

# Afstudeerverslag

**Applicatieontwikkelaar**  
Good-Lookz

**4People Communications**



Nuenen, 1 juni '17, v4

Opgesteld door: Furkan Demirci, 69274, furkan.demirci@live.nl

## **Overzichtsblad**

DEELNEMER: Furkan Demirci  
1<sup>e</sup> haagstraat 51a  
5707XN Helmond  
+316 - 43115967, furkan.demirci@live.nl

SCHOOL: ICT College – ROC Ter AA  
Keizerin Marialaan 2 - Postbus 490  
5702 NR Helmond - 5700 AK Helmond  
Tel: 0492 – 507900

LEERBEDRIJF: 4People  
Nuenderbeekselaan 13  
5673RC Nuenen  
+316 - 53904909

BPV-DOCENT: Rico van Dooren  
r.v.dooren@roc-teraa.nl

PRAKTIJKOPLEIDER: Jeroen Franssen  
Jeroen@4people.nl

OPLEIDING: Applicatieontwikkelaar, niveau 4

### **PERIODE VAN UITVOERING:**

Dit project is uitgevoerd in de periode van:  
06-02-2017 tot 22-06-2017

## Voorwoord

Ik ben Furkan Demirci en zit in het derde en laatste leerjaar van de opleiding applicatieontwikkeling op het ROC Ter AA. Voor u ligt mijn stageverslag, betreft mijn stageperiode binnen 4People Communications Nuenen. Deze stage is een verplicht onderdeel in het afstudeerjaar, aan de opleiding Applicatieontwikkeling. Deze volg ik aan het ROC Ter AA, te Helmond. De doelstelling van deze stage, is het toepassen van de geleerde theoretisch kennis in de praktijk. Op deze manier doe ik ervaring op in mijn eventueel toekomstig werkgebied. Een onderdeel van de afstudeerstage is een afstudeerverslag te schrijven en in dit verslag ga ik aantonen dat ik het vak applicatieontwikkeling waardig ben.

Met 4People Communications ben ik in contact gekomen door Nihat Sahin nadat we een gesprek hadden gehad. Ik wou graag apps maken voor mobiele telefoons en samen waren we dus uitgekomen bij 4People Communications. Na een gesprek bij 4People kreeg ik vrijwel direct de bevestiging dat ik daar stage mocht komen lopen.

Tijdens de stageperiode heb ik met veel plezier gewerkt binnen het bedrijf. Het is een leuk bedrijf waar een fijne werksfeer heerst. De volgende personen hebben mij tijdens de stage zeer goed ondersteund en begeleid, om die reden zou ik ze graag willen bedanken voor de geboden hulp:

- Dhr. Jeroen Franssen voor het ondersteunen en begeleiden van mijn stage en het beantwoorden van al mijn vragen.
- Mijn collega's Davey Cornelissen en Rick de Vries voor het helpen met mijn project als ik hulp nodig had.
- Dhr. Rico van Dooren, voor het ondersteunen en begeleiden van mijn stage en het geven van feedback.



## Inhoudsopgave

Overzichtsblad.....	2
Voorwoord .....	3
Inleiding .....	6
Stage bij welk bedrijf .....	6
Wat zijn de activiteiten van het bedrijf.....	6
Producten .....	6
Organigram.....	7
Afdeling .....	7
Project .....	7
1.    Ontwerpen van de applicatie .....	8
1.1    Stelt de vraag en/of informatiebehoefte vast.....	8
1.2    Maakt een plan van aanpak .....	8
1.3    Levert een bijdrage aan het functioneel ontwerp .....	8
1.4    Maakt een technisch ontwerp .....	8
1.5    Richt de ontwikkelomgeving in .....	9
2.    Realiseren van de applicatie.....	11
2.1    Legt een gegevensverzameling aan.....	11
2.2    Realiseert een applicatie .....	12
2.5    Test het ontwikkelde product .....	14
3.    Implementeren van de applicatie .....	15
3.1    Maakt of levert een bijdrage aan het implementatieplan .....	15
3.2    Stelt een acceptatietest op en voert deze uit .....	15
3.3    Implementeert een applicatie.....	15
3.4    Evalueert een implementatie.....	15
4.    Onderhouden en beheren van de applicatie .....	16
4.1    Onderhoudt applicaties.....	16
4.5    Beheert de content .....	18
4.6    Documenteert en archiveert gegevens.....	20
5.    Bronvermelding.....	21
6.    Verklarende woordenlijst.....	22
Bijlage A – Informatiebehoefte .....	23
Bijlage B - Interview.....	25
Bijlage C - Plan van aanpak.....	27

Bijlage D - Functioneel ontwerp .....	34
Bijlage E - Technisch ontwerp .....	54
Bijlage F – WEB-API documentatie .....	60
Bijlage G – Test plan .....	74
Bijlage H – Implementatieplan .....	86

# Inleiding

## Stage bij welk bedrijf

Bedrijf: 4People Communications.

4People Communications is verfrissend en scherp met opvallend creatief werk.

De scherpe mix tussen het doel, een opvallend idee/concept en commerciële haalbaarheid en uitvoering maakt de positionering meteen helder, bepalend, verrassend en doelgericht.

In 1992 is Jeroen Franssen begonnen als organisatiebureau voor evenementen en diensten (parttime). In 2000 is de naam veranderd in 4People, sindsdien is Jeroen fulltime gaan werken. Waar Jeroen zich vooral mee bezighield, is met evenementen voor horeca en evenementorganisatoren waar maar liefst 60 shows voor gedraaid zijn. In 2004 campagnes voor bedrijven. Daarna heeft 4People zich meer gefocust op innovatie voor derde en zichzelf.

## Wat zijn de activiteiten van het bedrijf

4People Communications is gespecialiseerd in het bedenken van creatieve commerciële concepten zowel digitaal en fysiek in allerlei vormen en inzetbare technieken om zo efficiënt naar de juiste doelgroep te communiceren.

Als ideeëngenerator voor organisaties en media- & reclamebureaus of als eigen project zijn er vele succesvolle opvallende "out of the box"-projecten, campagnes, digitale-, onlinediensten en producten gerealiseerd.

## Producten

De producten die er gemaakt worden zijn heel breed, van commercieel tot maatschappelijk. Van websites, programma's, apps tot daadwerkelijke fysieke producten.

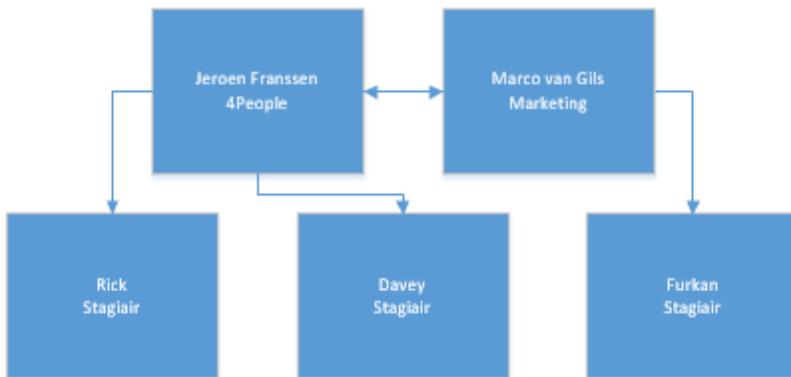
Een aantal voorbeelden:

- narrowcasting, verbinding tussen beeldschermen (bijvoorbeeld bij Centerparks).
- thisaction.com, om snel en gemakkelijk een kansspelsite op te zetten.
- student-concepts.com / idea-concepts.com / share2care.com, een soort ideeënmarktplaats voor studenten, bedrijven & mensen met een beperking.
- alert-ice.com, een hulparmband die bij noodgeval via chipcontact informatie weergeeft op een smartphone.
- 4esite.com, in een CMS framework om heel snel een website voor evenementen & andere doeleinden op te zetten.

Doelgroep: heel divers. Waar innovatie is, is een doelgroep.

Aantal medewerkers: Normaal werken er twee personen in het bedrijf. Nu met de stagiairs erbij werken er 5 personen.

## Organigram



## Afdeling

4People Communications is oorspronkelijk een tweemans-bedrijf. Dus een afdeling is hierbij overbodig. Natuurlijk heeft iedereen wel een eigen (hoofd)taak.

Functie: website/apps programmeren.

## Project

Projectomschrijving staat beschreven in de informatiebehoefte. **Zie bijlage A.**



# 1. Ontwerpen van de applicatie

## 1.1 Stelt de vraag en/of informatiebehoefte vast

Na een gesprek met Jeroen Franssen is er een opdracht opgesteld. Deze staat op papier gedocumenteerd en is bevestigd met een handtekening van de opdrachtgever en de opdrachtnemer. De opdracht is een cross-platformapplicatie waar gebruikers zich voor kunnen aanmelden door een account aan te maken. In de app is het mogelijk om kleding stukken op te slaan en om het te delen of uitlenen aan vrienden die gekoppeld zijn aan het account. Het is ook mogelijk om kledingstukken als een soort set op te slaan. Als er geen tijdgebrek is wordt er ook nog een spiegelfunctie toegevoegd waar vrouwen zijn make-up kunnen toepassen doormiddel van de front-camera te gebruiken. Hierbij zit ook een webshopachtige pagina waar de gebruiker verwijst wordt naar de originelen webshop van de sponsoren. Dit is de uitleg van de opdracht zoals het beschreven staat in het document en inleiding van dit verslag. Verder wordt het doel en de opdracht nog verder beschreven. Zie **Bijlage A** voor het document.

## 1.2 Maakt een plan van aanpak

Nadat de opdracht duidelijk was, is er een plan van aanpak opgesteld. In het plan van aanpak worden een aantal onderdelen beschreven. Als eerst is er een korte beschrijving over de opdracht beschreven. De projectdoelstelling en projectgrenzen zijn ook vastgesteld. In de projectdoelstelling wordt letterlijk de doelstelling van de opdracht beschreven. In de projectgrenzen wordt beschreven hoe de applicatie gecodeerd moet worden en hoe het later makkelijk moet kunnen worden uitgebreid. Verder worden de activiteiten, planning en kosten nog beschreven in het plan van aanpak. Zie **Bijlage C** voor het plan van aanpak.

## 1.3 Levert een bijdrage aan het functioneel ontwerp

In het functioneel ontwerp wordt er beschreven over het perspectief van het app. Ook zit er een flowchart bij en worden er doormiddel van wireframes de functionaliteit weergegeven. Een flowchart is een schematische voorstelling van een proces, waarmee de functionaliteit makkelijker te visualiseren is. In dit geval leek het handig om het visuele proces van de applicatie uit te leggen. Met behulp van de wireframes is er een makkelijk pad gelegd tussen de functies binnen de applicatie. Er wordt ook een beschrijving bij gegeven om makkelijker uit te leggen wat er zou moeten gebeuren. Zie **Bijlage D** voor het functionele ontwerp.

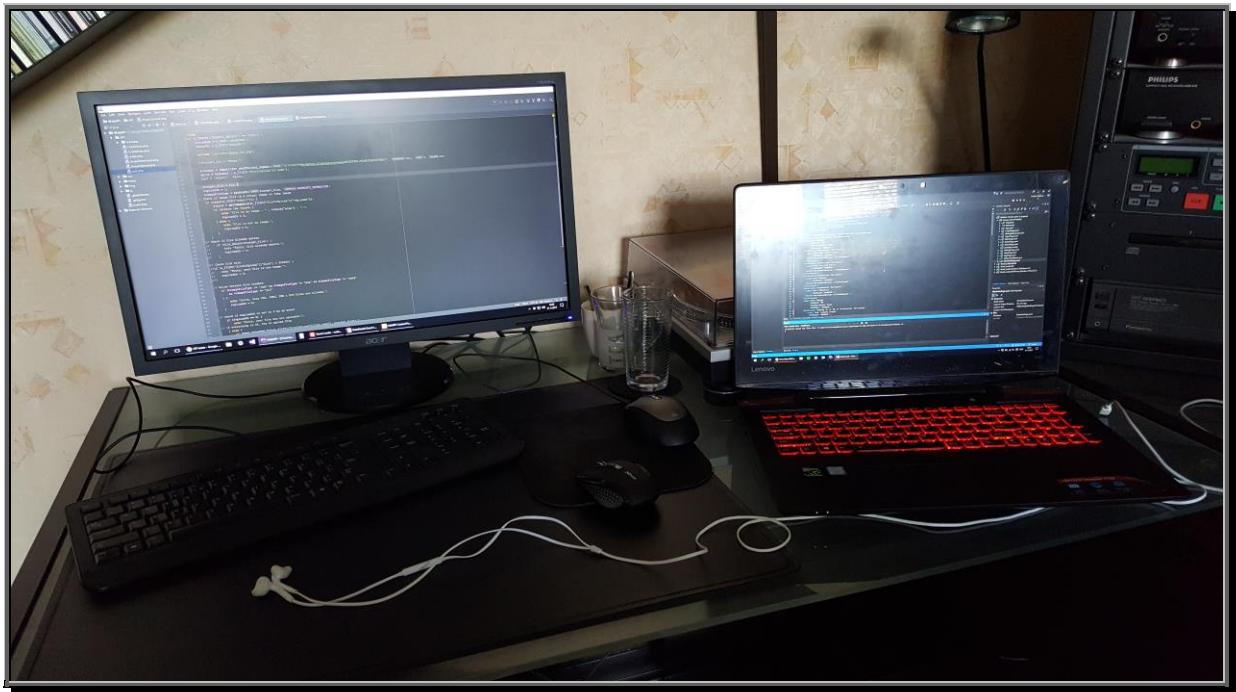
## 1.4 Maakt een technisch ontwerp

Nadat het functionele ontwerp is gemaakt, is er een technisch ontwerp opgesteld. Het technische ontwerp is gebaseerd op het functionele ontwerp, de technische aspecten staan uitgelegd in dit document. Hier wordt bijvoorbeeld uitgelegd hoe de WEB-API precies samenwerk met de applicatie. Hoe de data wordt gestuurd doormiddel van POST en GET requesten en responsen als JSON. Het technische ontwerp is bedoeld om de technische aspecten van de applicatie te documenteren. Zie **Bijlage E** voor het technische ontwerp.



## 1.5 Richt de ontwikkelomgeving in

Werkomgeving bij 4People:



Windows 10 met de volgende applicaties:

### Sublime text

Sublime Text is een cross-platform-editor voor tekst en broncode. Het uiterlijk van het programma is geïnspireerd op de teksteditor Vim. De functionaliteit kan worden uitgebreid door middel van plug-ins.



### Filezilla

Is een opensource-FTP-programma voor Windows, Mac, Linux en FreeBSD. Met FileZilla is het mogelijk over meerdere verbindingen, bestanden naar een FTP-server te sturen en op te halen, waardoor men sneller bestanden kan uploaden en/of downloaden.



FileZilla biedt de mogelijkheid tot een veilige verbinding, zo is er ondersteuning voor SFTP over SSH en FTP over SSL/TLS. Ook kan er met wachtrijen en proxyservers gewerkt worden. Er is een ook een mogelijkheid om de upload- en downloadsnelheden te begrenzen.

### MySQL

MySQL is een proprietair opensource-managementsysteem voor relationele databases (RDBMS). SQL is de taal die wordt gebruikt om een database van dit systeem op te bouwen, te bevragen en te onderhouden. MySQL werd allereerst vooral gebruikt voor internettoepassingen zoals fora en gastenboeken, meestal in combinatie met PHP. MySQL vormt de basis van vele internettoepassingen en standalone software.



## **Visual Studio**

Microsoft Visual Studio is een programmeerontwikkelomgeving van Microsoft. Het biedt een complete set ontwikkelingstools om computerprogramma's in diverse programmeertalen voor met name Windows-omgevingen te ontwikkelen. De talen Visual Basic.Net, C#, F# en C++ worden in de standaardeditie meegeleverd. Het wordt gebruikt om ASP.NET-webapplicaties, XML-webservices, desktopapplicaties en mobiele toepassingen te ontwerpen.

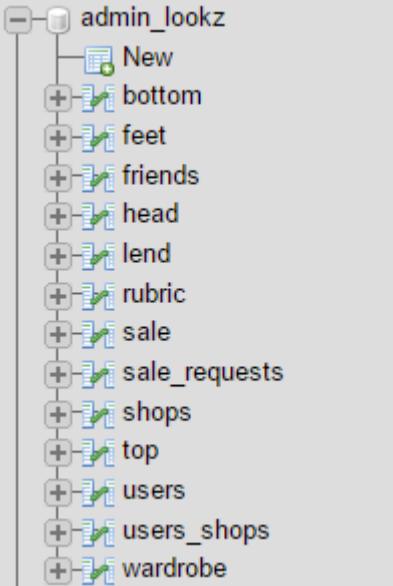
Voor een cross-platform applicatie kwam het beste Xamarin Forms uit omdat het gebruik maakt van C# en de geschreven codes gedeeld word met alle platformen dus is het niet nodig om voor elk apart platform te gaan coderen. Ook kwam dit goed uit omdat ik de kennis hierover al had opgebouwd over de taal C# en het programma Visual Studio.



## 2. Realiseren van de applicatie

### 2.1 Legt een gegevensverzameling aan

Met de app wordt er gecommuniceerd met de database via een WEB-API. Deze WEB-API is gemaakt met PHP en wordt gehost op de domein good-lookz.com. De WEB-API zorgt ervoor dat er GET en POST requesten kunnen worden ontvangen van de app en daarna een JSON als response terugstuurt. Hiervoor is ook documentatie gemaakt over de uitleg van de WEB-API en databaserelaties. Dit is te zien in **bijlage E** datamodel en **bijlage F**. In de database worden alle gegevens verzameld zoals accounts, koppelingen van vrienden, kledingen en kleding-sets. De kleding wordt als een Image in het server en in het database opgeslagen. In het technische ontwerp is een datamodel te zien met alle relaties van de database. Er wordt ook gebruikgemaakt van Application Properties, dit zorgt ervoor dat instellingen worden opgeslagen na het aanpassen.



```
$conn = mysqli_connect("localhost", "admin_lookz", "nicetry", "admin_lookz");

if (!$conn) {
    echo "Error: Unable to connect to MySQL." . PHP_EOL;
    echo "Debugging errno: " . mysqli_connect_errno() . PHP_EOL;
    echo "Debugging error: " . mysqli_connect_error();
    exit;
}
```

In de documentatie dat voor de WEB-API gemaakt is wordt uitgelegd hoe de POST- en GET-requesten zou worden moeten ontvangen en wat de API zou moeten terug sturen als response.

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    header("Access-Control-Allow-Origin: *");
    header('Content-Type: application/json');

    $loginName = $_POST['loginName'];
    $password = $_POST['password'];

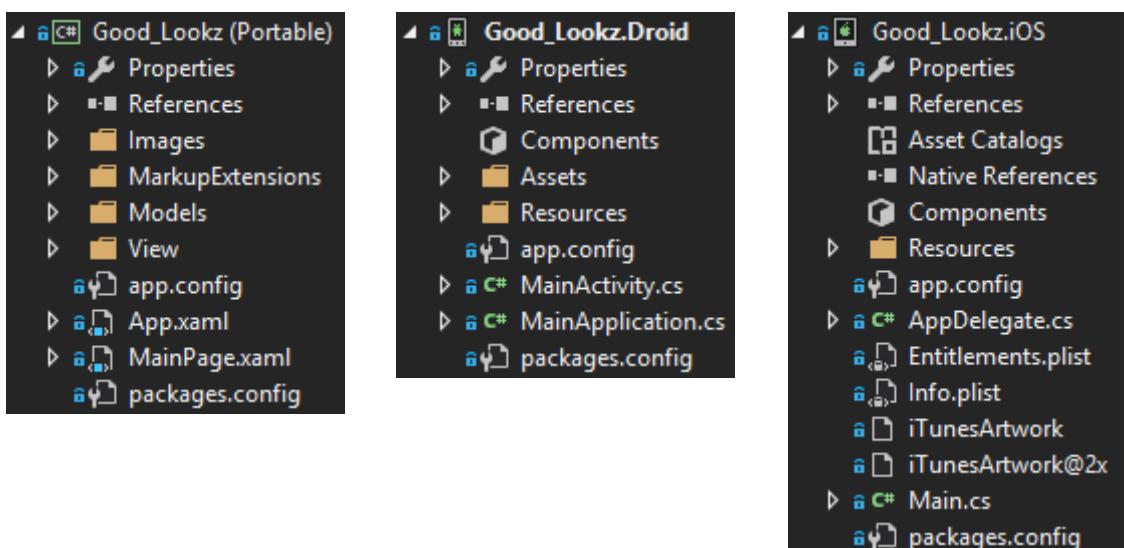
    $row_array['friends_check'] = false;
    array_push($return_arr, $row_array);
    echo json_encode($return_arr);
```

## 2.2 Realiseert een applicatie

De applicatie is ontwikkeld aan de hand van het plan van aanpak, functioneel- en technisch ontwerp. De eisen horen te voldoen aan de plan van aanpak. Good-Lookz app moet voldoen aan een werkend loginsysteem via de database en minimaal een wardrobe-functie of een camerafunctie. De camerafunctie moet als een spiegelbeeld worden gebruikt, zodat de gebruiker (vrouw) zijn make-up kan doen. In de wardrobe-functie is het mogelijk om kledingstukken op te slaan en te bekijken. Hier is het ook mogelijk om kledingstukken als een set op te slaan. Als een gebruiker is gekoppeld met zijn of haar vrienden is het mogelijk om elkaars kledingstukken te zien en is het mogelijk om te vragen om een kledingstuk te lenen. De structuur van de applicatie is makkelijk te begrijpen zodat er later door andere stagiaires de project makkelijk weer opgepakt kan worden. De mappen structuur bestaat uit:

- Een groot solution Good-Lookz dat bestaat uit 5 projecten (3 worden er gebruikt)
  - Good\_Lookz (Portable)
  - Good\_Lookz.Droid
  - Good\_Lookz.iOS
  - Good\_Lookz.Windows
  - Good\_Lookz.WinPhone

Meestal wordt er alleen in de portable class project gewerkt omdat dat shared code is over alle platformen. Als er specifiek code geschreven moet worden voor een platform dan wordt dat dus in Droid project of iOS project geschreven.



De portable class library bestaat uit 3 mappen. Images zijn voor pictogrammen die voor allebei de platformen gebruikt worden. In Models zitten alle classes van de applicatie dat JSON requesten opvangt. Hier worden ook verschillende values door middel van Get en Set opgeslagen.



