Delft University of Technology

Tam Tam in je broekzak

IN3405 Bachelor Project

8 juli 2012

Onderwijsinstituut: Technische Universiteit Delft

Bedrijf: Tam Tam B.V.

Studenten:

Bastiaan van Graafeiland	1399101
Wing Lung Ngai	1511483
Arvind Shailesh Jagesser	1550942

Commissie:

Begeleider TU Delft	Cor-Paul Bezemer
Begeleider Tam Tam	Wouter van Weelderen
Projectcoördinatoren	Gerd Gross & Peter van Nieuwenhuizen

Voorwoord

Dit document is het eindverslag van het vak *IN3405*, het eindproject van de bachelor Technische Informatica aan de TU Delft. In het eindproject wordt de opgedane kennis uit de andere vakken zoveel mogelijk toegepast gedurende een stage van 11 weken.

Wij hebben ervoor gekozen om de stage te doen bij Tam Tam in Rijswijk, waar een intern project is uitgevoerd. We willen Tam Tam bedanken voor het regelen van het project op korte termijn, en alle medewerkers die altijd bereid waren om vragen te beantwoorden en bestaande systemen uit te leggen. Verder bedanken we Bart Manuel en Wouter van Weelderen voor de coördinatie van het project en het opstellen van de backlog bij elke sprint. Ten slotte willen we Cor-Paul Bezemer bedanken voor de inhoudelijke begeleiding.

Delft, juli 2012

Bastiaan van Graafeiland
Wing Lung Ngai
Arvind Jagesser

Samenvatting

Tam Tam is een van de grootste Nederlandse full-service internet-bureaus, langs de A13 gevestigd op de grens van Delft en Rijswijk. Tam Tam draagt zorg bij voor allerhande online wensen die klanten maar hebben. Of het nu gaat om het verzorgen van een corporate internet site, een intranet, een mobiele applicatie, of zelfs een Facebook applicatie, niets is te gek. Tevens wordt onderhoud gepleegd aan reeds opgeleverde projecten.

Door het veelvuldige gebruik van smartphones binnen Tam Tam is het idee ontstaan van een mobile app die veelgebruikte bedrijfsapplicaties bevat. Deze mobile app zou bestaan uit een framework waarbinnen sub-apps worden gestart, die overeenkomen met huidige webapplicaties.

We hebben eerst onderzoek verricht naar de bestaande systemen, voorkeuren van de werknemers en standaard werkmethodes binnen Tam Tam. We hebben besloten te werken volgens de Scrum ontwikkelmethode. Vervolgens hebben we een algemene planning gemaakt, en een programma van eisen opgesteld. Hierin hebben we bepaald wat de prioriteit is van de verschillende eisen. De mobile app zou voor Android en iOS worden ontwikkeld, met een server back-end die beide systemen ondersteunt.

Vervolgens hebben we per sprint van twee weken een ontwerp gemaakt voor de eisen van die sprint, en dit ontwerp geïmplementeerd. Aan het eind van elke sprint hebben we de voortgang gepresenteerd en besproken met de begeleider. In de loop van het project zijn de eisen en prioriteit hiervan steeds bijgesteld in overleg. Zo is er besloten om ons volledig te richten op het framework, en geen sub-apps te ontwikkelen. Het is op deze manier makkelijker voor Tam Tam om het project voort te zetten.

Tijdens de laatste sprint hebben we ons vooral gericht op het testen van de frameworks en de server, en het opsporen van bugs door middel van exploratory testing en unit tests.

Het eindproduct variëert vrij veel van wat we oorspronkelijk hadden gepland na de oriëntatiefase. Het blijkt dat de Scrum ontwikkelmethode inderdaad de juiste aanpak was voor dit project. Via de tweewekelijkse sprint meeting zijn de begeleiders van Tam Tam steeds op de hoogte gesteld van het ontwikkelingsproces. Uiteindelijk kon het project op tijd worden afgerond, met de gewenste functionaliteit.

Inhoudsopgave

1 Inleiding

In de laatste jaren is het gebruik van smartphones enorm toegenomen. Waar het in 2005 nog bijzonder was om muziek te kunnen luisteren via je telefoon, dragen mensen nu een apparaat bij zich dat zich kan meten met een desktop computer. Hiermee is de vraag naar nuttige software voor smartphones, ofwel apps, ook toegenomen. Dit alles geldt niet alleen voor consumenten, maar ook voor bedrijven, waar werkprocessen efficiënter kunnen worden uitgevoerd met behulp van smartphones.

1.1 Tam Tam

Tam Tam is een van de grootste Nederlandse full-service internet-bureaus, langs de A13 gevestigd op de grens van Delft en Rijswijk. Tam Tam draagt zorg bij voor allerhande online wensen die klanten maar hebben. Of het nu gaat om het verzorgen van een corporate internet site, een intranet, een mobiele applicatie, of zelfs een Facebook applicatie, niets is te gek. Tevens wordt onderhoud gepleegd aan reeds opgeleverde projecten.

Zoals de meeste bedrijven, heeft ook Tam Tam een aantal al dan niet zelf ontwikkelde systemen, waarmee werknemers verschillende taken uitvoeren. Hierbij kan bijvoorbeeld worden gedacht aan het boeken van gemaakte uren op projecten (Tam Tam verkoopt uren, geen producten) of projectplanning. Deze systemen zijn echter allemaal ontwikkeld voor een browser in een desktop of laptop, en niet bepaald geschikt voor de kleinere beeldschermen van mobiele telefoons.

1.2 Het Project: Tam Tam in je broekzak

Iedere werknemer die een contract bij Tam Tam krijgt aangeboden, krijgt hierbij ook een mobiele telefoon, dit omdat er geen vaste telefoons aanwezig zijn. Hierdoor is de vraag naar een mobiele versie van interne systemen toegenomen. Het is gewenst deze systemen te gebruiken zonder eerst de computer op te moeten starten, of als er even geen computer binnen handbereik is.

Het doel van het project is om een framework te ontwikkelen voor een algemene Tam Tam app, die gemakkelijk moet kunnen worden uitgebreid met mobiele versies van de bestaande systemen. Gezien het persoonlijke karakter van de systemen moet er worden ingelogd door gebruikers, waarbij gebruikers hun eigen instellingen hebben. Verder is het wenselijk om berichten te kunnen ontvangen via de app, die door project- of teamleiders worden verstuurd.

1.3 Structuur verslag

Het verslag beschrijft de verschillende werkzaamheden uitgevoerd binnen het project. Eerst wordt beschreven hoe het project is georganiseerd en verlopen in hoofdstuk 2. In hoofdstuk 3 staat een aangepaste versie van het programma van eisen uit het oriëntatieverslag. Hoofdstukken 4, 5, 6 en 7 beschrijven het ontwerpen en implementeren van het systeem. In hoofdstuk 8 is te vinden hoe we zijn omgegaan met het commentaar van SIG. Het testen wordt beschreven in hoofdstuk 9. Vervolgens worden in hoofdstuk 9.3 de conclusies getrokken en worden er aanbevelingen gegeven voor eventuele voortzetting van het project, en geven wij onze persoonlijke ervaringen.

2 Project organisatie

In dit hoofdstuk wordt beschreven hoe het project is georganiseerd en verlopen. Eerst wordt de ontwikkelmethode die wij hebben gebruikt in dit project besproken. Daarna volgt een beschrijving van de project planning. Ten slotte wordt er nog een beknopte versie van het logboek van dit project gegeven.

2.1 Ontwikkelmethode - Agile Scrum

Dit project volgt de richtlijn van de Scrum softwareontwikkelingsmethode (1). Het project wordt opgesplitst in korte iteratieve sprints (2 weken). In het begin van elke sprint worden de requirements opnieuw vastgesteld in de product backlog. De backlog is een lijst van requirements die op prioriteit worden gesorteerd. De requirements met de hoogste prioriteit worden verwerkt in de komende sprint. Aan het einde van elke sprint wordt werkende software opgeleverd en in elke sprint vinden er opnieuw een requirements analyse, ontwerp en implementatie plaats.

2.1.1 Product Owner

De Tam Tam begeleider, Wouter van Weelderen, is de product owner. In het begin van elke tweewekelijkse sprint beslist de product owner samen met andere 3 developers van Tam Tam (Bart Manuel, Ramon Nagelhout, Roel Spruit) welke requirements de hoogste prioriteit krijgen. Deze requirements worden opgenomen in de backlog voor de komende sprint.

2.1.2 Development Team

Het development team is verantwoordelijk voor het ontwerp, de implementatie en het testen van de software. De leden van het team zijn wij zelf. In iedere sprint werkt het development team aan de requirements die in de backlog staan.

2.1.3 Scrum Master

Er werd geen Scrum Master aangewezen, omdat we met een klein development team dit eigenlijk niet nodig hadden. In de eerste sprint werkten we alle drie aan het Android framework en design. Daarna zijn we voornamelijk individueel aan de slag gegaan met de taken. Bastiaan werkte verder aan het Android framework, Wing aan de server en Arvind aan het iOS framework. Bij de tweewekelijkse sprint meeting was het duidelijk of we wel of niet aan belangrijke taken hadden gewerkt, omdat we hier de voortgang bespraken met de begeleider.

2.2 Project Planning

Het project duurt ongeveer 11 weken. Elke week bestaat uit circa 32-40 werkuren. De planning voor het project is in de volgende stappen onderverdeeld:

1. Oriëntatiefase (week 18 - 19) - Eerste 2 weken worden besteed aan het vooronderzoek.
    a. Tutorials – Basistutorials over mobile softwareontwikkeling bestuderen (2).
    b. Demo App – Een Android mobile app maken met basisfunctionaliteit.

c. Systeemanalyse – Het netwerksysteem van Tam Tam analyseren en bekijken welke applicaties en diensten worden aangeboden.
d. Gebruikersanalyse – Een enquête houden bij de medewerkers van Tam Tam.
e. Ontwerpkeuzes – Bekijken welke ontwerpkeuzes mogelijk zijn.

Deliverable: Oriëntatieverslag (zie Bijlage B met daarbij de systeemanalyse, de gebruikersanalyse en de ontwerpkeuzes), Demoversie van Android applicatie

2. Ontwikkelingsfase (week 20 - 25) – 3 iteratieve sprints (van 2 weken) worden besteed aan het ontwerp en de implementatie van de applicatie. In het begin van elke sprint wordt de backlog vastgesteld. In de backlog staan de taken voor de volgende sprint gesorteerd op prioriteit.
Deliverable: Uitbreiding van de vorige iteratie aan de hand van de backlog.

3. De testfase (week 26 - 27) – De laatste sprint wordt besteed aan het testen. Het doel is het inleveren van een stabiel eindproduct.
   a. Unit tests – Unit tests worden geschreven die de afzonderlijke klassen en methodes in isolatie testen.
   b. Exploratory testing – Het testen wordt handmatig uitgevoerd door de testers. De testers moeten beslissen of de software werkt als verwacht, en waar de bugs meestal voorkomen.

Deliverable: Unit Tests, Testverslag (zie hoofdstuk 0)

4. Afrondingsfase (week 28)
   a. Release - De release van de applicatie in de app stores.
   b. Code overdracht – De source code ter beschikking stellen aan Tam Tam.
   c. Eindverslag – Laatste documentatie inleveren.
   d. Presentatie – Eindpresentatie voor TU coördinator, TU begeleider en Tam Tam begeleider.

Deliverable: Eindverslag, Eindproduct

2.3 Logboek

De volgende tabel geeft aan welke taken zijn uitgevoerd tijdens welke fase. Een gedetailleerde versie van het logboek staat in bijlage D.

Wk	Dag	Fase	Taken
18 **19**	30 Apr - 4 Mei 7 Mei - 11 Mei	Oriëntatiefase	Introductie in Tam Tam Oriëntatieonderzoek en -verslag • Survey • User Analysis • System Analysis • Design Choices Demoversie van Android App • Design & Implementatie Development Server configureren

20
21
	14 Mei -18 Mei
	21 Mei -25 Mei
		Sprint 1	Front-end (Android)
			• UI
			  ○ login
			  ○ menu structuur
			  ○ pincode
			• ADFS Authenticatie
			• Encryptie van logingegevens
			Back-end
			• SQL Server configureren
			  ○ User Account/Database
			  ○ WCF Webservice tutorials
			1e Gesprek met TU begeleider Cor-Paul

22
23
	28 Mei - 01 Jun
	04 Jun - 08 Jun
		Sprint 2	Front-end (Android)
			• ServiceSingleton
			  ○ netwerk connectie naar server
			  ○ login
			  ○ applicaties beheren
			  ○ connectie buiten Tam Tam
			• UI
			  ○ applicatie beheren
			  ○ voor applicatie menu
			• Code refactoren (voorbereiden voor SIG)
			Back-end
			• Webservice
			  ○ uitbreiden voor applicaties abonneren
			  ○ Restful service met JSON
			• SQL Server – gebruikers data opslaan
			iOS Development
			• Basis structuur, Apps en settings
			1e keer code inleveren bij SIG

24
25
	11 Jun - 15 Jun
	18 Jun - 22 Jun
		Sprint 3	Front-end (Android)
			• Push notificatie
			  ○ connectie met C2DM
			  ○ connectie met development server
			• ServiceSingleton
			  ○ opsplitsen & code refactoren
			  ○ https functionaliteit
			• Externe applicaties laden
			Back-end
			• Webservice
			  ○ push notificatie server kant
			  ○ code refactoren
			• System Admin WebPage voor webservice
			iOS Development
			• Service Singleton
			  ○ Beheert alle connecties, login etc.
			• Pin, alle functionaliteit
			• RootViewController, laad webapps & Settings
			• Applicaties beheren
			• Login scherm
			• Settings scherm, logout etc.
			SVN opruimen
			Voortgangsgesprek met Cor-Paul

26 **27**	25 Jun - 29 Jun 02 Jun - 06 Jul	Sprint 4	Testing framework onderzoeken Unit testen - mobile apps Unit testen – webservice 2e keer code inleveren bij SIG iOS Development • Push Notifications en testen op een iPhone Stageverslag
28	09 Jul - 13 Jul	Afrondingsfase	Release Presentatie

Tabel 2-1: Beknopte versie van het logboek

3 Programma van Eisen

Vroeg in het project hebben we een programma van eisen opgesteld. In de loop van het project is deze lijst enigszins gewijzigd. In dit hoofdstuk wordt eerst beschreven hoe de requirements analyse is uitgevoerd. Daarop volgt een complete lijst van de uiteindelijke requirements. Ten slotte wordt uitgelegd waarom bepaalde requirements uit deze lijst niet zijn geïmplementeerd**.**

3.1 Werkwijze van requirements analyse

De requirements van deze applicatie zijn als volgt tot stand gekomen:

1. De opdrachtomschrijving (zie bijlage A) bestuderen.
2. Eerste gesprek met de opdrachtgever om een beter inzicht te krijgen in het bedrijf.
3. In de oriëntatiefase vindt een vooronderzoek plaats. (zie bijlage B)
    a. Gebruikersanalyse wordt gedaan in de vorm van een enquête.
    b. Systeemanalyse wordt gedaan op de bestaande infrastructuur van Tam Tam.
4. Aan het einde van de oriëntatiefase is een draftversie van de requirement lijst vastgesteld.
5. In het begin van iedere sprint worden de requirements samen met de projectbegeleiders opnieuw vastgesteld en op prioriteit gesorteerd.
6. De requirements met de hoogste prioriteit worden in de backlog gezet.

3.2 Lijst van Requirements

De requirements worden geclassificeerd volgens de MoSCoW-methode (3):

- Must Have: moet in het eindproduct terugkomen.
- Should Have: komt in het eindprodcut aan bod zolang het mogelijk is.
- Could Have: komt alleen aan bod als er nog tijd over is.
- Won't: wordt niet verwerkt in dit project, misschien in de toekomst wel.

De tijdsperiode geeft aan in welke sprints de requirements worden gerealiseerd. Een requirement met een vink betekent dat die requirement is volledig verwerkt in het eindproduct. Een requirement met een kruis betekent dat die requirement is niet verwerkt.

	Requirement	**Prioriteit**	**Tijdsperiode**	
1.	De mobile app werkt in Android telefoons	M	Alle sprints	✓
2.	De mobile app werkt in iPhone	M	Sprint 2-4	✓
3.	De mobile app werkt in iPad	W	-	✘
4.	De mobile app werkt in Android tablets	W	-	✘
5.	De mobile app werkt in Windows Phone	W	-	✘
6.	Nederlandse/Engelse schermtekst	M	Alle sprints	✓
7.	Mogelijkheid tot uitbreiden naar andere talen	S	Alle sprints	✓
8.	Inloggen met Tam Tam account	M	Sprint 1	✓
9.	Inloggen met ADFS authenticatie	S	Sprint 1	✘

10.	Gebruikersnaam en wachtwoord opslaan	S	Sprint 1	✓
11.	De mobile app wordt beschermd met pincode	C	Sprint 1	✓
12.	Pincode interval veranderen	C	Sprint 1	✓
13.	RESTful data uitwisseling met de web server ondersteunen	M	Sprint 2	✓
14.	Data uitwisseling met de web server met SSL encryptie	S	Sprint 3	✘
15.	Single-Sign-On ondersteuning voor native sub-apps	M	Sprint 2	✓
16.	Single-Sign-On ondersteuning voor web sub-apps	M	Sprint 2	✓
17.	Dynamische menustructuur voor sub-apps	M	Sprint 2	✓
18.	Native sub-apps toevoegen in de mobile app	S	Sprint 2	✓
19.	Web sub-apps toevoegen in de mobile app	M	Sprint 2	✓
20.	Sub-apps toevoegen aan of verwijderen uit het menu	M	Sprint 2	✓
21.	Bijhouden welke gebruikers zijn geabonneerd op welke sub-apps	M	Sprint 2	✓
22.	De nieuwe sub-apps weergeven in het menu	M	Sprint 2	✓
23.	Externe mobile apps aanroepen op de applicatiemenu	C	Sprint 3	✓
24.	Push Notification toevoegen op een telefoon	S	Sprint 3-4	✓
25.	Push Notification verwijderen als gebruiker is uitgelogd	S	Sprint 3-4	✓
26.	Push Notification verzenden vanaf de server	S	Sprint 3-4	✓
27.	Updaten via Android Market	S	-	✘
28.	Updaten via Apple Store	S	-	✘
29.	Native functionaliteit (zoals camera) ondersteunen	W	-	✘
30.	Functioneren onder langzame internetverbinding	S	Alle sprints	✓
31.	Tenminste twee sub-apps bevatten	C	-	✘

Tabel 3-1: Alle opgestelde requirements

3.3 Niet-geïmplementeerde requirements

In deze paragraaf wordt uitgelegd waarom bepaalde requirements uit de lijst niet zijn geïmplementeerd.

