2016•2017
FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN
master in de industriële wetenschappen: elektronica-ICT

Masterproef

Integratie van social media services en chat bots in een online platform voor zwemclub ZVO

Promotor :
dr. Kris AERTS

Promotor :
Dhr. KRISTOF RYHEUL

Jeroen Timmermans
Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

Gezamenlijke opleiding Universiteit Hasselt en KU Leuven

2016•2017

Faculteit Industriële ingenieurswetenschappen

master in de industriële wetenschappen: elektronica-ICT

Masterproef

Integratie van social media services en chat bots in een online platform voor zwemclub ZVO

Promotor :
dr. Kris AERTS

Promotor :
Dhr. KRISTOF RYHEUL

Jeroen Timmermans

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

universiteit hasselt | KU LEUVEN

Voorwoord

In het kader van mijn opleiding Master Industriële Ingenieurswetenschappen Elektronica-ICT kreeg ik de kans om mijn stageperiode te doorlopen voor Zwemclub De Valkaart te Oostkamp.

Gedurende deze stage kreeg ik de opdracht een online platform voor deze zwemclub te ontwikkelen. Ik had bij aanvang van deze masterproef zelfs nog nooit een website gemaakt, laat staan een online platform opgestart. Maar dit was net de reden waarom ik voor deze masterproef gekozen heb. Ik wou iets doen wat ik nooit gedaan heb. Ik heb me vaak beklaagd dat ik amper voorkennis had en heb vaak het gevoel gehad dat ik aan een koersrit was begonnen, maar nog maar net op een driewieler had leren rijden. Hierdoor ben ik een paar keer van mijn traject afgeweken en heeft het me allemaal meer tijd gekost dan verwacht. Uiteindelijk ben ik er toch in geslaagd de weg terug te vinden en heb ik de finish behaald. Als ik kijk naar wat ik nu weet en een jaar geleden, dan heb ik termen geleerd en inzichten gekregen in een wereld die vroeger voor mij onbegrijpelijk was. Ik ben nog altijd allesbehalve een meester in back- en frontendontwikkeling en zal nog vaak van mijn fiets vallen. Maar hoe vaak ik ook val, elke keer als ik opsta zal ik weer een stapje dichter staan om misschien zelf ooit een topontwikkelaar te worden.

Ik wil ook mijn externe promotor Mr. Kristof Ryheul bedanken voor de aangename en fijne samenwerking die ik mocht ervaren tijdens het realiseren van dit project. Mijn interne promotor Prof. Dr. Kris Aerts bedank ik voor het begeleiden en bijsturen van deze masterproef. Mijn ouders voor alle steun en kansen die ze me hebben gegeven. Zonder hun steun en geduld was ik er nooit in geslaagd om het van secundair Mechanica student tot master Elektronica-ICT te schoppen. Als laatste wil ik ook nog mijn dank betuigen aan mijn vriendin Anouck Royen voor de steun die ik altijd van haar gekregen heb gedurende dit gehele proces.

Inhoudstafel

Lijst van tabellen

Lijst van figuren

Verklarende woordenlijst

API: Application Programming Interface

Git: versiebeheersysteem

JSON: Javascript Object Notation

JWT: JSON Web Token

ML: Machine Learning

MVC: Model View Controller

NLP: Neural Language Processing

Npm: Package manager voor JavaScript

ORM: Object-Relational Mapping

OS x: besturingssysteem voor Mac

PHP: Hypertext Preprocessing, scriptingtaal voor het maken van webapplicaties

REST: Representational State Transfer

SDK: Software Development Kit

SSL: Secure Sockets Layer, encryptie protocol voor beveiligde communicatie.

Turing-compleet: een programmeertaal is Turing-compleet wanneer ze in staat is om eender welke berekening van een andere programmeertaal uit te voeren.

UX: User Experience

Webhook: API interface om callbacks te ontvangen

XML: Extensible Markup Language

Abstract

Zwemclub De Valkaart is een zwemclub gelegen in Oostkamp met ongeveer 500 leden. Om alle leden, aanwezigheden op trainingen, trainingsschema's, agenda's, … bij te houden en met de leden te communiceren hebben ze gebruik gemaakt van een softwareprogramma. Deze oplossing was verre van optimaal en wordt op dit moment zelfs niet meer gebruikt.

Het opzet van deze Masterproef is een systeem te ontwikkelen dat eenvoudig is in gebruik en toepasbaar is op mobile platformen en dat gebruik maakt van koppelingen met populaire sociale media. De applicatie moet de communicatie tussen leden en bestuur vereenvoudigen en het beheer van de club overzichtelijker maken.

Voor de ontwikkeling van de app is er eerst gekeken naar een geschikte programmeertaal om zowel de back- als frontend in te ontwikkelen. Uiteindelijk is er gekozen om een crossplatform applicatie te ontwikkelen en hiervoor het Ionic framework te gebruiken. Voor de backend is er gekozen om Node.js te gebruiken in combinatie met het Sails.js framework. Het social media aspect is uitgewerkt in de vorm van bots en geeft gebruikers naast de app een alternatief om met het platform te communiceren. Dit alles maakt gebruik van bestaande API's in combinatie met op maat gemaakte functies.

Door de combinatie van bestaande API's en een op maat gemaakte backend is er een platform ontstaan dat het gebruiksgemak aanzienlijk verhoogt en er voor zorgt dat meer gebruikers het systeem gebruiken.

Abstract in English

Swimming club De Valkaart is a swimming club located at Oostkamp Belgium, with approximately 500 members. To manage all, attendance lists, training schedules, agendas, … and share all of this with their members they used to use a software program. This solution was far from ideal and currently the software is no longer in use.

The purpose of this master thesis is to design and build a platform that is easy to use and uses a mobile application. The application must simplify the communicate between members and admins and keep the administration of the club clear.

For the development of the app there has first been searched for the appropriate programming language for both the back- and frontend. Eventually there has been chosen to develop a crossplatform app which uses the Ionic framework. For the backend the choice was made to use Node.js in combination with the SailsJS framework. The social media integration has been achieved by using bots and gives users, besides the app, an alternative interface to the platform. All of this makes use of existing APIs in combination with custom functions.

Thanks to the combination of existing APIs and a custom made backend it was possible to build a system that enhances the ease of use and encourages more users to use the system.

1. Inleiding

1.1. Situering

Wat is de club en hoe is ze georganiseerd?

'De Valkaart' (ZVO) is een zwemclub in Oostkamp.

De leden en organisatoren van de zwemclub bestaan momenteel uit:

- Voorzitter
- Bestuursleden
- Zwemmers
- Redders
- Trainers
- Penningmeester(s)
- Secretaris

Wat doen ze?

Bij het organiseren van een verenging komt altijd heel wat kijken, gaande van het regelen van dagdagelijkse taken tot de verloning van trainers en redders of het organiseren van evenementen. Bij zwemclub ZVO is dit niet anders.

Hoe doen ze het?

Momenteel wordt dit allemaal geregeld door gebruik te maken van social media, e-mail, telefoon, … Dit zorgt echter regelmatig voor misverstanden en is daarbovenop ook nog eens tijdrovend.
Wetende dat zwemmers, redders en trainers ook nog eens bestaan uit meerdere subcategorieën, wordt de structuur van de ledenlijst nog uitgebreider en de organisatie nog omslachtiger.

Wat willen ze?

In een tijd van mobiele applicaties en interactieve webapplicaties wenst zwemclub ZVO de huidige manier van werken te vereenvoudigen door gebruik te maken van een online platform in combinatie met mobiele applicaties waarop leden en bestuursleden op een eenvoudige en efficiënte manier met elkaar kunnen communiceren. Voornamelijk de communicatie tussen trainers en redders moet hiermee efficiënter worden.

1.2. Probleemstelling

De zwemclub heeft begin vorig jaar een onlineapplicatie, Smitapp, in gebruik genomen. Deze applicatie maakt het mogelijk voor bestuursleden om aanpassingen te doen in de onlinedatabase van de zwemclub. De Smitapp is echter zeer uitgebreid en net daardoor voor de meeste toepassingen nodeloos log en complex. Ook is er met de Smitapp geen rechtstreekse interactie mogelijk tussen het bestuur en de leden van de zwemclub. Hierdoor is het bijvoorbeeld niet mogelijk meldingen te sturen naar leden in geval van een afwezigheid van een trainer.
De huidige manier van werken biedt ook geen oplossing om de interactie tussen bestuur en leden minder complex en tijdrovend te maken. Een oplossing voor dit probleem zou eruit bestaan het systeem zodanig te ontwikkelen dat het deels het werk van het bestuur overneemt door zelfstandig beslissingen te nemen.

Voorbeelden hiervan zijn:

- het automatisch zoeken naar vervangers;
- het aanpassen van de agenda;
- aanwezigheidslijsten aanpassen.

De Smitapp bevat tegelijk ook te veel features voor bepaalde groepen. Een vereenvoudigde view voor deze groepen zou dus een oplossing kunnen bieden.
Het is duidelijk dat het grootste probleem ligt in het te complex zijn van de Smitapp en het gebrek aan mogelijkheden tot communicatie tussen bestuur en leden.

Tijdens het verloop van de masterproef is gebleken dat de Smitapp niet voldoet aan de wensen en noden van de zwemclub en is hierdoor ook de samenwerking stopgezet.

1.3. Doelstelling

De opzet van deze masterproef was een applicatie te ontwikkelen die datgene biedt waar de Smitapp in te kort schoot. Dit is mogelijk gemaakt door gebruik te maken van een mobiel platform in combinatie met bestaande webservices en een backend. Dit geheel zorgt voor een platform dat met een minimum aan uitleg begrepen kan worden door de gebruikers van het platform. Hierdoor is de gebruiksvriendelijkheid hoger dan die van de voorganger, de Smitapp.

Volgende doelstellingen zijn samen met de bestuurders opgesteld:

1. onderzoek naar een gepaste ontwikkelingsmethode,
   hieruit moet blijken welke programmeertaal de efficiëntste weg naar een werkend platform kan bieden;
2. mobiele applicatie voor tablet met goede UX;
3. eenvoudig aanvoelende applicatie of optimaal alternatief;
4. link met sociale platformen met het oog op delen van media, bijvoorbeeld een nieuwspagina in een app;

5. bijhouden van aan- en afwezigheden in de vorm van lijsten;
6. een hogere gebruiksvriendelijkheid gaat boven meer uitgebreide functionaliteit;
7. meldingen sturen, gebruikers op de hoogte brengen wanneer er wijzigingen plaatsvinden in hun agenda of wanneer ze een bericht hebben ontvangen;
8. planning trainers, plaats waar een trainer een les kan inplannen en dit kan delen met andere leden.

1.4. Gebruikte materialen en methodes

Voor dit project zijn er meerdere functies/wensen opgegeven waar het platform over moet beschikken. Deze zijn de volgende:

- agenda/planning,
- notificaties,
- lijst met aan- en afwezigheden,
- bijhouden van rapporten,
- gebruikers profielen,
- gebruikers authenticatie,
- nieuwspagina,
- doodle systeem of alternatief,
- connectie met database,
- communicatie tussen backend en mobile applicaties,
- ondersteuning voor userbase <500,
- REST services,
- integratie bestaande API's.

Om dit project in een zo kort mogelijke tijdspanne te kunnen realiseren is het voordelig voor elk van deze functies een bibliotheek of webservice ter beschikking te hebben. Hierdoor wordt het uitschrijven van bepaalde functionaliteiten uitgespaard en wordt de ontwikkelingstijd ingekort. Alle onderzoek en vergelijkingen zijn dan ook gedaan met het oog op het realiseren van deze functies.

Als eerste stap wordt in *2. Bestaande apps en platformen* een overzicht gegeven van enkele platformen die vergelijkbaar zijn met het te realiseren platform. De beschikbare functies worden vergeleken met de wensen van ZVO en het wordt duidelijk waarom er vraag is naar een opmaatgemaakt platform.

Vervolgens wordt er een in hoofdstuk 3*. Vergelijking programmeertalen* een vergelijking gemaakt welke programmeertaal het meest geschikt is om te gebruiken bij het maken van zowel de front- als backend.

Mogelijke categorieën zijn:

- script-talen (JavaScript),
- applicatietalen (Java,C#),
- webtechnologieën (JavaScript, PHP).

Voor het ontwikkelen van een mobiele applicatie is er een onderscheid gemaakt tussen drie groepen:

- Native, gebruikmakende van de oorspronkelijke ontwikkelingstaal en tools;
- Hybrid, een andere taal, maar wel toegang tot native componenten;
- Web app, web gebaseerde applicatie op gelijkaardige principes als een website.

Ook hier is er onderzocht welke het toepasselijkste is. Het onderscheid tussen deze drie methodes wordt nader toegelicht in *4. Crossplatform.*

Om berichten te kunnen sturen naar de gebruikers van de ZVOapp wordt er gebruik gemaakt van Notificaties. In *5. Notificaties* wordt een vergelijking gemaakt tussen enkele services.

Uiteindelijk worden er in de hoofdstukken *6. Backend* en *7. Frontend* de uitwerkingen en daarbij komende problemen nader besproken.

2. Bestaande apps en platformen

Voordat er overgaan wordt tot de ontwikkeling van het ZVO platform wordt er een vergelijking gemaakt tussen enkele bestaande platformen en apps.

2.1. Vergelijking

Onderstaande apps en platformen zijn gemaakt voor het beheren van sportclubs. De zwemclub ZVO slaagt er echter niet in succesvol gebruik te maken van reeds bestaande oplossingen. Onderstaande vergelijking zoekt uit waar deze platformen sterk in zijn en waar ze voor ZVO tekort komen.

2.1.1. Socie

Socie is een platform gemaakt voor ondersteuning van het sociale aspect binnen een club [1]. Onder andere zwemclub Esca maakt gebruik van Socie en heeft daarnaast ook een Socie app [2].
Volgens de website van Socie zijn de beste functies:

- jaarplanning, een functie die het mogelijk maakt een algemene kalender op te stellen;
- fotoalbums, voor het delen van foto's met leden;
- push-berichten.

Twee van de "beste" functies, fotoalbums en Push-berichten, kunnen eigenlijk al gerealiseerd worden door een gesloten Facebook groep aan te maken. Het voordeel van de app is natuurlijk dat leden niet verplicht zijn over Facebook te beschikken. De jaarplanningsfunctie kan nagebootst worden met een online agenda zoals Google Agenda. Op een relatief eenvoudige manier kunnen de drie sterkste punten van de app nagebootst worden door gratis platformen. Socie is zich hier blijkbaar van bewust en biedt de basis app, die de drie sterkste functies bevat, gratis aan.

Socie is een leuke oplossing om nieuws en berichten te delen met de leden van een club. Dit is echter niet voldoende voor de ZVOclub, die voornamelijk behoefte heeft aan oplossingen omtrent het beheren van de club *(zie 1.3 doelstellingen)*.

2.1.2. Sportadministratie

Sportadministratie legt, zoals de naam het zegt, de focus op administratie van een club. Het beschikt over de mogelijkheid om clubleden, kalenders, wedstrijden, aanwezigheden, … te beheren. De mogelijkheden leunen aan bij de wensen van de ZVOclub. Op grafisch vlak heeft de app veel weg van een view op een database en doet het denken aan de Smitapp, zie onderstaande afbeelding.

Figuur 1: Screenschot van sportadministratie software [3]

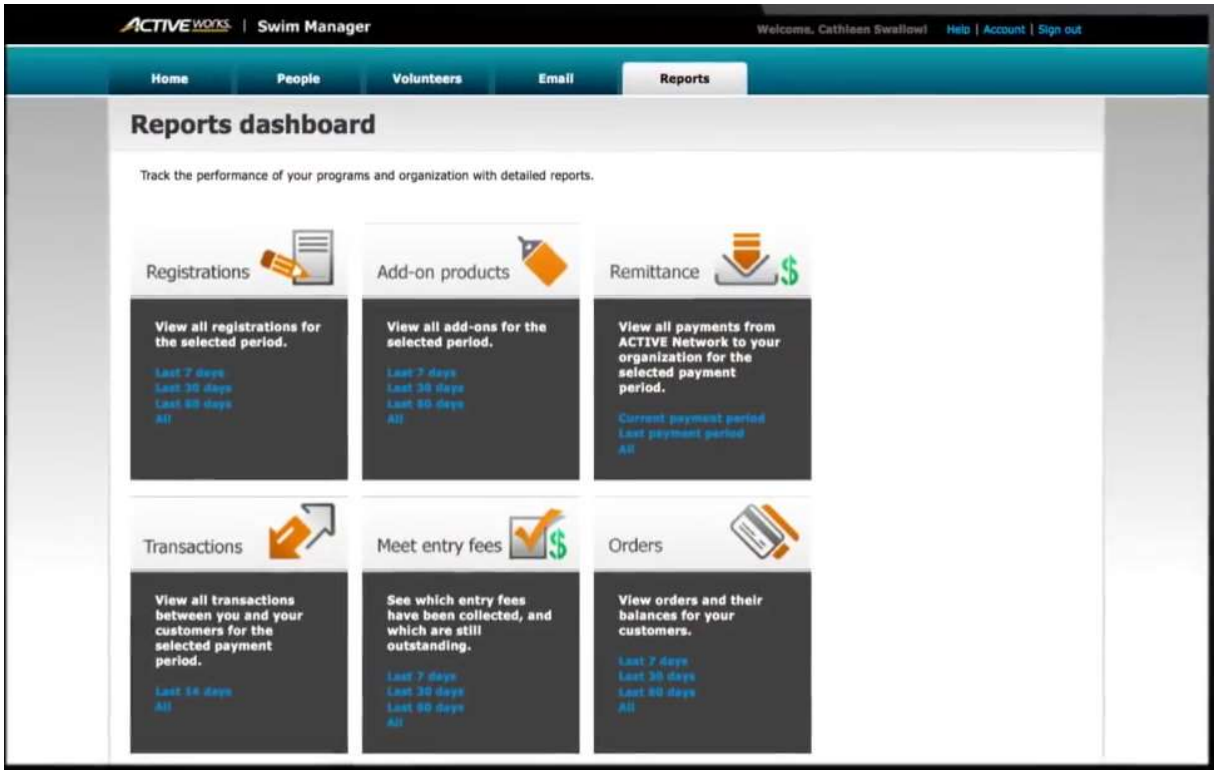
De Smitapp was omwille van zijn view te complex en wordt daarom niet langer gebruikt. De Sportadministratie app voorziet enkel een e-mail- en sms-module om met de leden te communiceren. Er is dus geen optie om via een mobiele applicatie de club te kunnen beheren of met leden te communiceren en voldoet daarmee dus niet aan alle eisen.

2.1.3. Meet Manager 6.0

Meet Manager [4] is een tool aangeboden door ActiveHy-Tek die de nadruk legt op het regelen van bijeenkomsten tussen zwemclubs en het bijhouden van zwemresultaten. Naast de managementsoftware bestaat er ook een Meet app, Meet Mobile [5]. De app kan gebruikt worden door leden of fans om live de zwemresultaten op de meet te volgen. De tool en app leggen volledig de focus op het afspreken met andere clubs en bijhouden van resultaten. Deze app kan een meerwaarde zijn maar biedt enkel functionaliteiten die door de ZVOclub niet als problemen worden gezien en lost daarmee geen bestaande problemen op.

2.1.4. Swim Manager

Naast Meet Manager biedt ActiveHy-Tek ook de software Swim Manager [6] aan. Onderstaande afbeelding geeft een overzicht van het Swim Manager Reports dashboard.



Figuur 2: Screenshot Swim Manager Reports dashboard [7]

In bovenstaande figuur is te zien dat Swim Manager vooral de nadruk legt op het beheren van leden en betalingen. Er wordt bijvoorbeeld niet aan de vereiste om ook aanwezigheidslijsten te beheren voldaan. De software legt te zeer de nadruk op het regelen van wedstrijden en evenementen en voldoet niet aan de wensen van de ZVOclub.

2.2. Conclusie

Hoewel er reeds applicaties bestaan, voldoen er geen volledig aan de eisen van de ZVOclub. Ze zijn ofwel te algemeen of leggen de nadruk op functies die niet noodzakelijk zijn voor de ZVOclub. Dit is ook niet verwonderlijk aangezien de applicaties focussen op de meest voor de hand liggende problemen bij het beheren van een sportclub. Hierdoor spreken ze een zo breed mogelijk publiek aan, maar voldoen ze niet aan alle specifieke eisen van de gebruikers. Het is te vergelijken met een appartement kopen of een huis zelf laten bouwen. Het appartement moet je grotendeels nemen zoals het is, het huis kan je volledig naar je eigen wensen laten bouwen.

3. Vergelijking programmeertalen

In onderstaande tekst wordt een vergelijking gemaakt tussen enkele programmeer- en scriptingstalen die gebruikt kunnen worden voor server-side (backend) ontwikkeling. Eerst wordt er gekeken naar welke talen momenteel het meest besproken worden en waarom ze zo populair zijn. Op basis hiervan worden enkele talen uitgekozen en wordt voornamelijk gekeken naar de voor- en nadelen van iedere taal en hun bijhorende (meest bekende) frameworks. Met de achterliggende gedachte "een goede ingenieur is ook een luie ingenieur", is er gezocht naar de taal en het bijhorende framework waarmee op een zo effectief mogelijke manier het gewenste resultaat gerealiseerd kan worden *(zie 1.4 Materiaal en methode)*. Het is misschien interessant om in iedere taal een testprogramma uit te werken en dit te vergelijken met de andere talen op vlak van snelheid, leesbaarheid, gebruiksgemak en leergraad. Dit zou echter een studie op zich worden en maar een minimale bijdrage hebben aan het oorspronkelijke project. Het meeste van deze informatie is bovendien terug te vinden op het internet.
Het voornaamste is om eerst te kijken naar welke onderdelen nodig zijn om de masterproef te realiseren. Er moet ook wel in het achterhoofd gehouden worden dat er niet zoiets bestaat als een 'beste' programmeertaal. Met ieder van deze talen is het namelijk mogelijk het project te realiseren, maar bepaalde talen zijn mogelijk effectiever dan anderen *(Zie 1.4 Materiaal en methode).*

3.1. Populariteit

Alvorens een vergelijking te maken, wordt er eerst gezocht naar de populairste talen op dit moment. Op basis hiervan kan er dan gekeken worden of ze voldoen aan de noden *(Zie 1.4 Materiaal en methode)* om het ZVO-platform te realiseren.

Om tot een ranking te komen, is er gebruik gemaakt van *Spectrum.ieee.org*. Deze website geeft een ranking met daarin alle momenteel populaire programmeertalen. Om tot een vergelijking te komen, wordt er door IEEE gebruik gemaakt van meerdere bronnen *(zie Figuur 3)*. Deze bronnen kunnen met een slider ingesteld worden om zo een gewogen gemiddelde te bekomen. Onderstaande figuur laat dit zien.

