# Project AMO19 - Blok 6 (2020)

In blok 6 ga je verder met het project van blok 5. Maar de eisen die we stellen worden wat aangepast en uitgebreid. Omdat dit document grotendeels ongewijzigd is ten opzichte van het projectdocument van blok 5, is er een hoofdstukje met een opsomming van de wijzigingen toegevoegd.

Inhoud

[Project AMO19 - Blok 6 (2020) 1](#_Toc56501714)

[Wijzigingen in het document ten opzichte van blok 5 2](#_Toc56501715)

[Beoordeling 2](#_Toc56501716)

[De techniek van de applicatie 3](#_Toc56501717)

[MVC in de backend 3](#_Toc56501718)

[Basisfunctionaliteit van de applicatie 4](#_Toc56501719)

[Indexpagina 4](#_Toc56501720)

[Filmpagina 4](#_Toc56501721)

[Persoonpagina 5](#_Toc56501722)

[Navigatie 5](#_Toc56501723)

[Waarderingsfunctionaliteit en favorietenpagina 6](#_Toc56501724)

[Opslag in de database 6](#_Toc56501725)

[Inlogcheck 6](#_Toc56501726)

[Registratiepagina 6](#_Toc56501727)

[Inlogpagina 7](#_Toc56501728)

[Uitloglink en nickname 7](#_Toc56501729)

[Favorietenpagina 7](#_Toc56501730)

[Waarderingsinterface 7](#_Toc56501731)

[Gemiddelde waardering 7](#_Toc56501732)

[GIT-repository en bestandsindeling 8](#_Toc56501733)

[Documentatie 9](#_Toc56501734)

[Plan van aanpak/planning 9](#_Toc56501735)

[Vormgeving 9](#_Toc56501736)

[Technisch ontwerp 11](#_Toc56501737)

[Database-export 11](#_Toc56501738)

[Configuratie 11](#_Toc56501739)

[Verslag 11](#_Toc56501740)

[Deadline en presentatie 12](#_Toc56501741)

## Wijzigingen in het document ten opzichte van blok 5

* dit hoofdstuk is toegevoegd ;-)
* het hoofdstuk *Werken in sprints* is verwijderd
* het hoofdstuk *De techniek van de applicatie* is iets aangepast, omdat er een kleine onjuistheid in stond over de werking van Vue
* het subhoofdstuk *MVC in de backend* is toegevoegd
* het hoofdstuk *Waarderingsfunctionaliteit en favorietenpagina* is aangepast (er is nog maar één versie: de moeilijke)
* het hoofdstuk *Git-repository en bestandsindeling* is gewijzigd
* het hoofdstuk *Documentatie* is gewijzigd
* het hoofdstuk *Deadline en presentatie* is toegevoegd

## Beoordeling

Aan het eind van het blok wordt je beoordeeld op de volgende onderdelen:

* applicatie (50%)
  + frontend; design + html bootstrap
  + vue; de basis van je applicatie
  + backend; php + sql
* documentatie (25%)
  + staat verderop
* oplevering (25%)
  + deployment
  + eindpresentatie

**Om je nog wat meer houvast te geven zal er in een toekomstige versie van dit document nog een onderverdeling (meer details) komen van bovenstaand lijstje.**

## De techniek van de applicatie

De applicatie moet worden gebouwd vanuit **Vue** (javascript framework). Vue genereert de frontend, in de vorm van **Bootstrap** (html/css/javascript framework). Vue communiceert door middel van AJAX met een PHP/MySQL-backend. Daarbij is het de bedoeling dat je in PHP Objectgeoriënteerd werkt.

In de basis maak je je applicatie met het javascript-framework Vue. De scripts die je maakt, zorgen er onder andere voor dat er wordt gecommuniceerd met de backend. Dat gebeurt met AJAX-calls.

Zo'n AJAX-call doet een PHP-request op de server. Het PHP-script doet eventueel een validatie (bij een ingezonden formulier), en indien nodig een actie op de database (iets ophalen, iets toevoegen of iets veranderen), en stuurt als response een **JSON-string** terug naar het Vue-script dat de AJAX-call deed. Dat terugsturen is niets anders dan echoën. Er wordt nu alleen geen html-pagina geëchood, maar een JSON-string, en door middel van een header-aanpassing maak je aan het ontvangende Vue-script bekend dat het een JSON-response is.