	Requirement	**Reden voor het afzien van implementatie van requirement**
3	De mobile app werkt in iPad	Deze requirement heeft een lage prioriteit. Een iPad heeft een groot scherm en hier zijn de bestaande applicaties van Tam Tam goed zichtbaar op.
4	De mobile app werkt in Android tablets	Deze requirement heeft een lage prioriteit. De Android tablets hebben ook een scherm dat groot genoeg is om de bestaande applicaties goed te weergeven.
5	De mobile app werkt in Windows Phone	Uit de gebruikersanalyse blijkt dat slechts 14% van Tam Tam een Windows Phone gebruikt. Het is nuttiger om de tijd voor het project te besteden aan de Android (36%) & iOS (46%) platforms.

9	Kunnen inloggen met ADFS authenticatie	ADFS authenticatie is een requirement vastgesteld door de begeleiders in het begin van het project, omdat zij dachten dat het noodzakelijk was. Het bleek vrij complex te zijn, zeker om te laten samenwerken met de bestaande infrastructuur, daarnaast biedt ADFS meer functionaliteit dan nodig is. We werden door PCS (de IT-afdeling en netwerkbeheer) geadviseerd een andere manier te gebruiken voor authenticatie. Uiteindelijk bleek dat Basic Authentication ook voldoet aan de eisen.
14	Data uitwisseling met de web server met SSL encryptie	Er moet eerst een certificaat worden aangeschaft. Mogelijk wordt dit nog nader onderzocht in de afrondingsfase.
29	Pushen van updates via Android Market	Er is geen Android Developer Account beschikbaar. Misschien wordt hier nog aan gewerkt in de afrondingsfase van het project. In ieder geval kan de mobile app op andere manieren worden gedistribueerd door deze te exporteren als APK bestand, waarna elke Android gebruiker dit bestand kan downloaden om de mobile app te installeren
30	Pushen van updates via Apple Store	Apple moet de code nog evalueren nadat de iOS mobile app is volledig af is. Dit heeft een verwerkingstijd van twee weken, binnen het project hebben wij hier geen tijd meer voor.
31	Gebruikt native functionaliteit zoals camera	Deze requirement heeft een lage prioriteit. Het is namelijk nog niet bekend welke native functies nuttig zouden kunnen zijn, maar dit is eventueel gemakkelijk uit te breiden.
33	Bevat ten minste twee sub-apps	Om een volledig framework plus sub-apps af te krijgen, hadden we simpelweg niet genoeg tijd. In de loop van het project bleek dat in de beschikbare tijd alleen het framework geïmplementeerd kon worden. We hebben gekozen om een stabiele release van het framework op te leveren waar anderen gemakkelijk op verder kunnen bouwen, in plaats van een half bruikbare sub-app die nog veel bugs bevat.

Tabel 3-2: Redenen voor niet-geïmplementeerde requirements

4 Ontwerp

Aangezien we hebben gewerkt volgens de Scrum ontwikkelmethode, hebben we niet eerst het gehele systeem ontworpen alvorens te beginnen met implementeren. In plaats hiervan hebben we per sprint eerst nagedacht over het ontwerp en daarna iets geïmplementeerd, zodat we aan het eind van elke sprint de voortgang konden presenteren aan de begeleiders. Aan het begin van elke sprint is het doel voor die sprint ruwweg vastgesteld in overleg met de begeleiders, waarna wij dit zelf hebben vertaald naar een concreet ontwerp.

Het ontwerp is een globale omschrijving van de verschillende elementen van het systeem, die dus niet afhankelijk zijn van implementatie-specifieke details zoals gebruikte API's of platform. Af en toe was het nodig om het ontwerp uit een eerdere sprint aan te passen om het beter te laten aansluiten op het huidige werk. Hierbij kan worden gedacht aan kolommen in de database, de manier waarop componenten samenwerken, of de verdeling van het werk tussen telefoon en server.

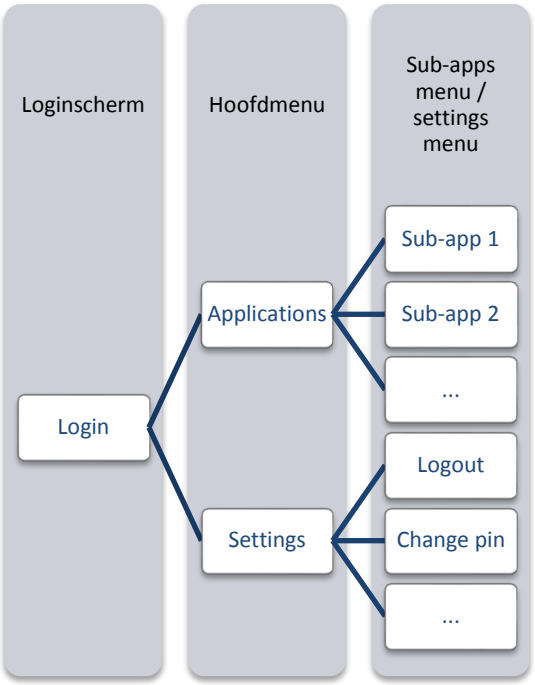
In dit hoofdstuk komt het ontwerp van de elementen van de mobile app en de server aan bod, en de verdeling tussen deze subsystemen.

4.1 Mobile app

Bij het ontwerpen van de mobile app zijn we begonnen met de user interface, dit was wenselijk omdat de user interface invloed heeft op het ontwerp van de uiteindelijke back-end.

4.1.1 User interface

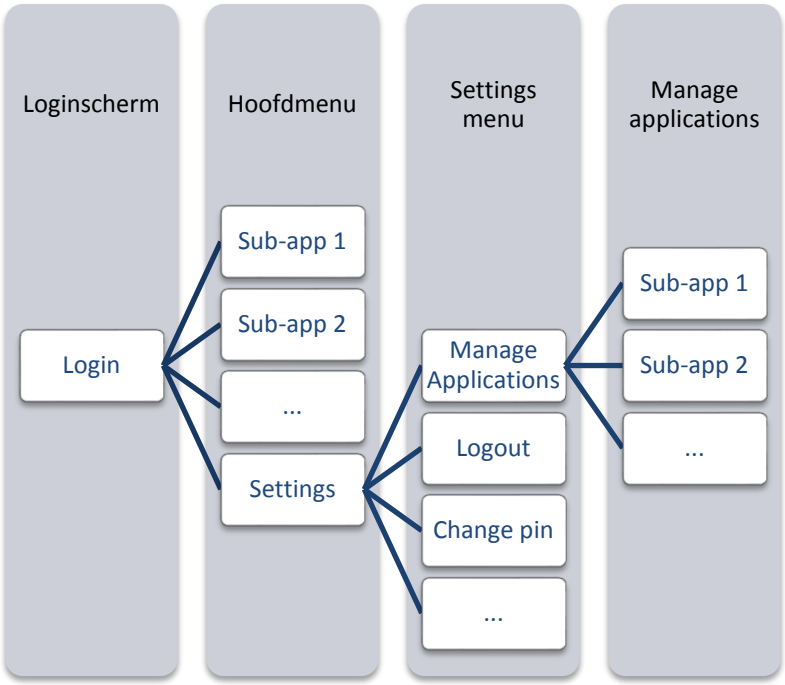
Uit de opdrachtomschrijving bleek al dat er in ieder geval een loginsysteem en menustructuur aanwezig moest zijn. We hebben allereerst een mockup gemaakt van de menustructuur met placeholder items. De indeling was als volgt:



Figuur 4-1: Eerste versie menustructuur

Op dat moment was het idee om alle sub-apps web-based te maken, en binnen de mobile app weer te geven in een embedded browser view. Ook werden alle bestaande sub-apps weergeven in het sub-apps menu. Later bleek echter dat de Tam Tam begeleider iets anders in gedachten had, namelijk dat vanuit de instellingen een lijst met alle sub-apps wordt weergegeven, waaruit de gebruiker dan kan selecteren welke hiervan in het hoofdmenu zullen verschijnen. Een apart menu voor de sub-apps was niet gewenst.

Uiteindelijk is het ontwerp in grote lijnen hetzelfde gebleven, maar worden alle sub-apps nu direct weergegeven in het hoofdmenu. Ook zijn er meerdere soorten sub-apps mogelijk: native, web of extern. Het bleek namelijk niet zo veel werk als verwacht om native sub-apps te ontwikkelen, aangezien het meeste werk bij de server wordt gedaan. De native sub-apps hebben een eigen layout die bij de mobile app hoort, de web sub-apps worden geladen in een browser view, en voor de externe sub-apps is er een link in de lijst.



Figuur 4-2: Uiteindelijke menustructuur

4.1.2 Back-end

De back-end van de app kan worden omschreven als de verbinding tussen de user interface op de telefoon, en de API op de server. Het ontwerpen hiervan is een continu proces geweest, waarbij steeds componenten moesten worden aangepast of toegevoegd om alles goed te laten samenwerken. Hiernaast moest ook worden voldaan aan de eisen die Android stelt, zoals het gebruik van bepaalde interfaces en klassen van Android, en de manier waarop data kan worden overgedragen tussen verschillende componenten.

Kern

Een van de voornaamste behoeftes voor de back-end was een centrale component met functionaliteit die gebruikt kan worden door alle andere componenten binnen de mobile app. Deze component bestaat weer uit drie sub componenten: utility, network en alerts. Het utility gedeelte bevat een aantal variabelen met informatie over de gebruiker en een aantal methodes die overal gebruikt moeten kunnen worden, zoals een controle of er internetverbinding is, of de URL van onze eigen server, waar veelvuldig gebruik van wordt gemaakt. Het network gedeelte verzorgt alle communicatie met de server. Hierbij kan worden gedacht aan het synchroniseren van instellingen, de authenticatie, en het verkrijgen van gebruikersgegevens. Ten slotte is er nog het alerts gedeelte, dat methodes bevat om feedback te geven aan de gebruiker door middel van pop-ups of notificaties.

Voor de bestaande web-based applicaties is het nodig om elke keer opnieuw in te loggen bij gebruik. Daarom was er behoefte aan een Single Sign On mechanisme, ter verbetering van de gebruikerservaring. Onze oplossing hiervoor was om eenmalig een inlogscherm te tonen, waarna de logingegevens in de telefoon worden opgeslagen, zodat de volgende keer het inlogscherm kan worden overgeslagen. Hierbij hoort uiteraard ook een mogelijkheid om uit te loggen. In dat geval worden alle gegevens van de gebruiker verwijderd van de telefoon en kan eventueel iemand anders inloggen.

Een voorstel van de Tam Tam begeleider was ook om de gebruiker een pincode te laten invoeren. De reden hiervoor was dat enige vorm van beveiliging gewenst is, maar dat een complete gebruikersnaam en wachtwoord elke keer invoeren ongewenst is. De pincode bestaat uit 4 cijfers, en kan door de gebruiker zelf worden gekozen. Op deze manier kan de app alleen door de eigenaar van de telefoon worden gebruikt, zonder al te veel ongemak.

Push Notifications - Android

Er moest een manier komen om berichten te sturen naar gebruikers. Dit is vergelijkbaar met SMS, maar dan naar bepaalde groepen van Tam Tam medewerkers. Voor de Android versie hebben we hiervoor besloten een bestaand systeem van Google te gebruiken: Cloud to Device Messaging, ofwel C2DM. Hiermee kan relatief eenvoudig een server gebruikt worden om berichten te sturen naar telefoons, die op korte termijn aankomen zonder dat de telefoon periodiek verbinding hoeft te maken met de server. De maximale lengte voor een bericht verstuurd vanaf de C2DM server (van Google) is ongeveer 1000 karakters. Wij hebben besloten dat dit voorlopig voldoende is, aangezien de service voornamelijk zal worden gebruikt voor korte mededelingen via notificaties. In bijlage E wordt C2DM in meer detail uitgelegd.

Push Notifications - iOS

Voor de Push Notifications was er alleen de optie om te communiceren met de APNS (Apple Push Notification Service) server van Apple. Met behulp van de APNS server is het mogelijk om je eigen bericht naar alle geregistreerde iPhones te sturen. Dit vereist wel wat moeite met certificaten en private keys.

4.1.3 Eisen aan de gebruiker

De voornaamste eis van de app als geheel is een actieve internetverbinding, aangezien vrijwel alle functionaliteit een verbinding met de server vereist. Als hieraan niet wordt voldaan, kan de gebruiker niet voorbij het inlogscherm komen en dus eigenlijk niets doen. Ten tweede is een geldig account bij Tam Tam benodigd, aangezien de app alleen voor werknemers is bedoeld. Ten slotte moet voor Android een Google-account zijn gekoppeld aan de telefoon om berichten te kunnen ontvangen via C2DM.

4.2 Server

Zowel de mobile app als de sub-apps hebben een centraal aanspreekpunt nodig voor een beter overzicht, dus is een virtuele server van Tam Tam aangevraagd als back-end ondersteuning. In de server bevinden zich de webservice, een database en een System Administration tool. In deze paragraaf wordt het ontwerp van de server applicaties besproken.

Vrijwel alle functionaliteiten van de mobile app zijn afhankelijk van onze server. Deze server heeft toegang tot de database, waar persoonlijke instellingen zijn opgeslagen. Hiernaast dient deze als link tussen de bestaande webservices van Tam Tam en de mobile app, zodat de mobile app maar op één manier hoeft te communiceren met onze server, in plaats van verschillende webservices met verschillende protocollen. Al het "vertaalwerk" wordt door onze server gedaan, en de mobile app hoeft alleen met deze server te communiceren.

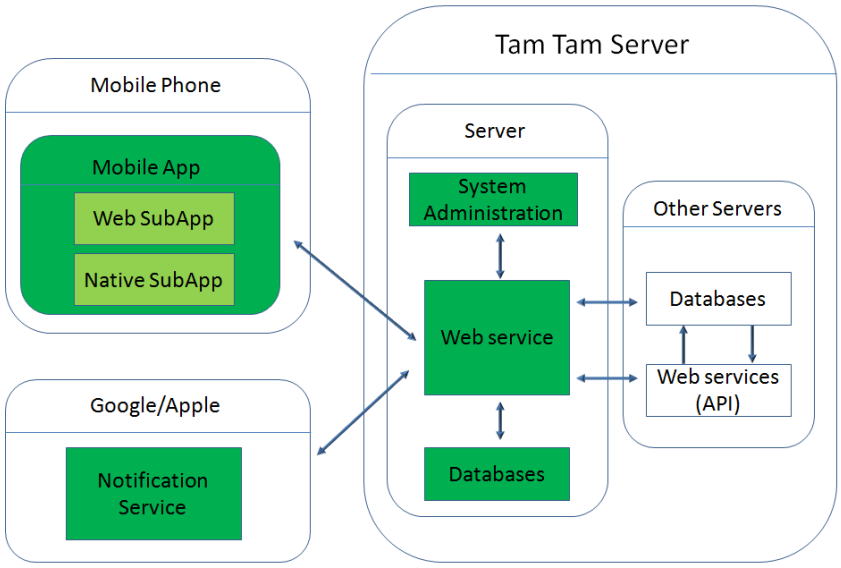
Omdat de services gebruikt zullen worden door twee verschillende mobiele platforms moeten deze flexibel worden ontworpen. Uiteindelijk is REpresentational State Transfer (RESTful) (4) web verbinding gekozen voor de data uitwisseling tussen de mobile app en de server applicaties. Een RESTful verbinding is platform-onafhankelijk, omdat deze verbinding gebruik maakt van het HTTP protocol. Op deze manier hoeven clients alleen maar JSON strings te kunnen inlezen.

4.2.1 Webservice

Dit is het centrale verbindingsknooppunt van het hele systeem. Alle data uitwisselingen met de mobile app worden behandeld door deze webservice. Verder heeft de webservice toegang tot de database, en maakt de System Administration tool gebruik van de service. Het is ook mogelijk om verbinding te maken met andere bestaande Tam Tam webservices, zoals de Hourbooking Service. Figuur 4-3 en Tabel 4-1 geven een goed overzicht van de verbindingen van webservice.

De API's van deze webservice zijn verdeeld in drie categorieën.

- **Administratieve API's**: deze API's beheren de database van het framework. Hiermee kunnen bijvoorbeeld nieuwe gebruikers worden toegevoegd, gebruikersinstellingen worden gewijzigd, of sub-apps worden geregistreerd.
- **Wrapper API's**: deze API's zijn wrappers van de API's van de bestaande Tam Tam webservices, zodat de mobile app deze functionaliteit indirect kan gebruiken, bijvoorbeeld uren boeken via de Hourbooking Service.
- **Push Notificatie**: deze API's kunnen gebruikt worden om push notifications te verzenden naar de Google/ Apple Server.



Figuur 4-3: De verbindingen van de webservice

De entiteiten die in verbinding staan met de webservice zijn als volgt:

Verbinding	**Beschrijving**
Mobile app	Alle data uitwisselingen tussen de mobile app (inclusief sub-apps) en de server worden behandeld door de webservice. De mobile app kan de RESTful web API hiervan gebruiken, zodat de internetverbinding simpel en light-weighted blijft.
Database	De database bevat accountgegevens van de gebruikers en andere systeem gegevens. De database is niet direct toegankelijk, alleen via de webservice. Op deze manier wordt voorkomen dat er ongeldige data in de database wordt gezet. De database verbindt zich alleen met de webservice, waardoor alle data modificatie uitsluitend wordt behandeld door de webservice.
System Administration	De System Administration tool biedt een eenvoudige gebruikersinterface voor het weergeven van de opgeslagen data in de database. De website maakt gebruikt van de API's van de webservice voor het lezen en het schrijven van data, en het versturen van push notifications.
Andere Tam Tam webservices	Sommige bestaande web applicaties maken al gebruik van hun eigen webservice. Als het nodig is om deze applicaties te porten naar een mobiele versie, kunnen deze webservices hergebruikt worden. Onze eigen webservice maakt verbinding met deze webservices, zodat de mobile app indirect de andere webservices ook kan gebruiken.
Andere Tam Tam databases	De webservice kan ook verbinding maken met andere databases van Tam Tam, indien hier behoefte aan is.

Notificatiediensten	Als een applicatie een verbinding heeft gemaakt met de webservice, kan de applicatie via de webservice notificaties verzenden naar de servers van Google/Apple. De servers van Google/Apple verbinden zich met de telefoons van de gebruikers, waardoor gebruikers push notificaties kunnen ontvangen.