Figuur 3: Datasource IEEE Ranking [8]

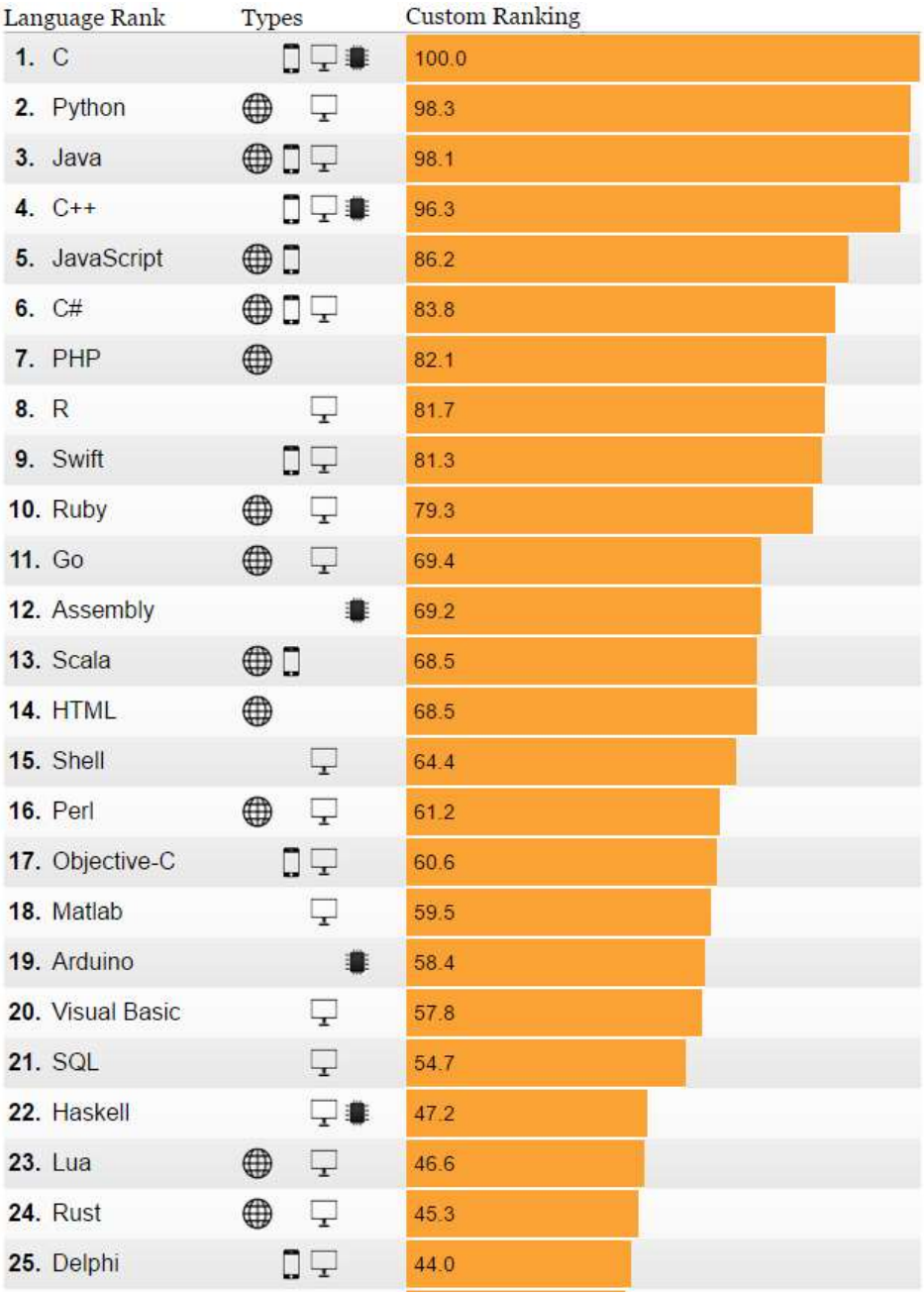
Deze bronnen bevatten het volgende:

- Google (search), het aantal hits bij een Google search;
- Google (trends), het aantal keer dat een zoekterm is ingegeven in Google search;
- Github (active), repositories van een bepaalde taal die actief gebruikt worden;
- Github (created), nieuw aangemaakte repositories;
- Stack Overflow (?s), aantal vragen waar taal in vernoemd wordt;
- Stack Overflow (views), hoeveel aandacht de vragen van de betreffende taal krijgen;
- Reddit, aantal posts dat betreffende taal vermeldt;
- Hackers News, idem Reddit;
- Career Builder, aantal recent toegevoegde job advertenties dat taal vernoemt;
- Dice, idem Career Builder;
- Twitter, aantal tweets dat taal vernoemt;
- IEEE Xplore, aantal artikels dat taal vernoemt.

Als eerste stap wordt er gekeken naar de algemene populariteit van de programmeertalen. Dit wordt gedaan door de sliders allemaal op 100% te zetten. Hierdoor tellen alle data sources even zwaar mee. Achteraf worden er dan sliders aangepast om zo te onderzoeken wat hun invloed op het geheel is. Dit geeft voor het jaar 2016 de volgende grafiek met bijpassende legende:

Tabel 1: Legende van Figuur 4 en 5

Symbool	Beschrijving
	Web
	Mobile
	Enterprise
	Embedded



Figuur 4: Ranking programmeertalen van populairst naar minst populair 1-25 voor het jaar 2016 [8]

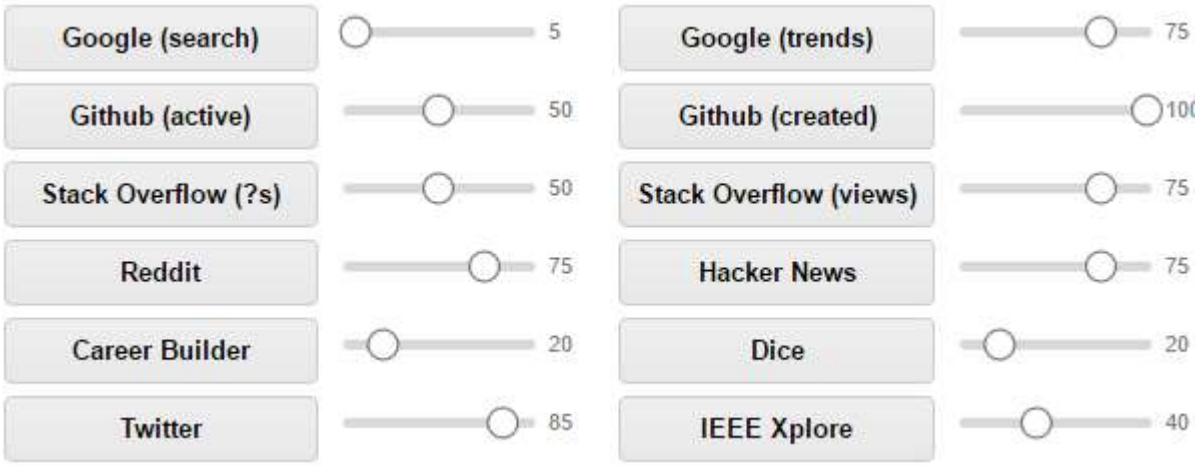
Figuur 5: Ranking programmeertalen van populairst naar minst populair 26-47 [8]

De types worden weergegeven door middel van symbolen *zie Tabel 1* en staan voor Web, Mobile, Enterprise en Embedded. Aangezien bovenstaande lijst veel te uitgebreid is en er ergens een grens moet getrokken worden, is er gekozen om enkel te focussen op de talen uit de top 25, zie *Figuur 4*.

Als we naar *Figuur 4* kijken, zien we dat niet alle talen webtalen zijn. Ze voldoen dus niet allemaal aan de eisen om de ZVOapp te realiseren. Er is ook te zien dat PHP en Python geen ondersteuning bieden voor de ontwikkeling op mobiele platformen. Hierdoor zijn ze niet geschikt om een mobiele applicatie te ontwikkelen, maar maakt het ze niet minder geschikt als backend taal. In de vergelijking zijn ook HTML, Arduino, SQl en soortgelijken opgenomen. Dit zijn eigenlijk geen echte programmeertalen, maar volgens de auteurs worden alle talen die een computer instructies geven als programmeertalen gezien. Niet alle talen in de lijst zijn Turing-compleet.

De vergelijking is momenteel nog te oppervlakkig. Door met de sliders te schuiven kan er meer toegespitst vergeleken worden. IEEE heeft dit gedaan voor “trending”, de talen die snel aan het toenemen zijn in populariteit, “jobs” en “opensource”.

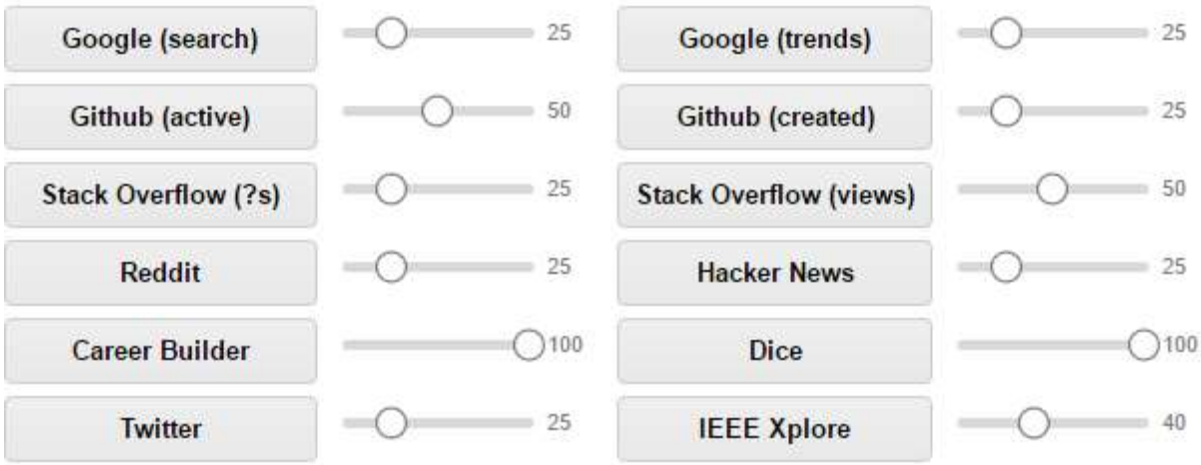
Voor trending worden de sliders als volgt geplaatst:



Figuur 6: Positie slider voor trending ranking [8]

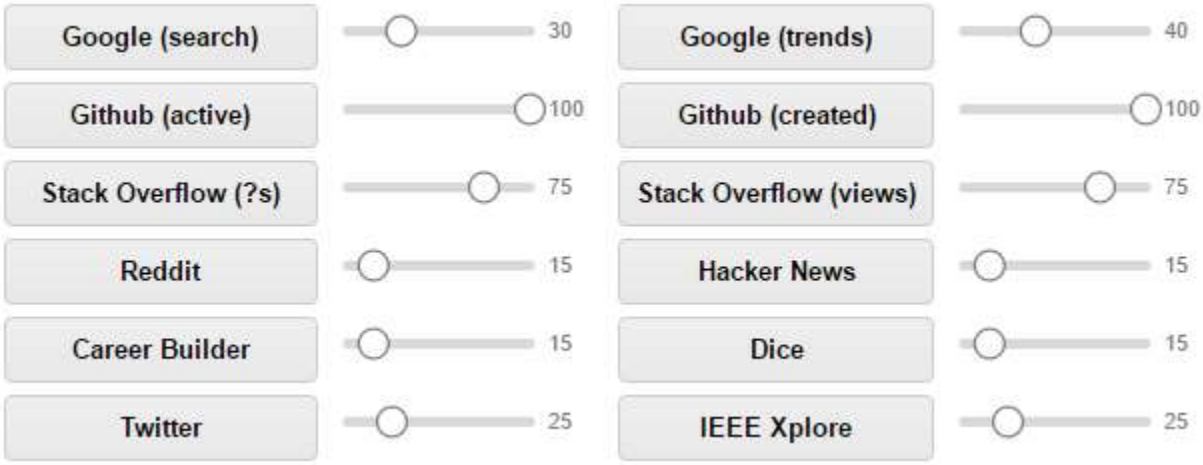
Trending wil zeggen dat de talen op korte termijn zijn gestegen in populariteit en momenteel veel besproken worden. De resultaten van Google (search), Career Builder, Dice en IEEE Xplore zijn afhankelijk van hoe lang bepaalde talen al bestaan en worden daarom op minder dan 50% gezet.

Voor Jobs geeft dit de volgende sliders:



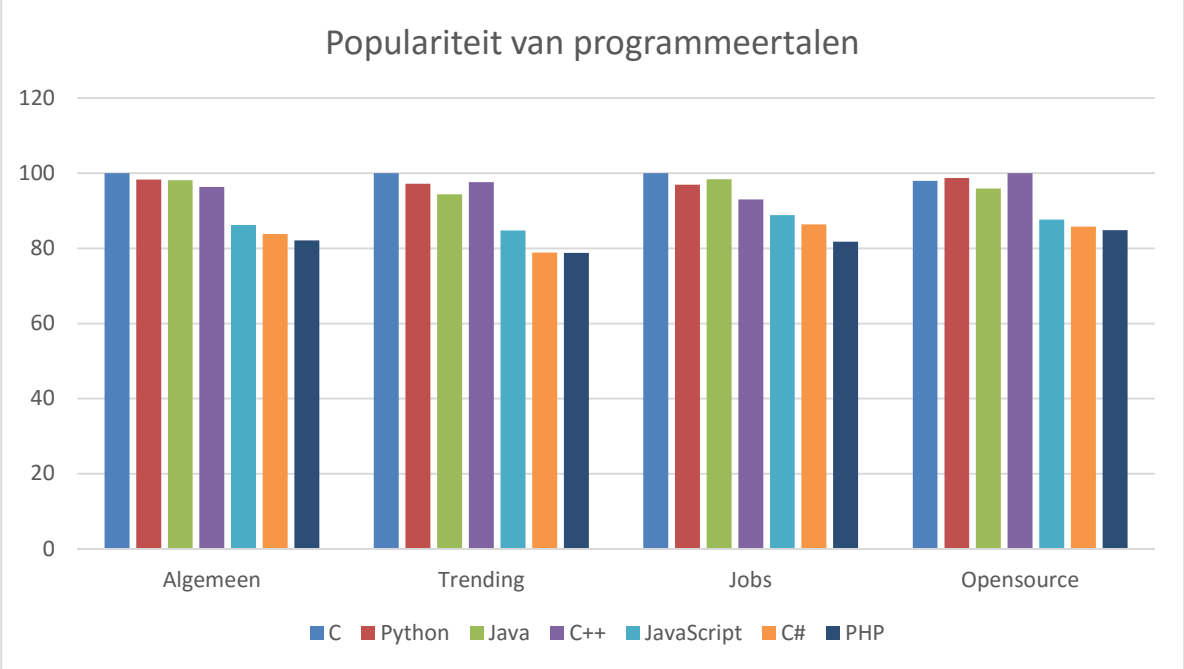
Figuur 7: Positie sliders voor Jobs ranking [8]

Voor jobs worden Career Builder en Dice, de job sites, als belangrijkste geacht. De andere bronnen zijn niet geheel onbelangrijk en worden daarom toch nog op ¼ gezet.



Figuur 8: Positie sliders voor Opensource [8]

Onderstaande grafiek geeft een beeld van hoe de populariteit van de “algemene” top verandert bij het aanpassen van de slides.



Figuur 9: Populariteit programmeertalen volgens IEEE [8]

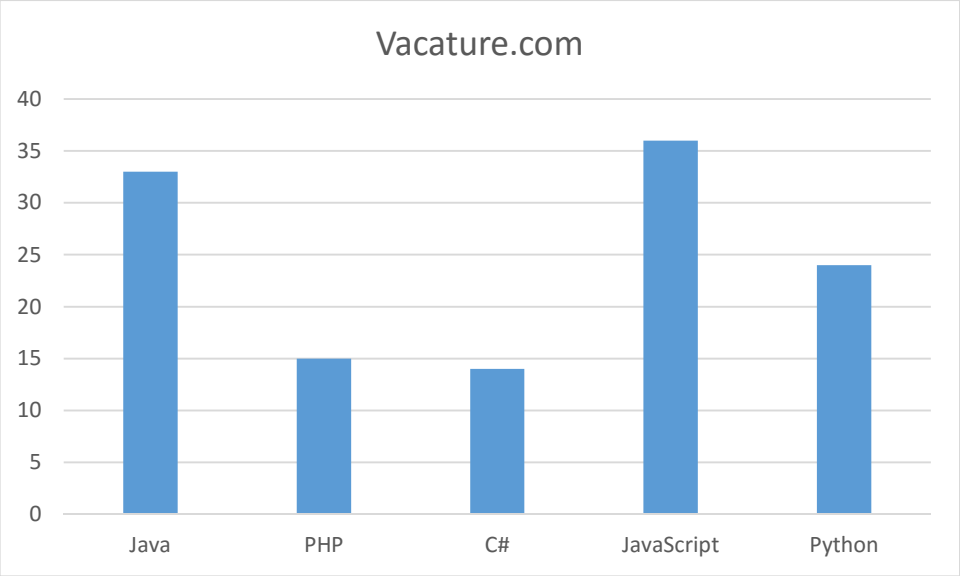
Voor zowel Trending, Jobs als Opensource zag de ranking er anders uit dan de algemene ranking. In bovenstaande vergelijking zijn echter enkel de talen opgenomen die bovenaan in de algemene ranking terugkwamen. Op deze manier wordt het duidelijk hoe de talen scoren in verhouding met elkaar en met aangepaste filters. Het valt op dat de vier uitschieters C, Python, Java en C++, uit de algemene vergelijking ook het hoogste scoren in de andere vergelijkingen. Ten opzichte van elkaar zijn er wel enkele verschillen maar er zijn geen extreme uitschieters. In de Trending vergelijking is te zien dat C als ‘meest trending’ taal naar voren komt.

Dit is opmerkelijk wetende dat C de oudste taal in deze grafiek is. Door de Stack Overflow (views) slider op 0% te zette, verschuift C naar de derde plaats in de trending grafiek.

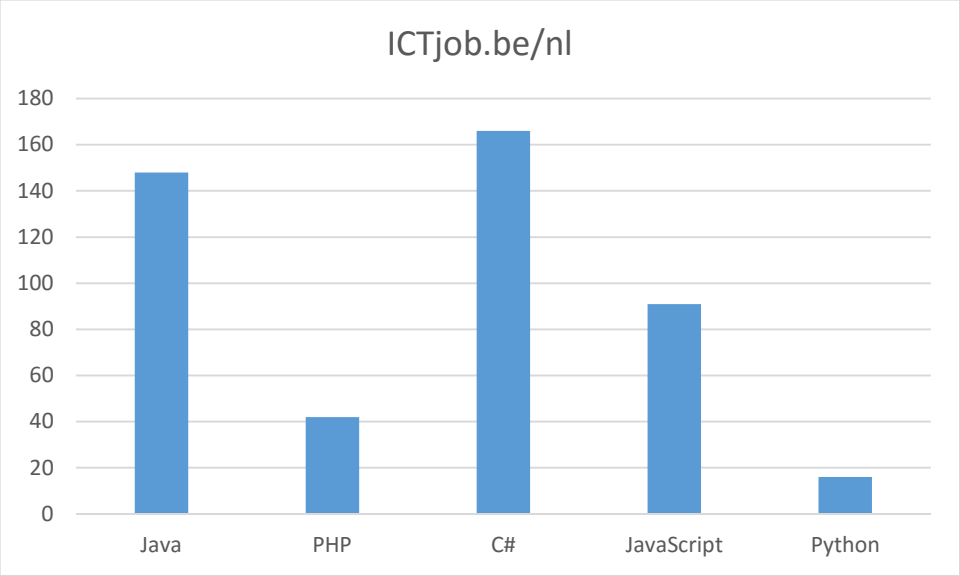
Het verband tussen 'trending zijn' en het aantal gelezen vragen in de afgelopen 12 maanden kan dus enigszins in twijfel getrokken worden.

3.2. Openstaande vacatures

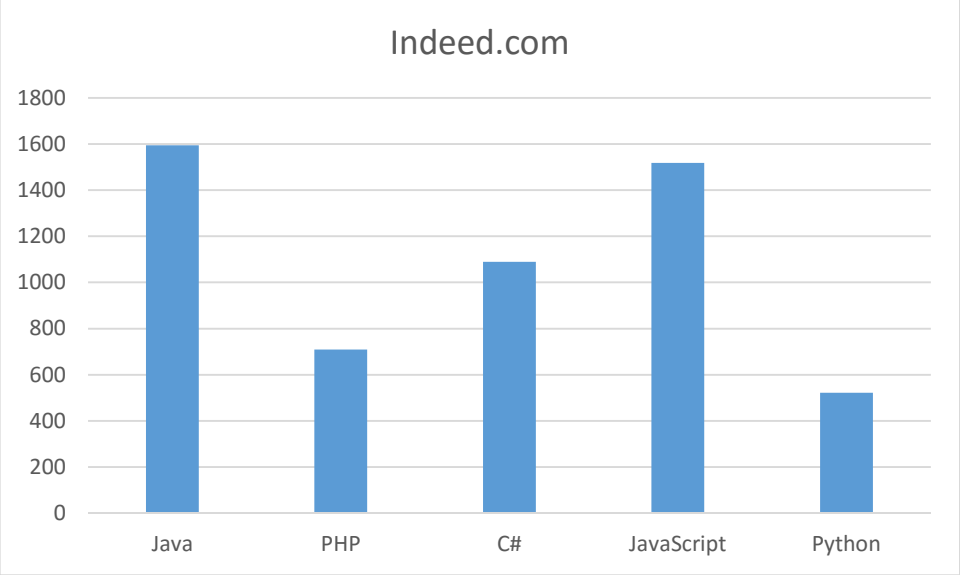
Uit voorgaande paragraaf is gebleken welke talen over het algemeen de 'hipste' en meest besproken talen zijn. Ook is er dankzij de resultaten van Dice en Career Builder een beeld van hoe deze scoren op de arbeidsmarkt. De gebruikte zoekmachine maakt echter enkel gebruik van Amerikaanse bronnen. Om hier een beter zicht op te krijgen is onderstaande vergelijking gemaakt op basis van Belgische job sites. In de Vergelijking in *Figuur 10* is rekening gehouden met welke talen eerder als geschikte talen werden geselecteerd. Onderstaande vergelijkingen geven voor elke taal het aantal openstaande vacatures mee, dit voor vacature.com, ICTjob.be/nl en indeed.com.



Figuur 10: Aantal vacatures/taal op Vacature.com



Figuur 11: Aantal vacatures/taal op ICTjob.be/nl



Figuur 12: Aantal vacatures/taal op Indeed.com

In de vacatures van Indeed en ICTjob zien we dezelfde talen terugkomen in de top drie. Weliswaar in een andere volgorde, maar haast niet te vergelijken met de resultaten op Vacature.com. Dit is gedeeltelijk te wijten aan de zoekmachine van Vacature.com. Deze leek nogal ruim te zoeken en gaf bij het zoeken naar JavaScript zelfs Java of .Net vacatures aan. Dit doet twijfelen aan de betrouwbaarheid van de zoekmachines. Na het nader doornemen van de vacatures is gebleken dat op Vacature.com alle vacatures die de opgegeven zoekterm bevatten, worden weergegeven. Op ICTjob wordt er bijvoorbeeld bij het zoeken naar Python enkel vacatures weergegeven waarbij Python een vereiste is. Een vereiste wil zeggen dat van de werknemer verwacht wordt dat hij deze taal machtig is. Wanneer een taal elders in de vacature terugkomt wordt het machtig zijn van deze taal eerder gezien als een pluspunt dan als een vereiste. Dit verklaart waarom er bij Vacature.com het aantal hits voor JavaScript en Python zo hoog scoren vergeleken met de andere jobsites.

Dit toont ook aan dat hoewel JavaScript en Python geen vereisten zijn voor bepaalde vacatures ze toch een meerwaarde bieden bij het vinden van een job.

Wanneer de resultaten van IEEE en de openstaande vacatures met elkaar vergeleken worden, valt op dat hoewel Python (2 de plaats) en PHP (7de plaats) zeer populair zijn volgens IEEE, ze op gebied van jobaanbiedingen minder scoren dan Java, C# en JavaScript. Het valt ook meteen op dat net deze twee talen gekend zijn omwille van hun lage moeilijkheidsgraad [9] vergeleken met de meer volwassen talen zoals Java en C#. Omwille van deze "lagere" moeilijkheidsgraad zijn ze zeer geliefd bij hobbyisten en voor het maken van kleine projecten. Zo winnen ze aan populariteit op het internet. Java en C# worden eerder gebruikt in enterprise toepassingen en grote projecten zoals bijvoorbeeld gameontwikkeling. Als laatste hebben we dan JavaScript. De taal is door de tijd een onmisbare component van frontend ontwikkeling geworden en kan als ontwikkelaar haast niet meer vermeden worden.

