



Разработчик на Spring Framework

Меня хорошо видно && слышно?



Защита проекта

Тема: Онлайн кинотеатр



Максим Савенко

Самозанятый разработчик :)



План защиты



Цели проекта

The diagram shows a vertical list of six blue rounded rectangular boxes. A dashed line on the left side of the boxes connects them in a continuous loop, starting from the top box, going down to the bottom box, and then curving back up to the top box.

Что планировалось

Используемые технологии

Что получилось

Схемы/архитектура

Выводы



Цели проекта

1. “Поставить точку” в обучении
 2. Использование технологий Spring Framework на практике
 3. Демонстрация полученных знаний
-

Что планировалось

1. Сайт просмотра видеоконтента (встраиваемые коды видеохостингов - youtube, vimeo и т.д)
2. Доступна фильтрация по категориям
3. Регистрация пользователей
4. Зарегистрированные зрители могут ставить оценки
5. Панель управления контентом (админка)

Используемые технологии

Бэкенд

Spring Data (JPA)

Postgresql

Spring Web (REST)

Spring Security (JWT токен)

Resilience4j

Amazon SDK (S3)

Spring Actuator

Micrometer Prometheus

Eureka

Spring Gateway

Spring Cache

Фронт

VueJS

Дополнительно

Liquibase

MapStruct

JJWT

Jsoup

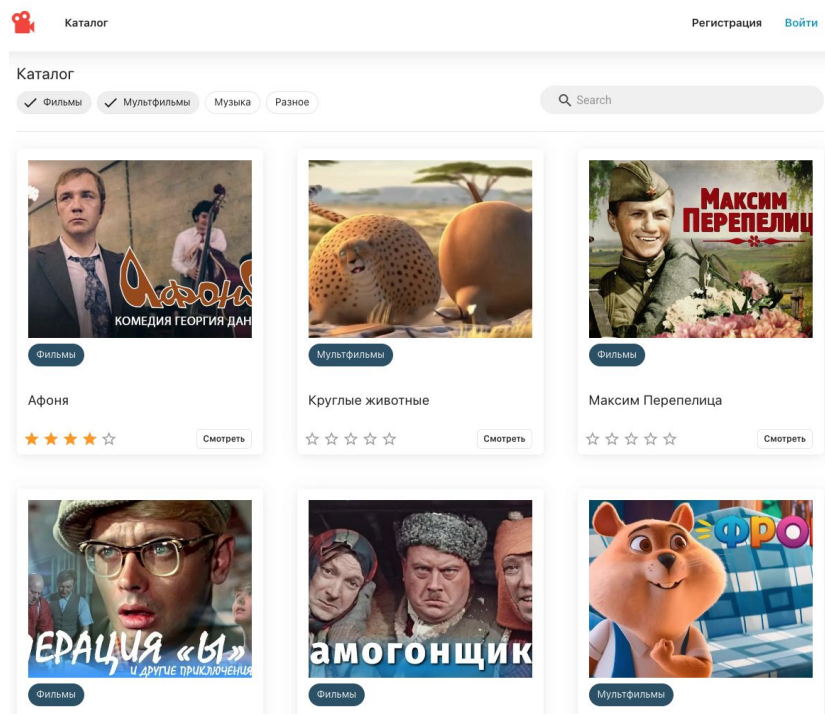
Springdoc Open API

Что получилось?

