Afstudeerverslag

Realtime Communication App

Student	Morsink M.D.
Studentnummer	2181277
Afstudeerrichting	ICT & Software (Voltijd)
Afstudeerperiode	3 februari t/m 25 juni 2014
Naam bedrijf/instelling	Info Support b.v.
Afdeling	Business Intelligence
Plaats	Veenendaal
Bedrijfsbegeleidster	Keurntjes M. Opleidingsadviseur
Docentbegeleidster	Heck P.
Titel	Realtime Communication App
Datum uitgifte	5-6-2014

Kruisboog 42, 3905 TG Veenendaal, Tel. +31 (0) 318 55 20 20 Fax +31 (0) 318 55 23 55
De Smalle Zijde 39, 3903 LM Veenendaal, Tel. +31 (0) 318 50 11 19 Fax +31 (0) 318 51 83 59
Rabobank Bergh 30.59.52.889, Postbank 47.38

Fontys Hogeschool ICT

InfoSupport

Afstudeerverslag

Realtime Communication App

Gegevens student:

Naam: Morsink M.D.
Studentnummer: 2181277
Afstudeerrichting: ICT & Software (Voltijd)
Afstudeerperiode: Van 3 februari t/m 25 juni 2014

Gegevens bedrijf:

Naam bedrijf/instelling: Info Support b.v.
Afdeling: Business Intelligence
Plaats: Veenendaal
Bedrijfsbegeleidster: Keurntjes M. Opleidingsadviseur

Gegevens docentbegeleidster:

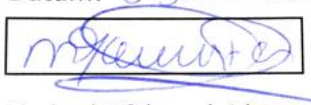
Naam: Heck P.

Gegevens verslag:

Titel afstudeerverslag: Realtime Communication App
Datum uitgifte afstudeerverslag: 5-6-2014

Getekend voor gezien door bedrijfsbegeleidster:

Datum: 5 juni 2014



De bedrijfsbegeleidster,

M. Keurntjes

Voorwoord

Dit afstudeerverslag is geschreven naar aanleiding van mijn afstudeerstage bij Info Support. Van begin februari tot en met eind juni 2014 heb ik mij verdiept in push-frameworks en heb ik een Proof of Concept ontwikkeld in de vorm van een mobiele veilingapp.

Ik heb een leuke, prettige en vooral leerzame tijd gehad bij Info Support. Voor toekomstige afstudeerders: Info Support is een fijne club vakmensen waar iedereen elkaar graag wil helpen. Er is veel kennis in huis en je kunt er ontzettend veel leren. Een aanrader!

De volgende personen wil ik graag bedanken voor hun bijdrage aan mijn afstuderen:

Casper Wind (Opdrachtgever, Info Support)
Bedankt voor het uitdagen van mijn competenties en het sturen naar een gewenst resultaat.

Willem Meints (Technisch begeleider, Info Support)
Bedankt voor het sparren over de technische obstakels en het uitdagen van mijn competenties.

Pascalle Hijl (Proces begeleider, Info Support)
Bedankt voor je persoonlijke begeleiding.

Marieke Keurntjes (Proces begeleider, Info Support)
Bedankt voor je persoonlijke begeleiding en het overnemen van de begeleiding van Pascalle.

Henk Brands (Business Unit Manager, Info Support)
Bedankt voor de opname in de unit Business Intelligence en de fieldmanagementgesprekken die we hebben gehad.

Petra Heck (Docent begeleider, Fontys Hogescholen)
Bedankt voor je feedback op mijn documenten en de begeleiding vanuit Fontys.

Roy Cornelissen (IT-Architect, Info Support)
Bedankt voor het schrijven van de leuke en uitdagende opdracht.

Evelien Frederiks (Recruiter, Info Support)
Bedankt voor de uitnodiging om te komen afstuderen bij Info Support en de prettige gesprekken.

Hans Isbrücker, Léon Bouquiet en Sander Wannet (Kamergenoten, Info Support)
Bedankt voor het meedenken over mijn opdracht en de geweldige sfeer op kamer 3.8.

Inhoudsopgave

Samenvatting

In de mobilemarkt gaan de technologische ontwikkelingen zeer snel. Een van de verschijnselen is de opkomst van apps die continu connected zijn, en realtime communiceren met een server. Dat wil zeggen dat informatie vanuit een server naar devices kan worden gepusht op het moment dat het beschikbaar wordt.

Er zijn inmiddels verschillende technologieën in de markt die push-communicatie vanuit een server mogelijk maken. Bovenop dit soort technologieën zijn push-frameworks beschikbaar die de implementatie vereenvoudigen, voorbeelden daarvan zijn Pusher en SignalR.

Info Support, een IT-dienstverlener, wilde graag meer ervaring opdoen met dit soort push-frameworks. Met deze ervaring kan Info Support innovatieve oplossingen met een realtime component ontwikkelen, waarmee Info Support haar klanten extra waarde kan bieden. Om ervaring op te doen en inzicht te krijgen in dit onderwerp is Info Support het afstudeerproject "Realtime communication app" gestart. In dit project is onderzoek gedaan naar beschikbare push-frameworks en de verschillen die er zijn. De centrale vraag die beantwoord is: "Welk push-framework kan Info Support het beste gebruiken voor het realtime pushen van informatie vanuit een server naar een mobiele app?".

Middels het onderzoek diende onderzocht te worden welke eisen Info Support stelt aan een push-framework en welke criteria belangrijk zijn bij het selecteren van zo'n framework voor een bepaalde oplossing. Op basis van de opgestelde criteria zijn de frameworks Parse Push, SignalR, Hydna, PubNub, Service2Media, BeaconPush, Pusher, PushSharp, Socket.IO en Azure Notifiactions Hub daarna vergeleken.

Uit het onderzoek bleek dat SignalR het best scoorde op de criteria, met dit framework is een Proof of Concept (PoC) gebouwd. Dit PoC moest aantonen dat dit push-framework geschikt is om in een praktijkoplossing te gebruiken. Het resultaat van het PoC is een Veiling app voor iOS en Android waarmee de werking van het push-framework SignalR wordt gedemonstreerd.

Als Info Support besluit push-communicatie te implementeren in een mobiele applicatie, kan er het best eerst gekeken worden welk push-framework het meest geschikt is. Dit kan op basis van de gestelde functionele requirements en de scoretabellen uit het onderzoek. Indien er geen specifieke eisen zijn, wordt SignalR aangeraden. Dit framework is snel, betrouwbaar, makkelijk te implementeren, gratis en ook voor web applicaties te gebruiken.

Op basis van de opgestelde criteria en hoe de frameworks scoren lijkt het voor Info Support niet interessant om zelf een push-framework te ontwikkelen.
De afleverzekerheid van SignalR en Azure Notification Hub was in alle tests 100%. Het kan wel interessant zijn om dit met een gesimuleerde slechte verbinding verder te testen, om te kijken wat voor invloed dit heeft.

SignalR laat in de testapp zien dat het transportmechanisme Server Sent Events wordt gebruikt voor push-communicatie. Volgens de documentatie van SignalR gebeurt dit wanneer er óf op de server, óf op het device geen Websocket transport beschikbaar is. Op zich is het positief dat SignalR zelf een geschikt transportmechanisme zoekt. Maar waarom er geen Websocket transport wordt gebruikt, en of dit überhaupt mogelijk is, is nog een belangrijke resterende vraag.

Summary

In the world of mobile, the technology develops very fast. One of the phenomena is the emergence of apps that are constant constantly connected and communicate in real-time with a server. Which means that information can be pushed from a server to devices, directly when it becomes available.

There are various technologies in the market that enable push-communication from a server. On top of these technologies there are push-frameworks available that simplify the implementation, examples are Pusher and SignalR.

Info Support, an IT services company, wanted to gain more experience with push-frameworks. With this experience, Info Support can develop innovative solutions with a real-time component, which allows Info Support to offer its customers additional value.

In order to gain experience and insight into this topic, Info Support started this 'Real-time communication app' project. In this project the available push-frameworks and the differences between them have been researched. The central question answered is: 'What push-framework is best used by Info Support for real-time pushing of information from a server to a mobile app?'.

Through a study there was to determine which demands Info Support proposes to a push-framework and what criteria are most important in choosing such a framework for a particular solution. The frameworks Parse Push, SignalR, Hydna, PubNub, Service2Media, BeaconPush, Pusher, PushSharp, Socket.IO and Azure Notifiactions Hub have been compared based on those criteria.

The outcome of the study showed, that SignalR was the most appropriate framework based on the criteria. Therefore SignalR was used to build a Proof of Concept (PoC). This PoC needed to show that this push-framework is suitable to use in practice. The result of the PoC is an auction app for iOS and Android that allows to demonstrate the operation of the push-framework.

If Info Support decides to implement push-communication in a mobile application, it's advised to look at what push-framework is most suited for a particular solution. This decision can be based on the specified functional requirements and the score tables of the research report. If there are no specific requirements, SignalR is recommended because it's a fast, reliable, easy to implement and free to use .NET framework and can also be used in web applications.

If you look at the criteria and the score of the frameworks, it doesn't seem interesting for Info Support to develop a push-framework itself. In all tests the delivery assurance of SignalR and Azure Notification Hub turned out to be 100%. What might be interesting though, is for Info Support to do further testing with a simulated bad connection to see what effect this has.

In the test app SignalR shows that the transport mechanism Server Sent Events is used for push-communication. According to the documentation of SignalR, this occurs when there is no Websocket transport available, either on the server or on the device. In itself it's a good thing that SignalR searches for an appropriate transport mechanism. But why no Websocket transport is being used, and if this is at all possible, still remains to be an important question.

Verklarende woordenlijst

.NET	Door Microsoft geschreven applicatieframework t.b.v. een samenwerking tussen applicaties geschreven in verschillende talen. .NET bestaat uit een groot aantal klassen die de ontwikkelaar kan gebruiken bij het schrijven van software.
Acceptance Test Driven Development	Een software ontwikkeltechniek waarbij eerst (vanuit de gebruiker gedacht) geautomatiseerde acceptatietests worden geschreven. Daar wordt de code pas geschreven om de test te laten slagen.
ASP.NET Identity Package	Het membership systeem voor .NET applicaties.
ASP.NET MVC Project	Een open-source web implementatie van het MVC patroon ontwikkeld door Microsoft.
Azure Cloud Service	Een service draaiend op Azure waarbij een WebRole en een WorkerRole een eigen IIS instantie krijgen.
Azure Notification Hub	Een hosted push-framework onderdeel van Windows Azure, de berichten worden via de infrastructuur van Azure, en via de platform notificatie service (PNS) van een mobiel operating system afgeleverd.
Backplane	Een backplane zorgt ervoor dat wanneer een applicatie over meerdere servers wordt gehost, een bericht via deze backplane naar alle servers wordt verspreid (Wasson & Patrick, 2014).
C#	Door Microsoft ontwikkelde programmeertaal, onderdeel van .NET.
Cross-platform	Wanneer een implementatie cross-platform is dan werkt deze op meerdere platformen.
Enterprise doeleinden	Bedoeld voor ondernemingen, niet bedoeld voor consumenten.
Git	Een gedistribueerd versiebeheersysteem voor code.
Git-TF	Tool om Git te combineren met Team Foundation Server.
HTTP protocol	Veelgebruikt protocol voor communicatie tussen een client en server.
IDE	Integrated Development Environment, een programma met een groot aantal samenwerkende tools om software programma's mee te ontwikkelen.
IIS	Internet Information Services, een verzameling serverdiensten bedoeld voor Windows machines. Ontwikkeld door Microsoft.

Lock	Een systeem waarbij threads op een andere thread moeten wachten tot een bepaald lock vrij is gegeven. Alleen de thread met het lock kan de taak uitvoeren.
Mock	Een nep-implementatie.
MS SQL	SQL variant van Microsoft.
MVC	Model, View, Controller, een software ontwerppatroon waarbij de verantwoordelijkheden van onderdelen duidelijk worden gescheiden.
Ontwikkelstraat Endeavour	Verzameling processen, tools en building blocks t.b.v. professioneel ontwikkelen van software. Ontwikkeld door Info Support (Info Support, 2014).
Payload	De hoeveelheid meegestuurde data van een bericht. Te vergelijken met de grootte van een attachment van een E-mail.
PL/SQL	Programmeertaal beschikbaar in Oracle databases.
PNS	Platform Notification Service. Service van een mobiel operating system waarmee push berichten kunnen worden verzonden naar gebruikers van een device met dat operating system.
Proof of Concept	Een hosted push-framework, de berichten worden via de infrastructuur van Pusher afgeleverd.
Pusher	Een hosted push-framework, de berichten worden via de infrastructuur van Pusher afgeleverd (Pusher, 2014).
Push-framework	Een framework wat de implementatie van push communicatie vereenvoudigt.
Realtime	Direct, zonder vertraging.
SCRUM	Een flexibele projectmethodiek om o.a. software te ontwikkelen waarbij er iteratief wordt gewerkt.
Server Sent Events	Een transportmechanisme waarmee push-communicatie te realiseren is.
SignalR	Een open-source C# push-framework, ontwikkeld door Microsoft.
Specification by Example	Een techniek waarbij er realistische business voorbeelden worden gebruikt om heldere specificaties te formuleren. Hierdoor wordt de kans vergroot dat de ontwikkelde software voldoet aan wat de klant verwacht.
Sprint	Een iteratie in SCRUM.

SSL	Encryptieprotocol waarmee communicatie via het HTTP protocol te beveiligen is.
Stored Procedure	Een vooraf opgeslagen functie toegevoegd aan een database die daarna herhaaldelijk is aan te roepen.
Team Foundation Server	Een Microsoft product te gebruiken voor o.a. versiebeheer, requirements management, project management, automatische builds etc.
Test code-coverage	De hoeveelheid code die wordt gedekt met testcode.
Test Driven Development	Een ontwikkelmethode voor software waarbij eerst tests worden geschreven en daarna pas de code.
Threads	Een mechanisme in software om meerdere taken tegelijktijdig uit te laten voeren.
Transportmechanisme	De techniek die wordt gebruikt om data van een systeem naar een ander systeem te transfereren.
T-SQL	Programmeertaal beschikbaar in Microsoft databases.
UML	Unified Modeling Language. Een veelgebruikte taal om software vanuit verschillende invalshoeken met diagrammen te beschrijven (Fowler, 2003).
Visual Studio	De IDE van Microsoft (Microsoft, 2014).
WebRole	Een IIS instantie in een Azure project dat verantwoordelijk is voor communicatie met de buitenwereld.
Websockets	Een transportmechanisme waarmee push-communicatie te realiseren is.
Windows Azure	Het cloud computing platform van Microsoft.
WorkerRole	Een IIS instantie in een Azure project dat verantwoordelijk is voor werk dat op de achtergrond moet worden uitgevoerd.
Xamarin	Bedrijf dat een .NET implementatie genaamd Mono op andere dan Windows architecturen laat draaien (Xamarin, 2014).
Xamarin Studio	De IDE van Xamarin (Xamarin, 2014).

1 Inleiding

Een veelgebruikt element in de interface van een mobiele app is een refreshknop, of een "pull-to-refresh" lijst waarmee de data in het scherm kan worden ververst. Dit zijn inmiddels achterhaalde oplossingen die hun bestaan danken aan het HTTP protocol, dat van oorsprong een pull model heeft (data wordt "gepulled" van een server).

Er is een grote opkomst van apps die continu connected zijn en realtime communiceren met een server. Zo kan informatie vanuit een server naar devices worden gepusht op het moment dat het beschikbaar wordt.

Inmiddels zijn er verschillende technologieën in de markt die push-communicatie vanuit een server mogelijk maken. Bovenop dit soort technologieën zijn push-frameworks beschikbaar die de implementatie vereenvoudigen, voorbeelden daarvan zijn Pusher en SignalR.

Info Support wilde graag meer ervaring opdoen met dit soort push-frameworks. Naar aanleiding daarvan is dit onderzoek gedaan naar beschikbare push-frameworks en de verschillen die er zijn. De hoofdvraag die beantwoord is: "Welk push-framework kan Info Support het beste gebruiken voor het realtime pushen van informatie vanuit een server naar een mobiele app?".

In dit afstudeerverslag wordt eerst het bedrijf waar de afstudeerstage heeft plaatsgevonden en de opdracht die tot het project geleid heeft, beschreven. Daarna komt de aanpak van het project aan bod, waarin de gebruikte projectmethodiek wordt toegelicht.

In het hoofdstuk Onderzoek wordt ingegaan op het doel van het onderzoek, hoe er te werk is gegaan en hoe de resultaten tot stand zijn gekomen.

Het hoofdstuk Proof of Concept beschrijft het proces en de keuzes die gemaakt zijn tijdens het ontwikkelen van het product.

Tot slot wordt er teruggekeken op de resultaten en wordt het project geëvalueerd.

2 Het bedrijf

Vanaf de oprichting in 1986 is Info Support uitgegroeid van een eenmansbedrijf tot een IT-dienstverlener met ongeveer 350 vakmensen in dienst. De werkzaamheden worden uitgevoerd vanuit vestigingen in Nederland en België.

Info Support zorgt voor softwareoplossingen die organisaties ondersteunen in het realiseren van hun bedrijfsdoelstelling. Het leveren van toegevoegde waarde voor de opdrachtgevers staat hierbij centraal.

Info Support biedt verschillende IT-oplossingen. Naast softwareontwikkeling en consultancy heeft het bedrijf ook een eigen ontwikkelstraat ontwikkeld genaamd Endeavour. Deze ontwikkelstraat wordt ingezet voor de eigen projecten maar ook aangeboden als oplossing voor andere ontwikkelaars. Ook heeft
Info Support een eigen kenniscentrum met IT docenten, om zowel eigen werknemers als externe klanten trainingen aan te kunnen bieden.

De software die ontwikkeld wordt is voornamelijk voor Enterprise doeleinden. Veel van de opdrachtgevers behoren tot de top 500 organisaties in Nederland en België.

Er zijn binnen Info Support vier businessunits. Iedere businessunit focust zich op een specifiek business domein. De vier businessunits zijn Handel & Industrie, Overheid, Finance en Zorg & Verzekeringen. Business Intelligence (BI) is een afdeling die meer over de andere units heen ligt, BI mensen kunnen ingezet worden in andere units om Data Warehouse oplossingen te ontwikkelen.

De teams die werkzaam zijn binnen een businessunit hebben specifieke kennis van die business, waardoor er beter kan worden meegedacht met de klant en er betere oplossingen worden ontwikkeld.

Figuur 1 Organisatiestructuur Info Support

Naast deze businessunits zijn alle professionals binnen Info Support ingedeeld in een Competence Center. Zo is er bijvoorbeeld het CCMSAD (Competence Center Microsoft Application Development) waar alle .NET ontwikkelaars lid van zijn. De Competence Centers hebben als doel om 'best practices', kennis en ervaring vanuit verschillende units en verschillende projecten te delen.

Ik was werkzaam op de hoofdlocatie in Veenendaal en tijdens mijn stage viel ik onder de businessunit Business Intelligence. Dit hield in dat ik als een werknemer van de unit werd beschouwd en ik bijvoorbeeld ook werd uitgenodigd voor het unitoverleg om mee te denken over de ontwikkelingen binnen deze afdeling.

Op 3 februari 2014 begon ik, samen met nog 31 andere studenten aan mijn afstudeerstage bij Info Support.

Figuur 2 Afstudeerders Info Support feb 2014

3 De opdracht

Voorafgaand was er al bekend dat er meerdere frameworks beschikbaar zijn om push-communicatie te realiseren. Het vergelijken en het maken van een afgewogen keuze voor één van deze push-frameworks is geen makkelijke taak. Elke framework heeft zijn voor- en nadelen, ook heeft elke oplossing zijn eigen specifieke behoeften.

In een literatuuronderzoek zijn verschillende frameworks onderzocht. Ook diende onderzocht te worden welke eisen Info Support stelt aan een push-framework en welke criteria belangrijk zijn bij het selecteren van zo'n framework voor een bepaalde oplossing. Op basis van de opgestelde criteria zijn de frameworks met elkaar vergeleken.

Vervolgens moest er op basis van de uitkomst van het onderzoek een Proof of Concept (PoC) met het meest geschikte framework gebouwd worden. Dit PoC moest aantonen dat dit push-framework geschikt is om in een praktijkoplossing te gebruiken.

Het PoC is een mobiele app geworden waarin een (eenvoudig) veilingscenario is uitgewerkt. In dit scenario is het mogelijk om realtime de actuele prijs van een veiling te volgen, te bieden, en indien er niemand hoger biedt, de veiling te winnen. De backend moest het winnende bod ontvangen en opslaan.

Op basis van het onderzoek en het PoC, wilde de opdrachtgever inzicht krijgen in het onderwerp. Er kan hierdoor in de toekomst een goede afweging worden gemaakt, welk push-framework, in welke situatie het beste gebruikt kan worden.

In welke taal en op welk platform het PoC ontwikkeld moest worden was niet vooraf vastgelegd. Daar kon, met goede argumenten, een voorstel voor worden gedaan. Ook de projectmethodiek mocht, mits beargumenteerd, zelf gekozen worden.

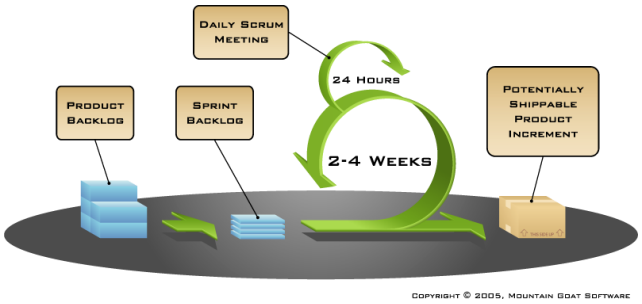
4 Aanpak

In dit hoofdstuk staat de aanpak van het project globaal beschreven. In Bijlage A: Plan van Aanpak is uitgebreider te lezen wat de initiële aanpak is geweest.

4.1 Projectmethodiek

Bij het schrijven van het Plan van Aanpak is een aanpak gekozen die goed bij de opdracht en omstandigheden past. Deze aanpak was iteratief en bevatte onderdelen van SCRUM.

"Scrum is een procesraamwerk dat sinds de jaren '90 gebruikt wordt om complexe productontwikkeling te managen. Scrum is geen proces of techniek voor het bouwen van producten; het is een raamwerk waar binnen je de verschillende processen en technieken kunt inzetten. Scrum maakt de effectiviteit van jouw productmanagement en ontwikkeltechnieken inzichtelijk zodat je die kunt verbeteren"
(Schwaber & Sutherland, 2011).



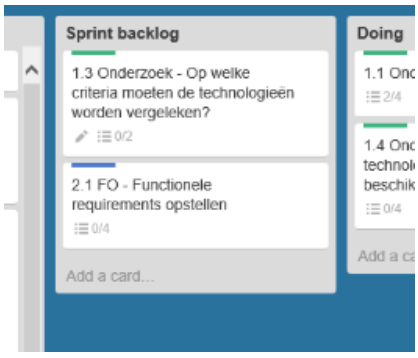
Figuur 3 Het SCRUM proces

4.2 Toepassing

Er is in sprints van 2 weken gewerkt, het hele project was één release.
De sprintreview en sprintplanning zijn in slanke vorm toegepast.

Niet alle onderdelen van SCRUM zijn toegepast, een daily stand up was uiteraard niet nodig wanneer er individueel gewerkt wordt en niet in een team.

Er is niet met user stories gewerkt maar er zijn requirements opgesteld waaruit use cases zijn geschreven. Hiervoor is gekozen omdat scenario's dan met meer detail uitgewerkt kunnen worden. Dit had met user stories, met een uitgebreide beschrijving, ook gekund maar daar is uiteindelijk niet voor gekozen omdat de opdrachtnemer meer ervaring had met use cases.



Figuur 4 Sprint backlog taken

De requirements en use cases zijn vertaald naar taken die, in overleg met de opdrachtgever, gedurende het project werden opgepakt (zie Figuur 4).

Hoe de scenario's van de use cases zijn beschreven wordt later toegelicht in het hoofdstuk Proof of Concept in het onderdeel Requirements en Functioneel Ontwerp.

Er is voor deze iteratieve aanpak gekozen, en bijvoorbeeld niet voor de traditionele watervalmethode, omdat eerdere ervaring heeft aangetoond dat het incrementeel opleveren van werk en veel contact met de opdrachtgever een goede invloed heeft op de verwachting en het resultaat.

Over de gehele looptijd is een fasering gelegd die opgedeeld is in de initiatiefase, onderzoeksfase, ontwikkelfase en afrondingsfase. Deze fasering is toegepast omdat er bepaalde zaken moesten worden afgerond voor er met het volgende kon worden begonnen. Zo moest het onderzoek (onderzoeksfase) eerst af zijn voor er met het PoC (ontwikkelfase) kon worden gestart. Een projectfasering is in SCRUM niet gebruikelijk, maar was voor een project met zoveel verschillende werkzaamheden als dit een duidelijke leidraad.

4.3 Communicatie

Na elke sprint werd het gemaakte werk aan de opdrachtgever gepresenteerd. Tijdens deze sessie werden de resultaten besproken en werd er teruggekeken op wat er goed ging en beter zou kunnen. Hierna werd de planning voor de komende sprint besproken. De opdrachtgever heeft door deze aanpak en het regelmatige contact goed kunnen sturen naar een gewenst resultaat.

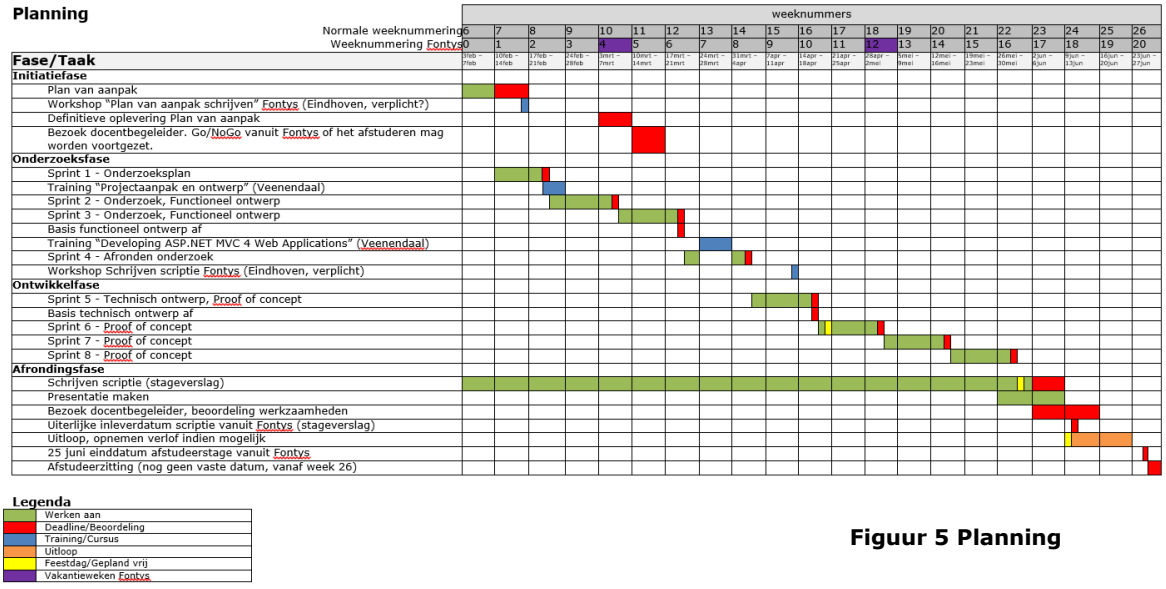
Alle betrokkenen ontvingen wekelijks op vrijdag een voortgangsupdate met de stand van zaken en nieuwe versies van de documenten.

Met de technisch begeleider vond om de week een meeting plaats om te sparren over de technische kant van het project. Dit heeft veel nuttige inzichten opgeleverd die later worden besproken in hoofdstuk 6.

Met de procesbegeleider, schoolbegeleider en Business Unit Manager is er ook regelmatig contact geweest om het proces van afstuderen te waarborgen.

4.4 Planning

Met de looptijd van het project en de producten die opgeleverd moesten worden, is de volgende planning opgesteld:



Figuur 5 Planning

(Grotere versie in Bijlage A: Plan van Aanpak)

Er is bewust gekozen om al na de eerste week, tijdens de onderzoeksfase, te starten met sprinten, om zo de routines van het iteratief werken al te stimuleren. Het onderzoek was prima op te delen in taken en er konden steeds nieuwe resultaten worden opgeleverd.

Het is een goede planning gebleken, deadlines zijn zonder stress gehaald en er zijn geen aanpassingen nodig geweest. Een uitzondering hierop is het schrijven van het afstudeerverslag, dit stond over het gehele project gepland. Uiteindelijk is er vanaf de 11e week besloten om hier op één vaste dag in de week aan te werken om de focus tijdens het schrijven te vergroten.

4.5 Uitvoering

Tijdens de uitvoering van het project, is er deels afgeweken van de aanpak beschreven in Bijlage A: Plan van Aanpak. Zo zou alle code en commentaar in het Engels geschreven worden, maar is later in overleg besloten om dit toch in het Nederlands te doen. De reden hiervoor is dat het bij Info Support gebruikelijk is dat de taal en het jargon van de business wordt aangehouden in ontwerp en code, zodat er geen ruis kan ontstaan door het gebruik van vreemde vertalingen.

Het doel was om de geschreven code te dekken met een test code-coverage van 80%. Het bleek dat het unittesten van sommige applicatiedelen, hoewel goed testbaar opgezet, door de complexiteit veel tijd kostte. Om deze redenen is besloten om het doel van 80% te laten varen en te testen waar de risico's het grootst waren en wat het meeste opleverde. Het schrijven van de unittests is op een TDD (Test Driven Development) wijze uitgevoerd. Dit is een ontwikkelmethode voor software waarbij eerst tests worden geschreven en daarna pas de code. Meer informatie over TDD is te lezen in het boek Test-Driven Development van Kent Beck, hij heeft de ontwikkelmethode in 2003 op papier gezet (Vliet, 2008).

5 Onderzoek

In dit hoofdstuk wordt samenvattend ingegaan op het onderzoekaspect van het project. Voor het volledige onderzoek wordt verwezen naar Bijlage B: Onderzoeksrapport.

5.1 Waarom was het onderzoek nodig?

Info Support had weinig ervaring met push-technologieën en wilde hier graag meer ervaring in opdoen. Hierdoor kan Info Support haar klanten extra waarde bieden met innovatieve oplossingen. Klanten verwachten namelijk steeds meer van mobiele apps.

Het onderzoek had als doel de volgende vraag te beantwoorden:

"Welk push-framework kan Info Support het beste gebruiken voor het realtime pushen van informatie vanuit een server naar een mobiele app?"

5.2 Wat is er onderzocht?

Om de hoofdvraag te kunnen beantwoorden zijn de volgende deelvragen beantwoord:

- Welke push-frameworks zijn er beschikbaar?
- Waar en voor welke doeleinden zou Info Support push-frameworks in kunnen zetten?
- Op welke criteria moeten de push-frameworks worden vergeleken?
- Weegt het zelf ontwikkelen van een push-framework op tegen de afhankelijkheid van het gebruiken van een bestaand framework?
- Hoe scoren de onderzochte push-frameworks op de voor Info Support belangrijke criteria?

Er is onderzocht hoe Info Support apps ontwikkelt en welke eisen Info Support stelt aan een "realtime" communicatietechnologie. Op basis daarvan zijn selectiecriteria opgesteld waarmee de gevonden push-technologieën zijn vergeleken.

Niet alle beschikbare push-technologieën konden met dezelfde diepgang worden onderzocht, het waren er simpelweg te veel. Er is gedurende het onderzoek met de opdrachtgever besproken welke technologieën er op basis van de selectiecriteria zouden afvallen en welke er nader onderzocht moesten worden.

5.3 Hypothese

De vooraf verwachte uitkomst van het onderzoek was:
"Als Info Support op een cross-platformwijze mobiele apps ontwikkelt, bijvoorbeeld met Xamarin, dan zal er een push-technologie gekozen moeten worden die goed past bij deze manier van ontwikkelen. Deze push-technologie zal moeten werken op alle mobiele platformen waarnaar geëxporteerd wordt. Omdat apps die met Xamarin ontwikkeld worden een .NET kern bevatten, zou SignalR, een .NET framework, weleens de beste optie kunnen zijn."

5.4 Uitvoering

Om alle vragen te kunnen beantwoorden is er gebruik gemaakt van de onderzoeksmethoden: deskresearch, interviews met experts en een experiment. Deze methoden zijn gekozen omdat het een kwalitatief onderzoek betrof.

Voor het onderzoeken van push-communicatie en de beschikbare push-frameworks, is deskresearch uitgevoerd. Ook voor de toetsing van de frameworks op een groot deel van de criteria was deskresearch het meest geschikt.

Het toetsen van de frameworks aan de overige criteria is gebeurd middels een experiment, omdat deze criteria alleen "runtime" te toetsen waren.

Om de deelvragen "*Waar en voor welke doeleinden zou Info Support push-frameworks in kunnen zetten?*" en "*Op welke criteria moeten de push-frameworks worden vergeleken?"* beantwoord te krijgen, zijn er interviews gehouden.

5.4.1 Toetsingscriteria

De criteria waaraan de push-frameworks getoetst moesten worden staan in tabel 1. Deze zijn in overleg met C. Wind en W. Meints, beiden werkend bij Info Support, tot stand gekomen. De kolomzwaarte geeft aan hoe belangrijk het criteria weegt voor Info Support en met deze percentages is later een gewogen score berekend.

NR	Beschrijving	Afkorting	Zwaarte [totaal 100%]
1	Is het framework crossplatform in te zetten met Xamarin?	**Xamarin**	Voor Info Support een eis
2	Is het framework schaalbaar in te zetten?	**Schaalbaarheid**	**20%**
3	Is de aflevering van een bericht gegarandeerd?	**Afleverzekerheid**	**15%**
4	Is versleutelde communicatie mogelijk?	**Versleuteling**	**15%**
5	Hoe snel worden berichten afgeleverd?	**Afleversnelheid**	**10%**
6	Wat voor eigen serverarchitectuur is er nodig?	**Benodigde serverarchitectuur**	**5%**
7	Verloopt de communicatiestroom via derden?	**Communicatie via derden**	**10%**
8	Kunnen berichten door het mobile device worden ontvangen als de app niet draait?	**Berichten ontvangen indien app niet actief**	**0%**
9	Hoe performt het framework op het mobile device wat betreft stroomverbruik?	**Stroomverbruik**	**5%**
10	Is de volgorde van aankomst van berichten gegarandeerd?	**Volgorde van aankomst**	**5%**
11	Is het framework te gebruiken in HTML/JavaScript?	**HTML/JavaScript**	**5%**
12	Wat kost het?	**Kosten/Licenties**	**5%**
13	Hoeveel Kb dataverkeer genereert het framework op de verbinding van het mobile device?	**Dataverkeer**	**5%**

Tabel 1

5.4.2 Scoreberekening

Er zijn uiteindelijk 10 push-frameworks gevonden en getoetst aan de criteria. De push-frameworks kregen, in een tabel, per criterium een score van 0 tot 10. Hoe er precies tot deze scores per criterium is gekomen, is te lezen in hoofdstuk 5 van Bijlage B: Onderzoeksrapport.