Afhankelijk van de response, verandert je Vue-script vervolgens de frontend. Die frontend moet gemaakt worden met behulp van het html/css-framework Bootstrap. Je maakt gebruik van het grid-systeem van bootstrap om ervoor te zorgen dat je frontend zich aanpast aan de grootte van het scherm van de bezoeker. Je gebruikt ook verschillende soorten bootstrap-elementen om een fraai en prettig werkende applicatie te krijgen.

### MVC in de backend

Bij het vak backend leer je momenteel hoe je een miniframework opzet met het MVC designpattern. Het is de bedoeling dat je dit gaat gebruiken voor de backend van je applicatie. Mocht je daarin al ver gevorderd zijn, dat is geen groot probleem. Het grootste deel van je code (als jet het goed gedaan hebt) zit in de models. En die neem je mee naar de nieuwe MVC-versie van je applicatie.

## Basisfunctionaliteit van de applicatie

Je gaat een filmencyclopedie(tje) bouwen (denk aan IMDB). De applicatie bestaat in de basis uit de volgende pagina's:

* Indexpagina (home)
* Filmpagina
* Persoonpagina
* Favorietenpagina

### Indexpagina

De indexpagina bevat een zoekinterface. De zoekinterface bevat de volgende elementen:

* text-input om in te kunnen voeren **wat** er gezocht wordt
* selectbox om te kiezen **waarin** er gezocht wordt; de opties zijn: filmtitel, acteur, regisseur, ... (zelf te kiezen extra zoekmogelijkheden)
* button om de zoekactie uit te voeren

Als er geen zoekactie is, toont de indexpagina alle in de database aanwezige filmtitels. Als er wel een zoekactie is, toont de indexpagina alleen filmtitels die aan de zoekactie voldoen. Als er geen filmtitels zijn moet dat uitgelegd worden aan de bezoeker.

De gevonden filmtitels worden getoond in een lijst.

Bij elke filmtitel staat vermeld wie de regisseur is, van welk jaar de film is, en in welk land de film gemaakt is. Elke filmtitel is aanklikbaar en bevat een link naar de filmpagina van de betreffende film.

De lijst is initieel nooit langer dan vijf titels. Onderaan de lijst staat een button/link om meer titels op te halen. Elke klik op die button haalt nog eens vijf titels op.

De button om de zoekactie uit te voeren, en de button onder aan de lijst om meer filmtitels op te halen, zijn AJAX-calls (GET). Ook het initieel ophalen van filmtitels (dus voordat er een zoekactie is geweest) gebeurt met AJAX. Zolang de informatie niet binnen is, wordt er een pauze-gifje getoond.

### Filmpagina

De filmpagina bevat alle informatie van een film. Die informatie wordt niet meteen meegestuurd met de response, maar vanuit de browser met behulp van een AJAX-call (GET). Zolang de informatie niet binnen is, wordt er een pauze-gifje getoond.

De pagina bevat de volgende informatie:

* filmtitel
* jaar van verschijnen
* land van verschijnen
* regisseur
* acteurs
* beschrijving
* ... (eventueel door jezelf toe te voegen extra informatie)

De naam van de regisseur en de namen van de acteurs werken als links naar de persoonpagina's van de betreffende acteurs en regisseurs.

### Persoonpagina

De persoonpagina bevat alle informatie over een persoon. Die informatie wordt niet meteen meegestuurd met de response, maar vanuit de browser met behulp van een AJAX-call (GET). Zolang de informatie niet binnen is, wordt er een pauze-gifje getoond.

De pagina bevat de volgende infomatie:

* naam van de persoon
* geboortedatum
* sterfdatum
* ... (eventueel door jezelf toe te voegen extra informatie)
* indien van toepassing: lijst met filmtitels (op dezelfde manier als op de indexpagina) die de persoon heeft geregisseerd
* indien van toepassing: lijst met filmtitels (op dezelfde manier als op de indexpagina) waarin de persoon heeft geacteerd

De filmtitels zijn weer links naar de filmpagina.

