|  |  |
| --- | --- |
| PHL_logoTXT | C:\Users\Koen\Pictures\Logo (Vlag).png |

**CITYGAME**

**HTML5 vs. Native**

**Departement Toegepaste Informatica**

[phl@phlimburg.be](mailto:phl@phlimburg.be)

[www.phl.be](http://www.phl.be)

**eXplore-it:** (Groep 2)

* Yoni Pervolarakis
* Geoffry Nagy
* Kristof Aerts
* Harm Froyen
* Sander Vanbrabant
* Kimberly Spitznagel
* Koen Vanderlinden
* Yannick Stevaert
* Ruben Leten

**Begeleider:**

Bram Heyns

**Academiejaar 2012 - 2013**

Table of Contents

HTML5 3

Het werkveld 3

Hybride applicatie 3

Monotouch 6

HTML5 vs. iOS 7

HTML5 vs. Android 8

HTML5 vs. Windows Phone 11

Webservice 12

# HTML5

HTML5 is de verzamelnaam voor HTML, CSS en JavaScript. Met de komst van de nieuwe standaardisering van HTML5, belooft men dat je een applicatie eenmalig ontwikkelt en deze snel en makkelijk kan uitbreiden naar verschillende platformen.

Wanneer men HTML5 en mobile in de mond neemt, kan men twee kanten op. Je hebt enerzijds de webapplicatie waarbij je via de browser de applicatie bezoekt. Anderzijds is er de hybride applicatie waarbij een framework gebruikt wordt om de HTML5 files te converteren naar een applicatie die native aanvoelt.

# Het werkveld

Het werkveld van onze bachelorproef is beperkt tot de vergelijking van HTML5 hybride applicaties en native applicaties. Hiervoor hebben we een applicatie ontwikkelt voor de platformen Android, iOS en Windows Phone 7.

# Hybride applicatie

***PhoneGap***

Het framework dat wij gebruikt hebben om de hybride applicatie te ontwikkelen is het opensource PhoneGap. PhoneGap maakt gebruikt van application container technologie. Dit wil zeggen dat PhoneGap zorgt voor een encapsulatie van de HTML pagina’s, de CSS sheets en JavaScripts. Phonegap doet dit door een instantie te gebruiken van een uitgeklede versie van een webbrowser. Deze webbrowser is steeds native aan het platform en is voor de meeste platformen gebaseerd op het opensource Webkit. De applicatie wordt dus weergegeven in een chromeless webbrowser. Dit wil zeggen dat de browser geen randen of browserbalk bevat. De user interface is dus een instantie van deze webbrowser met een maximale schermgrootte waarvan de gebruiker niets merkt.

*Toegang tot native features*

PhoneGap biedt toegang tot de meest gebruikte features op een toestel. Het is mogelijk om de camera op te roepen, GPS-lokalisatie te gebruiken of zelfs de telefooncontacten te importeren.

*Plugins*

Om specifieke functionaliteit aan te bieden in een applicatie, maakt men gebruik van plug-ins. In native applicaties, komen ze vrij veel voor en zijn ze uiteraard ontwikkeld in dezelfde programmeertaal als de applicatie.

Een hybride applicatie volgt hetzelfde stramien, maar met het verschil dat de plug-ins geschreven moeten worden in de taal van het target mobile operating system.

Wanneer de eisen niet beantwoordt kunnen worden met HTML5 technologie, kan men gebruik maken van deze plug-ins.

*Performantie*

De snelheid en responsiviteit van een hybride applicatie ligt beduidend lager dan zijn native tegenhanger. Dit heeft te maken met de encapsulatietechnologie die PhoneGap gebruikt om de applicatie weer te geven.

Native applicaties worden geschreven in dezelfde taal als de machine compiler taal van het mobile operating system. Dit is een duidelijk voordeel qua performantie.

Dit kan men opvangen door :

* Efficiënt inladen JavaScripts en DOM-elementen.
* Zo weinig mogelijk elementen opzoeken in de DOM-structuur maar bijhouden in variabelen.
* JavaScript mainthread zo logisch en efficiënt opbouwen.
* Gebruik van AJAX-calls
* Onderzoek verrichten naar efficiënter gebruik HTML, CSS en JavaScript per native webbrowser die PhoneGap gebruikt in zijn technologie.
* Visuele animaties behandelen met CSS3 attributen zodat ze verwerkt worden door de GPU. (taakverdeling CPU en GPU)

*Ontwikkeling*

Middelen

Aangezien HTML5 bestaat uit websitetechnologie, zijn er meer geschoolde mensen op de arbeidsmarkt om een HTML5 applicatie te ontwikkelen dan een native.

