

# OOAD – ActiBuddy



versie van dit document: 24 juni 2024

## A. Doelstellingen en context

Het project draait rond een eenvoudige toepassing, waarmee personen activiteiten kunnen voorstellen waaraan anderen kunnen deelnemen. Er zijn **personen** (admins en gebruikers), **activiteiten** (categorieën: sportief, cultureel en hobby) en **deelnemers** (i.e. een deelnemende gebruiker aan een activiteit). Het bestaat uit drie deelprojecten:

- een **class library** met alle klassen
- een **admin WPF** toepassing (beheer personen en activiteiten)
- een **customer WPF** toepassing (beheer eigen workouts en enkele statistieken/grafieken)

Dit project is niet zomaar een herhalingsoefening of integratie van geziene leerstof, maar een *vervolg* er op. Het is als het ware de tweede helft van het vak. Er komen heel wat nieuwe technieken aan bod die je – met behulp van onze tips, video's, links en codefragmenten – zelfstandig zal moeten uitzoeken. De voornaamste zijn:

- werken met een **SQL databank** in C#
- aanmaak en gebruik van een **class library**
- lezen en schrijven van **afbeeldingen** in een databank
- paswoord **hashing**
- werken met **Frame / Page**
- items **dynamisch toevoegen** aan een panel control
- gebruik van **nieuwe WPF controls** (Password, Calendar...)
- gebruik van **NuGet packages** ([GMap.NET](#))

Het is aan jou om met het project te bewijzen dat je deze technieken zelfstandig kan bestuderen en combineren met geziene technieken tot een volwaardige toepassing.

Voorgesteld stappenplan:

- zorg ervoor dat je alle onderdelen van de cursus goed onder de knie hebt, en zeker WPF panels en classes
- bestudeer grondig het hoofdstuk [10. SQL Databanken](#) uit de C#-Cursus; ga niet verder voor je **SInCompany** door en door begrijpt
- bestudeer **SInDemoFrame** en bekijk eventueel tutorails op Youtube zoals [deze video](#) over hoe je een WPF toepassing maakt met verschillende pagina's
- bestudeer **SInDemolItemsPanel** over hoe je items dynamisch vanuit C# in een panel control laadt
- installeer de databank en bestudeer de **ERD**; zie dat je alle velden goed begrijpt

## B. Wijzigingen t.o.v. eerste zit

Er zijn een paar wijzigingen t.o.v. het project van EP1:

1. de **databank wordt zonder data geleverd**, je krijgt enkel de structuur; vul de databank zelf met voldoende relevante en realistische data (i.e. minstens 2 admins, minstens 4 gebruikers, minstens 30 activiteiten verspreid over de drie categorieën en 50 deelnemers)
2. je moet bijgevolg een **SQL dump (schema + data)** toevoegen aan je project, anders kan het niet verbeterd worden
3. je moet van alle **chatGPT chats een pdf dump** maken en bij je oplossing steken
4. er wordt **geen technische ondersteuning** geboden; je kan uiteraard vragen stellen over de inhoud en interpretatie van de opgave, maar vragen over codefouten e.d. worden niet beantwoord

## C. Evaluatie

Het eindcijfer wordt berekend op basis van twee deelcijfers: de **geïmplementeerde functionaliteit** op 10 punten ("F10") en de **technische uitvoering** op 10 punten ("T10"):

$$\text{eindcijfer} = (\text{T10} * \text{F10} * 2 / 5 + \text{T10} + \text{F10}) / 3$$

De achterliggende gedachte is dat je onmogelijk kan geslaagd zijn als er bijna niks uitgevoerd is ( $\text{F10} = 0$ ), en evenmin als je alles uitgevoerd hebt maar technisch alles fout is. Enkele cijfervoorbeelden:

F10	T10	eind
0	10	3,3
10	0	3,3
5	5	6,7
4	8	9,2
6	6	9,6
6	8	11,8
8	7	13,1
9	9	17,1
10	10	20,0

Moraal van het verhaal: je scoort **zowel functioneel als technisch best meer dan 6/10** om te kunnen slagen.