De toetsing verliep in twee stappen. Eerst werden de criteria getoetst die op basis van literatuur te toetsen waren. Hierna werd besloten om de meest interessante frameworks verder te toetsen op criteria die runtime getoetst moesten worden middels een experiment.

Daarna werd op basis van de zwaarte van de criteria een gewogen score bepaald. Dit heeft als voordeel dat de scores op lange termijn bruikbaar blijven en bovendien ook bruikbaar zijn voor derden die misschien andere wegingsfactoren hebben.

5.4.3 Herhaalbaarheid

Een belangrijk feit bij het runtime testen is dat het een momentopname is. Er is bij dit onderwerp een grote afhankelijkheid van het netwerk tussen server en het mobiele device, wat de testresultaten kan beïnvloeden. Hoe deze runtime criteria exact zijn getoetst is te lezen in de Bijlage "Runtime testopzet" van Bijlage B: Onderzoeksrapport.

Door het nauwkeurig beschrijven van de toetsing en testopzet is het hele experiment herhaalbaar geworden, wat belangrijk is bij kwalitatief onderzoek.

5.5 Onderzoeksresultaten

Na de toetsing van de push-frameworks op de criteria die op basis van literatuur waren te toetsen bleek dat SignalR, PushSharp en Azure Notification Hub direct te gebruiken zijn met Xamarin. Van deze drie had SignalR de hoogste totaalscore. Azure Notification Hub had, door de zwaarte van de criteria, de hoogst gewogen score voor Info Support.

PushSharp en Azure Notification Hub zijn vergelijkbaar in werking, maar de tweede scoort aanzienlijk hoger. SignalR en Azure Notification Hub zijn daarom in een runtimetest verder getoetst op de runtime criteria afleverzekerheid, afleversnelheid en volgorde van aankomst.

Tijdens de runtimetests hadden beide frameworks altijd een afleverzekerheid van 100%.

SignalR is veruit het snelste in het afleveren van een pushbericht, gemiddeld ruim 10 keer zo snel. Indien er een payload met het bericht moet worden meegestuurd is SignalR het meest geschikt, hiermee is een extra payload tot 47Kb mogelijk. Met Azure Notification Hub ligt de max bij een bericht naar een iOS device slechts op 256 bytes, een behoorlijk verschil. Wat betreft de volgorde van aankomst, scoort SignalR (93,5% correct) beter dan Azure Notification Hub (83,4% correct).

Na deze toetsing op runtime criteria is gebleken dat SignalR een hogere totaalscore had dan Azure Notification Hub. Met de zwaarte verrekend is het verschil echter klein. Het zijn echter totaal verschillende oplossingen en er zijn toepassingen te bedenken waar een van deze frameworks beter tot zijn recht zou komen dan de ander.

De exacte resultaten en verschillen zijn te lezen in het hoofdstuk 'De push-frameworks vergelijken' van Bijlage B: Onderzoeksrapport.

6 Proof of Concept

In dit hoofdstuk wordt het Proof of Concept (PoC) beschreven dat tijdens deze afstudeerperiode is ontwikkeld.

6.1 Doel

De definitie van een PoC luidt: "*Basisimplementatie om aan te tonen dat de voorgestelde oplossing in de praktijk te gebruiken is.*" (encyclio.nl, 2014).

Dit was dan ook het doel van het te ontwikkelen PoC. SignalR kwam in het onderzoek naar voren als een geschikt push-framework, het PoC had als doel deze ontdekking te onderbouwen.

Om het push-framework in een praktijksituatie te testen is besloten om een PoC te ontwikkelen in de vorm van een realtime veilingapplicatie.

Een globale samenvatting van wat het systeem moest kunnen is:

- Gebruikers kunnen inloggen met de app
- Het veilinghuis start automatisch veilingen
- Veilingen lopen af
- Gebruikers kunnen bieden op veilingen
- Gebruikers ontvangen realtime updates van wijzigingen

Er is gekozen voor een veilingapplicatie omdat er in een veilingscenario een grote afhankelijkheid is van het realtime bezorgen van data. Hierdoor zou het gekozen push-framework zich goed kunnen bewijzen.

6.2 Proces

6.2.1 Requirements en Functioneel Ontwerp

Om tot een PoC te komen wat aan de verwachtingen van de opdrachtgever zou voldoen, zijn er tijdens de onderzoeksfase requirements opgesteld. De functionele requirements en de constraints zijn opgenomen in Bijlage C: Requirements.

Op basis van de opgestelde requirements is het functioneel ontwerp ontstaan. Dit bevat het domeinmodel, het usecasediagram (Figuur 6) en de uitgeschreven usecases.

Figuur 6 Usecase diagram

In plaats van bij elke usecase textuele scenario's te beschrijven, zijn er "Given When Then" voorbeelden toegevoegd. Dit is een onderdeel van een techniek die Specification by Example (SBE) heet. Hierbij wordt er gebruik gemaakt van realistische business voorbeelden om heldere specificaties te formuleren (Adzic, 2014). Voor meer informatie over SBE is het boek 'Specification by Example: How Successful Teams Deliver the Right Software' van Gojko Adzic een goed startpunt.

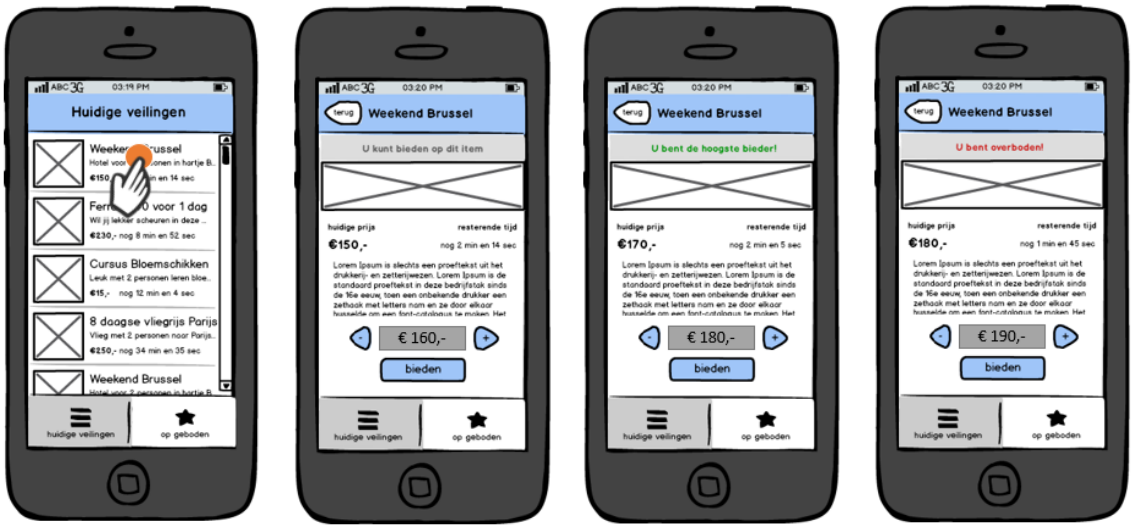
Een uitgewerkt Given When Then voorbeeld uit het Functioneel Ontwerp:

> ***Gegeven***
> *Jan heeft eerder op twee veilingen geboden.*
> ***Wanneer*** *Jan het scherm opent waar de veilingen bekeken kunnen worden waarop hij eerder heeft geboden…*
> ***Dan*** *ziet Jan de twee veilingen waarop hij eerder heeft geboden in lijst-vorm op het scherm.*

Het voordeel van deze werkwijze is dat er wordt afgedwongen om te schrijven in de taal van de klant. Als beide partijen dezelfde taal gebruiken, is er minder last van ruis en sluipen er dus ook minder fouten in de software als gevolg van vage specificaties.

Een ander voordeel is dat deze voorbeelden ook gebruikt kunnen worden als acceptatietests. Bij de techniek Acceptance Test Driven Development (ATDD) worden specificaties gekoppeld aan de daadwerkelijke code. De eerder genoemde Given, When, Then voorbeelden worden dan acceptatietests (Microsoft, 2013).

Een uitgebreide beschrijving alléén schetst wel een beeld van wat er te verwachten valt, maar laat nog geen visuele beleving zien. Om deze reden is er een conceptanimatie gemaakt waarin te zien is hoe de app gebruikt wordt door een gebruiker (Figuur 7).



Figuur 7 Schermafbeeldingen conceptanimatie

Door het tonen van deze animatie konden er, in een vroeg stadium, kleine functionele aanpassingen worden gedaan. Was deze animatie niet gemaakt, dan had dit aanpassen later in het project waarschijnlijk een grotere impact gehad.

6.2.2 Versiebeheer

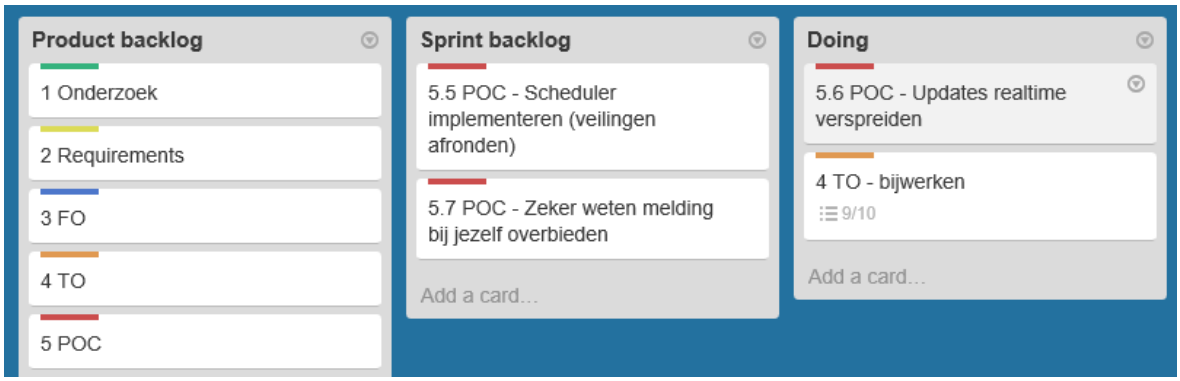
Team Foundation Server (TFS) is een software ontwikkel- en beheerpakket van Microsoft t.b.v. o.a. versiebeheer, requirement management, project management, automatische builds, testing enz (Microsoft, 2014). Bij Info Support is het gebruikelijk om .NET software te beheren met TFS. Info Support heeft TFS ook geïntegreerd in hun .NET ontwikkelstraat. Voor dit project is TFS gebruikt voor versiebeheer van de code.

Omdat de mobiele app een iOS versie bevatte werd de app ontwikkeld in Xamarin Studio (de IDE van Xamarin) op een Mac en niet op Windows in Visual Studio (over de keuze voor Xamarin meer in 6.4.2). Dit zorgde er wel voor dat de code niet meer aan TFS versiebeheer toe te voegen was. Dit is opgelost door een tool genaamd Git-TF te gebruiken, waarbij er via een lokale Git repository toch naar TFS ingecheckt kon worden.

6.2.3 Technisch ontwerp en realisatie

In de eerste sprint van de ontwikkelfase heeft de nadruk gelegen op het maken van de belangrijkste architectuurkeuzes. Dit waren de keuzes die vroeg gemaakt moesten worden omdat deze later moeilijk aan te passen zijn. Enkele van dit soort keuzes komen verder aan bod in 6.4 Ontwerp- en implementatiekeuzes.

Het verder technisch ontwerpen en ontwikkelen van het PoC was incrementeel en vond plaats tijdens de overige sprints in de ontwikkelfase. Steeds wanneer er weer een taak werd opgepakt (zie Figuur 8), werden de details van dat betreffende deel van het systeem ontworpen en gerealiseerd.



Figuur 8 Trello board

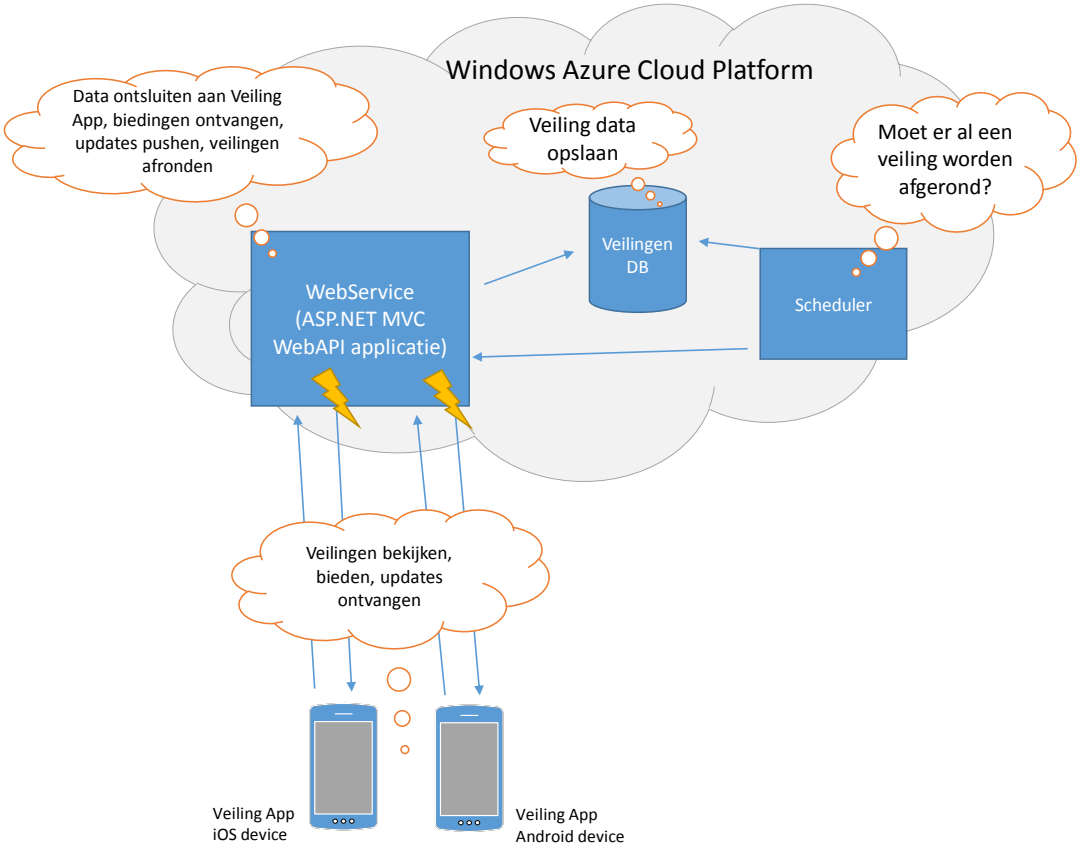
Bij het ontwerpen is gebruik gemaakt van de modelleringstaal UML (Unified Modelling Language). Dit is een veelgebruikte taal om software vanuit verschillende invalshoeken met diagrammen te beschrijven (Fowler, 2003).

Bij elk onderdeel is naar eigen inzicht een goed beschrijvend diagram toegevoegd. Dit waren vaak klassendiagrammen, maar soms was een sequence diagram of een niet-UML diagram beter geschikt om het ontwerp te verduidelijken.

In Bijlage D: Ontwerpdiagrammen zijn elke van deze diagrammen uit het Technisch Ontwerp opgenomen.

6.3 Globale werking systeem

Onderstaand ontwerp toont de onderdelen van het systeem en hun taken.



Figuur 9 Onderdelen van het systeem en hun taken

Er is gekozen om de WebService, de database en de Scheduler in Windows Azure (Azure) te hosten. Azure is een cloud hosting platform van Microsoft. De motivatie voor deze en andere technologiekeuzes wordt later in dit hoofdstuk toegelicht.

6.4 Ontwerp- en implementatiekeuzes

Hier volgen opvallende keuzes die gemaakt zijn in het ontwerp en problemen die zich voordeden tijdens de implementatie en de ontworpen oplossingen.

6.4.1 Microsoft ASP.NET MVC

Er is gekozen voor Microsoft technologie omdat Info Support vooralsnog een vooral Microsoft-georiënteerd bedrijf is. Ook is het SignalR framework, waar dit PoC omheen gebouwd is, een Microsoft product.

Een ASP.NET MVC project heeft als voordeel dat je met een paar klikken een werkende applicatie kan opzetten en dus snel van start kunt. Daarnaast is ook de MVC structuur al helemaal vooraf opgezet en klaar voor gebruik.

MVC (Model View Controller) is een veelgebruikt ontwerp patroon en zorgt voor structuur in software. Enkele voordelen van het gebruik van het MVC patroon is dat Models en Views herbruikbaar kunnen worden geschreven en de software makkelijk te onderhouden is (Dalling, 2009).

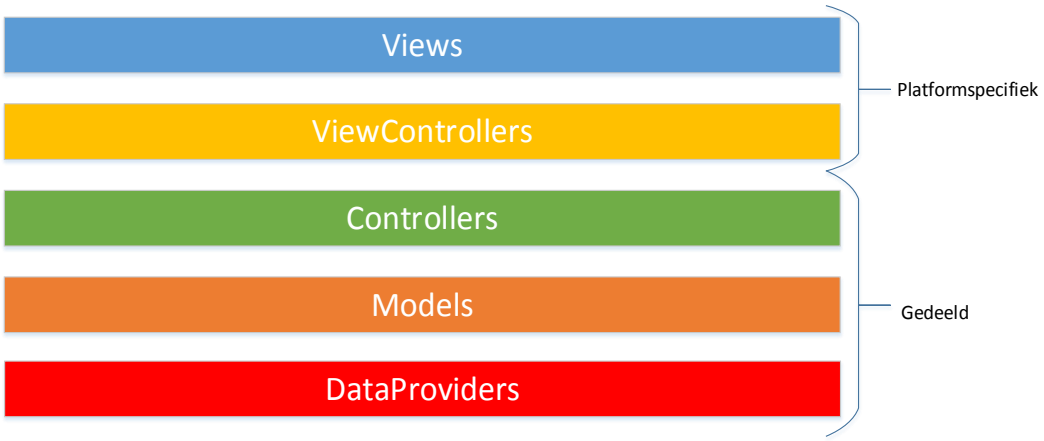
6.4.2 Xamarin

Info Support heeft als mobile strategie gekozen voor Xamarin. Ontwikkelen met Xamarin houdt in dat de code voor meerdere platformen gemeenschappelijk in C# wordt geschreven. Dit heeft als voordeel dat er een groot deel van de applicatie code hergebruikt kan worden op de platformen. Meer informatie over Xamarin is te lezen op https://xamarin.com/.

De mobiele Veiling App is voor de platformen iOS en Android ontwikkeld met Xamarin Studio. Een alternatief was geweest om de app op beide platformen native te ontwikkelen maar dit zou veel meer tijd hebben gekost. Ook zou de app minder goed te onderhouden zijn, al is dit waarschijnlijk niet nodig bij een PoC.

Ook bij de app is er voor een MVC structuur gekozen. Een native iOS en Android app hebben in de basis ook al een MVC structuur, waardoor het logisch is om deze structuur door te zetten.

Om zoveel mogelijk code binnen de app voor beide platformen te kunnen hergebruiken, is er zelf een bepaalde strategie bedacht. De volledige strategie is beschreven in Bijlage E: Codesharing strategie in Xamarin. Met hetzelfde doel (zo veel mogelijk code delen) is ook het lagenmodel van de app tot stand gekomen:



Figuur 10 Lagenmodel mobiele Veiling App

De View en ViewController-lagen bevatten voor elk platform een eigen deel. Views en ViewControllers zijn volledig platformspecifiek. In de Android app zijn ViewControllers Activities en Fragments, in de iOS app heten dit al ViewControllers. De ViewControllers bevatten geen business logica en zijn puur een doorgeefluik naar de controllers uit de laag 'Controllers'.

De Controllers in de 'Controllers'-laag zijn voor beide platformen. Elke controller wordt gekoppeld aan een ViewController d.m.v. een interface die de ViewController moet implementeren. Een controller bevat de business logica en verwerkt de input van een ViewController.

Doordat alle business logica in de gedeelde 'Controllers'-laag is gestopt, hoefde deze maar één keer geschreven en getest te worden. Ditzelfde geldt voor de Models en de DataProviders.

6.4.3 Schaalbaarheid

Er zijn twee typen van schalen mogelijk om de prestaties van een systeem te vergroten. De eerste is opschalen, waarbij de resources van een machine worden uitgebreid (denk aan snellere processor, meer geheugen etc.). De tweede is uitschalen, waarmee het bijplaatsen van machines wordt bedoeld (Michael, Moreira, Shiloach, & Wisniewski, 2007).

Doordat de Scheduler een aparte applicatie binnen het systeem is die bijhoudt wanneer veilingen moeten worden afgerond, wordt de WebService van deze taak ontlast. Hierdoor kan de WebService zich volledig bezig houden met het afhandelen van requests vanuit de Veiling App en is deze ook eenvoudig uit te schalen.

Ook de Scheduler zou uitgeschaald kunnen worden wanneer er geregeld wordt dat meerdere instanties van de Scheduler, het werk (afronden van veilingen) zouden verdelen.

Door de splitsing van de WebService en de Scheduler kunnen beide onderdelen dus onafhankelijk van elkaar worden geschaald, zodat er geen onnodige resources worden ingezet.

Om de WebService-applicatie daadwerkelijk uit te kunnen schalen moet er voor het SignalR gedeelte wel een backplane worden toegevoegd.
Een backplane zou ervoor zorgen dat meerdere instanties van de WebService applicatie de SignalR berichten allemaal ontvangen en versturen (Microsoft, 2014).

Het backend systeem is dus nog niet 100% schaalbaar in te zetten maar er zijn op architectuurniveau wel voorbereidingen voor getroffen.

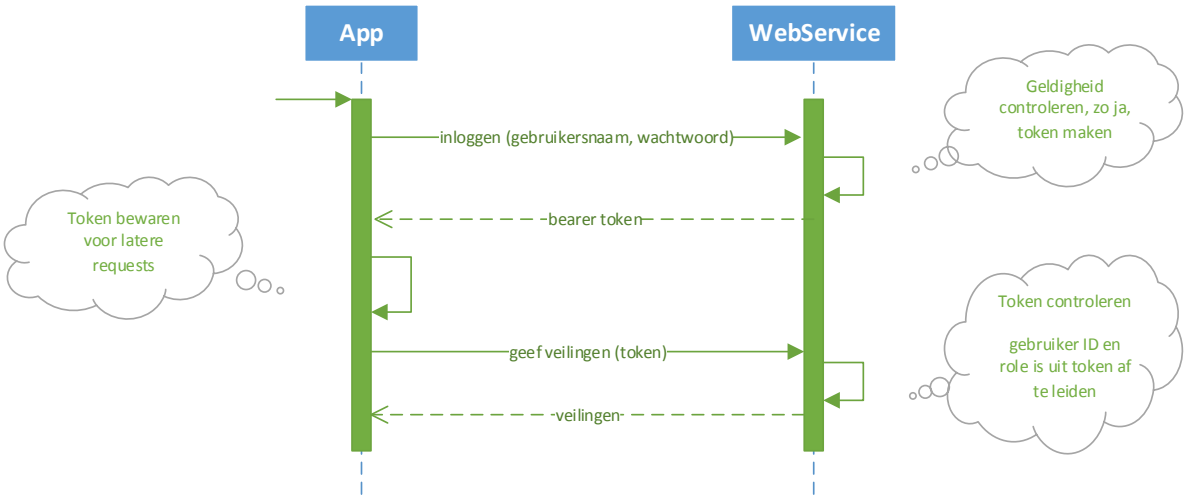
6.4.4 Windows Azure

Er is voor het hosten van de backend voor het platform Windows Azure gekozen, omdat dit veel installatie- en configuratiewerk uit handen neemt. Wanneer er zelf een Windowsserver ingericht en geconfigureerd zou zijn, was hier veel tijd in gaan zitten. Ook het deployen van het systeem zou meer werk zijn. Met Azure is het eerder genoemde schalen van onderdelen van een applicatie, eenvoudig in te stellen via de Azure Management Portal.

Het WebService project is als Azure Website gedeployed, omdat dit een lichtgewicht oplossing is en prima geschikt voor dit PoC. De Scheduler is als WebJob in Azure aan de website toegevoegd, eveneens een lichtgewicht oplossing. Een alternatief zou zijn om de WebService als WebRole en de Scheduler als WorkerRole in een Azure Cloud Service te laten draaien. Hier is nu niet voor gekozen omdat dit een veel zwaardere oplossing is, aangezien er voor de Web- en WorkerRole een aparte instantie van IIS wordt gereserveerd. Hierdoor zou deze oplossing ook duurder uitvallen.

6.4.5 Security

Het plaatsen van een bod mag alleen gebeuren door een geautoriseerde gebruiker, zodat een kwaadwillende geen bod kan plaatsen namens iemand anders. Dit wordt geregeld met het ASP.NET Identity package dat beschikbaar is voor ASP.NET MVC applicaties. Dit houdt in dat bij inloggen een token wordt verkregen wat bij volgende requests wordt meegestuurd. Dit heeft als voordeel dat er niet bij elk request een username, wachtwoord en/of ID overgestuurd hoeft te worden. Dit is veiliger.



Figuur 11 Inloggen

Belangrijk is dat het verkregen token niet te onderscheppen is door een kwaadwillende. Deze zou het token, tot deze verlopen is, kunnen misbruiken. Dit is voorkomen door de verbinding met SSL te versleutelen.

De Scheduler-acties op de WebService kunnen alleen door de Scheduler-applicatie worden aangeroepen. De Scheduler heeft een eigen user-account met de rol "Scheduler". Alleen als er is ingelogd met dit account kunnen de acties op de SchedulerController worden aangeroepen.

6.4.6 Afhandeling gelijktijdige biedingen

Om er voor te zorgen dat er per veiling altijd maar één hoogste bod kan zijn is er een mechanisme nodig dat dit regelt. Het zou bijvoorbeeld fout kunnen gaan als er in de WebService het huidige hoogste bod wordt opgehaald en er wordt gekeken of het nieuwe bod geldig is. Wanneer dit namelijk twee keer op precies hetzelfde moment met een even hoog bod zou gebeuren, is er een kans dat beide biedingen als geldig verklaard en opgeslagen worden.

Een oplossing zou kunnen zijn om runtime een static lijst met locks per veiling te bewaren, de thread die het lock van een veiling als eerste te pakken heeft zou dan als eerste zijn, waardoor andere threads in de wachtrij komen te staan. Dit zou het probleem oplossen, maar hiermee zou de schaalbaarheid verloren gaan. Als er namelijk twee instanties van de applicatie zouden draaien, zijn er ook twee static lijsten met locks, waardoor het mechanisme niet meer zou werken.

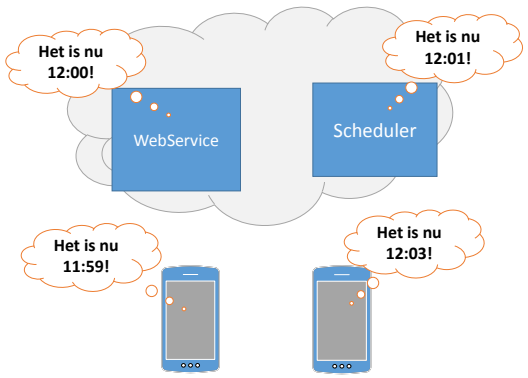
Beter is om dit op database-niveau te regelen met een "PlaatsBod" stored procedure. In deze stored procedure wordt de geldigheid van een bod gecontroleerd, waarna er een transactie wordt gestart en er een update lock wordt gelegd op het betreffende

veilingrecord. Vanaf nu kan een volgend bod op deze veiling, het record niet gelijktijdig updaten en wordt er een wachtrij gerealiseerd. Nadat er een update lock is gelegd, kan de geldigheid van het bod nogmaals worden gecontroleerd. Hierna kan het bod worden weggeschreven en de transactie worden afgesloten. Nu kan de update lock op het veilingrecord worden vrijgegeven. Het volgende bod kan dan op eenzelfde wijze worden afgehandeld.

Deze stored procedure is na te lezen in Bijlage F: Stored procedure PlaatsBod

6.4.7 Tijdsverschillen tussen devices en server

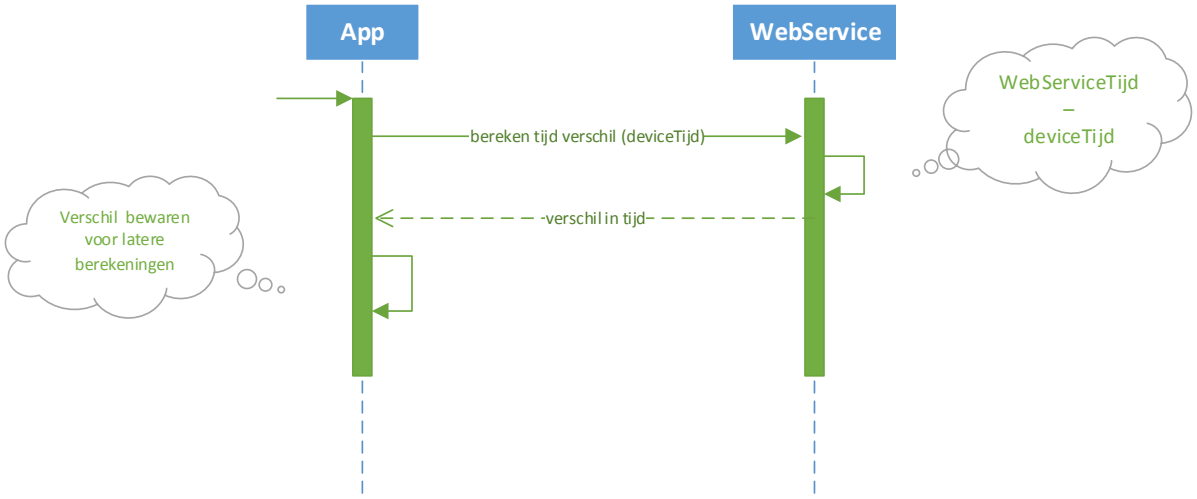
Wanneer een veiling afloopt, is het wenselijk dat alle biedende gebruikers dezelfde resterende tijd zien. Een probleem dat hier opspeelt is dat de tijd op de backend-server en de tijd op verschillende devices waar de app op draait, niet gegarandeerd gelijk lopen.



Figuur 12 Probleem tijdsverschillen

Een mogelijkheid om dit te omzeilen zou kunnen zijn om niet de tijd waarop een veiling afloopt te verspreiden, maar over hoeveel seconden dit is. Een groot nadeel hiervan is dat zodra deze info uit de database is gehaald, deze al verouderd is, waardoor er dus direct een foutmarge ontstaat.

Een betere oplossing voor dit probleem is om de WebService een functie te laten aanbieden waarmee andere devices kunnen opvragen hoeveel hun klok afwijkt van die van de WebService. Deze afwijking kan daarna in berekeningen met tijd worden meegenomen. Een nadeel van deze oplossing is dat de duur van het request voor enkele milliseconden afwijking zorgt, maar in het geval van een veilingscenario is dit geen probleem.



Figuur 13 Oplossing tijdsverschillen

6.4.8 Testbare code

Om code goed te kunnen unittesten, is het van belang dat klassen met afhankelijkheden van elkaar ontkoppeld worden. In dit project is dit gedaan door de klassen 'loosly coupled' te maken d.m.v. interfaces. De klassen waar een te testen klasse van afhankelijk is, kunnen dan eenvoudig vervangen worden door een 'mock object' (een nep implementatie).

Een alternatief om te ontkoppelen is dependency injection, een techniek waarbij ditzelfde bereikt wordt met veelal een speciaal hiervoor ontwikkeld framework (Chauhan, 2014). Er is in dit project niet gekozen voor deze manier van ontkoppelen omdat dit extra complexiteit met zich meebrengt en zo'n framework eerst bestudeerd moet worden voordat het goed gebruikt kan worden.

Een voorbeeld van ontkoppeling in dit project t.b.v. het testbaar maken van code:

```
public class VeilinghuisRepository {
    private IVeilinghuisDbContext _dbContext;

    public VeilinghuisRepository() {
        _dbContext = new VeilinghuisDbContext();
    }

    public VeilinghuisRepository(IVeilinghuisDbContext dbContext) {
        _dbContext = dbContext;
    }
}
```

VeilinghuisRepository gebruikt een object IVeilinghuidDbContext. In de eerste constructor wordt de echte implementatie aangemaakt. In de tweede constructor kan een vervangende implementatie of een mock worden meegegeven. In de testmethode hieronder wordt de tweede constructor gebruikt. Er wordt een mock implementatie meegegeven waardoor er geen echte database wordt aangeroepen en de klasse VeilinghuisRepository goed te testen is.

```
[TestMethod]
public void TestRepositoryGetVeiling() {
    //Arrange
    DynamicMock mockVeilinghuisDbContext = new DynamicMock(typeof(IVeilinghuisDbContext));

    VeilinghuisRepository repo = new VeilinghuisRepository((IVeilinghuisDbContext)mockVeilinghuisDbContext.MockInstance);

    // Act
    Veiling result = repo.GetVeiling(1L);

    //Assert
    //...
}
```

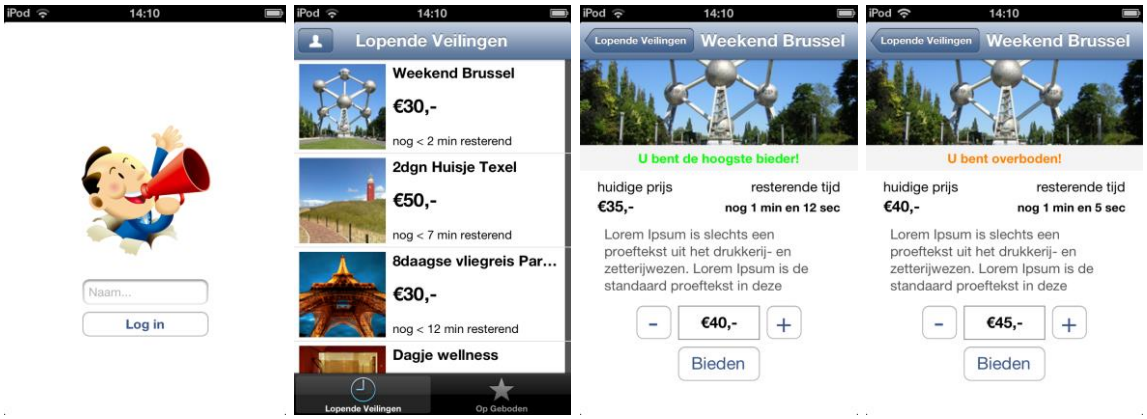
De testmethode is geschreven met behulp van NUnit, een .NET test framework dat functionaliteiten biedt om makkelijker te kunnen Unittesten (Poole, Terrell, & Busoli, 2007). DynamicMock is een klasse van NUnit om eenvoudig een mock te kunnen maken van een gegeven interface. MockInstance is een property van DynamicMock waarmee het mock-object kan worden gecast naar het 'te mocken' type.