Tabel 4-1: De verbindingen van de webservice

4.2.2 Database

De database bevat accountgegevens van de gebruikers en andere systeemgegevens. Hieronder staat een overzicht van de database tabellen.

Data	Beschrijving
Application	Alle informatie die de mobile app nodig heeft over de sub-apps.
AuthenticationToken	Tokens voor het gebruik van de notification service van Google. Deze tokens zijn nodig voor authenticatie.
GoogleNotification	Registration ID's en Device ID's van de gebruikerstoestellen die zijn geregistreerd voor Android push notifications.
Subscription	Relationele tabel waarin staat welke gebruiker geabonneerd is op welke sub-apps.
User	Gegevens van alle gebruikers van de mobile app.

Tabel 4-2: Tabellen in de database

4.2.3 System Adminstration

De System Administration tool biedt een eenvoudige gebruikersinterface aan voor het weergeven van de opgeslagen data in de database. Het is meestal niet nodig om data handmatig te veranderen via de website. Op dit moment is het alleen maar nodig voor het toevoegen van nieuwe sub-apps. Hiernaast kan deze website gebruikt worden om push notifications te sturen naar geregistreerde gebruikers.

De System Adminstration toolbiedt alleen de gebruikersinterface aan. Alle onderliggende functionaliteit wordt uitgevoerd door de API's van de webservice.

Functionaliteit	Beschrijving
Gebruikersgegevens weergeven	De persoonlijke gegevens van de gebruikers: voornamen, achternamen, gebruikersnaam, etc.
Applicatiegegevens weergeven	Gedetailleerde beschrijving van de sub-apps.
Abonnementsgegevens weergeven	Welke gebruiker geabonneerd is op welke sub-apps, onderverdeeld in interne en externe sub-apps.
Push notification verzenden	Notifications sturen naar de opgegeven gebruikersnaam. Alle telefoons waar deze gebruiker is geregistreerd, ontvangen het bericht.

Tabel 4-3: De functionaliteit van System Administration tool

5 Implementatie – Android

Met de implementatie van de Android-app zijn we al vrij snel begonnen. Na een ontwerp hebben we code uit een aantal tutorials samengevoegd, met de eerste demoversie als resultaat. Door het dynamische karakter van Scrum is het ook af en toe nodig geweest om het framework iets te wijzigen, maar dit bleef eigenlijk altijd bij refactoren van de code.

5.1 Basisstructuur

De manier waarop het Android framework is opgebouwd heeft veel invloed gehad op de implementatie van de mobile app. Voor meer informatie over het Android framework, zie bijlage E.

In ons geval is het loginscherm het eerste dat de gebruiker ziet, dus de bijbehorende *LoginActivity* hebben we gekozen als launch-activity. Vanuit het loginscherm wordt vervolgens het menu geladen, gerepresenteerd als *MenuActivity*, waar een lijst met sub-apps wordt weergegeven en een link naar de settings. Elke sub-app heeft een eigen activity, en tenslotte is er nog een *SettingsActivity* voor de instellingen. Android heeft een standaardklasse voor de settings, *PreferenceActivity*, maar we kwamen hier pas later achter en dus hadden we niet meer genoeg tijd om de bestaande klasse om te schrijven.

Aanvankelijk hadden we gekozen om de navigatie door menu's te doen door middel van Fragments. Dit zijn een soort containers, waarin andere UI elementen geplaatst kunnen worden, vergelijkbaar met Panels in Java. Elke menu view had zijn eigen Fragment en bij het navigeren wisselden de Fragments elkaar af binnen dezelfde Activity. Dit is later echter veranderd, omdat er nu nog maar twee verschillende menu's zijn (main en settings), die dusdanig van elkaar verschillen dat het delen van dezelfde Activity een onnodig complexe klasse zou opleveren. Hiervoor was er een main, settings en apps menu, die vrij veel op elkaar leken.

5.2 User Interface

Vrijwel alle user interface elementen in Android worden gedefinieerd in een XML-bestand. Deze bestanden kunnen vervolgens door een Activity worden ingelezen en op het scherm worden getoond. Het is ook mogelijk om elementen in programmacode te definiëren, maar dat hebben wij niet gedaan, omdat het laden van de XML-bestanden van tevoren een snellere responsietijd oplevert. Ook is het in XML overzichtelijker en makkelijker te ontwerpen door de ingebouwde drag-and-drop editor van Android.

5.3 Singletons

De back-end van de app bestaat voornamelijk uit de verbinding en communicatie met de server. Dit is namelijk voor vrijwel al het achtergrondwerk benodigd.

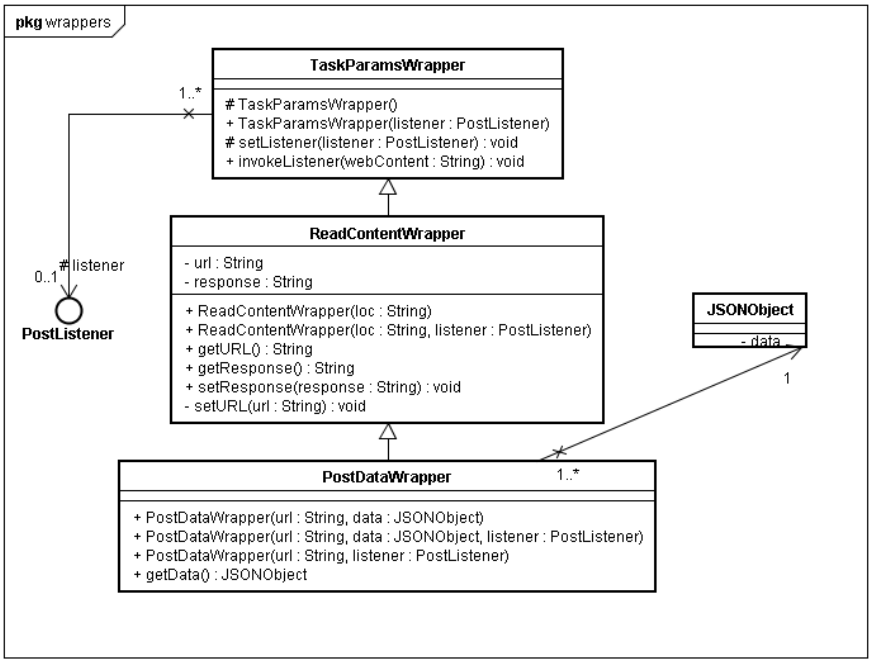
De basis voor veel van de back-end bestaat uit de centrale klassen beschreven in het ontwerp. Aanvankelijk hadden we gekozen om hiervoor de Android klasse Service te gebruiken. Dit lijkt op een Activity, maar heeft geen user interface en kan in de achtergrond actief blijven, zonder dat de app zelf is gestart. Het probleem hiermee bleek echter dat een Service eerst gestart en verbonden moet worden voordat een Activity gebruik kan maken van de Service. Dit gebeurt asynchroon, waardoor bijvoorbeeld bij het inloggen op problemen werd gestuit doordat de service al werd aangeroepen voordat deze hier klaar voor was.

De conclusie was dat het sneller en gemakkelijker zou zijn om hier helemaal zelf een systeem voor te maken. Dit is uiteindelijk geïmplementeerd in de vorm van singletons, omdat deze op elk moment direct beschikbaar zijn. In de singletons is functionaliteit opgenomen die meerdere onderdelen van de app nodig hebben, om zo redundante code te vermijden. Er zijn uiteindelijk drie verschillende singletons ontstaan; een voor netwerkverbindingen en achtergrondprocessen, een voor utility functies, en een voor het weergeven van alerts. Deze zijn allen subklassen van *BaseSingleton*, die een referentie naar het Application-object (een object dat de gehele app voorstelt) heeft om zo toegang te krijgen tot verschillende Android-onderdelen, zoals de *SharedPreferences* waar data kan worden opgeslagen in de telefoon. Ten slotte is er nog een extra *EntrySingleton* die een referentie bevat naar de andere singletons om de toegang eenvoudiger te maken.

NetworkSingleton en wrappers

De NetworkSingleton is de singleton met de meeste functionaliteit, en wordt het meest gebruikt. De klasse bevat de gebruikersnaam en wachtwoord, en maakt gebruik van een *HttpClient* om verzoeken naar de server te sturen. Daarnaast is er een inner class *ServiceTask*. Deze klasse behandelt alle netwerkverbindingen als subklasse van de Android klasse *AsyncTask*.

Een AsyncTask kan code uitvoeren in een ander thread (doInBackground), wat noodzakelijk is bij netwerkoperaties, omdat anders de gehele app zou vastlopen gedurende de operaties. Er kan ook code worden uitgevoerd zodra het achtergrondwerk is voltooid (onPostExecute). De argumenten die worden doorgegeven aan een AsyncTask zijn implementatieafhankelijk, en wij hebben hiervoor een set dataklassen gemaakt waarmee de ServiceTask voor meerdere doeleinden kan worden gebruikt, en bovendien gemakkelijk worden uitgebreid voor toekomstige doeleinden. Deze data klassen kunnen als volgt worden weergegeven:



Figuur 5-1: Overzicht van de wrappers

Een wrapper kan een *PostListener* bevatten, dit is een interface met één methode, onPostExecute(). Deze listener bevat de code die uitgevoerd dient te worden na afloop van het achtergrondwerk, dus in onPostExecute() van de ServiceTask wordt deze listener aangeroepen.

De TaskParamsWrapper wordt gebruikt als het achtergrondwerk geen netwerkoperaties bevat, maar ook niet op de UI-thread uitgevoerd mag worden vanwege langere verwerkingstijd. Op dit moment hebben we hiervoor nog geen toepassing gevonden in de app zelf, maar voor de volledigheid is het geïmplementeerd. De PostDataWrapper dient voor het versturen van POST-verzoeken naar een webserver. De data wordt als JSON-object meegegeven, en in ServiceTask omgezet naar een geldig POST-formaat. Na afloop wordt het antwoord van de server ook opgeslagen in de wrapper, om te gebruiken in onPostExecute(). De ReadContentWrapper wordt gebruikt om GET-verzoeken te sturen, die dus uit alleen een URL bestaan. De inhoud van de webpagina wordt vervolgens weer in de wrapper gezet.

De servicetask neemt als parameter één of meerdere TaskParamsWrappers (dus alle drie de klassen), die vervolgens in de gegeven volgorde worden uitgevoerd. Dit is noodzakelijk omdat soms bepaalde acties pas kunnen worden uitgevoerd nadat andere acties al zijn voltooid, bijvoorbeeld het synchroniseren met de server moet klaar zijn voordat het menu met sub-apps kan worden geladen.

Voor het maken van een verbinding met de gegeven URL bij Post/Get wordt de HttpClient gebruikt. De HttpClient bevat ook de gebruikersnaam en wachtwoord, verkregen uit de NetworkSingleton.

Ten slotte is er binnen NetworkSingleton nog een toepassing van de ServiceTask. In syncSubscriptions() wordt een viertal PostDataWrappers gemaakt, waarmee de naam van de ingelogde gebruiker, alle interne en externe applicaties, en de subscriptions (de te weergeven sub-apps) bij de server worden opgevraagd. Indien mogelijk kan er hierna nog een andere taak worden uitgevoerd, dit om na het succesvol inloggen verder te gaan naar het menuscherm, dit mag namelijk pas gebeuren nadat alles volledig is gesynchroniseerd.

UtilSingleton

Zoals de naam al zegt, bevat deze klasse utility functies, die binnen de app kunnen worden uitgevoerd. De klasse maakt veelvuldig gebruik van de referentie naar het Application-object uit BaseSingleton om toegang te krijgen tot Android-functionaliteit.

Als attributen zijn er de lijst met applicaties, de subscriptions, object voor de ingelogde gebruiker, en een boolean die aangeeft of er iemand is ingelogd. De eerste drie waarden worden verkregen door een sync met de server uit NetworkSingleton, en zijn belangrijk voor de weergave van het hoofdmenu. De boolean isLoggedIn wordt gebruikt bij het openen van het loginscherm. Als deze op *true* staat, is er al ingelogd, en is de state van de app nog bewaard gebleven. In dat geval hoeft er geen sync plaats te vinden en kan direct worden doorgegaan naar het menu.

Behalve de attributen bevat de klasse een aantal belangrijke methodes. Ten eerste hasInternetConnection(), waar de naam voor zichzelf spreekt, is overal cruciaal aangezien vrijwel alle delen van de app een actieve internetverbinding vereisen. Als dit niet het geval is, wordt de gebruiker automatisch uitgelogd, en wordt er geen poging tot inloggen meer gedaan. Daarnaast zijn er methodes om een string met de URL van de server te krijgen, of een string met de inhoud van de titelbalk, waarin de naam van de huidige gebruiker staat. Ten slotte is hier de methode om alle persoonlijke data van de app te wissen, wat gebruikt wordt om uit te loggen.

AlertSingleton

Soms is het nodig om feedback te geven aan de gebruiker die duidelijk naar voren komt. Dit kan in Android op meerdere manieren. Wij hebben gebruik gemaakt van *AlertDialog, Toast, Notification* en *ProgressDialog*. Een Toast (kort berichtje op het scherm dat vanzelf weer verdwijnt) kan direct worden aangemaakt via een statische methode, dus daar hoefde verder niets aan gedaan te worden. De andere alerts kosten meer werk om te weergeven, dus hebben we besloten om hiervoor methodes te schrijven die alle benodigde aanroepen doen, en deze samen te voegen in AlertSingleton.

De AlertDialog is een bericht dat op het scherm verschijnt, net als een Toast, maar deze heeft een eigen venster, dat niet uit zichzelf weggaat. De dialog bevat ook één of twee knoppen (Ok en Cancel) zodat de gebruiker kan reageren. Verder kan er een Listener worden verbonden, die wordt aangeroepen zodra een van de knoppen is ingedrukt. Om een AlertDialog via Android aanroepen te maken, is het nodig om voor elke eigenschap een aparte methode aan te roepen, zoals de titel, tekst, of listener. Onze methode neemt in één keer alle argumenten en roept vervolgens de benodigde methodes aan. De AlertDialog wordt voornamelijk gebruikt om een bevestiging van de gebruiker te vragen (uitloggen), of om zeker te weten dat de gebruiker de tekst heeft gezien (verkeerde inlog gegevens).

Een ProgressDialog wordt gebruikt wanneer er een taak moet worden uitgevoerd die te lang duurt om de gebruiker hiervan niet op de hoogte te stellen, terwijl het resultaat van die taak belangrijk is om verder te gaan. Hierbij kan worden gedacht aan inloggen, het duurt een paar seconden voordat de bevestiging van de server komt dat de inloggegevens geldig zijn. Gedurende deze taak wordt dan een ProgressDialog weergegeven zodat de gebruiker weet dat de applicatie niet is vastgelopen. Ook hier zijn veel methode-aanroepen nodig, dus hebben we een methode die alle argumenten in een keer aanneemt en doorgeeft. Er kan een listener worden doorgegeven, waardoor de dialoog een cancel-knop krijgt waardoor de listener wordt aangeroepen, dit om bijvoorbeeld het inlogverzoek te annuleren.

Ten slotte zijn er nog Notifications, een handige vorm van alerts omdat het bericht de gebruiker niet onderbreekt, maar wel blijvend is tot de gebruiker het afsluit. Notifications worden in de Action Bar buiten de app weergegeven (in het OS zelf) en kunnen bovendien een Intent met zich meedragen, die wordt uitgevoerd zodra er op de Notification wordt geklikt. Opnieuw worden alle argumenten doorgegeven aan één methode.

5.4 Push Notifications

Voor de implementatie van Push notifications bleek C2DM veruit de beste manier. C2DM gebruikt de bestaande verbinding met de Android market (Google Play) om berichten te ontvangen en is dus zeer energie efficiënt. Er zijn drie partijen in betrokken: de app op de telefoon, onze eigen server, en de C2DM server van Google.

Ten eerste moet de app registreren bij Google. Dit gebeurt met een Intent naar een bepaald adres, en is vrij simpel te doen. De app moet echter ook klaar zijn om berichten van Google te ontvangen. Hiervoor moet een *BroadcastReceiver* worden gebruikt. Voor de overzichtelijkheid hebben we twee receivers gemaakt, een voor de registratie en een voor de ontvangen berichten. Deze receivers moeten worden aangegeven in de Android manifest. Wanneer er een bericht/registratiebevestiging wordt verstuurd naar de telefoon, komt hier het in deze klassen aan als Intent, waar het verder wordt afgehandeld.

Bij het ontvangen van een registratiebevestiging sturen wij het Registration ID naar onze server, samen met het unieke Device ID en de gebruikersnaam. Op deze manier kan een bericht direct naar een gebruikersnaam worden verstuurd. Bij het ontvangen van een bericht wordt er automatisch een Notification gemaakt die het bericht weergeeft.

Ten slotte is het nog mogelijk om uit te schrijven van C2DM berichten. Dit gebeurt bij het uitloggen. Door een unregister-Intent te sturen naar de C2DM server zal de server een melding krijgen dat de telefoon zich heeft uitgeschreven, en kan deze hierop reageren.

5.5 Implementatie per requirement

Requirement	**Implementatie**
Inloggen met Tam Tam account	De ingevoerde logingegevens worden doorgestuurd naar de HttpClient in NetworkSingleton. Vervolgens wordt geprobeerd om een -speciaal hiervoor ontworpen- pagina van onze server te bereiken. De server vereist authenticatie, waarop de HttpClient antwoordt met de logingegevens. Als vervolgens de inhoud van de webpagina overeenkomt met de waarde van een vooraf bepaalde string, is het inloggen succesvol.
Gebruikersnaam en wachtwoord opslaan	Bij het succesvol inloggen worden de gebruikersnaam en het wachtwoord opgeslagen in de *SharedPreferences* van Android. Dit is een plek in de telefoon om data op te slaan, die alleen leesbaar is voor de mobile app die de data heeft opgeslagen.
De mobile app wordt beschermd met pincode	Bij de eerste keer inloggen wordt de gebruiker direct gevraagd om een pincode te kiezen. Deze pincode wordt ook opgeslagen in SharedPreferences, waarna een timer begint te lopen. Wanneer deze timer de ingestelde interval waarde heeft bereikt wordt opnieuw de pincode gevraagd. Hierna start de timer weer opnieuw.
Pincode interval kan worden gewijzigd	In de instellingen kan de lengte van het interval worden gekozen. Ook deze waarde wordt in SharedPreferences opgeslagen.
Single-Sign-On ondersteuning voor native en web sub-apps	Native sub-apps maken gebruik van een webservice binnen Tam Tam. Voor het ontvangen van data van de server wordt de HttpClient gebruikt, die de gebruikersnaam en wachtwoord al kent. De gebruiker hoeft dus niet opnieuw in te loggen. Web sub-apps worden gestart in een *WebView*, die de logingegevens uit de NetworkSingleton kan opvragen.
Dynamische menustructuur voor sub-apps	Voordat het menu wordt geladen, wordt een webservice op onze server gevraagd om een lijst van alle bestaande sub-apps, en een lijst van sub-apps die de gebruiker heeft geselecteerd (dit is per gebruiker opgeslagen in de database). Slechts de geselecteerde sub-apps worden weergegeven als menu items.

