Proyecto Final: Servidor para dispositivos IoT con Django y MQTT

Universidad Nacional Autónoma de México (UNAM)
Facultad de Ingeniería

Materia: Temas Selectos de Programación I
Profesor: Ing. Emiliano Nava Morales 1964
Alumno: Treviño Selles Jorge Eithan
Grupo: 5

Ciudad Universitaria, México
11 de diciembre de 2024

Índice

1. Introducción 3

2. Herramientas utilizadas 3

3. Protocolo MQTT 4

4. Desarrollo del servidor web 4

5. Desarrollo del dispositivo IoT 6

6. Conclusiones 7

7. Referencias 7

1. Introducción

Los dispositivos IoT (Internet of Things) son cada vez más comunes en la vida cotidiana, desde electrodomésticos hasta sistemas de seguridad. Estos dispositivos se conectan a una red para enviar y recibir datos, y en muchos casos, se pueden controlar remotamente. Para facilitar la interacción con estos dispositivos, se propone un servidor web que permita a los usuarios monitorear y controlar sus dispositivos IoT desde cualquier lugar con acceso a internet.

En este proyecto, se desarrollo un servidor web que permitió a los usuarios visualizar las medidas y controlar el tiempo de medidas de diversos sensores. Fue necesario utilizar diversas tecnologías que, en conjunto, permitieron la creación de un servidor web funcional y fácil de usar.

A través de este tipo de servidor se pueden controlar dispositivos IoT de manera remota, lo que facilita la interacción con estos dispositivos y permite a los usuarios monitorear y controlar sus dispositivos IoT desde cualquier lugar con acceso a internet. Esto tiene aplicaciones tanto en el hogar como en la industria, donde la automatización y el monitoreo remoto son cada vez más importantes en una era regida por la conectividad y la información.

2. Herramientas utilizadas

- **Python:** Lenguaje de programación utilizado para el desarrollo del servidor web.
- **Django:** Framework de desarrollo web utilizado para el desarrollo del servidor web.
- **SQLite:** Sistema de gestión de bases de datos utilizado para almacenar los datos de los sensores, utilizado por Django por defecto.
- **HTML:** Lenguaje de marcado utilizado para la creación de las páginas web.
- **CSS:** Lenguaje de estilos utilizado para dar formato a las páginas web.
- **JavaScript:** Lenguaje de programación utilizado para la interacción con el servidor web.
- **jQuery:** Biblioteca de JavaScript utilizada para simplificar la interacción con el servidor web.
- **ESP32:** Microcontrolador utilizado para tomar medidas de sensores y enviarlas al servidor web.
- **AWS EC2:** Servicio de Amazon Web Services utilizado para alojar el servidor web.
- **Linux:** Sistema operativo utilizado para el desarrollo del servidor web.
- **Mosquitto:** Broker MQTT utilizado para la comunicación entre el servidor web y los dispositivos IoT.

3. Protocolo MQTT

MQTT (Message Queuing Telemetry Transport) es un protocolo de mensajería ligero que se utiliza para la comunicación entre dispositivos IoT. MQTT es un protocolo de publicación/suscripción (PubSub) que permite a los dispositivos enviar y recibir mensajes de forma eficiente y confiable. MQTT es ampliamente utilizado en aplicaciones IoT debido a su simplicidad y eficiencia.

En este proyecto, se utilizó MQTT para la comunicación entre el servidor web y los dispositivos IoT. Los dispositivos IoT publican mensajes en un tema específico, y el servidor web se suscribe a ese tema para recibir los mensajes. Esto permite a los dispositivos IoT enviar datos al servidor web de forma sencilla y eficiente.

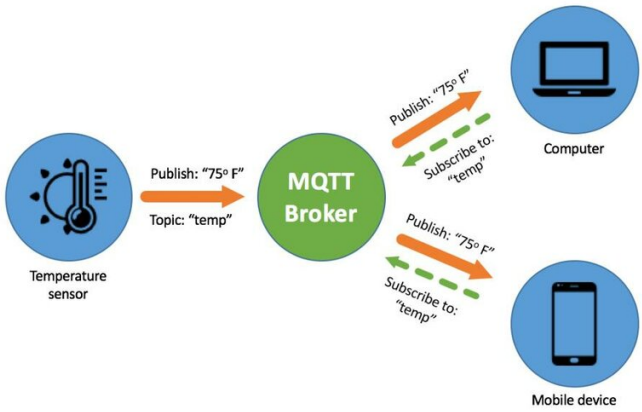

Figura 1: Protocolo MQTT

El broker MQTT utilizado en este proyecto fue Mosquitto, que es un broker MQTT de código abierto que se puede instalar en sistemas Linux. Mosquitto se encargó de gestionar la comunicación entre el servidor web y los dispositivos IoT, garantizando la entrega de los mensajes de forma segura y eficiente.