6.5 Resultaat

Het PoC is een softwareproject gebleken waarbij redelijk wat technische obstakels moesten worden genomen. Uiteindelijk is voor elk obstakel een passende oplossing gevonden en is aan alle requirements voldaan. Maar belangrijker, de opdrachtgever heeft werkend bewijs gekregen dat SignalR een geschikt framework is voor softwareoplossingen die afhankelijk zijn van hun realtime component.

Het resultaat is een Veilingapp voor iOS en Android waarmee de werking van het push-framework SignalR wordt gedemonstreerd. Het PoC laat zien dat de uitkomsten uit het onderzoek kloppen. Veilingupdates worden zeer snel verstuurd naar de devices. Als de app gebruikt wordt wanneer het device is verbonden met een WiFi verbinding (wat sneller is dan een mobiele dataverbinding als 3G (Artman, 2014)), lijkt een update zelfs zonder enige waarneembare vertraging verspreid en ontvangen te worden.

Ook de juiste volgorde van aankomst en de hoge afleverzekerheid van SignalR blijkt uit het PoC. Zelfs na het uitschakelen van de verbinding, mits niet langer 5 minuten (wat instelbaar is), worden gemiste berichten nog in de juiste volgorde afgeleverd.



Figuur 14 Schermafbeeldingen iOS app

De overige schermafbeeldingen zijn te bekijken in Bijlage G: Schermafbeeldingen App.

6.6 Acceptatie

Nadat het PoC voltooid was heeft de opdrachtgever handmatig de acceptatie verricht. Hiervoor was een checklist opgesteld waarmee hij heeft getoetst of alle functionaliteiten en constraints aanwezig waren en correct werkten. In Bijlage H: Screenshot Acceptatie Test, is hiervan een screenshot opgenomen.

7 Conclusies en aanbevelingen

De doelstelling van dit project was om ervaring met push-technologie op te doen. Middels een literatuuronderzoek zijn verschillende frameworks onderzocht.
Ook diende onderzocht te worden welke eisen Info Support stelt aan een push-framework. Daarnaast moest er worden bekeken welke criteria belangrijk zijn bij het selecteren van zo'n framework voor een bepaalde oplossing. Op basis van de opgestelde criteria moesten de frameworks daarna worden vergeleken.

Vervolgens moest er op basis van de uitkomst van het onderzoek een PoC met het meest geschikte framework worden gebouwd. Dit PoC moest aantonen dat dit push-framework geschikt is om in een praktijkoplossing te gebruiken.
Op basis van het onderzoek en het PoC, wilde de opdrachtgever inzicht krijgen in het onderwerp.

7.1 Is de doelstelling behaald?

Het onderzoek heeft veel interessante kennis over het onderwerp opgeleverd. De verschillen tussen frameworks zijn duidelijk geworden en op basis van opgestelde criteria is het framework SignalR het meest geschikt gebleken.

Het resultaat van het PoC is een Veilingapp voor iOS en Android waarmee de werking van het push-framework SignalR wordt gedemonstreerd. Het PoC laat zien dat de uitkomsten uit het onderzoek kloppen.

Met de onderzoeksresultaten en het PoC heeft de opdrachtgever inzicht gekregen in het onderwerp push-technologie en de doelstelling is hiermee behaald.

7.2 Aanbevelingen

Info Support past nog niet veel push-communicatie toe in mobiele applicaties omdat de vraag van klanten nog beperkt is. Gebruikers verwachten intussen wel steeds meer van mobiele applicaties en het krijgen van notificaties is al vanzelfsprekend.

Uit een onderzoek van Urban Airship, een onderzoeksbureau, is gebleken dat het implementeren van push-communicatie kan leiden tot een flinke toename in het dagelijks openen van een app, 3x snellere responsetijd dan bij email en 30% toename in het delen van informatie op sociale netwerken. (O'Neill, 2012).

Dus de vraag is niet óf Info Support push-communicatie moet integreren in producten, maar hóe zij dit kan doen.

Als Info Support push-communicatie wil implementeren in een mobiele applicatie, is het advies om te kijken naar de functionele requirements en de scoretabellen van de frameworks. Op basis daarvan kunnen zij kijken welk push-framework het meest geschikt is. Als er geen specifieke eisen zijn, wordt SignalR aangeraden. Dit framework is snel, betrouwbaar, makkelijk te implementeren, gratis en ook voor web applicaties te gebruiken.

Op basis van de opgestelde criteria en hoe de frameworks scoren, lijkt het voor Info Support niet interessant om zelf een push-framework te ontwikkelen.

Uit het bouwen van het PoC is gebleken dat de gebruikte codesharing-strategie goed werkt. Zonder veel extra werk is dezelfde app voor de platformen Android en iOS

geïmplementeerd. Het wordt dan ook aanbevolen om deze of een soortgelijke strategie te gebruiken als er een multi-platform app wordt gebouwd met Xamarin.

7.2.1 Vervolgonderzoek

De afleverzekerheid van SignalR en Azure Notification Hub was in alle tests 100%. Maar het is nog wel aan te raden om dit met een gesimuleerde slechte verbinding verder te testen. Zeker als er een applicatie ontwikkeld wordt waarbij de gebruikers veel onderweg zijn, is het van belang om te weten wat een slechte verbinding met de afleverzekerheid doet.

De criteria stroomverbruik en dataverkeer zijn nog niet getest. Als Info Support een applicatie gaat ontwikkelen waarbij deze criteria zwaar wegen, is een vervolgonderzoek naar deze punten aan te raden omdat hier nu nog geen oordeel over kan worden gegeven.

Evaluatie

De gekozen aanpak is heel passend en prettig werkbaar gebleken. Er waren genoeg contactmomenten met de opdrachtgever waardoor hij goed heeft kunnen sturen naar een gewenst resultaat, zowel tijdens het onderzoek als bij het PoC.

De planning is gehaald en er zijn bij deadlines weinig tot geen stressmomenten ontstaan. De ingeplande weekuitloop op het einde was niet nodig en is gebruikt om alles netjes af te ronden en voor de voorbereiding op het afstuderen.

Als ik het project nu nog een keer opnieuw zou beginnen, zou ik niet heel veel aan de aanpak veranderen. Wel zou ik achteraf gezien, in plaats van 10, liever 2 à 3 push-frameworks vergelijken waardoor er meer tijd was geweest om dieper in een framework te duiken.

Ook zou ik een volgende keer nieuwe uitdagingen willen toevoegen. Zo ben ik geïnteresseerd geraakt in het eerder genoemde Acceptance Test Driven Development Het lijkt me heel interessant om hiermee aan de slag te gaan.

De meeste problemen had ik met het helder schrijven van documenten. Over de resultaten ben ik tevreden, maar het kostte me soms veel tijd en moeite om alles kort en helder te formuleren. Aan het eind van de stage, tijdens het schrijven van dit verslag ging het al makkelijker dan tijdens het schrijven van het onderzoeksrapport. Er zit dus vooruitgang in.

Tijdens het onderzoek ben ik in de fout gegaan bij het toewijzen van scores voor de criteria aan de push-frameworks. Ik gaf zelf cijfers tussen de 1 en 10, maar had dit niet onderbouwd en het was ook niet te herleiden waarom één framework ergens bijvoorbeeld een 10 voor kreeg en een ander framework een 9. Na de terechte feedback die ik hierop kreeg, heb ik dit recht kunnen zetten door eerst vast te leggen hoeveel punten een framework op elk criteria kon scoren en hoe deze score berekend werd. Daarna heb ik de scores opnieuw berekend. Door deze verandering was het voor lezers van het rapport wel te herleiden hoe een score tot stand kwam.

Tijdens de afstudeerstage heb ik mijn kennis vergroot met het bijwonen van een behoorlijk aantal workshops, trainingen en presentaties. Zo waren er door de procesbegeleiders presentaties ingepland om bijvoorbeeld het schrijven van een PVA, Functioneel- en Technisch ontwerp en scriptie te trainen. Ook waren er wekelijks op woensdagavond ISKA's (Info Support Kennis Avonden) ingepland waarin collega's allerlei interessante presentaties over onderwerpen binnen het ICT vakgebied gaven. De passie waarmee dit gedaan wordt is heel inspirerend en je leert er veel van.

Veel technologieën waar ik mee gewerkt heb waren nog nieuw voor mij. Ik had bijvoorbeeld al wel ervaring met mobile- en webdevelopment, maar nog niet met Xamarin en ASP.NET. Oracle databases en PL/SQL (een programmeertaal voor Oracle databases) kende ik al wel, maar MS SQL en TSQL (de Microsoft varianten van SQL en PL/SQL) nog niet. Van het platform Windows Azure had ik nog nooit gehoord. Het zelf leren kennen en verkennen van alle nieuwe technologieën en technieken waarmee ik heb gewerkt, heb ik als zeer positief ervaren. Ik denk ook dat het zelfstandig kunnen aanleren van nieuwe technieken, één van de belangrijkste competenties is van een Software Engineer, in een zich zo snel evoluerend landschap als de ICT.

Literatuurlijst

Adzic, G. (2014, juni 4). *Key Ideas*. Opgehaald van specificationbyexample.com/: http://specificationbyexample.com/key_ideas.html

Artman, J. (2014, mei 19). *3G Speed Vs. WiFi*. Opgehaald van E-How: http://www.ehow.com/about_5317627_vs-wifi-speed.html

Chauhan, S. (2014, feb 4). *Implementation of Dependency Injection Pattern in C#*. Opgehaald van Dot Net Tricks: http://www.dotnet-tricks.com/Tutorial/dependencyinjection/67LX120413-Implementation-of-Dependency-Injection-Pattern-in-C

Dalling, T. (2009, mei 31). *Model View Controller Explained*. Opgehaald van tomdalling.com: http://tomdalling.com/blog/software-design/model-view-controller-explained/

encyclio.nl. (2014, mei 8). *Proof of concept*. Opgehaald van encyclo.nl: http://www.encyclo.nl/begrip/proof%20of%20concept

Fowler, M. (2003). *UML Distilled.* Boston: Addison Wesley.

Info Support. (2014, juni 4). *Ontwikkelstraat Endeavour*. Opgehaald van infosupport.com: https://www.infosupport.com/Ontwikkelstraat

Info Support. (2014, feb). *Over Info Support*. Opgehaald van infosupport.com: http://www.infosupport.com

Michael, M., Moreira, J. E., Shiloach, D., & Wisniewski, R. W. (2007). Scale-up x Scale-out: A Case Study using Nutch/Lucene. *IEEE International Parallel and Distributed Processing Symposium*, (p. 1).

Microsoft. (2013, september 15). *Acceptance Testing Driven Development (ATDD) Use SpecFlow*. Opgehaald van blogs.msdn.com: http://blogs.msdn.com/b/qingsongyao/archive/2013/09/15/acceptance-testing-driven-development-atdd-use-specflow.aspx

Microsoft. (2014, mei 20). *Learn About ASP.NET SignalR*. Opgehaald van asp.net: http://www.asp.net/signalr

Microsoft. (2014, mei 23). *Team Foundation Server*. Opgehaald van Microsoft Developer Network: http://msdn.microsoft.com/en-us/library/ff637362.aspx

Microsoft. (2014, juni 4). *Visual Studio*. Opgehaald van visualstudio.com: http://www.visualstudio.com/

O'Neill, C. (2012, augustus 24). *How push notifications can boost your mobile strategy*. Opgehaald van econsultancy.com: https://econsultancy.com/blog/10596-how-push-notifications-can-boost-your-mobile-strategy-3

Poole, C., Terrell, J., & Busoli, S. (2007). *What Is NUnit?* Opgehaald van NUnit: http://www.nunit.org/

Pusher. (2014, mei 13). *Build awesome realtime web and mobile apps*. Opgehaald van pusher.com: http://pusher.com/

Schwaber, K., & Sutherland, J. (2011, oktober). *De Scrumgids.* Opgehaald van Scrum.org: https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum%20Guide%20-%20NL.pdf

Vliet, H. v. (2008). Software Engineering: Principles and Practice. In H. v. Vliet, *Software Engineering: Principles and Practice* (p. 421). Wiley.

Wasson, M., & Patrick, F. (2014, maart 7). *Microsoft ASP.NET*. Opgehaald van Introduction to Scaleout in SignalR: http://www.asp.net/signalr/overview/signalr-20/performance-and-scaling/scaleout-in-signalr

Xamarin. (2014, juni 4). *Build better apps for iOS, Android and Mac with Xamarin Studio*. Opgehaald van Xamarin.com: http://xamarin.com/studio

Xamarin. (2014, juni 4). *Xamarin*. Opgehaald van Xamarin.com: http://xamarin.com/

Bijlage A: Plan van Aanpak

Inhoudsopgave

Management samenvatting

Dit plan van aanpak is een leidraad voor de uitvoering van dit project.

In de mobile markt gaan de technologische ontwikkelingen zeer snel. Eén van de verschijnselen is de opkomst van apps die continu connected zijn en "real time" communiceren met een server. Er zijn verschillende technologieën in de markt waarmee een dergelijk scenario kan worden geïmplementeerd. Info Support bv wil graag ervaring opdoen met dit soort technologieën, zodat Info Support haar klanten extra waarde kan bieden met innovatieve oplossingen.

In een literatuuronderzoek zal een aantal push frameworks worden vergeleken. Vervolgens zal er in de vorm van een veilingapplicatie een proof of concept ontwikkeld worden. Hierbij wordt voor de implementatie van het real time aspect één framework gebruikt.

Op basis van het onderzoek en het proof of concept, wil de opdrachtgever inzicht krijgen in het onderwerp. Hij kan dan in de toekomst een goede afweging maken, welk push framework in welke situatie gebruikt zou kunnen worden.

Omdat dit project tevens de afstudeerstage van de opdrachtnemer betreft, zijn er 2 opdrachtgevers. Info Support bv is de primaire opdrachtgever van het daadwerkelijk te ontwikkelen product. Fontys Hogescholen is de secundaire opdrachtgever en heeft samen met de technisch begeleider het doel: het uiteindelijke afstuderen van de opdrachtnemer.

De betrokkenen in dit project zijn:

- Casper Wind, primaire opdrachtgever Info Support bv.
- Fontys hogescholen, secundaire opdrachtgever
- Menno Morsink, opdrachtnemer
- Willem Meints, technisch begeleider
- Pascalle Hijl, procesbegeleider
- Petra Heck, schoolbegeleider
- Henk Brands, business unit manager

Dit plan beschrijft naast de inhoudelijke opdracht, ook de afspraken die er zijn en de wijze van communiceren.

Het project loopt van 3 februari 2014 tot en met 25 juni 2014.

De gekozen aanpak is iteratief en bevat onderdelen van SCRUM. Over de gehele looptijd ligt een fasering die opgedeeld is in de initiatiefase, onderzoeksfase, ontwikkelfase en afrondingsfase.

De producten die ontwikkeld en opgeleverd zullen worden zijn:

- Plan van aanpak
- Onderzoeksplan
- Onderzoeksrapport
- Functioneel ontwerp
- Technisch ontwerp
- Proof of concept
- Scriptie (eindverslag)
- Eindpresentatie

1 Opdracht

1.1 Opdrachtgever en opdrachtnemer

1.1.1 Opdrachtnemer

De opdrachtnemer in dit project is:

Adres:	Info Support b.v. Kruisboog 42 3905 TG Veenendaal
Contactpersoon:	Menno Morsink +31 (0)653 412 569 menno.morsink@infosupport.com

1.1.2 Opdrachtgevers

De opdrachtgevers van dit project zijn:

1.1.2.1 Info Support bv (primair)

Adres:	Info Support b.v. Afdeling Business Intelligence Kruisboog 42 3905 TG Veenendaal
Contactpersoon:	Casper Wind +31 (0)645 786 662 casper.wind@infosupport.com

1.1.2.2 Fontys Hogescholen (secundair)

Adres:	Fontys Hogescholen ICT Software Engineering Rachelsmolen 1 5612 MA Eindhoven
Contactpersoon:	Petra Heck +31 (0)885 089 716 +31 (0)622 962 053 p.heck@fontys.nl

1.2 Opdrachtdefinitie

In de mobile markt gaan de technologische ontwikkelingen zeer snel. Een van de verschijnselen is de opkomst van apps die continu connected zijn, en "real time" communiceren met een server. Dat wil zeggen dat informatie naar devices kan worden gepusht op het moment dat het beschikbaar komt.

Er zijn verschillende technologieën in de markt waarmee een dergelijk scenario kan worden geïmplementeerd. Voorbeelden daarvan zijn Pusher en SignalR.

Info Support bv wil graag ervaring opdoen met dit soort technologieën, zodat Info Support haar klanten extra waarde kan bieden met innovatieve oplossingen. Onderzocht dient te worden welke criteria belangrijk zijn in zo'n push mechanisme.

In een literatuuronderzoek dienen een aantal push frameworks vergeleken te worden op basis van de criteria.

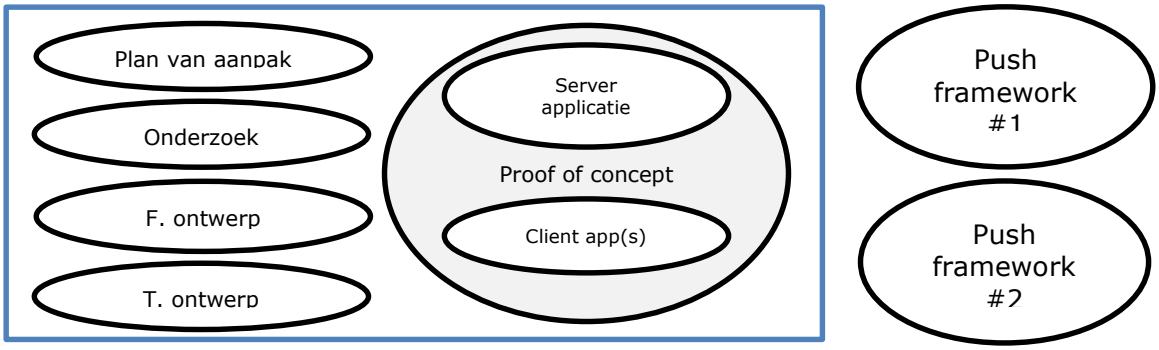
Vervolgens moet er op basis van de uitkomst van het onderzoek een proof of concept met het meest geschikte framework gebouwd worden.

Het proof of concept moet een app worden waarin een (eenvoudig) veilingscenario wordt uitgewerkt. In dit scenario moet het mogelijk zijn om real time de actuele prijs van een voorwerp te volgen, en een product te kopen. De back end moet het winnende bod ontvangen en opslaan.

Op basis van het onderzoek en het proof of concept, wil de opdrachtgever inzicht krijgen in het onderwerp. Hij kan dan in de toekomst een goede afweging maken, welk push framework in welke situatie gebruikt kan worden.

1.3 Afbakening

In het volgende context diagram laat de rechthoek zien welke onderdelen er opgeleverd zullen worden. De overige onderdelen spelen een rol, maar worden niet zelf ontwikkeld:



Wat wordt er **niet** opgeleverd?

1. Een push framework
2. Een volledig operationeel veilingsysteem in een productieomgeving

In de volgende subhoofdstukken worden de producten die gerealiseerd gaan worden nader toegelicht. Omdat er twee opdrachtgevers zijn zal er geprobeerd worden aan de eisen van beiden te voldoen. Indien dit niet mogelijk is zullen deze partijen een aparte versie geleverd krijgen.

1.3.1 Plan van aanpak

Het plan van aanpak heeft als doel duidelijkheid te scheppen over de aanpak van het project. Het bevat een duidelijke definitie, aanpakbeschrijving en planning. Dit document wordt naar beide opdrachtgevers gedistribueerd. Het project kan pas worden gestart wanneer beide opdrachtgevers akkoord gaan met de inhoud.

Als er een akkoord is op het plan van aanpak wordt het niet meer aangepast. Wanneer er na dit moment nog zaken veranderen qua aanpak, zal dit met memo's worden vastgelegd en verstuurd. Dit heeft als voordeel dat wijzigingen makkelijk te achterhalen zijn en het hele plan van aanpak niet steeds volledig doorgelezen hoeft te worden.

1.3.2 Onderzoeksplan

Het onderzoeksplan is het plan waarin de aanpak van het onderzoek beschreven staat. Dit document wordt gedistribueerd naar de primaire opdrachtgever. Wanneer deze een GO heeft gegeven kan er worden gestart met het onderzoek.

1.3.3 Onderzoeksrapport

Het onderzoeksrapport bevat de resultaten en adviezen die voortkomen uit het onderzoek. Ook dit document wordt gedistribueerd naar beide opdrachtgevers. Wanneer dit rapport door beide opdrachtgevers wordt geaccepteerd kan begonnen worden aan de ontwikkelfase van het project.

1.3.4 Functioneel ontwerp

Dit functioneel ontwerp zal in samenspraak met de primaire opdrachtgever worden opgesteld. Het beschrijft wat het proof of concept is, en kan. Dit document wordt gedistribueerd naar de primaire opdrachtgever.

1.3.5 Technisch ontwerp

In het technisch ontwerp zal beschreven worden welke technische keuzes er gemaakt zijn. Dit document wordt gedistribueerd naar de primaire opdrachtgever.

1.3.6 Proof of concept

Het proof of concept bestaat uit een server applicatie en één of meerdere client applicaties. Op basis van de uitkomsten van het onderzoek, zal er één push framework geselecteerd worden, waar het proof of concept gebruik van zal maken. Het functioneel en technisch ontwerp vormen samen de bouwtekening voor het proof of concept. Het zal aan de primaire opdrachtgever getoond worden.

1.3.7 Scriptie (stageverslag)

De scriptie, ook wel het stageverslag genoemd, is een document wat gedurende het hele project wordt samengesteld door de opdrachtnemer. Het beschrijft het hele proces wat de opdrachtnemer tijdens het project doorlopen heeft. In de afrondingsfase maakt hij er een helder geschreven document van volgens de richtlijnen van Fontys Hogescholen en levert deze op aan beide opdrachtgevers.

1.3.8 Presentatie

De presentatie wordt in de afrondingsfase door de opdrachtnemer ontwikkeld en geoefend. Tijdens de proefpresentatie bij Info Support wordt hierop feedback gegeven. De presentatie wordt als laatste gegeven tijdens de afstudeerzitting bij Fontys Hogescholen.

1.4 Afhankelijkheden

Dit project is afhankelijk van een aantal zaken. Deze zaken brengen risico's met zich mee die zijn beschreven in het hoofdstuk Risicomanagement.

De afhankelijkheden zijn:

1.4.1 Opdrachtgever

Beschikbaarheid van de opdrachtgever om tijdens het project voldoende sturing te geven aan de opdrachtnemer.

1.4.2 Begeleiders

Beschikbaarheid van de technisch-, proces- en schoolbegeleider. Deze personen sturen de opdrachtnemer en monitoren het afstudeerproces.

1.4.3 Push frameworks

De te onderzoeken, en later te gebruiken push frameworks. Er zijn push frameworks in de markt die niet alleen het framework (de code) leveren, maar de push communicatie volledig leveren als dienst. Dit wil zeggen dat de communicatie ook via hun servers loopt, wat een extra afhankelijkheid betekent.
Het kan dus grote gevolgen hebben wanneer de beschikbaarheid tijdens het project verandert.

1.5 Kwaliteitseisen

1.5.1 Algemeen

- Documenten worden geschreven in heldere Nederlandse taal.
- Indien beschikbaar worden templates gebruikt van Info Support bv.
- In een werkzaamhedenadministratie houdt de opdrachtnemer dagelijks bij wat er is gedaan.

1.5.2 Ontwerpen

- Diagrammen worden uitgewerkt volgens UML.

1.5.3 Code

- Code en commentaar worden geschreven in het Engels.
- Code wordt geschreven naar de codestandaarden van Info Support bv.
- Minimaal 80% van niet-UI code wordt getest d.m.v. Unit testen (UI code wordt gedekt met acceptatietesten).

1.5.4 Proof of concept

- Het proof of concept moet aantonen of het gebruikte push framework voldoet aan de gestelde criteria.
- Er zal een testplan gemaakt worden met testcases. Hierin zullen de use cases worden getest d.m.v. acceptatietests. In hoeverre dit geautomatiseerd kan en gaat worden, zal nog moeten worden overlegd met de technisch begeleider.

1.6 Uitgangspunten

Voor uitvoering van de in dit plan van aanpak beschreven delen zijn de onderstaande uitgangspunten van toepassing:

1. De opdrachtnemer voert het project intern uit bij Info Support op de locatie Kruisboog 42 te Veenendaal.
2. Alle onderdelen binnen de opdracht kunnen worden uitgewerkt vanuit de werkplek van de opdrachtnemer.

1.7 Randvoorwaarden

Aan uitvoering van de in dit plan van aanpak beschreven delen zijn de volgende randvoorwaarden gesteld:

1.7.1 Werkplek

Er is een werkplek beschikbaar voor de opdrachtnemer bij Info Support bv. Ook nog nader te bepalen ontwikkelsoftware is beschikbaar. Het beheer van de werkplek en deze software ligt bij de afdeling ICT Service van Info Support bv.

1.7.2 Beschikbaarheid opdrachtgever

Het is belangrijk dat de opdrachtgever voldoende beschikbaar is voor overleg en bespreking van de voortgang. Dit hoeft niet altijd in persoon te gebeuren maar kan ook via e-mail.

1.7.3 Beschikbaarheid opdrachtnemer

De opdrachtnemer dient gedurende de doorlooptijd van het project enkel aan dit project te werken. Hierop is ook de planning gebaseerd.

1.7.4 Kennis

Het is mogelijk voor de opdrachtnemer om tijdens het project een technische cursus te volgen, om de kennis over het onderwerp en de slagingskans te vergroten.

1.7.5 Budget

Er is voldoende budget beschikbaar voor de eventuele aanschaf van abonnementen/licenties van de push frameworks. Indien hier geen budget voor is kan er mogelijk ook met gratis proefaccounts worden gewerkt.

1.7.6 Infrastructuur

Er is een server(ruimte) beschikbaar om het proof of concept op te hosten. Ook benodigde devices voor de client app(s) dienen beschikbaar te zijn tijdens de ontwikkeling.

2 Aanpak

2.1 Algemeen

Voor het uitwerken van dit project zal er een aanpak worden gehanteerd die lijkt op SCRUM. Omdat het projectteam 1 persoon telt zijn een aantal tools uit SCRUM weggelaten. Er wordt alleen gebruikt wat een meerwaarde heeft voor het project. Zo wordt er bijvoorbeeld geen daily stand-up gehouden omdat de opdrachtnemer alleen werkt. Er zal wel gesprint worden, hierdoor heeft de opdrachtgever veel invloed op het uiteindelijk opgeleverde werk.

Er wordt geen onderscheid gemaakt tussen sprints en releases. Elke sprint is een release. Dit houdt in dat alle producten na elke sprint up-to-date zijn en kunnen worden opgeleverd of overgedragen.

In SCRUM is een user story een beschrijving van wat de gebruiker wil. In dit project zullen de beschrijvingen van wat de gebruiker wil worden opgenomen in het functioneel ontwerp als use cases. Tijdens een sprintplanning worden taken/onderdelen ingepland, geen user stories.

Tijdens elke sprint vinden in ieder geval de volgende activiteiten plaats:

Sprintplanning
Tijdens de sprintplanning plannen de opdrachtgever en opdrachtnemer in overleg de te ontwikkelen onderdelen in voor de komende sprint. Er zal gekeken worden naar het functioneel ontwerp, welke functionaliteiten de prioriteit krijgen. Dit is het moment voor de opdrachtgever (product owner) om zijn wensen te overleggen.

Sprintreview
Tijdens de sprintreview geeft de opdrachtnemer een demo aan de opdrachtgever van de ontwikkelde functionaliteit.

Retrospective
Daarna bespreken de opdrachtgever en opdrachtnemer het verloop van de afgelopen sprint en kijken ze wat er verbeterd kan worden in het proces.

De sprintplanning, sprintreview en retrospective zullen direct na elkaar plaatsvinden. Hierdoor is een contactfrequentie tussen opdrachtgever en opdrachtnemer eens per 2 weken voldoende.

2.2 Fasering

De gehele looptijd van het project (20 weken) is verdeeld over 4 fasen:

2.2.1 Initiatiefase

In deze fase maken de betrokkenen kennis en worden er afspraken gemaakt over het verdere verloop van het project. Het op te leveren mijlpaal product in deze fase is het Plan van aanpak.

2.2.2 Onderzoeksfase

In de onderzoeksfase wordt er begonnen met iteratief werken in sprints van 2 weken. In de eerste sprint, zal het onderzoeksplan worden uitgewerkt. Daarna wordt, per sprint, het onderzoek steeds verder uitgebreid zodat de opdrachtgever de richting van het onderzoek steeds kan bijsturen. Op deze manier krijgt hij precies de gewenste informatie over het onderwerp.

Naast de sprintplanning, review en retrospective worden de volgende activiteiten tijdens de sprints opgepakt:

Functioneel ontwerpen
Omdat de gewenste functionaliteit belangrijke input is voor het onderzoek, wordt er in deze fase al begonnen met het functioneel ontwerp.

Aan het eind van deze fase is het op te leveren mijlpaalproduct het onderzoeksrapport. De mogelijkheid bestaat wel dat het onderzoeksrapport tijdens de ontwikkelfase nog wordt bijgewerkt op basis van de ervaringen die worden opgedaan tijdens de ontwikkeling van het proof of concept.

2.2.3 Ontwikkelfase

Aan de hand van de uitkomst van het onderzoeksrapport zal de opdrachtnemer in de deze fase een proof of concept ontwikkelen. Dit mijlpaalproduct zal mede dienen als bewijs voor de conclusie uit het onderzoek.

Ook in deze fase wordt er iteratief gewerkt in sprints van 2 weken. Naast de sprintplanning, review en retrospective worden de volgende activiteiten tijdens de sprints opgepakt:

Technisch ontwerpen
De basis van het technisch ontwerp wordt in de eerste sprint van deze fase opgezet. Daarna wordt het eventueel nog uitgebreid aan de hand van de ingeplande onderdelen/taken per sprint.

Realiseren
Het proof of concept wordt verder ontwikkeld op basis van het functioneel en technisch ontwerp. Hierbij worden ook testen geschreven die, indien mogelijk, automatisch kunnen worden uitgevoerd.

Aan het eind van de ontwikkelfase zijn de mijlpaalproducten: het definitieve functioneel ontwerp, het definitieve technisch ontwerp en het proof of concept.

2.2.4 Afrondingsfase

In deze afrondende fase werkt de opdrachtnemer aan een final versie van het stageverslag en de eindpresentatie. Er zal een projectreview plaatsvinden om het project te evalueren. De mijlpaalproducten zijn: stageverslag en presentatie.

3 Risicomanagement

3.1 Risico's

Onderkende risico's met maatregelen ter beheersing
Voor uitvoering van de in dit plan van aanpak beschreven delen zijn de onderstaande risico's onderkend. Bij de risico's zijn de oorzaken en bijbehorende maatregelen ter beheersing opgenomen.

Nr.	Risico omschrijving				
	Oorzaak	Maatregel	P/S	Wie	Wanneer
1	**Een increment van het proof of concept is nog niet af tijdens levering.**				
	Er is te veel werk ingepland voor de sprint.	Sprintreview na elke sprint, volgende sprint minder inplannen.	P	Opdrachtgever, opdrachtnemer.	Wanneer er bij een sprintreview blijkt dat de planning niet is gehaald.
	Er is geen rekening gehouden met tussenkomende afspraken (zoals het volgen van een cursus).	Bij het maken van een sprintplanning rekening houden met andere afspraken. Stel de opdrachtgever op de hoogte.	P	Opdrachtnemer.	Wanneer er blijkt dat er in de bestaande planning iets moet worden ingevoegd.
2	**Het proof of concept zal gebreken vertonen in functionaliteit.**				
	Er is te weinig overleg met de opdrachtgever.	Regelmatig overleggen.	P	Opdrachtgever, opdrachtnemer.	Min 1 x per 2 weken.
	De wensen van de opdrachtgever zijn onvoldoende gespecificeerd vastgelegd.	De opdrachtgever laten bevestigen wat er de volgende sprint opgeleverd gaat worden.	P	Opdrachtgever, opdrachtnemer.	Bij start van elke sprint.
	Er is te weinig technische kennis bij de opdrachtnemer.	Advies in winnen bij technisch begeleider of een cursus volgen.	P,S	Opdrachtnemer.	Wanneer er blijkt dat er vereiste kennis ontbreekt.
3	**Het onderzoek of het proof of concept zal niet worden opgeleverd.**				
	De opdrachtgever is definitief niet meer beschikbaar.	Vervangende opdrachtgever vinden binnen Info Support bv.	S	Procesbegeleider.	Wanneer de opdrachtgever niet meer beschikbaar kan zijn.
	Onderdelen zijn verloren gegaan.	Backups maken en werken met een versiebeheersysteem.	P	Opdrachtnemer.	Gehele project.
	Het werk is niet af door ziekte van de opdrachtnemer.	In overleg met opdrachtgever de sprint schappen of doorschuiven.	S	Opdrachtgever, opdrachtnemer.	Wanneer er blijkt dat de opdrachtnemer langer dan 3 dagen ziek is.
4	**Het proof of concept voldoet niet aan de gestelde kwaliteitseisen.**				
	De kwaliteitseisen zijn niet helder vastgelegd.	Kwaliteitseisen per onderdeel duidelijk en concreet vastleggen.	P	Opdrachtgever, opdrachtnemer.	Tijdens het gehele project.
	Te weinig kennis bij de opdrachtnemer wat	Advies vragen aan de technisch begeleider.	S	Opdrachtnemer.	Wanneer de opdrachtgever

	betreft kwaliteit.				niet tevreden is met de geleverde kwaliteit.
5	**Het proof of concept is niet meer nodig.**				
	Uit het onderzoek blijkt dat Info Support bv geen baat heeft bij een proof of concept.	Uitzoeken wat er dan wel nodig is.	S	Opdrachtgever, opdrachtnemer.	Wanneer uit het onderzoek er geen baat meer is bij een proof of concept.
	Uit onderzoek blijkt dat er geen push framework bestaat dat aan de vastgestelde criteria voldoet.	Resultaat opnemen in onderzoek, de criteria bijstellen zodat er toch een proof of concept gerealiseerd kan worden.	S	Opdrachtgever, opdrachtnemer.	Wanneer uit het onderzoek blijkt dat er geen push framework aan de gestelde criteria voldoet.
6	**De vereiste infrastructuur is niet of maar deels beschikbaar.**				
	De afdeling ICT Services heeft de benodigdheden niet beschikbaar.	Zoeken naar alternatieven.	S	Opdrachtnemer, Technisch begeleider.	Wanneer blijkt dat de benodigde spullen niet beschikbaar zijn.
7	**Benodigd(e) push framework(s) tijdens het project niet bruikbaar.**				
	Bij de service van het framework heerst een storing.	De ervaring opnemen in het onderzoeksrapport. Verder werken aan niet afhankelijke componenten.	S	Opdrachtnemer.	Wanneer er een storing wordt geconstateerd.
	De leverancier van het framework is failliet en permanent niet meer beschikbaar.	Uitkomst opnemen in onderzoek, situatie bespreken met opdrachtgever, in opdracht verder focussen op overgebleven alternatieven.	S	Opdrachtgever, opdrachtnemer.	Wanneer de service van het framework permanent niet meer beschikbaar is.
8	**Het opgeleverde proof of concept zal fouten bevatten.**				
	Er zijn onvoldoende tests geschreven en uitgevoerd.	Schrijven van testen en uitvoeren hiervan opnemen in de taken per sprintplanning.	P	Opdrachtnemer.	Elke sprint.
	Er worden niet werkende onderdelen opgeleverd.	Alleen werkende, geteste onderdelen opleveren.	P	Opdrachtnemer.	Elke sprint.
9	**Het proof of concept is niet overdraagbaar.**				
	De kwaliteitseisen van code worden niet gehandhaafd.	Tooling gebruiken om statische kwaliteit van code te monitoren.	P	Opdrachtnemer.	Elke sprint.
	De kwaliteitseisen van documentatie worden niet gehandhaafd.	De technisch begeleider vragen om reviews.	P	Opdrachtnemer, technisch begeleider.	Elke sprint.
	Er ontbreekt een beschrijving van de in te richten omgeving.	Korte bondige readme file bijhouden met de specificaties van de omgeving(en).	P	Opdrachtnemer.	Elke sprint.