Duur

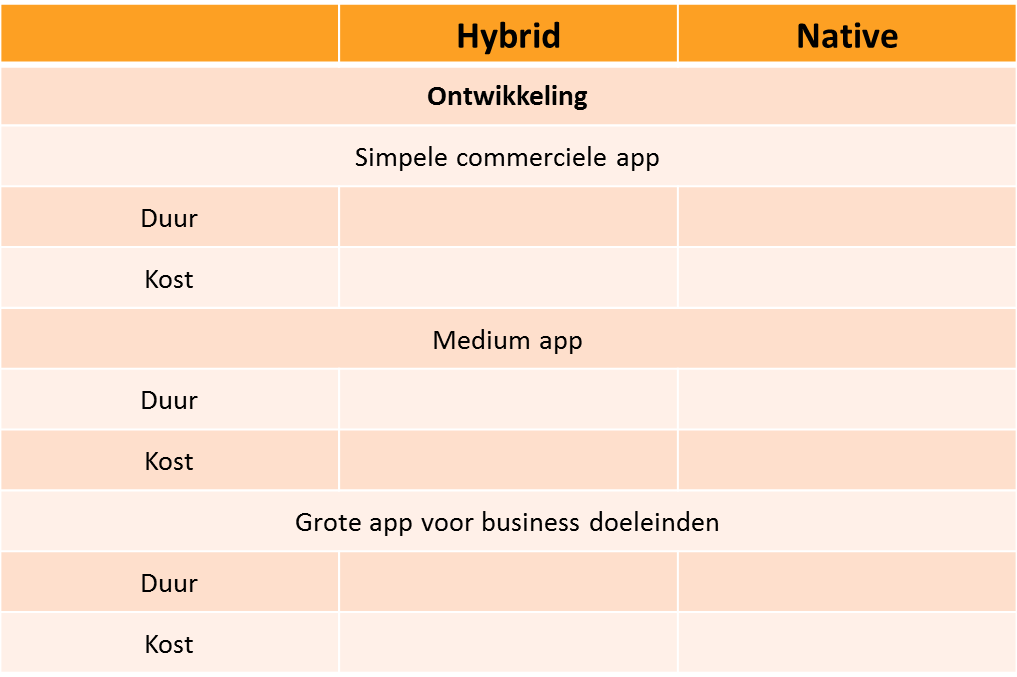
Een HTML5 applicatie is sneller ontwikkeld dan zijn native tegenhanger. Dit is te danken aan het feit dat HTML, CSS en JavaScript makkelijkere en overzichtelijkere omgevingen zijn.

Kost

De kost om een hybride applicatie te ontwikkelen ligt beduidend lager dan een native. Dit heeft te maken met enerzijds een grotere aanbod op de arbeidsmarkt en anderzijds de duur om een HTML5 applicatie te ontwikkelen.

Maar naarmate de complexiteit van een applicatie toeneemt, stijgen ook de kosten. Dit heeft te maken met de beperkingen van HTML5. Wanneer men de overstap moet maken naar de ontwikkeling van plug-ins, dan is dit een tijdrovend proces aangezien deze plugins geschreven moeten worden in de taal van het target mobile operating system.

In figuur 1 wordt dit geïllustreerd.



Figuur - Kost en duur ontwikkeling Hybrid vs native

*Keuze PhoneGap*

Om uit te klaren waarom we gekozen hebben voor PhoneGap tussen de vele beschikbare frameworks worden hier even opgesomd :

* Opensource
* Voldaan aan features
* Plug-ins beschikbaar
* Support en community
* Beschikbare debugging tools : WEINRE en Ripple
* Combinatie mogelijk met jQuery Mobile, een UI plug-in.