3.3. Conclusie

Uiteindelijk is het gelukt om van 47 talen een top vijf waardige kandidaten te komen. Van deze vijf talen zijn Java, C# en web gebaseerd (PHP + JavaScript) aan bod gekomen in de opleiding Elektronica-ICT. Om een keuze te maken uit deze talen wordt er in de volgende paragraaf dieper ingegaan op de sterktes en zwaktes van de taal op zich en die van hun populairste frameworks; en dit vooral in relatie tot wat nodig is om de ZVOapp te realiseren.

3.3.1. Top drie nader toegelicht

Java

Java is een populaire keuze bij het maken van applicaties voor grote bedrijven [10]. Java is een taal die nagenoeg gelijktijdig met het internet is ontstaan. Java is ook veel meer dan gewoon een programmeertaal, het is een platform/omgeving. Android van Google is hier een mooi voorbeeld van. Een groot voordeel van Java is de compatibiliteit, "Write once, run anywhere" is ook wel een gekend Java moto. Voor het merendeel van de Java web frameworks wordt gebruik gemaakt van een MVC principe. Dit wil zeggen dat de server de view zal renderen en dit als HTML naar de client zal sturen [11].

Pros:

- nagenoeg overal toepasbaar;
- object georiënteerd programmeren;
- grote gebruikers community;
- goede schaalbaarheid;
- snelheid [10].

Cons:

- meestal gebruikt om applicaties te bouwen voor grote ondernemingen of toch zeker op grotere schaal dan die van een lokale sportclub.

Bibliotheken/functionaliteiten

- Java Authentication and Authorization Service (JAAS), kan gebruikt worden om gebruikers authenticatie te integreren;
- Gcm-server.jar, bibliotheek die via GCM (Google Cloud Messaging) het mogelijk maakt berichten te sturen tussen mobiele apparaten en de server.

Populariteit frameworks

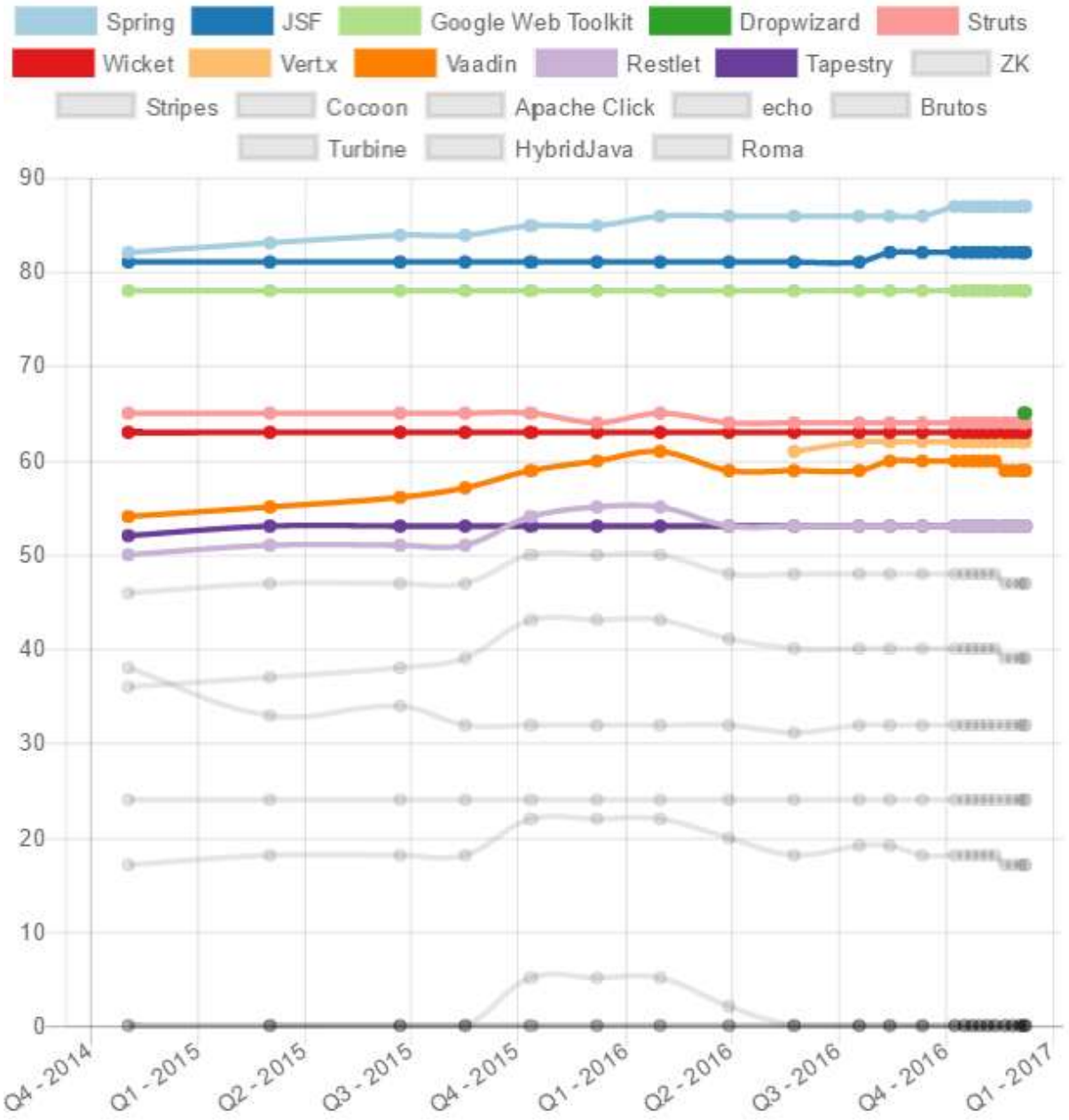
Onderstaande resultaten geven een beeld van welke frameworks momenteel het populairste zijn op basis van:

- GitHub score, is het aantal sterren dat een repository van een framework op GitHub krijgt. Indien het framework niet op GitHub staat wordt er geen rekening gehouden met de GitHub score;
- Stack Overflow score, is gebaseerd op het aantal vragen dat als tag de naam van het framework heeft.

Tabel 2: Java frameworks gerangschikt op populariteit [12]

Framework	Score
Spring	87
JSF	82
Google Web Toolkit	78
Dropwizard	65
Struts	64
Wicket	63
Vert.x	62
Vaadin	59
Restlet	53
Tapestry	53
ZK	47
Stripes	39
Cocoon	32
Apache Click	24
echo	17
Brutos	0
Turbine	0
HybridJava	0

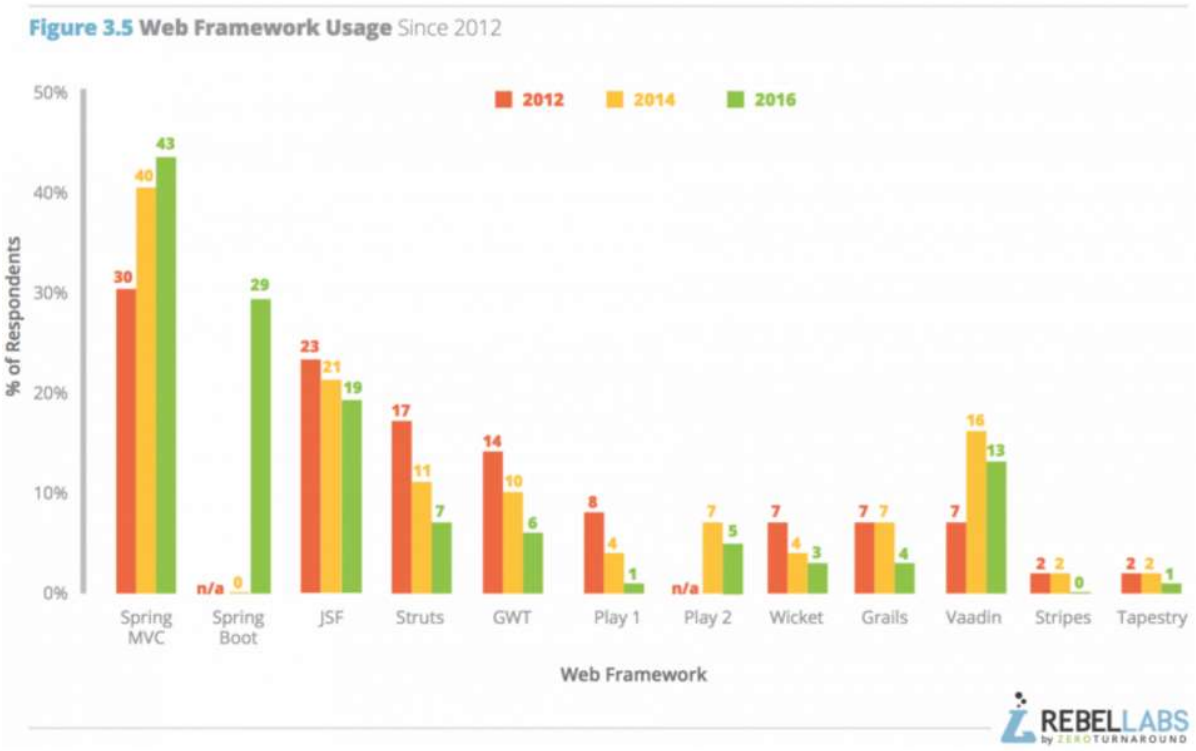
Roma	0



Figuur 13: Popupalariteit Java frameworks sinds kwartaal4-2014 [12]

Bij het nader onderzoeken van de populairste frameworks in bovenstaande vergelijking is gebleken dat hoewel sommige hier heel hoog scoren bijvoorbeeld Dropwizard en Struts, er niet zoveel informatie over was te vinden. Dit doet vermoeden dat bovenstaande resultaten afwijken van de werkelijke populariteit. De resultaten van het "Java Tools and Technologies Landscape Raport 1016" van zeroturnaround.com [13] bevestigen dit vermoeden en geven een andere kijk op de populairste frameworks.

Onderstaande grafiek komt uit het eerder vernoemde rapport [13]:



Figuur 14: Framework en hun populariteit sinds 2012 [13]

Bovenstaande grafiek is het resultaat van een ondervraging van 2040 ontwikkelaars (2016). De resultaten geven weer welk percentage van deze ontwikkelaars een bepaald framework gebruikt. Deze grafiek geeft een heel ander beeld dan Hotframeworks.com. Het laat zien dat een framework met veel Stackoverflow vragen geen garantie is dat het een "geliefd" framework is en eigenlijk enkel laat zien dat er veel vragen zijn. Zo scoort bijvoorbeeld Dropwizard relatief hoog in *Figuur 13* en komt het zelfs niet te pas in bovenstaande vergelijking. Een hoog aantal vragen kan duiden op veel gebruikers maar ook bijvoorbeeld op slechte documentatie van het framework. Uit bovenstaande resultaten wordt duidelijk dat Spring veruit het populairste framework is. Spring heeft sinds 2012 zelfs alleen nog maar gewonnen in populariteit terwijl alle andere hier hebben moeten inboeten.

Wanneer we de drie populairste frameworks van *Figuur 13* en *14* naast elkaar leggen komen volgende frameworks naar voren:

Spring

Is het meest populaire Java framework voor web ontwikkeling en bestaat al sinds 2003 [11]. Het wordt vooral gebruikt voor applicaties voor grote ondernemingen. Spring maakt gebruik van het MVC principe en is een RESTful web framework. Aangezien het een framework is dat de nadruk legt op web toepassingen is het zeker een meerwaarde voor het ontwikkelen van de ZVOapp. Deze is namelijk zeer afhankelijk van webservices.

JSF

JavaServer Faces wordt gebruikt voor het bouwen van webapplicaties en is voornamelijk frontend gericht. Daarbovenop ziet de frontend er achterhaald uit en leunt het niet aan met de doelstellingen van het ZVO platform. Daar is het vooral de bedoeling dat de frontend op mobiele apparaten wordt uitgewerkt.

Google Web Toolkit

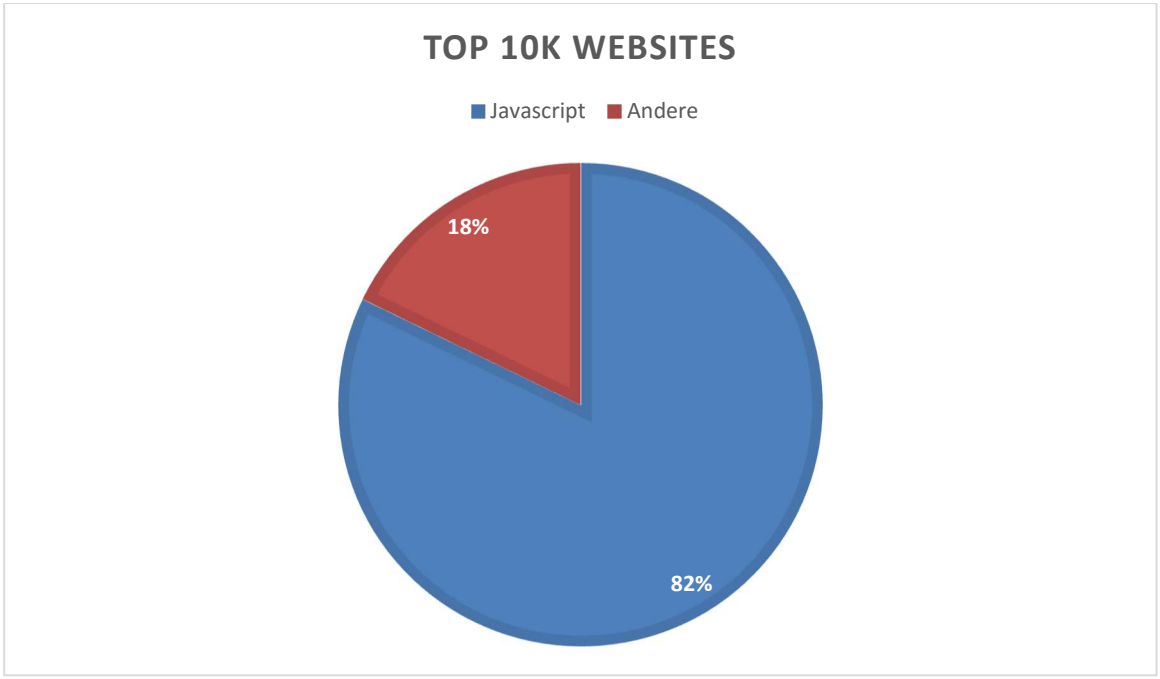
Is een framework dat het mogelijk maakt Javacode om te zetten in JavaScript. Hierdoor kan een browser de code verstaan. Dit is weer frontend gericht en niet van toepassing op de backend.

Dropwizard

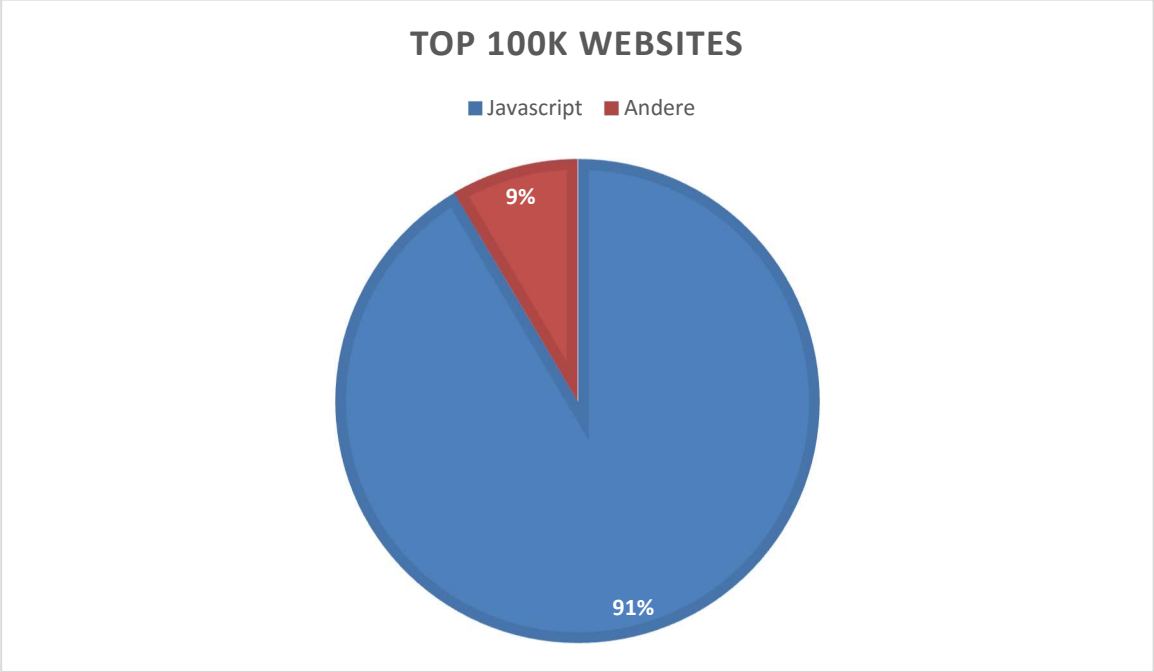
Dropwizard legt de focus op RESTful webservices en daarmee ook webapps. Het maakt gebruik van bestaande bibliotheken waaronder Jetty (http), Jersey (REST) en Jackson (JSON). Dropwizard legt daarmee een solide basis voor het bouwen van een backend die dankzij de RESTful webservices ook geschikt is in combinatie met mobiele toepassingen. Het moet dus mogelijk zijn de ZVObackend te ontwikkelen gebruikmakende van Dropwizard.

JavaScript

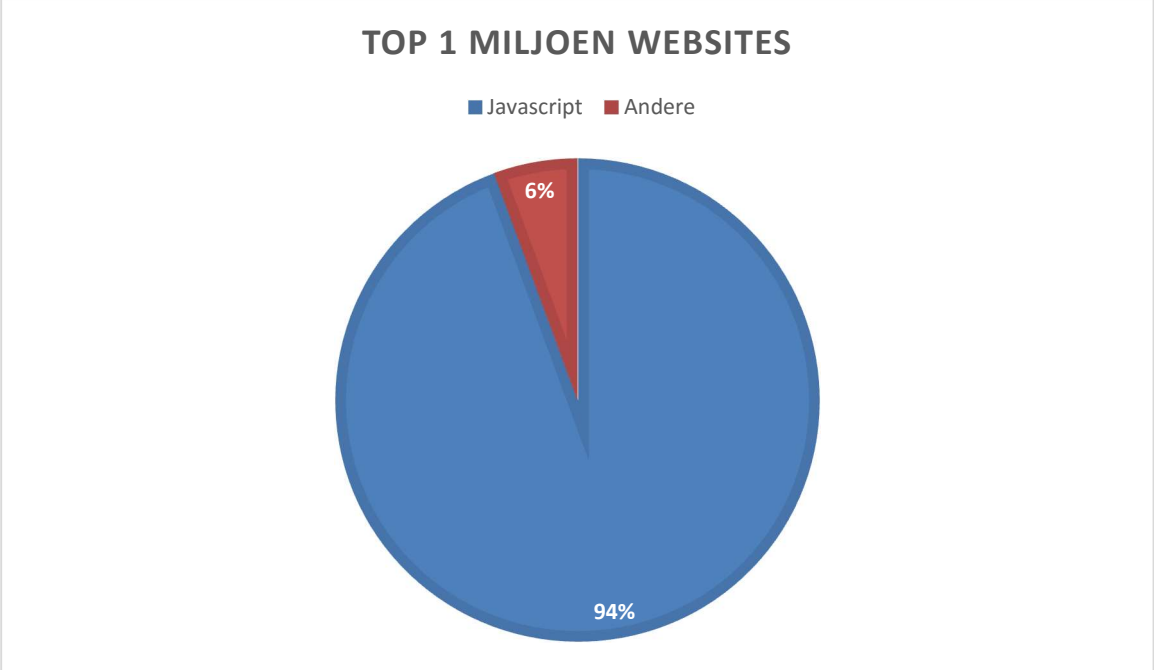
JavaScript is de populairste scriptingtaal voor de ontwikkeling van websites. JavaScript is tegenwoordig absoluut niet meer weg te denken bij het maken van een website. Onderstaande figuren maken dit duidelijk en zijn gebaseerd op de gegevens van Builtwith.com [14]. Ongeacht welk backend framework er gekozen wordt, voor de frontend zal er hoogstwaarschijnlijk JavaScript aan te pas komen.



Figuur 15: Percentage uit top 10K websites dat gebruik maakt van JavaScript (03/03/2016)



Figuur 16: Percentage uit top 100K websites dat gebruik maakt van JavaScript (03/03/2016)



Figuur 17: Percentage uit top 1mil websites dat gebruik maakt van JavaScript (03/03/2016)

Bovenstaande figuren laten het percentage aan websites zien dat gebruik maakt van JavaScript, dit voor de 10.000, 100.000 en 1 miljoen meest bezochte websites.

Deze hoge percentages zijn voornamelijk te danken aan het JavaScript gebruik bij frontend ontwikkeling. Dit zegt niets over het gebruik van JavaScript als backend ontwikkelingstaal, maar toont wel aan dat JavaScript zeer populair is bij web ontwikkelaars. Om een backend te ontwikkelen in JavaScript wordt er gebruik gemaakt van Node.js.

Node.js

In 2009 werd Node.js in het leven geroepen en werd het voor frontend-ontwikkelaars ook mogelijk backend logica te schrijven in JavaScript. Node.js is een JavaScript runtime die gebouwd is op de Google Chrome V8 JavaScript engine. Het is gelijkaardig in ontwerp en beïnvloed door systemen als Twisted van Python en de Even Machine van Ruby [15]. Node.js maakt door het toepassen van slimme 'trucjes' toch mogelijk om zaken te doen die JavaScript normaal gezien niet kan. Zo wordt het bijvoorbeeld mogelijk, hoewel Node.Js niet ontwikkeld is met threading in het achterhoofd, toch meerdere processorkernen aan te spreken door het gebruik van specifieke API's.

In de beginjaren van JavaScript was het logisch dat je als ontwikkelaar zowel PHP als JavaScript gebruikte, PHP zorgde voor de server en JavaScript voor de client-side. Door de opkomst van Node.js kan JavaScript ook gebruikt worden om services te schrijven en is JavaScript een concurrent van PHP geworden.