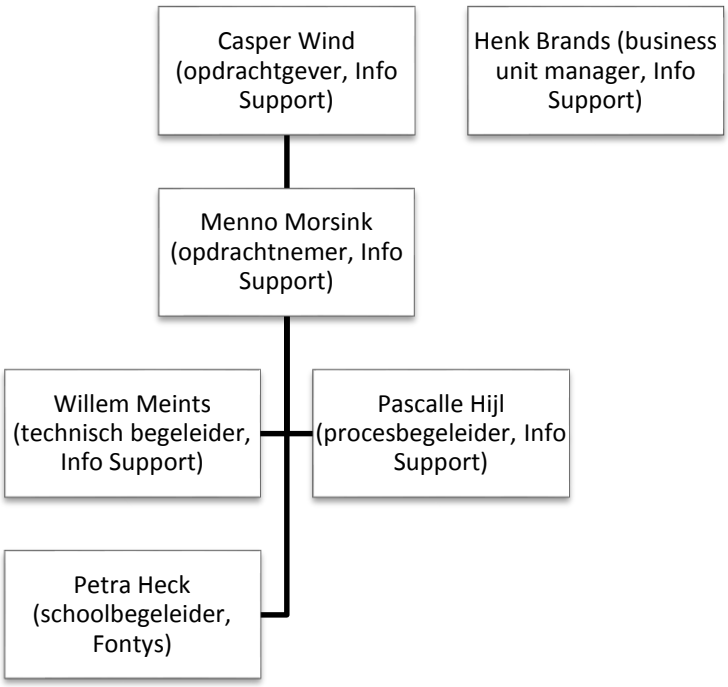
P/S = Preventief / Schadebeperkend

4 Beheersaspecten

4.1 Organisatie

4.1.1 Rollen

De personen die een rol spelen binnen dit project zijn in het volgende organogram weergegeven:



Opdrachtgever
De opdrachtgever is de klant. Hij is verantwoordelijke voor het duidelijk maken van de gewenste kwaliteit aan de opdrachtnemer. Hij stelt de eisen, bekijkt de mijlpaalproducten, geeft feedback en keurt het project goed. Tijdens het sprinten vervult de opdrachtgever de rol van product owner.

Opdrachtnemer
De opdrachtnemer (afstudeerder) is verantwoordelijk voor het goed opleveren van het project, en daarmee het tevredenstellen van de opdrachtgever.

Business Unit Manager
De business unit manager monitort de voortgang van het project. Voert fieldmanagementgesprekken met betrokkenen t.b.v. de beoordeling van de opdrachtnemer.

Technisch begeleider
De technisch begeleider is verantwoordelijk voor het de begeleiding van de opdrachtnemer op technisch/inhoudelijk vlak. Hij ondersteund bij het maken van het functioneel- en technisch ontwerp en beoordeeld deze documenten. Hij bepaald in overleg benodigde trainingen en beantwoord technische/inhoudelijke vragen van de opdrachtnemer.

Procesbegeleider
De procesbegeleider begeleid de opdrachtnemer tijdens het afstudeerproces met procesmatige zaken. Accordeert de weekstaten van de opdrachtnemer, controleert de voortgang en is tevens aanspreekpunt voor de schoolbegeleider.

Schoolbegeleider
De schoolbegeleider begeleid de afstudeerder vanuit het perspectief van de Fontys Hogescholen. Zij monitort de voortgang en controleert of er aan de gestelde eisen vanuit Fontys wordt voldaan. Zij is tevens aanspreekpunt van Fontys voor de andere rollen.

4.2 Informatie

4.2.1 Documentversies

Alle documenten krijgen per sprint, indien er wijzigingen zijn gemaakt, een nieuw versienummer.

4.2.2 Opslag van informatie

Alle documenten en data die tijdens dit project worden gegenereerd/verzameld worden opgeslagen op de netwerklocatie voor afstudeerders binnen Info Support bv. Van deze data wordt dagelijks automatisch een back-up gemaakt door ICT Services.

4.2.3 Vertrouwelijkheid van gegevens

Alle documenten en data die tijdens dit project worden gegenereerd/verzameld worden door de opdrachtnemer als vertrouwelijk beschouwd. Hij zal hier gepast mee omgaan en er zorg voor dragen dat er geen vertrouwelijke informatie op straat komt te liggen.

4.3 Tijd en voortgang

4.3.1 Algemeen

Het project start op maandag 3 februari en eindigt op woensdag 25 juni. Binnen deze periode is er de eis gesteld door Fontys Hogescholen dat er tenminste 85 dagen stage wordt gelopen. Eventuele trainingen en terugkomdagen op de Fontys Hogescholen tellen hierin ook mee.

4.3.2 Voortgangsupdate en controle

De opdrachtnemer stuurt wekelijks op vrijdag een voortgangsrapportage via de mail naar de andere rollen behalve naar de business unit manager. In deze rapportage staat welke werkzaamheden er die week zijn verricht, of er tegen problemen is aangelopen en wat er de komende week gedaan gaat worden. Tevens stuurt hij de gewijzigde mijlpaalproducten mee zodat de voortgang voor de andere rollen goed te monitoren is.

De eerder beschreven planning is leidend, aan de hand van deze planning en de voorgangsrapportages kan worden afgeleid of het project nog op schema ligt.

Planning

Fase/Taak	weeknummers																				
Normale weeknummering	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
Weeknummering Fontys	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	3feb – 7feb	10feb – 14feb	17feb – 21feb	24feb – 28feb	3mrt – 7mrt	10mrt – 14mrt	17mrt – 21mrt	24mrt – 28mrt	31mrt – 4apr	7apr – 11apr	14apr – 18apr	21apr – 25apr	28apr – 2mei	5mei – 9mei	12mei – 16mei	19mei – 23mei	26mei – 30mei	2jun – 6jun	9jun – 13jun	16jun – 20jun	23jun – 27jun
Initiatiefase																					
Plan van aanpak																					
Workshop "Plan van aanpak schrijven" Fontys (Eindhoven, verplicht?)																					
Definitieve oplevering Plan van aanpak																					
Bezoek docentbegeleider. Go/NoGo vanuit Fontys of het afstuderen mag worden voortgezet.																					
Onderzoeksfase																					
Sprint 1 - Onderzoeksplan																					
Training "Projectaanpak en ontwerp" (Veenendaal)																					
Sprint 2 - Onderzoek, Functioneel ontwerp																					
Sprint 3 - Onderzoek, Functioneel ontwerp																					
Basis functioneel ontwerp af																					
Training "Developing ASP.NET MVC 4 Web Applications" (Veenendaal)																					
Sprint 4 - Afronden onderzoek																					
Workshop Schrijven scriptie Fontys (Eindhoven, verplicht)																					
Ontwikkelfase																					
Sprint 5 - Technisch ontwerp, Proof of concept																					
Basis technisch ontwerp af																					
Sprint 6 - Proof of concept																					
Sprint 7 - Proof of concept																					
Sprint 8 - Proof of concept																					
Afrondingsfase																					
Schrijven scriptie (stageverslag)																					
Presentatie maken																					
Bezoek docentbegeleider, beoordeling werkzaamheden																					
Uiterlijke inleverdatum scriptie vanuit Fontys (stageverslag)																					
Uitloop, opnemen verlof indien mogelijk																					
25 juni einddatum afstudeerstage vanuit Fontys																					
Afstudeerzitting (nog geen vaste datum, vanaf week 26)																					

Het project loopt van 3 februari tot de afstudeerzitting in juni/juli. Zodra de definitieve datum voor deze zitting bekend is wordt de planning daarop aangepast. Tot die tijd wordt uitgegaan van de eerste mogelijkheid in week 26.

Legenda

	Werken aan
	Deadline/Beoordeling
	Training/Cursus
	Uitloop
	Feestdag/Gepland vrij
	Vakantieweken Fontys

4.4 Middelen

4.4.1 Benodigdheden per fase

Gehele looptijd project
De werkplek (locatie Kruisboog 42, kamer 3.8) is in beheer bij de afdeling ICT Services. De opdrachtnemer heeft de werkplek het gehele project nodig.

Toegang tot de te onderzoeken push frameworks. In samenspraak met de technisch begeleider zal de opdrachtnemer kijken of er voldoende kan worden onderzocht/ontwikkeld met gratis accounts of dat er betaalde accounts nodig zijn.

Initiatiefase
Geen specifieke middelen nodig.

Onderzoeksfase
Tijdens deze fase zijn er 2 trainingen ingepland. De training "Projectaanpak en ontwerp" en "Developing ASP.NET MVC 4 Web Applications" worden beiden door het Kenniscentrum van Info Support bv gefaciliteerd.

Ontwikkelfase
In deze fase kan het zijn dat de opdrachtnemer 2 test devices (Android + iOS) nodig om het proof of concept op te testen. In eerste instantie wordt er ontwikkeld met emulatoren. Wanneer er toch fysieke devices nodig blijken kan er met ICT Services overlegd worden of er devices van Info Support bv beschikbaar zijn.

Er is voor het ontwikkelen op een iOS device een developersaccount nodig is bij Apple. Het gebruikte device en de ontwikkelomgeving van de opdrachtnemer moeten dan gekoppeld worden aan dit account. Info Support bv heeft zo'n account en de opdrachtnemer zal samen met de technisch begeleider kijken welke acties er precies moeten worden uitgevoerd om van start te kunnen gaan.

Afrondingsfase
Geen specifieke middelen nodig.

4.5 Kwaliteit

4.5.1 Productkwaliteit

Om de productkwaliteit hoog te houden, zorgt de opdrachtnemer, dat documenten elke sprint up-to-date worden gehouden. Ook zal hij de ontwikkelde code dekken met unittesten en de use cases met acceptatietesten. De opdrachtgever zal na elke sprint beoordelen of het proof of concept het juiste demonstreert.

4.5.2 Proceskwaliteit

Om de proceskwaliteit te waarborgen stuurt de opdrachtnemer wekelijks een voortgangsrapport naar belanghebbenden. Mocht een van de begeleiders hierin opmerken dat er procesmatig iets niet goed dreigt te gaan, kunnen zij tijdig bijsturen.

4.6 Overlegvormen

Overleg tussen opdrachtnemer en opdrachtgever (sprintplanning, sprintreview, retrospective)
Dit overleg zal 1 keer per 2 weken in persoon plaatsvinden. De opdrachtgever maakt tijdens dit overleg zijn functionele eisen kenbaar. Het voorstel voor de locatie en tijd van het overlegmoment komt vanuit de opdrachtnemer. Tussendoor kan er mailcontact, en bij dringende zaken, telefonisch contact plaatsvinden. Wanneer de opdrachtnemer via de mail vragen stelt, geeft hij hierbij de gewenste feedbacktermijn op.

Overleg tussen opdrachtnemer en technisch begeleider
Dit overleg zal 1 keer per 2 weken in persoon plaatsvinden. Het voorstel voor de locatie en tijd van het overlegmoment komt vanuit de opdrachtnemer. Tussendoor kan er mailcontact, en bij dringende zaken, telefonisch contact plaatsvinden. Wanneer de opdrachtnemer via de mail vragen stelt, geeft hij hierbij de gewenste feedbacktermijn op.

Overleg tussen opdrachtnemer en procesbegeleider
Dit overleg zal 1 keer per 3 weken in persoon plaatsvinden. Het voorstel voor de locatie en tijd van het overlegmoment komt vanuit de begeleider. Tussendoor kan er mailcontact, en bij dringende zaken, telefonisch contact plaatsvinden.

Overleg tussen opdrachtnemer en schoolbegeleider
Er zal contact plaatsvinden via de mail wanneer dit nodig is. De opdrachtnemer stuurt de schoolbeleider tevens de mijlpaaldocumenten en houdt haar wekelijks op de hoogste van de voortgang. Wanneer de opdrachtnemer via de mail vragen stelt, geeft hij hierbij de gewenste feedbacktermijn op.

Overleg tussen opdrachtnemer en business unit manager
Eens in de 3 a 4 weken wordt de opdrachtnemer uitgenodigd door de business unit manager voor een voortgangsgesprek.

Overleg tussen procesbegeleider en schoolbegeleider
De schoolbegeleider zal 2 keer langskomen bij Info Support om de voorgang van het afstuderen te bespreken. Vanuit Fontys Hogescholen is er de wens dat er bij dit contactmoment minimaal de opdrachtnemer en de procesbegeleider aanwezig zijn. Eventueel kunnen de andere rollen ook bij dit overleg aanwezig zijn.

4.7 Communicatie

Fase/proces	Ondersteunen	Uitvoeren	Beslissen (goedkeuren)	Gebruiken (informeren)
Algemeen				
Aanvraag uitvoerende middelen	Technisch begeleider	Procesbegeleider		Opdrachtnemer
Weekstaten		Opdrachtnemer	Procesbegeleider	
Voortgangsrapportage		Opdrachtnemer		Opdrachtgever, Schoolbegeleider, Procesbegeleider, Technisch begeleider
Initiatiefase				
Plan van aanpak	Technisch begeleider, Schoolbegeleider	Opdrachtnemer	Schoolbegeleider, Opdrachtgever	Opdrachtgever, Schoolbegeleider, Business Unit Manager
Onderzoeksfase				
Onderzoekplan	Technisch begeleider	Opdrachtnemer	Opdrachtgever	Schoolbegeleider
Onderzoeksrapport	Technisch begeleider	Opdrachtnemer	Opdrachtgever	Opdrachtnemer
Ontwikkelfase				
Functioneel ontwerp	Technisch begeleider	Opdrachtnemer	Opdrachtgever, Technisch begeleider	
Technisch ontwerp	Technisch begeleider	Opdrachtnemer	Opdrachtgever, Technisch begeleider	
Proof of concept	Technisch begeleider	Opdrachtnemer	Opdrachtgever, Technisch begeleider	
Afrondingsfase				
Scriptie (eindverslag)	Schoolbegeleider	Opdrachtnemer	Schoolbegeleider	Business Unit Manager
Presentatie	Technisch begeleider	Opdrachtnemer		

5 Oplevering

Mijlpaalproduct	Eind opleverdatum
Plan van aanpak	7-3-2014
Onderzoeksplan	19-2-2014
Onderzoeksrapport	6-6-2014
Functioneel ontwerp	6-6-2014
Technisch ontwerp	6-6-2014
Proof of concept	6-6-2014
Scriptie (stageverslag)	6-6-2014
Presentatie	6-6-2014

Bijlage B: Onderzoeksrapport

Inhoudsopgave

Managementsamenvatting

Info Support past nog niet veel push communicatie toe in mobiele applicaties omdat de vraag nog beperkt is. Gebruikers verwachten intussen wel steeds meer van mobiele applicaties en het krijgen van notificaties is al vanzelfsprekend. Uit onderzoek is gebleken dat het implementeren van push communicatie kan leiden tot een flinke toename in het dagelijks openen van een app, 3x snellere response tijd dan bij email en 30% toename in het delen van informatie op sociale netwerken. De vraag is dus niet óf Info Support push communicatie moet integreren in producten, maar op welke manier zij dit kunnen doen.

Push-frameworks
Push-frameworks maken het integreren van push communicatie in mobiele applicaties een stuk makkelijker. Er zijn tien van deze frameworks getoetst op een aantal criteria. Deze criteria waren: schaalbaarheid, versleuteling, benodigde infrastructuur, communicatie via derden, berichten ontvangen indien app niet actief, beschikbaar in HTML/JavaScript, en kosten/licenties. Voor Info Support is het een eis dat het framework direct te gebruiken is met Xamarin. Aan de criteria die extra belang hebben voor Info Support is een zwaarte toegevoegd om naast een totaalscore ook tot een gewogen score te kunnen komen.

Drie smaken
Er zijn drie stromingen van oplossingen voor push communicatie naar mobiele devices:

- Oplossingen die push realiseren via het PNS (Platform Notification System).
- Oplossingen die een directe persistente verbinding tussen het device en de eigen server opzetten.
- Oplossingen die een persistente verbinding tussen het device en de server van het framework opzetten.

Transportmechanismen
De push-frameworks gebruiken onder de motorkap, een of meerdere van de volgende transportmechanismen om push communicatie te realiseren:

- Web Socket
- Server Sent Events
- Forever Frame
- Long Polling

Testresultaten
Na de toetsing van de push-frameworks op de criteria die op basis van literatuur waren te toetsen bleek dat SignalR, PushSharp en Azure Notification Hub direct te gebruiken zijn met Xamarin. Van deze drie had SignalR met 55 de hoogste totaalscore. Azure Notification Hub had met 4.0, door de zwaarte van de criteria, de hoogste gewogen score voor Info Support.

PushSharp en Azure Notification Hub zijn vergelijkbaar in werking en Azure Notification Hub scoort aanzienlijk hoger van deze twee. SignalR en Azure Notification Hub zijn daarom in een runtime test verder getoetst op de runtime criteria afleverzekerheid, afleversnelheid en volgorde van aankomst.

Tijdens de runtime tests hadden beiden frameworks altijd een afleverzekerheid van 100%. SignalR is veruit het snelste in het afleveren van een pushbericht, gemiddeld ruim 10 keer zo snel. Indien er een payload met het bericht moet worden meegestuurd is SignalR het meest geschikt, hiermee is een extra payload tot 47Kb

mogelijk. Met Azure Notification Hub ligt de max bij een bericht naar een iOS device zelfs op 256 bytes, een behoorlijk verschil. Wat betreft de volgorde van aankomst, scoort SignalR (93,5% correct) beter dan Azure Notification Hub (83,4% correct).

Na deze toetsing op runtime criteria blijkt dat SignalR met 80 een redelijk hoger totaal heeft dan Azure Notification Hub, met 64. Met de zwaarte verrekend is het verschil echter erg klein. Met 6.3 is SignalR net iets beter dan Azure Notification Hub met 6.0. Het zijn echter totaal verschillende oplossingen en er zijn toepassingen te bedenken waar een van deze frameworks beter tot zijn recht zou komen dan de ander.

	SignalR	Azure N.H.
Totaal	80	64
Info Support totaal (zwaarte zonder Xamarin)	**6,3**	**6,0**

Advies
Wanneer Info Support besluit push communicatie te implementeren in een mobiele applicatie, kan er het best eerst op basis van de gestelde functionele requirements en de score tabellen gekeken worden welk push-framework het meest geschikt is. Indien er geen specifieke eisen zijn gesteld, is SignalR aan te raden omdat het een snel, betrouwbaar, makkelijk te implementeren en gratis te gebruiken framework is wat ook in web applicaties te gebruiken is.

Op basis van de opgestelde criteria en hoe de frameworks scoren lijkt het voor Info Support niet interessant om zelf een push-framework te ontwikkelen.

Vervolgonderzoek
De afleverzekerheid was bij SignalR en Azure Notification Hub in alle tests 100%. Het kan interessant zijn om dit met een gesimuleerde slechte verbinding verder te testen, om te kijken wat voor invloed dit heeft.

SignalR laat in de testapp zien dat het transportmechanisme Server Sent Events wordt gebruikt voor push communicatie. Volgens de documentatie van SignalR gebeurt dit wanneer er, óf op de server, óf op het device geen websockets transport beschikbaar is. Op zich is het positief dat SignalR zelf een geschikt transportmechanisme zoekt. Alleen waarom er geen websockets transport wordt gebruikt, en of dit überhaupt mogelijk is, is nog een belangrijke resterende vraag.

De criteria stroomverbruik en dataverkeer zijn nog niet getest en zouden ook in een vervolgonderzoek meegenomen kunnen worden.

Verklarende woordenlijst

.NET	Een applicatieframework ontwikkeld door Microsoft.
AES	Advanced Encryption Standard, een versleutelingstechniek om data onleesbaar te maken.
Android	Mobiele OS van Google voor o.a. smartphones en tablets.
APNS	Apple Push Notification Service, push mechanisme van iOS.
Backplane	Een oplossing die, in het geval van een scale-out, de meerdere nodes met elkaar verbind. Zodat data bij alle nodes terecht komt.
BYOK	Bring Your Own Key, versleuteling waarbij de sleutel door de gebruiker/client zelf wordt gemaakt en bewaard.
Client	Applicatie of device dat communiceert met een server.
Client-side	De kant van de client.
Cloud	Het via internet beschikbaar stellen van hardware, software en services.
Enterprise applicaties	Applicaties bedoeld voor organisaties, niet voor consumenten.
Firewall	Systeem dat een netwerk beschermt tegen misbruik.
GCM	Google Cloud Messaging, push mechanisme van Android.
GitHub	Website waar groepen, gebruikmakend van het versiebeheersysteem Git, samen software kunnen ontwikkelen.
HTML5	De nieuwste HTML standaard.
HTTP	Hypertext Transfer Protocol, protocol voor communicatie tussen een webclient en een webserver.
HTTP POST	Een methode van HTTP waarbij extra data kan worden "mee-gepost".
iOS	Mobiele OS van Apple voor o.a. smartphones en tablets.
JVM	Java Virtual Machine.
Library	Een collectie van implementaties.
Mono	Open-source implementatie van .NET waarmee o.a. .NET code te draaien is op bijvoorbeeld Android en iOS.
MPNS	Microsoft Push Notification Service, push mechanisme van Windows Phone.
NuGet	Een package manager voor het .NET framework.
Open-source	Software waarvan de bron vrij beschikbaar is.
OS	Operating System.
Package	Een verzameling libraries.
Persistente verbinding	Een verbinding die na connectie open wordt gehouden.

PNS	Platform Notification Service, verzamelnaam voor diensten als APNS, GCM en MPNS.
Publish-Subscribe	Software architectuur patroon waarbij ontvangers zich abonneren op een bepaald kanaal. De zender publiceert berichten op een dit kanaal waarna de ontvangen alleen de berichten ontvangt waarop hij is geabonneerd.
Pull	Event waarbij data opgehaald wordt door een client bij een server.
Push	Event waarbij data verstuurd wordt vanuit een server naar een client.
Push-framework	Framework dat de implementatie van push communicatie vereenvoudigd.
Real time data	Data die beschikbaar is op het moment dat het ontstaat, ongeacht de locatie.
Redis	Een open-source softwaresysteem, dat in-memory key-value dataopslag beschikbaar maakt op een netwerk.
REST API	Een webservice die werkt volgens het REST (Representational state transfer) principe.
Runtime	Het moment waarop een applicatie wordt uitgevoerd.
Scale-out	Het uitbreiden (opschalen) van een systeem door het toevoegen van nodes.
SDK	Software Development Kit, verzameling hulpmiddelen die het mogelijk maakt om software te schrijven voor een bepaald systeem of doel.
Server	Centraal systeem/applicatie dat communiceert met clients.
Server-side	De kant van de server.
SSL	Secure Socket Layer, een encryptieprotocol om communicatie op het internet te beveiligen.
TCP	Transmission Control Protocol, een van de basisprotocollen van het internet.
TLS	Transport Security Layer, de opvolger van SSL. Een encryptieprotocol om communicatie op het internet te beveiligen.
UDP	User Datagram Protocol, een van de basisprotocollen van het internet.
Windows Azure	Cloud platform van Microsoft om virtuele machines, applicaties en services in de cloud te kunnen hosten.
WP7, WP8	Mobiele OS van Microsoft voor o.a. smartphones en tablets.
Wrapper	Een laag/schil over een software library heen die het aanspreken van de library mogelijk/makkelijker maakt.
Xamarin	Softwarepakket t.b.v. cross-platform mobile app ontwikkeling.

1 Inleiding

In de mobile markt gaan de technologische ontwikkelingen zeer snel. Een van de verschijnselen is de opkomst van apps die continu connected zijn, en "real time" communiceren met een server. Dat wil zeggen, informatie kan naar devices worden gepusht op het moment dat het beschikbaar komt. Er zijn verschillende technologieën in de markt waarmee zo'n scenario kan worden geïmplementeerd.

Info Support wil graag ervaring opdoen met dit soort push-technologieën, zodat zij haar klanten extra waarde kan bieden met innovatieve oplossingen. Klanten verwachten namelijk steeds meer van mobiele apps.

Dit onderzoek heeft het doel de volgende vraag te beantwoorden:

Welk push-framework kan Info Support het beste gebruiken voor het "real time" pushen van informatie vanuit een server naar een mobiele app?

Om hier een antwoord op te kunnen geven zijn de volgende deelvragen opgesteld:

Welke push-frameworks zijn er beschikbaar? Deze vraag wordt beantwoord in het hoofdstuk "Beschikbare frameworks".

Waar en voor welke doeleinden zou Info Support push-frameworks in kunnen zetten? Deze vraag wordt beantwoord in het hoofdstuk "Inzet push-framworks".

Op welke criteria moeten de push-frameworks worden vergeleken? Deze vraag wordt beantwoord in het hoofdstuk "Opzet voor vergelijking push-frameworks".

Hoe scoren de onderzochte push-frameworks op de voor Info Support belangrijke criteria? Deze vraag wordt beantwoord in het hoofdstuk "Push-frameworks vergelijken".

Weegt het zelf ontwikkelen van een push-framework op tegen de afhankelijkheid van het gebruiken van een bestaand framework? Deze vraag wordt beantwoord in het hoofdstuk "Is zelfbouw een optie?".

Tot slot is er de conclusie, die tot stand is gekomen met alle input van de eerder beantwoordde vragen.

2 Waarom zijn er push-frameworks?

Het versturen van "real time" informatie naar mobiele devices is sinds de komst van de smartphone de normaalste zaak van de wereld geworden. Voor die tijd hoorde je er nog niet veel over. Je zou kunnen spreken van een pull en een push tijdperk. In het pull tijdperk werd alle informatie opgevraagd van een server, tegenwoordig is het mogelijk om vanuit een server informatie te pushen naar devices.

Voordelen van push communicatie zijn: De gebruiker van een applicatie of systeem heeft de informatie direct wanneer deze beschikbaar komt. Daarnaast zijn er minder verbindingen nodig tussen een client en een server dan bij pull (Janssen, 2014). Omdat er minder verbindingen nodig zijn wordt er ook stroom bespaart (Blackberry, 2014).

Een onderzoek van Urban Airship toont aan dat het gebruik van push communicatie kan leiden tot:

- 540% toename in het dagelijks openen van een app.
- 3x snellere response tijd dan bij email.
- 30% toename in delen van informatie op sociale netwerken.

(O'Neill, 2012)

Push communicatie kan niet zomaar plaatsvinden, het is ingewikkelder om op te zetten dan pull communicatie. Bij pull communicatie kan een device gewoon een request versturen naar een server, het adres van deze server staat immers toch vast. Voor een server is het echter veel moeilijker om informatie naar een device te versturen. Een device heeft vaak geen vast adres op een netwerk en het device wacht ook niet op requests zoals een server dat doet. Ook houdt een firewall veel verkeer van buiten een netwerk tegen (Wikipedia, 2014).

Bij push communicatie wordt er, voordat er informatie beschikbaar komt, vanuit de client-side een verbinding geopend naar de server. Wanneer er informatie beschikbaar komt, kan de server deze direct over de al geopende verbinding versturen. (Wikipedia, 2014).

Het zelf opzetten van zo'n verbinding tussen client en server zou wel eens veel werk kunnen zijn. Er zijn veel zaken waar rekening mee moet worden gehouden, zoals bijvoorbeeld de beschikbare transport protocollen UDP of TCP, het opzetten en wegvallen van een verbinding, de hoeveelheid verbindingen die de server tegelijk aankan en welke platformen het moet kunnen gebruiken. Daarom zijn er push-frameworks ontstaan die dit soort zaken managen en het implementeren van push communicatie een stuk makkelijker maken.

3 Inzet push-frameworks

3.1 Mogelijke toepassingen

Welke mogelijkheden biedt push communicatie allemaal? Tijdens het doorlezen van de websites van een aantal push-frameworks, kom je een aantal voorbeelden tegen waar push communicatie voor ingezet kan worden:

- Chat
- Activity Streams
- Notifications
- Collaboration
- Presence of online users
- Multiplayer games
- Realtime data
- Dashboards
- 2nd Screen experiences
- Remote controlling

Een aantal toepassingen zoals chat en notificaties liggen erg voor de hand. Maar collaboratie is bijvoorbeeld een toepassing die erg nuttig kan worden ingezet bij bedrijfsapplicaties om communicatie en samenwerking te bevorderen. (Creative Bloq, 2013).

3.2 Wat doet Info Support al met push?

In interviews met Willem Meints en Henk Brands is gevraagd hoe Info Support apps ontwikkeld en wat voor scenario's er worden uitgewerkt. In deze interviews kwam het volgende naar voren:

Info Support ontwikkeld doorgaans Enterprise applicaties en geen apps die in de app stores beschikbaar zijn voor consumenten. De apps die ontwikkeld worden zijn vooral mobile extensies van administratieve systemen. Enkele voorbeelden van de scenario's die in de apps van Info Support gerealiseerd worden zijn: het inzien van een rooster, aanvragen van verlof en inspecties uitvoeren. Hierbij wordt offline data ingevoerd die later wordt gesynchroniseerd met de server. Het zijn geen standaard oplossingen die er ontwikkeld worden, het zijn vaak oplossingen voor specifieke business processen. (Meints, 2014).

Er zijn nog niet veel apps, ontwikkeld door Info Support, die scenario's implementeren d.m.v. push communicatie. De meeste apps zijn invoer gedreven. (Brands, 2014).

De vraag naar push functionaliteit is tot nu toe beperkt in de bestaande producten van Info Support. In het product KnowNow zouden comments en notificaties live gepusht kunnen worden naar een app. Ook het monitoren van serverlogs d.m.v. push notificaties is iets wat er mogelijk wel gaat komen. Dit zou het monitoren van allerlei zaken minder arbeidsintensief kunnen maken. (Meints, 2014).

3.3 Toekomst

Klanten, zoals mediabedrijven, zouden applicaties met een 2nd screen kunnen inzetten dat wordt aangestuurd d.m.v. push communicatie.
Mensen verwachten steeds meer van mobiele applicaties en het krijgen van notificaties is al vanzelfsprekend. Door websites als Gmail, Facebook en Twitter is het

intussen niet meer dan normaal dat je informatie krijgt toegespeeld op het moment dat het ontstaat. Dit lijkt een nieuwe standaard. (Meints, 2014).

De vraag is dus niet of Info Support push communicatie moet integreren in producten, maar op welke manier? Dit onderzoek helpt om te bepalen hoe push communicatie het beste door Info Support kan worden ingezet.

4 De techniek achter push communicatie

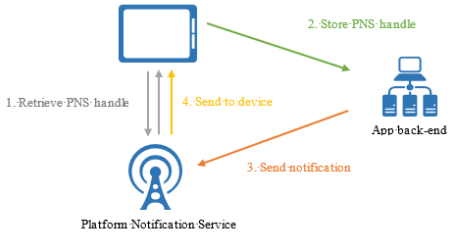
Hoe werkt push communicatie eigenlijk?

4.1 Drie smaken

Er zijn drie stromingen van oplossingen voor push communicatie naar mobiele devices:

4.1.1 Oplossingen die push communicatie realiseren via het PNS (Platform Notification System) van het device OS:

Een mobile operating system heeft zelf vaak een mechanisme aan boord t.b.v. push communicatie, ook wel PNS genoemd.



Bij Apple heet dit systeem APNS (Apple Push Notification Service). Een iOS device opent een beveiligde persistente verbinding met deze service. Over deze verbinding worden alle standaard push berichten verstuurd zoals mail en agenda notificaties. Ontwikkelaars kunnen via APNS ook gebruik maken van deze persistente verbinding voor het versturen van berichten. (Apple, 2014).

Google heeft eenzelfde soort service genaamd GCM (Google Cloud Messaging). Deze service heeft op een vergelijkbare manier een persistente verbinding met een Android device waarover push berichten kunnen worden afgeleverd. (Google, 2014).

Een belangrijke eigenschap is dat er dan push communicatie mogelijk is terwijl de app niet draait.

Elke PNS heeft ook zijn eigen maximale payload per bericht vastgesteld:

GCM (Android)	4 Kb
MPNS (Windows Phone)	3 Kb
APNS (iOS)	256 bytes

4.1.2 Oplossingen die een directe persistente verbinding tussen het device en de eigen server opzetten:

Dit biedt meer controle over de verbinding, en zou sneller kunnen zijn, maar heeft ook als nadeel dat de app geen berichten kan ontvangen als deze niet draait.

4.1.3 Oplossingen die een persistente verbinding tussen het device en de server van het framework opzetten:

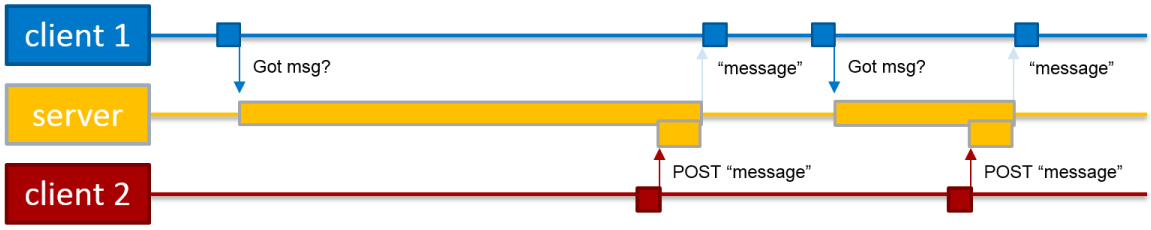
Hierbij wordt tussen het device en de server van het framework een persistente verbinding opgezet. Dit heeft als nadeel dat de app geen berichten kan ontvangen als deze niet draait.

