Opdracht: Kerst- en Wenslijst in Maui

In deze opdracht bouw je een .NET MAUI-app die zowel een persoonlijke wenslijst als een kerstlijst voor anderen beheert.

Je past de concepten toe die we in de lessen gezien hebben: MVVM, services, dependency injection, Shell-navigatie, repositories, enzovoort.

De app die je zal maken is bedoeld om een wenslijst bij te houden, alsook om een lijst van kerstgeschenken bij te houden voor anderen. Alle gegevens die je invoert (personen, wishlist-items en kerstlijst-items) moeten lokaal worden opgeslagen in een database

Je implementeert zowel de functionaliteit als een duidelijke, gebruiksvriendelijke UI. Omdat dit een opdracht is voor het vak softwareontwerp, ligt de focus op het onderliggende ontwerp en de kwaliteit van je implementatie. Een verzorgde UI is zeker een pluspunt, maar bepaalt niet de score.

De meegeleverde screenshots zijn louter illustratief. Je mag hier volledig van afwijken, zolang alle vereiste functionaliteit aanwezig blijft en de app logisch en consistent opgebouwd is.

De opdracht vereist dat de MAUI-app correct werkt op Windows. Je mag de app ook laten functioneren op Android of iOS (altijd leuk om je eigen app op je telefoon te zien), maar voor de eindbeoordeling telt uitsluitend de Windows-versie.

Werkt de app niet op Windows, dan kan de opdracht niet geëvalueerd worden en volgt automatisch een onvoldoende.

Opstartscherm

Dit is het eerste scherm dat de gebruiker ziet wanneer de app opstart.

Het bevat minstens twee knoppen: één die naar de Kerstlijst navigeert en één die naar de Wenslijst gaat.

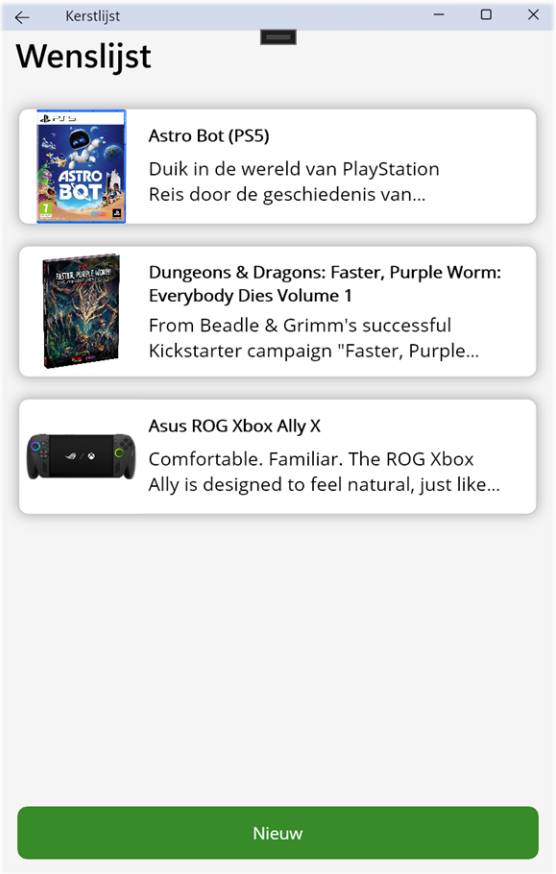
Hoe je dit scherm visueel vormgeeft, kies je zelf. Hou wel de geziene principes in het achterhoofd: geen events of logica in de code-behind. De volledige app, inclusief het startscherm, volgt het MVVM-patroon:

Wenslijst

De wenslijst zijn de dingen die je leuk zou vinden om te ontvangen als cadeau.

Wanneer je op het hoofdscherm op de knop ‘Wenslijst’ klikt, navigeert de app naar een scherm met een lijst (‘CollectionView’) van deze items.