Native sub-apps toevoegen in de mobile app	Nieuwe native sub-apps kunnen eenvoudig worden toegevoegd door hiervoor een Activity toe te voegen, en vervolgens een entry in de database op onze server te maken voor de sub-app met *isAndroid=true*. In deze entry krijgt de sub-app een ID. In MenuActivity, binnen de methode launchApplication bevindt zich een switch-statement over het ID van de te starten sub-app. In deze switch moet een extra case worden toegevoegd met het ID van de nieuwe sub-app, waarbinnen via een Intent de Activity van de sub-app moet worden gestart.
Web sub-apps toevoegen in de mobile app	In dezelfde database als waar de native sub-apps staan, kunnen web sub-apps worden toegevoegd, door *isWeb=true*. In dezelfde rij kan de URL van de web sub-app worden toegevoegd, waarna de sub-app vanzelf zal verschijnen in de instellingen. De mobile app maakt een sub-app object aan met de *isWeb* boolean en de URL. Bij het starten van de sub-app zal vervolgens een *WebView* worden geopend waarin de URL wordt geladen. Merk op dat voor het toevoegen van een web sub-app dus geen update van de mobile app nodig is.
Sub-apps toevoegen aan of verwijderen uit het menu	In de instellingen kan een lijst worden weergegeven met alle bestaande sub-apps. Hier kan de gebruiker vervolgens selecteren welke sub-apps moeten worden toegevoegd of verwijderd door deze aan- of af te vinken.
De nieuwe sub-apps weergeven in het menu	Elke keer na een synchronisatie wordt een volledige lijst met sub-apps van onze server gehaald, waardoor de nieuwe sub-apps vanzelf verschijnen bij de instellingen. Standaard worden deze niet weergegeven in het hoofdmenu.
Externe mobile apps aanroepen in het applicatiemenu	Met behulp van een Intent kunnen externe mobile apps worden gestart. Het argument voor deze Intent, een string, wordt uit onze database gehaald.
Push Notification toevoegen op een telefoon	Wanneer een gebruiker handmatig inlogt, wordt de mobile app automatisch geregistreerd bij Google. Dit gebeurt met een Intent richting de Google server, zoals eerder beschreven. Vervolgens worden berichten altijd ontvangen, zelfs als de mobile app niet is gestart.
Push Notification verwijderen als gebruiker is uitgelogd	Wanneer een gebruiker in de instellingen voor uitloggen kiest, wordt behalve het wissen van alle persoonlijke data ook een Intent naar Google gestuurd met daarin het verzoek om de telefoon uit te schrijven van verdere berichten. Als een server vervolgens een bericht stuurt naar deze telefoon, wordt deze medegedeeld dat de telefoon is uitgeschreven, en dient de server het registration ID te verwijderen uit de database.

Functioneren bij langzame internetverbinding	Netwerkactiviteiten vinden meestal plaats op een moment dat er pas kan worden doorgegaan als de activiteiten klaar zijn. De gebruiker ziet dan een progress dialog. Bij een langzame internetverbinding zal er dus nog steeds eerst gewacht worden tot alle benodigde informatie binnen is voordat er wordt doorgegaan, en hoewel dit langer zal duren, heeft het verder geen invloed op de mobile app.

Tabel 5-1: Beschrijving van de implementaties van de gehaalde requirements

6 Implementatie – iOS

6.1 iPhone

Het ontwerp van de mobile app voor de iOS versie is hetzelfde voor de Android versie, maar de implementatie is compleet anders. In deze paragraaf worden de technische details uitgelegd en hoe de requirements zijn geïmplementeerd voor iOS 5.0.

6.1.1 Basisstructuur

Controllers – de controller klassen bepalen welke 'View' er op je schermt zichtbaar is. Een controller klasse is gekoppeld aan een XIB file, met daarin alle data voor de View. Een View bestaat dus uit UI elementen en de Controller bepaalt wat er precies gebeurd als er bijvoorbeeld op een UI element wordt gedrukt.

De applicatie bevat de volgende Controllers: RootviewController, LoginViewController, WebViewController en SettingsViewController. Verder is er ook een Pincode Interface.

Als de applicatie opstart dan zet iOSBroekzakAppDelegate.m, een automatisch gegenereerde klasse, de RootViewController als de start voor de navigatie van de mobile app. De RootViewController is dus de start View voor de applicatie. Hier wordt beslist of de gebruiker moet inloggen, een pincode moet invullen of gewoon verder kan gaan met de applicatie. Hier worden voor het eerste gebruik ook alle variabelen gezet, zoals pin interval en alle andere basisinstellingen. Als er een nieuwe controller op de navigatie stack wordt gepushed kan deze nog alleen worden gepopped. Er zijn niet meerdere controllers op elkaar op de navigatie stack.

De LoginViewController handelt alle logica af met betrekking tot het inloggen. Er wordt een UIViewAlert, een soort van popup, naar voren gebracht als de gebruiker verkeerde gegevens heeft ingevoerd. Met behulp van de ServiceSingleton klasse, die bij 'Helpers' wordt uitgelegd, worden de gegevens gecontroleerd.

De WebViewController is gekoppeld aan de View waar de sub-apps worden geladen. In plaats van een eigen activity, zoals bij Android, is hier maar één View. Er wordt alleen een nieuwe URL geladen in de WebView die aangevraagd wordt van de database.

De SettingsViewController en de pincode logica zijn hetzelfde als bij Android.

Helpers – De helper klassen zijn klassen die niet de logica bepalen van de Views, maar wel door bijna elke controller worden gebruikt. In dit geval zijn het de klassen ServiceSingleton en SpinnerView. Bij elke connectie naar onzer server worden deze twee klassen aangeroepen om dit af te handelen.

Externals – zijn klassen die niet zelf zijn geschreven of zijn bewerkt. Deze waren nodig om bepaalde standaarden af te handelen, zoals beveiligingen voor iOS. Voor de encryptie en het veilig opslaan van wachtwoorden en gebruikersnamen gebruiken we de klassen KeychainItemWrapper, geschreven door Apple zelf. Voor het uitlezen van de JSON strings en het omzetten van een dictionary naar een JSON string gebruiken we SBjsonBase, geschreven door Stig Brautaset.

6.1.2 User Interface

User interfaces worden gemaakt door ingebouwde functionaliteiten van Xcode. Er wordt een XIB file aangemaakt, waar je UI elementen kan slepen en hun eigenschappen kan aanpassen. Ook is het mogelijk de UI elementen te koppelen aan bepaalde klassen en variabelen. Als er op een UI element wordt gedrukt, dan wordt er een delegate aangeroepen die je kunt opvangen in de gekoppelde klasse.

6.1.3 Implementatie van de connecties met de server

De ServiceSingleton handelt alle connecties voor de gegeven URL's af. De volgende functie laat je inloggen.

```
-   (void)   login:(id<ConnectionDelegate>)target   username:(NSString*)
username password:(NSString*)password {
    KeychainItemWrapper *keychainItem = [ … ]
    delegate = target;
    NSURLRequest *request = [ … ]
}
```

Deze functie wordt geroepen vanaf de LoginViewController. De LoginViewController geeft de ingevulde data door aan de ServiceSingleton login functie en hier wordt dit veilig opgeslagen in de Keychain voor iPhone. Nu er een connectie is aangevraagd met de server wordt er op een succesvolle connectie de volgende delegate aangeroepen:

```
-(void) connectionDidFinishLoading:(NSURLConnection *)connection
```

In deze methode hebben we gespecificeerd wat er precies op een bepaalde connectie moet gebeuren.

Er zijn meerde connecties mogelijk die elk een verschillende actie moeten hebben. Daarom voegen we de connectie toe aan een dictionary en bij succes, dus in `connectionDidFinishLoading`, vergelijken we dit met de toegevoegde connectie in de dictionary.

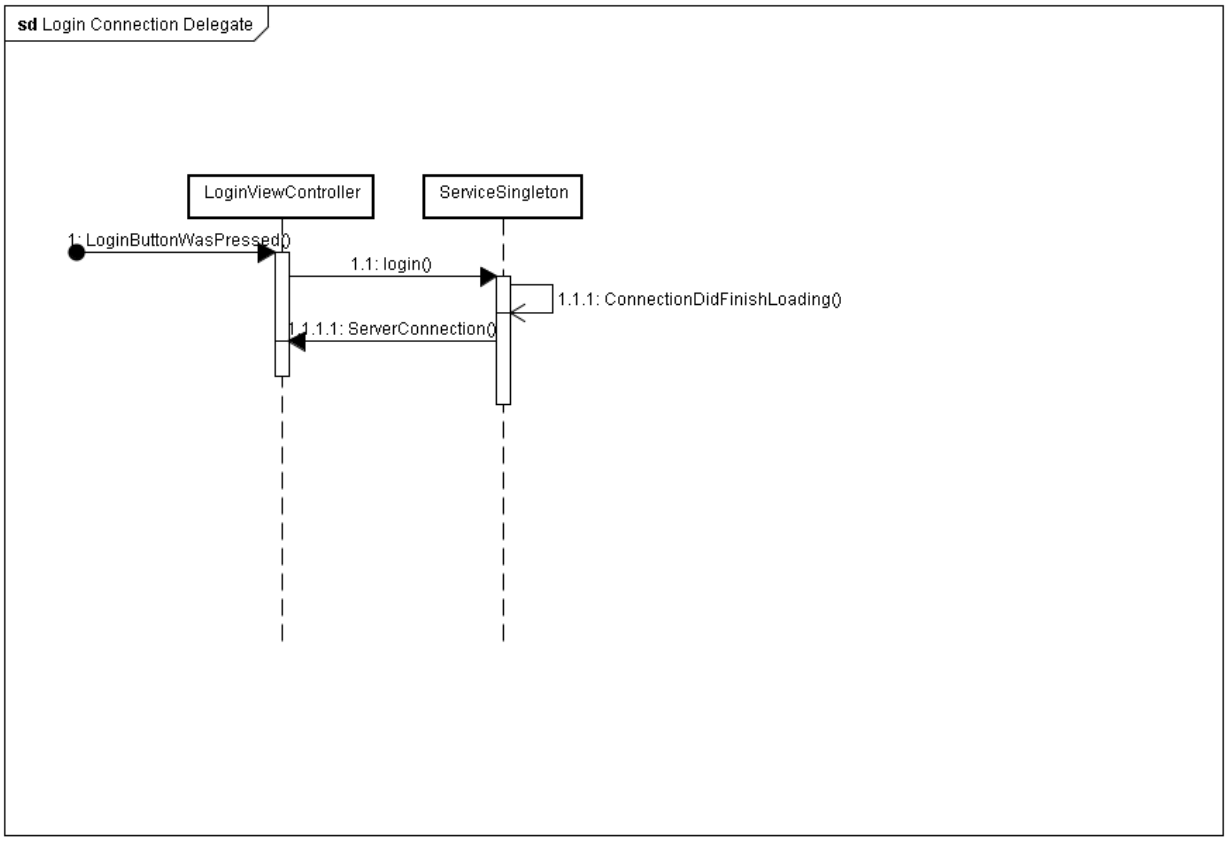
Kijk ook naar het eerste argument van de login functie: "`(id<ConnectionDelegate>)target`".

Dit is een referentie naar de klasse vanwaar de login functie is aangeroepen. Als de `connectionDidFinishLoading` delegate wordt aangeroepen sturen we een bericht terug naar de klasse die een connectie aanvroeg. Dit doen we door middel van onze eigen gedefinieerde delegate:

```
/// Delegate to notify other classes for a successful connection. This
is done asynchronously. This is so that the main thread does not get
cluttered and the application stays responsive.
```

```
@protocol ConnectionDelegate <NSObject>
@required
- (void) serverConnection: (BOOL)success process:(NSString*)
identification;
@end
```

Als de 'Request' klasse een bericht terug wil krijgen moet het de bovenstaande methode implementeren. Uit de argumenten kan de klasse te weten komen welke connectie er werd gevraagd en of het gelukt is of niet.



Figuur 6-1: Sequence Diagram voor het inloggen

6.1.4 Implementatie van de Push Notifications

Voor het implementeren van de Push Notifications moest er eerst een iPhone worden geregistreerd. Zodra dit was gebeurd kan er vanaf onze server een bericht naar de APNS server van Apple worden gestuurd, die dit bericht vervolgens weer doorstuurt naar alle geregistreerde apparaten. In het ontwerp is dit in meer detail besproken.

Met behulp van het Moon-APNS Framework, een programma geschreven in C# voor een .NET server, kan er een bericht worden gestuurd naar de APNS-server van Apple. Wat er nodig is qua informatie is de device token die is verkregen bij het registreren van de iPhone, de certificaten en de payload. De payload bestaat uit het bericht dat verstuurd moet worden en alle opties voor bijvoorbeeld geluid.

6.1.5 Implementatie per requirement

Requirement	Implementatie
Inloggen met Tam Tam account	De ingevoerde logingegevens worden doorgestuurd naar de ServiceSingleton. Vervolgens wordt geprobeerd om een speciaal hiervoor ontworpen pagina van onze server te bereiken. De server vereist authenticatie. Als de pagina geladen kan worden, is het inloggen succesvol.
Gebruikersnaam en wachtwoord opslaan	Dit gebeurt aan de hand van de KeychainitemWrapper klassen. Gebruikersnaam en wachtwoord worden veilig opgelsagen in de Keychain.
De mobile app wordt beschermd met pincode	In de RootViewController wordt er beslist of er een pincode moet worden ingevoerd of niet.
Pincode interval kan worden gewijzigd	In de Settings View kan er worden bepaald hoe lang het interval is. Dit wordt opgeslagen in NSUserDefaults.
Single-Sign-On ondersteuning voor native en web-apps	Aangezien het paswoord en username veilig zijn opgeslagen kunnen deze worden herbruikt.
Dynamische menustructuur voor sub-apps	De structuur voor de sub-apps worden uitgelezen uit de JSON string op de server. Het is dus dynamisch.
Native sub-apps toevoegen in de mobile app	Het framework is uitbreidbaar, dit moet wel zelf door de developers van Tam Tam worden geïmplementeerd.
Web sub-apps toevoegen in de mobile app	Als er nieuwe web sub-apps worden toegevoegd op de server dan worden deze ook automatisch toegevoegd in de mobile app. De data op het beginscherm is afhankelijk van de data op de server.
Sub-apps toevoegen aan of verwijderen uit het menu	In de Settings View is er de mogelijkheid om uit sub-apps te kiezen, die beschikbaar voor de gebruiker zijn. Je kunt hier apps toevoegen of verwijderen.
Bijhouden welke gebruikers zijn geabonneerd op welke sub-apps	Er wordt er een POST request gestuurd naar de server als de gebruiker een abonnement op een sub-app wijzigt.
De nieuwe sub-apps weergeven in het menu	In de Settings View heb je de mogelijkheid om uit sub-apps te kiezen, die beschikbaar voor de gebruiker zijn. Deze worden dan later beschikbaar.
Externe mobile apps aanroepen in het applicatiemenu	Het is mogelijk om onder andere Google Maps aan te roepen vanuit de Settings View.

Push Notification toevoegen op een telefoon	In de instellingen van de iPhone kan de gebruiker aangeven of hij een Push Notification wil ontvangen.
Push Notification verwijderen als gebruiker is uitgelogd	De gebruiker moet dit handmatig doen in de instellingen van de iPhone, of de mobile-app verwijderen. Dit omdat iOS alleen de mogelijkheid bied om te unregisteren. Na 24 uur of langer wordt er dan weer gevraagd of de gebruiker zich wil registreren.
Functioneren bij langzame internetverbinding	Voor elke connectie request wordt er een laadscherm aangeroepen. Het laadscherm verdwijnt als er een connectie delegate wordt aangeroepen.

Tabel 6-1: Requirements voor iOS die aansluiten op de webservices

7 Implementatie - Server

We hebben een virtuele machine toegewezen gekregen voor de ontwikkeling van server-side applicaties. Deze virtuele machine draait op Windows Server 2008 RC2, waarop IIS Server 7.5 en SQL Server 2008 zijn geïnstalleerd.

Het .NET Framework is gekozen als development environment voor alle server-side applicaties. Ten eerste is .NET het meest gebruikte development platform binnen Tam Tam. De server-side applicaties sluiten daardoor beter aan op het bestaande systeem, wat de implementatie minder complex maakt. Bovendien zijn er ook geen extra kosten voor Tam Tam om .NET te gebruiken. Verder zijn er veel .NET developers binnen Tam Tam waardoor het makkelijker zal zijn voor Tam Tam om verder te werken aan het project. Ten slotte is .NET zelf een krachtig applicatieframework, dat veel development toolkits aanbiedt. De implementatie gaat veel makkelijker met behulp van deze toolkits. In bijlage E wordt .NET in meer detail uitgelegd.

7.1 WebService

Windows Communication Foundation (WCF) wordt gebruikt voor de implementatie van de webservice. Met behulp van WCF kunnen API's van de webservice op verschillende manieren worden aangeroepen. Via een RESTful verbinding communiceert de webservice met de mobile app. De RESTful verbinding maakt gebruikt van het HTTP-protocol, namelijk de HTTP-POST methode om JSON objecten uit te wisselen.

Naast RESTful verbinding, kunnen de API's van de WCF webservice ook aangeroepen worden met een Simple Object Access Protocol (SOAP) verbinding. De SOAP verbinding is speciaal bedoeld voor andere .NET server-side applicaties. In bijlage E worden WCF webservice, de RESTful verbinding en de SOAP verbindingen in meer detail uitgelegd.

De WCF webservice kan ook gebruik maken van andere .NET componmenten zoals databases in een SQL server door de database als een ADO.NET Entity Data Model te importeren. In bijlage E wordt ADO.NET Entity Data Model in meer detail uitgelegd.

Omgedraaid, kan deze webservice via een SOAP verbinding andere .NET webservices gebruiken. Een voorbeeld hiervan is Hourbooking Sevice van Tam Tam. De Hourbooking Service is niet direct toegankelijk voor de mobile app. De WCF webservice kan de Hourbooking Service importeren, en wrappers om de methodes bouwen. Op deze manier kunnen mobile apps indirect de API's van de Hourbooking Service gebruiken via de WCF service.