```

Models
├── CreateAccount.cs
├── FriendsCredentials.cs
├── GetShops.cs
├── LendList.cs
├── LoginAccount.cs
├── LoginCredentials.cs
├── Notifications.cs
├── SaleList.cs
├── ShopsChosen.cs
├── WardrobeBottom.cs
├── WardrobeFeet.cs
├── WardrobeHead.cs
├── WardrobeSetDelete.cs
├── WardrobeSets.cs
├── WardrobeSetsItem.cs
└── WardrobeTop.cs

```

```

public class LoginAccount
{
    public string id { get; set; }
    public string username { get; set; }
    public string password { get; set; }
    public string first_name { get; set; }
    public string last_name { get; set; }
    public string email { get; set; }
    public string date { get; set; }
    public string gender { get; set; }
    public string offline { get; set; }
    public string active { get; set; }
}

public class SelectedFriendsCredentials
{
    private static string _id;
    private static string _friends_id;
    private static string _fullName;

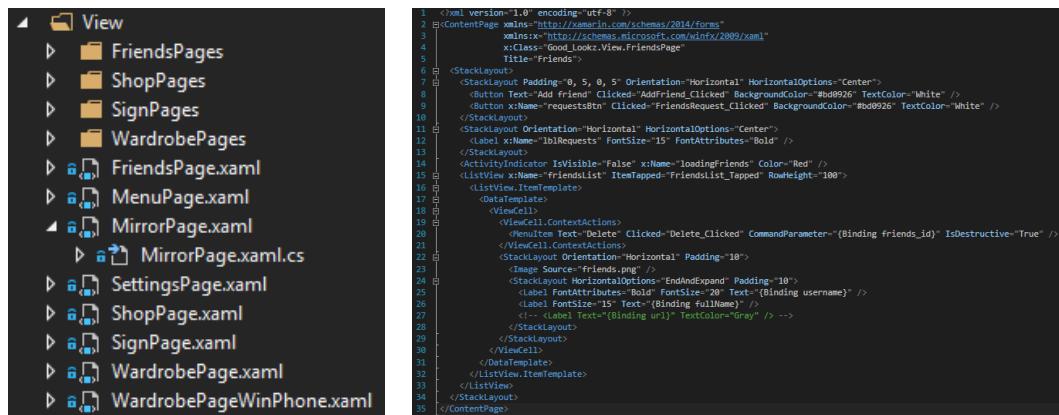
    public static string id
    {
        get { return _id; }
        set { _id = value; }
    }

    public static string friends_id
    {
        get { return _friends_id; }
        set { _friends_id = value; }
    }

    public static string fullName
    {
        get { return _fullName; }
        set { _fullName = value; }
    }
}

```

View map bestaat uit XAML files met code-behind voor visueel weergave maar ook voor de backend.



```

View
├── FriendsPages
├── ShopPages
├── SignPages
├── WardrobePages
├── FriendsPage.xaml
├── MenuPage.xaml
└── MirrorPage.xaml
    ├── MirrorPage.xaml.cs
    └── SettingsPage.xaml
    ├── ShopPage.xaml
    ├── SignPage.xaml
    ├── WardrobePage.xaml
    └── WardrobePageWinPhone.xaml

```

```

<xaml version="1.0" encoding="utf-8" />
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Good_Lookz.View.FriendsPage"
             Title="Friends">
    <StackLayout Padding="0, 5, 0, 5" Orientation="Horizontal" HorizontalOptions="Center">
        <Button Text="Add Friend" Clicked="AddFriend_Clicked" BackgroundColor="#Bd0926" TextColor="White" />
        <Button x:Name="requestBtn" Clicked="FriendsRequest_Clicked" BackgroundColor="#Bd0926" TextColor="White" />
    </StackLayout>
    <StackLayout Orientation="Horizontal" HorizontalOptions="Center">
        <Label x:Name="blbRequests" FontSize="15" FontAttributes="Bold" />
        <ActivityIndicator IsVisible="False" x:Name="loadingFriends" Color="Red" />
    </StackLayout>
    <ListView x:Name="friendsList" ItemTapped="FriendsList_Tapped" RowHeight="100">
        <ListView.ItemTemplate>
            <DataTemplate>
                <ViewCell>
                    <ViewCell.ContextActions>
                        <MenuItem Text="Delete" Clicked="Delete_Clicked" CommandParameter="{Binding friends_id}" IsDestructive="True" />
                    </ViewCell.ContextActions>
                    <StackLayout Orientation="Horizontal" Padding="10">
                        <Image Source="friends.png" />
                        <StackLayout HorizontalOptions="EndAndExpand" Padding="10">
                            <Label FontSize="10" Text="Full Name" />
                            <Label FontSize="15" Text="{Binding fullName}" />
                            <Label Text="(" Binding="url" TextColor="Gray" />
                        </StackLayout>
                    </StackLayout>
                </ViewCell>
            </DataTemplate>
        </ListView.ItemTemplate>
    </ListView>
    </StackLayout>
</ContentPage>

```

Er wordt ook documentatie geschreven tussen de code van de applicatie. Hier worden moeilijke beschreven code beter uitgelegd.

Dit is in het kort uitgelegd hoe de structuur werkt. Deze manier is voor alle bestaande/toekomstige pagina's makkelijk te volgen. Zodat als er een bepaalde fout op een bepaalde pagina zit dat je dan ook specifiek kunt zoeken in de desbetreffende map. Ook is er rekening mee gehouden dat in elk bestand, documentatie beschreven is wat een bepaalde functie doet of moet doen.

```

/// <summary>
/// A list must be created before filling it in with JSON data.
/// This prevents compiling error when it's trying to make the value clear.
/// </summary>
List<Models.WardrobeHead> gets_Head = new List<Models.WardrobeHead>();
List<Models.WardrobeTop> gets_Top = new List<Models.WardrobeTop>();
List<Models.WardrobeBottom> gets_Bottom = new List<Models.WardrobeBottom>();
List<Models.WardrobeFeet> gets_Feet = new List<Models.WardrobeFeet>();

```

## 2.5 Test het ontwikkelde product

Er wordt wekelijks getest na het wijzigen of schrijven van nieuwe code. Bij het testen van het product wordt er gekeken of de applicatie voldoet aan de plan van aanpak en aan de eisen en wensen van de opdrachtgever. Het testen van de WEB-API wordt gedaan door middel van POST- en GET-requesten die via een testwebsite wordt gestuurd. Hier wordt bekijken of de juiste nummers worden meegegeven en of het goed wordt bijgewerkt in het database. Het testen van de app wordt gedaan doormiddel van emulators en Android apparaten. Het testen van de WEB-API wordt gedaan op een speciaal test website dat specifiek geschreven is voor de WEB-API. Hier wordt bekijken of het goed draait en voldoet aan de verwachtingen. Als het testen fout gaat, wordt het opnieuw bekijken en verbeterd. Dit wordt allemaal gedocumenteerd in het testplan. Zie **Bijlage G**.

Hier is een voorbeeld te zien hoe ik de WEB-API test.

The screenshot shows a web browser window with four separate API tester forms side-by-side:

- Head:** Contains fields for id (text input), Select image to upload (button), id (text input), JSON (button), head\_id (text input), and a red DELETE button.
- Top:** Contains fields for id (text input), Size (radio buttons S, M, L, XL), Select image to upload (button), id (text input), JSON (button), top\_id (text input), and a red DELETE button.
- Bottom:** Contains fields for id (text input), Size (radio buttons S, M, L, XL), Select image to upload (button), id (text input), JSON (button), bottom\_id (text input), and a red DELETE button.
- Feet:** Contains fields for id (text input), Select image to upload (button), id (text input), JSON (button), feet\_id (text input), and a red DELETE button.

The browser title bar shows "FunkanDemirici/Good-Lookz.com - good-lookz.com / local - API tester".

## 3. Implementeren van de applicatie

### 3.1 Maakt of levert een bijdrage aan het implementatieplan

Er is een heldere volledige implementatieplan geschreven over dit project. De documenten zijn te vinden in **Bijlage H**. In de implementatieplan staat beschreven over het doel van de applicatie, oud en nieuw systeem, scope en afbakening. Hier wordt ook de gevolgen gedocumenteerd voor trainingen en opleiding, maar ook voor ICT/Facilitair. Er wordt goed uitgelegd hoe dit wordt aangepakt met de acceptatietest. Hier worden ook de risico's en maatregelen beschreven die kunnen voorkomen tijdens het implementeren.

### 3.2 Stelt een acceptatietest op en voert deze uit

In het implementatieplan is er een acceptatietest te zien dat wordt uitgevoerd tijdens het implementeren. In het acceptatietest zijn de belangrijkste punten genoteerd dat getest moet worden door een gebruiker en vervolgens bijgewerkt moet worden om tot een resultaat te komen. Dit is gedocumenteerd in een Excel tabel met een status of het goed- of afgekeurd is. Voor een afgekeurd onderdeel word er een evaluatie beschreven hoe dit moet worden opgelost. Indien er geen fouten zijn gevonden zal het implementatieplan geëvalueerd worden met de opdrachtgever en de gebruikers. De documenten zijn te vinden in **Bijlage H**.

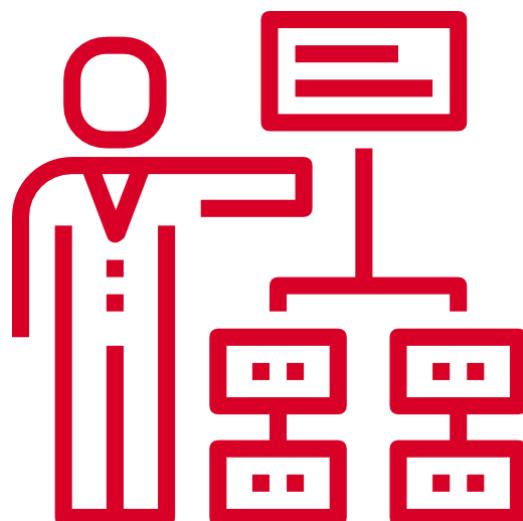
### 3.3 Implementeert een applicatie

De applicatie word geïmplementeerd naar aanleiding van de implementatie plan. Zie **Bijlage H**.

### 3.4 Evalueert een implementatie

Na het implementeren is er gekeken naar de resultaten van de acceptatietest, daaruit is gebleken dat alles is verlopen zoals gepland. Tevens is er met de betrokkenen besproken over de implementatie en is er nog kort geëvalueerd met de opdrachtgever.

Het bewijs van de evaluatie is te vinden in **Bijlage H**.



## 4 Onderhouden en beheren van de applicatie

### 4.1 Onderhoudt applicaties

#### Incidentmelding

Hier zijn de incidentmeldingen te zien.

Nr.	Incidentmelding	Beschrijving	Datum	status
1	Leen systeem	Een leen systeem dat alle kledingen van je gekoppelde vrienden laat zien in swipe functie binnen een pagina.	11-05-2017	Bijgewerkt
2	Image formaat	Images moeten een kleinere formaat hebben zodat er geen extra geheugen wordt gebruikt in het applicatie.	18-05-2017	Bijgewerkt
3	Image error	Als de gebruiker geen foto's had gaf die een error aan.	22-05-2017	Bijgewerkt

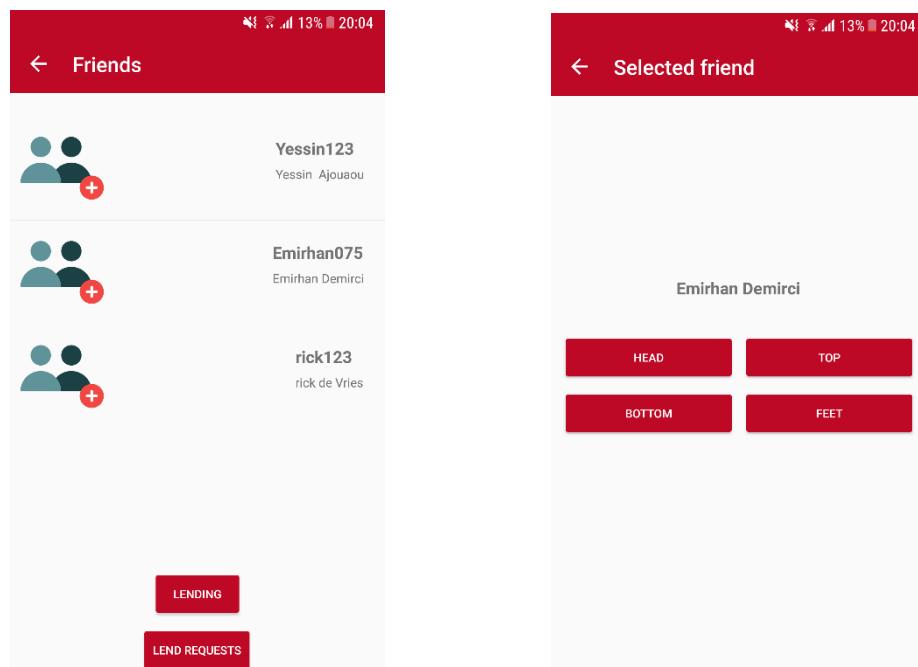
#### 1 – Leen systeem:

In het vorige leen systeem was het mogelijk om van een specifieke gekoppelde vriend de kledingen te bekijken en kledingstukken te lenen. Dit is handig als het gaat om 1 specifieke vriend.

#### Nieuwe eisen:

De gebruikers willen een functie hebben waar het mogelijk is om alle kledingen van alle gekoppelde vrienden te zien op een pagina met een swipe functie. Hier is het ook mogelijk om een kledingstuk te lenen of uit te lenen.

Aanpassingen zijn bijgewerkt in het documentatie. In het functioneel ontwerp is er een documentatie toegevoegd over de verandering.

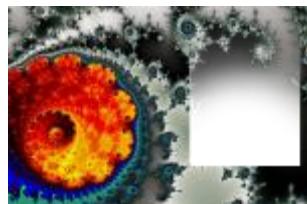


## 2 - Image formaat:

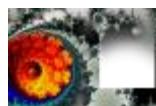
In het oude upload systeem van de WEB-API werden de images die geüpload waren niet aangepast dus bleef het de originele size en formaat behouden. Dit zorgt voor veel ruimte in de server en in de applicatie. Bijvoorbeeld een foto van 10mb laden in het applicatie gebruikt teveel data en teveel memory. Dit zorgt voor een Java memory error dat het interne ram geheugen vol zit.

### Nieuwe eisen:

Een nieuwe upload systeem dat er voor zorgt dat de geüploade images worden aangepast aan de size en formaat. Bijvoorbeeld een image van 4 mb word verlaagt naar 10kb. Dit gaat ervoor zorgen dat er niet veel data wordt gebruikt en ook niet veel memory. Aanpassingen zijn bijgewerkt in de documentatie. In de WEB-API documentatie ([zie bijlage F](#)) is het bijgewerkt.



image/png – 150 x 100 – 35.25KB



image/png – 75 x 50 – 9.75KB

## 3 – Image error:

Als de gebruiker geen foto's had gaf de applicatie tijdens het openen van de wardrobe page een error aan. Dit gebeurde omdat de applicatie niet de list kon vullen met image gegevens.