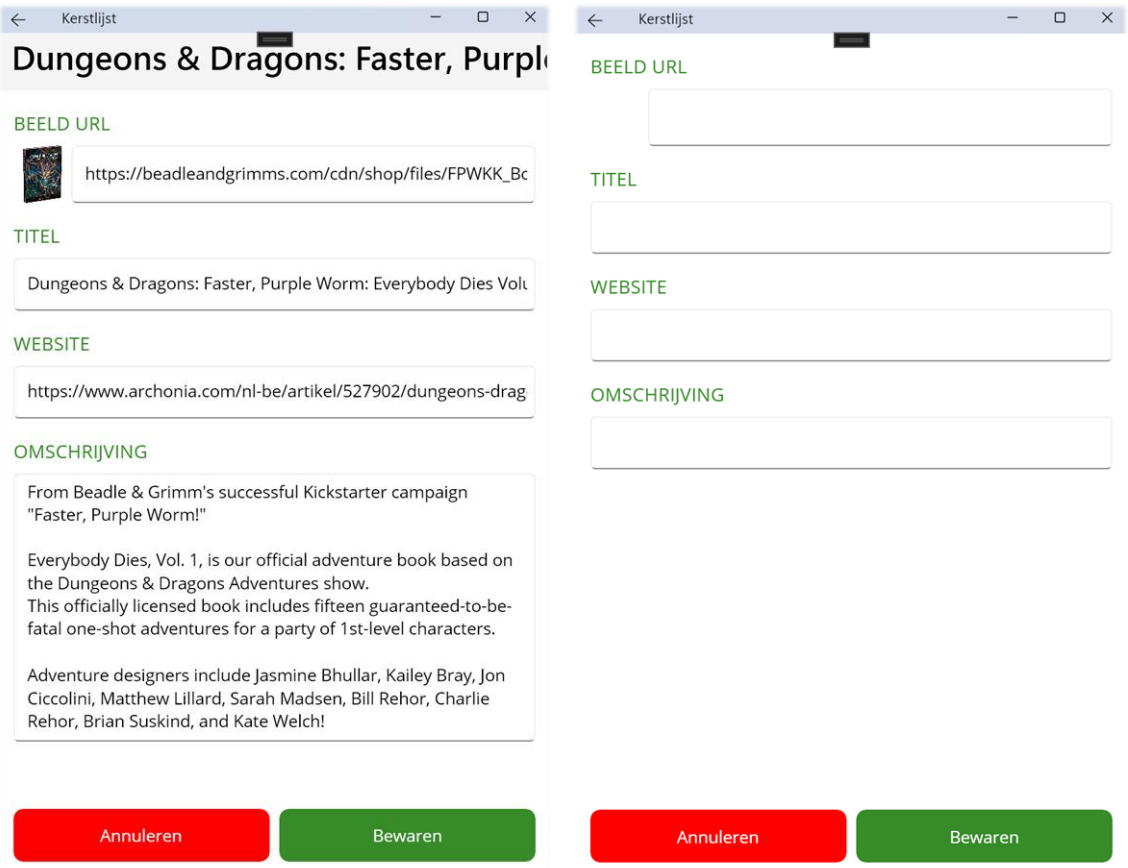
Per item zien we minstens het beeld, de titel en maximum 2 lijnen van de beschrijving. Dit scherm zou er als volgt kunnen uitzien:



Wanneer je op een item in de lijst klikt, navigeert de app naar een scherm waar de details van het gekozen wenslijstitem worden getoond.

Op dit scherm staat ook een knop ‘Nieuw’.
Wanneer je daarop klikt, opent hetzelfde detailscherm, maar dan zonder vooraf ingevulde gegevens, zodat je een nieuw wenslijstitem kunt toevoegen.

Onderstaande voorbeelden tonen hoe deze schermen eruit kunnen zien:



Wanneer het scherm een bestaand item toont, wordt de titel van dat item gebruikt als schermtitel (in de balk bovenaan).

Op het detailscherm zijn minstens de volgende velden beschikbaar:

1. *Titel*

   Dit is een tekstveld, de titel van het wens item. Dit is een verplicht veld.

2. *Website*

   Een tekstveld met de link naar meer informatie of een webshop. Ook dit is een verplicht veld.

3. *Omschrijving*

   Een groter tekstvak waarin meerdere regels tekst kunnen worden ingevoerd voor extra toelichting.

4. *Beeld Url*

   Dit zijn 2 velden samen: een tekstveld waar de gebruiker kan een url kan ingeven van een beeld met ervoor een image veld waar men het beeld direct kan tonen.