Hieruit kun je opmaken dat een persoon zowel acteur als regisseur kan zijn - en eventueel in dezelfde film. **Dat heeft gevolgen voor je databaseontwerp**.

### Navigatie

De drie pagina's hebben een navigatiebalk (of iets dergelijks) met daarop een link naar de indexpagina.

## Waarderingsfunctionaliteit en favorietenpagina

Deel van je applicatie wordt een waarderingsfunctionaliteit. Dat kan op allerlei manieren: 5-sterrensysteem of rapportcijfer, en misschien kan jezelf iets origineels bedenken. De waarderingen van een bezoeker betreffen alleen de films (dus niet de personen).

### Opslag in de database

Essentieel is dat het waarderen van een film persoonsgebonden is. Daarmee wordt bedoeld dat elke bezoeker elke film ten hoogste één keer kan waarderen (herwaarderen kan wel). Dat betekent dat elke waardering van een film door een bezoeker vastgelegd moet worden in de database in de vorm: *bezoeker x waardeert film y met cijfer (o.i.d.) z*. Dát betekent weer dat er ingelogd moet worden om te kunnen waarderen. En dát betekent weer dat er een nieuw useraccount geregistreerd moet kunnen worden.

### Inlogcheck

In de hieronder beschreven functionaliteit, zul je zien dat Vue moet controleren of een bezoeker is ingelogd. Wat er zoal in de applicatie getoond wordt, hangt daar namelijk van af.

Die check gaat met een AJAX-call vanuit Vue. Het controlemiddel is echter niet een wachtwoord, maar een *token*. Dat is een tijdelijk systeem-gegenereerd wachtwoord. Het token wordt door PHP gegenereerd als de bezoeker succesvol inlogt via het inlogformulier; het token wordt opgeslagen in de database - gekoppeld aan de id van de bezoeker, en het token wordt ook teruggestuurd in de JSON-response. Vue slaat het token vervolgens locaal op.

### Registratiepagina

Als een bezoeker NIET ingelogd is, bevat de navigatiebalk een link naar de registratiepagina. De registratiepagina bevat een formulier waarin de volgende gegevens ingevoerd kunnen worden:

* nickname
* e-mailadres
* controle e-mailadres
* wachtwoord
* controle wachtwoord

Verzenden van de formuliergegevens doe je met een AJAX-call (POST). De gegevens worden in de backend gevalideerd en eventueel opgeslagen in de database. De response van de call kan zijn: goedgekeurd of afgekeurd.

Indien afgekeurd, dan moet het formulier validatie-errors (die moeten dus ook in de response) tonen. De mogelijke validatie-errors zijn:

* ongeldige nickname (namelijk leeg)
* bestaand e-mailadres
* ongeldig e-mailadres
* ongelijk controle-adres
* ongeldig wachtwoord (wat de regels zijn, mag je zelf bedenken)
* ongelijk controle-wachtwoord

Indien goedgekeurd, dan moet de response ervoor zorgen dat de bezoeker ingelogd is (token!), en dat de pagina wordt getoond waar de bezoeker was, toen hij op de registratielink klikte. Indien je dat te lastig vindt, kies dan altijd voor de indexpagina.

### Inlogpagina

Als een bezoeker NIET ingelogd is, bevat de navigatiebalk een link naar de inlogpagina. De inlogpagina bevat een formulier waarin de volgende gegevens ingevoerd kunnen worden:

* e-mailadres
* wachtwoord

Verzenden van de formuliergegevens doe je met een AJAX-call (POST). De gegevens worden in de backend gevalideerd. Indien valide, dan wordt een token gegenereerd en in de database opgeslagen. De response van de call kan zijn: wel of geen token. Indien geen token, dan moet de pagina een error tonen: *inloggegevens onjuist.*

Indien goedgekeurd (dus wel een token), dan moet het token in een cookie worden opgeslagen en wordt de pagina getoond waar de bezoeker was, toen hij op de inloglink klikte. Indien je dat laatste te lastig vindt, kies dan altijd voor de indexpagina.

