Inlämning: Filmstudion, 8Yhp

Inlämningsdatum 11 feb av 23.59 **Poäng** 100 **Lämnar in** en länk till webbplats
Tillgänglig 31 jan kl 0:00–11 feb kl 23.59 12 dagar

Den här uppgiften låstes 11 feb kl 23.59.

Läs hela uppgiftsbeskrivningen innan du börjar!

I detta påhittade uppdrag så ska du skapa en webbapplikation riktad till föreningar som är anslutna till **Sveriges Förenade Filmstudios** **(https://www.filmstudio.se/)** (SFF) där man via ett API och klientgränssnitt kan boka/beställa filmer till sin förening.

SFF fungerar så att lokala filmintresserade bildar föreningar (en filmstudio), dessa föreningar ingår i medlemskap hos SFF som är förbund för alla filmstudios i Sverige. SFF köper rättigheter från filmdistributörerer att låna ett visst antal exemplar av olika filmer, som SFF sen skickar till lokala föreningar. Filmstudion visar sedan dem på på exempelvis kulturhus och mindre biografer runt om i landet.

Förut skedde detta via blanketter, och filmerna man kunde låna fraktades som stora filmrullar, varför filmer bara kunde visas samtidigt på ett begränsat antal ställen i taget. I dag skickas, och visas, filmerna digitalt - men avtalen ser fortfarande likadana ut; så SFF måste begränsa hur många filmstudios som samtidigt kan visa en viss film!

Tips

API:et är tänkt att vara öppet, men också innehålla funktionalitet för **administratörer** hos SFF samt registrerade **Filmstudios**, i *inlämningsuppgiften* så ska båda dessa fritt kunna registrera sig via Webb-API:et. var noga med att läsa kravlistan ordentligt och identifiera vilken information som ska vara öppen eller *begränsad*. Klientgränssnittet är endast för *filmstudios*, så ingen funktionalitet för icke-inloggade användare eller administratörer behöver finnas.

När kravlistan är så här lång är det viktigt att jobba med den i flera steg och att designa din tänkta lösning innan du börjar implementera webbapplikationen i kod. Tänk också på att kravlistan beskriver det som måste finnas - du får, och kan behöva, lägga till ytterligare information i de beskriva *resurserna* i kravlistan.

Tänk på att i din **reflektionsfil** alltid motivera dina ställningstaganden med kopplingar till exempelvis **krav**, upplevt användningsområde eller tekniska designöverväganden med koppling till läsbarhet, underhåll och vidareutveckling.

Implementera Webb-API:et

Ett tips på tillvägagångssätt är att börja med att identifiera och skissa upp hur ditt nya API skall fungera på papper. Lista resurser och vilka åtkomstpunkter som behöver finnas och stäm av att genom att implementera dessa som du tänkt så uppfyller du kravlistan. För API:et i denna uppgiften är din största uppgift att **implementera** ett API som i stort sett är färdig-designat, vilka åtkomstpunkter som ska finnas samt hur dessa *bör* implementeras i kod är givet i kravlistan.

Använd förslagsvis ett ER-diagram (eller skriv bara listor) för att rita upp programmets resurser och modeller - vad innehåller dem, hur är de kopplade, och hur de skiljer sig modellera från resurserna? Hur har du tänkt att hålla koll på antalet exemplar som finns tillgängliga för varje film?

När detta material finns kan det vara värt att även fundera på hur API:et kan användas från klientgränssnittet. Saknas något funktionalitet som kan behöva tillföras i API:et men som inte står uttryckligen i kravlistan?

Kodgranskning

Tänk på att denna uppgift i nästa modul kommer kodgranskas av dina klasskamrater - **dokumentera** därför gärna din lösning och vad som krävs för att få den att fungera. Samt begränsa inte din applikation till att endast fungera på bara ditt operativsystem. Om du exempelvis använder Entity Framework för att lagra tillgängliga användare borde du använda *SQL-Lite* eller *InMemoryDatabase*-Providern.

Lycka till!

Inlämning

I denna programmeringsuppgift ska du skapa och ladda upp en applikation som uppfyller den nedanstående kravlistan. Applikationen ska skapas i versionhanterinssystemet Git och laddas upp på Github som ett privat repo, kom ihåg att bjuda in din lärare!

När du är klar med uppgiften lämnar du in den lösning du skapat som en länk till det repo du skapat på Canvas.

