

Xamarin workshop

Radius College

Rio4-APO2 2016/2017

2017

Inhoud

[Xamarin Forms 4](#_Toc485800808)

[Opstarten Xamarin project 4](#_Toc485800809)

[Waarom Xamarin(Native)? 4](#_Toc485800810)

[Waarom Xamarin.Forms? 4](#_Toc485800811)

[Hoe maak ik een project aan? 5](#_Toc485800812)

[Hoe selecteer ik het juiste Xamarin Project? 6](#_Toc485800813)

[Het kiezen van de juiste template 7](#_Toc485800814)

[Toevoegen van components 8](#_Toc485800815)

[Emuleren 11](#_Toc485800816)

[Android virtual emulator 11](#_Toc485800817)

[Mogelijkheid 1: 11](#_Toc485800818)

[Mogelijkheid 2: 12](#_Toc485800819)

[Virtual emulator opties: 13](#_Toc485800820)

[Debugging 15](#_Toc485800821)

[Errors 15](#_Toc485800822)

[Foute versie 15](#_Toc485800823)

[Layouts 16](#_Toc485800824)

[List lay-out 16](#_Toc485800825)

[Toevoegen van items aan een lay-out 16](#_Toc485800826)

[Scroll view 16](#_Toc485800827)

[Grid lay-out 17](#_Toc485800828)

[Rijen en kolommen toevoegen 17](#_Toc485800829)

[Weergeven in een grid 18](#_Toc485800830)

[Doorsturen naar een nieuw form 19](#_Toc485800831)

[Toevoegen van afbeelding aan project 20](#_Toc485800832)

[Resources & Styles 21](#_Toc485800833)

[Styles 21](#_Toc485800834)

[Global styles: 21](#_Toc485800835)

[Style Inheritance 22](#_Toc485800836)

[Dynamic styles 23](#_Toc485800837)

[Resources dictionary 24](#_Toc485800838)

[Resources 25](#_Toc485800839)

[Static resource 25](#_Toc485800840)

[Dynamic resource 26](#_Toc485800841)

[Waarom static- of dynamic resource 27](#_Toc485800842)

[Databinding 29](#_Toc485800843)

[Waarom databinding? 29](#_Toc485800844)

[Implementeren van data-binding 30](#_Toc485800845)

[Databinding MVVM : 32](#_Toc485800846)

[One-way binding 33](#_Toc485800847)

[Listviews & lazy loading 34](#_Toc485800848)

[Lazy loading 34](#_Toc485800849)

[Listview 37](#_Toc485800850)

[Deployment 39](#_Toc485800851)

[Deployment naar android 39](#_Toc485800852)

[Wordt een uitgever 39](#_Toc485800853)

[Google API project aanmaken 40](#_Toc485800854)

[Registreer Google API Access 42](#_Toc485800855)

[Publiceren 43](#_Toc485800856)

[Troubleshooting 44](#_Toc485800857)

[FAQ 45](#_Toc485800858)

# Xamarin Forms

## Opstarten Xamarin project

Voordat we beginnen met het aanmaken van een project moeten we natuurlijk weten of we Xamarin of Xamarin.Forms gaan gebruiken.

### Waarom Xamarin(Native)?

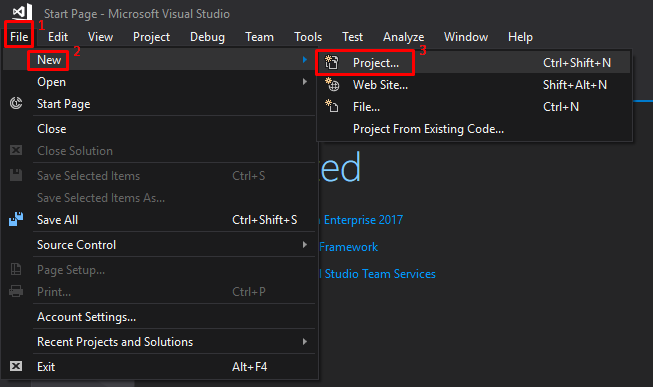
Xamarin ook wel Xamarin Native wordt gebruik op het moment dat je een platform specifieke applicatie wilt gaan maken. Wat is platform specifiek? Platform specifiek betekent dat je een applicatie schrijft die alleen op bijv. Windows, Android of iOS werkt. Het omzetten van Android en iOS zal dan ook niet zo makkelijk zijn aangezien je gebruik maakt van platform specifieke onderdelen.

### Waarom Xamarin.Forms?

Xamarin.Forms is het Framework dat je gebruikt op het moment dat je een Cross-Platform applicatie wilt gaan maken voor alle mogelijke besturingssystemen. Dit Framework zorgt ervoor dat de code die je schrijft in een Portable Class Library of een Shared Project wordt omgezet bij het builden naar code specifiek voor een besturingssystemen. Alle code die je schrijft zal voor een groot deel werken op alle systemen ondersteund door dit Framework. Op dit moment beschikt Xamarin over de volgende besturingssystemen: iOS, Android en Universal Windows Platform(UWP)

## Hoe maak ik een project aan?

Wij gaan nu de stappen doorlopen om een project aan te maken op het door u verkozen framework van Xamarin(.Forms).

Om een project aan te maken ga naar: **File > New > Project** (U kunt ook Ctrl+Shift+N indrukken)

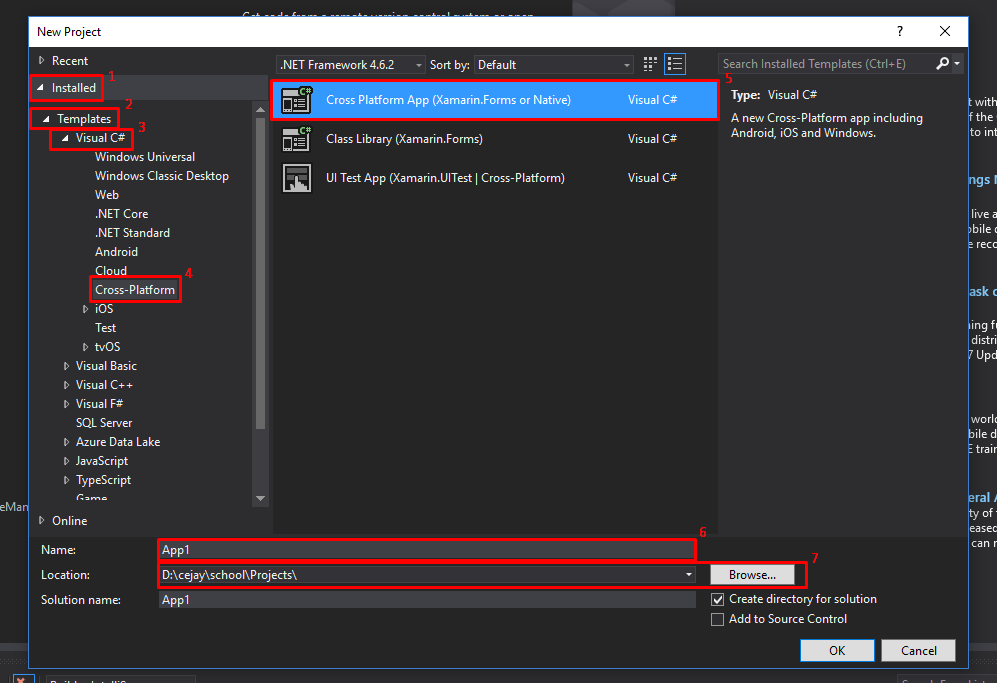
### Hoe selecteer ik het juiste Xamarin Project?

Op het moment dat u een nieuw project aanmaakt krijgt u een scherm te zien waarbij u verschillende templates te zien krijgt waaruit u kunt kiezen(Zie afbeelding). Om Xamarin te gebruiken moet u gebruik maken van de Cross Platform App template van Xamarin.

Ga naar: **Installed > Templates > Visual C# (Programmeer taal) > Cross-Platform >**

**Cross Platform App (Xamarin.Forms or Native)**

Zodra u de template heeft geselecteerd kunt u onderin een naam geven aan de Solution en App. Hierbij kunt u ook een specifieke locatie kiezen voor het project.

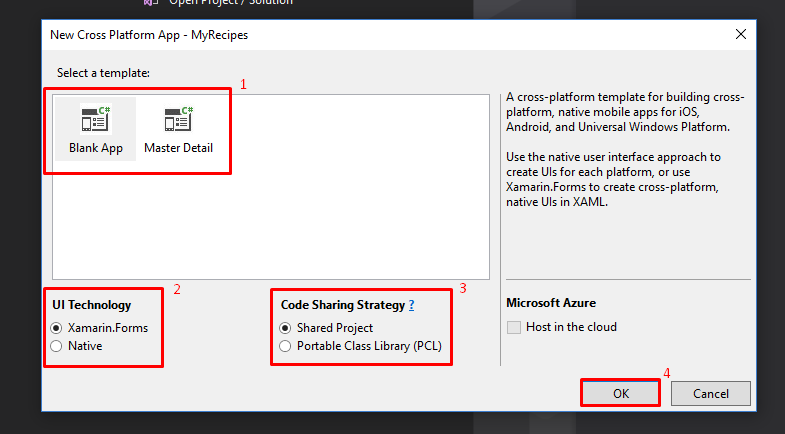


### Het kiezen van de juiste template

Zodra u het project hebt aangemaakt komt er een venster(Zie afbeelding) naar voren waarin u kunt kiezen tussen Blank App en MasterDetail. Wij kiezen standaard voor Blank App om zo te starten met een volledig leeg project. Het verschil tussen Blank App en MasterDetail is het volgende:  
Bij een blank app begin je vanaf scratch met het opbouwen van je applicatie, terwijl je bij MasterDetail al gebruik kunt maken van een bestaand navigatiepatroon.

