Afstudeerverslag

Bedrijf	42 BV
Projectnaam	Goudvink
Auteur	Wilco Stuyfersant
Afstudeerperiode	7 mei 2012 – 24 september 2012
Versie	1.0, 24 september 2012

Voorwoord

Dit afstudeerverslag, geschreven door Wilco Stuyfersant, vierde jaars informatica student aan De Haagse Hogeschool, beschrijft het proces rondom de verrichte werkzaamheden tijdens de afstudeerstage. Het doel van dit afstudeerverslag is om de gekozen werkwijze te beschrijven en de daarbij behorende keuzes en problemen waar ik tegenaan ben gelopen toe te lichten. Middels dit verslag en bijbehorende bijlages is de examencommissie in staat om mijn verrichte werkzaamheden tijdens het afstudeerproject te kunnen beoordelen.

Graag wil ik een aantal mensen bedanken voor de hulp die zij hebben geboden gedurende het afstudeerproject.

Als eerste wil ik mijn bedrijfsmentor, Robert Bor, bedanken voor de vele gesprekken en zijn daarbij continu verhelderende blik op de opdracht.

Ook wil ik mijn projectteam, bestaande uit Michel Noppert, Margret Kouwen en Paulien de Wind, bedanken voor de ontzettend prettige samenwerking gedurende de eerste maand van mijn afstuderen. We hebben een aantal zeer leerzame discussies gevoerd waar ik in de toekomst zeker nog wat mee kan.

Zowel Ferdy Perdaan als Jeroen Niesen wil ik bedanken voor de feedback die zij hebben gegeven op het afstudeerverslag. Hun kritische blik op het verslag heeft de uiteindelijke kwaliteit verhoogd.

Tenslotte wil ik de overige medewerkers van 42 BV bedanken die mij op moeilijke punten van het afstudeerproject hebben geholpen met hun verhelderende kennis over diverse probleemgevallen.

Inhoud

1 Inleiding

Aan de opleiding informatica aan De Haagse Hogeschool wordt verwacht dat de student in de laatste periode van zijn studie een afstudeertraject doorloopt. Hierbij wordt binnen het vakgebied een opdracht uitgevoerd voor een bedrijf. Het doel van deze afstudeeropdracht is de examencommissie in staat te stellen om de prestaties van de individuele student te kunnen beoordelen.

Dit verslag beschrijft hoe ik de zeventien weken afstuderen heb ervaren, wat mijn werkzaamheden zijn geweest, hoe ik met problemen en keuzes ben omgegaan en welke leermomenten ik daarbij heb gehad.

Ik heb mijn afstudeerproject als leerzaam en leuk ervaren en hoop dat de lezers van dit verslag dat op eenzelfde manier ervaren.

De hoofdstukindeling is als volgt:

- *Hoofdstuk 2 – Het bedrijf*
  Dit hoofdstuk gaat in op het bedrijf waar ik gedurende 17 weken mijn afstudeertraject heb gevolgd. Hierbij komen onder andere de geschiedenis, het bedrijfsprofiel en de bedrijfsvisie naar voren.

- *Hoofdstuk 3 – Opdracht en aanpak*
  Het derde hoofdstuk gaat dieper in op de opdracht die ik heb uitgevoerd tijdens het afstudeertraject. Tevens wordt gekeken naar de aanpak die ik vooraf had gedefinieerd.

- *Hoofdstuk 4 – Goudvink back-end*
  Hoofdstuk vier beschrijft de eerste fase van het afstudeertraject, de fase waarin ik mee heb ontwikkeld aan het back-end systeem van de Goudvink applicatie.

- *Hoofdstuk 5 – Onderzoek app types*
  In de tweede fase van het afstudeertraject heb ik een onderzoek verricht naar diverse app types. Het vijfde hoofdstuk gaat dieper in op de opzet en uitvoering van dit onderzoek en de daarbij behorende door mij gemaakte keuzes.

- *Hoofdstuk 6 – Prototype app*
  Dit hoofdstuk gaat dieper in op het proces wat ik heb doorlopen om een prototype van de Goudvink app te ontwikkelen.

- *Hoofdstuk 7 – Procesevaluatie*
  In het zevende hoofdstuk van dit document geef ik een evaluatie over het doorlopen proces gedurende mijn afstudeertraject.

- *Hoofdstuk 8 – Productevaluatie*
  In het achtste hoofdstuk van dit document geef ik een evaluatie over de producten die ik gedurende mijn afstudeertraject heb gemaakt.

- *Hoofdstuk 9 – Beroepstaken*
  Voorafgaand aan het afstudeertraject heb ik een aantal beroepstaken gekozen waarin ik mij zelf gedurende het afstuderen in heb bewezen. Dit hoofdstuk gaat dieper in op deze beroepstaken en hoe ik denk daaraan voldaan te hebben.

- *Bijlagen*
  Door het gehele verslag is gebruik gemaakt van meerdere documenten die gedurende dit afstudeerproject zijn gemaakt. Alle gebruikte documenten zijn opgenomen in de bijlage.

2 Het bedrijf

In dit hoofdstuk wordt 42 BV, het bedrijf waar ik mijn afstudeertraject heb gevolgd, nader beschreven.

2.1 Geschiedenis

De oprichter van 42 BV, Eric Meijer, was tot 2003 werkzaam als directeur van enkele Nederlandse bedrijfstakken van de multinational Software AG. Het hoofdkantoor van Software AG is oorspronkelijk gevestigd in Duitsland. Wegens een afnemende winst in het buitenland werd het Nederlandse kantoor gesloten, ondanks het feit dat deze nog wel winstgevend was. Eric Meijer besloot de ruimte in de markt die hierdoor vrij zou komen te benutten door een eigen bedrijf te starten.

In 2005 is 42 BV een samenwerkingsverband aangegaan met het Amsterdamse Kensas. De twee bedrijven zijn ondergebracht onder de holding genaamd M.A.D. Dit staat voor Mice and Dolphins, de twee meest intelligente diersoorten op aarde volgens the Hitchhiker's Guide to the Galaxy, het boek waar eveneens de naam 42 van afkomstig is. Tegenwoordig wordt de naam Kensas zo min mogelijk gebruikt, de naam 42 is hiervoor in de plaats gekomen. 42 BV is inmiddels uitgegroeid tot een bedrijf met 40 medewerkers. Enkelen hiervan werken op kantoor in Zoetermeer of Amsterdam, maar de meeste werken gedetacheerd bij de klant.

2.2 Bedrijfsprofiel

De missie van 42 BV is het ontwikkelen van hoogwaardige software waarmee haar klanten hun werkprocessen en bedrijfsresultaten kunnen verbeteren. 42 BV ontwikkelt maatwerk Java gebaseerde software, en doet dit het liefst open source. Verder helpt 42 BV grote klanten bij het integreren van Atlassian-producten. 42 BV gebruikt deze software zelf ook en is partner van Atlassian. Een aantal van deze producten worden later in dit verslag nog toegelicht.

Om het ontwikkelen van hoogwaardige software mogelijk te maken is 42 BV een bedrijf dat veel aandacht besteedt aan kwaliteit. Hierbij worden drie principes gehanteerd:

- De software is van lage complexiteit.
- De software is grondig getest.
- De software is vrij van conventieschendingen.

De gedachte hierachter is dat investering op het gebied van kwaliteit op de lange termijn kostenbesparend werkt, doordat de software makkelijker te onderhouden is.

2.3 Klanten

42 BV heeft na de oprichting enkele grote klanten van Software AG kunnen behouden, waaronder Schiphol en Center Parcs. Hierdoor kon 42 BV meteen beginnen met grotere opdrachten, iets wat lastig kan zijn voor beginnende bedrijven. Enkele andere (recente) grote klanten van 42 BV zijn Albert Heijn, de Consumentenbond, de Belastingdienst, ANWB, Essent en Nidera.

2.4 Bedrijfsvisie

De visie van het bedrijf is bepalend voor de keuze van aanpak van het bedrijf en is op sommige punten van mijn afstudeerproject van invloed geweest op de door mij gemaakte keuzen. Binnen de bedrijfsvisie van 42 BV is het gebruikelijk voorafgaand aan een project alle requirements op te schrijven, zonder daarbij de details te specificeren. Wanneer het project van start gaat wordt alleen de eerste requirement (globaal) besproken, waarna het projectteam aan slag gaat met ontwikkeling. Tijdens de ontwikkeling wordt alleen ontwikkeld wat nodig is om de huidige requirement werkend te krijgen, waarbij geen rekening wordt gehouden met eventuele overige requirements. Scrum sluit hier met zijn iteratieve en incrementele aanpak goed bij aan. Middels regelmatig feedback van de klant, achterhaald 42 BV of zij nog wel op het goede spoor zitten qua details van het project. Indien het misgaat, wordt door deze regelmatige feedback vroegtijdig aan de bel getrokken, waardoor de schade die in de factor tijd wordt geleden minimaal is. Deze aanpak stimuleert het klant contact warm te houden, omdat anders de details van het project niet bekend zijn. Tevens geeft dit de klant de mogelijkheid gedurende het ontwikkeltraject zijn visie op het project te veranderen. Laatst genoemde is iets wat wanneer alle details vooraf worden uitgedacht en gemodelleerd minder vanzelfsprekend is, de details zijn namelijk al bekend en het systeem kan daardoor sneller doorontwikkeld worden.

3 Opdracht en Aanpak

In dit hoofdstuk wordt de opdracht die ik bij 42 BV heb gedaan beschreven, zodat er een duidelijk beeld ontstaat over wat er gedaan moest worden. Dit wordt gedaan aan de hand van documenten die ik voor aanvang van het afstudeerproject heb opgesteld, zijnde het afstudeerplan en het plan van aanpak. Na de opdrachtomschrijving wordt de aanpak van het project toegelicht.

3.1 De opdracht

De afstudeeropdracht richt zich op het verkennen van de mobiele markt en het ontwikkelen van een declaratiesysteem, waarmee medewerkers van 42 BV gemaakte kosten voor het bedrijf kunnen declareren. Daarbij wordt voor dit declaratiesysteem een mobiele applicatie ontwikkeld. Deze opdracht is aan de hand van twee aanleidingen opgezet. Deze twee aanleidingen worden besproken in de volgende paragraaf.

3.1.1 Aanleiding

Er zijn twee oorzaken die aanleiding geven tot het afstudeerproject zoals deze voor aanvang van de afstudeerperiode was gedefinieerd. De eerste oorzaak is dat aan 42 BV steeds vaker de vraag wordt gesteld of de software die zij ontwikkelen ook geschikt is voor mobiele platformen. De tweede oorzaak is dat 42 BV nog geen declaratiesysteem heeft voor zijn medewerkers. Momenteel gebeurt dit nog middels papier.

Binnen de ICT wereld vinden continu nieuwe ontwikkelingen plaats. Een sterk groeiende ontwikkeling op het moment is die van mobiel werken, waarmee in dit geval wordt bedoeld plaats- en tijdsonafhankelijk werken. Deze ontwikkeling is door de komst van smartphones en tablets sinds een aantal jaar in opkomst, waardoor voor veel mensen uitsluitend werken achter de computer tot het verleden behoort. De klanten van 42 BV zien de sterke vooruitgang in deze ontwikkelingen en daardoor rijst steeds vaker de vraag in hoeverre zij de ontwikkelde applicaties kunnen gebruiken in een mobiele omgeving. Binnen 42 BV is geen specialistische kennis over het bouwen van mobiele applicaties aanwezig, met het gevolg dat mobiele oplossingen vaak door middel van een web-applicatie worden aangeboden aan de klant. Om in de toekomst beter mee te draaien in de zogenoemde appsmarkt wilt 42 BV zich nu gaan richten op de mobiele ontwikkelingen en de obstakels die daarbij komen kijken in kaart brengen. Zo kunnen zij de klant beter van dienst zijn en in hun behoeften voorzien.

42 BV heeft vernomen dat er binnen de mobiele markt verschillende app types bestaan, met elk hun eigen ontwikkelsituatie. Voordat 42 BV aan het ontwikkelen van mobiele apps voor haar klanten begint, wil het bedrijf weten in hoeverre de ontwikkeling van een app aansluit op de ontwikkelstraat zoals deze nu wordt toegepast binnen het bedrijf. Hierbij is het bedrijf specifiek op zoek naar welk van de app types het meeste aansluit op de huidige ontwikkelstraat. Deze ontwikkelstraat is nog niet eerder in kaart gebracht, waardoor dit ook tot het onderzoek behoort.

Als hulpmiddel om te testen of het onderzochte app type aansluit op de ontwikkelstraat wordt gebruik gemaakt van een nieuw (intern) project, genaamd Goudvink. Dit project betreft een declaratiesysteem waarin medewerkers van 42 BV kosten die zij hebben gemaakt voor bedrijfsdoeleinden kunnen

declareren. Om het gebruik van deze applicatie voor de medewerkers te stimuleren, wilt 42 BV voor dit declaratiesysteem een mobiele app ontwikkelen. Het Goudvinkproject bestond voor aanvang van de afstudeeropdracht nog niet.

Kort samengevat is zowel de groeiende vraag vanuit de markt naar mobiele applicaties en het momenteel ontbreken van een declaratiesysteem reden voor de afstudeeropdracht.

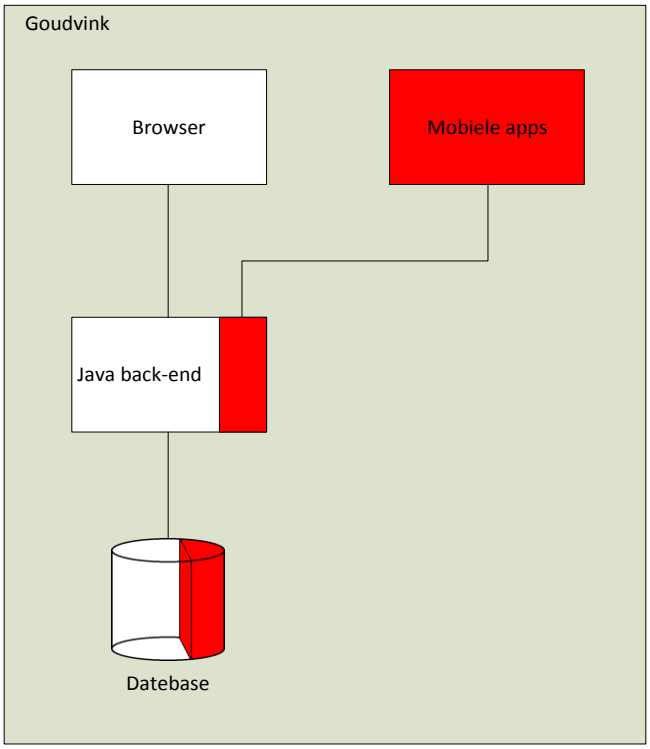
3.1.2 Doelstelling

Het primaire doel van deze afstudeeropdracht is 42 BV meer inzicht geven in de verschillende mogelijkheden binnen de appsmarkt. Dit inzicht wil het bedrijf ontwikkelen, voordat zij definitief met app ontwikkeling binnen deze markt aan de slag gaan. Hierbij is het voor het bedrijf belangrijk om te weten wat er bij komt kijken wanneer het bedrijf mobiele apps maakt voor haarzelf en haar klanten, welke mogelijkheden tot ontwikkelen er zijn en waar rekening mee moet worden gehouden. Het is vereist te achterhalen welke overeenkomsten en verschillen er zijn te vinden in de huidige ontwikkelstraat en die van de appsmarkt. Hierbij wordt gedacht aan overeenkomsten tussen gebruikte programmeertalen, nieuwe ontwikkelomgevingen en bijbehorende ontwikkelprocessen zoals het build- en distributieproces. Deze doelstelling wordt behaald door middel van een verkennend onderzoek naar app types.

Naast het meer inzicht verkrijgen in de appsmarkt wil 42 BV het verkregen inzicht op waarde toetsen, door de gekozen ontwikkelstraat van het app type toe te passen in de praktijk. Hiervoor wordt voor het Goudvink back-end systeem een mobiele app ontwikkeld.

3.1.3 Afstudeeropdracht

42 BV wilt meer inzicht verkrijgen over de diverse ontwikkelmogelijkheden binnen de appsmarkt. Om dit nieuw verkregen inzicht te toetsen in de praktijk, wilt 42 BV een mobiele app ontwikkelen waarbij gebruik wordt gemaakt van één van de onderzochte ontwikkelmogelijkheden. Deze mobiele app sluit aan op een intern project, genaamd Goudvink. Het gehele Goudvinkproject is op te delen in een aantal onderdelen, welke globaal zijn weergegeven in onderstaande afbeelding.



Binnen dit overzicht behoort enkel het rood gemarkeerde gedeelte tot de afstudeeropdracht. Hierdoor zijn binnen het gehele Goudvinkproject drie aspecten te herkennen die van belang zijn voor de afstudeeropdracht: het Java back-end systeem, de mobiele app en de database. Hier moet bij worden vermeld dat de database enkel tot het afstuderen behoort indien de overige teamleden van het Goudvinkproject hier niet toe in staat zijn. In een dergelijk geval wordt enkel het relevantie deel van de database gerealiseerd om niet de afstudeervoortgang te belemmeren.

Opvallend aan voorgaande afbeelding is dat maar een klein deel van het Java back-end onderdeel is gearceerd. Dit komt doordat de focus van deze afstudeeropdracht niet op het declaratiesysteem als geheel ligt, maar op de mobiele aspecten hiervan. Daardoor wordt tijdens deze afstudeeropdracht alleen datgene van de Java back-end gemaakt wat communicatie tussen de mobiele apps en database mogelijk maakt.

Naast het ontwikkelen van het Goudvink declaratiesysteem en bijbehorende app wordt een onderzoek gedaan naar de diverse ontwikkelmogelijkheden die bestaan om voor mobiel gebruik applicaties te ontwikkelen. Vooraf is al bekend dat deze verschillende ontwikkelmogelijkheden resulteren in specifieke

app types. Het onderzoek moet 42 BV uiteindelijk in staat stellen of het ontwikkelen van mobiele apps aansluit op de ontwikkelsituatie die momenteel binnen 42 BV gehanteerd wordt. Indien overeenkomsten bestaan tussen het ontwikkelen van mobiele apps en de huidige ontwikkelsituatie moet 42 BV in staat zijn aan de hand van het onderzoek te bepalen welk van de verschillende app types dan het meest aansluit bij het bedrijf.

Alles bij elkaar leidt dit tot een aantal werkzaamheden die volbracht worden tijdens de afstudeerperiode:

- Van het Goudvink declaratiesysteem wordt de functionaliteit ontwikkeld die nodig is om communicatie tussen de mobiele applicaties en database mogelijk te maken.
- Er wordt een verkennend onderzoek gedaan naar de verschillende app types die binnen de appsmarkt bestaan, waarna voor 42 BV een beeld ontstaat over de ontwikkelstraten van deze app types.
- Om het inzicht wat aan de hand van het onderzoek wordt verkregen is te toetsen, wordt door middel van één van de onderzochte ontwikkelmogelijkheden een prototype van de app gemaakt.

3.2 Plan van aanpak

Bij projecten wordt vaak een Plan van Aanpak opgesteld waarin zaken als opdracht details, afbakening en planning staan beschreven. Deze paragraaf gaat dieper in op hoe ik het Plan van Aanpak voor het Goudvinkproject heb opgesteld, hoe deze tot stand is gekomen, de oorspronkelijke planning en welke methodes en technieken hiervoor zijn gehanteerd.

3.2.1 Totstandkoming

Het Plan van Aanpak is in de eerste week van de afstudeerperiode opgesteld. Het document heeft als doel om een aantal zaken te verduidelijken en af te bakenen.

Zaken als de precieze aanleiding en doelstelling van het project waren mij nog niet geheel duidelijk. Daarbij leidde ik uit de opdracht omschrijving af dat het project een soort van fasering zou bevatten om de diverse problemen op te lossen. Voorafgaand en gedurende het opstellen van het Plan van Aanpak heb ik dan ook meerdere gesprekken gehad met de opdrachtgever om in te zoomen op de vragen die ik nog had. Hierbij kwam de opdrachtgever ook regelmatig met feedback. Dit heeft uiteindelijk tot een definitieve versie van het Plan van Aanpak geleid.

3.2.2 Technieken en methoden

Vanuit 42 BV worden standaard een aantal methoden en technieken voor haar werkzaamheden gehanteerd. Van deze gekozen Methoden en Technieken ben ik bekend met JIRA (projectmanagement tool) en Scrum (projectmanagementmethode). Het overzicht is als volgt:

Proces / Activiteit	*Methode / Techniek / Tool*
Projectmanagementmethode	Scrum
Projectmanagement	JIRA
Ontwikkelomgeving (voor de Goudvink back-end)	IntelliJ
Softwarekwaliteitsgarantie	Sonar
Rapportage	Confluence

42 BV werkt al een aantal jaar met deze methoden, technieken en tools, waardoor ik heb besloten mij hierbij aan te sluiten. Hieronder is per methode, techniek of tool een kleine uitleg gegeven:

Scrum

Binnen 42 BV wordt als projectmanagementmethode gebruik gemaakt van Scrum. Dit komt met name naar voren in de dagelijkse stand-ups met de diverse medewerkers die op een intern project zijn geplaatst. In deze stand-ups verteld elke medewerker zijn dag planning en tegen welke problemen hij de vorige dag is aangelopen. Tijdens het afstudeertraject is het gebruik van Scrum vooral tijdens de eerste fase naar voren gekomen. Het proces werd gedurende deze periode onderverdeeld in Sprints, korte perioden van twee weken waarin bepaalde taken werden uitgevoerd.

Sprintplanningen

Aansluitend op het Scrumproces wordt gebruik gemaakt van Sprintplanningen. Deze Sprintplanningen vinden plaats na de presentatie van een demo over de voorgaande Sprint. Deze demo wordt gepresenteerd aan de opdrachtgever en productowner. Doordat na de demo direct een Sprintplanning plaatsvindt, kan hierdoor voorafgaand aan de Sprintplanning met het oog op de besproken bedrijfsvisie dieper worden ingegaan op de details van de te maken requirement.

Aan de hand van de Sprintplanning wordt voor de Sprintbacklog bepaald aan welke user-story wordt gewerkt, wat overigens vaak de eerst volgende user-story op de productbacklog is. Vervolgens wordt de gekozen user-story opgedeeld in taken, die gezamenlijk de vooraf opgestelde requirement vormen. Aan de hand van een pokerplanning sessie wordt aan elke taak een puntenaantal toegekend. Dit puntenaantal representeert een hoeveelheid werk die verzet moet worden om de taak te volbrengen. Het puntenaantal wordt bepaald door het openleggen van een door ieder teamlid getrokken kaart. De praktijk wijst uit dat de getrokken punten in de eerste paar pokerplanning sessies erg van elkaar afwijken, doordat ieder teamlid een ander beeld heeft bij de te beoordelen taak. Om deze mis-match tussen de teamleden te voorkomen geeft ieder teamlid na het trekken van een kaart zijn idee over wat deze taak naar zijn idee precies inhoud. Dit dwingt ieder teamlid om zijn of haar idee te uiten, waarbij de discussies die hieruit voortkomen er voor zorgen dat een gezamenlijk beeld ontstaat over de taak. Door een gezamenlijk beeld samen te stellen is ieder teamlid na de pokerplanning sessie op de hoogte van wat iedere taak op de Sprintbacklog inhoud. Na de overeenstemming trekt iedereen nogmaals een kaart, waarbij de getrokken punten vrijwel gelijk aan elkaar zijn doordat het projectteam nu een gezamenlijk beeld heeft over de taak. Door het regelmatig houden van pokerplanning sessies, hoeft gedurende de Sprint veel minder tijd besteedt te worden aan het bespreken van taken. Dit verhoogt de efficiëntie van het projectteam.

Jira

Ter ondersteuning van het Scrumproces is gebruik gemaakt van projectmanagementtool Jira. Binnen Jira is de mogelijkheid tot invoeren van user stories met bijbehorende taken op een virtuele productbacklog. Na invoer worden deze taken aan projectleden van het projectteam gekoppeld, waarna zij verantwoordelijk zijn voor de uitvoering van deze taak. Doordat de productbacklog digitaal wordt bijgehouden en via het web te benaderen is, stelde dit de stakeholders van het project in staat om de voortgang van het project in te zien, zelfs wanneer zij niet op kantoor aanwezig waren. Wie deze stakeholders zijn is te vinden in ***paragraaf 3.2.2 – Rollen***.

IntelliJ

IntelliJ is de ontwikkelomgeving die voornamelijk tijdens de eerste fase van mijn afstuderen wordt gebruikt. Het betreft een commerciële IDE, ontwikkeld door JetBrains. In vergelijking met bijvoorbeeld Eclipse is standaard meer functionaliteit aanwezig binnen het pakket. Afhankelijk van het gekozen app type in de tweede fase, het onderzoek, kan worden gekeken of deze ontwikkelomgeving ook gebruikt kan worden.

Sonar

42 BV gebruikt Sonar om de code kwaliteit van de geleverde software te waarborgen. Sonar betreft een Atlassian product waarin projecten worden geïmporteerd, waarna het programma de software analyseert en afweegt tegen vooraf gedefinieerde kwaliteitscriteria. De ontwikkelaar krijgt vervolgens feedback op deze afweging door middel van overzichten. Deze kwaliteitscriteria kan aan de hand van een standaard template worden opgesteld, maar kan ook zelf worden ingevoerd of aangepast. Over het gebruik van Sonar wordt in ***hoofdstuk 4 - Goudvink*** meer informatie gegeven.

Confluence

Confluence is de omgeving waarbinnen het bedrijf informatie deelt tussen de medewerkers. Het bestaat uit meerdere delen, bijvoorbeeld de knowledge-base (technische bevindingen en project keuzes), evenement planning en bedrijfsinformatie. Alles wat voor mede collega's belangrijk kan zijn om te weten, wordt binnen deze omgeving opgeschreven en gedeeld.

3.2.3 Rollen

Tijdens het afstudeertraject worden diverse personen bij het project betrokken, waarbij de rollen als volgt zijn verdeeld:

Rol	*Persoon*	*Functie*
Opdrachtgever, Bedrijsmentor, stakeholder (architect Goudvink)	Robert Bor	Chief Technology Officer
Productowner (Goudvink), stakeholder	Eric Meijer	CEO
Developer (Goudvink)	Paulien de Wind	Software Developer
Developer (Goudvink)	Margret Kouwen	Software Developer
Scrum Master, Developer (Goudvink)	Michel Noppert	Software Developer
Afstudeerder	Wilco Stuyfersant	Software Developer
Begeleidend examinator	Nanny Jacobs	Docent aan de Haagse Hogeschool
Tweede examinator	Arno Nederend	Docent aan de Haagse Hogeschool

Paulien, Michel en Margret zijn alleen gedurende de eerste fase van het project bij het afstuderen betrokken. Zij maken deel uit van het Goudvinkproject.

3.2.4 Planning

Voordat ik aan de afstudeertaken ben begonnen heb ik een globale planning opgesteld. Hierbij heb ik ervoor gekozen de planning onder te verdelen vier in fasen, waarin per fase een bepaalde activiteit wordt verricht met een bijbehorende oplevering van een product of producten. De planning is als volgt:

Fase	*Week*	*Activiteit*
	1	Opstellen Plan van Aanpak.
1	2 t/m 4	Deel van de Java back-end van het Goudvinkproject ontwikkelen.
2	5 t/m 9	Onderzoek naar app types binnen de appsmarkt.
3	10 t/m 15	Ontwerpen en ontwikkelen van een mobiele app.
4	16 en 17	Schrijven en afronden afstudeerverslag.

Te zien is dat het hoofdgedeelte van de afstudeerperiode (week twee tot week 15) is opgedeeld in drie fasen. In ***paragraaf 3.1.3 – Afstudeeropdracht*** kwam al naar voren dat de afstudeeropdracht bestaat uit drie hoofdtaken, het uitvoeren van een onderzoek naar de verschillende app types binnen de appsmarkt, het ontwikkelen van een prototype van één van de app types en het ontwikkelen van een Java back-end server waar het prototype van de mobiele app mee kan communiceren.

Bij aanvang van de afstudeerperiode is het Goudvinkproject twee weken eerder van start gegaan. Dit project begint met het ontwikkelen van de functionaliteit om vanuit een externe bron te communiceren met de Goudvinkapplicatie. Gezien deze functionaliteit voor mij van belang is bij het ontwikkelen van de mobiele app, is besloten dat de afstudeeropdracht start met het mee ontwikkelen aan deze functionaliteit. Deze fase resulteert in een Java back-end waarin bepaalde functionaliteit, afhankelijk van de requirements, beschikbaar is om vanaf een externe bron mee te communiceren.

In de tweede fase van de afstudeeropdracht wordt het onderzoek gestart, waarbij de conclusie leidt tot het app type dat het meest aansluit op de behoeften van 42 BV. Deze fase heeft als resultaat een onderzoeksrapport.

De conclusie van het onderzoeksrapport geldt als input voor de derde fase, de fase waarin het gekozen app type wordt uitgewerkt tot een prototype van de mobiele app.

3.2.5 Risicofactoren

In de doelstelling is al naar voren gekomen dat het voornaamste doel van de afstudeeropdracht is dat 42 BV meer inzicht krijgt in de appsmarkt. Het risico hierbij is dat gedurende de eerste fase van de afstudeeropdracht teveel tijd wordt besteed aan het ontwerpen en ontwikkelen van de Goudvink back-end. De leden van het Goudvink team hebben allen aangegeven dat in verband met privé omstandigheden uitval gedurende een periode kan worden verwacht. In dit geval moet er meer werk worden verzet om een goed functionerende Java back-end te ontwikkelen, wat uiteindelijk de basis voor de mobiele app vormt. Wanneer de eerste fase uitloopt komt de hoeveelheid tijd die is vastgelegd om het onderzoek in uit te voeren in gevaar, hetgeen waar 42 BV voornamelijk in geïnteresseerd is. Afgesproken is dat afhankelijk van de situatie bepaalde functionaliteit komt te vervallen, zodat er genoeg tijd beschikbaar blijft voor het onderzoek.

Een andere belangrijke risicofactor dat ik heb opgenomen, betreft het ontwerpen en documenteren van de te ontwikkelen applicatie. Het is bij 42 BV ongebruikelijk om voorafgaand aan een project het gehele systeem tot in detail te ontwerpen, het ontwerpen wordt enkel gedaan indien dit in een complexe situatie verheldering biedt. Van de medewerkers van 42 BV wordt dan ook verwacht dat zij een groot deel van het systeem zonder ontwerp implementeren. Van een nieuwe medewerker wordt verwacht dat hij of zij zonder veel documentatie kan inlezen in een al bestaand project. Hier heb ik aan het begin van het afstudeerproject rekening mee gehouden. Aansluitend daarop worden ontwerpen pas gemaakt naarmate het project vordert. Dit wordt gedaan aan de hand van de user stories op de productbacklog. Wanneer de eerste requirement op de productbacklog wordt gemaakt is de kans groot dat de details van de requirement onderaan de backlog nog geheel onduidelijk zijn. Het risico wat ik hier bij loop is dat de ontwerpfase en ontwikkelfase dusdanig dicht op elkaar liggen, dat het risico bestaat dat ik te weinig tijd besteed aan het ontwerpen van een systeemdeel.

4 Fase 1 – Goudvink back-end

Dit hoofdstuk gaat dieper in op de afstudeeropdracht die ik bij 42 BV heb gedaan. Mijn werkzaamheden, problemen en keuzemomenten komen daarbij naar voren. Hieronder een kort overzicht van de verschillende activiteiten die worden besproken:

- *Definition of Done*
  In de eerste paragraaf van dit hoofdstuk wordt dieper ingegaan op de vooraf opgestelde Definition of Done. Hierbij wordt besproken waar de ontwikkelde requirements aan moeten voldoen alvorens deze worden gepresenteerd aan de stakeholders.

- *Bestaande deel Goudvink*
  Deze paragraaf gaat dieper in op het proces dat ik heb doorlopen om met het Goudvinkproject aan de slag te kunnen. Het gaat hier om het analyseren en begrijpen van een bestaande architectuur met bijbehorende code.

- *Goudvink requirements*
  Voor het Goudvinkproject zijn een aantal requirements opgesteld. In deze paragraaf wordt besproken waar deze requirements uit bestaan.

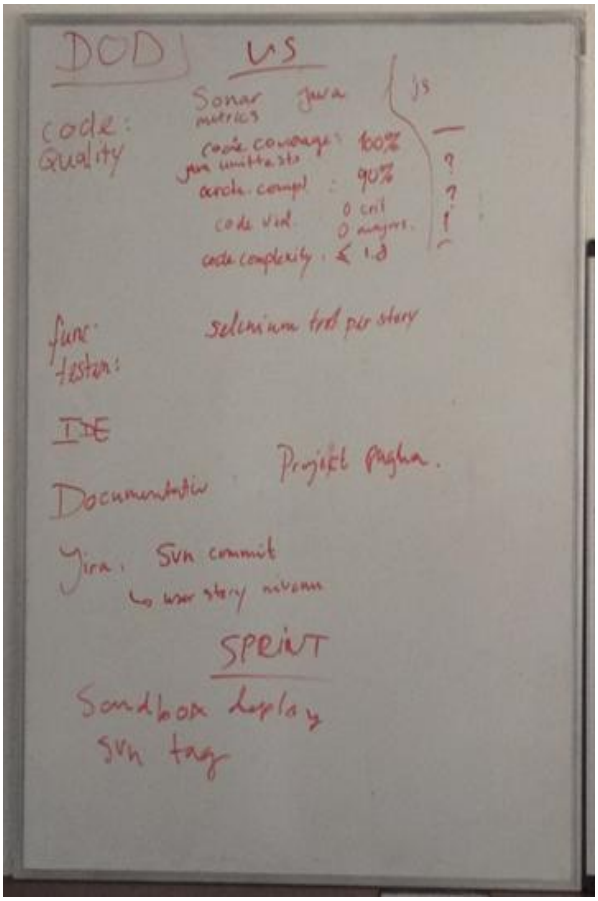
- *Sprint 1 – Invoeren uitgave*
  In de eerste sprint is gewerkt aan de eerste requirement. Hierbij heb ik continu de opgestelde Definition of Done in het achterhoofd gehouden. Deze paragraaf gaat dieper in op de door mij verrichte werkzaamheden.

- *Sprint 2 – Ontwikkelen declaratie requirement*
  Gedurende de tweede sprint is de declaratie requirement ontwikkeld. In deze paragraaf wordt dieper ingegaan op het proces dat is doorlopen tijdens deze sprint.

- *Resultaat fase één*
  Het uiteindelijke resultaat van de eerste fase komt in deze paragraaf aan bod.

4.1 Definition of Done

Voor aanvang van het Goudvinkproject is in samenspraak met de architect en productowner de Definition of Done (DoD) vastgesteld. De DoD geeft aan wanneer een deel van de applicatie klaar is voor oplevering. Deze oplevering is in de vorm van een demo, die aan het eind van elke Sprint wordt gepresenteerd. Door discussies te hebben over diverse aspecten van de code met zowel de architect en productowner is het Goudvink team tot een uiteindelijke DoD gekomen, die er als volgt uit ziet:



DoD

	Sonar Metrics & Java	*Waarde*
Code quality:	Code Coverage Java unittests	100%
	Arch. Compl.	90%
	Code Violations	0 Critical 0 Majors
	Code Complexity	≤ 1.8
Functioneel testen	Selenium test per story	
IDE		
Documentation	Project pagina	
Jira	Issues gedocumenteerd	User-story niveau
Sprint eisen	Sandbox	.war gedeployed
	SVN	Nieuwe release tag

Definition of Done – Opgesteld tijdens de eerste sprintplanning.

- *Code Coverage*

De Code Coverage verwijst naar het percentage van de methodes uit de applicatie die op een correcte werking worden getest. In eerste instantie is afgesproken om de minimale Code Coverage op 95% te stellen, wat inhoudt dat 95% van alle methoden in het project getest worden en succesvol moeten verlopen. Echter, met een afspraak van 95% kan in theorie de belangrijkste methode worden overgeslagen met testen, zonder dat dit verdere consequenties heeft voor de DoD. Gezien dit wel degelijk consequenties voor de kwaliteit van de code heeft, heb ik deze gedachte in de groep kenbaar gemaakt. Uiteindelijk is de Code Coverage op 100% gezet, met als uitzondering de methodes die niet getest kunnen worden (hetgeen waar de overige 5% in eerste instantie voor bedoeld was). Bij een Code Coverage van 100% wordt niet alleen getest op het gebruik van methoden, maar ook op de inhoud van de methoden zelf. Wanneer bijvoorbeeld een if-statement in de code wordt gebruikt, wordt verwacht dat zowel de if- als het else-statement wordt getest op correcte werking. Voor het Goudvinkproject wordt daarbij gebruik gemaakt van Java unittests. De front-end wordt getest middels een ander testing framework, Selenium. Hier ga ik verder niet op in omdat dit buiten de scope van de afstudeeropdracht valt.

- *Architecture Compliance*

Het gegeven percentage van de Architecture Compliance (Arch. Compl. 90%) wordt beïnvloed door het aantal bugs, code duplicatie en het gebruik van best practices. Wanneer er meerdere bugs of code duplicaties in de code voorkomen, gaat het Architecture Compliance percentage omlaag. Door het gebruik van best practices wordt het percentage hoger. Om deze criteria te kunnen meten wordt binnen de Sonar software gebruik gemaakt van standaard templates.

- *Code Violation*

Code Violations geven aan hoe vaak bepaalde codeconventies worden geschonden. Daarbij wordt de gradatie van schending gegeven. Zo geeft een Critical Violation aan dat de code hoogstwaarschijnlijk niet naar behoren functioneert zonder deze eerst op te lossen. Het verschil met Architecture Compliance is dat het in het geval van Code Violations ook kan gaan om de schending van een code conventie. Zo wordt het ontbreken van een spatie in de code als een minor violation gezien. Deze kwaliteitscriteria zijn door het bedrijf zelf samen te stellen, maar ook kan er gebruik worden gemaakt van standaard templates. Afgesproken is dat er per definitie geen Critical en Major violations mogen voorkomen in de code. De minor violations dienen in principe ook te worden voorkomen, maar hier ligt veel minder nadruk op. Mede door deze reden is afgesproken hier geen specifiek getal aan te hangen.

- *Code Complexity*

Zoals de naam al doet vermoeden geeft de Code Complexity aan hoe complex een bepaald stuk code is. Dit wordt gedaan aan de hand van het aantal vertakkingen per functie (bijvoorbeeld if statements, for en while loops, exception handlers). Hoe meer vertakkingen in een functie worden gebruikt, hoe complexer de code is. Uit ervaring zegt de Goudvinkarchitect dat een maximale Code Complexity van 1.8 geoorloofd is.

- *IDEdocumentatie*

Met IDEdocumentatie wordt niet verwezen naar een Integrated Development Environment, maar naar een Instant Development Experience. Deze term omschrijft dat een nieuwe ontwikkelaar in het team in staat moet zijn binnen enkele stappen aan de slag te kunnen met het project. Door middel van een subversion project check-out en een Maven commando moet het voor de ontwikkelaar mogelijk zijn zonder verdere configuraties het project te draaien. Maven is een automated build tool die binnen 42 BV veel wordt gebruikt. Door middel van het ***mvn tomcat7:run*** commando wordt een webserver opgestart waarna de gebruiker het project kan inzien. Deze eis is onder andere opgesteld zodat de productowner en opdrachtgever voorafgaand aan een Sprintdemo het project gemakkelijk kunnen inzien, waardoor zij eventuele feedback kunnen voorbereiden.

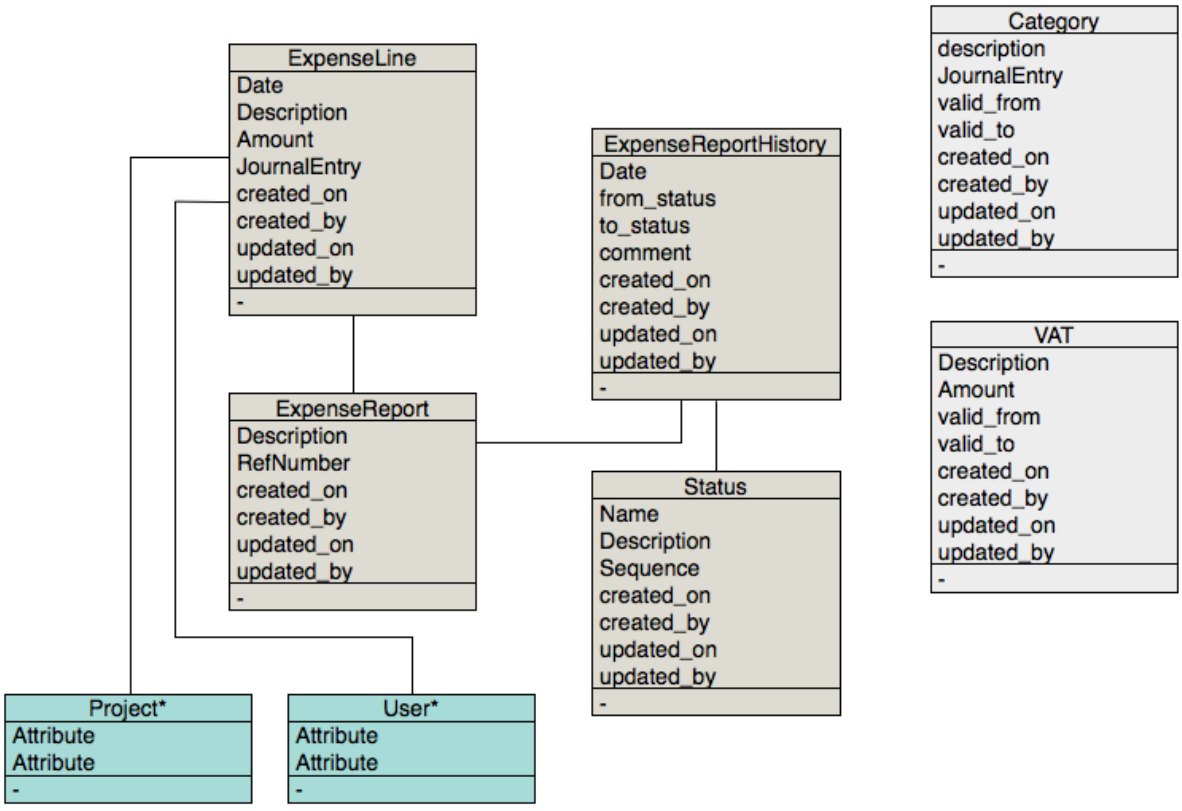
Door het naleven van de DoD wordt de kwaliteit van de code gedurende het project consequent op niveau gehouden. Opvallend aan deze DoD is dat er geen eisen worden gesteld aan de requirements van het project. Dit is typisch een gevolg van de gevolgde bedrijfsvisie, die beschreven is in ***Hoofdstuk 2.4 – Bedrijfsvisie.***

4.2 Bestaande deel van Goudvink

Bij aanvang van het afstudeertraject was het Goudvinkproject al van start gegaan. Om duidelijk te krijgen wat er tot zover was gemaakt en is afgesproken, ben ik begonnen met het oriënteren op het bestaande deel van Goudvink. Hierbij heb ik mij gefocust op zowel het deel dat al was ontwikkeld, als de requirements van Goudvink. Doordat het hier om een al bestaande architectuur en code gaat, beschrijft deze paragraaf alleen het proces van verkenning wat ik heb doorlopen om het bestaande deel tot mij te nemen.

