Master's Degree in Agile Software Development for the Web

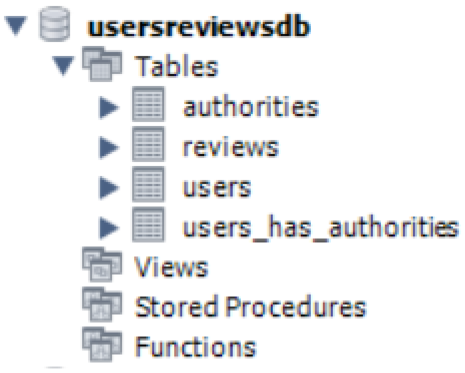
Final Work Report: Frameworks Backend y Microservicios

Extended Movie Management Web Application

Tomasz Sojka

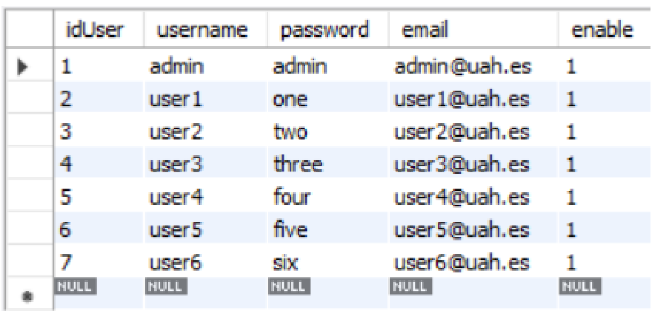
1. Task analysis and database design.

First, the task was analyzed and the database was designed. In compliance to the Principle of Decentralization: Applications based on the microservices architecture should have decentralized data management. Each service should manage its own database (if needed). Following this principle, the SQL file was prepared to generate database schema and tables that are separate from the database prepared for initial work.. There were 4 tables created, one for users, one for reviews of the films, one for user authorities and one to join the users with their authorities, by storing their ids. Then, the SQL file to initiate some data into tables was created. Both files were run in MySQLWorkbench successfully.

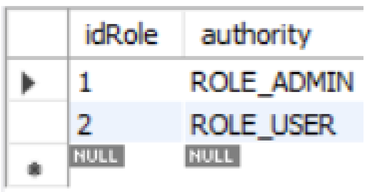


Created database schema.

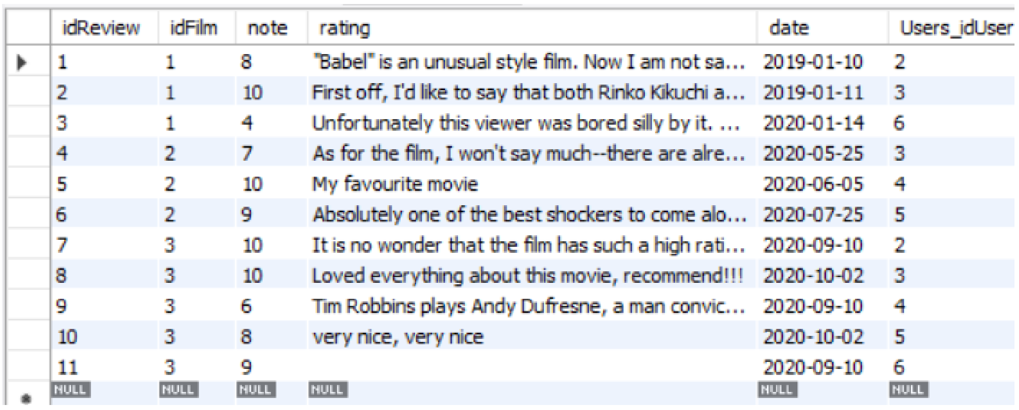
Afterwards, to test if everything worked well and tables contain initial data, query SELECT * was used.



idUser	username	password	email	enable
1	admin	admin	admin@uah.es	1
2	user1	one	user1@uah.es	1
3	user2	two	user2@uah.es	1
4	user3	three	user3@uah.es	1
5	user4	four	user4@uah.es	1
6	user5	five	user5@uah.es	1
7	user6	six	user6@uah.es	1
NULL	NULL	NULL	NULL	NULL



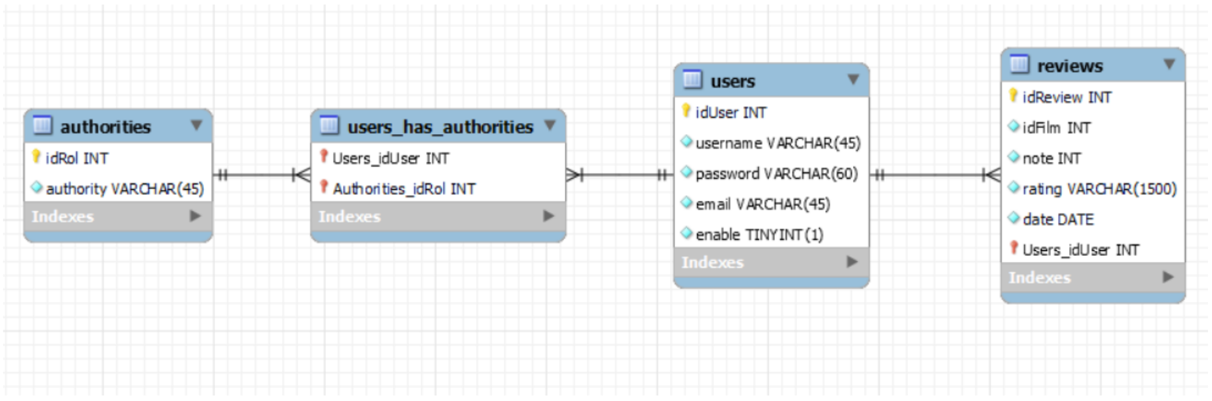
idRole	authority
1	ROLE_ADMIN
2	ROLE_USER
NULL	NULL