Er zijn een aantal **red flags** in het rood; als je aan één van deze punten flagrant niet voldoet, krijg je sowieso 0/20 voor het hele project. Enkele red flags vooraf:

- de **twee** deelapps (WpfAdmin en WpfCustomer) van het project moeten min of meer **werkend** en **functioneel** zijn; laat je één van de deelapps geheel of bijna geheel weg, dan kan je niet meer slagen
- je maakt enkel gebruik van **geziene technieken en controls uit de C#-Cursus**, dus geen databinding, geen DataGridView/GridView/ListView/..., geen transacties, geen DataTable, geen User Controls, geen ExpandoObject, geen EventHandler<T> of Action<T>...; indien je hier toch van wil afwijken, vraag eerst toestemming aan de docent

- je **begrijpt elke regel code** die je schrijft en vermeldt duidelijk waar je welke **bronnen** gebruikt hebt; code die je niet kan uitleggen op een mondeling, wordt als niet uitgevoerd beschouwd
- **exporteer alle chatGPT chats** met de [Save ChatGPT as PDF plugin](#) en steek die bij je oplossing
- je werkt op **Github** en commit zeer regelmatig, minstens één maal per gewerkte dag en minstens 5 in totaal; een oplossing die plots uit de lucht komt vallen wordt niet geaccepteerd
- je **uploadt** je code tijdig op Toledo

We bekijken nu de details van de technische en van de functionele evaluatie.

## Technische evaluatie (10pt)

XAML (2pt):

- de WPF controls zijn mooi uitgelijnd, het geheel **oogt verzorgd**
- de XAML code is correct opgebouwd met **WPF panels** zoals gezien in les04 (tip: vergroot/verklein de Window en controleer of alles er ok uit blijft zien)
- er is effectief gebruik gemaakt van **Page/Frame**

Class library en SQL (4pt):

- er is gebruik gemaakt van een **class library**
- het gebruik van **classes** is over de hele lijn **verplicht**
- in de **WPF app(s) komt nergens SQL code** voor; alle databank communicatie gebeurt via de class library
- er is **niet zomaar overal static** gebruikt; er is oordeelkundig gekozen tussen static / non-static
- de classes zelf **bevatten alle zinvolle methodes** (CRUD e.d.), er is zeker geen gebruik gemaakt van aparte Repository classes!
- de **classes zijn correct opgebouwd** (static / non-static, property vs methode, object returnen, plaatsing in juiste klasse, gebruik properties boven methodes...)
- de **visibility** (public, internal, protected, private) is overal zo beperkt mogelijk
- in de **class library komt nergens WPF code** voor (geen MessageBox, WPF controls...)
- de **connectiestring staat enkel in App.config** en is exact  
Data Source=(localdb)\mssqllocaldb;Initial Catalog=ActiBuddyDB;Integrated Security=True

Code (4pt):

- de **StyleCop** analyzer is correct geïnstalleerd en geeft geen grote fouten
- er is aandacht voor **code kwaliteit** (denk aan DRY, niet te diepe nesting, compacte en logische code...)
- de code bevat van voldoende **commentaar**
- **exception handling** is toegepast (vuistregel: zo laat mogelijk, dus standaard in de WPF app, niet in de class library)
- **overerving** is toegepast

## Functionele evaluatie (10pt)

De gedetailleerde uitleg van het gevraagde per onderdeel vind je in het volgende onderdeel; hier alvast een overzicht van de puntenverdeling:

Logins (1pt):

- **inloggen** werkt
- **redirect** naar de correcte pagina
- **ingelogde gebruiker** wordt correct bewaard
- **password hashing** is toegepast
- **formchecking** is toegepast
- **foutmelding bij foutief inloggen** (geen MessageBox!)
- **uitloggen** werkt

Admin (3pt):

- **beheer gebruikers** (overzicht, details, toevoegen, verwijderen, bewerken – 1pt)
- **beheer activiteiten** (overzicht, details, verwijderen – 2pt)