PhoneGap is opensource, meteen de grootste reden waarom we dit framework hebben gekozen. De community en support zijn uitstekend. PhoneGap biedt toegang tot alle features en functionaliteiten die we gebruiken de ontwikkeling van CityGame. Qua debugging tools heeft PhoneGap een eigen tool, namelijk WEINRE. Het is gebaseerd op de inspectietool van Google Chrome. Ripple is een desktop browser plugin waarmee je de applicatie snel kan debugging op visuele fouten en device eventhandling.

jQuery Mobile is een jQuery plugin die de handling van het user interface op zich neemt.

# Monotouch

Android en iOS hebben gewerkt met Monotouch.

Monotouch is een product van het bedrijf Xamarin. Het is een crossplatform gemaakt om te werken in .NET en C#. De bedoeling is dat er in plaats van Java of Objective-C gecodeerd wordt, het nu in C# gebeurd. Monotouch kan je bekijken als een vertaler, hij vertaalt de C# geschreven code naar een native code. Dit heeft als voordeel dat er code uitgewisseld kan worden.

Monotouch is verder een multiplatform. Deze kan dus zowel draaien op Linux, Mac als Windows. Verder kan je Monotouch vergelijken als de gewone gekende IDE met intellisense en debugmogelijkheden.

Monotouch heeft enkele voor- en nadelen die we even gaan toelichten.  
  
Monotouch heeft als voordeel dat dit een cross platform is, hierdoor kunnen wij code uitwisselen. Dit is echter wel niet altijd het geval. Wanneer er specifieke code wordt aangesproken, zoals een Toast-object of een viewDidLoad, kunnen we deze niet uitwisselen, maar een klasse kunnen we volledig samen gebruiken.  
Met wat basiskennis van C# is het heel gemakkelijk om te starten in Monotouch. Voor personen die liever in C# dan in Java programmeren is dit programma ideaal. Sommige functionaliteiten in C# zijn gemakkelijker aan te spreken dan in Java. Bijvoorbeeld het aanspreken van de webservice.

Vermits Monotouch nog niet zo bekend is, is er weinig documentatie te vinden op hun website. Op internet is ook niet veel te vinden vermits dit niet zoveel gebruikt wordt.  
Monotouch vertaalt C# naar de native taal in de achtergrond, maar het is niet mogelijk voor de programmeur om deze te bekijken.

Problemen die we tegenkwamen met Monotouch waren wanneer men een button definieerde in Xcode en deze dan opsloeg, hij alle klassen die gemaakt waren wiste. De enigste oplossing hiervoor was Xcode sluiten en heropstarten. Hierna waren alle klassen weer aanwezig.

# HTML5 vs. iOS

Het grote verschil dat we hebben gemerkt in iOS bij het coderen tussen HTML5 en iOS is bijvoorbeeld het gebruik maken van native API’s. Werkende met Monotouch kan je al geen gebruik maken van Objective-C API’s. Hierdoor moet er dus gezocht worden naar een vertaalde C# API.  
Verder worden iOS Apps via de AppStore vrijgegeven terwijl dit vroeger bij PhoneGap niet mogelijk was. Sinds de laatste update van PhoneGap kunnen hybride oplossingen ook in de AppStore terecht komen. Jammer genoeg heeft Apple strenge toelatingsvoorwaarden en laten ze niet snel hybride oplossingen toe. Uiteindelijk heeft Apple hier het laatste woord in. Updates worden ook gedistribueerd via de AppStore. HTML5 kan updaten via het internet (of url), maar ook via de AppStore als de applicatie toegelaten werd.   
Het nadeel van iOS tegenover HTML5 is dat iOS Apple Devices only is. Ook duurt het iets langer om de GPS, Camera of Accelerometer aan te spreken dan wanneer men een native oplossing heeft. Dit komt omdat Objective-C betere en snellere toegang heeft tot de hardware van het apparaat.

Het grote verschil tussen de 2 applicaties is het gebruik van de maps. HTML5 maakt gebruik van Google Maps terwijl iOS gebruik maakt van Apple Maps. Het was een bewuste keuze om gebruik te maken van Apple Maps. Google Maps had in december voor het eerst de api-keys beschikbaar gemaakt voor het gebruik van Google Maps in een iOS applicatie. Heden ten dage hebben we nog steeds geen toegang verkregen tot deze api-key. Sinds de laatste update van Apple voor Apple Maps is Apple Maps al een heel stuk verbeterd. Hierbij zijn de grootste bugs opgelost waardoor het stilaan zijn reputatie begint terug te krijgen.

