Full-Stack IoT Test

INTRODUCCIÓN

Esta pequeña API hecha en Java pretende simular el comportamiento de un sistema de IoT en una residencia. En teoría, se permite leer el la información de cada dispositivo y agregar, editar, eliminar o listar todos los dispositivos del lugar.

SOFTWARE NECESARIO Y SU INSTALACIÓN

Se necesita el siguiente software

- Java 8
- Python 3
- Spring Boot Tools
- MySQL Server 8.0 y MySQL Workbench
- NPM (con la CLI de Angular)

PROCESO DE INSTALACIÓN DE SOFTWARE

En este punto se mostrará como instalar algunas de las herramientas necesarias para que funcione el sistema.

Antes que nada, se debe descargar el código fuente del sistema, disponible en https://github.com/jorgefalconcampos hacerlo con:

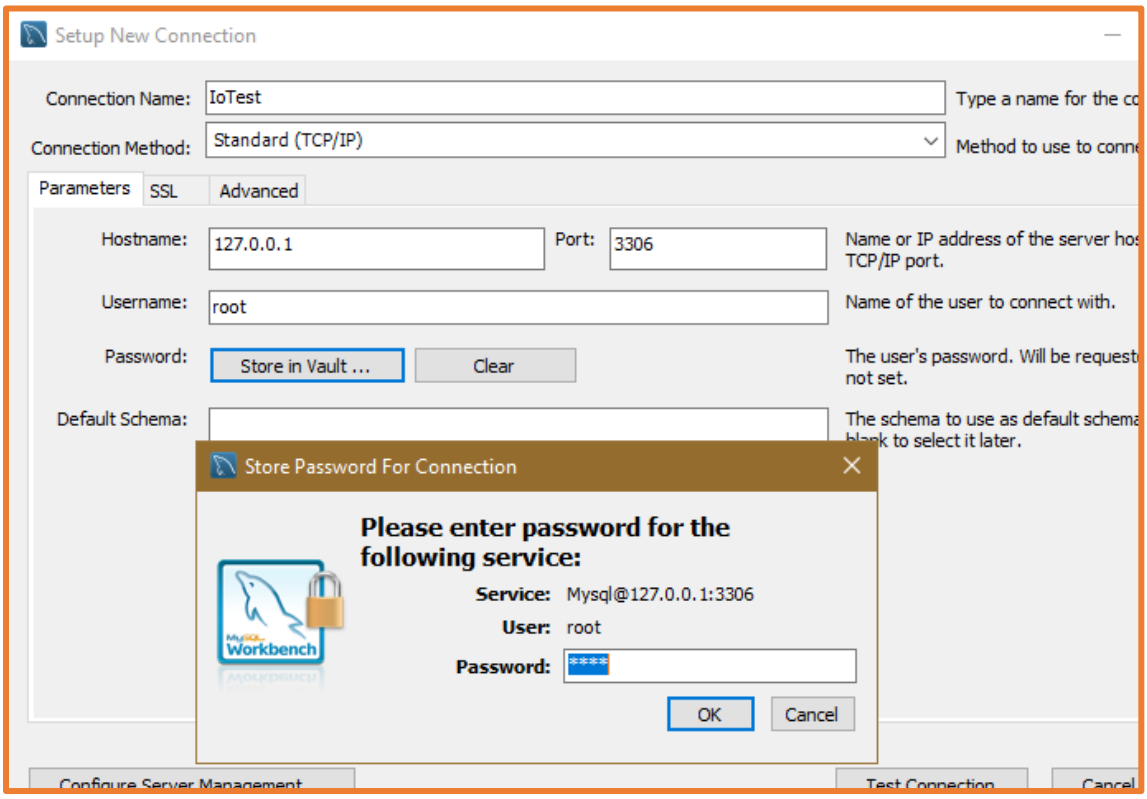
```
git clone https://github.com/jorgefalconcampos/IoTSystem.git
```

Teniendo el código fuente proceder a la instalación de las herramientas necesarias para ejecutar el sistema si es que no se cuentan instaladas en la PC.