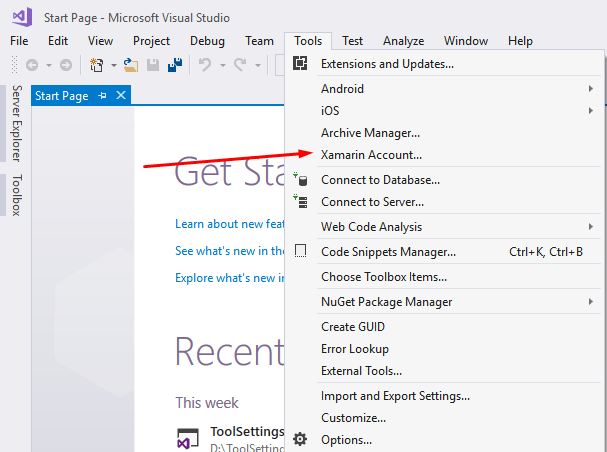
Onder UI Technology kunt u kiezen tussen Xamarin.Forms of Native wij kiezen voor Xamarin.Forms omdat wij een Cross-Platform app willen creeëren.

Onder Code Sharing Strategy kunt u kiezen tussen Shared Project of Portable Class Library(PCL)[[1]](#footnote-1)

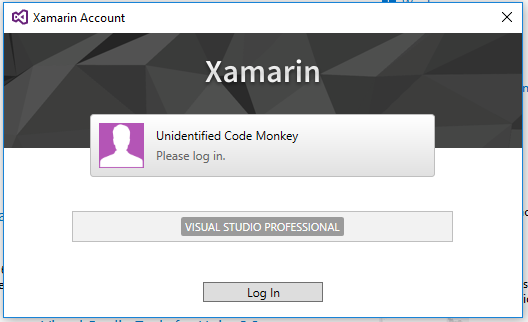


## Toevoegen van components

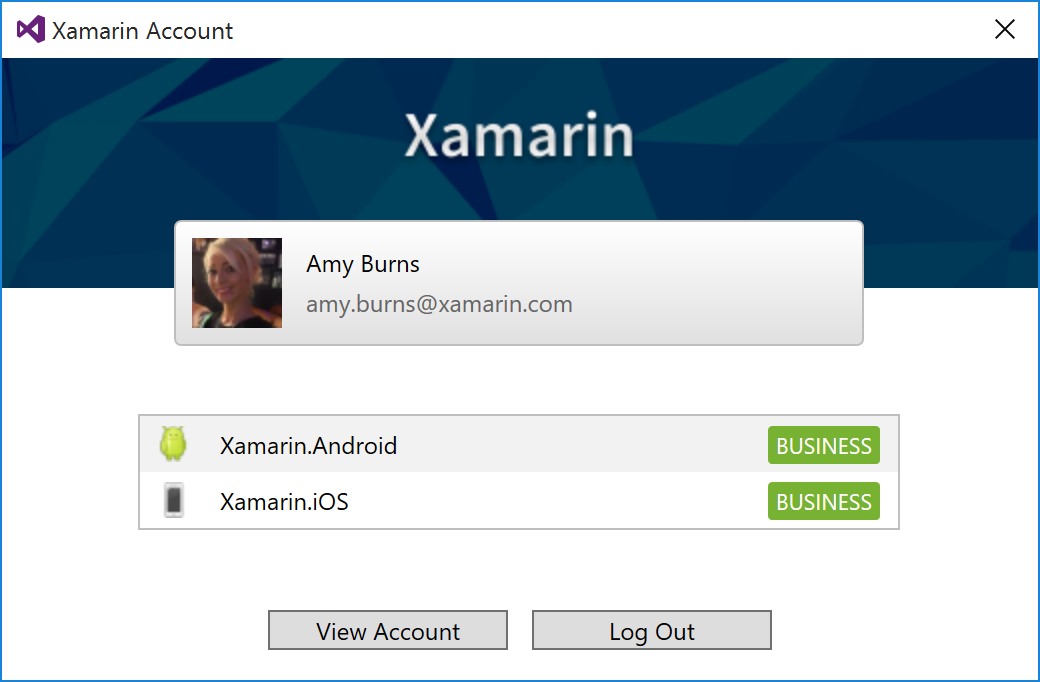
Om componenten aan je project toe te voegen, zodat je meer functionaliteiten kan implementeren in je applicatie, onderneem je de volgende stappen:



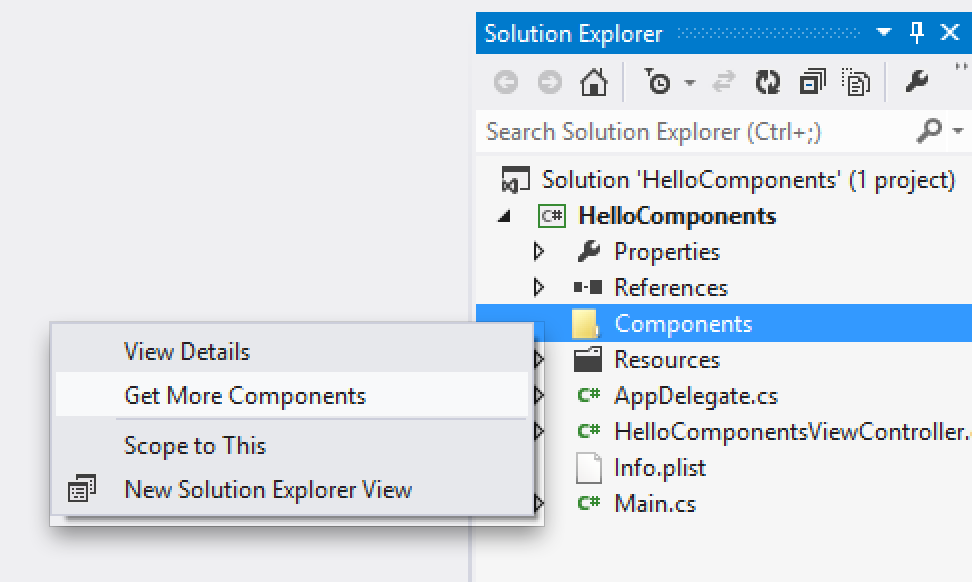
Start visual studio op en ga via het menu naar ‘Xamarin Account’.  
Het volgende scherm komt tevoorschijn.

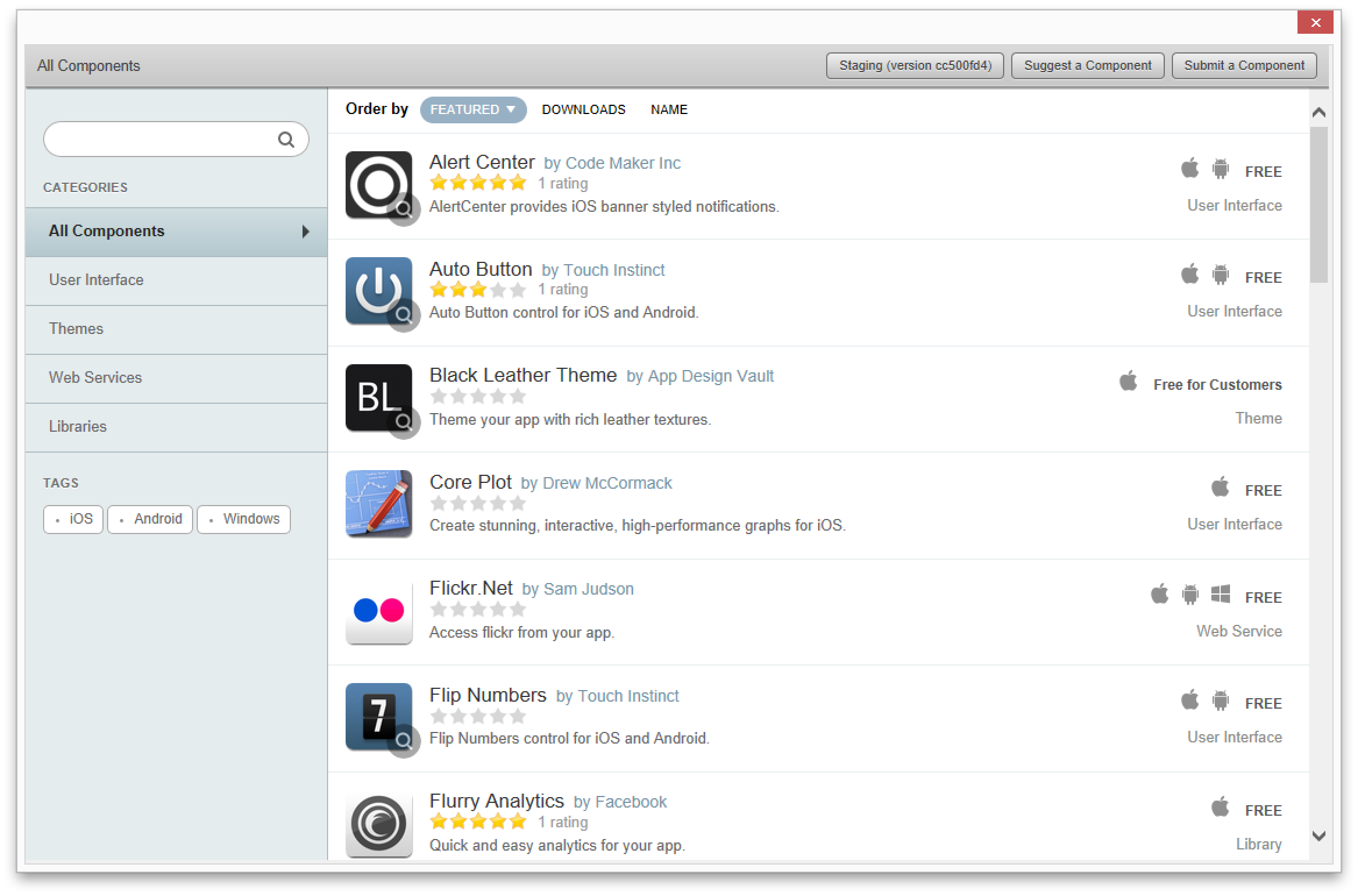
[](https://developer.xamarin.com/guides/cross-platform/xamarin-studio/components_walkthrough/Images/Image21.png)

Nadat u in het scherm op log in hebt geklikt, kunt u uw gegevens invoeren. Nadat u bent ingelogd, wordt uw abonnement geactiveerd en wordt u terug doorverwezen naar uw accountpagina.