Wanneer de gebruiker op de knop ‘Bewaren’ klikt, dan moet er eerst gevalideerd worden of alle gegevens geldig zijn (bv. of alle verplichte velden ingevuld zijn)

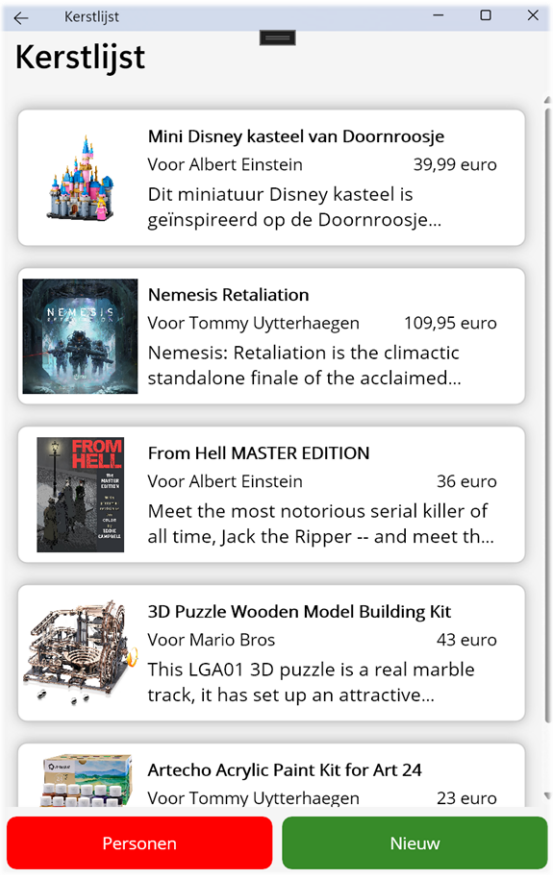
Kerstlijst

De kerstlijst bevat de cadeaus die je effectief zal geven, inclusief aan wie ze bedoeld zijn.

Wanneer je op het hoofdscherm op ‘Kerstlijst’ klikt, navigeer je naar een overzicht (‘CollectionView’) met alle kerstlijstitems — de cadeaus die je al hebt gekocht of nog van plan bent te kopen.

Per item zien we minstens het beeld, de titel, voor wie het is, de prijs en maximum 2 lijnen van de beschrijving.

Dit scherm zou er als volgt kunnen uitzien:

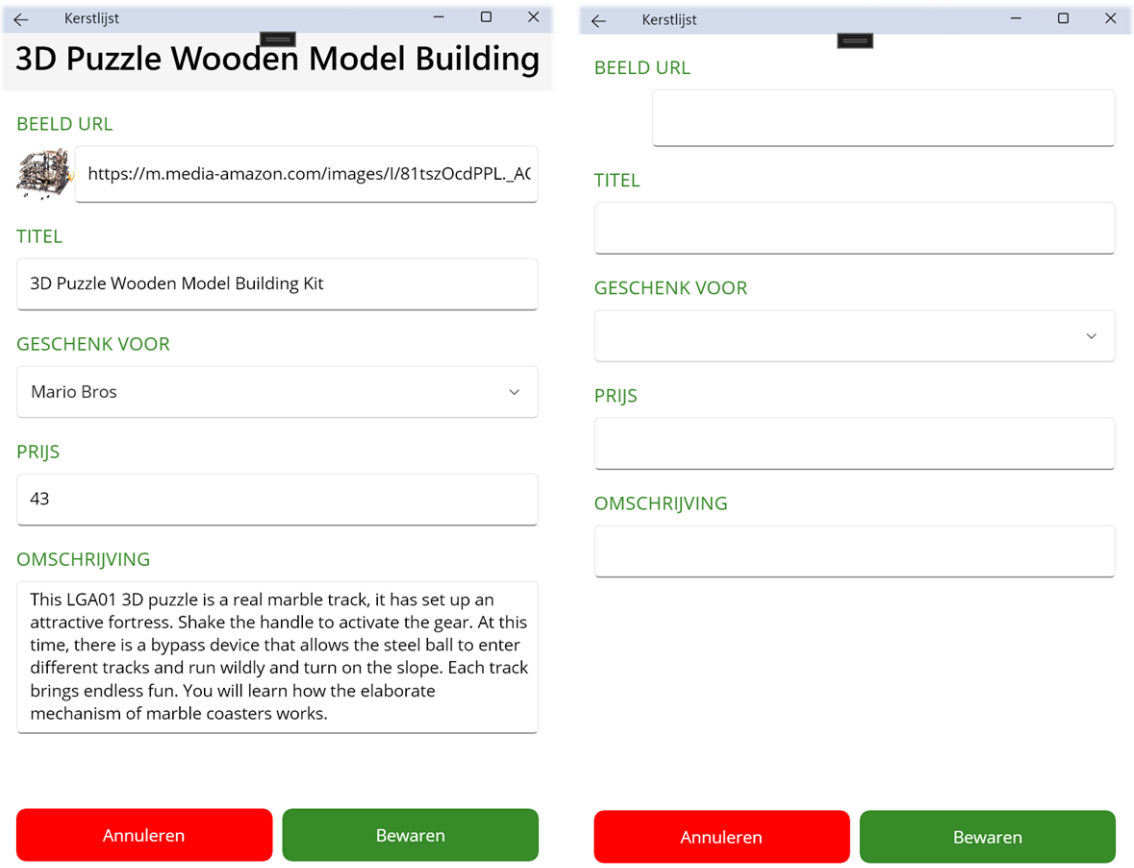


Op dit scherm bevinden zich ook 2 knoppen:

Knop ‘Nieuw’

Wanneer je op deze knop klikt, opent het scherm om een nieuw kerstlijstitem toe te voegen. Dit kan hetzelfde scherm zijn als het detailvenster, maar dan zonder vooraf ingevulde gegevens.

Dit scherm zou er als volgt kunnen uitzien:



Wanneer het scherm een bestaand item toont, wordt de titel van dat item gebruikt als schermtitel (in de balk bovenaan).

Op dit detailscherm zijn minstens de volgende velden beschikbaar:

1. *Titel*

   Een tekstveld met de naam van het kerstlijstitem. Dit is een verplicht veld.

2. *Prijs*

   Een tekstveld dat enkel een geldige prijs mag bevatten (of leeg mag zijn).
   Bevat het een ongeldig formaat of een negatief getal, dan mag het item niet bewaard worden.

3. *Omschrijving*

   Een groter tekstvak waarin meerdere regels tekst kunnen worden ingevoerd.

4. *Beeld Url*

   Dit zijn 2 velden samen: een tekstveld waar de gebruiker kan een url kan ingeven van een beeld met ervoor een image veld waar men het beeld direct kan tonen.

5. *Geschenk voor*

   Dit is een selectie lijst (‘Picker’) waar de gebruiker een lijst ziet met alle personen geregistreerd in het systeem (via de knop ‘Personen’ op het Kerstlijst scherm.
   Dit veld mag leeg blijven.

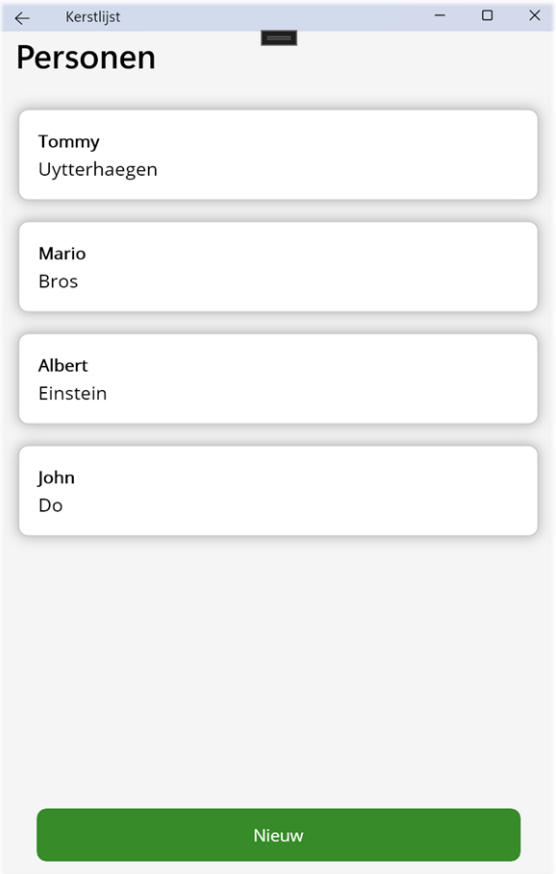
Wanneer de gebruiker op de knop ‘Bewaren’ klikt, dan moet er eerst gevalideerd worden of alle gegevens geldig zijn (bv. of alle verplichte velden ingevuld zijn, of velden bevatten wat verwacht is, …)

Knop ‘Personen’

Dit opent de lijst met personen waarvoor kerst cadeautjes gekocht worden.

Wanneer er op de kerstlijst op de knop ‘Personen’ geklikt wordt navigeert de app naar een scherm waar men een lijst (‘CollectionView’) ziet van personen.

