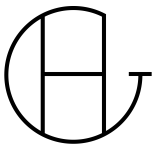


HoGent

Faculteit Bedrijf en Organisatie

# Vergelijkende studie en proof-of-concept in functie van een doelbewuste keuze tussen een cross platform mobiele applicatie en een mobiele website

Yannick Van Hecke

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Koen Hoof
Co-promotor:
Jeroen Gevenois

Instelling: Politiezone Gent - Dienst ICT

Academiejaar: 2016-2017

Tweede examenperiode

Faculteit Bedrijf en Organisatie

# Vergelijkende studie en proof-of-concept in functie van een doelbewuste keuze tussen een cross platform mobiele applicatie en een mobiele website

Yannick Van Hecke

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Koen Hoof
Co-promotor:
Jeroen Gevenois

Instelling: Politiezone Gent - Dienst ICT

Academiejaar: 2016-2017

Tweede examenperiode

# Samenvatting

# Voorwoord

# Inhoudsopgave

# Hoofdstuk 1

# Inleiding

## 1.1 Stand van zaken

In de vakliteratuur zijn er vandaag diverse artikelen te vinden over een vergelijkende studie tussen mobiele applicaties en responsive webapplicaties. De meeste van deze onderzoeken resulteren vaak een beperkte lijst van kenmerken die aangeven wanneer men het best voor welke optie kiest. Dit zonder dat hier wetenschappelijk onderzoek aan vooraf ging. Toch zijn deze wetenschappelijke artikelen op het internet beschikbaar. Zo onderzocht Paulo R. M. de Andrade (2015) in het artikel 'CROSS PLATFORM APP A COMPARATIVE STUDY' de voornaamste reden om te kiezen voor enerzijds de mobiele applicatie en anderzijds de responsive website.

De belangrijkste redenen uit het onderzoek om te kiezen voor een cross-platform mobiele applicatie zijn de volgende:

- Performantie
- User Interfaces in de lijn van het besturingssysteem
- Distributie via de app stores
- Push notificaties

In de vakliteratuur haalt men voor de hoofdzakelijkste redenen om te kiezen voor een responsive webapplicatie, de volgende aan:

- Eenvoudige ondersteuning voor meerdere systemen
- Het publiceren van de toepassing

## 1.2 Probleemstelling en Onderzoeksvragen

Momenteel wordt er bij de ICT-afdeling van de Gentse Politie aan verschillende projecten gewerkt waarbij het mobiel beschikbaar maken van interne gegevens een vereiste is. Om dit naar een gebruiksklaar eindproduct voor een mobiel apparaat om te zetten, bestaan er vandaag de twee opties.

Enerzijds kan er gekozen worden voor een cross-platform mobiele applicatie. Dit houdt in dat men binnen 1 ontwikkelomgeving een applicatie ontwikkelt die bruikbaar is op de 3 grote mobiele platformen: Android van Google, iOS van Apple en windowsphone of Windows Universal Platform van Microsoft. Hierbij wordt er code gedeeld tussen de 3 platformen.

Het weergeven van de data gebeurt op een platform-specifieke manier. Dit omdat men binnen de verschillende mobiele besturingssystemen werkt met andere UI elementen om de data in te tonen.

Anderzijds bestaat de mogelijkheid om te kiezen voor een responsive webapplicatie. Dit wil zeggen dat de inhoud van de webapplicatie aangepast wordt, naargelang het scherm waarbij men de mobiele website bekijkt. Zo wordt er op een smartphone mogelijks minder informatie weergegeven dan indien men de website bekijk op een computer of wordt in het geval van de smartphone de positie en afmetingen van de elementen aangepast.

In dit werk wordt onderzocht in welke gevallen de webapplicatie de beste optie is alsook in welke gevallen de mobiele applicatie een beter alternatief vormt.

Hierbij zullen volgende onderzoeksvragen beantwoord worden:

- Bij welke vorm van applicatie wordt het snelst resultaat behaald?
- Welke optie is het efficiënst?
  - Welke optie verbruikt de meeste hoeveelheid gegevens?
  - Welke optie geeft de eindgebruiker het snelst de gevraagd informatie?
- Welke optie kan het snelst ontwikkeld worden?

## 1.3 Opzet van deze bachelorproef

In functie van de snelheid van de ontwikkeling van beide producten, moet er een ontwikkelomgeving gekozen worden. Om een cross-platform mobiele applicatie te ontwikkelen, bestaat de keuze uit 2 technologieën . Enerzijds zijn er ontwikkelomgevingen met C# als achterliggende programmeertaal. Hierbij wordt de UI gedefinieërd in xml voor Android, xaml van windowsphone en Universal Windows Platform en storyboards voor iOS. Hierbij wordt de oorspronkelijke interactie en structuur van de UI en de logica van het mobiele besturingssysteem behouden. Visual Studio en Xamarin zijn 2 voorbeelden die gebaseerd zijn op deze technologie. Anderzijds kan men ook opteren voor een cross-platforme mobiele applicatie te bouwen op basis van html, css en javascript. Hierbij wordt er eerst een webservice van het te hosten project gemaakt, om vervolgens deze webservice op te starten in een emulator. De ontwikkelomgeving PhoneGap is op deze manier van werken gebaseerd (Adobe, 2017).

In dit onderzoek zal gekozen worden voor de eerste manier van werken, nl. de ontwikkelomgeving met C# als achterliggende programmeertaal. Dit vanwege de huidige kennis van het ontwikkelen van cross-platforme mobiele applicatie.