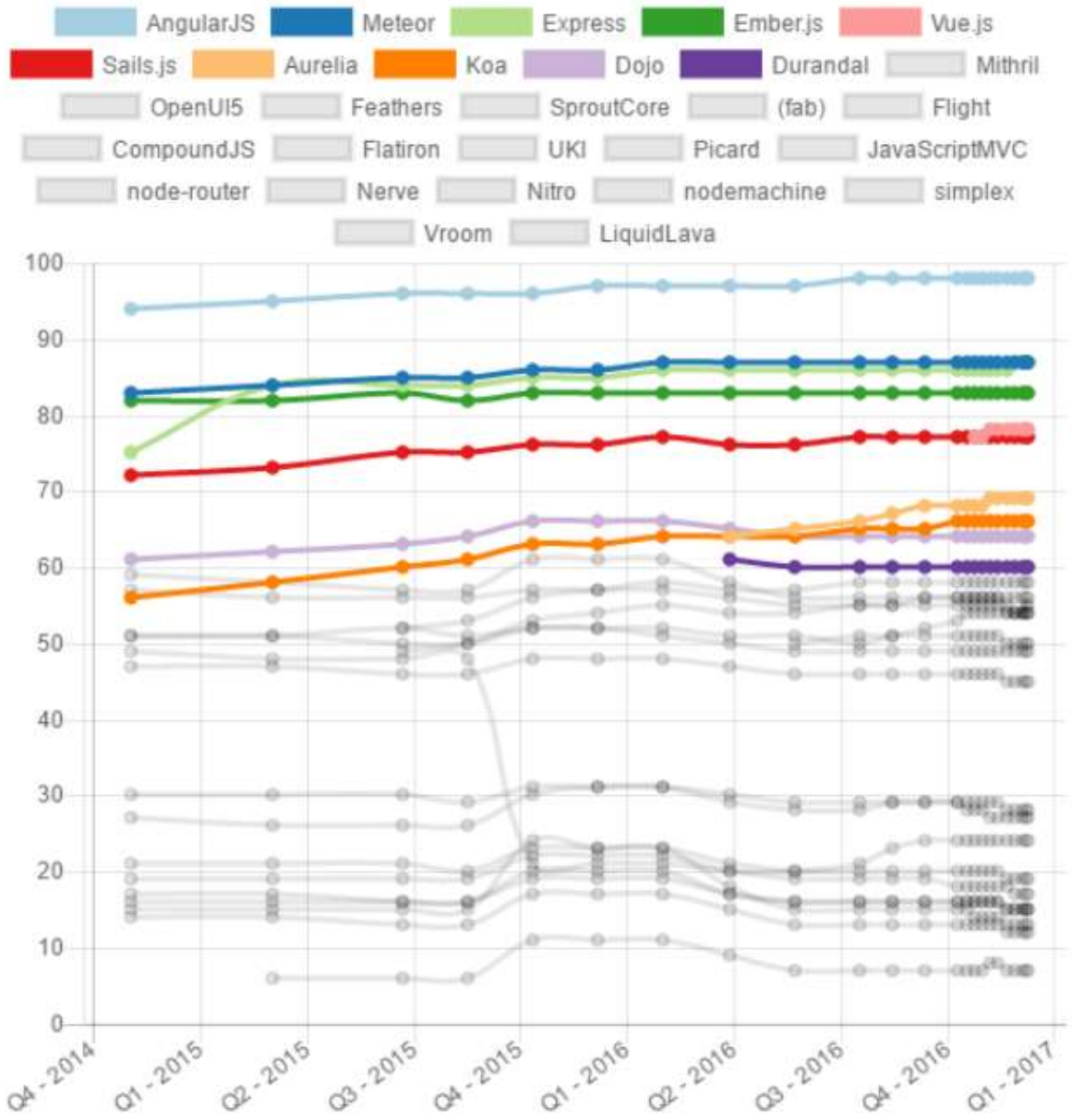
Node heeft op een relatief korte tijd een enorme community weten op te bouwen. NPM (Node Package Manager) is hier een voorbeeld van. NPM is zoals de naam het zegt een tool om packages te managen. Sinds de release in 2010 zijn er al over de 350.000 packages toegevoegd aan NPM. Dit is zeer veel vergeleken met bijvoorbeeld NuGet, een C# package manager dat sinds zijn release in 2010 slechts iets meer dan 70.000 packages wist te verzamelen.

Populariteit frameworks

Tabel 3: JavaScript frameworks gerangschikt op populariteit [16]

Framework	Score
AngularJS	98
Meteor	87
Express	87
Ember.js	83
Vue.js	78
Sails.js	77
Aurelia	69
Koa	66
Dojo	64
Durandal	60
Mithril	58
OpenUI5	56
Feathers	55
SproutCore	54
(fab)	54
Flight	50

CompoundJS	49
Flatiron	45
UKI	28
Picard	27
JavaScriptMVC	24
node-router	19
Nerve	17
Nitro	15
nodemachine	15
simplex	13
Vroom	12
LiquidLava	7



Figuur 18: Populariteit JavaScrip framerworks sinds kwart 4 -2014 [16]

Zoals te zien is in bovenstaande figuur en in *Tabel 3* heeft JavaScript een groot scala aan Frameworks. Doordat JavaScript van origine een frontend taal is, zijn het merendeel van de frameworks ook frontend frameworks met als populairste frontend framework Angular.js.

Net zoals bij de Java frameworks schiet deze vergelijking weer te kort. Zo is er nergens het framework/bibliotheek React te bespeuren. React is een frontend bibliotheek die zeer populair is en tegenwoordig zeker niet moet onderdoen voor andere populaire frameworks [17]. De afwezigheid van React is misschien nog te rechtvaardigen doordat de meesten React als een bibliotheek zien en niet als een framework.

Voor de backend zijn echter enkel de frameworks die gebruik maken van Node.js van belang. In de top vijf zijn dan Meteor, Express en Sails.js terug te vinden.

Om de vergelijking niet nodeloos lang te maken, wordt er enkel tussen deze drie frameworks een vergelijking gemaakt.

Frameworks

JavaScript frameworks zijn zeer divers, gaande van serverside, clientside tot mobiele app ontwikkeling. Onderstaande frameworks zijn enkele van de populairste die samenwerken met Node.js en het mogelijk maken web API's te ontwikkelen.

Meteor

Meteor is het populairste full stack framework voor JavaScript [18]. Meteor is eigenlijk meer dan een framework, het is een platform dat het mogelijk maakt om in een omgeving een web app te bouwen. Dit heeft als voordeel dat de ontwikkelaars kunnen focussen op de ontwikkeling van hun platform. Meteor maakt standaard gebruik van MongoDB. Het is mogelijk om via onofficiële packages gebruik te maken van andere database types zoals MySQL. Deze packages zijn echter amper in gebruik en meestal al meer dan een jaar niet meer geupdate. Daardoor ben je als ontwikkelaar eigenlijk genoodzaakt om gebruik te maken van MongoDB. Voor het ontwikkelen van het ZVO platform gaat de voorkeur naar een MySQL database *(zie 6.4 Database)* en wordt Meteor meteen een minder voor de hand liggende keuze.

Express

Express is een MVC framework dat veel vrijheid biedt op het vlak van configuratie mogelijkheden [18]. Het is veruit het populairste framework dat samen werkt met Node.js. Het biedt de vrijheid om [19]:

- eender welk type database te kiezen;
- elke vorm van user authenticatie te gebruiken;
- eender welke opbouw van de mappen structuur van je project te kiezen.

Met Express kan je alles doen wat je met Node.js ook kan, maar dan met minder code [19]. Het heeft van alle Node.js frameworks de grootste community en is volledig opensource. De keerzijde van al deze vrijheid is dat bij het opstarten van een Express project er nog geen basis is gelegd voor de structuur van je project. Sails.js, dat bovenop Express is gebouwd, maakt daarentegen wel zo'n structuur aan.

Sails.js

Sails.js is een robuust Node.js framework en biedt de mogelijkheid om alles te bouwen gaande van een kleine chat client tot een webapplicatie voor een groot bedrijf.

Sails maakt het makkelijk voor de gebruiker om meerdere databasetypen op een gelijkaardige manier aan te spreken. Deze module heet "Waterline" [20] en wordt nader besproken onder 5.6 *Database Sails Waterline*. Sails is ook geliefd door mensen met een voorkennis in Ruby, Django of Zend, vanwege de gelijkenissen met deze frameworks.

Sails is gebouwd op Node.js en maakt gebruik van Express om de http requests af te handelen.

Sails heeft als voordeel dat het meteen een basis legt voor de app. Bij het starten van een project heb je controllers, models, services en policies. Dit zorgt meteen voor structuur in het project en zet aan tot het maken van een overzichtelijk en makkelijk te onderhouden project. Natuurlijk is dit laatste afhankelijk van het kunnen van de ontwikkelaar die dit framework gebruikt. Door de basis die Sails.js boven op die van Express legt gaat de voorkeur uit naar Sails om te gebruiken als backend framework voor de ZVOapp. Door deze basis wordt de ontwikkelingstijd ingekort en de instapdrempel verlaagd.

C# (ASP.Net)

C# is net zoals Java aan bod gekomen in de masteropleiding van de Universiteit Hasselt en KU Leuven. Het wordt voornamelijk gebruikt voor het maken van desktopapplicaties en server applicaties. Qua syntax is C# sterk gelijkend aan Java. C# wordt voornamelijk gebruikt om applicaties te bouwen, zo wordt het door Unity en de Unreal Engine ondersteund voor het maken van games.

Populariteit frameworks

Tabel 4: C# frameworks gerangschikt op populariteit [21]

Framework	Score
ASP.NET	100
ASP.NET MVC	94
Nancy Pallet	64

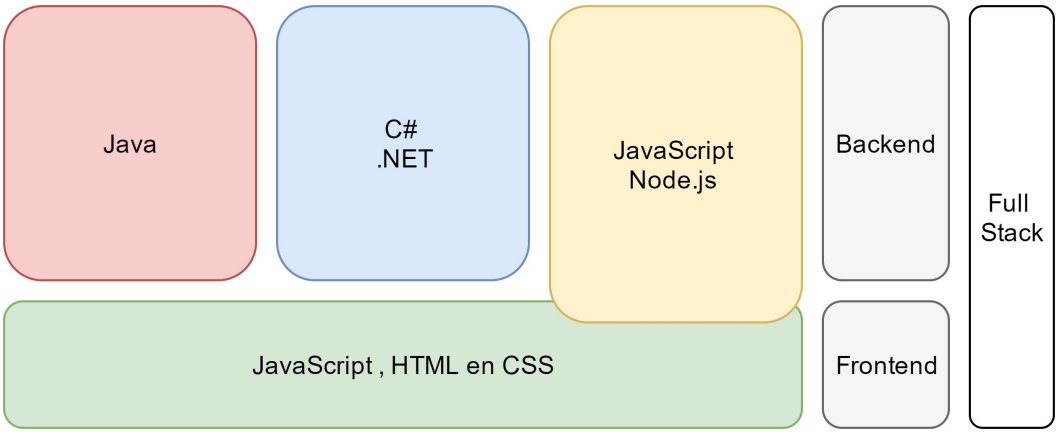
Zoals bovenstaande *Tabel 4* aantoont, is er bij C# niet veel keuze wat het framework betreft en wordt er meestal voor ASP.NET gekozen, waarbij ASP voor Active Server Pages staat. .NET is een webapplicatie framework dat afkomstig is van Microsoft en ook onderhouden wordt door Microsoft. Hiermee is het mogelijk om onder andere dynamische websites, webservices en webapplicaties te bouwen.

.NET applicaties kunnen naast C# ook in VisualBasic, F#, C++ en een resem andere talen gemaakt worden. Hoewel .NET onderhouden wordt door Microsoft is het een opensource framework en kan de .NET core ook uitgevoerd worden op Unix en MAC-besturingssystemen.

Voor het frontend gedeelte van de web app valt het op dat .NET zoals de meeste andere frameworks gebruik maakt van HTML5, CSS en JavaScript. Dit is nogmaals een bewijs dat een fullstack ontwikkelaar niet kan zonder kennis van JavaScript.

3.3.2. Conclusie

Uit bovenstaande vergelijking is nog maar eens gebleken dat er niet zoiets bestaat als een beste programmeertaal. Wel is gebleken dat afhankelijk van de eisen van een project de een al geschikter is als de andere. Het is ook duidelijk geworden dat in webontwikkeling haast geen ontkomen is aan JavaScript.



Figuur 19: Webontwikkeling stacks

Bovenstaande afbeelding laat zien hoe de meeste webapplicaties opgebouwd worden. Voor de ZVO backend is er uiteindelijk gekozen om met JavaScript en Node.js te werken, gebruik makend van het Sails framework. Doordat JavaScript, dankzij Node.js, zowel voor front- als backend gebruikt kan worden, wordt het mogelijk om het gehele project in dezelfde taal te ontwikkelen. Daarbovenop komt ook nog eens het open source karakter van Node.js en de duizenden bestaande modules (npm). Hoewel er uiteindelijk gekozen wordt voor Node.js wil dit niet zeggen dat het niet haalbaar is om dit project in een andere taal te ontwikkelen. Sterker nog een bedrijf dat beschikt over een team van ervaren .NET of Java ontwikkelaars kan bij een strakke deadline beter kiezen om hun vertrouwde taal en frameworks te gebruiken. Nu dat de knoop is doorgehakt welke programmeertaal gebruikt gaat worden voor de Backend gaan we in hoofdstuk *4.0 Cross platform* over tot het kiezen van een gepaste taal en framework voor het ontwikkelen van de ZVOapp.

4. Cross platform

Onder crossplatform verstaan we het ontwikkelen voor meerdere platformen vanuit dezelfde codebase in een crossplatform-ontwikkelingsomgeving. De tegenpool van crossplatform wordt native development genoemd en wil zeggen dat er gebruik wordt gemaakt van de programmeertaal en bijhorende tools die specifiek ontworpen zijn voor het gekozen platform bijvoorbeeld Android Studio (Java) voor Android.

4.1. Vergelijkingen en mogelijkheden

Crossplatform ontwikkeling maakt het gelijktijdig ontwikkelen op meerdere platformen mogelijk. Maar aangezien deze platformen niet 100% identiek zijn, gelden er tot een bepaald niveau voor ieder platform andere regels. Dit is voornamelijk van toepassing op het design van een applicatie. Zo gelden voor zowel Apple (iOS), Android als Windows Phone andere regels over hoe een applicatie er moet uitzien. Afhankelijk van de gekozen techniek zal een applicatie niet of volledig native aanvoelen. Bij mobiele applicaties kan crossplatform development onderverdeeld worden in drie groepen:

- Web,
- Hybrid,
- Crosscompiler.

Onderstaande tekst beschrijft deze nader.

4.1.1. Native vs. hybrid

Native apps zijn apps geschreven in de ontwikkelingsomgeving die ontworpen is voor een bepaald platform.
Hybrid apps zijn applicaties ontwikkeld met een tool die de mogelijkheid biedt op meerdere platformen tegelijkertijd te ontwikkelen. Deze tool maakt het mogelijk om zowel Android-, iOS- als Windowsapplicaties te schrijven in een programmeertaal.

4.1.2. Web app

Een web app is eigenlijk gewoon een site die geoptimaliseerd is voor gebruik op het kleine scherm van een smartphone. Aan de frontend is dit dan JavaScript en HTML. Web apps waren vooral populair in de beginjaren van crossplatform applicaties omdat ze gemakkelijk te maken waren. De web apps zijn immers niet meer dan een webpagina die oogt als een app. Om een web app weer te geven, wordt er gebruik gemaakt van de internetbrowser die op het toestel geïnstalleerd is. Hierdoor wordt de app afhankelijk van een andere app, namelijk de browser. Dit zorgt er voor dat de app moet inboeten in performance en minder soepel aanvoelt in vergelijking met een native applicatie. Daarnaast was het in het begin ook niet mogelijk om met een web app toegang te krijgen tot de hardware van het toestel. Tegenwoordig kan je wel al een aantal sensoren gebruiken [22]. Vooral de camera en GPS-sensor worden al veel gebruikt in mobiele sites. Ook op visueel vlak zien de apps er meestal anders uit dan native applicaties.

Dit laatste is niet echt een probleem en eerder afhankelijk van waar de ontwikkelaars naartoe willen op het gebied van design. Wat betreft de toepassing in de ZVOapp zou een webapp kunnen werken.

4.1.3. Hybrid

Hybrid apps overbruggen de grens tussen native en webapps en maken het mogelijk native apps te maken met webtechnologieën (HTML5, JavaScript en CSS) [23]. In plaats van op de browser zelf te draaien, maken hybrid apps gebruik van dezelfde engine als de browser. Door gebruik te maken van een web-to-native bridge wordt het mogelijk ook gebruik te maken van de hardware in het toestel, zoals de camera, microfoon, geheugenopslag, sensoren, … [23]. Voor de ZVOapp is het gebruik van een camera of specifieke sensoren geen vereiste, maar het houdt wel de weg naar uitbreidingen open. Naast toegang tot de hardware zorgt de bridge tussen web en native er ook voor dat de app kan communiceren met andere native apps zoals bijvoorbeeld de kalender. Dit is wel zeer interessant aangezien het bijhouden van een agenda een van de vereisten is voor de ZVOapp. Hierdoor is het mogelijk om via de hybrid app gegevens op te halen en dan de gebruiker toestemming te vragen om deze in hun native agenda app te steken i.e. Google Agenda voor Android. Hier komt bij kijken dat hoewel het mogelijk is om je hybrid te laten communiceren met een native component er soms beperkingen optreden in vergelijking met een echte native app. Om het bij de agenda te houden, op onderstaande afbeelding is te zien dat Android en iOS verschillen in de ondersteuning van bepaalde functies.

Tabel 5: Compatibiliteit iOS en Android, op native kalender Calendar-PhoneGap-Plugin [24]

Operation	Comment	iOS	Android
createCalendar		yes	yes
deleteCalendar		yes	yes
createEvent	silent	yes	yes (on Android < 4 dialog is shown)
createEventWithOptions	silent	yes	yes (on Android < 4 dialog is shown)
createEventInteractively	interactive	yes	yes
createEventInteractivelyWithOptions	interactive	yes	yes
findEvent		yes	yes
findEventWithOptions		yes	yes
listEventsInRange			yes
listCalendars		yes	yes
findAllEventsInNamedCalendars		yes	
modifyEvent		yes	
modifyEventWithOptions		yes	
deleteEvent		yes	yes
deleteEventFromNamedCalendar		yes	
openCalendar		yes	yes

In bovenstaande *Tabel 5* is te zien dat het op Android niet mogelijk is om agenda events aan te passen en dat dit op iOS wel kan. In dit geval is het natuurlijk wel op te lossen door op Android eerst een event te wissen en daarna het nieuwe event toe te voegen. Hoewel het probleem op te lossen is, moet hier al een onderscheid gemaakt worden in de programmatie van de Android en iOS app. In vergelijking met het maken van twee native apps met elk een volledig andere codebase valt dit kleine verschil in code tussen de Android en iOS app nog heel goed mee.

4.1.4. Crosscompile

Crosscompile zijn apps die geschreven worden in een andere taal en daarna omgezet worden naar native code. Dit zijn dus feitelijk echte native apps. De gebruiker zal geen verschil zien tussen een Crosscompile app en een volledig native ontwikkelde app. Een voorbeeld van een crosscompile platform is React native dat het mogelijk maakt om een native app in JavaScript te ontwikkelen. Waar web apps gebruik maken van HTML en CSS om views te maken, maakt een crosscompile gebruik van native componenten. Hierdoor wordt de app gebouwd met dezelfde bouwblokken als een echte native app.

In vergelijking met hybrid/web apps is de “Write once, run everywhere” filosofie niet meer van toepassing voor Crosscompile apps. De native componenten tussen Android en iOS zijn maar tot een bepaald niveau gelijkaardig. Hier moet tijdens de ontwikkeling rekening mee gehouden worden. Daarbovenop moet het crosscompile platform deze native componenten ook ondersteunen.

4.2. Populariteit

Hybrid apps voldoen aan al de eerder beschreven eisen en zijn dus geschikt om de ZVOapp mee te ontwikkelen. Er zijn echter net zoals bij de backend zeer veel opties. Daarom is er eerst onderzocht naar welke frameworks het meest besproken worden. Nadien wordt dan uit de populairste frameworks het meest geschikte voor de ontwikkeling van de ZVOapp gekozen. Zowel back- als frontend zullen in JavaScript ontwikkeld worden. Daarom wordt er enkel gekeken naar de frameworks die gebruik maken van JavaScript.

De crossplatformmarkt staat niet stil en er komen steeds meer en meer frameworks uit. De verschillen zijn soms miniem en het kiezen van een framework dat gaat blijven bestaan en groeien voelt meer aan als een gok dan een doordachte keuze. Er zijn echter enkele die al even bestaan en nog altijd lijken te groeien:

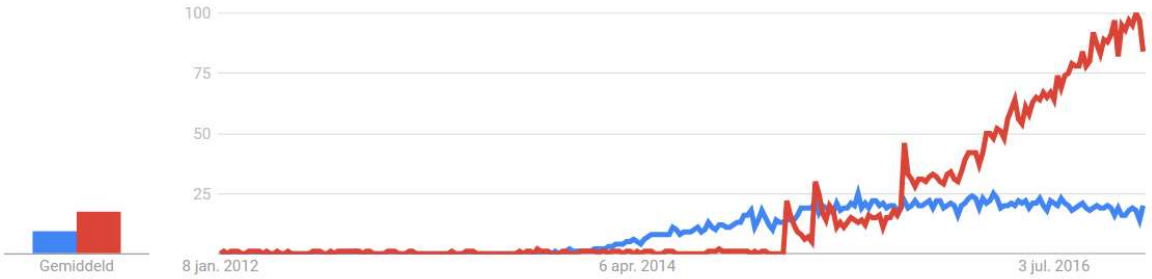
4.2.1. Ionic (Hybrid)

Ionic bestaat sinds 2013 en is dus nog relatief jong. Het legt voornamelijk de focus op de look en feel van een app en probeert als HTML5 SDK deze zo dicht mogelijk te laten aanleunen bij die van een native app. Hiervoor wordt gebruik gemaakt van AngularJS, een van de populairste frontend frameworks voor het maken van websites. Om toegang te krijgen tot de native componenten zoals bijvoorbeeld gps, gyro, kalender, … wordt gebruik gemaakt van Apache Cordova, PhoneGap of Trigger.io.

Ondertussen is er een preview versie van Ionic 2 die ook gebruik maakt van Angular 2. Deze zou in vergelijking met zijn voorganger het ontwerpproces moeten vereenvoudigen en de performance verhogen. Omdat de Ionic 2 nog in bèta was bij de ontwikkeling van de ZVOapp is er geen gebruik gemaakt van Ionic 2.

4.2.2. React Native (Crosscompile)

React Native is ontwikkeld door de ontwikkelaars van Facebook en wordt onder andere gebruikt voor het ontwikkelen van de Facebook en Instagram app. Zoals onderstaande grafiek van Google trends laat zien, is React native op korte tijd zeer populair geworden en heeft het zelfs Ionic weten in te halen.



Figuur 20: Populariteit zoekterm "Ionic framerwork" (blauw) VS "react native" (rood)

Zoals de naam al laat horen maakt React Native het mogelijk een native applicatie te bouwen. Om dit mogelijk te maken, maakt React gebruik van bridges naar native componenten. Voor de view maakt React geen gebruik van HTML5 en css maar van een zelf gemaakte taal JSX. JSX is een XML achtige extensie voor ECMAScript (de standaard voor scripting talen).

4.2.3. Conclusie

Bovenstaande, op JavaScript gebaseerde, crossplatform technieken zijn allen geschikt om de ZVOapp te realiseren. Ionic leek uiteindelijk het meest geschikt. Het is een relatief eenvoudig framework dat verder bouwt op de Angular.js userbase en modules. De apps zien er niet volledig native uit, maar Ionic weet dit goed genoeg te benaderen waardoor de ZVOapp er niet “ouderwets” uitziet. Het nadeel dat Ionic niet volledig native is, is langs de andere kant ook een voordeel. Hierdoor moet er tijdens de ontwikkeling geen rekening gehouden worden met platform specifieke eigenschappen. In het volgende hoofdstuk wordt er een vergelijking gemaakt tussen verschillende notificatie services die met de Ionic app kunnen communiceren.

5. Notificaties en alternatieven

Pushnotificaties, iedereen met een smartphone kent ze wel en heeft er al mee te maken gehad. Het zijn meldingen die een applicatie je stuurt om te laten weten dat er 'iets' gebeurd is of kan gaan gebeuren. Dat 'iets' is een event dat plaatsvindt binnen een applicatie, bijvoorbeeld iemand die je foto leuk vindt, je een berichtje stuurt of die vriend die je voor de zoveelste keer een uitnodiging stuurt om een spelletje te spelen. Kortom ze zorgen ervoor dat we bepaalde dingen niet vergeten of mislopen. Pushnotificaties zijn meestal handig, maar kunnen soms ook bij de gebruiker voor frustraties zorgen.