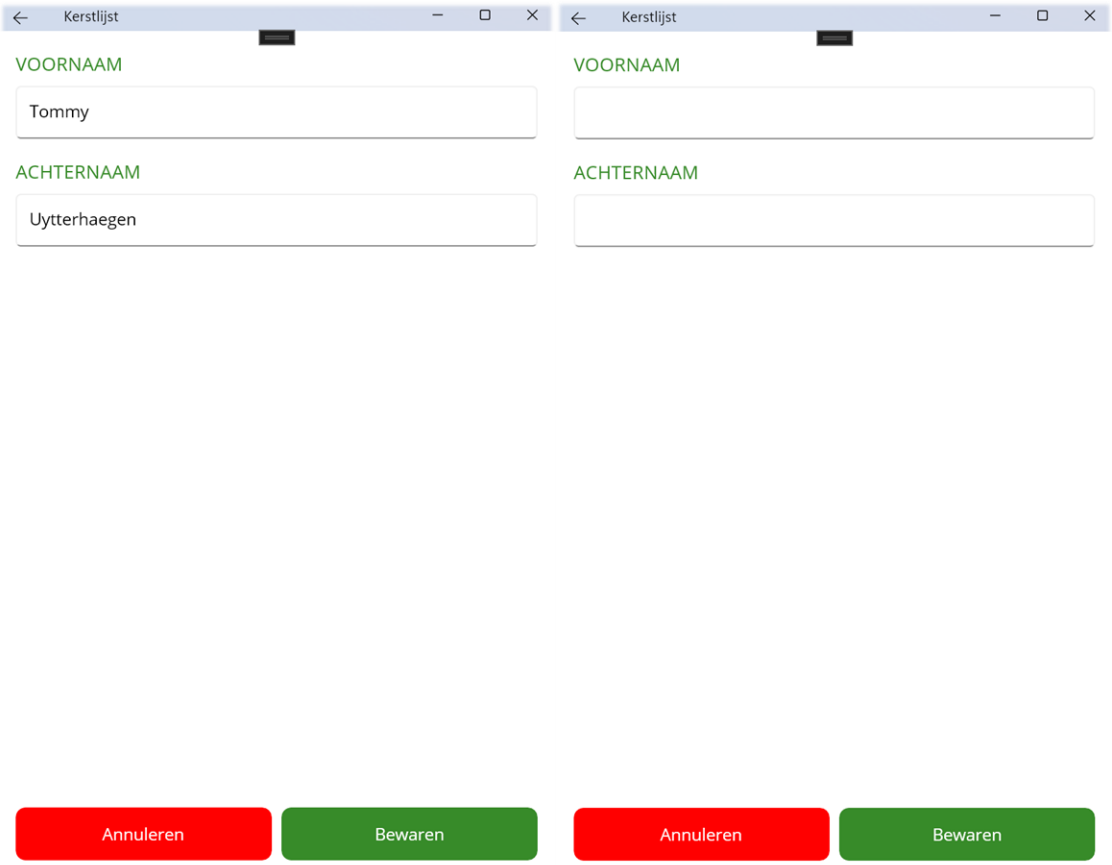
Per persoon zien we minstens de voornaam en de achternaam. Dit scherm zou er als volgt kunnen uitzien:



Wanneer je op een persoon in de lijst klikt, navigeert de app naar een scherm waarin de details van die persoon worden weergegeven.

Op dit scherm staat ook een knop ‘Nieuw’.
Wanneer je daarop klikt, opent hetzelfde detailscherm maar dan zonder vooraf ingevulde gegevens, zodat je een nieuwe persoon kunt toevoegen.

Hieronder zie je voorbeelden van hoe deze schermen eruit kunnen zien:



Wanneer het scherm een bestaande persoon toont, wordt de voornaam gebruikt als schermtitel (in de balk bovenaan).

Op dit detailscherm zijn minstens de volgende velden beschikbaar:

1. *Voornaam*

   Dit is een tekstveld, de voornaam van de persoon. Dit is een verplicht veld.

2. *Achternaam*

   Dit is een tekstveld, de familienaam van de persoon.

Wanneer de gebruiker op de knop ‘Bewaren’ klikt, dan moet er eerst gevalideerd worden of alle gegevens geldig zijn (bv. of alle verplichte velden ingevuld zijn)

Technische vereisten

Je past de architectuurprincipes toe die we in de lessen gezien hebben. Concreet betekent dit dat je werkt met:

- ViewModels
- Data- en command-bindings, met INotifyPropertyChanged waar nuttig
- Inversion of Control & Dependency Injection
- Navigation service in combinatie met IQueryAttributable
- Services, repositories, …
- Een 3-lagen-architectuur (UI, BL, DL)

De app moet zonder fouten compileerbaar zijn en correct werken op Windows.