idReview	idFilm	note	rating	date	Users_idUser
1	1	8	"Babel" is an unusual style film. Now I am not sa...	2019-01-10	2
2	1	10	First off, I'd like to say that both Rinko Kikuchi a...	2019-01-11	3
3	1	4	Unfortunately this viewer was bored silly by it. ...	2020-01-14	6
4	2	7	As for the film, I won't say much--there are alre...	2020-05-25	3
5	2	10	My favourite movie	2020-06-05	4
6	2	9	Absolutely one of the best shockers to come alo...	2020-07-25	5
7	3	10	It is no wonder that the film has such a high rati...	2020-09-10	2
8	3	10	Loved everything about this movie, recommend!!!	2020-10-02	3
9	3	6	Tim Robbins plays Andy Dufresne, a man convic...	2020-09-10	4
10	3	8	very nice, very nice	2020-10-02	5
11	3	9		2020-09-10	6
NULL	NULL	NULL	NULL	NULL	NULL

All of the initial data selected from the tables. As we can see, everything was inserted properly.

In the end, the EER Model was generated from the created database to visualize the relation between tables.

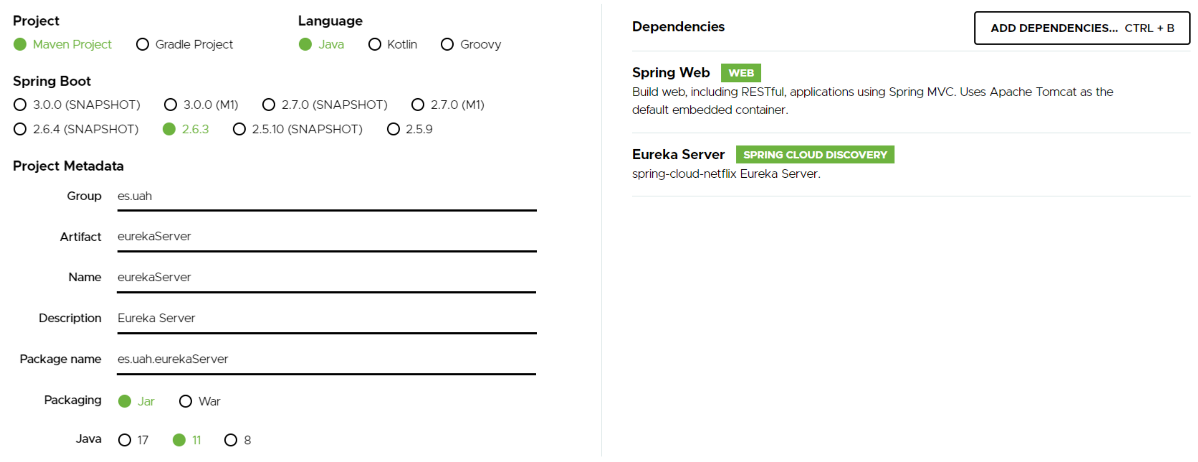


EER Diagram generated from ‘*usersreviewsdb*’ database.

2.Eureka server

Before preparation of backend and client services Eureka server was prepared to be able to provide the registration and location of the microservices of the application.

The base of the project was generated using Spring Initializr:



Project
[✓ Maven Project] Gradle Project

Language
[✓ Java] Kotlin Groovy

Spring Boot
3.0.0 (SNAPSHOT) 3.0.0 (M1) 2.7.0 (SNAPSHOT) 2.7.0 (M1)
2.6.4 (SNAPSHOT) [✓ 2.6.3] 2.5.10 (SNAPSHOT) 2.5.9

Project Metadata

Group	es.uah
Artifact	eurekaServer
Name	eurekaServer
Description	Eureka Server
Package name	es.uah.eurekaServer
Packaging	[✓ Jar] War
Java	17 [✓ 11] 8

Dependencies ADD DEPENDENCIES... CTRL + B

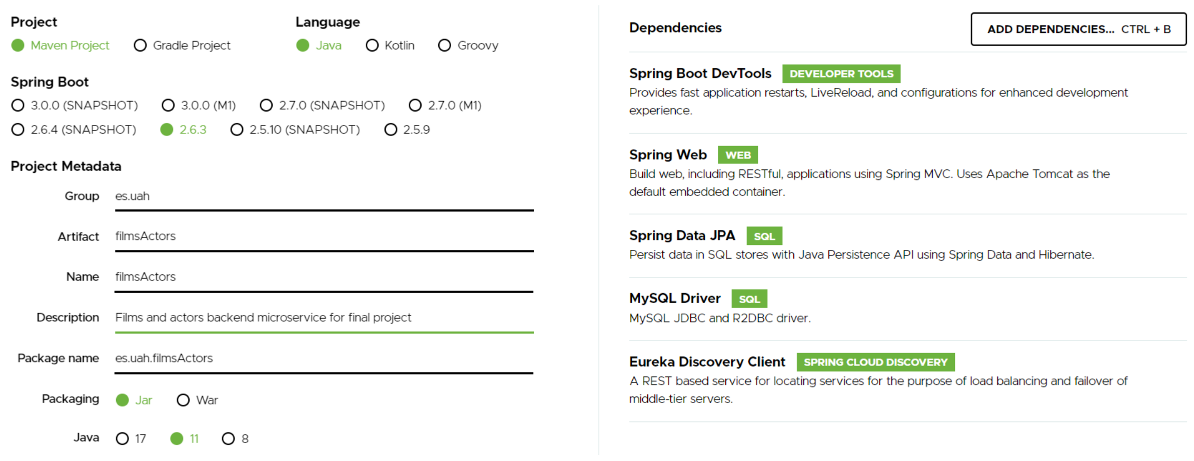
Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Eureka Server SPRING CLOUD DISCOVERY
spring-cloud-netflix Eureka Server.