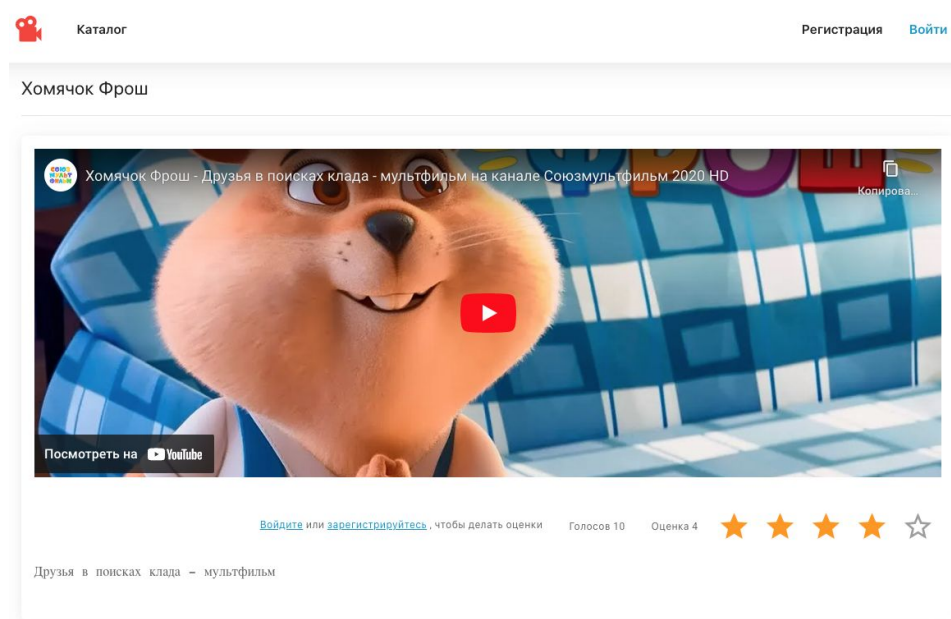


Главные страницы (демо)

1. Каталог



2. Просмотр контента / оценка контента



Админка - управление контентом (демо)

The image displays three overlapping screenshots of the 'MOVIES ONLINE' admin interface, version 0.0.1-SNAPSHOT. Each screenshot shows a sidebar with navigation links: 'На сайт', 'Видеоролики', 'Категории', and 'Пользователи'.

Скриншот 1: Управление пользователями
This screen shows a table for managing users. The table has columns for 'Id' and 'Пользователь'. Two users are listed with IDs 1 and 2.

Id	Пользователь
1	
2	

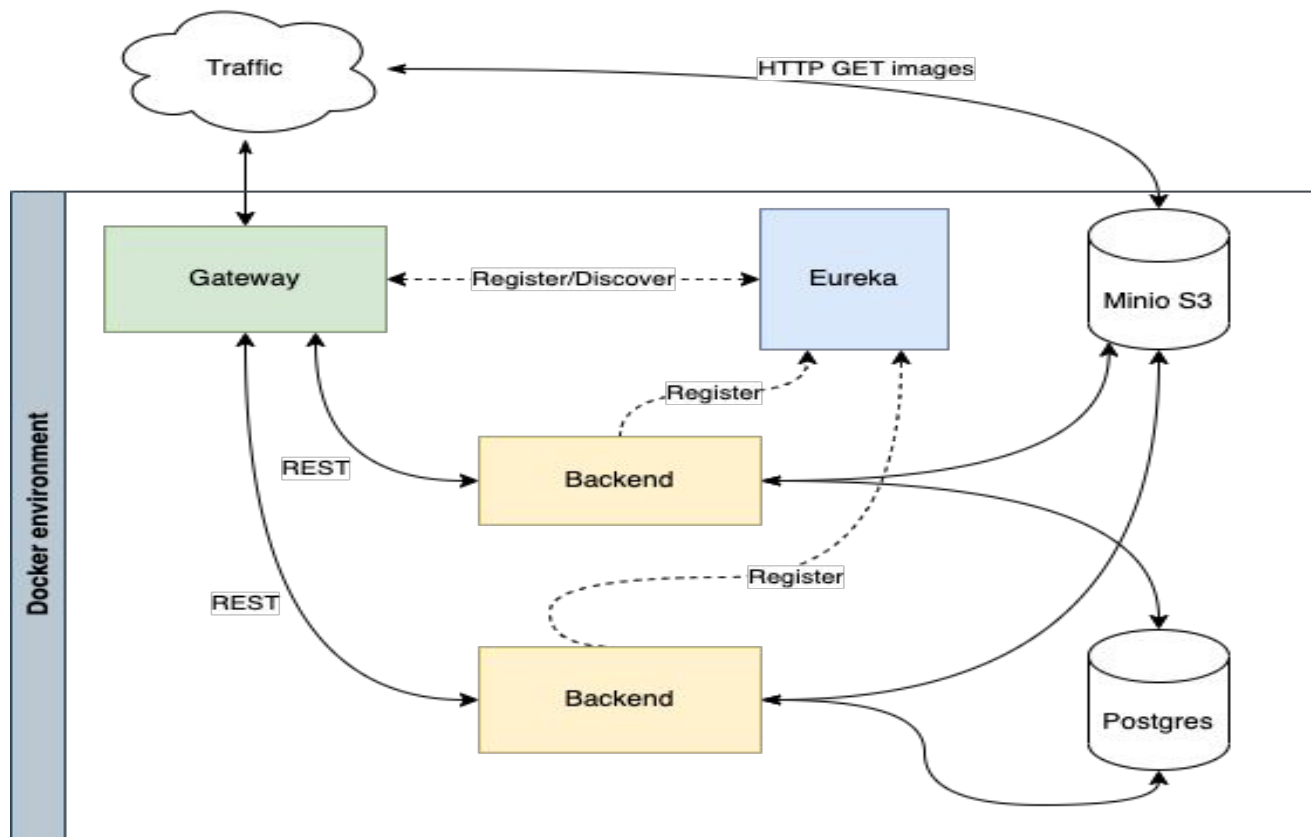
Скриншот 2: Управление категориями
This screen shows a table for managing categories. The table has columns for 'Id' and 'Категория'. Two categories are listed with IDs 1 and 2.

