Report of Homework 3

Group8: Emanuela Piga (s333550) - Dilleta Ceschini (s333117) - Shiva Pourfeilieh (S329207)

Exercise 1: Data Collection, Communication, and Storage

MQTT is a portable messaging protocol that operates on a publish-subscribe model. In this system, one device (the publisher) sends data to a central hub (the broker), and other devices (the subscribers) receive the data they are interested in. It is specifically designed to perform efficiently with low-power devices and slow networks, which are typical in IoT environments.

REST, on the other hand, is a communication protocol commonly used on the web. It follows a request-response model, where one device (the client) requests information and another (the server) provides it. While REST is simple and widely adopted for web services, it is not tailored to meet the specific demands of IoT systems.

For an application like this, where fast, continuous, and reliable updates from IoT devices are needed, MQTT is a much better fit than REST. It's effortless, efficient, and specifically designed for scenarios like monitoring temperature and humidity in real-time.

Firstly, MQTT is designed to minimize data overhead. It sends only the essential information in small, lightweight messages, which is perfect for devices like the Raspberry Pi that have limited processing power. In contrast, REST adds a lot of extra information to each message, like HTTP headers, which makes the messages heavier and slower to process.

Another key advantage of MQTT is its ability to provide real-time communication. The publisher can send data immediately when it becomes available, and subscribers receive it instantly. This is ideal for monitoring constantly changing data, such as temperature and humidity levels. REST, however, relies on a request-response model, where devices have to continuously ask for updates. This polling mechanism can introduce delays and isn't as efficient for real-time data transmission.

MQTT also excels in saving bandwidth. Its lightweight nature means it uses less internet data, making it suitable for environments where network resources might be limited or slow. REST, on the other hand, consumes more bandwidth due to its repeated request and response cycles.

Scalability is another area where MQTT shines. It can handle many devices seamlessly, all communicating through the same broker. This makes it well-suited for IoT systems with numerous sensors. REST, however, can struggle under the load of many devices, as each one has to individually request updates from the server, potentially leading to performance bottlenecks.

Finally, MQTT offers reliable data delivery, even in unstable network conditions, thanks to its Quality of Service (QoS) features. This ensures that messages are delivered as expected. REST does not have similar built-in mechanisms for ensuring reliable data delivery, which can result in data loss or delays in transmission.

Exercise 2: Data Retrieval & Visualization

In order to implement the required functionality of retrieving humidity and temperature data from the mac address, the method GET has been chosen.
This decision is justified given that the task only involves accessing and retrieving existing information, without making any changes to the data stored on the server.
Indeed, it was not required to insert, modify or delete data (actions for which the use of PUT, POST, DELETE would have been more appropriate); GET, rather, is the most suitable choice in this context, because it ensures a straightforward and efficient retrieval process, perfectly aligned with the goal of accessing the required data without any side effects.
This approach keeps the implementation simple and focused while adhering to RESTful principles.
Further reasons and considerations on why GET is the best choice:
beyond the reasons mentioned above, can be carried out a deeper analysis of the motivation driving this choice.
Firstly, the **/data/{mac_address}** endpoint is developed to retrieve historical temperature and humidity data for a specific device and GET, in REST APIs, is actually the method used to retrieve data from the server.
It does not modify any data on the server, but simply retrieves existing data according to the specified MAC address. Furthermore, since the aim of this endpoint is just to retrieve the required data, we just want to read information from the server, without changing anything. The GET approach just fetches historical data, perfectly fitting with the requirement.
Parameters like *start_date* and *end_date* are defined by the endpoint, to filter the data by the date range, and it is a common function performed by GET.
By maintaining a simple implementation and following the required practices, this method guarantees both logic and efficiency; the choice is reliable and in line with the design of REST APIs.