

# MANUAL DE USO LOCALHOST DEL SISTEMA DE INTEGRACIÓN DE MÓDULOS IoT (SIMIoT) -

SIMIoT funciona en conjunto con dos aplicaciones, el sistema de registro de módulos IoT o SRMIoT y el Api de Estandarización. Previo a poder registrar las lecturas de cada sensor a través del api de estandarización es necesario registrar el dispositivo el SRMIoT.

## Sistema de registro de Módulos IoT - Localhost

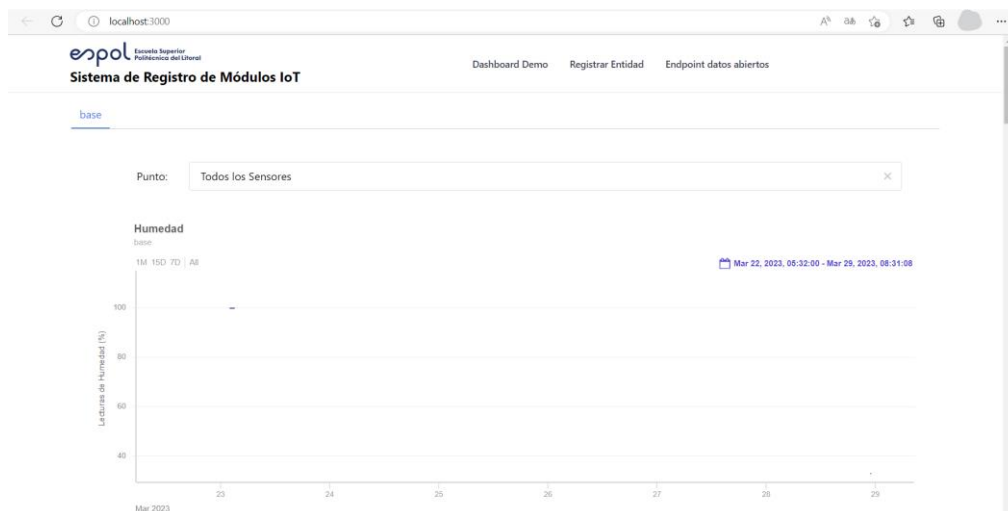
En caso de querer una versión en LocalHost, para crear una versión de producción de su aplicación primero se deben instalar las dependencias y luego ejecutar el comando:

**npm run build**

Luego se ejecuta:

**npm run preview**

Después, se podrá visualizar en <http://localhost:3000>.



Para que se pueda visualizar correctamente, se deben seguir los pasos mencionados en el documento “Manual de Instalación y Ejecución” hasta Api Standarization, ya que si desplegamos el IoT-register-Module, no se podrá correr de manera local.

Para visualizar los datos de la gráfica, se deben añadir datos a la base de datos desplegada en Hasura, para ello se debe:

1. Ir a Data/LTS Database/public/registros\_sensores, ve a Insert Row

The screenshot shows the Hasura Data Manager interface. On the left, the 'Data Manager' sidebar lists databases and tables. The main area displays the 'registros\_sensores' table with the 'Insert Row' tab selected. The form includes fields for 'id' (set to 'uuid'), 'point\_id' (set to 'character varying'), 'timestamp\_registro' (set to 'timestamp without time zone'), and 'registro' (set to a JSON object). Radio buttons allow selecting 'NULL' or 'Default' for each field. 'Save' and 'Clear' buttons are at the bottom.

2. Una vez allí, se llenan los datos necesarios para actualizar la tabla. Point\_id es obligatorio, los demás pueden ser dejados como default, estos points se pueden visualizar en la tabla point. Luego se da click en save, lo que hará que se registre el nuevo sensor.

This screenshot shows the same 'registros\_sensores' table form, but now with data entered. The 'point\_id' field is set to 'id\_humidity'. The 'registro' field contains a JSON object: 

```
{
  "id": "id_humidity",
  "dis": "Humedad",
  "kind": "integer",
  "unit": "%",
  "value": 329.9,
  "siteRef": "id_cenaim",
  "equipRef": "base",
  "humidity": true
}
```

. The 'Save' button is highlighted in orange.

3. En caso que se quiera agregar más datos a la tabla, solo cambie los valores y de click en "Insert Again".

Browse Rows

Insert Row

Modify

Relationships

Permissions

id

☐ uuid
☐ NULL
☒ Default

point\_id

☒ id\_humidity
☐ NULL
☐ Default

timestamp\_registro

☐ timestamp without time zone
☐ NULL
☒ Default

registro

☒

```

{
  "id": "id_humidity",
  "dis": "Humedad",
  "kind": "integer",
  "unit": "%",
  "value": 1040.9,
  "siteRef": "id_cenaim",
  "equipRef": "base",
  "humidity": true
}

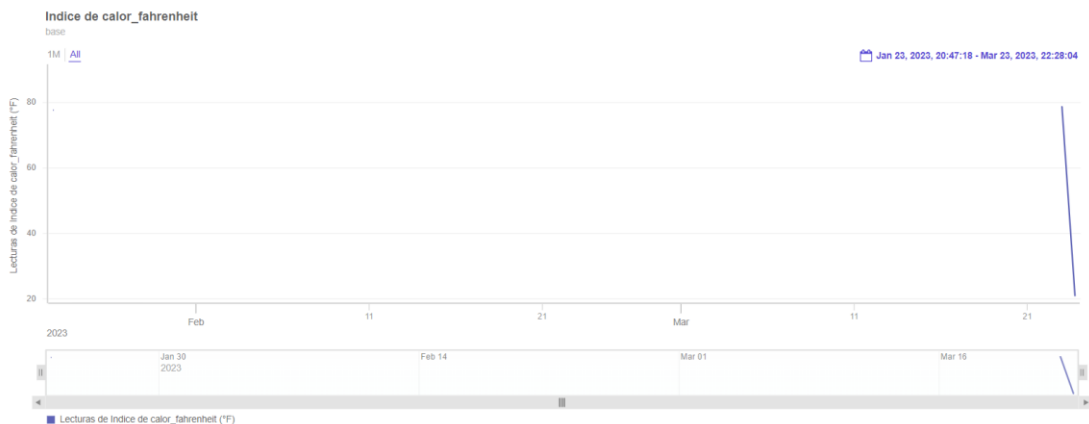
```

☐ NULL
☐ Default

Insert Again

Clear

Para visualizar una buena gráfica, se deben tener datos suficientes en la base de datos, mientras exista al menos 1 dato registrado, se mostrarán las gráficas:



Caso contrario, aparecerá un mensaje, en el cual se menciona que no existen datos que mostrar.

SIN DATOS QUE MOSTRAR

SIN DATOS QUE MOSTRAR

En caso de que se requiera modificar el código, cada vez que se realice un cambio, se debe ejecutar:

**npm run build**  
**npm run preview**

De esta manera, se pueden visualizar los nuevos cambios realizados.