Gebruiker (6pt)

- **activiteitenkaart** (GMap kaart met activiteiten, zoekfunctie en filters, mogelijkheid in- of uit te schrijven – 3pt)
- **beheer eigen activiteiten** (overzicht, toevoegen, bewerken, deelnemers verwijderen, activiteit verwijderen – 3pt)

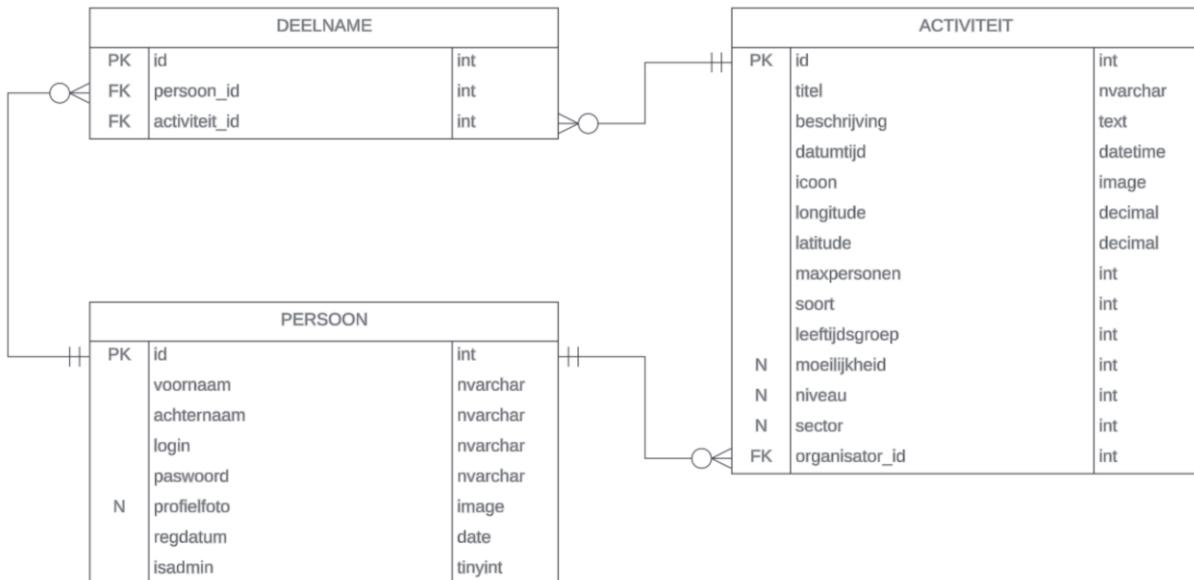
Op onderdelen die **crashen of niet builden** of werken worden *geen punten* gegeven. Test daarom je toepassing grondig op fouten.

# D. Voorbereiding

## D.1. Databank opzetten

### Schema importeren

De databank is meegegeven met de opgave als *ActiBuddyDB.sql*. Overloop alle stappen uit , zie [10.SQL databanken deel 3](#) om de databank te installeren. Het ERD:



PK = primary key, FK = foreign key, N = nullable

Uitleg over de tabel ACTIVITEIT:

- er zijn drie types activiteiten: sport, cultuur en hobby; het onderscheid wordt gemaakt met *soort* veld (1 = sport, 2 = cultuur, 3 = hobby)
- voor elke activiteit moet een icoon opgegeven zijn; dit wordt later gebruikt op de kaart
- drie velden zijn niet op alle types van activiteiten van toepassing, en kunnen dus null zijn
  - *moeilijkheid* (0 = n.v.t., 1 = makkelijk, 2 = gemiddeld, 3 = zwaar): enkel voor sport
  - *niveau* (0 = n.v.t., 1 = beginner, 2 = halfgevorderd, 3 = gevorderd): enkel voor hobby
  - *sector* (0 = n.v.t., 1 = muziek, 2 = theater, 3 = dans, 4 = andere): enkel voor cultuur

Uitleg over de tabel DEELNAME:

- bestaat uit één gebruiker die ingeschreven is voor één activiteit

Uitleg over de tabel PERSOON:

- het bevat zowel admins als gebruikers; het onderscheid wordt gemaakt met het *isadmin* veld (1 = admin, 0 = gebruiker) – enkel admins kunnen inloggen op het admingededeelte, enkel gebruikers kunnen inloggen op het gebruikersgedeelte
- de paswoorden zijn SHA256-geëncrypteerd (zie verder), onleesbaar voor hackers.