Ook is de performance van de 2 mappen verschillend. iOS heeft Apple Maps al een deel gecached waardoor het inzoomen en uitzoomen zonder internet niet veel problemen geeft. Dit is weliswaar een probleem voor de HTML5 versie. Deze moet eerst de hele map laden waardoor de performance wat minder is.

Bij de native oplossing werd er gebruik gemaakt van verschillende kleuren pinpoints. Om deze pinpoints te verkrijgen moeten er verschillende klassen gemaakt worden die de originele klassen overriden zodat we verschillende kleuren kregen. Bij Google is dit iets simpeler door een afbeelding te kiezen.

Verder was er niet zo’n groot verschil tussen iPad en iPhone programmeren. De lay-out moet wel telkens geherdefinieerd worden. Dit wilt zeggen als er een button bijkomt op de iPhone applicatie dat deze ook moet getekend worden op de iPad applicatie.

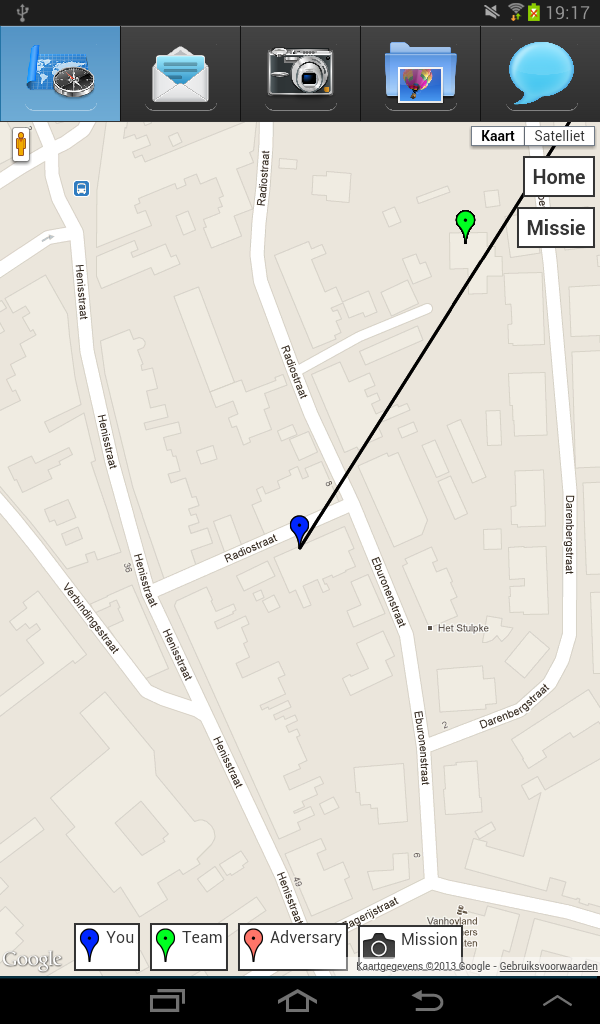
Qua code veranderd er niet veel behalve bij het ophalen van foto’s bij het album. iPad verwacht dat er een popover komt terwijl bij iPhone dit het hele scherm in neemt.

|  |  |
| --- | --- |
| Screen Shot 2013-02-19 at 11.29.31.png | Screen Shot 2013-02-19 at 11.30.47.png |

*Verschil album tussen iPhone en iPad*

# HTML5 vs. Android

Het grote verschil tussen HTML5 en Android is de lay-out. Neem nu als voorbeeld de tabbar op de 2 platforms. Op de 2 onderstaande afbeeldingen zie je een duidelijk verschil van lay-out. De tabbar bij HTML5 zijn eigenlijk buttons waarvan de lay-out volledig gedefinieerd is via CSS. Hoewel de Android applicatie geprogrammeerd is via Monotouch, heeft deze dezelfde look-and-feel als een native Android applicatie. Dit komt omdat Monotouch de code in C# vertaalt naar de native code van Android.