Voor de ZVOapp wordt er ook gebruik gemaakt van events. Onderstaande punten gaan dieper in op de toepassingsmogelijkheden en bekijken ook alternatieve paden om dit te realiseren.

5.1. Waarom notificaties

De ZVOapp gaat grotendeels draaien rond planning i.e. het delen van agenda's, melden van afwezigheden en het zoeken naar vervangers. Om dit te realiseren moet de gebruiker op de hoogte gehouden worden van veranderingen in zijn/haar planning. Je zou dit kunnen doen door een pagina in de app te voorzien die zegt wat er in de planning veranderd is. In een ideale wereld zou de gebruiker dan deze pagina elke dag bekijken om zeker te zijn dat hij geen veranderingen mist. De werkelijkheid is echter helemaal anders. Mensen hebben zo al veel aan hun hoofd en er zijn veel leukere dingen te doen dan continu op de app van je zwemclub te kijken. Er moet dus voor gezorgd worden dat we de gebruiker naar de applicatie lokken en hij daardoor ook meer met de applicatie betrokken raakt [25].

5.2. De mogelijkheden

5.2.1. Standaard push notificaties

De kleine balkjes of figuurtjes die meestal bovenaan je scherm verschijnen zijn de standaardvorm van notificaties.

Om dit te realiseren kan er een heel systeem gebouwd worden voor het afhandelen en zenden van notificaties naar mobiele apparaten en webbrowsers, of je kan gebruik maken van bestaande diensten zoals Onesignal. De eerste optie is echter zeer veel werk, om nog maar te zwijgen voor de bijkomende tijd en kosten om het systeem te onderhouden. Een beter alternatief is gebruik te maken van optie twee, dit kan tegen betaling of zelfs volledig gratis. Hierdoor kan er meer tijd en geld gestoken worden in het maken van de applicatie.

Enkele van deze diensten zijn:

Onesignal

Onesignal biedt de mogelijkheid notificaties te sturen naar, iOS, Android, Amazon Fire, Windows Phone 8.1, MacOS apps en Chrome apps. Kortom zo goed als alles wat een ontwikkelaar maar kan wensen. Daar bovenop is er ook nog eens SDK-ondersteuning voor 11 crossplatform frameworks, dit alles volledig gratis [26]. Het bedrijf beweert zijn geld te verdienen uit het verzamelen van data en aangepaste oplossingen te voorzien voor grote bedrijven [27]. Onesignal beschikt over duidelijke documentatie en een overzichtelijke web interface voor het zenden van berichten. Samen met de SDK-ondersteuning maakt het een geschikte kandidaat voor het gebruik in de ZVOapp.

Tabel 6: Onesignal

Pushservices	Kost	Platformen	Extra's
Onesignal	Gratis	iOS Android Windows Web	Zeer uitgebreide documentatie

Facebook pushnotificaties

Facebook biedt ook een dienst aan die het mogelijk maakt pushnotificaties te sturen via je eigen applicatie [28]. Dit staat los van het Facebook platform en gebruikers moeten dus geen Facebookaccount of de Facebook app hebben om hier gebruik van te kunnen maken. Het nadeel van Facebook pushnotificaties is dat er enkel een iOS en Android SDK zijn. Hierdoor is het niet mogelijk dit platform te integreren met crossplatform applicaties zoals Ionic of alleszins niet op een eenvoudige manier. Om de laatste reden is het beter om gebruik te maken van een alternatieve dienst. Tevens is de documentatie van Facebook ook beperkter dan die van de concurrentie.

Tabel 7: Facebook pushnotifications

Pushservices	Kost	Platformen	Extra's
Facebook pushnotificaties	Gratis	iOS Android	Ondersteuning voor afbeeldingen en GIF's

Ionic notificaties

Ionic biedt ook een eigen service aan om pushnotificaties te sturen. In combinatie met Ionic Auth [29] is het mogelijk berichten naar gebruikers te sturen op basis van hun gebruikerID, e-mailadres, FacebookID, … [30]. Andere services bieden meestal enkel de optie om berichten te sturen op basis van een token. Het is dan aan de gebruiker van deze services zelf om ervoor te zorgen dat de backend een gebruiker kan linken aan een token.

Ionic biedt ook een service aan om user login via Ionic af te handelen, genaamd Ionic Auth. In samenwerking met Ionic notificaties heeft dit als voordeel dat Ionic ook pushnotificaties naar gebruikers kan sturen op basis van hun gebruikerID of loginnaam. Dit is een pluspunt vergeleken met andere diensten waarbij het linken van users aan tokens zelf moet afgehandeld worden door de backend. Het nadeel is echter dat wanneer een server al een bestaand loginproces heeft, dit moet aangepast worden om met Ionic Auth te kunnen weken.

De nadelen en voordelen heffen elkaar uiteindelijk op waardoor de keuze voor Ionic notificaties eerder gaat liggen in de prijs en de ondersteuning naar andere platformen. En dit is nu net waar Ionic dan weer minder in scoort. Zo kunnen er maar tot 10.000 berichten gratis verzonden worden per maand. Deze 10.000 berichten zouden ruimschoots genoeg moeten zijn voor de ZVOapp, maar de ondersteuning voor andere platformen laat ook afweten. Als je voor Ionic push kiest, zit je als het ware vast aan Ionic en kan je enkel push notificaties naar een Ionic app verzenden.

Tabel 8: Ionic notificaties

Pushservices	**Kost**	**Platformen**	**Extra's**
Ionic notificaties	Gratis tot 10.000 berichten/maand	iOS Android (Phonegap)	Targetting opties zoals: Tokens userIDs Email Facebook Google Twitter Linkedin Github

Vergelijking

Tabel 9: Vergelijking pushnotificatie services

Pushservices	Kost	Platformen	Extra's
Onesignal	Gratis	iOS Android Windows Web	Zeer uitgebreide documentatie
Facebook pushnotificaties	Gratis	iOS Android	Ondersteuning voor afbeeldingen en GIF's
Ionic notificaties	Gratis tot 10.000 berichten/maand	iOS Android (Phonegap)	Targetting opties zoals: Tokens userIDs Email Facebook Google Twitter Linkedin Github

5.2.2. Notificaties via andere apps

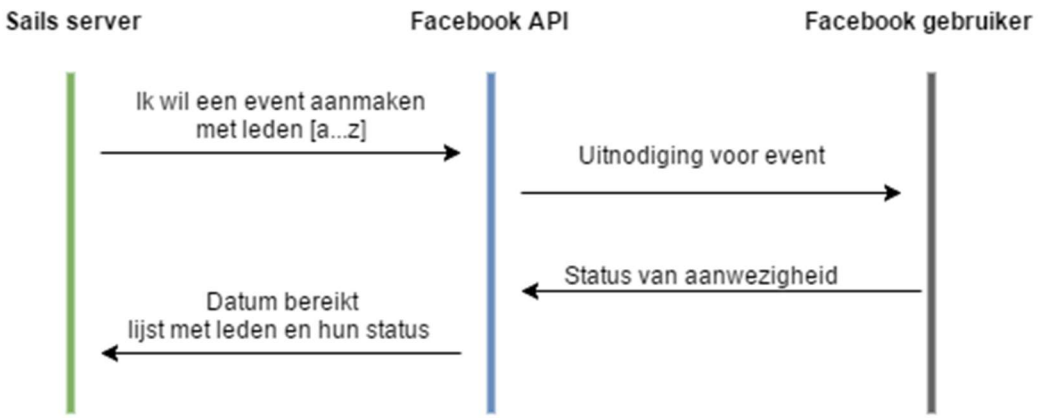
Applicaties zoals Facebook, WhatsApp, Twitter en consoorten slagen er in een zeer breed publiek aan te spreken en worden op regelmatige basis gebruikt. Het leek dan ook interessant te kijken in welke mate de gebruiker, via deze platformen, op de hoogte kan gehouden worden van wijzigingen in het ZVO-platform.

Facebook

Facebook heeft een zeer uitgebreide API en maakt het zelfs mogelijk om gebruikers in je app of website te laten inloggen door middel van hun Facebookaccount.

De API gaat echter veel verder dan het linken van een Facebookaccount aan een andere dienst. Het beschikt over allerhande interfaces om gegevens uit de gebruiker zijn account te halen, of in naam van de gebruiker content te delen op Facebook.

Facebook beschikt ook over events. Zo'n event kan aangemaakt worden door een gebruiker en hier kunnen dan andere gebruikers laten weten of ze dan wel of niet gaan deelnemen aan het event. Hieruit ontstond de volgende denkpiste, deze bestond er uit het systeem aan de server te koppelen en gebruikers uit te nodigen voor events. Onderstaande *Figuur 21* zorgt voor meer duidelijkheid.



Figuur 21: Denkpiste Facebook events

Figuur 21 laat zien hoe de server een request naar de Facebook API stuurt om een event aan te maken en hier enkele leden aan toe te voegen. De Facebookserver zou dan de betreffende leden uitnodigen voor het event en de leden zouden hierop reageren en laten weten of ze wel of niet aanwezig zullen zijn. Dit zou dan door de Facebook API worden doorgegeven aan de backend (Sails server). Deze kan op zijn beurt de gegevens dan verwerken. Dit zou een mogelijke oplossing zijn geweest om vervangers te vragen of ze kunnen invallen voor de afwezige trainer. Deze denkpiste is echter niet meer te realiseren omdat Facebook spijtig genoeg niet meer toelaat events aan te maken door middel van hun Graph API [31]. Het is wel nog mogelijk een event uit te lezen, maar dit heeft geen meerwaarde in combinatie met de ZVOapp. Na het doorlezen van de Graph API is ook gebleken dat Facebook via deze API voornamelijk de nadruk legt op het uitlezen van gegevens van een bestaand Facebookaccount en dus niet de mogelijkheid biedt te communiceren met een gebruiker via zijn eigen account.

5.3. Conclusie

Als notificatie service steekt Onesignal, dankzij zijn uitgebreide support en documentatie er met kop en schouders bovenuit. Alternatieven die wel in de buurt komen, worden dan weer van de troon gestoten omwille van de prijs. Naast gewone pushnotificaties kunnen gebruikers ook nog op een andere manier notificaties ontvangen *(zie 7.2 Bots).* In het volgende hoofdstukken gaan we over op de toepassing van alle tools die in voorgaande hoofstukken zijn beschreven.

6. Backend

In dit hoofdstuk wordt de algemene werking van de backend beschreven en de onderdelen en services die nauw in verbinding staan met de backend. Zoals de databse en de Google Calendar service.

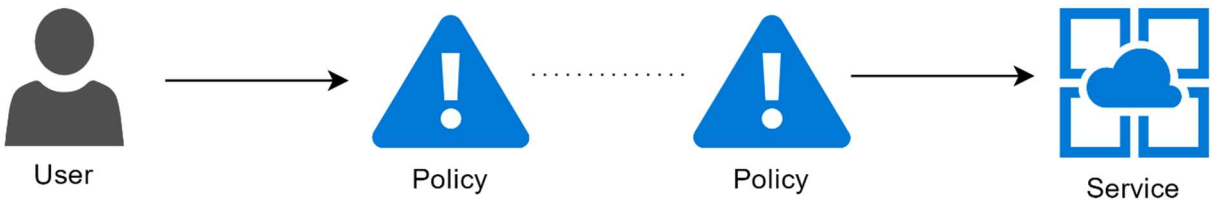
6.1. Security

Om de communicatie tussen gebruikers en het platform veilig te doen verlopen zijn de volgende maatregelen genomen.

6.1.1. Sails Policies

Policies maken het mogelijk om voorwaardes op te leggen waaraan voldaan moet worden voordat een request toegang krijgt tot een controller of een bepaalde functie van een controller.

Een voorbeeld van een veel gebruikte policy is gebruikersauthenticatie. Als een gebruiker een service wil aanspreken dan moet er eerst gecontroleerd worden of deze gebruiker een geldige gebruiker is. Als dit niet is, wordt de request verworpen. De policies zijn terug te vinden onder config/policies.js.



Figuur 22: het verband tussen user, policies en een service

Onderstaande code is een voorbeeld van hoe een policy eruitziet wanneer een service enkel aangesproken mag worden door een gebruiker die ingelogd is en admin rechten heeft.

```
AanwezigheidController:{
    getAanwezigheidslijst: ['authToken','admin']
}
```

Voordat de functie getAanwezigheidslijst wordt uitgevoerd, zal er eerst aan ‘authToken’ voldaan moeten worden en daarna aan de ‘admin’ policies. ‘authToken’ en ‘admin’ zijn terug te vinden onder api/policies.

Onderstaande code is een voorbeeld van hoe de admin policy er zou kunnen uitzien:

```
var requireAdminError = [{name: 'requireAdminError', message: 'Administrator
rechten nodig!'}]
  User.findOne({id: req.tokenID})
    .populateAll()
    .exec(function (err, response) {
      if (response) {
        if( response.isAdmin() === true ) {
            console.log('Admin true');
            next();
          }
        else{
            res.json(401, requireAdminError);
          }
        }

      if(err){
        res.json(401, err);
      }
    })
```

Wanneer de rol van de gebruiker admin is, zal er overgegaan worden naar de service of een volgende policy. In alle andere gevallen zal er een '401 Unauthorized' naar de client gestuurd worden.

6.1.2. Bot Authenticatie

Vooraleer een gebruiker toegang kan krijgen tot zijn ZVO-profiel en bijhorende functies, dient hij eerst herkent te worden door het systeem. Om dit op te lossen, is er voor elke chatbot een authenticatie proces ontworpen. Dit proces kent gelijkenissen met dat wat voor de Ionic app gebruikt wordt. Het grootste verschil is echter dat met de Ionic app bij elke request een token verzonden moet worden. Met de bot is dit enkel nodig bij de authenticatie van de gebruiker. Een bericht afkomstig van een bot wordt standaard al voorzien van een vooraf opgestelde token. Deze token is platform afhankelijk en enkel gekend door de backend en het betreffende platform, bijvoorbeeld Fb Messenger en Telegram.

Om een gebruiker te herkennen moet hij eerst via de bot kunnen inloggen. Dit zou kunnen opgelost worden door de gebruiker via de chat te vragen wat zijn gebruikersnaam en paswoord zijn. Het zou werken, maar is verre van ideaal. Inloggegevens zouden in de chat blijven staan en door iedereen die toegang heeft tot het chatgesprek gelezen kunnen worden. In plaats daarvan is gekozen om op dezelfde wijze te werk te gaan als bij een app of website. Door de gebruiker een link te sturen naar een beveiligde pagina kan hij op een veilige manier inloggen.

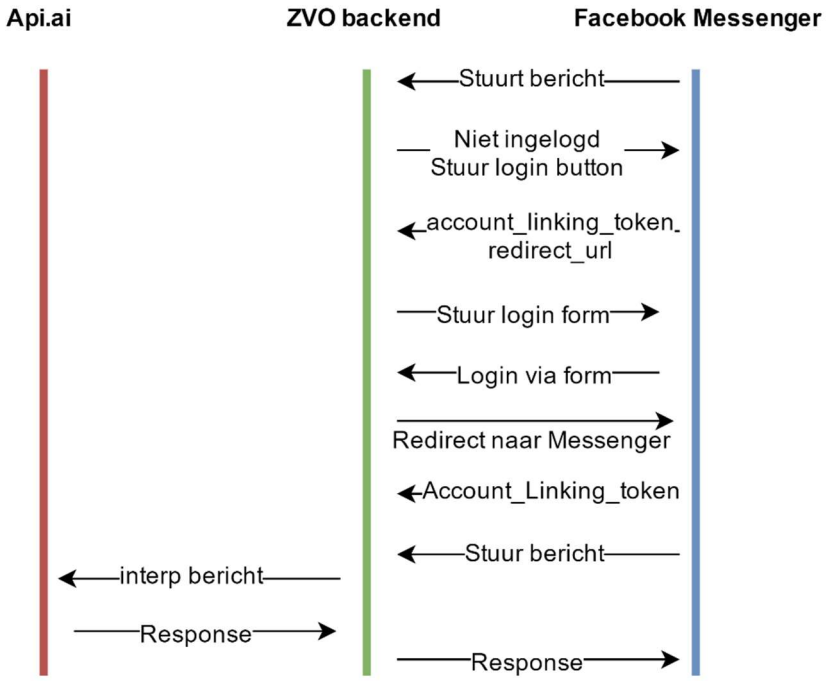
Door dit loginproces voldoen de integraties van Api.ai niet meer en moest er voor elke chat dienst een afzonderlijke webhook op de backend voorzien worden. Deze webhook zorgt ervoor dat chatberichten eerst naar de backend gestuurd zullen worden. De backend zal dan op zijn beurt gebruik maken van Api.ai om de berichten te interpreteren.

Indien een chat gebruiker nog niet voorkomt in de database zal hij worden gevraagd om eerst in te loggen. Dit inlogproces is platformafhankelijk en wordt in onderstaande paragrafen besproken.

De communicatie tussen Api.ai en de backend wordt beveiligd door middel van een token dat enkel gekend is tussen Api.ai en de backend. De integratie van API.ai wordt nader besproken in 7.*2.4 Bot uitwerking.*

Facebook Messenger

Onderstaande *Figuur 23* verduidelijkt de relatie tussen Facebook Messenger, de backend en Api.ai bij het verifiëren van de gebruiker.



Figuur 23: Implementatie Messenger Account Linking

Wanneer de Messenger gebruiker een bericht stuurt naar de bot wordt er eerst gekeken of deze ingelogd is. Dit gebeurt in een Sails policy die nakijkt of het FacebookID al aanwezig is in de database. Indien dit niet het geval is, zal het Facebook Account Linking proces gestart worden.

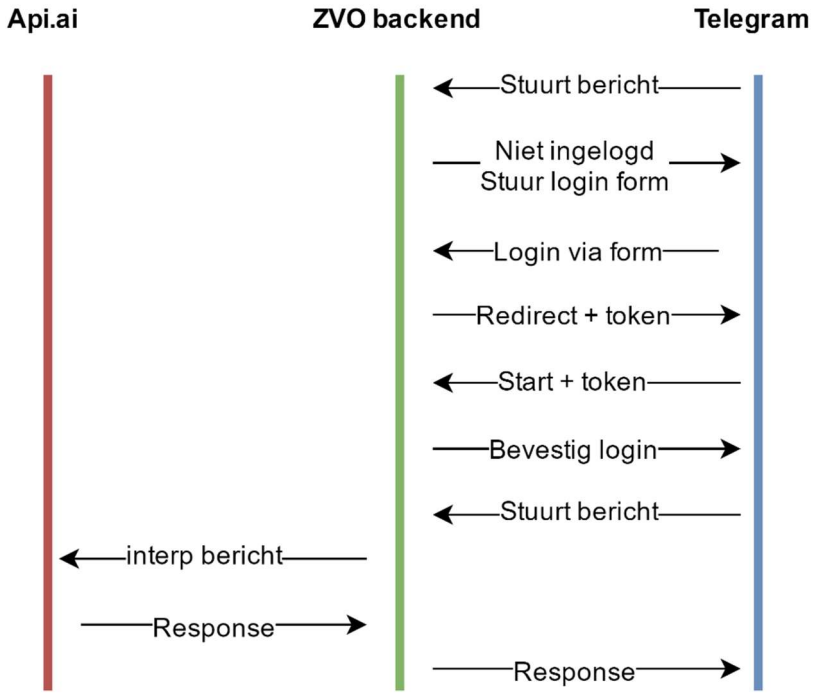
Dit houdt het volgende in:

- de backend stuurt een button naar de Messenger gebruiker met als type “Account_link” en als redirect url een link naar de backend;
- de Facebookserver herkent deze knop als een Account Linking knop, en stuurt bij het indrukken van de knop een “account_linking_token” en een “redirect_url” naar de link achter de knop;
- de server zal daarop reageren en een login form terugsturen naar de Messenger gebruiker;
- de Messenger gebruiker kan nu inloggen met zijn ZVO-accountgegevens;

- als de gebruiker de juiste gegevens heeft ingegeven, zal de server dit bevestigen door de eerder verkregen redirect url met een authorization_code terug te sturen naar Facebook. Deze authorization_code is een parameter die bij een succesvolle login wordt verwacht door de Facebook-server en de Messenger gebruiker terug zal brengen naar Messenger.
  De authorization_code bestaat uit een token met daarin het ID, de naam en de rol van desbetreffende gebruiker;
- Facebook zal nu naar de webhook een Account_linking callback sturen met als parameters, “Status”: “linked” en
  “authorization_code”: ”Token_met_daarin_ID_Naam_Rol”;
- de backend zal deze callback opvangen in een Policy en weet nu dat ook Facebook de gebruiker heeft aanvaard. Het FacebookID zal nu opgeslagen worden in de database en zo dus ook gelinkt worden aan het ID in de meegeleverde token.

Telegram

In tegenstelling tot Messenger heeft Telegram geen standaardoplossing voor het linken van externe gebruikers aan de bot. In de plaats hiervan heeft Telegram een “Deep Linking” functionaliteit. Dit maakt het mogelijk door middel van een link met extra parameters de gebruiker naar de bot te leiden. Deze extra parameter kan dan gebruikt worden om informatie van een externe bron naar de bot te sturen. Het volgende schema laat zien hoe Deep Linking wordt toegepast om de gebruiker te identificeren:



Figuur 24: Login proces Telegram

Het volledige Telegram login proces betreft het volgende:

1. de gebruiker probeert een bericht te sturen naar de bot.
2. Telegram stuurt dit bericht naar de webhook van de ZVO-backend.
3. vooraleer de gebruiker zijn bericht wordt verwerkt, wordt er door middel van een Sails policy nagegaan of het ID van de Telegram gebruiker al voorkomt in de database.
4. het ID wordt niet herkend en het authenticatie proces wordt gestart.
5. de backend stuurt een bericht naar de gebruiker met de vraag om in te loggen en een bijhorende "login" knop die de gebruiker naar een login form brengt.
6. de gebruiker probeert nu in te loggen en stuurt zijn gegevens naar de ZVO-backend door op "Login" te klikken.
7. indien de gegevens correct zijn wordt er voor de gebruiker een token gegenereerd.
8. de token wordt in de database opgeslagen samen met het ID van de gebruiker. *Omdat Telegram het niet toelaat "." te gebruiken in een parameter, is het niet mogelijk te werken met een JWT als token. In een JWT zitten namelijk twee punten, deze zorgen voor een onderscheid tussen de Header, Payload en Signature. In de plaats daarvan wordt er een random string gegenereerd met een lengte van 32 karakters. Deze string die de token voorstelt is random en daarom dus niet echt uniek, in principe zou er moeten gecontroleerd worden of deze al in de database voorkomt. Maar aangezien de token uit 32 karakters bestaat en voor elk karakter 36 mogelijkheden zijn, is de kans dat een token zich voordoet 1/(36^32). Deze kans is zo klein dat het overbodig wordt dit te controleren.*
9. er wordt nu een redirect link naar de Telegram gebruiker gestuurd (Deep Linking) aan deze link wordt de parameter "Start", met als waarde de token, toegevoegd.
10. de gebruiker wordt terug naar het gesprek met de bot gebracht en kan nu onderaan de chat op "Start" klikken.
11. door op start te klikken, wordt de token opnieuw naar de ZVO-backend gestuurd en wordt op basis van de token het TelegramID toegevoegd aan de database. De gebruiker wordt vanaf nu herkent door het systeem en krijgt de melding dat hij succesvol is ingelogd.

6.2. Hosting

6.2.1. Web Hosting

Om de backend naar de buitenwereld beschikbaar te maken, moet er gebruik gemaakt worden van een hosting service. Aangezien de backend draait op Node.js moet de hosting service dit ook ondersteunen.

In onderstaande tabel worden enkel Node.js hosting platformen vergeleken op basis van:

- prijs van het goedkoopste pakket om de backend te kunnen testen en de prijs voor definitieve hosting;
- locatie, van de servers;
- extra services die worden aangeboden.

