Spring Microservices - Terminals Pooling

- You need to create 2 microservices, one as the client and one as the server.
- The server should have a controller that verifies and logs request parameters it receives from the client, sleeps for 1 second before returning HTTP 200 OK status to the client with the following verification:
  - Server must verify that while it is locking a terminal, it doesn't receive other request with the same terminalId.
  - Server must also verify that the sequence number of the terminal it receives is correct.
  - If either the terminalId or the sequence number is not correct, it will return HTTP 400 (Bad Request) error to the client.
- The client should have a controller that expose a PUT method that does the following:
  - Upon receiving a PUT request, the client build a message payload.
  - The payload contains:
    - terminalId
    - sequenceNo
    - timestamp
  - The client has 5 terminals with the following IDs: 1001, 1002, 1003, 1004, 1005.
  - Each time the client receives the request, it will construct the message payload using an available terminal.
  - When a terminal is being used, it should be locked and not used by other threads until it is releases.
  - Each time a terminal is used, you need to increase the sequence number which is unique per terminal.
  - The sequence number has to be looped from 0 to 7. So that we it reaches 7, it will go back to 0.
  - For example, if you have 2 threads calling the client at the same time, you will have 2 message payloads with the following data:
    - Message payload 1:

      ```
      {
        "terminalId": "1001",
        "sequenceNo": 0,
        "timestamp": 1552904148143
      }
      ```

    - Message payload 2:

      ```
      {
        "terminalId": "1002",
        "sequenceNo": 0,
        "timestamp": 1552904148143
      }
      ```

  - Send the payload to the server.
  - If all the terminals are being used, the client should wait and retry until there is an available terminal. The maximum wait time is 30 seconds. If after 30 seconds and there is still no available terminal, client will return HTTP 503 (Service Unavailable) error.
- To test, we will deploy 1 instance of the server and 3 instances of the client. The 3 clients will share the same pool of terminals. So if client 1 is locking terminal 1001, client 2 should not use the same terminal.
- We will run a multi threaded class to test the robustness of the code.
- You are free to use queue, cache, database or anything else you see fit.