[](https://developer.xamarin.com/guides/cross-platform/xamarin-studio/components_walkthrough/Images/Image25.png)

Nadat u bent ingelogd, zullen we een nieuw IOS-project aanmaken. Nadat u uw project heeft aangemaakt. Klik met de rechtermuisknop op components en klik daarna op get more components.

[](https://developer.xamarin.com/guides/cross-platform/xamarin-studio/components_walkthrough/)

Hierna wordt de onderdelenlijst geopend. [](https://developer.xamarin.com/guides/cross-platform/xamarin-studio/components_walkthrough/)

Vanuit dit venster kunnen we:

* Zoeken naar componenten op naam
* Filteren van componenten per categorie of tag
* Sorteren van componenten op downloads/naam
* Voorstellen maken voor onderdelen
* Versturen van componenten
* Componenten kiezen om in je project te gebruiken

# Emuleren

Na het maken van je eerste Android of iOS applicatie is het tijd om hem te gaan testen. Om de applicatie te gaan testen heb je natuurlijk een ruimte nodig waarin hij kan worden gedebugd.

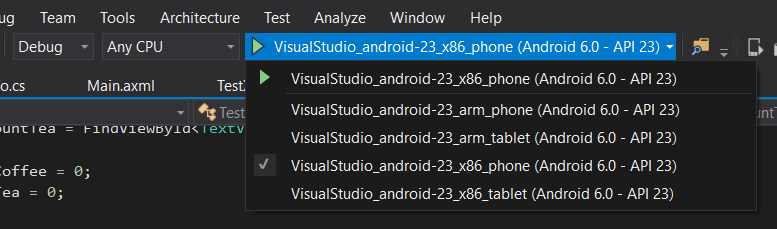
*Wat is emulator?*

Een emulatie is stukje software die hardware nabootst. Zo kan je dus een telefoon nabootsen via je computer! Zo kan je applicatie veilig worden getest, zonder dat je echte telefoon kapot kan gaan.

## Android virtual emulator

Er zijn een aantal mogelijkheden om je emulator voor Android op te starten.

### Mogelijkheid 1:

Voor dat je gaat debuggen kan je een emulator selecteren, deze gaat Visual Studio eerst opstarten voor de applicatie wordt gedebugd. Als je een applicatie voor een Android tablet hebt gemaakt wordt het aangeraden de VisualStudio\_android-23\_x86\_tablet te gebruiken. Als je een applicatie voor een Android telefoon hebt gemaakt wordt het aangeraden de VisualStudio\_android-23\_x86\_phone te gebruiken. Deze emulators zijn het snelst.  
  


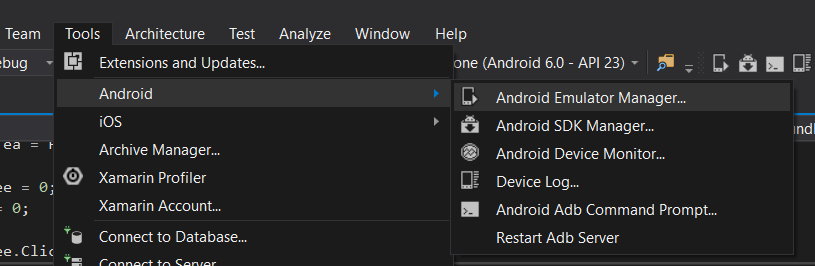
Als je dan op f5 of start drukt, wordt de emulator opgestart gevolgd met het debuggen van de applicatie.

LET OP: Sluit de emulator niet af! Zo wordt de opstarttijd van het debuggen veel langer omdat de emulator opnieuw wordt opgestart. Stop gewoon het debuggen van de applicatie, de emulator blijft openstaan.

### Mogelijkheid 2:

De Android Emulator Manager

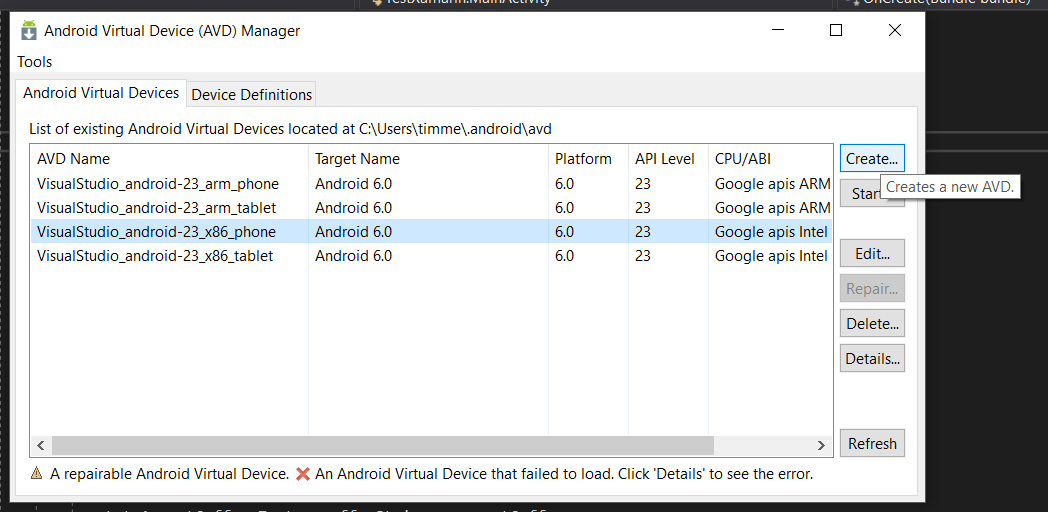
Om de Android Emulator Manager te openen, druk op tools -> Android -> Android Emulator manager



De Android Emulator Manager wordt nu geopend.

Om vervolgens de gewilde emulator te openen, selecteer de desbetreffende emulator en klik op Start.  
Net als bij mogelijkheid 1 wordt het aangeraden de x86 variant te gebruiken omdat die aanzienlijk sneller is.

De emulator wordt nu opgestart



## Virtual emulator opties:

Als het opstarten is gelukt, krijg je dit scherm te zien. Het opstarten is gelukt!



Rechts van het scherm staat er een balk met verschillende opties.

Emulator afsluiten:  
Sluit de emulator af.

Volume omhoog:  
Haalt het volume omhoog.

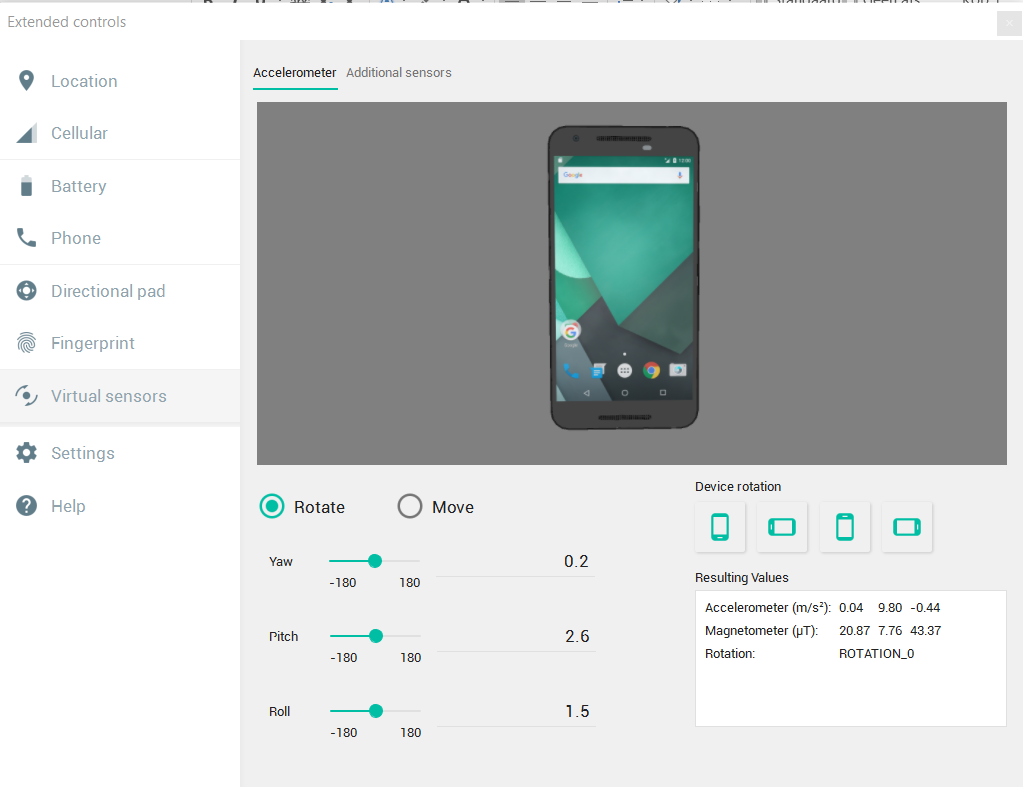
Volume omlaag:  
Haalt het volume omlaag

Roteren naar links:  
Roteert het scherm naar links.

Roteren naar rechts:  
Roteert het scherm naar rechts

Screenshot:  
Maakt een screenshot van de emulator

Inzoomen:  
Veranderd de muis in een zoom knop, doormiddel van te klikken zoom je in, rechtermuis knop is uitzoomen.

Terug knop:  
Gaat een scherm terug.

Home:  
Gaat naar het home menu.

Overview:  
Laat de actieve applicaties zien.

Geavanceerde opties:

Location:  
Hier kan je de locatie van het apparaat instellen

Cellular:  
Hier kan je de instellingen van je 4/3g instellen

Battery:  
Hier kan je de instellingen van je batterij instellen zoals  
charging en het aantal procent

Phone:  
Hier kan je een sms naar je telefoon sturen en bellen met een nummer naar keuzen.

Directional pad:  
Hier kan je het directional pad van je telefoon gebruiken (als de telefoon er een heeft natuurlijk)

Fingerprint:  
Hier kan je de fingerprint scanner activeren van de telefoon (als de telefoon er een heeft natuurlijk). Zo kan je de telefoon laten denken dat een vinger op de knop wordt gehouden.

Virtual sensors:  
Hier kan je in een 3d omgeving van de ligging van de telefoon beïnvloeden, zo kan je bijv. een event triggeren met schudden of als de telefoon verticaal ligt enz.