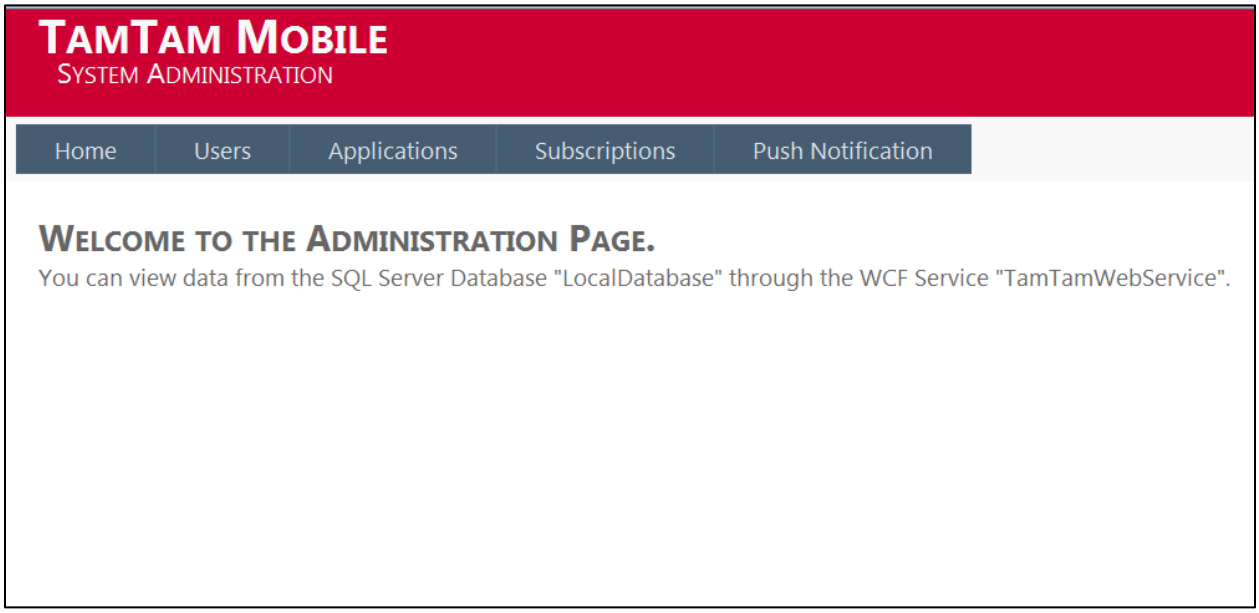
7.2 Database

Gebruikersgegevens, applicatiegegevens en andere systeemgegevens worden opslagen in databases van SQL Server 2008. In vergelijking met andere gratis alternatieven is SQL Servermeer compatibel met WCF. Alle data uitwisselingen vanuit de database worden uitsluitend behandeld door de webservice.

7.3 System Administration

System Administration is geïmplementeerd als een ASP.NET 4.0 website. Deze website biedt alleen de gebruikersinterface aan. Alle onderliggende functionaliteit wordt uitgevoerd door de API's van de webservice.

De website bevat een menu-boject in het toppanel. De menu-items "Users", "Applications", "Subscriptions" en "Push Notification" verbinden zich met een webpagina. In deze webpagina's wordeen de fucntionaliteiten besproken in paragraaf 4.2.3 geimplementeerd. Hieronder is een screenshot van de System Administration tool.



Figuur 7-1 Screenshot van Systeem Administration tool

8 Codebeoordeling SIG

We hebben voor de eerste codebeoordeling van SIG de code van het Android framework en de code van de server opgestuurd. Het iOS framework was op dat moment nog in een te vroeg stadium om mee te zenden.

8.1 Aanbevelingen SIG

De code van het systeem scoort vier sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code bovengemiddeld onderhoudbaar is. De hoogste score is niet behaald door een lagere score voor Component Independence en Duplicatie.

Voor Component Independence wordt er gekeken naar de hoeveelheid code alleen binnen een component wordt gebruikt, oftewel de hoeveelheid code die niet aangeroepen wordt vanuit andere componenten. Hoe hoger het percentage code welke vanuit andere componenten wordt aangeroepen, des te groter de kans dat aanpassingen in een component propageren naar andere componenten, wat invloed kan hebben op toekomstige productiviteit. Wat in dit geval opvalt is de vrij forse class 'ServiceSingleton' die in meerdere componenten wordt gebruikt. Deze class implementeert utility functies zoals 'hasInternetConnection', maar bevat ook functionaliteit voor authenticatie en het tonen van alerts. Het opsplitsen van deze class in classes met een specifiekere functionaliteit zorgt ervoor dat deze aparte stukken functionaliteit makkelijker te begrijpen, te testen en daardoor te onderhouden worden.

Voor Duplicatie wordt er gekeken naar het percentage van de code welke redundant is, oftewel de code die meerdere keren in het systeem voorkomt en in principe verwijderd zou kunnen worden. Vanuit het oogpunt van onderhoudbaarheid is het wenselijk om een laag percentage redundantie te hebben omdat aanpassingen aan deze stukken code doorgaans op meerdere plaatsen moet gebeuren. In dit systeem is er relatief veel duplicatie te vinden tussen bijvoorbeeld de 'CRMActivity' en de 'HourBookingActivity' classen, maar ook tussen 'AppListAdapter' en 'CheckableAppListAdapter'. Het is aan te raden dit soort duplicatie op te sporen en dit waar mogelijk te verwijderen.

Over het algemeen scoort de code bovengemiddeld, hopelijk lukt het om dit niveau te behouden tijdens de rest van de ontwikkelfase. Als laatste nog de opmerking dat er geen (unit)test-code is gevonden in de code-upload. Het is sterk aan te raden om in ieder geval voor de belangrijkste delen van de functionaliteit automatische tests gedefinieerd te hebben om ervoor te zorgen dat eventuele aanpassingen niet voor ongewenst gedrag zorgen.

8.2 Effect van de aanbevelingen

De hoeveelheid werk die benodigd was om al het commentaar te verwerken was niet al te groot. Daarnaast konden wij ons prima vinden in de genoemde punten.

We hebben dan ook de ServiceSingleton in drie verschillende klassen gesplitst, zoals beschreven in hoofdstuk 5. Het geheel bleek hierdoor inderdaad een stuk overzichtelijker. Verder was het niet nodig om de duplicatie in *CRMActivity* en *HourBookingActivity* te verwijderen, omdat deze klassen niet bij het uiteindelijke product zullen horen en waren gemaakt om het aansluiten van nieuwe sub-apps te testen.

De duplicatie tussen *AppListAdapter* en *CheckableAppListAdapter* is echter wel overbodig. Het verschil zit alleen in de methode *getView()*. Nu is *AppListAdapter* de superclass van *CheckableAppListAdapter* geworden, en CheckableApplistAdapter override de getView() methode.

9 Testen

9.1 Testen voor Android

Voor het testen van de Android mobile app hebben we gebruik gemaakt van het geïntegreerde testing framework. Dit framework verzorgt de afhankelijkheid van de Android klassen binnen de mobile app.We hebben unit tests geschreven voor de belangrijkste klassen, en exploratory testing toegepast op zowel de emulator als een fysiek Android device. Acceptance testing is niet uitgevoerd omdat het eindproduct niet direct in productie zal worden genomen.

9.1.1 Unit tests

De unit tests zijn vooral bedoeld om klassen apart in een geïsoleerde omgeving te testen. Dit is van toepassing op zelfgemaakte niet-triviale klassen die geen gebruik maken van Android componenten, dus de klassen uit het model package. Er is getest op het correct inlezen van JSON strings, en het wijzigen van de lijst van sub-apps in het hoofdmenu. Dit wordt deels gedaan door het testen van de klassen zelf, en deels door het testen van de UtilSingleton. AlertSingleton wordt niet getest omdat dit slechts een wrapper is voor Android methodes, die geacht worden te werken.

Het totaal aantal tests hier is vrij laag, dit omdat een groot deel van de functionaliteit binnen Android het doorgeven van data is. Verder zijn er veel triviale dataklassen. De overige functionaliteit bestaat voornamelijk uit netwerkactiviteiten, waarvoor dus ook geen unit tests voor kunnen worden ontworpen. Hiervoor passen we exploratory testing toe.

9.1.2 Monkey testing

Een applicatie van het Android testing framework is de monkey applicatie. Deze voert willekeurig acties uit binnen de mobile app, zoals het drukken op plaatsen op het scherm, of specifieke knoppen. Een serie van acties kan later weer opnieuw worden afgespeeld door middel van een seed.

Op deze manier kan de mobile app worden getest op robuustheid.

9.1.3 Exploratory testing

Veel functionaliteit is lastig te testen met unit tests. Hierbij kan worden gedacht aan of er wel de correcte dialogen worden weergegeven, of de juiste menu's worden geladen, en navigatie door de mobile app in het algemeen. In de volgende tabel is het test script voor exploratory testing verder toegelicht.

Situatie	Actie	Verwacht resultaat
Loginscherm na het starten van de mobile app	• Druk op "Log in" zonder iets in te vullen	Er verschijnt een foutmelding die zegt dat er geen gegevens zijn ingevoerd.

	• Voer een ongeldige gebruikersnaam en/of wachtwoord in, maar zorg dat beide velden zijn ingevuld • Druk op “Log in”	Er verschijnt een foutmelding die zegt dat de logingegevens niet kloppen.
	• Voer geldige logingegevens in • Druk op “Log in”	Er verschijnt een progressdialog waarin staat dat gegevens worden gesynchroniseerd met de server, waarna wordt gevraagd om een pincode te kiezen.
	• Voer logingegevens in • Druk op “Log in” • Druk in de progressdialog die volgt op “Cancel”	Het inloggen is afgebroken en het loginscherm blijft zichtbaar.
	• Zet de internetverbinding uit • Voer logingegevens in • Druk op “Log in”	Er verschijnt een melding dat er geen internetverbinding is.
Scherm waarin gevraagd wordt een pincode te kiezen	• Voer een pincode van minder dan 4 cijfers in en druk op “OK”	Er verschijnt een melding dat de pincode 4 cijfers lang moet zijn.
	• Voer een pincode van 4 cijfers in en druk op “OK”	Er verschijnt een melding dat de pincode met succes is ingesteld.
Hoofdmenu	• Druk op “Settings” • Druk op “Log out”	Er wordt teruggekeerd naar het loginscherm
Loginscherm na uitloggen	• Druk op de back-toets van de telefoon	De mobile app wordt afgesloten in plaats van terug te gaan naar het settings menu
Hoofdmenu	• Druk op “Settings” • Druk op “Manage applications” • Selecteer een aantal sub-apps (placeholders) • Ga terug naar het hoofdmenu	De geselecteerde sub-apps staan nu in het hoofdmenu
	• Druk op “Settings” • Druk op “Manage applications” • Verwijder één of meerdere sub-apps door ze af te vinken • Ga terug naar het hoofdmenu	De afgevinkte sub-apps staan niet meer in het hoofdmenu

	• Druk op “Settings” • Druk op “Manage applications” • Selecteer een aantal externe sub-apps • Ga terug naar het hoofdmenu • Druk op een externe sub-app	De geselecteerde externe app wordt uitgevoerd. Indien deze niet was geïnstalleerd, wordt de pagina in Google Play van deze app geopend, zodat de app gedownload kan worden.

Tabel 9-1: Exploratory testing voor het Android framework

9.2 Testen voor iPhone

Het testen van het framework gebeurt op de volgende manieren: testklassen, testen in de emulator en testen op een fysiek device.

9.2.1 Test Klassen

Er zijn Unit Tests gemaakt om te kijken of JSON strings goed worden uitgelezen en of de PIN logica naar behoren werkt. De rest van de functionaliteit is getest door exploratory testing in de emulator, en op een iPhone 3GS en 4GS.

9.2.2 Testen in de emulator

Xcode heeft de mogelijkheid om te debuggen in de editor. Bij elke run kan er een breakpoint worden gezet om te kijken waar de fouten zijn. Sommige foutmeldingen zoals BAD_EXCESS en SIGBART zijn af en toe onduidelijk. Meestal komt het, omdat er geen ‘@’ voor een string staat. Helaas kan het ook wat anders zijn en moet er wat meer onderzoek gedaan worden.

In vergelijking met de Android emulator werkt de iPhone emulator erg soepel en snel. De emulator is meestal binnen 10 seconden opgestart. Dit kan wel af en toe vastlopen, maar het hindert niet erg. Ook kan gemakkelijk worden gewisseld tussen verschillende versies (4.3, 5.0 etc.) als dit nodig is.

9.2.3 Testen op een fysiek device

Het testen op een fysiek device leverde wat problemen op. Omdat we niet de laatste versie van Mac OS X hadden, konden de laatste versies van de development tools en SDK’s niet worden geïnstalleerd op de MacBook. Dit betekende dat we de SDK voor iOS 5.0 moesten gebruiken. Apple is heel erg strikt met haar versie beheer en wil dat iedereen update naar de laatste OS voor zowel Mac als iPhone. Het is niet toegestaan om terug te gaan naar een oudere versie. Dit is begrijpelijk om fragmentatie tussen de versies te voorkomen, maar hierdoor werd het wel lastig om het framework op een echt device te testen. De enige iPhone die we tot onze beschikking hadden had de laatste versie OS 5.1.1. Hiermee was het niet mogelijk om de gemaakte mobile app te deployen op de iPhone. Het was wel mogelijk om de mobile app te exporten naar een IPA file en deze te installeren op een gejailbreakte iPhone. Later hadden we onze eigen iPhone gejailbreakt en de mobile app hierop getest. Alles werkte naar behoren.

9.3 Webservice

Voor het testen van de webservice worden unit tests en exploratory tests uitgevoerd.

9.3.1 Unit tests

Om de unit tests te maken voor de webservice wordt Microsoft Visual Studio gebruikt. De functie “Create Unit Tests” genereert automatisch de testclasses en testmethoden. Vervolgens hoeven alleen de testmethodes worden aangepast. Vrijwel alle methodes hebben betrekking op het lezen van of het schrijven naar de database. Er wordt vooral getest op het correct inlezen en schrijven van data.

9.3.2 Exploratory testing

De tester moet ook zorgen voor betrouwbare datauitwisseling tussen de mobile app en de webservice. Om te correctheid van de datauitwisseling te testen, gebruiken wij Fiddler Web Debugger. Meer informatie over Fiddler is in bijlage E.

Elke methode van de webservice heeft een unieke URL. De mobile app kan deze methoden aanroepen via HTTP POST. Vervolgens wordt een JSON string als response gegeven. Met Fiddler kan HTTP POST gemakkelijk worden gesimuleerd. Om een methode te testen worden HTTP POST’s handmatig gemaakt in Fiddler en verstuurd naar de webservice. De tester controleert of de juiste data is binnengekomen bij de webservice en de juiste JSON Objecten worden teruggestuurd. Alle methoden van de webservice worden op deze manier getest.

10 Conclusies en aanbevelingen

10.1 Conclusie

Het eindproduct variëert vrij veel van wat we oorspronkelijk hadden gepland na de oriëntatiefase. Het blijkt dat de Scrum ontwikkelmethode inderdaad de juiste aanpak was voor dit project. Via de tweewekelijkse sprint meeting zijn de begeleiders van Tam Tam steeds op de hoogte gesteld van het ontwikkelingsproces.

Het ontwerp en de implementatie van de componenten zijn niet vastgelegd in het begin maar zijn steeds aangepast na elke sprint. Buiten ons development team hebben we hulp gekregen van de begeleiders, het advies van professionele developers van Tam Tam en de codebeoordeling van SIG. Hierdoor konden we kritisch kijken naar ons ontwikkelingsproces en het framework verbeteren in de loop van de tijd.

Het doel van het project was uiteindelijk om een goed framework af te krijgen, in plaats van een half functioneel framework met een sub-app die ook nog niet goed werkt. Op deze manier kan er makkelijker worden verder gewerkt aan het project in de toekomst. Voor de mobile app is het gelukt om een werkend framework te leveren in beide Android en iOS platform. De mobile app bevat een dynamisch menu voor het laden van zowel native als web sub-apps. De mobile app kan data uitwisselen met de server. De gebruikers kunnen inloggen met hun Tam Tam account, en hoeven maar één keer in te loggen (Single-Sign-On). Ook kan de mobile app push notifications ontvangen.

Bij de webservice hebben we bij de volgende onderdelen het meeste succes geboekt: het kunnen communiceren van onze eigen aangewezen server met de andere servers van Tam Tam, HTTPS verbindingen en een API om Push Notifications te sturen naar Android en iPhone. Ook de webservice kan vrij gemakkelijk worden uitgebreid.

Onze begeleider bij Tam Tam vond het project geslaagd als er aan het einde van het project een werkend framework stond, en dit is ook gelukt. Zelf wilden we graag nog ten minste één mobiele web applicatie afhebben, maar met drie man is dit toch iets te veel gebleken. Met een groep van vier, of zonder de requirement van een framework voor twee platformen, zou dit wel mogelijk zijn geweest.

Als er een sub-app wordt ontwikkeld, kan deze worden toegevoegd aan de mobile app, waarna deze in principe in gebruik kan worden genomen. In de volgende sectie worden aanbevelingen gedaan met betrekking tot toekomstige uitbreiding.

10.2 Aanbevelingen

Gezien de scope van het project, waar we alleen het framework hebben gemaakt, kunnen er nog geen harde conclusies worden getrokken qua bruikbaarheid van de app. Wel is het framework af en is het mogelijk om een aantal aanbevelingen te doen.

Het is mogelijk om het framework uit te breiden met zowel native als web sub-apps. Deze hebben beide hun voor- en nadelen, dus het ligt aan de situatie welke keuze het beste is. Native sub-apps zullen apart moeten worden ontwikkeld voor elk platform, maar het native gedeelte zal voornamelijk uit een user interface bestaan. Het meeste achtergrondwerk kan worden gedelegeerd naar de server. Web sub-apps hoeven maar één keer ontwikkeld te worden, en kunnen op dezelfde server als de webservice gehost worden. Hierdoor kunnen ze direct de methodes van de webservice gebruiken, zonder een RESTful verbinding te hoeven maken. In de oriëntatiefase is echter gebleken dat mobiele webapps minder responsief zijn dan native apps, en vaak traag laden. De keuze voor native of web ligt aan hoe zwaar de voor- en nadelen meetellen. Het is aan te bevelen een experimentele sub-app eerst als webapp te ontwikkelen, en als deze goed blijkt te werken of populair is, de overstap te maken naar native.