Nadat de gekozen ontwikkelingomgeving vastligt, wordt er aan de hand van de functionele requirements beslist welke de minimale versies van de besturingssystemen zijn. Hierbij wordt getracht het aantal ondersteunende apparaten zo hoog mogelijk te houden. Tijdens het onderzoek wordt, zoals in de onderzoeksvragen reeds vermeld, ook rekening gehouden met de tijd en kosten die de ontwikkeling en publiek stellen van beide applicaties met zich meebrengen.

Vervolgens wordt de gebruikte methodologie toegelicht, waarna de casus uitgelegd wordt. Deze casus is de basis die zal gebruikt wordt in het onderzoek naar efficiëntie van de cross-platforme mobiele applicatie en de responsive webapplicatie.

Nadat de casus duidelijk uiteengezet is, worden de gemaakte applicaties verduidelijkt aan de hand van de geschreven code. Vervolgens worden de resultaten getoond en wordt de conclusie die uit het onderzoek voortvloeit, medegedeeld. Deze conclusie wordt ook gebruikt door de Gentse politie in hun keuze tussen een mobiele applicatie of een responsive website.

Na de conclusie worden er nog richtlijnen voor toekomstig onderzoek gegeven in deze materie.

# Hoofdstuk 2

# Methodologie

## 2.1 Literatuuronderzoek

Ter voorbereiding van deze vergelijkende studie is er een literatuuronderzoek verricht. Hierbij in dat er op internet gezocht werd naar verschillende wetenschappelijke artikelen over voorgaande vergelijkende studies tussen cross platforme mobiele applicatie en responsive website. Dit literatuuronderzoek heeft al voornaamste reden het geven van een stand van zaken in het ontwikkelen van cross-platforme mobiele applicaties en responsive webapplicaties.

## 2.2 Vastleggen casus

De eerste stap in het onderzoek naar de vergelijkende studie tussen een cross-platforme mobiele applicatie en een responsive website, was het vastleggen van de casus. Dit bepaalde de functionele requirements van beide applicaties en gebeurde in overleg met de co-promotor. De casus omvatte een toepassing rond personeelsgegevens binnen de Gentse Politie, waarbij de gegevens momenteel enkel intern beschikbaar zijn. Het doel van de casus is om deze, mits de nodige beveiliging, beschikbaar te stellen van het personeel dat niet altijd met het intern netwerk verbonden is. Hierbij wordt er vooral gedacht aan operationele mensen.

## 2.3 Framework kiezen voor cross platform mobiele applicatie

Omdat vanuit de stage reeds gekozen werd voor de ontwikkelomgeving van Microsoft en dit hierdoor reeds een vertrouwde manier van werken was, was de keuze om de cross platform mobiele applicatie te ontwikkelen met Visual Studio 2015 een evidentie.

Hierbij wordt er gebruik gemaakt van een apart deelproject binnen de applicatie waarin code gemeenschappelijk wordt gemaakt voor de 3 platform-specifieke applicaties. Op deze manier kan men code hergebruiken, waardoor de efficiëntie van de geschreven code stijgt.

## 2.4 Framework kiezen voor responsive webapplicatie

Het framework, dat voor de responsive webapplicatie gebruikt werd, is het ASP MVC 5-framework van Microsoft. Hierdoor kan de ontwikkeling van de webapplicatie ook in Visual Studio gebeuren, hetgeen voor de vergelijking van de ontwikkeltijd een eerlijker resultaat opleverde, dan wanneer men zich nog dient in te werken in een nieuwe technologie en ontwikkelomgeving.

## 2.5 Opzetten REST API

Het opzetten van de REST API was de volgende stap in het onderzoeksproces. Aanvankelijk werd ervoor gekozen om eerst de basisfunctionaliteiten te implementeren en pas nadien de authenticatie toe te voegen. Dit zorgt ervoor dat de data niet ongeoorloofd beschikbaar is voor derden. Hierbij dient men zich eerst te authenticeren aan de hand van de gebruikersnaam en het wachtwoord van het gebruikersaccount binnen de Politiezone Gent. Indien men zich succesvol geauthenticeerd heeft bij het domein aan de hand van de Active Directory, krijgt men van de REST API een token terug. De geldigheid van deze token kan men vrij kiezen. Deze token is verplicht bij te voegen bij alle requests die verstuurd worden naar de server. Het testen van de REST API zal eerst gebeuren aan POSTMAN, een REST-client. Zo is men zeker dat de REST API correct werkt, alvorens over te gaan te het ontwikkelen van de mobiele applicatie.

## 2.6 De ontwikkeling van een cross platforme mobiele applicatie

Nadat het testen van de REST API voltooid is, begint de ontwikkeling van de cross platforme applicatie. Hierbij is het de bedoeling om de code die gemeenschappelijk is voor de 3 deelprojecten, nl. het ophalen van gegevens uit de webservice en deze op te slaan in een lokale databank, alsook het mappen naar objecten van deze gegevens af te zonderen van de platform-specifieke deelprojecten in een gemeenschappelijk deelproject. Op deze manier wordt de ontwikkeltijd gereduceerd ten opzichte van het ontwikkelen van 3 afzonderlijke mobiele applicaties. Het weergeven van de opgehaalde data wordt, samen met het ontwerp van de schermen voor de platforme-specifieke user

interfaces, staat in de platform-specifieke projecten gecodeerd. Dit omdat er geen uniforme manier bestaat om de data weer te geven voor de 3 besturingssystemen binnen de gekozen ontwikkelomgeving.