Settings:  
Hier kan je de settings van je emulator aanpassen

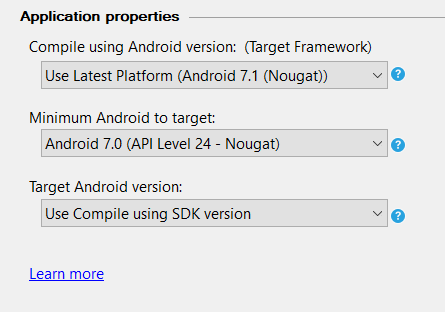
Help:  
Hierin staan alle shortcuts, beschrijving van de emulator en de generale help pagina.

## Debugging

### Errors

Het kan soms zijn dat er errors voorkomen. Hieronder staat een lijst van fouten die ik zelf ben tegen gekomen, mocht jou fout er niet tussen staan probeer te zoeken op het Xamarin, MSDN of stackoverflow forum. Probeer te zoeken naar Xamarin Android [error]. Mocht je geen oplossing vinden, maak een nieuw project aan en probeer je code te copy pasta-en. Mocht dit nog steeds niet werken download Xamarin opnieuw, maak een nieuw project aan en typ de code over. Ik heb dit zelf helaas ook een paar keer moeten doen.

### Foute versie

Het kan zo zijn dat de emulator een andere versie gebruikt dan de versie waarop jij debugt, dat kan fout gaan. Ga naar de properties van je project, ga naar ‘Application’ en selecteer de versie van je android emulatie.

# Layouts

## List lay-out

ListView is een belangrijk UI component van Android applicaties, die overal gebruikt wordt voor 'short lists' vanuit de menu opties tot aan 'long lists' van bijvoorbeeld contacten of internetbrowser favorieten.

ListView biedt een simpele manier om een 'scroll list' van rijen te presenteren die geformatteerd met een ingebouwde stijl kunnen worden.

Ook heeft ListView een Adapter nodig om data in te kunnen laden die in rijen worden weergeven.

### Toevoegen van items aan een lay-out

De makkelijkste manier om items toe te voegen, doe je door middel van een array met strings te vullen:

var listView = new ListView();

listView.ItemsSource = new string[]{

"mono",

"monodroid",

"monotouch",

"monorail",

"monodevelop",

"monotone",

"monopoly",

"monomodal",

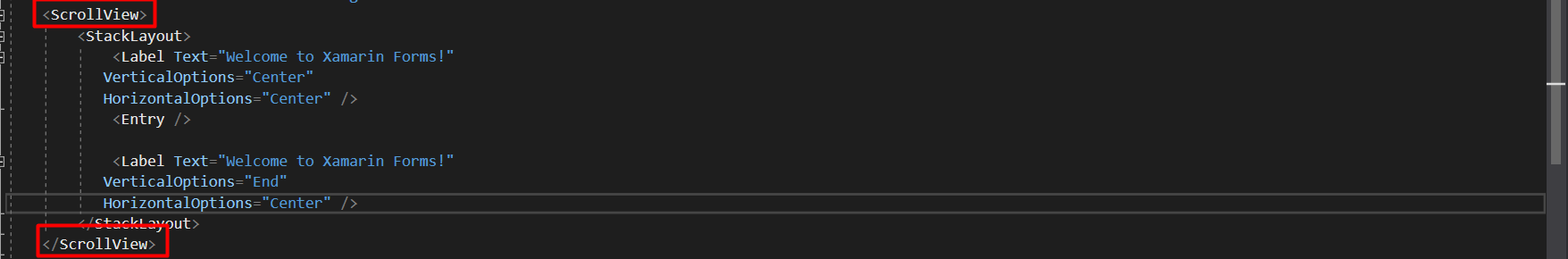
"mononucleosis"

};

## Scroll view

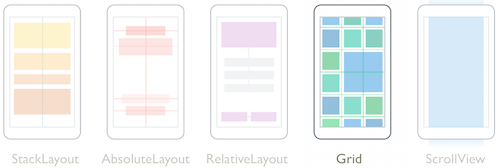
De ScrollView is een geweldige layout-optie één die je bijna op elke pagina wil gebruiken. Het zorgt ervoor dat in principe je pagina zo lang mogelijk kan maken.

Om een ScrollView te implementeren heb je maar 2 simpele regels code nodig!



Simple! Het resultaat zorgt ervoor dat je naar onder en boven kan swipen!

## Grid lay-out

Grid lay-out geeft iemand de mogelijkheid om de pagina weer te geven met rijen en kolommen. Rijen en kolommen kunnen zelf worden ingedeeld met hun eigen vaste grote. De grid lay-out zou niet de verwarrend moeten zijn, gezien de traditionele tabellen. Verder is het geen concept van rijen, kolommen of cel formateren. Anders dan HTML tabellen, is Grid uitsluitend bedoeld om inhoud uit te leggen.

Grid kan worden gebruikt voor het weergeven van je pagina in een raster. Dit kan handig zijn in de volgende omstandigheden:

* Ordenen van knoppen.
* Organiseren van knoppen / keuzes in een raster, zoals bijvoorbeeld de iOS- of Android startschermen.
* Arrangeren van de weergaven zodat alles op gelijke grootte is, zoals bijvoorbeeld in sommige werkbalken.

In tegenstelling tot traditionele tabellen, maakt het bij Grid niet uit hoeveel en in welke grote de rijen en kolommen inhouden. In plaats daarvan heeft Grid RowDefinitions en ColumnDefinitions collecties. Deze bevatten definities van hoeveel rijen en kolommen worden uitgelegd. Uitzicht wordt toegevoegd aan Grid met opgegeven rij- en kolomindices, die bepalen welke rij en kolom een ​​weergave moet worden geplaatst.

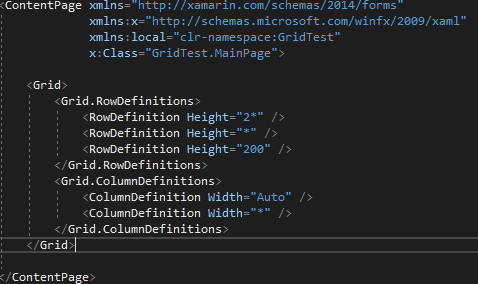
Alle code bij deze workshop zal in XAML zijn.

### Rijen en kolommen toevoegen

Rij- en kolominformatie wordt opgeslagen in de eigenschappen van RowDefinitions & ColumnDefinitions van Grid, die respectievelijk zijn van respectievelijk verzamelingen van RowDefinition en ColumnDefinition-objecten. RowDefinition heeft een enkele eigenschap, hoogte en ColumnDefinition heeft een enkele eigenschap, breedte. De opties voor hoogte en breedte zijn als volgt:

* Auto  
  Maakt automatisch de maten om de rij of kolom aan te passen. Word gegeven als GridUnitType.Auto in C# of als Auto in XAML.
* Proportional  
  Maakt de maten van de rijen en kolommen als een deel van de resterende ruimte. Gegeven als een waarde en GridUnitType.Star in C# en als #\* in XAmL, waarbij # de gewenste warde is. Een rij of kolom worden opgegeven met \*, en zal er voor zorgen dat de beschikbare ruimte wordt gevuld.
* Absolute  
  Maakt de maten van de rijen en kolommen met een specifieke hoogte en breedte waardes. Word gegeven als GridUnitType.Absolute in C# en als # in XAML, waarbij # uw gewenste waarde is.

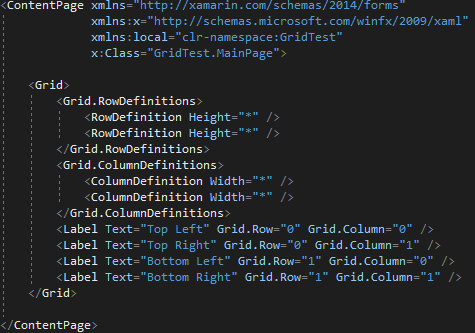
Ga ervan uit dat een app drie rijen en twee kolommen nodig heeft. De onderste rij moet precies 200px lang zijn en de bovenste rij moet tweemaal zo lang zijn als de middelste rij. De linker kolom moet breed genoeg zijn om de inhoud te passen en de rechter kolom moet de resterende ruimte invullen.



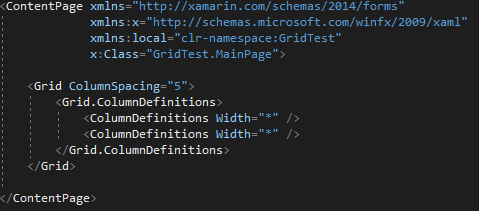
### Weergeven in een grid

Om weergaven in een raster te plaatsen, moet u deze als kinderen toevoegen aan het raster en vervolgens opgeven welke rij en kolom ze behoren.

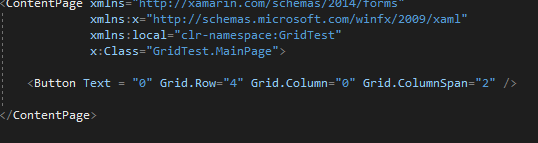
In XAML, gebruik Grid.Row en Grid.Column op elke afzonderlijke weergave om plaatsing te specificeren. Merk op dat Grid.Row en Grid.Column locatie specificeren op basis van de nul gebaseerde lijsten van rijen en kolommen. Dit betekent dat in een 4x4-rooster de linker bovenste cel (0,0) is en de rechter onderkant (3,3) is.



Grid heeft eigenschappen om de afstand tussen rijen en kolommen te regelen. De volgende eigenschappen zijn beschikbaar om het raster aan te passen: ColumnSpacing - de hoeveelheid ruimte tussen kolommen. RowSpacing - de hoeveelheid ruimte tussen rijen. De volgende XAML specificeert een raster met twee kolommen, één rij en 5 px tussenruimte tussen kolommen.



