Inlämning: Filmstudion

Robin Karlsson (WU21)

Datum: 2022-02-11

Github repo:
https://github.com/niborium/Filmstudion

REST

Jag använder API-kontroller som webb-API-endpoint. En kontroller är en klass för att ta emot och svara på en API-endpoint. Jag använder olika HTTP-verb som GET, POST, PUT, Patch etc. Jag använder också Dependency Injection Method för att koppla ihop mina tjänster med tjänstegränssnitt. Gränssnitt används i mina kontroller, så min kod är mer testbar och ren. Jag använder repository pattern och Entity framework core "in memory Database".

När ett RESTful API anropas kommer servern att överföra till klienten en representation av tillståndet för den begärda resursen.

Vad servern gör när du, klienten, anropar en av dess API:er beror på två saker som du behöver tillhandahålla servern:
1) En identifierare för resursen du är intresserad av. Detta är URL:en för resursen, även känd som endpoint. Faktum är att URL står för Uniform Resource Locator.
2) Operationen du vill att servern ska utföra på den resursen, i form av en HTTP-metod eller verb. De vanliga HTTP-metoderna är GET, POST, PUT och DELETE.

Representationen kan vara i ett JSON-format, och förmodligen är detta verkligen fallet för de flesta API:er. Det kan också vara i XML- eller HTML-format.

Har 3 kontrollers för att tillgodose behovet (FilmsController, FilmStudiosController och UsersController) och endpointsen härleds till respektive controller för enkel navigation/struktur.

Implementation

"User"-modellen jag använder ändrades när den användes i webb-API. Web API svarar aldrig på några lösenord eller lösenordsrelaterad information när användarrelaterad data hämtas. Förutom ibland, reagerar webb-API annorlunda när man hämtar data baserat på autentiseringar. Här är några exempel:

API Controller Namn	Metod	URL	Förklaring
Films	Get	Api/films/	Får en lista av alla filmer (för alla) - För autentiserade användare Med listan över tillgängliga kopior av viss film För autentiserade användare med listan över tillgängliga kopior av viss film.
Films	Get/{id}	Api/films/{id}	Hämtar en film efter dess id (för alla). -För autentiserade användare med listan över tillgängliga exemplar av denna film

Finns även andra exempel som (/api/filmstudios) där användare eller en autentiserad filmtstudio kan hämta alla filmstudios samt enskild filmstudio men som inte innehåller listan RentedFilmCopies eller egenskaper City. Samt vid Krav 4 (”*ENDAST innehåller egenskaperna Username, Role och UserId”*) och Krav 5 (”*förutom egenskapet ’password’ returneras”*) så gömmer vi den känsliga informationen som returneras.

Säkerhet

Jag har använt JWT-tokensystemet för att ”bevisa” anroparen av API-endpoint. JWT-bearer token används i autentiseringshuvudet så att jag kan verifiera användaren och användarnivån. Sedan ger webb-API information till en anropare.

I webb-API-kontrollern använder vi attributet [Authorize], vilket betyder att vi måste tillhandahålla en JWT-token för att verifiera användaren. När användare autentiserar sig med användar-id och lösenord ger jag dem en genererad JWT-token. Jag har skapat en tjänst (i startup.cs) som dekrypterar och verifierar JWT-tokenet. När du anropar en api-endpoint med den autentiseringsbearertoken, dekrypterar och verifierar och anropar API-endpointen, då är informationen tillgänglig för användaren.

Inloggningen (klientgränssnittet) består av ett formulär som läsar in och skickar e-post och lösenord (request body) till /api/Users/authenticate. Responsen tas emot och sparar Bearer token i min jwtholder och även user.id i id (app). När utloggning sker så rensas dessa.