Booktion

2025-06-08

Petri Närhi (SYS)

Adam Areberg (IA)

Ingrid Merz (SYS)

Malmö Universitet - Webbtjänster

Om tjänsten

Vi har byggt en tjänst som kombinerar filmer och böcker för att se om det finns en tillhörande film till en bok eller vice versa. Det innehåller en sökfunktion. När användaren tagit fram rätt film kan användaren rösta upp filmen eller boken, röstningen visar om boken eller filmen är mer värd att läsa/se. Det finns även ett leaderboard där man kan se nuvarande topp 5 uppröstade filmer och böcker.

Den tekniska lösningen

Vi har använt Java för backenden, då 2 av 3 av oss är systemutvecklarstudenter och har det som förstaspråk. För frontenden har vi HTML, CSS och JavaScript. Ramverket vi använt är SpringBoot för REST-baserade tekniker. Vi laddade ner ett Maven-projekt från springboot.io där vi inkluderade dependencies för att utveckla ett RESTful API. PostgreSQL användes för tjänstens databas.

Datakällorna är 2 externa API:er:

- omdb - API från IMDB
- Google Books API

Dessa används för att få fram film- respektive bokresultat när man söker, och vår tjänst kombinerar dessa i samma sökresultat.

Vi behandlar datakällorna genom att ge ett sökresultat med den bästa matchningen av film och bok, så att rösterna läggs till i databasen endast ifall ett acceptabelt sökresultat hittas. Sökresultaten presenteras snyggt med tillhörande poster och beskrivning i tydliga sektioner.

Användarmanual

Vid behov, ladda ner SpringBoot, Java, Maven. API-nycklarna finns tillagda i application.properties i zip:en men inte i repositoryt.

1. Klona projektet: git clone https://github.com/PetriNarhi/Booktion
2. Gå till src/main/java/com/api/controller/BooktionApplication.java
3. Bygg och starta servern: mvn spring-boot:run eller klicka på Run i IntelliJ.
4. Öppna webbläsaren och gå till localhost:8081

API-dokumentation

Vi har valt att bygga en REST API på våra separata endpoints för (/search), röstning (/vote), (/leaderboard/), samt hämtning av röster för specifika titlar (/votes, /votes/total). Det här ger en tydlig separation av ansvar i vår API, enkel utbyggnad och följer etablerade webbstandarder. Vi har ett par olika typer av endpoints vilket är ett krav för att en API ska vara RESTful. Vi har designat det så här för tydlighets skull.

Dokumentation

OBS: Man måste starta applikationen för att kunna besöka:
http://localhost:8081/swagger-ui/index.html