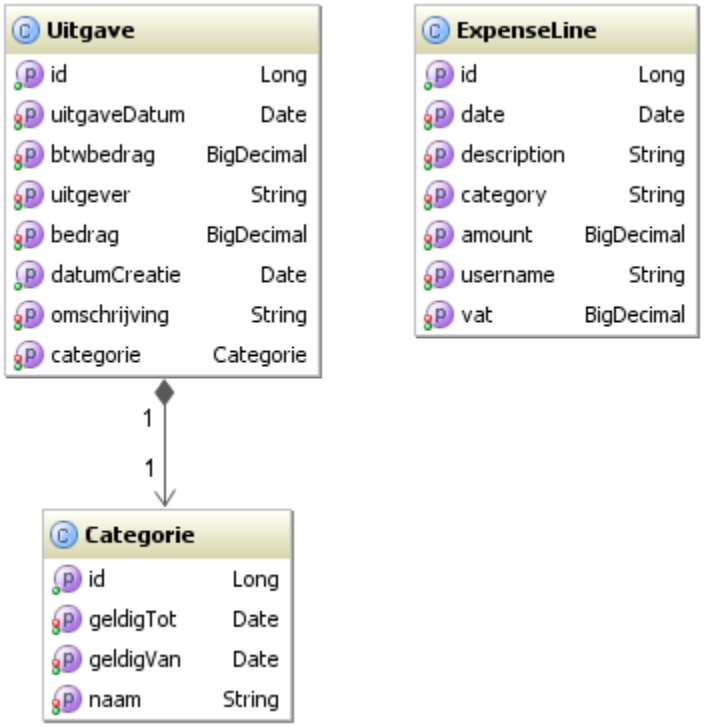
4.2.1 TODD

Via de Confluence, een applicatie waarmee binnen het bedrijf informatie wordt gedeeld, ben ik erachter gekomen dat Goudvink wordt gebaseerd op TODD, een Proof of Concept wat een jaar eerder door een medewerker van 42 BV is gemaakt. Het betrof hier een Proof of Concept om te kijken hoe het ontwikkelen van een dergelijk systeem verloopt en hoe dit er vervolgens uit komt te zien. Doordat de medewerker die hier aan heeft ontwikkeld op een extern project werd gezet, is het TODD project stil komen te liggen. Een jaar later zijn andere medewerkers gevraagd hiermee aan de slag te gaan, waarbij is besloten de applicatie vanaf scratch opnieuw te ontwikkelen. Een deel van de code uit dit project is te gebruiken voor Goudvink. Het klassendiagram wat ik van TODD tegenkwam is als volgt:



Klassendiagram TODD

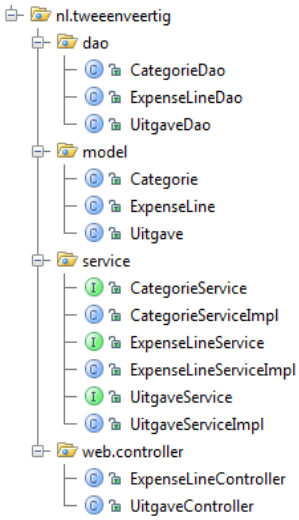
Dit klassendiagram wijkt af van wat ik binnen het Goudvinkproject vond, waardoor ik tot zover met zekerheid kon zeggen dat Goudvink geen letterlijke kopie is van TODD. Het Goudvinkproject bevatte namelijk de volgende entiteiten:



Klassendiagram Goudvink

4.2.2 Lagenarchitectuur

Tijdens de analyse van het bestaande stuk van Goudvink kwam heel sterk naar voren dat er gekozen is voor een bepaalde architectuur. Het sterkst is dit terug te zien in de mappenstructuur van het Goudvinkproject, waarin duidelijk voor een vier lagen architectuur is gekozen, bestaande uit een modellaag, een Data Access Object (DAO) laag, een servicelaag en een controllerlaag. De mappenstructuur is als volgt:



Mappenstructuur Goudvink

Om met het Goudvink project aan de slag te kunnen is het voor mij belangrijk te weten waar de lagen in deze vier lagen architectuur voor bedoeld zijn. Dit heb ik achterhaald aan de hand van de al geschreven code. De modellaag bevat alle objecten die in de applicatie gebruikt worden en weet daarbij niet van het bestaan van andere lagen af. De modellaag kende in dat stadium van het project nog geen functionaliteit, waardoor het alleen de structuur van de objecten definieert. Ik verwacht dat in de loop van het project de specifieke data-logica (uit de business logica) ook in de modellaag wordt geïmplementeerd.

De tweede laag, de DAOlaag, is verantwoordelijk voor het aanpassen van bestaande objecten, waarbij deze momenteel alleen bestaat uit CRUD (Create, Read, Update, Delete) functionaliteit. In het begin stadium van Goudvink doet de derde laag, de servicelaag, niets meer dan het doorsturen en afhandelen van objectwijzigingen die doorkomen van de laatste laag, de controllerlaag. In de servicelaag zit dus geen functionaliteit, wat de laag zoals deze nu is geen extra waarde laat toevoegen. Toen ik hier naar vroeg bij een collega bleek dat hier wel over is nagedacht en dat de servicelaag in een later stadium meer code bevat. De servicelaag krijgt meer functionaliteit wanneer een object meer functionaliteit behoeft dan alleen het wijzigen van het object zelf. De DAOlaag heeft dus met opzet alleen maar CRUD functionaliteit. Deze extra functionaliteit wordt ook wel de transactiescripts van de business logica genoemd. Gezien dit soort functionaliteit in het begin stadium van Goudvink nog niet van toepassing was, kwam dit niet duidelijk terug in de structuur van het project. Gezegd is al dat naast dat de servicelaag verantwoordelijk is voor functionaliteiten anders dan het wijzigen van objecten, de laag

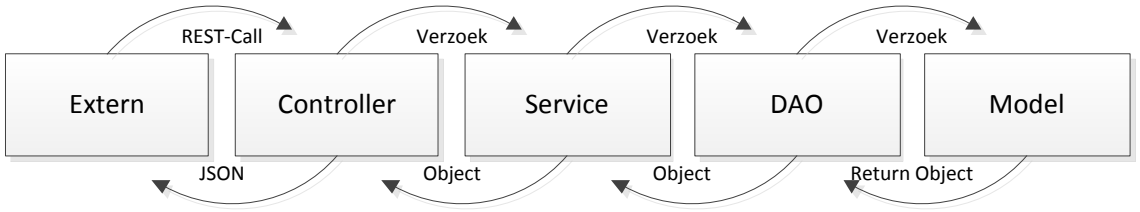
tevens verantwoordelijk is voor het doorsturen en afhandelen van object wijzigingen die worden gestuurd vanaf de controllerlaag. Deze controllerlaag is verantwoordelijk voor het afhandelen van aanvragen die van buiten het systeem worden gestuurd. Gezien ik in de derde fase van mijn afstuderen moet communiceren met de Java back-end van Goudvink, is het voor mij interessant meer te weten te komen over deze vierde laag. Deze laag ben ik dan ook verder gaan analyseren.

Voordat ik hier op in ga, eerst een overzicht. De voorgaand benoemde architectuur geeft een duidelijke scheiding tussen verschillende lagen, waarbij met name over de verantwoordelijkheden per laag is gesproken. Het overzicht van de verantwoordelijkheden binnen deze architectuur is als volgt:

Model	De modellaag definieert de structuur van gebruikte data. Tevens bevat deze laag de specifieke data logica.
DAO	De DAO laag is verantwoordelijk voor het uitvoeren van de CRUD verzoeken op de objecten.
Service	Indien een object meer functionaliteit behoeft dan de CRUD functionaliteit van de DAOlaag, wordt dit geïmplementeerd in de servicelaag. Deze laag is tevens verantwoordelijk voor het afhandelen van verzoeken vanuit de controllerlaag en is daardoor de kern van de applicatie.
Controller	De controllerlaag is verantwoordelijk voor het opvangen en afhandelen van verzoeken die van buiten het systeem komen.

4.2.3 Verloop REST-Call

Zoals gezegd is de controllerlaag verantwoordelijk voor het opvangen en afhandelen van verzoeken die van buiten het systeem worden aangevraagd. Ik vond het belangrijk meer over het verloop van een dergelijke aanvraag te weten te komen, omdat ik hier in de derde fase van mijn afstudeeropdracht nog mee te maken ga krijgen. In eerste instantie dacht ik dat het verloop van een aanvraag als volgt verliep:



Verloop REST-Call (eerste gedachte)

Te zien is dat de externe bron door middel van een REST-Call een wijziging aan een object wil doorvoeren. REST (REpresentational State Transfer) is een vorm van software architectuur bedoeld voor gebruik bij web services. Een web service is een methode van communicatie ontwikkeld om verschillende apparaten met elkaar te laten communiceren over een netwerk. REST stelt de ontwikkelaar in staat door middel van http methoden (POST, GET, PUT, DELETE) en URL parameters bepaalde informatie van een server op te vragen of te wijzigen. Op het internet las ik bronnen die

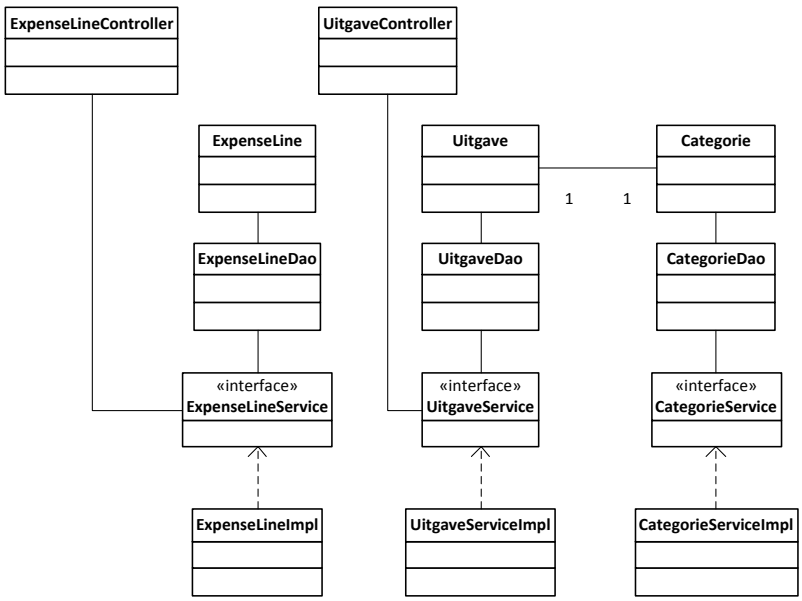
stelden dat REST ideaal werkt in web services waarbij veel systeem resources benaderd moeten worden. Dit is ook het geval bij Goudvink, wat het dus geen vreemde keuze maakt.

Een belangrijk detail wat ik met name voor mijn eigen opdracht opmerkte, is dat de REST-Calls gebruik maken van http methoden. Gezien http in principe de basis vormt voor data communicatie binnen de web wereld, biedt iedere webbrowser hier ondersteuning voor. De kans is groot dat ik hier in de laatste fase van mijn afstuderen een voordeel mee behaal, omdat het http protocol heel universeel is en dus waarschijnlijk ook werkt op mobiele apparaten.

Terug naar het REST-Call verloop, door middel van de controller en servicelaag wordt de DAOlaag verzocht een object te wijzigen, waarna het object wordt teruggegeven. Eenmaal terug bij de controllerlaag wordt het object omgezet in een JSON-object (JavaScript Object Notation), waarna deze wordt teruggestuurd aan de externe bron. JSON wordt door de simpliciteit en snelheid van de datastructuur tegenwoordig veel gebruikt om data over het web sturen, waardoor deze datastructuur ook door het grote gros van de webbrowsers wordt ondersteund. Waarschijnlijk levert dit mij, net als het gebruik van REST-Calls, mij in de derde fase van het afstudeerproject een voordeel op omdat ook deze universele ondersteuning kent.

4.2.4 Relaties

Doordat ik aan de hand van de mappenstructuur niet duidelijk de onderliggende relaties tussen de verschillende klassen kon zien, maar dit beeld nodig heb om aan de slag te kunnen met ontwikkelen voor Goudvink, besloot ik de relaties in kaart te brengen door middel van een high level klassendiagram. Zonder dit inzicht weet ik niet aan welke klassen ik ongestoord aanpassingen kan maken, zonder dat andere delen van het systeem daardoor niet meer werken. Op maken van een dergelijk diagram stelde mij tevens in staat de gekozen architectuur beter te begrijpen.

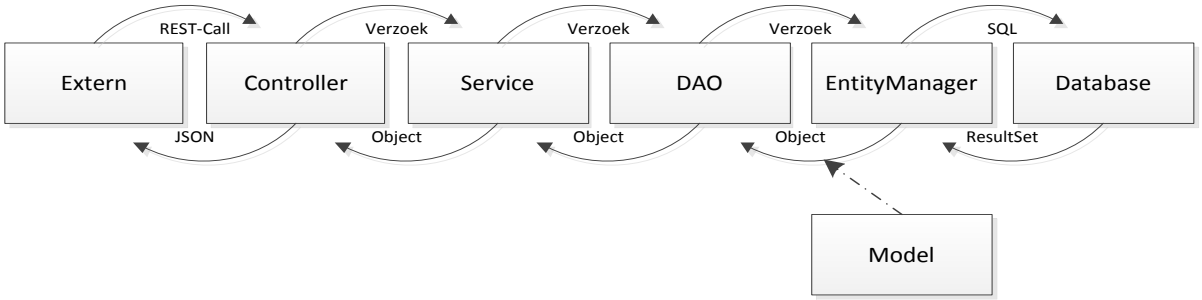


High level klassendiagram Goudvink

Uit dit klassendiagram kon ik opmaken dat het systeem uit drie onderdelen bestaat, te weten het uitgave deel, het expenseline deel en het categorie deel. Deze drie delen staan, op de uitgave en categorie entiteiten na, los van elkaar en kennen daardoor dus geen gezamenlijke functionaliteit.

Door de relaties tussen de verschillende klassen in kaart te brengen liep ik ook tegen een ander detail in het project aan, wat mij eerder nog niet was opgevallen. Dit detail ontkracht mijn eerdere statement over het REST-Call verloop. Elke DAO klasse maakt veelvuldig gebruik van een EntityManager object. Dit object is afkomstig uit de Java Persistence API (JPA) en heeft als doel het opslaan en verkrijgen van applicatie data. Dat Goudvink gebruik maakt van JPA is ook terug te zien in het gebruik van annotaties van dit framework. Zo is bijvoorbeeld elke entiteit met @Entity aangeduid (waarna deze worden gepersisteerd). Na hier wat meer op doorgezocht te hebben blijkt dat JPA de afhandeling van CRUD functionaliteiten binnen Goudvink verzorgd en is daarbij tevens zo geconfigureerd dat ook de databasewijzigingen worden afgehandeld. Hierdoor was mijn eerdere bevinding onjuist, waarbij ik dacht dat de DAOlaag alle CRUD functionaliteit van de entiteiten binnen de modellaag afhandelde. Dit doet de DAOlaag wel, maar maakt daarbij gebruik van de EntityManager, die dus tussen beide lagen is gelokaliseerd.

Doordat de DAO klassen gebruik maken van de EntityManager om de objecten in de modellaag te wijzigen, hebben zij geen directe connectie meer met laatst genoemde laag. Dit resulteerde in een ander verloop van de Rest-Call, waarbij de EntityManager ook een rol speelt.



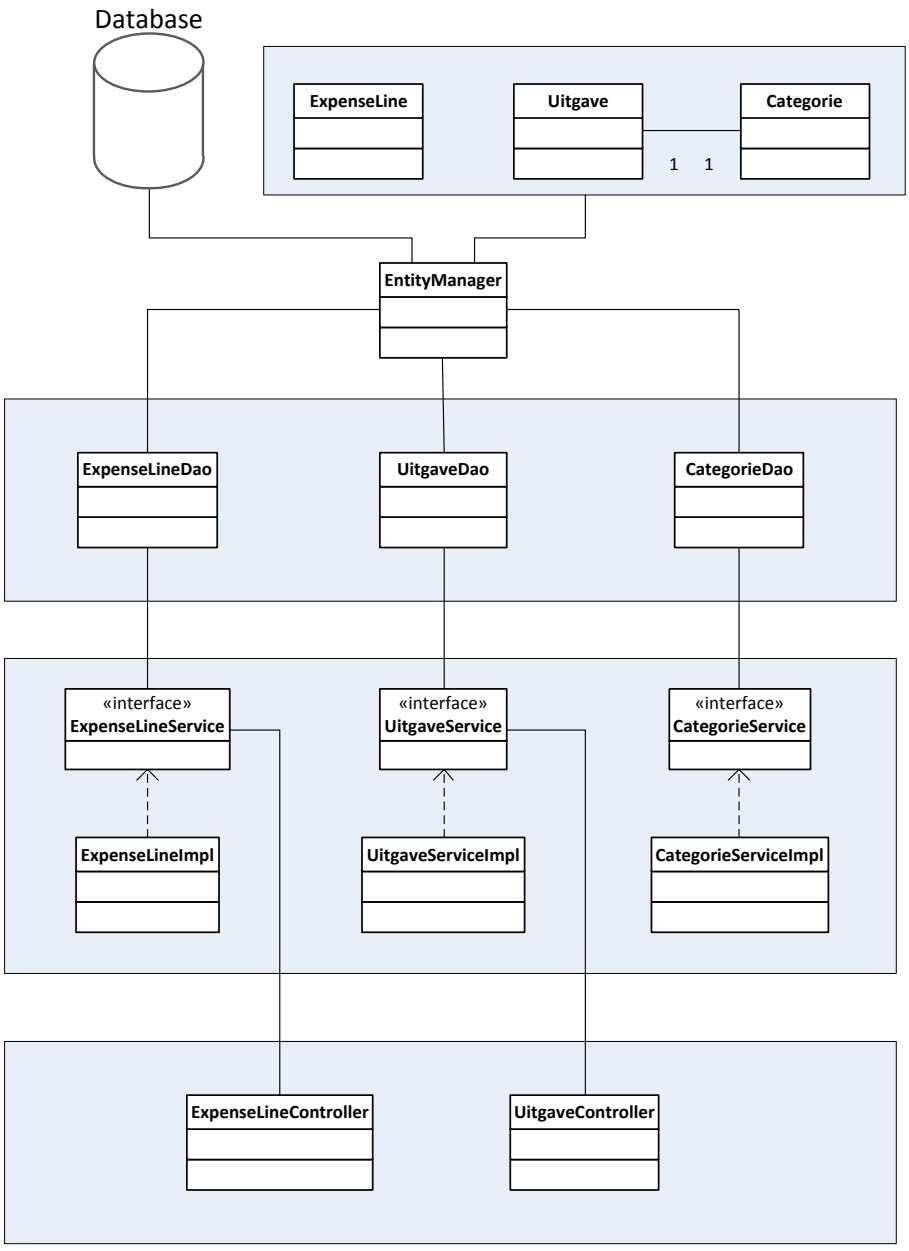
Verloop REST-Call

Te zien is dat de DAO via de EntityManager een wijziging aan een object maakt, waarna de EntityManager dit door middel van een SQL script doorvoert in de database. Na het terugkrijgen van de ResultSet van de database maakt de EntityManager gebruik van het model uit de modellaag om de ResultSet te mappen naar een object. Hierbij maakt de EntityManager gebruik van de eerder genoemde JPA annotatie op de verschillende klassen, namelijk @Entity. Vervolgens wordt het object teruggegeven tot aan de controllerlaag, die van het object weer een JSON-object maakt.

Het feit dat gebruik wordt gemaakt van de EntityManager lijkt een klein verschil, maar is in werkelijkheid een groot verschil voor de applicatie. Hierdoor is bijvoorbeeld geen SQL code binnen het project te vinden. Het voordeel van een laag als deze is dat de applicatie onafhankelijk is van het type database wat wordt gebruikt. In het geval van de EntityManager, afkomstig uit JPA, is het namelijk zo dat deze

meerdere SQL-dialecten spreekt waardoor dit framework met bijvoorbeeld zowel een PostgreSQL database kan werken, als een Microsoft SQL server.

Doordat nu gebruik wordt gemaakt van de EntityManager moest ook het voorgaande klassendiagram worden herzien. Hierbij heb ik tevens blokken toegevoegd (de blauwe kaders) welke de vier architectuurlagen representeren. Het gewijzigde high level klassendiagram is als volgt:

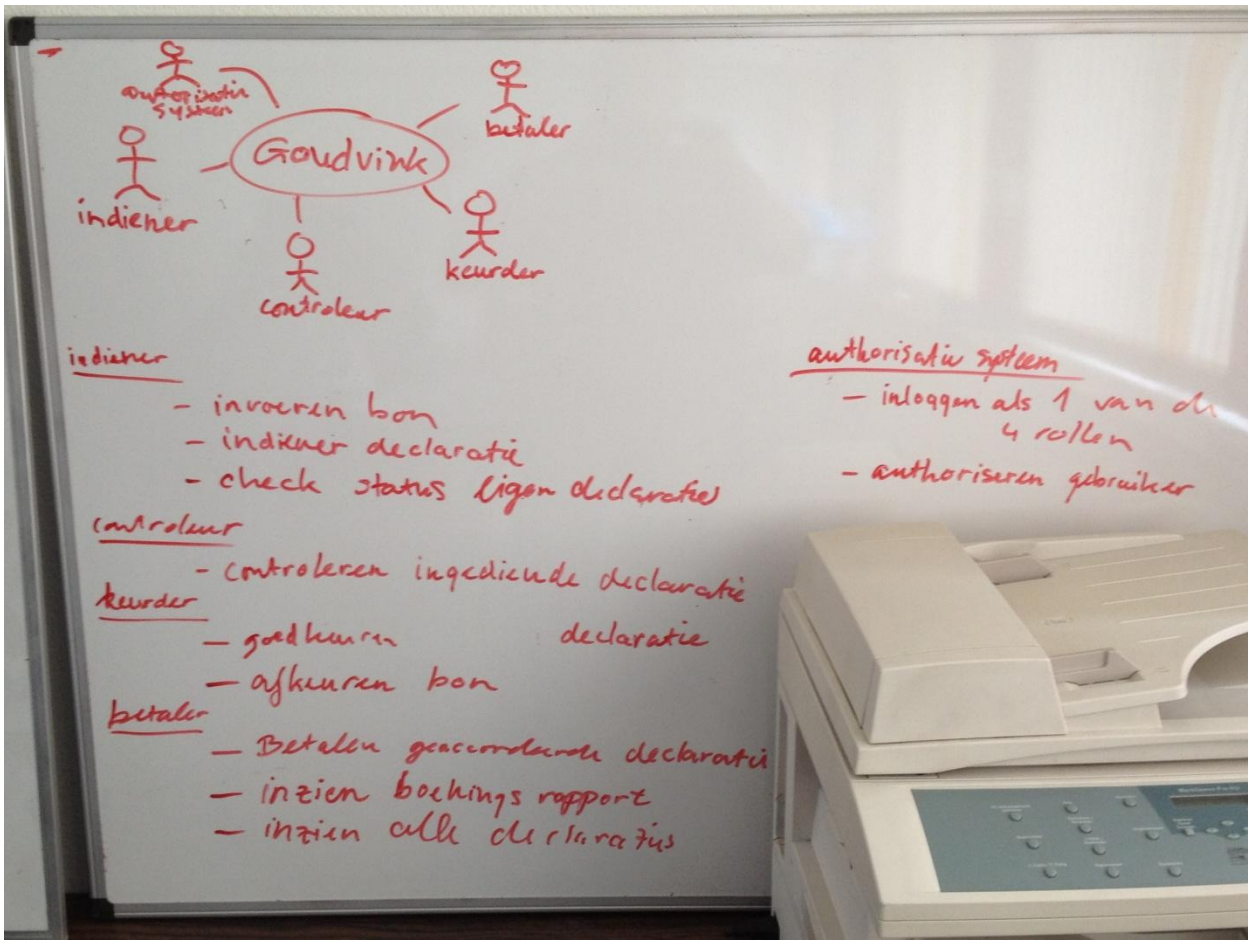


High level klassendiagram Goudvink

Voor mij was het nu duidelijk hoe de gekozen architectuur en het bestaande deel van Goudvink in elkaar zat. Wat mij echter nog niet duidelijk was, is waarom het project uit drie verschillende delen bestaat. Hierdoor was het voor mij zaak te achterhalen wat het doel is van de verschillende delen van de applicatie. In andere woorden, om zelf aan het project mee te helpen moest ik de requirements van het Goudvinkproject zien te achterhalen. Hiervoor ben ik in gesprek gegaan met de overige teamleden van het Goudvinkproject. Meer hierover in de volgende paragraaf.

4.3 Goudvink Requirements

In de eerste stakeholdervergadering, waar ik niet bij aanwezig ben geweest, zijn de requirements van het Goudvinkproject vastgesteld. Door middel van een gesprek met de opdrachtgever en productowner hebben de overige leden van het projectteam een aantal requirements achterhaald, waarna deze zijn opgeschreven op een whiteboard. De requirements zijn als volgt:



Goudvink requirements – Opgesteld tijdens de eerste stakeholdervergadering

Gezien ik niet bij deze eerste stakeholder vergadering aanwezig ben geweest, ben ik om te achterhalen waar ik gedurende de eerste fase van het afstudeerproject naar toe werk in gesprek gegaan met het Goudvink team. Zij vertelden mij dat gedurende de eerste fase van mijn afstudeertraject waarschijnlijk de eerste drie requirements van de lijst worden geïmplementeerd. Deze requirements zijn als volgt:

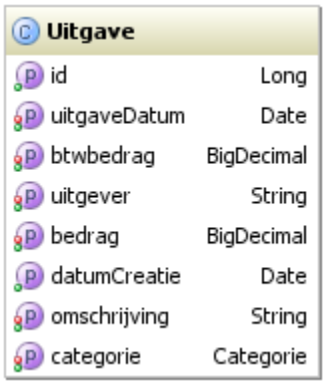
- Invoeren bon (invoeren uitgave)
- Indienen declaratie
- Check status eigen declaratie

Om het project zo goed mogelijk bij het communicatiedomein van de klant aan te laten sluiten is ervoor gekozen alle domeinnamen in het Nederlands te houden. Gezien de vreemde situaties die dit oplevert met bijvoorbeeld methodenamen (die dan half Engels, half Nederlands zijn) is dit een punt waarvan ik liever had gezien dat de domeinnamen ook in het Engels waren vertaald. Toch bleef het een specifieke eis van de stakeholders om de Nederlandse domeinnamen aan te houden.

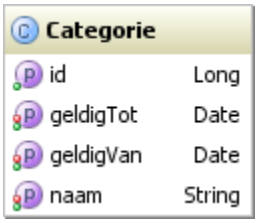
Invoeren bon

Het doel van de Goudvink applicatie is het declareren van kosten die medewerkers van 42 BV hebben gemaakt voor bedrijfsdoeleinden digitaal te laten verlopen. Deze kosten werden in het begin stadium van het Goudvinkproject ook wel *bonnetjes* genoemd. Echter bleek dit geen handige naamkeuze te zijn, waardoor is overgestapt op *uitgaven*. Het invoeren van een *uitgave* betreft het aanmaken van een nieuw *uitgave*-object, waarna deze wordt opgeslagen in de database.

Eerder is al aan bod gekomen dat een *uitgave* object uit de volgende attributen bestaat:



Voor de *categorie* is gekozen dit als een losstaand object te implementeren. De reden hiervoor is dat een aantal categorieën alleen gedurende een bepaalde periode geldig zijn. Dit is terug te zien in de klasse *categorie*, die de datumvelden *geldigVan* en *geldigTot* bevat:



De functionaliteit *invoeren uitgave* gaat in de productieomgeving voornamelijk gebruikt worden door de gebruiker met de rol *indiener*. Binnen het Goudvink systeem worden vier gebruikers rollen gedefinieerd, namelijk de *indiener*, *controleur*, *keurder* en *betaler*. Hier wordt echter gedurende de eerste fase van mijn afstuderen geen rekening mee gehouden, het gehele rollen systeem wordt achterwegen gelaten. Dit heeft te maken met de bedrijfsvisie. Besproken is namelijk dat er minimale kennis is over (opvolgende) requirements en alleen hetgeen wordt ontwikkeld wat noodzakelijk is om het huidige requirement werkend te krijgen. Het invoeren van een uitgave betreft maar één rol, namelijk die van

indiener, waardoor in de eerste paar Sprints geen rekening wordt gehouden met het bestaan van andere rollen. Pas wanneer een requirement wordt geïmplementeerd die een andere rol dan *indiener* heeft, wordt het rollen systeem geïmplementeerd.

Voor deze requirement moet een gebruiker een *uitgave* in kunnen voeren in het systeem, waarbij niet alle attributen van belang zijn. Zo worden de attributen *ID, uitgever* en *datumCreatie* niet getoond aan de gebruiker, maar worden deze automatisch bepaald of gegenereerd. Naast dat de functionaliteit voor het aanmaken van een nieuwe *uitgave* ontwikkeld wordt, moet de gebruiker ook in staat zijn *uitgaven* aan te passen (CRUD functionaliteit) en een overzicht van zijn *uitgaven* kunnen opvragen.

Indienen declaratie

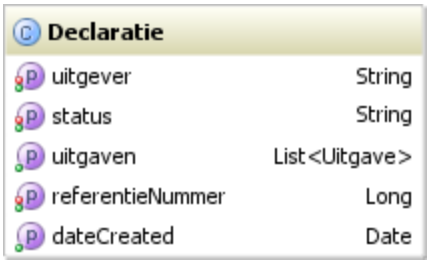
Voor de tweede requirement moet de gebruiker in staat zijn om een *declaratie* in te dienen. In het voorgaande TODD project werd een *declaratie* ook wel *expenseline* genoemd (zie ***Paragraaf 4.2.1 - TODD***). Eerder was echter al afgesproken de domeinnamen in het Nederlands te houden, waardoor *expenseline* wordt veranderd naar *declaratie*.

Een declaratie is een verzameling van uitgaven, waarvan de gebruiker definitief heeft besloten deze te declareren. De attributen zijn als volgt:

Wanneer de gebruiker een aantal *uitgaven* heeft ingevoerd kan hij besluiten hier een *declaratie* van te maken. Het systeem maakt vervolgens een nieuwe *declaratie* aan, waaraan de niet gekoppelde *uitgaven* worden gekoppeld. De gebruiker hoeft hier verder geen gegevens voor in te voeren, alleen de knop *declaratie indienen* te activeren. Het blijkt dat de gemaakte *expenseline* functionaliteit uit de TODD applicatie dusdanig afwijkt van de gestelde requirements, dat het beter lijkt deze functionaliteit opnieuw te ontwikkelen. *Expenseline* is daarom uit de Goudvink applicatie gehaald, waardoor alleen het *uitgave* deel overblijft. Ook het *categorie* deel wordt opnieuw geïmplementeerd. Naast het aanmaken van de declaratie kan de gebruiker een declaratieoverzicht tonen.

Check status eigen declaratie

Naast dat de gebruiker in staat moet zijn een declaratie in te dienen, moet hij tevens in staat zijn de status van deze declaratie te controleren. Hierdoor krijgt het declaratie object tegen de tijd dat dit requirement wordt geïmplementeerd een nieuw attribuut, *status*:



Het *status* attribuut kan, door het ontbreken van rollen in het Goudvinkproject bij de implementatie hiervan, alleen de waarde *ingediend* hebben.

Zoals te lezen zijn over deze laatste requirement al een stuk minder details bekend zijn dan van de eerste requirement, invoeren uitgave. Dit heeft alles te maken met de eerder uitgelegde bedrijfsvisie, waarin zo snel mogelijk met de ontwikkeling van het eerste deel van de applicatie wordt begonnen. Binnen de bedrijfsvisie wordt voorafgaand aan iedere Sprintplanning dieper ingegaan op de te ontwikkelen requirement.

Totaal overzicht requirements

Voor eigen referentie heb ik een totaal overzicht van de requirements die op dat moment bekend waren gemaakt.

Systeemdeel	*Requirement*
Invoeren bon	Het systeem kan een nieuw uitgaveobject aanmaken.
	Er kan CRUD-functionaliteit worden toegepast op uitgaveobjecten.
	Het systeem is in staat uitgaveobjecten op te slaan in de database. Wijzigingen moeten tevens doorgevoerd kunnen worden.
	Het systeem kan een overzicht van uitgaven tonen.
Indienen declaratie	Het systeem kan een nieuwe declaratie indienen, waarbij alle niet gekoppelde uitgaven worden gekoppeld.
	Het systeem kan een declaratieoverzicht tonen
Check status eigen declaratie	Declaraties moeten gecontroleerd kunnen worden op status.
	Status kan voorlopig alleen de status ingediend hebben.

4.4 Sprint 1 – Invoeren uitgave

Tijdens de Sprintplanning is afgesproken dat de eerste requirement, het invoeren van een uitgave wordt ontwikkeld. In de Proof of Concept variant van het Goudvinkproject, het eerder genoemde TODD, is al gekeken naar deze functionaliteit. Het voordeel wat het team hierbij heeft is dat een deel van de benodigde code al is ontwikkeld tijdens dat project. Afgesproken is delen van deze ontwikkelde code te gebruiken in het Goudvinkproject. Daarbij moet wel gelet worden op dat de gemaakte code ook voldoet aan de vooraf opgestelde Definition of Done (DoD). Voor deze Sprint ben ik verantwoordelijk gesteld voor het behalen van het Code Quality onderdeel van de DoD. Dit houdt in dat alle code getest moet zijn, er geen critical en major violations in het project zitten en dat de code complexiteit beperkt is. Dit sluit goed aan op mijn afstudeeropdracht, omdat het voor de derde fase van mijn afstuderen belangrijk is dat ik ervan uit kan gaan dat de back-end van Goudvink correct functioneert.

Ik heb besloten met het eerste item te beginnen, namelijk de Code Coverage. Voor de DoD is afgesproken dat de Code Coverage 100% moet zijn, wat inhoud dat 100% van de methoden getest moeten zijn. Hierbij moeten de tests succesvol verlopen, alvorens een deel van de applicatie opgeleverd kan worden. Het testen van code heeft als doel de kwaliteit van de code inzichtelijk te krijgen. Op basis van de bevindingen worden vervolgens verbeteringen uitgevoerd. De kwaliteit van code kan op verschillende manieren geïnterpreteerd worden, kwaliteit kan namelijk zowel snelheid, als betrouwbaarheid, als juistheid van de code betekenen. In het geval van Goudvink wordt gekeken naar of de methodes correct functioneren. Hierbij wordt dus gekeken of de methoden het gewenste resultaat geven, wat ik heb bepaald aan de hand van de opgestelde requirements. Op de vraag welke code ik ga testen was maar één antwoord mogelijk, namelijk alles.

4.4.1 Test niveau

Alvorens ik aan de slag kon met testen van de code, heb ik bepaald op welk niveau van de code ik ga testen. Hier heb ik twee keuzen, of ik test vanaf de servicelaag gelijk alle onderliggende lagen, of ik test de diverse lagen los van elkaar. Voor beide manieren van testen zijn voor en nadelen te benoemen en een keuze is van invloed op de aanpak van het testen.

Wanneer besloten wordt vanaf de servicelaag alle onderliggende lagen mee te testen, houdt dit in dat wanneer een onderdeel vanaf deze laag wordt getest, tevens de DAOlaag en modellaag worden mee getest. Als voorbeeld geef ik de test of het aanmaken van een uitgave-object goed werkt. Wanneer ik via de servicelaag test of ik het uitgave object op de juiste manier kan aanmaken, test dit eveneens of de DAOlaag deze actie goed aanroept via de EntityManager en of de modellaag de eigenschappen van het object wel op de juiste manier toekend. Hierbij moet voorafgaand aan het testen extra aandacht worden besteedt aan de diverse paden die doorlopen kunnen worden binnen de applicatie, zodat elk mogelijk testscenario wordt getest. Het voordeel van deze manier van testen is dat er minder tests benodigd zijn om de gehele applicatie te testen. Het nadeel is dat de tests meerdere units van de code tegelijk test, waarbij de tests qua omvang groter worden. Dit heeft een aantal zaken tot gevolg. Zo is de kans dat er fouten in de tests zitten groter en kan het onduidelijk zijn in welke laag het probleem zich precies voordoet mocht een test falen.

Wanneer alle drie de lagen apart worden getest, wordt binnen de modellaag getest of bijvoorbeeld een nieuwe uitgave zijn eigenschappen wel op de juiste manier krijgt toegekend en wordt de DAOlaag getest op het correct aanroepen van deze aanmaak functie. Doordat de servicelaag de DAOlaag aanroept, wordt op de servicelaag getest of dit het aanroepen van deze functie wel correct functioneert. Het voordeel van deze manier van testen is dat er per unit in de applicatiecode wordt gekeken of deze correct functioneert. Daarbij zijn de tests kort (in veel gevallen enkele regels). Het nadeel is echter dat er veel meer tests geschreven moeten worden, waarbij in sommige gevallen de vraag kan ontstaan in hoeverre deze tests de code kwaliteit nu daadwerkelijk verhogen. Dit laatste licht ik toe aan de hand van een code voorbeeld. Gegeven zijn de onderstaande methoden:

```
public BigDecimal getBedrag() {
    return bedrag;
}

public void setBedrag(BigDecimal bedrag) {
    this.bedrag = bedrag;
}
```

Wanneer ik de *Get* en *Set* methoden van het attribuut *Bedrag* wil testen, doe ik dat door middel van de *Set* functie van *bedrag* een waarde te koppelen. Vervolgens test ik door middel van de *compateTo()* functie van *BigDecimal* en de *getBedrag()* functie of het verschil in bedrag wel de verwachte nul is.

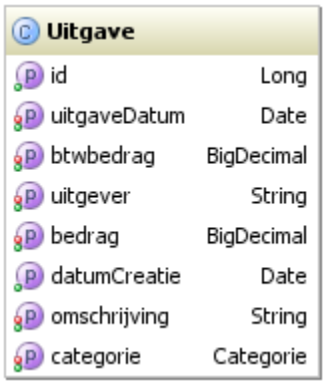
```
@Test
public void testBedragGetSet(){
    Uitgave testedUitgave = new Uitgave();
    testedUitgave.setBedrag(new BigDecimal("9.10"));
    assertTrue(new BigDecimal("9.10").compareTo(testedUitgave.getBedrag()) == 0);
}
```

In principe wordt in deze test weinig functioneels getest, waardoor de vraag kan ontstaan of een dergelijke test nu wel de code kwaliteit verhoogd. In het geval waarbij de drie lagen los van elkaar worden getest zijn dit soort tests wel nodig om aan de eerder opgestelde DoD te voldoen.

Ondanks de discussiepunten die hierover kunnen ontstaan heb ik toch besloten voor de tweede optie te kiezen, de optie waarbij ik elke laag los test. Hiervoor zijn enkele redenen te benoemen. In het begin stadium van het testproces klopt het inderdaad dat ik door de keuze van deze test aanpak veel meer tests moet schrijven. Het voordeel wat ik hier echter mee behaal is dat ik het gehele testproces voor mijzelf en komende testers (mede als mijn projectleden) overzichtelijk houdt. Doordat elke unit loststaand wordt getest, is het in het geval dat een test faalt vrijwel direct duidelijk waar de fout zich voor doet. Wanneer in een later stadium in de basis van de code van Goudvink iets wordt aangepast, dan kan door middel van deze manier van testen heel specifiek worden gezegd op welke andere delen van de applicatie code dit invloed heeft. Naast het feit dat hierdoor de tests dus overzichtelijker worden, is de kans dat een testscenario wordt overgeslagen kleiner dan dat bij de eerste optie, de optie waarin via de servicelaag de overige lagen worden getest.

4.4.2 Testen

Gezien ik had besloten om de drie lagen losstaand van elkaar te testen, ben ik begonnen met het testen van de modellaag. Deze laag is de basis van de applicatie, waardoor het zo is dat wanneer deze laag niet goed functioneert, de overige lagen dat ook niet doen omdat die gebruik maken van de modellaag. Het *uitgave* object was in dit stadium het enige object in de modellaag. De enige functionaliteit die dit model tot nu toe bevatte was de *constructor* en de *Get* en *Set* methoden van de diverse attributen. Aan de hand van de requirements ben ik voor consistentie nogmaals de attributen doorgelopen.



Het *categorie* object is eerder deze sprint verwijderd (zie ***Paragraaf 4.3 – Goudvink Requirements***), waardoor de klasse niet meer klopte met wat er voor de requirements was opgesteld. Hiertoe besloot ik, de bedrijfsvisie volgend, een hot fix uit te voeren. Daarbij heb ik *categorie* in eerste instantie niet als volwaardig losstaand object geïmplementeerd, maar als een enum. Dit heb ik gedaan omdat de *categorie* implementatie met een *geldigVan* en *geldigTot* datumveld nog niet van toepassing is bij het invoeren van een *uitgave*. Hier komt de Scrumaanpak en bedrijfsvisie van toepassing doordat wordt gezegd alleen datgeen te maken wat nodig is om de huidige requirement werkend te krijgen, de overige details komen later. Nadat ik dit had doorgevoerd heb ik de eerste test voor de applicatie geschreven. De belangrijkste test voor dit deel van de applicatie is of de uitgave wel op een correcte manier wordt aangemaakt:

```
@Test
public void testUitgaveCreation(){
    Date date = parseDate("2012/01/01");

    Uitgave testedUitgave = new Uitgave(date, "iets", new BigDecimal("9.10"), new BigDecimal("1.73"), Categorie.BRANDSTOFKOSTEN);


    assertEquals(date, testedUitgave.getUitgaveDatum());
    assertTrue(new BigDecimal("9.10").compareTo(testedUitgave.getBedrag()) == 0);
    assertTrue(new BigDecimal("1.73").compareTo(testedUitgave.getBtwbedrag()) == 0);
    assertEquals("iets", testedUitgave.getOmschrijving());
    assertEquals(Categorie.BRANDSTOFKOSTEN, testedUitgave.getCategorie());

    assertNull(testedUitgave.getId());
    assertNull(testedUitgave.getDateCreated());
}
```

Bij de voorgaande requirements is besproken dat de gebruiker van Goudvink een aantal attributen niet zelf hoeft in te voeren, ter herhaling het *ID*, de *uitgever* en de *dateCreated*. Dit simuleer ik door alleen de benodigde parameters aan de constructor van *uitgave* mee te geven. De datum die wordt

geïnstantieerd in de derde regel is de *uitgaveDatum* van de *uitgave*. Wanneer het object eenmaal geïnstantieerd is, worden de ingevulde waarden op correctheid gecontroleerd door deze te vergelijken met de eerder ingevoerde waarden. Te zien is dat ik aan het eind van deze test controleer of het *ID* en *dateCreated* wel NULL zijn. De reden daarvoor is dat bij het aanmaken van het object deze waarden automatisch gegenereerd worden. Dit gebeurd echter pas bij het wegschrijven van het object naar de database, waardoor de waardes in eerste instantie NULL zijn. Het automatisch genereren van deze waardes is een JPA functionaliteit, die wordt aangeroepen door middel van JPA annotaties. Van ID zijn deze annotaties als volgt:

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
```

De eerste annotatie geeft aan dat dit attribuut een primary key, of een onderdeel daarvan betreft. Met de tweede annotatie wordt aangegeven dat het om een automatisch gegenereerde waarde gaat, waarbij het generator type wordt gegeven. Pas wanneer JPA's EntityManager het object persisteert wordt het ID gegenereerd, waardoor deze methode in de modellaag niet getest kan worden. Hetzelfde geldt voor de *dateCreated*, alleen wordt hier gebruik gemaakt van *@PrePersist*. Voorafgaand aan het persisteren van het object wordt automatisch de methoden geannoteerd met *@PrePersist* aangeroepen.