## Data aanmaken

Tabellen Activiteit en Deelname zijn leeg; je zult ze dus zelf met data moeten vullen.

- *Tabel Persoon:* de records uit de Fitness databank van het vorige project zijn reeds toegevoegd, maar **verander achteraf de foto's**, want die zijn bedoeld voor een Fitness toepassing en kloppen niet echt met deze opgave. Neem realistische foto's.
- *Tabel Activiteit:* voorzie minstens 30 activiteiten van verschillende soorten (sport, cultuur en hobby); beperk je om het verbeteren makkelijker te maken tot activiteiten **in augustus 2024** en verspreid **binnen het Brussels Gewest**
- *Tabel Deelname:* minstens 50 ingeschreven deelnemers op verschillende activiteiten

Alle demodata moet **realistisch** zijn, dus geen blablabla, test123 enz...

## D.2. Solution setup

Maak een nieuwe lege solution met de naam **SInActiBuddy**. Maak daarin drie projecten aan:

1. een WPF App (.NET8) met de naam **WpfAdmin**
2. een WPF App (.NET8) met de naam **WpfUser**
3. een Class Library (.NET8) met de naam **CLActiBuddy**.

Het is zeer belangrijk dat de projecttypes (.NET8) exact kloppen, want anders zullen een aantal zaken niet of anders werken.

Voeg ook de StyleCop.Analyzers package toe aan de projecten. Voeg ook de correcte .editorConfig toe aan je solution, meegeleverd met de opgave. Voeg tenslotte de volgende GMap NuGet packages toe voor de kaart:



**GMap.NET.Core** by Jurgen De Leon Rodriguez

2.1.7

GMap.NET Windows Forms, Presentation & Avalonia is an excellent open source, powerful, free and cross-platform .NET control. Allows the use of routing, geocoding, directions and maps from Google, Yahoo!, Bing, Ope...



**GMap.NET.WinPresentation** by Jurgen De Leon Rodriguez

2.1.7

GMap.NET Windows Forms, Presentation & Avalonia is an excellent open source, powerful, free and cross-platform .NET control. Allows the use of routing, geocoding, directions and maps from Google, Yahoo!, Bing, Ope...

## D.3. Connection string

Je mag tijdens de ontwikkeling je eigen connectiestring gebruiken, maar vóór je het uploadt/pusht:

```
Data Source=(localdb)\mssqllocaldb;Initial Catalog=ActiBuddyDB;Integrated Security=True
```

Voeg het toe aan de App.Config van beide WPF toepassingen, zie [10.SQL databanken deel 5](#). Probeer gerust al eens een eenvoudige SELECT query uit om te testen.

## D.4. Class library

We volgen nu alle stappen uit [10. SQL Databanken deel 6 en 7](#).

- maak klassen Activiteit, Deelname en Persoon
- voeg alle nodige properties toe
- voeg alle nodige methodes en constructors toe
- voeg enumeraties toe (b.v. voor soort activiteit: sport, cultuur of hobby)

Aandachtspunten:

- let op de aggregaties, zie [10.SQL databanken deel 7.2](#)
- alle relevante methodes en properties voor de klassen moeten in de klassen zelf staan, dus geen aparte klassen als PersonRepository, DbLayer enz...!
- denk goed na over keuze static methode / objectmethode / readonly property; over het algemeen is de volgorde van voorkeur:  
**readonly property > objectmethode > statische methode**

Als je alles uit gemak statisch maakt, zul je niet veel punten krijgen voor je class library!