Vaak bij het werken met een raster is er een element dat meer dan één rij of kolom moet bezetten. In XAML kan een button als volgt worden toegevoegd:



## Doorsturen naar een nieuw form

`App.Navigation.PushAsync(new MyCustomContentPage());`

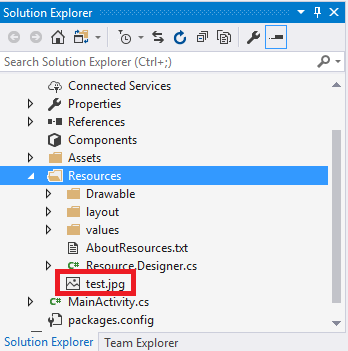
Met dit stukje code kan je een naar een nieuwe page door in de pushasync een nieuwe class aan te maken met de page die je wilt openen. Bijvoorbeeld

Als je een class Secondpage hebt kun je App.Navigation.PushAsync(new Secondpage()); gebruiken. De code maakt dan een nieuwe page aan en switched hiernaartoe. Ook kan je new page veranderen in een al bestaande page.

## Toevoegen van afbeelding aan project

Om een afbeelding toe te voegen aan je project zijn er verschillende manieren beschikbaar.

* Voor iOS plaats je je gewenste afbeelding in de “Resources” map die je kunt vinden in de Solution Explorer.
* Voor Android plaats je de afbeelding in de “Resources/Drawable” map.
* Voor Windows Phone plaats je de afbeelding in de root van je project.
* Voor Windows/UWP plaats je ook de afbeelding in de root van het project.



Om de afbeelding in te laden gebruik je de volgende code:

var testImage = new Image { Aspect = Aspect.AspectFit };

testImage.Source = ImageSource.FromFile("test.jpg");

Om verschillende afbeeldingen voor de verschillende operating systems kun je de volgende code gebruiken:

testImage.Source = Device.OnPlatform(

iOS: ImageSource.FromFile("Images/testt.jpg"),

Android: ImageSource.FromFile("test.jpg"),

WinPhone: ImageSource.FromFile("Images/test.png"));

Hierbij wordt steeds dezelfde afbeelding ingeladen op de verschillende systemen.

# Resources & Styles

## Styles

Xamarin Forms applicaties hebben meestal meerdere controls, die controls hebben dan allemaal een identiek uiterlijk. Het instellen van zo’n uiterlijk kan heel vervelend zijn en veel werk in beslag nemen. Daarom is er Xamarin Forms de styles functionaliteit, met deze functionaliteit word het makkelijker gemaakt om stijlen toe te passen in het design van je applicatie door het toepassen van grouping en settings properties aan passen.

### Global styles:

Over het algemeen zijn alle Xamarin.Forms applicaties gemaakt met behulp van het implementeren van een ‘Application’ subclass. Om een style te kunnen toe te kunnen wijzen in de ‘ResourceDictionary’ met XAML, moet de standaard App class vervangen worden door een XAML App Class en de code erachter.

Dit stuk code laat zien hoe dit is gedaan in C#:

public App ()

{

var buttonStyle = new Style (typeof(Button)) {

Setters = {

...

new Setter { Property = Button.TextColorProperty, Value = Color.Teal }

}

};

Resources = new ResourceDictionary ();

Resources.Add ("buttonStyle", buttonStyle);

...

}

Button Style property ResourceDictionary en het toevoegen van de button style aan de Dictionary

## Style Inheritance

Style inheritance is het toepassen van stijlen die al bestaan met behulp van de Style.BasedOn Stijlen die erven van andere base stijlen kan setters instances bevatten voor nieuwe properties, of het gebruiken om het te overschrijven van base stijlen. Stijlen die geërfd worden van een basis stijl moeten hetzelfde type hebben. Bijvoorbeeld als een basis style target “View” is, elke stijl die daarvan wordt geërfd heeft automatisch de TargetType=”View”.

De volgende code laat zien hoe je dit doet in XAML:

<Style x:Key="labelStyle" TargetType="Label"

BasedOn="{StaticResource baseStyle}">

...

<Setter Property="TextColor" Value="Teal" />

</Style>

<ContentPage.Content>

<StackLayout Padding="0,20,0,0">

<Label Text="These labels"

Style="{StaticResource labelStyle}" />

...

<Button Text="So is the button"

Style="{StaticResource buttonStyle}" />

</StackLayout>

</ContentPage.Content>

In C# geef je in plaats van een TargetType een typeof() mee, zie hier een voorbeeld

var labelStyle = new Style (typeof(Label)) {

BasedOn = baseStyle,

Setters = {

...

new Setter { Property = Label.TextColorProperty, Value = Color.Teal }

}

Children = {

new Label { Text = "These labels", Style = labelStyle },

...

new Button { Text = "So is the button", Style = buttonStyle }

}

De baseStyle van een View instance heeft de properties HorizontalOptions en VerticalOptions. De baseStyle is niet direct gekoppeld aan een control. Maar labelStyles en Button Styles erven van de baseStyle.

### Dynamic styles

De DynamicResource extensie is bijna hetzelfde als StaticResource want ze gebruiken allebei een dictionary key om de values op te vragen voor de stijl. Het enige verschil is dat bij een StaticResourcemaar een keer een in de dictionary kijkt of er aanpassing zijn gemaakt van bijvoorbeeld de values. Bij DynamicResources is dat andersl, namelijk dat de connectie met dictionary blijft bestaan en zo kan de app dus live zijn style aanpassen als deze value is aangepast.

De volgende code laat zien hoe de Dynamic Stijlen werkt in XAML

<ResourceDictionary>r

<Style x:Key="baseStyle" TargetType="View">

...

</Style>

<Style x:Key="blueSearchBarStyle"

TargetType="SearchBar"

BasedOn="{StaticResource baseStyle}">

...

<ContentPage.Content>

<StackLayout Padding="0,20,0,0">

<SearchBar Placeholder="These SearchBar controls"

Style="{DynamicResource searchBarStyle}" />

...

</StackLayout>

</ContentPage.Content>

Bijvoorbeeld als de OnButtonClicked event wordt aangeroepen, zal de searchBarStyle wisselen tussen de blueSearchBarStyle en greenSearchBarStyle. Dit zal als resultaat een andere kleur geven aan het uiterlijk van de app.

public DynamicStylesPage ()

{

InitializeComponent ();

Resources ["searchBarStyle"] = Resources ["blueSearchBarStyle"];

}

void OnButtonClicked (object sender, EventArgs e)

{

if (originalStyle) {

Resources ["searchBarStyle"] = Resources ["greenSearchBarStyle"];

originalStyle = false;

} else {

Resources ["searchBarStyle"] = Resources ["blueSearchBarStyle"];

originalStyle = true;

}

}

## Resources dictionary

ResourceDictionary is een object of manager om alle resources bij te houden en te hergebruiken, aanpassen, toevoegen of verwijderen.

Je kan styles, control templates, data templates, colors en converters opslaan.

In XAML definieer je resources met StaticResource markup extensie. In c# definieer je met string-based index, maar het is makkelijker om direct om resources gelijk aan visueel element te koppelen zonder uit ResourceDictionary te halen.

Op welke niveau je definieert is belangrijk,

* Als je op het control definieert kan het control en zijn inherited controls bij komen bij de resource
* Als je op het scherm definieert kunnen alle controls op het scherm bij de resource
* Als je op de app definieert kunnen alle schermen bij het resource.



Je roept het aan via x:Key=”naam van resources”

Als je op een lager niveau dezelfde resource oproept neemt de laagste de hoogste over, zodat je specifieke aanpassingen kan maken.

## Resources

Met resources kunt u data opslaan die u later in het programma kunt hergebruiken. U kunt een resource hergebruiken met twee types:

* Static Resource
* Dynamic Resource

**Waarom resources**Omdat veel stijlen en opmaak hetzelfde moeten worden gehouden zijn er variabelen nodig om het hierin op te slaan. Ook moet het meteen duidelijk zijn in de code welke opmaak erop is toegepast. Dit wordt gedaan met behulp van resources.   
  
Resources kunt u aanmaken in XAML of C#. Deze resources moeten dan eerst wel in een ResourceDictionary worden opgeslagen zodat het in blokken kan worden aangeroepen.  
  
**Noodzakelijk voor resources**Omdat elke resource moet worden geïdentificeerd moet deze een unieke key hebben. Dit wordt gedaan met static- of dynamic resource. Hierdoor weet het programma welke resource kan worden hergebruikt. Deze resource key geeft u als type mee in de resource tag.  
  
**Variabel**Het belangrijkste van een resource is het feit dat de resources variabel zijn. Hierbij kunt u eigenschappen aan resources toepassen en deze wijzigen. Wel moet er dan rekening gehouden worden met static resources.

### Static resource

Static resources worden voor het compileren uitgevoerd. Dit betekent dat static resources eenmaal wordt uitgevoerd. Tijdens het uitvoeren van de applicatie kunnen hierbij dan geen veranderingen worden aangepast. Static resources kunnen hierbij dan niet worden gerefereerd. De volgende XAML code laat zien hoe u een static resource aanroept:



**StaticResource:** *De resource type die u aangeeft voor het aanroepen van een bepaalde resource.*

**Key:** *De identiteit van de resource die u aanroept.*

Een static resource zet u als volgt op:

1. Maak een nieuwe **.XAML** aan in **Visual Studio**
2. Maak een **ResourceDictionary tag** aan

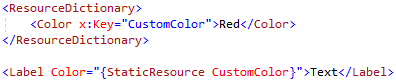


1. Maak in de **ResourceDictionary** een resource aan. Dit mag elke resource zijn die u wilt. In dit geval wordt er een **resource** **Color** aangemaakt met een **x:Key=”CustomColor”.**



1. Als laatst roept u de resource aan met **static resource**.

Eindresultaat

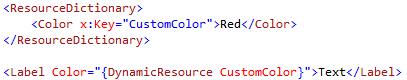


### Dynamic resource

Met dynamic resources kunt u een resource wijzigen tijdens het uitvoeren van de applicatie. Wat bij static resources het geval was dat je voor het compileren resources kunt uitvoeren en wijzigen is bij dynamic resources niet het geval.   
  
Deze kunt u dan alleen maar voor gebruik maken tijdens uitvoering. Het aanroepen van een dynamic resource lijkt heel veel als het aanroepen van de static resource. Hierbij is het enige verschil dat u in plaats van StaticResource een DynamicResource aanroept. De volgende opzet XAML code laat zien hoe.



