howest
we develop people

Academiejaar **2019-2020**
Basisopleiding Bachelor Toegepaste Informatica
Zittijd: 1 Kans: 1

Module: **Web Development**
Lector: M. Blomme, D. Casier
Nagelezen door: O. Sourie

EINDOPDRACHT – HOWEST MOVIE SHOP

De eindopdracht van deze module bestaat uit **een individueel project** waarbij men op basis van een voorbeeld een movie shop dient na te maken volgens de geziene .NET core technieken in de les.

Bekijk het nodige bronmateriaal (screencast, opgave, ...) en assimileer de gegevens om tot een goed eindresultaat te komen.
De gegeven percentages zorgen voor een indicatie van belangrijkheid van de onderdelen.

Indienen

Deze opdracht wordt ten laatste **zondag 24 mei 23:59** ingediend.

Indienen gebeurt op gitlab via je eigen gecreëerde repo.
(Zie document project-setup.pdf)

De **laatste commit op de master-branch voor de deadline** wordt aanzien als de finale versie. Er worden geen andere versies (ouder of nieuwer) aanvaard als de te verbeteren versie

Testen van de git

Test zo snel mogelijk uit of je in staat bent een commit & push uit te voeren op je eigen git repository. Indien niet meld je dit **voor 5 mei** in de voorziene monitoraten.

Testen van database scripts

Test eveneens zo snel mogelijk of je in staat bent de database op te zetten.
Indien niet meld je dit **voor 5 mei** in de voorziene monitoraten.

Belangrijk!

Dit is een **individuele opdracht.** Deels of volledig kopiëren van een andere student, plagiaat, hacking of elke andere vorm van bedrog zijn niet toegelaten. Deze zullen worden bestraft met een nulscore voor de volledige module.
GitLab ondersteunt plagiaat validatie!

Wat dien je in?
Twee repos:

- **howest-movie-shop:**
- **howest-movie-lib**

Zie verder in het document voor een uitgebreide beschrijving.

Deadline + hulp

- Start project **30 april 23:59**
- Eerste monitoraat: **4 mei**
- Tweede monitoraat **18 mei**
- Deadline: **24 mei 23:59**

Tussentijds kunnen op het teams kanaal .NET CCCP/IBC vragen gesteld worden. Deze zullen minder actief beantwoord worden vergeleken met de twee gegeven monitoraten.

Vragen via mail worden **niet** beantwoord.

OPGAVE

Bekijk de screencast voor onderstaande door te nemen. Er kunnen ten allen tijde regels worden toegevoegd, aangepast of weggelaten. Deze zullen steeds via Leho gecommuniceerd worden.

Html & css

Je vertrekt vanaf de verkregen **startbestanden** op GitLab.
https://op-gitlab.howest.be/TI/2019-2020/s4-.net-technology/cccp-ibc/exercises/project/start-bestanden

- **CSS:** in principe dien je geen CSS meer te schrijven, alle nodige stijlregels zijn aanwezig. Zo krijg je ook onmiddellijk feedback bij het implementeren van de correcte HTML elementen.

- **HTML:** het startproject bevat enkele HTML bestanden die helpen de nodige views op te bouwen. In principe hoef je geen nieuwe html meer te schrijven. Haal de gewenste stukken code uit de HTML bestanden. Hierdoor zullen de views automatisch een correcte stijl toegekend krijgen.

Git hygiëne (5% v/h eindtotaal)

- Correct gebruik van git: o.a. **voldoende frequent commits. We verwachten minimum 20 commits**
- Nuttige **commit messages**
- De **eerste commit** is het creëren van een blanko mvc project.
    - dotnet new mvc –o “NaamProject” -au Individual

Functionele vereisten

Hieronder een uitgebreide oplijsting van de minimale verwachtingen. Vergelijk met de screencast, maar let op niet alles wordt gedemonstreerd!

Lijst met movies (30% v/h eindtotaal)

- De lijst met movies wordt dynamisch samengesteld uit de aangeleverde database van movies.
- De **eerste** maal worden de movies opgehaald uit de **database**.
  Nadien worden de movies telkens opgehaald uit een sessie tijdens de duur van de sessie.
- Men kan **zoeken op een woord uit de titel** van de movie. Als hetgeen men intikt in de zoekbar ergens in de titel aanwezig is, wordt het resultaat getoond. Na het klikken op de search knop wordt de zoekopdracht uitgevoerd.