Tabel 10: Vergelijking PaaS services

	Locatie	Prijs	Extras
Heroku	EU, US , Toky Japan	**Testen:** gratis voor 1 worker met 512 MB RAM, slaapt na 30 min inactiviteit. **Final:** Zou misschien kunnen lukken met een hobby Account van 7 Dolar per Dyno en 512 MB RAM.	Deployen via git, Heroku git of dropbox. Addons van Heroku zelf of externe services zoals ,database, Search, Monitoring, Security, …
OpenShift	US, Canada, EU, Israel en Rusland	**Testen:** Bronze plan met 3 gears, 1GB oplsag per gear, SSL **Final:** Bronze plan, maar bij betalen voor extra gears (€ 0,02 per gear/uur) en opslag (€ 1 /per GB elke maand)	Deployen via git push. Toevoegen meerdere database types waaronder MySQL, MongoDB, PostrgresSQL
Xervo		geen gratis optie miniumum specs: 512<MB RAM , 1 servo, 1 GB database voor $ 33,80 per maand	Opslag is een geintegreerde MongoDB.

Conclusie

Uiteindelijk is er gekozen om gebruik te maken van Heroku. Het grootste voordeel van Heroku is dat het development pakket volledig gratis is en dat zonder een gebruiksperiode op te leggen. Hierbij komt wel de downtime, maar dit is meer dan voldoende om testen te kunnen uitvoeren. Daarnaast zijn er ook nog de talloze addons zoals Database addons die meestal ook over een gratis optie beschikken, SSL en HTTPS support en servers gelegen in Europa. Voor de definitieve hosting kan er nog gekozen worden voor een ander platform, maar met een maximum van 14 Euro per maand is ook hier Heroku een goede oplossing.

6.2.2. Local Hosting

Tijdens het ontwikkelen werd eerst gebruik gemaakt van een localhost. Dit was voldoende om de ZVOapp te testen op de backend.

Het lokaal testen van bots was echter niet mogelijk omdat bots geen callbacks kunnen sturen naar lokaal gehoste webhooks. In het begin leek de enige optie de code te pushen naar Heroku en zo de bot en de code te testen. Doordat de Heroku server iedere keer de git push moest verwerken en heropstarten, was dit zeer tijdrovend en daardoor een inefficiënte manier van werken. Daarbovenop moest dit bij de minste fout in de code herhaald worden en zat het totaalaantal deploys op de backend al snel over de 200.

Om bovenstaand probleem op te lossen is er gebruik gemaakt van de npm module Localtunnel [32]. Deze module maakt het mogelijk de localhost van buitenaf te bereiken, door een public URL te hosten. Deze URL moet dan ook gedeeld worden met Telegram en Messenger zodat de bots het nieuwe adres van de webhook ook kennen. Hoewel de webhook adressen voor de bots moeten aangepast worden, is dit nog altijd sneller dan een deploy op Heroku. Zolang de tunnel blijft werken moeten de URL's niet aangepast worden. Spijtig genoeg crasht de Localtunnel nogal regelmatig en moet dit dus wel gedaan worden. Blijkbaar hebben ook anderen last van dit probleem [33]. Als dit probleem blijft aanhouden is ngrok [34] een alternatief.

6.3. Google agenda integratie

Het beheren van een club is haast niet mogelijk zonder een goede planning. Het was daarom ook een van de vereisten om een planning/agenda systeem te integreren in het ZVO platform. Dit kan gerealiseerd worden door een bijpassende database aan te maken. Om deze database dan aan te spreken moeten er de nodige functies op de backend toegevoegd worden voor het interpreteren en wijzigen van de agenda data. Daarbovenop moet er dan ook een view (frontend) gemaakt worden om de functies op de backend aan te spreken. Kortom een hoop werk die misschien grotendeels uitbesteed kan worden aan een andere service.

6.3.1. Agenda API

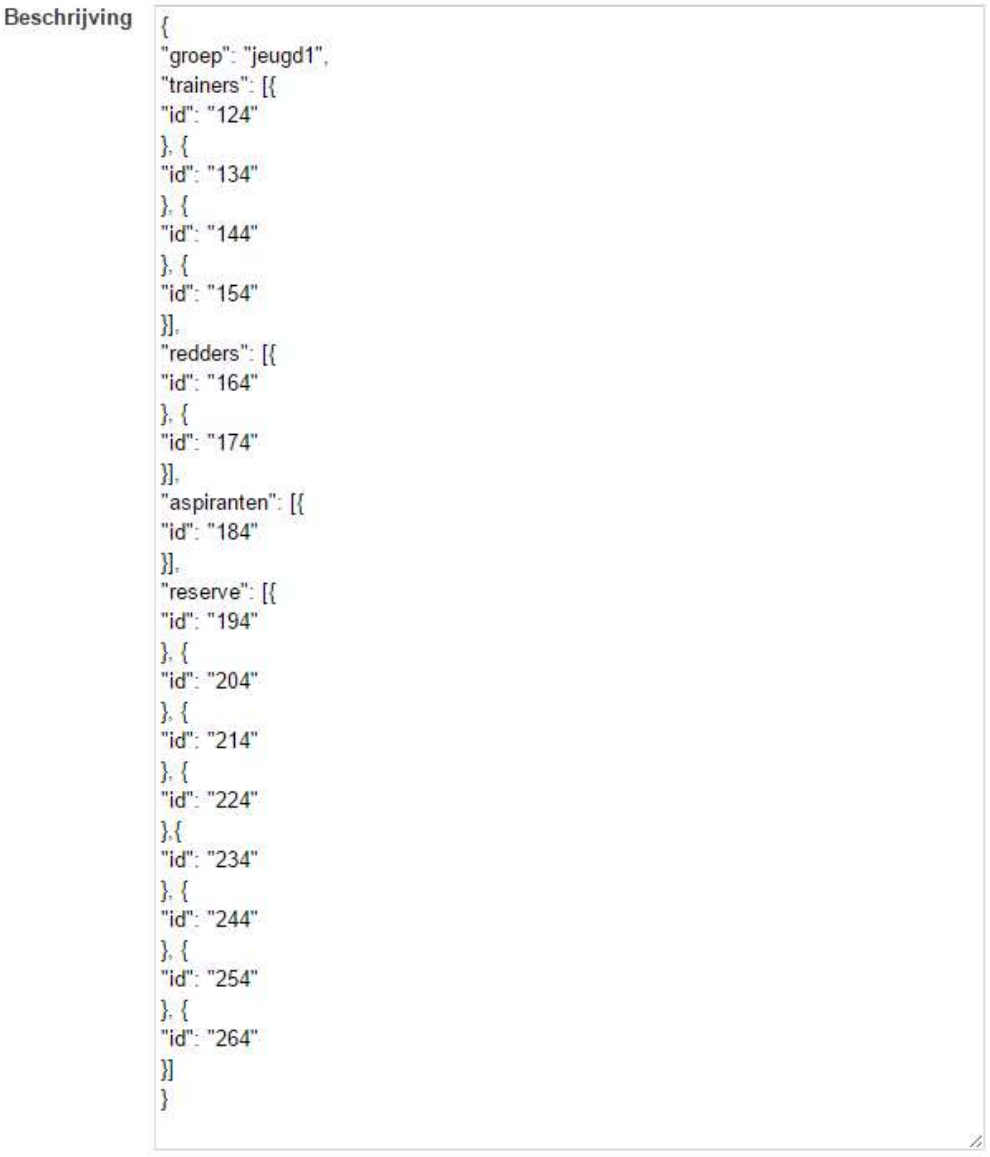
Als agenda service/API-aanbieder is gekozen om gebruik te maken van de Google Calendar (Google Agenda) API. Deze is de meest uitgebreide en best gedocumenteerde agenda service die er te vinden is en deze is ook gratis te gebruiken.

Google Agenda heeft als voordeel dat het voor iedereen beschikbaar en eenvoudig in gebruik is. Hierdoor kunnen bestuursleden makkelijk van op hun computer of gsm de algemene agenda beheren. Deze agenda is echter een algemene agenda en geeft gebruikers nog altijd niet de mogelijkheid een persoonlijke agenda op te vragen. Om dit op te lossen is er gebruik gemaakt van de Google Calendar API. Deze API maakt het mogelijk alle events in de agenda te doorzoeken op basis van bepaalde zoektermen. Om deze zoekfunctie te kunnen gebruiken, moet eerst de gebruiker gelinkt kunnen worden aan het event. Google biedt de optie aan om deelnemers aan een event toe te voegen op basis van hun e-mailadres. Hoewel dit de meest voor de hand liggende optie lijkt, komen hier enkele problemen bij kijken. Zo zou het niet meer mogelijk zijn om enkel op basis van het event te weten wat de rol van de gebruiker is. Stel dat een gebruiker op sommige dagen als redder opstaat en andere als trainer.

Dan kan de backend niet uit het event afleiden welke van de twee nu bij het betreffende event hoort. De backend kan dan enkel aan Google vragen of het e-mailadres van de gebruiker voorkomt in een bepaald event.

Door het gebruik van e-mailadressen ontstaat er een tweede probleem. Het gebruik hiervan is enkel toe te passen indien de e-mailadressen uniek zijn. Dit wil zeggen elke gebruiker heeft zijn eigen e-mailadres. Dit geeft problemen bij gezinnen die lid zijn van de club en jonge kinderen hebben die zelf nog geen e-mailadres hebben.

De optie om e-mailadressen te gebruiken was dus uitgesloten en daarom is er naar een alternatief gezocht. De oplossing bestaat er uit de “event description” te voorzien van alle IDs van de deelnemende leden. Hierdoor kan er gezocht worden op basis van gebruikersID en hoeft de backend enkel een request naar Google te sturen met als search parameter het ID van de desbetreffende gebruiker. Google zal dan een response met alle events terugsturen naar de backend. De structuur van zo een event-beschrijving ziet er als volgt uit:



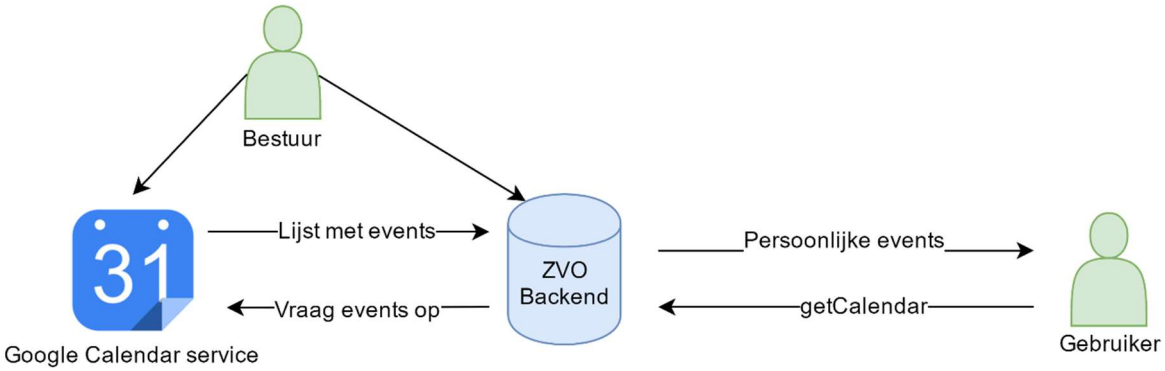
Figuur 25: Voorbeeld van een Google agenda event beschrijving

De structuur maakt gebruik van de JSON syntax, wat het mogelijk maakt voor de backend om de beschrijving als een object uit te lezen. Dankzij het uitlezen als een object wordt het makkelijker voor de backend (en vooral de programmeur) om de inhoud van de beschrijving te interpreteren. De backend aanziet het object namelijk als een array, dit maakt het mogelijk het object te doorlopen door middel van een filter.

Zo een filterfunctie maakt het bijvoorbeeld mogelijk om na te kijken wat de rol van de gebruiker is in het betreffende agenda event. Het kunnen controleren van de rol zorgt ervoor dat de rol opgenomen kan worden in de persoonlijke agenda. Hierdoor is het voor de gebruiker meteen duidelijk wat zijn taak is op het betreffende event.

Deze events worden dan door middel van een JavaScript module 'ical-generator' omgezet in het .ICS formaat. ICS is een bestandstype dat het mogelijk maakt om kalender data te delen met andere gebruikers en applicaties. Het ZVO platform maakt hiervan gebruik om persoonlijke agenda's aan te maken en op te slaan als tekst in de database. Wanneer een gebruiker zijn agenda opvraagt zal deze tekst naar de gebruiker gestuurd worden in de .ICS-bestandsindeling. Dit bestand kan de gebruiker inladen in de meeste agenda applicaties, waaronder Google Agenda, Apple Calendar, Microsoft Outlook, … De gebruiker moet hier geen weet van hebben. Wanneer hij het bestand downloadt zal het besturingssysteem i.e. Android, Windows, iOS, … voorstellen het te openen met een aanwezige agenda applicatie. Dit leunt aan bij de vereiste om de instapdrempel en gebruikservaring zo eenvoudig mogelijk te houden.

Onderstaande illustratie verduidelijkt het verband tussen de Google API en de gebruikers van het platform.



Figuur 26: Illustreert het verband tussen gebruikers en Google Calendar service

6.4. Database

In dit hoofdstuk wordt dieper ingegaan op de gekozen database en de gebruikte ORM en de voor- en nadelen die hierbij komen kijken.

6.4.1. MySQL

Voor het ZVO platform bestaat de database voornamelijk uit gebruikers en gegevens in relatie tot die gebruiker i.e. aanwezigheidslijsten, rapporten, profielen, …

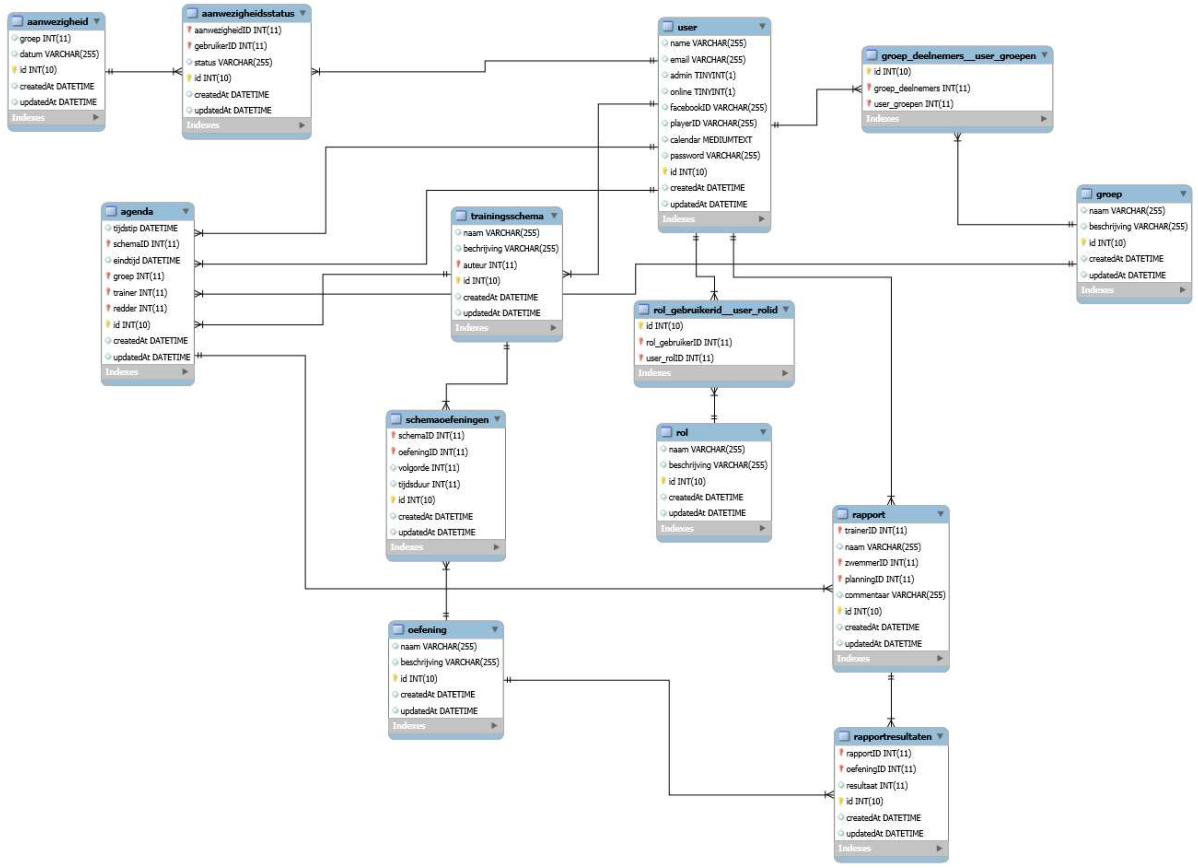
Vanuit dit standpunt is er gekozen om gebruik te maken van een relationele database, meer bepaald een MySQL database. Een bijkomend voordeel van MySQL is dat de reeds bestaande database ook een MySQL database was. Dit vergemakkelijkt indien nodig het samenvoegen van de bestaande en de nieuwe database.

Om in te zien waarom er gekozen is voor een MySQL database, nemen we even de gebruikers en hun rollen. Een rol kan toebehoren tot meerdere gebruikers, maar een gebruiker kan ook meerdere rollen hebben. Dit wordt een many-to-many relatie genoemd. Door een relationele database wordt het mogelijk om queries te maken die alle rollen van een bepaalde gebruiker of alle gebruikers met een bepaalde rol geven. Hoewel many-to-many relaties ook mogelijk zijn in een niet relationele database als MongoDB, verkiezen we toch SQL omdat dit daar heel goed in is.

6.4.2. Ontwerp

Bij het ontwerpen van een MySQL database is het gebruikelijker eerst een schema te maken van de database. Nadien wordt dan een systeem gebouwd dat ervoor zorgt dat de gewenste handelingen kunnen gedaan worden met de database. Wanneer er gebruik gemaakt wordt van een MongoDB is het gebruikelijker de software te schrijven en wordt het tijdens het schrijven duidelijk hoe de database er uit gaat zien. Tijdens de ontwikkeling van de ZVO-backend is er, hoewel deze gebruik maakt van een MySQL database, gebruik gemaakt van de tweede werkwijze. Deze werkwijze was dankzij de ORM ook goed toepasbaar op een MySQL database. Daarbovenop was het bij de start van het ontwikkelingsproces niet helemaal duidelijk welke functionaliteiten effectief ingebouwd moesten worden.

Uiteindelijk heeft de database de volgende structuur gekregen:



Figuur 27: Schema MySQL database

Hoewel bovenstaande *Figuur 27* niet scherp genoeg is om alles in detail weer te geven, volstaat het om er het volgende uit af te leiden. In het schema is links de Agenda tabel te zien waar vijf pijlen naartoe lopen. Deze tabel was aangemaakt voordat er gekozen was om Google Calendar te implementeren. De vijf pijlen duiden op het aantal relaties dat de tabel heeft met andere tabellen. Door de implementatie van Google Calendar maakt de backend niet langer gebruik van de Agenda tabel. Deze kan dus eigenlijk met al zijn verbindingen uit het schema gewist worden. Dit is nog niet gedaan, voor het geval dat na verloop van tijd de voorkeur toch naar een agenda in de database gaat.

Om bovenstaand schema te genereren, is gebruik gemaakt van de “Reverse engineer” functie van MySQL Workbench. Hierbij is duidelijk geworden dat het enkel mogelijk was de modellen van de tabellen te genereren.

Het automatisch genereren van de verbindingen was niet mogelijk omdat de database niet wist welke elementen foreign keys waren. Hieruit ontstaat de conclusie dat Waterline geen foreign keys aanmaakt bij het aanmaken van join tabellen.

6.5. Sails Waterline

Is de Object-relational mapping of kort ORM die standaard bij het Sails framework wordt geïnstalleerd. Een ORM maakt het mogelijk om een database te querien door de data aan te spreken als een object.

Hierdoor hoeft de programmeur geen queries meer te schrijven en kan de database aangesproken worden in de taal die ook gebruikt wordt voor de rest van het project. Door de data als model te zien zet het de gebruiker aan om de code in de MVC vorm te schrijven, waardoor de code over het algemeen overzichtelijker wordt. Niet dat MVC een garantie is voor mooie code, maar indien juist geïmplementeerd heeft het zeker een meerwaarde.

Afhankelijk van de specificaties van de ORM kunnen er verschillende database types aangesproken worden. Voor het aanspreken van zo'n database maakt Waterline gebruik van adapters. Zo een adapter zet de gebruikte methodes om in een query die door de database begrepen kan worden. Waterline biedt adapters aan voor de volgende datastores [35]:

- PostgreSQL,
- MySQL,
- MongoDB,
- Memory,
- Disk,
- Microsoft SQL Server,
- Redis,
- Riak,
- IRC,
- Twitter,
- JSDom,
- Neo4j,
- OrientDB,
- ArangoDB,
- Apache Cassandra,
- GraphQL,
- Solr,
- Apache Derby.

Waterline laat het toe meerdere adapters te gebruiken en deze aan elkaar te linken. Een gebruikers account kan bijvoorbeeld opgeslagen worden in MySQL en de afbeeldingen die hij upload in een MongoDB. Waterline zal dan zorgen voor de relatie tussen de gebruiker en zijn foto's.

6.5.1. Tekortkomingen

Hoewel Waterline een meerwaarde was voor het project waren er ook problemen met de ORM enkele hiervan waren:

Many-to-many

Bij het aanmaken van een many-to-many relatie is het niet mogelijk tabellen toe te voegen aan de join tabel, bijvoorbeeld bij de tabel aanwezigheidslijsten en de tabel gebruikers.

Hierbij moet er in de join tabel de status van de gebruiker toegevoegd worden. De enige manier om dit op te lossen is door de join tabel als een schema toe te voegen. Met deze join tabel kan er dan een one-to-many relatie gemaakt worden met de gebruikers tabel en met de aanwezigheidslijsten tabel. De ORM Node-orm2 is wel instaat many-to- many relaties aan te maken met extra kolommen in de join tabel.

Deep querying

Het bovenstaande probleem en de bijhorende oplossing brengt ons bij het volgende probleem, Waterline ondersteunt geen deep querying. Dit probleem doet zich voor wanneer een query op basis van parameters in de ene tabel, rijen uit de andere tabel moet teruggeven. Het volgende voorbeeld verduidelijkt dit:

De relatie tussen gebruikers en hun rapporten is een one-to-many relatie. Een gebruiker kan meerdere rapporten hebben, maar een rapport behoort toe tot maar een gebruiker. Stel dat we alle gebruikers willen opvragen die een resultaat groter dan 7 hebben gehaald op de crawl oefening (met ID = 5). Met deep linking zouden we dit dan met de volgende query kunnen bekomen:

```
gebruiker.find({
  'rapporten': {
    oefeningID:5,
    resultaat: { '>': 7}
  }
})
  .populate('rapporten')
```

Bovenstaande query geeft echter een leeg array terug en is dus onbruikbaar.
Om het toch mogelijk te maken deze lijst weer te geven kan het volgende gedaan worden:

```
rapport.find({
    oefeningID:5,
    resultaat: { '>': 7}
)}
```

Hiermee krijgen we een array met rapport objecten. Voor elke rapport in dit array kan dan het volgende gedaan worden:

```
gebruiker.find({
    rapporten : idVanRapport
)}
```

Uiteindelijk bekomen we de gewenste data, maar het resulteert al gauw in een wirwar van forloops en queries. Een andere oplossing is om zelf specifieke queries te schrijven, maar de syntax van deze queries is database-afhankelijk. Hierdoor gaat het voordeel van de ORM een beetje verloren.