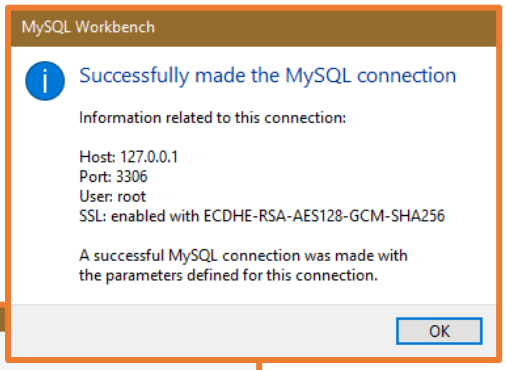
La principal de estas herramientas es **Spring Boot Tools**, que permite la creación de proyectos Spring Boot fácilmente. Se descarga desde **https://spring.io/tools** y se descarga la versión de Eclipse. Se descargará un archivo .JAR que debemos descomprimir y quedará entonces una carpeta llamada **sts-4.8.1.RELEASE.**

Dentro de esa carpeta estará el programa ejecutable de Spring Tools (SpringToolSuite4.exe) que es básicamente un Eclipse "alterado" con plugins de Sprint que facilita el desarrollo; la apertura del programa Spring Boot Tools y del proyecto se verá un poco más adelante.

Antes de crear e iniciar el proyecto de Spring Boot necesitaremos una BDD MySQL. Se deberá instalar **MySQL Server** de preferencia en su versión 8 y **MySQL Workbench**. Una vez instalados dichos requisitos, abrir Workbench y crear una nueva conexión BDD llamada **IoTest.** Dejar los ajustes por defecto del usuario root y puerto 3306 y darle una contraseña sencilla como "1234" solo para efectos prácticos y de desarrollo.

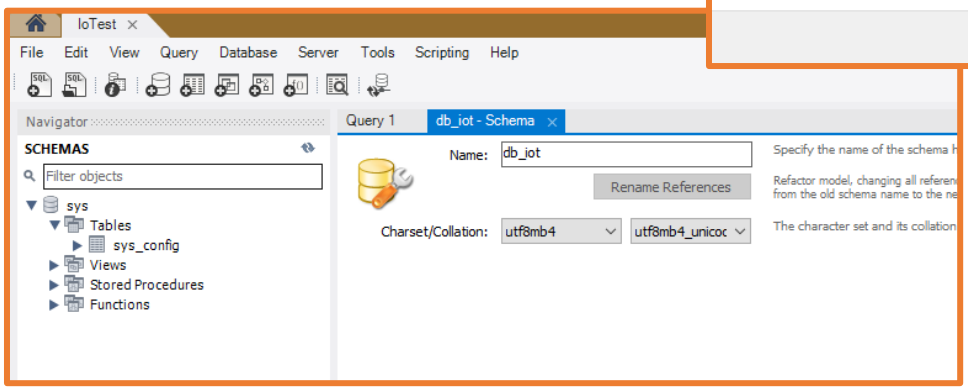


Testear la conexión si es necesario.

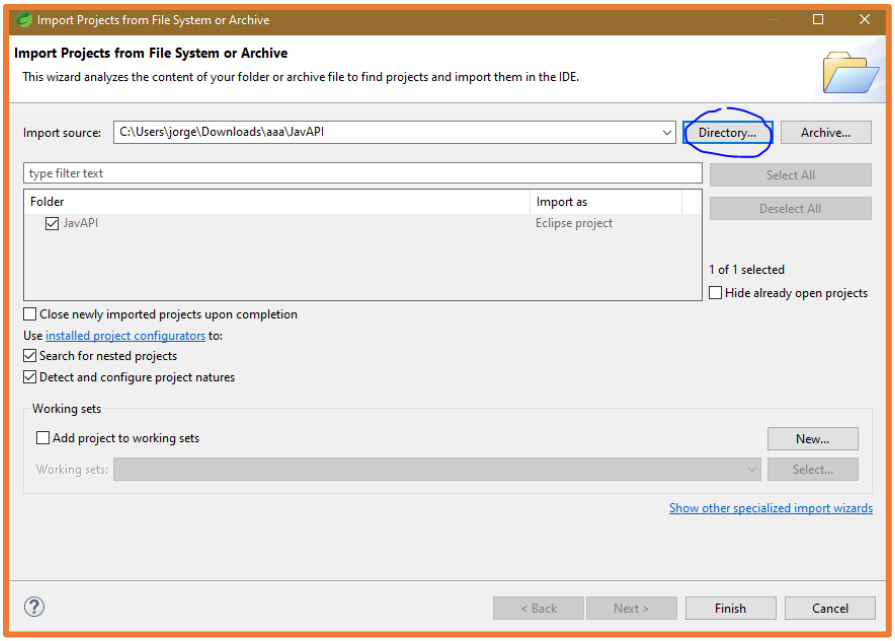


Finalmente, crear la BDD "db_iot" con el carácter set **utf8mb4** y la colación **utf8mb4_unicode**

```
CREATE SCHEMA `db_iot` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci ;
```

