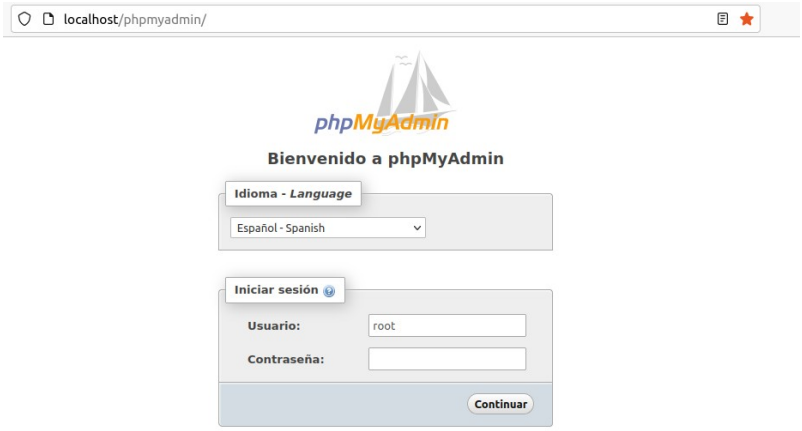
El Backend se diseño en NODE.JS y es el encargado de atender a las peticiones del cliente por medio del Frontend y comunicarse con los datos de la base de datos.

El diseño de la base de datos y su implementación fueron aportados por la cátedra, sin embargo en este caso se implemento utilizando un cliente phpmyadmin al que se accedió por http://localhost/phpmyadmin/. Ingresando con usuario y contraseña se cargó toda estructura de datos y las tablas brindadas por la cátedra.

De esta manera se realizaron 2 proyectos independientes uno para el Backend y otro para el Frontend. Además la base de datos corre de forma independiente también.

Descripción de la base de datos

Se instaló el servidor de mySql en mi máquina local para el proyecto al que se puede acceder por medio de un cliente para la configuración inicial. El ingreso a la base de datos utilizando un cliente se realiza como se mencionó anteriormente por medio de http://localhost/phpmyadmin/:



Cargando usuario y contraseña se observa a modo de resumen la base de datos DAM con sus respectivas columnas. El armado de cada tabla se realizó a partir de los archivos otorgados por la cátedra.

De esta manera con los parámetros siguientes el Backend puede acceder a la base de datos DAM descripta:

```
JS index.js > pool.getConnection() callback
var mysql = require('mysql');
var configMysql = {
    connectionLimit: 10,
    host: 'localhost',
    port: 3306,
    user: 'root',
    password: 'vtwqh7717',
    database: 'DAM'
}
```

Esta configuración se encuentra el el archivo /mysql/index.js del proyecto **backend**

Descripción del Backend

Para hacer funcionar el Backend se debe ingresar a la carpeta del proyecto **backend** y luego en un terminal escribir: node index.js. Luego de esto el backend comenzará a funcionar. Por terminal se visualizan diferentes mensajes, en este caso se informa que la API-REST comenzó a funcionar adecuadamente:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                                  bash

Se envió al cliente la última medición para el dispositivo 1
Se envió al cliente la última medición para el dispositivo todas
Se envió al cliente la última medición para el dispositivo todas
Se envió al cliente el listado de mediciones para el dispositivo 1
Se envía al cliente el listado de logs de electroválvula 1
Se envió el último log para la electroválvula 1
Se envió el listado de Dispositivos solicitado por el cliente
^C
rome@rome-HP-Laptop-15-bs0xx:~/Escritorio/maestriaIot/damp/TP/backend$ node index.js
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                                  node

Se envió al cliente la última medición para el dispositivo todas
Se envió al cliente el listado de mediciones para el dispositivo 1
Se envía al cliente el listado de logs de electroválvula 1
Se envió el último log para la electroválvula 1
Se envió el listado de Dispositivos solicitado por el cliente
^C
rome@rome-HP-Laptop-15-bs0xx:~/Escritorio/maestriaIot/damp/TP/backend$ node index.js
API Funcionando
```

El backend proporciona varios endpont que permiten la comunicación con el Frontend o con cualquier cliente que acceda a estos endpoint por medio de peticiones http GET o POST.