```
/// <summary>
/// A list must be created before filling it in with JSON data.
/// This prevents compiling error when it's trying to make the value clear.
/// </summary>
List<Models.WardrobeHead> gets_Head = new List<Models.WardrobeHead>();
List<Models.WardrobeTop> gets_Top = new List<Models.WardrobeTop>();
List<Models.WardrobeBottom> gets_Bottom = new List<Models.WardrobeBottom>();
List<Models.WardrobeFeet> gets_Feet = new List<Models.WardrobeFeet>();
```

### Nieuwe eisen:

Tijdens het ophalen van de images gaat de code door een try en catch method. Hier gaat de applicatie proberen de list in te vullen met gegevens, maar als het niet lukt wordt het dus opgevangen in de catch. Dit is opgelost en bijgewerkt (zie onderstaande foto).

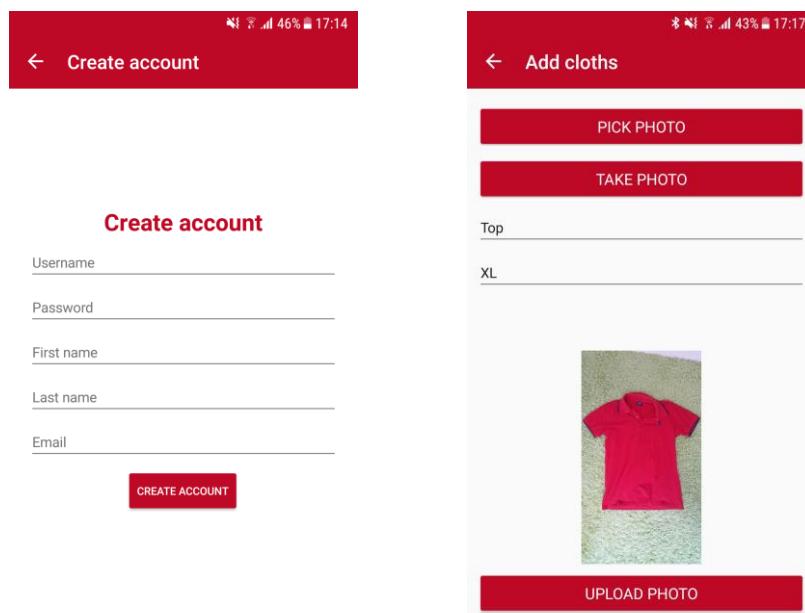
```
try
{
    CarouselViewHead.ItemsSource = gets_Head;
}
catch
{
}
```

## 4.5 Beheert de content

### Regels en procedures

Tijdens het gebruiken van de app is het logisch en makkelijk te zien wat er moet worden gebeuren. Zoals een input veld met een placeholder waar de gebruiker meteen kan zien wat hier nu in hoort te staan. Ook wordt er door de app zelf geregeld dat het niet mogelijk is om een letter in te voeren bij een prijs input.

Voor het uploaden van images zijn er regels voor opgeschreven dat de gebruiker altijd een type kleding aangeeft zoals head, top, bottom en feet. Ook wordt hier gecontroleerd of het een image file is. Als het niet overeenkomt dan wordt dit aangepakt en geeft de applicatie een error message aan. Omdat zulke situaties door de app zelf wordt aangepakt zijn hier geen regels voor gedocumenteerd.



Voor de volgende programmeur die hier aan verder gaat werken is er ook volledige documentatie geschreven over de WEB-API hier wordt uitgelegd hoe het werkt en wat de programmeur te wachten staat. Zie **Bijlage F**.

### Correcte opmaak

Xamarin forms zorgt ervoor dat de layout responsive blijft voor elk ondersteunende apparaat. Ook images dat in het applicatie wordt weergegeven zou voor elke soort resolutie moeten aanpassen dat het goed wordt weergegeven. Dit zijn meestal hoge resolutie png images die dus niet de kwaliteit verliest als het heel groot wordt weergegeven. Als het gaat om kleding die door de gebruikers wordt geüpload dan worden de images altijd op een vaste formaat weergegeven, zodat er niet veel data wordt gebruikt tijdens het laden van de foto's. Dit vermindert ook de bekende java memory error.

## Correcte rechten

Zoals elke app installatie vraagt de app welke permissions het nodig heeft voordat het kan worden geïnstalleerd. Dit kan nog niet gedemonstreerd worden omdat de app nog niet compleet af is. Er is wel een rechten en voorwaarden pagina gemaakt dat na het creëren van een account tevoorschijn komt. Als de gebruiker dit pagina sluit gaan we ervan uit dat het is geaccepteerd. Indien ze er niet mee eens zijn is het hun eigen keuze om wel of niet de applicatie te gebruiken.

## Beheerde content

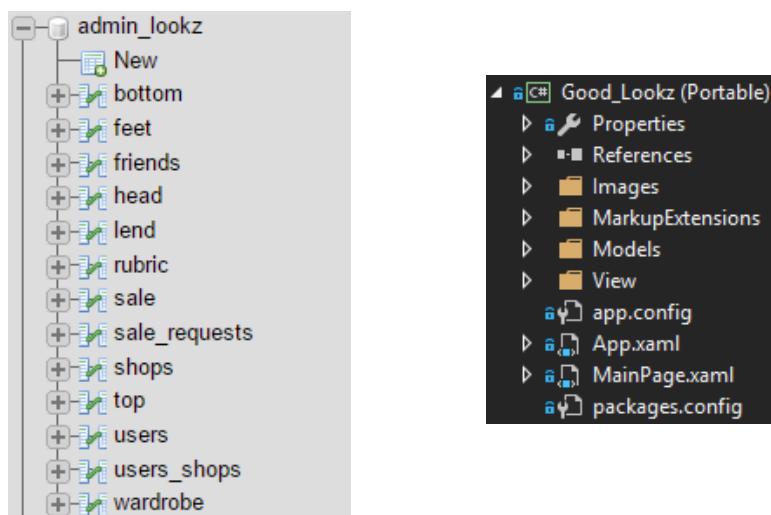
De content dat wordt beheerd door de gebruiker zelf zijn de accounts die worden aangemaakt, foto's die worden geüpload en items die voor sale staan. De gebruiker heeft zelf macht over de geüploade foto's, zo is het ook mogelijk om ze te verwijderen. Bij het verwijderen worden de foto's ook echt definitief van de database en server verwijderd. Ook de items die voor sale worden gezet zijn dan ook mogelijk om deze te verwijderen. Om deze content te beheren heb je geen speciaal software bij nodig dan alleen de app zelf. Zo beheert de gebruiker zijn of haar gegevens.

Filename	Filesize	Filetype	Last modified	Permissions	Owner/Group
..					
delete.php	1.877	PHP File	19-5-2017 12:09:29	0644	501 501
f0ibllpcwq.jpg	6.479	JPG File	19-5-2017 12:11:15	0644	501 501
Hjv3r7VrY.jpg	13.142	JPG File	23-5-2017 15:42:56	0644	501 501
MxUNznTTN.jpg	13.142	JPG File	19-5-2017 12:11:28	0644	501 501
u13HNxCIRd.jpg	5.550	JPG File	19-5-2017 12:11:24	0644	501 501

Voor de ontwikkelaar zelf is eigenlijk alles te beheren wat beheerd kan worden. Een paar voorbeelden zijn dan de gegevens van alle gebruikers in het database en geüploade foto's van alle gebruikers. Dit is ook in het rechten en voorwaarden beschreven dat 4People hier toegang voor heeft.

De software om dit content te beheren zijn:

- Visual Studio voor de applicatie zelf.
- FileZilla voor de WEB-API en voor alle geüploade foto's van de gebruiker.
- PhpMyAdmin voor de gegevens in het database.



## 4.6 Documenteert en archiveert gegevens

De applicatie en het WEB-API staan allebei lokaal opgeslagen in het computer bij 4People Communications. Ook staat de WEB-API opgeslagen in het Cloud server van 4People Communications.

WEB-API is opgeslagen in de Cloud server Good-Lookz. Om hier bij te kunnen is er een software nodig genaamd FileZilla.

Alle documentatie wordt achtergelaten in het bedrijf. Een belangrijk document is de handleiding WEB-API (**Bijlage F**). Deze is nodig om uitleg te geven over de WEB-API zodat de volgende ontwikkelaar er direct mee kan werken.

WEB-API documentatie wordt gearchiveerd in het lokaal project van de WEB-API zelf.

account	6-5-2017 14:05	File folder
friends	9-5-2017 16:21	File folder
includes	31-3-2017 13:59	File folder
lend	23-5-2017 10:03	File folder
notifications	9-5-2017 16:21	File folder
sale	23-5-2017 10:03	File folder
shops	3-5-2017 17:00	File folder
wardrobe	3-4-2017 19:44	File folder
wardrobeSet	23-4-2017 18:53	File folder
array.php	5-4-2017 22:30	PHP File 1 KB
index.php	28-4-2017 16:48	PHP File 2 KB
test.php	23-5-2017 10:03	PHP File 37 KB
uploads.php	23-5-2017 10:02	PHP File 1 KB
WEB API documentatie.docx	23-5-2017 13:16	Microsoft Word D... 49 KB

De applicatie staat in het map Good-Lookz dat zich bevindt op het desktop. Hier staan ook de Functionele en technische ontwerpen van de applicatie. Ook staat hier een notitie over hoe de volgende gebruiker moet beginnen met een link naar de Xamarin forms documentatie.

Good Lookz	16-3-2017 13:14	File folder
packages	10-5-2017 13:25	File folder
Good Lookz.sln	16-3-2017 13:14	Microsoft Visual S... 27 KB

Notitie in het project:

**Good-Lookz**

Getting started.

Voor dat je begint adviseer ik eerst om de documentatie en de structuur van xamarin forms te leren.  
<https://developer.xamarin.com/guides/xamarin-forms/>

Na dat je een beeld hebt met hoe het werkt zou je nu aan de slag kunnen.

Applicatie bestaat uit XAML en C#. Ik neem aan dat je al wat kennis hierover hebt anders zou je dat even moeten opbouwen met C# tutorials.

De XAML is voor de front-end view. C# is voor de backend (code-behind). C# classes zitten allemaal in de models map. View samen met de backend zitten in View map. Probeer deze structuur aan te houden.

Als je wat vragen hebt heb je hier mijn email address: furkan.demirci@live.nl

## 5 Bronvermelding

<http://www.encyclo.nl>

<https://www.google.nl/>

<http://www.w3schools.com>

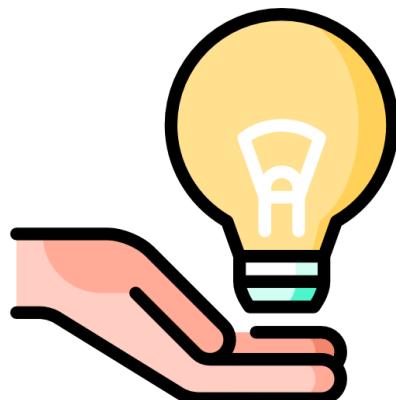
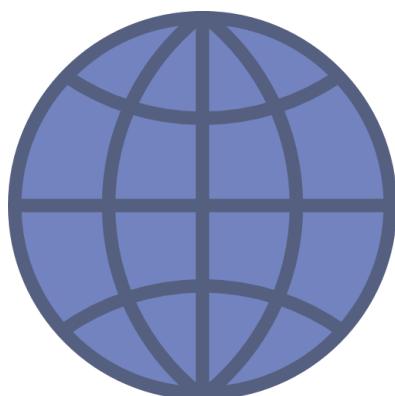
<http://www.stackoverflow.com>

<http://www.4people.nl>

<http://good-lookz.com/API/test.php>

<https://developer.xamarin.com/guides/xamarin-forms/>

[https://www.verot.net/php\\_class\\_upload.htm](https://www.verot.net/php_class_upload.htm)



## 6 Verklarende woordenlijst

Woord:	Betekenis:
API	Een application programming interface (API) is een verzameling definities op basis waarvan een computerprogramma kan communiceren met een ander programma of onderdeel (meestal in de vorm van bibliotheken). Vaak vormen API's de scheiding tussen verschillende lagen van abstractie, zodat applicaties op een hoog niveau van abstractie kunnen werken en het minder abstracte werk uitbesteden aan andere programma's. Hierdoor hoeft bijvoorbeeld een tekenprogramma niet te weten hoe het de printer moet aansturen, maar roept het daarvoor een gespecialiseerd stuk software aan in een bibliotheek, via een afdruk-API.
GET	Bij computers is GET een verzoek methode ondersteund door het HTTP-protocol dat door het World Wide Web wordt gebruikt. Door het ontwerp vraagt de GET-aanvraagmethode dat een webserver de gegevens die in het bericht van het verzoekbericht zijn opgenomen, waarschijnlijk zullen accepteren voor het opslaan. Het wordt vaak gebruikt bij het uploaden van een bestand of bij het indienen van een ingevuld webformulier.
POST	Bij computers is POST een verzoek methode ondersteund door het HTTP-protocol dat door het World Wide Web wordt gebruikt. Door het ontwerp vraagt de POST-aanvraagmethode dat een webserver de gegevens die in het bericht van het verzoekbericht zijn opgenomen, waarschijnlijk zullen accepteren voor het opslaan. Het wordt vaak gebruikt bij het uploaden van een bestand of bij het indienen van een ingevuld webformulier.
Xamarin forms	Xamarin forms is een cross-platform UI toolkit waarmee ontwikkelaars gemakkelijk indeling van gebruikersinterface kunnen maken die over Android, iOS en Windows Phone kunnen worden gedeeld.
XAML	XAML is een declaratieve opmaaktaal die kan worden gebruikt om gebruikersinterfaces te definiëren. De gebruikersinterface is gedefinieerd in een XML-bestand met behulp van de XAML-syntaxis, terwijl het runtime gedrag is gedefinieerd in een apart code achter bestand.

## **Bijlage A – Informatiebehoefte Project**

Een app voor Android, iOS en Windows Mobile.

De app heet Good-Lookz en bestaat uit 4 delen:

1. Loginsysteem met vriendenlijst
2. Frontcamera-spiegelfunctie
3. Garderobe met sharefunctie
4. Webshop

### **1 - Loginsysteem met vriendenlijst**

Inloggen is mogelijk als de gebruiker een eigen account aanmaakt. Met een account kan de gebruiker zoeken naar andere accounts en als een vriend opslaan. Zo is het mogelijk om garderobes met elkaar te koppelen.

### **2 – Frontcamera-spiegelfunctie:**

Een functie waar de gebruiker de frontcamera kan openen en als spiegel kan gebruiken. Er komen ook extra functies in zoals flash light-rand om het camerascherm, een zoomfunctie en het moet selfies kunnen maken.

### **3 - Garderobe met sharefunctie:**

Bij een garderobe kan een gebruiker de kledingstukken opslaan en delen met vrienden die met elkaar gekoppeld zijn. Gekoppelde vrienden kunnen de gedeelde kledingen zien en in een interface kledingstukken met elkaar bekijken. Ook zit er een functie in waar gekoppelde vrienden kledingstukken kunnen verkopen of uitlenen aan elkaar.

### **4 - Webshop**

Webshop waar bedrijven zich kunnen inschrijven en de app klanten kunnen linken naar hun website.



## Opdracht

Er zijn 3 onderdelen gemaakt van de 4:

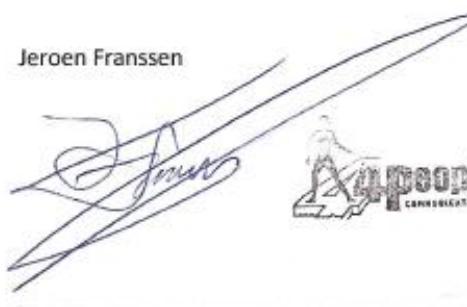
1. Loginsysteem met vriendenlijst
2. ~~Frontcamera-spiegelfunctie~~
3. Garderobe met sharefunctie
4. Webshop

Onderdeel 1, 3 en 4 waren algemeen al heel groot en is er dus geen tijd besteed aan onderdeel 2.

Om het te laten communiceren met de database moest er ook een WEB-API gemaakt worden specifiek voor Good-Lookz. De WEB-API hoort dus ook bij dit project. Gemaakte onderdelen:

1. WEB-API met JSON response
2. Loginsysteem met vriendenlijst
3. Garderobe met sharefunctie
4. Webshop met een 2<sup>de</sup> hands winkel

Jeroen Franssen



Furkan Demirci



## Bijlage B - Interview

### *Algemene vragen*

#### **Hoe kwam je aan het idee?**

Voor vrouwen is het lastig om zich eigen op te maken in de auto wat betreft belichting. Vrouwen willen graag veel kleren voor weinig geld en iedere dag wel iets anders aan. Vanuit deze 2 ideeën is het idee voor de app ontstaan. Hoe kan ik makkelijk mijn gebruikte kleding aanbieden of leuke gebruikte dingetjes vinden zoals accessoires.

#### **Wie is de doelgroep?**

Vrouwen of mannen vanaf 13 jaar tot gemiddeld 40 jaar. Er is niet echt een grens. App is downloadbaar voor iedereen.

#### **Wat staat er te verwachten?**

Multifunctioneel app die aan de boven genoemde behoeftes voldoet.

#### **Welke platformen en waarom?**

Voor mobiel Android/iOS, tablet en eventueel later op een desktop.

#### **Met hoeveel man wordt aan dit project gewerkt?**

Vormgever, project coördinator, programmeur, marketing en sales.

#### **Welke ontwikkeling omgeving wordt er gebruikt?**

Bureau en van de zaak computers met programma's zoals Visual Studio.

## **Project vragen**

### **Wat is wel belangrijk en wat is niet belangrijk?**

Mirror, wardrobe en shops zijn het belangrijkste functies.

### **In welke taal moet dit worden gecodeerd?**

De app zelf word gecodeerd in deels XAML en deels C#. Front-end XAML en Backend C#. Het WEB API word gecodeerd in PHP en SQL. Ook stuurt het een JSON terug.

### **Moet er een API worden gebouwd?**

Ja WEB API (web service), dit is nodig om een connectie te maken met het database buiten de app via het internet. Dit is te doen door middel van URI requesten te sturen en JSON response terug te krijgen.

### **Uiteindelijke resultaat?**

Vrouwvriendelijk werkende app voor Android en iOS.

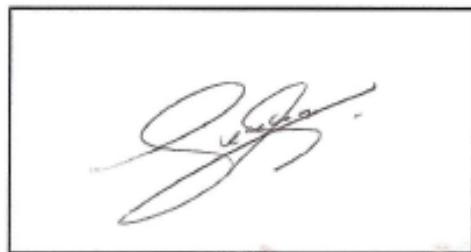
### **Moet het makkelijk uitbreidbaar kunnen worden?**

Ja het moet zo ontworpen worden dat het later makkelijk is om nieuwe functionaliteit erbij toe te voegen.

Jeroen Franssen



Furkan Demirci



## Bijlage C - Plan van aanpak

### Plan van aanpak Good-Lookz

Project : App, Xamarin C#

Opdrachtgever : 4people Communications Jeroen Fransen

Auteur : Furkan Demirci

Datum : 31-03-2017

Versie : 1.0

---

#### Inhoud

Opdracht.....	28
Projectdoelstelling.....	28
Gemaakte afspraken .....	28
Interview.....	28
Projectgrenzen .....	28
Uit te voeren activiteiten .....	29
Activiteit 1 .....	29
Activiteit 2 .....	29
Activiteit 3 .....	29
Activiteit 4 .....	30
Planning en kosten .....	31
Risico's en showstoppers .....	33
Goedkeuringen .....	33