```
@PrePersist
protected void setCreationDate() {
    dateCreated = new Date();
}
```

Ik ben hierbij tegen het probleem aangelopen dat ik in dit geval er van uit moet gaan dat JPA op een correcte manier functioneert. Gezien JPA een externe dependency betreft is dit niet altijd gewenst. Om gedurende de rest van het testproces een goede richting te kiezen besloot ik mijn opties nader te specificeren.

4.4.3 Component of unittests

In een situatie zoals beschreven in voorgaande paragraaf kan worden gekozen tussen twee verschillende tests, namelijk unittests of componenttests. Op deze twee verschillende tests heb ik mij georiënteerd. Beide worden nader toegelicht.

Bij unittesten is het gebruikelijk de dependencies binnen de code te mocken. Dit houdt in dat de dependency wordt nagemaakt waarbij vooraf wordt gedefinieerd dat het gemockte object bij een bepaalde aanvraag een vooraf bepaalde waarde teruggeeft. Bij het testen is het vaak gewenst dat de tests onafhankelijk zijn van externe systemen of frameworks. Dit komt doordat bij gebruikt van een extern systeem, de tester er niet zeker van kan zijn dat het externe systeem zich exact zo gedraagt als verwacht. Een voorbeeld hiervan is de database. Wanneer een test gebruik maakt van data uit de database, is de test afhankelijk van de inhoud van een database, waarbij geen rekening wordt gehouden met dat de inhoud buiten de tests om kan veranderen. Hierdoor moet, om zeker te zijn van een succesvol resultaat, de database gemockt worden.

In het geval van Goudvink geldt hetzelfde voor het gebruik van de EntityManager. Om er absoluut zeker van te zijn dat de EntityManager de verwachtte resultaten teruggeeft, moet het verwachtte gedrag vooraf worden gedefinieerd. Dit is het punt waar het mocken van een dependency bij komt kijken. Om absoluut zeker te zijn van het resultaat dat de dependency teruggeeft, kan deze dependency worden nagemaakt. Gezien de DAOlaag veelvoudig gebruik maakt van de EntityManager, betekent het gebruik van unittesting dat de EntityManager gemockt moet worden. Gezien de omvang en de complexiteit van de interactie met de EntityManager staat dit niet in verhouding met wat het systeem functioneel doet. Tevens wordt het JPA framework als vertrouwd framework gezien binnen 42 BV, waardoor het mocken van dit framework wordt voorkomen.

Wanneer het mocken van alle dependencies niet gewenst is, kan worden gekozen voor componenttesten. Hierbij worden alle dependencies die je anders mockt, vervangen door de daadwerkelijke dependency. In het geval van de EntityManager betekent dit dat de DAOlaag direct communiceert met het echte EntityManager-object, in plaats van een gemockte versie daarvan. Hierbij wordt er van uitgegaan dat het gebruikte object goed functioneert. Dit is bij een extern framework niet altijd gewenst, maar in het geval van veelgebruikte frameworks kunnen hier uitzonderingen op worden gemaakt. Het gebruik van componenttests betekent overigens niet dat er geen mocks gemaakt kunnen worden. Zo kan nog steeds worden besloten om bijvoorbeeld de database of externe systemen te mocken. Hier kom ik *in paragraaf 4.4.4 – Database* nog op terug.

Het doel van het testen binnen Goudvink is dat wordt gecontroleerd of de queries die op de database worden afgevuurd (door middel van de EntityManager), wel het gewenste resultaat teruggeven. Doordat je bij unittesten de EntityManager namaakt, test je wel dat de bedoelde query aan de EntityManager wordt gegeven, maar niet of de query die de EntityManager verstuurd de correcte data teruggeeft. Immers definieert de tester voorafgaand in de mock zelf het resultaat dat gegeven moet worden.

Uiteindelijk heb ik gekozen voor het componenttesten. Binnen het Goudvinkproject is het de bedoeling dat getest wordt of de gemaakte code goed functioneert, niet of de code in combinatie met een extern framework correct is. Het is dus de bedoeling dat op de DAOlaag wordt getest of de vraag aan de EntityManager het juiste object uit de database oplevert, niet of de EntityManager de vraag op een juiste manier vertaald en opstuurt.

4.4.4 Database

Goudvink maakt gebruik van een fysieke database, gelokaliseerd op een fysieke server. Tijdens het testen gebruik maken van een fysieke database brengt nadelen met zich mee. Het grootste nadeel is dat ik als tester bij het starten van een applicatie niet zeker kan zijn wat voor data er in de externe database staat. Natuurlijk kan ik voorafgaand aan het starten van mijn tests altijd controleren of bepaalde data in de database staat, maar dit is niet handig. Een andere optie die ik heb is zelf de benodigde data toevoegen, maar dan raak ik het overzicht van de totaal waardes in de database kwijt. Daarbij is het zo dat ik door een fysieke database te gebruiken een extra dependency opneem in het testtraject. De tests werken immers niet wanneer de database is uitgeschakeld. Door een lokaal draaiende database op te zetten, kan ik vrij zeker zijn van de bestaande data in de database. Daarbij zou ik niet afhankelijk zijn van een extern systeem. Wanneer ik echter voor deze optie kies, raak ik de portabiliteit van de applicatie kwijt. Indien andere ontwikkelaars de tests in dat geval draaien, falen de tests in eerste instantie doordat zij de lokale database niet hebben.

Hierdoor heb ik gekozen om gebruik te maken van een in-memorydatabase. Naast dat dit sneller werkt in vergelijking met een fysieke database, is het qua tests ook makkelijker te gebruiken. Ik ben namelijk niet afhankelijk van de werking van een extern systeem en kan daarbij zeker zijn van een beginsituatie. Een in-memorydatabase wordt gemaakt en opgestart voorafgaand aan het uitvoeren van de tests. Vervolgens wordt de in-memorydatabase gevuld aan de hand van een SQL-script zodra ik de software build. Het SQL script definieer ik zelf, wat betekent dat wanneer ik in het SQL script testdata zet, deze testdata elke build automatisch wordt toegevoegd in de in-memorydatabase. Wanneer de applicatie wordt gesloten verwijderd dit automatisch de in-memorydatabase, waardoor de volgende build met een nieuwe start kan beginnen, ongeacht wat in een voorgaande build in de database is gewijzigd. Een nadeel wat het gebruik van een in-memorydatabase met zich mee brengt, is dat het niet hetzelfde SQL-dialect spreekt als gewone databases. Met JPA maakt dit echter niet uit, omdat dit framework de meeste SQL-dialecten kent en daardoor automatisch een vertaalslag maakt naar andere SQL-dialecten. Het verschil in SQL-dialecten wordt pas een probleem zodra de Goudvinkapplicatie SQL die de EntityManager gebruikt zelf opgeeft.

Toch heb ik besloten voor de laatste optie te kiezen, omdat het gemak en de snelheid wat een in-memorydatabase met zich meebrengt veel meer opweegt dan de voor- en nadelen van een fysieke database. Daarbij kent het bestaande deel van Goudvink nog niet de complexiteit waarbij door de ontwikkelaar zelf SQL code aan de EntityManager wordt meegegeven.

Om gebruik te kunnen maken van een in-memorydatabase moest ik de configuratie van de test omgeving aanpassen. Dit moest ik doen zodat de testomgeving een andere database gebruikt dan de

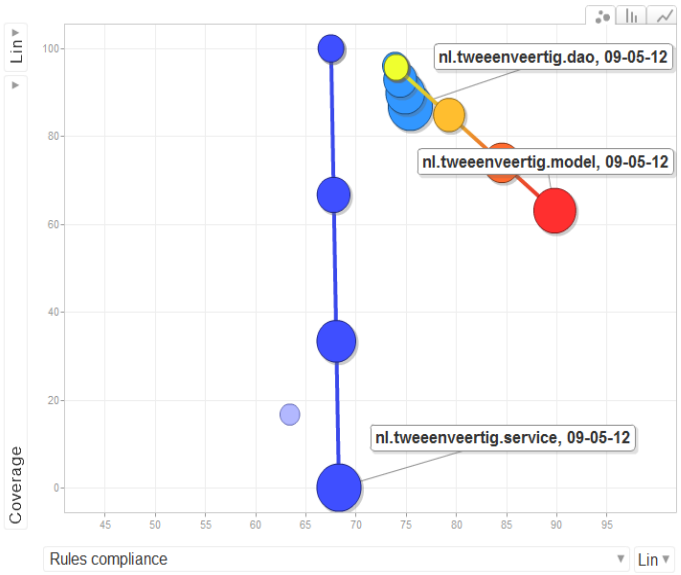
ontwikkelomgeving. Ik heb dit gedaan door een klein stuk code toe te voegen aan het ***test-application-context.xml*** bestand als volgt:

```
<jdbc:embedded-database id="dataSource">
    <jdbc:script location="classpath:create-schema.sql"/>
    <jdbc:script location="classpath:test-data.sql"/>
</jdbc:embedded-database>
```

Hierin vertel ik de JDBC-driver dat deze gebruik moet maken van een embedded-database, waarbij ik de locatie van zowel het te gebruiken tabellen script (*create-schema.sql*) als het script van de test data (*test-data.sql*) meegeef.

4.4.5 Test resultaat

Bij het testen van methoden die in het begin stadium van Goudvink al bestonden heb ik er specifiek voor gezorgd dat tests onafhankelijk van elkaar werken, waardoor zij in willekeurige volgorde kunnen worden uitgevoerd. Hier heb ik specifiek op gelet zodat een veranderende test niet van invloed is op de overige tests. Het verbeteren van de Code Coverage en daarmee dus de verhoging van het aantal (succesvolle) tests, is gedurende de eerste paar dagen goed te zien in de Motion Chart, te vinden in onderstaande afbeelding. Deze Motion Chart is gegenereerd middels Sonar. De datum die wordt weergegeven is de datum vanaf het moment dat ik de meting start. Het gaat in dit geval om een periode van een aantal dagen (09-05-2012 tot 14-05-2012).



Bron – Sonar Motion Chart

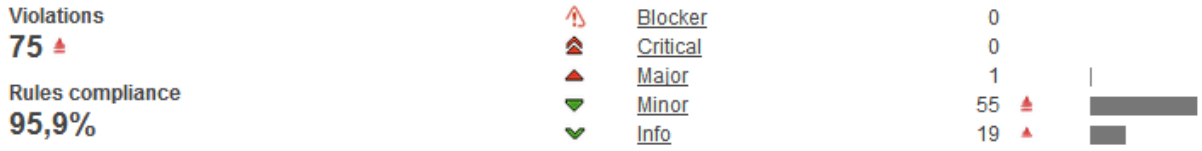
Eigenschap	*Betekenis*
X-as	De Rules Compliance van het project
Y-as	De Code Coverage binnen het project.
Bolletje	Een package uit het project.
Bolletje (grootte)	Complexiteit van de package
Bolletje (kleur)	Gemiddelde complexiteit van de methodes

4.4.6 Code Violations

Naast het testen van de code ben ik ook met een ander onderdeel van de DoD aan de slag gegaan, de Code Violations. Eerder genoemd is al dat 42 BV gebruik maakt van Atlassian's Sonar om de kwaliteit van de code in te zien. Daarbij is verteld dat de ontwikkelaar feedback over de kwaliteit van de geschreven code krijgt door middel van overzichten.

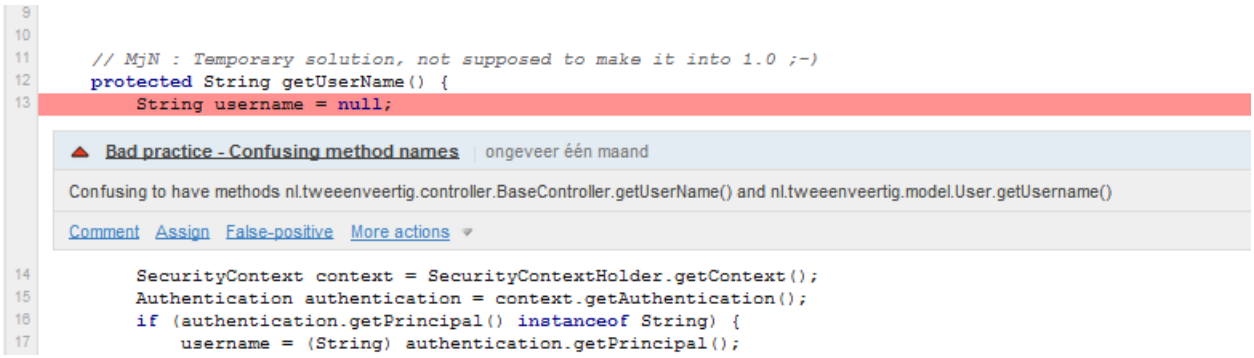
De aanpak die ik bij het verhelpen van violations heb gekozen is door heel simpel per violation naar een oplossing te zoeken. In veel gevallen gaat het ook om een herhalende violation, waardoor als eenmaal een oplossing is gevonden een aantal andere violations ook verholpen kunnen worden.

Om te verduidelijken wat het verhelpen van Code Violations precies inhoud geef ik een voorbeeld:



Bron: Sonar violations report Goudvink (28-08-2012)

Het bovenstaande overzicht betreft het aantal violations die door Sonar zijn aangetroffen binnen de applicatie code. Uit de afbeelding is op te maken dat het Goudvinkproject één major violation, 55 minor violations en 19 kritische opmerkingen bevat. In dit geval is de major violation snel op te lossen. Wanneer op de violation wordt geklikt is het volgende te zien:



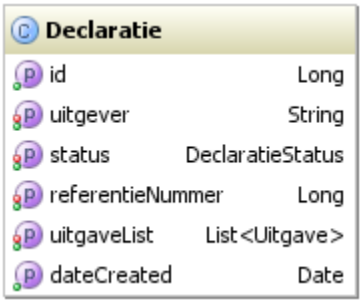
Bron: Sonar violations report Goudvink (28-08-2012)

Het gaat in het geval van deze Major violation alleen om een refactor van een methode naam. Het hoeft echter niet te betekenen dat dit altijd zo is. Soms gaat het om hele structuren in de code die geherformuleerd moeten worden om aan een bepaalde code conventie te voldoen.

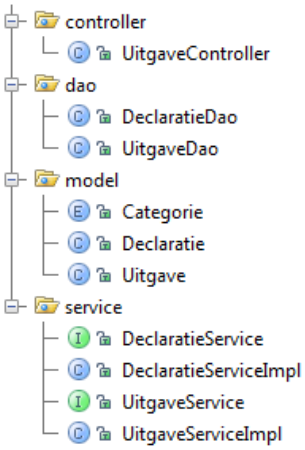
4.5 Sprint 2 - Ontwikkelen declaratie requirement

In de tweede Sprint wordt de tweede requirement van de applicatie ontwikkeld. Gedurende een groot deel van deze sprint was de Scrummaster door omstandigheden afwezig, waardoor ik de taken van hem voor een groot deel heb overgenomen. Dit deed zich met name voor tijdens de sprintplanningen, de sprintdemo en de dagelijkse stand-up. Meer hierover in ***Hoofdstuk 7 – Procesevaluatie***.

Voor de tweede requirement moet een gebruiker in staat zijn om een declaratie in te dienen. Een declaratie bestaat uit een verzameling van uitgaven waarvan de gebruiker definitief heeft besloten deze te declareren. De attributen zijn als volgt:



Om aan de architectuur van Goudvink te voldoen heb ik bij de ontwikkeling van de *indienen declaratie* requirement dezelfde architectuur aangehouden als bij de *invoeren uitgave* requirement. Doordat de onderliggende relaties voor mijzelf goed duidelijk waren, heb ik hier geen klassendiagram voor gemodelleerd. Ik heb vier nieuwe klassen toegevoegd aan het project, de *Declaratie* klasse in de modellaag, de *DeclaratieDao* klasse in de DAOlaag en de *DeclaratieService* en *DeclaratieServiceImpl* in de servicelaag.



Aan de hand van de eerder opgestelde requirement heb ik een aantal functionaliteiten vastgesteld die dit gedeelte van de code minimaal moet hebben om de opgestelde requirement te behalen. Deze functionaliteiten zijn als volgt:

1. Een declaratie kan worden aangemaakt.
2. Er is een afhandeling naar de database, waarbij CRUD functionaliteit mogelijk is.
3. Een lijst van declaraties kan worden opgevraagd.
4. Een lijst van declaraties van een specifieke uitgever kan worden opgevraagd.
5. De waardes van de declaratie worden automatisch gegenereerd.
6. Het *referentienummer* moet worden gegenereerd en de *uitgever* bepaald. Dit is een uitzondering op de bij punt vijf besproken functionaliteit, omdat hier meer bij komt kijken. Hier kom ik op terug in de volgende paragraaf.

Functionaliteit één tot en met vier is de verantwoordelijkheid van de DAO klasse. Voor functionaliteit vijf is de modellaag verantwoordelijk, waarbij middels JPA annotaties de *id* en *dateCreated* attributen worden bepaald. De servicelaag is verantwoordelijk voor het genereren van het *referentienummer* en het bepalen van de *uitgever*.

4.5.1 Referentienummer

Tijdens de ontwikkeling van de *indienen declaratie* requirement liep ik tegen een probleem aan waarbij voorafgaand aan de Sprintplanning niet was afgesproken waar een *referentienummer* uit bestaat. Het referentienummer is bedoeld voor de gebruiker, waarbij de gebruiker het referentienummer op een envelop met de bonnetjes van de te declareren uitgaven schrijft alvorens hij deze inlevert. Daarna dient het referentienummer alleen voor verdere communicatie tussen indiener en de andere rollen van het systeem.

De opbouw van het referentienummer is nog niet bekend en het eerst volgende gesprek met de productowner is pas over een aantal weken. Ik moest hier dus een tussenoplossing bedenken, waarvan ik denk dat dit in de buurt komt van wat de productowner wil en wat minimale aanpassing nodig heeft mocht dit laatste niet het geval zijn. Daarbij moet het referentienummer uniek zijn voor iedere declaratie. Ik heb een aantal oplossingen voor dit probleem bedacht, waarbij ik rekening heb gehouden met het feit dat voor gebruikersgemak het getal niet te lang mag zijn.

Uiteindelijk is gekozen voor een getal, opgebouwd aan de hand van de datum van declaratie en het declaratie nummer van die dag. Hierbij wordt de datum verkort en omgekeerd opgeschreven zodat sorteren op referentienummer nog steeds correct verloopt. Declaratie nummer één op 5 december 2012, wordt dus 1212051. Declaratie nummer twee op 5 december 2012 heeft dan als referentienummer 1212052. Dit getal heeft als voordeel dat het meer betekenis heeft dan een oplopend getal vanaf nul, doordat de datum van declaratie is opgenomen in het getal, waarbij in de toekomst makkelijker kan worden terug gezocht op datum van declaratie.

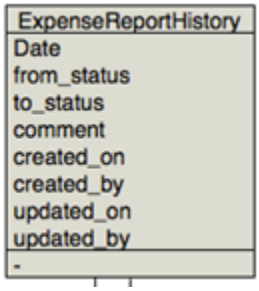
Aansluitend hierop heb ik besloten de methode die het referentienummer genereert op een manier te programmeren dat er vanuit de rest van de applicatie maar één afhankelijkheid is van deze methode, de

methode die de declaratie aanmaakt. Dit stelt het team in staat om de code maar op één plek aan te passen wanneer de productowner het oneens is met de gekozen oplossing voor het referentienummer. Bij de ontwikkeling van de methode die het referentienummer genereerd heb ik de servicelaag voor het eerst anders gebruikt dan een doorgeefluik. Voorheen deed de servicelaag namelijk niets anders dan de aanvragen van de controllerlaag afhandelen met de DAOlaag. Het bepalen van een referentienummer behoort niet in de modellaag thuis, omdat de modellaag alleen functionaliteit heeft om van zichzelf eigenschappen aan te passen. Voor een referentienummer moet het object echter weten wat het voorgaande declaratienummer is op een bepaalde dag. Tevens hoort deze functionaliteit niet thuis in de DAOlaag, de DAOlaag is enkel verantwoordelijk voor het ophalen en wijzigen van objecten. Hierdoor bleef de servicelaag over.

4.5.2 Declaratie changelog

Doordat ik tijdens de tweede Sprint al in een vroeg stadium klaar was met het ontwikkelen van de *indienen declaratie* requirement besloot ik verder te gaan met een ander requirement. Over de beslissing om door te gaan met een andere requirement is nog een hele discussie ontstaan met één van de teamleden. Hier heb ik uiteindelijk een leermoment uit getrokken, waar ik op terug kom in ***Paragraaf 4.7 - Leermomenten***. Uit het klassendiagram van TODD is op te maken dat in de Proof of Concept van Goudvink een declaratie changelog (*ExpenseReportHistory*) is ontwikkeld. De Scrummaster had mij gevraagd of ik hier gedurende de laatste paar dagen van de Sprint mee aan de slag kon gaan.

Ik heb gekozen om dezelfde aanpak als bij de *declaratie invoeren* requirement te kiezen. Doordat deze requirement niet voorafgaand aan de Sprint was uitgedacht, heb ik deze requirements qua modellaag gebaseerd op het voorgaande TODD project.



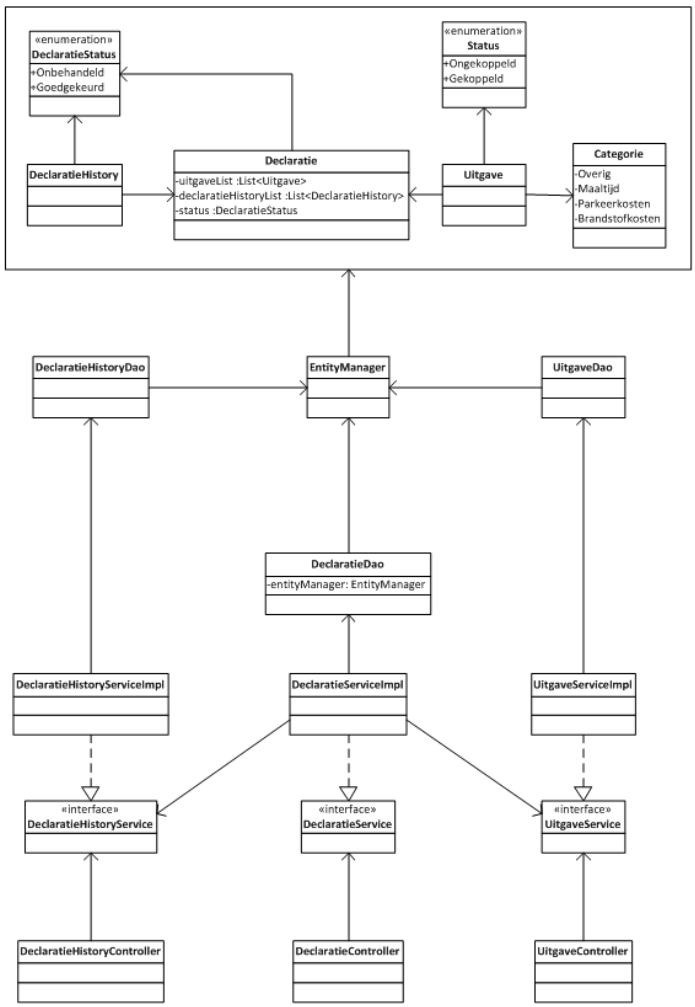
Te zien is dat ik heb besloten een aantal attributen weg te laten, te weten *comment*, *created_on*, *created_by*, *updated_on* en *updated_by*. Deze attributen kunnen allen worden achterhaald aan de hand van de attributen die ik zelf heb gekozen, waardoor zij dus niet nodig zijn.

Door de vier lagen architectuur moeten een aantal nieuwe klassen worden toegevoegd in het systeem. Door de hoeveelheid relaties die nu ontstaan in het project kan makkelijk het overzicht worden verloren. Daarom heb ik voorafgaand aan het implementeren van de *declaratie changelog* requirement een nieuw klassendiagram opgesteld. Deze heb ik later na ontwikkeling van de requirement op bepaalde punten gecorrigeerd, zodat ik deze kon gebruiken voor de overdracht aan het team. Dit wordt nader besproken in de volgende paragraaf.

4.5.3 Overdracht Declaratie Changelog

Binnen IntelliJ is een optie beschikbaar waarmee de ontwikkelaar een patch kan maken van de wijzigingen die hij of zij heeft aangebracht in een project. De overdracht van de code van de declaratie changelog heb ik dan ook gedaan door gebruik te maken van deze optie. De patch stelt het projectteam in staat om op een later tijdsstip de *declaratie changelog* requirement te implementeren. Naast het beschikbaar stellen van de patch heb ik ook een overdrachtsgesprek gehouden met de teamleden, om duidelijk te maken waar de patch uit bestaat en wat deze kan. Ook zaten er nog enkele onopgeloste bugs in de code van deze requirement, waar ik door het naderende einde van de Sprint niet aan toe ben gekomen. Deze bugs heb ik in projectmanagement tool Jira gezet, zodat het team hier op een later tijdstip naar kan kijken. Ook heb ik ze in de overdracht benoemd, met de oplossing die ik denk dat het beste is.

Gezien het een vrij lang verhaal betrof en ik zeker wilde zijn dat ik geen punten oversloeg, besloot ik tijdens het gesprek een klassendiagram te tekenen die de weergave van het systeem gaf zoals die op dat moment bestond. Als voordeel werden hierdoor ook de nieuwe relaties tussen de diverse klassen, waaronder de declaratie changelog klassen, duidelijker. Om het beeld voor mijzelf goed duidelijk te krijgen heb ik voorafgaand aan de overdracht sessie het eerder gemaakte high level klassendiagram gecorrigeerd en voor mijzelf doorgenomen. Het diagram op volledige grootte is te vinden in ***Bijlage B – Goudvink diagram***.



4.6 Resultaat fase één

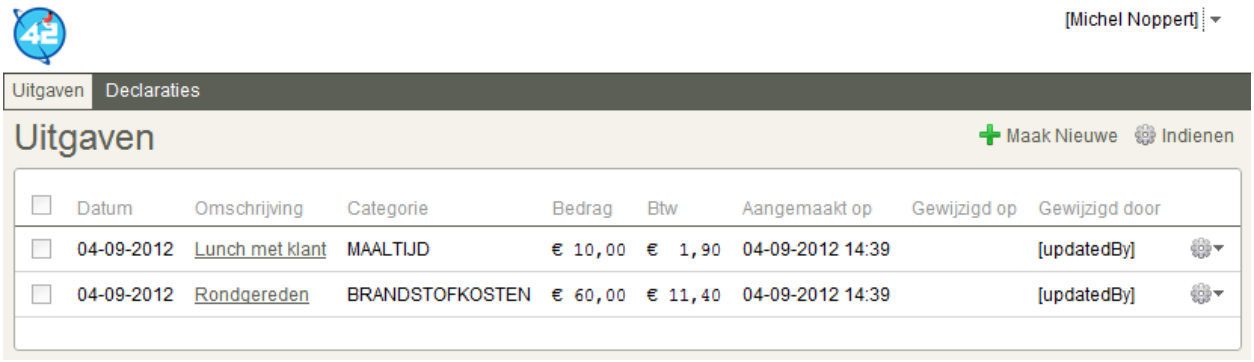
Gedurende de eerste fase van mijn afstudeerproject heb ik meegeholpen aan de ontwikkeling van het Goudvinkproject. Hierbij heb ik mij voornamelijk gericht op de functionaliteit die voor mij in fase drie van het afstuderen benodigd is om door middel van de prototype app te communiceren met de database. Tevens heb ik mij gericht op het behalen van de voorafgaand aan deze fase opgestelde Definition of Done. In deze paragraaf geef ik een kort overzicht van de resultaten die ik gedurende deze fase heb behaald. Ter afsluiting van de eerste fase heb ik aan mijn projectleden een evaluatieformulier gestuurd zodat zij mijn werkzaamheden gedurende deze periode konden beoordelen. Deze evaluatieformulieren zijn te vinden in ***Bijlage C - Evaluatieformulieren***.

4.6.1 Requirements resultaat

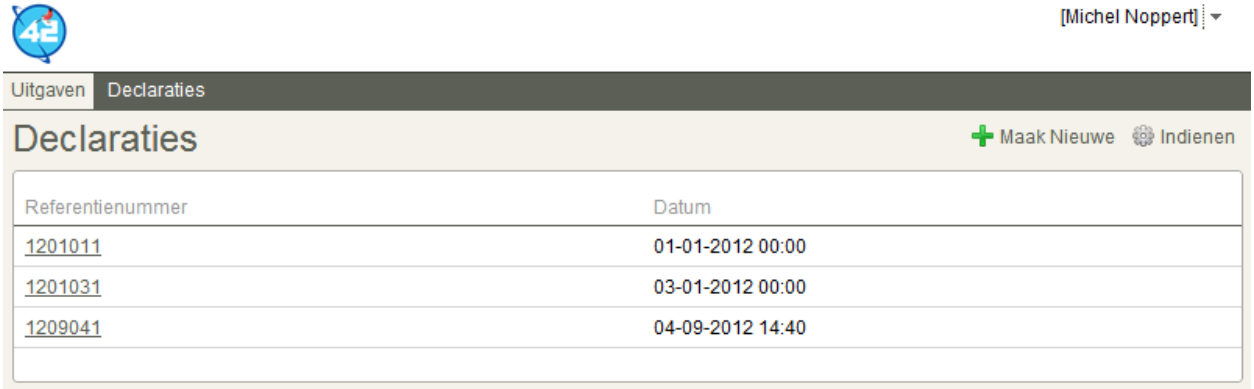
De requirements waarvan ik en het projectteam voorafgaand aan deze fase hadden verwacht dat deze aan bod kwamen zijn uiteindelijk succesvol afgerond. Dit is te zien aan de hand van de screenshots bijgevoegd in onderstaande sub-paragrafen. Hierbij vermeld ik nadrukkelijk dat ik op geen enkele wijze betrokken ben geweest bij de visuele weergave van de declaraties en uitgaven door middel van een user interface. Onderstaande screenshots zijn dan ook enkel bedoeld om aan te tonen waar het systeem op het moment van mijn vertrek toe in staat was.

Invoeren uitgave

Uitgavenoverzicht



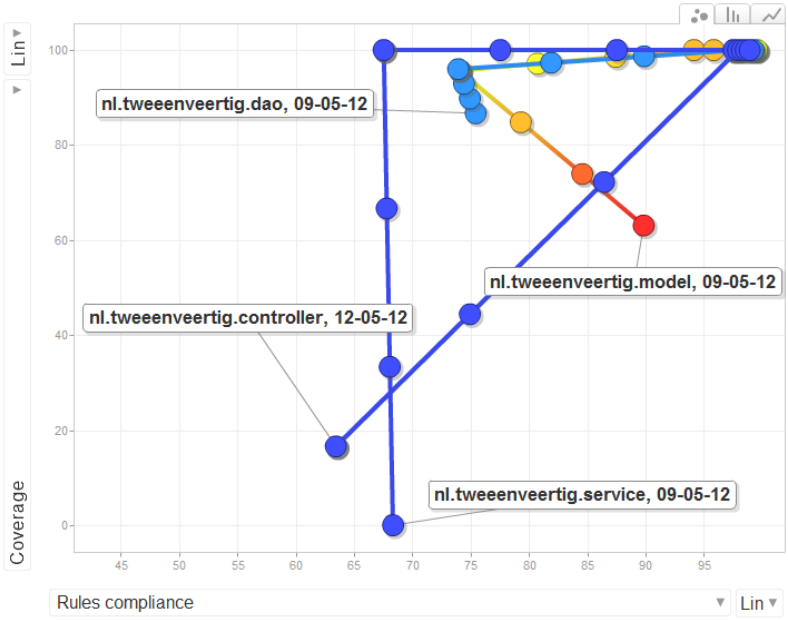
Declaratieoverzicht



Declaratiedetails

4.6.2 Testresultaat

Het resultaat van het continu testen van de ontwikkelde code waar ik mij tijdens deze fase onder andere mee heb beziggehouden, is in onderstaande Motion Chart te zien. Te zien is dat toen ik eenmaal klaar was met deze fase, de Code Quality specifieke onderdelen van de DoD nageleefd zijn.



Bron – Sonar Motion Chart

Eigenschap	*Betekenis*
X-as	De Rules Compliance van het project
Y-as	De Code Coverage binnen het project.
Bolletje	Een package uit het project.
Bolletje (grootte)	Complexiteit van de package
Bolletje (kleur)	Gemiddelde complexiteit van de methodes

4.7 Leermomenten

Gedurende deze fase heb ik een aantal leermomenten gehad, waarvan ik een aantal nader toelicht in de volgende sub-paragrafen.

4.7.1 Gebruik technische backlog

Naar voren is gekomen dat toen ik vroegtijdig klaar was met de ontwikkeling van de *indienen declaratie* requirement, ik door ben gegaan met het ontwikkelen van de *declaratie changelog*. Deze requirement stond niet op de sprintbacklog, maar de Scrummaster wist uit het voorgaande TODD project dat een changelog van de declaraties bijgehouden moest worden. De Scrummaster heeft mij gevraagd of ik mij met deze requirement kon bezig houden. Hierbij ontstond een probleem waar ik gedurende deze periode tegenaan ben gelopen. Toen ik het resultaat van mijn werk aan de *declaratie changelog* liet zien aan de overige teamleden, bleek met name de Scrummaster hier niet blij mee te zijn. Met zijn vraag of ik met de declaratie changelog aan de slag kon gaan had hij nooit bedoeld dat ik de requirement daadwerkelijk ging ontwikkelen, maar eerst een, naar zijn woorden, 'uitgebreid onderzoek' had gedaan naar de mogelijkheden tot implementatie hiervan.

Eerst een 'uitgebreid onderzoek' doen sluit echter niet aan op de gekozen werkwijze van dit project, waarbij een onderdeel zo snel mogelijk wordt ontwikkeld, zodat deze op een later tijdstip kan worden verbeterd aan de hand van feedback van de stakeholders. Hierbij impliceer ik overigens niet dat ik niet heb nagedacht over de diverse mogelijkheden tot implementatie van een changelog. Ik heb meerdere opties afgewogen en uiteindelijk de naar mijn mening beste gekozen. Toch was de Scrummaster het hier niet mee eens, mede doordat deze requirement niet was meegenomen in de Sprintplanning.

Later ben ik hier nog met mijn bedrijfsmentor over in gesprek gegaan. Van hem kreeg ik te horen dat het normaliter in een situatie waarin geen backlog items meer beschikbaar zijn, de bedoeling is dat je eerst met de klant in overleg gaat. In dit overleg vraag je wat de klant graag als volgend onderdeel ontwikkeld ziet, waarna dit onderdeel vervolgens op de technische backlog wordt geplaatst. Dit laatste betekend voor het bedrijf dat het item aan een sprint kan worden toegekend als de overige onderdelen van de backlog af zijn. Dit was een goed Scrum-leermoment voor mij en is daarmee iets wat ik in een soortgelijke situatie ook zeker toe kan passen in een volgend project.

4.7.2 SVN branching

Afgesproken met de Scrummaster was dat ik wel verder ga met het ontwikkelen van de declaratie changelog, maar dat ik deze niet mag committen naar de SVN server. Dit bracht namelijk 'onnodige complicaties' met zich mee. Door echter niet te committen, ben ik zelf hierdoor tegen andere problemen aangelopen. Deze problemen deden zich voor toen op een gegeven moment zich een bug voordeed in de code die ik voorafgaand aan de declaratie changelog had gemaakt. Om deze bug voor de overige medewerkers te verhelpen, moest ik de code die ik had aangepast committen. Echter, dit is dan inclusief de code die ik had geschreven voor de declaratie changelog, wat op zijn beurt weer problemen met zich mee zou brengen omdat ik hier nog aan bezig was. Dit is op dat moment op een hele ongemakkelijke manier opgelost, door een stuk code door te sturen aan een ander projectlid, die vervolgens de bestaande code verving waarna hij wel kon committen. Vanaf het begin wist ik al dat dat

een foutieve manier van werken is, maar gezien het naderende einde van de sprint, zorgde het wel voor dat alle focus op het ontwikkelen kon blijven.

Later ben ik erachtergekomen dat de juiste aanpak voor een soortgelijke situatie is om gebruik te maken van een SVN branch. Door gebruik te maken van een SVN branch had de changelog requirement ongehinderd door kunnen worden ontwikkeld, waarbij de voordelen van SVN op een later tijdstip hun dienst hadden bewezen wanneer het team de changelog implementeerde. Daarbij kan door gebruik te maken van een branch heel specifiek één deel van de code worden gecommit naar de SVN, waardoor het voorgaande probleem zich niet had voor hoeven doen.

4.7.3 Componenttesten

Gedurende deze fase heb ik besloten componenttests te gebruiken in plaats van unittests. Na de eerste fase ben ik er echter achter gekomen dat het binnen de gekozen architectuur de bedoeling is dat wanneer de controllerlaag getest wordt, de servicelaag wordt gemockt. Dit houdt in dat wanneer de controllerlaag wordt getest, gebruik gemaakt moet worden van unittests in plaats van componenttests. Dit komt door het volgende:

De tester kan zeker zijn van het feit dat de objecten die de servicelaag teruggeeft aan de controllerlaag, geen invloed of connectie meer hebben met de EntityManager. Dit komt omdat de aanvraag aan de servicelaag in een transactie wordt uitgevoerd, waarna bij teruggave van het object uit de servicelaag de connectie van het object naar de EntityManager wordt verbroken.

Dit is niet het geval bij het object dat de DAOlaag teruggeeft aan de servicelaag. Doordat de aanvraag aan de DAOlaag vanuit de servicelaag niet in een transactie verloopt, behoud het teruggegeven object zijn verbinding met de EntityManager. Wanneer de servicelaag een aanpassing op het object doet, wordt dit tevens door de EntityManager doorgevoerd in de database. Hierdoor kan de DAOlaag niet worden gemockt voor de servicelaag, tenzij de tester ook alle objecten mockt. Dit is uit kostenoverwegingen niet te doen.

5 Fase 2 – Onderzoek app types

In de tweede fase van het afstudeerproject is een verkennend onderzoek gedaan naar de verschillende mogelijkheden voor het ontwikkelen van applicaties voor een mobiele omgeving zoals smartphones. Om een duidelijk beeld te krijgen van wat onderzocht moest worden ben ik begonnen met het gedetailleerd uitwerken van de aanleiding en doelstelling in een onderzoeksplan. Als belangrijkste onderdeel van het onderzoeksplan is een centrale hoofdvraag met bijbehorende deelvragen gedefinieerd. Door voorafgaand aan het onderzoek een hoofdvraag te formuleren, is een duidelijke leidraad in het onderzoek ontstaan die zich richt op het beantwoorden van de hoofdvraag. Doordat het gehele onderzoek gebaseerd is op de hoofdvraag wordt in de volgende paragraaf toegelicht hoe ik tot deze hoofdvraag ben gekomen.

5.1 Totstandkoming hoofdvraag

De hoofdvraag is opgesteld aan de hand van de aanleiding en doelstelling van het onderzoek (zie ***Hoofdstuk 3 – Opdracht en aanpak***), die verder zijn aangevuld aan de hand van diverse gesprekken met de opdrachtgever. Het relevante deel voor het onderzoek van de aanleiding, die eerder in dit document is besproken, is als volgt:

> ***Aanleiding***
>
> Binnen de ICT wereld vinden continu nieuwe ontwikkelingen plaats. Een sterk groeiende ontwikkeling op het moment is die van mobiel werken, waarmee in dit geval bedoeld wordt plaats- en tijdsonafhankelijk werken. Deze ontwikkeling is door de komst van smartphones en tablets sinds een aantal jaar in opkomst, waardoor voor veel mensen uitsluitend werken achter de computer tot het verleden behoort. De klanten van 42 BV zien de sterke vooruitgang in deze ontwikkelingen en daardoor rijst steeds vaker de vraag in hoeverre zij de ontwikkelde applicaties kunnen gebruiken in een mobiele omgeving. Binnen 42 BV is geen specialistische kennis over het bouwen van mobiele applicaties aanwezig, met het gevolg dat mobiele oplossingen vaak door middel van een web-applicatie worden aangeboden aan de klant. Om in de toekomst beter mee te draaien in de zogenoemde appsmarkt wilt 42 BV zich nu gaan richten op de mobiele ontwikkelingen en de obstakels die daarbij komen kijken in kaart brengen. Zo kunnen zij de klant beter van dienst zijn en in hun behoeften voorzien.
>
> 42 BV heeft vernomen dat er binnen de mobiele markt verschillende app types bestaan, met elk hun eigen ontwikkelsituatie. Voordat 42 BV aan het ontwikkelen van mobiele apps voor haar klanten begint, wil het bedrijf weten in hoeverre de ontwikkeling van een app aansluit op de ontwikkelstraat zoals deze nu wordt toegepast binnen het bedrijf. Hierbij is het bedrijf specifiek op zoek naar welk van de app types het meeste aansluit op de huidige ontwikkelstraat. Deze ontwikkelstraat is nog niet eerder in kaart gebracht, waardoor dit ook tot het onderzoek behoort.

Bron: Paragraaf 3.1.1, huidig document

Op basis van de aanleiding, doelstelling en gevoerde gesprekken met de opdrachtgever heb ik een aantal zaken afgeleid die voor 42 BV belangrijk zijn. Op volgorde van de aanleiding en doelstelling zijn deze als volgt:

1. *Klanten vragen zich af in hoeverre zij de ontwikkelde applicaties kunnen gebruiken in een mobiele omgeving.*
   De klanten van 42 BV zijn geïnteresseerd in hoeverre ontwikkelde applicaties kunnen worden gebruikt in een mobiele omgeving. Om te weten waar de mobiele omgeving toe in staat is, is het belangrijk de mogelijkheden hiervan te achterhalen.

2. *Binnen 42 BV is geen specialistische kennis aanwezig over het ontwikkelen van apps.*
   42 BV ziet het als een probleem dat zij geen specialistische kennis hebben over het ontwikkelen van apps. Daaruit kan ik afleiden dat het bedrijf niet aan de slag wil met iets wat zij (vooralsnog) niet kennen. Het is dus belangrijk een vergelijking te maken tussen het ontwikkelen van apps en de ontwikkelsituatie die momenteel door 42 BV gehanteerd wordt voor het ontwikkelen van applicaties. Dit geeft het bedrijf inzicht in welke kennis zij nog op moeten doen om aan de slag te kunnen met het ontwikkelen van applicaties.

3. *42 BV is geïnteresseerd in de obstakels die het ontwikkelen van apps met zich mee brengt.*
   Het bedrijf verwacht tegen obstakels aan te lopen tijdens het ontwikkelen van apps. Zij zijn dus geïnteresseerd in het vinden van de problemen (nadelen van apps) waar ontwikkelaars mogelijk tegen aan kunnen lopen tijdens de ontwikkeling.

4. *42 BV wilt de klant beter in hun behoefte kunnen voorzien.*
   Om de klant in zijn behoefte te kunnen voorzien moet achterhaald worden wat de klant eigenlijk wilt. De reden dat er meerdere app types bestaan is doordat er verschillende besturingssystemen bestaan binnen de appsmarkt. Specifieker kan dus worden gezegd dat achterhaald moet worden in welke besturingssystemen de klanten van 42 BV het meest geïnteresseerd zijn. Dit is belangrijk voor 42 BV om te weten, omdat dit in grote mate bepaald welk van de onderzochte app types geschikt zijn.

5. *Er bestaan verschillende ontwikkelmogelijkheden.*
   42 BV geeft aan dat er verschillende app types met bijbehorende ontwikkelsituaties zijn voor het ontwikkelen van apps. Hierbij onderscheidt het bedrijf vier app types, achtereenvolgend Native, Web, Semi Native en tussentaal. Gezien het bedrijf hier nog geen verder onderzoek naar heeft gedaan is de kans aanwezig dat er meerdere app types bestaan. De invloed die een extra app type kan hebben op de uitkomst van het onderzoek, heeft mij ertoe aangezet eerst verder onderzoek te doen of er niet meer dan vier app types onderscheiden kunnen worden.

Uit deze lijst zijn drie belangrijke aspecten te filteren, namelijk de ontwikkelmogelijkheden, de huidige ontwikkelsituatie van 42 BV en tenslotte de klant. Deze drie aspecten heb ik terug laten komen in de hoofdvraag. Ik ben uitgekomen op de volgende hoofdvraag:

> *Welke mobiele applicatie ontwikkelmogelijkheden zijn voor 42 BV, markttechnisch en met het oog op de huidige ontwikkelstraat, interessant om in de praktijk toe te passen?*

Om de hoofdvraag te kunnen beantwoorden is deze onderverdeeld in deelvragen, die op hun beurt zijn onderverdeeld in onderzoeksvragen. Beantwoording van de deelvragen resulteert in een antwoord op de hoofdvraag. Door de deelvragen onder te verdelen in onderzoeksvragen is een leidraad gecreëerd, die ik gedurende het onderzoek heb kunnen volgen.

De hoofdvraag gaat in op drie verschillende onderwerpen, namelijk de mobiele applicatie ontwikkelmogelijkheden, de ontwikkelstraat van 42 BV en de commerciële markt. Middels de geformuleerde hoofdvraag en de eerder genoemde aanleiding zijn de volgende deelvragen gedefinieerd:

Deelvraag 1 Wat zijn de voor en nadelen van de diverse ontwikkelmogelijkheden?

- Welke ontwikkelmogelijkheden bestaan er?
- Wat zijn de beperkingen per ontwikkelmogelijkheid met betrekking tot native applicatie en portabiliteit?
- Tegen welke problemen lopen andere ontwikkelaars aan (user experiences)?

Deelvraag 2 Wat zijn de gelijkenissen tussen de huidige ontwikkelsituatie van 42 BV en die van de appsmarkt?

- Hoe ziet de huidige ontwikkel situatie van 42 BV er uit?
- Welke aspecten uit de huidige ontwikkel situatie zijn terug te vinden in de appsmarkt?

Deelvraag 3 Welke vraag naar applicaties, specifiek voor bepaalde platformen, is er vanuit de mobiele markt?

- Naar welke platformen is er vanuit de klanten van 42 BV veel vraag?
- Wat zegt het marktaandeel van besturingssystemen binnen de mobiele markt?

Gedurende het onderzoek hebben zowel het bedrijf als ik een voortschrijdend inzicht ontwikkeld, waardoor de initiële opzet van het onderzoek is veranderd. Ik heb besloten de deel- en onderzoeksvragen te herzien, zodat zij beter aansloten op het onderzoek en de vraag van het bedrijf. Op de reden tot wijzigen en het resultaat van deze wijziging kom ik in ***Paragraaf 5.8 – Aspect benadering*** en ***Paragraaf 5.9 – Herzien hoofd- en deelvragen*** terug. Volgende paragrafen gaan uit van de oude situatie.

5.2 Onderzoeksopzet

Na het definiëren van de hoofd en deelvragen moest er gekozen worden voor het hanteren van een kwalitatieve of kwantitatieve opzet voor het onderzoek. Bij kwalitatieve onderzoeken worden de te onderzoeken eenheden (onderzoekseenheden) in de omgeving als geheel onderzocht. Hierbij staat vaak de beleving centraal, waardoor deze onderzoeksopzet geschikt is om individuele onderzoekseenheden te onderzoeken. Bij kwantitatief onderzoek worden cijfermatige gegevens verzameld, waardoor deze onderzoeksopzet geschikt is voor een onderzoek betreffende een samengestelde groepen. Per deelvraag heb ik gekeken naar welke onderzoeksopzet het beste past bij die specifieke deelvraag.

> ***Deelvraag 1: Wat zijn de voor en nadelen van de diverse ontwikkelmogelijkheden?***

De eerste deelvraag bevat drie onderzoeksvragen die antwoord geven op welke ontwikkelmogelijkheden er bestaan, wat de beperkingen met betrekking tot native applicatie ontwikkeling en portabiliteit zijn en tegen welke problemen andere ontwikkelaars aanlopen. Bij alle drie de onderzoeksvragen wordt ingegaan op de individuele onderzoekseenheid, achtereenvolgens namelijk de ontwikkelmogelijkheden, de beperkingen van de ontwikkelmogelijkheden en de ervaring van andere ontwikkelaars. Hierdoor wordt voor deze deelvraag gekozen voor een kwalitatieve opzet.

> ***Deelvraag 2: Wat zijn de gelijkenissen tussen de huidige ontwikkelsituatie van 42 BV en die van de appsmarkt?***

Bij deze deelvraag is het belangrijk dat de huidige ontwikkelsituatie van 42 BV in kaart wordt gebracht, iets dat door de verschillende aspecten, waarbij gebruik wordt gemaakt van één of meerdere tools of technieken, kwalitatieve gegevens betreft. 42 BV hanteert een eigen ontwikkelstraat, waarbij het voor het onderzoek relevant is te achterhalen welke tools en technieken voor diverse aspecten uit de ontwikkelstraat worden gebruikt. Dit betreft kwalitatieve gegevens omdat wordt ingegaan op de diverse aspecten van de ontwikkelstraat (de onderzoekseenheden).

> ***Deelvraag 3: Welke vraag naar applicaties, specifiek voor bepaalde platformen, is er vanuit de mobiele markt?***

Bij de derde deelvraag wordt een uitzondering gemaakt op de kwalitatieve onderzoeksopzet. Zowel de eerste als tweede onderzoeksvraag heeft betrekking tot het verkrijgen van kwantitatieve gegevens. In het eerste geval wordt de voorkeurskeuze van een groep (de klanten van 42 BV) gevraagd op welk type device of besturingssysteem zij de app mobiel willen gebruiken. Dit betreft numerieke gegevens. Bij de tweede onderzoeksvraag is het zaak te onderzoeken wat het marktaandeel binnen de huidige mobiele markt momenteel doet. Ook dit betreft numerieke gegevens waardoor wordt gekozen voor een kwantitatieve opzet.

5.3 Onderzoeksmethoden

Met de keuze tussen een kwalitatieve en kwantitatieve onderzoeksopzet in gedachten, heb ik de te hanteren onderzoeksmethoden vastgesteld per deelvraag.

> ***Deelvraag 1: Wat zijn de voor en nadelen van de diverse ontwikkelmogelijkheden?***

Om de onderzoeksvragen van de eerste deelvraag te beantwoorden, is gekozen voor zowel de literatuuronderzoekmethode als de interviewmethode. De reden voor het kiezen van de literatuuronderzoekmethode is dat veel van de informatie over dit onderwerp te vinden is via het internet, wat dit (mits betrouwbaar) een zeer waardevolle bron maakt voor het onderzoek. Bij beantwoording van de eerste twee onderzoeksvragen is dan ook gekozen voor de literatuuronderzoeksmethode.

De derde onderzoeksvraag betreft het achterhalen van user experiences. Om user experiences te achterhalen heb ik mij aangesloten bij een App developer community, welke regelmatig bijeenkomsten houdt waarbij app ontwikkelaars met elkaar in gesprek kunnen gaan. De informatie die ik heb verkregen binnen de community is vergaard door middel van interviews, waarbij ik dieper in heb kunnen spelen op antwoorden die zijn gegeven. Tevens is voor deze onderzoeksvraag gebruik gemaakt van literatuuronderzoek middels het internet. Er zijn diverse fora te vinden op internet waar ontwikkelaars hun ervaringen en bijbehorende problemen delen.

Populatie

Bij de derde onderzoeksvraag is sprake van een populatie, namelijk de populatie ontwikkelaars op het gebied van app development. Gezien de tijd die besteed kan worden aan het onderzoek heb ik ervoor gekozen niet de gehele populatie te onderzoeken op user experiences. Ik heb mij specifiek gericht op drie deel populaties. Ten eerste richt ik mij op de deelpopulatie van app developers die regelmatig actief gebruik maken van het internet in de vorm van het stellen van vragen op fora en het formuleren van problemen op bijvoorbeeld een persoonlijke blog. Als tweede deelpopulatie benader ik de leden van de eerder genoemde App developer community waar ik mij bij aansluit. De derde deelpopulatie beslaat de App developers die in mijn directe omgeving te vinden zijn.

> ***Deelvraag 2: Wat zijn de gelijkenissen tussen de huidige ontwikkelsituatie van 42 BV en die van de appsmarkt?***

Doordat het achterhalen van de huidige ontwikkelsituatie van 42 BV kwalitatieve gegevens betreft, heb ik voor deze deelvraag gekozen om gebruik te maken van de interviewmethode. Deze methode stelt mij in staat om direct van de te onderzoeken bron te achterhalen wat zij in de dagelijkse bedrijfssituatie gebruiken. Hierbij heb ik bewust besloten om deze methode niet in de vorm van een officieel interview plaats te laten vinden.

Bij een interview hebben mensen al gauw het idee dat ze voor het blok worden gezet en veel tijd voor je vrij moeten maken, iets dat ik wilde voorkomen omdat dit wellicht het onderzoek belemmerd. Als ik

naar de gekozen methoden en technieken vraag in de voor de medewerker natuurlijke omgeving (in de wandelgangen, achter de computer, enzovoort), zijn zij meer geneigd een eerlijk en accuraat antwoord te geven. Dit heeft mij tevens de mogelijkheid gegeven om zaken die ik in de wandelgangen bij 42 BV hoor, of die tijdens de dagelijkse stand-ups naar voren zijn gekomen, mee te nemen in het onderzoek.

Het meenemen van onderwerpen in het onderzoek die in de wandelgangen of tijdens stand-ups naar voren zijn gekomen, wordt ook wel de observatiemethode genoemd. Als specifieke vertakking binnen de observatiemethode wordt het 'observeren in het veld' gebruikt. Bij de observeren in het veld methode wordt in de alledaagse situatie informatie vergaard door gedragingen van de onderzochte personen te observeren. Het betreft in dit onderzoek een observatie van gedrag, doordat medewerkers die tegen problemen aanlopen (iets wat tijdens de stand-ups wordt verteld) daar hun eigen oplossing voor bedenken en uitvoeren. Het gaat in het geval van dit onderzoek dus niet om het observeren van een herhalende gedraging, maar om het vergaren van informatie betreffende de ontwikkelstraat van 42 BV. Dat iedere medewerker situatie afhankelijk een andere oplossing toepast heeft als nadeel dat de observatiemethode in mindere mate herhaalbaar is, terwijl dit officieel gezien wel één van de voorwaarde is voor de observatiemethode.

Populatie

Bij de tweede deelvraag bestaat de populatie uit de medewerkers van 42 BV die dagelijks met de ontwikkelstraat te maken hebben. Hierdoor vallen medewerkers van bijvoorbeeld de marketing en administratie buiten de populatie.

> ***Deelvraag 3: Welke vraag naar applicaties, specifiek voor bepaalde platformen, is er vanuit de mobiele markt?***

De voorkeur voor onderzoeksmethode om de laatste deelvraag te beantwoorden gaat uit naar de survey-onderzoeksmethode, waarbij een vragenlijsten betreffende het mobiel gebruik van applicaties aan de klanten van 42 BV wordt gestuurd. Een andere onderzoeksmethode die is overwogen betreft de (kwalitatieve) interviewmethode. Deze methode zou mij in staat stellen dieper in te gaan op waarom de voorkeur van klanten uit gaat naar bepaalde devices. Toch is deze methode niet gekozen omdat de waarom vraag niet van belang is voor het bedrijf om te weten.

Populatie

Bij de eerste onderzoeksvraag is een populatie te definiëren, de klanten van 42 BV. Tevens is er een strata te definiëren, namelijk de klanten die al eens hebben aangegeven geïnteresseerd te zijn in mobiele apps.

Voor de tweede onderzoeksvraag is de populatie beperkt tot groep mensen die gebruik maken van smartphones in het bedrijfsleven. Hierdoor valt de groep consumenten die smartphones alleen buiten het bedrijf gebruiken buiten de populatie.

5.4 Betrouwbaarheid

Voor een onderzoek is het van belang dat de informatie die wordt onderzocht en gebruikt betrouwbaar is. Het is belangrijk dat uitspraken die worden gedaan in het onderzoeksrapport zijn gevalideerd en op waarheid berusten. Hierbij moet gedefinieerd worden wanneer een uitspraak voldoende is gevalideerd en wanneer deze als betrouwbaar wordt bevonden.

Om uitspraken te verantwoorden en te valideren wordt vaak gebruik gemaakt van externe bronnen. Indien de uitspraken in het onderzoek worden bevestigd door een bron, verhoogt dit de algehele validiteit van het onderzoek. Dit geldt echter niet wanneer de gebruikte bron onbetrouwbaar is, waardoor het zaak is om te bepalen wanneer in dit onderzoek een bron als betrouwbaar wordt gezien.

Tijdens dit onderzoek zijn een aantal betrouwbaarheidscriteria gebruikt om de betrouwbaarheid van een bron vast te stellen. Zo is erop gelet of de bron een primaire of secundaire bron betreft. Een primaire bron komt daarbij uit de eerste hand. Een website van één van de onderzochte frameworks wordt gezien als een betrouwbare bron, omdat de informatie op zijn website direct van het bedrijf of groep achter het onderzochte framework komt. Indien de bron een secundaire bron betreft, oftewel dat de informatie door iemand anders dan de maker van het framework is opgeschreven, moet de bron aan meerdere voorwaarden voldoen.

In dit laatste geval heb ik ervoor gekozen dat een uitspraak valide is wanneer deze door minimaal twee losstaande bronnen kan worden bevestigd. Wanneer twee bronnen losstaand van elkaar eenzelfde uitspraak bevestigen, is de uitspraak valide genoeg voor gebruik binnen dit onderzoek. Dat twee bronnen losstaand van elkaar een uitspraak bevestigen betekend dat een bron, die op zijn beurt gebruik maakt van één andere bron, niet telt als betrouwbare bron. Hierdoor wordt voorkomen dat een bron geschreven uit eigenbelang van de auteur als betrouwbaar wordt gezien. Hier zijn echter uitzonderingen op.