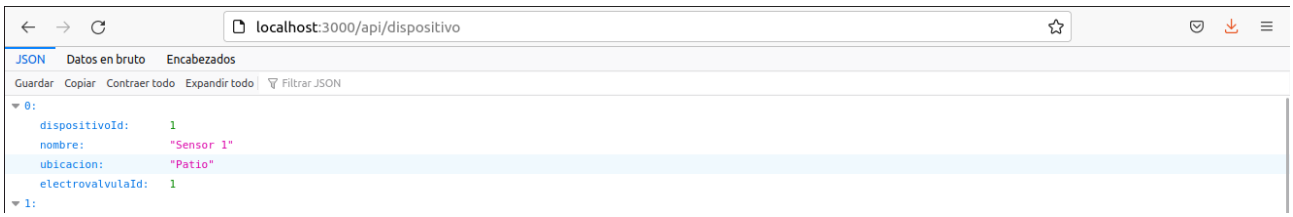
Los endponit son:

GET

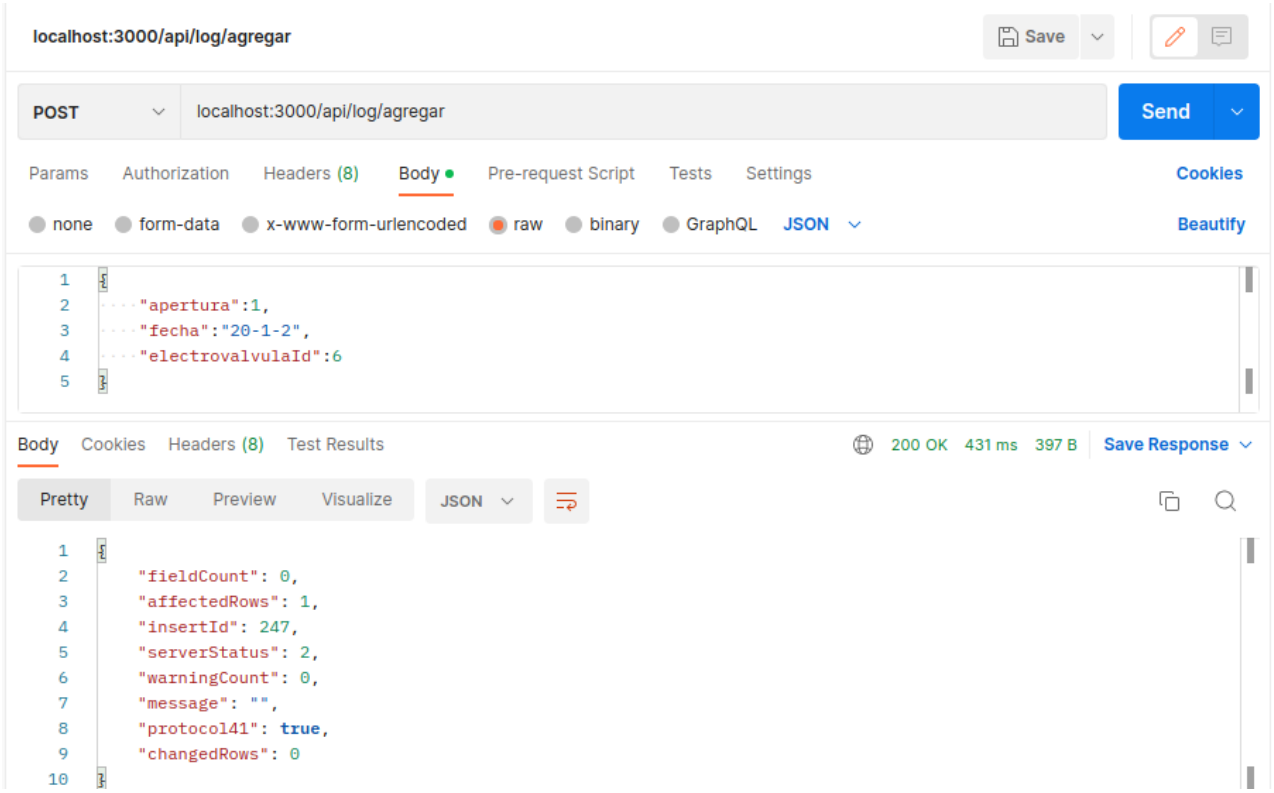
- Lista de dispositivos, en este caso son 6 dispositivos los cargados.