## **Opdracht**

Good-Lookz is een app voor vrouwen waar het mogelijk is om kledingen op te slaan als een soort kleding kast en het mogelijk maakt om het te delen met je vrienden. Vrienden kunnen met elkaar verbonden worden door een vriendverzoek te sturen. Ook is het mogelijk om de kleding te bekijken van je gekoppelde vrienden.

Als er geen tijdgebrek is dan wordt er ook een mirror functie gebouwd waar vrouwen hun make-up kunnen doen en ook een foto kunnen maken. In de mirror functie is het mogelijk om je helderheid aan te passen.

Hiervoor moet ook een WEB API gebouwd worden om een connectie te maken tussen de app en de database. WEB API moet worden geschreven in PHP en moet een JSON response terug schrijven.

## **Projectdoelstelling**

De bedoeling is om een werkende app te maken met de benoemde functies:

- Login system
- Shop system
- Wardrobe system

De Front-end en Backend moet ook gemaakt worden met Visual Studio Xamarin forms framework. Front-end moet compleet worden geschreven in XAML en backend van het app moet worden geschreven in C#.

WEB API moet worden gebouwd om een connectie te maken met de database. Dit wordt gedaan door middel van URI requesten en JSON responsen.

## **Gemaakte afspraken**

Er zijn 3 hoofdonderdelen afgesproken die moeten worden gemaakt. Dat zijn de Login systeem, shop systeem en wardrobe systeem. Ook moet er een WEB API gemaakt worden met meerdere functies om het app uiteindelijk werkend te maken. Dit wordt gedaan door middel van URI requesten en JSON responses.

## **Interview**

Er is een interview gehouden met verschillende vragen. Algemene vragen en project vragen. Document zit als bijlage in het verslag (zie **Bijlage B**).

## **Projectgrenzen**

De app moet goed gecodeerd worden zodat het veilig blijft en uitbreidbaar blijft. De volgende stagiaires die dit verder moeten opbouwen is het de bedoeling dat ze er zonder moeite iets aan kunnen toevoegen. De WEB API moet ook goed gecodeerd worden voor evt. uitbreidingen later.

# **Uit te voeren activiteiten**

## **Activiteit 1**

Het creëren van de applicatie project via Visual Studio. Hier wordt eerst een Cross-platform project aangemaakt. Na het aanmaken van het project wordt er elke soort NuGet packages voor Android en iOS geüpdateert.

## **Activiteit 1.2**

Na het creëren van je project heb je de basis bestanden om er mee te kunnen beginnen. Eerst wordt er documentatie gelezen over soorten Xamarin forms lay-outs en wordt het toegepast in het project.

## **Activiteit 1.3**

Vervolgens wordt er documentatie gelezen in Xamarin forms hoe het framework nu precies werkt en wat relaties zijn tussen soorten bestanden. Hier wordt ook de informatie opgehaald hoe Front-end en Backend nu met elkaar communiceren.

## **Activiteit 2**

Het realiseren van de applicatie

### **Activiteit 2.1**

Na het ophalen van informatie over de framework en lay-out options, is het bouwen begonnen. Hier wordt als eerst de Main page gemaakt. Op deze page wordt de naam en het logo van de app getoond. Ook komen er 2 buttons met login of create account opties.

### **Activiteit 2.2**

Login venster wordt gemaakt samen met een create account venster. Alleen visueel omdat het nog niet werkend kan worden gemaakt.

### **Activiteit 2.3**

Menu page venster wordt aangemaakt met button die naar de volgende pagina's worden gelinkt. Dit wordt zowel visueel als werkend gemaakt.

### **Activiteit 2.4**

Shop page en wardrobe page worden gemaakt. Alleen visueel omdat werkend nog niet mogelijk is.

## **Activiteit 3**

Het realiseren van de WEB API. Dit is nodig om een connectie te maken met de database.

### **Activiteit 3.1**

Eerst wordt er documentatie gelezen over hoe een WEB API precies werkt en hoe het met Xamarin communiceert. WEB API wordt geschreven in PHP. API moet URI requesten krijgen en JSON string terug sturen.

### **Activiteit 3.2**

Vervolgens wordt de database geschreven met alle relaties die erbij horen. Database wordt gemaakt en gedocumenteerd in Visio.

### **Activiteit 3.3**

Login checker en account worden gemaakt en moeten een POST method krijgen. Als dit goed is uitgevoerd hoort die een JSON string terug te sturen.

### **Activiteit 3.4**

Image upload voor kleding moeten worden aangemaakt. Tijdens het uploaden wordt er een unieke naam gegeven. De image wordt opgeslagen in het server en database met het URL. Ook dit moet een JSON string terug sturen nadat het succesvol is uitgevoerd.

### **Activiteit 3.5**

Vrienden moeten met elkaar verbonden worden door middel van unieke ID's. Dit word ook opgeslagen in de database en dit hoort ook een JSON string terug te sturen.

## **Activiteit 4**

De app laten functioneren. Dit wordt gedaan doormiddel van JSON responsen op te halen en in variables te zetten.

### **Activiteit 4.1**

Login en create account functie werkend maken door URI requesten naar de WEB API te sturen en na het succesvol uitvoeren een JSON response terug krijgen. Deze data ook weer in een variabel opslaan.

### **Activiteit 4.2**

Vriendverzoek functie werkend maken door middel van ID koppelingen. Zoeken naar een specifieke email adres en daarna beide ID koppelen in de database. Dit wordt waargemaakt door de WEB API die dit voor de app doet. Stuurt JSON response ook weer terug.

### **Activiteit 4.3**

Wardrobe functie werkend maken door middel van download able images te gebruiken. Images die in de database staan horen ook een link te hebben. De link laat alleen de afbeelding zien en met Xamarin is het mogelijk om de URL als image source te gebruiken. Voor het uploaden geld: foto sturen naar het WEB API en wordt daar dan automatische bijgewerkt in de database.

### **Activiteit 4.4**

Shop functie werkend maken waar het kopen van kledingen mogelijk is. Officiële shops van toegevoegde bedrijven of 2<sup>de</sup> hands winkel waar vrienden kleding kunnen verkopen of te koop kunnen zetten.

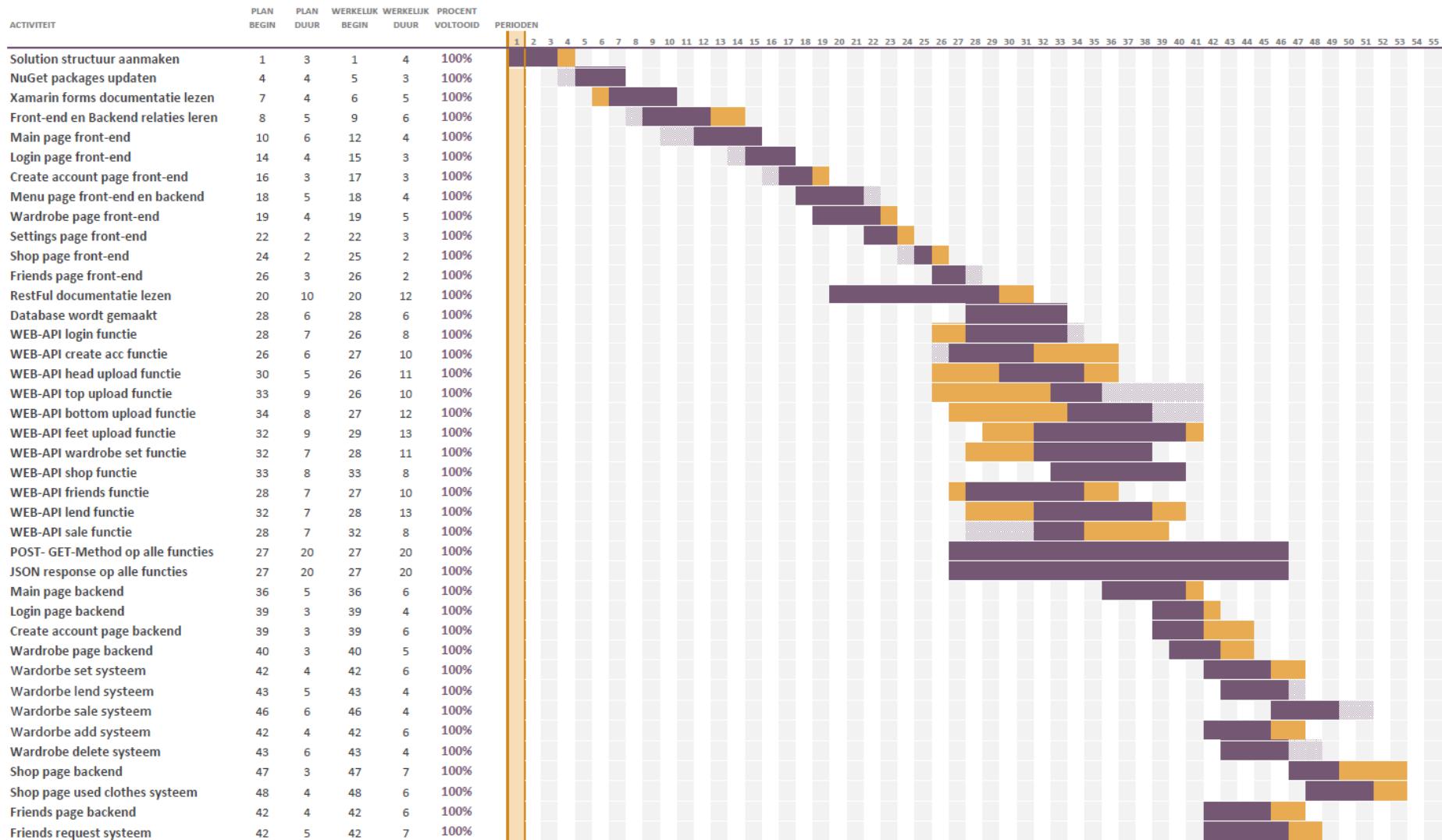
## Planning en kosten

Er waren geen extra kosten hierbij betrokken.

WBS id	predecessor	task	requirement level	developer	Risc	duration, hr (plan)	duration, hr (do)	evaluation (check)	action (act)
			(MoSCoW)						
1	1	Good-Lookz app Wireframe maken.	Must	Furkan Demirci	Onvoldoende informatie of onjuiste	5	5	Volledig Wireframes gemaakt	✓
2	2	Good-Lookz app Diagram maken.	Must	Furkan Demirci	Onvoldoende informatie of onjuiste	6	6	Diagram gemaakt	✓
3	3	Flowchart maken over de app	Must	Furkan Demirci	Onvoldoende informatie of onjuiste	20	20	Wireframes zijn de flowcharts	✓
4	4	Functioneel ontwerp invullen en gereed maken.	Must	Furkan Demirci	Onvoldoende informatie of onjuiste	10	10	Ingevuld en gereed	✓
5	5	Bestaande Database reverse engineeren en opslaan in PNG.	Must	Furkan Demirci	Onvoldoende informatie of onjuiste	0,3	0,3	Gemaakt in visio	✓
6	6	Technisch ontwerp invullen en gereed maken.	Must	Furkan Demirci	Onvoldoende informatie of onjuiste	10	10	Technische ontwerp gemaakt	✓
7	7	Beginnen met het maken van de app. Front-end only	Must	Furkan Demirci	Xamarin framework wil niet werken	60	60	Gedaan	✓
8	8	App Indeling met soorten layouts	Should	Furkan Demirci	Layouts zijn niet responsive	40	40	Gedaan	✓
9	9	Maken van de Database met connecties	Must	Furkan Demirci	Database crash, Relations die niet meer goed werken	50	50	Database volledig gemaakt	✓
10	10	WEB API maken met login en create acc functies	Must	Furkan Demirci	Onvoldoende informatie of kennis op gebied van API	50	50	WEB-API gemaakt	✓
11	11	WEB API voor foto upload en download	Must	Furkan Demirci	Onvoldoende informatie of kennis op gebied van API	50	50	WEB-API gemaakt	✓
12	12	App bestand structuur	Should	Furkan Demirci	Navigatie doet het niet meer	20	20	WEB-API gemaakt	✓
13	13	WEB API voor vriend request en shops	Must	Furkan Demirci	Onvoldoende informatie of kennis op gebied van API	50	50	WEB-API gemaakt	✓
14	14	WEB API met database connectie	Must	Furkan Demirci	Database connecties niet meer werken door ip address veranderingen	20	20	WEB-API gemaakt	✓
15	15	WEB API JSON responses	Must	Furkan Demirci	WEB API stuurt verkeerde JSON op die niet valid zijn	20	20	WEB-API gemaakt	✓
16	16	Eigen sqlite van de app	Should	Furkan Demirci	Geen connectie tussen mobiel database en app	18	0	Niet van toepassing	X
17	17	App aparte galerij	Should	Furkan Demirci	Haalbaarheid hangt er van af	20	0	Niet van toepassing	X
18	18	Implementatie plan schrijven	Must	Furkan Demirci	Onvoldoende informatie of onjuiste	6	6	Gemaakt en uitgevoerd	✓
19	19	Test plan schrijven	Must	Furkan Demirci	niet alle onderdelen zijn testbaar of zijn vergeten	7	7	Testplan geschreven	✓
20	20	Acceptatie test schrijven en opstellen/uitvoeren	Must	Furkan Demirci	Dat er stiekem toch veel bugs en problemen eruit komen.	2	2	Gedaan en uitgevoerd	✓
total						464,3	426,3		

## Good-Lookz app en WEB-API

Periodetrend: 1 Plan Werkelijk % Voltooid Werkelijk (meer dan gepland) % Voltooid (meer dan gepland)



## Risico's en showstoppers

Er zijn risico's die we kunnen tegenkomen zoals:

- Beveiliging leaks
- WEB API SQL injecties
- Database verbinding niet meer werkt

Showstoppers:

- Xamarin framework wil niet meer goed functioneren
- Computer crash (Back-ups worden gemaakt)
- Een vastloper waardoor ik niet meer verder kom (extra hulp moet dan worden ingeschakeld)

## Goedkeuringen

**Opdrachtgever:** 4People Communications

**Opdrachtnemer:** Furkan Demirci

**Vertegenwoordiger:** Jeroen Franssen

**Vertegenwoordiger:** Furkan Demirci

**Datum:** 31-03-2017

**Datum:** 31-03-2017

Handtekening:


Handtekening:



## Bijlage D - Functioneel ontwerp

# Functioneel Ontwerp

## Good-Lookz

datum: 31-03-2017

Versie: V.1

door: Furkan Demirci

klas: IC.14AOA

### Goedkeuring

Naam: 4People Communications	Naam: Furkan Demirci
Functie: Manager	Functie: Applicatie Ontwikkelaar
Datum: 31-03-2017	Datum: 31-03-2017
Goedgekeurd (handtekening)	Goedgekeurd (handtekening)



## Inhoud

Inleiding .....	36
Business perspectief.....	36
Huidige situatie.....	36
Gewenste situatie.....	36
Business requirements .....	36
Gebruikersperspectief .....	37
Diagram .....	37
Wireframes.....	38
Main venster .....	38
Login venster .....	39
Account aanmaak venster.....	39
Menu venster .....	41
Mirror venster .....	43
Wardrobe venster .....	44
Settings venster .....	46
Over dit document .....	53
Afkortingen.....	53
Gebruikte materialen .....	53

## Inleiding

Dit functioneel ontwerp is in opdracht van 4People Communications. Voor de applicatie Good-Lookz. Good-Lookz is een App voor Android/iOS waar klanten zich voor kunnen aanmelden. In de app is het mogelijk om de wardrobe functie te gebruiken om zijn kleding te kunnen opslaan en delen met vrienden. Niet alleen de app zelf moet gerealiseerd worden, maar ook de WEB API om connectie te hebben met de database van de app. De WEB API moet zo worden geprogrammeerd dat het alleen URL requesten moet krijgen en na het uitvoeren van de gewenste taak een JSON terug stuurt naar de app.

Om een duidelijk beeld te krijgen van wat er gerealiseerd moet worden, moesten er verschillende wireframes en usecases gebouwd worden. Voor het maken van de usecases en de wireframes heb ik twee verschillende programma's gebruikt. Voor de usecases heb ik Visio 2013 gebruikt en voor de wireframes heb ik Pencil project gebruikt.



Visio 2013



Pencil project

## Business perspectief

### Huidige situatie

In de business perspectief zien we een grote kans. Een kans dat dit een hit gaat worden voor man en vrouw. Je hebt sociaal media zoals facebook en twitter om natuurlijk van alles met elkaar te delen, maar specifiek op kleding heb je niet, dus dacht 4People communications om een app te maken waar mensen zijn kledingen kunnen opslaan en delen met vrienden. Waarom een app en niet een website? 4People communications ziet de toekomst in apps, vooral omdat er vaak veel apps wordt gedownload via de appstore of Play store. De app wordt gratis om te gebruiken maar de klant moet zich alleen registreren om er gebruik van te maken.

### Gewenste situatie

Dat vrouwen en mannen deze app gaan gebruiken in hun dagelijkse leven. Om kledingen met elkaar te kunnen delen, uitlenen of verkopen.

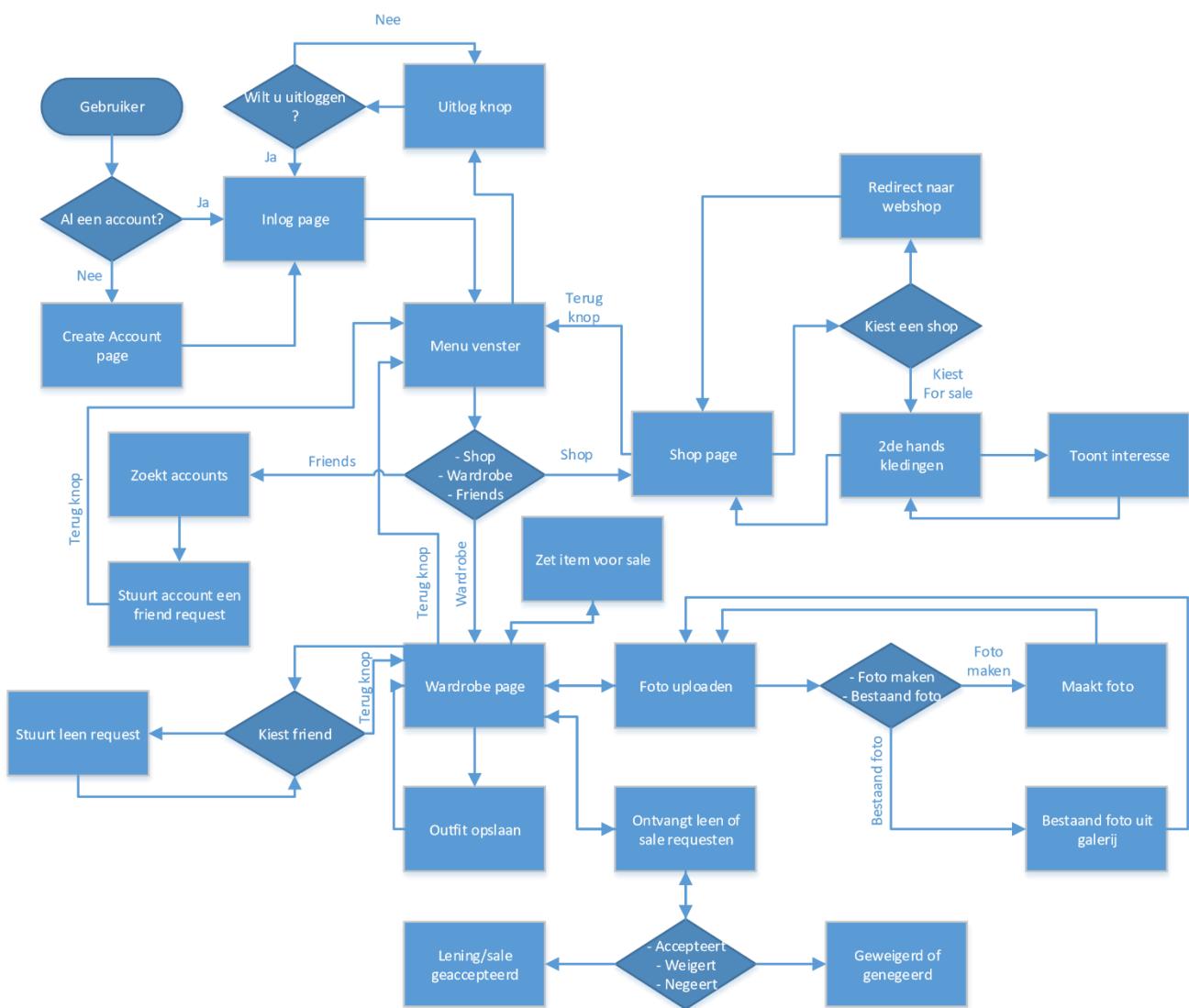
### Business requirements

De opdrachtgever wil een werkende app met een wardrobe functie om je kledingen op te slaan en om te delen onder je vrienden. Om dit allemaal waar te maken moet er een WEB API worden gebouwd om een connectie te maken met de database.

# Gebruikersperspectief

Hier ziet u de diagram over de gehele app. Voor een beter overzicht staan alle links onder de bijlage kop. Hier in de diagram is te zien hoe de app functioneert met een gebruiker. Gebruiker begint eerst bij het inloggen en gaat dan direct door naar het menu venster om daar ook nog uit mogelijk heden te kunnen kiezen.

## Diagram

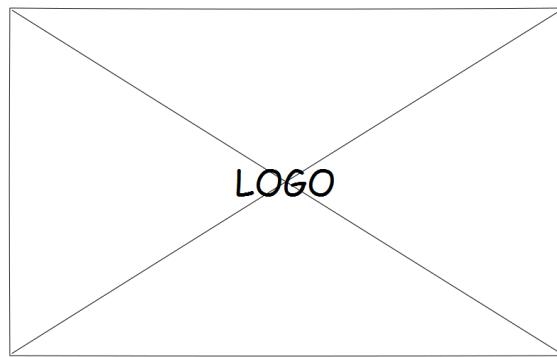


## Wireframes

Ook zijn er wireframes gemaakt om een soort beeld te geven hoe de functionaliteit eruit gaat zien. Wireframes zijn gemaakt in het programma Pencil.

### Main venster

Als de gebruiker de app voor het eerst opent is er een mogelijkheid om in te loggen of om een account aan te maken. Hier wordt de gebruiker dan verwezen naar een inlog venster of een account aanmaak venster.



### Login venster

Als de gebruiker al een account heeft kan hij/zij inloggen in de app. Dit hoeft maar eenmalig, omdat je account dan wordt opgeslagen op je telefoon.



## Login

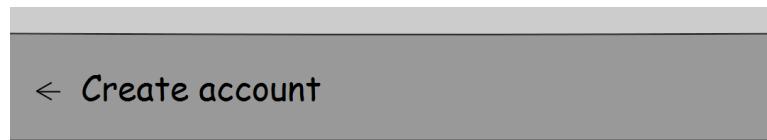
The login form consists of three main components: a "Username" input field, a "Password" input field, and a central "LOGIN" button. The "Username" field is positioned above the "Password" field. Both fields are represented by white rectangular boxes with thin black borders. The "LOGIN" button is located below the password field, enclosed in a light gray rectangular box with a slightly darker gray border.