## 2.7 De Ontwikkeling van een responsive webapplicatie

De volgende stap in deze vergelijkende studie tussen een cross platforme mobiele applicatie en responsive webapplicatie, is het ontwikkelen van de responsive webapplicatie. Hierbij wordt dezelfde databank gebruikt van de webservice van de cross platforme mobiele applicatie. Verder wordt ook het domeinmodel van de webservice en de mobiele applicatie overgenomen. Hierna worden de controllers geïmplementeerd met authenticatie en authorisatie aan de hand van de REST API. Dit om vervolgens het responsive design van de webpagina's af te werken.

## 2.8 Vergelijken van de tijd die nodig is om beide vormen van toepassingen te ontwikkelen

Wanneer beide vormen van de toepassing ontwikkeld zijn, zal er gekeken worden naar tijd die nodig was om deze tot stand te brengen. Dit is overigens een onderzoeksvraag van de casus en zal gebeuren aan de hand van een versiebeheersysteem. Hierbij zal de tijd tussen het verschil tussen de eerste en de laatste commit van ieder project berekend worden.

## 2.9 Meten van het gegevensverbruik

De verbruikte hoeveelheid gegevens wordt gemeten door het opvragen van de grootte van het antwoord dat de toepassing binnenkrijgt van de webservice tijdens het uitvoeren van beide applicaties. Vooraf wordt bepaald hoeveel keer dit gebeurt. Nadien wordt dit proces herhaald bij de responsive website. Indien alle nodige gegevens verzameld zijn, wordt een conclusie getrokken over welke vorm van applicatie het meest zuinig omgaat met het internetgebruik.

## 2.10 Meten van de snelheid

De snelheid van beide vormen van applicatie is een criterium waarop men beoordeelt tussen de keuze tussen de app en de website. Men wil namelijk zo snel mogelijk de

gevraagde gegevens ter beschikking hebben. Met de ingebouwde stopwatch in het .NET-framework wordt dit gemeten. Nadien worden deze resultaten meegenomen in de besluitvorming.

## 2.11 Extra kosten bij de realisatie en publicatie

Om te kiezen voor de cross-platforme mobiele applicatie of de responsive website, dient rekening gehouden te worden met de kosten die nodig zijn voor de realisatie en publicatie. Hierbij wordt hoofdzakelijk gedacht aan de kosten die de publicatie in de app stores met zich meebrengt, alsook de kosten voor de hosting van enerzijds de webservice en anderzijds de webapplicatie. Deze kosten zijn online beschikbaar.

## 2.12 Besluitvorming

Eens alle bovenvermelde gegevens bekend zijn, worden deze met elkaar vergeleken om zo tot een gefundeerde lijst van argumenten te komen. Deze lijst zal vermelden in welke gevallen de cross-platforme mobiele applicatie de betere oplossing voor de klant is. Overigens wordt er een lijst met karakteristieken voor de responsive webapplicatie bekend gemaakt.

# Hoofdstuk 3

# Framework cross-platform mobiele applicatie

## 3.1 Twee grote categorieën

Ter voorbereiding van het vergelijkende onderzoek tussen de cross-platform mobiele applicatie en de responsive website, is er voor beide mogelijke manier gezocht om deze vormen van applicaties te ontwikkelen. Voor de cross-platforme mobiele applicatie zijn deze mogelijke vormen een methode die zeer sterk aansluit bij het ontwikkelen van html-website. Hierbij wordt de html code in combinatie met css voor de opmaak en JavaScript voor de logica door de compiler omgezet naar een cross- plaforme mobiele applicatie. Op deze laatste manier wordt er gedurende dit werk niet verder op ingegaan.

## 3.2 Architectuur van de cross-platform mobiele applicatie

Voor de opbouw van de cross-platform mobiele applicatie is geopteerd om te kiezen voor een architectuur waarbij men zoveel mogelijk code gemeenschappelijk kan hergebruiken. Concreet houdt dit in de men in het Shared-project zowel de definitie van de domeinklassen, als de code die verantwoordelijk is voor het ophalen van de data uit de REST-service en de mapping definieërt.

## 3.3 Voordelen van de gekozen architectuur

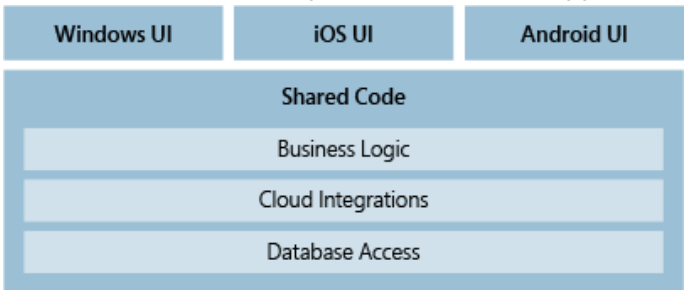
### 3.3.1 Ontwikkeling en project structuur

De grote voordelen van de architectuur van de cross-platform mobiele applicatie zijn: Specifieke deelprojecten voor de afzonderlijke mobiele besturingssystemen en het gemeenschappelijk stellen van bepaalde delen van de programmacode. Dit biedt voor de ontwikkelaar als pluspunt dat binnen de gekozen ontwikkelomgeving, nl. Visual Studio, dat de oorspronkelijke project-structuur behouden blijft.

Deze manier van werken vereenvoudigt de overstap van de oorspronkelijke ontwikkelingomgevingen, nl. Android Studio voor Android en xcode voor iOS, naar Visual Studio. Ook de syntax van de code van de deelprojecten werd hierbij overgenomen en vertaald naar C#.