Om te controleren of een secundaire bron betrouwbaar is, is de informatie gevalideerd middels een tweede bron. Dit is echter niet de enige manier om als betrouwbare bron te worden gezien binnen dit onderzoek. Om dit toe te lichten geef ik als voorbeeld het gebruik van Wikipedia, een bekende online encyclopedie. De meningen zijn veelal verdeeld of Wikipedia nu wel of niet een betrouwbare bron is. Toch is in een aantal delen van het onderzoeksrapport Wikipedia als bron gebruikt, omdat de gebruikte pagina's wel degelijk betrouwbaar zijn. Dit komt doordat de gebruikte pagina's veelvoudig zijn aangepast en bijgewerkt door meerdere auteurs. Lastiger is te achterhalen of deze auteurs een bepaald belang hebben bij het schrijven van informatie, maar er kan van worden uitgegaan dat de informatie die er staat veelvoudig is gereviseerd. Hierdoor is de kans minimaal dat de informatie wordt gebaseerd op eigenbelang. Daarbij wordt op de gebruikte pagina's binnen Wikipedia steeds vaker gebruik gemaakt van bronvermeldingen. Om voorgaande te verduidelijken geef ik het volgende voorbeeld. Eén van de in dit onderzoek gebruikte pagina's, is de Wikipedia pagina over smartphones. Deze pagina is in totaal meer dan 500 keer aangepast door diverse auteurs, waarbij in totaal 125 bronnen zijn vermeld (het gaat hier om de Engelse variant van de pagina). Hierdoor voldoet ook een pagina als deze aan de betrouwbaarheidseisen die ik aan dit onderzoek heb gesteld.

5.5 Probleemdomein

Door het probleemdomein te definiëren wordt duidelijk waarom er nu eigenlijk problemen zijn bij het ontwikkelen van apps voor een klant.

Bedrijfspresentatie – 42 gaat mobiel (door Wilco Stuyfersant)

Wanneer een bedrijf een app ontwikkeld voor de klant, staan de klant en het bedrijf in principe lijnrecht tegenover elkaar qua aanpak van ontwikkeling. Waar de klant graag de native experience en (app afhankelijk) native functionaliteit wilt in zijn app, wilt het bedrijf juist portabiliteit en aansluiting op de ontwikkelstraat. Met name op portabiliteit hebben de wensen van de klant (native experience en functionaliteit) invloed. Een hogere mate van native experience en functionaliteit heeft namelijk tot gevolg dat de portabiliteit van de app achteruit gaan, doordat de app in dat geval voor één specifiek besturingssysteem wordt geschreven. Hierdoor moet het bedrijf meer kosten maken om de app geschikt te maken voor meerdere platformen. Om het probleemdomein aan te pakken zijn verschillende app types met een bijbehorende ontwikkelsituatie bedacht, waar het uitgevoerde onderzoek over gaat.

5.6 Definiëren app types

Naar voren is gekomen dat 42 BV in eerste instantie vier verschillende app types definieert, namelijk de app types native, semi-native, web en tussentaal. Zoals eerder genoemd heb ik besloten zelf onderzoek te verrichten naar deze app types te verrichte, zodat de uitkomst van het onderzoek niet kan worden beïnvloed door ontbrekende app types. Door de app types kort toe te lichten zijn volgende paragrafen makkelijker te begrijpen. Het overzicht is als volgt:

Categorie	*App type*	*Omschrijving*
Native	Android & iOS	Bij de ontwikkeling van dit app type wordt gebruik gemaakt van de ontwikkel SDK van de native besturingssystemen (bijvoorbeeld de Android SDK en de iOS SDK). Apps van dit type werken alleen op het bijbehorend besturingssysteem, maar hebben wel toegang tot native functionaliteit van het besturingssysteem.
Web	Mobile web Client-Side web	Bij het app type Web wordt gebruik gemaakt van diverse web talen (HTML, JavaScript en CSS) om een app via een webbrowser benaderbaar te maken. Apps van dit type zijn door alle besturingssystemen die beschikken over een web browser te benaderen, maar hebben geen toegang tot native functionaliteit.
Cross-platform	Hybrid	Door gebruik te maken van de Web view mogelijkheden van het Native besturingssysteem, wordt een zogenoemde 'bridge' gemaakt tussen de Native taal en de programmeertaal van het gebruikte framework. Hierdoor is communicatie tussen bijvoorbeeld web taal (JavaScript) en verschillende besturingssystemen mogelijk, waardoor gebruik van native functionaliteit mogelijk is. Door gebruik te maken van een web taal kan de app voor diverse platformen beschikbaar worden gemaakt.
	Interpreted	Middels een per besturingssysteem verschillende abstractie laag tussen de Native APIs en de gebruikte programmeertaal, kan worden gecommuniceerd met componenten uit het besturingssysteem. Door gebruikt te maken van de abstractielaag is de app na ontwikkeling op meerdere platformen toegankelijk, waarbij toegang is tot de native functionaliteit.
	Cross-Compiling	Door gebruik te maken van de cross-compiling methode kan de ontwikkelaar door het realiseren van een bepaalde input voor een generator code laten genereren. Doordat de generator de input omzet naar code voor diverse platformen is de app na ontwikkeling beschikbaar voor meerdere besturingssystemen. Ook in dit geval kan gebruik worden gemaakt van native functionaliteit.

Bovenstaande tabel is een korte samenvatting van de gedefinieerde app types. Voor de volledige beschrijving verwijs ik door naar ***Bijlage F – Onderzoeksrapport, Hoofdstuk 4.1***.

5.7 Aspect benadering

Gedurende het onderzoek heeft zich een moment voorgedaan waarbij over de diverse app types allerlei verschillende informatie was vergaard. Waar bijvoorbeeld bij het type Hybrid veel bekend was over de portabiliteit en de werking van de architectuur hierachter, was bij de Android app juist weer veel bekend over de opbouw en structuur van een app. Als voorbeeld het volgende:

Bij het app type Hybrid werd dieper ingegaan op de werking van het Hybrid principe, en hoe dit app type bijvoorbeeld toegang kan krijgen tot de native functionaliteiten. Bij het 'type' Android wordt echter ingegaan op één van de implementaties van het app type Native en niet zozeer de voordelen en werking van het app type. Dit was niet het enige verschil, zo had ik bij het app type Interpreted een aantal frameworks geselecteerd die op eenzelfde manier werken als wat het app type representeert, terwijl ik bijvoorbeeld bij het app type Cross-Compiling al dieper was ingegaan op een framework door bijvoorbeeld ontwikkelomgeving, programmeertaal en testmogelijkheden te achterhalen. Doordat het in deze twee voorbeelden totaal verschillende gegevens betreft, kon geen vergelijking worden gemaakt tussen de diverse app types. Om dit probleem op te lossen heb ik een tweetal keuzen gemaakt, de app types worden onderverdeeld aan de hand van aspecten, en per app type wordt een te onderzoeken framework gekozen. Beide keuzes licht ik nader toe.

Door de app types onder te verdelen in aspecten, is de informatie gevonden over de app types gelijk aan elkaar waardoor een vergelijking plaats kan vinden. Bij de onderverdeling van app types komen zowel technische aspecten uit de ontwikkelstraat, als app type specifieke aspecten naar voren. Onder technische aspecten uit de ontwikkelstraat wordt verstaan aspecten zoals programmeertaal, ontwikkelomgeving en testmogelijkheden. Onder app type specifieke aspecten wordt verstaan of er bijvoorbeeld toegang is tot de native functionaliteit van de smartphone. Alle aspecten bij elkaar definiëren het app type. Zie ***Bijlage F – Onderzoeksrapport, Hoofdstuk 4.2*** voor het volledige overzicht van aspecten.

Doordat de app types door meerdere frameworks worden gerepresenteerd en al deze frameworks een andere invulling aan de opgestelde aspecten geven, is ervoor gekozen om één framework per app type te onderzoeken. Per app type is het meest belovende framework gekozen, wat is bepaald aan de eerder verkregen informatie in het onderzoek. Gedurende het onderzoeken van de verschillende app types zijn bepaalde namen vaker naar voren gekomen dan andere, waardoor ik per framework een duidelijk beeld had of hier wel of niet voldoende informatie over gevonden kon worden. Aan de hand van de functionaliteiten die het framework biedt is gekozen voor het meest belovende framework. Dit laatste is ook in samenspraak met 42 BV gedaan, zodat het bedrijf een eventuele voorkeur uit kon spreken.

Door de aspectenstructuur binnen de app types en het gekozen framework aan te brengen is het mogelijk geweest uiteindelijk een vergelijking te maken tussen de verschillende frameworks. Deze vergelijking werd mogelijk doordat van elk framework dezelfde informatie bekend was, waardoor ik deze informatie in een vergelijkingstabel heb kunnen plaatsen.

5.8 Herzien hoofd- en deelvragen

Het bedrijf heeft gedurende de eerste periode van het onderzoek een voortschrijdend inzicht ontwikkeld. Aansluitend op de in de vorige paragraaf beschreven verandering van structuur, heeft dit voortschrijdend inzicht ervoor gezorgd dat het bedrijf een nieuwe richting aan het onderzoek wil geven. In deze nieuwe richting worden de onderwerpen die belangrijk zijn voor het bedrijf beter belicht dan in de oude richting. Doordat dit van invloed is geweest op de initiële opzet van het onderzoek, heb ik de deel- en onderzoeksvragen moeten herzien. Tevens is daarbij ook de hoofdvraag geherformuleerd als volgt:

> *Welke mobiele applicatie ontwikkelmogelijkheden, resulterend in app types, sluiten aan op de werkprocessen en vaardigheden van 42 BV en zijn daarbij commercieel gezien verzilverbaar om in de praktijk toe te passen?*

De vernieuwde deelvragen zijn gebaseerd op de eerder opgestelde deelvragen, maar zijn dusdanig geherformuleerd, dat zij beter aansluiten op het onderzoek en de nieuwe wensen van 42 BV. Om de hoofdvraag te kunnen beantwoorden heb ik informatie nodig over drie verschillende onderwerpen, namelijk informatie over de app types, de werkprocessen en vaardigheden van 42 BV en de commerciële appsmarkt. Tussen het eerste en tweede aspect moet tevens een vergelijking worden gemaakt, om de mate van overeenkomsten tussen de twee te kunnen bepalen. Hierdoor ben ik uiteindelijk op de volgende deelvragen met bijbehorende onderzoeksvragen uitgekomen:

Deelvraag 1 *Welke app types bestaan er en hoe ziet de ontwikkeling van deze app types eruit?*

- Welke app types bestaan er?
- Hoe ziet de ontwikkelstraat per app type eruit?
- Tegen welke overige problemen lopen ontwikkelaars aan (user experiences)?

Deelvraag 2 *Hoe ziet de huidige ontwikkelstraat van 42 BV eruit en welke beoordeling kan, in vergelijking daarmee, per app type worden gegeven?*

- Hoe ziet de huidige ontwikkelstraat van 42 BV eruit?
- In welke mate zijn aspecten en gebruikte technologieën uit de huidige ontwikkelstraat terug te vinden bij de diverse app types?

Deelvraag 3 *Welke vraag naar applicaties, specifiek voor bepaalde platformen, is er vanuit de apps markt?*

- Naar welke platformen is er vanuit de klanten van 42 BV veel vraag?
- Wat zegt het marktaandeel van besturingssystemen binnen de mobiele markt?

5.9 Toepassen wegingsfactor

De diverse aspecten van zowel de huidige ontwikkelsituatie als die van de app types zijn onderzocht en bekend, waardoor de resulterende waardes met elkaar vergeleken kunnen worden. Om hier een conclusie in de vorm van een advies naar het bedrijf in mogelijk te maken, heb ik in overleg met het bedrijf per aspect een wegingsfactor in de vorm van te behalen punten bepaald. Het is mogelijk een totaalscore van 100 punten te halen, waarvoor het app type aan alle eisen en wensen van het bedrijf moet voldoen. De punten zijn verdeeld over de verschillende te beoordelen aspecten, waardoor dus per aspect een bepaalde wegingsfactor wordt gegeven. Bij het bepalen van deze wegingsfactor is uitgegaan van de prioriteiten die het bedrijf stelt aan de aspecten. Om dit te verduidelijken het volgende voorbeeld:

42 BV geeft aan dat het bedrijf een hoge prioriteit stelt aan de gebruikte ontwikkelomgeving binnen een framework. Hierdoor kan op het aspect ‘ontwikkelomgeving’ in totaal 10 van de 100 punten worden gescoord. Binnen 42 BV wordt voornamelijk gebruik gemaakt van twee ontwikkelomgevingen, namelijk Eclipse en IntelliJ. Wanneer tijdens de ontwikkeling van een app middels het gekozen framework gebruik kan worden gemaakt van zowel Eclipse als IntelliJ, scoort het app type 100% van de score op dit aspect. Indien alleen gebruik kan worden gemaakt van één van de twee ontwikkelomgevingen, Eclipse of IntelliJ, dan scoort het app type 80% van de score. Er zijn nog diverse andere varianten op de score van dit aspect mogelijk, bijvoorbeeld wanneer geen van beide ontwikkelomgevingen gebruikt kan worden (0%) of een ontwikkelomgeving gebruikt wordt die lijkt op Eclipse of IntelliJ (60%). De wegingsfactor is in samenspraak met de opdrachtgever vastgesteld.

Door de hiervoor genoemde methode wordt per aspect een score bepaald, die bij elkaar opgeteld een totaalscore vormen. Aan de hand van deze totaalscore kan een overzicht worden gemaakt, waaruit kan worden afgeleid welk app type het meest aansluit bij de wensen van 42 BV. Het overzicht is te vinden op de volgende pagina. Opvallend aan dit overzicht is dat de individuele frameworks zijn beoordeeld middels percentages, zoals ook naar voren is gekomen in het voorgaande voorbeeld. De wegingsfactor van de aspecten wordt echter bepaald middels een puntenaantal. Dit heb ik met opzet gedaan, zodat de wegingsfactor van de diverse aspecten kan worden aangepast, zonder dat daardoor ook de frameworks opnieuw beoordeelt moeten worden. Wanneer een lezer het aspect ‘ontwikkelomgeving’ belangrijker vind dan de tien punten die het momenteel toegekendt heeft gekregen, zou hij dit kunnen aanpassen naar 20 punten, zonder ook de puntverdeling van de diverse frameworks aan te passen. In het geval van een 100% score hoeft dat namelijk niet, doordat deze mee schaalt met de wegingsfactor. Door op deze manier te werk gegaan heb ik het doorvoeren van eigen inzichten in het overzicht laagdrempelig gehouden, wat het onderzoek voor meer mensen interessant kan maken.

Te beoordelen aspect	*Telt mee voor (%)*	*Punten*	*Android*		*iOS*		*Web*		*PhoneGap*		*Titanium SDK*		*Corona SDK*	
Ontwikkelomgeving		10	100%	10,00	60%	6,00	100%	10	100%	10	80%	8,00	80%	8
Programmeertaal		10	100%	10,00	30%	3,00	70%	7	70%	7,00	70%	7,00	0%	0
Test mogelijkheden		10	100%	10,00	70%	7,00	70%	7	70%	7,00	50%	5	0%	0
Distributiewijze		4	40%	1,60	40%	1,60	100%	4	40%	1,60	40%	1,60	40%	1,60
Automated Build Tool		10	100%	10,00	70%	7,00	100%	10	100%	10	30%	3,00	0%	0
Sonar		6	100%	6,00	0%	0	50%	3	50%	3	50%	3	0%	0
Sub-totaal		*50*		*47,60*		*25*		*41,00*		*38,60*		*27,60*	*1,20*	*9,60*
Community		8												
Tools	50,00%		100%	4,00	100%	4,00	100%	4,00	100%	4,00	60%	2,40	100%	4,00
Omvang	25,00%		100%	2,00	100%	2,00	100%	2,00	70%	1,40	40%	0,80	40%	0,8
Cultuur	25,00%		100%	2,00	0%	0	100%	2,00	100%	2,00	100%	2,00	0%	0,0
Maturity		12												
Leeftijd	50,00%		100%	6,00	100%	6,00	100%	6,00	80%	4,80	70%	4,20	30%	1,80
Issue management	25,00%		100%	3,00	100%	3,00	70%	2,10	100%	3,00	100%	3,00	30%	0,90
Documentatie	25,00%		100%	3,00	100%	3,00	100%	3,00	70%	2,10	70%	2,10	70%	2,10
Sub-totaal		*70*		*67,60*		*42,60*		*60,10*		*55,90*		*42,10*		*19,20*
Overige baten														
Native APIs		9	100%	9,00	100%	9,00	0%	0	70%	6,30	70%	6,30	40%	3,60
User Interface		9	100%	9,00	100%	9,00	40%	3,6	0%	0	80%	7,20	80%	7,20
Offline mogelijkheden		3	100%	3,00	100%	3,00	30%	0,9	100%	3	100%	3	100%	3
Portabiliteit		9	0%	0,00	0%	0,00	100%	9	90%	8,10	70%	6,30	70%	6,30
Totaal		100,00		88,60		63,60		73,60		73,30		64,90		39,30

Bron – Onderzoeksrapport (punten overzicht)

In het overzicht is te zien dat er drie hoofdcategorieën worden beoordeeld, namelijk de ontwikkelstraat specifieke eisen (ontwikkelomgeving, programmeertaal, et cetera), de framework specifieke eisen (community en maturity) en de app type specifieke eisen (Native APIs, User Interface, et cetera).

De eerste categorie, de ontwikkelstraat specifieke eisen, zijn het belangrijkst voor het bedrijf, omdat hiermee de aansluiting op de huidige bedrijfssituatie wordt gevonden.

De tweede categorie is belangrijk voor de kwaliteit en ontwikkelsnelheid van de software. De kwaliteit wordt hierdoor bepaald omdat binnen deze categorie wordt aangetoond hoe snel een framework zich doorontwikkeld en hoe met issues wordt omgegaan. De snelheid wordt bepaald doordat binnen deze categorie wordt aangetoond hoeveel beschikbare tools binnen het framework bestaan. Ook wordt aangegeven in hoeverre hulp kan worden gevonden binnen de community en de hoeveelheid documentatie die beschikbaar is. Dit verkort het leerproces van de ontwikkelaars.

De app type specifieke eisen, zijn op portabiliteit na het meest van belang voor de klanten omdat dit uiteindelijk hetgeen is wat zij te zien krijgen. Dit ziet de klant in de vorm van de gebruikte User Interface, maar ook bijvoorbeeld in de mogelijkheid tot gebruik van camera en GPS functionaliteiten.

5.10 Onderzoek conclusie

De conclusie van het onderzoeksrapport bestaat uit een aantal delen. In eerste instantie heb ik aangegeven dat uitgaande van de verzamelde cijfers, het antwoord op de hoofdvraag Android is. Dit zou echter geen correct advies zijn als dit het enige is wat ik stel, omdat ik als onderzoeker weet dat er meer zaken spelen dan alleen de gegeven cijfers.

Gedurende het onderzoek heeft 42 BV aangegeven dat zij zelf al een beeld hebben bij het commerciële aspect, waardoor de derde deelvraag van het onderzoek is komen te vervallen. Ondanks dat het bedrijf dit commerciële aspect zelf kan meewegen, heb ik voor de conclusie van het onderzoek wel rekening gehouden met deze factor. Dit heb ik gedaan om het bedrijf een richting te kunnen geven in het vervolg proces wat zij moeten doorlopen. Hierdoor is de conclusie uitgebreider geworden.

In het eerste deel van de conclusie heb ik de ideaalsituatie heel beknopt beschreven, deze bestaat uit vier punten als volgt:

1. De gebruiker heeft het gevoel dat hij met een Native applicatie werkt (Native Experience).
2. Alle functionaliteiten van de smartphone kunnen worden benut (Native APIs).
3. De applicatie kan na ontwikkeling direct op meerdere platformen draaien (Portabiliteit).
4. De ontwikkelstraat van het app type moet aansluiten op de ontwikkelstraat die momenteel wordt gehanteerd binnen 42 BV (Ontwikkelsituatie).

Bron – Onderzoeksrapport appsmarkt (conclusie)

Deze vier eisen aan de ideaalsituatie houden geen rekening met het commerciële aspect van de mobiele markt. Het onderzoek wijst uit dat er app types bestaan die aan maximaal drie van de vier gestelde punten van de ideaalsituatie voldoen, maar nooit allemaal.

	Native Experience	*Native APIs*	*Portabiliteit*	*Ontwikkelsituatie*
Android	++	++	--	++
iOS	++	++	--	--
Web	--	--	++	+
PhoneGap	--	+	++	+-
Titanium	+	+	+	-
Corona	+	-	+	--

Te zien is dat het ideale app type (nog) niet bestaat. Om toch aan de slag te kunnen moet het bedrijf bepalen aan welke drie van de vier punten uit de ideaalsituatie zij de meeste prioriteit stellen. Dit moet ook in overleg met de klant worden gedaan, omdat elke klant een eigen visie op het gestelde probleemdomein heeft. Per gestelde prioriteit is een ander advies van toepassing. Daarom heb ik besloten een overzicht van de verschillende mogelijke keuzen te maken die overwogen kunnen worden, waarbij ik het meest geleden nadeel aangeef. Afhankelijk van de keuze van het bedrijf kan het gekozen framework per klant veranderen, omdat de wensen van deze klanten elke keer anders kunnen zijn. Ter verduidelijking het volgende voorbeeld te vinden op de volgende pagina.

Het Native app type scoort op drie van de vier punten uit de ideaalsituatie aanzienlijk goed. Zo heeft het app type toegang tot de Native APIs, is de User Interface native (Native Experience) en sluit de ontwikkeling heel goed aan op de huidige ontwikkelsituatie van 42 BV. Er is echter één nadeel, de portabiliteit van een native app is heel slecht. Een app ontwikkeld voor Android werkt niet op iOS en vice-versa, waardoor om ondersteuning te bieden voor beide besturingssystemen meerdere apps ontwikkeld moeten worden. Dit heeft als gevolg hogere ontwikkelingskosten en een langer ontwikkeltraject.

Dit is het moment waarop de communicatie met de klant vanuit het bedrijf belangrijk is. Indien de klant heel veel prioriteit stelt aan de eerste drie genoemde punten, en portabiliteit minder belangrijk vind, kiest de klant er zelf voor om dieper in de buidel te tasten om de ontwikkelingskosten van de native app ontwikkeling te bekostigen. Indien de klant dit niet ziet zitten en de app die zij willen niet per se bijvoorbeeld de Native Experience hoeft te bevatten, kan het bedrijf beter kiezen voor de Hybrid oplossing in combinatie met een extern User Interface framework. Op deze manier heb ik ook mijn advies naar het bedrijf kenbaar gemaakt. Doordat de derde deelvraag uit het onderzoek is gehaald, heb ik dit laatste niet zelf kunnen concluderen. Hier moet het bedrijf, dan wel in overleg met de klant, een eigen beslissing in maken.

6 Fase 3 – Prototype app

In de derde fase van het afstudeertraject is een prototype van de app ontwikkeld. Doordat deze fase deels in overlapping met de tweede fase plaatsvond, is in eerste instantie niet een app type gekozen op basis van de conclusie van het onderzoek. De conclusie was namelijk nog niet getrokken bij aanvang van deze fase, meer hierover in ***Paragraaf 7.1 – Procesevaluatie***.

Doordat de opdrachtgever gedurende de eerste periode van deze fase op vakantie ging, moest voorafgaand hieraan een beslissing worden gemaakt met welk app type ontwikkeling ik tijdelijk een prototype ga uitwerken. Ondanks dat het onderzoek nog niet af is geeft het bedrijf aan dat hun interesse uitgaat naar iOS, waardoor is besloten dat ik mij gedurende de eerste periode hier mee bezig houd. De keuze tot iOS is echter tijdelijk en staat daardoor nog niet vast, na de vakantie van de opdrachtgever kon deze nog worden gewijzigd aan de hand van het onderzoeksrapport. Dit laatste is uiteindelijk ook gebeurd, na de vakantie van de opdrachtgever is gesteld dat het bedrijf het meest geïnteresseerd is in een prototype gemaakt middels PhoneGap. Meer hierover in ***Hoofdstuk 7 – Procesevaluatie***.

6.1 Doel prototype

Het doel van de te maken prototypes is het inzicht wat gedurende het onderzoek is verkregen te toetsen in de praktijk. In het onderzoeksrapport zijn een aantal problemen naar voren gekomen waar het bedrijf in geïnteresseerd is in hoeverre dit de ontwikkeling van apps belemmert, wanneer zij een app type in een realistische bedrijfssituatie toepassen. De vraag die hierbij het doel van het prototype definieert is als volgt:

> ***Tegen welke problemen loopt het bedrijf aan wanneer zij beginnen met app ontwikkeling door middel van het gekozen framework?***

Een aantal punten zijn al onderzocht in het onderzoeksrapport, maar het bedrijf verwacht ook dat er onverwachtse problemen of keuzemomenten voordoen in een praktisch gezien realistische situatie. Als praktisch gezien realistische situatie wordt gebruik gemaakt van een recent bedrijfsproject, Goudvink. Eerder is al aan bod gekomen dat het bedrijf voor deze desktop applicatie ook een mobiele variant wilt ontwikkelen. Deze app wordt daarom als uitgangspunt voor de te ontwikkelen prototype genomen.

6.2 iOS ontwikkeling

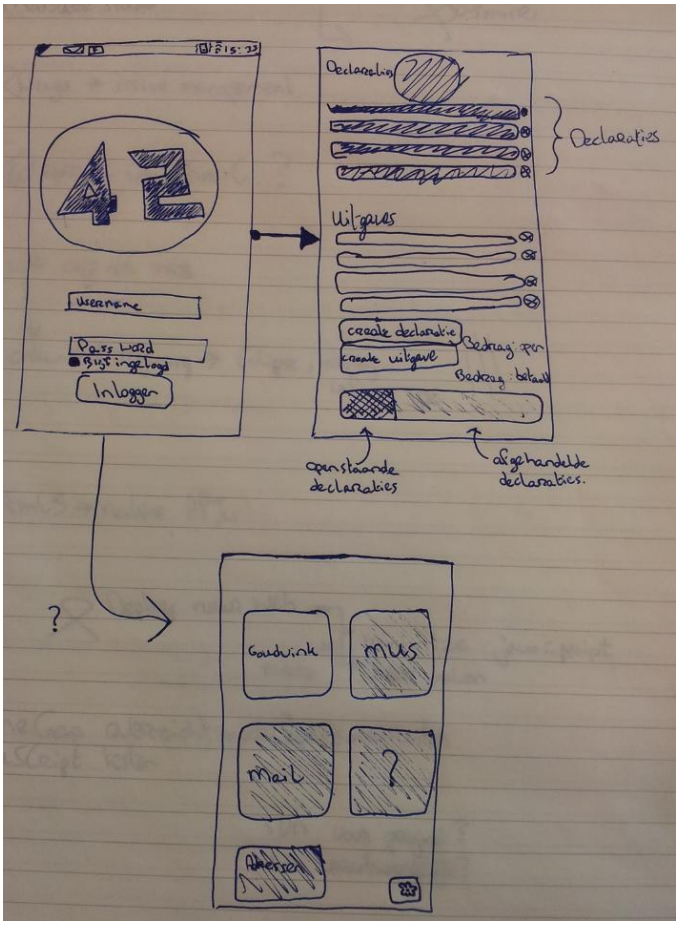
De fase is begonnen met het ontwikkelen van een app voor iOS. Gedurende de eerste periode van deze fase heb ik er bewust voor gekozen nog geen details voor de functionele requirements op te stellen voor de app. De nadruk ligt hier op details, omdat ik wel degelijk kennis van de te maken app had doordat ik eerder in mijn afstudeerperiode heb mee ontwikkeld aan de back-end van het Goudvinkproject.

De reden dat ik in eerste instantie geen details van de functionele requirements op wilde stellen, is dat ik mij gedurende deze eerste periode niet heb willen focussen op de te maken functionele requirements, maar op het verkennen van de mogelijkheden van de Objective-C programmeertaal. Daarbij is er nog geen conclusie voor het onderzoek getrokken. Het onderzoek moet onafhankelijk zijn

van de te maken app, maar door eventueel bekende (details van de) requirements is de conclusie te makkelijk te beïnvloeden in het voordeel van de gestelde requirements. Gedurende de overlap tussen fase twee en drie heeft dus geen officieel moment plaats gevonden om de definitieve functionele requirements van de app vast te leggen, dit is pas op een later tijdsstip gedaan.

Om toch in de goede richting te werken heb ik de ontwikkelde test-app wel gebaseerd op het Goudvinkproject, waardoor alleen de specifieke app details mogelijk niet overeen komen met de wensen van de opdrachtgever.

Om met Objective-C gewend te raken ben ik begonnen met het ontwikkelen van een User Interface (UI), waarbij ik gebruik heb gemaakt van een aantal schetsen die ik had bedacht om een passende user interface op te zetten. De eerste schets was als volgt:

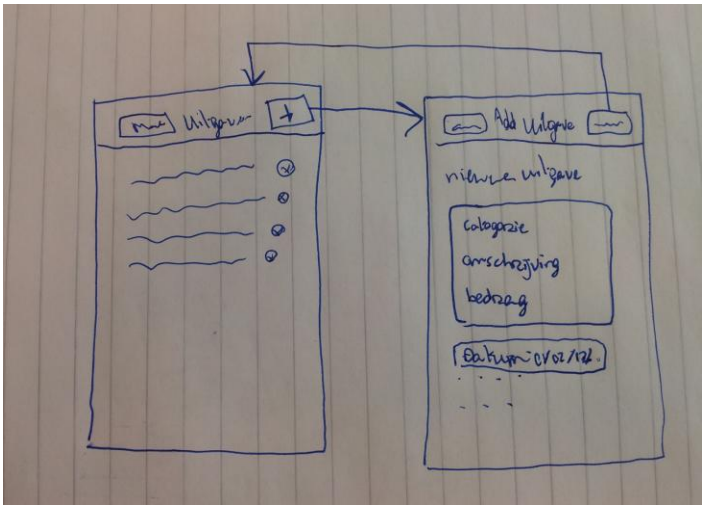


Eerste User Interface schets

Te zien is dat bij het tweede scherm zowel de uitgaven als de declaraties getoond worden, twee knoppen worden toegevoegd om zowel een nieuwe uitgave als een nieuwe declaratie aan te maken, en een snelle overzichtsbalk onderaan het scherm aanwezig is om in één keer een duidelijk beeld te krijgen van hoeveel declaraties nog openstaan. Echter, na een tijdje hieraan bezig te zijn geweest liep ik al gauw

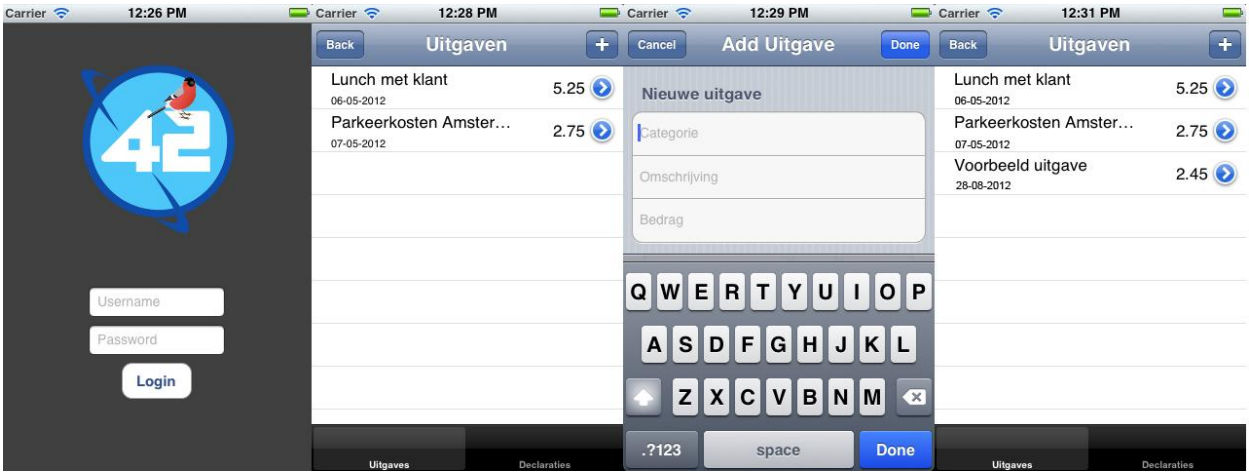
tegen het probleem met schermgroottes aan. Het scherm is niet groot genoeg om de gehele user interface zoals ik die had bedacht weer te geven. Daarbij kwam ik door regelmatig gebruik bepaalde iOS specifieke zaken op te zoeken erachter dat schermen binnen iOS vaak op een heel andere manier worden opgebouwd dan ik tijdens het maken van mijn schetsen had bedacht.

Binnen iOS apps wordt namelijk vaak gebruik gemaakt van een navigatie balk bovenaan de app, waarbij per scherm één specifiek onderwerp van de app wordt getoond. De schets die voor mij hier uit voort kwam is als volgt:



Uitgavenoverzicht en uitgave toevoegen User Interface schets

Door steeds nieuwe user interfaces te schetsen en bij te werken heb ik steeds verder kunnen ontwikkelen aan de app. Tevens zijn een aantal functionaliteiten toegevoegd aan de user interface. Het resultaat van de ontwikkelde user interface en functionaliteit is als volgt:

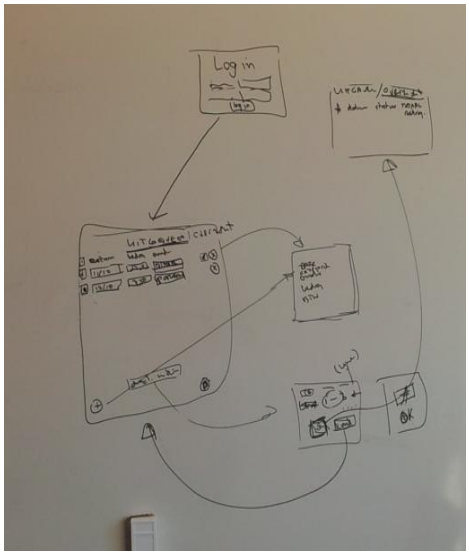


Bron – Ontwikkelde iOS app

Op het moment dat het onderzoeksrapport af was en de opdrachtgever terug van zijn vakantie, is besloten dat aan de hand van de resultaten van het onderzoek het bedrijf het meest geïnteresseerd is in ontwikkeling van apps middels het PhoneGap framework. Doordat hierdoor een ander app type (namelijk Hybrid) wordt gebruikt moet en dit niet is te combineren met iOS ontwikkeling, heb ik laatst genoemde stop moeten zetten. Voorafgaand aan de PhoneGap ontwikkeling heb ik de functionele requirements achterhaald. Meer hierover in de volgende paragraaf.

6.3 Vaststellen requirements

Het vaststellen van de functionele requirements van de app is gedaan aan de hand van een gesprek met de Goudvink productowner, Eric Meijer. Voorafgaand had ik al voorkennis van het Goudvinkproject, doordat ik hier zelf aan had mee ontwikkeld. In dit gesprek hebben we dan ook niet specifiek gekeken naar hoe bepaalde functionaliteiten moeten werken (details van de requirements), maar meer naar welke functionaliteiten terugkomen in de app en hoe deze worden weergegeven. Het leek mij een goede aanpak dit aan de hand van user interfaces te doen. Dit gesprek resulteerde uiteindelijk in een schets op het whiteboard.



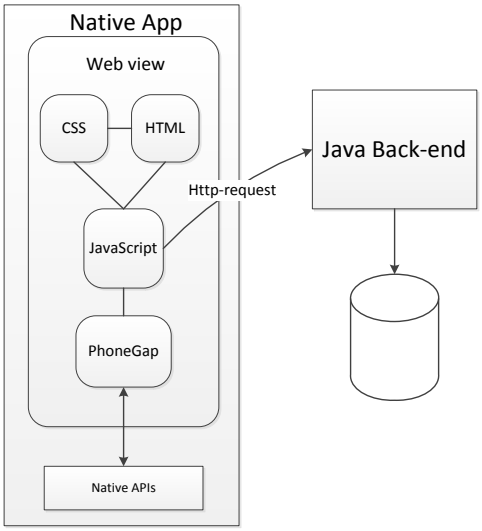
Bron – Functional Requirements vergadering

Het is geheel begrijpelijk dat de afbeelding hierboven nauwelijks leesbaar is, maar zelfs wanneer deze is uitvergroot is dit niet het geval. Doordat het een schets betreft gaat het dan ook met name om het verhaal wat per getekend scherm in de schets verteld is.

Alle informatie die ik heb verkregen gedurende het gesprek met de productowner, inclusief de verduidelijking van de schermen, heb ik uitgewerkt in een requirements document. In dit document heb ik per scherm de benodigde functionaliteit uitgeschreven. Vervolgens is een overzicht van alle functionaliteiten gemaakt, waarbij ik per functionaliteit een prioriteit heb toegekend aan de hand van het gesprek met de productowner. Dit document is te vinden in ***Bijlage E – Goudvink app requirements***.

6.4 Werking PhoneGap

In het onderzoeksrapport had ik de achterliggende werking van PhoneGap apps uitgewerkt. De werking van het systeem heeft voor een aantal problemen gezorgd. PhoneGap apps worden geschreven middels HTML, CSS en JavaScript. Bij het opstarten van een PhoneGap app creëert de native app code een web view waarin de JavaScript code zal draaien. Doordat deze web view wordt aangemaakt door de native app code, ontstaat er een zo genoemde bridge tussen de native code en de app code, waarover gecommuniceerd kan worden. PhoneGap profiteert van deze mogelijkheid tot communicatie door een extra Javascript API te implementeren in de web view, die de communicatie tussen de native code en web view afhandelt.



Bedrijfspresentatie – 42 gaat mobiel (door Wilco Stuyfersant)

Het grote nadeel bij gebruik van PhoneGap is dat door gebruik te maken van een web view, er geen native experience kan worden geboden. Het in de volgende paragraaf beschreven probleem wordt deels veroorzaak door de werking van PhoneGap.

6.5 Same Origin Policy

Een probleem waar ik gedurende de fase van PhoneGap ontwikkeling tegenaan liep is de Same Origin Policy. Dit is een belangrijk beveiligingsprincipe geïmplementeerd in browsers om Cross Site Scripting tegen te gaan. Cross Site Scripting is een techniek waarbij JavaScript code wordt geïnjecteerd in webpagina's om op die manier ongewenste activiteiten uit te voeren, waaronder het stelen van data.

Scripts van een ander domein dan waarin het script normaal gesproken draait worden dus standaard geblokkeerd. Deze beperking leverde mij gedurende deze fase een probleem op omdat ik gegevens vanaf een extern domein probeer op te vragen van de Goudvink server, die in een ander domein is gelokaliseerd. Doordat ik in eerste instantie niet zeker wist of het inderdaad om de Same Origin Policy gaat, heb ik dit eerst vastgesteld. Om dit te doen heb ik een test scenario opgesteld.

Indien het om de Same Origin Policy beveiliging gaat, moet het mogelijk zijn om informatie van de server op te vragen, mits de browser denkt dat het systeem zich in hetzelfde domein bevind als de Goudvink back-end server. De gepoogde aanvraag van *localhost* naar *test.goudvink.42.nl* werkt niet. Om te simuleren dat de browser denkt dat de app zich in hetzelfde domein bevind als *test.goudvink.42.nl* heb ik een virtual host opgezet, wat gedaan wordt door de host file van de lokale computer zodanig aan te passen dat *test.phonegap.42.nl* verwijst naar het *localhost* adres (*127.0.0.1*). Wanneer ik vervolgens via de browser naar *test.phonegap.42.nl* ga, wordt ik buiten het weten van mijn browser om gewoon doorverwezen naar *localhost*. Via de lokaal draaiende webserver kon ik vervolgens via de applicatie verbinding proberen te maken met de Goudvink back-end. Voor de browser ziet het er in dat geval uit alsof *test.phonegap.42.nl* een request suurt aan *test.goudvink.42.nl*. Toch werkte deze oplossing niet.

Ik besloot een collega die hier eerder een onderzoek naar had uitgevoerd te vragen om hulp. Uit zijn onderzoek bleek dat hij ook een aanpassing aan de server bestanden heeft moeten doen om dit werkende te krijgen. In de configuratie van de server moet *localhost* als geaccepteerde host worden toegevoegd. Dit is niet een gewenste configuratie, omdat dit betekend dat iedere computer ter wereld wordt geaccepteerd door de server, doordat vrijwel elke computer een *localhost* heeft. Gezien het in dit geval nog steeds om een testscenario ging, om vast te stellen dat het inderdaad om de Same Origin Policy gaat, heb ik besloten de *test.phonegap.42.nl* (wat tevens mijn *localhost* adres is door de eerdere aanpassing in het host bestand) tijdelijk toe te voegen aan de serverconfiguratie. Uiteindelijk bleek het inderdaad om een serverconfiguratie te gaan waardoor het opvragen van data na het toevoegen van het *test.phonegap.42.nl* adres wel werkte.

Hierdoor heb ik een tijdelijke oplossing weten te creëren voor mijn probleem. De nadruk ligt hierbij op tijdelijk, omdat deze oplossing niet gaat werken bij mobiel gebruik. Om deze oplossing werkende te krijgen heb ik namelijk het host bestand van de computer moeten aanpassen zodat de browser *test.phonegap.42.nl* als *localhost* ziet. Dit is niet gewenst, omdat dit zou betekenen dat iedere gebruiker zijn eigen host bestand aan moet passen alvorens hij de Goudvink app kan gebruiken.

Doordat ik nu wist wat het probleem precies was heb ik heel specifiek naar een oplossing kunnen zoeken. Om *test.goudvink.42.nl* als geaccepteerde bron in mijn PhoneGap project toe te laten, moet de platform specifieke configuratie van het besturingssysteem worden aangepast. In dit geval moest ik een Android PhoneGap project in het Android configuratie bestand (*AndroidManifest.xml*) aangeven dat *test.goudvink.42.nl* een vertrouwde bron is.

Hierdoor is het probleem aan de clientkant opgelost en kan er data worden opgevraagd van de server. Nu resteerde alleen nog de oplossing aan de serverkant, waarvoor ik tijdelijk had gekozen een oplossing door te voeren die beveiligingsissues met zich mee kan brengen. Het bedrijf is al vaker tegen dit probleem aangelopen, waarbij nooit een definitieve keuze is genomen om het probleem voorgoed op te lossen. Na een overleg hierover heeft het bedrijf geconcludeerd dat het tijd wordt dat dit probleem wordt aangepakt en daartoe hebben zij een andere medewerker op dit probleem gezet. Deze medewerker gaat een definitieve oplossing voor dit probleem zoeken en doorvoeren. Tot die tijd kan ik verder zoals het nu geïmplementeerd is.

6.6 Data opslag

Vrijwel iedere applicatie werkt met data die op de één of andere manier moet worden opgeslagen voor gebruik. Zo ook de Goudvink app. Bij gebruik van PhoneGap kon ik kiezen tussen verschillende opslag methodes.

HTML5 Local Storage

Deze optie is ideaal om te gebruiken wanneer de app ook offline gebruikt dient te worden. De data die wordt opgeslagen kan naar gelieve een dag, een week of zelfs jaren worden opgeslagen. In het geval van Goudvink maakt deze optie het mogelijk data tijdelijk offline op te slaan, om deze vervolgens bij de eerst volgende connectie met het internet te synchroniseren met de server. Deze synchronisatie feature dient door de ontwikkelaar geïmplementeerd te worden.

Een nadeel van deze methode is dat door nieuwe wetgevingen de toestemming van de gebruiker om bestanden op te slaan is vereist, alvorens bestanden opgeslagen mogen worden. Daarbij zijn deze opgeslagen bestanden vaak door de gebruiker in te zien en aan te passen.

HTML5 Sessionstorage

De tweede optie is de sessionstorage. Door middel van sessionstorage kan de ontwikkelaar data opslaan, die vervolgens gedurende de hele sessie beschikbaar blijft. Een sessie eindigt wanneer de browser of tab wordt gesloten, of na een bepaalde periode van inactiviteit. Deze optie is ideaal om op een gemakkelijke manier data tussen verschillende pagina's van de app te delen. Daarbij is het wel zo dat dit alleen binnen hetzelfde domein kan.

Lokale SQLite database

Door de communicatie bridge tussen PhoneGap en de Native APIs, is het met PhoneGap mogelijk om data lokaal in een database op te slaan. Het voordeel ten opzichte van de localstorage van HTML5 is dat het zoeken aan de hand van een query sneller is dan door een volledige mapping heen te loopen. Dit speelt echter pas een rol wanneer de database behoorlijke groottes aan begint te nemen.

Gekozen oplossing

Gezien het offline gebruik van de app, met daarbij een synchronisatie optie, een functionele feature betreft die niet eerder is vastgesteld, besloot ik aan de opdrachtgever de vraag voor te leggen of deze functionaliteit gewenst is. Mijn verwachting daarbij was dat de opdrachtgever in eerste instantie deze functionaliteit niet benodigd vind. De reden hiervoor is dat het binnen Nederland vrij aannemelijk is dat onze doelgroep altijd wel een internet connectie beschikbaar heeft, zeker met smartphones. Het is naar de mening van de opdrachtgever inderdaad nog niet van toepassing om deze feature in de Goudvink app te implementeren. Lokale opslag is daardoor niet specifiek van belang, wat mijn keuze uiteindelijk op sessionstorage heeft laten vallen. Wellicht dat het offline werken wel een leuke toevoeging is in een later stadium van de Goudvink app. Voor het prototype is het echter achterwege gelaten en wordt gebruik gemaakt van HTML Sessionstorage, die data bij kan houden totdat het scherm gesloten wordt.

6.7 Performance

Tijdens ontwikkeling van de Goudvink app ben ik een situatie tegengekomen, die in een later stadium van het Goudvinkproject een performanceprobleem op kan leveren.

De situatie is als volgt: wanneer ik door middel van mijn app de declaraties van een bepaalde gebruiker opvraag, resulteert dit in een JSON (JavaScript Object Notation) object waarin de volgende zaken staan.

1. Alle declaraties van de opgevraagde gebruiker (logisch).
2. Alle uitgaven die bij deze specifieke declaraties horen (minder logisch).
3. Alle changelog items van de uitgaven (geheel niet logisch).

Informatie twee en drie zijn minder tot niet logisch doordat deze niet benodigd zijn om de declaraties aan de gebruiker te tonen. Hierdoor wordt meer informatie uit de database opgevraagd dan nodig.

Momenteel gezien levert dit voor het Goudvinkproject nog geen probleem op, omdat de database alleen nog is gevuld met beperkte test-data. De database gaat echter behoorlijk snel groeien wanneer Goudvink in productie gaat. Het feit dat door één request aan de database drie verschillende lijsten van objecten in hun geheel worden opgevraagd, terwijl eigenlijk maar één lijst nodig is, kan in de toekomst voor een performanceprobleem zorgen. Daarbij viel mij ook op dat de JSON objecten die ik teruggestuurd kreeg geen identifier waardes toegekend krijgen. Ook dit laatst genoemde is nadelig voor de performance. Het betreft hier echter geen performanceprobleem voor de serverkant, maar voor de clientkant. Wanneer er namelijk een zoek actie wordt uitgevoerd binnen het resulterende JSON object, moet alle aanwezige data worden gecontroleerd op een match. Bij gebruik van identifiers stopt het zoeken zodra een match is gevonden.

