WhatsON

Anton Ziaziulin

1 General idea

Our lives are based on information, especially on filtered qualitative information. The main source for such information is news. As we live in the world full of events we need to be informed well so that we'll be up to date.

According to the upper I came up with a thought to build something news-based. As I know Java EE well the idea landed on creating a web system (or environment) that would mail registered users with news. After thinking a moment I decided to build the app in REST technology since it is flexible, simple and highly extendable. Spring Boot was the first and only technology i came up with to build the system. Moreover the microservice architecture has been used, which is sneaking to developing contexts rapidly. Why haven't I used monolith approach? It would have been too heavy for my system. Due to microservice approach I was able to build the environments partly, like from blocks, integrating fresh service to the whole system.

The main functionality of the system is daily mailing based on user preferences, user registration and settings handling. With the integration of News API provided by google.

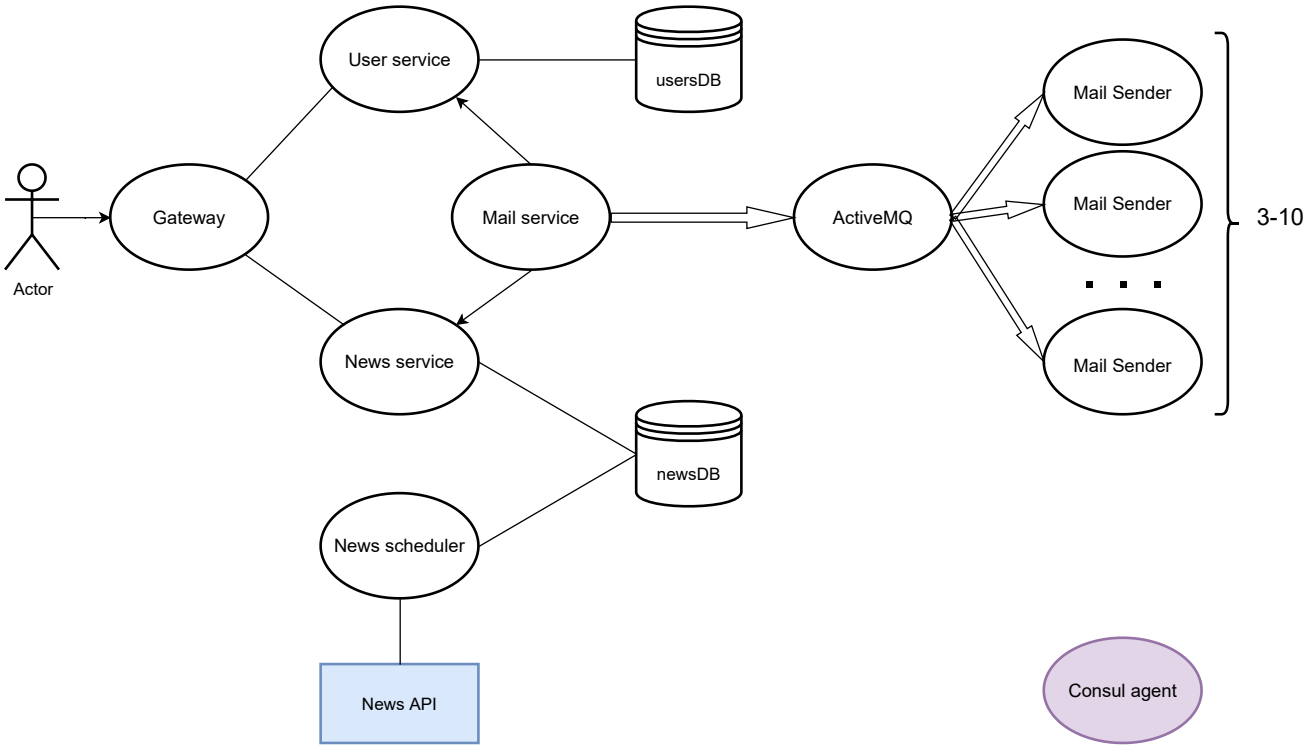
2 Technologies

- Spring Boot Core
- Spring Boot Web
- Spring Data MongoDB
- Spring Boot Validation
- Spring Boot Test
- Spring Boot Actuator
- Spring Boot Devtools
- Spring Boot ActiveMQ
- Spring Boot Mail
- Spring Boot WebFlux
- Spring Cloud Consul
- Spring Cloud Gateway
- Docker
- Consul
- OpenAPI
- Swagger

- TestContainers
- JUnit 5
- Lombok
- MapStruct

2.1 Architecture and design

Below is logical scheme of built system:



First of all, actor (user) is available to communicate with system through API Gateway, which routes http request to user and news services. User services maintain users and communicates with usersDB which is a MongoDB. It provides several endpoints for registering user and changing its settings: mailing enable/disable option, username, name and email. News service is responsible for retrieving data from newsDB. To simplify, it gets articles which are stored in mongoDB instance.

News scheduler is responsible for requesting News API server for getting fresh news. It doesn't have any endpoints. The main goal for it is to get articles and save it into newsDB. This service is scheduled: every hour it makes requests to API server to retrieve news in polish, russian and english languages. Also, for developing aims it deletes older than 2-days long articles from the database every night.

Mail Service is the core of application. It connects both user service and news service to make user-based mail content. The task for requesting both services is also scheduled for every evening. Then the mail content is proceeded to the message broker. In my case it is Apache ActiveMQ.

Mail Senders are concurrent sessions to listen for messages from the queue and then to send the mails directly to users. The number of sessions is between 3 and 10. Of course in production this numbers should be increased.

As my system is built in microservice architecture the problem of finding hosts of services after (re)deploying them was actual. To solve this problem I used Service Discovery mechanism provided by Consul, which is present in the scheme as Consul agent.

2.2 Improvements

Here are some improvements that are possible to made for the system:

1. Add cache support
2. Add module for social media like support (chatting, sharing, friends, etc)
3. Add options for sending news (sms, social media bots, etc)
4. Add Spring security
5. Add Mail verification
6. Add OAuth 2.0 or JWT

2.3 How to start the environment on localhost

Firstly, you need to obtain your api in News API. Then provide it to application.properties file in news-scheduler module. After that install Consul and run it on your localhost. After that run ActiveMQ (the easiest way is to use docker image). Note that I have another ports in properties for message-broker.url in Mail Service and Mail Sender. You may need to change them before start as default port for ActiveMQ are slightly different. Then run two instances (or one for not complicating) of MongoDB. Provide url to it/them in User Service, News Service and News Scheduler. Then via IntelliJ or another IDE simply run all services and observe what's going on!