- Een blanko zoekveld == alle movies.
- Men kan de lijst verfijnen door te **filteren op properties (title, year).**
  Men kan steeds kiezen voor oplopende of aflopende sortering
- Zoeken en filteren zijn dus aan elkaar gekoppeld.
- Men kan een **movie toevoegen aan de winkelwagen**
- De movielijst wordt **weggeschreven én uitgelezen in een sessie**.
  De aantallen worden er bijgehouden.
- Men kan movies bestellen zonder in te loggen.
- Inloggen dient enkel te gebeuren om de bestelling definitief te maken (zie verder).
- Men kan slecht **één** exemplaar van een film bestellen.

Detail van een movie (10% v/h eindtotaal)

- Bij het **klikken op de afbeelding** van een movie ordt een detailpagina getoond.
- **De detail pagina bevat de volgende informatie:**
  - Afbeelding van de movie
  - Rating
  - Genre
  - Year of release
  - Plot
  - Alle **acteurs** (geen andere rollen)
- Men kan het movie an hieruit ook **toevoegen aan de winkelwagen**

Winkelwagen (30% v/h eindtotaal)

- Wanneer een element wordt toegevoegd aan de winkelwagen is **er visuele feedback** door de badge counter te updaten
- In geval van nog **geen items aanwezig** wordt een link voorzien (Please order something to continue) die verwijst naar de hoofdpagina.
- In geval men **nog niet ingelogd is** wordt een login link voorzien die verwijst naar de hoofdpagina.
- Toegevoegde items worden opgelijst met hun **stukprijs**. Alternerende kleuren zorgen voor leesbaarheid. Er wordt een **eindtotaal** voorzien.
- Bij het verlaten van de pagina (sluiten van de browser) en terug openen is de winkelwagen opgeslagen. **Men is zijn gegevens niet kwijt!**
- Het bestelproces bestaat **uit 3 stappen**. Indien men tijdens het proces het venster sluit, opent het opnieuw op **de eerste stap**.
- **De naam, leveringsadres en betaalmethode** worden opgevraagd voor het voltooien van de bestelling.
- Er wordt een bevestiging getoond bij het vervolledigen van een bestellingsaanvraag. Deze bevat **de naam van de persoon, en de betaalwijze.**
- Tijdens het ingeven van de persoonlijke informatie is de badge counter niet meer zichtbaar in de menubar.
- Wanneer een bestelling afgerond is moet het winkelwagentje terug leeg zijn.

Movie of the day (5% v/h eindtotaal)

- Bij het laden van de pagina **wordt willekeurig** een movie uit de lijst getoond.
- De **afbeelding** en de **titel** worden telkens aangepast.

- Bij het **laden** van dezelfde pagina is er telkens een andere movie zichtbaar.

Orders pagina (10% v/h eindtotaal)

De users met **admin**rechten hebben een extra **orders** link bovenaan in de menu bar. Deze orders pagina bevat de volgende informatie:

- Een overzicht van alle orders.
- Ieder order records bevat:
  - Order id
  - Order date
  - Movie titles
  - TotalPrice
  - Customer name
  - Address
  - Delete knop.
- De admin moet een volledig record kunnen verwijderen.

Inloggen/registreren (5% v/h eindtotaal)

- De menubar bevat ten alle tijden inlogmogelijkheden.
- Men moet kunnen inloggen/uitloggen/registreren.
- Eveneens kan men inloggen/registreren met een Google account.
- Een geregistreerde klant heeft automatisch de rol **member.**
- Na het uitloggen wordt de data in de sessie verwijderd.

Varia (5% v/h eindtotaal)

- Teller van het aantal movies op de homepage
- Algemene code quality, structuur, afwerkingsgraad, …
- Uitwerking MoviePriceSeeder

Technische vereisten

Concreet bestaat jullie oplossing uit **twee project.**
Beide projecten hebben elk hun eigen repo op GitLab.

1. **howest-movie-shop:** dit bevat de webapplicatie en maakt gebruik van de **howest-movie-lib** om de database te bevragen.

2. **howest-movie-lib:** dit bevat allerhande nuttige services om de database te bevragen.

Hieronder een uitgebreide oplijsting van technische verwachtingen.

