# Presentación: Tecnologías para una Aplicación IoT

## Introducción

En este proyecto, desarrollaré una aplicación IoT que gestiona una base de datos en un servidor Linux remoto. Utilizaremos un **ESP32** para recopilar datos, un **servidor Flask** para manejar las solicitudes y una **base de datos PostgreSQL** para almacenar los datos. También emplearé herramientas como **PuTTY**, **Postman**, y **Arduino IDE** para facilitar el desarrollo y las pruebas. A continuación, se describen las tecnologías elegidas y su papel en el proyecto.

El lenguaje de programación será Arduino en la placa , Python en el server y Node mediante Angular en la web.

## ESP32

* **Descripción**: El ESP32 es un microcontrolador potente y versátil que cuenta con conectividad Wi-Fi y Bluetooth. Es ideal para aplicaciones IoT debido a su capacidad para manejar múltiples sensores y dispositivos conectados.
* **Uso en el Proyecto**:
  + **Recolección de Datos**: Se conectará a varios sensores (como sensores de distancia, temperatura y presión) para recopilar datos en tiempo real.
  + **Comunicación**: Enviará los datos a un servidor Flask a través de solicitudes HTTP POST.

## Flask

* **Descripción**: Flask es un microframework para Python que permite crear aplicaciones web de manera sencilla y flexible.
* **Uso en el Proyecto**:
  + **Interfaz del Servidor**: Se utilizará Flask para recibir las solicitudes del ESP32. Las funciones definidas en Flask procesarán los datos recibidos y los almacenarán en la base de datos PostgreSQL.
  + **Desarrollo Ágil**: Su estructura modular permite agregar nuevas funcionalidades y endpoints según sea necesario.

PostgreSQL

* **Descripción**: PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto, conocido por su robustez, escalabilidad y cumplimiento de estándares SQL.
* **Uso en el Proyecto**:
  + **Almacenamiento de Datos**: Todos los datos de los sensores y las lecturas se almacenarán en PostgreSQL, permitiendo un acceso estructurado y eficiente a la información.
  + **Consultas Avanzadas**: Se podrá realizar análisis y consultas complejas sobre los datos almacenados, facilitando la obtención de informes y estadísticas.

## Servidor Linux Remoto

* **Descripción**: Un servidor Linux remoto proporcionará un entorno estable y seguro para alojar la aplicación Flask y la base de datos PostgreSQL.
* **Uso en el Proyecto**:
  + **Despliegue de la Aplicación**: La aplicación Flask se ejecutará en el servidor remoto, permitiendo el acceso desde cualquier dispositivo conectado a la red.
  + **Gestión de Base de Datos**: PostgreSQL se instalará en el servidor, donde se gestionarán todas las operaciones relacionadas con los datos.

PuTTY

* **Descripción**: PuTTY es un cliente SSH y telnet que permite conectarse a servidores de manera segura a través de una interfaz de línea de comandos.
* **Uso en el Proyecto**:
  + **Acceso al Servidor**: Se utilizará PuTTY para acceder al servidor Linux remoto, donde se podrá configurar el entorno, instalar dependencias y desplegar la aplicación Flask.
  + **Gestión Remota**: Permite realizar tareas administrativas y de mantenimiento en el servidor de forma segura.

Postman

* **Descripción**: Postman es una herramienta de desarrollo que facilita la creación, envío y gestión de solicitudes HTTP.
* **Uso en el Proyecto**:
  + **Pruebas de API**: Se utilizará Postman para probar los endpoints de la aplicación Flask, asegurando que los datos se envían y reciben correctamente.
  + **Simulación de Solicitudes**: Permite simular las solicitudes del ESP32, facilitando la depuración y el desarrollo de la API.

Arduino IDE

* **Descripción**: Arduino IDE es un entorno de desarrollo integrado que permite programar placas Arduino y microcontroladores compatibles como el ESP32.
* **Uso en el Proyecto**:
  + **Desarrollo del Código del ESP32**: Se usará para escribir el código que controla el ESP32, gestionando la lectura de sensores y la comunicación con el servidor Flask.
  + **Carga de Programas**: Permite cargar el código directamente en el ESP32 y realizar pruebas de funcionalidad.