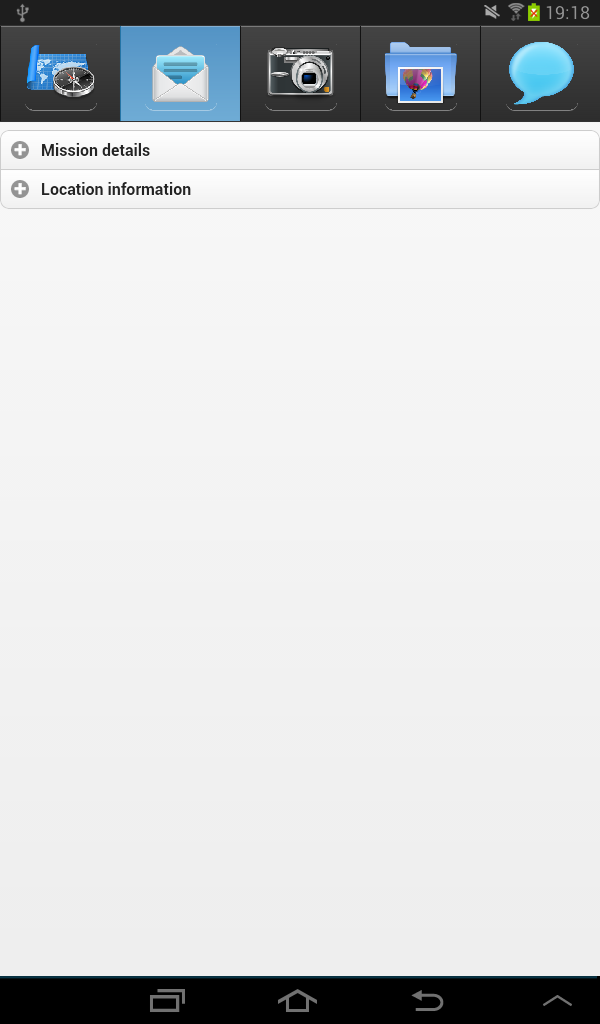
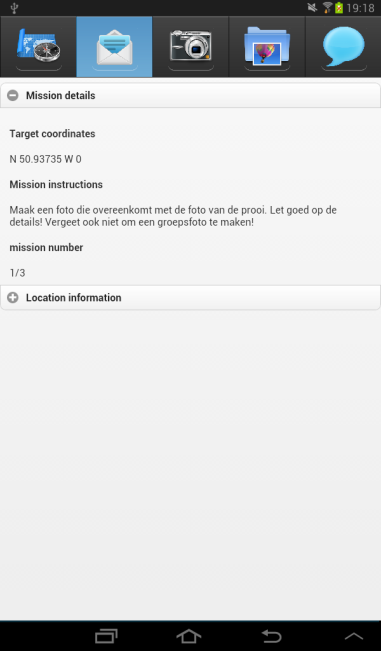


*Tabbar bij HTML5*

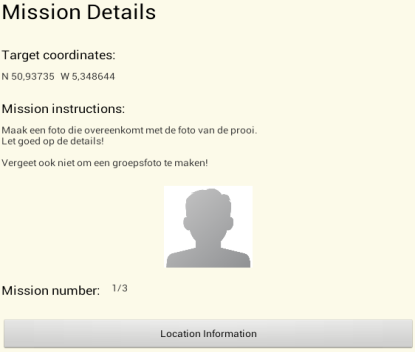


*Tabbar bij Android*

Een ander voorbeeld van de verschillen in lay-out is de accordeon. In de mockups werd een accordeon gebruikt om de details van de missie te bekijken. Dit was echter niet zo simpel om te maken in Monotouch. Vermits PhoneGap werkt met HTML5 en we weten dat HTML5 gemakkelijk een accordeon kan maken, heeft HTML5 dit wel kunnen implementeren. In plaats van een accordeon, hebben wij beslist om hiervoor verschillende schermen te maken.