**Refereer in C#**Het refereren met dynamic resource werkt wel en bij static resources niet omdat deze tijdens het uitvoeren van de applicatie kan worden gewijzigd. Hierbij wordt er van de volgende XAML code vanuit gegaan:



Om de CustomColor te refereren doet u als volgt:

  
Hiermee kunt u gemmakelijk de resources hergebruiken op elke gepaste object in code.

### Waarom static- of dynamic resource

Omdat een static resource alleen voor uitvoering van de applicatie wordt uitgevoerd is het vaak handig om met behulp van static resources een template te maken.   
  
Dit omdat er in templates niets veranderd hoeft te worden omdat deze resources niet dynamisch zijn. Het grote voordeel hierbij is dat de snelheid wordt behouden. Dit is niet het geval bij dynamic resources. Wel kunt u de applicatie updaten met behulp van dynamic resources. Hiermee wordt bedoeld dat u controls kunt maken die u wanneer u wilt kan aanpassen.  
  
Met static resources met behulp van ResourceDictionary kunt u verschillende thema’s maken. Als u dit met dynamic resources probeert kunnen de thema’s er heel anders uitzien omdat er tijdens uitvoering van de applicatie resources kunnen worden gewijzigd en niet statisch zijn.

Het beste is om altijd te beginnen met static resources. Alleen als u zeker weet dat u dynamic resources nodig heeft gebruikt u deze.   
  
De volgende voor- en nadelen nogmaals op een rijtje:  
  
**Static resource**

|  |  |
| --- | --- |
| + | - |
| Snelheid | Niet aan te passen tijdens uitvoering van de applicatie |
| Geschikt voor templates | Kan langer duren voordat de applicatie wordt opgestart |

**Dynamic resource**

|  |  |
| --- | --- |
| + | - |
| Updaten van de programma | Snelheid |
|  | Kan geen uitvoeringen doen voor het compileren |
|  | Niet geschikt voor templates |

# Databinding

## Waarom databinding?

Om te weten waarom je databinding zou gebruiken raden we aan om eerst te kijken naar het hoofdstuk: Wat is databinding?

Er zitten zowel voordelen als nadelen aan databinding, voordat je databinding gaat gebruiken raden we je aan om deze voordelen en nadelen goed door te lezen.

Voordelen:   
- Makkelijk om te gebruiken in .net.  
- Aanrader als je live op de hoogste gehouden wilt worden.   
- Werkt goed bij simpele interfaces die niet veel anders doet dan data weergeven.

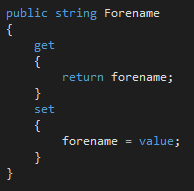
Nadelen:   
- Code kan ongestructureerd worden.   
- Er kunnen met databinding geen berekening worden uitgevoerd.

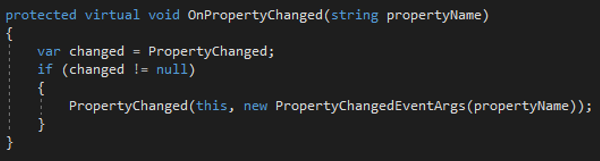
Voordat je databinding gaat gebruiken is het dus eerst van belang dat je weet wat je gaat maken. Het kan in sommige gevallen meer tijd gaan kosten dan dat het oplevert.

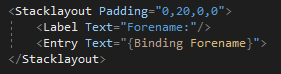
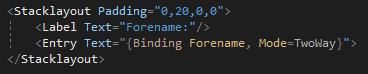
## Implementeren van data-binding

In dit gedeelte gaan we het hebben over hoe je data-binding kan gaan implementeren in Xamarin.

Als eerst gaan we het hebben over de implementatie in de CS file achter je XAML file. Om te beginnen moet we het ViewModel laten overerven van INotifyPropertyChnged. Dit doe je zo:  
  
  
  
Zoals uitgelegd in paragraaf {PLAATS PARAGRAAF NUMMMER HIER!} bind je met data-binding een property aan een andere property. Hierdoor zal je voor elke property die je gebind wilt hebben in je CS file een public property aanmaken met een get set en hier ook een private property voor maken. Dit doe je zo:

(Wanneer we deze property later binden in de XAML zal automatisch de set of get van de gebinde property aanroepen.)  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Hierna zul je een methode en een event moeten aanmaken voor wanneer de set aangeroepen wordt van de gebinde variabele. Dit doe je zo:  
  
  
(Dit zal mogelijk al automatische toegevoegd zijn wanneer we de eerste stap hadden uitgevoegd zo niet voeg dit alsnog toen.)  
  
  
(Deze methode zal moeten worden aangeroepen bij elke set van een gebind property en deze updaten voor alle elementen in de view die het gebinde property gebuikt.)  
  
Als laatst zal je in de CS file een aanroep moeten doen aan de methode in elke set van elk gebinde property. Dit doe je zo:  
  
  
(Dit zal de vorige methode die we hebben gemaakt aanroepen.)  
  
Hierna moeten we in de Mainpage.cs dit toevoegen onder de InitializeComponent:  
  
  
(Dit zal er voor zorgen dat de view in staat is data-binding te gebruiken.)

Als laatst moet we in de XAML de property gaan binden aan die in de CS file. Dit doe je zo:  
  
  
(Dit zal ervoor zorgen dat er een label met een textbox verschijnt. In de textbox zal dan de property waarde te zien zijn die momenteel gebind is hieraan, in dit geval “Forename”.)  
  
  
  
  
  
  
  
  
Data-binding in Xamarin kan one-way of two-way dit wordt uitgelegd in paragraaf {PLAATS PARAGRAAF NUMMMER HIER!}. Om dit aan te geven moet je dit achter de binding aangeven. Dit doe je zo:  
  
  
(Hier geef je dan aan of je het TwoWay of OneWay data-binding wilt hebben. Als je dit niet aangeeft is het standaard TwoWay.)  
  
Als je al deze stappen hebt uitgevoerd heb je een werkend voorbeeld van databinding, en weet je hoe je dit overal kan implementeren.

## Databinding MVVM :

Zet dit in je Content Page

“xmlns:sys="clr-namespace:System;assembly=mscorlib"”

Dit verbind sys met DLL (mscorlib) windows lib.  
MSCORLIB, (Microsoft Core Library)geeft je de mogelijkheid on in de XAML designer functies te gebruiken die je anders zou gebruiken in de C# code.

Het verbonden item SYS wordt gebruikt om de huidige datum op te vragen maar statisch gemaakt en het blijvend binden. Dit houd dus ook in dat het blijft updaten.

Zie de code als volgt.

<StackLayout BindingContext="{x:Static sys:DateTime.Now}">

<Label Text="{Binding Year, StringFormat='The year is {0}'}" />

<Label Text="{Binding StringFormat='The month is {0:MMMM}'}" />

<Label Text="{Binding Day, StringFormat='The day is {0}'}" />

<Label Text="{Binding StringFormat='The time is {0:T}'}" />

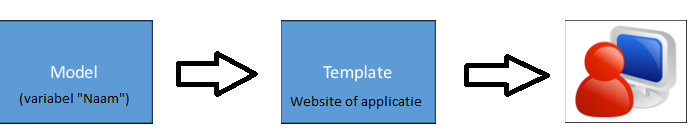
</StackLayout>

Door het aan de Stack Layout het te binden kan je dus alles wat verbonden moet zijn met het sleutelwoord “Binding” de functies gebruiken die het object heeft die je meegeeft.

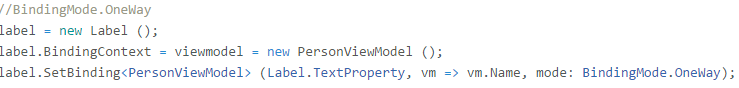
## One-way binding

One way binding betekend simpel gezegt dat data alleen maar opgehaald mag worden. Wanneer de data veranderd, veranderd dit ook in de data die op het beeld staat.

Stel je hebt een variabel genaamt “Naam” waarin een naam is opgeslagen, kan deze dus niet aangepast worden als er een one way binding is toegepast, maar alleen maar opgehaald worden.



Een voorbeeld van hoe one way binding toegepast kan worden:



# Listviews & lazy loading

## Lazy loading

Lazy Loading:

Bij lazy loading laad je het object pas als je het nodig hebt. Bijvoorbeeld een plaatje van facebook wat eerst nog buiten beeld stond en pas wordt geladen als het in beeld komt.

* Voordelen:

Omdat objecten pas geladen worden als ze nodig zijn, is je applicatie sneller bij het opstarten.

* Nadelen:

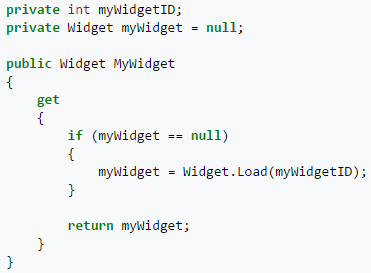
Met lazy loading krijg je misschien niet meteen alle data die je wel nodig hebt, waardoor je vaker dingen moet opvragen.

Eager Loading:

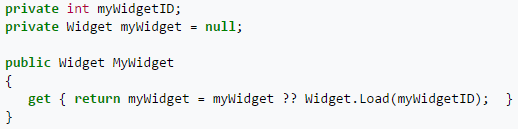
Bij eager loading laad je alle objecten bij de start-up van de applicatie.

Lazy initialization:

* Het object dat lazy loaded moet worden, wordt in eerste instantie op null gezet. Elke aanvraag voor het object wordt gekeken of het null is en dan wordt er een “on the fly” object, voordat het returned wordt.



* Of met de  [null-coalescing operator](https://en.wikipedia.org/wiki/Null_coalescing_operator): '??'



Deze methode is het makkelijkste om te implementeren maar als null een geldige return waarde is dan is er een placeholder nodig om aan te geven dat het object nog niet geïnitialiseerd is.

Implementatie Lazy Loading:

Er zijn 4 veel voorkomende manieren om lazy loading te implementeren:

lazy initialization; een ‘virtual proxy’; een ‘ghost’ en een ‘value holder’.

* Virtual proxy:

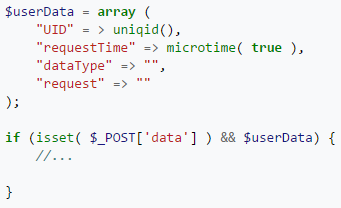
Een virtual proxy is een object met dezelfde interface als het originele object. Als de methode voor de eerste keer wordt geroepen dan wordt het originele object geladen en wordt het gedelegeerd.