# ¿Por qué usar bases de datos Relacional?

Usar una base de datos relacional en un proyecto de Internet de las Cosas (IoT) que involucra sensores y estaciones de medición es una decisión estratégica que ofrece varias ventajas. A continuación, se detallan las razones por las cuales una base de datos relacional es adecuada para este tipo de proyecto, considerando la complejidad y la variabilidad de los sensores y las estaciones de medición:

**1. Estructura y Organización de Datos**

* **Relaciones Complejas**: En un sistema IoT con múltiples tipos de sensores y estaciones de medición, es probable que haya relaciones complejas entre diferentes entidades (sensores, estaciones, ubicaciones). Las bases de datos relacionales permiten definir estas relaciones a través de claves primarias y foráneas, lo que facilita la organización y gestión de los datos.
* **Modelado de Entidades**: modelar entidades como Sensores, Estaciones de Medición, y Ubicaciones de manera clara y estructurada. Cada entidad puede tener atributos específicos (por ejemplo, tipo de sensor, ubicación de la estación, etc.), y las relaciones entre estas entidades se pueden representar con facilidad.

**2. Flexibilidad para Múltiples Tipos de Sensores**

* **Escalabilidad**: Con una base de datos relacional, es fácil agregar nuevos tipos de sensores o estaciones de medición sin tener que reestructurar completamente la base de datos. Se puede crear tablas específicas para cada tipo de sensor y relacionarlas con las estaciones y ubicaciones pertinentes.
* **Consistencia de Datos**: La normalización de datos en una base de datos relacional ayuda a mantener la consistencia. Por ejemplo, si un tipo de sensor cambia su forma de medir o agrega nuevos parámetros, solo necesita actualizar la estructura de la tabla relacionada, lo que minimiza el riesgo de inconsistencia.

**3. Consultas Potentes y Análisis de Datos**

* **Lenguaje SQL**: Las bases de datos relacionales utilizan SQL (Structured Query Language), que permite realizar consultas complejas para extraer, filtrar y agregar datos de manera eficiente. Esto es particularmente útil para análisis de datos y generación de informes, como monitorear el rendimiento de diferentes estaciones de medición o analizar tendencias a lo largo del tiempo.
* **Agrupaciones y Resúmenes**: Se Puede fácilmente agrupar datos por estación o ubicación, lo que facilita la visualización de datos relevantes y la toma de decisiones basadas en esos datos.

**5. Soporte para Múltiples Ubicaciones**

* **Gestión de Ubicaciones**: Las bases de datos relacionales pueden manejar múltiples ubicaciones y permitir la asignación de varias estaciones a una misma ubicación. Esto es ideal para aplicaciones como la gestión de estacionamientos o el control de curado del hormigón en diferentes obras.
* **Escalabilidad**: Si tu sistema crece y necesitas agregar más estaciones o sensores, una base de datos relacional puede escalar eficientemente para manejar un mayor volumen de datos y consultas sin comprometer el rendimiento.

**Descripción de las entidades**

1. **Estaciones**:
   * **id\_estacion**: Identificador único de la estación (clave primaria).
   * **nombre**: Nombre de la estación.
   * **ubicación**: Ubicación geográfica de la estación.
2. **Sensores**:
   * **id\_sensor**: Identificador único del sensor (clave primaria).
   * **nombre**: Nombre del sensor.
   * **tipo**: Tipo de sensor (por ejemplo, temperatura, humedad, etc.).
   * **id\_estacion**: Identificador de la estación a la que está asociado el sensor (clave foránea).
3. **Lecturas**:
   * **id\_lectura**: Identificador único de la lectura (clave primaria).
   * **valor**: Valor registrado por el sensor.
   * **fecha\_hora**: Fecha y hora en que se registró la lectura.
   * **id\_sensor**: Identificador del sensor que realizó la lectura (clave foránea).