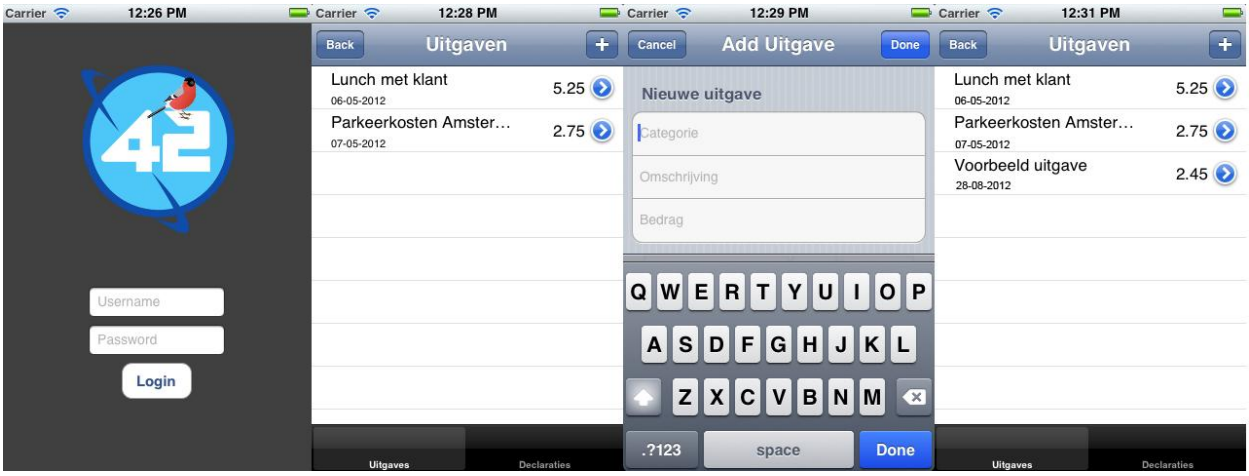
Door de kennis die ik heb opgedaan gedurende de eerste fase van mijn afstuderen, wist ik dat het eerst genoemde probleem, de hoeveelheid data die wordt teruggestuurd, waarschijnlijk te maken had met de Hibernate (een JPA implementatie) configuratie van de Java back-end. In dit geval bevat namelijk het declaratie object meerdere uitgaven, die op zijn beurt weer een relatie heeft met meerdere geschiedenis items. Relationele data objecten kunnen in Hibernate *Eager fetched* zijn, wat inhoud dat als hier een aanvraag op komt Hibernate gelijk alle objecten inclusief relatie objecten teruggeeft. Toen ik hier echter naar op zoek ging bleek dat deze configuratie niet was gespecificeerd in het back-end systeem van Goudvink, wat inhoud dat Hibernate zijn default optie aanhoudt *Lazy fetched* (alleen de informatie teruggeven zonder bijbehorende relatie objecten).

Ondanks dat er bij mij dus verwarring ontstond over waarom dan toch alle informatie werd teruggegeven, is het mogelijk dat Hibernate met opzet zo is geconfigureerd dat alle data wordt teruggegeven. In dit laatste geval functioneert het systeem naar behoren. Ik besloot de architect van Goudvink te vragen in hoeverre hij hier van af wist. Hierbij heb ik ook kenbaar gemaakt dat ik zowel aan de serverkant als clientkant een performanceprobleem voorzie in de toekomst. Dit probleem was echter nog niet bekend bij de architect en ook hij voorziet hier problemen. Hier moet naar gekeken worden, maar hier ben ik gezien de afstudeeropdracht niet verantwoordelijk voor. Ook de architect benadrukt dat dit buiten de scope valt en dat ik dus verder moet roeien met de riemen die ik heb.

Gezien de app die ik maak een prototype betreft, heb ik besloten voorlopig dit probleem bij ontvangst binnen de app op te lossen. Daarbij manipuleer ik de data zo, dat deze bij ontvangst wordt uitgesplitst en opgeslagen in aparte arrays. Hierdoor is binnen de applicatie dus toegang tot een losse declaratiearray, een losse uitgavenarray en een losse changelogarray. Tevens voeg ik direct identifiers aan de nieuwe objecten toe, wat voorkomt dat ik met een hele grote ongestructureerde data variabele aan de slag moet gaan. Daarbij maakt het toevoegen van de identifiers het voor mij makkelijker om de benodigde data in de arrays te zoeken.

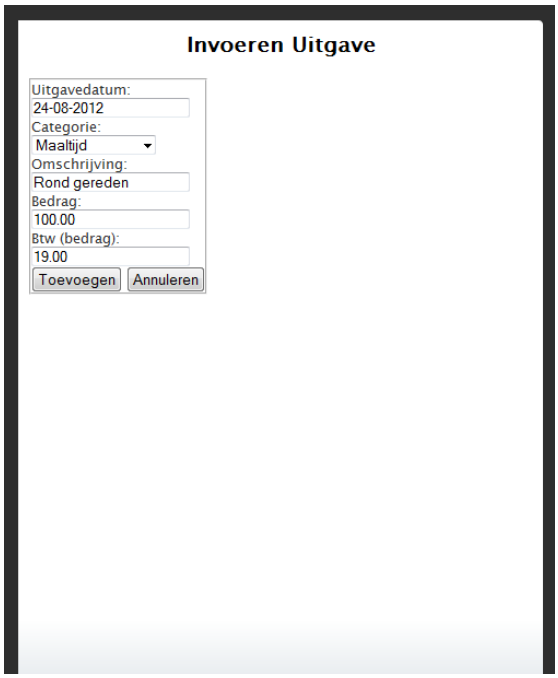
6.8 Resultaat fase drie

Gedurende de derde fase van de afstudeeropdracht heb ik mij zowel bezig gehouden met iOS als PhoneGap ontwikkeling. Het resultaat van iOS ontwikkeling is al naar voren gekomen in ***Paragraaf 6.2 – iOS ontwikkeling*** en is als volgt:



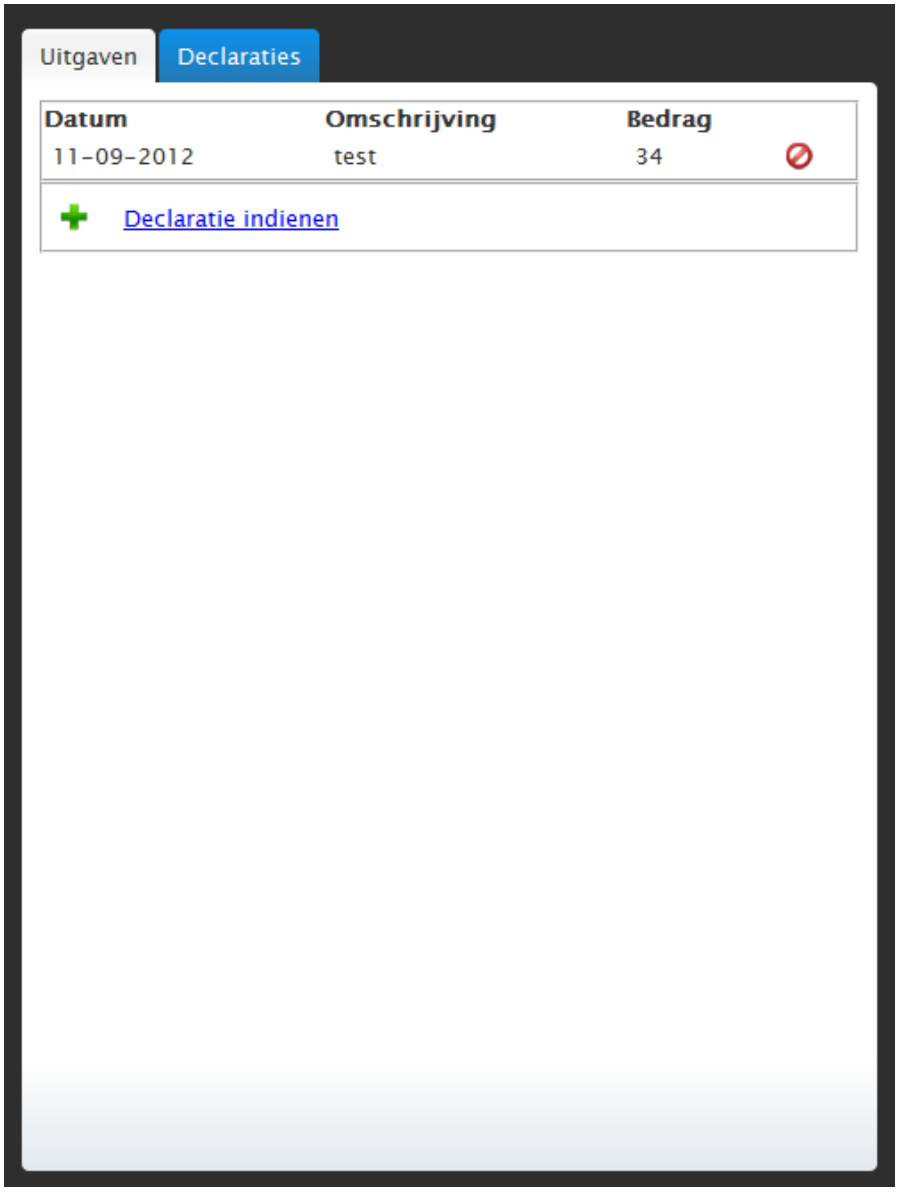
Resultaat iOS ontwikkeling

Gedurende het tweede deel van de derde fase heb ik mij bezig gehouden met PhoneGap ontwikkeling. De hoeveelheid tijd waarmee ik mij met PhoneGap heb bezig kunnen houden is door uitloop van het onderzoek gering gebleven. Toch is een deel van de Goudvink app ontwikkeld, waarbij minimale aandacht aan de vormgeving is besteed. Het resultaat is zowel hieronder als op de volgende pagina te vinden.



PhoneGap app – Invoeren uitgave

PhoneGap app – Declaratie overzicht



PhoneGap app – Uitgave overzicht

7 Procesevaluatie

Het proces is naar mijn mening over het algemeen goed verlopen. Zoals gedurende elk project zijn er goede punten, maar ook punten van kritiek te benoemen. Deze punten komen per fase aan bod in de volgende sub-paragrafen.

7.1 Fase één

Gedurende de eerste fase ben ik tegen het feit aangelopen dat er teveel kennis bij één persoon zat. Dit kwam naar voren door de afwezigheid van de Scrummaster, de persoon die van alle projectleden de meeste kennis en ervaring heeft op zowel technologiegebied als het Goudvinkproject. Gedurende de eerste paar dagen was dit probleem gemakkelijk uit te stellen door aan een ander deel van de sprintbacklog te werken, maar steeds vaker liep ik tegen problemen aan waarbij het toch handig was als de Scrummaster aanwezig was. Een voorbeeld van een dergelijk probleem is de applicatie van de ontwikkelomgeving deployen naar de testomgeving. Niemand binnen de projectgroep had enig idee hoe dit moest, waardoor de oplevering aan het eind van de sprint in gevaar kwam. Dit liet mij realiseren dat teveel kennis bij één persoon zat, wat als gevolg had dat we hier als projectgroep actief mee aan de slag gingen. Dit heeft er uiteindelijk voor gezorgd dat bijvoorbeeld niet enkel de Scrummaster meer kon deployen naar de testomgeving, maar de gehele projectgroep.

Ondanks dat de bedrijfsvisie die 42 BV hanteert met betrekking tot Scrum op momenten zijn nut bewijst en heel fijn werkt, heeft het ook situaties gekend die laten zien dat het niet altijd vlekkeloos verloopt. Gedurende de eerste fase van het afstuderen heb ik zoals eerder beschreven de rol van Scrummaster tijdelijk op mij genomen. Tijdens één van de sprintplanning zag ik vooraf al aankomen dat ik gedurende die sprint waarschijnlijk eerder klaar zou zijn met mijn werkzaamheden dan dat de sprint is afgelopen. Doordat vooraf geen details van de volgende requirement bekend zijn, heb ik gedurende die sprintplanning ervoor gekozen een ‘nader te specificeren’ requirement op de backlog te zetten.

De foute keuze die ik hier heb gemaakt is om de sprint te starten alvorens alle teamleden geheel bekend zijn met alle items die op de backlog staan. Mijn hoop was daarbij dat de Scrummaster tegen de tijd dat ik klaar was met de te maken requirement, weer aanwezig zou zijn. Hij had namelijk wel een duidelijk beeld bij wat er als volgt ontwikkeld moest worden. Door de keuze toch te starten met de sprint heb ik mij afhankelijk gemaakt van de terugkeer van de Scrummaster. In dit geval had ik er naar mijn mening dan ook beter voor kunnen kiezen een extra vergadering met de productowner of opdrachtgever aan te vragen, zodat zowel voor mijzelf als de overige teamleden duidelijk wordt waar het volgende item op de backlog uit gaat bestaan. Hierdoor waren wij niet afhankelijk geweest van een afwezig teamlid. Tevens benadrukte dit nogmaals hoe belangrijk het is dat kennis gedeeld moet worden over het hele team. Indien dit niet mogelijk is, moet per direct naar een oplossing worden gezocht, zodat een heel project team niet afhankelijk wordt van één teamlid. Hier ben ik vanaf dat moment ook actief mee aan de slag gegaan.

Ondanks dat de rol van Scrummaster niet altijd vlekkeloos verliep, ben ik toch blij dat ik de kans heb aangegrepen om de rol op mij te nemen. Het is uiteindelijk toch een hele ervaring om binnen een officieel projectteam als Scrummaster op te kunnen treden.

7.2 Fase twee

Ook gedurende de tweede fase zijn er zowel goede momenten als minder goede momenten geweest. Met deze laatste categorie begin ik. Een punt van kritiek op mijn voorbereiding voor het onderzoeksrapport is dat ik vooraf wel voldoende heb nagedacht over de onderzoeksopzet, keuze van methoden en verkleining van het probleemdomein door onderverdeling over hoofd en deelvragen, maar niet voldoende heb nagedacht over welke specifieke informatie ik precies van de app types wil vinden. Dit uitte zich na een aantal weken. Naar voren is gekomen dat er zich een moment heeft voorgedaan waarop ik mij realiseerde dat er veel verschillende informatie over de app types bekend was, waardoor het niet mogelijk was deze gegevens met elkaar te vergelijken.

Het voortschrijdend inzicht, resulterend uit mijn onderzoek, heeft het bedrijf in staat gesteld snel een beslissing te kunnen nemen over welke punten uit het onderzoek wel en niet belangrijk zijn geweest. Er zitten dan ook twee kanten aan dit verhaal. Enerzijds is meer werk verricht dan nodig was, doordat door het voortschrijdend inzicht bepaalde informatie kwam te vervallen. Daarbij moest een groot gedeelte aan nieuwe informatie gezocht worden om alsnog de apps met elkaar te kunnen vergelijken op de vastgestelde punten. Voornamelijk door dit laatste is er een grote achterstand ontstaan op de planning. Wanneer vooraf was besloten welke informatie belangrijk is voor het bedrijf, had ik mij vanaf het begin hier op kunnen focussen.

De andere kant van het verhaal is dat door in eerste instantie veel verschillende informatie op te zoeken, het bedrijf beter in staat is geweest aan te geven wat zij wel en niet belangrijk vinden. Door vooraf over specifieke onderwerpen na te denken had er misschien tijd bespaard kunnen worden waardoor de uitloop op de planning minder had kunnen zijn, maar het is lastig om te zeggen dat dit per definitie voorkomen had kunnen worden. Gedurende de eerste periode van het onderzoek zijn namelijk heel veel onderwerpen gevonden die voorafgaand aan dit deel nog niet bekend waren. Hierdoor heeft zowel het bedrijf als ik een voortschrijdend inzicht ontwikkeld, wat het verdere verloop van het onderzoek qua snelheid bevorderde. Tevens is het maar de vraag of het bedrijf en ik zonder dit voortschrijdend inzicht wel in staat waren geweest de belangrijkste aspecten te bepalen, zonder daarbij bepaalde aspecten te vergeten. Naar mijn mening heeft dit dan ook de kwaliteit van het rapport in grote mate verbeterd, omdat de kans dat aspecten zijn vergeten naar mijn idee aanzienlijk kleiner is door het onderzoek wat tot dat moment was gedaan.

De uiteindelijk gemaakte keuzes hebben ervoor gezorgd dat de onderzoeksfase een aantal weken is uitgelopen. Echter, door regelmatige gesprekken met de opdrachtgever is door deze uitloop wel een onderzoeksrapport opgeleverd waar zowel 42 BV, als andere lezers van het rapport iets mee kunnen. Met name dit laatst genoemde stelt mij in staat om volledig achter de gemaakte keuze te staan om meer tijd aan het onderzoeksrapport te besteden om de kwaliteit ervan te verbeteren. De kwaliteit van het rapport is namelijk in grote mate verbeterd ten opzichte van wat het anders was gebleven.

7.3 Fase drie

Met het oog op het voorafgaand aan het afstudeertraject opgestelde afstudeerplan, is deze fase geheel anders verlopen dan verwacht. Waar in eerste instantie was bedacht deze fase in te richten door veel verschillende prototype apps te maken, is met name door een lang uitgelopen onderzoek besloten deze fase te beperken tot één app.

Aan het begin van deze fase is in overleg met de opdrachtgever besloten, ondanks dat het onderzoek nog niet geheel af was, dat ik mij gelijktijdig met de afronding van het onderzoek bezig houdt met iOS ontwikkeling. Dit is met het oog op het onderzoek een vreemde keus geweest. Ondanks dat de conclusie van het onderzoek nog niet was getrokken, kon wel al met redelijke zekerheid gesteld worden dat iOS met het oog op ontwikkelstraat en mogelijkheden van het app type toch één van de minder interessante opties is voor het bedrijf. Toch is er vanuit het bedrijf nog steeds veel interesse voor dit app type, doordat er vanuit het commerciële aspect van het onderzoek veel vraag is naar iOS. Gelijktijdig met deze beslissing is afgesproken dat de keuze afhankelijk van de onderzoeksconclusie nog kan veranderen naar aanleiding van de uiteindelijke conclusie van het onderzoek. Dit is ook gebeurd.

Mede door deze wisseling van framework is de uiteindelijke prototype app qua functionaliteit niet volledig afgekomen. Dit vind ik enerzijds jammer, indien ik verder had kunnen gaan met iOS ontwikkeling was de Goudvink app dan wel helemaal, dan wel voor een groot deel afgekomen. Toch is dit niet erg geweest voor het bedrijf met het oog op het doel van het prototype. Het doel was namelijk achterhalen tegen welke onverwachtse problemen kan worden aangelopen tijdens de ontwikkeling van een app middels het geselecteerde framework. Door met PhoneGap aan de slag te gaan zijn een aantal problemen ontdekt waar eerder niet aan gedacht was zoals de Same Origin Policy. Tevens heb ik door de switch van framework ervaring op kunnen doen met twee ontwikkelmogelijkheden, namelijk die van iOS en die van PhoneGap.

Om niet tweemaal dezelfde app te ontwikkelen heb ik gedurende deze twee periodes rekening gehouden met afwisseling in de ontwikkeling. Waar ik tijdens iOS ontwikkeling mij met name heb bezig gehouden met het ontwerpen van een gebruikersvriendelijke interface (mede om met de mogelijkheden van iOS bekend te raken), heb ik mij tijdens de PhoneGap ontwikkeling meer op de technische aspecten van ontwikkeling gericht.

Ik ben dan ook van mening dat het, zeker voor mijn eigen leerervaring, een goede afwisseling is geweest. Gedurende de weken dat ik voor iOS heb ontwikkeld heb ik een stuk ervaring op kunnen doen met en native besturingssysteem. Indien ik mij alleen op PhoneGap had georiënteerd zou ik deze ervaring missen, waardoor ik met het oog op het probleemdomein niet het verschil tussen de twee had kunnen bevestigen. Achteraf gezien ben ik dan ook tevreden over deze laatste fase.

8 Productevaluatie

Ondanks dat de gemaakte app niet geheel af is, ben ik zeer tevreden over het eindresultaat van het afstudeertraject.

8.1 Java back-end

Gedurende de eerste fase van de opdracht is een deel van Java back-end ontwikkeld voor de Goudvink applicatie. De focus lag hierbij op het ontwikkelen van het systeemdeel wat communicatie tussen de front-end en de database mogelijk maakt, zodat ik middels dit systeemdeel in de laatste fase van het afstudeerproject met de database kon communiceren. Gedurende deze fase zijn twee relevante requirements voor het mobiele app van Goudvink ontwikkeld, namelijk het invoeren en tonen van uitgaven en het indienen en tonen van declaraties. Tevens heb ik een extra requirement kunnen ontwikkelen, namelijk de functionaliteit om van een declaratie een changelog bij te houden. Laatst genoemde was echter minder van belang voor de derde fase.

Bij de afronding van deze fase van het afstudeertraject was het systeemdeel wat benodigd is voor gebruik in de derde fase dus ontwikkeld. Daarbij is voldaan aan de vooraf opgestelde Definition of Done. Echter is niet alleen het ontwikkelde systeemdeel resultaat van de eerste fase, ook heb ik veel kennis opgedaan over bijvoorbeeld het testen van applicaties en het gebruik van het JPA framework. Hierdoor heb ik een hoop geleerd over diverse technieken die bij 42 BV vrijwel in de dagelijkse situatie worden toegepast. Doordat zowel de benodigde requirements zijn gemaakt, als voldaan is aan de Definition of Done, ben ik tevreden over dit product.

8.2 Onderzoeksrapport

Voor het onderzoeksrapport is vooraf een hoofdvraag opgesteld die gedurende het onderzoek beantwoord moest worden. De hoofdvraag is op het commerciële aspect na ruimschoots beantwoord, wat het uiteindelijke rapport een waardevol document maakt voor het bedrijf. In hoeverre het bedrijf beslissingen gaat nemen op basis van het rapport is voor mij helaas nog niet bekend.

Tijdens het onderzoek heb ik rekening moeten houden met diverse stakeholders, waardoor het op sommige momenten lastig bleek te zijn om van alle informatie een gestructureerd document te maken. Toch ben ik uiteindelijk in staat geweest een document samen te stellen, waar niet alleen 42 BV iets mee kan, maar ook andere bedrijven eventueel een conclusie uit kunnen trekken. Hierdoor is naar mijn inzien het opgeleverde product van grote waarde voor het bedrijf. Ik ben dan ook erg te spreken over het uitgevoerde onderzoek en resulterend onderzoeksrapport.

8.3 Goudvink app

Gedurende de derde fase van het afstudeertraject heb ik een mobiele applicatie ontwikkeld voor het Goudvink declaratie systeem. Ondanks dat de app niet in zijn geheel af is gekomen, ben ik toch erg tevreden over deze fase. De fase was niet lang, maar toch heb ik geleerd te werken met twee heel verschillende programmeertalen, namelijk Objective-C middels het iOS besturingssysteem en JavaScript middels het PhoneGap framework. Daarbij heb ik toch het idee dat ik het maximale eruit heb gehaald wat mogelijk is.

9 Beroepstaken

Voorafgaand aan het afstuderen heb ik een aantal beroepstaken gekozen waar ik mij gedurende deze afstudeeropdracht in heb bewezen. In de volgende paragrafen licht ik per beroepstaak toe hoe ik aan deze beroepstaak gewerkt heb.

9.1 Uitvoeren analyse door definitie van requirements

Inventariseren, analyseren en beschrijven van de gewenste informatievoorziening, zodat wordt vastgelegd welke processen objecten creëren en gebruiken.

Ik heb op verschillende momenten in het afstudeertraject aan deze beroepstaak gewerkt. Gedurende de eerste fase ben ik in gesprek gegaan met het projectteam van het Goudvinkproject, om de requirements die zij met de opdrachtgever hadden besproken te achterhalen. Gezien de bedrijfsvisie van 42 BV was gedurende deze eerste periode niet veel bekend over de requirements, waardoor ik mij heb moeten aanpassen aan de situatie. Door continu in contact te blijven met de stakeholders en voorafgaand aan de sprintplanningen de requirements die gedurende die sprint worden gemaakt in kaart te brengen, zijn gedurende de eerste fase op meerdere momenten requirements geanalyseerd en uitgewerkt.

Gedurende de laatste fase van het afstudeertraject heb ik aan deze competentie gewerkt door de functionele requirements van de Goudvink app uit te werken. Het uiteindelijke resultaat hiervan is te vinden in ***Bijlage E – Goudvink app requirements***.

9.2 Bouwen applicatie

Bouwen en documenteren van de systeemdelen.

Deze competentie heb ik behaald door gedurende de eerste fase van het afstudeertraject mee te ontwikkelen aan het back-end systeem van de Goudvink applicatie. De focus lag hierbij op functionaliteit ontwikkelen om communicatie tussen de database en een front-end applicatie mogelijk te maken. Gedurende het ontwikkelen van de desbetreffende requirements heb ik regelmatig modellen ontworpen om het te maken, of bestaande, systeemdeel duidelijk in kaart te brengen.

Ook gedurende de laatste fase van het afstudeertraject heb ik aan deze competentie gewerkt, namelijk tijdens de ontwikkeling van een mobiele app. Door te leren werken met Objective-C, JavaScript en het PhoneGap framework zijn twee verschillende prototypes ontwikkeld die in staat zijn met het Goudvink back-end systeem te communiceren. Zo kunnen in de iOS app uitgaven worden toegevoegd en weergegeven, en kan de PhoneGap app uitgaven en declaraties toevoegen en tevens weergeven.

9.3 Uitvoeren van en rapporteren over het testproces

Uitvoeren testplan en opstellen rapportage.

Gedurende de eerste fase van het afstudeertraject is aan het behalen van deze competentie gewerkt. In eerste instantie zijn keuzes gemaakt over het niveau waarop de tests plaats moeten vinden, waarna een keuze is gemaakt tussen het maken van unittests of componenttests. Gezien een Code Coverage van 100% is afgesproken, is gedurende deze fase elke methode getest op een correcte werking hiervan. Om de correcte werking te definiëren is gebruik gemaakt van de opgestelde requirements. Middels rapporten gegenereerd door Sonar, kunnen de bevindingen en dekkingsgraad van de diverse tests worden ingezien.

9.4 Ontwerpen softwarearchitectuur

Bepalen en beschrijven van de verschillende subsystemen en componenten waarin een applicatie is onderverdeeld en de relaties tussen die delen.

Deze beroepstaak is komen te vervallen en vervangen door de ‘Selecteren van standaard software’ beroepstaak. Voorafgaand aan het afstudeertraject had ik verwacht een softwarearchitectuur te moeten ontwerpen voor de te maken mobiele apps. Dit is echter doordat het geen volwaardige app maar een prototype betreft, in veel mindere mate teruggekomen dan vooraf verwacht. Hierdoor heb ik mij onvoldoende kunnen bewijzen in deze beroepstaak.

9.5 Selecteren van standaard software

Adviseren over en/of selecteren van software, zoals een applicatie, framework, databasemanagementsysteem of besturingssysteem.

Aan deze competentie heb ik gewerkt door voor 42 BV een onderzoek te verrichten naar welk app type zij het beste kunnen gebruiken wanneer het bedrijf mobiele apps wilt ontwikkelen. Voor het onderzoek heb ik rekening gehouden met de eisen en wensen van het bedrijf op het gebied van ontwikkelstraat, community eisen en app specifieke elementen. Voor dit onderzoek zijn diverse stakeholders te benoemen waar rekening mee is gehouden, namelijk de ontwikkelaars, het management en de klanten van 42 BV. Bij het onderzoek is rekening gehouden met de kennis en middelen die 42 BV standaard in huis heeft. Hierbij is tevens gekeken naar welke verschillende problemen ondervonden kunnen worden tijdens ontwikkeling van apps. Deze problemen zijn teruggekoppeld op de huidige bedrijfssituatie, waardoor een conclusie kon worden getrokken.

Bijlagen

Bijlage A – Plan van Aanpak

1 Inleiding

Dit document is opgesteld naar aanleiding van de *Goudvink* afstudeeropdracht. Deze opdracht zal het verkennen van de *appsmarkt* betreffen door middel van het project Goudvink. In dit Plan van Aanpak (PvA)zullen deze termen nader worden toegelicht, evenals de opdracht zelf. Het PvA zal de aanpak van het afstudeerproject beschrijven waarmee de opdracht tot een succesvol einde wordt gebracht.

In het document zijn bij het eerste gebruik bepaalde begrippen schuingedrukt. Definities van deze begrippen zoals zij in deze context bedoeld zijn, zijn terug te vinden in de begrippenlijst.

Het document is opgebouwd aan de hand van de volgende hoofdstukken:

- *Hoofdstuk 1 – Inleiding*
- *Hoofdstuk 2 – De afstudeeropdracht*
  In dit hoofdstuk zal de opdracht nader worden beschreven, waarbij zowel de aanleiding als de doelstelling naar voren komen. Ook zullen de diverse fases worden besproken welke worden doorlopen tijdens dit afstudeerproject.
- *Hoofdstuk 3 – Afbakening*
  De afbakening van het project wordt in dit hoofdstuk besproken. Hierbij wordt toegelicht welke zaken wel en niet binnen de projectscope vallen.
- *Hoofdstuk 4 – Randvoorwaarden*
  Aan welke eisen 42 BV en de afstudeerder moeten voldoen komt in dit hoofdstuk naar voren. Hierbij worden ook afspraken genoemd welke vooraf zijn gemaakt.
- *Hoofdstuk 5 – Risicofactoren*
  Elk project heeft met bepaalde risicofactoren te maken. In dit hoofdstuk wordt dieper ingegaan op de specifieke risicofactoren voor dit afstudeerproject.
- *Hoofdstuk 6 – Rollen*
  In dit hoofdstuk zullen de betrokken personen bij het afstudeerproject worden benoemd en aan rollen worden gekoppeld.
- *Hoofdstuk 7 – Planning*
  De globale planning komt in dit hoofdstuk aan bod.
- *Begrippenlijst*
  Een lijst met definities van woorden zoals deze bedoeld zijn in de geplaatste context. Begrippen uit de begrippenlijst zijn bij het eerste gebruik cursief neergezet.

2 De afstudeeropdracht

In dit hoofdstuk wordt de afstudeeropdracht nader toegelicht. Wat is de aanleiding van de opdracht, wat is het doel en uit welke fasen zal deze afstudeeropdracht bestaan zijn daarbij belangrijke zaken die naar voren komen. Het hoofdstuk is als volgt opgebouwd:

- Aanleiding
- Doelstelling
- Werkwijze

2.1 Aanleiding

Er zijn twee hoofdfactoren die aanleiding geven tot deze afstudeeropdracht. De eerste van deze hoofdfactoren is dat 42 BV bij diverse projecten vanuit de klant de vraag krijgt of de te maken applicatie ook beschikbaar wordt gesteld voor mobiel gebruik. De tweede factor is dat er binnen het bedrijf nog geen declaratiesysteem bestaat. Beide aanleidingen worden nader toegelicht.

Binnen de ICT wereld vinden continu nieuwe ontwikkelingen plaats. Eén van de sterkst groeiende op het moment is die van *mobiliteit*, waarmee in dit geval bedoeld wordt plaats- en tijdsonafhankelijk werken. Dit architectuur principe is door de komst van smartphones en tablets sinds een aantal jaar sterk in opkomst, waardoor uitsluitend werken achter de computer tot het verleden behoord. De klanten van 42 BV zien de sterke vooruitgang in deze ontwikkelingen en daardoor rijst steeds vaker de vraag in hoeverre zij hun gebruikte applicaties kunnen integreren in een mobiele omgeving. Binnen 42 BV is geen specialistische kennis over dit onderwerp aanwezig waardoor mobiele oplossingen vaak door middel van een browser-applicatie worden aangeboden aan de klant. Om in de toekomst beter mee te kunnen in de zogenoemde appsmarkt wil 42 BV zich nu gaan richten op de mobiele ontwikkelingen en de obstakels die daarbij komen kijken in kaart brengen. Zo kunnen zij de klant beter van dienst zijn en in hun behoeften voorzien.

Als hulpmiddel tot het richten op de appsmarkt zal gebruik worden gemaakt van een nieuw (intern) project, genaamd Goudvink. Dit project betreft een declaratiesysteem waarin medewerkers van 42 BV kosten die zij hebben gemaakt voor bedrijfsdoeleinden kunnen declareren. Een dergelijk systeem bestaat nog niet binnen het bedrijf.

Kort gezegd is zowel de grotere vraag vanuit de markt naar mobiele applicaties en het momenteel ontbreken van een declaratiesysteem reden voor de afstudeeropdracht.

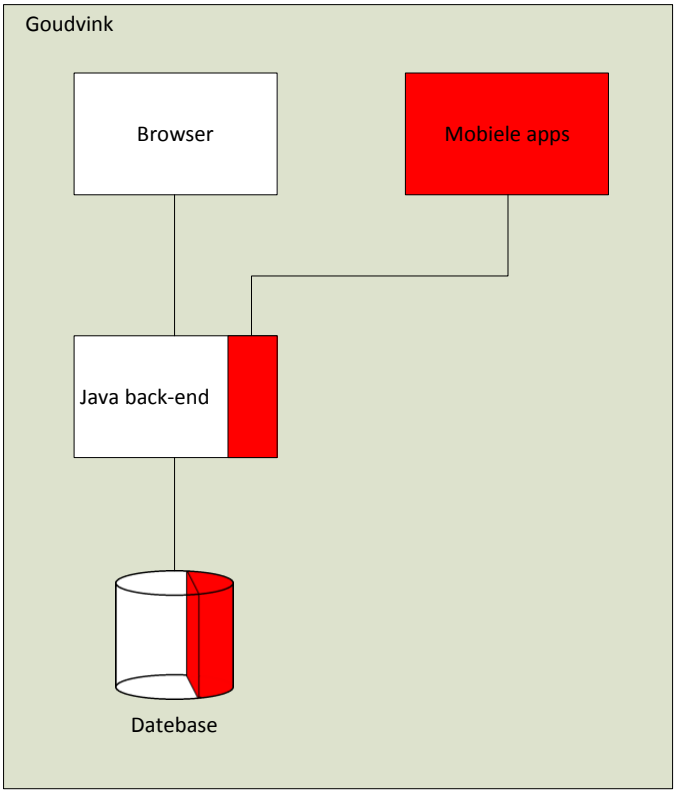
2.2 Doelstelling

Zoals de aanleiding doet vermoeden wil 42 BV eerst meer inzicht in de mogelijkheden binnen de appsmarkt voordat zij definitief hiermee aan de slag gaan. Wat komt er bij kijken wanneer het bedrijf mobiele apps maakt voor zichzelf en haar klanten, welke mogelijkheden tot ontwikkelen zijn er en waar moet rekening mee worden gehouden zijn daarbij belangrijke vragen. Daarbij is het nodig te achterhalen welke overeenkomsten en verschillen er zijn te vinden in de huidige ontwikkelsituatie en die van de appsmarkt. Hierbij kan gedacht worden aan overeenkomsten tussen verschillende talen, nieuwe ontwikkelomgevingen en bijbehorende processen. Deze doelstelling zal worden behaald door middel van een verkennend onderzoek welke nader wordt uitgelegd in de volgende paragraaf.

Naast het meer inzicht verkrijgen in de appsmarkt wil 42 BV ook een mobiel declaratiesysteem die de medewerkers in staat stelt om kosten die zij maken voor bedrijfsdoeleinden te declareren.

2.3 Werkwijze

Project Goudvink is op te delen in een aantal onderdelen, wat globaal is weergegeven in onderstaande afbeelding. Het roodgekleurde gedeelte valt wat Goudvink betreft binnen de scope van de afstudeeropdracht. Hierdoor zijn binnen Goudvink twee aspecten te herkennen die van belang zijn: het *Java back-end systeem* en de *mobiele apps*. De database is alleen onder bepaalde omstandigheden onderdeel van de afstudeeropdracht. Meer hierover in **Hoofdstuk 5 – Risicofactoren**.



Naast het Java back-end systeem en de mobiele apps ontwikkeling zal om de doelstelling te behalen onderzoek worden gedaan naar de verschillende ontwikkelmogelijkheden binnen de appsmarkt. Dit in combinatie met de schoolaspecten van het afstuderen leidt tot de fasering, te vinden op de volgende pagina's.

Fase 1 Ontwikkelen Java Back-end

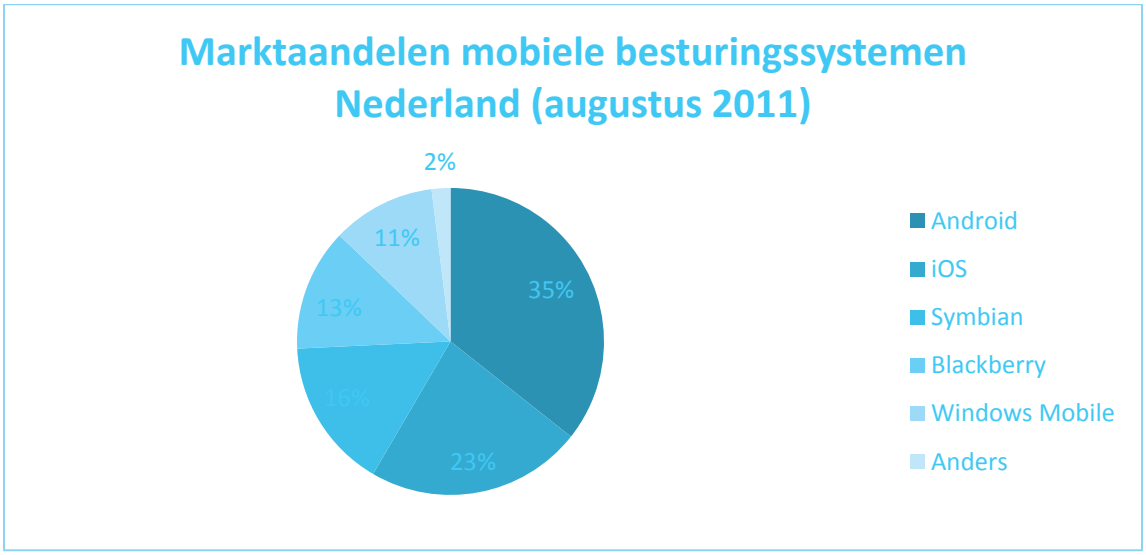
Tijdens aanvang van het afstuderen zal project Goudvink al van start zijn gegaan. Voor het ontwikkelen van mobiele apps is het van belang dat de Java back-end is ontwikkeld waarmee de apps kunnen communiceren. In totaal zullen er vier mensen aan project Goudvink werken. Om een duidelijke scheiding aan te brengen zal de afstudeeropdracht zich richten op de mobiele kant van Goudvink. Van de Java back-end zullen dan ook alleen de onderdelen binnen de afstudeeropdracht vallen die communicatie tussen de mobiele apps en de database mogelijk maken. De overige drie medewerkers richten zich op de *front-end*, het grootste gedeelte van de Java back-end en de Database.

Tijdens deze fase zal deels in groepsverband worden gewerkt, waarbij gebruik wordt gemaakt van de *Scrum methodiek*. Deze fase zal diverse (nog nader te bepalen) activiteiten op het gebied van communicatie tussen mobiele apps en de database bevatten. Tijdens de informatica opleiding wordt geleerd om voor aanvang van een project het *ontwerp* van het te ontwikkelen systeem in detail uit te denken. Bij 42 BV is het gebruikelijk door middel van de Scrum methodiek een *productbacklog* op te stellen waarvan vervolgens per onderdeel (en alleen waar nodig) het ontwerp wordt uitgedacht. In hoeverre dit binnen de scope van de afstudeeropdracht komt te liggen is nog niet bekend. Wel zal worden uitgegaan van de ontwerpen en standaarden van 42 BV.

Fase 2 Onderzoek naar verschillende ontwikkelmogelijkheden

In de tweede fase van het project zal een verkennend onderzoek plaatsvinden naar de verschillende mogelijkheden die bestaan om binnen de appsmarkt applicaties te ontwikkelen. Deze fase zal aan de opvolgende fase een verdere invulling geven.

Binnen de appsmarkt is een grote diversiteit aan apparaten en *besturingssystemen* aanwezig wat het ontwikkelen van een *universele applicatie* sterk beïnvloed. Met name het besturingssysteem bepaald op welke apparaten de nieuwe applicatie zal werken. In augustus 2011 heeft het bedrijf TNS NIPO een onderzoek verricht naar het marktaandeel van mobiele besturingssystemen binnen Nederland. Zij hebben een screening gehouden onder 16.000 smartphonebezitters, wat de resultaten te vinden op de volgende pagina opleverde. Tijdens dat onderzoek is geen rekening gehouden met tablets.



Uit de resultaten blijkt dat er een aantal grote besturingssystemen te herkennen zijn: Google's Android, Apple's iOS, Symbian (oorspronkelijk een samenwerkingsverband waar meerdere fabrikanten deel van uitmaakten zoals Nokia, Ericsson en Samsung, maar vandaag de dag alleen Nokia), RIM's Blackberry en Windows Mobile.

42 BV kiest er vooralsnog voor alleen Google's Android en iOS te onderzoeken. Hier zijn verschillende redenen voor te benoemen.

- Android en iOS hebben samen het grootste marktaandeel in handen.
- Vanuit de klant is er veel vraag naar iPhone apps (welke draaien onder het iOS besturingssysteem).
- 42 BV ziet veel gelijkenissen tussen ontwikkelen voor Android en de huidige ontwikkelmethodieken van het bedrijf. Beide maken namelijk gebruik van de Java programmeertaal.

Aan het ontwikkelen voor meerdere besturingssystemen zit een groot nadeel. Zoals al eerder genoemd bepaald vooral het besturingssysteem of een app wel of niet werkt. De besturingssystemen verschillen dermate van elkaar dat applicaties gemaakt voor bijvoorbeeld Android niet zullen werken op iOS en vice-versa.

Er bestaan diverse mogelijkheden om een applicatie geschikt te maken voor mobiel gebruik op meerdere mobiele besturingssystemen.

- *Volledig browser-based*
  Er kan worden gekozen om de gehele applicatie browser gebaseerd te maken, waardoor voor de front-end gebruik wordt gemaakt van bijvoorbeeld HTML en Javascript. Browser-based applicaties zijn universeel over vrijwel alle platformen (zij hebben een hoge portabiliteit) en is dus een makkelijke oplossing. Deze oplossing wordt momenteel binnen 42 BV gebruikt voor klanten die hun applicatie mobiel willen gebruiken.

- *Native*
  De applicatie kan in een native programmeertaal worden geschreven wat inhoud dat hij specifiek voor Android of iOS is. Dit betekend als men wil dat de applicatie op beide besturingssystemen werkt, hij twee keer 'opnieuw' gemaakt moet worden. Eenmaal in Java (voor Android) en eenmaal in Objective C (voor iOS).

- *Semi-native*
  Wanneer een applicatie semi-native wordt gemaakt, wordt voor meerdere besturingssystemen een applicatie gemaakt. Deze applicaties roepen vervolgens de browserfunctionaliteiten van het besturingssysteem aan. Het voordeel hiervan is dat er maar een klein gedeelte van de code in een native taal geschreven hoeft te worden. De rest van de functionaliteit zal door middel van een universele browsertaal worden geschreven. Dit houdt onder andere in dat de inhoud van de applicatie centraal is aan te passen.

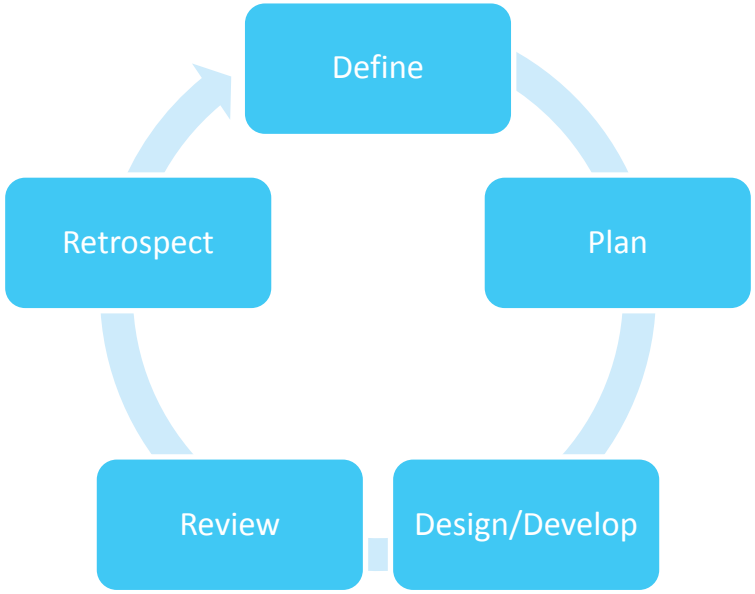
- *Tussentaal*
  Er zijn mogelijkheden tot gebruik van een 'tussentaal' welke bij het compileren ervoor zorgt dat de applicatie voor meerdere besturingssystemen geschikt wordt gemaakt. De mogelijkheden hiervan zijn echter nog niet geheel duidelijk.

Tijdens het onderzoek wordt over deze vier onderwerpen informatie verzameld, de voor en nadelen bepaalt en user-experiences opgezocht. Uit de resultaten van dit onderzoek moet blijken welk van deze technieken voor 42 BV interessant zijn. Vervolgens wordt uit de meest interessante opties een selectie gemaakt waarmee de volgende fase invulling krijgt. Omdat in deze fase vrijwel geen ontwikkeling plaatsvindt (hooguit kleine *"Hello-World"* applicaties) zal er geen gebruik worden gemaakt van sprints of Scrum.

Fase 3 Ontwikkelen prototypes

Vanuit de tweede fase zal een selectie aan tools en technieken komen welke verschillende manieren representeren hoe apps ontwikkeld kunnen worden. Aan de hand van de selectie wordt tijdens deze fase een aantal *prototypes* ontwikkeld van de mobiele app van Goudvink. Dit resulteert in meerdere apps welke communiceren met de eerder ontwikkelde Java back-end.

De fase zal worden opgedeeld in korte periodes waarin prototypes worden opgeleverd. Het proces zal worden ingericht door middel van de Scrum methodiek. Deze periodes worden binnen Scrum ook wel Sprints genoemd. Elke Sprint zal uit de volgende onderdelen bestaan:



- ***Define***
  Tijdens het eerste onderdeel van de cyclus zal de productbacklog worden opgesteld. De productbacklog bestaat uit een set van user stories waar het te maken systeem uit bestaat. Deze set van user stories zal hoogstwaarschijnlijk veranderen gedurende het project waardoor het van belang is dat deze aan het begin van elke Sprint wordt bijgewerkt. Een belangrijk punt daarbij is dat deze fase niet per definitie de enige fase is die mogelijkheid geeft tot wijzigen van de productbacklog. Dit is in elke fase mogelijk.

- ***Plan***
  Tijdens dit onderdeel zal de komende Sprint worden ingepland aan de hand van de productbacklog. Er worden een aantal user stories gekozen (door de productowner) waarbij rekening wordt gehouden met de voortgang van de voorgaande Sprint. Dit resulteert in een Sprint backlog.

- ***Design/Develop***
  Eerder genoemd is al dat het bij 42 BV gebruikelijk is aan de hand van een user-story op de backlog de benodigde ontwerpen te maken waarna men deze user-story ontwikkeld.

    - ***Design***
      Van de user stories zullen waar nodig ontwerpen worden gemaakt. Dit zal alleen worden gedaan wanneer de user-story zo complex is dat een uitwerking van het ontwerp voor meer duidelijkheid zorgt. Hierbij wordt de kennis die is opgedaan tijdens de informatica opleiding toegepast. Er kan worden gedacht aan bijvoorbeeld *Use-case diagrammen* en *klassendiagrammen,* waarbij gedurende elke Sprint voorgaande ontwerpen worden aangepast waar nodig.

    - ***Develop***
      Aan de hand van de Sprint backlog zal een user-story worden ontwikkeld. Voorafgaand hieraan vind waar nodig een ontwerpperiode plaats. Het ontwikkelen wordt gedaan middels het *Test-driven Development* principe. Test-driven development is gebaseerd op de herhaling van kleine ontwikkelcyclussen waarbij eerst een test case wordt geschreven voordat de bijbehorende code wordt ontwikkeld.

- ***Review***
  Tijdens de Review wordt de Sprint door middel van een demo getoond aan de productowner. Hierbij geeft de productowner feedback welke vervolgens wordt meegenomen in de volgende sprint.

- ***Retrospect***
  Tijdens dit onderdeel wordt gekeken naar het procesverloop van de Sprint. Welke zaken goed en minder goed zijn verlopen komen daarbij aan bod. Tevens wordt gekeken hoe het proces veranderd kan worden zodat het een volgende keer beter verloopt. Dit zal in wekelijkse gesprekken met de bedrijfsmentor plaatsvinden.

Op het moment van schrijven is nog erg weinig bekend over Goudvink. Wel zijn een aantal *gebruikersinterfaces* opgesteld voor de browser applicatie waaruit enige functionaliteit is af te leiden. De overige requirements worden achterhaald door middel van interviews met diverse *stakeholders*.

Fase 4 Afstudeerverslag

Gedurende de hele periode zal gedocumenteerd worden over de verrichte werkzaamheden. Aan het eind van de afstudeerperiode is tijd gereserveerd waarin het afstudeerverslag kan worden geschreven en afgerond.

3 Afbakening

De afstudeeropdracht beperkt zich tot het mobiele aspect van Goudvink. Hierdoor zal de focus vooral komen te liggen op het uit te voeren onderzoek en de daarbij behorende ontwikkeling van prototypes. Het Java back-end systeem valt voor het grootste deel buiten de scope van dit project en van dit systeem zal alleen de back-end functionaliteit worden gemaakt welke benodigd is voor communicatie met de mobiele applicatie van Goudvink.

Binnen de appsmarkt zijn vele besturingssystemen te onderscheiden, maar het onderzoek richt zich alleen op Google's Android en Apple's iOS. Overige besturingssystemen worden vooralsnog buiten beschouwing gelaten.

Qua ontwikkelmogelijkheden zal het onderzoek enkel richten op de vier genoemde mogelijkheden: Browser-based, Native, Semi-native en tussentaal.

4 Randvoorwaarden

Om dit project tot een geslaagd einde te brengen zijn er enkele voorwaarden waar aan moet worden voldaan.

De afstudeerder

- Afgesproken is dat de afstudeerder 5 dagen per week aanwezig zal zijn gedurende een periode van 17 weken (85 werkdagen).
- Tijdens de afstudeerperiode worden de werkzaamheden binnen het bedrijf beperkt tot Goudvink en onderdelen daarvan. Er zal niet worden gewerkt aan andere projecten.
- Oplevering van documenten aan 42 BV wordt gedaan middels *Confluence* en bepaalde documenten bedoeld voor school worden als PDF bestand geüpload.
- Er worden diverse prototypes gemaakt.
- Er worden regelmatig presentaties gegeven over de voortgang. Hierbij kan worden gedacht aan een presentatie over de resultaten van het onderzoek en een presentatie over de gemaakte prototypes.