Momenteel staat er nog een aantal placeholder sub-apps in het project, om te verduidelijken hoe het toevoegen hiervan werkt. Dit zijn de klassen CRMActivity en HourBookingActivity. C2DMActivity is een placeholder klasse die het registreren bij C2DM verduidelijkt.

De push notifications bestaan nu uit slechts berichten die in een notification op de telefoon worden weergegeven, maar het is mogelijk om dit uit te breiden voor meer functionaliteit. Zo kunnen herinneringen voor het boeken van uren bijvoorbeeld worden gelinkt aan een hourbooking sub-app, zodat de gebruiker via de notification direct de uren kan boeken zonder de mobile app te hoeven openen.

Voor het testen van een SSL-verbinding is de HttpClient in NetworkSingleton zo ingesteld dat alle certificaten worden geaccepteerd. Voor een productieversie moet dit uiteraard worden teruggedraaid, dit is eenvoudig te doen door het *EasySSLSocketFactory* object in de het https scheme te vervangen door *SSLSocketFactory.getSocketFactory()*, op dezelfde manier waarop de *PlainSocketFactory* wordt ingevuld.

Verder is gebleken dat vlak nadat we C2DM hadden geïmplementeerd, dit systeem is opgevolgd door Google Cloud Messaging for Android (GCM) (5). C2DM kan nog steeds worden gebruikt, maar het is ten zeerste aan te bevelen om naar het GCM over te stappen. Dit kost relatief weinig werk, maar wij zijn hier te laat achter gekomen om het zelf te doen.

10.3 Persoonlijke ervaringen

Direct vanaf het begin van het project was het duidelijk dat een project extern bij een bedrijf doen een stuk anders was dan op de TU. Er wordt meer waarde gehecht aan componenten die nuttig zijn en werken, in plaats van dat de requirements gebaseerd zijn op leerdoelen. Het viel ook op dat ondanks dat we al vakkennis uit de gehele bachelor Technische Informatica hadden, we toch een aanzienlijk deel van de tijd bezig zijn geweest met het uitzoeken hoe dingen werken.

Het was echter wel een leuke uitdaging om van een algemene opdrachtomschrijving zelf een ontwerp te maken, waarmee de opdrachtgever het vervolgens eens moest zijn. De regelmatige vergaderingen hebben hierbij veel geholpen.

Mobile development was voor ons allen nieuw, maar het is wel een snel groeiend begrip binnen ons vakgebied. Uiteindelijk zijn we dan ook blij dat we deze opdracht als eindproject hebben gekozen. We hebben hier veel geleerd, maar vooral ook praktische ervaring opgedaan.

11 Referenties

1. **Mountain Goat Software.** What is Scrum? [Online] [Cited: July 8, 2012.] http://www.mountaingoatsoftware.com/topics/scrum.

2. **Open Handset Alliance.** *Android Developers.* [Online] [Cited: juli 07, 2012.] http://developer.android.com.

3. **Coley Consulting.** MoSCoW Prioritisation. [Online] [Cited: July 8, 2012.] http://www.coleyconsulting.co.uk/moscow.htm.

4. **Rodriguez, Alex.** RESTful Web services: The basics. [Online] November 06, 2008. [Cited: July 8, 2012.] https://www.ibm.com/developerworks/webservices/library/ws-restful/.

5. **Google.** *Google developers.* [Online] [Cited: juli 8, 2012.] https://developers.google.com/android/c2dm/index.

6. **Hollemans, Matthijs.** Apple Push Notification Services (APNS) Overview. *Ray Wenderlich.* [Online] 09 05 2011. [Citaat van: 07 07 2012.] (bron: http://www.raywenderlich.com/3443/apple-push-notification-services-tutorial-part-12/push-overview.

A. Opdrachtomschrijving

1 Bedrijfsomschrijving

Tam Tam is een van de grootste Nederlandse full-service internet-bureaus, langs de A13 gevestigd op de grens van Delft en Rijswijk. Tam Tam draagt zorg bij voor allerhande online wensen die klanten maar hebben. Of het nu gaat om het verzorgen van een corporate internet site, een intranet, een mobiele applicatie, of zelfs een Facebook applicatie, niets is te gek. Tevens wordt onderhoud gepleegd aan reeds opgeleverde projecten.

Voor al dit soort projecten wordt er van alles bijgehouden en geadministreerd, van uren-registratie tot documentatie van bugs. Dit wordt allemaal gedaan met intern gebouwde applicaties. Eén van de onderstaande opdrachten heeft hiermee te maken.

Tam Tam kijkt ook naar waar de markt nu behoefte aan heeft, en waar in de toekomst behoefte aan zal zijn. De volgende stap die Tam Tam wil maken heeft alles te maken met het mobiel benaderen van de zojuist genoemde beheer-applicaties. Ook hier kun jij de bepalende stap maken!

Wat je van ons naast goede begeleiding mag verwachten? Werkplekken inclusief apparatuur, een open en informele sfeer op de werkvloer, meer dan 120 collega's vol ervaring, en natuurlijk een stagevergoeding.

Natuurlijk zijn er buiten de genoemde opdrachten ook andere projecten mogelijk, dus spreken de opdrachten je niet meteen aan, maar heb je wel een leuk idee? Laat het weten!

Neem voor meer informatie contact op met:

Nanda Berkhout

2 Projectomschrijving

Tam Tam in je broekzak

Ontwikkel een algemene Tam Tam app voor iPhone en iPad en evt. Android phones en tablets (en evt. Windows Phone 7 en Windows 8 Metro) die een verzameling aan apps en informatie biedt aan medewerkers van Tam Tam.

Denk dan aan:

- Uren registreren.
- Workflow stappen goedkeuren (bijv. facturatie).
- Zoeken naar klantgegevens op het intranet of in het Customer Relationship Managementsysteem (CRM)
- Project Referentie sheets voor ad-hoc presentaties (iPad only)
- Nieuws
- Tweets
- (Personalizable) Dashboards
- etc.

De app moet een generiek deel hebben (inloggen, menustructuur) en met name op een coherente manier toegang bieden aan de onderliggende apps. Er kan dus gedacht worden aan het ontwikkelen van een framework hiervoor. Bij een eerste release moeten er minimaal 2 goede apps in zitten, en het moet via updates mogelijk zijn om gemakkelijk nieuwe onderliggende apps te pushen.

B. Oriëntatieverslag

1 System Analysis

In de oriëntatiefase hebben we onderzoek verricht naar de bestaande applicaties en de achterliggende systemen. Er is echter naar verwachting meer onderzoek nodig om alles goed op elkaar te laten aansluiten.

1.1 Bestaande applicaties

Werknemers van Tam Tam hebben de beschikking over een aantal administratieve applicaties:

- portal.tamtam.nl – verzamelplaats voor alle applicaties en bedrijfsnieuws
- hourbooking.tamtam.nl – uren boeken
- planning.tamtam.nl – project planning
- crm.tamtam.nl – Customer Relations Management
- skynet.tamtam.nl – Development Platform
- webmail.tamtam.nl – Email/Contact/Calendar

1.2 Eigenschappen applicaties

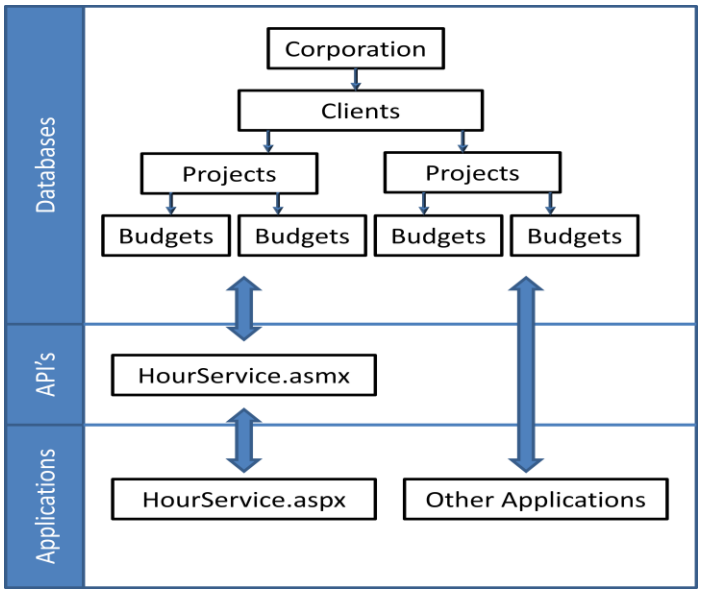
Web-based: Alle applicaties zijn web-based. Er zijn nog geen mobiele applicaties.

Server: De applicaties draaien op Windows servers.

Programming Language: Vrijwel alle applicaties zijn ontwikkeld met behulp van ASP.NET/Silverlight.

Database: Data is opgeslagen in Microsoft SQL server.

1.3 Structuur



Figuur 1-2 Systeemstructuur van Tam Tam

Niet alle applicaties hebben bruikbare API's. Ze gebruiken een directe connectie met de database, waardoor de mobile versie hiervan moet ook databaseverbinding behandelen. In dit geval is implementatie veel moeilijker.

Voor de applicatie "Hourbooking" is in ieder geval een bruikbare API aanwezig. Over de overige applicaties is nog weinig bekend.

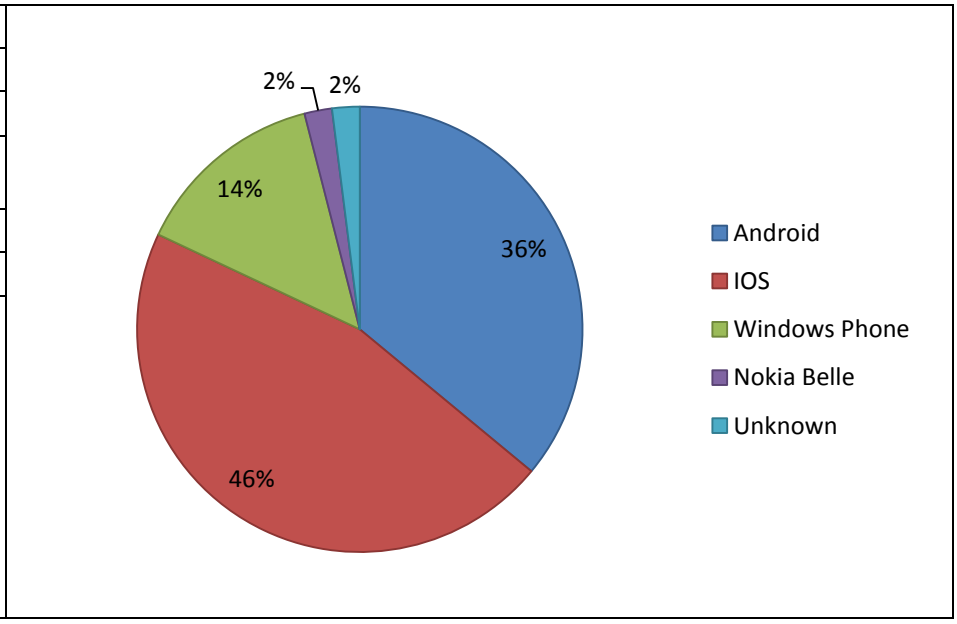
2 User Analysis

In de oriëntatiefase hebben we een kleine survey gehouden, om erachter te komen wat voor telefoons de doelgroep heeft. Deze informatie is belangrijk om fundamentele ontwerpkeuzes te maken, zoals het kiezen van het platform, en welke API versie gebruikt zal gaan worden.
De vragenlijst:

1. What kind of mobile device do you have & Screen size (e.g. Samsung Galaxy S II i9100 800 x 480)?
2. What is your OS & Version (e.g. Android 4.0.1)?
3. Which Tam Tam web application do you use the most (e.g. Hour Registration)?
4. Which other Tam Tam web applications would you like to have on your mobile device?
5. Do you have any suggestions, tips or tricks for us :)?

2.1 Platform

Platform	Gebruikers
Android	18
IOS	23
Windows Phone	7
Nokia Belle	1
Overig	1
Totaal	50

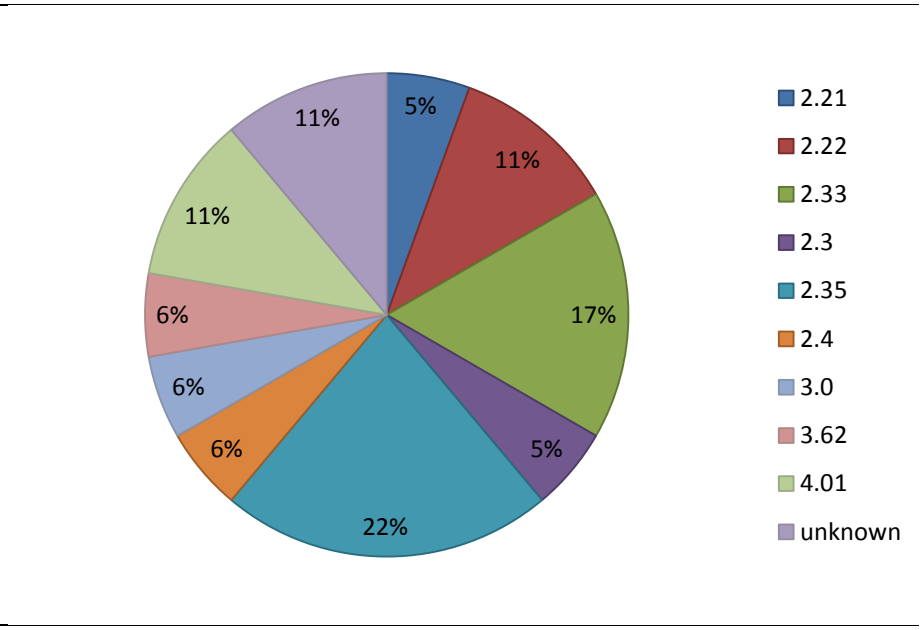


Figuur 2-1: Aantal gebruikers per platform

iOS is in de meerderheid met 46% van de werknemers. 36% gebruikt Android, en 14% heeft Windows Phone. Hieruit blijkt dat de applicatie in ieder geval voor iOS en Android ontwikkeld zal moeten worden om voldoende gebruikers te hebben. Gezien het lage aandeel van Windows Phone, zal dit platform buiten beschouwing blijven.

2.2 Versie

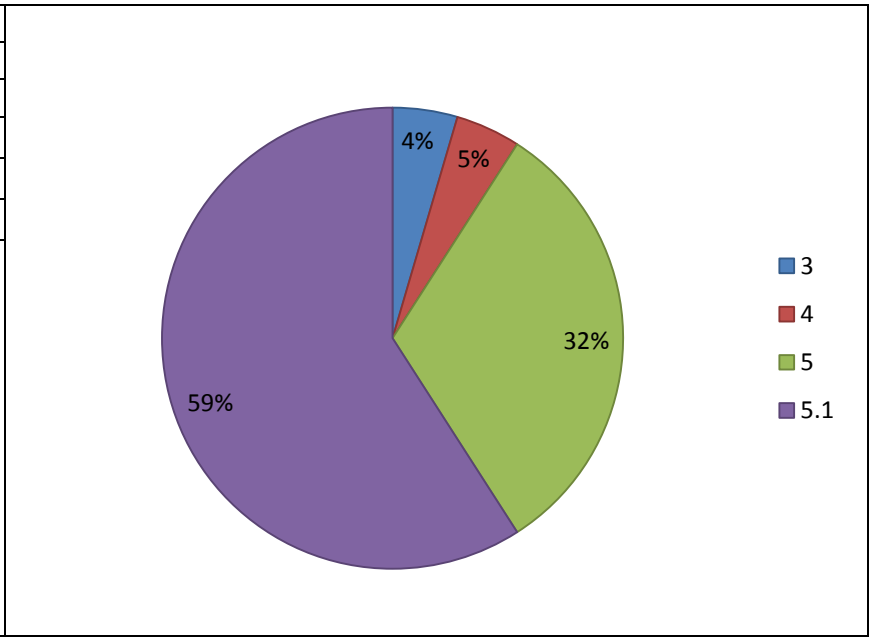
Versie	Gebruikers
2.21	1
2.22	2
2.33	3
2.3	1
2.35	4
2.4	1
3.0	1
3.62	1
4.01	2
Overig	2
Totaal	18



Figuur 2-2: Aantal gebruikers per Android versie

Ongeveer 2/3 van de Android gebruikers hebben versie 2.x. De Tam Tam applicatie zal dus ontwikkeld worden met een API die deze versie ondersteunt.

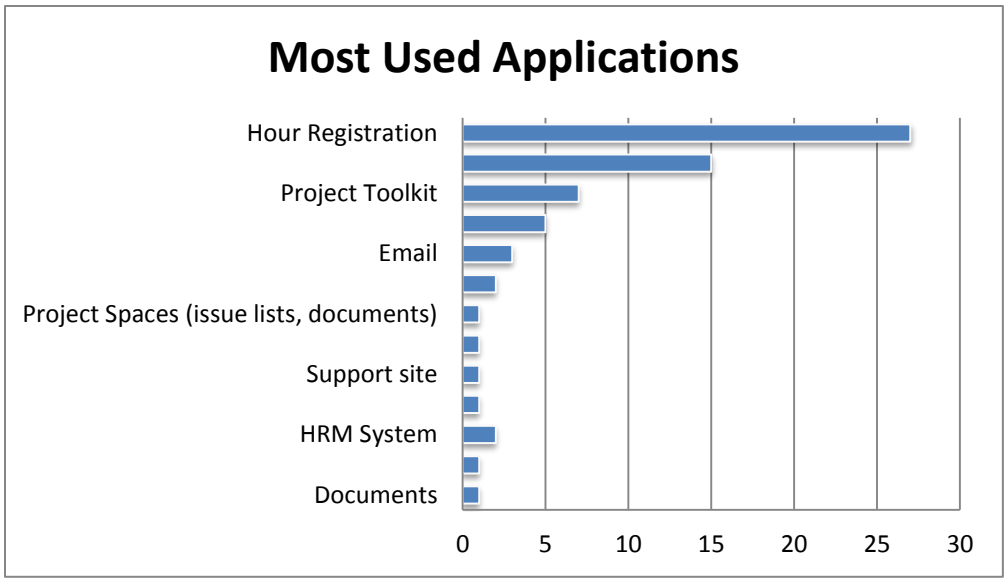
Versie	Gebruikers
3	1
4	1
5	7
5.1	13
Overig	1
Totaal	23



Figuur 2-3: Aantal gebruikers per iOS versie

Versie 5.x wordt door 90% van de werknemers gebruikt. De applicatie kan dus voor deze versie worden ontwikkeld.