4. Desarrollo del servidor web

El servidor web se desarrolló utilizando el framework Django, que es un framework de desarrollo web de alto nivel que fomenta el desarrollo rápido y limpio. Django proporciona una arquitectura de desarrollo basada en el patrón de diseño Modelo-Vista-Controlador (MVC) que facilita la creación de aplicaciones web complejas. Django se estructura en aplicaciones, que son módulos independientes que se pueden reutilizar en diferentes proyectos.

El servidor web constó únicamente de una aplicación, llamada iot_app, que se encargó de manejar las vistas y los modelos de la aplicación. La aplicación se dividió en las siguientes partes:

- **Vistas:** Las vistas son funciones que se encargan de manejar las peticiones HTTP y

devolver una respuesta. En este caso, las vistas se encargaron de mostrar las páginas web y procesar las peticiones de los usuarios.

- **Modelos:** Los modelos son clases que representan las tablas de la base de datos. En este caso, se crearon dos modelos: **Device** y **Log**. El modelo **Device** se encargó de almacenar la información de los dispositivos IoT, como su identificador y su estatus de conexión. El modelo **Log** se encargó de almacenar los mensajes de registro enviados por los dispositivos IoT.
- **Plantillas:** Las plantillas son archivos HTML que se utilizan para generar las páginas web. En este caso, se crearon plantillas para mostrar la lista de dispositivos, el detalle de un dispositivo y los mensajes de registro.
- **Suscriptor MQTT:** Se creó un suscriptor MQTT que se encargó de recibir y gestionar los mensajes de los dispositivos IoT. El suscriptor MQTT se conectó al broker MQTT y se suscribió a los temas específicos de cada dispositivo IoT.
- **URLs:** Las URLs son patrones de URL que se utilizan para enrutar las peticiones HTTP a las vistas correspondientes. En este caso, se crearon URLs para mostrar la lista de dispositivos, el detalle de un dispositivo y los mensajes de registro.
- **Configuración:** Se configuró el archivo **settings.py** para definir la configuración del servidor web, como la base de datos, las plantillas y los archivos estáticos.
- **Administrador:** Se creó un administrador de Django para gestionar los dispositivos IoT y los mensajes de registro. El administrador permitió a los usuarios agregar, modificar y eliminar dispositivos IoT, así como ver los mensajes de registro.

Para cada uno de los dispositivos IoT, se crearon tres temas de MQTT específicos:

- $devices/ < device\_id > /logs$**:** Tema utilizado para enviar mensajes de registro al servidor web.
- $devices/ < device\_id > /connect$**:** Tema utilizado para que el ESP32 envíe su estatus de conexión al servidor web.
- $devices/ < device\_id > /response$**:** Tema utilizado para que el servidor web configure el tiempo de medidas del ESP32.

La comunicación entre el servidor web y los dispositivos IoT se realizó de la siguiente manera:

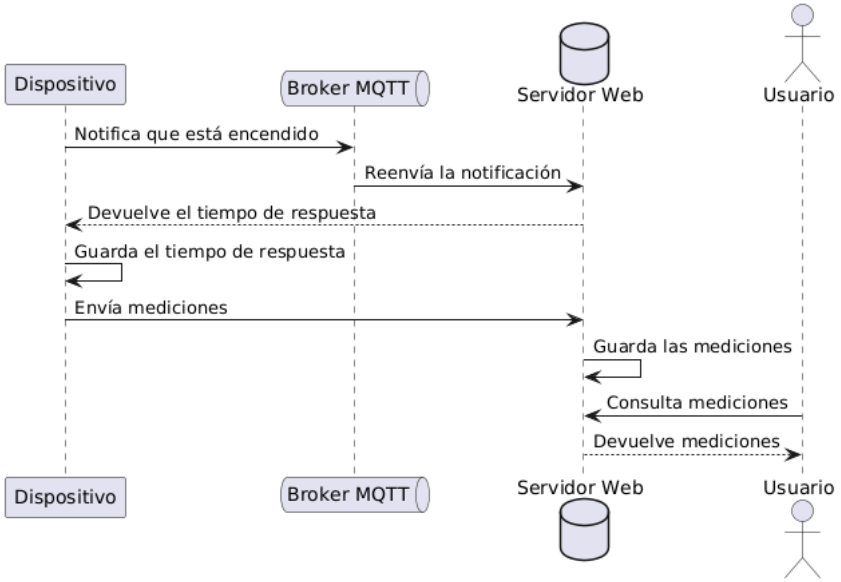

Figura 2: Comunicación entre el servidor web y los dispositivos IoT

Para actualizar el tiempo de respuesta, se hace de la siguiente manera:

