Filmstudio

Milica Blagojevic

REST

Jag har använt flera klasser och interface för att definiera och strukturera mina resurser.

- IFilm - representerar en film och innehåller förutom beskrivande egenskaper en lista av FilmCopy.
- IFilmStudio - representerar en filmstudio och ärver från IUser för att koppla filmstudion till en användarroll. För att kunna hantera uthyrning av filmer har jag kopplat FilmStudio till många filmkopior.
- IFilmCopy – med hjälp av egenskapen IsRented håller jag koll på om den enstaka filmkopian är utlånad. Den har relation både till film och till filmstudion. Den har även TimeWhenRented som sätts när filmen lånas och hjälper mig att ha koll på vilken studio det är som lånade filmen först. Jag ville ha koll på detta för att lösa problemet som uppstår när admin minskar på antal filmkopior och det drabbar även de utlånade kopiorna. Då försvinner kopian som är hos studion som har haft den längst.

Jag har använt de REST åtkomstpunkter som var listade i uppgiftskraven. Där det var ett krav att endast till exempel admin skulle få åtkomst har jag använt Authorize Role admin ovanför min Controller och då krävs det att man i anropet skickar token som tillhör den inloggade admin. Om det är så att enligt uppgiftskraven alla kan få åtkomst men olika information beroende på rollen har jag använt User.IsInRole för att hantera de olika scenarion. Om det är så att uppgiften kräver att det är just den inloggade studion som ska få utökad information exempelvis när man ska hämta en specifik filmstudio har jag använt User.FindFirst(ClaimTypes.NameIdentifier)?.Value för att kontrollera id.

Implementation

De interna modellerna används för att organisera data på ett sätt som gör det mögligt att utföra funktionalitet som behövs. De synliga resurserna är de objekt som skickas till klienten -mina DTO klasser.

Jag har till exempel en intern class FilmStudio som jag inte vill returnera fullständigt till klienten speciellt om klienten inte är autentiserad. Därför har jag skapat en extern FullFilmStudioDTO som innehåller all information en admin behöver ha tillgång till men jag har också en extern FilmStudioMiniDTO som endast innehåller id och namn på studion vilket inte är någon hemlighet och returneras för en oautentiserad användare eller en filmstudio som inte är den sökta. Genom att använda DTO objekt minskar jag risken att exponera känsliga data och data kan begränsas beroende på användarrollnivå. Detta gör att klienten inte behöver få långa och svårlästa svar från APIet, om det inte är nödvändigt, då vi kan välja

hur mycket som ska inkluderas och då kan hela relationer väljas bort i vissa fall som till exempel när jag inte inkluderar en lista med rentedFilmCopies.

Säkerhet

Säkerheten var en stor del av projektet och jag har implementerat det genom autentisering, auktorisering och begränsning av åtkomst till känslig information.

ASP.NET Identity
Identity ansvarar för registrering och inloggning av användaren och den hanterar lösenordet på ett säkert sätt.

Autentisering med JWT
API:et använder JWT-baserad autentisering, vilket innebär att användare måste logga in och få en token för att kunna göra vissa anrop. Token genereras vid inloggning och skickas i Authorization-headern vid efterföljande förfrågningar.

Auktorisering via roller
Användare har olika roller (admin och filmstudio) som styr vilka resurser de har åtkomst till. Detta implementeras via [Authorize(Roles = "admin")] och [Authorize(Roles = "filmstudio")] på endpoints.

DTO-objekt för att begränsa exponerad information
Endast nödvändig information returneras till klienten via DTO-objekt. Till exempel är lösenordet något som man aldrig vill skicka tillbaka till klienten.

Inloggning i frontend sker genom att användaren skickar sitt användarnamn och lösenord till API:et. Om uppgifterna är korrekta returneras en JWT-token och ett Id som lagras i localStorage i webbläsaren. Med hjälp av detta görs alla andra anrop från frontend. Utloggningen sker genom att localStorage töms.

Jag tycker att jag har löst alla uppgiftskrav så som jag tolkade de och är därför nöjd med min lösning. Hade jag haft mer tid skulle jag ha skrivit bättre kod, speciellt i frontend. Jag har inte lagt mycket tid på detta så det är kod som kan minskas genom att skriva fler mindre funktioner och återanvända dem. Jag skulle även vilja lösa uthyrning av filmer på bättre sätt, att knappen byter text och färg exempelvis istället för en alert som det är nu. I min backend har jag några varningar, “Possible null reference assignment”, som jag skulle vilja lösa. Det skulle säkert gå att minska lite kod i backend också men jag är nöjd med att jag har fått alla endpoints att fungera.