De schematische voorstelling van de architectuur van de cross-platform mobiele applicatie ziet er schematisch als volgt uit:

Figuur 3.1: Architectuur van de cross-platforme mobiele applicatie Holmes (2017)




### 3.3.2 User Interface

Voor het definiëren van de user interface heeft Microsoft besloten om de bestaande manier van werken over te nemen van de oorspronkelijke platformen, nl. Android, iOS en Universal Windows Platform. Dit brengt voor de ontwikkelaar zowel een voor- als nadeel met zich mee. Het heeft als voordeel dat, indien men reeds mobiele applicatie voor Android, iOS en windowsphone kan ontwikkelen, men geen nieuwe opmaaktaal voor de user interface dient aan te leren. Het grote nadeel van deze manier is dat men 3 keer een UI moet definiëren, dit moet behorende code-behind voor het opvangen van de events uit de UI.

## 3.4 Nadelen van de gekozen architectuur

### 3.4.1 User Interface

Zoals eerder vermeld, brengt het gebruik van de standaard-methodologie van de 3 mobiele platformen een grotere hoeveelheid werk voor de ontwikkelaar met zich mee. Dit omdat Android, iOS en windowsphone elk hun eigen manier van UI te definiëren en deze niet hergebruikt kan worden.

Voor het opvangen van de events van de UI kan men wel opteren om in de architectuur een facade [1] te voorzien, waarnaar de oproepen van events uit de user interface gedelegeerd worden naar de facade.

Op deze manier beperkt de code in de klassen die de events van de user interface opvangen tot het strikt noodzakelijke.

### 3.4.2 Kosten

De ontwikkeling van een cross-platform mobiele applicatie is een project waaraan de nodige kosten verbonden zijn. Hieronder volgt een oplijsten van de kosten per fase in het softwareontwikkelingsproces.

#### Kosten voor de ontwikkeling

Reeds bij de ontwikkeling van de cross-platform mobiele applicatie zijn er mogelijks kosten. Zo dient men een licentie op een ontwikkelomgeving aan de schaffen. Indien men de ontwikkelomgeving kiest die voor dit onderzoek gebruikt werd, nl. Visual Studio 2015 Professional, komt dit momenteel 499 dollar.

#### Kosten voor testen

Voor een applicatie in de applicatiewinkel van een mobiel besturingssysteem terecht komt, dient deze eerst uitvoerig getest te worden. Op deze manier worden de mogelijke fouten uit de software gehaald. Fouten in de applicatie kunnen ervoor zorgen dat de applicatie onverwacht beeïndigd wordt of vastloopt. Dit beeïndigen of vastlopen kan tot gevolg hebben dat de applicatie slechte reviews krijgt in de applicatiewinkels en bijgevolg minder downloads en inkomsten voor de ontwikkelaar oplevert.

[1] Fasade Design Pattern dofactory.com (2017)

Indien men ervoor kiest om voor de drie grote platformen te ontwikkelen, heeft men naast de kost van de licentie voor Visual Studio of een ander ontwikkelomgeving, ook de kost van een computer waar het besturingssysteem macOS op geïnstalleerd is in combinatie met Xcode, de ontwikkelomgeving van Apple. Xcode is een IDE [2] waarin men applicatie kan ontwikkelen voor iOS, macOS, tvOS en watchOS, de besturingssytemen voor respectievelijk iPhone en iPad, iMac en de macbook in alle versie, de Apple TV en de Apple Watch. De ontwikkelomgeving is gratis beschikbaar de App Store in macOS (xco).

### Kosten voor publicatie

Men heeft echter wel nog kennis nodig heeft van Android, iOS en windowsphone om deze applicatie te kunnen ontwikkelen. Zo moeten er nog steeds 3 applicaties ontwikkeld worden, hetzij wel in een afgeslankte versie. Dit in tegenstelling tot de responsive webapplicatie waarbij men slechts 1 keer de gebruikersinterface dient te definiëren.

Verder dient ook men de applicatie voor elk mobiel besturingssysteem te publiceren in de appstores van de verschillende mobiele besturingssystemen. Hieraan zijn eenmalige of jaarlijkse kosten aan verbonden. Zo dient men voor de publicatie in de App Store van Apple zich eerst inschrijven voor het Apple Developer Program (Inc, 2017) . Deze inschrijving kost 99 Amerikaanse dollar per jaar. Verder kunnen bedrijven of organisaties zich hier ook bij aansluiten, om zo meerdere ontwikkelaars aan te laten sluiten op 1 ontwikkelaarsaccount voor de App Store. Dit verlaagd de kosten.

Voor Android dient men bij de publicatie van de app (Google, 2017) eenmalig 25 Amerikaanse dollar betalen voor de inschrijving. Verder kan men aan de hand van Developer Console het aantal downloads van de app, het aantal gerapporteerde crashes en de gemiddelde rating die de gebruikers gaven bekijken. Aan de hand van de recensies die de gebruikers op de app, kan de ontwikkelaar de feedback met eventuele verbeterpunten bekijken.

Indien men de applicatie wenst te publiceren in de 'Microsoft Marcketplaces' van Microsoft (2017d) zoals de Windows Store, Office Store, Azure Marketplaces en nog meer aan te kondigen winkels, dient men voor een individueel account eenmalig ongeveer 19 Amerikaarse dollar te betalen aan Microsoft. Voor bedrijfsaccounts komt deze kost neer op +/- 99 dollar.

[2] Integrated Develop Environment

# Hoofdstuk 4

# Ontwikkeling cross platforme mobiele applicatie

## 4.1 Analyse van het project

Als start van het project werd er een analyse gemaakt door de co-promotor van de functionele requirements.