3. Backend microservices implementation.

Because each microservice must be configured as a Eureka client and send a request requesting registration on that server at the time of boot, previously prepared films and actors' microservice had to be initialized again with the use of Eureka Discovery Client dependency.

The base of the project was generated using Spring Initializr:


Project
[✓ Maven Project] Gradle Project

Language
[✓ Java] Kotlin Groovy

Spring Boot
3.0.0 (SNAPSHOT) 3.0.0 (M1) 2.7.0 (SNAPSHOT) 2.7.0 (M1)
2.6.4 (SNAPSHOT) [✓ 2.6.3] 2.5.10 (SNAPSHOT) 2.5.9

Project Metadata

Group	es.uah
Artifact	filmsActors
Name	filmsActors
Description	Films and actors backend microservice for final project
Package name	es.uah.filmsActors
Packaging	[✓ Jar] War
Java	17 [✓ 11] 8

Dependencies ADD DEPENDENCIES... CTRL + B

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

MySQL Driver SQL
MySQL JDBC and R2DBC driver.

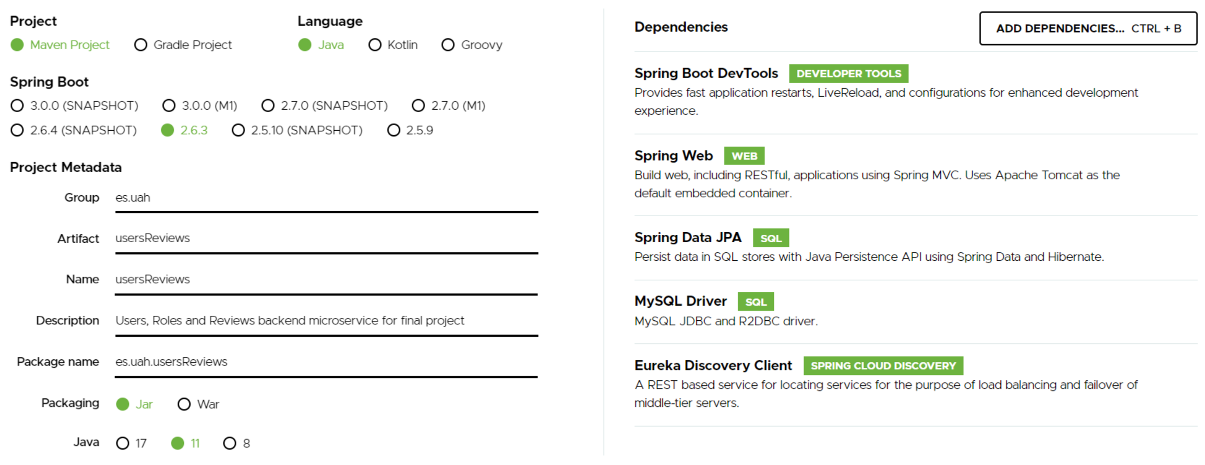
Eureka Discovery Client SPRING CLOUD DISCOVERY
A REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.

As it can be seen above, java 11 is used and the modules that were included are the following:

- Spring Boot DevTools,
- Spring Web,
- Spring Data JPA,
- MySQL Driver
- Eureka Discovery Client.

Code of films and actors' microservice remained the same.

Afterwards, base of the users and reviews' microservice was generated using Spring Initializr, using the same dependencies:


Project
[✓ Maven Project] Gradle Project

Language
[✓ Java] Kotlin Groovy

Spring Boot
3.0.0 (SNAPSHOT) 3.0.0 (M1) 2.7.0 (SNAPSHOT) 2.7.0 (M1)
2.6.4 (SNAPSHOT) [✓ 2.6.3] 2.5.10 (SNAPSHOT) 2.5.9

Project Metadata

Group	es.uah
Artifact	usersReviews
Name	usersReviews
Description	Users, Roles and Reviews backend microservice for final project
Package name	es.uah.usersReviews
Packaging	[✓ Jar] War
Java	17 [✓ 11] 8

Dependencies ADD DEPENDENCIES... CTRL + B

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

MySQL Driver SQL
MySQL JDBC and R2DBC driver.

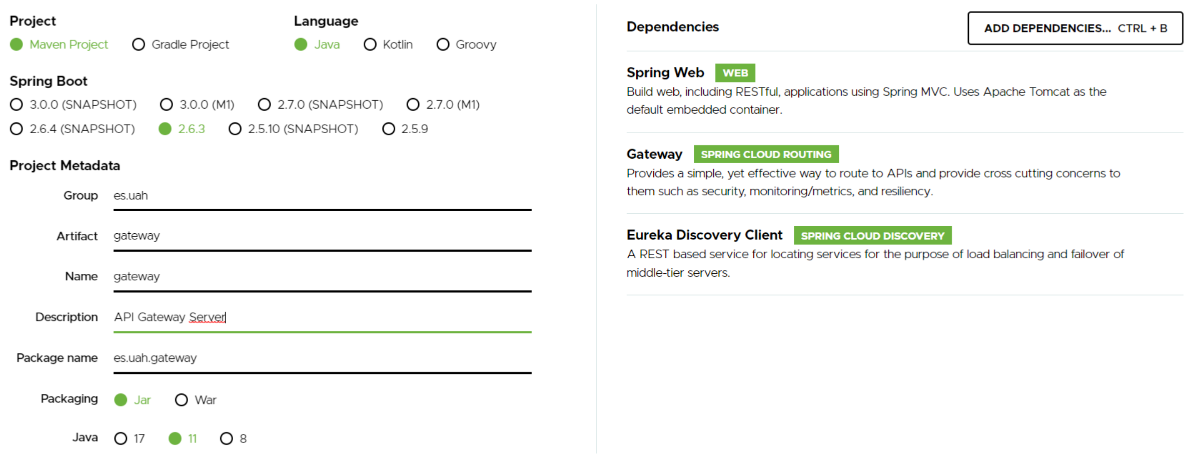
Eureka Discovery Client SPRING CLOUD DISCOVERY
A REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.