Betygsättning

Efter rättning kommer du få feedback med korta kommenterar för varje rad i poängtabellen. För att ett krav ska vara poänggivande måste även de krav som angetts som kriterium vara uppfyllda.

För betyget **Godkänd** krävs minst *44* poäng från poängtabellen. För betyget **Väl Godkänd** krävs *70* poäng. Erhålls åtminstone *22* poäng så erbjuds ett försök att komplettera din inlämning, annars ges betyget underkänd.

Kravlista

Följande utförliga kravlista beskriver lösningen som du ska skapa och lämna in och fungerar som poängtabell för uppgiften.

Webb API

Din inlämning ska innehålla ett webb-api skapat med ASP.NET och C#. Här har du i förväg fått utförlig information för hur du bör implementera ditt API.

1: Det inlämnande git-repot ska innehålla ett Webb-API skapats med ASP.NET och som går att starta med .NET 5 eller .NET 6.
Max poäng: 1p

- *Läranderesultat: 3*

2: API:et ska tillhandahålla resurserna "film" och "filmstudio".
Max poäng: 2p

- *Läranderesultat: 3, 4*
- Kriterium: Krav 1 måste också uppfyllas
- Klassen “Film” ska finnas på följande plats Models/Film/Film.cs
- Klassen “FilmStudio” ska finnas på följande plats Models/FilmStudio/FilmStudio.cs
- Klassen Film ska ha uppfyllt interfacet IFilm.
- Klassen FilmStudio ska ha uppfyllt interfacet IFilmStudio.

3: En filmstudio ska kunna registrera sig via API:et
Max poäng: 6p

- *Läranderesultat: 3, 4, 5*
- Kriterium: Krav 2 måste också uppfyllas
- Detta anrop ska använda följande metod: POST
- Registreringen ska ske vid ett anrop mot följande endpoint: /api/filmstudio/register
- BODY:n i detta anrop ska kunna innehålla ett json-stringifierat objekt som uppfyller interface:et IRegisterFilmStudio.
  - *Detta objekt innehåller nödvändig data för att korrekt registrera filmstudion.*
- Ett lyckat anrop ska returnera statuskod 200.
- Ett lyckat anrop ska returnera ett json-objekt i sin body som, parse:at till C#, är ett objekt som uppfyller interface:et IFilmStudio.

4: En administratör ska kunna registrera sig via API:et, en administratör är inte en filmstudio.
Max poäng: 2p

- *Läranderesultat: 3, 4, 5*
- Kriterium: Krav 2 måste också uppfyllas
- Detta anrop ska använda följande metod: POST
- Registreringen ska ske vid ett anrop mot följande endpoint: /api/users/register
- BODY:n i detta anrop ska kunna innehålla ett json-stringifierat objekt som uppfyller interface:et IUserRegister.
  - *Detta objekt innehåller nödvändig data för att korrekt registrera en ny användare.*

- Om egenskapen IsAdmin är true, och alla Username och Password innehåller passande information, så registreras användaren som admin.
- Vid lyckad tegistration av en admin returneras ett json-objekt i body:n som, parse:at till C#, är ett objekt som ENDAST innehåller egenskaperna Username, Role och UserId som finns i IUser-interface:et.
  - *Det är viktigt att gömma känslig information. Vid returnering av förbjuden data görs poängavdrag*
- Ett lyckat anrop ska returnera statuskod 200.
  - *Det är viktigt att gömma känslig information. Vid returnering av förbjuden data görs poängavdrag*

5: Både filmstudios och administratörer ska kunna autentisera sig via API:et.
Max poäng: 3p

