Concepts to practice

- Microservices
- REST
- Messaging
- NoSQL
- Mappers
- Dependency Injection
- Unit testing
- Swagger

Tech-stack

- Spring (Data, AMQP, API)
- RabbitMQ
- MongoDB
- Maven

Description

This project will consist of 2 microservices.

- A service/api to respond the requests
- A consumer service that will read the messages published in the RabbitMQ queue, and perform the operations on the database.

Model/Entity

A member of a gym. Should have the following attributes: name, gender, age, height, weight, body fat percent, active membership? and registration date.

MICROSERVICES

1. **Service/API**

   Create three 4 endpoints respecting the following pattern:

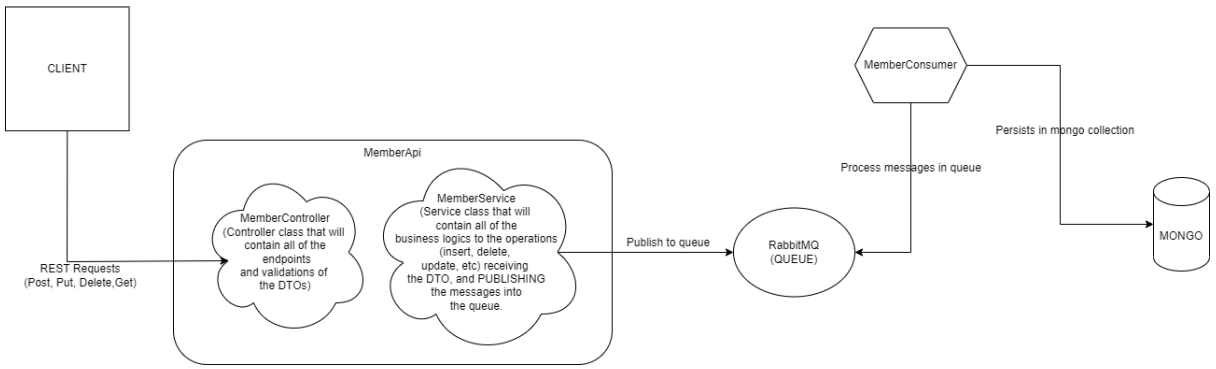
- (GET) Endpoint to list one or all of the members.
- (POSTz) Endpoint to create a new member.
- (PUT) Endpoint to edit an existing member.
- (DELETE) Endpoint to delete an existing member.

All of the endpoints must have their own DTOs (data transfer object) which is different from the model/entity class (having a different class to transport data instead of using the model itself is a good practice).

MANDATORY TO ADD SWAGGER TO PROJECT POM.XML

2. **Consumer**

The consumer must be able to read from the relay and perform the modifications on the database. The mongo connection should be created in (and only in) the consumer. No other service should be able to connect to the database.


CLIENT
REST Requests
(Post, Put, Delete,Get)
MemberApi
MemberController
(Controller class that will
contain all of the
endpoints
and validations of
the DTOs)
MemberService
(Service class that will
contain all of the
business logics to the operations
(insert, delete,
update, etc) receiving
the DTO, and PUBLISHING
the messages into
the queue.
Publish to queue
RabbitMQ
(QUEUE)
MemberConsumer
Process messages in queue
Persists in mongo collection
MONGO