Geen Foreign keys

Waterline maakt associaties aan zonder de IDs, die de tabellen linken, als foreign key te definiëren. Hierdoor is het mogelijk relaties aan te maken met elementen die niet of niet meer bestaan in de database. Indien de database overgezet moet worden naar een ander platform, moeten deze foreign keys handmatig worden ingesteld. Op 20 december 2014 is dit probleem aangekaart op github [36], maar tot op heden is hier nog altijd geen response op gekomen.

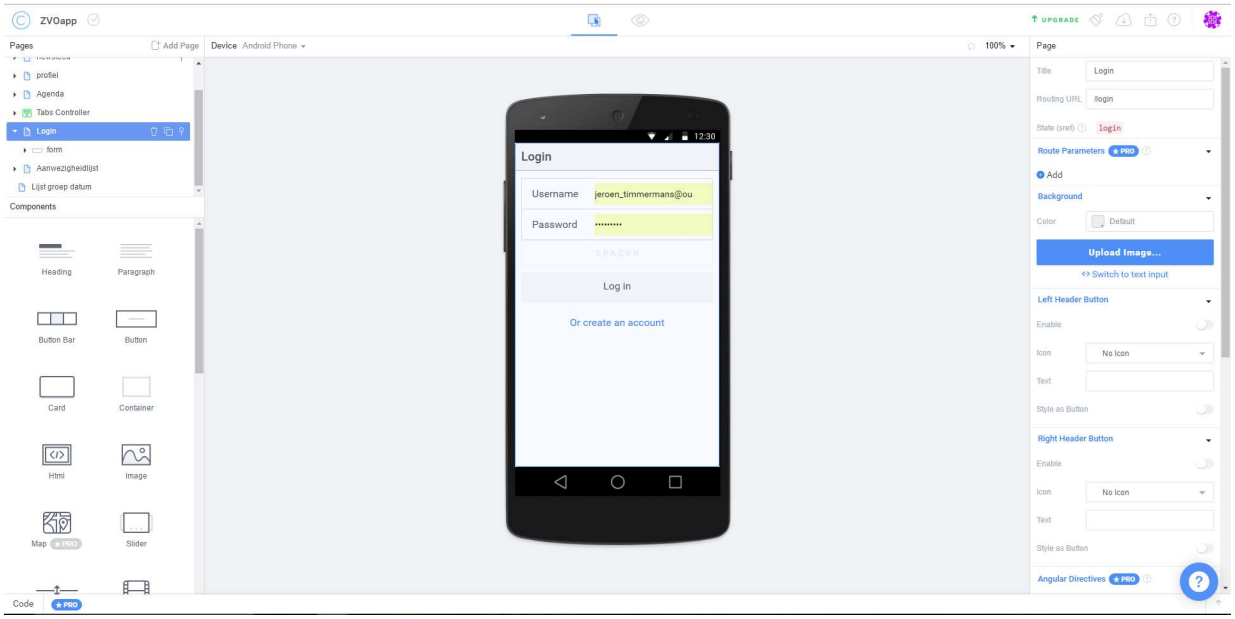
7. Frontend

De frontend is op te delen in drie onderdelen: een website, app en bots. Van deze drie zijn er twee uitgewerkt, namelijk de app en de bots. In dit hoofdstuk wordt er dieper ingegaan op de gebruikte methodes en uitwerking van de app en bots.

7.1. Ionic

7.1.1. Ionic Creator

Voor het maken van de view is gebruik gemaakt van Ionic Creator.



Figuur 28: Interface Ionic Creator app

Ionic Creator maakt het mogelijk om op een eenvoudige manier de view van een Ionic app samen te stellen. De Creator voorziet de gebruiker van enkele basiselementen zoals: tekstinput, labels, buttons, lijsten, afbeeldingen, …. De beschikbare elementen zijn voldoende om een schets te maken van de app. Elke pagina in de app is te vergelijken met een pagina op een website. Deze 'pages' hebben elke een routing URL waarmee het mogelijk wordt naar de pagina te verwijzen. Deze verwijzingen kunnen gebruikt worden om met behulp van buttons andere pagina's te openen. Dit zorgt ervoor dat naast het uiterlijk van de view ook de flow getest kan worden in Ionic Creator. Dit alles kan op drie manieren geëxporteerd worden: als zip, app package of via de Creator mobile app.
Een app als zip exporteren zorgt ervoor dat er een zip bestand wordt aangemaakt met daarin de html, CSS en .js files van alle pagina's. Met een app package krijgt de gebruiker een native package voor iOS of Android. De Creator mobile app is een app van Ionic die het mogelijk maakt op je view meteen te testen op een device. Deze laatste is ook heel handig als er in teamverband wordt gewerkt. Met het gratis account is het enkel mogelijk om gebruik te maken van de zip export. Gelukkig is de zip export de enige export die echt nodig is om de view te kunnen integreren in de app.

7.1.2. Deployen

Om de ZVOapp te kunnen installeren moet ze ook beschikbaar zijn voor gebruikers om ze te downloaden. Dit kan om verschillende manieren. Onderstaande paragrafen gaan dieper in op de mogelijkheden.

Native store

De app store (iOS) en Play store (Android) maken het mogelijk om apps te downloaden en te installeren op smartphones. Vooraleer een app in de store beschikbaar is voor gebruikers, moet deze eerst opgeladen (deployen) worden op de appstore. Hierbij komt heel wat meer kijken dan simpelweg de app uploaden. Afhankelijk van iOS of Android gaat dit als volgt te werk:

Android

Vooraleer de app geüpload kan worden moet ze eerst gesigned worden. Dit signen zorgt ervoor dat de app een unieke handtekening toegewezen krijgt, genaamd een Keystore. Deze handtekening is een bewijs van oorsprong van de app en moet voor altijd bijgehouden worden.

Eens de app ondertekend is, kan ze toegevoegd worden aan de store. Hiervoor is een Google Play developer account nodig. Indien de ontwikkelaar nog geen account heeft dient hij een eenmalige kost van 25 Dollar te betalen.

iOS

Net zoals Android is er voor iOS ook een developer account verijst, voor iOS kost dit echter 99 Dolar. Daarnaast moet er ook gebruik gemaakt worden van de Xcode software die enkel werkt op een Mac OS. Windows gebruikers moeten dus een Mac aanschaffen ofwel een virtuele versie van OS X draaien op hun Windows pc. Xcode maakt het mogelijk om de app te signen en te uploaden voor review. Als de app toegelaten wordt door Apple zal deze in de store verschijnen.

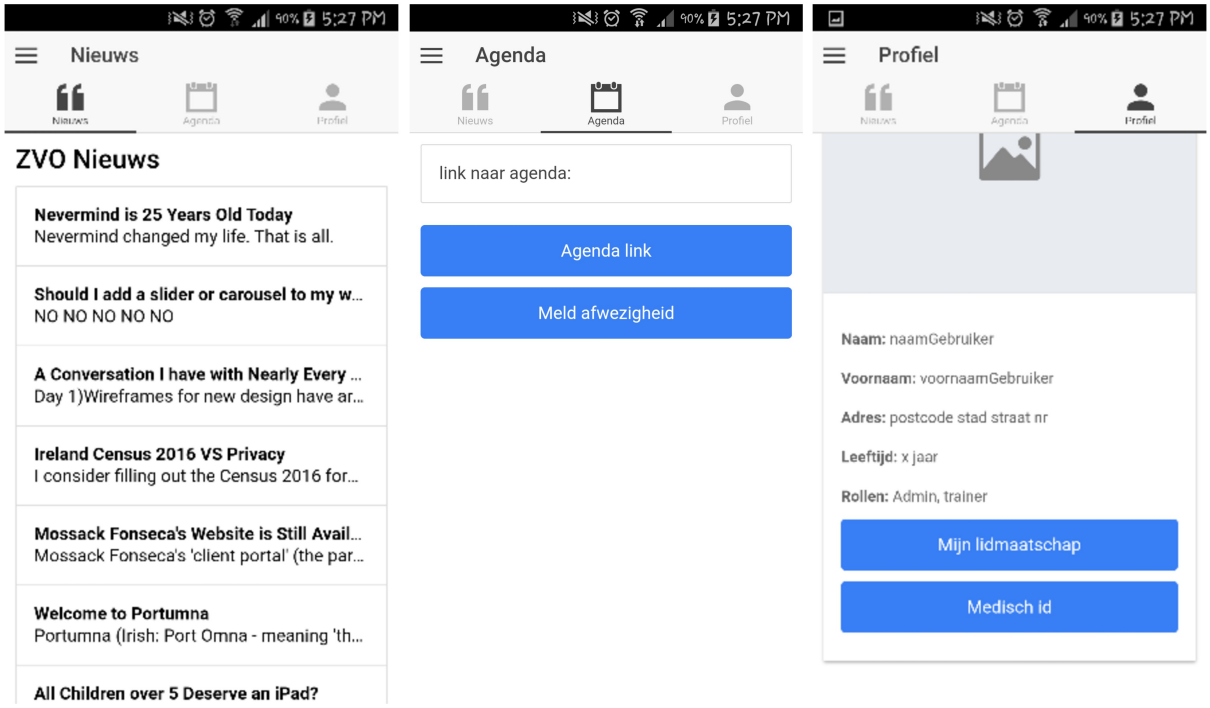
Ionic deploy

Ionic deploy [37] kan naast of samen met de store gebruikt worden. De service maakt het mogelijk updates te deployen zonder dat deze terug via de appstore hoeven te gaan. Dit is wel op voorwaarde dat de updates zich beperken tot de web assets zoals HTML, JavaScript en CSS. Wanneer er updates uitgevoerd worden aan de native componenten van de app, moet dit nog altijd via de store gebeuren.

7.1.3. Uitwerking

Onderstaande tekst en afbeeldingen geven een korte beschrijving van de app.

Home



Figuur 29: Screenshots Ionic app home menu

In bovenstaande *Figuur 29* zijn de drie tabs: Nieuws, Agenda en Profiel te zien. De werking achter deze tabs is als volgt:

Nieuws

In de nieuws tab kunnen berichten worden weergegeven die door leden of beheerders gepost worden. Het valt op dat de nieuwsberichten in de afbeelding in het Engels zijn. Dit komt omdat de app momenteel gebruik maakt van een demo link [38]. Er is gebruik gemaakt van een demo link omdat het ZVO platform momenteel nog niet beschikt over een nieuws service. Een alternatief voor de huidige nieuwspagina zou een Facebookpagina kunnen zijn. De berichten moeten dan niet meer door de ZVOserver worden afgehandeld en er hoeft ook geen interface meer gemaakt te worden voor het posten van berichten. Dankzij de Facebook page-plugin [39] kan de pagina dan worden ingeladen in de ZVOapp.

Agenda

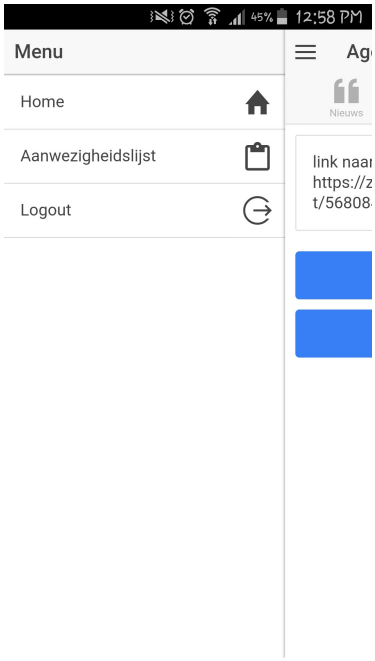
De agenda tab geeft leden de mogelijkheid om de link naar hun persoonlijke agenda op te vragen en een afwezigheid te melden. Voorlopig wordt de link enkel weergegeven in de app. Momenteel moeten de gebruikers deze link zelf nog in hun agenda app ingeven, dit zorgt nog voor te veel verwarring. In de toekomst kan de app, bij het opvragen van de agenda, verwijzen naar een agenda app die de link automatisch kan openen.

Profiel

Deze pagina spreekt eigenlijk voor zich. Ze stelt gebruikers instaat hun profiel te bekijken. Daarnaast is er ook de mogelijkheid om informatie omtrent het lidmaatschap en het Medisch ID van de gebruiker op te vragen.

Menu

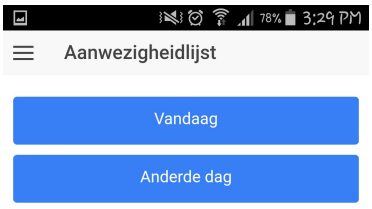
Naast het home scherm is er ook side menu:



Figuur 30: Side menu Ionic app

Bovenstaande *Figuur 30* laat het side menu zien. Momenteel is er enkel de optie voorzien om aanwezigheidslijsten op te zoeken en uit te loggen. In de toekomst kan dit uitgebreid worden met menu's voor het opvragen van rapporten, examens, groepen, ledenbeheer, …

Aanwezigheidslijsten



Figuur 31: Aanwezigheidslijsten menu

Het bovenstaande *Figuur 31* spreekt voor zich. De gebruiker kan lijsten van diezelfde dag opvragen of zelf een bepaalde dag kiezen, zie onderstaande *Figuur 32*.

Figuur 32: Android datum kiezer

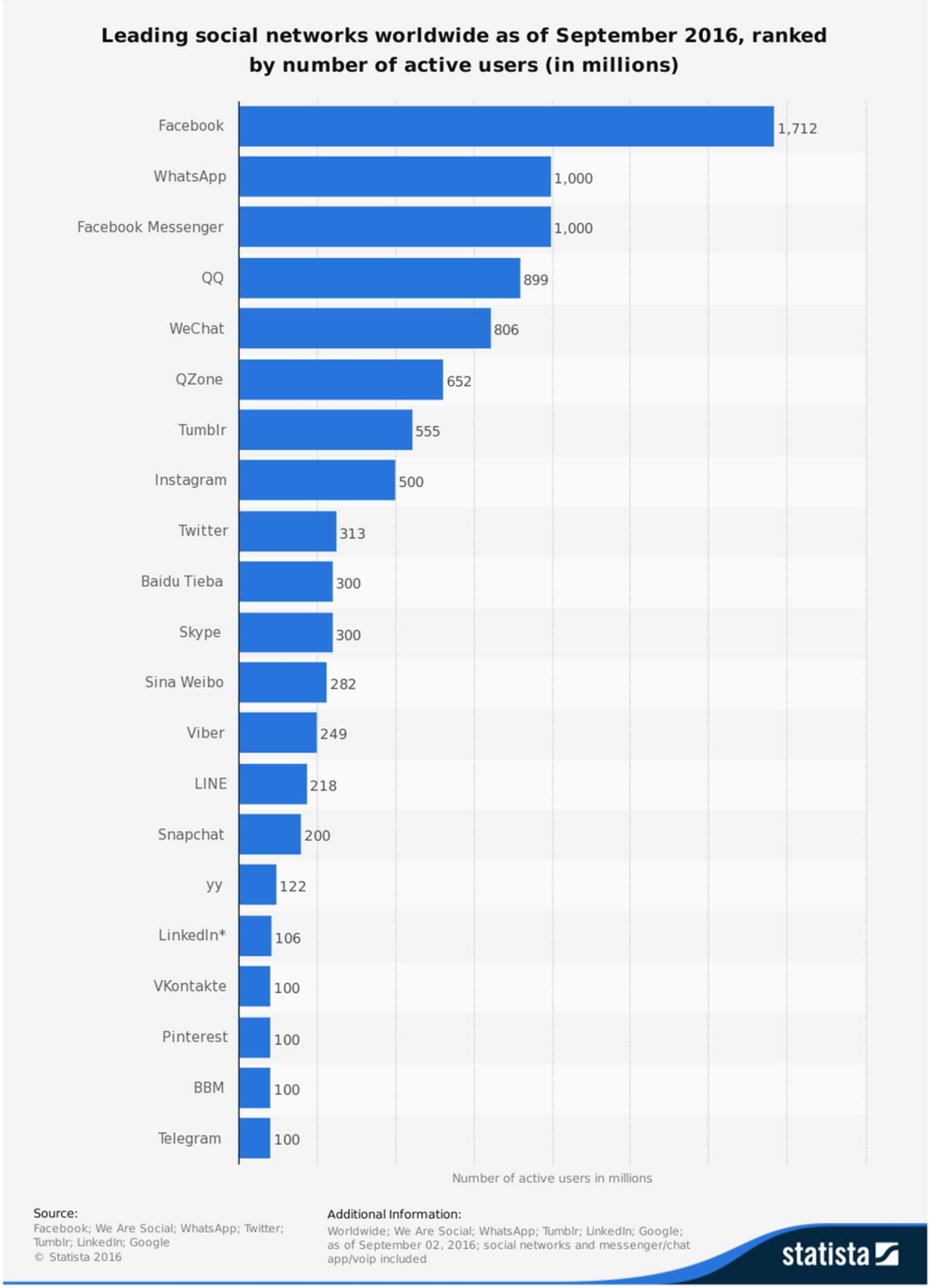
Om een datepicker in de app te integrere,n is er gebruik gemaakt van de Ionic-datepicker module [40]. Deze module maakt het mogelijk om op een eenvoudige manier een dateipicker te integreren.

7.2. Bots

Naast de ZVOapp is er ook een bot ontwikkeld. Het concept van bots is helemaal niet nieuw meer. Zo zag ELIZA al in 1966 het leven als een van de eerste chatbots [41]. ELIZA kon gesprekken voeren door een database die bestond uit mogelijke input zinnen te vergelijken met de input van de gebruiker.

Op basis van de input werd dan een vooropgesteld antwoord gegeven. Het merendeel van de moderne chatbots werkt nog altijd op basis van dit principe. Dankzij de vorderingen op het vlak van NLP en ML kunnen de huidige bots sneller leren dan hun voorgangers. Ook is er recentelijk een toename in chatplatformen die bots ondersteunen waaronder enkele grote spelers zoals Facebook Messenger, WeChat, Skype, … Door de populariteit van chat apps *(zie Figuur 33)* en de gigantische userbase die ze hiermee hebben opgebouwd, kan een bot nu ook profiteren van deze populariteit en meteen aangesproken worden door de enorme userbase.

In onderstaande statistieken uit *Figuur 33* is te zien dat de drie populairste chatplatformen meer maandelijkse gebruikers hebben dan de drie populairste social media platformen.



Figuur 33: Populariteit social media en messenger platformen [42]

Hieruit wordt duidelijk dat het gebruik van chatapps dat van social media platformen overstijgt.

Los van dat bots misschien ooit de nieuwe “apps” gaan worden, is het momenteel een geschikte weg om een app of platform te koppelen met bestaande social media. Daarom is er gekozen om te experimenteren met het concept van bots en naast de ZVOapp ook een ZVObot te ontwikkelen.

De bedoeling van de bot is niet om de gebruiker het gevoel te geven dat hij met een intelligente entiteit aan het communiceren is. De bot zal eerder dienen als een app in een social media (Messenger) omgeving.

7.2.1. Chatplatform

Om een bot te kunnen gebruiken, moet de gebruiker ook gebruik maken van het chat platform waarmee de bot geïntegreerd is. Uit *Figuur 33* wordt duidelijk welke momenteel de populairste chatplatform zijn, de top vijf is namelijk als volgt:

1. WhatsApp,
2. Facebook Messenger,
3. QQ,,
4. WeChat,
5. Skype.

In deze lijst komen er drie bekenden naar voren, namelijk WhatsApp, Messenger en Skype. QQ en WeChat zijn afkomstig uit Azië en minder tot niet bekend in België. Hoewel WhatsApp als populairste naar voren komt, is er momenteel nog geen ondersteuning voor bots. Sterker nog WhatsApp heeft zelf geen officiële API. Daarentegen heeft Facebook Messenger (dat net zoals WhatsApp eigendom van Facebook is) wel ondersteuning voor bots. Skype heeft net zoals Facebook ook ondersteuning voor bots, maar is vergeleken met Facebook Messenger niet zo populair en moet het meer van zijn videochat hebben. Van de drie platformen die wel bekend zijn, blijft dus enkel Messenger nog over als het ideale chatplatform om de ZVObot mee te integreren.

7.2.2. Bot platformen en NLP

Voor het maken van een chatbot bestaan er verscheidene platformen, de ene al wat uitgebreider als de ander. Gaande van gesprekken met standaardzinnen en knopjes tot het gebruik van machine learning en NLP (Neural Language Processing).

Motion.ai

Maakt het mogelijk een bot te bouwen zonder dat er code hoeft geschreven te worden. Door het gebruik van een grafische programmeeromgeving wordt de instapdrempel aanzienlijk verlaagd. Motion.ai ondersteunt geen NLP, maar biedt wel de mogelijkheid om JavaScript-modules te gebruiken. In principe kan er dan in zo een module een NLP zoals Wit.ai geïntegreerd worden. Dit laatste is echter niet ideaal en daarom kan er beter voor een ander platform gekozen worden.

De volgende platformen zijn standaard geïntegreerd in motion.ai:

- Webchat,
- SMS,
- Facebook Messenger,
- Slack;
- Email;
- Custom / API (een eigen app of nog niet ondersteund platform).

Api.ai

Api.ai is een van de meest geavanceerde platformen op het vlak van tekstverwerking. Het kan gebruikt worden om spraakherkenning, NLP en conversatie management te implementeren in een app. Het bedrijf is opgericht in 2010 en in 2016 overgenomen door Google. Api.ai biedt standaard al integraties aan voor verscheidene platformen waaronder:

- Facebook Messenger,
- Slack,
- Skype,
- Kik,
- Line,
- Telegram,
- Microsoft Cortana,
- Twitter.

Dankzij deze integratie kan er een bot gemaakt worden die interactie biedt met gebruikers van het platform. Door middel van "Intents" kan de ontwikkelaar het platform voorzien van voorbeeld user input. De ontwikkelaar kan dan definiëren wat de bot moet doen. Bijvoorbeeld een conversatie met een weer bot:

- gebruiker: "wat is het weer voor morgen?";
- Api.ai herkent het woord morgen als een Date parameter en stuurt de datum van morgen naar een API die informatie over het weer kan geven;
- Api.ai gebruikt de ontvangen data om een nieuw bericht samen te stellen;
- "Morgen gaat het voornamelijk zonnig worden en in de namiddag gemiddeld 20 graden Celsius zijn".

Zoals te zien in bovenstaand voorbeeld is de user input in het Nederland. Het is dan ook een sterk punt van Api.ai, dat het overweg kan met Nederlandse input. Voorlopig is er, wanneer er gewerkt wordt in het Nederlands, nog geen mogelijkheid om machine learning te gebruiken. Dit heeft als nadeel dat de bot zinnen met typfouten of anders geformuleerde zinnen niet zal herkennen. De reeds bestaande integratie is zeker een meerwaarde, maar wanneer een bot complexer wordt, zijn er toch beperkingen *(zie Bot Authenticatie).*

Wit.ai

Bot service die je zelf moet hosten. Het gebruik ervan is dus in zekere zin gratis. Wit.ai is net zoals Api.ai. Het maakt gebruik van machine learning en NLP om volzinnen te interpreteren. Dit maakt Wit.ai ook geschikt voor het gebruik in toepassingen die aangestuurd worden op basis van stemcommando's. Het heeft ook de mogelijkheid om Nederlandse kernwoorden te herkennen, bijvoorbeeld een adres dat in een zin staat beschreven. In het Nederlands is de API wel veel beperkter en is het bijvoorbeeld niet mogelijk om een datum in een zin te herkennen. Bijvoorbeeld in het Engels zou Wit.ai dan het woord "tomorrow" omzetten naar de datum van de volgende dag.

Aangezien dit niet mogelijk is in het Nederlands, is er momenteel niet een echte meerwaarde aan de ondersteuning van de Nederlandse taal. Api.ai staat hierin wel al verder *(zie Api.ai).*

Wit.ai biedt SDK's aan voor Node.js, Python en Ruby. Bot platformen zoals Meya maken hier gebruik van om NLP en ML te integreren in hun platform. Wit.ai legt enkel de focus op het verwerken van tekst. Api.ai doet dit ook maar gaat hier nog een stap verder in door bepaalde platformen zoals Messsenger en Telegram te integreren.