### Uitloglink en nickname

Indien de bezoeker WEL ingelogd is, dan bevat de navigatiebalk de nickname van de bezoeker en een uitloglink. De uitloglink zorgt ervoor dat de bezoeker uitlogt: het token moet gewist worden in de database. Ook dit gaat weer met een AJAX-call (POST).

### Favorietenpagina

De favorietenpagina wordt toegevoegd aan de navigatiebalk. Maar alleen als de bezoeker WEL ingelogd is. Indien de bezoeker wel is ingelogd, maar nog geen favorieten heeft, dan wordt dit item inactief (en vaag) getoond.

De favorietenpagina bevat weer de gebruikelijke lijst met filmtitels (op dezelfde manier als de indexpagina), maar nu staan ze op volgorde van voorkeur van de bezoeker. Uiteraard is de waardering door de bezoeker ook zichtbaar in de lijst.

### Waarderingsinterface

Indien de bezoeker WEL ingelogd is, dan bevat de filmpagina de eventuele waardering van de bezoeker voor de film en de mogelijkheid om de film te (her)waarderen.

(Her)waarderen leidt tot een AJAX-call (POST).

Indien de bezoeker NIET ingelogd is, dan bevat de filmpagina een trigger (bijvoorbeeld geblurd waarderingsinstrument met mouseover-tekst) die de bezoeker duidelijk moet maken dat hij moet inloggen/registreren om te kunnen waarderen.

### Gemiddelde waardering

Voeg op de filmpagina ook de **gemiddelde** waardering van een film toe. Voor de duidelijkheid: dit wordt dus berekend aan de hand van de waarderingen van je bezoekers; het is niet de bedoeling dat dit een gegeven uit IMDB is of zo.

## GIT-repository en bestandsindeling

Om je project voor te bereiden moet je een GIT-repository maken op GitHub classrooms: <https://classroom.github.com/a/bc5wgWQA>. Accepteer de uitnodiging en clone daarna je repo naar de juiste plek op je laptop - ergens onder de Apache webroot.

Je repo bevat de documentatie, de Vue-applicatie, en de backend. Aan je repo moet je ook een Kanban-planbord toevoegen.

De repo is als volgt ingedeeld:

* je projectnaam (bevat ook je naam)
  + documentatie
    - vormgeving
    - plan van aanpak/planning
    - technisch ontwerp
    - configuratieoverzicht
    - database-ontwerp en vulling (SQL-exportbestand)
    - verslag
  + Vue-applicatie
  + Backend
    - webroot
      * .htaccess
      * index.php
    - php classes
      * core (o.a. Router, View, MyPDO)
      * applicatie (o.a. models, controllers)
    - includes
      * init.php (o.a. met autoloader)
      * configuratie (php of ini)

Zoals je ziet is de webroot voor de backend geen onderdeel van de Vue-applicatie. Dit blijkt niet goed te kunnen, dus de beschrijving in het projectdocument van blok 5 was niet juist. Het betekent ook dat de API-calls naar de backend in de Vue-applicatie niet met een relatieve url kunnen. Dat moet een absolute url worden. Het pad naar de webroot van de backend moet configureerbaar zijn, zodat docenten het makkelijk kunnen aanpassen naar hun locale situatie.

Je eigen naam en de naam van het project moeten dus beide onderdeel zijn van de naam van de repo.

Telkens als je iets afrondt, (onderdeel van de documentatie, onderdeel van de code), moet je je repo verversen door de nieuwe bestanden te committen en te pushen naar de repo. We verwachten min of meer dagelijks nieuwe commits. Je kunt wel eens een dagje overslaan natuurlijk, maar één of twee keer per week is in elk geval te weinig.

## Documentatie

Een zeer belangrijk deel van je project - je wordt er op beoordeeld - is je documentatie. Doe niet alle documentatie in één bestand, maar maak meerdere bestanden. Met je documentatie laat je zien hoe je te werk gaat. Je doet er verslag mee van je voorbereiding. Om een voorbeeld te noemen: het is niet goed als we eerst een database-export zien verschijnen op je repo, en later een document waarin je database-ontwerp staat beschreven.