- *Läranderesultat: 3, 4, 5*
- Kriterium: Krav 3, 4 måste också uppfyllas
- Detta anrop ska använda följande metod: POST
- Autentiseringen ska ske vid ett anrop mot följande endpoint: /api/users/authenticate
- BODY:n i detta anrop ska kunna innehåll ett json-stringifierat objekt som uppfyller interface:et IUserAuthenticate.
  - *Detta objekt innehåller nödvändig data för att korrekt autentisera en användare.*
- Om denna autentisering godkänns så ska ett JSON-objekt som motsvarar interface:t IUser, förutom egenskapen ‘password’, returneras.
  - *Det är viktigt att gömma känslig information. Vid returnering av förbjuden data görs poängavdrag*
- Ifall användarnamn och lösenord som används vid autentiseringen tillhör en registrerad admin så ska egenskapen Role vara en sträng med ordet “admin”.
  - *Stora eller små bokstäver på denna sträng spelar ingen roll.*
- Ifall användarnamn och lösenord som används vid autentiseringen tillhör en registrerad filmstudio så ska egenskapen Role i objektet vara en sträng med ordet “filmstudio”. Egenskapen filmStudioId ska också finnas och egenskapen filmStudio ska innehålla ett objekt som motsvarar interface:et IFilmStudio och innehåller åtminstone data i egenskaperna FilmStudioId, Name och City.
- Vidare autentiseringen i applikationen ska ske via headern "Authentication" vid anrop till API:et.

6: Det ska gå att lägga till nya filmer via Webb-API:et om man är autentiserad som administratör.
Max poäng: 2p

- *Läranderesultat: 3, 4, 5*
- Kriterium: Krav 4, 5 måste också uppfyllas

- Detta anrop ska använda följande metod: PUT
- Anropet ska göras mot följande endpoint: /api/films
- Body:n i anropen ska vara ett JSON-objekt som parse:at motsvarar ett objekt som uppfyller interface:et ICreateFilm.
- Ett anrop av en autentiserad admin ska ge statuskod 200
- Ett anrop av en autentiserad admin ska returnera ett JSON-stringifierat objekt som motsvarar IFilm, inklusive listan av FilmCopies.

7: Via Webb-API:et ska alla filmstudios kunna hämtas.
Max poäng: 3p

- *Läranderesultat: 3, 4, 5*
- Kriterium: Krav 3 måste också uppfyllas
- Detta anrop ska använda följande metod: GET
- Anropet ska kunna göras mot följande endpoint: /api/filmstudios
- Ett lyckat anrop med en oautentiserad användare eller en autentiserad filmstudio ska ge tillbaka ett json-objekt som, parse:at till C#, är en array av objekt som uppfyller interface:et IFilmStudio.cs men som INTE innehåller listan RentedFilmCopies eller egenskapen City.
- Ett lyckat anrop med en autentiserad admin ska innehålla alla egenskaper i objektet, även listan med RentedFilmCopies och dess innehåll.

8: Via Webb-API:et ska informationen om en enskild filmstudio kunna hämtas
Max poäng: 2p

- *Läranderesultat: 3, 4, 5*
- Kriterium: Krav 3, 4, 5 måste också uppfyllas
- Detta anrop ska använda följande metod: GET
- Anropet ska kunna göras mot följande endpoint: /api/filmstudio/{id} där {id} motsvarar FilmStudioId på eftersökt filmstudio.
- Ett lyckat anrop med en oautentiserad användare eller en autentiserad filmstudio men som inte är filmstudion som eftersöks ska ge tillbaka ett json-objekt som, parse:at till C#, är en array av objekt som uppfyller interface:et IFilmStudio.cs men som INTE innehåller listan RentedFilmCopies eller egenskapen City.
- Ett lyckat anrop med en admin ska ge returnera ett json-objekt som, parse:at till C#, är en array av objekt som uppfyller interface:et IFilmStudio.cs.
- Ett lyckat anrop med en autentiserad filmstudio som har samma FilmStudioId som den filmstudio som eftersöks ska ge returnera ett json-objekt som, parse:at till C#, är en array av objekt som uppfyller interface:et IFilmStudio.cs.
    - *Detta innebär även data i RentedFilmCopies*

9: Via Webb-API:et ska alla filmer kunna hämtas.

Max poäng: 3p

- *Läranderesultat: 3, 4, 5*
- Kriterium: Krav 4, 6 måste också uppfyllas
- Detta anrop ska använda följande metod: GET
- Anropet ska kunna göras mot följande endpoint: /api/films
- Ett anrop av en autentiserad användare (filmstudio eller admin) ska ge tillbaka en array med json-objekt som ifall vi parse:ar det till ett objekt i C# motsvarar ett objekt som uppfyller interface:et IFilm.
- Ett anrop av en icke-autentiserad användare ska ge tillbaka en array med json-objekt som ifall vi parse:ar dem till objekt i C# motsvarar objekt som har alla properties i IFilm-interface:et förutom egenskapen `FilmCopies`.