4.2 Transportmechanismen

Push communicatie kan op meerdere manieren tot stand worden gebracht.

4.2.1 Long polling

Ook wel Ajax Long Polling genoemd, is eigenlijk geen push; het is een variant op de traditionele polling techniek waarbij HTTP requests van de client door de server worden open gehouden. Zolang de server geen update heeft, houdt deze de verbinding open door simpelweg nog geen response te versturen. Zodra de server een update voor de client beschikbaar heeft, beantwoord deze het request dat nog open stond. Telkens wanneer de verbinding is gesloten door een response, of een time-out, start de client weer een request. (Wikipedia, 2014)



(Cornelissen & Taling, 2013)
Deze techniek wordt ook gebruikt onder andere noemers zoals Pushlet.

4.2.2 Forever frame

Deze pushtechniek gebruikt een feature genaamd "chuncked encoding" uit de HTTP1.1 specificatie, om informatie van een server naar een client te sturen. Deze feature is origineel bedoeld om grote documenten (waarvan de server nog niet weet hoe groot), incrementeel over te sturen. De techniek wordt vaak in een "hidden iframe" HTML element in een webpagina verwerkt zodat de verbinding wordt gestart zodra de pagina is geladen. Deze verbinding is niet bi-directioneel. (Schiemann, 2007).

4.2.3 Websocket

Het Websocket protocol maakt bi-directionele communicatie tussen een client en server mogelijk op een enkele persistente verbinding. Het protocol wordt gestandaardiseerd door IETF (Internet Engineering TaskForce) en is nog in ontwikkeling.

Het begint met een "handshake" vanuit de client gevolgd door het over en weer sturen van berichten in via een TCP socket. Het doel van dit protocol is om een mechanisme te bieden wat niet vereist dat er meerdere HTTP connecties open blijven.
Een handshake ziet er als volgt uit:

Client HTTP request:

```
GET /mychat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat
Sec-WebSocket-Version: 12
Origin: http://example.com
```