Meya

Meya is net zoals Api.ai een volledig platform voor het maken van bots. Waar Api.ai zelf instaat voor het machine learning en NLP gebeuren, wordt dit bij Meya uitbesteed aan wit.ai of luis.ai. Meya legt de nadruk op het aanmaken van bots en het gebeuren hierrond. Zo is Meya een van de weinige of misschien het enige bot platform dat de mogelijkheid voorziet om gebruikers te identificeren met Facebook Account-Linking *(zie 6.1.2 Bot Authenticatie)*. Meya biedt integratie met volgende platformen:

- Facebook Messenger,
- Intercom,
- Kik,
- Slack,
- Smooch,
- Telegram,
- Twillio,
- Twitter,
- Web (live chat),
- Webhook.

Daarnaast biedt Meya ook nog cloud dataopslag en de mogelijkheid om taken te plannen (scheduling). Een dusdanig uitgebreid platform komt natuurlijk met een prijskaartje.

In onderstaande *Tabel 11* wordt het prijskaartje van Meya nader toegelicht.

Tabel 11: Prijs tabel Meya

$0/mo	$49/mo	Popular $99/mo	$199/mo	Let's talk
FREE	STARTUP	BUSINESS	ORGANIZATION	ENTERPRISE
500 messages /mo	25,000 messages /mo[5]	100,000 messages /mo[6]	Flex pricing[5] 250,000 messages /mo or 10,000 users /mo	Flex pricing[5] > 250,000 messages /mo or 10,000 users /mo
1 bot*	5 bots	Unlimited bots	Unlimited bots	Unlimited bots
1 collaborator	2 collaborators	Unlimited collaborators	Unlimited collaborators	Unlimited collaborators
Community support	Bronze support	Silver support	Gold support	Platinum support
Most messaging channels	All messaging channels	All messaging channels	All messaging channels	All messaging channels
--	Analytics	Analytics	Analytics	Analytics
--	Broadcast	Broadcast	Broadcast	Broadcast
--	--	Customer service integrations	Customer service integrations	Customer service integrations
--	--	Export/Github	Export/Github	Export/Github

Meya is pas op 7 april 2016 gelanceerd en is dus nog zeer jong en de prijzen kunnen zeker nog veranderen. Zo had Api.ai in het begin van 2016 een gelijkaardig prijskaartje en heeft het nu enkel nog een gratis of "Let's talk" plan.

7.2.3. Conclusie

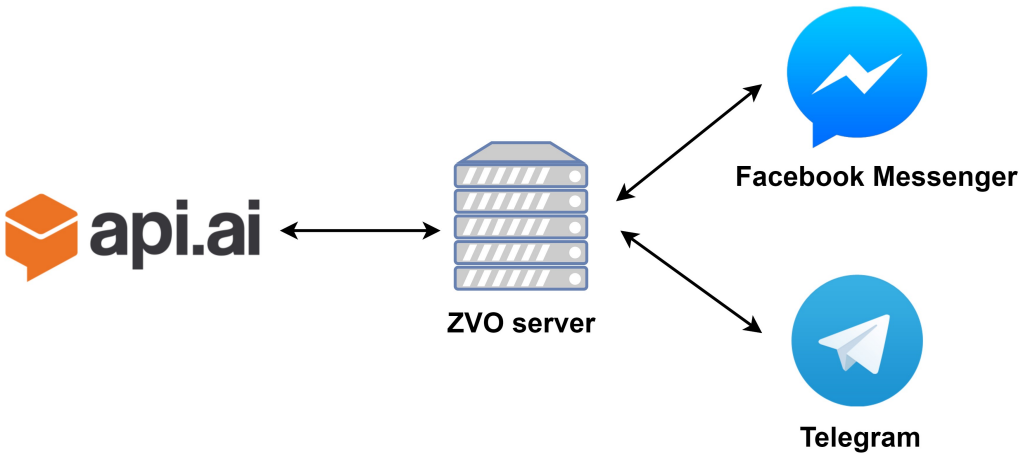
De botmarkt is nog in volle groei en platformen zijn nog in volle ontwikkeling. Sommige proberen een zo volledig mogelijk platform te vormen (Meya) waar andere voornamelijk de focus leggen op NLP en machine learning (Wit.ai, Api.ai). Elke platform is in theorie geschikt voor het maken van een ZVObot, maar Api.ai schiet er voorlopig bovenuit. De mogelijkheid om Nederlandse input te verwerken is een groot pluspunt vergeleken met andere platformen. Daarbovenop is Api.ai overgekocht door Google en als Google een bedrijf overkoopt dan moeten ze wel iets goed doen. Sinds de overname van Google is er ook de mogelijkheid om Api.ai te integreren in Google Home, dat gebruik maakt van Google Assistant, Googles versie van Siri. Kortom Api.ai is nog in volle groei en een geschikte API voor het maken van een ZVObot.

7.2.4. Bot uitwerking

Bij de ontwikkeling van de bot is er gekozen voor twee chat platformen. Door de bot op twee platformen uit te werken, kan er een vergelijking gemaakt worden tussen deze platformen. In *7.2.1 Chatplatform* kwam Facebook messenger als ideale kandidaat naar voren. Naast Messenger is ook gekozen om gebruik te maken van Telegram. Hoewel Telegram helemaal onderaan de vergelijking in *Figuur 31* terugkomt, is het in vergelijking met sommige andere chatplatformen zeer innovatief. Zo was het al op juni 2015 mogelijk om op Telegram gebruik te maken van bots en kan dit op Facebook Messenger pas sinds april 2016.

Structuur

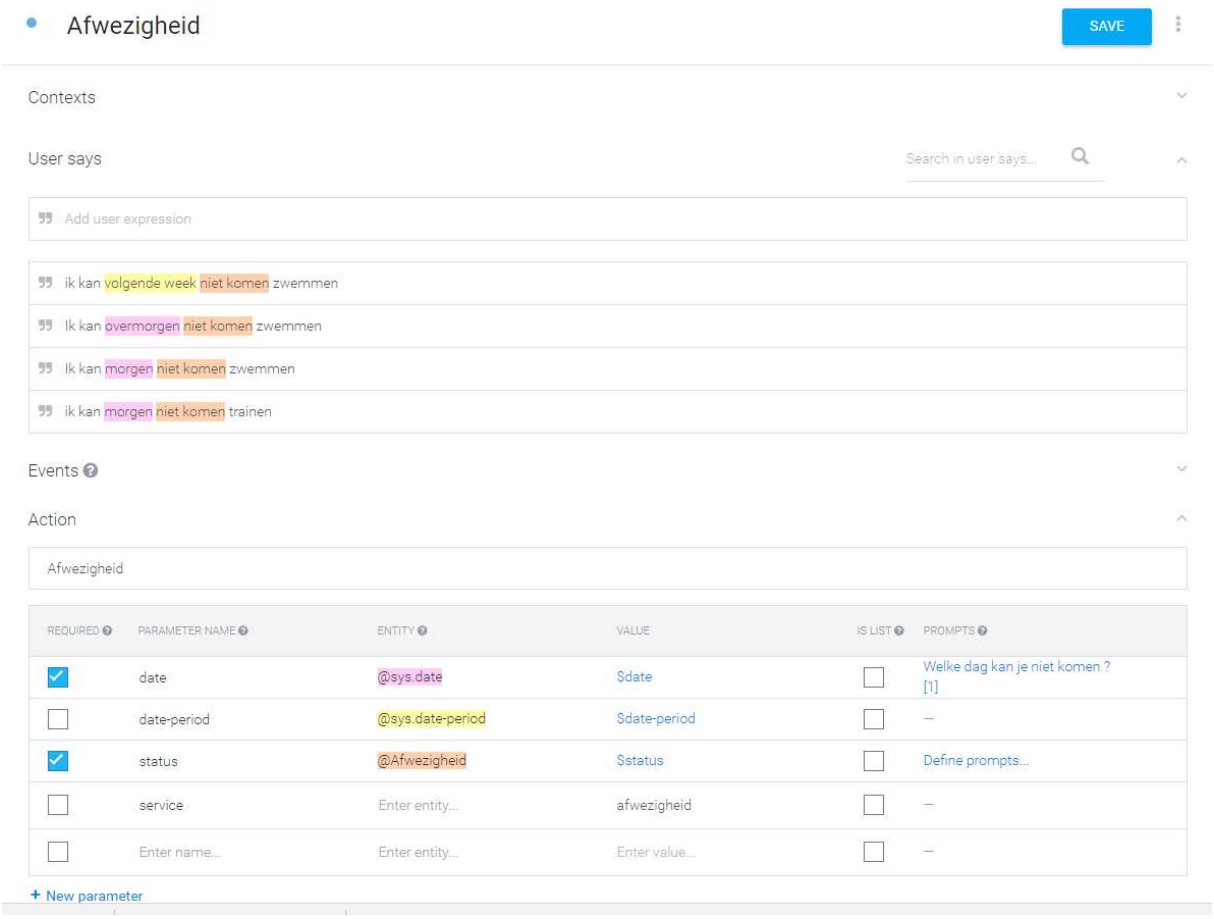
Onderstaande *Figuur 34* is een visuele representatie van het verband tussen de ZVO server, de chatplatformen en API.ai.



Figuur 34: opbouw bot service

Bovenstaande *Figuur 34* laat zien dat de bots enkel rechtstreeks communiceren met de ZVO server, de reden hiervoor is nader besproken in *6.1.2. Bot Authenticatie*. De server gebruikt API.ai enkel voor het interpreteren van berichten.

Het verloop van de conversaties wordt grotendeels gedefinieerd in API.ai in de vorm van intents, onderstaande *Figuur 35* is een voorbeeld van zo een intent.



Figuur 35: Voorbeeld afwezigheidsintent

Figuur 35 is de Afwezigheidsintent die het mogelijk maakt voor gebruikers om zich afwezig te melden. De intent bevat de volgende onderdelen:

Context
Context kan gebruikt worden om parameters van een andere intent in te lezen of door te geven. Bijvoorbeeld intent A vraagt de naam van een gebruiker en voegt deze toe als output Context. Dan kan intent B deze naam uitlezen en gebruiken in zijn response.

User says
Hier staan alle input zinnen die de Afwezigheidsintent zullen activeren.

Events
Via Events wordt het mogelijk een intent te activeren zonder gebruik te maken van user input. Dit wordt bijvoorbeeld gebruikt om een welkombericht te sturen.

Dit wordt momenteel niet gebruikt door de ZVOserver, deze stuurt namelijk de berichten rechtstreeks door naar het chatplatform.

Action
Actions maken het mogelijk parameters te definiëren die voor kunnen komen in een input zin. Deze parameters kunnen een Entity worden toegewezen. Een Entity beschrijft de waarde van de parameter. Bijvoorbeeld voor date zal @sys.date de tekst “morgen” proberen om te zetten in een date object. Prompts kunnen gebruikt worden om een gebruiker naar parameters te vragen, wanneer deze niet aanwezig waren in de input.

Wanneer de Afwezigheidsintent wordt geactiveerd door een chatgebruiker dan ziet dit er als volgt uit:

Figuur 36: Gebruiker meldt afwezigheid via Telegram bot

We zien dat de gebruiker wil laten weten dat hij niet kan komen zwemmen. Deze input zin wordt door de ZVOserver doorgestuurd naar API.ai. API.ai herkent de tekst als de Afwezigheidsintent, maar kan de date parameter niet vinden. Daarop stuurt API.ai een prompt met de vraag “welke dag kan je niet komen?” naar de server. De server stuurt dit door naar de gebruiker. De gebruiker antwoord nu met een specifieke dag “maandag”. Dit komt opnieuw bij API.ai terecht waarop API.ai reageert met een laatste bericht om de gebruiker te bedanken. De Server ontvangt dit bericht en ziet aan de response van API.ai dat dit het laatste bericht was en er ook een date object is geleverd.

De server probeert nu met de gegeven datum een afwezigheid in te plannen. Indien het lukt de gebruiker afwezig te melden, zal de server het, “Het staat genoteerd, bedankt” bericht doorsturen naar de gebruiker. Indien het niet lukt de gebruiker afwezig te melden, zal de server reageren met een aangepast bericht om de gebruiker te verwittigen dat er iets is fout gegaan.

8. Besluit

Oorspronkelijk maakte de ZVOclub gebruik van een nodeloos log en complex aanvoelende applicatie. Met als gevolg dat hier amper gebruik van werd gemaakt en de database slecht onderhouden werd.

Door de combinatie van API's en een op maat gemaakte backend is er een platform ontstaan dat taken zoals aanwezigheidslijsten beheren, agenda's raadplegen, zoeken naar ververvangers, … volledig digitaliseert en in sommige gevallen zelfs automatiseert. Door het gebruik van een Messengerbot en op maat gemaakte app wordt de interactie met leden en het gebruik en onderhoud van dit platform vereenvoudigd. In vergelijking met de voorgaande oplossing zet dit sneller aan tot gebruik met als gevolg dat de database beter wordt onderhouden en benut.

Naar de toekomst toe kan de ZVOclub de services die nu door de app en de bot gebruikt worden integreren in hun website. Het platform heeft een basis gelegd die er voordien niet was en laat de weg open voor uitbreidingen.

9. Reflectie

Het is gelukt om met de gebruikte tools een basis te leggen voor het ZVO platform. Sommige van deze tools bleken achteraf beter geschikt dan andere.

9.1. Backend

Sails maakt het dankzij de basis die het legt makkelijker voor beginnende Node.js backend ontwikkelaars. De ingebouwde modules zoals Policies, Blueprints, en Rest routes maken het aanmaken van Rest services aanzienlijk eenvoudiger. Maar Sails heeft ook zijn gebreken en bij nader inzien is het Sails framework niet ideaal geweest. Waterline, de ORM waar Sails rond gebouwd is, heeft sommige tekortkomingen die voor frustraties zorgen. Het zou kunnen opgelost worden door een ORM te gebruiken zoals Node-ORM2. Maar om deze ORM te koppelen aan Sails moet er zoveel aan de basis van Sails herschreven worden dat het uiteindelijk makkelijker is om Sails te vervangen door een ander framework. Voor iemand die ervaring heeft met Node.js is het beter om het Express framework te gebruiken en de nodige modules hier zelf aan te koppelen. Het is zeker meer werk dan met Sails, en voor beginners misschien te hoog gegrepen, maar op de lange termijn gaat het zeker lonen.

9.2. Frontend

De frontend bestaat uit twee delen. Aan één zijde is er de app en aan de andere kant zijn er de bots.

App

Ionic bleek achteraf een goede keuze te zijn. De app voelt niet stroef aan en qua design waren er genoeg mogelijkheden om de app toch nog modern te houden. Ionic Creator was ook een meerwaarde in de ontwikkeling van de app.

Bot

Het gebruik van API.ai heeft zeker het ontwikkelingsproces vergemakkelijkt en ingekort. De implementatie van API.ai in de backend werkt naar behoren, maar oogt nog stroef. Bij een aanvraag van een service wordt de response nu volledig door de backend afgehandeld. Het zou overzichtelijker zijn moest de backend API.ai voorzien van data en API.ai de response afhandelen. De conversaties zouden dan volledig in API.ai beschreven zijn.

De bot maakt voornamelijk gebruik van tekst input, dit werkt, maar het is niet altijd even duidelijk voor de gebruiker. Door gebruik te maken van inchat buttons had de interactie met de bot vereenvoudigd kunnen worden.

9.3. Persoonlijk

In het begin van deze masterproef had ik amper kennis van webontwikkeling. Een opdracht die bestond uit het maken van een website of mobiele app die gebruik maakte van open webservices zoals Google Calendar, Facebook, Dropbox, Pinterest, Instagram, …, leek een ideale manier om me te verdiepen in de wereld van webontwikkeling.

Deze masterproef heeft zeker mijn ogen geopend en me doen inzien wat er allemaal komt kijken bij het ontwikkelen van een web platform. Het heeft me vooral laten inzien hoe moeilijk het is om als beginneling je in te werken in de gigantische wereld van webontwikkeling. Daarnaast is ook gebleken hoe moeilijk het is om de requirements tussen opdrachtgever en uitvoerder op een lijn te krijgen. Dit is al zichtbaar aan het verschil tussen de beschrijving van de opdracht en de werkelijke wensen van de club. De opdracht leek de nadruk te leggen op een sociaal platform gericht op de communicatie tussen gebruikers en beheerders. Uiteindelijk bleken de grootste tijdrovers het beheren van de club te zijn.

Ik heb daarom het gehele platform op een andere manier bekeken dan de opdracht eerst beschreef en mij gefocust op het bouwen van een platform dat met minimale input een maximale uitput kan leveren. Een voorbeeld hiervan is de jaarkalender die in Google Calendar wordt opgeslagen en automatisch persoonlijke agenda's en aanwezigheden genereert.

Ik heb ook de vruchten kunnen plukken van de opkomst van bots op populaire chatplatformen en deze geïntegreerd met het ZVO-platform. Dit heeft me de kans geven te verdiepen in een vakgebied waar zeker potentieel in zit.

Zelf zie ik het platform nog niet als 'compleet', er is ruimte voor verbetering. Het platform legt in ieder geval al een solide basis voor het implementeren van nieuwe functies.

Het is gelukt om met de gebruikte tools een basis te leggen voor het ZVO platform. Sommige van deze tools bleken achteraf beter geschikt dan andere.

Bibliografie

[1] „socie.nl," Tizin Mobile, [Online]. Available: https://www.socie.nl/nl/features/. [Geopend 2016].

[2] „ecsa-zwemmen.nl," [Online]. Available: http://www.esca-zwemmen.nl/. [Geopend 2017].

[3] „sportadministratie.be," [Online]. Available: http://www.sportadministratie.be/Docs/sportadministratie_sponsoraccount.pdf. [Geopend 2016].

[4] „activeswim.com," ActiveHy-Tek, [Online]. Available: http://www.activeswim.com/swim-meet-software/meet-manager.htm. [Geopend 2016].

[5] „activeswim.com," ActiveHy-Tek, [Online]. Available: http://www.activeswim.com/swim-meet-software/meet-mobile. [Geopend 2016].

[6] „activeswim.com," ActiveHy-Tek, [Online]. Available: http://www.activeswim.com/swim-meet-software/swim-manager. [Geopend 2016].

[7] A. Network, „youtube.com," Active Hy-Tek, 19 september 2014. [Online]. Available: https://youtu.be/p4CztAeXK6A. [Geopend 2016].

[8] „spectrum.ieee.org," IEEE Spectrum, 26 juli 2016. [Online]. Available: http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2016. [Geopend 2016].

[9] K. Bouwkamp, „codingdojo.com," 27 januari 2016. [Online]. Available: http://www.codingdojo.com/blog/8-most-in-demand-programming-languages-of-2015/.

[10] C. Wodehouse, „Upwork.com," [Online]. Available: https://www.upwork.com/hiring/development/the-java-platform/. [Geopend 2016].

[11] M. Dowsden, „opensource.com," 22 oktober 2015. [Online]. Available: http://opensource.com/life/15/10/4-java-web-frameworks. [Geopend 2016].

[12] „hotframeworks.com," hotframeworks, [Online]. Available: http://hotframeworks.com/languages/java. [Geopend 2016].

[13] S. Maple, „zeroternaround.com," Zeroternaround, 14 July 2016. [Online]. Available: https://zeroturnaround.com/rebellabs/java-tools-and-technologies-landscape-2016-trends/. [Geopend 2016].

[14] „trends.builtwith.com," 2016. [Online]. Available: https://trends.builtwith.com/docinfo/Javascript. [Geopend 2016].

[15] „nodejs.org," [Online]. Available: https://nodejs.org/en/about/. [Geopend 2016].

[16] „hotframeworks.com," hotframeworks, [Online]. Available: http://hotframeworks.com/languages/javascript. [Geopend 2016].

[17] D. Rathore, „denubook.com," [Online]. Available: https://www.dunebook.com/5-best-javascript-frameworks-to-learn-in-2017/. [Geopend 2017].

[18] „nodeframework.com," [Online]. Available: http://nodeframework.com/. [Geopend 2016].

[19] noeticsunil, „noeticforce.com," 16 juli 2016. [Online]. Available: http://noeticforce.com/best-nodejs-frameworks-for-web-and-app-development. [Geopend 2016].

[20] mikermcneil, „github.com," [Online]. Available: https://github.com/balderdashy/waterline.

[21] „hotframeworks.com," hotframeworks, [Online]. Available: http://hotframeworks.com/languages/c-sharp. [Geopend 2016].

[22] „mobilehtml5.org," 17 september 2015 . [Online]. Available: http://mobilehtml5.org/. [Geopend januari 2017].

[23] T. Agrimbau, „toptal.com," [Online]. Available: http://www.toptal.com/android/developing-mobile-web-apps-when-why-and-how. [Geopend 2016].

[24] E. Verbruggen, „gtihub.com," [Online]. Available: https://github.com/EddyVerbruggen/Calendar-PhoneGap-Plugin. [Geopend 2016].

[25] T. Sokoll, „linkedin.com," 30 november 2015. [Online]. Available: https://www.linkedin.com/pulse/importance-push-notification-mobile-apps-tj-sokoll. [Geopend 2016].

[26] „documentation.onesignal.com," OneSignal, [Online]. Available: https://documentation.onesignal.com/docs. [Geopend 2016].

[27] „onesignal.com," OnseSignal, [Online]. Available: https://onesignal.com/about. [Geopend 2016].

[28] „developers.facebook.com," Facebook, [Online]. Available: https://developers.facebook.com/docs/analytics/push-campaigns/. [Geopend 2016].

[29] „docs.ionic.io," Ionic, [Online]. Available: https://docs.ionic.io/services/auth/custom-auth.html. [Geopend 2016].

[30] „docs.ionic.io," Ionic, [Online]. Available: https://docs.ionic.io/api/endpoints/push.html#post-notifications. [Geopend 2016].

[31] „developers.facebook.com," Facebook, [Online]. Available: https://developers.facebook.com/docs/graph-api/reference/event. [Geopend 2016].

[32] defunctzombie, „npmjs.com," [Online]. Available: https://www.npmjs.com/package/localtunnel. [Geopend 2017].

[33] „github.com," [Online]. Available: https://github.com/localtunnel/localtunnel/issues/156. [Geopend 2017].

[34] „ngrok.com," ngrok, [Online]. Available: https://ngrok.com/product. [Geopend 2017].

[35] mikermcneil, „npmjs.com," [Online]. Available: https://www.npmjs.com/package/waterline. [Geopend januari 2017].

[36] „github.com," [Online]. Available: https://github.com/balderdashy/sails-mysql/issues/172. [Geopend 2017].

[37] „docs.ionic.io," Ionic, [Online]. Available: https://docs.ionic.io/services/deploy/. [Geopend 2017].

[38] „mark.ie," [Online]. Available: http://www.mark.ie/articles/feed/json. [Geopend 2017].

[39] „developers.facebook.com," Facebook, [Online]. Available: https://developers.facebook.com/docs/plugins/page-plugin. [Geopend 2017].

[40] rajeshwarpatlolla, „github.com," [Online]. Available: https://github.com/rajeshwarpatlolla/ionic-datepicker. [Geopend 2017].

[41] „en.wikipedia.org," [Online]. Available: https://en.wikipedia.org/wiki/ELIZA. [Geopend 2017].

[42] „statista.com," statista, september 2016. [Online]. Available: https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/. [Geopend 2016].

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Integratie van social media services en chat bots in een online platform voor zwemclub ZVO

Richting: **master in de industriële wetenschappen: elektronica-ICT**
Jaar: **2017**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Timmermans, Jeroen

Datum: **18/01/2017**