Id	Категория
1	
2	

Скриншот 3: Управление видеороликами
This screen shows a table for managing videos. The table has columns for 'Id', 'Название', 'Категория', 'Рейтинг', and 'Действия'. Four videos are listed.

Id	Название	Категория	Рейтинг	Действия
1	Афоня	Фильмы	★★★★☆	✎ 🗑
2	Самогонщики	Фильмы	★★★★☆	✎ 🗑
3	Операция «Ы» и другие приключения Шурика	Фильмы	★★★★☆	✎ 🗑
4	Хомячок Фрош	Мультфильмы	★★★★☆	✎ 🗑

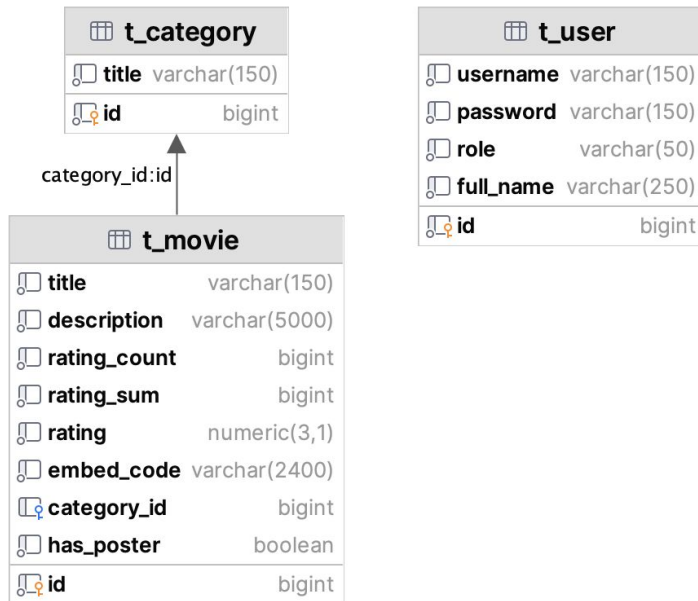
Компонентная схема



База данных



Всего три таблицы



Open API - документируем REST

Swagger SMARTBEAR /backend/api-docs Explore

OpenAPI definition v0 OAS3
/backend/api-docs

Servers

user-controller

- GET /api/v1/user/{id}
- PUT /api/v1/user/{id}
- DELETE /api/v1/user/{id}
- GET /api/v1/user
- POST /api/v1/user

