Dossier

- Student: Robbe Pauwels
- Studentennummer: 202396579
- E-mailadres: mailto:robbe.pauwels2@student.hogent.be

- Demo: <[DEMO_LINK_HIER]>
- GitHub-repository: <GITHUB_REPO__HIER>
- Web Services:
    - Online versie: [<ONLINE_VERSIE_HIER>]

Logingegevens

Lokaal

Inloggen gebruiker met ADMIN en USER rol

- Gebruikersnaam/e-mailadres: robbe.pauwels2@student.hogent.be
- Wachtwoord: 12345678

Inloggen gebruiker met ADMIN rol

- Gebruikersnaam/e-mailadres: test.gebruiker@hogent.be
- Wachtwoord: 12345678

Online

Inloggen gebruiker met ADMIN rol

- Gebruikersnaam/e-mailadres: robbe.pauwels2@student.hogent.be
- Wachtwoord: 12345678

Inloggen gebruiker met ADMIN en USER rol

- Gebruikersnaam/e-mailadres: test.gebruiker@hogent.be
- Wachtwoord: 12345678

Projectbeschrijving

Ik heb gekozen om een Film API te maken waarbij het mogelijk is om films op te vragen, toe te voegen, te updaten en te verwijderen. Deze films bevatten ook personen waaronder 1 regisseur en 1 of meerdere acteurs, awards die de film gewonnen heeft en Locaties waar er gefilmd is. Als ingelogde gebruiker is het mogelijk om deze op te vragen en toe te voegen maar enkel gebruikers met een ADMIN rol kunnen deze updaten en verwijderen. Het is ook mogelijk om personen, awards en locaties op zich zelf toe te voegen en deze dan toe te voegen aan de film met hun ID in plaats van ze gelijk met de film aan te maken.

 image

API calls

Film

- `GET /api/film`: alle films ophalen
- `GET /api/film/:id`: Film met een bepaald id ophalen
- `POST /api/film`: Film toevoegen
- `PUT /api/film/:id`: Film met bepaald id updaten. Dit kan enkel een gebruiker met de ADMIN rol doen.
- `DELETE /api/film/:id`: Film met bepaald id deleten. Dit kan enkel een gebruiker met de ADMIN rol doen.

Persoon

- `GET /api/persoon`: alle personen ophalen
- `GET /api/persoon/:id`: Persoon met een bepaald id ophalen
- `POST /api/persoon`: Persoon toevoegen
- `DELETE /api/persoon/:id`: Persoon met bepaald id deleten. Dit kan enkel een gebruiker met de ADMIN rol doen.

Locatie

- `GET /api/locatie`: alle locaties ophalen
- `GET /api/locatie/:id`: Locatie met een bepaald id ophalen
- `POST /api/locatie`: Locatie toevoegen
- `DELETE /api/locatie/:id`: Locatie met bepaald id deleten. Dit kan enkel een gebruiker met de ADMIN rol doen.

Award

- `GET /api/award`: alle awards ophalen
- `GET /api/award/:id`: Award met een bepaald id ophalen
- `POST /api/award`: Award toevoegen
- `DELETE /api/award/:id`: Award met bepaald id deleten. Dit kan enkel een gebruiker met de ADMIN rol doen.

Session

- `POST /api/sessions`: Gebruiker inloggen
- `POST /api/sessions`: Gebruiker uitloggen

User

- `POST /api/users`: Registreer een nieuwe gebruiker
- `PUT /api/users/:id`: Updaten gebruiker met bepaald id
- `GET /api/users`: Vraag alle gebruikers op. Dit kan enkel een gebruiker met de ADMIN rol doen.
- `GET /api/users/:id`: Registreer een nieuwe gebruiker. Dit kan enkel een gebruiker met de ADMIN rol doen.
- `DELETE /api/users/:id`: Verwijderen gebruiker. Dit kan enkel een gebruiker met de ADMIN rol doen.