10: Via Webb-API:et ska informationen om en enskild film kunna hämtas.

Max poäng: 2p

- *Läranderesultat: 3, 4, 5*
- Kriterium: Krav 4, 6 måste också uppfyllas
- Detta anrop ska använda följande metod: GET
- Anropet ska kunna göras mot följande endpoint: /api/films/{id} där path-variabeln id motsvarar FilmId för filmen som ska hämtas.
- Ett anrop av en autentiserad användare (filmstudio eller admin) ska ge tillbaka ett json-objekt som ifall vi parse:ar det till ett objekt i C# motsvarar ett objekt som uppfyller interface:et IFilm.
- Ett anrop av en icke-autentiserad användare ska ge tillbaka ett json-objekt som ifall vi parse:ar det till ett objekt i C# motsvarar ett objekt som har alla properties i IFilm-interface:et förutom egenskapen `FilmCopies`.

11: Det ska gå att ändra informationen om en film via Webb-API:et om man är autentiserad som administratör:

Max poäng: 5p

- *Läranderesultat: 3, 4, 5*
- Kriterium: Krav 4, 6, 10 måste också uppfyllas
- Detta anrop ska använda följande metod: PATCH/PUT/POST
- Anropet ska göras till följande endpoint: /api/films/{id} där path-variabeln id motsvarar FilmId:t för filmen som ska uppdateras.
- Ett lyckat anrop av en autentiserad admin ska ge statuskod 200.
- Ett lyckat anrop av en autentiserad admin ska returnera det uppdaterade objektet som är ett json-objekt som parse:at till C# uppfyller interface:et IFilm.
- Ett anrop av en icke-autentiserad användare ska ge statuskod 401.

12: En autentiserad administratör ska kunna ändra antalet tillgängliga exemplar som går att låna av varje film.

Max poäng: 4p

- *Läranderesultat: 3, 4, 5*
- Kriterium: Krav 4, 6, 10 måste också uppfyllas
- Detta anrop ska använda följande metod: PATCH/PUT/POST
- Anropet ska göras till följande endpoint för att fungera: /api/films/{id} där path-variabeln id motsvarar id:t för filmen som ska uppdateras.
- I body:n för detta anrop ska ett JSON-objekt som, parse:at till ett C#-objekt, som uppfyller interface:et IFilm kunna skickas.
- Ett lyckat anrop av en autentiserad admin ska ge statuskod 200 och returnera det uppdaterade objektet. Detta objekt är återigen ett json-objekt som parse:at till C# uppfyller interface:et IFilm.
- Ett anrop av en icke-autentiserad användare ska ge statuskod 401.

13: En autentiserad filmstudio ska kunna låna ett exemplar av en film via Webb-API:et om det finns exemplar tillgängliga.

Max poäng: 4p

- *Läranderesultat: 3, 4, 5*
- Kriterium: Krav 3, 4, 5, 6, 8 måste också uppfyllas
- Detta anrop ska kunna använda följande metod: POST
- Anropet ska göras till följande endpoint för att fungera: /api/films/rent?id={id}&studioid={studioid} där parametern {id} motsvarar id:t för filmen som ska lånas och {studioid} id:t för studion som ska låna filmen.
- Ett lyckat anrop ska returnera statuskod 200.
- Om anropet görs av en icke-autentiserad admin, eller av en autentiserad filmstudio vars id inte överensstämmer med id:t som anges i anropet, ska anropet returnera statuskod 401 eller annan felkod (absolut inte 200) och lånet ska inte godkännas.
- Om filmen inte finns ska statuskod 409 returneras.
- Om filmen inte har några lediga kopior ska statuskod 409 returneras.
- Om filmstudion redan hyr en kopia av samma film ska statuskod 403 returneras och lånet ska inte gå igenom.

14: En autentiserad filmstudio ska kunna lämna tillbaka ett lånat exemplar av en film via Webb-API:et.

Max poäng: 2p

- *Läranderesultat: 3, 4, 5*
- Kriterium: Krav 3, 4, 5, 6, 8, 13 måste också uppfyllas

- Detta anrop ska kunna använda följande metod: POST
- Anropet ska göras till följande endpoint för att fungera: /api/films/return?id={id}&studioid={studioid} där parametern {id} motsvarar id:t för filmen som ska lånas och {studioid} id:t för studion som ska låna filmen.
- Ett lyckat anrop ska returnera statuskod 200.
- Om anropet görs av en icke-autentiserad användare, eller om filmstudion som är inloggad inte överensstämmer med filmstudion vars id anges i anropet, ska anropet returnera statuskod 401.
- Om filmen inte finns ska statuskod 409 returneras.