profile-controller

- GET /api/v1/profile
- PUT /api/v1/profile
- PUT /api/v1/profile/password

movie-controller

- GET /api/v1/movie/{id}
- PUT /api/v1/movie/{id}
- DELETE /api/v1/movie/{id}
- GET /api/v1/movie
- POST /api/v1/movie

localhost:19900/backend/api-docs

```
{
  "openapi": "3.0.1",
  "info": {
    "title": "OpenAPI definition",
    "version": "v0"
  },
  "servers": [
    {
      "url": "http://localhost:19900/backend",
      "description": "Generated server url"
    }
  ],
  "paths": {
    "/api/v1/user/{id}": {
      "get": {
        "tags": [
          "user-controller"
        ],
        "operationId": "getUser",
        "parameters": [
          {
            "name": "id",
            "in": "path",
            "required": true,
            "schema": {
              "type": "integer",
              "format": "int64"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "OK",
            "content": {
              "*/*": {
                "schema": {
                  "$ref": "#/components/schemas/UserDto"
                }
              }
            }
          }
        }
      }
    }
  }
}
```

Подготовка к запуск в облаке



Сборка с помощью GoogleContainerTools / JIB

Все компоненты “заворачиваются” в докер образы

Конфигурирование через переменные окружения

Мониторинг (demo)

NETFLIX
EUREKA

Instances currently registered with Eureka

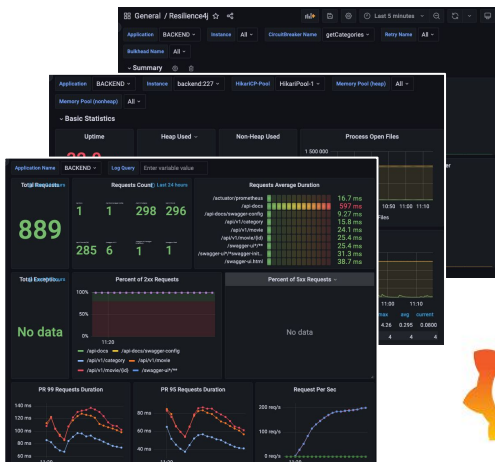
Application	AMIs	Availability Zones	Status
BACKEND	n/a (2)	(2)	UP (2) - backend:502 , backend:227
GATEWAY	n/a (1)	(1)	UP (1) - gateway:555



Prometheus

eureka-docker (3/3 up) [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration
http://172.18.0.2:9900/actor/prometheus	UP	<code>application="GATEWAY"</code> <code>instance="gateway:555"</code> <code>job="eureka-docker"</code>	1.880s ago	9.669ms
http://172.18.0.4:9905/actor/prometheus	UP	<code>application="BACKEND"</code> <code>instance="backend:502"</code> <code>job="eureka-docker"</code>	1.56s ago	12.368ms
http://172.18.0.3:9905/actor/prometheus	UP	<code>application="BACKEND"</code> <code>instance="backend:227"</code> <code>job="eureka-docker"</code>	3.243s ago	18.762ms



Grafana



Где это все можно взять/посмотреть?

<https://github.com/maxixcom/2022-08-otus-spring-savenko-project>



Выводы и планы по развитию



1. Задуманное реализовано
2. Дальнейшего развития не предвидится

Что можно было бы сделать лучше, если бы было больше времени?



1. Не реализовано управление JWT токенами - обновление, отмена (SSO Keycloak?)
2. Автоматическое получение и сохранение картинок из embed кода.

Спасибо за внимание!

Инструкции для работы с презентацией