Movie blog documentation

Andrei Colhon group 252 HPC

Project

The software built serves as an e-blog for movie enthusiasts. Such a person can register via email and see the list of movies. Then, based on the comments left by the user these are prioritized in order to be easy to see movies of interest. Further, the user can see comments left by other users and get real time notifications.

Implementation

The **web server** has been designed in a microservices architecture approach. This approach is service-oriented and it is composed of loosely coupled elements that have bounded contexts. This loosely coupled approach will enable updating services independently. Thus, updating one microservice will not require major changes in the other services. Eventually all services will talk to each other through different APIs. The solution utilizes REST API architecture.

The current solutions part exposes four REST services: customer service (user information), movie service (movies information), comment service (comments for movies), notification service (notifications). They were written in Python, using Flask.

The **web app** consumes the specified REST services and uses microfrontend architecture, based on webpack module federation. Implementation was done in Angular 13 and includes two components: the shell-service (which deals with customer authentication/registration) and movie-service (deals with information about movies and comments).

Integration between the web server and other services is realized through an API gateway, also implemented in Python - Flask.

Containers are used to deploy the solution, through Docker and Docker-Compose. Each service (from the backend) has its own Dockerfile for deployment, while Docker-Compose is used to specify the path to each such docker file and start all the docker containers together.

Microservices patterns

Microservice Pattern

From the beginning on we tried to model the system in a decoupled and service oriented way, so this pattern was ideal for that.

API Gateway Pattern

The API Gateway defines how clients access the services in a microservice architecture and is the single entry point for all clients.
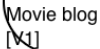
Figure 1. Container diagram (from C4 model) of the entire application/system