Code of the users and reviews' microservice was opened in Intellij Idea IDE, a new prepared usersreviewsdb database was added to the project and assigned to the entity manager factory. Afterwards, JPA Buddy plugin was used to generate entities from the assigned database. One to many relationship was added from User to Review entity and Many to Many relationship between User and Role entities.The code of the generated entities was slightly changed. Afterwards, all of the DAO, Service and Controller classes had to be added and connected. All methods were implemented to provide all the required operations.

4. Gateway

To consume both backend microservices APIs and join them under one port, a project using Spring Cloud Gateway was prepared.

The base of the project was generated using Spring Initializr:



Project
[✓ Maven Project] Gradle Project

Language
[✓ Java] Kotlin Groovy

Spring Boot
3.0.0 (SNAPSHOT) 3.0.0 (M1) 2.7.0 (SNAPSHOT) 2.7.0 (M1)
2.6.4 (SNAPSHOT) [✓ 2.6.3] 2.5.10 (SNAPSHOT) 2.5.9

Project Metadata

Group	es.uah
Artifact	gateway
Name	gateway
Description	API Gateway Server
Package name	es.uah.gateway
Packaging	[✓ Jar] War
Java	17 [✓ 11] 8

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Gateway SPRING CLOUD ROUTING
Provides a simple, yet effective way to route to APIs and provide cross cutting concerns to them such as security, monitoring/metrics, and resiliency.

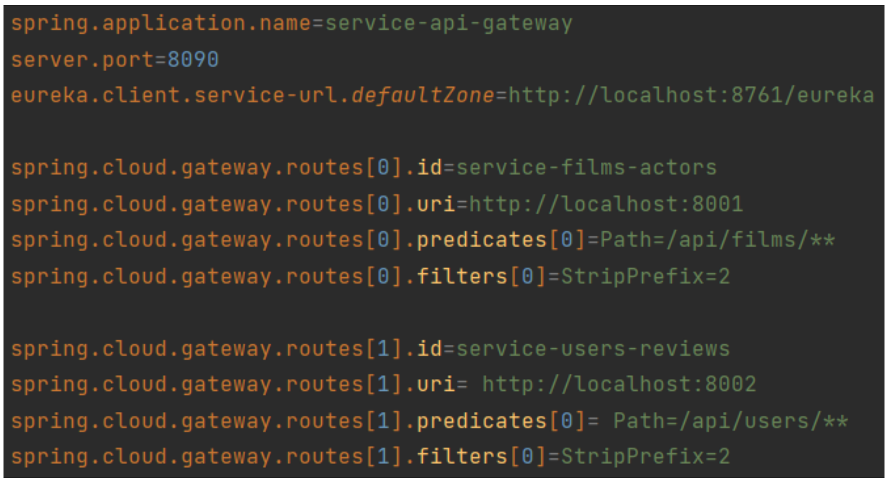
Eureka Discovery Client SPRING CLOUD DISCOVERY
A REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.

As it can be seen above, java 11 is used and the modules that were included are the following:

- Spring Web,
- Gateway,
- Eureka Discovery Client.

Project is also using Eureka Discovery Client dependency to be tracked by Eureka Server.

Routing of the APIs was prepared in the properties of the application:



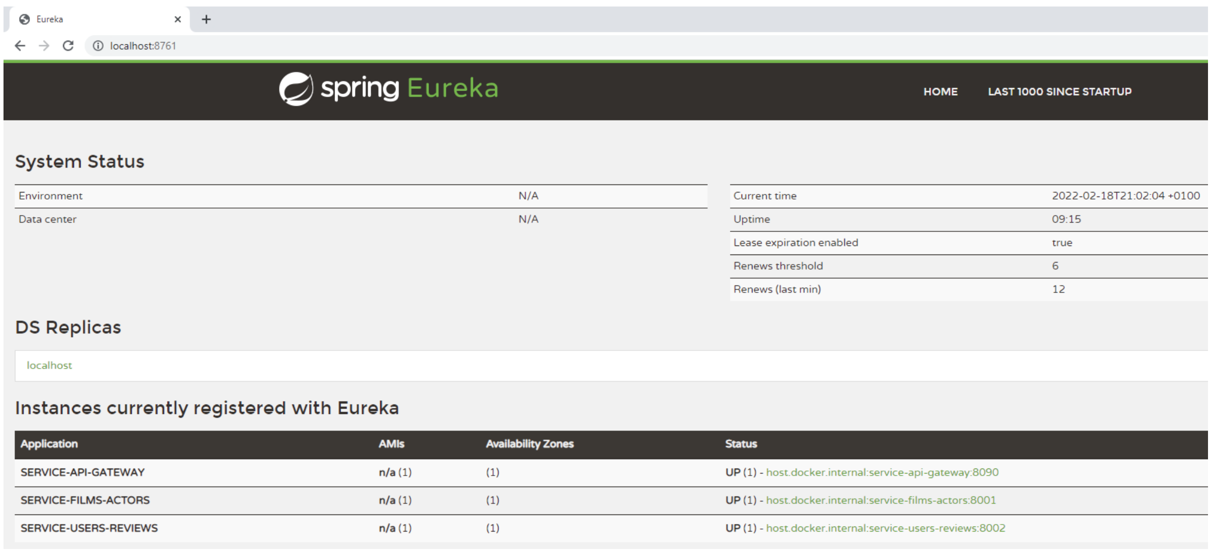
```
spring.application.name=service-api-gateway
server.port=8090
eureka.client.service-url.defaultZone=http://localhost:8761/eureka

spring.cloud.gateway.routes[0].id=service-films-actors
spring.cloud.gateway.routes[0].uri=http://localhost:8001
spring.cloud.gateway.routes[0].predicates[0]=Path=/api/films/**
spring.cloud.gateway.routes[0].filters[0]=StripPrefix=2

spring.cloud.gateway.routes[1].id=service-users-reviews
spring.cloud.gateway.routes[1].uri= http://localhost:8002
spring.cloud.gateway.routes[1].predicates[0]= Path=/api/users/**
spring.cloud.gateway.routes[1].filters[0]=StripPrefix=2
```

As it can be seen above, both microservices will be available under port 8090, with different paths.

Later it was checked if the service is visible in Eureka Server:


Eureka

localhost:8761

spring Eureka

HOME LAST 1000 SINCE STARTUP

System Status

Environment	N/A
Data center	N/A

Current time	2022-02-18T21:02:04 +0100
Uptime	09:15
Lease expiration enabled	true
Renews threshold	6
Renews (last min)	12

DS Replicas

localhost

Instances currently registered with Eureka

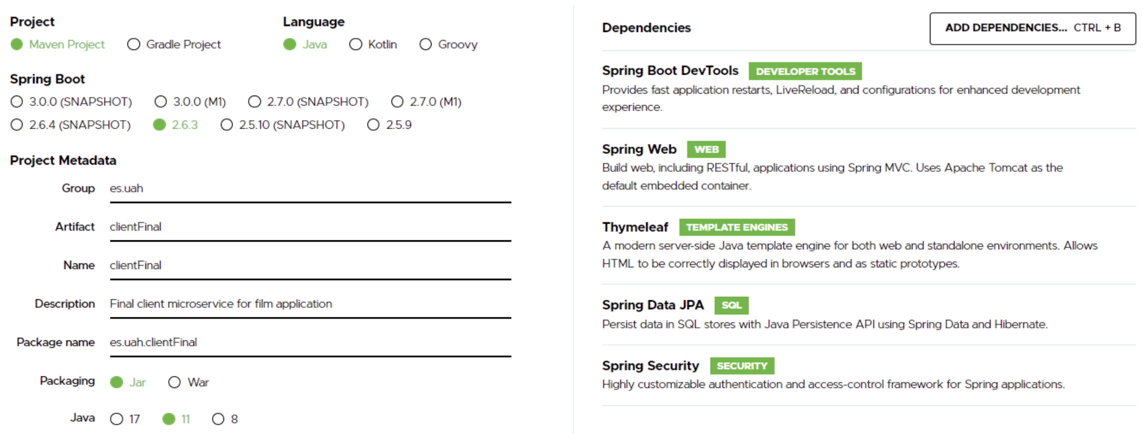
Application	AMIs	Availability Zones	Status
SERVICE-API-GATEWAY	n/a (1)	(1)	UP (1) - host.docker.internal:service-api-gateway:8090
SERVICE-FILMS-ACTORS	n/a (1)	(1)	UP (1) - host.docker.internal:service-films-actors:8001
SERVICE-USERS-REVIEWS	n/a (1)	(1)	UP (1) - host.docker.internal:service-users-reviews:8002

Part of the information about available services. All 3 services are tracked.

All implemented operations were tested by browser and POSTMAN application.

5. Client microservice implementation.

Because the final client microservice will use Spring Security features, client microservice used in initial project had to be had to be initialized again:


Project: [✓ Maven Project] Gradle Project

Language: [✓ Java] Kotlin Groovy

Spring Boot: 3.0.0 (SNAPSHOT) 3.0.0 (M1) 2.7.0 (SNAPSHOT) 2.7.0 (M1) 2.6.4 (SNAPSHOT) [✓ 2.6.3] 2.5.10 (SNAPSHOT) 2.5.9

Project Metadata

Group	es.uah
Artifact	clientFinal
Name	clientFinal
Description	Final client microservice for film application
Package name	es.uah.clientFinal
Packaging	[✓ Jar] War
Java	17 [✓ 11] 8

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Thymeleaf TEMPLATE ENGINES
A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

Spring Security SECURITY
Highly customizable authentication and access-control framework for Spring applications.