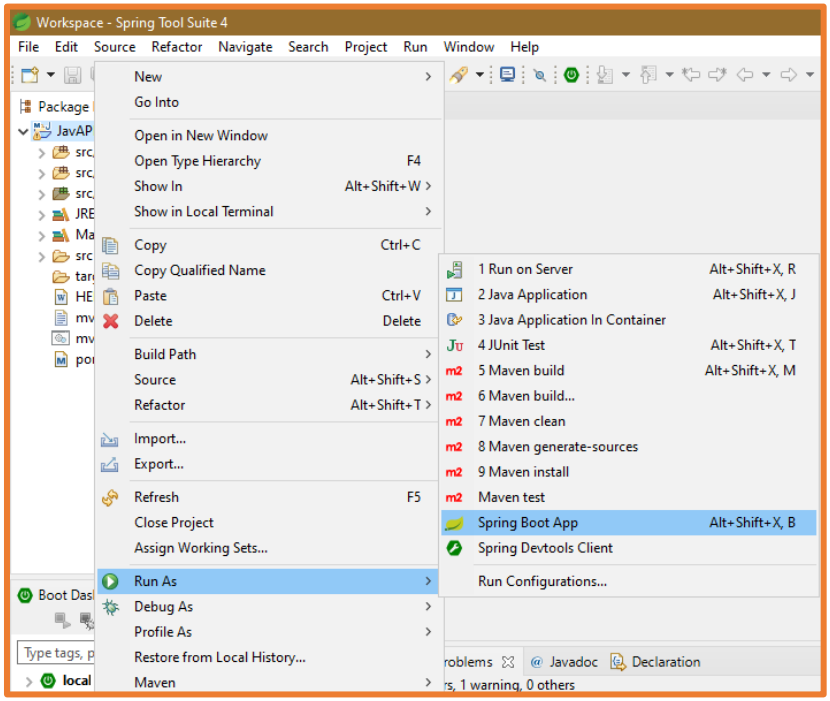


Nuestra BDD estará lista y podremos correr entonces la API. Regresamos al .exe de Spring Boot Tools que teníamos previamente y nos pedirá una carpeta donde se alojará el proyecto. Crear una carpeta llamada "Workspace" en cualquier locación que se desee y hacer clic en Ok.



Después hacemos clic en **File > Open Project From File System** y seleccionamos el directorio llamado "JavAPI" que se encuentra en "IoT System/BackEnd" que contiene el código fuente descargado de GitHub. Hacemos clic en Finish, esperamos a que cargue y listo.

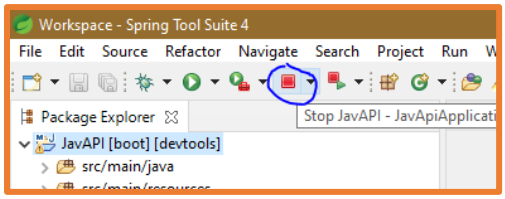
Después correr el proyecto haciendo clic derecho en el folder **JavAPI > Run As > Spring Boot App**



Puede que el firewall de Windows necesite permiso para que la aplicación pase a través del mismo. Si aparece uno o más mensajes de ese tipo, permitirlos y continuar con la ejecución.

Ahora la API estará corriendo en http://localhost:8080 .

Para detener la ejecución, hacer clic en Stop; al hacer esto los datos de la BDD se borrarán.



Después necesitamos instalar el CLI de Angular para la parte del front-end, instalarlo con:

```
npm install -g @angular/cli
```

Después instalar las dependencias con:

```
npm install
```

Una vez instalada la CLI de Angular y las dependencias necesarias, para correr el front-end en http://localhost:4200/ ir a la carpeta "IoT System/FrontEnd" y ejecutar el siguiente comando:

```
ng serve
```

Para ejecutar el dispositivo IoT, ir a la carpeta **IoT Device** y crear un entorno virtual llamado "devicesEnv" con el comando:

```
python -m venv devicesEnv
```

El paso anterior es opcional, si no se desea crear entorno virtual saltarse 2 pasos e ir directo a instalación de dependencias. Para crear el entorno virtual se deberá contar con el paquete **virtualenv**, instalarlo con:

```
pip install virtualenv
```

Y una vez instalado, crear el entorno virtual, el cual esté una vez creado, se activará con:

```
source devicesEnv/Scripts/activate
```

Y después instalar las dependencias necesarias del dispositivo IoT pero solo dentro del entorno virtual, con:

```
pip install requests PTable
```