* Ghost:

Een ghost is een object dat wordt geladen in een gedeeltelijke staat. De ghost mag alleen de identifier van het originele object bevatten, maar laad pas zijn eigen data als er voor de eerste keer een van de properties benaderd wordt.

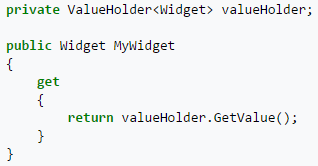
Voorbeeld:

Stel dat een gebruiker op het punt is om inhoud te vragen via een online formulier. Op het moment van de creatie weten wij dat de data toegankelijk is, maar welke actie of inhoud is onbekend.



* Value holder:

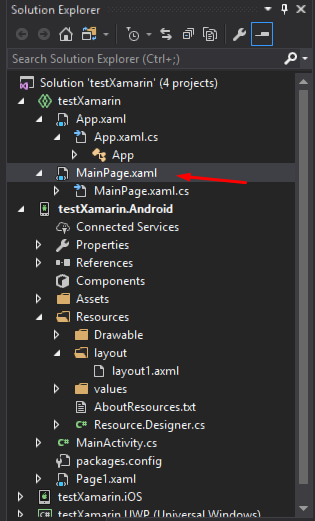
Een value holder is een generiek object dat het gedrag van de lazy loading hanteert en verschijnt op de plaats van de datavelden van het object.



## Listview

ListView aanmaken:

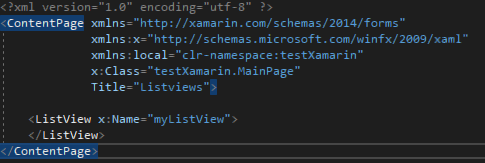
1. Begin eerst met het aanmaken van een project met naam naar keuze.
2. Navigeer dan naar MainPage.xaml.



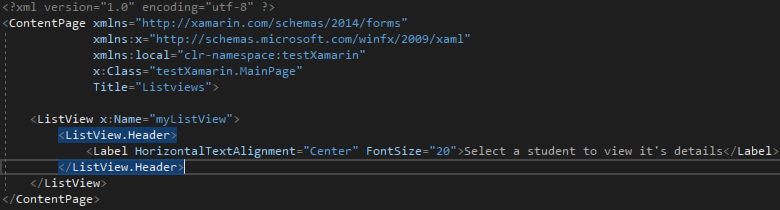
1. Maak vervolgens een ListView item aan naam naar keuze.

(Probeer voor het gemak de naam zo kort mogelijk te houden)

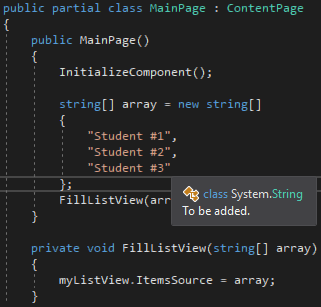
(De ListView past zich automatisch aan de schermgrootte aan omdat je geen grootte of breedte waardes aangeeft)

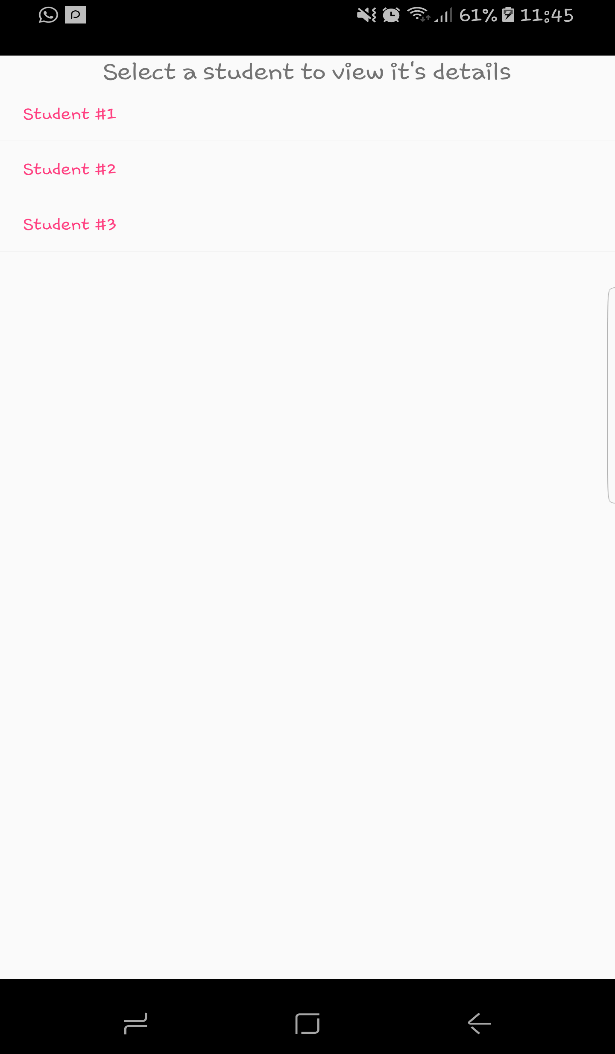


1. Om de ListView een onderwerp te geven gebruiken we een simpele methode om een grote gecentreerde titel toe te voegen met tekst naar keuze.



1. Ten slotte kunnen we heel gemakkelijk in de code de ListView invullen.



Resultaat:

# Deployment

## Deployment naar android

Ook al zijn er veel markets voor het uitbrengen van apps, Google Play is de grootste met de meeste bezoekers in de wereld voor Android apps. Google Play is een plaats waar je apps kan uitbrengen, adverteren, verkopen en waar je uitgebrachte apps kan analysere.

### Wordt een uitgever

Als je een applicatie wilt publiceren op Google Play, dan is een uitgever account nodig. Je kan jezelf aanmelden voor een uitgever account door de volgende stappen uit te voeren.

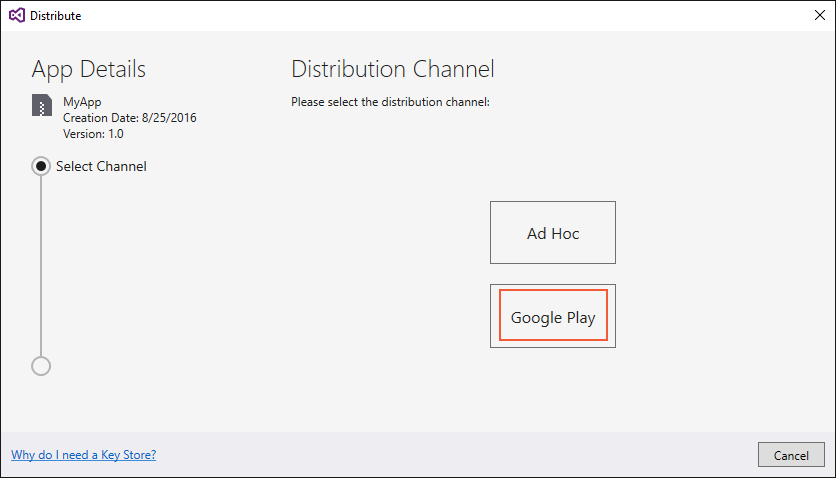
1. Bezoek de Google Play Developer Console[[2]](#footnote-2)
2. Vul standaard informatie in over je developers identiteit
3. Lees en accepteer de Developer Distribution Agreement
4. Betaal $25,- registratie kosten
5. Bevestig je aanmelding

Xamarin Android 7.0 Heeft een ingebouwde workflow voor het publiceren van apps naar Google Play via Visual Studio. Als je een versie van Xamarin Android voor 7.0 gebruikt, moet je de APK uploaden naar Google Play via het developer Console.  
Ook moet je minstens een applicatie hebben geüpload naar Google Play om gebruikt te maken van de ingebouwde workflow.

1. Log in op je Google Play Developer account om een project aan te maken.
2. Maak een OAuth Client, waarmee je je app machtigt.
3. Vul een resulting Client ID en Client secret in bij Visual studio
4. Registreer je account
5. Log in met je certificate
6. Publiceer jou op Google Play

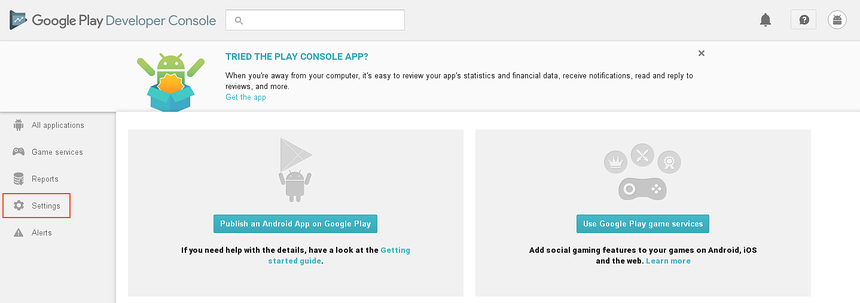
Om je app te publiceren volg je de volgende stappen:

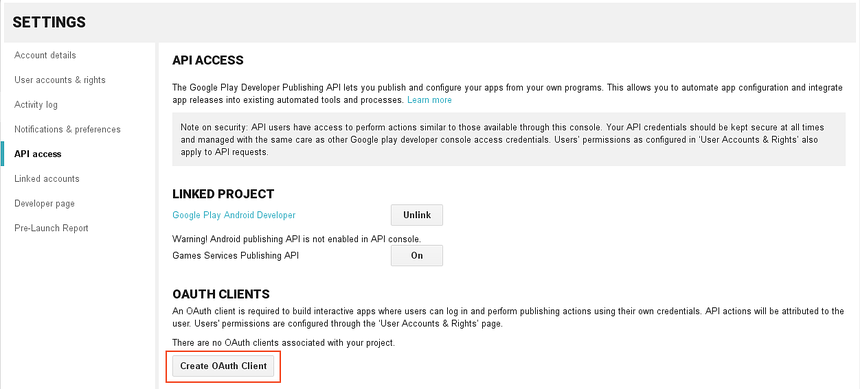
Rechtermuisklik op je project en klik op archive. Zodra je hierop hebt geklikt opent je archive manager.