Server HTTP response:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
```

(IETF, 2014)

De handshake is deels conform HTTP. Op deze manier wordt ervoor gezorgd dat de server zowel HTTP-verbindingen als Websocket verbindingen op dezelfde poort aankan. De velden die in de handshake staan en wat er na de handshake gebeurt is echter niet conform HTTP.

Het Websocket protocol is opgenomen in de HTML5 specificatie. Nog niet alle browsers ondersteunen dit volledig. (Wikipedia, 2014)

4.2.4 Server sent events

Server Sent Events (SSE) is een push communicatie patroon waarbij Server Sent Events worden gedefinieerd.

Er wordt een HTTP connectie geopend waarop alleen push berichten kunnen worden ontvangen, er wordt niets op gestuurd vanuit de client. Voor het verzenden vanuit de client is er een extra verbinding nodig. Omdat het aantal connecties naar een server vaak door browsers wordt gelimiteerd kan dit problemen opleveren. Hier is al wel een "work around" voor bedacht waarbij connecties naar een server kunnen worden gedeeld.

SSE gaat uit van een nieuw MIME-type, namelijk content-type:text/event-stream. Alleen browsers die dit content-type erkennen ondersteunen deze vorm van push communicatie.

Ook SSE is opgenomen in de HTML5 specificatie, maar wordt nog niet door alle browsers volledig ondersteund. (Roth, 2010)

SSE wordt gestandaardiseerd door W3C en is nog in ontwikkeling, al is er sinds december 2012 geen nieuwe versie meer verschenen. (W3C, 2012)

5 Opzet voor vergelijking push-frameworks

Dit hoofdstuk beschrijft de opzet van het onderzoek. De criteria waarop getoetst zal worden komen eerst aan bod. Daarna wordt uitgelegd hoe scores tot stand komen en de wijze waarop de criteria worden getoetst. Tot slot worden er een paar rekenvoorbeelden gegeven.

5.1 Criteria

De criteria zijn tot stand gekomen in overleg met C. Wind en W. Meints, beiden werkend bij Info Support. De kolom zwaarte geeft aan hoe belangrijk het criteria weegt voor Info Support. Met dit cijfer wordt later de gewogen score berekend.

NR	Beschrijving	Afkorting	Zwaarte [totaal 100%]
1	Is het framework crossplatform in te zetten met Xamarin?	**Xamarin**	**Voor Info Support een eis**
2	Is het framework schaalbaar in te zetten?	**Schaalbaarheid**	**20%**
3	Is de aflevering van een bericht gegarandeerd?	**Afleverzekerheid**	**15%**
4	Is versleutelde communicatie mogelijk?	**Versleuteling**	**15%**
5	Hoe snel worden berichten afgeleverd?	**Afleversnelheid**	**10%**
6	Wat voor eigen serverarchitectuur is er nodig?	**Benodigde serverarchitectuur**	**5%**
7	Verloopt de communicatiestroom via derden?	**Communicatie via derden**	**10%**
8	Kunnen berichten door het mobile device worden ontvangen als de app niet draait?	**Berichten ontvangen indien app niet actief**	**0%**
9	Hoe performt het framework op het mobile device wat betreft stroomverbruik?	**Stroomverbruik**	**5%**
10	Is de volgorde van aankomst van berichten gegarandeerd?	**Volgorde van aankomst**	**5%**
11	Is het framework te gebruiken in HTML/JavaScript?	**HTML/JavaScript**	**5%**
12	Wat kost het?	**Kosten/Licenties**	**5%**
13	Hoeveel Kb dataverkeer genereert het framework op de verbinding van het mobile device?	**Dataverkeer**	**5%**

Toelichting zwaarte

Het criterium Xamarin is een eis van Info Support en deze heeft daarom geen zwaarte gekregen. Wanneer deze bijvoorbeeld een zwaarte van 95% heeft zou dit een vertekenend beeld geven. Info Support is consulting-partner met Xamarin. Standaard bouwt Info Support alle mobile apps met Xamarin. Android, iOS, WP7 en WP8 worden dan ondersteund. Als een klant toch iets "echt" native wil laten ontwikkelen dan moet er een hele goede technische reden zijn waarom het niet met Xamarin ontwikkeld kan worden. Het kost namelijk meer tijd dan ontwikkelen met Xamarin. (Meints, 2014)

Schaalbaarheid (20%), is een zeer belangrijk criterium voor Info Support. Enterprise applicaties moeten zonder problemen kunnen groeien wanneer deze succesvol zijn.

Afleverzekerheid (15%), is een belangrijk criterium voor Info Support. De betrouwbaarheid van berichtenaflevering dat een push-framework biedt is een aspect waar de betrouwbaarheid van de hele applicatie van afhangt.

Versleuteling (15%), is ook een belangrijk criterium voor Info Support. Wanneer er met privacy gevoelige data wordt gewerkt kan het bij bepaalde oplossingen een eis zijn dat berichten onleesbaar verzonden moeten kunnen worden.

Afleversnelheid (10%). De snelheid waarmee een bericht wordt afgeleverd, bepaald mede de performance van een applicatie. Voor specifieke applicaties kan het zelfs bepalend zijn om wel of niet voor een push-framework te kiezen.

Communicatie via derden (10%). Het kan zijn dat vanuit privacy overwegingen een server van een derde partij voor het versturen van berichten niet gewenst is.

Berichten ontvangen indien app niet actief (0%), dit criterium heeft geen zwaarte gekregen omdat het een zeer specifiek functioneel onderdeel is. Er zal per applicatie gekeken moeten worden of dit een eis is.

De overige criteria hebben een zwaarte van 5% omdat het punten zijn die weliswaar van belang zijn, maar de keuze voor een framework door Info Support alleen zal beïnvloeden wanneer een applicatie specifieke eisen heeft.

5.2 Scoreberekening

De push-frameworks krijgen, in een tabel, per criterium een score van 0 tot 10. Daarna wordt op basis van de zwaarte van de criteria een gewogen score bepaald. Dit heeft als voordeel dat de scores op lange termijn bruikbaar blijven en bovendien ook bruikbaar zijn voor derden die misschien andere wegingsfactoren hebben.

5.3 Wijze van toetsing criteria

Het toetsen gebeurt in twee stappen. Eerst worden de criteria getoetst die op basis van literatuur te toetsen zijn. Hierna wordt besloten om de meest interessante frameworks verder te toetsen op criteria die "runtime" getoetst moeten worden.

Een belangrijk feit bij het "runtime" testen is dat het een momentopname zal zijn. Er is bij dit onderwerp een grote afhankelijkheid van het netwerk tussen server en het mobile device, wat de testresultaten kan beïnvloeden. Hoe deze criteria precies worden getoetst is te lezen in Bijlage C: Runtime testopzet.

Per criterium is aangegeven hoe een push-framework op dat punt getoetst wordt.

5.3.1 Xamarin

Bij het toetsen van dit criterium, zal op basis van de specificaties en documentatie van het push-framework gekeken worden of het framework gebruikt kan worden.

De score van een framework op dit criterium wordt op de volgende wijze bepaald:
0 – Voor helemaal geen support
5 – Voor support, maar met een eigen wrapper
10 – Voor out-of-the-box support

5.3.2 Schaalbaarheid

Bij het toetsen aan dit criterium zal, op basis van de specificaties en documentatie van het push-framework, gekeken worden of het framework rekening houdt met schaalbaarheid. Hierbij wordt gekeken naar scale-out mogelijkheden. Indien er geen expliciete informatie over te vinden is, kan er bij open-source frameworks gekeken worden wat er ongeveer nodig is om dit zelf realiseren.

De score van een framework op dit criterium wordt op de volgende wijze bepaald:
0 – Voor helemaal niet schaalbaar
5 – Voor schaalbaar na een eigen toevoeging
10 – Voor schaalbaar zonder dat hier iets voor gedaan hoeft te worden

5.3.3 Afleverzekerheid

Hoe dit criterium wordt getoetst is te lezen in Bijlage C: Runtime testopzet.

De score van een framework op dit criterium wordt op de volgende wijze bepaald:
0 – Voor < 80%
5 – Voor tussen 80% - 99%
10 – Voor 100%

5.3.4 Versleuteling

Om dit criterium te toetsen moet worden gekeken of het framework ondersteuning biedt voor versleuteling van berichten.

De score van een framework op dit criterium wordt op de volgende wijze bepaald:
0 – Voor versleuteling moet volledig zelf worden verzorgd
5 – Voor versleutelde verbinding of versleutelde berichten
10 – Voor zowel versleutelde verbinding als berichten

5.3.5 Afleversnelheid

Hoe dit criterium wordt getoetst is te lezen in Bijlage C: Runtime testopzet.

De score van een framework op dit criterium wordt op de volgende wijze bepaald:
0 – Voor gemiddelde aflevering > 3s
5 – Voor gemiddelde aflevering binnen 3s
10 – Voor gemiddelde aflevering binnen 0,5s

5.3.6 Benodigde serverarchitectuur

Bij het toetsen aan dit criterium, zal op basis van de specificaties en documentatie van het push-framework gekeken worden of een bepaalde infrastructuur benodigd is.

De score van een framework op dit criterium wordt op de volgende wijze bepaald:
0 – Indien niet toe te passen op een Windows serverarchitectuur
4 – Indien er een aparte server voor moet worden ingericht
9 – Indien er een externe service moet worden afgenomen
10 – Indien out of the box toe te passen in eigen backend draaiend op een Windows serverarchitectuur

5.3.7 Communicatie via derden

Om dit criterium te toetsen moet worden gekeken of de flow van berichten via een server van een derde partij loopt.

De score van een framework op dit criterium wordt op de volgende wijze bepaald:
0 – Wanneer de communicatie via meer dan 1 derde partij verloopt
5 – Wanneer de communicatie via 1 derde partij verloopt
10 – Wanneer de communicatie direct via eigen server naar client verloopt

5.3.8 Berichten ontvangen indien app niet actief

Bij het toetsen aan dit criterium, zal op basis van de specificaties en documentatie van het push-framework gekeken worden hoe het framework met dit aspect omgaat.

De score van een framework op dit criterium wordt op de volgende wijze bepaald:
0 – Voor geen communicatie mogelijk wanneer de app is afgesloten
10 – Voor wel communicatie mogelijk wanneer de app is afgesloten

5.3.9 Stroomverbruik

Om dit criterium te toetsen zal in verder onderzoek de testopzet in Bijlage C: Runtime testopzet moeten worden uitgebreid.

De score van een framework op dit criterium zou op de volgende wijze kunnen worden bepaald:
0 – Voor het framework dat de meeste stroom verbruikt tijdens de test
1-9 – Relatief aan minst en meest scorende
10 – Voor het framework dat de minste stroom verbruikt tijdens de test

5.3.10 Volgorde van aankomst

Hoe dit criterium wordt getoetst is te lezen in Bijlage C: Runtime testopzet.

De score van een framework op dit criterium wordt op de volgende wijze bepaald:
0 – Voor een gemiddeld correcte volgorde van < 90%
5 – Voor een gemiddeld correcte volgorde van 90% - 99%
10 – Voor een gemiddeld correcte volgorde van 100%

5.3.11 HTML/JavaScript

Bij het toetsen aan dit criterium, zal op basis van de specificaties en documentatie van het push-framework gekeken worden of het framework in JavaScript aangesproken kan worden.

De score van een framework op dit criterium wordt op de volgende wijze bepaald:
0 – Voor niet toe te passen binnen HTML/JavaScript
10 – Voor out of the box toe te passen binnen HTML/JavaScript

5.3.12 Kosten/Licenties

Dit zal bekeken aan de hand van de prijsopgaven die de push-frameworks zelf publiceren op hun websites.

De score van een framework op dit criterium wordt op de volgende wijze bepaald:
0 – Voor het framework wat kosten met zich mee brengt bij redelijk gebruik
10 – Voor een framework dat open-source en gratis is

5.3.13 Dataverkeer

Om dit criterium te toetsen zal in verder onderzoek de testopzet in Bijlage C: Runtime testopzet moeten worden uitgebreid.

De score van een framework op dit criterium zou op de volgende wijze kunnen worden bepaald:
0 – Voor het framework dat de meeste dataverkeer verbruikt tijdens de test
1-9 – Relatief aan minst en meest scorende
10 – Voor het framework dat de minste dataverkeer verbruikt tijdens de test

5.4 Rekenvoorbeelden

Het criterium "Is het framework crossplatform in te zetten met Xamarin?":
Wanneer er bijvoorbeeld drie push-frameworks worden getoetst en framework A is out of the box te gebruiken in Xamarin, dan krijgt dit framework de volle 10 punten op dit criteria. Framework B is bijvoorbeeld alleen te gebruiken wanneer er zelf een toevoeging wordt ontwikkeld en krijgt daarom 5 punten. Framework C is absoluut niet te gebruiken in Xamarin en krijgt daarom geen punten.

Het criterium "Is de aflevering van een bericht gegarandeerd?":
Wanneer er bijvoorbeeld drie push-frameworks worden getoetst en framework A scoort 100%, dan krijgt dit framework de volle 10 punten op dit criteria. Framework B scoort 81% en krijgt daarom 5 punten. Framework C scoort 70% van A krijgt daarom 0 punten.

6 Beschikbare push-frameworks

In dit hoofdstuk worden 10 push-frameworks onder de loep genomen. Er wordt gekeken hoe ze werken en er wordt per criterium een score gegeven.

6.1 SignalR (v2.0)

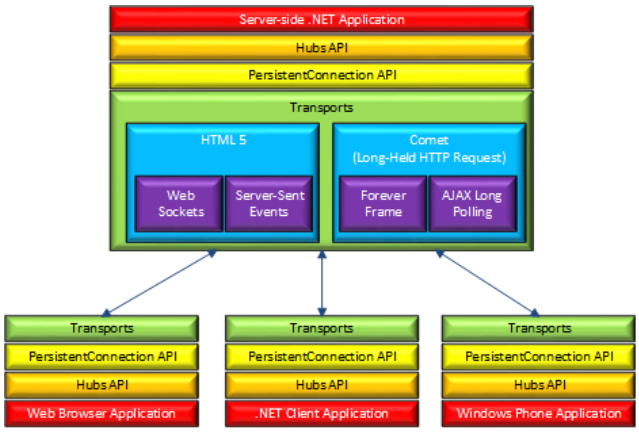
SingalR is een push-framework geschreven in .NET. Het is open source, gratis te gebruiken, maakt gebruik van de HTML5 transportmechanismen Websockets en SSE. Wanneer deze niet beschikbaar zijn wordt er teruggevallen op andere mechanismen waaronder long polling. SignalR zet een eigen persistente verbinding op tussen server en client. Wanneer de client offline is, kan deze dus geen push berichten ontvangen via SignalR.

SignalR kent twee abstractielevels die gebruikt kunnen worden voor de implementatie:

- Met de PersistentConnection API kunnen de endpoints op een low-level niveau worden ingericht.
- Met de Hubs API kan op een high-level manier tussen server en client gecommuniceerd worden, zoals bij remote procedure call (RPC).

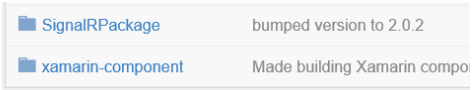
Hoewel dit geen vooraf opgesteld criterium is, is het een extra voordeel dat d.m.v. de Hubs API een client ook calls kan doen richting de backend.

Zaken als "guaranteed delivery" en transaction management worden niet in dit framework gedekt.



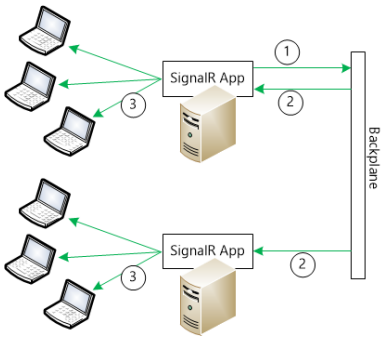
(Microsoft, 2014)

Is het te gebruiken in Xamarin?
Op hun forum laat het Xamarin team weten dat er zeer binnenkort een SignalR component beschikbaar wordt binnen de Xamarin component store. Het bericht is geplaatst in nov 2013. In de open-source GitHub repository is te zien dat er al gewerkt is aan het component.



Ook zonder dit component zou SignalR in Xamarin te gebruiken moeten zijn aangezien het een C# project is waarvan de .dll's in te laden zouden moeten zijn in een Xamarin project. Op verschillende fora waaronder deze is ook te lezen dat mensen SignalR al werkend hebben gekregen in een Xamarin project. Score op dit criterium: 10.

Schaalbaarheid
Wat betreft schaalbaarheid zegt SignalR dat een scale-out mogelijk is wanneer gebruik wordt gemaakt van een backplane. Een backplane zorgt ervoor dat wanneer een applicatie over meerdere servers wordt gehost, een bericht via deze backplane naar alle servers wordt verspreidt. Zo'n backplane kan volgens SignalR gerealiseerd worden met Windows Azure Service Bus, Redis of SQL Server. Het gebruik van een backplane kan de afleversnelheid van een bericht wel vertragen door de verlengde flow (Wasson & Patrick, 2014). Score op dit criterium is 5, omdat de backplane zelf nog moet worden toegevoegd.



Versleuteling
Als een applicatie gevoelige informatie verstuurt en SignalR gebruikt, dient het Secure Socket Layers protocol (SSL) gebruikt te worden. SSL gebruikt encryptie om veilig data uit te wisselen tussen een client en server. (Fitzmacken & Fletcher, 2014). Verder geeft SignalR op deze pagina een uitgebreide uitleg over hoe allerlei security gerelateerde zaken geregeld zouden moeten worden. Score op dit criterium is 0, omdat SSL en de versleuteling zelf nog moet worden geregeld.

Benodigde infrastructuur
Er is geen extra infrastructuur nodig wanneer besloten wordt SignalR in een applicatie te implementeren. Pas wanneer er capaciteitsproblemen worden geconstateerd zal er opgeschaald moeten worden zoals eerder beschreven. SignalR 2.0 werkt alleen op een in .NET framework 4 omgeving. Het zou met tooling als Mono ook mogelijk moeten zijn om te kunnen runnen op een omgeving zonder .NET (Mono, 2014). Score op dit criterium 10, omdat het out of the box aan een eigen backend, draaiend op Windows Server, toe te voegen is.

Communicatie via derden
Bij het gebruik van SignalR wordt er een persistente verbinding opgezet tussen client en de eigen server. Er komt geen derde partij aan te pas in de flow van berichten. Score op dit criterium is 10, omdat er geen derde partij in de communicatielijn voorkomt.

Berichten ontvangen indien app niet actief
Bij SignalR wordt de persistente verbinding door de app gestart open gehouden. Dit houdt ook in dat wanneer de app wordt afgesloten de verbinding wordt verbroken en er geen berichten meer kunnen worden ontvangen. Er zijn oplossingen om een deel van de app op de achtergrond te laten runnen als een soort service, waardoor de verbinding open gehouden zou kunnen worden. Dit zal wel nadelig zijn voor het stroomverbruik. De mogelijkheden hiervoor zijn per platform van het device ook verschillend. Score op dit criterium is 0, omdat dit out of the box niet mogelijk is.

HTML/JavaScript
SignalR is geschikt om te gebruiken in HTML/JavaScript clients. Op deze pagina geven ze uitgebreide uitleg. Score op dit criterium is 10, dit is direct toepasbaar.

Kosten/Licenties
SignalR is een open-source .NET framework (Apache 2.0 License) en is gratis te gebruiken (Microsoft, 2014). Score op dit criterium is 10, omdat het gratis te gebruiken is.

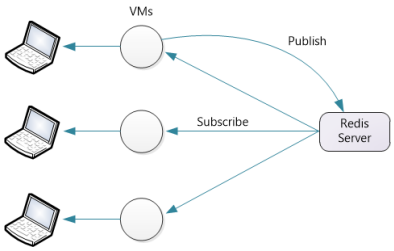
6.2 Socket.IO (v0.9)

Socket.IO is een push-framework voor Node.JS (een JavaScript server taal). Het doel van Socket.IO is om de verschillen tussen de transportmechanismen te doen vervagen. (LearnBoost, 2014)

Het framework zorgt er zelf voor dat het best beschikbare transportmechanisme wordt gebruikt. Ondanks dat dit framework bedoeld is voor een Node.JS server en JavaScript clients, zijn er ook client libraries ontwikkeld door derden voor allerlei andere programmeertalen zoals .NET, Java en Objective-C. (LearnBoost (2), 2014)

Is het te gebruiken in Xamarin?
Voor dit framework kan via NuGet een package genaamd SocketIO4Net.Client gedownload worden. De package is vrij uitgebreid en wordt uitgegeven als "stable release". Wanneer het aan een Xamarin project toegevoegd wordt blijkt dat het niet compatible is met het target framework van het project. Om dit werkend te krijgen moet hier nog iets aan worden gedaan. Score op dit criterium: 5.

Schaalbaarheid
Er is over het schaalbaar inzetten van Socket.IO niet veel beschreven. Behalve een in Portugees geschreven artikel over Socket.IO in combinatie met Redis. Het bestaan van dit artikel suggereert dat net als SignalR, Socket.IO schaalbaar ingezet zou kunnen worden met een backplane op basis van Redis. Details hierover zijn verder niet bekend. Score op dit criterium is 5, omdat het schaalbaar maken zelf nog moet worden geregeld.



Versleuteling
In dit artikel is te lezen hoe een server met Node.JS en Socket.IO opgezet kan worden waarbij het SSL protocol zorgt voor versleutelde communicatie. Verder zijn er op de GitHub pagina van Socket.IO nog artikelen geschreven over hoe andere zaken als autorisatie en sessie afhandeling kan worden opgezet. Score op dit criterium is 0, omdat SSL en versleuteling van berichten zelf moet worden geregeld.

Benodigde infrastructuur
Om Socket.IO te kunnen implementeren, is een server nodig met daarop Node.JS geïnstalleerd. Score op dit criterium 4, omdat Node.JS nog moet worden geïnstalleerd en geconfigureerd op de eigen backend server.

Communicatie via derden
Bij het gebruik van Socket.IO wordt er een persistente verbinding opgezet tussen client en de eigen server. Er komt geen derde partij voor in de flow van berichten. Score op dit criterium 10, omdat er geen derde partij in de communicatielijn voorkomt.

Berichten ontvangen indien app niet actief
Bij Socket.IO wordt de persistente verbinding, die door de app gestart is, open gehouden. Dit houdt ook in dat wanneer de app wordt afgesloten de verbinding wordt verbroken en er niet zomaar berichten meer kunnen worden ontvangen. Score op dit criterium Score op dit criterium 0, omdat dit out of the box niet mogelijk is.

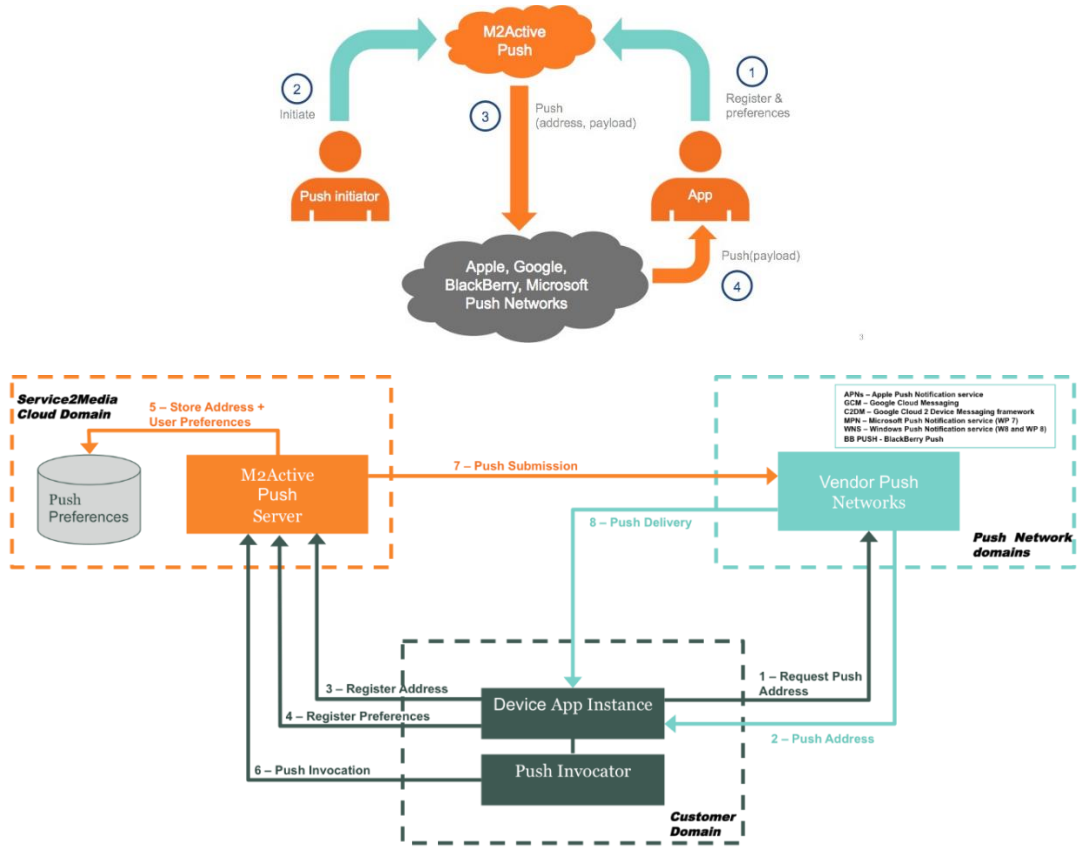
HTML/JavaScript
Het framework is expliciet bedoeld voor clients geschreven in HTML/JavaScript. Score op dit criterium Score op dit criterium 10, dit is out of the box toe te passen.

Kosten/Licenties
Socket.IO is een open-source JavaScript framework (MIT license) en is gratis te gebruiken. Score op dit criterium Score op dit criterium is 10, omdat het gratis te gebruiken is.

6.3 *Service 2 Media*

Service 2 Media maakt het implementeren van native push communicatie, via het PNS (Platform Notification System) van het device, makkelijker. De infrastructuur van Service 2 Media zit tussen je eigen server en PNS van het client device in zoals in de afbeeldingen te zien is. Het ontvangen van berichten gebeurt nog steeds op de native manier en de app hoeft niet te runnen om het bericht te kunnen ontvangen.

Voor er berichten gepusht kunnen worden dient de normale registratie stap van het PNS van het device te worden doorlopen. Daarna volgt een afrondende registratiestap bij Service 2 Media. Hierna kan er d.m.v. een enkele call naar de Service 2 Media server een bericht worden gepusht naar de clients, ongeacht welk OS deze draaien.



(Service 2 Media, 2014)

Opvallend is dat ze de dienst in 2014 onder een andere naam CERTA Push hebben gelanceerd. De informatie over dit push-framework is van beide sites gehaald.

Is het te gebruiken in Xamarin?
Dit framework verstuurt berichten via de PNS van het client device. De client moet de benodigde registratiestap zelf bij het PNS uit te voeren. Er is hiervoor geen SDK

beschikbaar en dit moet dus allemaal zelf nog gedaan worden binnen het Xamarin project. Score op dit criterium is 5, omdat er geen support voor Xamarin is maar het wel met een wrapper kan worden toegevoegd aan het project.

Schaalbaarheid
De bottleneck wat betreft schaalbaarheid ligt bij dit framework aan de capaciteit van de infrastructuur die Service 2 Media zelf inzet. Verwacht mag worden van een PNS van bijv. Google en Apple dat deze altijd genoeg capaciteit hebben. Score op dit criterium is 10, omdat er geen extra werk nodig is om het schaalbaar in te zetten.

Versleuteling
Voor de versleuteling van berichten communiceert Service2Media alleen via het SSL protocol. Om ervoor te zorgen dat Service2Media alleen berichten doorstuurt die "echt" zijn, moet vooraf op de portal worden ingesteld van welke IP adressen berichten mogen worden gestuurd. Om berichten ook onleesbaar te maken voor het PNS van het device OS bieden ze een extra versleutelingsstap Secure Push aan. Score op dit criterium is 10, omdat zowel de verbinding als de berichten zijn versleuteld.

Benodigde infrastructuur
Er is behalve de infrastructuur waar de eigen backend op draait geen extra infrastructuur nodig. Score op dit criterium is echter 9, omdat er een externe dienst wordt afgenomen.

Communicatie via derden
Tussen de communicatie van client en server zijn nog twee partijen betrokken. Namelijk de infrastructuur van Service 2 Media en die van het PNS van het client device OS. Sinds kort is het ook mogelijk om de service op eigen infrastructuur te hosten (kosten €60.000), in dit geval zou alleen het PNS er nog tussen zitten. Score op dit criterium is (indien niet zelf hosten) 0, omdat de communicatie via meer dan 1 partij loopt.

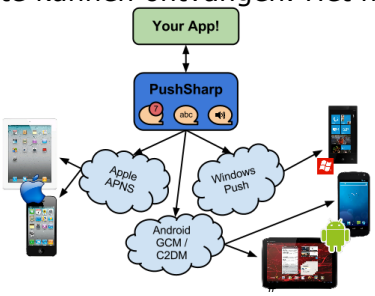
Berichten ontvangen indien app niet actief
Omdat het uiteindelijk het PNS is die het bericht aflevert bij het client device, is het mogelijk om berichten te ontvangen wanneer de app niet actief is. Score op dit criterium: 10.

HTML/JavaScript
Dit framework is niet geschikt gemaakt om te gebruiken in een client die geschreven is in HTML/JavaScript (Service 2 Media, 2014). Score op dit criterium is 0, omdat het niet toepasbaar is in HTML/JavaScript.

Kosten/Licenties
Service 2 Media biedt push messaging op twee manieren aan. Als service, kosten vanaf €0,0001 per bericht. En als "host it yourself" oplossing, sinds kort is het ook mogelijk om de service op eigen infrastructuur te hosten, een licentie hiervoor kost €60.000,- per server. Score op dit criterium is 0, omdat er kosten gemaakt zullen worden bij redelijk gebruik.

6.4 PushSharp (v2.1.2)

PushSharp is een framework geschreven in .NET. Het is een server library voor het verzenden van push berichten naar een client device, via het PNS van het device. Door deze vorm hoeft een client app op een device niet te runnen om een push bericht te kunnen ontvangen. Het framework is open source en wordt gehost op GitHub.



(Dick, 2014)

Is het te gebruiken in Xamarin?
Dit framework verstuurt berichten via de PNS van het client device. De client moet de benodigde registratiestap zelf bij het PNS uitvoeren. Er is hiervoor een SDK beschikbaar welke toegevoegd kan worden aan een Xamarin project. Score op dit criterium: 10.

Schaalbaarheid
Over schaalbaarheid hoeft bij dit framework geen zorgen te worden gemaakt omdat de PNS van het device de berichten aflevert. Het enige waarvoor gezorgd hoeft te worden, is dat de eigen backend server genoeg capaciteit heeft om de berichten te versturen naar de verschillende PNS. Score op dit criterium is 5, omdat de eigen server nog schaalbaar opgezet dient te worden.

Versleuteling
Over versleuteling van berichten is niets te vinden in het framework. Mogelijk moet er zelf dus nog een versleutelingsstap worden toegevoegd. Score op dit criterium is 0, omdat de beveiligde verbinding en versleuteling van berichten zelf geregeld dient te worden.

Benodigde infrastructuur
Om PushSharp te kunnen gebruiken dient op de eigen backend server het .NET framework geïnstalleerd te zijn. Indien dit niet mogelijk is kan het framework ook aangesproken worden op een Mono omgeving, het framework is op deze mogelijkheid gebouwd (Dick, 2014). Score op dit criterium is 10, omdat het out of the box toe te passen is in een Windows Server omgeving.

Communicatie via derden
De flow van berichten loopt via het PNS van het device OS. Score op dit criterium is 5, omdat het berichtenverkeer via 1 derde partij loopt.

Berichten ontvangen indien app niet actief
Omdat berichten via het PNS van het device OS worden afgeleverd, kunnen berichten ontvangen worden wanneer de app niet actief is. Score op dit criterium: 10.

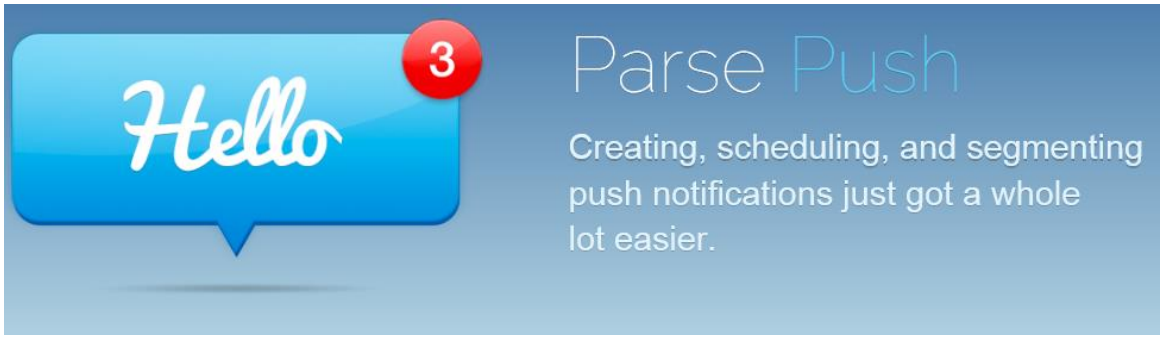
HTML/JavaScript
Het framework is niet gebouwd om berichten te ontvangen in een browser. Een uitzondering hierop is de Chrome browser van Google, deze kan berichten ontvangen via het PNS van Google. Score op dit criterium is 0, omdat dit niet in alle moderne browsers gebruikt kan worden.

Kosten/Licenties
Het framework is open-source (Apache 2.0 license) en is dan ook gratis te gebruiken. Score op dit criterium is 10, omdat het kosteloos is.

6.5 *Parse*

Parse is een framework waar de volledige backend van een app kan worden gehost. Binnen deze oplossing bieden zij ook het versturen van push berichten aan op basis van een channel model (publish/subscribe). De push berichten worden via het PNS van het device bezorgd. Ook hier geldt dat door deze vorm, de app op het device niet hoeft te runnen om het bericht te ontvangen.

Om Parse te kunnen implementeren in een client app zijn er libraries beschikbaar in de meest gebruikte talen. Naast een gratis instap, zijn er kosten verbonden aan het gebruik van dit framework. De kosten lopen op afhankelijk van de benodigde capaciteit.



(Parse, 2014)

Is het te gebruiken in Xamarin?
Er kan via NuGet een package van Parse gedownload worden. Wanneer het aan een Xamarin project toegevoegd wordt blijkt dat het niet compatible is met het target framework van het project. Om dit werkend te krijgen moet hier nog iets aan worden gedaan. Score op dit criterium: 5.

Schaalbaarheid
De bottleneck wat betreft schaalbaarheid ligt bij dit framework aan de capaciteit van de infrastructuur die Parse zelf inzet. Ze zeggen zelf dat hun dienst voor massale schaal is opgezet. Verwacht mag worden van een PNS van bijv. Google en Apple dat deze altijd genoeg capaciteit hebben. Score op dit criterium is 10, omdat er geen stappen nodig zijn om het schaalbaar in te zetten.

Versleuteling
Over versleuteling van push berichten is niets te vinden op de website van Parse. Wel zeggen ze dat de verbinding tussen device en PNS standaard SSL gebruikt, maar dit zegt verder niks over de versleuteling van het bericht tussen de eigen backend en de server van Parse. Mogelijk moet er zelf dus nog een versleutelingsstap worden toegevoegd. Score op dit criterium is 5, omdat alleen de verbinding beveiligd is.

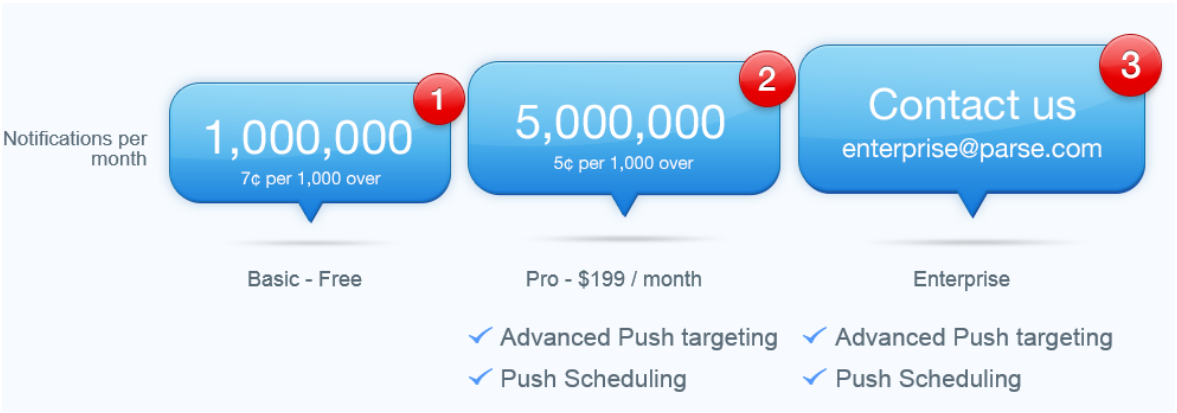
Benodigde infrastructuur
Score op dit criterium is echter 9, omdat er een externe service wordt afgenomen.

Communicatie via derden
De infrastructuur van Parse en die van het PNS van het device OS zit tussen de eigen server en de client(s). Score op dit criterium is 0, omdat het berichtenverkeer via meer dan één derde partij loopt.

Berichten ontvangen indien app niet actief
Omdat berichten via het PNS van het device OS worden afgeleverd, kunnen berichten ontvangen worden wanneer de app niet actief is. Score op dit criterium: 10.

HTML/JavaScript
Het push gedeelte van het framework is niet geschikt om te gebruiken in een HTML/JavaScript client. Het is gericht op de mobiele platformen waarvoor native SDK's beschikbaar zijn. Score op dit criterium: 0.

Kosten/Licenties
Parse push kan voor 1 miljoen berichten per maand gratis worden uitgeprobeerd. Wanneer dit wordt overschreden kost dit $0,07 per 1.000 berichten. Het Pro abonnement kost $199 per maand. Hiermee kunnen 5 miljoen berichten per maand worden verstuurd, elke 1.000 extra berichten kost $0,05. Voor nog hogere capaciteit kan in overleg een Enterprise abonnement worden samengesteld. Score op dit criterium is 0, omdat er kosten mee worden gemaakt wanneer er redelijke capaciteit wordt gevraagd.



6.6 Hydna

Hydna is een framework dat push communicatie mogelijk maakt d.m.v. een persistente connectie tussen de client en de server van Hydna. Dit houdt in dat de app moet runnen om push berichten te kunnen ontvangen. Hydna gebruikt het best beschikbare transportmechanisme dat er beschikbaar is tussen server en client. Is communiceren via Websocket bijvoorbeeld niet mogelijk, dan wordt er teruggevallen op andere technieken over HTTP en TCP.

Er zijn veel server en client libraries beschikbaar om het framework te implementeren. Berichten worden verzonden en ontvangen op basis van het channels en events model (publish/subscribe).

De infrastructuur van Hydna zit tussen de eigen server en de client app(s). Hydna kondigt op hun website aan dat het framework binnenkort open source zal worden, waarna het dus op eigen infrastructuur te hosten zal zijn. Ook zeggen ze dat het schaalbaar is.

Naast een gratis instap, zijn er aan het gebruik van het framework (als gehoste dienst) kosten verbonden. De kosten lopen op naarmate er meer capaciteit wordt afgenomen.

(Hydna, 2014)

Is het te gebruiken in Xamarin?
Voor dit framework kan via NuGet een package gedownload worden. Op deze manier zou het ook gebruikt kunnen worden in een Xamarin project. Er is een stable release te downloaden of er kan een pre-release versie gebruikt worden, wat aangeeft dat de client library voor .NET nog in ontwikkeling is. Wanneer het aan een Xamarin project wordt toegevoegd blijkt dat het niet compatible is met het target framework van het project. Een andere optie zou kunnen zijn om de native libraries die er beschikbaar zijn te gebruiken en hier een wrapper voor te schrijven. Score op dit criterium is 5, omdat hier nog iets aan gedaan moet worden om het werkend te krijgen.

Schaalbaarheid
De bottleneck wat betreft schaalbaarheid ligt bij dit framework aan de capaciteit van de infrastructuur die Hyda zelf inzet. Ze zeggen zelf dat hun platform schaalbaar en gedistribueerd is opgezet. Score op dit criterium is 10, omdat er zelf niet aan schaalbaarheid gewerkt hoeft te worden.

Versleuteling
Alle betaalde abonnementen van Hydna maken gebruik van Transport Layer Security (TLS), de opvolger van SSL. Hiermee wordt gezorgd dat de verbinding is versleuteld. Score op dit criterium is 5, omdat alleen de verbinding beveiligd is en niet de berichten.

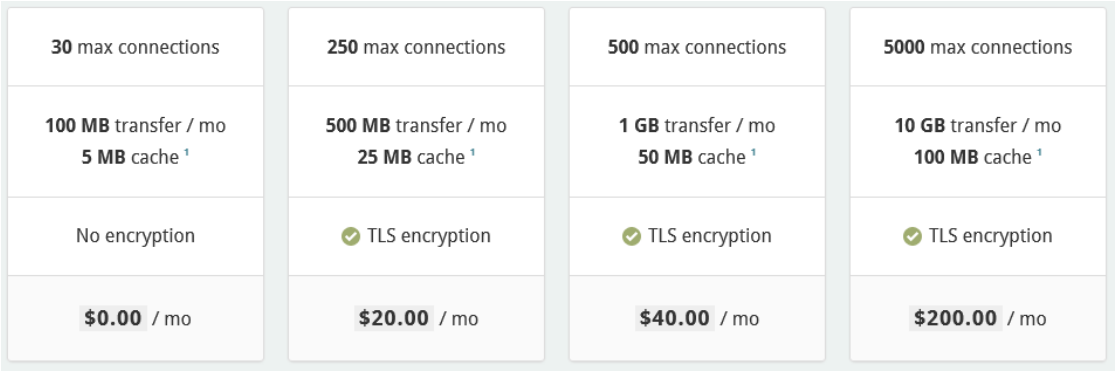
Benodigde infrastructuur
Score op dit criterium is echter 9, omdat er een externe service wordt afgenomen.

Communicatie via derden
Bij het gebruik van Hydna wordt er een persistente verbinding opgezet tussen client en de server van Hydna. Deze partij zit dus in de flow van berichten. Score op dit criterium 5, omdat het berichtenverkeer via 1 derde partij loopt.

Berichten ontvangen indien app niet actief
Bij Hydna wordt de persistente verbinding door de app gestart open gehouden. Dit houdt ook in dat wanneer de app wordt afgesloten de verbinding wordt verbroken en er niet zomaar berichten meer kunnen worden ontvangen. Score op dit criterium is 0, omdat dit zomaar niet mogelijk is.

HTML/JavaScript
Er is een JavaScript client SDK beschikbaar om een HTML/JavaScript client te kunnen implementeren. Score op dit criterium: 10.

Kosten/Licenties
Naast de betaalde abonnementen is er een gratis variant om de dienst te kunnen testen. Bij de betaalde abonnementen loopt het maandbedrag op naarmate er meer verbindingen, dataoverdracht en beveiliging nodig is. Naast deze abonnementen kan er ook in overleg een afname op maat worden bepaald.



De kosten bij gebruik van dit framework kunnen aardig oplopen wanneer er veel gebruikers van de applicatie zijn. Score op dit criterium is 0, omdat er kosten worden gemaakt bij redelijk gebruik.

6.7 PubNub

PubNub is een framework dat push communicatie mogelijk maakt d.m.v. een persistente connectie tussen de client en de server van PubNub. Dit houdt in dat de app moet runnen (of op de achtergrond moet draaien) om push berichten te kunnen ontvangen.

Er zijn meer dan 50 server en client libraries beschikbaar om het framework te implementeren. Berichten worden verzonden en ontvangen op basis van het channels model (publish/subscribe).

De infrastructuur van PubNub zit tussen de eigen server en de client app(s). Ook zegt het schaalbaar te zijn.

Naast een gratis instap, zijn er aan het gebruik van het framework kosten verbonden. De kosten lopen op naarmate er meer capaciteit wordt afgenomen.

(PubNub, 2014)

Is het te gebruiken in Xamarin?
Voor dit framework kan via NuGet of GitHub een .NET package gedownload worden. Ook zijn er op GitHub de packages beschikbaar die direct in een Xamarin.iOS en Xamarin.Android project kunnen worden gebruikt, dit is echter wel voor een verouderde versie van Mono. Score op dit criterium is 5, omdat er nog gesleuteld moet worden om het te kunnen builden.

Schaalbaarheid
De bottleneck wat betreft schaalbaarheid ligt bij dit framework aan de capaciteit van de infrastructuur die PubNub zelf inzet. Ze zeggen zelf dat hun platform schaalbaar en gedistribueerd is over 14 datacenters. Score op dit criterium is 10, omdat er zelf niets aan schaalbaarheid gedaan hoeft te worden.

Versleuteling
Voor de versleuteling kan er gebruik gemaakt worden van het ingebouwde AES algoritme. Daarnaast is het optioneel om het SSL protocol te gebruiken, deze optie maakt het versturen van een bericht wel 2x zo duur. Score op dit criterium is 10, omdat zowel de verbinding als de berichten versleuteld kunnen worden zonder dat hier iets voor hoeft te worden gedaan.

Benodigde infrastructuur
Score op dit criterium is 9, omdat er een externe service wordt afgenomen.

Communicatie via derden
De servers van PubNub zitten tussen de flow van de berichten. Score op dit criterium is 5, omdat het berichtenverkeer via een derde partij verloopt.

Berichten ontvangen indien app niet actief
Bij PubNub wordt de persistente verbinding door de app gestart open gehouden. Dit houdt ook in dat wanneer de app wordt afgesloten de verbinding wordt verbroken en er niet zomaar berichten meer kunnen worden ontvangen. Score op dit criterium is 0, omdat dit niet mogelijk is.

HTML/JavaScript
Met de 13 JavaScript SDK's die beschikbaar zijn voor PubNub, is het bouwen van een client op basis van HTML/JavaScript goed mogelijk. Score op dit criterium: 10.

Kosten/Licenties
Naast het gratis testabonnement zijn er vijf betaalde abonnementen. De betaalde abonnementen hebben standaard 1 miljoen berichten per maand. Elk miljoen berichten extra kost $1,- en indien SSL gebruikt wordt $2,-. Verder lopen de vaste maandelijkse bedragen op naarmate er meer devices dagelijks connected zijn. Deze "daily active devices" houdt in het aantal unieke devices date er in 24 uur connected is geweest. Score op dit criterium is 0, omdat er bij redelijk gebruik kosten zullen worden gemaakt.

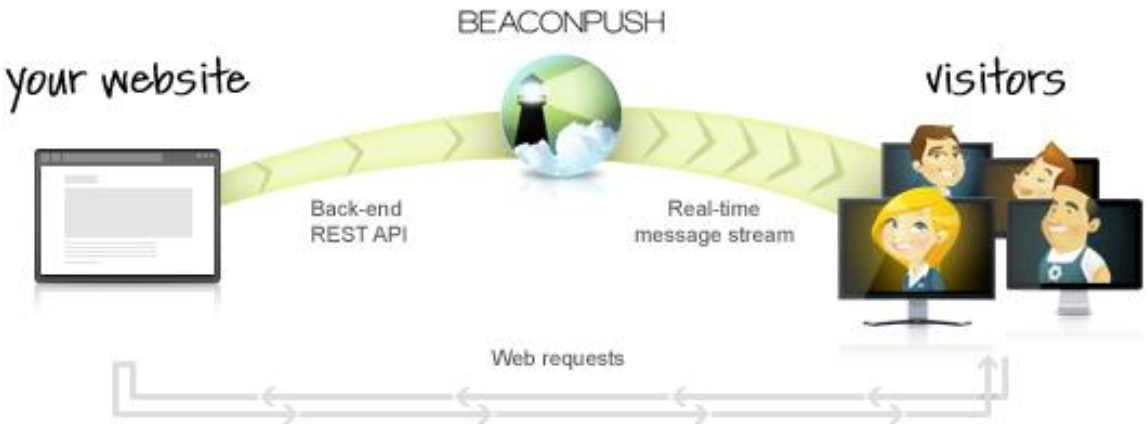
Sandbox	Go Cloud				Global Cloud
	Tier 1	Tier 2	Tier 3	Tier 4	starts at
Free	$15/mo	$49/mo	$125/mo	$399/mo	$1299/mo
20 Daily Active Devices	100 Daily Active Devices	3,000 Daily Active Devices	10,000 Daily Active Devices	40,000 Daily Active Devices	Millions of Devices Learn More
1 million messages included + $1 / million ($2 / million SSL)					GET A QUOTE

6.8 BeaconPush

BeaconPush is een JavaScript framework dat zich richt op push communicatie tussen server en browsers. Het werkt op basis van het channels model (publish/subscribe). Om een push bericht te verzenden kan vanuit de eigen server de REST API van BeaconPush worden aangesproken.

Het brengt een persistente connectie tot stand tussen de client en de BeaconPush server d.m.v. Websocket technologie. De infrastructuur van BeaconPush zit dus tussen de eigen server en de clients. Wanneer communicatie via Websockets niet mogelijk is wordt er teruggevallen op andere transportmechanismen. De benodigde persistente verbinding houdt ook in dat een client browser online moet zijn om berichten te kunnen ontvangen. BeaconPush garandeert dat berichten die niet bezorgd kunnen worden door afwezigheid van de client, later alsnog afgeleverd worden.

BeaconPush vermeld expliciet dat zij hun servercapaciteit inkopen bij Amazon, waardoor ze de dienst schaalbaar kunnen leveren.



(BeaconPush, 2014)

Is het te gebruiken in Xamarin?
Er is wel een .NET package te downloaden via NuGet, maar deze is alleen voor de server side implementatie. Er is van BeaconPush alleen een JavaScript client SDK beschikbaar. Om dit in Xamarin te kunnen gebruiken zou er een eerst wrapper moeten worden geschreven zodat de SDK in C# aan te spreken is. Score op dit criterium: 5.

Schaalbaarheid
De bottleneck wat betreft schaalbaarheid ligt bij dit framework aan de capaciteit van de infrastructuur die BeaconPush zelf inzet. Ze zeggen zelf dat ze hun dienst hosten bij Amazon AWS en dat deze hierdoor betrouwbaar en snel op te schalen is. Score op dit criterium is 10, omdat er zelf niets gedaan hoeft te worden om het schaalbaar in te zetten.

Versleuteling
Versleuteling van berichten d.m.v. SSL is alleen mogelijk wanneer het "On-site" abonnement wordt afgenomen. Score op dit criterium is 5, omdat alleen de verbinding is beveiligd en versleuteling van de berichten zelf gedaan dient te worden.

Benodigde infrastructuur
Er is geen bijzondere infrastructuur nodig om BeaconPush te kunnen gebruiken om een bericht te versturen, het benaderen van deze REST API kan vanuit elke omgeving. Wanneer het "On-site" abonnement wordt afgenomen is een server met minimaal 1CPU, 1GB RAM en een JVM (Java Virtual Machine) geïnstalleerd benodigd. Score op dit criterium is 9, omdat er een externe service wordt afgenomen.

Communicatie via derden
Bij het gebruik van BeaconPush wordt er een persistente verbinding opgezet tussen client en de server van BeaconPush. Deze partij zit (afhankelijk van het abonnement, zie kosten/licenties) wel of niet in de flow van berichten. Score op dit criterium (indien niet "On-site" abonnement) is 5, omdat het berichten verkeer via een derde partij verloopt.

Berichten ontvangen indien app niet actief
Het is bij een browserapplicatie niet mogelijk om nog berichten te ontvangen wanneer deze niet geopend is. Score op dit criterium: 0.

HTML/JavaScript
Dit framework richt zich exclusief op HTML/JavaScript clients. Score op dit criterium: 10.

Kosten/Licenties
Sinds 1 december is dit framework gestopt met het aanbieden van het framework als een service waarbij voor het aantal verzonden berichten werd betaald. Er kan nu alleen nog een gratis developer abonnement en een "On-site" abonnement worden afgenomen. Bij het On-site abonnement wordt BeaconPush op de eigen infrastructuur gehost en loopt het maandelijkse bedrag op naarmate het aantal gelijktijdige app gebruikers stijgt. Score op dit criterium is 0, omdat er kosten gemaakt zullen worden.

	Flotilla	Fleet	Armada	Custom
Concurrent users	5 000	10 000	20 000	Millions
Messages included	Unlimited	Unlimited	Unlimited	Unlimited
Everything included from Cloud Edition	✓	✓	✓	✓
Monthly price per server	$199	$299	$799	Contact us