As it can be seen above, java 11 is used and the modules that were included are the following:

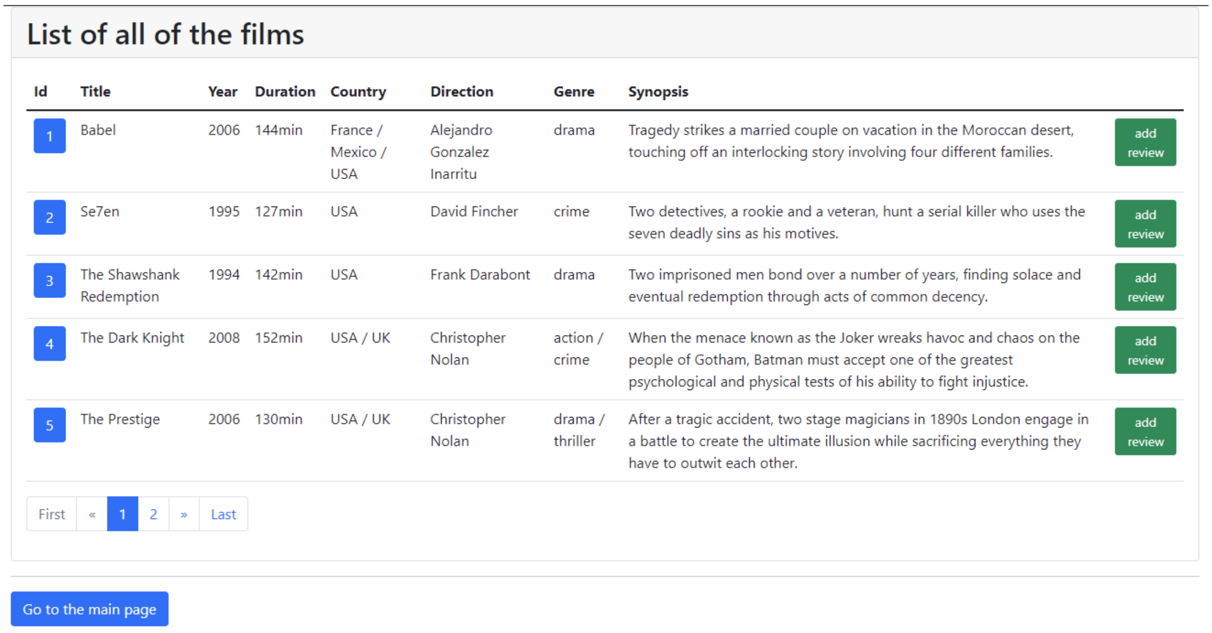
- Spring Boot DevTools,
- Spring Web,
- Thymeleaf,
- Spring Data JPA,
- Spring Security.

The part of the code that was already implemented for the initial project had to be slightly changed. Namely, the process of film removal had to be extended. Even though the '*filmsactorsdb'* and '*usersreviewsdb'* databases are not connected by foreign keys, reviews are clearly connected to the films. When a film is deleted, all reviews about it are also deleted from the database.
Moreover, the films controller class was extended by adding the endpoint for adding a review to a film. The option to add a review by user is available when the list of films is shown. That is why it was

easier to implement an endpoint in the films controller class. The rest of the films and actors remained unchanged from the initial version.

All the required operations on reviews and users were implemented.

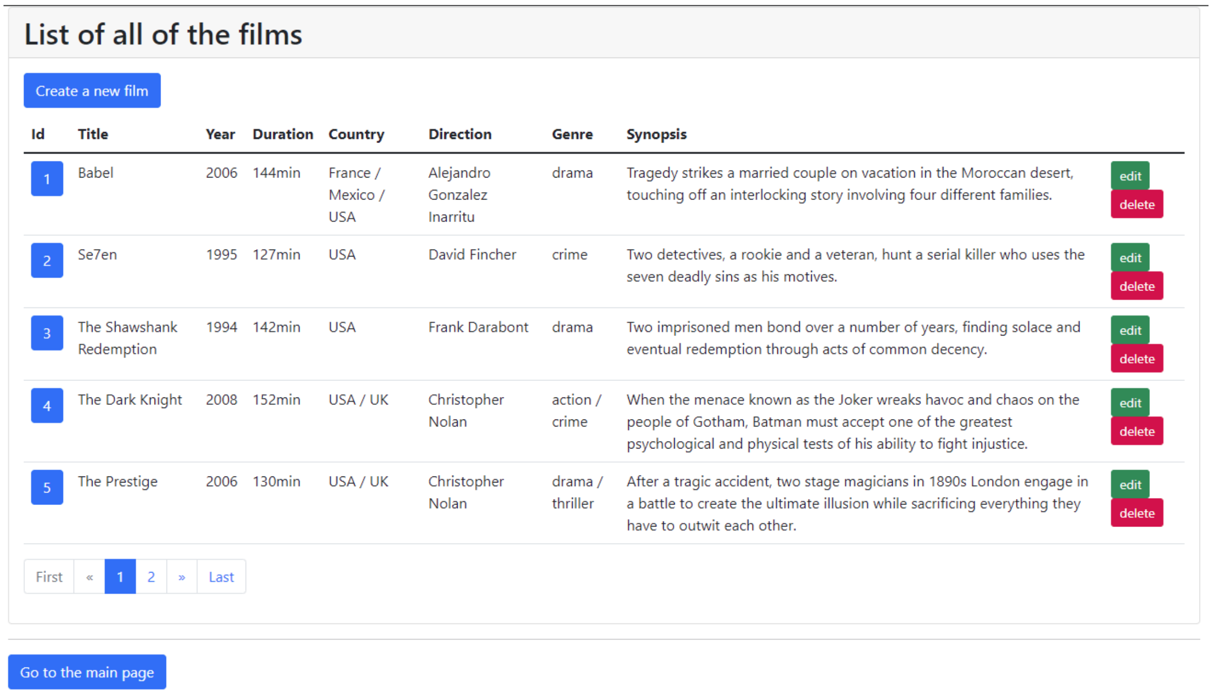
It was required that the application will have two different types of users: administrator and user. To divide their functionalities and authenticate who is using the application, additional classes were implemented. One to provide a custom authentication, one to configure which pages are available for a guest and one was a login controller. To show a proper content, depending on a user type, on the pages available for logged in users, thymeleaf extra spring-security was used in html templates.



