Microservices

Based on the codebase created during the previous module, implement follow functionality

Description

The main idea behind the creation of this microservice is as follows.
Every time a training session is planned or cancelled for a trainer; this data is transferred to a newly created microservice.
The Microservice calculates the number of working hours of each trainer for each month and stores this information in in-memory database.
In addition, when requesting the number of training hours from any of the trainers in a particular month, the microservice retrieves this data from the database and returns it to the requester.

1. Implement separate Spring boot Application (Microservice).
2. Application should implement REST endpoint to accept trainer's workload with follow contract:
    1. Request
        1. Trainer Username
        2. Trainer First Name
        3. Trainer Last Name
        4. IsActive
        5. Training date
        6. Training duration
        7. Action Type (ADD/DELETE)
    2. Response
        1. 200 OK
3. Implement service function corresponding to mentioned below REST endpoint. Service should calculate as in-memory saved structure (in memory DB) trainer's monthly summary of the provided trainings. The model should be the follow;
    1. Trainer Username
    2. Trainer First Name
    3. Trainer Last Name
    4. Trainer Status
    5. Years (List)
        1. Months (List)
            1. Training summary duration
4. Update Existing Main Microservice implementation to call Secondary Microservice every time that new training added or deleted to the system.
5. Implement discovery module according to guide Eureka Discovery Service.
6. Use circuit breaker design pattern in your implementation.
7. Implement Authorization − Bearer token for Microservices integration Use JWT token implementation.
8. Two levels of logging should be implemented - transactions and each operation transaction level - which endpoint was called, which request came and the service

response - 200 or error and response message + at this level, a *transactionId* is generated, by which you can track all operations for this transaction the same *transactionId* can later be passed to downstream services.

Notes:

1. For REST API implementation use second level of Richardson maturity model.
2. Try to understand in which case training can be deleted.