Database

- Als database heb je de keuze uit **Postgress** of **SqlServer**. Voor beide zijn er scripts voorzien. Maak gebruik van de aangeleverde SQL scripts om een database op te zetten.
- Er worden 2 scripts aangeleverd:
  - **movies.sql:** structuur **en** data van de movie gerelateerde tabellen
  - **shop.sql:** structuur van de shop gerelateerde tabellen

- Er zijn ook afbeeldingen voorzien, zie folder **images**.
  De tabel **movies** bevat een veld **cover_url**, waarin de **naam** van afbeelding staat, dus **niet het volledig pad**.

Entity framework

- Het entity framework moet gebruikt worden om de database te consumeren.

Services

- Het bevragen van de dbContext gebeurt enkel en alleen in service klasses.
- Deze services (uitzendering sessionService) bevinden zich allemaal in het library project.

Library

- **Alle communicatie** met de database gebeurt in een library.
- Maak hiervoor de nodige services aan zoals gezien in de les.
- Creëer een duidelijke folder **Library** met daar in de volgende mappen:
    - **Services**: per gebruikte tabel één service.
    - **Models**: automatisch gescaffold

Handlers

- Om teveel aan code te vermijden in de controllers maak je voor elke controller **één handler** aan.
- Dit is een gewone klasse in folder **Library/Handlers.**
- Een handler maakt gebruik van de services om de gewenste data te verzamelen.

Controllers

- Een controller maakt **geen** gebruik van services maar enkel en alleen van zijn **uniek** toegekende **handler.**
- Concreet heb je de volgende controllers nodig:
    - MovieController = hoofdpagina
    - ShoppingCartController = shopping cart pagina's
    - OrderController = orders pagina voor de admin.
    - DetailsController = detailpagina voor een movie.

ViewModels

- Database models worden niet in de **controllers** of **views** gebruikt!
- Data afkomstig van formulieren of data in views worden enkel via **viewmodels** doorgegeven/ontvangen.
- De **handler** voorziet hierbij functies om database modellen om te zetten in viewmodels.

Sessions

- De applicatie moet gebruik maken van een sessie.
- De sessie gaat niet verloren bij het afsluiten van de pagina.

- Maak zelf een session service aan in het project **movie-shop** in de folder **Library/Services** om alle functies m.b.t. tot sessie te centraliseren.
- De volgende gegevens worden bijgehouden:
    - Lijst van movies
    - De opgezochte query + sorteer criteria
    - Ordered movies.
- Data afkomstig van formulieren of data in views worden enkel via **viewmodels** doorgegeven.
- De **handler** voorziet hierbij funties om database modellen om te zetten in viewmodels.

Master/View sections

- Maak voor alle shoppagina's gebruik van een LayoutShop master.
- Je hoeft de Layout van het **authenticatie** gedeelte **NIET** aan te passen of te wijzigen naar de LayoutShop master.
- De LayoutShop master bevat op z'n minst alle onderdelen die op elke pagina terugkomen.
    - Menubar + footer
- **Tip:** het loginmenu: <partial name="_LoginPartial" />

Identity

- De applicatie moet gebruik maken van Authenticatie en Autorisatie met Identity.
- Na het uitvoeren van de migrations worden de Identity tabellen aangemaakt.
- De **AuthSeeder** wordt uitgevoerd als de applicatie start, dus na een dotnet run.
- De applicatie bevat twee rollen, **admin** en **member**
- De **admin@howest.be** user wordt met de seeder voorzien in de Identity User tabel.
- Na het registeren krijg je de rol **member** toegekend.

OAuth

- De applicatie moet gebruik maken van de Google sign-in functie.
- Inloggen/registreren met Google is mogelijk.

Seeders

- De applicatie bevat 2 seeders: **AuthSeeder** en **MoviePriceSeeder**
- De seeders moet worden uitgevoerd als de applicatie start (tip: Startup.cs)
- **De AuthSeeder** is volledig uitgewerkt en voegt een admin user, **admin@howest.be**, toe met het paswoord **Test123!** en ook 2 rollen: **admin** en **member**.
- **De MoviePriceSeeder** is een **NIET** uitgewerkte seeder klasse en **moet worden uitgewerkt**. Zorg ervoor dat er tijdens de start van de applicatie (na een dotnet run) telkens nieuwe **random prijzen** voor de movies worden ge-insert in de tabel **shop_movie_price**. Elke movie heeft 1 prijs. De tabel **shop_movie_price** bevat dus 250 records. Met 250 het aantal movies.