## E. WpfAdmin

Deze app bestaat uit drie onderdelen:

- loginpagina
- beheer personen (gebruikers en admins)
- beheer activiteiten

Er worden in tweede zit geen screenshots of mockups gegeven! Inspireer je op de mockups uit de Fitness opgave in eerste zit en lees goed de beschrijving.

### E.1. LoginWindow

De app start op met een login-scherm. Na succesvol inloggen wordt het hoofdvenster geopend. Een korte mogelijke walkthrough voor de logins (andere scenario's zijn mogelijk):

- voeg aan je project een nieuw venster **LoginWindow** toe
- stel dit in App.xaml in als **StartupUri**
- in het loginvenster vraag je login en paswoord: zoek bijhorende gebruiker in de databank via een **query** `SELECT ... FROM Persoon WHERE ...`
- **indien geen** record gevonden wordt, is de login / paswoord combinatie niet correct
- **indien wel:** bewaar de gevonden admin in [Application.Current.Properties](#), open een nieuwe MainWindow, en haal de gebruiker er weer uit; sluit tenslotte het loginvenster

Het is aangeraden **login en paswoord vooraf in te vullen** in de XAML of C# in het loginformulier; dat scheelt tijd tijdens het testen (ook voor ons docenten).

#### Foutcontrole

- formchecking: zijn login / paswoord correct ingevuld?
- is de gebruiker gevonden?
- is het wel een admin?

Foutmeldingen toon je in een tekst, niet in een MessageBox!

#### Paswoord hashing

Paswoorden mogen niet zomaar letterijk opgeslagen worden in de databank: iemand die inbreekt op de databank, kan zo de favoriete paswoorden van iedereen lezen. Zoek een methode om een paswoord om te zetten naar de SHA256 versie; hiermee kan je het paswoord ingetikt door de gebruiker vergelijken met de gehashte versie in de databank. En voor wie echt zich wil verdiepen in het onderwerp: hashing alleen is eigenlijk nog niet genoeg - Google eens op [password salt](#).