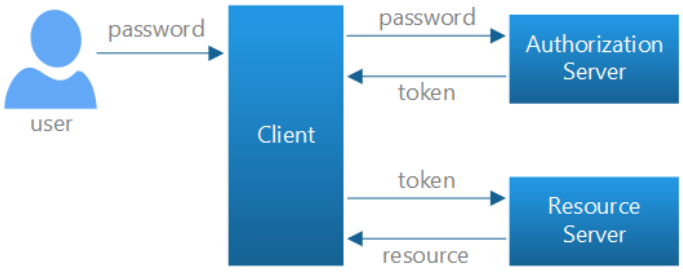
Deze analyse bestaat uit het sturen van een query naar de databank die draait in een webservice. Op deze manier worden de gegevens omgezet naar JSON [1], een makkelijk genereerbaar gegevensformaat dat eenvoudig te lezen is voor toepassingen en vaak gebruikt wordt in webservice. Het grote voordeel van JSON is dat het makkelijk leesbaar/te genereren is voor/door toepassingen.

## 4.2 Ontwikkeling van de ASP.NET MVC Web api

De ASP.NET MVC Web api heeft in deze toepassing 2 functies. De REST API heeft als functie gebruikers zich te laten authenticeren bij het Active Directory domein van de Gentse Politie. Nadien levert de REST API de gegevens die binnen de applicatie nodig zijn voor de gebruikers. Deze authenticatie gebeurt aan de hand van tokens die een bepaalde tijd geldig zijn (in deze casus een week). Nadien moet de gebruiker zich opnieuw authenticeren aan de hand van de inloggegevens. Dit om opnieuw een token met geldigheid van 1 week te verkrijgen. De tokens zijn verplicht toe te voegen aan de request. Indien dit niet gebeurt, stuurt de server een antwoord dat men geen toegang heeft tot de aangevraagde resources. De schematische voorstelling van deze techniek, kan u terugvinden op de volgende pagina.

[1] De afkorting staat inl staat voor JavaScript Object Notation

Figuur 4.1: Token-authenticatie met ASP.NET MVC Web api




## 4.3 Ontwikkeling van de cross platforme mobiele applicatie

### 4.3.1 Gedeelde code

Om de applicaties zo efficiënt mogelijk te ontwikkelen, wordt er code gedeeld tussen de deelprojecten van de 3 mobiele platformen. Concreet gaat dit over de code die nodig is voor het ophalen van de gegevens uit de webservice gemeenschappelijk gesteld wordt voor zowel het Android-, als het iOS- en Windowsphone-project. Ook het domeinmodel wordt gemeenschappelijk gedefinieërd voor de drie mobiele applicaties. In het project gebeurd dit door in ieder deelproject een referentie toe te voegen naar het gemeenschappelijke project.

#### Code om data op te halen uit de REST-api

De code die absoluut gedeeld wordt wegens efficientiëredenen, is de code om de data op te halen uit de REST-service. Hieronder vindt u het voorbeeld dat als basis diende voor de ontwikkeling van de cross platforme mobiele applicatie. De volledige code met uitgebereide stappen om dit voorbeeld te ontwikkelen is beschikbaar op (Microsoft, 2017b).

Ter voorbereiding van de logica om data uit een REST-api op te halen hebben we een klasse nodig die de entiteit uit het domein van de toepassing voorziet. In dit geval gaat het om de klasse Persoon met onder andere de velden Voornaam, Familienaam, Geslacht.

```
namespace MobileApp
{
  public class Persoon
  {
    public string Voornaam {get; set;}
    public string Familienaam {get; set;}
    public string Geslacht {get; set;}
  }
}
```

Nu deze domeinklasse gedefinieërd is, kunnen we verdergaan met een klasse aan te maken voor het effectief ophalen van gegevens uit de REST-service. Hierbij is het belangrijk om op te merken dat deze code asynchroon op de main-thread van de applicatie wordt uitgevoerd. Voorlopig ziet de klasse er als volgt uit:

```
using System.Threading.Tasks;
using Newtonsoft.Json;
using System.Net.Http;
namespace MobileApp
{
  public class DataService
  {
    public async Task<string> getData(string
       queryString)
    {
      // Implementatie DataService.getData(string
         queryString)
    }
  }
}
```

Nu we de methode gedefinieërd hebben, kunnen we deze verder implementeren. Deze implementatie zal bestaan uit 2 delen. In eerste instantie wordt de data opgehaald uit de REST-api, nadien zal de opgehaald data omgevormd worden tot objecten van het type "Persoon". Dit type is hierboven reeds gedefinieërd.

De volgende code die we toevoegen aan de klasse "DataServiceönder // Implementatie DataService.getData(string queryString)", is de code om de data als JSON binnen te halen. Hiervoor moet er eerst een instantie van de klasse HttpClient (uit de namespace System.Net.Http) aangemaakt worden.

```
HttpClient client = new HttpClient();
```

Nu de instantie van HttpClient geïnstanceerd is, kan deze gebruikt worden om data op te halen bij de webservice. Het ophalen van de data gebeurt aan de hand van onderstaande code:

```
namespace MobileApp
{
  // Imports statements

  public class DataService
  {
    public async List<Person> getData(string
       queryString)
    {
      // een instantie van de klasse HttpClient
         aanmaken
      HttpClient client = new HttpClient();

      // data ophalen uit de webservice
      HttpResponseMessage response = await
         client.GetAsync(queryString);

      // data converteren naar persoon-objecten
      List<Person> personen =
         JsonConvert.DeserializeObject<List<Person>>(response);

      // persoon-objecten teruggeven
      return personen;
    }
  }
}
```

### 4.3.2 Platform specifieke code

De deelprojecten voor de drie mobiele applicaties bestaat enerzijds uit de definitie van de schermen voor de mobiele applicaties. Deze definitie gebeurt in xml voor Android, xaml voor windowsphone en storyboard voor iOS.