De documentatie moet bestaan uit een aantal onderdelen:

### Plan van aanpak/planning

Eerste deel hiervan is een takenlijst (wat moet er allemaal gebeuren) op twee niveau's: een hoofdindeling en een onderverdeling. Aan elke taak koppel je een geschatte tijd.

Tweede deel is een planning: je zet daarbij de taken op de juiste volgorde en voegt data toe: wanneer begin je eraan en wanneer is het af. Je voert de geplande taken ook in in het aan je repo gekoppelde Kanban-planbord.

**Het is helemaal niet erg als je planning aan het einde van de rit niet erg blijkt te kloppen.** Wat wel belangrijk is, is dat je door een reflectieverslag laat zien welke inschattingen van jou niet goed bleken te zijn en waar dat dan aan lag.

Om je op weg te helpen, hieronder een voorbeeld van de uitsplitsing in taken voor de indexpagina (waarbij ik wellicht dingen vergeet):

* Nadenken over het ontwerp x uur
* XD-ontwerp indexpagina x uur
* Mock-up indexpagina x uur
* Prototype indexpagina x uur
* AJAX-call bij geen zoekactie x uur
* AJAX-call bij wel zoekactie x uur
* AJAX-call bij *meer titels* x uur
* Inhoud verwachte JSON x uur
* Vue-script indexpagina x uur
* Bootstrap-code indexpagina x uur
* Database-tabel *films* x uur
* Vulling voor de database x uur
* ERD x uur
* PHP class *Film* x uur
* PHP-script voor de AJAX-calls x uur

In welke volgorde je werkt, hangt af van je persoonlijke voorkeur. Wat in elk geval een rol zal spelen, is in hoeverre je je sommige dingen nog helemaal niet in de les hebt gehad.

### Vormgeving

Aan het begin van het project ga je nadenken over hoe de applicatie er uit gaat zien. Dit gebeurt in een proces waarbij je een aantal tussenproducten maakt. In feite zet je tijdens de ontwerpfase het idee van de applicatie om naar een visueel ontwerp. Daarom is het van belang om te weten wat de basisfunctionaliteiten zijn van het product. De volgende tussenproducten zien we graag terug in de documentatie:

* Een mock-up
* Een moodboard
* Een XD-bestand

#### Een mock-up

Een mock-up is een tijdens de ontwerp- of productiefase op schaal of op ware grootte gemaakt model van een ontwerp of product. In de software-industrie komt het begrip tevens voor bij het vroeg in het ontwikkelproces testen van de gebruikersinterface van het software-ontwerp.

Een mock-up krijgt pas de term prototype als het ontwerp ook echt werkt. Meestal zijn de mock-ups dus voorbeelden qua uiterlijk.

Een mock-up kan je heel simpel maken met alleen een pen en wat papier. Daarvoor zou je ook gebruik kunnen maken van de volgende templates: https://sneakpeekit.com/. Voeg aan de documentatie een mock-up van de webpagina’s toe. Doe dit zowel voor de desktop, als voor de mobile.

#### Een moodboard

Een moodboard is een digitale collage van posterformaat. Het wordt gebruikt om een concept, idee, een gedachte of gevoel te visualiseren. Deze collage kan gebruikt worden om een sfeer weer te geven met behulp van afbeeldingen of notities. Het is een uitstekend middel om ideeën en gedachtegangen weer te geven zonder direct in te gaan op de functionaliteit en andere concrete uitwerkingen. Een moodboard versterkt dus een creatief denkvermogen. Als een moodboard wordt gebruikt om ontwerpen en ideeën uit te werken, is deze bruikbaar in drie fasen:

* Het concept board kan een uitstraling en gevoel weergeven
* Het design board kan helpen om een concept uit te werken naar een ontwerp
* Het styleboard zal het ontwerp verder vormgeven naar een meer definitieve vormgeving

Daarnaast is een moodboard uitstekend geschikt om powerpoint presentaties te maken of te gebruiken in een verkoopgesprek. Een digitaal moodboard wordt ingericht met behulp van fotobewerkingsprogramma’s als Photoshop, Illustrator en XD. Het is mogelijk om afbeeldingen van Google, Flickr of Pinterest te kopiëren en op het moodboard te plaatsen. Voeg in de documentatie een moodboard met niet alleen afbeeldingen, maar ook de gekozen kleuren en typografie. Eventueel kan je een aparte document toevoegen met afbeeldingen van voorbeeld websites.