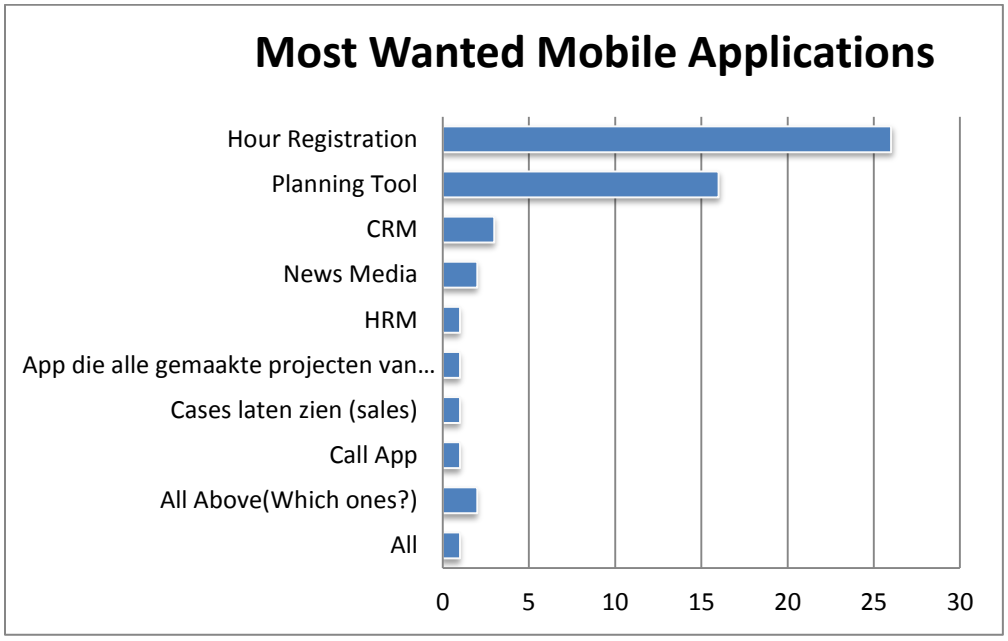
2.3 Meest gebruikte applicaties



Figuur 2-4: Meest gebruikte applicaties

De meest gebruikte applicaties bleken “Hour Registration”, ”Planning Tools” en “Project Toolkits” te zijn. De eerste applicatie die we zullen implementeren zal dan ook “Hour registration” te zijn.

2.4 Meest gewenste applicaties



Figuur 2-5: Meest populaire applicaties voor een mobiele versie

Ook de meest gewilde applicaties bleken “Hour Registration” en ”Planning Tools” te zijn.

3 Design Choices

Met het oog op de System Analysis en User Analysis, moet er een beslissing worden genomen met betrekking tot de ontwikkeling. In dit hoofdstuk worden de alternatieve oplossingen met elkaar vergeleken en dan wordt de meeste geschikte combinatie van frameworks, programmeertalen, gebruikersinterfaces en database uitgekozen.

3.1 Framework

Voor het framework kunnen we kiezen tussen de volgende alternatieven:

- 100% Native
- Native + Web
- 100 % Web

Native + Web lijkt momenteel het beste alternatief. Ten eerste is 46% van de werknemers in het bezit van iOS, maar we hebben geen iOS development platform tot onze beschikking. 100% native zou veel extra werk kosten voor het poorten van Android naar iOS. Wellicht is er dan ook weer extra onderzoek nodig voor het maken van een verbinding met de server.

Verder zou 100% web kunnen leiden tot trage applicaties, en is een aantal native functionaliteiten niet beschikbaar voor websites (camera, trilfunctie). Een combinatie van de twee zal de bovengenoemde mankementen beperken. Met een native framework, waarbinnen de webapplicaties worden getoond, zal navigatie een stuk sneller gaan. Het omzetten naar iOS zal minder werk kosten en native functionaliteit kan worden gebruikt. Om ook met trage internetverbinding redelijk te kunnen werken, zal zelfs een groot deel van de webcontent lokaal worden opgeslagen, en slechts de gegevens van de server worden gehaald. Ten slotte kunnen frameworks zoals JQuery Mobile HTML-pagina's eruit laten zien als native apps, met support voor meerdere schermgroottes.

3.2 Web sub-apps

- Basic Web(HTML5 + JavaScript + CSS)
- Basic Web + Cross Platform Framework (jQuery Mobile, Phonegap etc.)

Sub-apps moeten gemakkelijk kunnen worden ingevoegd in het framework van de native app. De client-side van web sub-apps zal naar alle waarschijnlijkheid in HTML/Javascript worden ontwikkeld, en we kunnen hiernaast ook gebruik maken frameworks als JQuery Mobile. Het voordeel hiervan is dat het eenvoudig is om snel een applicatie te maken die aanvoelt als een native app, terwijl alle platforms worden ondersteund. Het zou echter ook zo kunnen zijn dat de snelheid hieronder komt te lijden. Dit wordt nog onderzocht.

Het is echter niet noodzakelijk om alle web sub-apps op dezelfde manier te ontwikkelen. Gezien het onafhankelijke karakter van de sub-apps zijn toekomstige ontwikkelaars hier geheel vrij in.

3.3 Server Side

- PHP
- ASP.NET

Aanvankelijk waren we van plan om PHP te gebruiken, aangezien we hier al ervaring mee hebben, maar niet met ASP.NET. PHP bleek echter niet veel gebruikt te worden binnen Tam Tam, en ASP is meer geschikt voor de Microsoft-omgeving die gebruikt wordt. Ook zijn de meeste huidige applicaties hierin ontwikkeld, dus verbinding met databases en bestaande webservices zal dan makkelijker zijn.

3.4 Database

- MySQL
- Microsoft SQL Server

Hoewel we meer ervaring hebben met MySQL, zal het opnieuw beter zijn om ons aan te passen aan de bestaande systemen, dus kiezen we hier voor MS SQL Server.

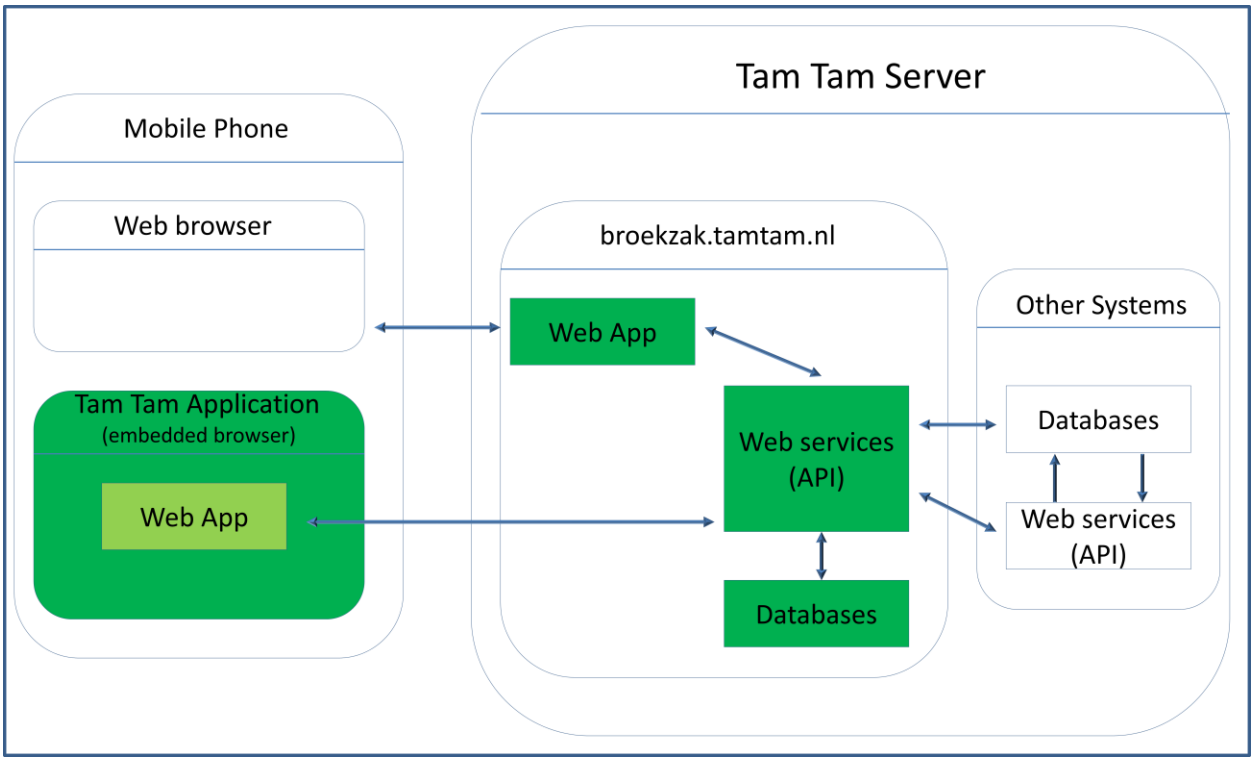
3.5 User Interface

- List View
- Tab View (Deprecated)
- ActionBarSherlock (externe plugin)
- ActionBar (Android 3.0+)

Dit gaat tot dusver alleen nog over Android, aangezien we daarmee beginnen. De meest logische keuze lijkt het gebruik van lijsten, omdat dat gangbaar is in de meeste bestaande apps. Andere alternatieven zijn niet meer geschikt, of niet ondersteund door alle versies van Android.

4 "Tam Tam in je broekzak" App overzicht

De data flow van het hele systeem is als volgt:



Figuur 4-1: Voorlopige structuur van het geheel

Op de telefoon (client side) wordt de mobile app uit de appstore geïnstalleerd. Deze bestaat uit een framework met een browser view voor web sub-apps. Deze communiceren vervolgens met de webservice op de server.

Op de server verbindt de webservice de applicatie door met de database/andere Tam Tam servers. Deze webservice is ontwikkeld in ASP.NET. Ook wordt hier een database bijgehouden met informatie zoals instellingen en geschiedenis.

In het geval dat de applicatie om wat voor reden dan ook niet werkt op een bepaald platform of versie van een platform, kan er ook een mobiele website worden ontwikkeld waarin de web sub-apps kunnen worden gebruikt.

C. Plan van aanpak

1 Opdrachtomschrijving

1.1 Inleiding

Binnen Tam Tam wordt er een aantal web applicaties gebruikt door de werknemers. Hierbij kan worden gedacht aan het registreren van werkuren, het plannen van projecten, of het goedkeuren van facturen die naar klanten gestuurd zullen worden. Deze applicaties zijn echter niet geoptimaliseerd voor mobiel gebruik.

1.2 De opdrachtgever

Tam Tam, hoofdvestiging (Delft).

1.3 Contactpersonen

Mentor binnen Tam Tam: Wouter van Weelderen

Begeleider TU Delft: Cor-Paul Bezemer

1.4 Doelstelling project

Er moet een mobile app worden ontwikkeld, ten minste voor de Android en iOS platformen. Deze app moet gemakkelijk in gebruik zijn en beschikbaar voor alle werknemers en als het mogelijk blijkt te zijn de gemaakte applicaties te submitten in de appstore.

1.5 Opdrachtformulering

De mobile app moet een generiek deel hebben, dat de “look & feel” heeft van native apps, oftewel het framework. Hiernaast zal er minstens één van de webapps een mobiele versie krijgen, die beschikbaar is in de mobile app. Deze zal ook goed moeten aansluiten op bestaande systemen.

1.6 Op te leveren producten

Een complete en bruikbare Tam Tam mobile app, die voor alle werknemers gemakkelijk te verkrijgen is. De meest logische manier van distributie is het plaatsen in de app-stores van de platformen. De mobile app moet het uitvoeren van administratieve taken vergemakkelijken.

2 Aanpak

2.1 Inleiding

Tam Tam is welbekend met het uitvoeren van softwareprojecten. Wij zullen ons dan ook zoveel mogelijk houden aan de vormen van aanpak die gangbaar zijn binnen het bedrijf.

2.2 Methodiek

Voor de meeste projecten binnen Tam Tam wordt gebruik gemaakt van de Agile Scrum ontwikkelmethode. Deze methode zullen wij dan ook aanhouden, waarbij er sprints worden gehouden van 2 weken. Aan het begin van elke sprint zal overleg plaatsvinden met de mentor binnen Tam Tam om de doelstellingen te bepalen.

2.3 Technieken

Voor het ontwikkelen van Android componenten wordt Java gebruikt, daar dit wordt verplicht door het Android framework. Voor de server gebruiken we ASP.NET, omdat dit de meest gebruikte techniek is binnen Tam Tam. Voor iOS ontwikkeling gebruiken we Objective-C, omdat ook dit wordt afgedwongen door Apple.

2.4 Werkzaamheden

Per sprint zullen we bezig zijn met zowel ontwerpen als implementeren en testen. Hiernaast zullen we de begeleiders op de hoogte houden van de voortgang, en een logboek bijhouden van de activiteiten per dag.

2.5 Betrokkenen

Wouter van Weelderen: mentor binnen Tam Tam

Ramon Nagelhout: heeft de meeste huidige applicaties ontwikkeld, en heeft kennis van de structuur

Roel Spruit: heeft veel ervaringen met mobile development

Afdeling PCS: het opzetten van en toegang krijgen tot servers, databases e.d.

2.6 Faciliteiten

In het kantoor hebben wij een bureau toegewezen gekregen. Van hieruit hebben wij toegang tot het bedrijfsnetwerk, en kunnen Tam Tam werknemers eenvoudig worden benaderd voor feedback.

2.7 Planning

Conform de Scrum ontwikkelmethode, zal eerst een lijst met eisen aan het totaalproduct worden opgesteld in overleg met Tam Tam. Hieruit zal vervolgens elke twee weken een selectie worden gemaakt, die dan in de opeenvolgende twee weken voltooid zal moeten worden. Een algemene planning is als volgt:

Week	Dag	Fase	Doorlooptijd	Tasks	Event & Deliverable
18	30 Apr - 4 May	Oriëntatiefase	2 weeks	Onderzoek naar probleem & oplossingsmethoden	
19	7 May - 11 May				Oriëntatieverslag (5-10 bladzijden)
20	14 May - 18 May	Implementatiefase	7 weeks	Ontwerp, Implementatie & Test	
21	21 May - 25 May				
22	28 May - 01 Jun				
23	04 Jun - 08 Jun				
24	11 Jun - 15 Jun				1e keer code inleveren bij SIG
25	18 Jun - 22 Jun				
26	25 Jun - 29 Jun				2e keer code inleveren bij SIG
27	02 Jun - 06 Jul	Afrondingsfase	2 weeks	Documentatie & Release	Stageverslag
28	09 Jul - 13 Jul				Presentatie(30 à 35 minuten)

Tabel 2-1: Algemene planning voor de duur van het project

3 Kwaliteitsborging

3.1 Documentatie

Alle door ons geschreven code zal zo worden gedocumenteerd, dat eventuele andere ontwikkelaars het project kunnen uitbreiden zonder eerst alle code hoeven door te lezen. Verder zal in het eindverslag worden weergegeven hoe de componenten samen werken, en hoe het systeem is uit te breiden.

3.2 Versiebeheer

Voor het opslaan en delen van al het project gerelateerde werk wordt een Subversion server van Tam Tam gebruikt.

3.3 Unit tests

Voor het testen van de code en het opsporen van eventueel ongewenst gedrag bij uitbreiding zullen er unit tests worden geschreven voor zowel het framework als de webservice.

3.4 Proces

We werken volgens de Scrum ontwikkelmethode om het project zo vloeiend mogelijk te laten verlopen, aangezien dit het makkelijkst zal werken voor Tam Tam.

D. Logboek

Week 19

Planning:

07-05 – 11-05	Bastiaan	Wing	Arvind
Maandag	Demoversie uitbreiden, documentatie schrijven, interview Ramon	Onderzoek server Survey verwerken Interview Ramon	Android ontwerp, vragenlijst tutorials doornemen.
Dinsdag	Meeting plannen, diagrammen data flow	Server FTP Website uploaden evt. authenticatie	ASPX.NET, Android deployment
Woensdag	Begin plan van aanpak, http authenticatie	Verbinden met HourBooking Webservice	Aanpak Android design. Bekijken van tutorials.
Donderdag	Plan van aanpak + o oriëntatie verslag conceptversies afmaken, design choices + requirements, onderzoek jQuery	Onderzoek SQL Server Documentatie: Design Choices + System Analysis + User Analysis	
Vrijdag	Orientatieverslag afmaken, verder met jQuery	Documentatie: Orientatieverslag	

Logboek:

07-05 – 11-05	Bastiaan	Wing	Arvind
Maandag	Javadoc voor demo, structuur verbeterd	Survey Doc ASP.NET TUTORIALS	ASP.NET 4.0 test
Dinsdag	Authenticatie onderzocht, diagrammen data flow gemaakt	Server configureren FTP uploaden configureren Website online.	Verder met onderzoek naar ASP. Hoe effectief een web app te maken.
Woensdag	Authenticatie opzet geïmplementeerd, plan van aanpak	Verbinden met HourBooking Webservice	Skeleton Design uitdenken en hoe te koppelen met Hourbooking.
Donderdag	jQuery begonnen, verslagen conceptversies	Documentatie: Design Choices + System Analysis + User Analysis	
Vrijdag	Orientatieverslag gestuurd naar leden meeting maandag, jQuery onderzoek	Documentatie: Orientatieverslag	jQuery design en implementatie voor Hourbooking.

Week 20

Planning:

07-05 – 11-05	Bastiaan	Wing	Arvind
Maandag		Onderzoek SQL Server Ontwerp: Server Requirements.	Creëren van de PIN Interface & startup logic.
Dinsdag		Gesprek: TU begeleider	Creëren van de PIN Interface.
Woensdag		Onderzoek SQL Server Server-side User Account/Database	Creëren van de PIN Interface.
Donderdag	Hemelvaartsdag		Creëren van de PIN Interface.
Vrijdag		Server-side Gebruiker Identificatie Server-side Gebruiker Account/Database	Creëren van de PIN Interface.