Anderzijds is ook de code die de gebruikersevent van de GUI opvangen. In deze code wordt er vervolgens een request gestuurd naar de api voor authenticatie en data.

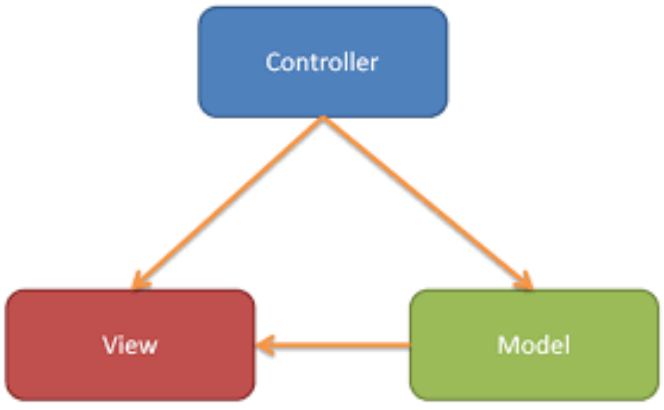
# Hoofdstuk 5

# Framework responsive website

## 5.1 Architectuur van de responsive website

De architectuur van de responsive webapplicatie is het ASP.NET MVC 5 framework van Microsoft (2017a). MVC [1] is een combinatie van design patterns waarbij de business-logica, de user interface en de controllers de belangrijkste onderdelen zijn. De business-logica bestaat uit domein-objecten die op hun beurt de reële situatie waarbinnen de applicatie gebruikt wordt, weerspiegelt. Het voornaamste doel van het MVC-framework is een scheiding voorzien tussen het grafische onderdeel van de toepassing (de schermen), de business logica of model en de controllers die instaan voor het delegeren van gegevens tussen de views en het model. Schematisch ziet het MVC-framework er als volgt uit:

Figuur 5.1: Schematische voorstelling van het MVC-framework Microsoft (2017c)




[1] Model View Controller

## 5.2 Voordelen van de gekozen architectuur

Het ASP.NET MVC 5 framework van Microsoft (2017a) werd gekozen vanwege volgende voordelen:

- Het beheren van complexe applicatie wordt, door de opdeling in Model, View en Controller, eenvoudiger.
- Verbeterde ondersteuning voor Test Driven Development [2]
- Gecentraliseerde beheer van requests door middel van Front Controller Pattern.
- Goede werking voor webapplicaties met grote ontwikkelteams, waarbij de controle over het applicatiegedrag een hoge vereiste is.

## 5.3 Nadelen van de gekozen architectuur

De grootste nadelen van de ASP.NET MVC 5 architectuur zijn volgens Stormpath de volgende:

- Het moeilijk debuggen van de toepassing in vergelijking met een event-driven variant.
- Url's van een ASP.NET MVC 5 applicatie zijn eenvoudiger te begrijpen voor een zoek-robot.

## 5.4 Kosten

Hieronder volgt een overzicht van de voornaamstekosten per fase in de ontwikkeling van de responsive website.

### 5.4.1 Kosten voor de ontwikkeling

De ontwikkeling van de responsive website is een proces dat niet kostenloos verloopt. Zo moet er een licentie op de ontwikkelomgeving (in dit geval Visual Studio) gekocht worden. De kosten van deze licentie is afhankelijk van de gekozen. Zo kost een enkele licentie op Visual Studio Professional 2017 499 dollar.

[2] Test Driven Development is een ontwikkelmethodologie waarbij voorafgaande aan de code eerst de testen geschreven worden. Hierdoor zijn de testen gebaseerd op de oorspronkelijke analyse in plaats van op de geschreven code

### 5.4.2 Kosten voor testen

In de testfase zijn er geen kosten verbonden aan de responsive website, aangezien Visual Studio zelf een lokale webserver opzet voor de hosting van de website. Deze lokale webserver is te bereiken via http://localhost:poortnummer met als poortnummer een 5-cijferig nummer dat vast gekozen is door de ontwikkelomgeving voor de toepassing.

### 5.4.3 Kosten voor publicatie

Voor de publicatie van de responsive website zijn er volgende optie voor handen:

- Men kan kiezen om te hosten op de lokale computer.
- Men kan ervoor kiezen om de webapplicatie op een eigen webserver te hosten.
- Men kan een externe webserver gebruiken zoals Microsoft Azure.

In volgende subsecties worden de kosten zoals electriciteit en koeling van de server buiten beschouwing gelaten omdat deze mogelijks te grote verschillen opleveren om objectief te kunnen oordelen over de drie publicatiemethoden. Tevens wordt er vanuit gegaan dat de kosten voor publicatie van de Web api en responsive website gelijk zijn.

#### Kosten bij hosting op lokale computer

De eerste manier van publicatie is het hosten van de webapplicatie op de lokale computer waarop men de responsive website ontwikkelt. Aan deze manier van publiceren zijn er geen bijkomende kosten verbonden. Het voornaamste nadeel is dat de computer die de website host, permanent aan dient te staan om de beschikbaarheid te garanderen. Verder is de kost van het installeren van een webservice minimaal, aangezien die functionaliteit reeds ingebouwd is in de ontwikkelomgeving.

### Kosten bij hosting op eigen webserver

Indien men ervoor kiest om de responsive website op een eigen webserver te hosten, dient men enkel de kosten voor de aankoop en installatie van de webserver in rekening te brengen. Dit omdat er verder geen kosten zijn voor de hosting van de webserver.