  http://localhost:3000/api/dispositivo
- Última medición registrada para un dispositivo (:id representa un entero de 1 a 6):
  http://localhost:3000/api/medicion/:id
- Todas las mediciones registradas para un dispositivo (:id representa un entero de 1 a 6):
  http://localhost:3000/api/medicion/:id/todas
- Último log registrado para un dispositivo electroválvula (:id representa un entero de 1 a 6):
  http://localhost:3000/api/log/:id
- Todas los logs registrados para un dispositivo (:id representa un entero de 1 a 6):
  http://localhost:3000/api/log/:id/todas
- Agregar una medición para un dispositivo electroválvula:
  POST
  http://localhost:3000/api/medicion/agregar
- Agregar una medición para un dispositivo electroválvula:

http://localhost:3000/api/log/agregar

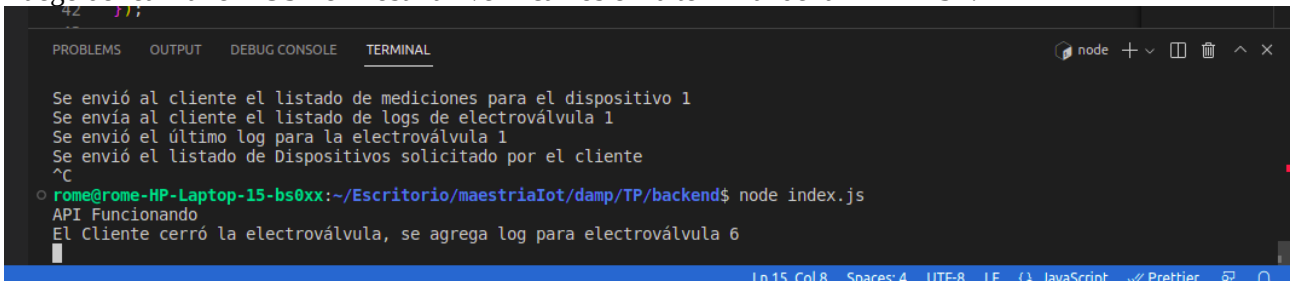
A continuación se muestra un ejemplo de método GET (utilizando el navegador) para listar los dispositivos (para no hacer muy grande la imagen no se ven todos los dispositivos pero efectivamente son 6):



A continuación se muestra un ejemplo de método POST, en este caso nos valemos de la herramienta Postman. En el ejemplo se agrega un nuevo log al registro de logs:

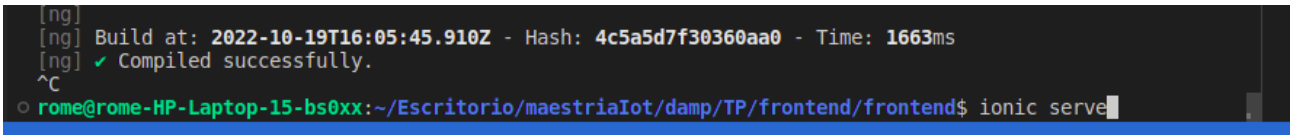


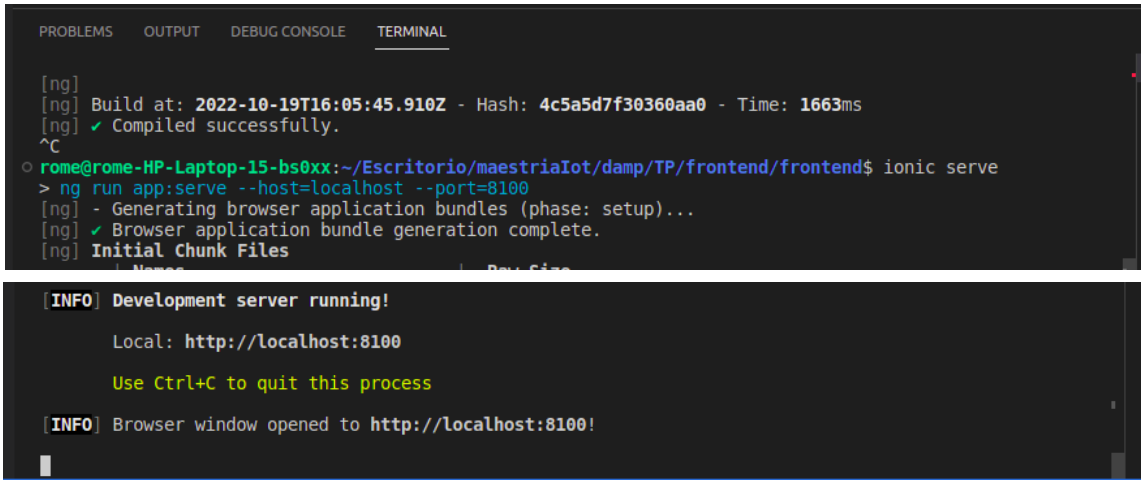
Luego de realizar el POST en Postman verificamos en la terminal de la API-REST:



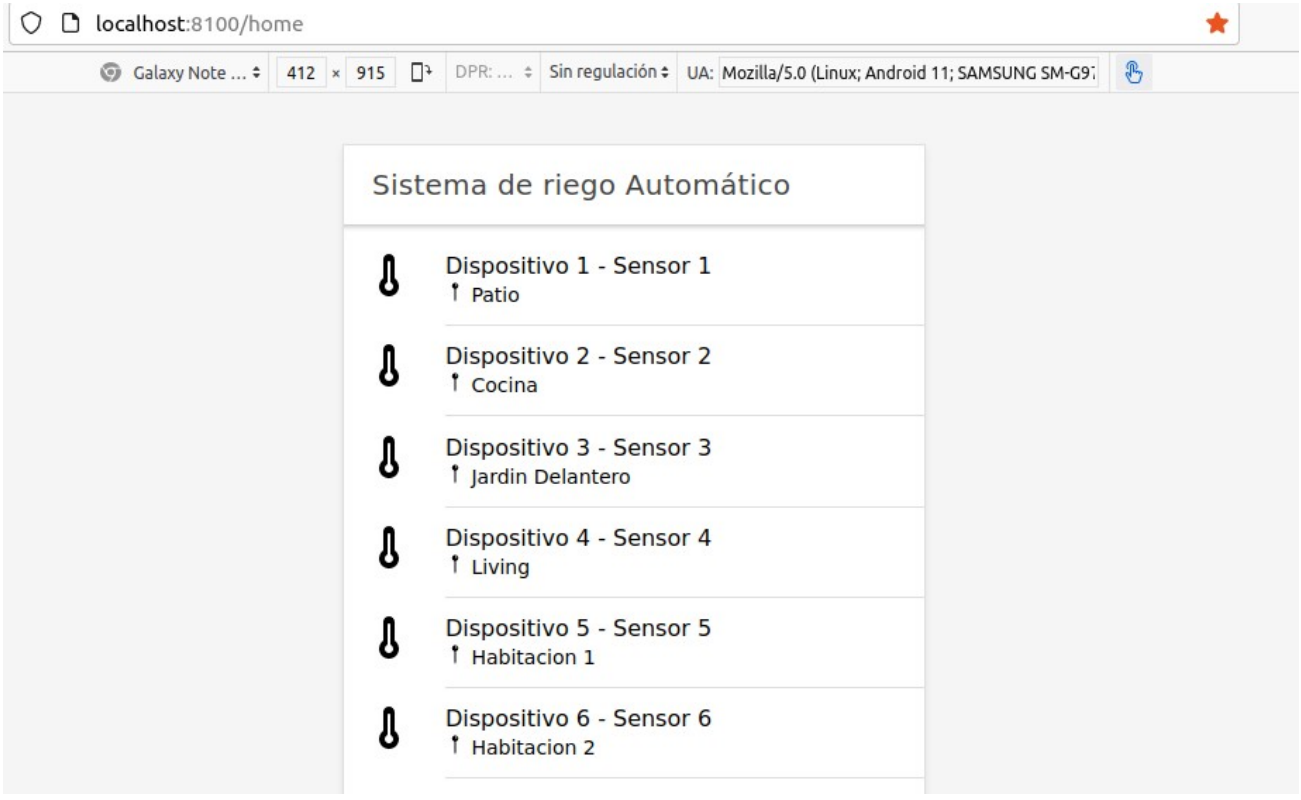