Logboek:

07-05 – 11-05	Bastiaan	Wing	Arvind
Maandag	Login activity in app	Onderzoek SQL Server Design: Server Requirements.	Pin design
Dinsdag	Gesprek TU begeleider	Gesprek: TU begeleider	Gesprek: TU begeleider
Woensdag	Verder met login activity	Onderzoek SQL Server Server-side Gebruiker Account/Database	Pin re-design
Donderdag	Hemelvaartsdag	/	Hemelvaartsdag
Vrijdag	Encryptie van logingegevens	Server-side Gebruiker Identificatie Server-side Gebruiker Account/Database	Pin Design

Week 21

Planning:

21-05 – 25-	Bastiaan	Wing	Arvind

05			
Maandag	Encryptie verder uitzoeken, authenticatie bespreken met Jasper	Server-side Gebruiker Account/Database	Werk aan PIN en alle logica erachter, zoals interval, startup en wijzigen.
Dinsdag	ADFS onderzoek	ADFS Authenticatie	Werk aan PIN en alle logica erachter, zoals interval, startup en wijzigen.
Woensdag	‘’	ADFS Authenticatie	Werk aan PIN en alle logica erachter, zoals interval, startup en wijzigen.
Donderdag	‘’	WCF Tutorials	Werk aan PIN en alle logica erachter, zoals interval, startup en wijzigen.
Vrijdag	Android service verbinden aan WCF service	WCF Tutorials	Werk aan PIN en alle logica erachter, zoals interval, startup en wijzigen.

Logboek:

21-05 – 25-05	**Bastiaan**	**Wing**	**Arvind**
Maandag	ADFS onderzoek	Server-side Gebruiker Account/Database	
Dinsdag	‘’	Server-side Gebruiker Account/Database	
Woensdag	‘’	Server-side Gebruiker Account/Database	
Donderdag	‘’	WCF Tutorials	
Vrijdag	Services	WCF Tutorials	Planning af, alles gerelateerde aan de PIN is af.

Week 22

Planning:

28-05 – 01-06	**Bastiaan**	**Wing**	**Arvind**
Maandag	2e pinksterdag		2e pinksterdag
Dinsdag	Service verder implementeren	Abonnement op sub-apps Menustructuur (Frontend)	Leer Objective -c
Woensdag	Httpclient login	Abonnement op sub-apps Menustructuur (Frontend)	Leer iOS
Donderdag	Login app, webservice connection	Abonnement op sub-apps Webservice (Backend)	Tutorials iOS

Vrijdag	Werken op TU, service implementeren	Abonnement op sub-apps Webservice (Backend)	Maak een View

Logboek:

28-05 – 01-06	**Bastiaan**	**Wing**	**Arvind**
Maandag	2e pinksterdag		2e pinksterdag
Dinsdag	Service uitgebreid	Abonnement op sub-apps Menustructuur (Frontend)	Objective – c, twee goede tutorials gevonden.
Woensdag	Webservices kunnen nu ook van buiten Tam Tam worden bereikt	Abonnement op sub-apps Menustructuur (Frontend)	Macbook opgehaald, en geconfigureerd.
Donderdag	Singleton class bruikbaar gemaakt, begin aan login systeem	Abonnement op sub-apps Webservice (Backend)	Een UITableView gemaakt.
Vrijdag	Begin gemaakt aan android service	Abonnement op sub-apps Webservice (Backend)	iOS Tutorials.

Week 23

Planning:

04-06 – 08-06	**Bastiaan**	**Wing**	**Arvind**
Maandag	Service verder uitbreiden, netwerkoperaties goed krijgen	Abonnement op sub-apps	iOS Tutorials.
Dinsdag	AsyncTask geschikt maken voor algemeen gebruik	Gebruikerdata Lezen (Frontend) from Database (Backend)	iOS Tutorials.
Woensdag	Thuis werken	Gebruikerdata Lezen (Frontend) from Database (Backend)	Hard coded navigatie tussen de RootViewController en SettingsViewController
Donderdag	Code refactoren voor SIG	Basic Authenticatiie (Back-end)	Settings UI opgevuld.
Vrijdag	Code verder refactoren, Android lint checken, eventueel authenticatie	Code Refactoren & Documentatie	Settings designs, iets anders dan Android.

Logboek:

04-06 – 08-06	**Bastiaan**	**Wing**	**Arvind**
Maandag	Mogelijkheid voor niet-parallelle	Abonnement op sub-apps	RootViewController logica toegevoegd.

	netwerkoperaties, login gegevens bewaren, one way encryption		
Dinsdag	AsyncTask algemeen gemaakt, commentaar in code gezet, internet verbinding requirement	Gebruikerdata Lezen (Frontend) from Database (Backend)	Navigate naar SettingsViewController en WebViewController, google.com. wordt geladen.
Woensdag	Update subscriptions bug gefixed	Gebruikerdata Lezen (Frontend) from Database (Backend)	Connectie met de server gemaakt en data verzonden.
Donderdag	Activity functionaliteit uitgebreid voor wrapper, alert dialog functie gemaakt, comments afgemaakt	Basic Authenticatie (Back-end)	Data van de server gelezen, LoginViewController gemaakt.
Vrijdag	Login afgemaakt, code gestuurd naar SIG	Code Refactoren & Documentatie	De front end van de applicatie grotendeels afgemaakt.

Week 24

Planning:

11-06 – 15-06	**Bastiaan**	**Wing**	**Arvind**
Maandag	Push notifications framework	Webpagina loaden in webview, geauthenticeerd	Server functionaliteiten implementeren, ServiceSingelton uitbreiden.
Dinsdag	Registratie app/device bij C2DM	Externe apps laden in menu Notificatie opslaan in Database	Server functionaliteiten implementeren, ServiceSingelton uitbreiden.
Woensdag	Server implementatie push, algemene POST methode in singleton	Externe apps laden in menu	Loading Screen maken
Donderdag	Server implementatie C2DM	Push Notificatie (Backend)	Encryptie uitzoeken
Vrijdag			Manier vinden om data op te slaan.

Logboek:

11-06 – 15-06	**Bastiaan**	**Wing**	**Arvind**
Maandag	Push notifications onderzoek, sender ID aangemeld	Webpagina loaden in webview, geauthenticeerd	Paswoord word opgslagen door de KeychainitemWrapper.

			Andere data wordt opgeslagen door NSUserDefaults.
Dinsdag	Push notifications registratie C2DM	Externe apps laden in menu Notificatie opslaan in Database	Protocols en custom delegates gemaakt.
Woensdag	Post in singleton gezet, server emulator gemaakt	Externe apps laden in menu	Asynchrone connecties d.m.v. delegates.
Donderdag	Server implementatie grotendeels af	Push Notificatie (Backend)	Logica voor de connecties op succes en failure.
Vrijdag			

Week 25

Planning:

18-06 – 22-06	**Bastiaan**	**Wing**	**Arvind**
Maandag	Voortgangsgesprek met Cor-Paul	Voortgangsgesprek met Cor-Paul	iOS framework afmaken, alles behalve push notifications.
Dinsdag	Afhandelen van C2DM	Externe apps laden in menu	iOS framework afmaken, alles behalve push notifications.
Woensdag	Project opruimen	Systeem Administration tool	iOS framework afmaken, alles behalve push notifications.
Donderdag	Service singleton opsplitsen, Listeners maken voor wrappers (postexecute)	Systeem Administration tool	iOS framework afmaken, alles behalve push notifications.
Vrijdag			iOS framework afmaken, alles behalve push notifications.

Logboek:

18-06 – 22-06	**Bastiaan**	**Wing**	**Arvind**
Maandag	TU werkzaamheden		Logica voor de Settings View.
Dinsdag	Package name gefixed, pendingintent bugs gefixed (cache problem), register/unregister bij inloggen/uitloggen,	Externe apps laden in menu	Logout functie

	HTTPClient voor multiple threads (en SSL?), afhandelen van send errors serverside		
Woensdag	Onnodige dingen verwijderd, SVN ingedeeld, server update (return voor send notification)	Systeem Administration tool	Dynamisch maken van de Root View aan de hand van de uitgelezen data.
Donderdag	HTTPS functionaliteit + wrapper listener	Systeem Administration tool	PIN gemaakt.
Vrijdag			PiN Interval, iOS port is grotendeels afgekomen, in het weekend is het meeste afgemaakt.

Week 26

Planning:

25-06 – 29-06	**Bastiaan**	**Wing**	**Arvind**
Maandag	Testing framework research	Research Unit Test in WCF	Push Notifications certificaten
Dinsdag	Verder met testen	Research Unit Test in WCF	Push Notifications , Provisioning profile
Woensdag	Connection loss bug fixen	Unit Test in WCF	Push Notifications iPhone registreren.
Donderdag	Singletons opsplitsen, unit tests maken	Unit Test in WCF	Versturen van Push Notifications
Vrijdag			

Logboek:

25-06 – 29-06	**Bastiaan**	**Wing**	**Arvind**
Maandag	Begonnen met test classes	Research Unit Test in WCF	Correcte certificaten aangemaakt, IPA Archive geëxporteerd.
Dinsdag	Orientation change bug gefixed, meer testing gedaan	Research Unit Test in WCF	Setup voor Xcode, Push Notifcations verder afmaken.
Woensdag	Connection bug gefixed, login layout	Unit Test in WCF	IPA gestest, Xcode verder voorbereiden voor Push Notifications.
Donderdag	Singletons gesplitst, testing	Unit Test in WCF	iPhone geregistreerd.

Vrijdag			

Week 27

Planning:

02-07 – 06-07	**Bastiaan**	**Wing**	**Arvind**
Maandag	Documentatie	Documentatie	
Dinsdag	Documentatie	Documentatie	Documentatie
Woensdag	Documentatie	Documentatie	Documentatie
Donderdag			Refactoren & Push Notifications .
Vrijdag			Refactoren & Push Notifications.

Logboek:

02-07 – 06-07	**Bastiaan**	**Wing**	**Arvind**
Maandag	Documentatie	Documentatie	Documentatie
Dinsdag	Documentatie	Documentatie	Documentatie
Woensdag	Documentatie	Documentatie	Documentatie
Donderdag			
Vrijdag			Refactoren

E. Achtergrondinformatie

1 Android

1.1 Framework

Er zijn enkele klassen die standaard in elke Android applicatie moeten worden gebruikt:

Activity – dit is de meest gebruikte klasse. Alles wat een gebruiker ziet of doet gaat via een Activity. Deze klasse laat UI elementen zien, heeft een verbinding met de back-end, en kan andere Activities opstarten. In principe heeft elk gedeelte van de app een eigen Activity. Bij het wisselen tussen Activities wordt standaard een stack opgebouwd, waardoor de back-button gebruikt kan worden om naar de vorige Activity terug te gaan.

Intent – een klasse die informatie bevat over een actie die moet worden uitgevoerd. Vaak is dit het starten van een Activity, maar ook het sturen van de registratie voor C2DM en het starten van andere applicaties gebeurt via een Intent. Behalve informatie over de actie kan een Intent ook extra data meedragen, die door een andere Activity kan worden uitgelezen. Activities kunnen alleen maar worden gestart met een Intent.

Verder wordt voor iedere app automatisch een manifest aangemaakt. In dit XML-bestand staat allerlei informatie over de app, zoals versienummer, compatibiliteit met Android-versies, en een lijst met de activities. Ook wordt hierin aangegeven welke activity moet worden gestart bij het starten van de app.

Voor de user interface wordt ook veelal gebruik gemaakt van XML-bestanden, hoewel het ook via programmacode kan. Binnen de XML-bestanden kunnen verschillende UI elementen met parameters gedefinieerd worden.

1.2 C2DM

C2DM maakt het relatief eenvoudig om vanaf een server berichten te sturen aan geregistreerde telefoon. Gebruikers worden automatisch geregistreerd voor berichten zodra ze inloggen, en uitgeschreven zodra ze uitloggen. De maximale lengte voor een bericht verstuurd vanaf de C2DM server (van Google) is ongeveer 1000 karakters. Om het sturen van berichten mogelijk te maken moeten de volgende stappen eenmalig worden uitgevoerd:

1. De server stuurt een POST-request naar de C2DM server met daarin de gebruikersnaam en wachtwoord van een C2DM-enabled Google account
2. De C2DM server stuurt een Authentication token terug, dit wordt door de server opgeslagen
3. De app stuurt de C2DM server een registratieverzoek
4. Als de telefoon aan de eisen voldoet (Market geïnstalleerd, Google account ingelogd) stuurt de C2DM server de telefoon een Registration ID
5. De app stuurt het Registration ID naar de server

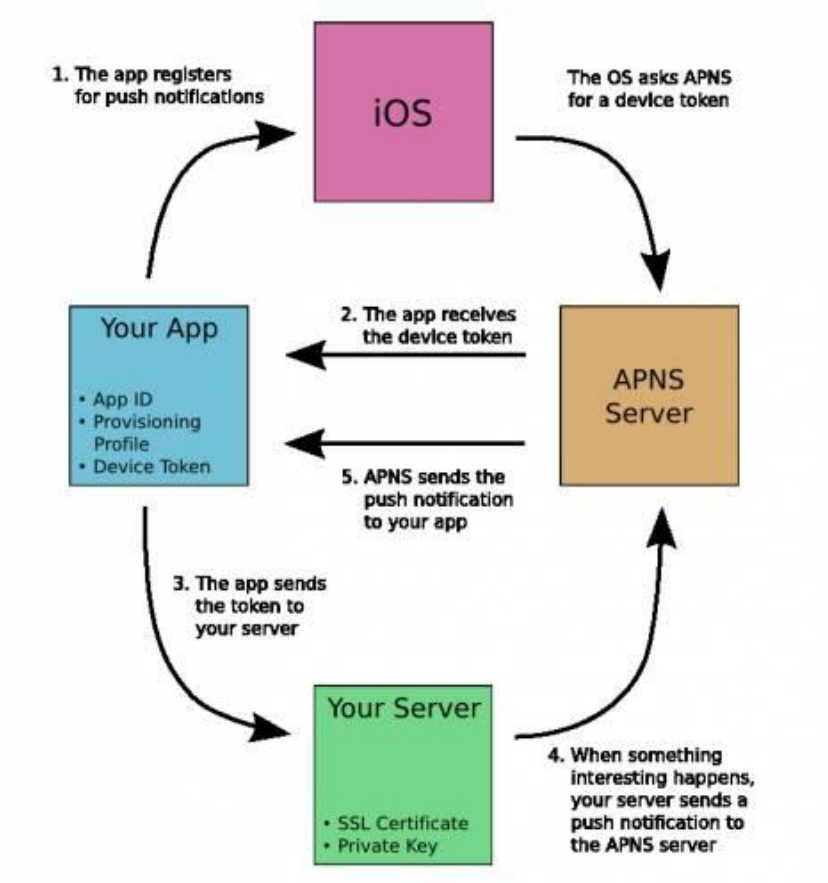
Nu kan de server berichten sturen naar de C2DM server. Dit wordt gedaan middels een POST-request, waarbij het Registration ID en Authentication token moeten worden meegezonden. De C2DM server stuurt het bericht vervolgens door naar de telefoon, waarna de mobile app hiervan op de hoogte wordt gesteld.

Voor het gebruik van C2DM is er wel een tweetal eisen aan de gebruiker: er moet een Google account zijn gekoppeld aan de telefoon, en de Google play applicatie moet zijn geïnstalleerd. De berichtenservice maakt namelijk gebruik van de bestaande continue verbinding met Google play.

2 iPhone

2.1 APNS

Voor de iOS versie kun je Push Notifications sturen door the communiceren met de APNS server van Apple. Hiervoor heb je certificaten en private keys nodig, deze moet je zelf aanmaken en in de developers portal registreren om de Application Profile en Provisioning Profile op te zetten. Door middel van een bericht van onze server naar de APNS server kan er een Push Notification worden gestuurd naar alle geregistreerde apparaten. Zie het figuur hieronder voor een goed overzicht (6).



Figuur 2-1: Communicatie met de APNS Server voor Push Notifications

3 Server

3.1 .NET Framework

.NET Framework is een applicatieframework ontwikkeld door Microsoft. Binnen .NET Framework worden grotere hoeveelheden van libraries en toolkits aangeboden. De integratie tussen .NET componenten is zeer gewenst. In dit project hebben wij gekozen om uitsluitende de .NET technologie te gebruiken voor de implementatie van de server-side applicaties. De volgende .NET componenten komen aan bod in ons project:

- WCF service: deze technologie wordt gebruikt voor de implementatie van ons webservice. Meer details in paragraaf 3.2.
- ASP.NET: deze technologie wordt gebruikt voor de implementatie van de System Administration tool. ASP.NET is het meest gebruikelijke framework voor de ontwikkeling van websites in Windows platform.
- ASMX webservice: sommige bestaande webservice van Tam Tam is geïmplementeerd met ASMX technologie. De aansluiting tussen ASMX webservice en onze webservice is gedaan via SOAP verbinding.
- IIS Server: de WCF service, de ASP.NET webpage en de ASMX webservice draaien op IIS Server.
- Microsoft SQL Server: Ons database maakt gebruikt van de Microsoft SQL Server. Deze technologie is geïntegreerd met .NET framework.

3.2 WCF service

In plaats van ASMX technologie, wordt WCF technologie steeds meer gebruikt voor de implementatie van .Net webservice. De client applicatie verbindt zich met WCF webservice via het zogenaamde endpoint. In het endpoint kan het verbindingsytpe gespecifieerd worden. Onze webservice maakt gebruikt van RESFful verbinding en SOAP verbinding. Hiernaast kan WCF webservice database importeren van de Microsoft SQL Server.

3.2.1 SOAP verbinding

Naast RESTful verbinding, kunnen de API's van de WCF webservice ook aangeroepen worden met een Simple Object Access Protocol (SOAP) verbinding. De SOAP verbinding is speciaal bedoeld voor andere .NET server-side applicaties. Met een SOAP verbinding kunnen .NET applicaties de WCF webservice importeren, en de API's aanroepen als eigen methodes.

3.2.2 RESTful verbinding

De RESTful verbinding maakt gebruikt van het HTTP-protocol, namelijk de HTTP-POST methode om JSON objecten uit te wisselen. RESTful data uitwisseling is eenvoudig, snel en platform-compatibel.

- Eenvoudig: de datapakketten kunnen makkelijk ingelezen worden door de mobile app.
- Snel: de datapakketten zijn klein zodat een langzame internetverbinding geen probleem is.

- Platform-compatibel: RESTful verbinding werkt in Windows, Android en iOS platforms.

3.2.3 ADO.NET Data Entity Model

Databases van de Microsoft SQL Server kunnen geïmporteerd worden in de vorm van ADO.NET Data Entity Model. Language Integrated Query (LINQ) zorgt voor een eenvoudig interface tussen de webservice en de ADO.Net Data Entity Model.

3.2.4 Fiddler Web Debugger

Exploratory testing van de webservice is gedaan met behulp van Fidder. Fiddler is een Web Debugging Tool die alle HTTP(S) verkeer tussen de computer en het internet kan bijhouden. Met Fiddler kunnen methodes van het HTTP-Protocol zoals POST en GET gesimuleerd worden. Omdat de verbinding tussen de webservice en de mobile app volledig is gebaseerd op het HTTP-Protocol, is Fiddler een handige test toolkit voor onze webservice.