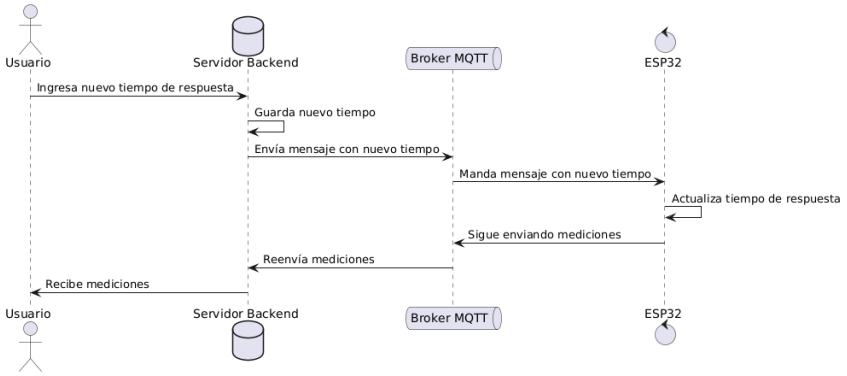

Figura 3: Comunicación entre el servidor web y los dispositivos IoT

5. Desarrollo del dispositivo IoT

El dispositivo IoT se desarrolló utilizando el microcontrolador ESP32, que es un microcontrolador de bajo costo y alto rendimiento que se utiliza en aplicaciones IoT. El ESP32 se programó utilizando el entorno de desarrollo de Arduino, que es un entorno de desarrollo de código abierto que se utiliza para programar microcontroladores.

El código utilizado para el ESP32 constó de las siguientes partes:

- **Conexión WiFi:** Se configuró la conexión WiFi del ESP32 para conectarse a la red WiFi del usuario.
- **Conexión MQTT:** Se configuró la conexión MQTT del ESP32 para conectarse al broker MQTT y publicar mensajes en los temas específicos.

- **Medidas de sensores:** Se programó el ESP32 para tomar medidas de los sensores y publicarlas en el tema de mensajes de registro.
- **Control de tiempo de medidas:** Se programó el ESP32 para recibir mensajes del servidor web y configurar el tiempo de medidas de los sensores.
- **Estatus de conexión:** Se programó el ESP32 para enviar su estatus de conexión al servidor web a través del tema de conexión.

El ESP32 se comunicó con el servidor web a través de los temas MQTT específicos, enviando mensajes de registro, recibiendo mensajes de configuración y enviando su estatus de conexión. El ESP32 se encargó de tomar medidas de los sensores y enviarlas al servidor web, permitiendo a los usuarios monitorear y controlar los dispositivos IoT desde cualquier lugar siempre y cuando tengan acceso a internet.

6. Conclusiones

El desarrollo de un servidor web para dispositivos IoT es una herramienta útil que permite a los usuarios monitorear y controlar sus dispositivos IoT desde cualquier lugar con acceso a internet. El servidor web desarrollado en este proyecto permitió a los usuarios visualizar las medidas y controlar el tiempo de medidas de diversos sensores, facilitando la interacción con los dispositivos IoT.

El uso de MQTT para la comunicación entre el servidor web y los dispositivos IoT fue fundamental para garantizar la entrega de los mensajes de forma segura y eficiente. El uso de Django como framework de desarrollo web permitió la creación de un servidor web funcional y fácil de usar, con una arquitectura MVC que facilitó el desarrollo de la aplicación.

En resumen, este proyecto demostró la importancia de la conectividad y la comunicación en el desarrollo de aplicaciones IoT, y cómo un servidor web puede facilitar la interacción con los dispositivos IoT. El servidor web desarrollado en este proyecto es una herramienta útil que puede tener aplicaciones tanto en el hogar como en la industria, donde la automatización y el monitoreo remoto son cada vez más importantes en una era regida por la conectividad y la información.

7. Referencias

- Django REST framework. (2024). *The web framework for perfectionists with deadlines — Django*. Django. https://www.djangoproject.com/
- Eclipse Foundation. (2024). *Eclipse Mosquitto™ An open source MQTT broker*. Eclipse Foundation. https://mosquitto.org/
- Espressif Systems. (2024). *ESP32 - Espressif Systems*. Espressif Systems. https://www.espressif.com/en/products/socs/esp32
- MQTT. (2024). *MQTT*. MQTT. https://mqtt.org/