List of all of the films

Id	Title	Year	Duration	Country	Direction	Genre	Synopsis	
1	Babel	2006	144min	France / Mexico / USA	Alejandro Gonzalez Inarritu	drama	Tragedy strikes a married couple on vacation in the Moroccan desert, touching off an interlocking story involving four different families.	add review
2	Se7en	1995	127min	USA	David Fincher	crime	Two detectives, a rookie and a veteran, hunt a serial killer who uses the seven deadly sins as his motives.	add review
3	The Shawshank Redemption	1994	142min	USA	Frank Darabont	drama	Two imprisoned men bond over a number of years, finding solace and eventual redemption through acts of common decency.	add review
4	The Dark Knight	2008	152min	USA / UK	Christopher Nolan	action / crime	When the menace known as the Joker wreaks havoc and chaos on the people of Gotham, Batman must accept one of the greatest psychological and physical tests of his ability to fight injustice.	add review
5	The Prestige	2006	130min	USA / UK	Christopher Nolan	drama / thriller	After a tragic accident, two stage magicians in 1890s London engage in a battle to create the ultimate illusion while sacrificing everything they have to outwit each other.	add review

List of all of the films - the view for a logged in user with role 'ROLE_USER'.



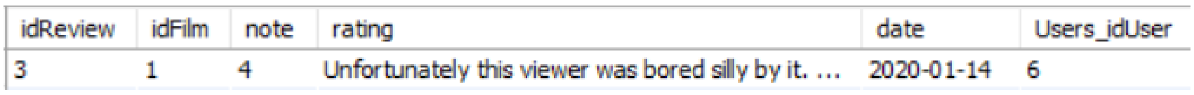
List of all of the films

Id	Title	Year	Duration	Country	Direction	Genre	Synopsis	
1	Babel	2006	144min	France / Mexico / USA	Alejandro Gonzalez Inarritu	drama	Tragedy strikes a married couple on vacation in the Moroccan desert, touching off an interlocking story involving four different families.	edit delete
2	Se7en	1995	127min	USA	David Fincher	crime	Two detectives, a rookie and a veteran, hunt a serial killer who uses the seven deadly sins as his motives.	edit delete
3	The Shawshank Redemption	1994	142min	USA	Frank Darabont	drama	Two imprisoned men bond over a number of years, finding solace and eventual redemption through acts of common decency.	edit delete
4	The Dark Knight	2008	152min	USA / UK	Christopher Nolan	action / crime	When the menace known as the Joker wreaks havoc and chaos on the people of Gotham, Batman must accept one of the greatest psychological and physical tests of his ability to fight injustice.	edit delete
5	The Prestige	2006	130min	USA / UK	Christopher Nolan	drama / thriller	After a tragic accident, two stage magicians in 1890s London engage in a battle to create the ultimate illusion while sacrificing everything they have to outwit each other.	edit delete

List of all of the films - the view for a logged in user with role 'ROLE_ADMIN'.

As can be seen, the view of the same page is different for different types of users. Administrators can Create a new film or delete or edit existing films. Users can add a review of the film. Both can list films and view the details of the movie.

The hardest to implement was the process of adding (more the part of saving) a new review by user. It is because reviews are connected both with users and films. Id of the film had to be saved from the table before going forward to the form page. Then after submitting the form user identity (the one who submitted it) had to be retrieved from SecurityContextHolder. Because SecurityContextHolder stores user email and the id of the user is what is needed, the user had to be searched in the database first and his id had to be returned. After all, the review can be saved with the rating and note passed in the form, today's date and id of the film and id of the user.



idReview	idFilm	note	rating	date	Users_idUser
3	1	4	Unfortunately this viewer was bored silly by it. ...	2020-01-14	6

Example review record in the database.

After all, a basic guest page was added, where listed movies can be seen in a more graphical way.

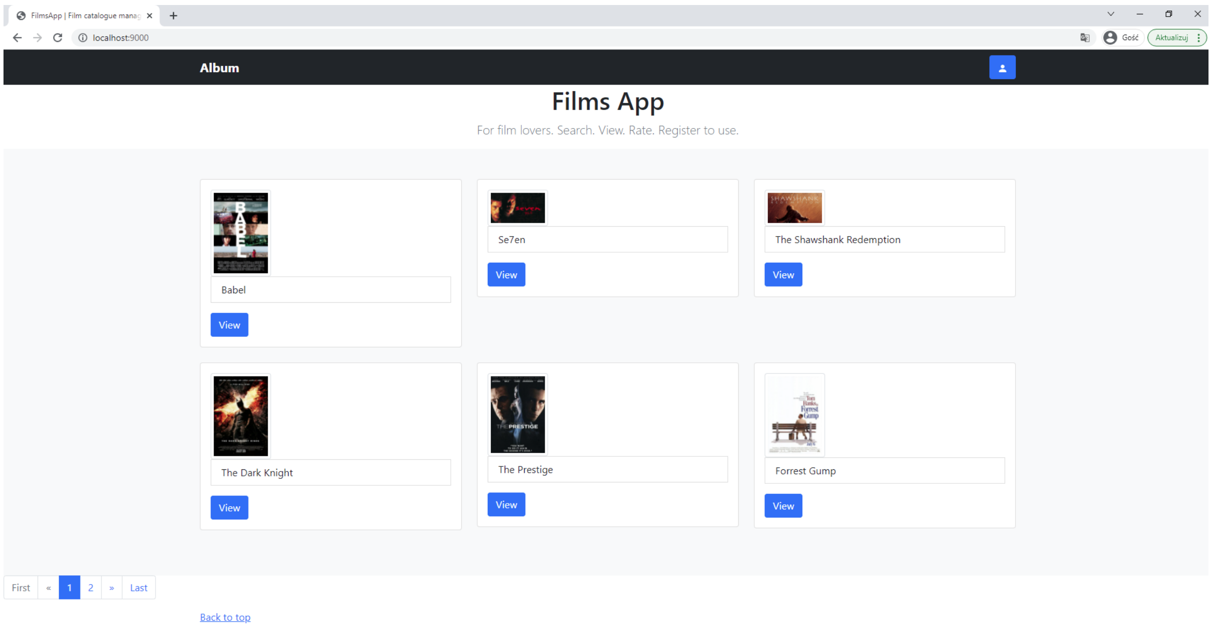
All the requirements were met – all the needed functionalities were implemented.

