Message queue exercise document

Topic-based communication with the message queue system implemented in this exercise works really well for this kind of use case. Implementing something like this using HTTP would be more difficult. It could be done, for example, by implementing an API in place of the message queue where services can send messages and interested services can request for them. This would require a database and it would be overall slower since services need to send requests etc. This would be a more modular solution. The system could also be simply hard coded in a way where the messages are sent directly with HTTP from ORIG to OBSE and IMED. Then from IMED also to OBSE. This is not very elegant and if the system were bigger, it would quickly become a mess.

On the other hand, the topic-based communication solution works perfectly for this. The main advantage here is the loose coupling. We simply send messages to the RabbitMQ-service with a specific topic. Zero or more services can then receive messages from topics they are interested in. The sender needs to publish the message only once unlike in the latter HTTP example where it had to be sent individually to each service. Also, adding new services would be easy. At least in my implementation, I used the same modular functions for publishing and receiving messages in each service that needed them.

There was a lot to learn in this exercise. First, I learned how the topic-based communication is implemented using RabbitMQ and specifically how it is implemented in Node. Second, I learned how to implement a bigger system with multiple services using Docker Compose. I have never had to have this many services in one system and them communicating with a message queue, a Docker volume and also providing the results with an HTTP-service. Overall, the architecture was interesting.