## E.2. MainWindow

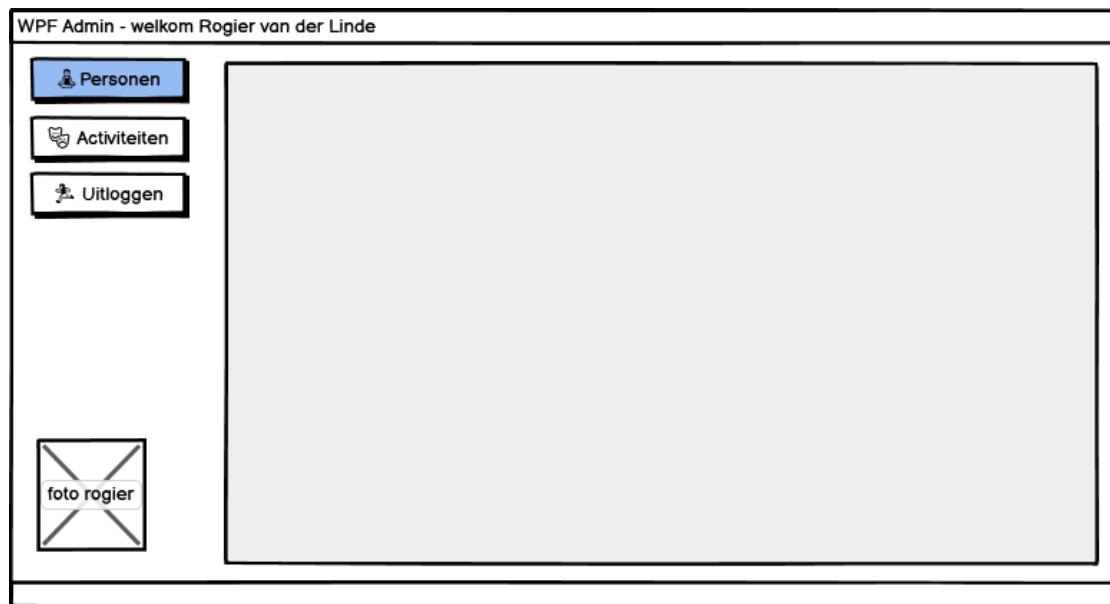
### Page/Frame

Om te vermijden dat je voor elk scherm een nieuw venster moet openen, zullen we gebruik maken van één WPF Frame control waarin je dan verschillende WPF Pages laadt (dus WPF Page, geen WPF Window). Je vindt online uitleg genoeg (zoals [deze video](#)). Een demo vind je onder **SInDemoFrame** (zie bijlage op Toledo bij project).

Bouw de MainWindow op met:

- drie knoppen linksboven: Personen, Activiteiten, Uitloggen
- een profielfoto van de ingelogde gebruiker linksonder
- een Frame control rechts.

Je kan dan telkens naargelang de actie van de gebruiker de juiste Page in dit Frame inladen. Daarna zullen we de Pages één voor één invullen, en nog bijkomende Pages toevoegen (zoals voor de detailpagina's).



### Lezen/schrijven van afbeeldingen in de databank

Voor het lezen en schrijven van afbeeldingen in de databank kan je de hulp van chatGPT inschakelen.

### Aandachtspunten

- Het gebruik van Frame / Page is verplicht; zoniet wordt op dit deel geen punten gegeven
- Laad standaard de Personen overzichtspagina in (zie volgend onderdeel)

### E.2.1 Page: personen overzicht

Na inloggen komt de admin rechtstreeks op de customers overzichtspagina terecht. Maak daarvoor een nieuwe Page aan, en laad die standaard in het Frame in. In deze Page toon je standaard het overzicht van personen:

- links een ListBox met alle personen
- rechts een panel waar alle details verschijnen als ik een persoon selecteer, met daaronder drie knoppen voor bewerken/verwijderen/nieuw

Inspireer je vervolgens op de code van SInCompany om de code in de Page uit te werken.

## E.2.2 Pages: persoon bewerken, verwijderen, nieuwe

Werk nu de knoppen bewerk, verwijder en nieuw uit. Let op: dit zijn drie nieuwe Pages die in het Frame ingeladen worden, geen Windows zoals in SInCompany!

### Aandachtspunten

- Let op dat je hier (en volgende delen waar foto's aan bod komen) ook de foto kan aanpassen – dit is een belangrijk onderdeel van dit project.