[](https://developer.xamarin.com/guides/android/deployment,_testing,_and_metrics/publishing_an_application/part_3_-_publishing_an_application_on_google_play/Images/vs/01-distribution-channel.png)  
Het distribution channel opent, hier klik je op Google Play. Hierna kom je in het Signing Identity scherm, waar je je eigen identity selecteert en verder gaat. Daarna is het tijd om je Google Play Developer account toevoegen door middel van het +-teken.

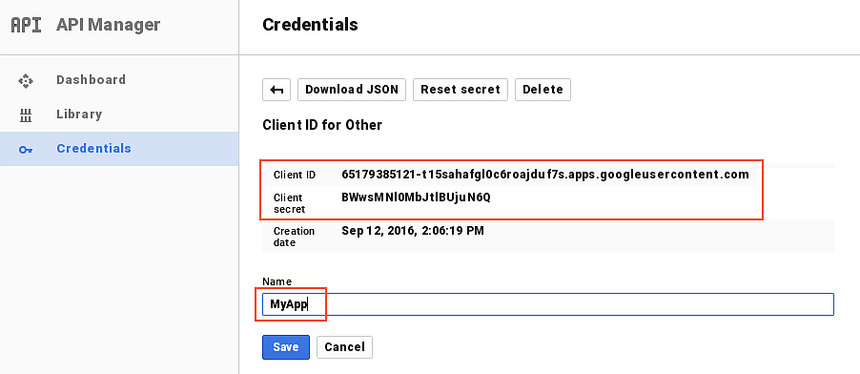
### Google API project aanmaken

Eerst log je in met je Google Play Developer account.

[](https://developer.xamarin.com/guides/android/deployment,_testing,_and_metrics/publishing_an_application/part_3_-_publishing_an_application_on_google_play/Images/01-google-play-developer-console.png)  
Op de settings pagina, selecteer je API Access en creëer je een nieuw project.  
Na een korte periode van een minuut is je project gegenereerd en gelinkt aan je Google Play Developer Account. De volgende stap is het maken van een OAuth Client voor de app (Als je deze nog niet hebt aangemaakt). Wanneer gebruikers vragen naar toegang tot hun persoonlijke data in jouw app, is jouw OAuth Client ID gebruikt voor het verifiëren van je app. Klik op Create OAuth Client voor het maken van een nieuwe OAuth Client.

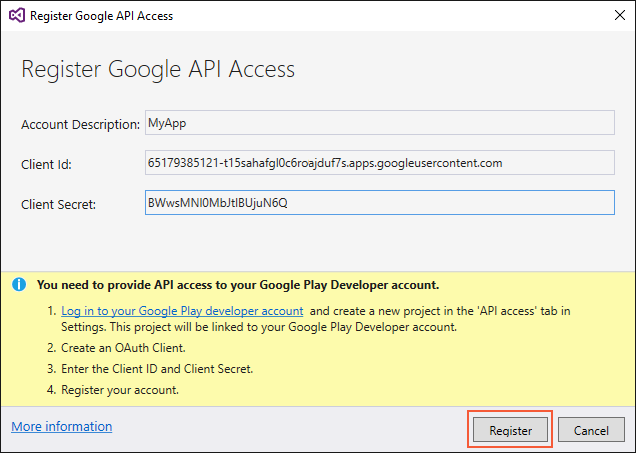
[](https://developer.xamarin.com/guides/android/deployment,_testing,_and_metrics/publishing_an_application/part_3_-_publishing_an_application_on_google_play/Images/03-create-oauth-client.png)

Na een paar seconden is je Client ID gegenereerd. Om dit ID te zien, ga je naar View in Developers Console. De Client ID wordt getoond naast de naam en aanmaakdatum. Klik op het Edit OAuth Client icoon zodat je de client van de app te zien krijgt. De standaard naam van de OAuth client is Google Play Android Developer. Dit kan veranderd worden naar de naam van de Xamarin Android app, of natuurlijk een andere bijpassende naam. In het voorbeeld is de naam van de app veranderd naar: MyApp.

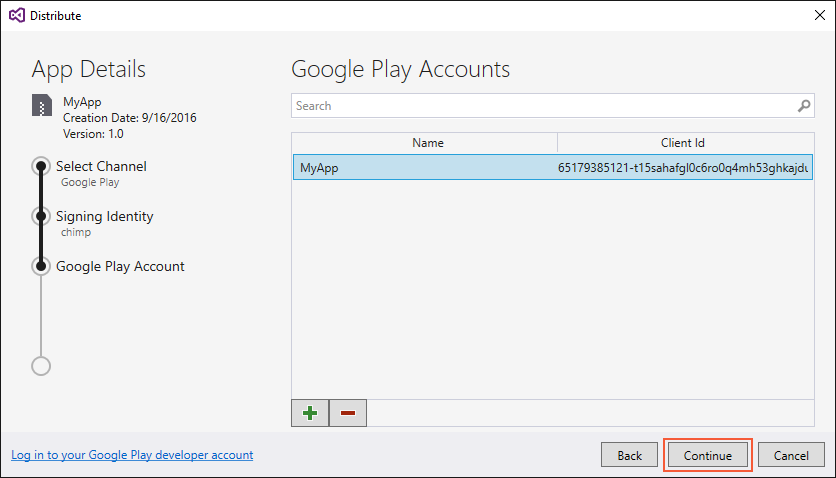
[](https://developer.xamarin.com/guides/android/deployment,_testing,_and_metrics/publishing_an_application/part_3_-_publishing_an_application_on_google_play/Images/06-client-id-and-secret.png)

Nadat je dit hebt opgeslagen, kan je een JSON-bestand downloaden, waarin deze gegevens zitten opgeslagen. Deze heb je later nodig om je API Access te registreren.

### Registreer Google API Access

Je Cliend ID en Client Secret gebruik je om je Google Play API Account te voltooien. Zodra je op registreren hebt geklikt opent er een webbrowser.  
[](https://developer.xamarin.com/guides/android/deployment,_testing,_and_metrics/publishing_an_application/part_3_-_publishing_an_application_on_google_play/Images/vs/05-enter-client-id-and-secret.png)

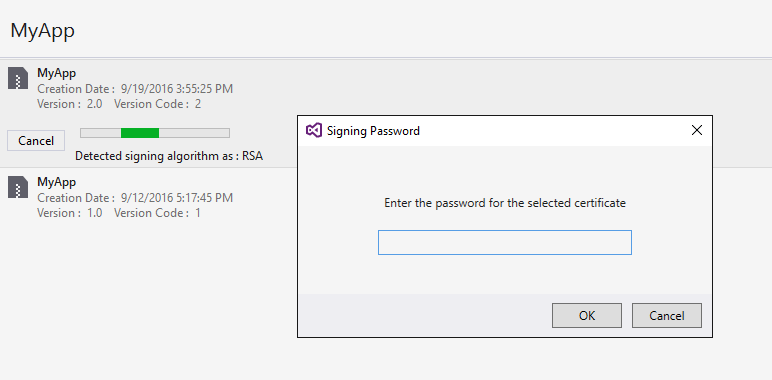
### Publiceren

[](https://developer.xamarin.com/guides/android/deployment,_testing,_and_metrics/publishing_an_application/part_3_-_publishing_an_application_on_google_play/Images/vs/07-account-added.png)

Het volgende scherm wordt getoond, hierin zie je je Google Play accounts staan. Selecteer het juiste Google Play account en klik op continue. Vervolgens wordt het Google Play Trackscherm getoond.   
Hierin worden je vier opties geboden voor het publiceren van je app.

* Alpha: Wordt gebruikt om een vroege versie van een applicatie beschikbaar te stellen voor een kleine groep testers.
* Beta: Wordt gebruikt om een mindere vroege versie van een applicatie beschikbaar te stellen voor een kleine groep testers.
* Rollout: Laat een percentage van de gebruikers een update ontvangen van de app, zodat je langzaam updates kan testen met een gedeelte van je gebruikers.
* Production: Selecteer dit als de app klaar is voor de Google Play Story.

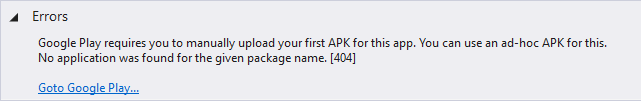
Kies welke track je wil gebruiker voor het uploaden van de app en klik op upload. Als je rollout hebt geselecteerd, moet je ook een percentage invullen.

[](https://developer.xamarin.com/guides/android/deployment,_testing,_and_metrics/publishing_an_application/part_3_-_publishing_an_application_on_google_play/Images/vs/09-certificate-password.png)

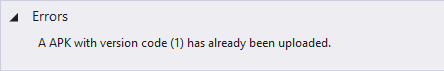
Nadat je je track hebt gekozen, wordt je gevraagd het wachtwoord in te vullen. Zodra je deze hebt ingevoerd, zal je app worden geüpload naar de Google Play Store. Tevens laat het scherm je een voortgang zien door middel van een progress bar.

### Troubleshooting

Wees ervan bewust dat de APK eerst handmatig moet worden toegevoegd, voordat je gebruik kan maken van het publishen.

[](https://developer.xamarin.com/guides/android/deployment,_testing,_and_metrics/publishing_an_application/part_3_-_publishing_an_application_on_google_play/Images/vs/12-upload-error.png)

Mocht je deze error krijgen, zal je handmatig een APK up moeten loaden naar de Google Play Store.[[3]](#footnote-3)

[](https://developer.xamarin.com/guides/android/deployment,_testing,_and_metrics/publishing_an_application/part_3_-_publishing_an_application_on_google_play/Images/vs/13-version-code-error.png)

Als je daarna last hebt van deze error, zal je je applicatie moeten rebuilden, dan zou deze error moeten verdwijnen.

# FAQ

1. Zie: https://xamarinhelp.com/portable-class-library-pcl-vs-shared-projects/ [↑](#footnote-ref-1)
2. https://play.google.com/apps/publish/signup/ [↑](#footnote-ref-2)
3. https://developer.xamarin.com/guides/android/deployment,\_testing,\_and\_metrics/publishing\_an\_application/part\_3\_-\_publishing\_an\_application\_on\_google\_play/ [↑](#footnote-ref-3)