Reflektioner Filmstudion

Rest

Resurser som nämns i reflektionen

FilmAsUnauthorized

FilmId

Name

ReleaseDate

Country

Director

NumberOfCopies

Film

FilmId

Name

ReleaseDate

Country

Director

NumberOfCopies

FilmCopies

CreateFilm

Name

ReleaseDate

Country

Director

NumberOfCopies

FilmstudioAsUnauthorized

FilmstudioId

Name

FilmStudio

FilmstudioId

Name

City

RentedFilmCopies

RegisterFilmStudio

FilmStudioCity

FilmStudioName

Password

Username

UserRegister

IsAdmin

Username

Password

AdminResults

UserId

Role

FilmstudioId

Filmstudio

Token

UserAuthenticate

UserName

Password

Api/Films [GET]

Åtkomstpunkten skriver ut alla filmer som finns i databasen. Dock finns det två olika resurser som den skriver ut. Om användaren inte är autentiserad som något får all data förutom listan med filmstudios som har lånat filmen, om användaren är autentiserad som en admin eller en filmstudio så får de ut all data även listan med filmstudios som har lånat filmen.

Api/Films/{ID} [GET]

Åtkomstpunkten skriver ut enbart den filmen som anges i endpointen. En vanlig användare får enbart ut FilmAsUnauthorized, medan om du är en admin eller är en filmstudio så skriver den ut Film.

Api/Films [PUT]

Om man är autentiserad som en admin så skickar du in en resurs in i api:et med modellen CreateFilm som sedan lägger till filmen i databasen om resursen som blir inskickad i api:et är tillåtet. Om man inte är autentiserad så får man error koden 401 ”Unauthorized”.

Api/Films/{ID} [Patch]

Om man är autentiserad som en admin så skickar man in en resurs som ändrar på en film, Filmen är den filmen som anges som ID i endpointen, om man inte är autentiserad så får man error koden 401 ”Unauthorized”.

Api/Films/{ID} [PUT]

Om man är autentiserad som en admin så skickar man in ett objekt som ändrar enbart på objektet ”NumberOfCopies”. Filmen är den filmen som anges som ID i endpointen

/api/films/rent?id={id}&studioid={studioid} [POST]

Om man är en autentiserad filmstudio så kan man låna en film till sin egna studio genom att skicka in ID och studioID i endpointen. Om den autentiserade studion har samma studioID som den som anges i endpointen är studion tillåten att låna filmen, om den autentiserade studion inte har samma studioId så är studion inte tillåten att låna filmen

/api/films/return?id={id}&studioid={studioid} [POST]

Om man är en autentiserad filmstudio så kan man lämna tillbaka en film som finns i ens lista av FilmCopies genom att skicka in ID:et på filmen och StudioId i endpointen. Om den autentiserade studion har samma studioID som den som anges i endpointen är studion tillåten att lämna tillbaka filmen, om den autentiserade studion inte har samma studioId så är studion inte tillåten att lämna tillbaka filmen.

Api/Filmstudio [GET]

Åtkomstpunkten skriver ut alla filmstudios som finns i databasen. Dock finns det två olika resurser som den skriver ut. Om användaren inte är autentiserad som något eller som en filmstudio skriver den ut resursen FilmStudioAsUnauthorized, om användaren är autentiserad som en admin så får de ut all data även listan med filmstudios som har lånat filmer.

Api/Filmstudio/{Id} [GET]

Åtkomstpunkten skriver ut enbart den filmstudion som anges i endpointen. En vanlig användare får enbart ut FilmAsUnauthorized, medan om du är en admin eller har samma id på din filmstudio som du är autentiserad som så skriver den ut resursen Filmstudio.

Api/Filmstudio/register [POST]

I åtkomstpunkten så skickar du in en objekt in i api:et med modellen registerFilmStudio som sedan lägger till filmstudion i databasen om resursen som blir inskickad i api:et innehåller rätt objekt som är giltig. När detta händer skapas även en användare med Filmstudions Användarnamn och lösenord via UserManager

Api/Users/Register [POST]

I åtkomstpunkten skickar du in ett objekt med modellen UserRegister som tar användarnamnet och lösenordet och skapar en admin utav resursen. Om objekten som skickas in är gilltiga och inte redan finns i användardatabasen så skapas en admin och skickar tillbaka resursen AdminResults

Api/Users/authenticate [POST]

I åtkomstpunkten skickar du in en objekt med modellen UserAuthenticate. Åtkomstpunkten kollar sedan med hjälp av SigninManager om de uppgifter som fanns i resursen stämmer överrens med någon av de användarna som är inskrivna i databasen. Om det gör det så skrivs en 201 ut med UserResults. Om inte så blir det en badrequest.

Api/mystudio/rentals [GET]

Åtkomstpunkten skriver ut alla dina filmer i en filmCopy modell. För att hitta dina lånade filmer kollar apiet på tokenen som du skickar med i Api:et

Implementation

Många av de resurser som skickas in i api:et har olika resultat när de sparas och när de skickas ut igen. Detta har jag lyckats med med hjälp av automapper.

Film, FIlmAsUnauthorized + CreateFIlm = Samma resurs bara att Film har Filmcopies med (OBS FIlmAsUnauthorized har FilmId som inte createFilm har)

Filmstudio, FilmstudioAsUnauthorized = Samma resurs bara att filmstudio har city och rentedFilmCopies

FilmStudio, RegisterFilmstudio = Register filmstudio har användarnamn och lösenord som inte filmstudio har, medan filmstudio har city och rentedFilmCopies som RegisterFilmStudio inte har

User, AdminResults = AdminResults skriver enbart ut UserId Role och UserName medan User har en massa olika objekt som behöver gömmas, ett exempel är Password

Säkerhet

Jag har använt mig av JWT bearer tokens, Usermanager, signinmanager och Asp.net.core.Identity för säkerheten i min applikation. Det fungerar så att UserManager sparar användare i Databasen medan signinmanager letar upp i databasen efter användare om man kallar på den, för att applikationen ska känna igen en användare så använder man sig av JWT bearer tokens som håller koll på roller mm. Sen Identity hjälper med att dölj/visa objekt i api:et beroende på vilken roll som Användaren med tokenen har. Detta har hjälpt mig att hålla säkerheten på saker som att skriva ut rätt resurs för rätt användare. Med en enkel IF sats så kunde jag välja om man var admin skulle man skriva ut en resurs som innehöll listor som rentedFilmCopies medan man inte var admin så fick man ut en resurs med utan listan.

Inloggning på klientgränssnittet är rätt enkelt. När du försöker logga in på klientgränssnittet så använder endpointen api/users/authenticate. Om Användaruppgifterna som skickas in i endpointen stämmer överens med en riktig användare så skickar endpointen ut en Token och du får en ny vy samt tokenen skickas till den nya vyn. För att kunna se filmerna med en låneknapp så var man tvungen att ha en token som du får när du loggar in. Denna sparas i en variabel, man kunde välja att spara den på browsern genom att använda localstorage dock valde jag bort det denna gången då jag tycker det var onödigt när jag använde mig av en SPA. När man väl loggar ut sen när man är färdig så laddas sidan bara om då Token är sparad i en variabel som försvinner vid omladdning