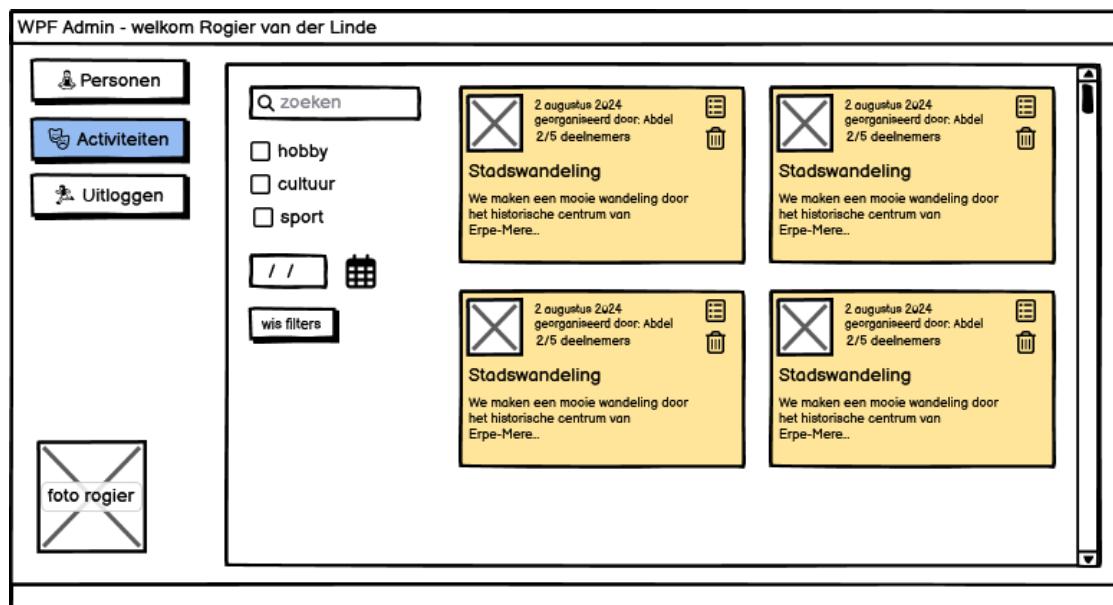
## E.2.3 Page: activiteiten overzicht

Voor de activiteiten gebruiken we geen ListBox, maar een **rasterweergave**. Bestudeer vooraf **SInDemItemsPanel** (zie Toledo bij project) over hoe je items dynamisch vanuit C# in een panel control laadt. Geef alle activiteiten weer:

- voorzie **checkboxes** om te kunnen filteren op soort
- voorzie een **zoekvenster**; zoek in titel én beschrijving
- voorzie een **datepicker** waar je kunt zoeken op datum
- sorteert op datum, **nieuwste eerst**
- geef per activiteitsblokje de foto, datum, de titel, het eerste deel (b.v. 100 karakters) van de beschrijving, de organisator, het maximum aantal personen en het aantal ingeschreven deelnemers weer, alsook iconen voor details bekijken en verwijderen

### Aandachtspunten

- De weergave in een raster is verplicht; zoniet wordt op dit deel geen punten gegeven
- Voorzie scrollbars
- Je mag zelf de kaart (afbeelding, knoppen, teksten...) schikken zoals je wil, zolang alles er maar op staat



## E.2.4 Page: activiteit verwijderen en details bekijken

Maak nu de pages voor verwijderen en details bekijken van activiteiten. Pages voor bewerken of toevoegen is niet nodig.

## F. WpfGebruiker

Deze app bestaat uit drie onderdelen:

- loginpagina
- activiteiten (alle activiteiten)
- organiser (eigen activiteiten)

Er worden in tweede zit voor dit deel geen screenshots of mockups gegeven! Inspireer je op de mockups uit de Fitness opgave in eerste zit en lees goed de beschrijving.

## F.1. LoginWindow

Voorzie ook voor deze WPF app een loginvenster.