42 BV

- Om voor iOS applicaties te kunnen ontwikkelen dient gebruik te worden gemaakt van Apple's besturingssysteem, Mac OS. Om dit mogelijk te maken stelt 42 BV een *MacBook* ter beschikking gedurende de daartoe benodigde periode.
- Om de prototypes te kunnen testen zal 42 BV bijbehorende devices beschikbaar stellen.
- Aan het distribueren van apps naar devices zijn kosten verbonden, de zogenoemde *developers licenties*. Dergelijke licenties dienen voor het testen beschikbaar te zijn.
- Wekelijks zullen er voortgangsgesprekken zijn met de opdrachtgever/bedrijfsleider.
- Dagelijks zullen er Scrum meetings worden gehouden. Dit zal in de eerste fase in combinatie met het projectteam zijn. In de overige fases zullen dit *stand ups* zijn met medewerkers van 42 BV die op een intern project zitten.
- Bij vragen kan contact op worden genomen met medewerkers van 42 BV. Dit zal vrijwel altijd door middel van een afspraak gaan.

5 Risicofactoren

Tijdens de afstudeerstage zijn een aantal risicofactoren aanwezig welke het afstuderen in gevaar kunnen brengen. Het is daarom belangrijk dat deze risico's vooraf bekend zijn, zodat hier rekening mee kan worden gehouden.

- *Doelstelling wordt niet behaald*

  In de doelstelling van deze opdracht wordt beschreven dat het hoofdzaak is dat 42 BV meer inzicht in de appsmarkt krijgt en dat zij als hulpmiddel gebruik maken van het Goudvink declaratiesysteem. Een mogelijk risico bij het werken aan Goudvink is dat teveel tijd wordt besteed aan het ontwerpen en ontwikkelen van het Java back-end systeem. Wanneer in de eerste fase teveel tijd aan ontwerpen en programmeren wordt besteedt komt de hoeveelheid tijd voor het te verrichten onderzoek in gevaar, hetgeen waar 42 BV juist in geïnteresseerd is. Een reden te meer om vooraf een duidelijk gedefinieerde planning te hebben.

  Een ander risico binnen dit kader is het beperkte aantal uur waar medewerkers aan het Goudvinkproject werken. Hierbij kan ook *uitval* van medewerkers plaatsvinden waardoor bijvoorbeeld de database van Goudvink niet opgeleverd kan worden. In dit geval zal het maken van een database ook binnen de projectscope vallen. Dit is voor veel situaties denkbaar. Dit betekend echter niet dat wanneer teamleden uitvallen de overige onderdelen automatisch binnen de scope van de afstudeeropdracht vallen. Zo komt het browser gedeelte van de afstudeeropdracht te vervallen wanneer de browserfunctionaliteit vanuit Goudvink niet kan worden opgeleverd. Hier moet per situatie een keuze over worden gemaakt.

  Gedurende de rest van het afstudeerproject is het risico aanwezig dat het afstudeerproject uit het oog wordt verloren waardoor er *operationele taken* worden uitgevoerd. Dit zal de planning in gevaar kunnen brengen en zal ten allen tijde voorkomen moeten worden. De volledige focus zal op het afstudeerproject moeten blijven.

- *Te weinig documentatie en ontwerp*
  Het is bij 42 BV ongebruikelijk om voorafgaand aan een project veel tijd te besteden aan het ontwerpen van een te ontwikkelen systeem. De reden hiervoor is de toepassing van de Scrum methodiek. Ontwerpen worden gemaakt naarmate het project vordert en dit gebeurt aan de hand van user stories op de backlog. Wanneer de eerste user-story op de productbacklog wordt gemaakt is de kans groot dat de details van de user-story onderaan de backlog nog geheel onduidelijk zijn. Uit ervaring blijkt dat tijdens de opleiding informatica voorafgaand aan het project veel tijd wordt besteed aan het systeemontwerp, maar gedurende het project vrijwel niet. Wanneer de ontwerp- en ontwikkelfases dicht op elkaar liggen is het risico aanwezig dat te weinig tijd wordt besteed aan het ontwerp van de mobiele apps van Goudvink. Omdat eerdere fases in groepsverband plaatsvinden zal dit risico met name spelen tijdens de derde fase.

- *MacBook en testdevices worden niet beschikbaar gesteld*
  Zoals eerder al is aangegeven, is voor de ontwikkeling van apps voor het besturingssysteem iOS een computer vereist die op Apple's besturingssysteem (Mac OS) draait. Wanneer een dergelijke computer niet beschikbaar wordt gesteld kunnen er geen prototypes voor iOS worden gemaakt. Doordat alle drie de te onderzoeken ontwikkelmogelijkheden een variatie in besturingssysteem vereisen zal hierdoor een groot deel van het afstudeerproject wegvallen. Wanneer dit het geval is zal hoogstwaarschijnlijk een ander mobiel besturingssysteem worden gekozen welke verder onderzocht zal worden.

  Een ander risico is dat 42 BV test devices niet beschikbaar kan stellen. Wanneer test devices niet beschikbaar worden gesteld houdt dit in dat het testen van de apps alleen door middel van simulators plaats kan vinden.

- *Verandering van inzicht*
  Een ander risico wat aanwezig is, is dat tijdens de afstudeerperiode de inzichten op het project anders worden waardoor de opdracht veranderd. Deze kans is in redelijke mate aanwezig. In voorgaande afstudeeropdrachten bij 42 BV is het altijd gebeurd dat het project veranderde naarmate het vorderde. Hoe hier op in wordt gespeeld is per situatie afhankelijk, maar het zal ten alle tijden in overleg met de *betrokkenen* (inclusief school) worden gedaan.

- *Bedrijfsmentor en opdrachtgever niet beschikbaar*
  42 BV is een bedrijf wat zijn werknemers regelmatig *detacheert* aan klanten. Zo kan het voorkomen dat de opdrachtgever/bedrijfsmentor wordt geplaats bij een nieuwe klant. Wanneer dit het geval is, zal een andere werknemer de begeleidende taak op zich nemen.

6 Rollen

Medewerker	*Rol*	*Functie*
Robert Bor	Opdrachtgever, Bedrijfsmentor	Chief Technology Officer
Eric Meijer	Productowner (Goudvink)	CEO
Michel Noppert	Scrum Master, Developer (Goudvink)	Software Developer
Marvin Koot	Stakeholder (Goudvink)	Software Developer
Paulien de Wind	Developer (Goudvink)	Software Developer
Margret Kouwen	Developer (Goudvink)	Software Developer
Wilco Stuyfersant	Scrum Master, Developer	Software Developer

Aan bovenstaande rollen zal gedurende het project de volgende invulling worden gegeven.

- *Opdrachtgever/Bedrijfsmentor*
  Wekelijks zal een gesprek plaatsvinden over het project. Hierbij zullen de problemen waar ik tegenaan ben gelopen en de voortgang worden besproken. Ook worden eventueel veranderde inzichten betreffende de opdracht tijdens deze gesprekken kenbaar gemaakt.

- *Productowner*
  De productowner zal bepalen aan welke functionaliteiten de mobiele app van Goudvink moet voldoen.

- *Scrum Master*
  De Scrum Master is verantwoordelijk voor het Scrum proces tijdens het project. Hij zorgt dat de productiviteit van het team zo hoog mogelijk ligt en onderhoudt de communicatie met de stakeholders. In de eerste fase zal Michel Noppert de rol van Scrum Master op zich nemen. In de derde en vierde fase zal dit de afstudeerder zijn.

- *Developer*
  De developer is verantwoordelijk voor de uitvoer van het ontwikkelproces. In dit geval betreft dat het opstellen van requirements, het maken van ontwerpen en het schrijven van de programmacode.

7 Planning

De afstudeerstage zal een tijdsperiode van 17 weken beslaan. In deze weken zullen een aantal activiteiten worden verricht, welke globaal zijn beschreven in **paragraaf 2.3 – Werkwijze**.

Aan de hand van de vooraf gedefinieerde fases kan de volgende globale planning worden opgesteld.

Week	*Periode*	*Activiteit*
1	7 mei – 11 mei	Opstellen Plan van Aanpak.
2 t/m 4	14 mei – 1 juni	Deel van de Java back-end ontwikkelen.
5 t/m 9	4 juni – 6 juli	Onderzoek naar mogelijke ontwikkeltechnieken binnen de appsmarkt.
10 t/m 15	9 juli – 17 augustus	Ontwikkelen en ontwerpen van meerdere prototypes.
16 en 17	20 augustus – 31 augustus	Schrijven en afronden afstudeerverslag.

Een belangrijke side-note aan deze planning is dat er tijdens deze afstudeerperiode een periode van drie weken vakantie is. Deze drie weken zijn nog niet in de planning meegenomen, maar kunnen op een later tijdstip worden toegevoegd. Mocht dit het geval zijn, zal er rekening worden gehouden met de fasering van het project om een zo gunstig mogelijk tijdstip te kiezen. Daarbij zal ook de einddatum (welke nu op 31 augustus staat) drie weken verplaatsen.

Begrippen lijst

Dit hoofdstuk geeft de begrippenlijst weer van begrippen zoals deze bedoeld zijn in de geplaatste context. De begrippen zijn gesorteerd op alfabetische volgorde.

Appsmarkt
De markt waarin de klant steeds vaker vraagt naar mobiele apps.

Besturingssysteem
Een programma wat tijdens het opstarten van een apparaat wordt geladen. Dit programma voorziet in de functionaliteit om andere programma's uit te voeren. In het grootste deel van het document wordt hiermee gerefereerd naar de volgende besturingssystemen: Android, iOS, Blackberry, Symbian en Windows mobile.

Betrokkenen
Alle partijen die bij het afstudeerproject zijn betrokkenen.

Confluence
Een omgeving die het bedrijf in staat stelt informatie te delen en daarbij discussies te voeren.

Detacheren(detacheert)
Het plaatsen van medewerkers bij een derde partij.

Developers licenties
De licentie die is vereist om mobiele apps te distribueren naar een fysiek apparaat.

Front-end
De visuele gebruikersinterface die interactie tussen de gebruiker en het back-end systeem mogelijk maakt.

Gebruikersinterfaces
De intermediair die interactie tussen de gebruiker en systeem mogelijk maakt.

Goudvink
Een intern project van 42 BV welke het declareren van kosten gemaakt voor bedrijfsdoeleinden mogelijk maakt.

Java back-end systeem
Het systeem wat de verwerking van ontvangen data van de front-end mogelijk maakt.

MacBook
Een notebook van het merk Apple, welke is voorzien van het Mac OS besturingssysteem.

Mobiliteit
Plaats- en tijdsonafhankelijkheid.

Mobiele app
Een applicatie die op een mobiel device (bijvoorbeeld smartphone of tablet) kan draaien.

Ontwerp
De structuur van een systeem.

Operationele taken
De taken die in de dagelijkse bezigheden van 42 BV vallen, maar niet met het afstudeerproject te maken hebben.

Productbacklog
Een set aan functionaliteiten waar het te maken systeem uit bestaat. Aan de hand van de productbacklog worden sprints ingedeeld.

Prototypes
Test applicaties die het declareren van kosten mogelijk maken. De prototypes worden gemaakt aan de hand van de verschillende ontwikkelmethodes die naar voren komen tijdens het onderzoek.

Scrum methodiek
Scrum is een iteratieve Agile software ontwikkelmethodiek die is bedoeld voor het managen van softwareprojecten waarbij applicatieontwikkeling plaatsvind.

Sprints
Een korte iteratie (van minimaal 1 week en maximaal een maand) waarin een deel van het product wordt opgeleverd.

Stakeholder
Een belanghebbende bij dit afstudeerproject.

Stand ups
Een korte ‘vergadering’ waarbij elke aanwezige medewerker vertelt wat hij/zij de dag daarvoor heeft gedaan, wat hij/zij gaat doen en tegen welke problemen hij/zij denkt aan te lopen.

Test-driven Development
Een ontwikkelmethodiek die is gebaseerd op de herhaling van kleine ontwikkelcyclussen waarbij eerst een test case wordt geschreven voordat de bijbehorende code wordt ontwikkeld.

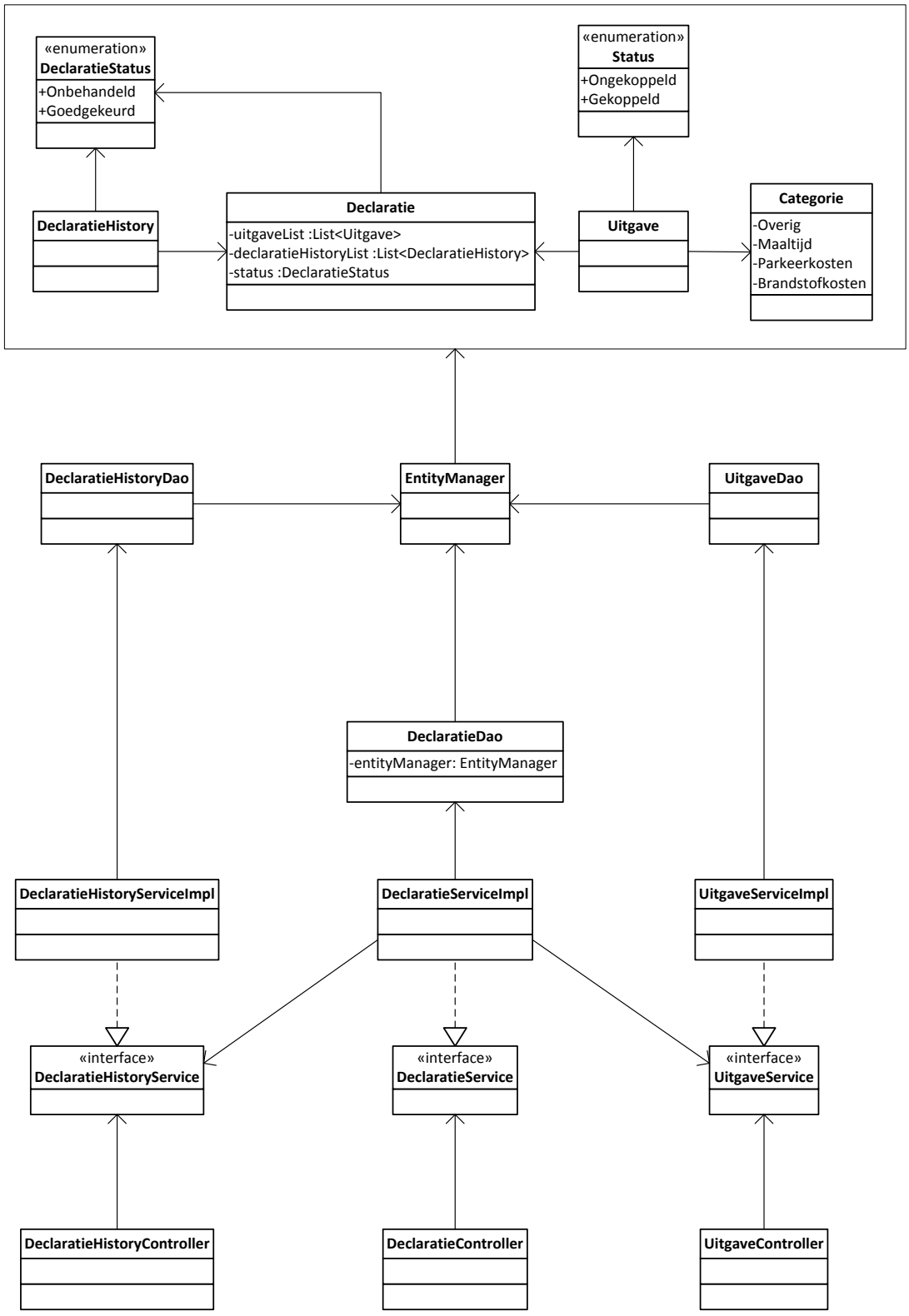
Uitval
Bij uitval van een medewerker is hij/zij gedurende een langere periode niet meer beschikbaar voor het Goudvinkproject.

Universele applicatie
Een applicatie die op meerdere besturingssystemen tegelijk kan draaien.

Use-case diagrammen en klassendiagrammen

Use-casediagram en klassendiagram zijn termen uit de Unified Modeling Language (UML) die een modelmatige taal representeren. Deze taal maakt objectgeoriënteerde analyses voor een informatiesysteem mogelijk.

Bijlage B – Goudvink diagram



Bijlage C – Evaluatieformulieren

Margret Kouwen

Naam geëvalueerde medewerker	Wilco Stuyfersant
Email geëvalueerde medewerker	wilco.stuyfersant@42.nl
Project	goudvink
Functie medewerker	developer
Periode inzet van	Tuesday, April 24, 2012
Periode inzet tot	Tuesday, June 5, 2012
Naam evaluator	Margret Kouwen
Email evaluator	margret.kouwen@42.nl
Functie evaluator	developer
Bedrijfsnaam	42 BV
Opdrachtomschrijving	tijdelijk meeontwikkelen met project zodat de backend redelijk bekend terrein is voor stageaire; gevolgd door onderzoek naar mobiele applicatie
Technisch-inhoudelijke vaardigheden	Goed
Functioneel kwaliteitsbewustzijn	Goed
Technisch kwaliteitsbewustzijn	Goed
Productiviteit	Goed
Software ontwikkelstraat	Goed
Teamspirit	Zeer goed
Sociale vaardigheden	Zeer goed
Communicatieve vaardigheden	Zeer goed
Initiatief	Zeer goed
Presteren onder druk	Goed
Leiderschap	Goed

Commerciële vaardigheden	Goed
Sterke punten (bij voorkeur minstens 3)	Toont initiatief; communiceert goed; kan goed relativeren
Verbeterpunten (bij voorkeur minstens 3)	assertiviteit;
Algemeen / eventuele opmerkingen	fijne collega om mee te werken

Michel Noppert

Naam geëvalueerde medewerker	Wilco Stuyfersant
Email geëvalueerde medewerker	wilco.stuyfersant@42.nl
Project	Goudvink
Functie medewerker	Developer
Periode inzet van	Tuesday, April 17, 2012
Periode inzet tot	Tuesday, June 5, 2012
Naam evaluator	Michel Noppert
Email evaluator	michel.noppert@42.nl
Functie evaluator	Developer
Bedrijfsnaam	42 bv
Opdrachtomschrijving	Een backend deel implementatie bouwen voor het Goudvinkproject.
Technisch-inhoudelijke vaardigheden	Voldoende
Functioneel kwaliteitsbewustzijn	Voldoende
Technisch kwaliteitsbewustzijn	Voldoende
Productiviteit	Voldoende
Software ontwikkelstraat	Voldoende
Teamspirit	Goed
Sociale vaardigheden	Goed
Communicatieve vaardigheden	Goed

Initiatief	Voldoende
Presteren onder druk	Voldoende
Leiderschap	Voldoende
Commerciële vaardigheden	Voldoende
Sterke punten (bij voorkeur minstens 3)	zelfstandig, assertief, leergierig
Verbeterpunten (bij voorkeur minstens 3)	bezint eer ge begint met implementeren (niet adhoc maar gaan bouwen)
Algemeen / eventuele opmerkingen	Ben zeer benieuwd naar het onderzoek en de mobiele implementaties. Zie je dan ook graag terug over een aantal sprints...

Paulien de Wind

Naam geëvalueerde medewerker	Wilco Stuyfersant
Email geëvalueerde medewerker	wilco.stuyfersant@42.nl
Project	Goudvink
Functie medewerker	Developer
Periode inzet van	Monday, May 7, 2012
Periode inzet tot	Tuesday, June 5, 2012
Naam evaluator	Paulien de Wind
Email evaluator	paulien.de.wind@42.nl
Functie evaluator	Developer
Bedrijfsnaam	42 bv
Opdrachtomschrijving	Ontwikkelen van backend functionaliteit voor Goudvink, een aplicatie om declaraties mee te monitoren.
Technisch-inhoudelijke vaardigheden	Voldoende
Functioneel kwaliteitsbewustzijn	Voldoende

Technisch kwaliteitsbewustzijn	Goed
Productiviteit	Goed
Software ontwikkelstraat	Goed
Teamspirit	Zeer goed
Sociale vaardigheden	Zeer goed
Communicatieve vaardigheden	Zeer goed
Initiatief	Goed
Presteren onder druk	Goed
Leiderschap	Goed
Commerciële vaardigheden	NVT
Sterke punten (bij voorkeur minstens 3)	Communiceert helder en to the point. Is zorgvuldig met documentatie, JIRA, unit tests, scrum. Is rustig, genuanceerd, kan relativeren en is daarmee een rustgevende factor in teamverband.
Verbeterpunten (bij voorkeur minstens 3)	Weinig technische ervaring, maar daar ben je stagiaire voor. Na veranderingen aan de backend graag ook de frontend testen.

Bijlage D – Afstudeerplan

Informatie afstudeerder en gastbedrijf (*structuur niet wijzigen***)**

Afstudeerblok: 2012-1.2 (start uiterlijk 7 mei 2012)
Startdatum uitvoering afstudeeropdracht:
Inleverdatum afstudeerdossier volgens jaarrooster: 24 september 2012

Studentnummer: 08024197
Achternaam: dhr. Stuyfersant (*) *weghalen niet van toepassing*
Voorletters: W.
Roepnaam: Wilco
Adres: Bijvoetplan 23
Postcode: 2728DC
Woonplaats: Zoetermeer
Telefoonnummer: 0654277353
Mobiel nummer: 06542772353
Privé emailadres: wilco.s@live.nl

Opleiding: Informatica
Locatie: Zoetermeer (*) *weghalen niet van toepassing*
Variant: voltijd

Naam studieloopbaanbegeleider: Vincent Broeren
Naam begeleidend examinator: Nanny Jacobs
Naam tweede examinator: Arno Nederend

Naam bedrijf: 42 BV
Afdeling bedrijf:
Bezoekadres bedrijf: Koraalrood 33
Postcode bezoekadres: 2718SB
Postbusnummer: 6003
Postcode postbusnummer: 2702AA
Plaats: Zoetermeer
Telefoon bedrijf: 088-4242042
Telefax bedrijf: 088-4242099
Internetsite bedrijf: http://www.42.nl

Achternaam opdrachtgever: ~~mw~~ / dhr Bor (*) *weghalen niet van toepassing*
Voorletters opdrachtgever*:* R.
Titulatuur opdrachtgever:
Functie opdrachtgever:
Doorkiesnummer opdrachtgever:
Email opdrachtgever:

Achternaam bedrijfsmentor: ~~mw~~ / dhr Bor (*) *weghalen niet van toepassing*
Voorletters bedrijfsmentor: R.
Titulatuur bedrijfsmentor:
Functie bedrijfsmentor:
Doorkiesnummer bedrijfsmentor:
Email bedrijfsmentor:
NB: bedrijfsmentor mag dezelfde zijn als de opdrachtgever

Doorkiesnummer afstudeerder*:*
Functie afstudeerder (deeltijd/duaal):
Titel afstudeeropdracht: Onderzoek & verkenning appsmarkt

Opdrachtomschrijving

1. **Bedrijf**

Geschiedenis

De oprichter van 42 BV, Eric Meijer, was tot 2003 werkzaam als directeur van enkele Nederlandse bedrijfstakken van de multinational Software AG, waarvan het hoofdkantoor in Duitsland is gevestigd. Wegens een afnemende winst in het buitenland werd het Nederlandse kantoor gesloten, ondanks het feit dat deze nog wel winstgevend was. Eric Meijer besloot de ruimte in de markt die hierdoor vrij zou komen te benutten voor een eigen bedrijf.

In 2005 is 42 BV een samenwerkingsverband aangegaan met het Amsterdamse Kensas. De twee bedrijven zijn ondergebracht onder de holding genaamd M.A.D. Dit staat voor Mice and Dolphins, de twee meest intelligente diersoorten op aarde volgens the Hitchhiker's Guide To the Galaxy, het boek waar eveneens de naam 42 van afkomstig is. Tegenwoordig wordt de naam Kensas zo minimaal mogelijk gebruikt, de naam 42 is hiervoor in de plaats gekomen.

42 BV is inmiddels uitgegroeid tot een bedrijf met 40 medewerkers. Enkelen hiervan werken op kantoor in Zoetermeer of Amsterdam, maar de meeste werken intern bij de klant.

Bedrijfsprofiel

De missie van 42 BV is het ontwikkelen van hoogwaardige software waarmee haar klanten hun werkprocessen en bedrijfsresultaten kunnen verbeteren. 42 BV ontwikkelt maatwerk Java-enterprise producten, en doet dit het liefst met open source. Verder helpt 42 BV grote klanten bij het integreren van
Atlassian-producten. 42 BV gebruikt deze software zelf ook en is partner van Atlassian.

De visie van 42 BV om de missie te verwezenlijken is 42 BV een bedrijf zijn dat:

- Software, advies en ondersteunende diensten levert waarbij bruikbaarheid en gebruiksvriendelijkheid centraal staan.
- De medewerkers een boeiende, inspirerende en inhoudelijk interessante werkomgeving biedt waar complexe software met behulp van state of the art gereedschappen wordt ontwikkeld.
- Een klantenkring bedient die de geboden diensten in hoge mate waardeert
- Door beheerste groei continuïteit biedt aan klanten en medewerkers.

Klantprofiel

42 BV heeft na de oprichting enkele grote klanten van Software AG kunnen behouden, waaronder Schiphol en CenterParcs. Hierdoor kon 42 BV meteen beginnen met grotere opdrachten, iets wat lastig kan zijn voor beginnende bedrijven. Enkele andere (recente) grote klanten van 42 BV zijn Albert Heijn, de Consumentenbond, de Belastingdienst, ANWB, Essent en Nidera.

2. Probleemstelling

Er zijn twee probleem factoren die tot deze afstudeeropdracht leiden waarvan de eerste de hoofdzaak is.

- Vanuit de markt is er steeds meer vraag naar mobiele applicaties.
- Momenteel is er geen declaratiesysteem.

Steeds meer vraag naar mobiele applicaties:

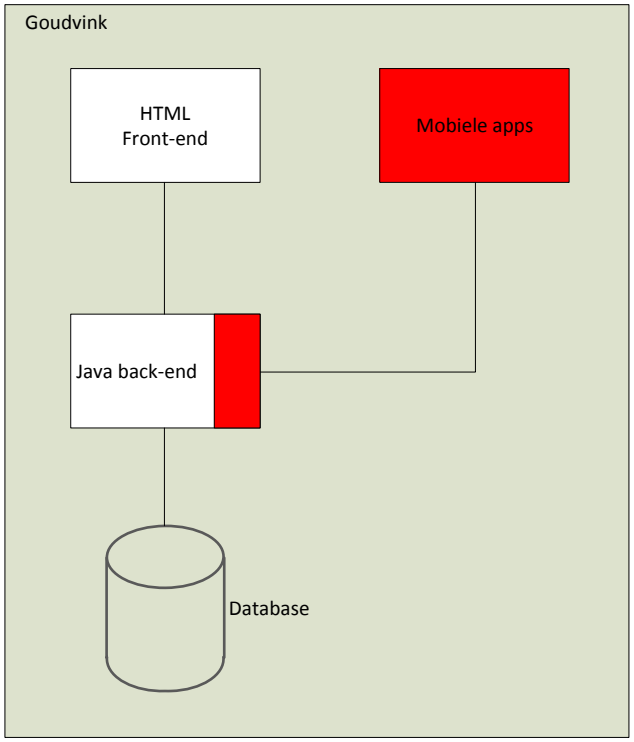
Binnen de ICT wereld vinden continu nieuwe ontwikkelingen plaats. Eén van de sterkst groeiende op het moment is die van mobiliteit. Met mobiliteit wordt in dit geval bedoeld plaats- en tijdsonafhankelijk werken. Dit architectuur principe is door de komst van smartphones en tablets sinds een aantal jaar sterk in opkomst, waardoor uitsluitend werken achter de computer tot het verleden behoord. De klanten van 42 BV zien een sterke vooruitgang in deze ontwikkelingen en daardoor rijst steeds vaker de vraag in hoeverre zij hun gebruikte applicaties kunnen integreren in een mobiele omgeving. Binnen 42 BV is echter geen specialistische kennis over dit onderwerp aanwezig waardoor mobiele oplossingen vaak door middel van een browser-applicatie worden aangeboden aan de klant. Om in de toekomst beter mee te kunnen in de zogenoemde *'appsmarkt'* wilt 42 BV zich nu gaan richten op de mobiele ontwikkelingen en de obstakels die daarbij komen kijken in kaart brengen. Hierbij moet ook gedacht worden aan de kwaliteit waar 42 BV voor staat. Alle applicaties die worden gemaakt moeten grondig worden getest voordat zij in gebruik worden genomen. 42 BV vraagt zich af in hoeverre er testmogelijkheden zijn tijdens de ontwikkeling van apps.

Momenteel is er geen declaratiesysteem:

Binnen 42 BV wordt gebruik gemaakt van diverse interne applicaties doe zijn onderverdeeld in de "Volière". Zij hebben allemaal een vogelnaam, als voorbeeld MUS, het uren registratie systeem. Wat 42 BV momenteel nog niet heeft is een declaratiesysteem waarin werknemers kosten die zij hebben gemaakt voor bedrijfsdoeleinden kunnen declareren. Dit is wel gewenst en in combinatie met bovenstaande probleemstelling wil 42 BV de ontwikkeling van dit systeem benutten om de appsmarkt te verkennen. Het systeem zal dus als hulpmiddel dienen om de appsmarkt te betreden en zal de naam "Goudvink" krijgen.

3. Doelstelling van de afstudeeropdracht

Het doel van de afstudeeropdracht is om 42 BV meer inzicht te geven over de mogelijkheden binnen de sterk opkomende appsmarkt. Het hulpmiddel daarbij is het nieuwe declaratiesysteem Goudvink. Voor de ontwikkeling van dit systeem zullen twee medewerkers van 42 BV en ikzelf worden aangesteld. Beide medewerkers zullen beperkt beschikbaar zijn voor de ontwikkeling, waardoor binnen het project genoeg ruimte over blijft waarin ik als afstudeerder een rol kan spelen. Er zal een duidelijke scheiding aanwezig zijn, wat te zien is in onderstaand diagram.



Waar de medewerkers zich zullen richten op de HTML front-end omgeving, zal ik mij richten op de mobiele omgeving van het systeem (het rode gedeelte in het diagram). Omdat beide delen het centrale Java backend systeem zullen gebruiken is op dit gebied een goede samenwerking noodzakelijk. Over de verdere invulling hiervan is nog niet veel bekend, maar hier zal nog steeds een duidelijke scheiding aanwezig zijn. Ik zal mij alleen richten op de interfaces die nodig zijn om de requests van de mobiele app af te handelen.

De focus van deze afstudeerstage ligt niet op het ontwikkelen maar op 42 BV inzicht geven in de mogelijkheden binnen de appsmarkt. Aansluitend op bovengenoemd systeem wordt er door mij een onderzoek verricht naar de verschillende mogelijkheden om voor mobiele applicaties te ontwikkelen. Binnen de appsmarkt is een grote diversiteit aan apparaten en besturingssystemen aanwezig waar de apps voor gemaakt kunnen worden. Dit maakt het moeilijk om een app universeel werkend te krijgen op alle denkbare apparaten. Er zijn twee grote besturingssystemen te herkennen, namelijk Google's Android en Apple's iOS. De afstudeeropdracht zal zich op deze twee besturingssystemen richten. Beide besturingssystemen verschillen ten opzichte van elkaar waardoor een applicatie gemaakt voor iOS niet zal werken op Google's Android en vice-versa.

Om een applicatie toch geschikt te maken voor mobiel gebruik op meerdere apparaten zijn er diverse mogelijkheden.

- **Volledig browser-based**
  Er kan worden gekozen om de gehele applicatie browser gebaseerd te maken, waardoor voor de front-end gebruik wordt gemaakt van bijvoorbeeld HTML en Javascript. Browser-based applicaties zijn universeel over vrijwel alle platformen (hoge portabiliteit) en is dus een makkelijke oplossing.

- **Native**
  De applicatie kan in een native programmeertaal worden geschreven wat inhoud dat hij specifiek voor Android of iOS is. Dit betekend als men wilt dat de applicatie op beide besturingssystemen werkt hij twee keer 'opnieuw' gemaakt moet worden. Eén maal in Java (voor Android) en eenmaal in Objective C (voor iOS).

- **Semi-native**
  Wanneer een applicatie semi-native wordt gemaakt wordt voor meerdere besturingssysteem een applicatie gemaakt welke de browserfunctionaliteiten van het besturingssysteem aanroepen. Het voordeel hiervan is dat er maar een klein gedeelte van de code in een native taal geschreven hoeft te worden. De rest van de functionaliteit zal namelijk door middel van een universele browsertaal worden geschreven en zal dus centraal aan te passen zijn.

- **Tussentaal**
  Er zijn mogelijkheden tot gebruik van een 'tussentaal' welke bij het compileren ervoor zorgt dat de applicatie in beide talen beschikbaar is. De mogelijkheden hiervan zijn echter nog niet duidelijk.

Uit een onderzoek moet blijken welk van deze technieken interessant zijn voor 42 BV. Het onderzoek zal dan ook resulteren in een selectie van tools en mogelijkheden waarmee ik tijdens de rest van het project aan de slag zal gaan. Er zullen voor dezelfde app meerdere prototypes worden gemaakt, elke keer door gebruik te maken van een andere tool. Door middel van documentatie over mijn uitgevoerde processen tijdens het prototypen zal het voor 42 BV een stuk duidelijker worden wat de voor en nadelen zijn van de diverse technieken en tools.

Het doel zal bereikt zijn wanneer meerdere applicaties beschikbaar zijn die medewerkers van 42 BV in staat stellen declaraties in te voeren en wanneer 42 BV een duidelijk beeld heeft van de voor en nadelen van de verschillende technieken. Dit laatste zal worden ondersteund door een presentatie en een advies. Om de visie op het te realiseren werk gelijk te houden zal ik regelmatig contact hebben met de verschillende stakeholders.

4. Resultaat

Wanneer de afstudeeropdracht is afgerond zal 42 BV over een aantal (werkende) prototypes beschikken van de mobiele app van Goudvink. Er zal een presentatie over de diverse mogelijkheden worden gegeven waarna 42 BV een verantwoorde keuze kan maken over welke techniek zij voortaan zullen toepassen bij het ontwikkelen van mobiele apps. Dit zal tevens worden ondersteund door proces documentatie met daarin mijn persoonlijke oordeel en advies.

5. Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten

Er zullen tijdens deze afstudeerstage diverse werkzaamheden worden uitgevoerd. Deze werkzaamheden zullen worden uitgevoerd door middel van een Scrum-aanpak, welke de vorm zal aannemen van meerdere sprints van elk 2 weken. Tijdens iedere sprint wordt een backlog bijgehouden van openstaande taken, hieruit worden een aantal taken voor de sprint gepland. Iedere sprint wordt afgesloten met een demo, waarmee aangetoond wordt wat er gedaan is. Er zijn diverse redenen waarom er wordt gekozen voor scrum, te vinden onder de paragraaf “Methodiek”.

De werkzaamheden zullen plaatsvinden in de onderstaande volgorde:

Plan van aanpak schrijven (1 week)

De eerste stap van de afstudeerstage zal het schrijven van een Plan van Aanpak (PVA) betreffen waarin ik duidelijk aangeef hoe ik denk dat het project zal gaan verlopen. Hierin zal onder andere een globale planning worden gedefinieerd.

Ontwikkelen Java backend (4 weken, met overloop in volgende fase)

Het ontwikkelen van de Java backend vereist zoals eerder vermeld een samenwerking met de twee medewerkers van 42 BV welke ook op dit project worden gezet. Omdat zij vroeg beginnen met de ontwikkeling van dit systeem zal dit gedeelte van de opdracht als eerste worden gedaan. Ik zal mij daarbij met name richten op de code die communicatie tussen de database en mobiele applicatie mogelijk maakt. De backend zal geprogrammeerd worden middels Test Driven Development om te voldoen aan de kwaliteitseisen van 42 BV op het gebied van software testing.

Oriëntatie op probleemdomein (4 weken, met aan het begin overloop uit de vorige fase)

De oriëntatie op het probleemdomein zal bestaan uit een onderzoek naar mobiele applicaties en de manier waarop zij worden gemaakt. Hierbij refereer ik terug naar punt 3, doelstelling van de opdracht, waarin ik het verschil beschrijf tussen native applicatie programmeren en browser-based programmeren. Er zullen diverse technieken en tools worden onderzocht waarna er een selectie zal worden gemaakt. Daarnaast zijn er requirements aan het declaratiesysteem welke nog in kaart gebracht dienen te worden.

Ontwerpen (1 week)

Tijdens de opleiding Informatica leren wij hoe een systeem vooraf wordt ontworpen door middel van UML diagrammen. Ik zal tijdens deze afstudeerstage de architectuur van de mobiele apps ontwerpen en de kennis die ik tijdens mijn opleiding heb opgedaan toepassen. Tevens zal hierbij worden gekeken naar de communicatie die tussen de Java backend en de mobiele apps plaatsvind. Deze ontwerpen, in combinatie met de vooraf gedefinieerde requirements, zullen de prototype fase vergemakkelijken doordat er vooraf goed wordt nagedacht over de functionaliteiten en structuur van de applicatie.

Ontwikkelen prototypen mobiele apps (5 weken)

Tijdens deze activiteit worden de resultaten die voortkomen uit de selectie van het onderzoek gebruikt om prototypes van de mobiele apps van Goudvink te ontwikkelen. Er worden dus meerdere applicaties gemaakt welke met de backend Java kunnen communiceren. Hierbij kan gedacht worden aan het doorgeven van data die is ingevoerd in de app. Hierbij zal gebruik worden gemaakt van de tijdens de oriëntatie en ontwerpfase opgestelde requirements en ontwerpen. Mochten er nog wijzigingen plaatsvinden dan zullen deze worden doorgevoerd in de opgestelde ontwerpen.

Documenteren (gehele periode)

Gedurende de hele afstudeerperiode zal er over mijn werkzaamheden worden gedocumenteerd. Hierbij komen niet alleen mijn activiteiten aan bod, maar ook tegen welke problemen ik aanloop tijdens het ontwikkelen van de mobiele apps. Aan de hand van deze informatie kan 42 BV op een later tijdstip besluiten of zij verder willen in de appsmarkt en welke techniek zij daarbij het beste kunnen gebruiken. Dit zal resulteren in een document met per techniek de voor en nadelen en mijn persoonlijke oordeel en advies.

Tevens zal een deel van de documentatie resulteren in het afstudeerverslag welke aan het eind van de afstudeerperiode opgeleverd dient te worden.

Schrijven afstudeerverslag (2 weken)

Zoals geschreven zal er gedurende de hele periode worden gedocumenteerd over de verrichte werkzaamheden. Om de afronding van het afstudeerverslag te realiseren zal hiervoor aan het eind van de afstudeerperiode tijd worden gereserveerd.

5. Methodiek

Tijdens de afstudeeropdracht zal ik gebruik maken van de Scrum ontwikkelmethodiek. Hier zijn enkele redenen voor te benoemen:

1. **De mogelijkheid om te wijzigen in het projectontwerp.**

Scrum biedt de mogelijkheid om wijzigingen die tijdens het project worden gemaakt aan het ontwerp mee te nemen in de planningen van het project. Het komt in de praktijk regelmatig voor dat bepaalde zaken die in eerste instantie heel logisch leken in een later stadium veranderen. Een voordeel van Scrum is dat deze wijzigingen gemakkelijk meegenomen kunnen worden, vanwege de korte sprints en frequente communicatie tussen de verschillende partijen. De ontwerpfase ligt daardoor wat minder vast dan bij andere ontwikkelmethodieken.

2. **Scrum is de bedrijfsstandaard en de medewerkers zijn hieraan gewend.**

42 BV gebruikt sinds de start van het bedrijf Scrum als ontwikkelmethodiek voor haar projecten. Dit betekent dat de processen die hierbij horen welbekend zijn bij de medewerkers van het bedrijf. Gecombineerd met het feit dat ik eveneens ervaring heb met de ontwikkelmethodiek zal dit de communicatie tussen mij en de rest van de betrokkenen bij dit project vergemakkelijken.

3. **Interne applicaties zijn gebaseerd op de Scrum methodiek.**

42 BV maakt gebruik van diverse applicaties van Atlassian die helpen bij het managen van een project en het bijhouden van de voortgang hiervan. Deze tools zijn gebaseerd op het gebruik van Scrum. 42 BV verlangt dat deze tools gebruikt worden, wat het gebruik van Scrum gemakkelijker laat aansluiten dan de andere ontwikkelmethodieken.

6. **Op te leveren (tussen)producten**

Tijdens de afstudeerperiode zullen een aantal producten worden opgeleverd. Deze producten zijn als volgt:

- Plan van aanpak
- Een vooronderzoek in de vorm van een rapport met de voor een nadelen van de diverse technieken voor het ontwikkelen van mobiele applicaties.
- Een overzicht van de requirements die zijn gesteld aan de mobiele applicaties van Goudvink.
- Ontwerp diagrammen
- Een deel van het Java backend systeem welke communicatie mogelijk maakt tussen de mobiele front-end en database.
- Een aantal prototypen van de mobiele app van Goudvink.
- Definitief rapport over de onderzochte technieken met daarin mijn oordeel en advies.

7. **Te demonstreren competenties en wijze waarop**

- 1.4 Uitvoeren analyse door definitie van requirements

  Aan het project Goudvink zitten requirements verbonden welke in kaart gebracht dienen te worden.

- 3.1 Ontwerpen softwarearchitectuur

  Voordat ik met het daadwerkelijke programmeren van de mobiele applicaties begin zal ik aan de hand van de requirements een softwarearchitectuur ontwerpen. Dit zal ik door middel van de UML methodiek doen.

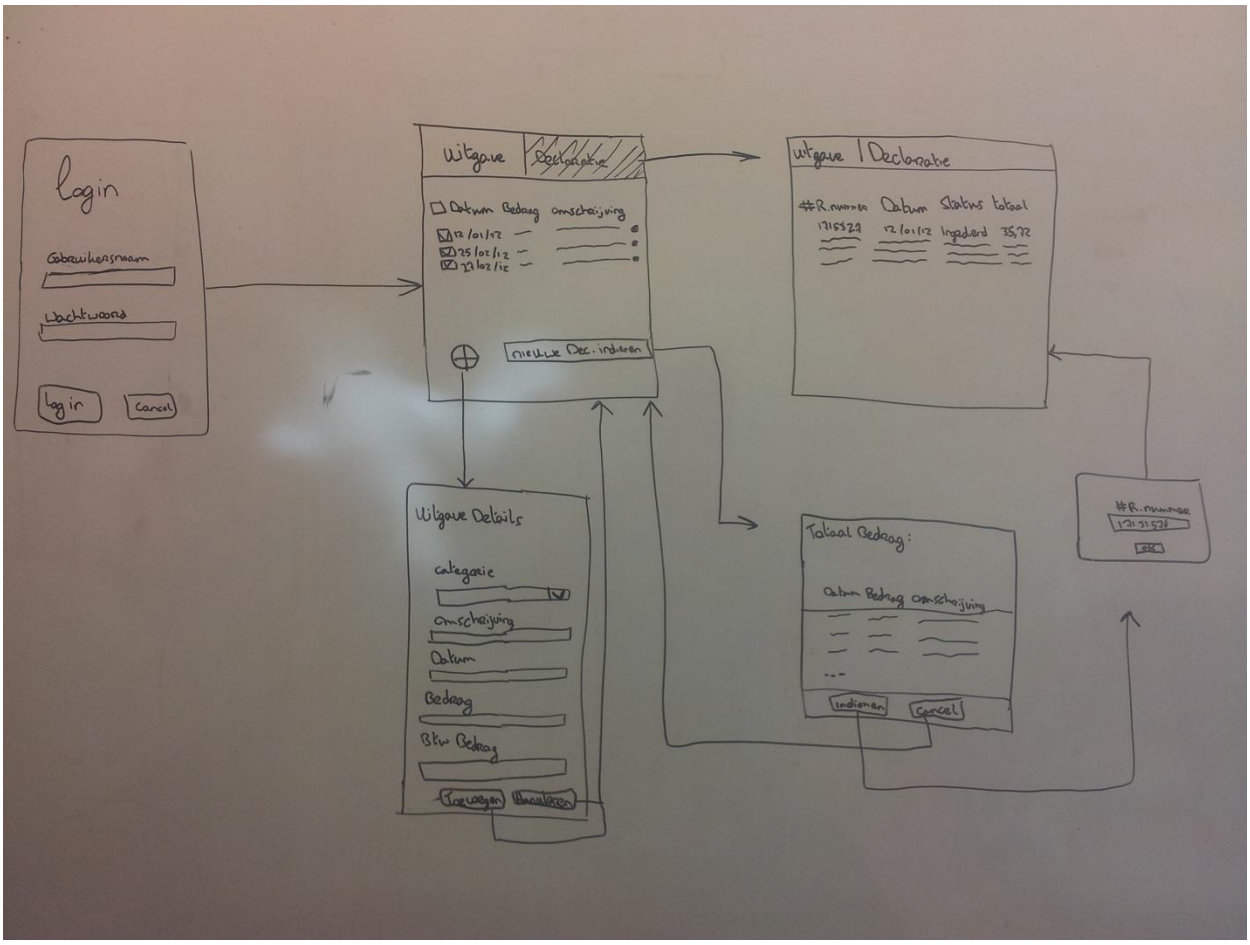
- 3.3 Bouwen applicatie

  Na de ontwerpfase zal de ontworpen systeem architectuur gebouwd worden. De applicatie zal bestaan uit een mobiele applicatie en een gedeelte van de backend software welke voor de communicatie zal zorgen.

- 3.5 Uitvoeren van en rapporteren over het testproces

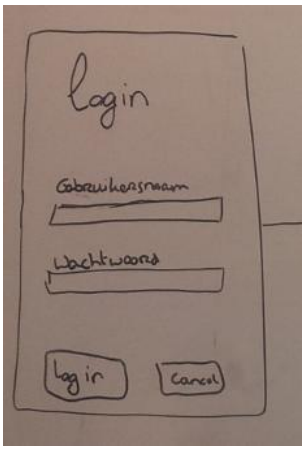
  De code die geschreven wordt moet ook worden getest. Hiervoor zullen unittests geschreven worden, mogelijk nog aangevuld met device specifieke testen (welke nog nader worden onderzocht). De Javatestcode wordt met behulp van de build automation tool Maven in kaart gebracht. Dit levert een duidelijk overzicht van de mate waarop de code is getest. De resultaten van alle testcases worden gedocumenteerd in een rapport.

Bijlage E – Goudvink app requirements



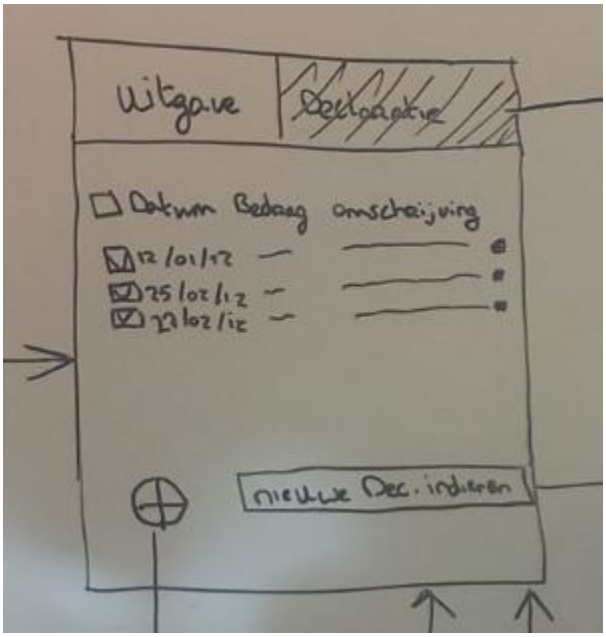
Login

Door middel van het login scherm van de Goudvink app kan de gebruiker zich middels een gebruikersnaam en wachtwoord authentiseren met de Goudvink server. Hierbij moet gelet worden op het feit dat Goudvink gebruik maakt van Crowd-authenticatie. Crowd is een Atlassian product die het single sign-on principe mogelijk maakt. Het login scherm komt er als volgt uit te zien:



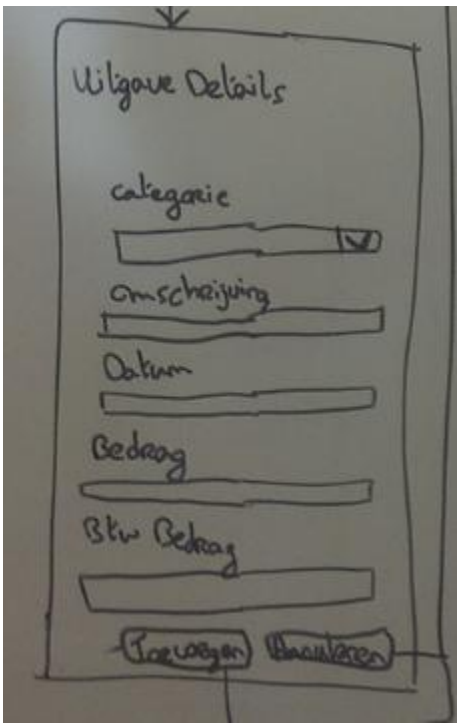
Uitgaven overzicht

Het uitgaven overzicht wordt het hoofdscherm van de Goudvink app. Middels het uitgaven scherm kan de gebruiker zien welke uitgaven hij of zij nog open heeft staan. Het bovenste deel van het scherm betreft een tabblad, waar gekozen kan worden voor *uitgave* of *declaratie*. Middels de declaratie tab kan de gebruiker naar het *declaratie overzicht* scherm. Onder de tabbladen is een tabel te vinden met de informatie over de diverse uitgaven. Naast de tabelheaders is een *selecteer alles knop* te vinden, waarmee de alle uitgaven geselecteerd of gedeselecteerd worden. Aan de linker kant van elke tabel rij is een *selectie box* te vinden waarmee de gebruiker uitgaven kan selecteren. Aan het eind van iedere rij in de uitgaven tabel is een *delete* knop te vinden, waarmee de gebruiker een uitgave kan verwijderen. Wanneer een gebruiker een nieuwe declaratie in wil dienen selecteert hij of zij de gewenste uitgaven en klikt op de knop *nieuwe declaratie indienen*. Standaard staan alle uitgaven geselecteerd. Deze actie zal vervolgens het *declaratie bevestigings scherm* aanroepen. Naast de *nieuwe declaratie indienen* knop is een *uitgave toevoegen* knop te vinden. Deze knop zal verwijzen naar het *uitgave toevoegen* scherm. Wanneer de gebruiker op één van de uitgaven in de tabel klikt, wordt de *uitgave aanpassen* actie aangeroepen, waarvan het scherm hetzelfde eruit zal zien als *uitgave toevoegen*. Het scherm ziet er als volgt uit:



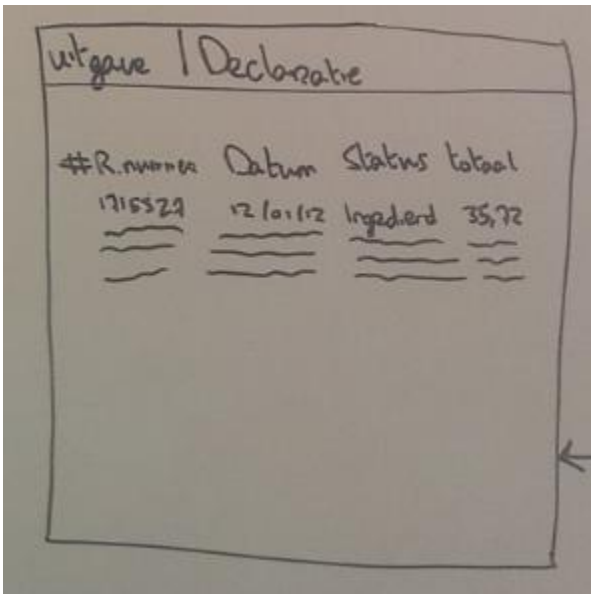
Uitgave toevoegen

Voor het toevoegen van een uitgave wordt een leeg *uitgave details* scherm aangeroepen. Dit scherm wordt teven gebruikt om een bestaande uitgave aan te passen. Nadat de gebruiker de vereiste gegevens heeft ingevuld klikt hij op de knop *toevoegen*, waarna het scherm vervolgens weer het *uitgave overzicht* scherm laat zien. Indien de gebruiker de actie wilt stoppen klikt hij op annuleren, waarna het *uitgave overzicht* scherm wordt getoond.



Declaratie overzicht

Net als het *uitgaven overzicht*, heeft ook het declaratie overzicht een tabblad aan de bovenkant van het scherm. Hierin kan zowel gekozen worden voor het *uitgave overzicht* als het *declaratie overzicht*, waarvan de laatst genoemde op dat moment actief is. Onder de tabs wordt een tabel getoond met daarin de declaraties met de status *ingediend* de ingelogde gebruiker. Naast de declaraties met de status *ingediend*, worden ook de laatste twee declaraties met de status *compleet* getoond. Dit wordt gedaan zodat er voor de gebruiker geen verwarring ontstaat in de vorm van verdwijnende declaraties, wanneer deze door de controleur in de tussentijd zijn geaccepteerd. Vanaf dit scherm zijn geen verdere acties mogelijk.

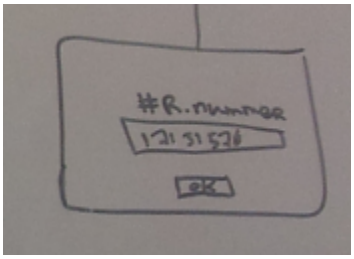


Declaratie bevestigingsscherm

Wanneer de gebruiker ervoor kiest in het *uitgaven overzicht* scherm een declaratie toe te voegen, krijgt hij of zij een bevestigingsscherm te zien. In dit bevestigingsscherm ziet de gebruiker het *totaal bedrag* dat hij of zij wilt declareren, waarna de gekozen uitgaven worden getoond middels een tabel. De hoeveelheid uitgaven die worden getoond hangt af van de grootte van het scherm. Indien de gebruiker akkoord gaat met hetgeen het scherm toont, kiest hij of zij voor de *indienen* knop. Dit resulteert in een actie die de declaractie definitief indient. Vervolgens wordt aan de gebruiker het *verkrijgen referentienummer* scherm getoond. Indien de gebruiker besluit de actie te stoppen kiest hij voor de *cancel* knop, waarna hij wordt teruggestuurd naar het *uitgaven overzicht*.