### Kosten bij hosting op externe webserver

Naast de mogelijkheid om een website zelf te hosten, kan dit ook uitbesteed worden naar een externe service. Hiervoor worden mogelijks wel kosten voor aangerekend. Bij wijze van voorbeeld vindt u de prijsberekening voor 1 maand Azure hieronder:

Uitgewerkte prijsberekening voor hosting op Azure

| Omschrijving | Region | Prijscategorie | Prijs (€) |
|---|---|---|---|
| SQL Database (Single Database) | West-Europa | Basis | 4,20 |
| API management | West-Europa | Standaard | 41,30 |
| Support | nvt | nvt | 1,33 |
| | | Totaal | 46,83 |

# Hoofdstuk 6

# Ontwikkeling responsive website

## 6.1 De domeinlaag

Als eerste stap in de ontwikkeling van de responsive website, is ervoor geopteerd om het business-logica te implementeren. Dit houdt in dat het opgebouwde domein uit de ASP.NET MVC Web api overgenomen werd in de responsive webapplicatie.

## 6.2 Gebruiken van de bestaande database

Eens de ontwikkeling van de responsive webapplicatie afgerond is, word deze gekoppeld aan een reeds bestaande databank van de Gentse Politie. Op deze manier is de reeds aanwezige data direct bruikbaar in de ontwikkelde applicatie.

Om dit tot een goed einde te brengen, wordt er in plaats van "Code-First"de ontwikkelstrategie "Database-First"toegepast in het project. Zo kan men de webapplicatie en de bestaande database makkelijker op elkaar afstemmen.

## 6.3 Controller voor het tonen van de views

De controllers hebben een dirigende rol binnen de toepassing, m.a.w. ze verwerken enkel aanvragen vanuit de views en vragen aan onderliggende klassen gegevens op uit de databank. Die gegevens worden later doorgegeven in de views, die op hun beurt de gegevens in een layout plaatsen om op een aangename manier weer te geven aan de gebruiker van de toepassing.

## 6.4 Koppeling met Asp Mvc Web api

Omdat ook in de responsive webapplicatie authenticatie en authorizatie aan de hand van tokens een vereiste is, dient er een koppeling te zijn tussen de responsive website enerzijds en anderzijds de web api, die tevens door de cross-platform mobiele applicatie gebruikt wordt. Deze koppeling is voorzien aan de server-side van de webapplicatie, die een api-request met de inloggegevens van de gebruiker verstuurd naar de web api van de mobiele applicatie. De access-token en vervaldatum worden opgeslagen door middel van een session. Bij elke request is het de bedoeling dat er wordt nagekeken aan de server-side of de token ingevuld is en of het tijdstip waarop de token vervalt niet in het verleden ligt. Indien dit het geval is, wordt de gebruiker omgeleid naar de inlogpagina, waar hij zich opnieuw dient te authenticeren aan de hand van de gebruikersnaam en wachtwoord uit de active directory.

## 6.5 Vereiste tijd om de toepassing te ontwikkelen

De ontwikkeling van de webapplicatie wordt berekend aan de hand van de tijd tussen de eerste en de laatste commit op het versiebeheersysteem. In dit geval gebeurde de eerste commit op 15 maart 2017. De laatste commit gebeurde op 23 maart 2017. De ontwikkeling van deze webapplicatie heeft 6 dagen in beslag genomen. Het uitgebereide overzicht van hoelang de ontwikkeling van elke feature in beslag nam, vindt u terug in bijlage B.

# Hoofdstuk 7

# Resultaten cross platforme mobiele applicatie

In dit hoofdstuk worden de resultaten van het onderzoek besproken, waarbij er 2 belangrijke aspecten besproken worden. Enerzijds wordt de performantie in verband met gegevensgebruik en snelheid besproken. Tevens komen ook de tijd die nodig is om de toepassingen te ontwikkelen en de mogelijke problemen tijdens de ontwikkeling aan bod in de hoofdstuk. Deze resultaten dienen overigens als basis om later een besluit onder welke voorwaarden wanneer men best kiest voor de cross-platforme mobiele applicatie en onder welke voorwaarden men beter kiest voor de responsive website.

## 7.1 Tijd die nodig is om de toepassing te ontwikkelen

Het eerste gegeven dat in de keuze tussen een cross platforme mobiele applicatie en responsive webapplicatie van belang is, is de tijd die nodig is om deze toepassing tot stand te brengen. Deze tijd wordt samengerekend met de tijd die nodig is om de REST-api te ontwikkelen. Dit aangezien de REST dient als basis voor de mobiele applicatie.

De eerste stap, in de ontwikkeling van de cross platforme mobiele applicatie, was het opzetten van REST-api. Deze REST-api heeft een twee-ledige functie, zoals reeds in het Hoofdstuk "Ontwikkeling cross platforme mobiele applicatie"vermeld staat. Hierbij werd eerst het beschikbaar maken van de data uit een achterliggende databank gerealiseerd. Nadien is de beveiliging aan de hand van authenticatie-token toegevoegd. De ontwikkeling van deze REST-api heeft 7 dagen in beslag genomen. De ontwikkeltijd van elk onderdeel in de toepassing kan u terugvinden in bijlage B.

## 7.2 Snelheid van de applicatie

De snelheid van de mobiele applicatie wordt gemeten aan de hand van een ingebouwde stopwatch. Hieronder volgt een samenvatting van de testresultaten. Overigens kan u de uitgebereide testresultaten terugvinden in Bijlage A.