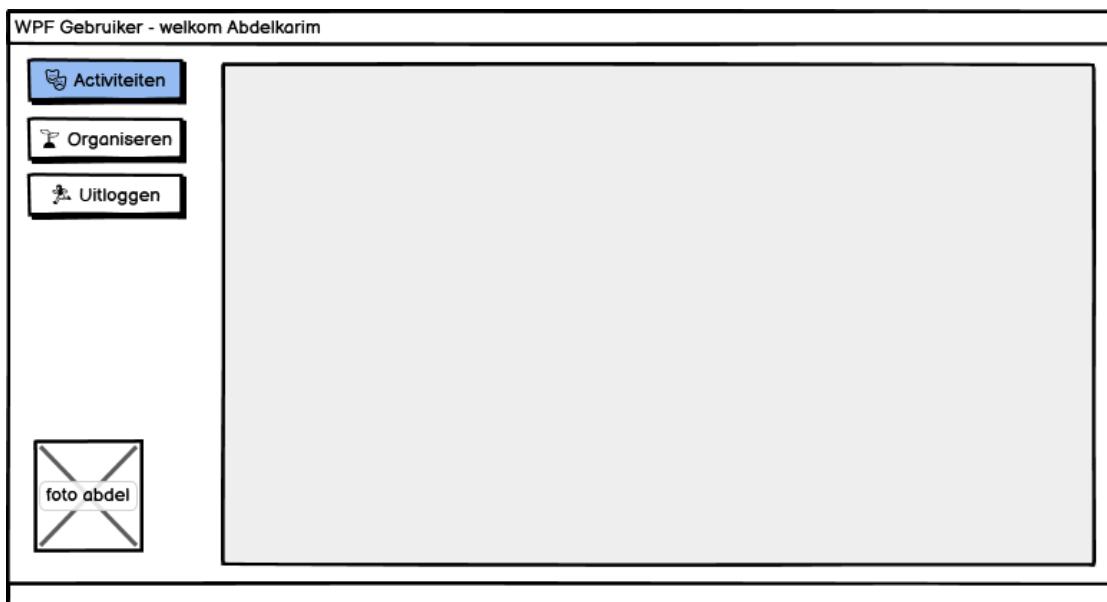
## F.2. MainWindow

### Page/Frame

Ook deze app is opgebouwd met Frame/Page zoals WpfAdmin.

Bouw de MainWindow op met:

- drie knoppen linksboven: Activiteiten, Organiseer, Uitloggen
- een profelfoto van de ingelogde gebruiker linksonder
- een Frame control rechts.



### Aandachtspunten

- Het gebruik van Frame / Page is verplicht; zoniet wordt op dit deel geen punten gegeven
- Laad standaard de Activiteiten overzichtspagina in (zie volgend onderdeel)

### F.2.1 Page: Activiteiten

Toon alle activiteiten op een **kaart**:

- voorzie filtermogelijkheid per soort activiteit en leeftijdscategorie
- voorzie een zoekvenster (zoek in titel en beschrijving)
- plaats alle resultaten op de kaart; gebruik de iconen van de activiteit
- als je een activiteit aanklikt, moet je alle details zien en een knop om in- of weer uit te schrijven

WPF Gebruiker - welkom Abdelkarim

[Activiteiten](#)

[Organiseren](#)

[Uitloggen](#)

[foto abdel](#)

Leeftijdscategorie ▾  hobby  cultuur  sport  zoeken

**stadswandeling**

datum: 2 augustus 2024  
organisator: Axel  
max. deelnemers: 5

We maken een mooie wandeling door het historische centrum van Erpe-Mere. We bezoeken we de vermaarde vlasweverijen, de oude abdij en het volkshistorisch museum, om af te sluiten met een drankje op een terras.

[Inschrijven](#)

nog 3 van 5 plaatsen over

8 activiteiten gevonden

### Aandachtspunten

- Gebruik voor de kaart de [GMap.NET](#) package

## F.2.2 Page: Organiseer

Geef een overzicht van alle activiteiten die zelf georganiseerd zijn in een rasterweergave. Je kan de code grotendeels overnemen van het admingededeelte. Voorzie knoppen voor toevoegen, bewerken en verwijderen.

*Voor deze pagina is geen mockup voorzien; je moet het zelf ontwerpen.*

### Aandachtspunten

- De weergave in een raster is weer verplicht; zoniet wordt op dit deel geen punten gegeven

## F.2.3 Pages: eigen activiteit toevoegen en verwijderen

Werk zelf deze pagina's uit.

### Aandachtspunten

- Vraag bij verwijderen eerst een bevestiging, niet met MessageBox, maar via een aparte Page

## G. Afwerking

### G.1. Overerving

Implementeer nu overerving. Maak voor elke soort een klasse die overerft van de klasse Activiteit; maak die laatste abstract. Herschrijf tenslotte je code gebruikmakend van deze nieuwe klassen.

## G.2. Exception handling en commentaar

Helemaal op het einde is het moment om exception handling en commentaar toe te voegen, en je code te optimaliseren/verbeteren.

## H. Indienen

De deadline voor indienen is **maandag 19 augustus 20u**. Je moet het pushen op je repository én uploaden op Toledo. Let er op dat volgende zaken zeker aanwezig zijn:

- de **volledige code**
- een lijst gebruikte bronnen, inclusief een **pdf dump van alle AI chats** met AI bots met behulp van de [Save ChatGPT as PDF plugin](#)
- een **SQL dump** van je databank (schema én data!)

Happy coding!