Health

- `GET /api/health/ping`: Controleren of service online staat
- `GET /api/health/version`: Huidige versie opvragen

User

- `GET /api/users`: Alle users opvragen. Dit kan enkel een gebruiker met de ADMIN rol doen.
- `GET /api/users/:id`: 1 specifieke user opvragen. Dit kan enkel een gebruiker met de ADMIN rol doen.
- `POST /api/users`: user toevoegen.
- `PUT /api/users/:id`: user updaten. Dit kan enkel een gebruiker met de ADMIN rol doen.
- `DELETE /api/users/:id`: user deleten. Dit kan enkel een gebruiker met de ADMIN rol doen.

Behaalde minimumvereisten

Web Services

Datalaag

- ☑ voldoende complex en correct (meer dan één tabel (naast de user tabel), tabellen bevatten meerdere kolommen, 2 een-op-veel of veel-op-veel relaties)
- ☑ één module beheert de connectie + connectie wordt gesloten bij sluiten server
- ☑ heeft migraties - indien van toepassing
- ☑ heeft seeds

Repositorylaag

- ☑ definieert één repository per entiteit - indien van toepassing
- ☑ mapt OO-rijke data naar relationele tabellen en vice versa - indien van toepassing
- ☑ er worden kindrelaties opgevraagd (m.b.v. JOINs) - indien van toepassing

Servicelaag met een zekere complexiteit

- ☑ bevat alle domeinlogica
- ☑ er wordt gerelateerde data uit meerdere tabellen opgevraagd
- ☑ bevat geen services voor entiteiten die geen zin hebben zonder hun ouder (bv. tussentabellen)
- ☑ bevat geen SQL-queries of databank-gerelateerde code

REST-laag

- ☑ meerdere routes met invoervalidatie
- ☑ meerdere entiteiten met alle CRUD-operaties
- ☑ degelijke foutboodschappen
- ☑ volgt de conventies van een RESTful API
- ☑ bevat geen domeinlogica
- ☑ geen API calls voor entiteiten die geen zin hebben zonder hun ouder (bv. tussentabellen)
- ☑ degelijke autorisatie/authenticatie op alle routes

Algemeen

- ☑ er is een minimum aan logging en configuratie voorzien
- ☑ een aantal niet-triviale én werkende integratietesten (min. 1 entiteit in REST-laag >= 90% coverage, naast de user testen)
- ☑ node_modules, .env, productiecredentials... werden niet gepushed op GitHub
- ☑ minstens één extra technologie die we niet gezien hebben in de les
- ☑ maakt gebruik van de laatste ES-features (async/await, object destructuring, spread operator...)
- ☑ de applicatie start zonder problemen op gebruikmakend van de instructies in de README
- ☑ de API draait online
- ☑ duidelijke en volledige README.md
- ☑ er werden voldoende (kleine) commits gemaakt
- ☑ volledig en tijdig ingediend dossier

Projectstructuur

Web Services

Ik heb de mappen structuur van de voorbeeld applicatie gevolgd om zo alle verschillende lagen mooi gescheiden te houden.

Extra technologie

Web Services

Ik heb gekozen om `apidoc` te gebruiken voor de documentatie. Ik vondt dit een goeie en veilige keuze omdat hier weinig kon fout lopen dat het project in zijn geheel zou beinvloeden. `https://apidocjs.com/`

Reflectie

Ik vondt het een zeer uitdagend project. In het begin was het lastig om er goed aan te kunnen beginnen omdat ik niet kon inschatten wat er verwacht werd en dit redelijk lang onduidelijk bleef voor mij. Na een tijd is het duidelijker geworden wat er verwacht werd en hoe ik mijn project er aan kon laten voldoen. Ik heb veel bijgeleerd over hoe je code kan testen, verbeteren en verbindingen maken met databanken. De cursus vondt ik redelijk duidelijk alleen was het op het einde lastig om te weten of ik alles correct werkende had gekregen.