#### XD

Binnen een website is de werking van en interactie tussen functionaliteiten belangrijk. De wireframes en mockups gaan hier niet heel diep op in. Toch is het van belang om deze user interface (UI) te testen bij een testgroep voordat er gestart wordt met de bouw van de website. Dit kan met een prototype. Een prototype wordt vooral ingezet bij complexe concepten.

Omdat een prototype een simulatie is van het eindproduct, kan een prototype gebruikt worden om de werking en gebruiksvriendelijkheid van de applicatie te testen. Het testen van de interactie tussen de functionaliteiten is belangrijk voor de eindgebruiker en voor de ontwikkelaars. De ontwikkelaars kunnen met het prototype feedback geven op de werking en zijn met het prototype in staat om een betrouwbare en transparante berekening te geven van de ontwikkelingskosten. De designer gebruikt het prototype als het ware als communicatiemiddel naar het ontwikkelteam.

In XD werk je de mock-ups van de webpagina’s verder uit naar een werkende prototype. Daarbij maak je ook gebruik van de moodboard. Je XD bestand bestaat uit de volgende drie onderdelen:

* Design
  + Minstens twee ontwerpen van de webpagina’s. Zowel voor desktop, als voor de mobile.
  + Assets met daarin de gekozen kleuren, typografie en componenten.
* Prototype
  + Een interactie ontwerp.
* Share
  + Een XD-file.
  + Een link met invite om de prototype te delen.
  + Export naar JPEG en/of PDF.

### Technisch ontwerp

Dit bestaat uit een aantal onderdelen:

#### Omschrijving AJAX en JSON

De interface tussen Vue en de backend wordt gevormd door JSON. Schrijf een document waarin je per pagina de AJAX-calls beschrijft: welk backendscript wordt aangeroepen, welke parameters worden meegestuurd, welke JSON-data worden als response verwacht?

#### Database-ontwerp

Aan de hand van het functioneel ontwerp, ontwerp je je database. Dat betekent: van elke tabel een beschrijving (naam van de tabel en namen van de velden) en van elk veld een beschrijving (type en bijzonderheden). Daarnaast een opsomming van de relaties en de aard van de relaties. En tot slot een overzichtelijke ERD (zo'n plaatje dat je met PhpMyAdmin kunt maken). Zorg ervoor dat je ontwerp wordt goedgekeurd door je docent backend.

### Database-export

Als je databaseontwerp is goedgekeurd, is het zaak om inhoud te verzamelen voor je database. Je kunt die direct invoeren in PhpMyAdmin. Zorg voor een backup van je database, want het vullen is vervelend werk. Doe dat vooral ook als je je database-ontwerp aanpast. Plaats in je documentatie ook een SQL-exportbestand.

### Configuratie

Beschrijf de benodigde hardware, software en acties om de applicatie te kunnen deployen naar een productieomgeving.

### Verslag

In je projectverslag moeten de volgende onderwerpen aan bod komen:

* Beschrijving van tenminste vier moeilijkheden (bij voorkeur alle aspecten van de applicatie bestrijkend: design, html, vue, backend) die je tegenkwam en hoe je ze te lijf bent gegaan.
* Reflectie op je planning: waarin wijkt de gerealiseerde afhandeling van je taken af van je beginplanning en waar lag dat aan?
* Reflectie op je applicatie:
  + werkt je applicatie zoals in dit document omschreven?
  + op welke deelresultaten ben je trots?
  + heb je eigen functionaliteit toegevoegd?

Daarnaast verwachten we dat je verslag een voorblad, een inleiding en een conclusie bevat. In de inleiding leg je in je eigen woorden de functionaliteit en onderliggende techniek van de applicatie uit.

## Deadline en presentatie

De deadline is vrijdag 15 januari. Dat is aan het eind van de voorlaatste lesweek van blok 2. De presentaties zijn in de week daarna, dus in de laatste lesweek van blok 2.