Descripción del Frontend

Para hacer funcionar el Frontend se debe ingresar a la carpeta frontend del proyecto y ejecutar en un terminal: ionic serve

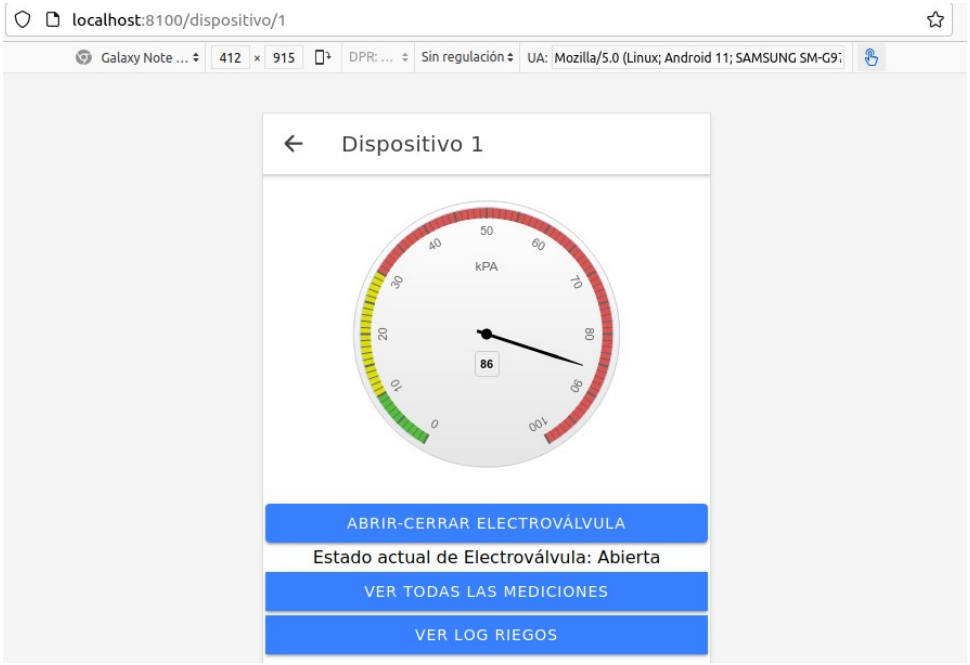




Cuando empieza a funcionar el SPA se abre en un navegador la API.
En esta pantalla se visualiza el listado de dispositivos y es la pantalla principal de la aplicación:



Al hacer click con el mouse sobre un dispositivo se accede a la pantalla propia del dispositivo:



En esta pantalla se observa para un dispositivo específico (en este caso el sensor 1):

- El último valor registrado en la base de datos en KPA.
- El último estado de la electroválvula registrado en la base de datos: Abierta o cerrada.