*Accordeon bij HTML5*

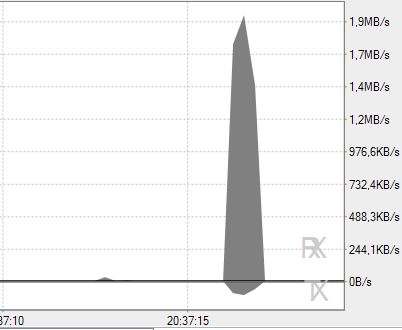
**

*Oplossing voor accordeon bij Android*

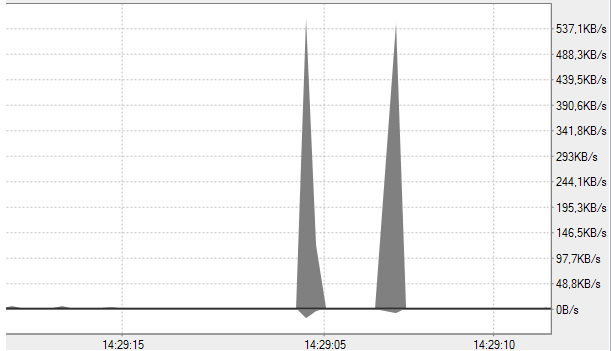
Op vlak van performantie is er een zeer groot verschil tussen Android en HTML5. Vermits we werken met PhoneGap is er geen zekerheid om te vermelden dat de performantie bij HTML5 veel lager ligt. De lagere performantie kan ook gewijd worden aan het feit dat PhoneGap hiertussen staat.

Via de tool van Android, genaamd DDMS, konden wij vrij beperkt performantietesten uitvoeren. Wij hebben hierdoor echter wel significante informatie verkregen dat een verschil maakt bij de keuze tussen PhoneGap en Android.

Allereerst werd het netwerkverkeer bij beide applicaties getest. Deze werden gemeten op het moment dat we de map uitzoomden. Ze werden beiden even ver uitgezoomd. Jammer genoeg herschaalt DDMS de grafiek zelf, waardoor je ook naar de cijfers moet kijken. Het resultaat spreekt echter voor zich.



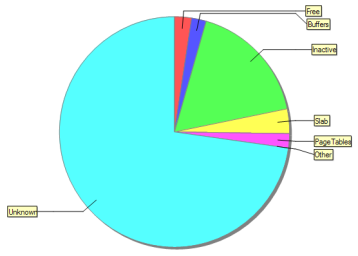
*Netwerkverkeer bij HTML5*



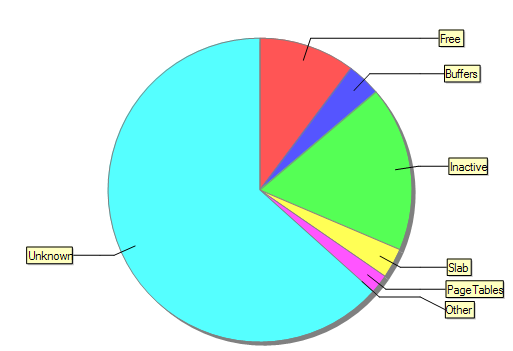
*Netwerkverkeer bij Android*

Na de meting van het netwerkverkeer werd het geheugenverbruik gemeten. Het moment van de meting werd bewust gekozen, namelijk net nadat de gebruiker zich heeft aangemeld. Op dat moment wordt zowel de locatie opgevraagd via GPS, als locaties gedownload, mapview geladen en informatie over de missie opgevraagd bij de webservice. Door op dat bepaald moment te meten, kunnen wij veronderstellen dat dit het maximumverbruik is doorheen de hele applicatie.

Zoals je kan zien op onderstaande afbeeldingen, is het vrije geheugen (rood) significant verminderd bij HTML5 in vergelijking met Android. Hierdoor kan ook de lage performantie gewijd worden.

**

*Geheugenverbruik bij HTML5*

**

*Geheugenverbruik bij Android*

# HTML5 vs. Windows Phone

De Windows Phone is nog relatief nieuw en heeft het bekende .NET framework ter beschikking. Hierdoor komen zeer veel elementen ons bekend voor. Voor dit project gaan we echter de native applicatie vergelijken met de hybride versie, gemaakt in html.

De hybride versie voor Windows Phone had echter de laagste prioriteit en de sprint die wel gemaakt was, konden we niet testen wegens een bug in PhoneGap (debuggen op het toestel ging niet omdat de Zune software, die nodig is voor de deployment, een fout gaf voor de PhoneGap applicatie).

Om toch een vergelijkende studie te geven, ga ik de grootste verschillen tussen Windows Phone en Monotouch geven. Omdat Mono ook werkt in de .NET omgeving, zijn dit er niet veel. De twee belangrijkste zijn de volgende.