De gemiddelde resultaten van de testen op een Android-toestel kan u hieronder terugvinden.

| Statistieken | Android | windowsphone |
|---|---|---|
| Gemiddelde | 6.552,9562 | 2361,3135 |
| Standaardafwijking | 2.657,2925 | 407,9001 |

## 7.3 Gegevensverbruik van de cross platforme mobiele applicatie

| Statistieken | Android | windowsphone |
|---|---|---|
| Gemiddelde | 190,00 | 207,00 |
| Standaarafwijking | 0,00 | 0,00 |

## 7.4 Mogelijke problemen bij de ontwikkeling

Tijdens de ontwikkeling van de cross platforme mobiele applicatie doken er af en toe ook enkele problemen op. Deze worden ook besproken omdat deze van belang zijn in de keuze tussen de mobiele applicatie en de webapplicatie.

Een eerste zaak die uit het onderzoek naar voor kwam, is dat men bij de ontwikkeling van een iOS-applicatie op windows moet beschikken over een verbinding tussen een windows-pc en een macOS-toestel. Deze verbinding is noodzakelijk voor het kunnen weergeven van het storyboard of de user interface in de ontwikkelomgeving en om de applicatie te kunnen debuggen. Naast het feit dat de structuur van de applicatie soms moeilijk te begrijpen is in vergelijking met de andere mobiele platformen (Android en Windowsphone), viel ook de verbinding soms weg. Dit maakt het uiteraard niet eenvoudig om de iOS-applicatie te ontwikkelen.

# Hoofdstuk 8

# Resultaten responsive website

## 8.1 Tijd die nodig is om de toepassing te ontwikkelen

Aangezien de tijd die nodig is om de toepassing te ontwikkelen bij de cross platforme mobiele applicatie gemeten wordt, is dit ook een vereiste voor de responsive webapplicatie. Dit om later een objectieve conclusie te kunnen trekken. Ook hierbij rekent ook de tijd voor de ontwikkeling van de REST api mee. Deze ontwikkeling nam 7 dagen in beslag.

Na de ontwikkeling van de REST api ging de ontwikkeling van de responsive webapplicatie van start. Dit duurde 7 dagen, wat de totale ontwikkelingstijd van de responsive website samen met de koppeling tussen de responsive webapplicatie en de REST api op 15 dagen brengt. De voornaamste redenen dat de ontwikkeling van de responsive webapplicatie sneller verliep in vergelijking met de cross platforme mobiele zijn de volgende:

- Betere kennis van en meer ervaring met ASP.NET MVC Webapplicatie.
- De uniforme manier van definiëren van de user interface.

## 8.2 Snelheid van de applicatie

Zoals reeds in de methodologie aangegeven, wordt de snelheid van de applicatie gemeten aan de hand van een stopwatch ingebouwd in C#. Deze stopwatch wordt gestart wanneer de aanvraag in de juiste controller en wordt gestopt wanneer men de html en css terugstuurt naar de browser.

Om een betere vergelijking te kunnen maken tussen de mobiele applicatie en de responsive webapplicatie, worden beide resultaten vermeld.

| Statistieken | Android | Windowsphone |
|---|---|---|
| Gemiddelde | 6.552,9562 | 2361,3135 |
| Standaardafwijking | 2.657,2925 | 407,9001 |

## 8.3 Gegevensverbruik van de responsive website

In de vergelijking tussen een mobiele applicatie en responsive website, is de hoeveelheid verbruikte data ook een parameter in de beslissing Het gegevensverbruik van de responsive website wordt gemeten door in de browser de opgehaalde hoeveelheid gegevens te bekijken. De hoeveelheid data (transferred) en de hoeveelheid gegevens van het DOM [1] wordt hierbij samengeteld.

| Statistieken | Android | windowsphone |
|---|---|---|
| Gemiddelde | 596,85 | 885,2 |
| Standaarafwijking | 19,2826 | 18,3004 |

## 8.4 Mogelijke problemen van de responsive website

De ontwikkeling van de responsive website verliep over het algemeen vlotter dan de ontwikkeling van de cross platforme mobiele applicatie. Dit is te verklaren door de betere kennis en meer ervaring in het ontwikkelen van responsive website. Verder werd er eerst zonder de REST api gewerkt, hetgeen de tijd die nodig is om de ontwikkeling te voltooien laat stijgen. Overigens zijn er geen noemenswaardige problemen tijdens de ontwikkeling van de responsive website opgedoken.

[1] Document Object Model

# Hoofdstuk 9

# Conclusie

In dit werk werd een vergelijkende studie en proof-of-concept in functie van een doelbewuste keuze tussen een cross platforme mobiele applicatie en een responsive website opgezet. Hierbij werden verschillende

# Bibliografie


Adobe (2017). Get started.

dofactory.com (2017). Facade .net design pattern in c# and vb - dofactory.com.

Google (2017). Get started with publishing | android developers.

Holmes, J. (2017). Building a simple photo gallery in asp.net mvc framework.

Inc, A. (2017). Apple developer program.

Microsoft (2017a). Asp.net mvc overview.

Microsoft (2017b). Build apps with native ui using xamarin in visual studio.

Microsoft (2017c). Cross-platform mobile development in visual studio.

Microsoft (2017d). Register as an app developer.

Paulo R. M. de Andrade, A. B. A. (2015). Cross platform app a comparative study. *International Journal of Computer Science & Information Technology (IJCSIT)*, 7(1):33–40.

Stormpath. Has asp.net core killed web forms? - stormpath user identity api.

# Lijst van figuren

# Lijst van tabellen