IoT Device Monitoring and Alerting System with Python

This is a project for sending log messages from an IOT device to a backend system using MQTT (Broker, Subscriber and Publisher) with Django.

BY:
Sunday AJAYI
sunnexajayi@gmail.com

HOW I FACED THE PROBLEM

After reviewing the issue, I realized we will be dealing with the following features:

1. **MQTT Publisher:** This will be the IoT device that will push the log messages

2. **MQTT Subscriber:** This will be the backend system that will receive the log messages, store into a database via API and the take necessary actions (we needed like send alert notifications, push notifications, etc.)

3. **API System:** A backend system built with Python that manages the API calls for:

    a. Pushing the log messages to the database

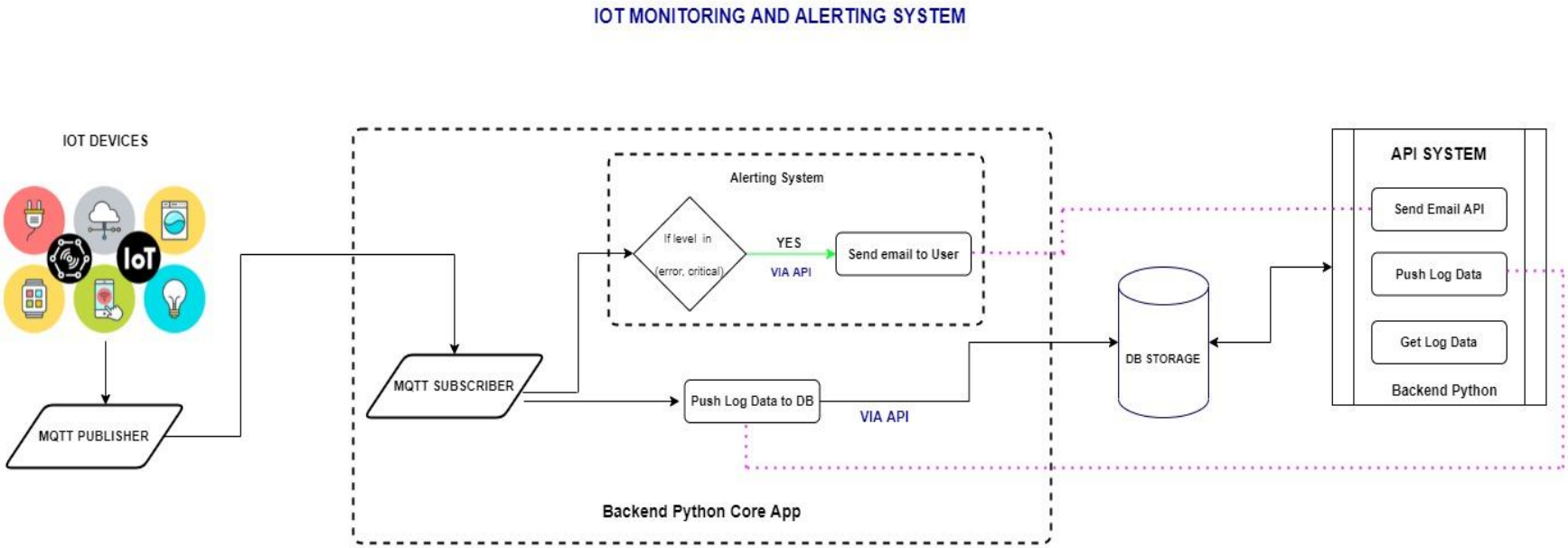
    b. Sending of emails

    c. Pulling the log messages with specific filters (such as searching with date range, device_id, location_id, etc.

4. **A database system:** I made use of sqlite DB for the assessment. The database stores the logs messages pushed from the IoT device. In a production environment, I will deploy a managed Postgres database.

5. **MQTT Broker:** An intermediary entity that enables MQTT clients to communicate. Specifically, an MQTT broker receives messages published by clients, filters the messages by topic, and distributes them to subscribers. I used Mosquitto.

WORK FLOW


IOT MONITORING AND ALERTING SYSTEM
IOT DEVICES
IoT
MQTT PUBLISHER
MQTT SUBSCRIBER
Alerting System
If level in
(error, critical)
YES
VIA API
Send email to User
Push Log Data to DB
VIA API
Backend Python Core App
DB STORAGE
API SYSTEM
Send Email API
Push Log Data
Get Log Data
Backend Python

RESULTS

- I was able to create a temporary MQTT publisher script and pass the expected json payload which I ran to publish the log messages to the subscriber listening to the set topic.
- The MQTT subscriber was able to get the published message which I piped to the database via the API system.
- I was able to send alerts via email whenever the level was error or critical
- I was able get record of the log messages stored on database via the created API.

RECOMMENDATION AND FUTURE WORK

1. I will Authenticate the APIs used and build a User management system : Activate , deactivate , onboard user,etc
2. I will make use of more robust managed cloud relational database such as PostgreSQL.
3. I will make use of a managed MQTT Server
4. Convert the backend python core application run as a service (daemon)
5. Deployment the whole backend applications (core application and API system) to a proper production server.

BY:
Sunday AJAYI
sunnexajayi@gmail.com