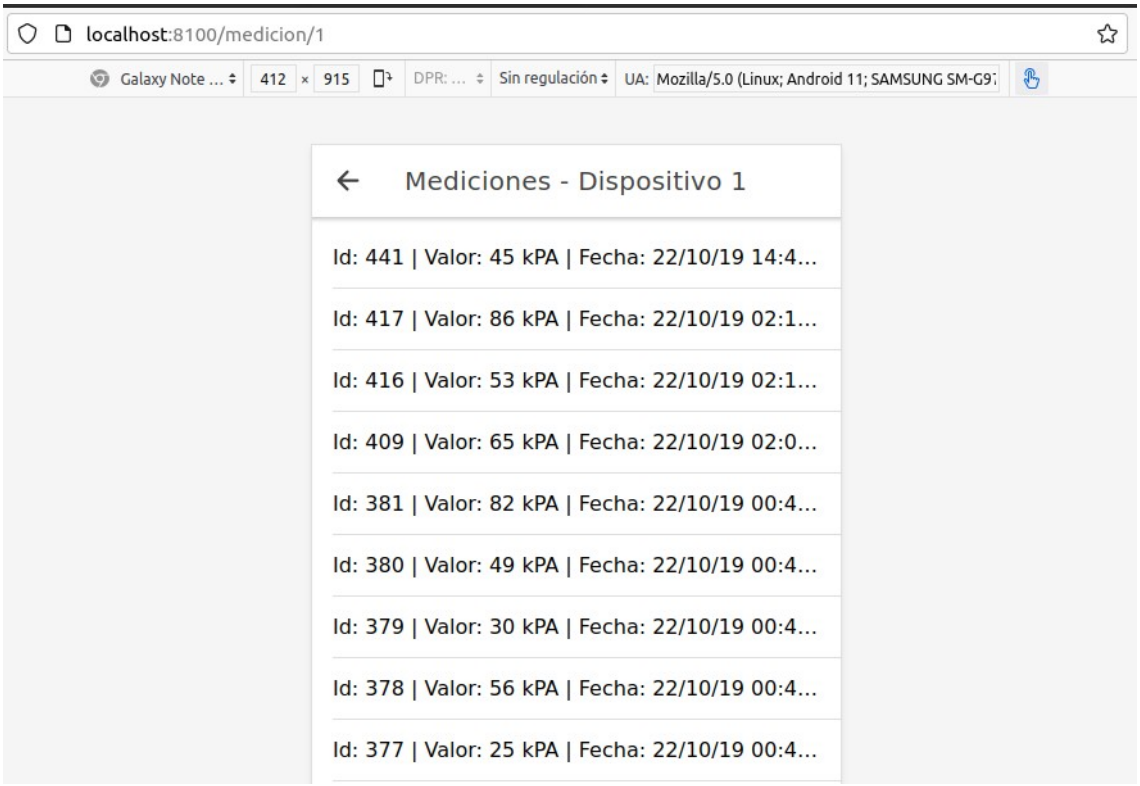
Además se cuenta con tres botones que permiten:

- ABRIR-CERRAR ELECTROVÁLVULA: permite cambiar el estado de la electroválvula.
- VER TODAS LAS MEDICIONES: permite visualizar el listado de todas las mediciones registradas en la base de datos.
- VER LOG RIEGO: permite visualizar el listado de todos los registrados en la base de datos.