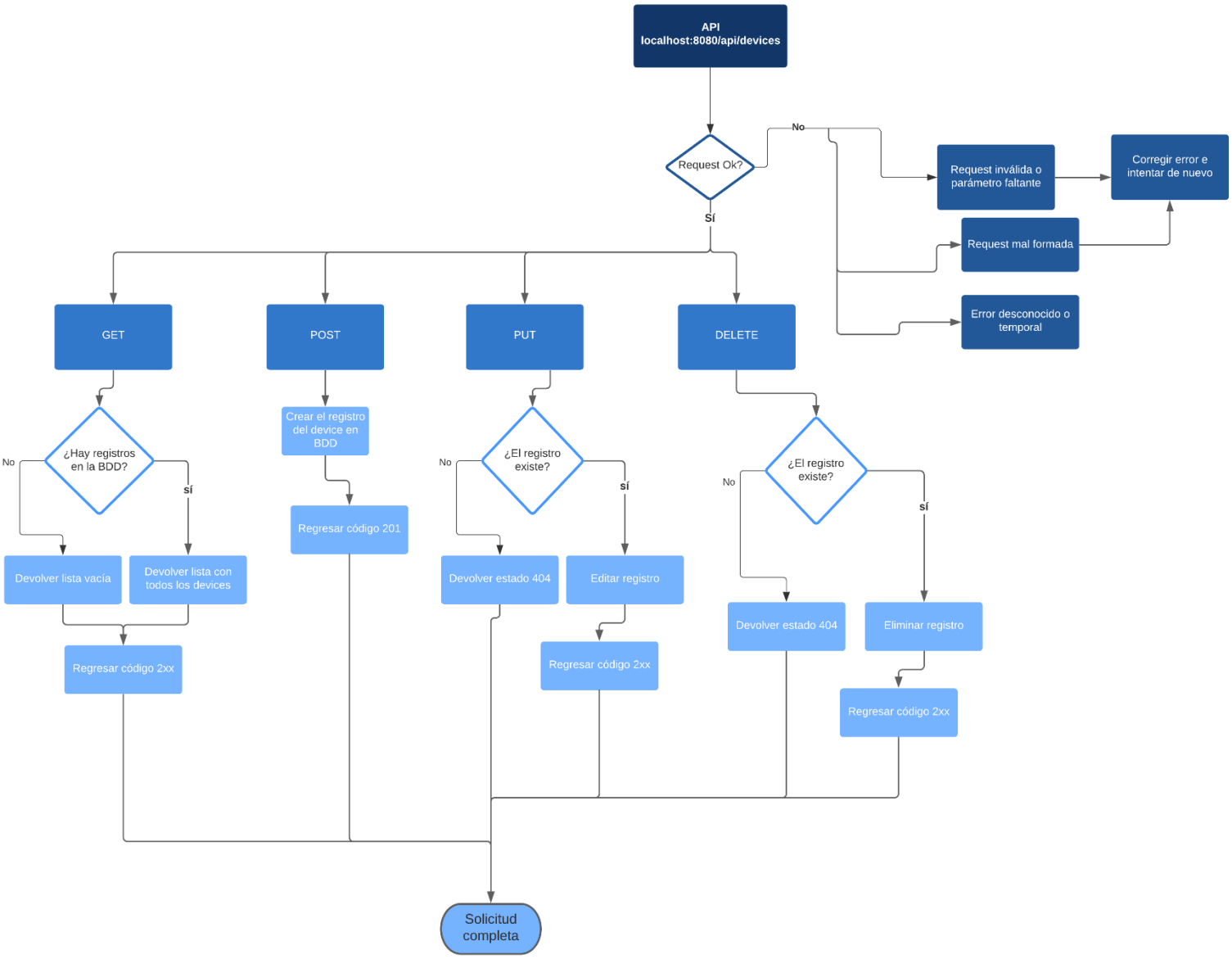
Finalmente, ejecutar el dispositivo IoT con

```
python device.py
```

Este script leerá los cambios que se hagan en el frontend del primer dispositivo creado.

Regresar a http://localhost:4200/ y probarlo.

En la página siguiente se muestra un diagrama de flujo de la API.


API
localhost:8080/api/devices
Request Ok?
No
Sí
Request inválida o parámetro faltante
Corregir error e intentar de nuevo
Request mal formada
Error desconocido o temporal
GET
POST
PUT
DELETE
¿Hay registros en la BDD?
No
sí
Devolver lista vacía
Devolver lista con todos los devices
Regresar código 2xx
Crear el registro del device en BDD
Regresar código 201
¿El registro existe?
No
sí
Devolver estado 404
Editar registro
Regresar código 2xx
¿El registro existe?
No
sí
Devolver estado 404
Eliminar registro
Regresar código 2xx
Solicitud completa