15: En autentiserad filmstudio ska via Webb-API:et kunna hämta vilka filmer som denna studio för närvarande har lånat
Max poäng: 2p

- *Läranderesultat: 3, 4, 5*
- Kriterium: Krav 3, 4, 5, 6, 8, 13 måste också uppfyllas
- Detta anrop ska kunna använda följande metod: GET
- Anropet ska göras till följande endpoint för att fungera: /api/mystudio/rentals
- Ett lyckat anrop ska returnera statuskod 200.
- Ett lyckat anrop ska returnera en array bestående av objekt som motsvarar interface:t IFilmCopy.
- Dessa objekt ska överrensstämma med de filmer som den autentiserade filmstudion för tillfället hyr.
- Om anropet görs av en icke-autentiserad filmstudio ska anropet returnera statuskod 401.

Klientgränsnitt

Inlämningen ska också innehålla ett klientgränssnitt där webb-api:et kan användas via en webbläsare. Här har du fått friare tyglar gällande implementation och användning av API:et från klientgränssnittet.

16: Det inlämnade git-repot ska innehålla ett fungerande kliengränsnitt för att interagera med Webb-API:et.
Max poäng: 1p

- *Läranderesultat: 3*
- Kriterium: Krav 1 måste också uppfyllas

17: Klientgränssnittet tillåter åtkomst till API:et för registerade filmstudios att logga-in samt

logga-ut
Max poäng: 4p

- *Läranderesultat: 3, 4, 5*
- Kriterium: Krav 16 måste också uppfyllas

18: Från klientgränssnittet kan besökaren se alla tillgängliga filmer från API:et.
Max poäng: 2p

- *Läranderesultat: 3, 4*
- Kriterium: Krav 16 måste också uppfyllas

19: En filmstudio som är inloggad kan tydligt se sina lånade filmer i klientgränssnittet.
Max poäng: 2p

- *Läranderesultat: 3, 4, 5*
- Kriterium: Krav 16, 17 måste också uppfyllas

20: En filmstudio som är inloggad kan låna ett exemplar av en film.
Max poäng: 4p

- *Läranderesultat: 3, 4, 5*
- Kriterium: Krav 16, 17 måste också uppfyllas

21: En filmstudio som är inloggad kan lämna tillbaka ett lånat exemplar av en film.
Max poäng: 4p

- *Läranderesultat: 3, 4, 5*
- Kriterium: Krav 16, 17 måste också uppfyllas

Dokumentatation & reflektion

Till inlämningen bör du också dokumentera samt, för ett högre betyg, visa din förståelse över uppgiften via objektiva reflektioner

22: Det inlämnade git-repot innehåller en fil med namnet "readme.md" där du beskrivit hur ditt

API ska startas och användas.
Max poäng: 1p

- *Läranderesultat: 3, 4, 5*

23: Det inlämande git-repot innehåller en fil med namnet "reflections" i formatet md, txt eller pdf.
Max poäng: 1p

- *Läranderesultat: 3, 4, 5, 6*

24: I reflections-filen under rubkriken "REST" har du förklarat och motiverat hur du implementerat dina listade åtkomstpunker och resurser (klasser & interface) för att uppfylla kraven på Webb-API:et.
Max poäng: 7p

- *Läranderesultat: 3, 4*
- Kriterium: Krav 23 måste också uppfyllas

25: I reflections-filen under rubkriken "Implementation" jämför du och motiverar vilka interna modeller som finns och om/hur de skilljer sig från de synliga resuserna i Webb-API:et.
Max poäng: 7p

- *Läranderesultat: 3, 4*
- Kriterium: Krav 23 måste också uppfyllas

26: I reflections-filen under rubkriken "Säkerhet" har du motiverat dina säkerhetsåtgärder i applikation. Svara på hur har du kontrollerat och begränsat vilken information som finns tillgänglig vid anrop till API:et? samt hur inloggning och utloggning fungerar i klientgränsnittet.
Max poäng: 12p

- *Läranderesultat: 4, 5*
- Kriterium: Krav 23 måste också uppfyllas