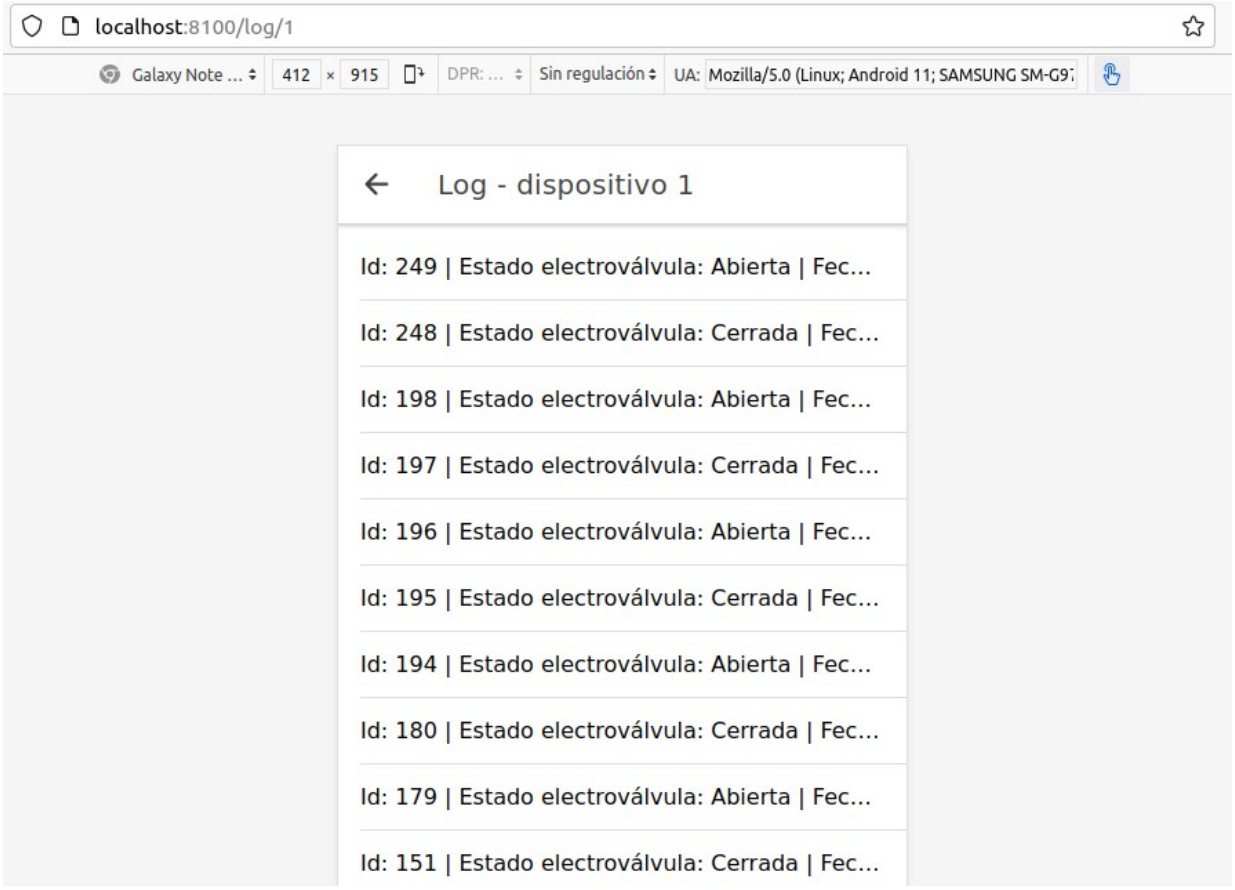
Luego de presionar el botón ABRIR-CERRAR ELECTROVÁLVULA según el caso se mostrará un mensaje sobre si se registró o no una medición y/o log y el estado actual de la electroválvula:

Presionando nuevamente:

Ventana VER TODAS LAS MEDICIONES



Ventana VER LOG RIEGO



Los endponit son:

- Lista de dispositivos:

http://localhost:8100/home

- Ventana de dispositivo (:id representa un entero de 1 a 6): http://localhost:8100/dispositivo/:1
- Todas las mediciones registradas para un dispositivo (:id representa un entero de 1 a 6): http://localhost:8100/medicion/:id
- Todas los logs registrados para un dispositivo (:id representa un entero de 1 a 6):
- http://localhost:8100/log/:id

Aclaraciones finales

Por último se destaca que en los dos proyectos se incluyeron todos los elementos pedidos en el TP que no se describen aquí pero que se comentan debidamente.