Username
Password
LOGIN

### Account aanmaak venster

Hier is het mogelijk om een account aan te maken. Na het aanmaken van het account kan de gebruiker inloggen en verder gebruik maken van de app.

Hier zijn nog extra invulvelden bijgekomen zoals First name, Last name en Email.



## Create account

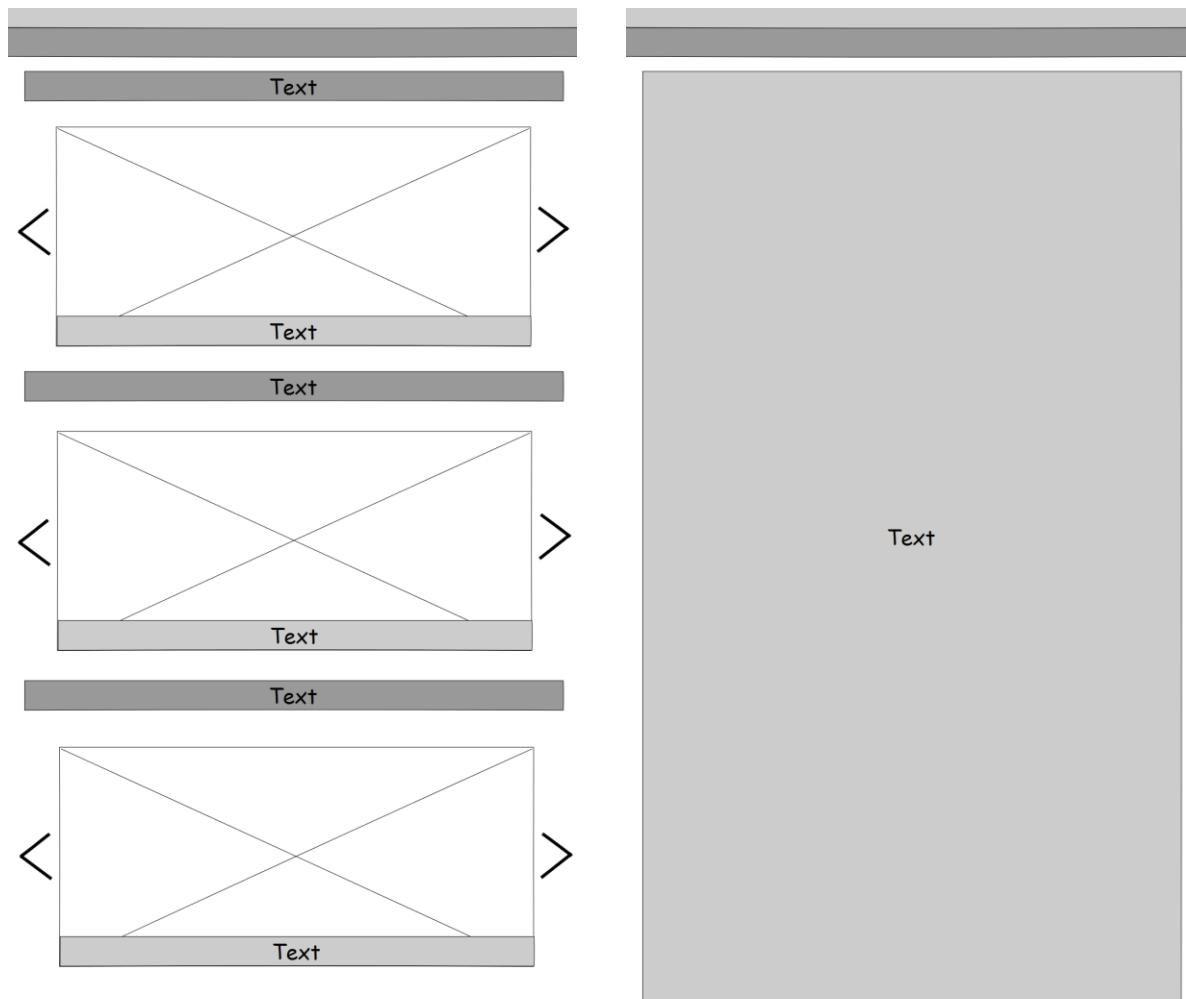
Username

Password

CREATE ACCOUNT

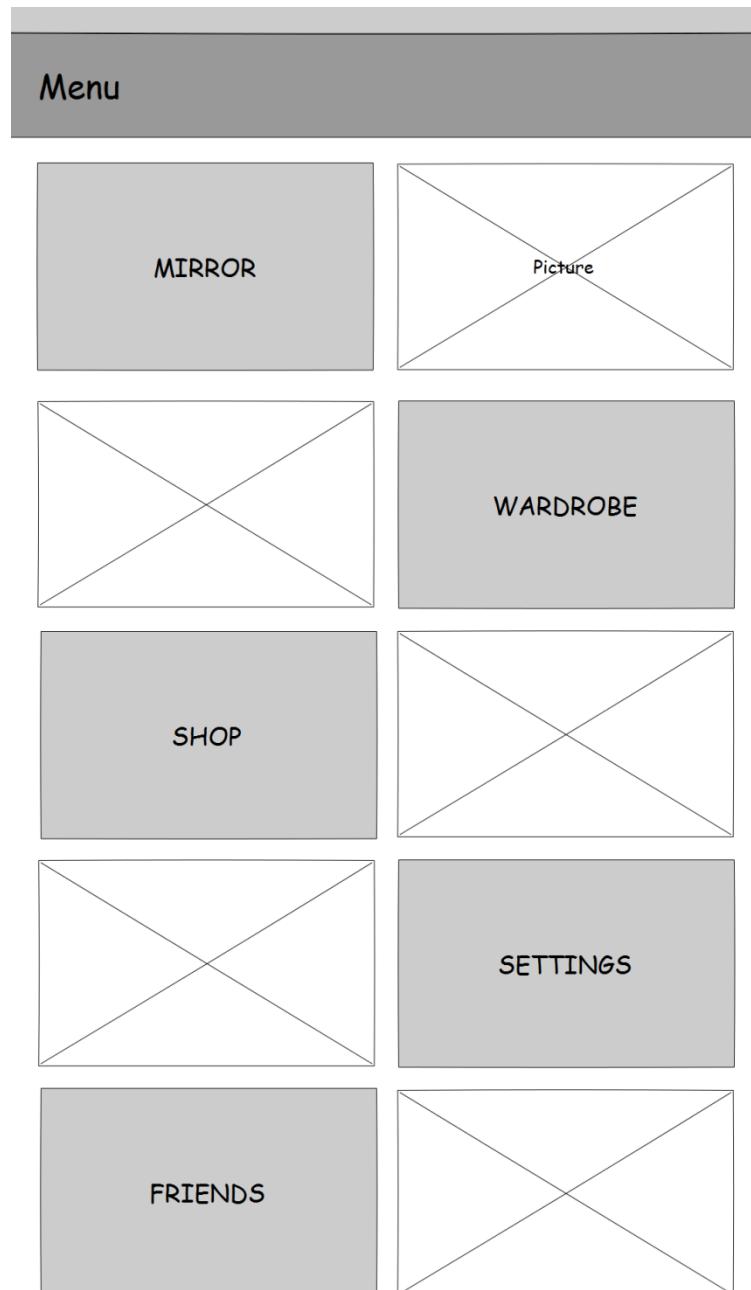
### Eerste keer ingelogd

Als een gebruiker voor de eerste keer inlogt komen er 2 pagina's tevoorschijn. Eerste pagina is het kiezen van de shops die bij de gebruiker past. Tweede pagina zijn de rechten en voorwaarden te zien.



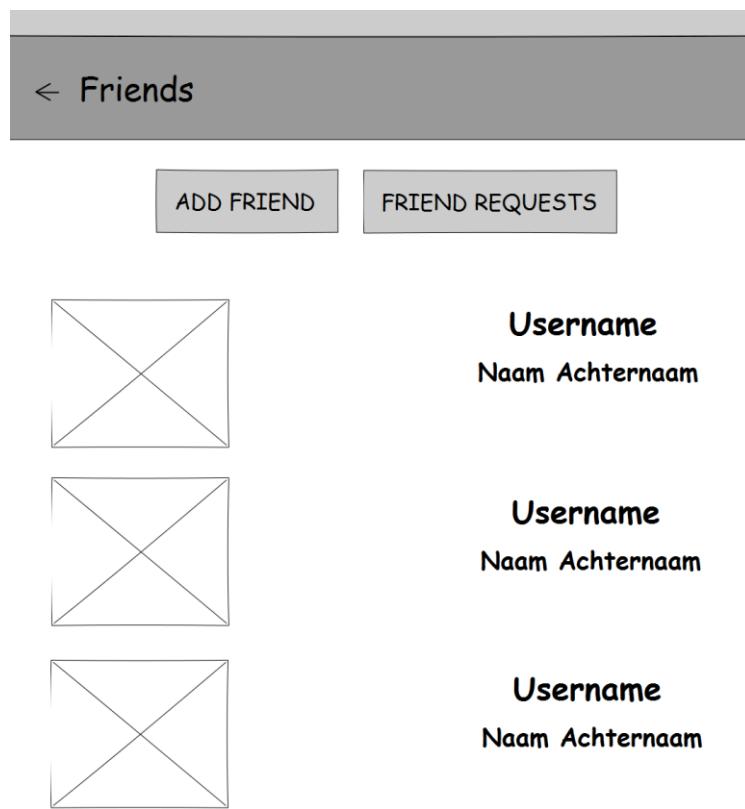
### Menu venster

Na het creëren van het account of na het inloggen met een bestaand account kom je in het menu venster. Hier is het mogelijk om uit bepaalde opties te kiezen. Zoals hieronder aangegeven: Wardrobe, Shop, Settings en Friends.



### Friends venster

In het Friends venster is het mogelijk om accounts op te zoeken en als een vriend toe te voegen. Er wordt dan eerst een request gestuurd en daarna geaccepteerd door de ontvangene. Ook is het mogelijk om vrienden te verwijderen.



### Add friends en friends request venster

Hier is het mogelijk om users op te zoeken met een username. Dit wordt telkens geladen tijdens het typen. Na dat de gebruiker het account heeft kunnen vinden is het mogelijk om daarna een friend request te sturen.

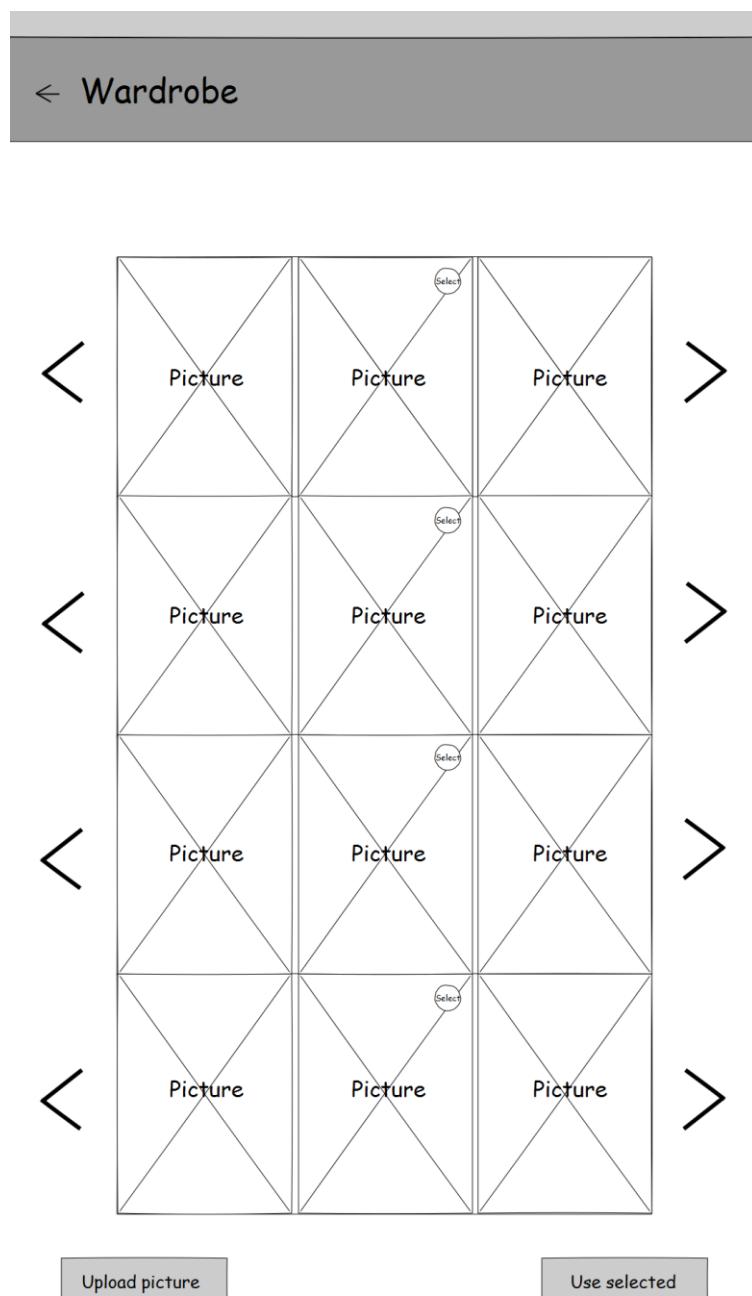
Na het sturen van een request krijgt de gebruiker een friends request binnen. Hier kan de gebruiker de friend request accepteren, weigeren of negeren.

< Add friends		< Friends requests	
Search users			
<input type="checkbox"/>	Username	<input type="checkbox"/>	Username
<input type="checkbox"/>	Username	<input type="checkbox"/>	Username
<input type="checkbox"/>	Username	<input type="checkbox"/>	Username
<input type="checkbox"/>	Username	<input type="checkbox"/>	Username

### Wardrobe venster

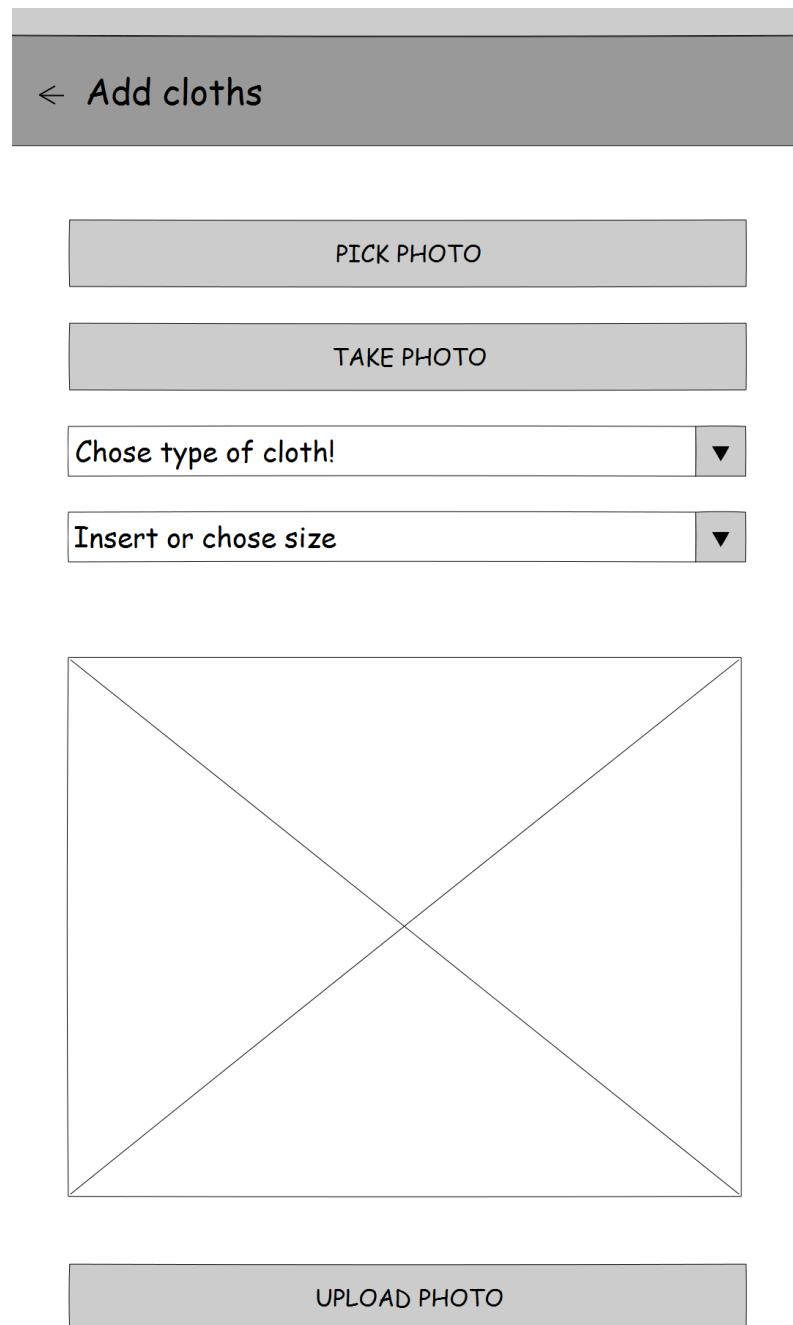
Hier in het wardrobe venster kan de gebruiker foto's uploaden van zijn kledingen. Dit doe je door een bestaande foto te kiezen uit de galerij of door een nieuwe foto te maken via de camera. Hier is het dan mogelijk dat de gebruiker de geselecteerde kleding kan opslaan als een set en is het mogelijk om kledingen uit te lenen aan zijn of haar vrienden.

Er zijn nog 5 extra buttons gekomen. Boven is er een Friends button waar de gebruiker de kleding van zijn vrienden kan bekijken. Boven is er ook een item selling button gekomen waar de gebruiker kan bekijken welke van zijn items er voor sale staan. Onder aan de pagina zijn er 3 buttons. Een Add button waar de gebruiker kleding kan uploaden. Een Save set button waar de gebruiker de gekozen kleding als een set kan opslaan. Als laatst een Sets button voor het bekijken van alle opgeslagen sets



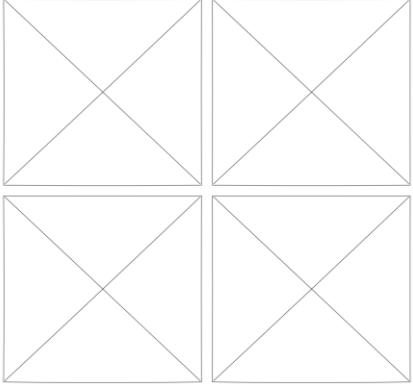
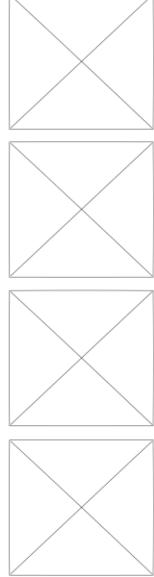
### Wardrobe Add venster

Hier kan een gebruiker vanuit de galerij of via de camera app een foto kiezen/maken. Tijdens het uploaden vraagt de app om voor een type kleding te kiezen en om een size mee te geven.



**Set vensters**

Hier zijn de pagina's te zien waar de gebruiker zijn gekozen items als een set kan opslaan en een naam kan geven. In de set list venster staan alle opgeslagen sets van de gebruiker met naam en datum. Als er een item is geselecteerd in de list dan wordt er een venster geopend met de kleding en de naam dat voor die set is opgeslagen.

<a href="#">← Save set</a>	<a href="#">← Sets</a>	<a href="#">← Selected set</a>								
	<table border="1"><thead><tr><th>Set name</th><th>Date</th></tr></thead><tbody><tr><td>Set name</td><td>Date</td></tr><tr><td>Set name</td><td>Date</td></tr><tr><td></td><td></td></tr></tbody></table>	Set name	Date	Set name	Date	Set name	Date			
Set name	Date									
Set name	Date									
Set name	Date									
<input type="text" value="Set name"/> <input type="button" value="SAVE"/>		<input type="text" value="Set name"/>								

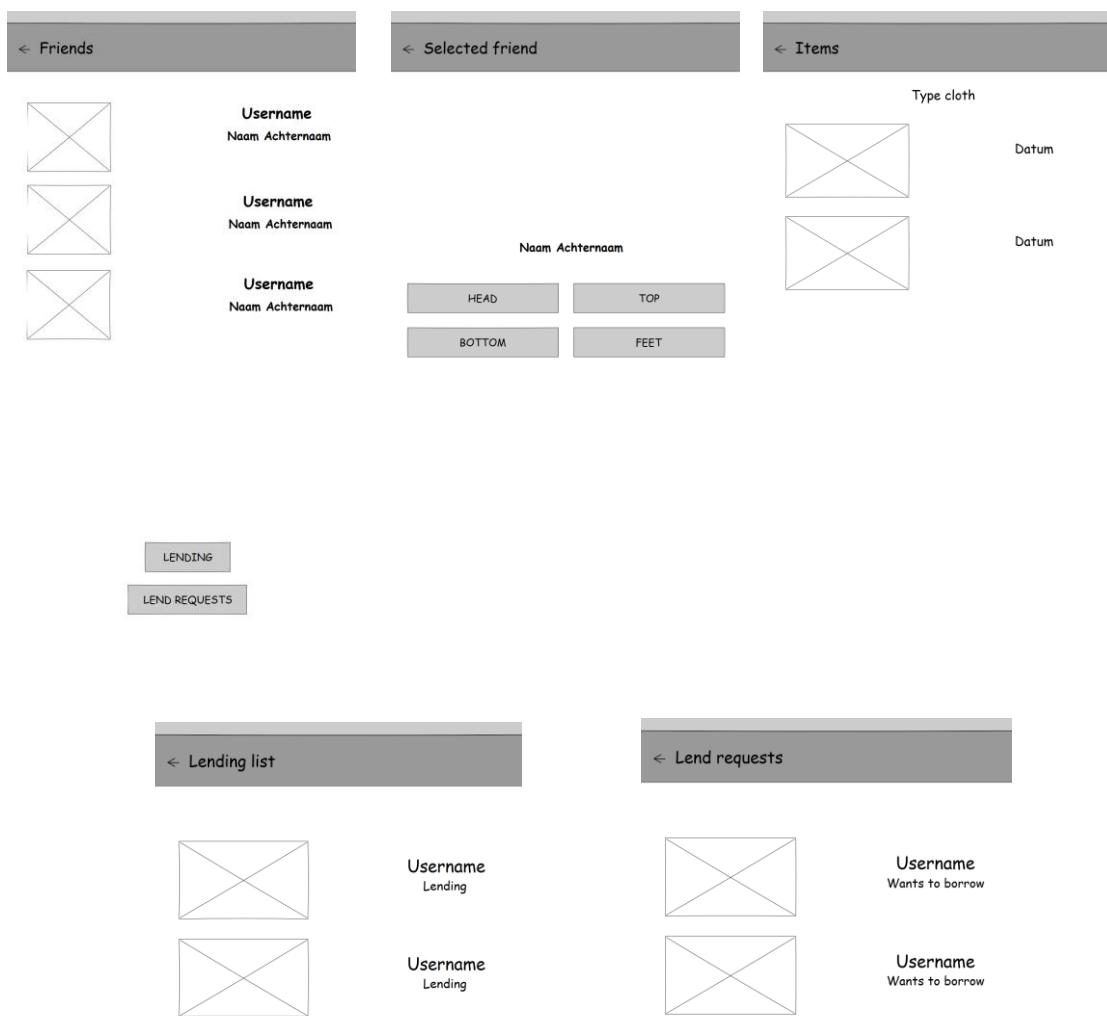
## Wardrobe friends venster

In de friend page zijn alle vrienden van de gebruiker te zien. Hier is het mogelijk om van een vriend de items te bekijken. Hier is het ook mogelijk om een request te sturen of de gebruiker een kledingstuk mag lenen.

In de Lending list venster staan alle items die de gebruiker op dat moment aan het uitlenen is. De gebruiker kan ook items in de list verwijderen.

In de LendRequest venster komen alle lend requesten te staan. Hier is het ook mogelijk om het te accepteren, weigeren of negeren.

Het nieuwe lend systeem dat na de incidentmelding is toegepast doet precies hetzelfde als de vorige systeem. Wat er is veranderd aan het systeem is alleen de view van de kleding. Nu kan de gebruiker alle kleding zien in een swipe functie zoals bij de Wardrobe venster.



### Wardrobe item selling

In de item selling venster staan alle items dat de gebruiker voor sale heeft gezet. Hier is het ook mogelijk om het te verwijderen. In de sale request venster komen alle sale requesten te staan. Hier is het ook mogelijk om het te verwijderen. Na het openen van een request komen er gegevens te staan zoals prijs en bied prijs.

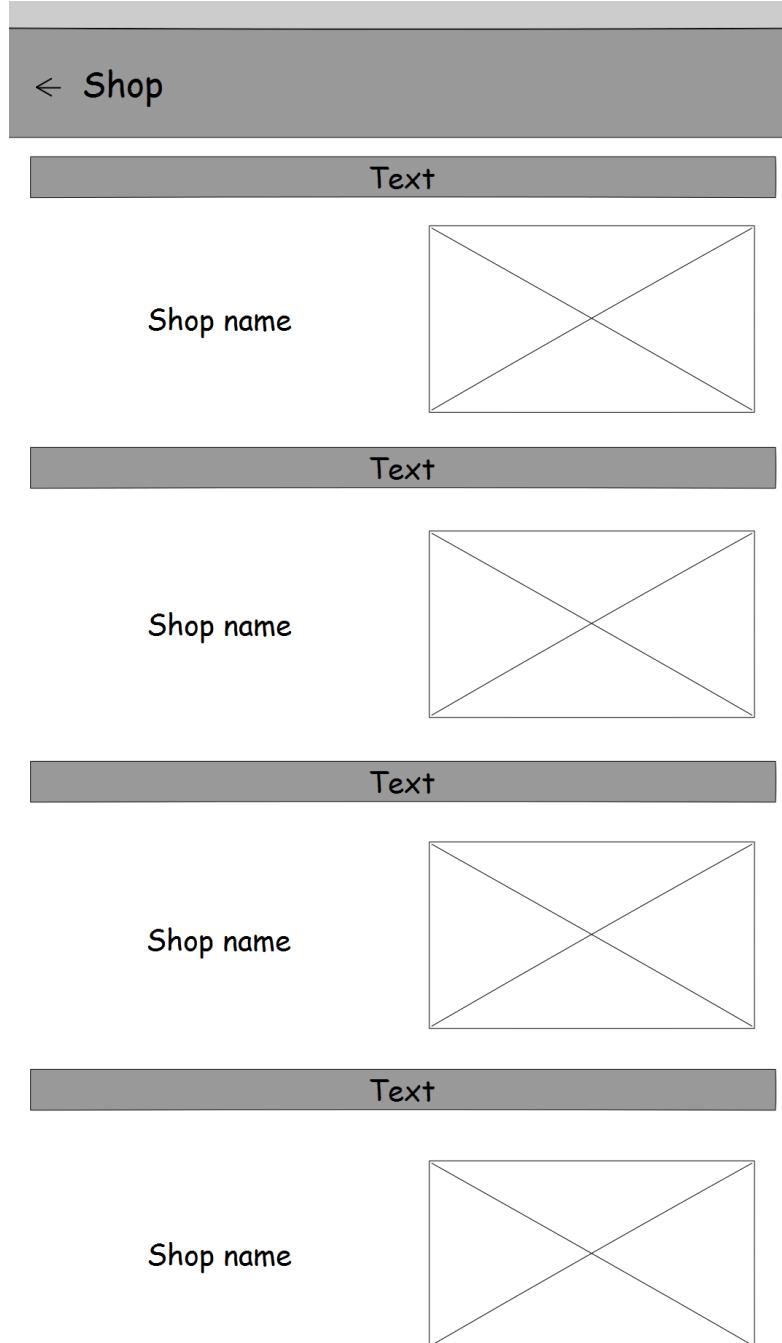
The screenshot shows a mobile application interface with three main sections:

- Items selling:** Shows two items, each with a delete icon (cross) and fields for Price and Description.
- Sale requests:** Shows two requests, each with a delete icon (cross) and fields for Username and Price.
- Selected request:** A detailed view of a selected request. It shows the Username and Price, and includes fields for Own price: Price and Bidding: Price. It also displays a message from the user: "Username Is interested in your item".

At the bottom, there are buttons for "SALE REQUESTS", "Communicating can be started using mail", and "SEND EMAIL".

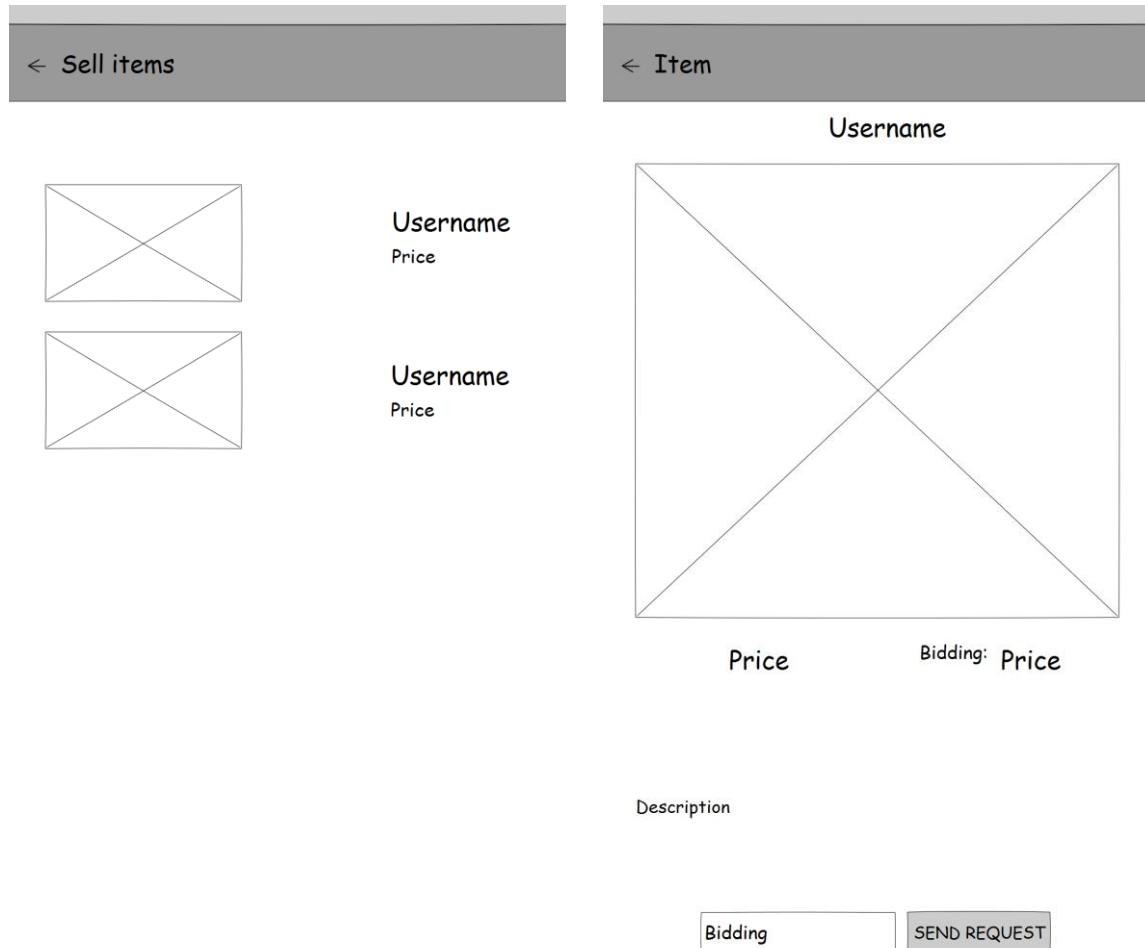
### Shop venster

Hier in de shop venster zijn de gekozen shops van de gebruiker per rubriek te zien en is het mogelijk om naar de webshop van de bedrijven te gaan. Dit kan per gebruiker verschillen. In de laatste shop staan de used clothes van alle gekoppelde vrienden. Hier is het mogelijk om interesse te tonen aan een kledingstuk dat de gebruiker zou willen kopen.



### Used Clothes venster

Hier in de used clothes venster staat alle kleding die voor sale staan van alle vrienden. Als de gebruiker geïnteresseerd is in een item is het mogelijk om een request te sturen met of zonder een bieding.



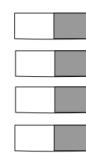
## Settings venster

Hier in het settings venster is het mogelijk om je instellingen van de app te wijzigen en hier kan de gebruiker ook informatie over de app vinden.



### Notify

- New friend request
- For sell notify
- Request
- Vibration



### Change

- Loyalty brands >
- Body size >
- For sell distance >

### Help

- Problem >
- Helpdesk >

### Privacy

- Privacy policy >
- Terms and Conditions >

### The app

- Cooperate >
- Feed back >
- Share this app >

## Over dit document

### Afkortingen

Afkorting	Omschrijving
API	Een application programming interface (API) is een verzameling definities op basis waarvan een computerprogramma kan communiceren met een ander programma of onderdeel (meestal in de vorm van bibliotheken).
Database	Een database, gegevensbank of databank is een digitaal opgeslagen archief, ingericht met het oog op flexibele raadpleging en gebruik. Databases spelen een belangrijke rol voor het archiveren en actueel houden van gegevens bij onder meer de overheid, financiële instellingen en bedrijven, in de wetenschap, en worden op kleinere schaal ook privé gebruikt.
Wireframes	Een applicatie wireframe, ook bekend als een pagina of scherm schematische blauwdruk, is een visuele gids die het geraamte van een applicatie vertegenwoordigt.

### Gebruikte materialen

Onderdeel	Omschrijving
Pencil project	Pencil Project is een programma. Waarin je gemakkelijk van allerlei verschillende thema's kunt gebruiken om bijv. een wireframe te maken of een 2D ontwerp, voor een website of applicatie etc.
Visio 2013	Microsoft Visio is een applicatie voor het maken van technische en logische schema's. Het programma is bedoeld voor technici en stelt deze in staat om relatief eenvoudig stroomschema's, (kantoor)plattegronden, databasemodellen en andere schematische documenten en diagrammen te maken.
Word 2016	Microsoft Word, of meestal alleen Word, is een van de meest gebruikte tekstverwerkers ter wereld. Het is gemaakt door Microsoft.

## Bijlage E - Technisch ontwerp

# Technische Ontwerp

## Good-Lookz

datum: 31-03-2017

Versie: V.1

door: Furkan Demirci

klas: IC.14AOA

### Goedkeuring

Naam: 4People Communications	Naam: Furkan Demirci
Functie: Manager	Functie: Applicatie Ontwikkelaar
Datum: 31-03-2017	Datum: 31-03-2017
Goedgekeurd (handtekening)	Goedgekeurd (handtekening)



The table contains four rows of information. The first row has two columns: 'Naam: 4People Communications' and 'Naam: Furkan Demirci'. The second row has two columns: 'Functie: Manager' and 'Functie: Applicatie Ontwikkelaar'. The third row has two columns: 'Datum: 31-03-2017' and 'Datum: 31-03-2017'. The fourth row has two columns: 'Goedgekeurd (handtekening)' and 'Goedgekeurd (handtekening)'. Below the table, there are two handwritten signatures. The left signature is over the first column of the fourth row, and the right signature is over the second column of the fourth row. There is also a small logo for '4PEOPLE COMMUNICATIONS' at the bottom of the left column.

## Inhoud

Inleiding .....	56
Eisen .....	56
Beslissingen .....	56
Afspraken.....	56
Plan van Aanpak .....	56
Haalbaarheidsstappen.....	56
Datamodel.....	57
Database structuur.....	57
URI requesten en JSON response .....	58
Over dit document .....	59
Afkortingen.....	59
Gebruikte materialen .....	59

## Inleiding

Dit technische ontwerp is in opdracht van 4People Communications. Voor de app Good-Lookz. Good-Lookz is een App voor Android/iOS waar klanten zich voor kunnen aanmelden. In de app is het mogelijk om de wardrobe functie te gebruiken om zijn kleding te kunnen opslaan en delen met vrienden. Niet alleen de app zelf moet gerealiseerd worden, maar ook de WEB API om connectie te krijgen met de database van de app. De WEB API moet zo worden geprogrammeerd dat het alleen URL requesten moet krijgen en na het uitvoeren van de gewenste taak een JSON terug stuurt naar de app. Hier in het technische ontwerp ga ik de technische kant van de app uitleggen.

## Eisen

De app Good-Lookz moet op een Android en iOS werken. De applicatie moet gebruiksvriendelijk en simpel gemaakt worden. De app wordt compleet gecodeerd in XAML en C# via Xamarin forms. Dit is een framework voor Visual Studio om makkelijk apps te creëren voor cross-platform. XAML is voor het Front-End (GUI) en C# is voor het Backend. De app moet data kunnen opslaan zoals accounts en foto's. Dit doen we doormiddel van een WEB API. De WEB API word compleet geschreven in PHP en moet URI requesten krijgen van de app via C#. In response stuurt de WEB API een JSON string terug en moet de app het kunnen lezen via C#. In dit WEB API worden query's gebouwd om met de database te communiceren. Uploaden van foto's moet via C# gestuurd worden naar de WEB API. Om foto's te lezen (bijvoorbeeld in het wardrobe venster) is het mogelijk om de foto's te downloaden van de database via een URL.

## Beslissingen

Waarom we Xamarin forms gebruiken is omdat C# populair en leuk taal is om mee te coderen en ook omdat het heel makkelijk is om cross-platform te maken door het functionaliteit te delen met dezelfde code. Er moet hiervoor een WEB API gebouwd worden, omdat het anders niet mogelijk is om met de database op een veilige manier te communiceren.

## Afspraken

Ik ben de enige programmeur die aan de app werkt. Alle benoemde functies moet worden gemaakt door mij. Wat ik niet moet maken is het CMS van het WEB API, dit word door de volgende stagiaire gemaakt.

## Plan van Aanpak

Plan van aanpak is in een apart bestand geschreven. De interview met de praktijk begeleider staat in **Bijlage B**. Zie **Bijlage C** voor de plan van aanpak.

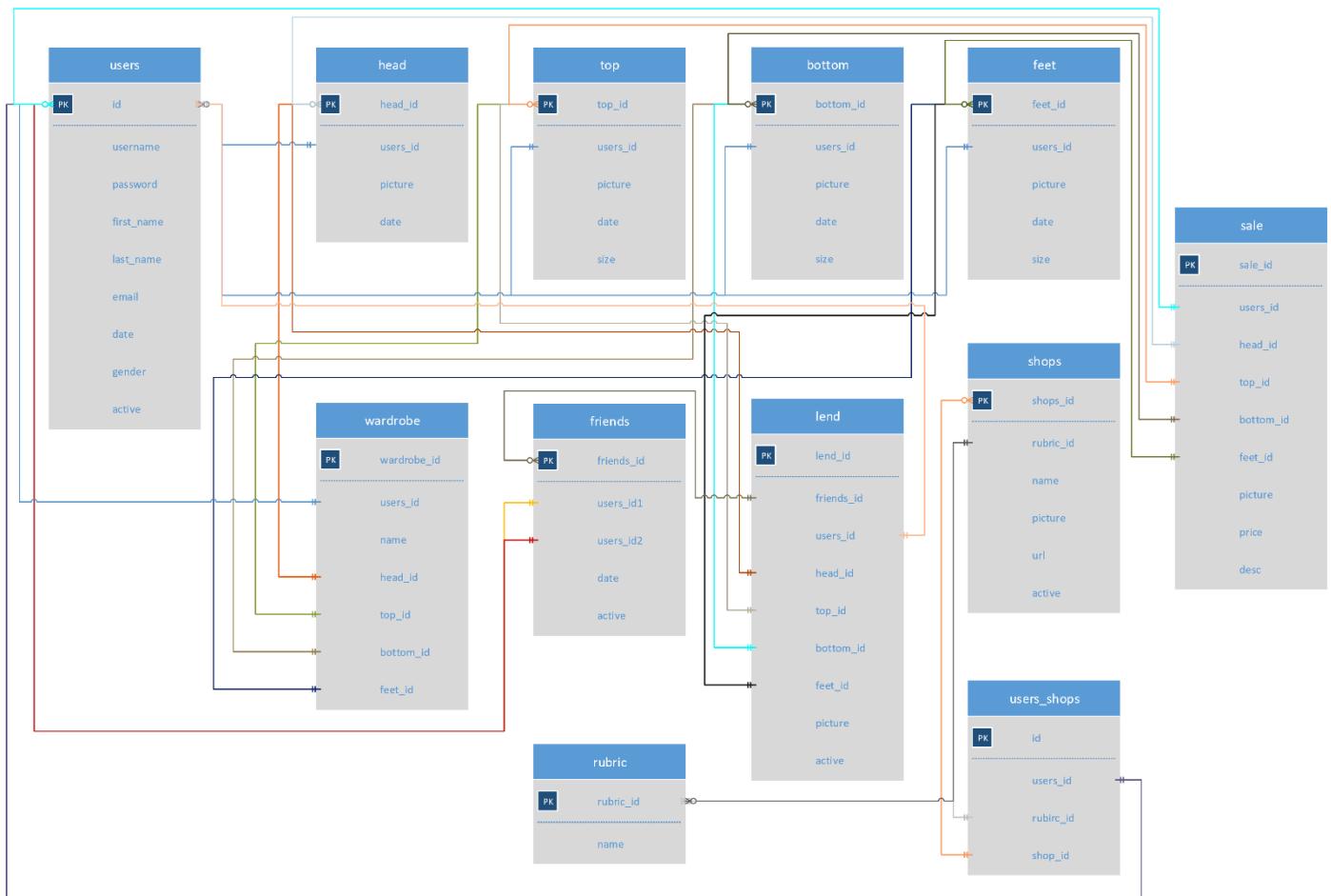
## Haalbaarheidsstappen

De app wordt getest op virtuele emulators of op onze eigen mobiele apparaten. Ook wordt er getest of er communicatie is tussen de app en WEB API. Dit doormiddel van URI requesten en JSON responsen. In de WEB API zelf wordt er ook getest of er een connectie is met het database. Als er een fout ontstaat wordt dit altijd vermeld met een FALSE response.

## Datamodel

### Database structuur

Hieronder is de communicatie aangegeven tussen tabellen. Samen met de WEB API word de verbindingen ook via PHP SQL query's geschreven.



## URI requesten en JSON response

De app moet kunnen communiceren met een WEB API. Dit moet door middel van URI requesten. Je hebt soorten requesten zoals:

- GET – this operation is used to retrieve data from the web service.
- POST – this operation is used to create a new item of data on the web service.
- PUT – this operation is used to update an item of data on the web service.
- PATCH - this operation is used to update an item of data on the web service by describing a set of instructions about how the item should be modified. This verb is not used in the sample application.
- DELETE - this operation is used to delete an item of data on the web service.

De WEB API moet ook data kunnen terug sturen. Dit moet worden gedaan doormiddel van JSON strings. Een voorbeeld is:

```
{"id": "19", "username": "ThisIsAnExample", "password": "meatbal123", "first_name": "Furkan", "last_name": "Demirci", "email": "test@test.test", "date": "2017-03-30", "gender": "0", "active": "1"}
```

De app moet dit lezen door middel van JSON decode in C# en in variables zetten.

JSON response recht gezet is het gewoon te zien dat dit data is via strings:

```
{
    "id": "19",
    "username": "ThisIsAnExample",
    "password": "meatbal123",
    "first_name": "Furkan",
    "last_name": "Demirci",
    "email": "test@test.test",
    "date": "2017-03-30",
    "gender": "0",
    "active": "1"
}
```

## Over dit document

### Afkortingen

Afkorting	Omschrijving
API	Een application programming interface (API) is een verzameling definities op basis waarvan een computerprogramma kan communiceren met een ander programma of onderdeel (meestal in de vorm van bibliotheken).
Database	Een database, gegevensbank of databank is een digitaal opgeslagen archief, ingericht met het oog op flexibele raadpleging en gebruik. Databases spelen een belangrijke rol voor het archiveren en actueel houden van gegevens bij onder meer de overheid, financiële instellingen en bedrijven, in de wetenschap, en worden op kleinere schaal ook privé gebruikt.
URI	Een uniform resource identifier ( <b>URI</b> ), is een internet-protocolelement, gebaseerd op eerdere voorstellen van Tim Berners-Lee. Het is een eenmalige benaming van een "bron", een stuk informatie, data of dergelijke.
JSON	<b>JSON</b> of JavaScript Object Notation, is een gestandaardiseerd gegevensformaat. <b>JSON</b> maakt gebruik van voor de mens leesbare tekst in de vorm van data-objecten die bestaan uit een of meer attributen met bijbehorende waarde.

### Gebruikte materialen

Onderdeel	Omschrijving
Visio 2013	Microsoft Visio is een applicatie voor het maken van technische en logische schema's. Het programma is bedoeld voor technici en stelt deze in staat om relatief eenvoudig stroomschema's, (kantoor)plattegronden, databasemodellen en andere schematische documenten en diagrammen te maken.
Word 2016	Microsoft Word, of meestal alleen Word, is een van de meest gebruikte tekstverwerkers ter wereld. Het is gemaakt door Microsoft.

## Bijlage F – WEB-API documentatie

# WEB-API Documentatie

## Good-Lookz

datum: 31-03-2017

Versie: V.1

door: Furkan Demirci

klas: IC.14AOA

**RESTful API**  
GET PUT POST DELETE

## Inhoud

Inleiding .....	62
Create account: .....	62
Check account: .....	62
Shop toevoegen:.....	63
Shops download: .....	63
Friends send request: .....	64
Friends receive request: .....	64
Friends accept request: .....	64
Friends check request: .....	65
Friends delete request: .....	65
Wardrobe head upload: .....	66
Wardrobe head download: .....	68
Wardrobe head delete: .....	68
Wardrobe top upload:.....	69
Wardrobe top download:.....	69
Wardrobe top delete:.....	69
Wardrobe bottom upload: .....	70
Wardrobe bottom download: .....	70
Wardrobe bottom delete: .....	70
Wardrobe feet upload:.....	71
Wardrobe feet download:.....	71
Wardrobe feet delete:.....	71
Wardrobe set upload: .....	72
Wardrobe set download: .....	72
Wardrobe set update: .....	72
Wardrobe set delete: .....	73

## Inleiding

De WEB API is gemaakt met PHP om data op te slaan in de database. Dit is een communicatie middel van de app Good-Lookz met de database. WEB API bestaat uit GET- en POST-requesten hoort JSON strings met data terug te sturen.

### Create account:

POST methode sturen met:

```
$username = $_POST['username'];
$password = $_POST['password'];
$first_name = $_POST['first_name'];
$last_name = $_POST['last_name'];
$email = $_POST['email'];
$gender = $_POST['gender'];
```

Bekijkt daarna of de ingevulde username of email al bestaat in de database. Als het al bestaat stuurt die een JSON terug:

```
[{"exists":true}]
```

Als het niet bestaat in de database wordt er een nieuwe record aangemaakt in de database. Ook wordt de laatste record dat gemaakt is verwijderd uit de database. (Dit doet die omdat er voor een rare reden 2 keer in de database wordt geüpload). Vervolgens stuurt die een JSON response terug.

```
[{"created":true}]
```

### Check account:

POST methode sturen met:

```
$loginName = $_POST['loginName'];
$password = $_POST['password'];
```

Controleert in het database of de ingevoerde gegevens overeenkomen met een account. Als die een account heeft gevonden en als de active op 1 staat stuurt die een JSON response terug met alle gegevens van de gebruiker:

```
[{"id": "33", "username": "ErikDoeRustig", "password": "123456", "first_name": "Erik", "last_name": "Rustig", "email": "geenmail@lol.com", "date": "2017-04-06", "gender": "0", "offline": "0", "active": "1"}]
```

Als active op 0 staat betekent dit dat het account geblokkeerd is:

```
[{"blocked":true}]
```

Als het account niet bestaat dan stuurt die een account false terug:

```
[{"account":false}]
```

## Shop toevoegen:

Om een shop te uploaden is er POST methode nodig (dit gebeurt alleen via het CMS en dus niet via de app):

```
$shopName = $_POST['shopName'];
$shopURL = $_POST['shopURL'];
$rubricSelect = $_POST['rubricSelect'];
$_FILES['fileToUpload'];
```

ShopName: Naam van de winkel.

ShopURL: Link om naar de webshop van de winkel te komen.

RubricSelect: Een van de 3 rubrieken wordt meegegeven (Care, Fashion en Accessories).

FileToUpload: Foto bestand die wordt meegegeven.

Na het uploaden wordt er een echo aangegeven (Geen JSON omdat dit allemaal via de CMS gaat):

```
echo "New record created successfully";
echo "The file " . basename($_FILES["fileToUpload"]["name"]) . " has been
uploaded.;"
```

## Shops download:

Get request nodig om shops te downloaden:

Rubriek 1, 2 of 3 sturen:

URL = [http://www.good-lookz.com/API/shops/shopsDownload.php?rubric\\_id=1](http://www.good-lookz.com/API/shops/shopsDownload.php?rubric_id=1) te openen en wordt er een JSON response met alle shops van dat rubriek terug gestuurd:

```
[{"shops_id": "17", "name": "H&M", "picture": "http://www.good-
lookz.com/API/shops/img/q5LrrvBkrwhm.jpg", "url": "www.hm.nl", "active": "1"}, {"shops_id": "16", "name": "Zara", "picture": "http://www.good-
lookz.com/API/shops/img/unuHeTSWWYzara.jpg", "url": "www.zara.com", "active": "1"}]
```

### **Friends send request:**

Een POST methode sturen met een users\_id waarmee de gebruiker al is ingelogd via de app:

```
$usrOrMail = $_POST['usrOrMail'];
$users_id = $_POST['users_id'];
```

Daarna kijkt een checker of het een username of mail is. Vervolgens in de database zoeken naar het ingevulde UsrOrMail. Als de UsrOrMail overeen komt met de database stuurt die een JSON response terug:

```
[{"friends_send":true}]
```

Als er iets fout gaat in de database tijdens het inserten:

```
[{"friends_send":false}]
```

Als de ingevulde naam niet overeen komt:

```
[{"friends_name":false}]
```

### **Friends receive request:**

Een GET methode sturen met een users\_id waarmee de gebruiker al is ingelogd via de app:

```
$users_id2 = $_GET['users_id'];
```

JSON response:

```
[{"id":"19","username":"furkan75","first_name":"furkan","last_name":"demirci","friends_id":"16"}]
```

Als er geen friends requesten zijn:

```
[{"friends_receive":false}]
```

### **Friends accept request:**

Een POST methode sturen met een accepted true of false en het friends\_id van de gekozen request:

```
$accepted = $_POST['accepted'];
$friends_id = $_POST['friends_id'];
```

Bij true wordt active naar 1 gezet en bij false wordt de request verwijderd. Ook dit stuurt JSON terug:

```
[{"friends_accept":true}]
```

```
[{"friends_decline":true}]
```

Als alles mis gaat:

```
[{"error":"accepted has no value"}]
```

### **Friends check request:**

Een GET methode sturen met een users\_id waarmee de gebruiker al is ingelogd via de app:

```
$users_id = $_GET['users_id'];
```

In de friends table wordt er gezocht naar de gestuurde users\_id als er iets gevonden is waar active ook op 1 staat wordt er een JSON response gestuurd:

```
[{"id": "1", "username": "Furkan075", "first_name": "Furkan", "last_name": "Demirci", "friends_id": "17"}]
```

Als er geen verbindingen zijn in de database:

```
[{"friends_check": false}]
```

### **Friends delete request:**

Een POST methode sturen van het gekozen friends\_id:

```
$friends_id = $_POST['friends_id'];
```

Daarna wordt het meteen verwijderd in de database. Stuurt ook een JSON response:

```
[{"friends_delete": true}]
```

Als het mis gaat:

```
[{"friends_delete": false}]
```

## Image upload

Voor het uploaden van images wordt er een script gebruikt dat volledige controle geeft over de geüploade foto. Hier kan bijvoorbeeld de naam, size, formaat en kwaliteit veranderd worden van de geüploade foto's.

Script documentatie is hier te vinden:

<https://github.com/verot/class.upload.php/blob/master/README.md>

### Korte uitleg

In de documentatie van het script staan er een paar voorbeelden zoals:

```
$foo = new Upload($_FILES['form_field']);
if ($foo->uploaded) {
    // save uploaded image with no changes
    $foo->Process('/home/user/files/');
    if ($foo->processed) {
        echo 'original image copied';
    } else {
        echo 'error : ' . $foo->error;
    }
    // save uploaded image with a new name
    $foo->file_new_name_body = 'foo';
    $foo->Process('/home/user/files/');
    if ($foo->processed) {
        echo 'image renamed "foo" copied';
    } else {
        echo 'error : ' . $foo->error;
    }
    // save uploaded image with a new name,
    // resized to 100px wide
    $foo->file_new_name_body = 'image_resized';
    $foo->image_resize = true;
    $foo->image_convert = gif;
    $foo->image_x = 100;
    $foo->image_ratio_y = true;
    $foo->Process('/home/user/files/');
    if ($foo->processed) {
        echo 'image renamed, resized x=100
              and converted to GIF';
        $foo->Clean();
    } else {
        echo 'error : ' . $foo->error;
    }
}
```

Een simple script dat volledige controle geeft over het geüploade foto. Wat in de WEB-API gebeurd is de resize van een afbeelding samen met een x grootte en de naam van de image wordt veranderd.

Zo staat het in de WEB-API beschreven:

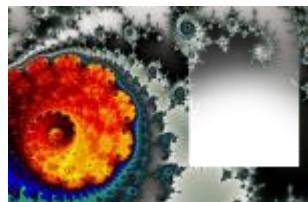
```
$random1 =
substr(str_shuffle(str_repeat("0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ", 10)), 0, 10);

$handle->image_resize = true;
$handle->image_ratio_y = true;
$handle->image_x = 300;
$handle->file_new_name_body = $random1;
```

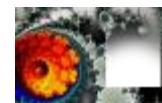
Eerst wordt er dus een random string aangemaakt en daarna de size met een nieuwe naam. Wat er daarna volgt is de query die het in de database stopt.

```
$sql = 'INSERT INTO admin_lookz.head (users_id, picture, date) VALUES (' .
$users_id . ', "http://good-lookz.com/API/wardrobe/head/img/' . $handle-
>file_dst_name . '", "' . date("Y-m-d") . '")';
if (mysqli_query($conn, $sql)) {
    $row_array['img_upload'] = true;
} else {
    $row_array['img_database'] = false;
}
```

## Image voorbeelden



image/png – 150 x 100 – 35.25KB



image/png – 75 x 50 – 9.75KB

## Wardrobe head upload:

Een POST methode sturen met een users\_id waarmee de gebruiker al is ingelogd via de app:

```
$users_id = $_POST['users_id'];  
$FILES['fileToUpload']['name'];
```

Na het ontvangen van een users\_id en een foto wordt het meteen geüpload in de database en wordt er een JSON response gestuurd:

```
[{"img_upload":true}]
```

Als het fout gaat:

```
[{"img_upload":true}]
```

Als img wel is geüpload maar niet in de database is gezet:

```
[{"img_database":false}]
```

## Wardrobe head download:

Een GET methode sturen met een users\_id waarmee de gebruiker al is ingelogd via de app:

```
$users_id = $_GET['users_id'];
```

Ophalen gaat via de database en stuurt een JSON response:

```
[{"head_id":"30","users_id":"23","picture":"http://good-lookz.com/API/wardrobe/head/img/55uJk8C31Jhead.jpg","date":"2017-04-07"}]
```

## Wardrobe head delete:

Een POST methode met een head\_id die geselecteerd is in de app:

```
$head_id = $_POST['head_id'];
```

Na het succesvol verwijderen in de database en server wordt er een JSON response gestuurd:

```
[{"head_delete":true}]
```

Als het niet succesvol is:

```
[{"head_delete":"No image to delete"}]
```

## Wardrobe top upload:

Een POST methode sturen met een users\_id waarmee de gebruiker al is ingelogd via de app:

```
$users_id = $_POST['users_id'];
$size = $_POST['size'];
$_FILES['fileToUpload']['name'];
```

Na het ontvangen van een users\_id en een foto wordt het meteen geüpload in de database en wordt er een JSON response gestuurd:

```
[{"img_upload":true}]
```

Als het fout gaat:

```
[{"img_upload":false}]
```

Als img wel is geüpload maar niet in de database is gezet:

```
[{"img_database":false}]
```

## Wardrobe top download:

Een GET methode sturen met een users\_id waarmee de gebruiker al is ingelogd via de app. Eventueel kan er ook een size mee worden gestuurd om het te filteren.

```
$users_id = $_GET['users_id'];
$size = $_GET['size'];
```

Ophalen gaat via het database en stuurt een JSON response:

```
[{"top_id":"25","users_id":"23","picture":"http://good-lookz.com/API/wardrobe/top/img/u6lgH3pEFilife-line-deer-dames-trui-met-noorse-print-en-wind.jpg","date":"2017-04-07","size":"S"}]
```

## Wardrobe top delete:

Een POST methode met een top\_id die geselecteerd is in de app:

```
$top_id = $_POST['top_id'];
```

Na het succesvol verwijderen in de database en server wordt er een JSON response gestuurd:

```
[{"top_delete":true}]
```

Als het niet succesvol is:

```
[{"top_delete":"No image to delete"}]
```

## Wardrobe bottom upload:

Een POST methode sturen met een users\_id waarmee de gebruiker al is ingelogd via de app:

```
$users_id = $_POST['users_id'];
$size = $_POST['size'];
$_FILES['fileToUpload']['name'];
```

Na het ontvangen van een users\_id en een foto wordt het meteen geüpload in de database en wordt er een JSON response gestuurd:

```
[{"img_upload":true}]
```

Als het fout gaat:

```
[{"img_upload":false}]
```

Als img wel is geüpload maar niet in de database is gezet:

```
[{"img_database":false}]
```

## Wardrobe bottom download:

Een GET methode sturen met een users\_id waarmee de gebruiker al is ingelogd via de app. Eventueel kan er ook een size mee worden gestuurd om het te filteren.

```
$users_id = $_GET['users_id'];
$size = $_GET['size'];
```

Ophalen gaat via de database en stuurt een JSON response:

```
[{"bottom_id":"26","users_id":"24","picture":"http://good-lookz.com/API/wardrobe/bottom/img/u8ppWT4k8CHerenbroeken-Meyer-Korte-broek-Palma-incl-riem.jpg","date":"2017-04-07","size":"234"}]
```

## Wardrobe bottom delete:

Een POST methode met een bottom\_id die geselecteerd is in de app:

```
$bottom_id = $_POST[bottom_id];
```

Na het succesvol verwijderen in het database en server wordt er een JSON response gestuurd:

```
[{"bottom_delete":true}]
```

Als het niet succesvol is:

```
[{"bottom_delete":"No image to delete"}]
```

## Wardrobe feet upload:

Een POST methode sturen met een users\_id waarmee de gebruiker al is ingelogd via de app:

```
$users_id = $_POST['users_id'];
$size = $_POST['size'];
$_FILES['fileToUpload']['name'];
```

Na het ontvangen van een users\_id en een foto wordt het meteen geüpload in de database en wordt er een JSON response gestuurd:

```
[{"img_upload":true}]
```

Als het fout gaat:

```
[{"img_upload":false}]
```

Als img wel is geüpload maar niet in het database is gezet:

```
[{"img_database":false}]
```

## Wardrobe feet download:

Een GET methode sturen met een users\_id waarmee de gebruiker al is ingelogd via de app. Eventueel kan er ook een size mee worden gestuurd om het te filteren.

```
$users_id = $_GET['users_id'];
$size = $_GET['size'];
```

Ophalen gaat via de database en stuurt een JSON response:

```
[{"feet_id": "18", "users_id": "25", "picture": "http://good-lookz.com/API/wardrobe/feet/img/zPKuKL61eWwww.emmenmode.nl-Antony-Morato-MMFW00367-sneaker-schoenen-blauw-005991-70-32.jpg", "date": "2017-04-07", "size": "43"}]
```

## Wardrobe feet delete:

Een POST methode met een feet\_id die geselecteerd is in de app:

```
$feet_id = $_POST[feet_id];
```

Na het succesvol verwijderen in de database en server wordt er een JSON response gestuurd:

```
[{"feet_delete":true}]
```

Als het niet succesvol is:

```
[{"feet_delete": "No image to delete"}]
```

## Wardrobe set upload:

Een POST methode met de volgende gegevens:

```
$users_id = $_POST['users_id'];
$name = $_POST['name'];
$head_id = $_POST['head_id'];
$top_id = $_POST['top_id'];
$bottom_id = $_POST['bottom_id'];
$feet_id = $_POST['feet_id'];
```

Na het inserten in de database wordt er een JSON response gestuurd:

```
[{"set_upload":true}]
```

Als het misgaat stuurt die een SQL error.

## Wardrobe set download:

Een POST methode sturen met een users\_id waarmee de gebruiker al is ingelogd via de app:

```
$users_id = $_POST['users_id'];
```

Gaat controleren in de database of er iets bestaat. Als er iets bestaat wordt de volgende JSON response gestuurd:

```
[{"wardrobe_id":"10","users_id":"17","name":"Winter_set","head_id":"43","top_id":"25","bottom_id":"14","feet_id":"73"}]
```

Als het niet bestaat in de database wordt er ook niks terug gestuurd.

## Wardrobe set update:

Een POST methode met de volgende gegevens:

```
$wardrobe_id = $_POST['wardrobe_id'];
$name = $_POST['name'];
$head_id = $_POST['head_id'];
$top_id = $_POST['top_id'];
$bottom_id = $_POST['bottom_id'];
$feet_id = $_POST['feet_id'];
```

Na het updaten in de database wordt er een JSON response gestuurd:

```
[{"set_update":true}]
```

Als het fout gaat:

```
[{"set_update":false}]
```

### Wardrobe set delete:

Een POST methode met een wardrobe\_id voor om een specifiek set te verwijderen of een users\_id om alle sets te verwijderen (dus niet allebei!).

```
$wardrobe_id = $_POST['wardrobe_id'];
$users_id = $_POST['users_id'];
```

Als het verwijderen uit de database gelukt is wordt er een JSON response gestuurd:

```
[{"set_delete":true}]
```

Als het mislukt is:

```
[{"set_delete":false}]
```

## Bijlage G – Test plan

# TEST PLAN

## Good-Lookz

datum: 31-03-2017

Versie: V.1

door: Furkan Demirci

klas: IC.14AOA



# TEST PLAN

## Documenthistorie

Datum	Versie	Beschrijving	Auteur
01-maart-2017	0.1	Aanmaak	Furkan Demirci
01-april-2017	0.5	Aanvulling	Furkan Demirci
01-meい-2017	0.9	Aanvulling	Furkan Demirci
01-juni-2017	1.0	Initiële versie	Furkan Demirci

## Accordering document

Namens Jeroen Franssen



Namens Furkan Demirci



## Inhoud

Bijlage F – Test plan .....	74
Inleiding .....	77
Doele van dit document .....	77
Referenties .....	77
Opdrachtformulering .....	77
Randvoorwaarden .....	77
Teststrategie .....	78
Samenvatting productrisico-analyse .....	78
Testsoorten .....	78
Verdeling van de testinspanning .....	79
Testaanpak .....	79
Testuitvoering .....	79
Testtools .....	79
Bijlage 1 .....	80
Bijlage 2 .....	82
Bijlage 3 .....	84

## Inleiding

### Doel van dit document

Dit Test Plan beschrijft de testaanpak die binnen het ontwikkelteam zal worden gehanteerd. Hier worden bugs, security leaks, gebruiksvriendelijkheid en functionaliteit getest.

### Referenties

Titel	Versie	Auteur	Vindplaats
Functioneel ontwerp	1.0	Furkan Demirci	<b>Bijlage D</b>
Technische ontwerp	1.0	Furkan Demirci	<b>Bijlage E</b>
Plan van aanpak	1.0	Furkan Demirci	<b>Bijlage C</b>

## Opdrachtformulering

De testinspanningen binnen het ontwikkelteam hebben de volgende doelen:

- Een bijdrage leveren aan de kwaliteit van de applicatie.
- Security lekkage verhinderen.
- Gebruiksvriendelijk mogelijk maken.

### Randvoorwaarden

Randvoorwaarden om dit test plan waar te maken zijn:

- Excel sheet met de verwachten resultaten.
- Implementeerde applicatie die voldoet aan de functionaliteit.
- Apparatuur zoals een Android device of een emulator.
- Source code van het project.

## Teststrategie

Hier worden de onderstaande kenmerken uitgevoerd en getest.

### Samenvatting productrisico-analyse

Kenmerk / - onderdeel	Risicoklasse	Argumentatie
Functionaliteit	-	-
WEB-API	HOOG	Communicatie middel en brein van de app.
Login systeem	HOOG	Een veilig login systeem zonder data lekkage.
Wardrobe functie	HOOG	Goed functionerend Wardrobe functie zonder error.
Shop functie	HOOG	Goed functionerend Shop functie zonder bugs.
Gebruiksvriendelijkheid	MIDDEN	Makkelijk bruikbaar app.
Performance	LAAG	Niet al te sloom en haperend.
Beveiliging	HOOG	Lekkage controleren.
Beheerbaarheid	MIDDEN	Makkelijk beheerbaar zonder bugs.
Betrouwbaarheid	LAAG	Betrouwbaar app.

### Testsoorten

De volgende testsoorten worden uitgevoerd.

Testsoort	Beschrijving	Doel
Bouwtest	Testsoort die door programmeurs wordt uitgevoerd. Hierbij wordt geschreven code getest of het in meerdere soorten situaties geen exceptions aangeeft.	Verifiëren dat de code bug free is en in meerdere soorten situaties nog steeds goed kan functioneren.
Monkey test	Testsoort die door gebruikers wordt uitgevoerd. De gebruiker voert willekeurige gegevens in en bekijkt hoe de applicatie hierop gaat reageren.	Verifiëren dat de applicatie zulke "monkey" situaties goed ontvangt zonder enige exceptions.
Security test	Testsoort die door gebruikers en programmeurs wordt uitgevoerd. Hier wordt er gekeken naar lekkages van data zodat het niet in de handen kan komen van hackers.	Een waterdichte WEB-API en applicatie waar geen data vanaf kan worden gehaald. Injectie vrij SQL query's en JSON lekkage.

## Verdeling van de testinspanning

Beschikbare testtijd dat verdeeld wordt over de onderdelen en testsoorten.

Kenmerk / - onderdeel	Risicoklasse	Bouwtest	Monkey test	Security test
Functionaliteit	-	●●●	●●	●●
WEB-API	HOOG	●●●	●●●	●●●
Login systeem	HOOG	●●●	●	●●●
Wardrobe functie	HOOG	●●●	●●	●●
Shop functie	HOOG	●●●	●	●
Gebruiksvriendelijkheid	MIDDEN	●●	●●	●●
Performance	LAAG	●	●	●●
Beveiliging	HOOG	●●●	●●●	●●●
Beheerbaarheid	MIDDEN	●●	●●	●●●
Betrouwbaarheid	LAAG	●	●	●

## Testaanpak

In het bijlage is een Excel bestand te zien waar de gewenste resultaten staan. Hier wordt de resultaat gedocumenteerd en vergeleken of het overeen komt met het gewenste resultaat. Als het overeen komt wordt dit goedgekeurd. Bij een foutkeuring wordt het onderwerp opnieuw aangepakt en opnieuw getest totdat het wordt goedgekeurd.

### Testuitvoering

#### 1. Bouwtest

Dit wordt door programmeurs getest door middel van geschreven code in verschillende soorten situaties te testen. Een voorbeeld hiervan is: Wat als het geen data binnen krijgt wat het eigenlijk wel verwacht. De reactie van de code wordt beschreven in het Excel document (zie bijlage).

#### 2. Monkey test

Hier wordt willekeurige gegevens ingevoerd door de gebruiker om te testen hoe de applicatie hierop gaat reageren. De reactie wordt ook beschreven in het Excel document (zie bijlage).

#### 3. Security test

Het voorkomen van security lekkages zoals email adressen en wachwoorden. Dit wordt door gebruikers als programmeurs uitgevoerd. Deze test wordt vooral op de WEB-API gericht omdat die ervoor zorgt dat er data's vanuit de database naar de applicatie kan worden verstuurd.

## Testtools

Testen van de applicatie is de source code nodig en een emulator of een Android device. Voor de WEB-API moeten er POST- en GET-requesten gestuurd kunnen worden. Dit wordt door middel van een website getest dat specifiek gemaakt is om de WEB-API te testen.

## Bijlage 1

nr.	Test onderdeel	Type	Verwachte resultaat
1	WEB-API login en create account	Bouwtest	JSON response met gegevens van het account.
2	WEB-API shop add/download	Bouwtest	JSON response met alle shop gegevens van een rubriek.
3	WEB-API head Upload	Bouwtest	Formaat foto verkleinen. JSON response voor de app.
4	WEB-API head Download	Bouwtest	Alle gegevens van de foto in een JSON response.
5	WEB-API top Upload	Bouwtest	Formaat foto verkleinen. JSON response voor de app.
6	WEB-API top Download	Bouwtest	Alle gegevens van de foto in een JSON response.
7	WEB-API bottom Upload	Bouwtest	Formaat foto verkleinen. JSON response voor de app.
8	WEB-API bottom Download	Bouwtest	Alle gegevens van de foto in een JSON response.
9	WEB-API feet Upload	Bouwtest	Formaat foto verkleinen. JSON response voor de app.
10	WEB-API feet Download	Bouwtest	Alle gegevens van de foto in een JSON response.
11	WEB-API kleidng set Upload	Bouwtest	Values in een POST method sturen en opslaan in database.
12	WEB-API kleidng set Download	Bouwtest	Gegevens van alle data's in een JSON response.
13	WEB-API friends systeem	Bouwtest	Alle data wordt goed ontvangen en in het database gestopt.
14	WEB-API lend systeem	Bouwtest	Alle data wordt goed ontvangen en in het database gestopt.
15	WEB-API sell systeem	Bouwtest	Alle data wordt goed ontvangen en in het database gestopt.
16	Applicatie ontvangt JSON	Bouwtest	JSON wordt goed uitgelezen en in variables gestopt.
17	Applicatie stuurt requesten	Bouwtest	POST en GET requesten sturen is succesvol.
18	Login systeem	Bouwtest	Login werkt stuurt POST request naar WEB-API succesvol.
19	Account creatie systeem	Bouwtest	Account aanmaken werkt stuurt POST requesten succesvol.
20	Shop page werkt	Bouwtest	laat de beschikbare winkels zien die in het database staan.
21	Wardrobe image download	Bouwtest	Images van de user word gedownload en laten zien.
22	Wardrobe image upload	Bouwtest	images die worden gemaakt of gekozen worden geüpload.
23	Wardrobe als set upload	Bouwtest	4 kledingen kunnen kiezen en opslaan als een set.
24	Wardrobe als set download	Bouwtest	De sets die waren opgeslagen kunnen downloaden als JSON.
25	Wardrobe items/sets deleten	Bouwtest	item id en set id als POST method sturen en deleteen.
26	Wardrobe leen systeem	Bouwtest	Gekoppelde vrienden kunnen elkaars kleding lenen.
27	Wardrobe sell systeem	Bouwtest	Een item voor sale zetten met een prijs en beschrijving.
28	Ontvangen van requesten	Bouwtest	Requesten van leningen en sell items binnen krijgen.
29	Friends systeem send	Bouwtest	Vrienden kunnen zoeken en toevoegen.
30	Friends systeem accept	Bouwtest	Vrienden kunnen accepteren of weigeren.

31	Login en account	Monkeytest	Applicatie moet het goed aanpakken zonder enige error.
32	Wardrobe upload	Monkeytest	Foto's uploaden met een size indien nodig.
33	Friends search	Monkeytest	Applicatie laat geen data zien als het ook niet bestaat.
34	Prijs involveld	Monkeytest	Gebruiker kan alleen cijfers invoeren.
35	Login JSON response	Securitytest	Response zou geen belangrijke informatie moeten leaken.
36	Foto's JSON response	Securitytest	Toegankelijk zolang de users hun eigen id weten.
37	SQL injecties	Securitytest	SQL injecties zijn mogelijk dus nog niet beveiligd.
38	Wachtwoord hashing	Securitytest	Wachtwoorden zijn nog niet gehashed.

Types	Status
Bouwtest	Goedgekeurd
Monkeytest	Afgekeurd
Securitytest	

## Bijlage 2

nr.	Gewenste resultaat
1	JSON response met alle gegevens van het account zoals naam, email, username, users_id enz.
2	JSON response met alle shop gegevens gevraagd om het rubriek
3	Uploaden van foto's worden verkleint qua size en formaat. JSON response met true of false.
4	JSON response met gegevens van de foto en het foto zelf als een link.
5	Uploaden van foto's worden verkleint qua size en formaat. JSON response met true of false.
6	JSON response met gegevens van de foto en het foto zelf als een link.
7	Uploaden van foto's worden verkleint qua size en formaat. JSON response met true of false.
8	JSON response met gegevens van de foto en het foto zelf als een link.
9	Uploaden van foto's worden verkleint qua size en formaat. JSON response met true of false.
10	JSON response met gegevens van de foto en het foto zelf als een link.
11	Verwacht een POST request met de geselecteerde gegevens in het applicatie. Word opgeslagen in het database
12	JSON response met de gegevens van elke set dat door de user is gemaakt.
13	De geselecteerde data word verstuurd naar het WEB-API en in het database opgeslagen.
14	De geselecteerde data word verstuurd naar het WEB-API en in het database opgeslagen.
15	De geselecteerde data word verstuurd naar het WEB-API en in het database opgeslagen.
16	JSON response word ontvangen door de applicatie en in een list met variables gestopt doormiddel van GET SET.
17	Applicatie kan Http Cliënt request sturen naar de WEB-API. Dit kan een POST of GET request zijn.
18	POST request sturen naar de WEB-API. Controleert of de gegevens overeenkomen. Stuurt JSON true of false terug.
19	POST request sturen naar de WEB-API. Maakt een nieuwe record in het database. Stuurt JSON true of false terug.
20	Laat de shops zien die bij de gebruiker hoort. OnClick event dat de user omleid naar de gekozen shop website.
21	Images worden gedownload via een link dat mee wordt gestuurd als een JSON response. Images worden gestreamd.
22	Een foto kiezen uit je galerij of een foto maken via de camera. Size ingevoerd indien nodig. Wordt geupload.
23	4 gekozen kledingen als een set kunnen opslaan. Gegevens worden gestuurd naar de WEB-API
24	Opgeslagen sets worden als een JSON response gestuurd en in de applicatie laten zien aan de gebruiker.
25	De id van de geselecteerde item wordt verstuurd naar de WEB-API en daar verwijdert uit het database en server.
26	Vrienden kunnen elkaars kledingen lenen. Geselecteerde kleding word verstuurt en in het database gestopt.
27	Geselecteerde item voor sale zetten met een prijs en beschrijving.
28	Requesten binnen krijgen van leningen en sell items voor de specifieke gebruiker.
29	Vrienden kunnen zoeken en toevoegen.
30	Vrienden kunnen accepteren of weigeren

31	Applicatie moet een displayalert geven als er verkeerde informatie ingevoerd is. Goed aanpakken zonder error.
32	Elk soort foto is mogelijk om te uploaden. Naast het uploaden moet er een size ingevoerd worden indien nodig.
33	Friends page moet geen data kunnen laten zien als het ook niet bestaat in het database.
34	Alleen cijfers kunnen worden ingevoerd. Er is geen limit.
35	Response zou geen belangrijke informatie moeten leaken.
36	Toegankelijk zolang de user hun eigen id weten.
37	De WEB-API moet beveiligd worden tegen SQL injecties.
38	Wachtwoorden moeten worden gehashed bij het aanmaken van een nieuwe account.

### Bijlage 3

nr.	Ontvangen resultaat	Status	Evaluaties
1	JSON response met gegevens van het account	Goedgekeurd	
2	JSON response met alle shop gegevens van een rubriek	Goedgekeurd	
3	Formaat foto op origineel grootte. JSON response voor de app	Goedgekeurd	Foto moet worden verkleint ( <b>BIJGEWERKT</b> )
4	Alle gegevens van de foto in een JSON response.	Goedgekeurd	
5	Formaat foto op origineel grootte. JSON response voor de app	Goedgekeurd	Foto moet worden verkleint ( <b>BIJGEWERKT</b> )
6	Alle gegevens van de foto in een JSON response.	Goedgekeurd	
7	Formaat foto op origineel grootte. JSON response voor de app	Goedgekeurd	Foto moet worden verkleint ( <b>BIJGEWERKT</b> )
8	Alle gegevens van de foto in een JSON response.	Goedgekeurd	
9	Formaat foto op origineel grootte. JSON response voor de app	Goedgekeurd	Foto moet worden verkleint ( <b>BIJGEWERKT</b> )
10	Alle gegevens van de foto in een JSON response.	Goedgekeurd	
11	Values in een POST method sturen en opslaan in database.	Goedgekeurd	
12	Gegevens van alle data 's in een JSON response	Goedgekeurd	
13	Alle data wordt goed ontvangen en in het database gestopt	Goedgekeurd	
14	Alle data wordt goed ontvangen en in het database gestopt	Goedgekeurd	
15	Alle data wordt goed ontvangen en in het database gestopt	Goedgekeurd	
16	JSON wordt goed uitgelezen en in variables gestopt	Goedgekeurd	
17	POST en GET requesten sturen is succesvol	Goedgekeurd	
18	Login werkt stuurt POST request naar WEB-API succesvol	Goedgekeurd	
19	Account aanmaken werkt stuurt POST requesten succesvol	Goedgekeurd	
20	laat de beschikbare winkels zien die in het database staan	Goedgekeurd	
21	Images van de user word gedownload en laten zien	Goedgekeurd	
22	images die worden gemaakt of gekozen worden geüpload	Goedgekeurd	
23	4 kledingen kunnen kiezen en opslaan als een set	Goedgekeurd	
24	De sets die waren opgeslagen kunnen downloaden als JSON	Goedgekeurd	
25	item id en set id als POST method sturen en deleten.	Goedgekeurd	
26	Gekoppelde vrienden kunnen elkaars kleding lenen	Goedgekeurd	
27	Een item voor sale zetten met een prijs en beschrijving.	Goedgekeurd	
28	Requesten van leningen en sell items binnen krijgen	Goedgekeurd	
29	Vrienden kunnen zoeken en toevoegen	Goedgekeurd	
30	Vrienden kunnen accepteren of weigeren	Goedgekeurd	

31	Applicatie moet het goed aanpakken zonder enige error	Goedgekeurd	
32	Foto's uploaden met een size indien nodig	Goedgekeurd	
33	Applicatie laat geen data zien als het ook niet bestaat	Goedgekeurd	
34	Gebruiker kan alleen cijfers invoeren	Goedgekeurd	
35	Response laat geen belangrijke informatie zien	Goedgekeurd	
36	Toegankelijk zolang de users hun eigen id weten.	Goedgekeurd	
37	Niet veilig tegen SQL injecties.	Goedgekeurd	SQL injecties vrij maken. ( <b>BIJGEWERKT</b> )
38	Wachtwoorden zijn nog niet gehashed	Goedgekeurd	Wachtwoord hashen tijdens account creëren. ( <b>BIJGEWERKT</b> )



## Bijlage H – Implementatieplan

### Implementatieplan

Good-Lookz – App

Opdrachtgever: 4people Communications

Opdracht: Implementeren van de Good-Lookz applicatie

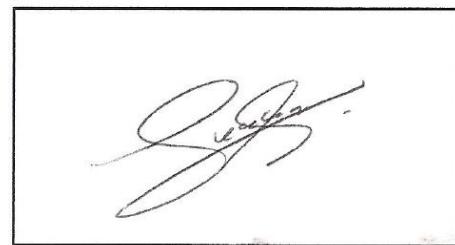
Datum: 20-05-2017



Jeroen Franssen



Furkan Demirci



Datum	Versie	Auteur	Aanpassingen
10-03-2017	0.1	Furkan Demirci	Implementatie document opstellen
20-04-2017	0.5	Furkan Demirci	Implementatie document invullen
31-05-2017	1.0	Furkan Demirci	Implementatie document afronden

## Inhoudsopgaven

Implementatie plan .....	86
Aanleiding:.....	88
Inleiding:.....	88
Doelgroep:.....	88
Doel van De Applicatie .....	89
Oud systeem.....	89
Nieuw systeem .....	89
Scope en afbakening .....	89
Gevolgen voor trainingen, opleidingen.....	90
Gevolgen voor ICT/Facilitair .....	90
Risico's en maatregelen .....	90
Aanpak van de implementatie .....	91
Acceptatie test .....	92
Planning van de implementatie .....	93

## **Aanleiding:**

Dit implementatie plan is na aanleiding van het nieuwe app Good-Lookz. Voor het live brengen van de applicatie en de beheerders/ontwikkelaars ondersteuning te leveren met het nieuwe systeem.

## **Inleiding:**

Voor de applicatie Good-Lookz moeten de eindgebruikers goed begeleid worden voor het nieuwe systeem. Veel hoeft er niet veranderd te worden aangezien dit maar een klein bedrijf is. Dus hoogstens moeten twee medewerkers worden ingelicht en begeleid worden met het nieuwe systeem.

## **Doelgroep:**

Dit plan is geschreven voor de beheerders en de ontwikkelaars die dit project verder oppakken en gebruiken.

## Doel van De Applicatie

Het is de bedoeling om een app te maken voor Android en iOS. In de app is het mogelijk om een account te maken en om in te loggen. In het Wardrobe page is het mogelijk om kleding toe te voegen en om kleding te verwijderen. Ook moet de gebruiker de geselecteerde kleding voor sale kunnen zetten. Gekoppelde vrienden kunnen elkaar kleding dan zien. Ook moeten gekoppelde vrienden elkaar kleding voor lening vragen. WEB-API is de brein van de applicatie en dit hoort de taken goed uit te voeren voor een functioneel applicatie. Meer hierover in **Bijlage A**

## Oud systeem

Er was geen oude systeem voor dit applicatie. Dit was een compleet nieuw project dat nog gemaakt moest worden.

## Nieuw systeem

De nieuwe systeem bestaat uit een inlog functie waar het ook mogelijk is om een account aan te maken. Na het inloggen wordt het menu weergegeven waar de gebruikers kunnen navigeren naar de wardrobe page, shop page, friends page en settings page. In de wardrobe page is het mogelijk om kleding te uploaden, bekijken en verwijderen. Gekoppelde vrienden kunnen elkaar kleding bekijken en kledingstukken lenen. Gebruikers kunnen ook kleding voor sale zetten met een prijs. Gekoppelde vrienden kunnen elkaar items dat voor sale staat ook bekijken en interesse tonen. In de shop page worden per rubriek de gekozen shops laten zien van de gebruiker.

## Scope en afbakening

Het implementatieplan geldt voor de applicatie en WEB-API.

Onder de applicatie valt:

- Login system
- Friends system
- Wardrobe system
- Lend system
- Sell system
- Shop system

Onder de WEB-API valt:

- Account
- Friends
- Lend
- Notifications
- Sale
- Shops
- Wardrobe
- Wardrobe sets

Wat er buiten valt:

- Mirror system

## Gevolgen voor trainingen, opleidingen

De doelgroep voor dit implementatieplan is niet heel groot. Het is bedoeld voor twee werknemers en de bedrijfsmanager.

De training/instructies worden gegeven door de ontwikkelaar aan de betreffende doelgroep. Ook staat er in de code zelf documentatie over wat een functie doet en of moet doen.

## Gevolgen voor ICT/Facilitair

Tijdens de implementatie moeten de nieuwe gebruikers een account aanmaken zodat ze kunnen inloggen en gebruik kunnen maken van de Applicatie. Ook krijgt men uitleg over wat er allemaal gedaan kan worden in de WEB-API/applicatie met een account. Om het te testen is er een Android device of een emulator op een pc nodig.

## Risico's en maatregelen

Hier de risico's en/of showstoppers betreft het implementeren.

Nr.	Beschrijving risico	Beschrijving Impact en kans	Te nemen maatregel	Eigenaar
1	Software gaat niet samen met de host pakket.	Niet kunnen live brengen.	Andere hosting server gebruiken waar het wel op werkt.	Jeroen Franssen
2	Tijdtekort.	Project kan niet geïmplementeerd worden.	Onderdelen laten vallen.	Furkan Demirci
3	Java memory error exception.	Applicatie crashed.	Niet veel resources op een pagina laden. Memory leak voorkomen.	Furkan Demirci
4	Oude android versies niet goed ondersteunt.	Applicatie wordt niet goed uitgevoerd.	Niet compatible maken op oudere apparaten.	Furkan Demirci
5				
6				
7				
8				
9				

## Aanpak van de implementatie

Als eerst werd er een host aangevraagd bij de opdrachtgever (Jeroen Franssen). Vervolgens koppelt de opdrachtgever de FTP gegevens terug die gebruikt kunnen worden om een verbinding te krijgen met de server. Dit wordt gedaan door middel van een software programma genaamd FileZilla.

Voor de applicatie kan er een emulator of een Android device gebruikt worden. Dit wordt gedaan door middel van Visual studio. Hier moet de programma eerst een verbinding hebben met de emulator of Android device.

Nu er verbinding mogelijk is kan het project live gebracht worden. Nadat het project live en werkend is, kunnen de gebruikers samen met de ontwikkelaars de acceptatietest uitvoeren. Nu kunnen de gebruikers kijken of de applicatie/website voldoet aan de opgestelde eisen van de acceptatietest.

Na aanleiding van de resultaten van de acceptatietest, zal er besproken worden of de applicatie/website acceptabel is voor verder gebruik. Dit gaat samen in overleg met de opdrachtgever en de ontwikkelaars. Als de acceptatie test wordt afgekeurd zullen de afgekeurde punten opnieuw bijgewerkt en/of aangepast worden. Vervolgens zal het aangepaste project opnieuw live gebracht worden en er zal weer een acceptatie test plaatsvinden. Mocht de acceptatie goedgekeurd zijn dan zullen de eindgebruikers en leiding bijgeschoold worden over het nieuwe systeem. Dit wordt gedaan door een ontwikkelaar en/of ontwikkelaars. Dit is afhankelijk van de hoeveelheid eindgebruikers. Dit wordt gedaan door mond tot mond communicatie betreft de uitleg.

Na dat alle eindgebruikers zijn bijgeschoold zal er een evaluatie plaats vinden tussen de ontwikkelaars en de opdrachtgever.

## Acceptatie test

Hieronder de acceptatietest die getest zijn door de gebruikers.

nr	Test onderdeel	Status	Evaluatie
1	Login systeem	Goedgekeurd	
2	Account create systeem	Goedgekeurd	
3	Wardrobe pagina open	Goedgekeurd	
4	Toevoegen van kledingen	Goedgekeurd	
5	Laden van pictures's	Goedgekeurd	
6	Pictures zijn interactief	Goedgekeurd	
7	Deleteren van picture's	Goedgekeurd	
8	Swipe functie	Goedgekeurd	
9	Opslaan als een set	Goedgekeurd	
10	Opgeslagen sets bekijken	Goedgekeurd	
11	Kleding voor sale zetten.	Goedgekeurd	
12	Accounts opzoeken.	Goedgekeurd	
13	Friend request sturen.	Goedgekeurd	
14	Friends list	Goedgekeurd	
15	Delete friends.	Goedgekeurd	
16	Kleding lijst van vrienden bekijken.	Goedgekeurd	
17	Kleding leen request sturen.	Goedgekeurd	
18	Leen request accepteren.	Goedgekeurd	
19	Leen request deleten.	Goedgekeurd	
20	Uitgeleende kledingen lijst.	Goedgekeurd	
21	Shop pagina open	Goedgekeurd	
22	Gekozen shops worden weergegeven	Goedgekeurd	
23	Shops zijn interactief (opent webshop)	Goedgekeurd	
24	For sale pagina open	Goedgekeurd	
25	Kledingen die voor sale staan worden weergegeven	Goedgekeurd	
26	Sale items zijn interactief	Goedgekeurd	
27	Bieden op de sale items	Goedgekeurd	
28	request sturen dat je geïnteresseert bent	Goedgekeurd	
29	Requesten ontvangen.	Goedgekeurd	
30	Uitloggen mogelijk	Goedgekeurd	

## Planning van de implementatie

Hier is de planning van de implementatie te zien met de categorieën, geschatte tijden en werkelijke tijden.

### Good-Lookz

Project	Categorie	Toegewezen aan	Geschat Starten	Geschat Voltooien	Geschatte duur (in dagen)	Werkelijk Starten	Werkelijk Voltooien	Werkelijke duur (in dagen)	Notities
Good-Lookz applicatie	Host gegevens	Jeroen Franssen	15-3-2017	16-3-2017	1	15-3-2017	16-3-2017	1	Ontvangen van host gegevens.
Good-Lookz applicatie	FTP gegevens	Furkan Demirci	15-3-2017	16-3-2017	1	15-3-2017	16-3-2017	1	Naam en wachtwoord voor de server
Good-Lookz applicatie	Live zetten	Furkan Demirci	18-3-2017	20-5-2017	62	16-3-2017	22-5-2017	66	Uploaden op de server
Good-Lookz applicatie	Acceptatietest	Rick de Vries	22-5-2017	23-5-2017	1	23-5-2017	24-5-2017	1	Testen van het geïmplementeerde project
Good-Lookz applicatie	Acceptatietest	Davey Cornelissen	22-5-2017	23-5-2017	1	23-5-2017	24-5-2017	1	Testen van het geïmplementeerde project
Good-Lookz applicatie	Resultaten	Furkan Demirci	26-5-2017	27-5-2017	1	24-5-2017	25-5-2017	1	Resultaat van het acceptatietest
Good-Lookz applicatie	Bijscholen	Jeroen Franssen	31-5-2017	1-6-2017	1	26-5-2017	27-5-2017	1	Uitleg geven over het geïmplementeerde project
Good-Lookz applicatie	Bijscholen	Davey Cornelissen	31-5-2017	1-6-2017	1	26-5-2017	27-5-2017	1	Uitleg geven over het geïmplementeerde project
Good-Lookz applicatie	Bijscholen	Rick de Vries	31-5-2017	1-6-2017	1	26-5-2017	27-5-2017	1	Uitleg geven over het geïmplementeerde project
Good-Lookz applicatie	Evaluatie	Furkan Demirci	1-6-2017	2-6-2017	1	28-5-2017	29-5-2017	1	Evaluieren over de gehele implementatie
Good-Lookz applicatie	Evaluatie	Jeroen Franssen	1-6-2017	2-6-2017	1	28-5-2017	29-5-2017	1	Evaluieren over de gehele implementatie

## Evalueert een implementatie

Na het implementeren en bijscholen is er besproken over de resultaten van de acceptatietest. Alle onderdelen waren goedgekeurd. Hier wordt ook over het bijscholen gesproken samen met de betrokkenen.

## Gesprek met praktijkbegeleider en betrokkenen

Hier hebben wij een gesprek gehad over de resultaten. Met de praktijkbegeleider en betrokkenen zijn er over de volgende punten gesproken.

- Hoe ging de implementatie
- Wat kon er beter
- Wat zou je de volgende keer anders doen
- Hoe ging het bijscholen

### Hoe ging de implementatie

#### Jeroen (praktijkbegeleider):

Implementatie is goed gegaan, door gezamenlijk de stappen te doorlopen is het vlekkeloos neergezet.

#### Rick:

Goed gestructureerd te werk gaan en dus niet aan allerlei dingen begonnen zonder het functie waar je mee bezig was af te maken.

### Wat kon er beter

#### Jeroen (praktijkbegeleider):

De vormgeving van bepaalde knoppen zouden mooier zijn als dat gifjes waren. Leen systeem kon beter aangepakt worden.

#### Rick:

Communicatie kan altijd beter.

Doordat jij goed gestructureerd te werk bent gegaan kon het niet beter gedaan worden.

### Wat zou je de volgende keer anders doen

#### Jeroen (praktijkbegeleider):

Communicatie lijnen kort houden om sneller het goede resultaat te creëren. Zodat we op dezelfde golf lengte zitten.

#### Rick:

Sneller contact leggen met opdrachtgever wanneer er een functie af is.

### Hoe ging het bijscholen

#### Jeroen (praktijkbegeleider):

Goed helder gegaan. Hieruit zijn nieuwe ideeën ontstaan voor de nieuwe volgende versie.

#### Rick:

Ging goed, was duidelijk wat de product betreft. Makkelijk zelf te gebruiken. WEB-API handleiding was goed in elkaar gezet om makkelijk te gebruiken.