6. Manual

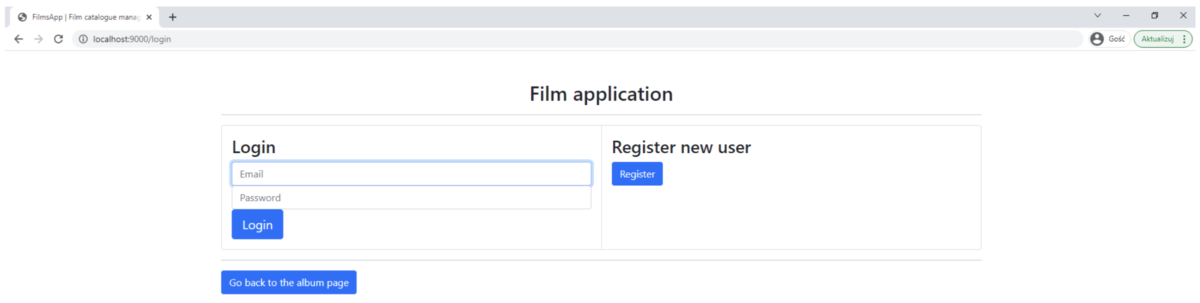
To use the application, first start Eureka Server, then both of the backend JPA microservices (filmsActors and usersReviews), then Gateway and the client microservice in the end. When all are started go to:

http://localhost:9000/

You will see the guest page, available for everybody.

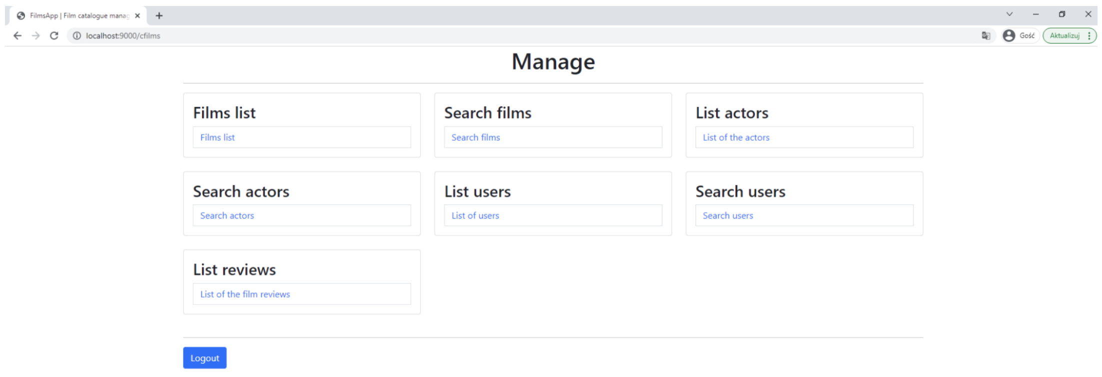


From the guest page list of all the films can be seen. Each film has a button to see the details, however you need to be logged into the user account first. Both “View” buttons and the button on the right upper corner with a person icon will redirect you to the login page.

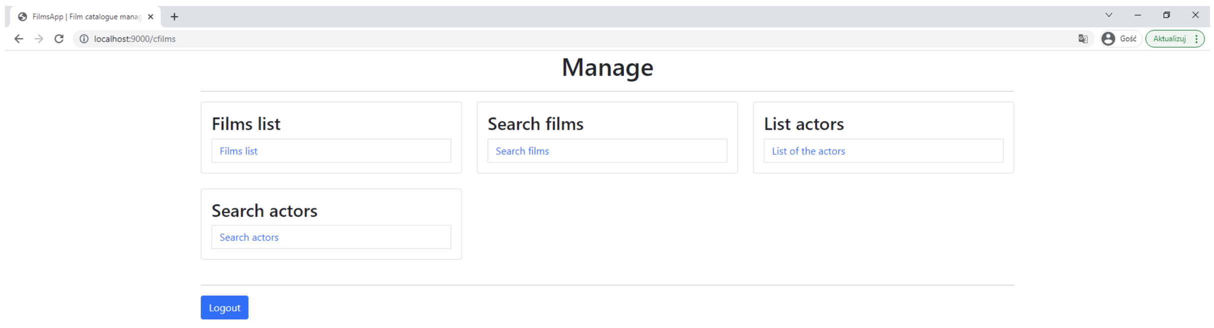


Login page, where users can login to an existing account or register a new one. Registered users are of the “ROLE_USER” authority. Admin accounts can be created only by another admin.

After logging in the film management page is shown. The content depends on the user role.



The film management page of admin.



The film management page of user.