Windows Phone (Microsoft) probeert dynamischer te werken door asynchrone methoden te gebruiken. Dit voordeel is uitgebuit bij de aanspreking van de webservice.

Ten laatste is er de opbouw van het design. Mono laat het design maken in de programeeromgeving van het gekozen platform (Xcode werkt met storyboards en Android met een xml bestand).

Monotouch werkt dus niet volledig in het .NET framework, enkel in de achterliggende code, en hier roept het de lokale code op, waardoor het niet makkelijk uit te wisselen is.

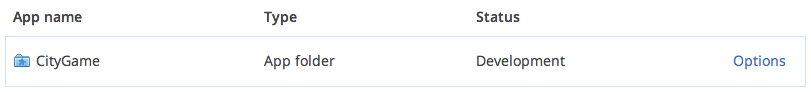
# Webservice

De web service behoort niet echt tot het onderzoek om hybride oplossingen te vergelijken met native oplossingen. Toch zijn er hier enkele problemen opgetreden.

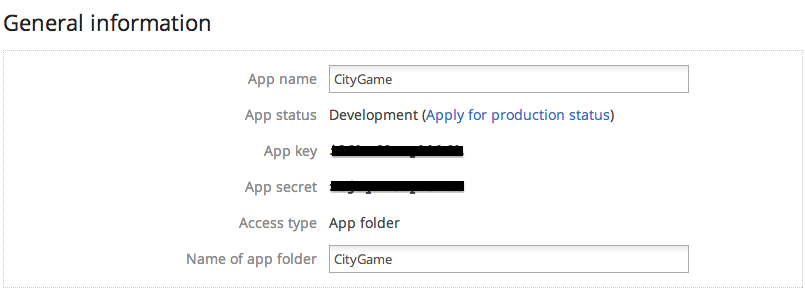
Veel van deze problemen zijn voortgekomen uit stommiteiten tijdens het programmeren. Het enige grote probleem dat zich bij de web service is opgetreden is bij het uploaden van een foto naar Dropbox.

Om deze feature te implementeren hebben we gebruik gemaakt van de Dropbox API.

Hievoor moeten we eerst op de Dropbox site een Dropbox App aanmaken.



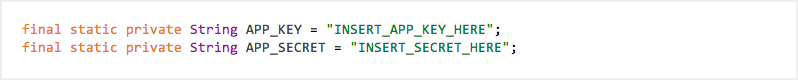
Bij het aanmaken van deze app, krijgen we een “App key” en een “App secret” die nodig zijn om te kunnen authentiseren van de gebruiker wanneer deze verbinding wilt maken met Dropbox.



Vervolgens moet op <https://www.dropbox.com/developers/core/sdk> de juiste SDK gedownload worden. In ons geval is dit de Java-versie en niet de Android-versie.

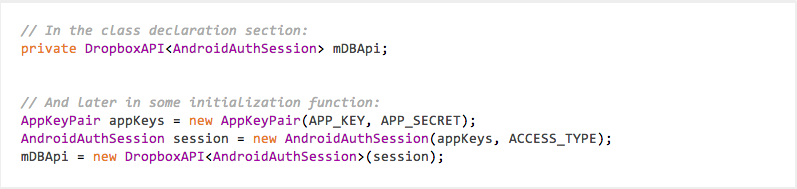
Op Dropbox staat geen documentatie over hoe de Java-versie geïmplementeerd moet worden, enkel Androi, iOS, Pyhton en Ruby. Omdat Android geprogrammeerd wordt in Java, gaan we dus deze documentatie gebruiken.

Volgens deze documentatie moet men beginnen met een variabelen aan te maken voor de “App key”, “App secret” en het access type.





Vervolgens wordt een Dropbox sessie aangemaakt.



Hier moet enkel “AndroidAuthSession” aangepast worden naar “WebAuthSession”.

Deze manier is echter niet volledig.

Volgens verschillende bronnen zou voor een Java applicatie onderstaande code gebruikt moeten worden:





Het probleem hierbij is dat er eerst naar een URL gegaan moet worden om daar een gebruikersnaam en wachtwoord in te geven.

Wanneer de web service verbinding probeert te maken, moet dit dus automatische gebeuren. Dit is hetgene wat we niet aan het werken hebben gekregen.