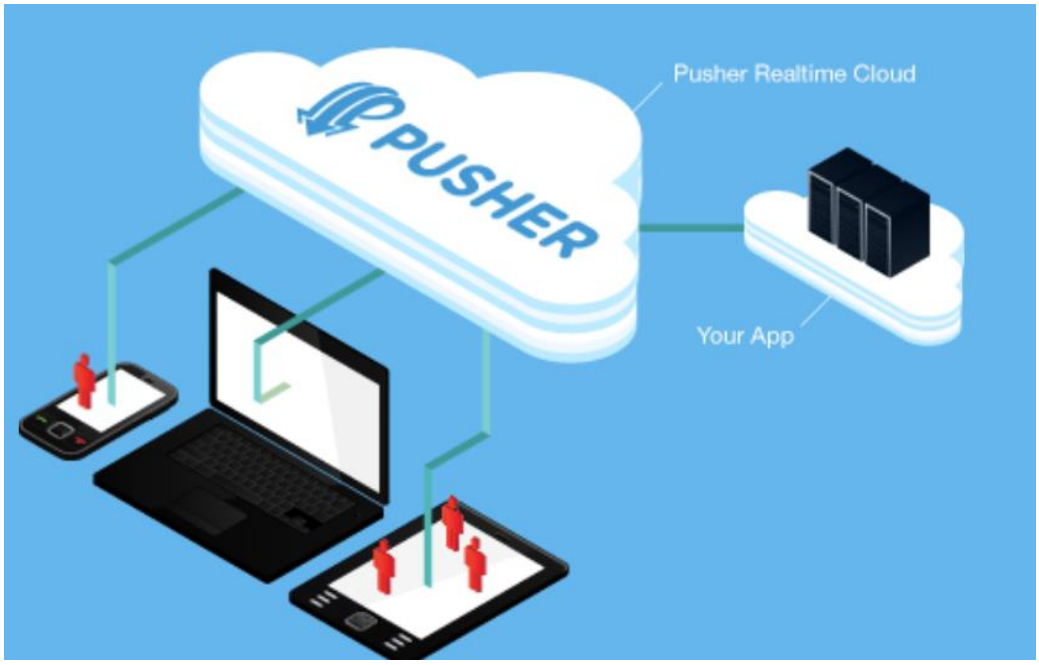
6.9 Pusher

Pusher is een framework dat push communicatie mogelijk maakt d.m.v. een persistente connectie tussen de client en de server van Pusher. Deze persistente connectie wordt opgezet d.m.v. Websocket technologie. Wanneer communicatie via Websockets niet mogelijk is wordt er teruggevallen op andere transportmechanismen. De wijze van communicatie houdt wel in dat de app moet runnen om push berichten te kunnen ontvangen.

Er zijn veel server en client libraries beschikbaar (sommigen ook ontwikkeld door derden) om het framework te implementeren. Berichten worden verzonden en ontvangen op basis van het channels en events model (publish/subscribe).

De infrastructuur van Pusher zit tussen de eigen server en de client app(s). Ook zegt het schaalbaar te zijn omdat ze hun servercapaciteit inkopen bij Amazon.

Naast een gratis instap, zijn er aan het gebruik van het framework kosten verbonden. De kosten lopen op naarmate er meer capaciteit wordt afgenomen.



(Pusher, 2014)

Is het te gebruiken in Xamarin?
Voor dit framework zijn er via NuGet twee verschillende client packages te downloaden worden. Deze blijken echter niet zomaar compitabel met Xamarin. Er zal een build gerealiseerd moeten worden die wel compitabel is voor het gebruikt kan worden. Score op dit criterium is 5, omdat het niet out of the box te gebruiken is met Xamarin.

Schaalbaarheid
De bottleneck wat betreft schaalbaarheid ligt bij dit framework aan de capaciteit van de infrastructuur die Pusher zelf inzet. Ze zeggen zelf dat ze hun dienst hosten bij Amazon EC2 en dat deze hierdoor schaalbaar kan worden aangeboden. Score op dit criterium is 10, omdat er niets gedaan hoeft te worden om het schaalbaar in te zetten.

Versleuteling
Voor de versleuteling van berichten is bij Pusher SSL beschikbaar vanaf het "Startup" abonnement ($49 per maand). Verder hebben ze in hun framework op het gebied van security, authenticatie van gebruikers en channels ingebouwd. Score op dit criterium is 5, omdat de verbinding wel beveiligd is maar de versleuteling van berichten zelf dient te worden geregeld.

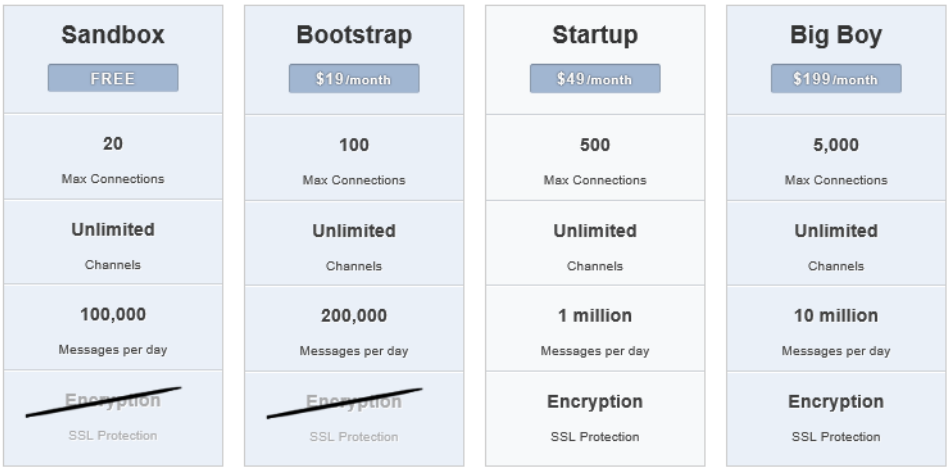
Benodigde infrastructuur
Score op dit criterium is 9, omdat er een externe service wordt afgenomen.

Communicatie via derden
Bij het gebruik van Pusher wordt er een persistente verbinding opgezet tussen client en de server van Pusher. Deze partij zit dus in de flow van berichten. Score op dit criterium is 5, omdat het berichten verkeer via een derde partij verloopt.

Berichten ontvangen indien app niet actief
Bij Pusher wordt de persistente verbinding door de app gestart open gehouden. Dit houdt ook in dat wanneer de app wordt afgesloten de verbinding wordt verbroken en er niet zomaar berichten meer kunnen worden ontvangen. Score op dit criterium: 0.

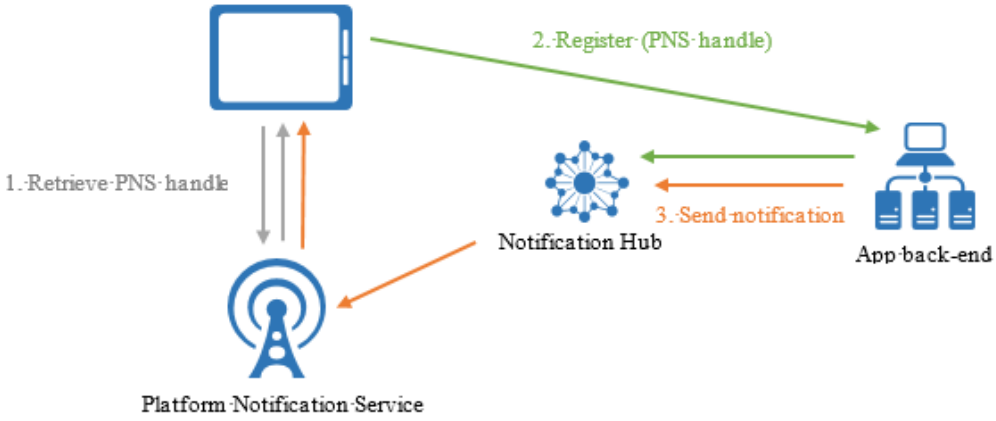
HTML/JavaScript
Er is een JavaScript client SDK beschikbaar om een HTML/JavaScript client te kunnen implementeren. Score op dit criterium 10.

Kosten/Licenties
Naast de drie betaalde abonnementen is er een gratis variant om de dienst te kunnen testen. Bij de betaalde abonnementen loopt het maandbedrag op naarmate er meer verbindingen en een groter aantal berichten per dag nodig is. Vanaf het Startup abonnement is SSL beschikbaar.
Naast deze abonnementen kan er ook in overleg een Enterprise abonnement op maat worden opgesteld. Score op dit criterium is 0, omdat er bij redelijk gebruik kosten gemaakt zullen worden.



6.10 Azure Notification Hub

Azure Notification Hub is een service van Microsoft binnen het Azure platform. Het is een service voor het verzenden van push berichten naar een client device, via het PNS van het device. De infrastructuur van Microsoft zit tussen je eigen server en PNS van het client device zoals op de afbeelding te zien is. Na een gratis instap zijn er verschillende abonnementen af te nemen. Omdat Azure Notification Hub een push bericht uiteindelijk via het PNS van het device verstuurt hoeft een client app op een device niet te runnen om een push bericht te kunnen ontvangen.



Is het te gebruiken in Xamarin?
Er is een Xamarin component beschikbaar genaamd "Azure Mobile Service" waarmee de client kant van deze service in de app kan worden geïmplementeerd. Op dit moment ondersteund dit component alleen iOS en Android, maar Microsoft kondigt aan dat Windows Phone 8 ondersteuning niet lang op zich laat wachten. (Microsoft, 2014). Score op dit criterium: 10.

Schaalbaarheid
De dienst is volledig schaalbaar af te nemen bij het Azure platform. Op de site zegt Microsoft: "*Meldingshubs zijn geoptimaliseerd voor zeer grootschalig werken. Als u meldingshubs gebruikt, kunt u razendsnel schalen naar miljoenen apparaten en miljarden pushmeldingen zonder dat u de toepassing opnieuw hoeft te ontwerpen of 'sharden' (horizontaal partitioneren)".*
Verwacht mag worden van een PNS, dat uiteindelijk het bericht zal afleveren bij het device, zoals bijv. Google en Apple dat deze altijd genoeg capaciteit hebben. Score op dit criterium 10, omdat er niets toegevoegd hoeft te worden om het schaalbaar in te kunnen zetten.

Versleuteling
Microsoft is eind 2013 een samenwerking aangegaan met security bedrijf Thales. Deze samenwerking heeft geleid tot een nieuwe vorm van versleuteling binnen het Windows Azure platform, namelijk het "bring your own key" (BYOK) principe. Hierbij heeft de klant de mogelijkheid om de data die in de Cloud rond gaat te versleutelen met een eigen sleutel (Thales e-Security, 2013). Alleen de partij met de sleutel kan de data nog ontcijferen. Score op dit criterium is 10, omdat zowel de verbinding als de berichten versleuteld kunnen worden zonder dat dit zelf geregeld moet worden.

Benodigde infrastructuur
Er is geen specifieke infrastructuur nodig. Het verzenden van push berichten kan vanuit elk soort backend server via de Notification Hub REST API. Deze kan d.m.v. een

HTTP POST call worden gebruikt. Score op dit criterium is echter 9, omdat er een externe service wordt afgenomen.

Communicatie via derden
Tussen de communicatie van client en server zijn nog twee partijen betrokken. Namelijk de infrastructuur van Microsoft Azure en die van het PNS van het client device OS. Score op dit criterium is 0, omdat de communicatie via meer dan één partij loopt.

Berichten ontvangen indien app niet actief
Omdat het uiteindelijk het PNS is die het bericht aflevert bij het client device, is het mogelijk om berichten te ontvangen wanneer de app niet actief is. Score op dit criterium: 10.

HTML/JavaScript
Al hoewel het framework hier niet primair voor bedoeld is, is in dit artikel te lezen dat er mogelijkheden zijn om een HTML/JavaScript client te bouwen, die push berichten kan ontvangen via dit framework. Score op dit criterium is 0, omdat dit niet out of the box is toe te passen in HTML/JavaScript.

Kosten/Licenties
Azure Notification Hub wordt aangeboden in drie verschillende abonnementen. Met het gratis abonnement kan de functionaliteit eerst uitgeprobeerd worden. De twee betaalde abonnementen lopen op in prijs naarmate er meer berichten verstuurd moeten kunnen worden en naarmate er meer geschaald moet kunnen worden. Score op dit criterium is 0, omdat er bij redelijk gebruik kosten gemaakt zullen worden.

	GRATIS	BASIS	STANDAARD
Prijs per maand	Gratis	€15 per eenheid (dagelijks prorata)	€149 per eenheid (dagelijks prorata)
Actieve apparaten [1]	500 actieve apparaten per naamruimte	Onbeperkt	Onbeperkt
Pushes [1]	100.000 per maand (3.333 per dag)	500.000 per eenheid per maand (16.667 per eenheid per dag)	5 miljoen per eenheid per maand (166.667 per eenheid per dag)
Schalen	N/A	Tot 9 eenheden	Onbeperkt

7 De push-frameworks vergelijken

7.1 Resultaten na toetsing d.m.v. literatuur

Als de eerder d.m.v. literatuur getoetste criteria in een tabel zijn verwerkt, ontstaat de volgende uitslag:

	SignalR	Socket.IO	Service 2 Media	PushSharp	Parse	Hydna	PubNub	BeaconPush	Pusher	Azure Notification Hub
Xamarin (eis)	10	5	5	10	5	5	5	5	5	10
Schaalbaarheid (20%)	5	5	10	5	10	10	10	10	10	10
Versleuteling (15%)	0	0	10	0	5	5	10	5	5	10
Benodigde infrastructuur (5%)	10	4	9	10	9	9	9	9	9	9
Communicatie via derden (10%)	10	10	0	5	0	5	5	5	5	0
Berichten ontvangen indien app niet actief (0%)	0	0	10	10	10	0	0	0	0	10
HTML/JavaScript (5%)	10	10	0	0	0	10	10	10	10	0
Kosten/Licenties (5%)	10	10	0	10	0	0	0	0	0	0
Totaal	55	44	44	50	39	44	49	44	44	49
IS totaal (zwaarte zonder Xamarin)	**3,5**	**3,2**	**4,0**	**2,5**	**3,2**	**4,2**	**5,0**	**4,2**	**4,2**	**4,0**

De onderste rij (IS totaal) is het totaal inclusief de voor Info Support gewogen score. Hierbij is Xamarin buiten de berekening gelaten en de zwaarte per criterium gebruikt om tot een totaal te komen.

De formule die is gebruikt om dit te berekenen ziet er als volgt uit:

InfoSupportScore = *(Schaalbaarheid * 20%) +*
*(Versleuteling * 15%) +*
*(Benodigde infra. * 5%) +*
*(Comm. via derden * 10%) +*
*(Berichten ontv. App niet actief * 0%) +*
*(HTMLJavaScript * 5%) +*
*(KostenLicenties * 5%)*

Het totaal van SignalR is het hoogste, 55. Na het meenemen van de zwaarte in de berekening scoort PubNub het hoogste, 5,0.

Er zijn drie push-frameworks out of the box te implementeren met Xamarin, dit zijn: SignalR, PushSharp en Azure Notification Hub.

PushSharp en Azure Notification Hub zijn vergelijkbaar in werking. Van deze twee scoort Azure Notification Hub flink hoger. SignalR en Azure Notification Hub zullen daarom in een runtime test verder worden onderzocht.

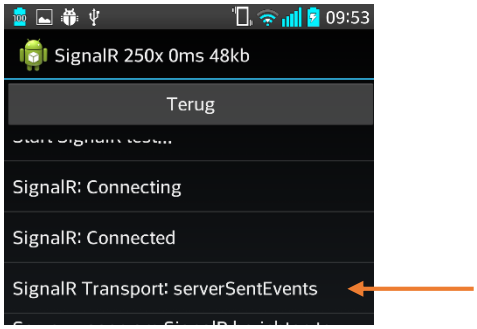
7.2 Resultaten runtime test

In Bijlage C: Runtime testopzet is de volledige testopstelling beschreven. In Bijlage D: Runtime testresultaten zijn alle uitslagen van de test opgenomen.

De onder deze testomstandigheden, opvallende uitkomsten zijn:

- Beiden frameworks hebben altijd een afleverzekerheid van 100%. In verder onderzoek zou dit met gesimuleerde slechte verbinding verder kunnen worden getest.
- SignalR is veruit het snelst (10x zo snel) in het afleveren van een pushbericht.
- Azure Notification Hub kan een extra payload tot 2Kb aan (bij een pushbericht naar Android). Bij een payload van 3kb wordt het bericht niet meer verstuurd.
- SignalR kan een extra payload tot 47Kb aan. Bij een payload van 48Kb wordt het bericht niet meer verstuurd.
- Bij beide frameworks heeft een extra payload nauwelijks invloed op de afleversnelheid.
- Bij SignalR is de volgorde van aankomst zekerder dan bij Azure Notification Hub.

SignalR laat in de testapp zien dat het transportmechanisme Server Sent Events wordt gebruikt voor push communicatie.



Om punten te kunnen geven op de geteste criteria worden de uitslagen opgeteld en gedeeld door het aantal tests. Test #5 t/m #10 konden niet uitgevoerd worden met Azure Notification hub en zijn daarom buiten de berekening gelaten.

SignalR – Gemiddelde score na test #1 tm #4

Afleverzekerheid (Aankomstpercentage):	**100%**	**10 punten**
Gemiddelde afleversnelheid (incl. webservice call):	**59ms**	**10 punten**
Volgorde van aankomst:	**93,5%**	**5 punten**

Azure Notification Hub – Gemiddelde score na test #1 tm #4

Afleverzekerheid (Aankomstpercentage):	**100%**	**10 punten**
Gemiddelde afleversnelheid (incl. webservice call):	**700ms**	**5 punten**
Volgorde van aankomst:	**83,4%**	**0 punten**

Als de runtime scores van SignalR en Azure Notification Hub aan de tabel worden toegevoegd ontstaat de volgende uitslag:

	SignalR	Azure N.H.
Xamarin (eis)	10	10
Schaalbaarheid (20%)	5	10
Afleverzekerheid (15%)	10	10
Versleuteling (15%)	0	10
Afleversnelheid (10%)	10	5
Benodigde infrastructuur (5%)	10	9
Communicatie via derden (10%)	10	0
Berichten ontvangen indien app niet actief (0%)	0	10
Stroomverbruik (5%)	0	0
Volgorde van aankomst (5%)	5	0
HTML/JavaScript (5%)	10	0
Kosten/Licenties (5%)	10	0
Dataverkeer (5%)	0	0
Totaal	80	64
Info Support totaal (zwaarte zonder Xamarin)	**6,3**	**6,0**

Met oranje is in de tabel aangegeven dat stroomverbruik en dataverkeer uiteindelijk niet zijn getest. Dit zou in verder onderzoek opgepakt kunnen worden.
De onderste rij is de voor Info Support gewogen score. De formule die is gebruikt om dit te berekenen ziet er als volgt uit:

InfoSupportScore =
*(Schaalbaarheid * 20%) +*
*(Afleverzekerheid * 15%) +*
*(Versleuteling * 15%) +*
*(Afleversnelheid * 10%) +*
*(Benodigde infra. * 5%) +*
*(Comm. via derden * 10%) +*
*(Berichten ontv. App niet actief * 0%) +*
*(Stroomverbruik * 5%) +*
*(Volgorde van aankomst * 5%) +*
*(HTMLJavaScript * 5%) +*
*(KostenLicenties * 5%) +*
*(Dataverkeer * 5%)*

De frameworks zijn vergeleken en het blijkt dat op de getoetste criteria SignalR met 80 een flink hoger totaal heeft dan Azure Notification Hub. Met de zwaarte verrekend is het verschil echter erg klein, met 6,3 is SignalR net iets beter dan Azure Notification Hub met 6,0. Het zijn echter totaal verschillende oplossingen en er zijn toepassingen te bedenken waar één van deze frameworks beter tot zijn recht zou komen dan de ander.

Zo zou een toepassing waar berichten ontvangen moeten kunnen worden buiten de app, of waar security erg belangrijk is, beter kunnen werken met Azure Notification Hub. Maar wanneer er bijvoorbeeld een oplossing nodig is waarbij er ook een website gemaakt wordt, of er geen extra kosten mogen worden gemaakt, kan SignalR beter kan worden gebruikt.

8 Is zelfbouw een optie?

In het hoofdstuk “De techniek achter push communicatie”, zijn de transportprotocollen bekeken die gebruikt kunnen worden om push communicatie te implementeren. Dit zijn low-level technieken en het is veel werk om dit zelf toe te passen in een applicatie. Bovendien is het dan niet zomaar herbruikbaar.

De onderzochte push-frameworks leggen een abstractie laag over het gebruikte transportprotocol zodat het makkelijker te implementeren is. Soms gebruiken ze zelfs meerdere protocollen als back-up. Bovendien is de oplossing die een push-framework biedt ook direct herbruikbaar in volgende projecten.

De vraag is “Is zelfbouw een optie?”, kan het beste beantwoord worden door af te vragen of er functionaliteiten ontbreken in de al bestaande frameworks. Of wanneer er op een bepaald criterium zo slecht wordt gescoord dat er geen framework is dat aan de behoefte van een bepaalde applicatie voldoet.

Dan zijn er nog de kosten van ontwikkeling en onderhoud, vergeleken met de open source frameworks die er al zijn. Dan lijkt het een investering die niet nodig is.

Op basis van de opgestelde criteria en de scores van de frameworks lijkt het voor Info Support niet nodig om zelf een push-framework te gaan bouwen.

9 Conclusies en aanbevelingen

Na de toetsing van de push-frameworks op de criteria, die op basis van literatuur te toetsen waren, bleek dat SignalR, PushSharp en Azure Notification Hub direct te gebruiken zijn met Xamarin. Van deze drie had SignalR met 55 de hoogste totaalscore. Azure Notification Hub had met 4.0, door de zwaarte van de criteria, de hoogste gewogen score voor Info Support.

PushSharp en Azure Notification Hub zijn vergelijkbaar in werking en Azure Notification Hub scoort het hoogst van deze twee. SignalR en Azure Notification Hub zijn in een runtime test verder getoetst op de runtime criteria afleverzekerheid, afleversnelheid en volgorde van aankomst.

Tijdens de runtime tests hadden beide frameworks altijd een afleverzekerheid van 100%. SignalR is veruit het snelst in het afleveren van een pushbericht, gemiddeld ruim 10 keer zo snel. Indien er een payload met het bericht moet worden meegestuurd is SignalR het meest geschikt, hiermee is een extra payload tot 47Kb mogelijk. Met Azure Notification Hub ligt de max. bij een bericht naar een iOS device zelfs op 256 bytes, een behoorlijk verschil. Wat betreft de volgorde van aankomst, scoort SignalR, met 93,5% correct, beter dan Azure Notification Hub (83,4% correct).

Na deze toetsing op runtime criteria blijkt dat SignalR met 80 een hoger totaal heeft dan Azure Notification Hub, met 64. Met de zwaarte verrekend is het verschil echter erg klein, SignalR met 6.3 scoort net iets beter dan Azure Notification Hub met 6.0. Het zijn echter totaal verschillende oplossingen en er zijn toepassingen te bedenken waar één van deze frameworks beter tot zijn recht zou komen dan de ander.

	SignalR	Azure N.H.
Totaal	80	64
Info Support totaal (zwaarte zonder Xamarin)	**6,3**	**6,0**

Advies
Wanneer Info Support besluit push communicatie te implementeren in een mobiele applicatie, kan er op basis van de gestelde functionele requirements en de scoretabellen het best eerst gekeken worden welk push-framework er het meest geschikt is. Indien er geen specifieke eisen zijn gesteld, is SignalR aan te raden omdat het snel, betrouwbaar, makkelijk te implementeren en gratis te gebruiken framework is wat ook in web applicaties te gebruiken is.

Op basis van de opgestelde criteria en hoe de frameworks scoren lijkt het voor Info Support niet nodig om zelf een push-framework te ontwikkelen.

Vervolgonderzoek
De afleverzekerheid was bij SignalR en Azure Notification Hub in alle tests 100%. Het zou interessant kunnen zijn om dit met een gesimuleerde slechte verbinding verder te testen, om te kijken wat voor invloed dit heeft.

SignalR laat in de testapp zien dat het transportmechanisme Server Sent Events wordt gebruikt voor push communicatie. Volgens de documentatie van SignalR gebeurt dit wanneer er, óf op de server, óf op het device geen websockets transport beschikbaar is. Op zich is het positief dat SignalR zelf een geschikt transportmechanisme zoekt, alleen waarom er geen websockets transport wordt gebruikt, en of dit überhaupt mogelijk is, is nog een belangrijke resterende vraag.

De criteria stroomverbruik en dataverkeer zijn nog niet getest en zouden ook in een vervolgonderzoek meegenomen kunnen worden.

Verwijzingen

Adobe. (2014, januari 22). *XMLSocket - AS3* . Opgehaald van help.adobe.com: http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/flash/net/XMLSocket.html

Apple. (2014, februari 25). *About Local Notifications and Push Notifications*. Opgehaald van developer.apple.com: https://developer.apple.com/library/mac/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Introduction.html

BeaconPush. (2014, februari 18). *Real-time in the cloud*. Opgehaald van beaconpush.com: http://beaconpush.com/

Blackberry. (2014, maart 5). *Benefits of push technology*. Opgehaald van developer.blackberry.com: http://developer.blackberry.com/bbos/java/documentation/benefits_of_push_technology_1478211_11.html

Brands, H. (2014, maart 17). Interview #2. (M. Morsink, Interviewer)

Cornelissen, R., & Taling, M. (2013, maart 18). Real Time Event Feeds with NServiceBus and SignalR. Veenendaal.

Creative Bloq. (2013, september 25). *12 amazing tools for online collaboration*. Opgehaald van creatuvebloq.com: http://www.creativebloq.com/design/online-collaboration-tools-912855

Dick, J. (2014, januari 28). *PushSharp* . Opgehaald van GitHub: https://github.com/Redth/PushSharp

Fitzmacken, T., & Fletcher, P. (2014, januari 3). *Introduction to SignalR Security*. Opgehaald van asp.net: http://www.asp.net/signalr/overview/signalr-20/security/introduction-to-security

Google. (2014, februari 25). *Google Cloud Messaging for Android*. Opgehaald van developer.android.com: http://developer.android.com/google/gcm/gcm.html

Hydna. (2014, februari 18). *A scalable real-time platform*. Opgehaald van hydna.com: https://www.hydna.com/

IETF. (2014, februari 17). *The WebSocket Protocol.* Opgehaald van ietf.org: http://tools.ietf.org/html/rfc6455

Janssen, C. (2014, maart 5). *Push Technology*. Opgehaald van techopedia.com: http://www.techopedia.com/definition/5732/push-technology

LearnBoost (2). (2014, februari 18). *Socket.IO protocol*. Opgehaald van github.com/LearnBoost /socket.io-spec: https://github.com/LearnBoost/socket.io-spec

LearnBoost. (2014, februari 18). *Introducing Socket.IO*. Opgehaald van socket.io: http://socket.io

Meints, W. (2014, februari 25). Interview #1. (M. Morsink, Interviewer)

Microsoft. (2014, maart 3). *Azure Mobile Services*. Opgehaald van components.xamarin.com: http://components.xamarin.com/view/azure-mobile-services/

Microsoft. (2014, februari 18). *Introduction to SignalR*. Opgehaald van asp.net: http://www.asp.net/signalr/overview/signalr-20/getting-started-with-signalr-20/introduction-to-signalr

Microsoft. (2014, januari). *SignalR*. Opgehaald van github.com: https://github.com/SignalR/SignalR

Mono. (2014, maart 10). *Compatibility*. Opgehaald van mono-project.com: http://www.mono-project.com/Compatibility

O'Neill, C. (2012, augustus 24). *How push notifications can boost your mobile strategy*. Opgehaald van econsultancy.com: https://econsultancy.com/blog/10596-how-push-notifications-can-boost-your-mobile-strategy-3

Parse. (2014, februari 18). *Parse Push*. Opgehaald van parse.com: https://parse.com/products/push
PubNub. (2014, februari 18). *Build Realtime Apps that Scale*. Opgehaald van pubnub.com: http://www.pubnub.com/
Pusher. (2014, februari 18). *Build awesome realtime web and mobile apps*. Opgehaald van pusher.com: http://pusher.com
Roth, G. (2010, maart 31). *HTML5 Server-Push Technologies, Part 1*. Opgehaald van today.java.net: https://today.java.net/article/2010/03/31/html5-server-push-technologies-part-1#sse
Schiemann, D. (2007, november 5). *The forever-frame technique*. Opgehaald van Comet Daily: http://cometdaily.com/2007/11/05/the-forever-frame-technique/
Service 2 Media. (2014, februari 18). *Technical overview*. Opgehaald van service2media.com: http://www.service2media.com/app-platform/product-overview/push-messaging/technical-overview/
Thales e-Security. (2013, november 6). *Thales helps secure Microsoft's next-generation cloud service* . Opgehaald van thales-esecurity.com: https://www.thales-esecurity.com/company/press/news/2013/november/thales-helps-secure-microsofts-next-generation-cloud-service
W3C. (2009, oktober 29). *Server-Sent Events*. Opgehaald van w3.org: http://www.w3.org/TR/2009/WD-eventsource-20091029/
W3C. (2012, december 11). *Server-Sent Events*. Opgehaald van w3c.org: http://www.w3.org/TR/eventsource/
Wasson, M., & Patrick, F. (2014, maart 7). *Microsoft ASP.NET*. Opgehaald van Introduction to Scaleout in SignalR: http://www.asp.net/signalr/overview/signalr-20/performance-and-scaling/scaleout-in-signalr
Wikipedia. (2014, maart 5). *Firewall*. Opgehaald van wikipedia.org: http://nl.wikipedia.org/wiki/Firewall
Wikipedia. (2014, februari 17). *Internet Relay Chat*. Opgehaald van Wikipedia: http://nl.wikipedia.org/wiki/Internet_Relay_Chat
Wikipedia. (2014, februari 17). *Push technology*. Opgehaald van wikipedia.org: http://en.wikipedia.org/wiki/Push_technology
Wikipedia. (2014, februari 17). *Websocket*. Opgehaald van wikipedia.org: http://en.wikipedia.org/wiki/WebSocket

Bijlage A: Interview #1

Interview met Willem Meints 25-2-2014

1. **Wat voor soort mobiele applicaties ontwikkelt Info Support precies?**

Dit zijn enkel Enterprise applicaties. Het zijn dan vooral mobile extensies van administratieve systemen. Enkele voorbeelden van scenario's in onze mobile apps zijn: het inzien van een rooster, aanvragen van verlof, inspecties uitvoeren, offline data invoeren en dit later syncen . Het zijn geen standaard oplossingen, het zijn vaak specifieke oplossingen voor business processen.

2. **Zijn er al apps ontwikkeld door Info Support waar scenario's met push communicatie in voorkomen?**

Dit antwoord is uiteindelijk niet opgenomen omdat het informatie bevat over partners van Info Support die niet gepubliceerd dienen te worden.

3. **In wat voor scenario's denk je dat push technologie nog gebruikt gaat worden in de producten van Info Support?**

De vraag naar push functionaliteit is tot nu erg beperkt in de bestaande producten van Info Support.
In het product KnowNow zouden comments en notificaties live gepusht kunnen worden naar een app. Ook het monitoren van serverlogs d.m.v. push notificaties is iets wat er mogelijk wel gaat komen. Dit zou het monitoren van allerlei zaken minder arbeidsintensief kunnen maken.
Een mediabedrijf (naam verwijderd), een klant van Info Support, zou mogelijk in de toekomst applicaties met een 2nd screen kunnen inzetten.

4. **Is het binnen Info Support denk je belangrijk dat een push-framework zowel in een native app, als in een web app gebruikt kan worden?**

Geen must denk ik, maar het zou wel heel handig zijn.

5. **Ontwikkelt Info Support alles zelf, van interactie ontwerp, tot UI, tot backend, tot etc., of zijn er samenwerkingen met andere partijen?**

Voor het grootste deel wordt zelf alles ontwikkeld, maar soms wordt er samen met een design bureau gewerkt. Zij ontwerpen dan de interactie en de UI. Per project wordt gekeken of dit uitbesteed wordt.

6. **Zijn er algemene richtlijnen bij het ontwikkelen van mobiele apps? (zoals, we ondersteunen alleen iOS, Android en WP8)**

Standaard bouwen we in mobile apps in Xamarin, we ondersteunen dan Android, iOS, WP7 en WP8. Wil een klant toch dat we iets "echt" native ontwikkelen dan moet er een hele goede technische reden zijn waarom het niet in Xamarin gebouwd kan worden. Het kost namelijk meer tijd dan ontwikkelen met Xamarin.

7. **Er wordt ontwikkelt met Xamarin, staat dit vooraf vast als technische keuze? Of kan er in bepaalde gevallen ook besloten worden om native te ontwikkelen?**

Zie antwoord vraag 6.

8. Apps kunnen ook multi-platform worden ontwikkeld in HTML5 (bijv. met phonegap). Ontwikkelt Info Support ook dit soort oplossingen?

Hooguit experimenteren we hiermee. De ervaringen van het ontwikkelen met dit soort oplossingen is tot nu toe niet positief gebleken.

Bijlage B: Interview #2

Interview met Henk Brands 17-3-2014

1. **Hoe ziet de afdeling Mobile er binnen Info Support uit?**

De afdeling telt drie mensen en is nog vrij nieuw. In het begin heb je nog maar een klein portfolio, dit duurt even om op te bouwen. Wanneer we mobiele app's maken dan is dit vaak een stukje front-end van een veel grotere Enterprise applicatie.

2. **Info Support maakt vooral Enterprise applicaties voor klanten, houdt dit in dat consumenten dus helemaal niet in aanraking komen met de producten?**

Het kan uiteindelijk wel door consumenten worden gebruikt, we hebben klanten die een app wilden waarmee ze met hun klanten in contact komen. Voorbeelden hiervan zijn een slaapcoach en tickets app.

3. **Zijn er al apps ontwikkeld door Info Support waar scenario's met push communicatie in voorkomen?**

Voor zover ik weet niet, de meeste apps zijn invoer gedreven.

4. **De push-frameworks die ik tot nu toe heb onderzocht zeggen dat je push technologie kunt gebruiken voor onder andere de volgende functionaliteiten: chat, notificaties, activity streams, samenwerkingstools, realtime data in dashboards, 2nd screen oplossingen en remote controlling. In wat voor scenario's denkt u dat push technologie nog gebruikt gaat worden in de toekomstige producten van Info Support?**

Dit is een lastige vraag aangezien het niet is te voorspellen welke apps info support in de toekomst zal bouwen, daarbij is niks uitgesloten (nou ja games zijn uitgesloten).

5. **Denk je dat klanten zelf gaan vragen om oplossingen met push toepassingen of moeten we als Info Support dit aanbieden?**

Dit zou een integraal onderdeel moeten zijn van de requirements van de app.

6. **Heeft Info Support partners voor het ontwikkelen van mobile toepassingen?**

We hebben een nauwe samenwerking met (naam verwijderd). Zij ontwerpen in opdracht van Info Support de GUI van een app. Dit werkt twee kanten op, wanneer (naam verwijderd) een opdracht van een eigen klant heeft en de app (en backend) moet gerealiseerd worden, dan kan Info Support hier mee helpen.

Bijlage C: Runtime testopzet

Testopstelling
Server:

- Azure cloud service (size "small" met daarop een ASP.NET WCF webservice.
- Azure Notification Hub met capacity tier "FREE".

Client device:

- LG G2 draaiend op Android 4.2.2.
- Alle mogelijke background processen afgesloten.
- In Xamarin ontwikkelde TestSuite app.
- Alle tests zijn gedaan op het WiFi netwerk WLANFREE van Info Support. Locatie kamer 3.8.

Flow van de test

1. Tester geeft te gebruiken push-framework op.
2. Tester geeft het aantal berichten dat gepusht moet worden op.
3. Tester geeft het aantal milliseconden op dat er gewacht moet worden tussen de requests naar de server.
4. Tester geeft het aantal kb op dat als dummy payload aan het bericht moet worden toegevoegd.
5. Tester start de test.

6. Het volgende wordt herhaald voor het aantal berichten dat is opgegeven:
    a. App stuurt bericht met volgnummer, tijd van versturen en payload naar de server.
    b. Server laat het gekozen push-framework het bericht pushen naar de app.
    c. App ontvangt het pushbericht, berekent de time of flight van het bericht en bewaart het bericht (op volgorde) in een lijst.

7. Het volgende wordt 10x herhaald t.b.v. het berekenen van de tijd die het kost om de server aan te roepen:
    a. App stuurt de huidige tijd naar de server, de server retourneert de opgegeven tijd.
    b. App berekent de time of flight en telt deze op bij het totaal.

8. Tester start de analyse van de test.
9. App berekent gemiddelde duur webservice call.
10. App berekent de afleverzekerheid.
11. App berekent de afleversnelheid.
12. App berekent de correctheid van de volgorde van aankomst.

Berekening gemiddelde duur webservice call
De gemiddelde duur van een webservice call wordt op de volgende manier berekend:

DuurVanCallX = OntvangenOp – VerzondenOp

TotaleDuur = DuurVanCallX + DuurVanCallX enz…

GemiddeldeDuur = TotaleDuur / 10

Berekening afleverzekerheid
Het aankomstpercentage wordt op de volgende manier berekend:

*AankomstPercentage = (AantalAangekomen / AantalVerstuurd) * 100*

Berekening gemiddelde afleversnelheid
De gemiddelde afleversnelheid wordt op de volgende manier berekend:

DuurVanPushX = OntvangenOp – VerzondenOp

TotaleDuur = DuurVanPushX + DuurVanPushX enz…

GemiddeldeDuur = TotaleDuur / AantalOntvangen

Berekening correctheid volgorde van aankomst
De correctheid van de volgorde van aankomst wordt op de volgende manier berekend:

Er wordt door alle ontvangen berichten gelopen en gekeken hoeveel berichten er op goede volgorde zijn ontvangen, het volgende levert AantalGoed + 1 op:

Bericht.VolgNr > VorigBericht.VolgNr

Daarna wordt de correctheid van volgorde van alle berichten berekend

*Correctheid = (AantalGoed / TotaalOntvangen) * 100*

Deze aanpak zorgt ervoor dat een aankomstvolgorde van 1,2,4,**3**,5,6,7,8,9,10 een correctheid van 90% oplevert.

Invloed van ingestelde delay

De delay die wordt ingesteld kan een grote invloed hebben op de resultaten. De reden hiervoor is dat er bij een delay van 0ms een opstopping ontstaat op de backend-server die de requests niet meer direct kan afhandelen. Hierdoor is de time of flight van een bericht niet meer reëel omdat het in de wacht heeft gestaan. De correctheid van volgorde van aankomst kan hiermee wel goed worden getest omdat het push-framework de berichten zeer snel na elkaar moet pushen.

Bijlage D: Runtime testresultaten

Eindresultaat

Test #5 t/m #10 konden niet uitgevoerd worden met Azure Notification hub en zijn daarom buiten de berekening gelaten.

	Aankomstpercentage	Gemiddelde afleversnelheid	Volgorde van aankomst
SignalR test # 1 t/m #4	**100%**	**59ms**	**93,5%**
Azure Notification Hub test # 1 t/m #4	**100%**	**700ms**	**83,4%**

#1 – 250 berichten, 200ms delay, 0kb extra payload

	Gemiddelde duur webservice call	Aankomstpercentage	Gemiddelde afleversnelheid	Volgorde van aankomst
SignalR run 1	19ms	100%	54ms	100%
SignalR run 2	25ms	100%	43ms	100%
SignalR run 3	15ms	100%	46ms	100%
Azure Notification Hub run 1	28ms	100%	593ms	98%
Azure Notification Hub run 2	29ms	100%	1064ms	92%
Azure Notification Hub run 3	19ms	100%	565ms	98%
SignalR gemiddeld na 3 runs		**100%**	**47,7ms**	**100%**
Azure Notification Hub gemiddeld na 3 runs		**100%**	**740,7ms**	**96%**

#2 – 250 berichten, 0ms delay, 0kb extra payload

De gemiddelde duur webservice call en gemiddelde afleversnelheid zijn grijs gemaakt omdat deze bij een delay van 0ms niet realistisch meer zijn. Dit komt door de opstopping die er ontstaat op de server. Meer hierover staat beschreven in Invloed van ingestelde delay.

	Gemiddelde duur webservice call	Aankomstpercentage	Gemiddelde afleversnelheid	Volgorde van aankomst
SignalR run 1	472ms	100%	2902ms	86%
SignalR run 2	391ms	100%	1219ms	94%
SignalR run 3	255ms	100%	1526ms	93%
Azure Notification Hub run 1	2625ms	100%	14099ms	75%
Azure Notification Hub run 2	2912ms	100%	19600ms	67%
Azure Notification Hub run 3	1964ms	100%	12699ms	64%
SignalR gemiddeld na 3 runs		**100%**		**91%**

Azure Notification Hub gemiddeld na 3 runs		**100%**		**68,7%**

#3 –250 berichten, 200ms delay, 2kb extra payload

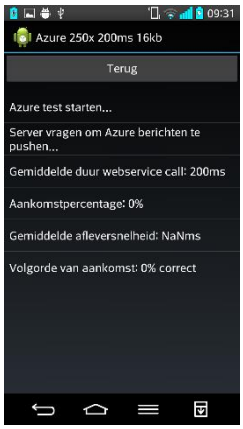
	Gemiddelde duur webservice call	Aankomstpercentage	Gemiddelde afleversnelheid	Volgorde van aankomst
SignalR run 1	129ms	100%	70ms	100%
SignalR run 2	26ms	100%	74ms	100%
SignalR run 3	32ms	100%	66ms	100%
Azure Notification Hub run 1	75ms	100%	543ms	98%
Azure Notification Hub run 2	19ms	100%	900ms	91%
Azure Notification Hub run 3	10ms	100%	537ms	99%
SignalR gemiddeld na 3 runs		**100%**	**70ms**	**100%**
Azure Notification Hub gemiddeld na 3 runs		**100%**	**660ms**	**96%**

#4 –250 berichten, 0ms delay, 2kb extra payload

	Gemiddelde duur webservice call	Aankomstpercentage	Gemiddelde afleversnelheid	Volgorde van aankomst
SignalR run 1	856ms	100%	4835ms	91%
SignalR run 2	24ms	100%	2788ms	91%
SignalR run 3	244ms	100%	787ms	97%
Azure Notification Hub run 1	1855ms	100%	13911ms	66%
Azure Notification Hub run 2	2339ms	100%	14220ms	77%
Azure Notification Hub run 3	2343ms	100%	17020ms	75%
SignalR gemiddeld na 3 runs		**100%**		**93%**
Azure Notification Hub gemiddeld na 3 runs		**100%**		**72,7%**

#5 – 250 berichten, 200ms delay, 16kb extra payload

	Gemiddelde duur webservice call	Aankomstpercentage	Gemiddelde afleversnelheid	Volgorde van aankomst
SignalR run 1	302ms	100%	189ms	100%
SignalR run 2	35ms	100%	117ms	100%
SignalR run 3	114ms	100%	118ms	100%
Azure Notification Hub run 1	-	-	-	-
SignalR gemiddeld na 3 runs		**100%**	**141,3ms**	**100%**
Azure Notification Hub gemiddeld na 3 runs		**-**	**-**	**-**



Tijdens het testen blijkt dat Azure Notification hub berichten naar Android met een payload tot en met 2kb kan versturen, vanaf 3kb wordt het niet verstuurd.

#6 – 250 berichten, 0ms delay, 16kb extra payload

	Gemiddelde duur webservice call	Aankomstpercentage	Gemiddelde afleversnelheid	Volgorde van aankomst
SignalR run 1	445ms	100%	2297ms	99%
SignalR run 2	943ms	100%	5415ms	96%
SignalR run 3	961ms	100%	4925ms	92%
Azure Notification Hub run 1	-	-	-	-
SignalR gemiddeld na 3 runs		**100%**		**95,7%**
Azure Notification Hub gemiddeld na 3 runs		**-**		**-**

#7 – 250 berichten, 200ms delay, 32kb extra payload

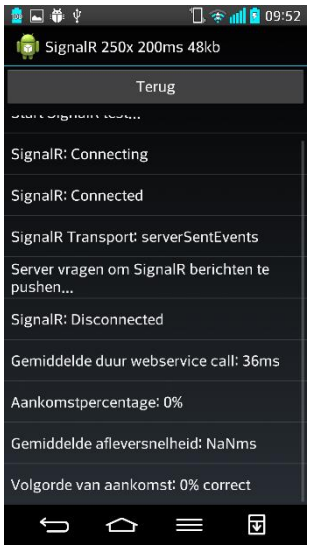
	Gemiddelde duur webservice call	Aankomstpercentage	Gemiddelde afleversnelheid	Volgorde van aankomst
SignalR run 1	23ms	100%	237ms	100%
SignalR run 2	11ms	100%	169ms	100%
SignalR run 3	25ms	100%	167ms	100%
Azure Notification Hub run 1	-	-	-	-
SignalR gemiddeld na 3 runs		**100%**	**191ms**	**100%**
Azure Notification Hub gemiddeld na 3 runs		**-**	**-**	**-**

#8 – 250 berichten, 0ms delay, 32kb extra payload

	Gemiddelde duur webservice call	Aankomstpercentage	Gemiddelde afleversnelheid	Volgorde van aankomst
SignalR run 1	901ms	100%	7416ms	91%
SignalR run 2	1018ms	100%	7790ms	92%
SignalR run 3	1227ms	100%	8831ms	90%
Azure Notification Hub run 1	-	-	-	-
SignalR gemiddeld na 3 runs		**100%**		**91%**
Azure Notification Hub gemiddeld na 3 runs		**-**		**-**

#9 – 250 berichten, 200ms delay, 48kb extra payload

	Gemiddelde duur webservice call	Aankomstpercentage	Gemiddelde afleversnelheid	Volgorde van aankomst
SignalR run 1	-	-	-	-
Azure Notification Hub run 1	-	-	-	-
SignalR gemiddeld na 3 runs				
Azure Notification Hub gemiddeld na 3 runs		**-**	**-**	**-**



Tijdens het testen blijkt dat SignalR berichten met een payload tot en met 47kb kan versturen, vanaf 48kb wordt het niet verstuurd.

#10 – 250 berichten, 0ms delay, 48kb extra payload

	Gemiddelde duur webservice call	Aankomstpercentage	Gemiddelde afleversnelheid	Volgorde van aankomst
SignalR run 1	-	-	-	-
Azure Notification Hub run 1	-	-	-	-
SignalR gemiddeld na 3 runs				
Azure Notification Hub gemiddeld na 3 runs		**-**	**-**	**-**

Bijlage C: Requirements

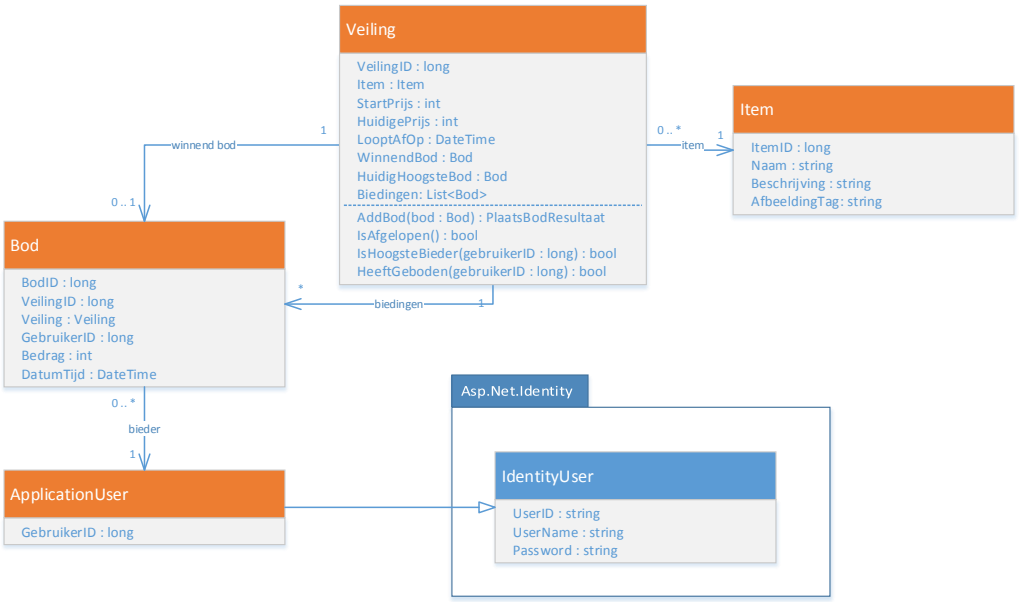
Functioneel	MoSCoW	NR
De eerste keer dat een gebruiker de app start wordt hij gevraagd om zijn naam.	M	F01
Het veilinghuis start elke 5 min automatisch een veiling.	M	F02
De resterende tijd van een veiling loopt af.	M	F03
Het veilinghuis rond een veiling af wanneer de tijd is verstreken.	M	F04
Een gebruiker kan een veiling bekijken.	M	F05
Een gebruiker kan op een veiling bieden.	M	F06
Een gebruiker kan in de app een veiling volgen.	M	F07
Een gebruiker kan stoppen een veiling te volgen.	M	F08
Een gebruiker ontvangt in de app veiling updates wanneer er veranderingen zijn in veilingen waar hij op heeft geboden.	M	F09
Een gebruiker kan de veilingen bekijken waar hij eerder op heeft geboden.	C	F10
Het veilinghuis verwijderd veilingen ouder dan 3 dagen.	C	F11

Contraints	MoSCoW	NR
Een gebruiker heeft een ongelimiteerde hoeveelheid geld waarmee hij op een veiling kan bieden.	M	C01
De initiële resterende tijd van een veiling is 1 uur.	M	C02
Een bod op een veiling is alleen geldig als het hoger is dan het huidige hoogste bod.	M	C03
Wanneer de resterende tijd van een veiling is verlopen kan er niet meer op geboden worden.	M	C04
De minimale verhoging van een bod is tot 10 steeds +1, na 10 steeds +5, na 100 steeds +10.	C	C05
Wanneer de resterende tijd van een veiling minder is dan 1 min wordt de resterende tijd verlengd met 20 sec na het uitbrengen van een bod.	C	C06
Wanneer een veiling wordt verlengd wordt de resterende tijd nooit langer dan 1 min.	C	C07
Een gebruiker kan een bod uitbrengen op een veiling waar hij zelf de hoogste bieder is, maar moet eerst bevestigen dat hij zichzelf overbied.	W	C08

Bijlage D: Ontwerpdiagrammen

Hier volgen enkele diagrammen uit het Technisch Ontwerp:

Domeinmodel (onderdeel van ontwerp WebService)

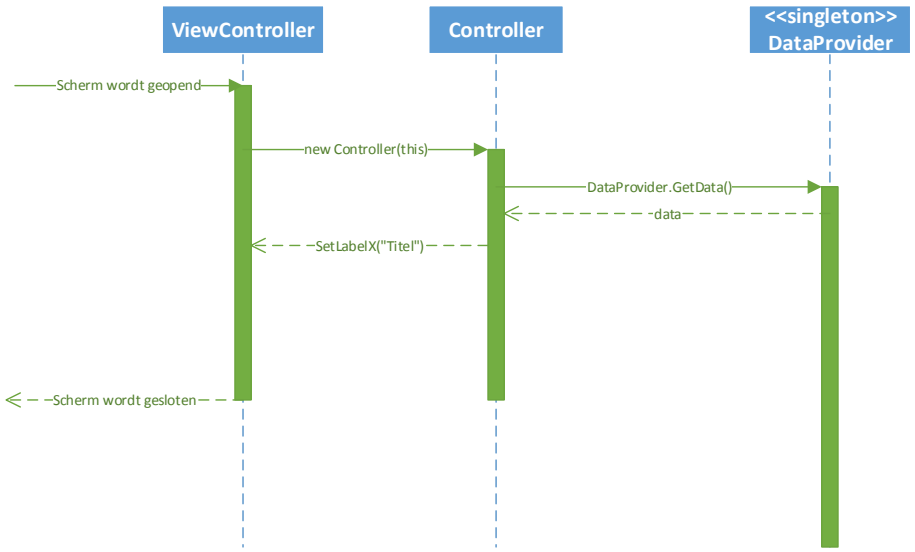


Openen scherm (onderdeel van ontwerp app)

Wanneer er een scherm aangeroepen op een device wordt er een ViewController geïnstantieerd. Deze maakt op zijn beurt een Controller aan en geeft bij de creatie zichzelf als IViewController mee.

De controller leeft nu zolang de ViewController leeft.

Wanneer de Controller data nodig heeft roep deze een singleton dataprovider aan. Wanneer dit voor het eerst gebeurt wordt de dataprovider automatisch geïnstantieerd en blijft deze de rest van de tijd bestaan.

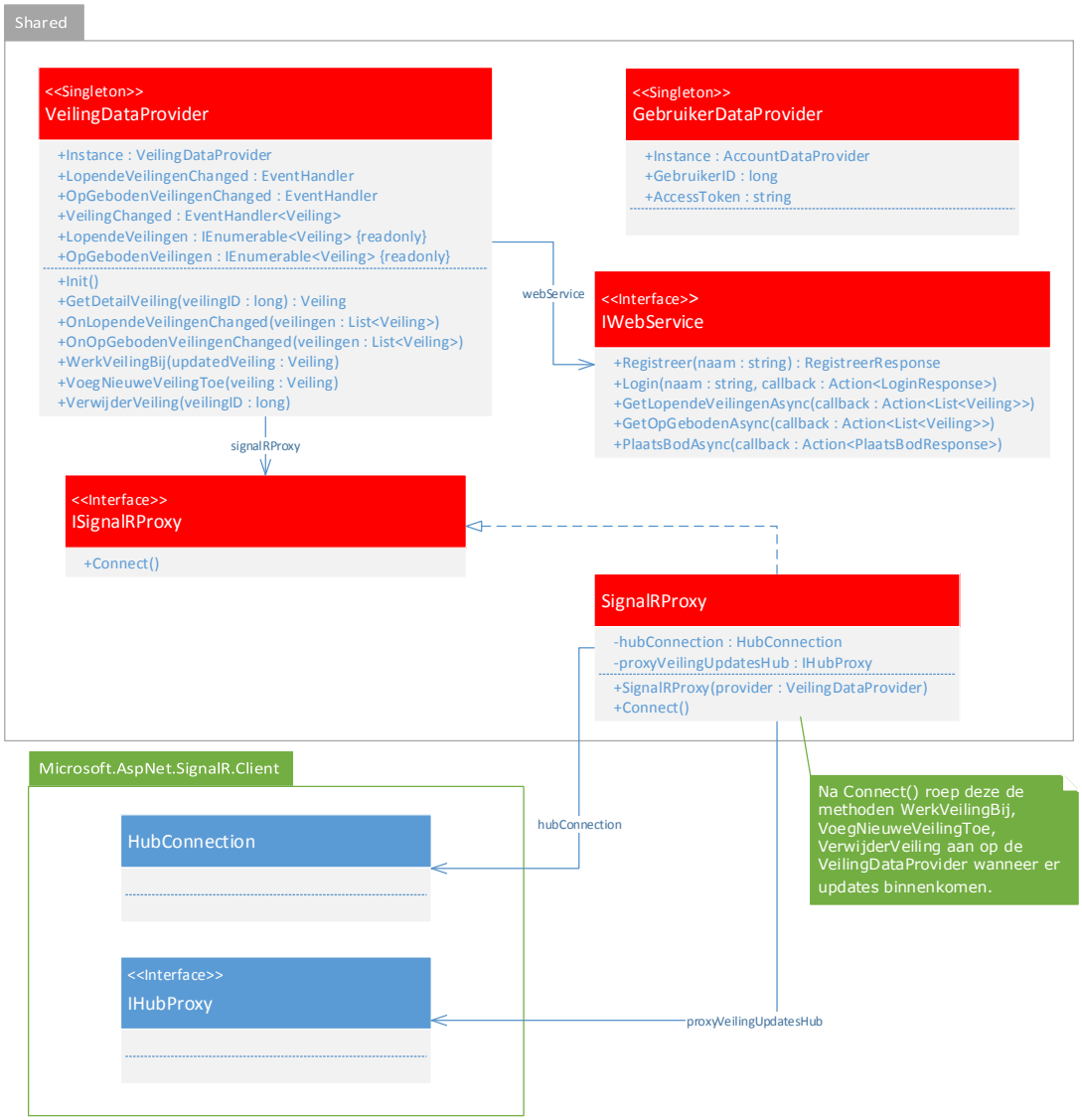


DataProviders en realtime updates (onderdeel van ontwerp app)

De DataProviders zijn singleton klassen. De reden waarom hiervoor is gekozen is omdat ze vanuit meerdere plekken in de applicatie te benaderen moeten zijn en ze één resource (de data) beheren die maar 1x mag bestaan.

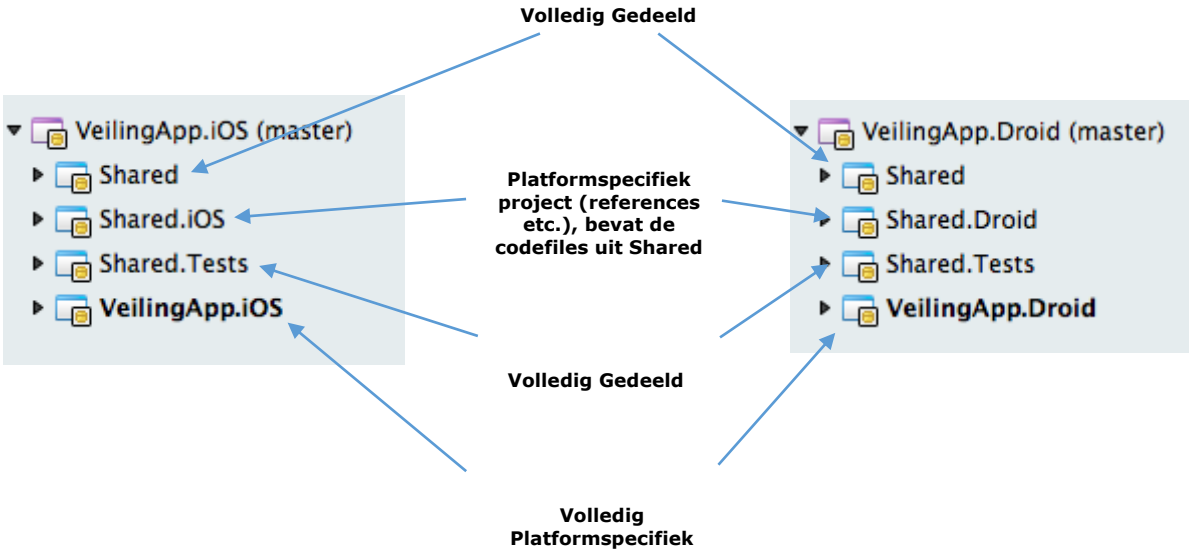
De VeilingDataProvider heeft een IWebService object waarmee er gecommuniceerd kan worden met de WebService. Ook heeft hij een ISignalRProxy waar updates van veilingen mee ontvangen kunnen worden.

Deze beiden objecten hebben een interface zodat deze in het unittest project kunnen worden vervangen door mock objecten.



Bijlage E: Codesharing strategie in Xamarin

Om zoveel mogelijk code binnen de app te kunnen gebruiken voor beide platformen, is er een bepaalde strategie nodig. Deze was als volgt:



Beide platformen kregen een eigen solution, die in dezelfde folder was geplakt. Binnen deze gedeelde solutionfolder was er een C# project genaamd **Shared** dat alle gedeelde code van de app bevatte. Dit project had tevens een NUnit testproject genaamd **Shared.Test**, beide projecten waren aan beide solutions toegevoegd.

De solutions hadden een eigen platformspecifiek App project. In de iOS solution was dit **VeilingApp.iOS**, in de Android solution was dit **VeilingApp.Droid**.

Het project **Shared** kon niet zomaar worden gebruikt door de platformspecifieke App projecten omdat deze een verschillend target framework hadden (MonoDroid/MonoTouch). Om dit mogelijk te maken kreeg elke solution een plaformspecifiek library project, wat de codefiles van het project **Shared** bevatte. Dit project heette in de iOS solution **Shared.iOS**, in de Android solution heette dit project **Shared.Droid**.

Door deze strategie te hanteren was het mogelijk om:

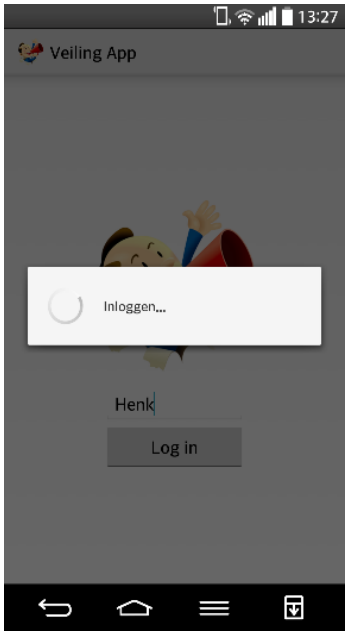
- De gedeelde code te builden voor beide platformen
- De gedeelde code te testen met een NUnit testproject

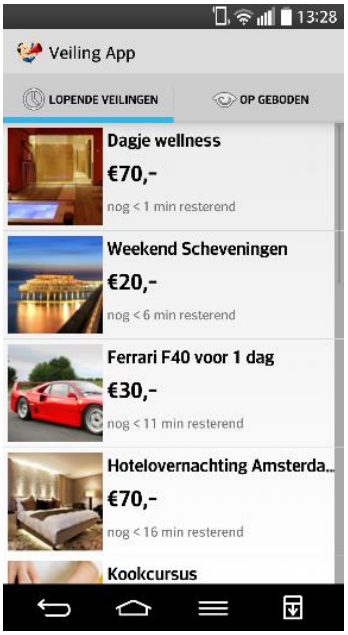
Bijlage F: Stored procedure PlaatsBod