SoftwareONTWERP

Hoewel de app duidelijk moet werken, ligt de echte klemtoon van deze opdracht op softwareontwerp. Je toont dat je:

- het domein correct en zinvol modelleert in een klassendiagram (personen, wishlist-items, kerstlijst-items, …)
- verantwoordelijkheden kan verdelen over meerdere klassen (geen GOD-klassen)
- patronen en principes inzet om de structuur uitbreidbaar, flexibel en begrijpbaar te houden

Hoe je dat aanpakt, is aan jou. Je zult onder meer moeten nadenken over:

- hoe je schermen structureert en organiseert (startscherm, personenbeheer, wishlist, kerstlijst, detailpagina's, …)
- hoe je het domein modelleert zodat het logisch en toekomstbestendig blijft (bv. personen die gelinkt zijn aan kerstlijst-items)
- hoe je objecten aanmaakt zonder overal new te gebruiken (factory/service/repository, dependency injection, …)
- hoe je input behandelt en validatie organiseert binnen MVVM (bindings, commands, INotifyPropertyChanged)
- hoe je data opslaat, ophaalt en beheert zonder dat code verspreid en duplicatief wordt (services, repositories, eventueel facades)
- hoe je navigatie opbouwt zodat het leesbaar en onderhoudbaar blijft (Shell + NavigationService + IQueryAttributable)

Patronen (Patterns)

Je mag patronen gebruiken die we in de lessen gezien hebben (mvvm, IoC, DI, facade, service, repository, factory, …), alsook anderen die je kent en relevant vindt.
Hou in het achterhoofd dat het geen checklist is. Forceer er geen patronen in, maar kies degenen die nuttig zijn om flexibiliteit, uitbreidbaarheid en duidelijkheid te ondersteunen.
Aan het einde moet je kunnen uitleggen waarom jouw ontwerp eruitziet zoals het eruitziet, waarom bepaalde klassen bestaan, en waarom die op die manier gekoppeld zijn. Een ontwerp is pas goed als jij het kan verdedigen.

Bekijk zeker het eindresultaat van de 'joke' oefening die we gemaakt hebben in de les.
Kijk, **begrijp** en **pas** de gebruikte concepten **toe**.
Er zijn een aantal dingen waar elke Maui app gebruik van maakt zoals MVVM, IoC, DI, NavigationService, Services, Repository, … ook deze.

Inleveren

Je

- vult het formulier in (https://forms.cloud.microsoft/e/7mbRFJNGB7)
- zet je app + diagram op **GitHub**
- geeft me (**tommy.uytterhaegen@hogent.be**) **toegang**
- stuurt de GitHub link, nogmaals, als inzending op de opdracht

Hier wil ik een werkende Maui app terugvinden, op een leesbare & logische manier, alsook een diagram van je ontwerp. Zorg ervoor dat jouw code & ontwerp met elkaar kloppen …

Heel **belangrijk** is ook dat je beknopte commentaar zet bij relevante code waarin je je keuzes toelicht. Als je er niet in slaagt je keuze uit te leggen, wil dit meestal zeggen dat je deze niet bewust gemaakt hebt.

Opgelet: Tijdens de mondelinge verdediging moet je kunnen aantonen dat je jouw oplossing zelf hebt opgebouwd en de onderliggende concepten begrijpt, alsook eventuele niet gebruikte concepten uit de les.

Als blijkt dat je je oplossing niet zelf hebt opgebouwd, dan zal dit **automatisch** leiden tot een **onvoldoende**.

Het doel van de verdediging is net om te evalueren of je de geziene stof beheerst, overtuig me dus dat jij het ontwerp gemaakt hebt en de implementatie begrijpt.

O.M.G. Dit is zo tof!

Extra features zijn welkom, maar mogen nooit ten koste gaan van een helder ontwerp. Start vanuit de vraag: “Hoe ontwerp ik een structuur die uitbreidbaar blijft?” en niet vanuit “Hoe krijg ik dit zo snel mogelijk aan de praat?”

TIP: Bekijk (en begrijp) de Joke-app oplossing die we hadden op het einde van les 3 (https://github.com/tommy-uytterhaegen/HoGent)