```
CREATE PROCEDURE PlaatsBod @VeilingID bigint, @GebruikerID bigint, @BiedBedrag int
AS
BEGIN
    --Controlleren of gebruiker bestaat
    DECLARE @UsersCount int = 0;
    SELECT @UsersCount = count(*) FROM dbo.AspNetUsers WHERE GebruikerID = @GebruikerID;

    IF (@UsersCount != 1) BEGIN RAISERROR('Gebruiker bestaat niet', 16, 1); END ELSE BEGIN
        --Controlleren of veiling bestaat
        DECLARE @VeilingCount int = 0;
        SELECT @VeilingCount = count(*) FROM dbo.Veilings WHERE VeilingID = @VeilingID;

        IF (@VeilingCount != 1) BEGIN RAISERROR('Veiling bestaat niet', 16, 1); END ELSE BEGIN
            --Controleren of veiling niet afgelopen is
            SET @VeilingCount = 0;
            SELECT @VeilingCount = count(*) FROM dbo.Veilings WHERE VeilingID = @VeilingID AND LooptAfOp > GETUTCDATE();
            IF(@VeilingCount != 1) BEGIN RAISERROR('Veiling is afgelopen', 16, 1); END ELSE BEGIN

                --Controleren of bod vooraf hoog genoeg is
                DECLARE @HogereBiedingenCount int = 1;
                SELECT @HogereBiedingenCount = count(*) FROM dbo.Bods WHERE VeilingID = @VeilingID AND Bedrag >= @BiedBedrag;
                DECLARE @StartPrijs int = 0;
                SELECT @StartPrijs = StartPrijs FROM dbo.Veilings WHERE VeilingID = @VeilingID;

                IF (@HogereBiedingenCount != 0 OR @BiedBedrag < @StartPrijs) BEGIN RAISERROR('Bod te laag', 16, 1); END
                ELSE BEGIN
                    SET TRANSACTION ISOLATION LEVEL READ COMMITTED
                    BEGIN TRANSACTION;

                    --Veiling rij updaten, hierbij wordt een update lock gelegd voor andere threads,
                    --deze moeten wachten tot deze thread klaar is.
                    UPDATE dbo.Veilings SET HuidigeHoogsteBieder = @GebruikerID WHERE VeilingID = @VeilingID;

                    --Controleren of veiling nog steeds niet afgelopen is
                    SET @VeilingCount = 0;
                    SELECT @VeilingCount = count(*) FROM dbo.Veilings WHERE VeilingID = @VeilingID AND LooptAfOp > GETUTCDATE();
                    IF(@VeilingCount != 1) BEGIN RAISERROR('Veiling is afgelopen', 16, 1); END ELSE BEGIN

                        --Controleren of bod nog steeds hoog genoeg is
                        SELECT @HogereBiedingenCount = count(*) FROM dbo.Bods
                        WHERE VeilingID = @VeilingID AND Bedrag >= @BiedBedrag;

                        IF (@HogereBiedingenCount != 0) BEGIN ROLLBACK TRANSACTION; RAISERROR('Bod te laag', 16, 1); END
                        ELSE BEGIN
                            --Bod opslaan
                            INSERT INTO dbo.Bods (VeilingID, GebruikerID, Bedrag, DatumTijd, Veiling_VeilingID)
                            VALUES (@VeilingID, @GebruikerID, @BiedBedrag, GetDate(), @VeilingID);

                            COMMIT;
                        END
                    END
                END
            END
        END
    END
END
```

Bijlage G: Schermafbeeldingen app

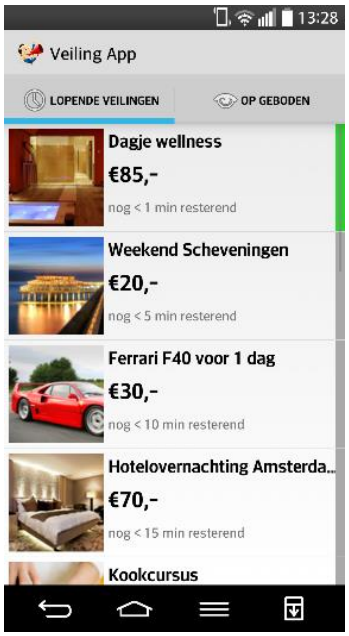
Android

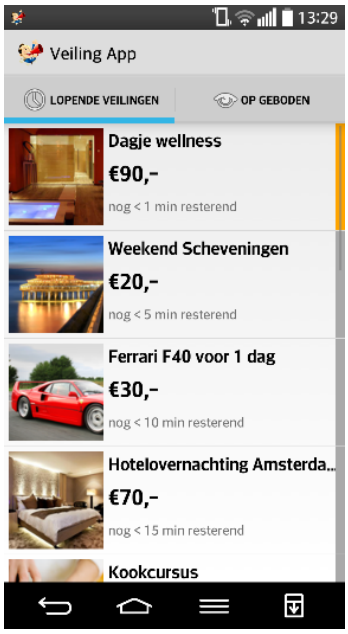


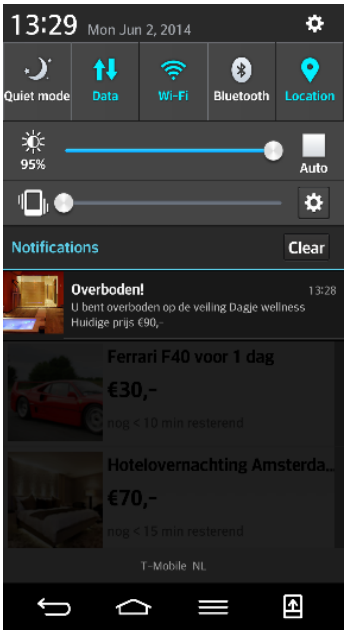



13:28
Dagje wellness
U bent de hoogste bieder!
Huidige prijs
Resterende tijd
€75,-
nog 0 min en 40 sec
Lorem Ipsum is slechts een proeftekst uit het drukkerij- en zetterijwezen. Lorem Ipsum is de standaard proeftekst in deze bedrijfstak sinds de 16e eeuw, toen een onbekende drukker een zethaak met letters nam en ze door elkaar
€80,-
Bieden

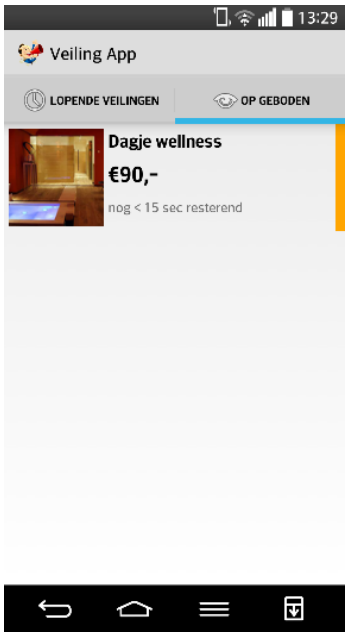

13:28
Dagje wellness
U bent overboden!
Huidige prijs
Resterende tijd
€80,-
nog 0 min en 42 sec
Lorem Ipsum is slechts een proeftekst uit het drukkerij- en zetterijwezen. Lorem Ipsum is de standaard proeftekst in deze bedrijfstak sinds de 16e eeuw, toen een onbekende drukker een zethaak met letters nam en ze door elkaar
€85,-
Bieden

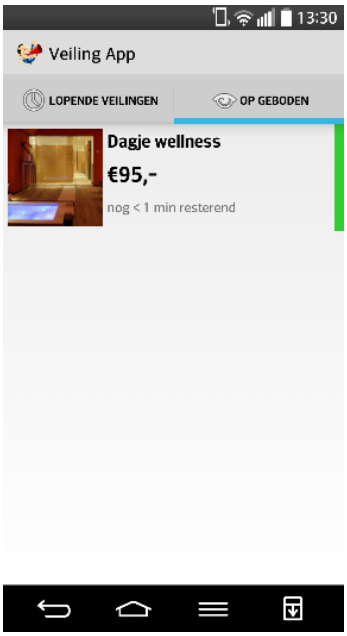

13:28
Veiling App
LOPENDE VEILINGEN
OP GEBODEN
Dagje wellness
€85,-
nog < 1 min resterend
Weekend Scheveningen
€20,-
nog < 5 min resterend
Ferrari F40 voor 1 dag
€30,-
nog < 10 min resterend
Hotelovernachting Amsterda...
€70,-
nog < 15 min resterend
Kookcursus

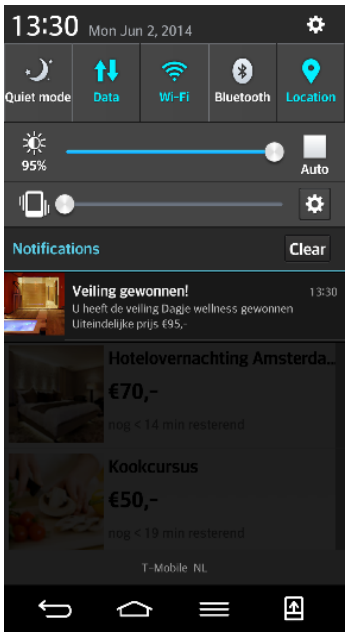

13:29
Veiling App
LOPENDE VEILINGEN
OP GEBODEN
Dagje wellness
€90,-
nog < 1 min resterend
Weekend Scheveningen
€20,-
nog < 5 min resterend
Ferrari F40 voor 1 dag
€30,-
nog < 10 min resterend
Hotelovernachting Amsterda...
€70,-
nog < 15 min resterend
Kookcursus


13:29
Mon Jun 2, 2014
Quiet mode
Data
Wi-Fi
Bluetooth
Location
95%
Auto
Notifications
Clear
Overboden!
13:28
U bent overboden op de veiling Dagje wellness
Huidige prijs €90,-
Ferrari F40 voor 1 dag
€30,-
nog < 10 min resterend
Hotelovernachting Amsterda...
€70,-
nog < 15 min resterend
T-Mobile NL


13:29
Dagje wellness
U bent overboden!
Huidige prijs
Resterende tijd
€90,-
nog 0 min en 22 sec
Lorem Ipsum is slechts een proeftekst uit het drukkerij- en zetterijwezen. Lorem Ipsum is de standaard proeftekst in deze bedrijfstak sinds de 16e eeuw, toen een onbekende drukker een zethaak met letters nam en ze door elkaar
€95,-
Bieden

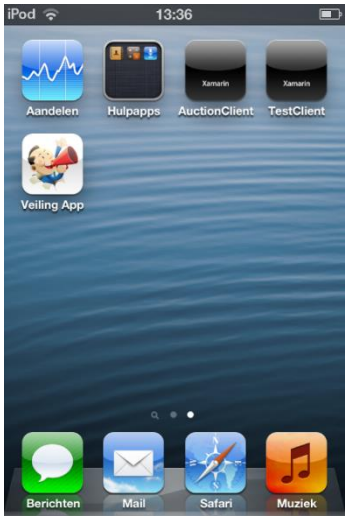

13:29
Veiling App
LOPENDE VEILINGEN
OP GEBODEN
Dagje wellness
€90,-
nog < 15 sec resterend


13:30
Veiling App
LOPENDE VEILINGEN
OP GEBODEN
Dagje wellness
€95,-
nog < 1 min resterend

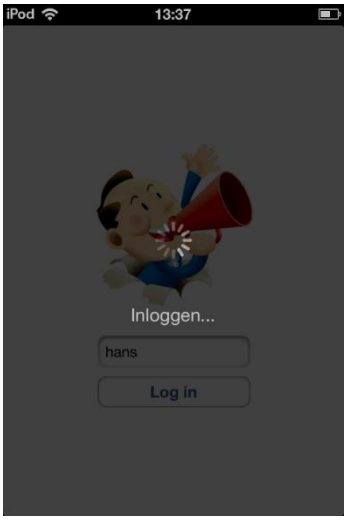

13:30 Mon Jun 2, 2014
Quiet mode
Data
Wi-Fi
Bluetooth
Location
95%
Auto
Notifications
Clear
Veiling gewonnen!
13:30
U heeft de veiling Dagje wellness gewonnen
Uiteindelijke prijs €95,-
Hotelovernachting Amsterda...
€70,-
nog < 14 min resterend
Kookcursus
€50,-
nog < 19 min resterend
T-Mobile NL


13:30
Dagje wellness
U heeft de veiling gewonnen!
Uiteindelijke prijs
€95,-
Lorem Ipsum is slechts een proeftekst uit het drukkerij- en zetterijwezen. Lorem Ipsum is de standaard proeftekst in deze bedrijfstak sinds de 16e eeuw, toen een onbekende drukker een zethaak met letters nam en ze door elkaar
We nemen zo spoedig mogelijk contact met u op

iOS


iPod
13:36
Aandelen
Hulpapps
AuctionClient
TestClient
Veiling App
Berichten
Mail
Safari
Muziek


iPod
13:36
InfoSupport
Solid Innovator


iPod
13:37
Inloggen...
hans
Log in


iPod
13:37
Lopende Veilingen
Ferrari F40 voor 1 dag
€30,-
nog < 3 min resterend
Hotelovernachting...
€70,-
nog < 8 min resterend
Kookcursus
€50,-
nog < 13 min resterend
Dagje Efteling
Lopende Veilingen
Op Geboden


iPod
13:37
Lopende Veili...
Ferrari F40 voor 1...
U kunt bieden op deze veiling
huidige prijs
€30,-
resterende tijd
nog 1 min en 58 sec
Lorem Ipsum is slechts een proeftekst uit het drukkerij- en zetterijwezen. Lorem Ipsum is de standaard proeftekst in deze
-
€35,-
+
Bieden

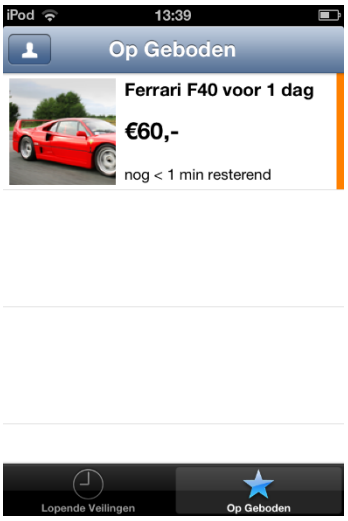

iPod
13:37
Lopende Veili...
Ferrari F40 voor 1...
U bent de hoogste bieder!
huidige prijs
€35,-
resterende tijd
nog 1 min en 51 sec
Lorem Ipsum is slechts een proeftekst uit het drukkerij- en zetterijwezen. Lorem Ipsum is de standaard proeftekst in deze
-
€40,-
+
Bieden


iPod
13:38
Lopende Veili...
Ferrari F40 voor 1...
U bent overboden!
huidige prijs
€40,-
resterende tijd
nog 0 min en 55 sec
Lorem Ipsum is slechts een proeftekst uit het drukkerij- en zetterijwezen. Lorem Ipsum is de standaard proeftekst in deze
-
€45,-
+
Bieden


iPod
13:38
Lopende Veilingen
Ferrari F40 voor 1 dag
€40,-
nog < 1 min resterend
Hotelovernachting...
€70,-
nog < 6 min resterend
Kookcursus
€50,-
nog < 11 min resterend
Dagje Efteling
Lopende Veilingen
Op Geboden


iPod
13:38
Lopende Veilingen
Ferrari F40 voor 1 dag
€45,-
nog < 1 min resterend
Hotelovernachting...
€70,-
nog < 6 min resterend
Kookcursus
€50,-
nog < 11 min resterend
Dagje Efteling
Lopende Veilingen
Op Geboden


iPod
13:39
Lopende Veilingen
Ferrari F40 voor 1 dag
€50,-
Overboden!
U bent overboden op de veiling Ferrari F40 voor 1 dag, huidige prijs €50,-
OK
€50,-
nog < 11 min resterend
Dagje Efteling
Lopende Veilingen
Op Geboden

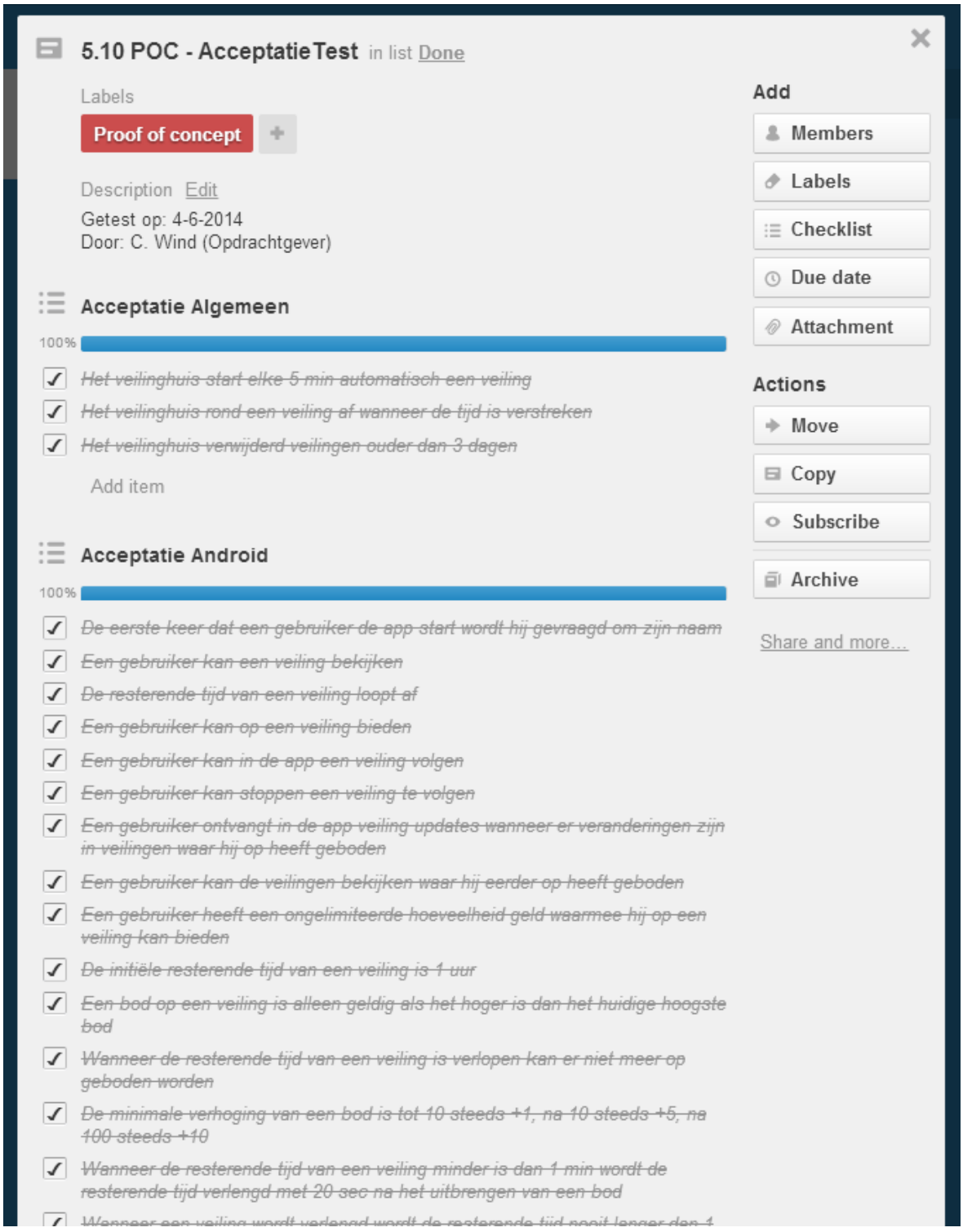

iPod
13:39
Op Geboden
Ferrari F40 voor 1 dag
€60,-
nog < 1 min resterend
Lopende Veilingen
Op Geboden


iPod
13:39
Op Geboden
Ferrari F40 voor 1 dag
€65,-
nog < 1 min resterend
Lopende Veilingen
Op Geboden


iPod
13:40
Aandelen
Hulpapps
Xamarin
AuctionClient
Xamarin
TestClient
Veiling App
Berichten
Mail
Safari
Muziek


iPod
13:40
Lopende Veilingen
Hotelovernachting...
€70,-
nog < 4 min resterend
Veiling gewonnen!
U heeft de veiling Ferrari F40 voor 1 dag gewonnen voor €65,-
OK
€60,-
nog < 14 min resterend
WK Trip Brazilië
Lopende Veilingen
Op Geboden

Bijlage H: Screenshot Acceptatie Test



5.10 POC - AcceptatieTest in list Done

Labels

Proof of concept +

Description Edit

Getest op: 4-6-2014
Door: C. Wind (Opdrachtgever)

Acceptatie Algemeen

100%

- ✓ ~~Het veilinghuis start elke 5 min automatisch een veiling~~
- ✓ ~~Het veilinghuis rond een veiling af wanneer de tijd is verstreken~~
- ✓ ~~Het veilinghuis verwijderd veilingen ouder dan 3 dagen~~

Add item

Acceptatie Android

100%

- ✓ ~~De eerste keer dat een gebruiker de app start wordt hij gevraagd om zijn naam~~
- ✓ ~~Een gebruiker kan een veiling bekijken~~
- ✓ ~~De resterende tijd van een veiling loopt af~~
- ✓ ~~Een gebruiker kan op een veiling bieden~~
- ✓ ~~Een gebruiker kan in de app een veiling volgen~~
- ✓ ~~Een gebruiker kan stoppen een veiling te volgen~~
- ✓ ~~Een gebruiker ontvangt in de app veiling updates wanneer er veranderingen zijn in veilingen waar hij op heeft geboden~~
- ✓ ~~Een gebruiker kan de veilingen bekijken waar hij eerder op heeft geboden~~
- ✓ ~~Een gebruiker heeft een ongelimiteerde hoeveelheid geld waarmee hij op een veiling kan bieden~~
- ✓ ~~De initiële resterende tijd van een veiling is 1 uur~~
- ✓ ~~Een bod op een veiling is alleen geldig als het hoger is dan het huidige hoogste bod~~
- ✓ ~~Wanneer de resterende tijd van een veiling is verlopen kan er niet meer op geboden worden~~
- ✓ ~~De minimale verhoging van een bod is tot 10 steeds +1, na 10 steeds +5, na 100 steeds +10~~
- ✓ ~~Wanneer de resterende tijd van een veiling minder is dan 1 min wordt de resterende tijd verlengd met 20 sec na het uitbrengen van een bod~~
- ✓ ~~Wanneer een veiling wordt verlengd wordt de resterende tijd nooit langer dan 1~~

Add
Members
Labels
Checklist
Due date
Attachment

Actions
Move
Copy
Subscribe
Archive

Share and more...