Verkrijgen referentienummer

Wanneer een gebruiker een declaratie heeft ingediend, krijgt hij of zij middels dit scherm het referentienummer van desbetreffende declaratie te zien. De enig mogelijke actie in dit scherm is de *ok* knop, die vervolgens het *declaratie overzicht* scherm laat zien.



Overzicht

Scherm	*Functie*	*Prioriteit*
Login	Authentiseren door middel van gebruikersnaam en wachtwoord.	Laag
	Mogelijkheid tot afsluiten app.	Laag
Uitgave overzicht	Lijst van uitgaven middels een tabel, waarbinnen de velden Datum, Bedrag en Omschrijving van de uitgave worden weergegeven.	Hoog
	Een mogelijkheid tot switchen tussen het *uitgaven* en *declaratie overzicht*.	Gemiddeld
	Selectie optie voor ieder item in de uitgaventabel, met standaard een geselecteerde waarde.	Gemiddeld
	Per item in de uitgaventabel een functionaliteit om de uitgave aan te passen.	Gemiddeld
	Verwijder knop voor ieder item in de uitgaventabel.	Hoog
	Functionaliteit om een nieuwe declaratie in te dienen, waarna wordt doorverwezen naar het *declaratie bevestigingsscherm*.	Gemiddeld
	Functionaliteit om een nieuwe uitgave toe te voegen, waarbij gebruik wordt gemaakt van het *uitgave toevoegen* scherm.	Hoog
Uitgave toevoegen	Een scherm wat zowel voor bestaande als nieuwe uitgaven gebruikt kan worden. Afhankelijk van de functie die dit scherm aanroept zijn de velden leeg of gevuld.	Hoog
	In het scherm zijn vijf velden gedefinieerd, categorie, omschrijving, datum, bedrag en btw bedrag. Voor categorie wordt gebruik gemaakt van een combobox. Het datum veld zal in de eerste versie van de Goudvink app handmatig ingevuld worden. Zowel de bedrag als btw bedrag velden mogen alleen getallen en een komma accepteren.	Hoog
	Functionaliteit die door middel van een toevoegen knop kan worden aangeroepen. Deze functionaliteit stuurt het nieuwe of te update uitgave object aan de back-end server voor verdere verwerking.	Hoog
	De gebruiker heeft de mogelijkheid de actie te annuleren middels een annuleren knop, resulterend in het *uitgave overzicht* scherm.	Hoog
Declaratie overzicht	Lijst van declaraties van de gebruiker middels een tabel. Alle declaraties met de status ingediend worden getoond, evenals de twee laatste declaraties met de status compleet.	Gemiddeld
	Een mogelijkheid tot switchen tussen het *uitgaven* en *declaratie overzicht*.	Gemiddeld
Declaratie bevestingsscherm	Een totaal bedrag van alle uitgaven bij elkaar weer.	Gemiddeld
	Een tabel met daarin een overzicht van de te declareren uitgaven, waarbij de hoeveelheid regels afhankelijk is van de grootte van het scherm.	Laag
	Functionaliteit om de declaratie definitief in te dienen, resulterend in het scherm *referentienummer verkrijgen*.	Gemiddeld
	Functionaliteit om het proces te beëindigen zonder de declaratie in te dienen, resulterend in het *uitgaven overzicht* scherm.	Gemiddeld
Verkrijgen referentienummer	Toont het referentienummer van de zojuist aangemaakte declaratie.	Gemiddeld
	Functionaliteit om het scherm te bevestigen, resulterend in het scherm *declaratie overzicht*.	Gemiddeld

Bijlage F – Onderzoeksrapport

Samenvatting

Binnen de ICT wereld vinden continu nieuwe ontwikkelingen plaats. De klanten van 42 BV zien een snelle ontwikkeling op het gebied van mobiel werken, waardoor zij het bedrijf steeds vaker de vraag stellen in hoeverre gemaakte applicaties tevens gebruikt kunnen worden in een mobiele omgeving. Binnen 42 BV is geen specialistische kennis over het ontwikkelen van mobiele applicaties aanwezig, met het gevolg dat mobiele oplossingen vaak door middel van een web-applicatie worden aangeboden aan de klant. Om in de toekomst beter mee te kunnen draaien in de zogenoemde appsmarkt, heeft 42 BV een onderzoek laten uitvoeren naar de diverse mogelijkheden tot het ontwikkelen van app types. Dit onderzoeksrapport is voortgekomen uit het uitgevoerde onderzoek en bevat alle onderzochte informatie op het gebied van de app types met de bijbehorende ontwikkelstraat, de diverse voor en nadelen van de app types en de reflectie die is gedaan op de huidige ontwikkelsituatie van 42 BV.

Inhoud

1 Inleiding

Het leven in een tijdperk waarbij in een sneltrein vaart veel ontwikkelingen gaande zijn op het gebied van diverse technologieën betekend voor veel bedrijven dat zij continu moeten innoveren om met de markt mee te kunnen. Het feit dat de veranderingen vaak relatief snel op elkaar volgen, maakt het voor bedrijven vaak lastig een juiste keuze van richting te maken tussen de vele, vaak onduidelijke opties, die het bedrijf heeft. 42 BV, het bedrijf waar dit onderzoek voor wordt uitgevoerd, ondervind soortgelijke problemen. Het technologiegebied waarbinnen zij tegen dit probleem aanlopen is die van het ontwikkelen van applicaties, die bedoeld zijn voor mobiel gebruik. Het bedrijf vindt het moeilijk een manier van ontwikkelen te kiezen, omdat de opties die het bedrijf heeft, met de bijbehorende voor en nadelen per optie, niet bekend zijn. 42 BV heeft daartoe besloten door middel van een onderzoek te achterhalen wat voor het bedrijf de beste manier van ontwikkeling is, wanneer zij mobiele applicaties ontwikkelen.

1.1 Aanleiding

Binnen de ICT wereld vinden continu nieuwe ontwikkelingen plaats. Eén van de sterkst groeiende op het moment is die van mobiel werken, waarmee in dit geval wordt bedoeld plaats- en tijdsonafhankelijk werken. Deze ontwikkeling is door de komst van smartphones en tablets sinds een aantal jaar sterk in opkomst, waardoor voor veel mensen uitsluitend werken achter de computer tot het verleden behoort. De klanten van 42 BV zien de sterke vooruitgang in deze ontwikkelingen en daardoor rijst steeds vaker de vraag in hoeverre zij de ontwikkelde applicaties kunnen gebruiken in een mobiele omgeving. Binnen 42 BV is geen specialistische kennis over het ontwikkelen van mobiele applicaties aanwezig, met het gevolg dat mobiele oplossingen vaak door middel van een web-applicatie worden aangeboden aan de klant. Om in de toekomst beter mee te draaien in de zogenoemde appsmarkt, wil 42 BV zich nu gaan richten op de mobiele ontwikkelingen en de obstakels die daarbij komen kijken in kaart brengen. Zo kunnen zij de klant beter van dienst zijn en in hun behoeften voorzien.

Daarbij heeft 42 BV vernomen dat er binnen de appsmarkt verschillende app types bestaan, met elk hun eigen ontwikkelsituatie. Voordat 42 BV aan het ontwikkelen van mobiele apps voor haar klanten begint, wil het bedrijf weten in hoeverre de ontwikkeling van een app aansluit op de ontwikkelstraat zoals deze nu wordt toegepast binnen het bedrijf. Hierbij is het bedrijf specifiek op zoek naar welk van de app types het meeste aansluit op de huidige ontwikkelstraat. Deze ontwikkelstraat is nog niet eerder in kaart gebracht, waardoor dit ook tot het onderzoek behoort.

1.2 Doelstelling

42 BV wil weten wat er bij ontwikkeling voor mobiele applicaties komt kijken. Vooraf is al bekend dat er verschillende manieren bestaan om universele applicaties te ontwikkelen, die gebruikt kunnen worden op meerdere mobiele besturingssystemen. Van deze manieren moeten de voor en nadelen worden onderzocht, waarbij expliciet wordt gekeken naar de gelijkenissen met de huidige ontwikkelsituatie.

1.3 Hoofd en deelvragen

Bij een onderzoek is het gebruikelijk een centrale hoofdvraag te definiëren, die vervolgens wordt beantwoord door middel van het onderzoek. Daarnaast wordt vaak gebruik gemaakt van deelvragen, die samen de hoofdvraag beantwoorden. Deze onderverdeling van de hoofdvraag wordt voornamelijk gedaan om het onderzoek duidelijker in kaart te brengen. Het dwingt de onderzoeker daarbij ook om vooraf na te denken over welke informatie nodig is om de hoofdvraag te kunnen beantwoorden.

Op basis van de vraag vanuit 42 BV, is de hoofdvraag van dit onderzoek als volgt geformuleerd:

> *Welke mobiele applicatie ontwikkelmogelijkheden, resulterend in app types, sluiten aan op de werkprocessen en vaardigheden van 42 BV en zijn daarbij commercieel gezien verzilverbaar om in de praktijk toe te passen?*

Om de centrale hoofdvraag te beantwoorden zijn een aantal deelvragen opgesteld. Deze zijn vervolgens onderverdeeld in onderzoeksvragen.

Deelvraag 1 Welke app types bestaan er en hoe ziet de ontwikkeling van deze app types eruit?

- Welke app types bestaan er?
- Hoe ziet de ontwikkelstraat per app type eruit?
- Tegen welke overige problemen lopen ontwikkelaars aan (user experiences)?

Deelvraag 2 Hoe ziet de huidige ontwikkelstraat van 42 BV eruit en welke beoordeling kan, in vergelijking daarmee, per app type worden gegeven?

- Hoe ziet de huidige ontwikkelstraat van 42 BV eruit?
- In welke mate zijn aspecten en gebruikte technologieën uit de huidige ontwikkelstraat terug te vinden bij de diverse app types?

Deelvraag 3 Welke vraag naar applicaties, specifiek voor bepaalde platformen, is er vanuit de apps markt?

- Naar welke platformen is er vanuit de klanten van 42 BV veel vraag?
- Wat zegt het marktaandeel van besturingssystemen binnen de mobiele markt?

Middels de eerste deelvraag wordt in kaart gebracht welke app types bestaan en hoe de ontwikkeling van deze app types er uit ziet. Hierbij worden per app type een aantal aspecten nader toegelicht die voor 42 BV van belang zijn. De resultaten zullen voor een groot deel bepalen welke app types interessant zijn voor 42 BV om verder te onderzoeken.

Middels de tweede deelvraag wordt de huidige ontwikkelstraat bij 42 BV vergeleken met die van de verschillende app types. Dit wordt gedaan aan de hand van het beoordelen van diverse, voor 42 BV belangrijke, aspecten. De overeenkomsten tussen de ontwikkelstraat van bepaalde app types en de huidige ontwikkelstraat van 42 BV kunnen een definitief bepalende factor voor het onderzoek zijn. De verwachting is dat deze deelvraag deels samen valt met de eerste deelvraag.

Middels de derde deelvraag wordt in kaart gebracht welke vraag er vanuit de klant is. Daarbij wordt het huidige marktaandeel van mobiele besturingssystemen onderzocht, wat 42 BV een nauwkeurig beeld geeft over welke besturingssystemen op het moment het populairst zijn.

Gebruikte begrippen in de hoofd- en deelvragen kunnen als volgt worden gedefinieerd:

App types
Binnen de apps ontwikkeling zijn verschillende mogelijkheden tot ontwikkelen te definiëren. Deze mogelijkheden resulteren in verschillende app types. 42 BV onderscheidt er vier: Volledig browser-based (resulteert in een Web app), Native (resulteert in een Native app), Semi-native (resulteert in een Semi-Native app) en tussentaal. De definitie van deze app types worden in dit rapport besproken.

Appsmarkt
De markt waarin de klant steeds vaker naar mobiele applicaties (apps) vraagt.

Ontwikkelstraat
De manier waarop applicaties worden ontwikkeld. Hierbij kan gedacht worden aan gebruikte methodieken, programmeertaal en programma's.

1.4 Leeswijzer

Dit onderzoeksrapport volgt het traject dat is gevolgd gedurende het onderzoek. Het is vanaf het begin van het onderzoek gestart en wordt aangevuld naarmate het onderzoek verder gaat. Het document bevat hierdoor zowel het onderzoeksplan als het onderzoek zelf. In het kort is het document opgebouwd aan de hand van volgende hoofdstukken:

- ***Hoofdstuk 1 – Inleiding***
  Dit deel bestaat uit het schetsen van de situatie waaruit dit onderzoek voorkomt. Dit is gedaan aan de hand van de aanleiding en doelstelling. Vervolgens wordt het onderzoek gedefinieerd aan de hand van een hoofdvraag en daarbij behorende deelvragen.

- ***Hoofdstuk 2 – Onderzoeksopzet***
  Het tweede deel van het document richt zich op de aanpak van het onderzoek. Hoe te werk is gegaan en hoe de informatie wordt verkregen die nodig is om een conclusie te onderbouwen, komen daarbij aan bod.

- ***Hoofdstuk 3 - Achtergrondinformatie***
  In dit hoofdstuk wordt achtergrondinformatie gegeven, met betrekking tot het domein waarin het onderzoek plaatsvindt.

- ***Hoofdstuk 4 - Het onderzoek***
  In het vierde deel komen de bevindingen van het onderzoek aan bod. Hierbij worden de diverse deelvragen die zijn opgesteld beantwoord en toegelicht.

- ***Hoofdstuk 5 – Conclusie***
  Het gehele document wordt afgesloten met een conclusie die aan 42 BV een advies geeft.

- ***Bronvermelding***
  Tijdens het onderzoek is veelvoudig gebruik gemaakt van bronnen. In het document worden bronnen aangegeven aan de hand van een getal aan het einde van de zin. Dit getal refereert naar bijbehorende voetnoot, welke de naam van de bron weergeeft. Deze voetnoot kan vervolgens in de bronvermelding op alfabetische volgorde worden teruggevonden.

2 Onderzoeksopzet

Om de hoofdvraag van dit onderzoek te beantwoorden is deze opgedeeld in deelvragen. Na beantwoording van deze deelvragen ontstaat een antwoord betreffende de hoofdvraag. Voor het onderzoek kan gekozen worden voor een kwalitatieve of een kwantitatieve opzet, wat de verdere keuze in aanpak beïnvloedt. Bij kwalitatief onderzoek staat de subjectieve beleving van de onderzoekseenheid centraal, waarmee dit type onderzoek interpretatief van aard is[1]. Daarbij worden bij dit type onderzoek de onderzoekseenheid in de omgeving als geheel onderzocht, wat de onderzoeksmethode ideaal maakt voor het onderzoeken van individuen. Bij kwantitatief onderzoek verzamelt de onderzoeker cijfermatige of numerieke gegevens en staat het individu veel minder centraal[2]. Bij dit type onderzoek ligt de nadruk meer op de beleving van een groep onderzoekseenheden.

Voor het grootste deel is in dit onderzoek de keuze gemaakt voor een kwalitatieve opzet, waarvoor een aantal redenen zijn te benoemen. Ten eerste is het belangrijk dat de huidige ontwikkelsituatie van 42 BV in kaart wordt gebracht, iets dat kwalitatieve gegevens betreft. Binnen 42 BV is een gestandaardiseerde ontwikkelstraat van toepassing. Voor het onderzoek is het relevant te achterhalen waarom deze ontwikkelstraat zo gestandaardiseerd is en waar de diverse onderzoekaspecten voor gebruikt worden. Dit betreft kwalitatieve gegevens omdat wordt ingegaan op de individuele onderzoekseenheden.

Ten tweede wordt bij de app types gezocht naar user experiences. Hierin is een grote mate van persoonlijke ervaring en meningen van individuele personen van toepassing, iets wat normaliter niet kwantitatief wordt gemeten.

Als laatste reden voor het kiezen van een kwalitatieve opzet is gegeven dat de diverse ontwikkelsituaties van de verschillende app types onderzocht moeten worden. Evenals bij de ontwikkelstraat van 42 BV wordt ook hierbij de individuele onderzoekseenheid onderzocht.

Een uitzondering op dit onderzoek type kan worden gemaakt voor de derde deelvraag. Zowel de eerste als tweede onderzoeksvraag heeft betrekking tot kwantitatieve gegevens. In het eerste geval worden diverse klanten gevraagd voor welke apparaten zij mobiel gebruik van de applicatie willen. Bij de tweede onderzoeksvraag is het zaak te onderzoeken wat het marktaandeel binnen de huidige mobiele markt momenteel doet. Dit betreft cijfermatige gegevens waardoor in eerste instantie gekozen wordt voor een kwantitatieve opzet.

[1] Hoogers-1
[2] Hoogers-2

2.1 Onderzoeksmethoden

Met de keuze tussen kwalitatief en kwantitatief onderzoek in gedachte zijn er een aantal onderzoeksmethoden gekozen per deelvraag.

Deelvraag 1 – Welke app types bestaan er en hoe ziet de ontwikkeling van deze app types eruit?

- Welke app types bestaan er?
- Hoe ziet de ontwikkelstraat per app type eruit?
- Tegen welke overige problemen lopen ontwikkelaars aan (user experiences)?

Voor dit deel van het onderzoek wordt met name de methode literatuuronderzoek gebruikt. Hierbij wordt gebruik gemaakt van het internet. Wanneer bruikbare boeken over dit onderwerp zijn geschreven worden deze wellicht ook gebruikt. De user experiences worden op diverse manieren achterhaalt, te beginnen met blog-posts op internet. Daarnaast oriënteer ik mij op het aansluiten bij een app-developer community. Een suggestie die vanuit 42 BV is gedaan is de community Appsterdam, uit deze community hoop ik waardevolle informatie te krijgen betreffende deze onderwerpen. Deze informatie wordt vergaard door middel van interviews.

Populatie

Bij de derde onderzoeksvraag wordt gebruik gemaakt van een populatie, de ontwikkelaars op gebied van app development. Niet de gehele populatie wordt onderzocht, er wordt gekeken naar twee deelpopulaties die gebruikt worden tijdens dit onderzoek. De eerste deelpopulatie is de populatie die regelmatig actief gebruik maakt van het internet. Met actief gebruik maken van het internet wordt in deze context bedoeld het stellen van vragen op fora, het formuleren van problemen of het bijhouden van blog-posts. De tweede deelpopulatie zijn de app ontwikkelaars in mijn eigen directe omgeving. Dit kunnen kennissen en collega's zijn die apps ontwikkelen. Om de populatie representatiever te maken oriënteer ik mij op het aansluiten bij een app-developers community. Hierdoor worden ook de ontwikkelaars die ik tijdens de bijeenkomsten van de (Appsterdam) community ontmoet bij deze deelpopulatie betrokken.

Deelvraag 2 – Hoe ziet de huidige ontwikkelstraat van 42 BV eruit en welke beoordeling kan, in vergelijking daarmee, per app type worden gegeven?

- Hoe ziet de huidige ontwikkelstraat van 42 BV eruit?
- In welke mate zijn aspecten en gebruikte technologieën uit de huidige ontwikkelstraat terug te vinden bij de diverse app types?

Om de eerste onderzoeksvraag te beantwoorden worden interviews met medewerkers van 42 BV gehouden en wordt gebruik gemaakt van de ervaring die ik heb opgedaan binnen het bedrijf. Deze gesprekken worden niet als officiële interviews geïntroduceerd, maar als informele gesprekken. Dit geeft mij de mogelijkheid om zaken die ik in de wandelgangen bij 42 BV hoor, of die tijdens de stand-ups naar voren komen, mee te nemen in het onderzoek. Dit wordt ook wel de "Observeren in het Veld"-

methode genoemd[3], doordat in de alledaagse situatie informatie wordt vergaard. Het betreft hier een observatie van gedrag doordat collega's die tegen problemen aanlopen daar hun eigen oplossing voor bedenken en uitvoeren.

Vervolgens wordt gekeken naar welk van deze aspecten van de huidige ontwikkelstraat zijn terug te vinden binnen de appsmarkt. Om hier voor 42 BV een keuze in te laten maken, zullen de diverse app types worden beoordeeld op bepaalde aspecten. Het verkrijgen van deze informatie over app types wordt gedaan door middel van literatuuronderzoek. Als voorbeeld wordt onderzocht of de testmethoden die 42 BV in de huidige situatie toepast ook kunnen worden bij het ontwikkelen van de diverse app types. Hierbij wordt ook rekening gehouden met de kennis die momenteel al binnen het bedrijf aanwezig is.

Populatie
Bij deze deelvraag bestaat de populatie uit de medewerkers van 42 BV die dagelijks met de ontwikkelstraat te maken hebben. Medewerkers van bijvoorbeeld marketing en administratie vallen hierdoor buiten de populatie.

Deelvraag 3 – Welke vraag naar applicaties, specifiek voor bepaalde platformen, is er vanuit de apps markt?

- Naar welke platformen is er vanuit de klanten van 42 BV veel vraag?
- Wat zegt het marktaandeel van besturingssystemen binnen de mobiele markt?

Als laatste wordt achterhaald waar veel vraag naar is vanuit de klanten van 42 BV. De uitkomst hiervan wordt vervolgens afgewogen tegen wat de marktcijfers zeggen. Gezien al diverse onderzoeken naar het marktaandeel van mobiele besturingssystemen zijn gedaan, wordt bij dit laatste punt gebruik gemaakt van externe onderzoeken.

Populatie
De populatie van de eerste onderzoeksvraag zijn de klanten van 42 BV. Echter wordt niet de gehele populatie gevraagd over dit onderwerp. De onderzochte deelpopulatie bestaat dan ook alleen uit klanten die geïnteresseerd zijn in mobiele applicaties. Gezien intern al redelijk bekend is wat de klanten voor wensen hebben, is hier misschien geen onderzoek voor nodig.

[3] Hoogers-1

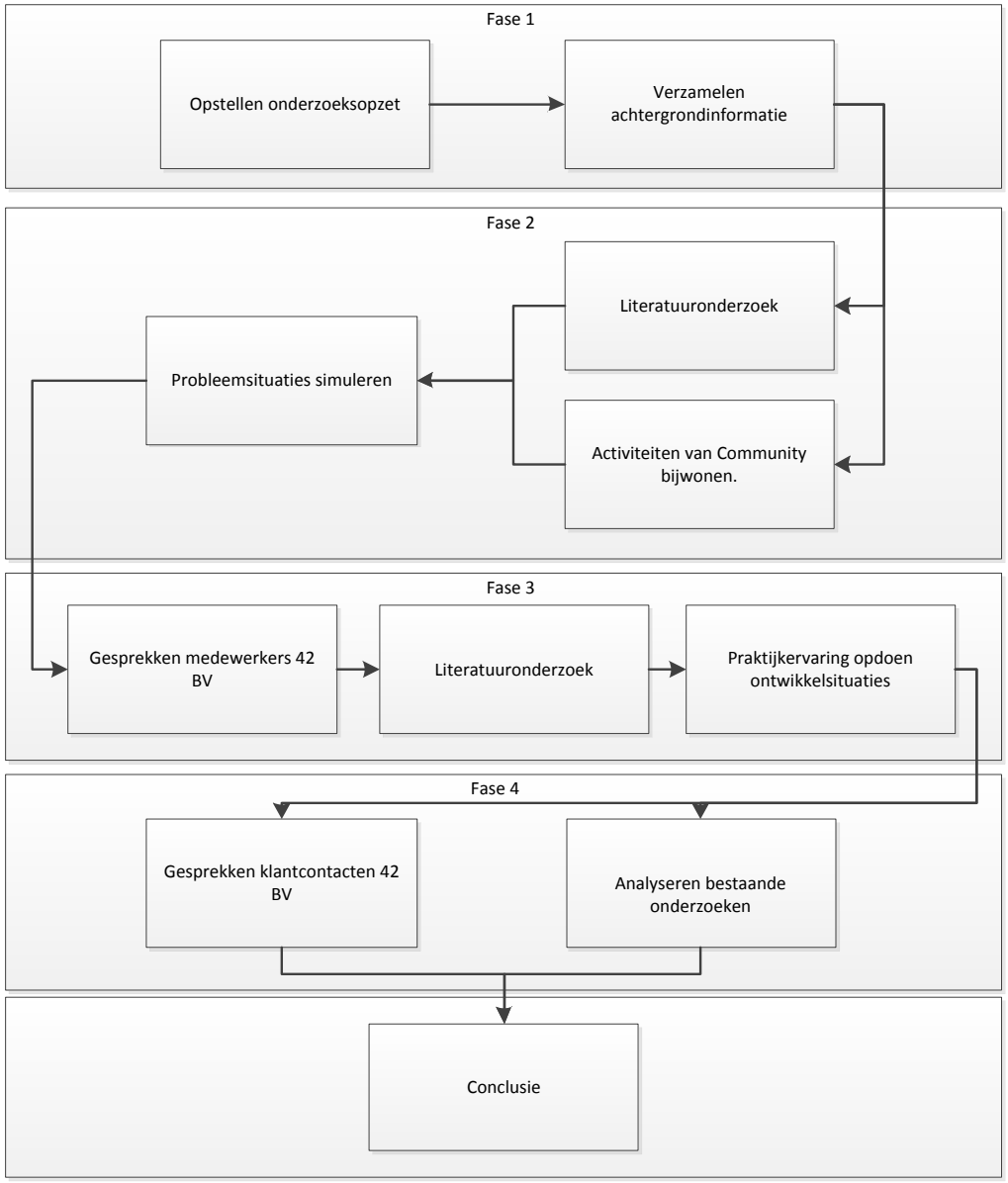
2.2 Kwaliteit waarborging

De kwaliteit van het totale onderzoeksproces wordt bepaald door verschillende aspecten. Om de betrouwbaarheid en validiteit van het onderzoek te waarborgen wordt gebruik gemaakt van triangulatie, wat inhoudt dat verschillende onderzoeksmethoden worden toegepast. De verscheidenheid aan onderzoeksmethoden en bronnen komt de uiteindelijke kwaliteit van het onderzoek ten goede.

Door met name bij de eerste deelvraag veelvuldig gebruik te maken van de ervaring die andere ontwikkelaars hebben in de praktijk, is het gevaar dat bij de conclusie teveel nadruk op individuele beleving komt te liggen in plaats van op waarheid beruste feiten. Dit komt doordat persoonlijke ervaringen, vergaart via het internet, niet gevalideerd zijn. Om de negatieve ervaringen te valideren wordt nooit uitgegaan van één bron, maar wordt getracht meerdere bronnen te vinden over bepaalde onderwerpen. Daarbij moet worden gelet dat in dit document alleen de hoofdbron als verwijzing wordt weergegeven. Indien het valideren door middel van meerdere bronnen niet lukt, wordt geprobeerd de probleemsituatie te valideren door deze situaties lokaal te simuleren. Dit geeft meer waarde aan de gestelde ervaring. Wanneer gecontroleerd is dat het probleem inderdaad onopgelost is, is de kwaliteit van de vernomen ervaring hoger.

2.3 Planning

Om het onderzoek tot een goed einde te brengen wordt gewerkt aan de hand van een vijf fases tellende planning. Deze fases, met bijbehorende activiteiten zijn weergegeven in onderstaand diagram. Zoals de tijdstabel doet vermoeden zullen de fases en bijbehorende activiteiten aan de hand van de deelvragen en onderzoeksvragen worden ingedeeld.



Tijd (in dagen)	*Fase*	*Hoofdactiviteit*
4	1	Voorbereiding
7	2	Beantwoorden deelvraag 1
5	3	Beantwoorden deelvraag 2
3	4	Beantwoorden deelvraag 3
4	5	Conclusie

3 Achtergrondinformatie

Voordat wordt onderzocht hoe mobiele applicaties worden ontwikkeld is het goed om te weten waarom deze tegenwoordig zo populair zijn. Mobiele applicaties worden gemaakt voor smartphones, maar waar komen deze apparaten nu eigenlijk vandaan en hoe komt het dat zij zo populair zijn geworden?

3.1 De Smartphone

Een smartphone is een mobiele telefoon die, in tegenstelling tot de feature phone, uitgebreidere computer mogelijkheden biedt. De eerste smartphone stampt uit 1992, de IBM Simon[4]. Naast mobiele telefonie had deze smartphone functionaliteiten als een adres boek, een wereld klok, een email programma en konden faxen worden verzonden en ontvangen. In 1996 (en de daarop volgende jaren) kwam Nokia met haar eerste smartphone serie, de Nokia Communicator's[5].

Laat in de jaren '90 had het grote merendeel van de mobiele telefoons alleen standaard telefoon functies. Ondanks dat smartphones al bestonden waren deze niet veel gebruikt. Mensen die meer mogelijkheden benodigde droegen daarom een PDA (Personal Digital Assistant) met zich mee.

In 2007 introduceerde Apple de eerste iPhone. Deze 'smartphone' werd getypeerd door het grote multi-touch scherm, die de telefoon in staat stelde om directe vinger invoer als hoofdinvoer te gebruiken. Deze directe vinger invoer kwam in de plaats van de gebruikelijke stylus en toetsenbord invoer en bracht voor veel mensen een groot voordeel met zich mee ten overstaande van vorige smartphones. De iPhone was een unieke combinatie van mobiele telefoon, pda en de toen al populaire iPod (een mp3 speler). Op de vraag aan consumenten wat nu de eerste smartphone is wordt veelal teruggewezen naar deze iPhone. Toch stellen critici, dat de eerste iPhone geen echte smartphone was. De definitie van smartphone stelt namelijk dat third-party software geïnstalleerd moet kunnen worden, terwijl de eerste iPhone alleen toegang had tot first-party software. Daarbij was de eerste generatie iPhones niet in Nederland te verkrijgen[6].

[4] Business 2 Community
[5] Encyclopedie-1
[6] Verkuil

3.2 Smartphone Hype

Dat smartphones de afgelopen vijf jaar in populariteit zijn gestegen is duidelijk[7]. Wanneer aan de dagelijkse gebruiker van mobiele telefoons wordt gevraagd naar de oorzaak van deze zogenoemde Smartphone Hype wordt veelal verwezen naar de iPhone, waarvan de tweede generatie in Nederland uitkwam in juli 2008. Deze kwam tegelijkertijd met de release van Apple's App Store, een online winkel met software die elke iPhone officieel in staat stelt third-party software te installeren[8].

Of de iPhone ook de daadwerkelijke oorzaak is geweest van de smartphone hype is moeilijk te zeggen en de meningen zijn veelal verdeeld. Dit komt doordat in 2007 meerdere smartphones (waaronder de Nokia N95) zijn uitgebracht die populair werden. Er wordt verder dan ook geen uitspraak gedaan over de oorzaak van de hype.

3.3 App Stores

Wat wel met zekerheid is te zeggen, is dat smartphones met de komst van de App Store een stuk populairder zijn geworden. Deze App Store, als één van de eerste in zijn soort, stelt de gebruiker in staat om op één centraal punt mobiele applicaties te downloaden, in plaats van dat de gebruiker daarvoor meerdere websites bezoekt. Deze mobiele applicaties worden ook wel Apps genoemd en variëren van het lezen van een boek tot het spelen van spelletjes en het opvragen van de weersvoorspelling[9]. Voorafgaand werden deze Apps door third-party partijen aangeboden, wat tot spreiding binnen de markt leidde. Al bij de release van de App Store leek het naar alle waarschijnlijkheid een immens populair platform te worden. In de eerste drie dagen na release waren er al meer dan 10 miljoen downloads, terwijl de keuze voor apps vrij beperkt was. Bij aanvang van de release bestond de App Store uit 500 apps[10]. Deze apps in combinatie met de smartphone brengen voor veel mensen, veel gemak met zich mee. Zij kunnen terwijl zij in de trein zitten, hun email lezen, presentaties voorbereiden en tegelijkertijd muziek luisteren. Maar daar houden de mogelijkheden niet op, ook kan men de beurskoersen bijhouden, spelletjes spelen en nog veel meer.

Sinds de ontwikkeling van de App Store is de smartphone markt alleen maar verder gegroeid en al snel werden nieuwe mogelijkheden geïntroduceerd[11]. In 2008 kwam Google met een nieuw open-source besturingssysteem, Android. Veel grote telefoon fabrikanten (waaronder HTC, Motorola en Samsung) starten hiermee een smartphone reeks welke de markt verder heeft doen uitbreiden[12]. Apps voor Android werden verzameld in Google's Marketplace en ook fabrikant RIM kwam met een eigen app store voor de BlackBerry serie, BlackBerry App World. Door de komst van de nieuwe smartphones is het aandeel van smartphones ten overstaande van feature phones een stuk groter geworden.

[7] Mediawijzer
[8] Encyclopedie-2
[9] Media Optima Forma
[10] Apple-1
[11] Mediawijzer
[12] Encyclopedie-1

3.4 Marktaandeel

Eén van de meest recente onderzoeken (binnen Nederland), op het gebied van het aandeel van smartphones in de mobiele markt, is die van het bedrijf Telecompaper[13]. Dit bedrijf heeft in het tweede kwartaal van 2011 een onderzoek gedaan naar het percentage smartphone-gebruikers onder de Nederlandse consument. Het resultaat van dit onderzoek is dat vier op de tien Nederlandse consumenten (42%) een smartphone bezit en gebruikt. Daarmee is het aandeel smartphone-gebruikers in een korte tijd flink gestegen. Het derde kwartaal van 2010 bedroeg dit aantal namelijk nog 30%. Het onderzoek is uitgevoerd onder 19.200 consumenten in de leeftijdscategorie van 15 tot 64 jaar.

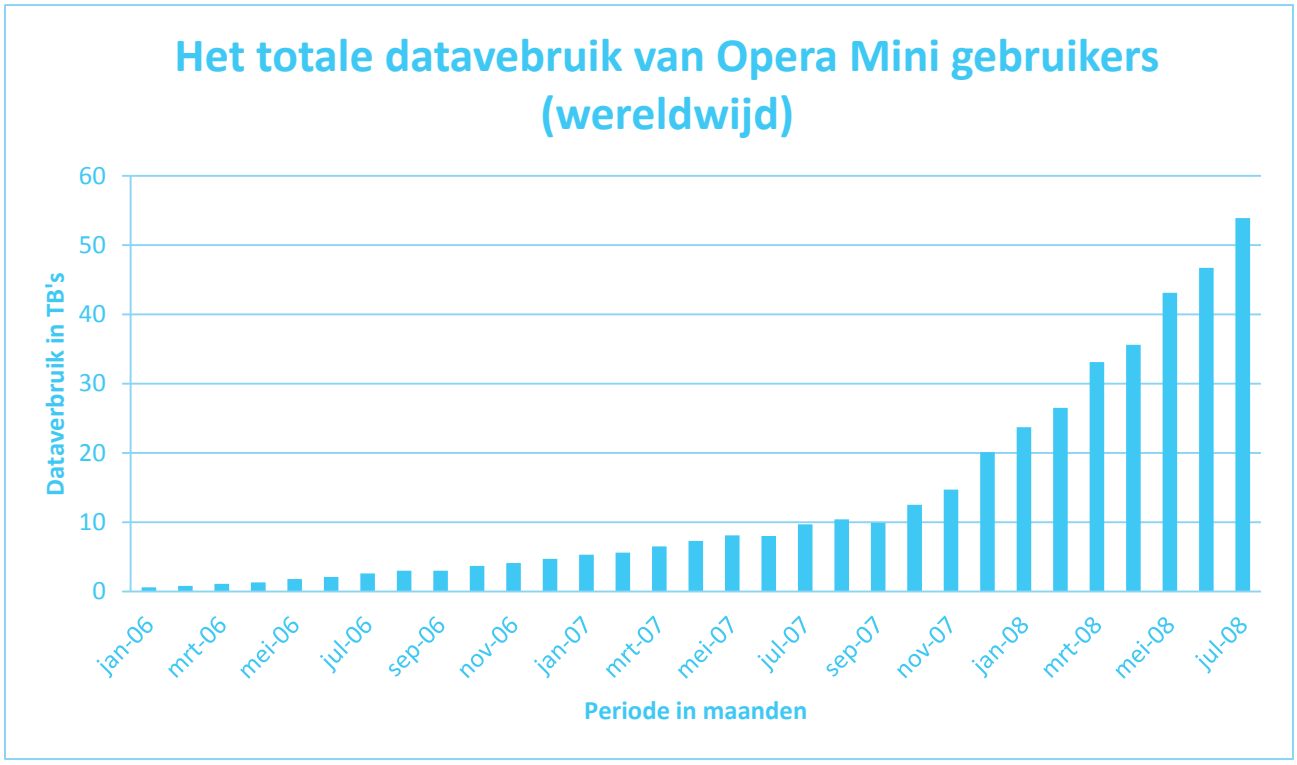
Ook wereldwijd wordt het aandeel aan smartphones alsmaar groter. Onderzoek- en adviesbureau Gartner onderzoekt regelmatig het marktaandeel op basis van verkoopstatistieken van mobiele telefoons. Waar de verkoop van smartphones in 2007 nog maar goed voor 11 procent van het wereldaandeel was, was dit 19.3% in 2010 en 26% in het derde kwartaal van 2011[14].

[13] Niezink
[14] Gartner-1,2,3

3.5 Mobiel internet

Smartphones brengen een groot gemak met zich mee, de mobiliteit. Mensen kunnen door de komst van de smartphone plaats en tijdonafhankelijk werken. Er is echter wel één vereiste, een mobiele internet connectie. Smartphones kunnen ook zonder internet connectie worden gebruikt, maar veel functionaliteit valt in dat geval weg. Voorbeelden van wegvallende functionaliteit zijn email ophalen, beurskoersen bijhouden, het nieuws lezen, enzovoort.

De groei van het marktaandeel van smartphones is dan ook niet alleen te wijten aan de nieuwe smartphones, de komst van diverse app stores en de nieuwe mogelijkheden die apps bieden. Om deze ontwikkelingen mogelijk te maken zou namelijk ook het mobiele netwerk mee moeten groeien. Mobiel internet was voor het eerst toegankelijk in 1996 op de Nokia 9000 Communicator, maar is pas een echt grote groei aan het meemaken sinds 2006[15]. Opera Mini heeft statistieken vrijgegeven van de periode 2006 tot 2008 betreffende mobiel data verkeer die de mobiele webbrowser heeft gegenereerd. De statistieken laten een toename in data verbruik zien van bijna 9000%[16]. Deze toename zou niet plaats kunnen vinden zonder groei in het mobiele netwerk. Dit wordt nog eens extra benadrukt door de directeur van telecomprovider T-Mobile[17]. Volgens de cijfers die hij heeft genoemd in 2011, heeft T-Mobile in het jaar daarvoor (2010) 142 miljoen euro geïnvesteerd in het mobiele netwerk.



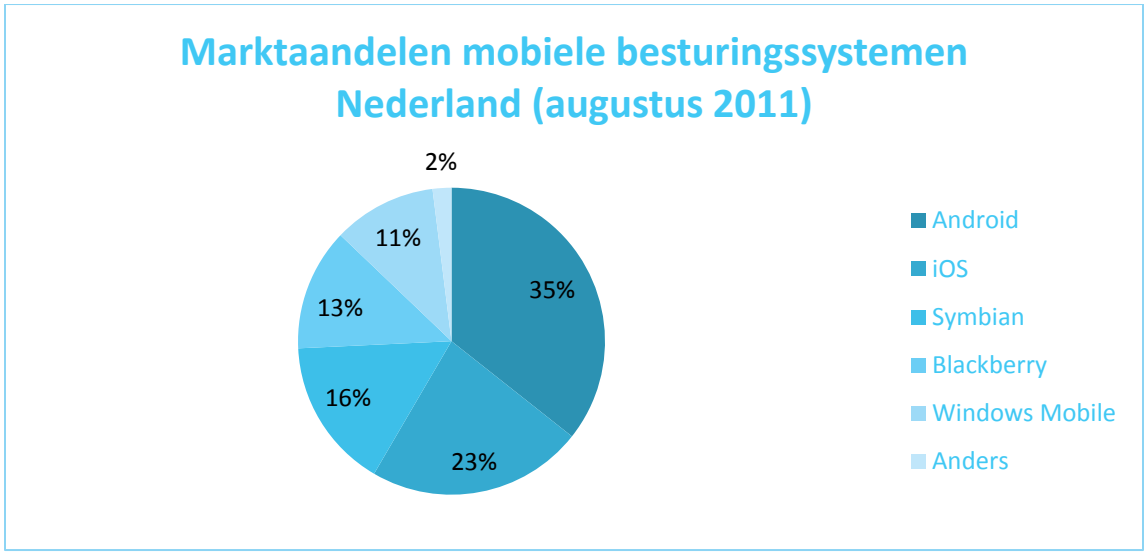
[15] Norman
[16] Von Tetzchner
[17] FD

3.6 Vooronderzoek

Aan het begin van het afstudeerproject heb ik een klein vooronderzoek gedaan om het project af te bakenen en te controleren of een soortgelijk onderzoek in het verleden al is gedaan. Dit is gedaan door middel van gesprekken met 42 BV en internet research. In dit vooronderzoek zijn een aantal belangrijke keuzes en afbakeningen gemaakt betreffende app ontwikkelmogelijkheden en besturingssystemen. Deze keuzes en afbakeningen zijn ook terug te vinden in het Plan van Aanpak.

3.6.1 Besturingssystemen

Binnen de appsmarkt is een grote diversiteit aan apparaten en besturingssystemen aanwezig wat het ontwikkelen van een universele applicatie sterk beïnvloed. Voornamelijk het besturingssysteem bepaald op welke apparaten de nieuwe applicatie gaat werken. In augustus 2011 heeft het bedrijf TNS NIPO een onderzoek verricht naar het marktaandeel van mobiele besturingssystemen binnen Nederland[18]. Zij hebben een screening gehouden onder 16.000 smartphonebezitters. De vraag daarbij was welk besturingssysteem zij op hun smartphone draaien. Hier is geen rekening gehouden met tabletgebruikers. Het onderzoek leverde de volgende resultaten op:



Uit de resultaten blijkt dat er een aantal grote besturingssystemen te herkennen zijn: Google's Android, Apple's iOS, Nokia's Symbian, RIM's BlackBerry en Microsoft's Windows Mobile.

42 BV kiest er vooralsnog voor alleen Google's Android en iOS te onderzoeken. Hier noemt het bedrijf verschillende redenen voor:

- Vanuit de klant is er veel vraag naar iPhone apps (welke draaien onder het iOS besturingssysteem).

[18] Bleijendaal

- 42 BV ziet veel gelijkenissen tussen ontwikkelen voor Android en de huidige ontwikkelmethodieken van het bedrijf. Beide maken namelijk gebruik van de Java programmeertaal.

Deze twee redenen in combinatie met de resultaten van het vooronderzoek, waaruit blijkt dat Android en iOS samen veruit het grootste marktaandeel in handen hebben, maakt de keuze van beperking tot deze twee besturingssystemen markttechnisch gezien logisch.

3.6.2 App ontwikkelmogelijkheden

Voor het ontwikkelen van apps kunnen verschillende app ontwikkelmogelijkheden worden gebruikt. Zoals eerder gesteld bepaald het besturingssysteem van een smartphone welke apps wel en welke apps niet werken op het desbetreffende apparaat. De verschillende app ontwikkelmogelijkheden stellen de ontwikkelaar in staat om voor één specifiek besturingssysteem, of voor meerdere besturingssystemen tegelijk een app te ontwikkelen.

Voorafgaand aan het onderzoek stelt 42 BV dat er binnen de apps ontwikkeling vier van deze verschillende mogelijkheden zijn te definiëren. Deze zijn als volgt:

- *Volledig browser-based*
  Er kan worden gekozen om de gehele applicatie browser gebaseerd te maken, waardoor voor de front-end gebruik wordt gemaakt van bijvoorbeeld HTML en Javascript. Browser-based applicaties zijn universeel over vrijwel alle platformen (hoge portabiliteit) en is dus een makkelijke oplossing. Deze oplossing wordt momenteel binnen 42 BV gebruikt voor klanten die hun applicatie mobiel willen gebruiken.

- *Native*
  De applicatie kan in een native programmeertaal worden geschreven wat inhoud dat hij specifiek voor Android of iOS is. Dit betekend als men wil dat de applicatie op beide besturingssystemen werkt, hij twee keer ‘opnieuw’ gemaakt moet worden. In dit geval is dat eenmaal in Java (voor Android) en eenmaal in Objective C (voor iOS).

- *Semi-native*
  Wanneer een applicatie semi-native wordt gemaakt, wordt voor meerdere besturingssystemen een applicatie gemaakt. Deze applicaties roepen vervolgens de browserfunctionaliteiten van het besturingssysteem aan. Het voordeel hiervan is dat er maar een klein gedeelte van de code in een native taal geschreven hoeft te worden. De rest van de functionaliteit wordt door middel van een universele (web) taal geschreven. Dit houdt onder andere in dat de inhoud van de applicatie centraal is aan te passen.

- *Tussentaal*
  Er zijn mogelijkheden tot gebruik van een ‘tussentaal’ welke bij het compileren ervoor zorgt dat de applicatie voor meerdere besturingssystemen geschikt is.

4 Deelvraag één – App types

In dit hoofdstuk komen de resultaten naar voren die tijdens de beantwoording van de eerste deelvraag van het onderzoek zijn verkregen. De eerste deelvraag betreft het onderzoeken van de bestaande app ontwikkelmogelijkheden. De deelvraag is als volgt geformuleerd:

> *Welke app types bestaan er en hoe ziet de ontwikkeling van deze app types eruit?*

Bij deze deelvraag horen zoals eerder gesteld een aantal onderzoeksvragen. Deze onderzoeksvragen worden individueel behandeld in de volgende sub-paragrafen. Het geheel wordt afgesloten met een passende conclusie, die dient als antwoord op de eerste deelvraag.

4.1 Welke app types bestaan er?

Om de eerste deelvraag te beantwoorden moet eerst worden achterhaald welke app types er bestaan binnen de appsmarkt. Uit het vooronderzoek blijkt dat 42 BV voorafgaand aan dit project al heeft nagedacht over de verschillende app ontwikkelmogelijkheden. Deze mogelijkheden resulteren in app types en zijn genoemd in ***Paragraaf 3.6 – Vooronderzoek***.

Tijdens het zoeken naar de app types en bijbehorende ontwikkelstraten (cross platform mobile development), kwam ik uit op een interessante presentatie van Peter Friese, een software ontwikkelaar met meer dan 15 jaar ervaring in de software ontwikkeling[19]. Eén van zijn expertises is het ontwikkelen van cross-platform apps. In deze presentatie benadrukt Peter nogmaals hoe belangrijk smartphones in ons dagelijks leven zijn geworden, dat de mobiele web wereld nog steeds groeit en dat deze de komende jaren naar alle waarschijnlijkheid blijft groeien. Verder worden in deze presentatie zes verschillende app types genoemd, met een daarbij behorende methode van ontwikkeling. Deze app types zijn ofwel platform specifiek, dan wel cross-platform. Vergelijkend met andere, soortgelijke artikelen, is dit een goed beginpunt om te starten. De genoemde app types kunnen worden onderverdeeld in drie hoofdcategorieën:

Categorie	*App type*
Native	De Native Experience
Web	Mobile Web
	Client-side web
Cross-Platform	Hybrid app
	Interpreted app
	Cross-compiling

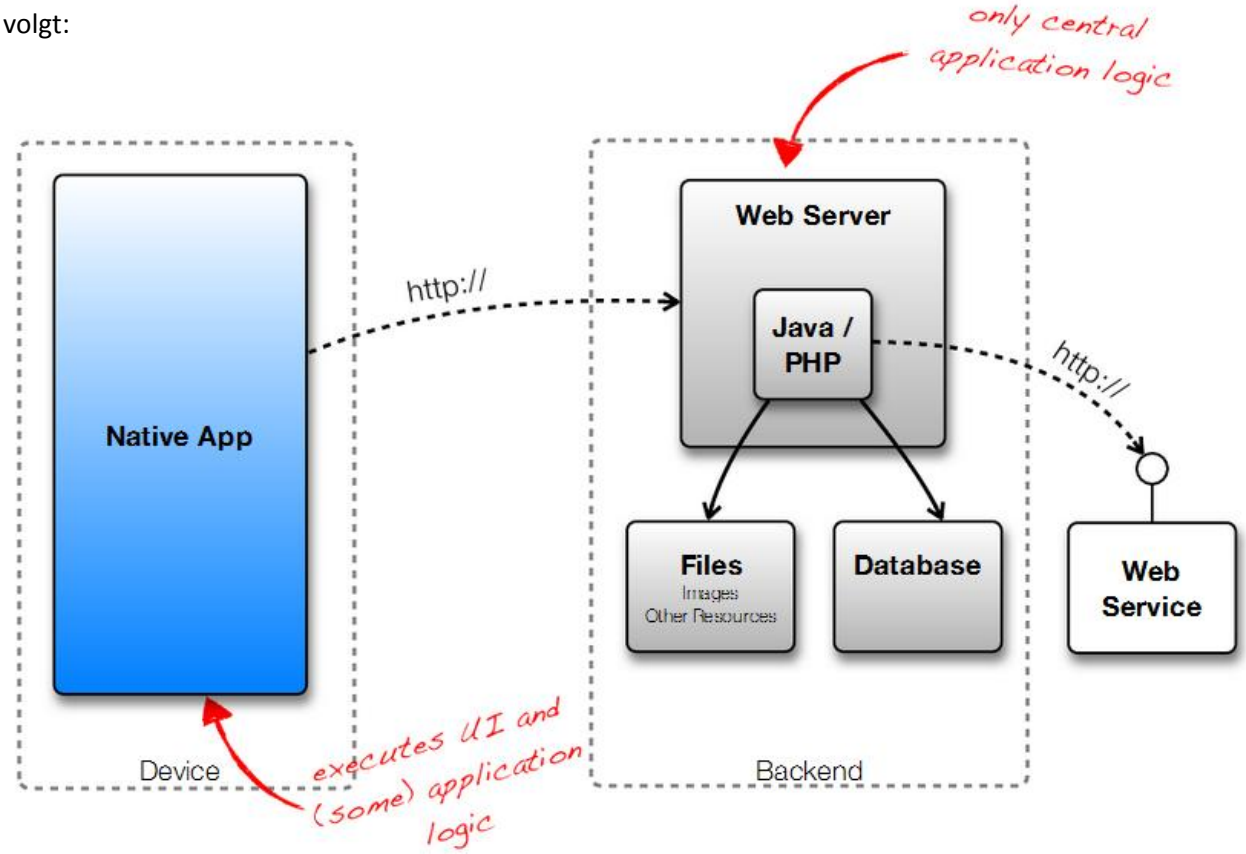
In de volgende paragrafen zullen de genoemde app types per categorie nader worden toegelicht.

[19] Friese-1,2

4.1.1 Native

Binnen de Native categorie is in principe maar één app type te vinden, namelijk die van de Native Experience. Dit app type is echter wel onder te verdelen door de diverse platformen die bestaan binnen de Native categorie. De belangrijkste platformen zijn als volgt: Android, iOS, BlackBerry, BlackBerry PlayBook OS, Symbian en Windows Phone[20].

Bij het kiezen van dit app type wordt voor een specifiek apparaat een app ontwikkeld die de User Interface (UI) en de daarbij behorende logica uitvoert. Vervolgens wordt door middel van http requests met de Web server gecommuniceerd. In de inleiding van dit hoofdstuk kwam de presentatie van Peter Friese naar voren, in dezelfde presentatie heeft Peter per app type handige overzichtsdiagrammen gebruikt, die de structuur van de app types goed weergeeft. Voor de Native app type is het overzicht als volgt:



Bron – Cross-platform mobile development presentatie door Peter Friese

Eerder gezegd is al dat voor 42 BV een beperking tot twee platformen vooralsnog interessant is, namelijk de beperking tot Android en iOS. Doordat deze twee sterk van elkaar verschillen wordt in het vervolg van dit document uitgegaan van twee verschillende app types, de Android app en de iOS app.

[20] Consumenten Bond-1, Encyclopedie-1

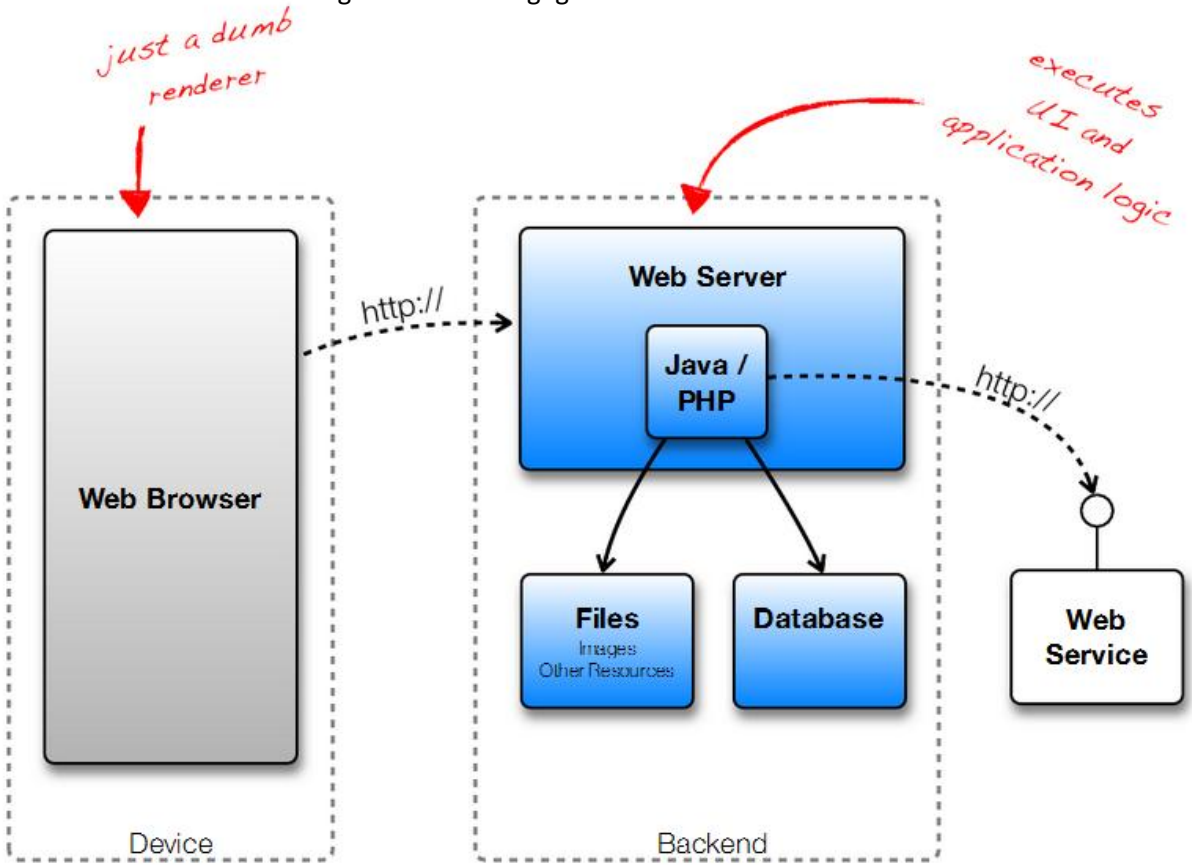
4.1.2 Web

Binnen de webcategorie zijn twee app types te herkennen, namelijk de Mobile Web app en Client-Side web app. Bij de web methode wordt een website ontwikkeld welke kan worden benaderd door een webbrowser. Gezien het browsen van webpagina's door alle moderne smartphones wordt ondersteund is deze methode in principe de meest geschikte methode om zo een groot mogelijk publiek te bereiken. Daar tegen over staat wel de hoeveelheid verschillende browsers op de markt. Waar de ene browser support heeft voor moderne technologieën als CSS3 en HTML5, zijn er ook nog een groot aantal die alleen voorgaande versies van deze technologieën ondersteunen.

De twee verschillende app types binnen de Web methode worden in de volgende sub-paragrafen besproken.

The Mobile Web

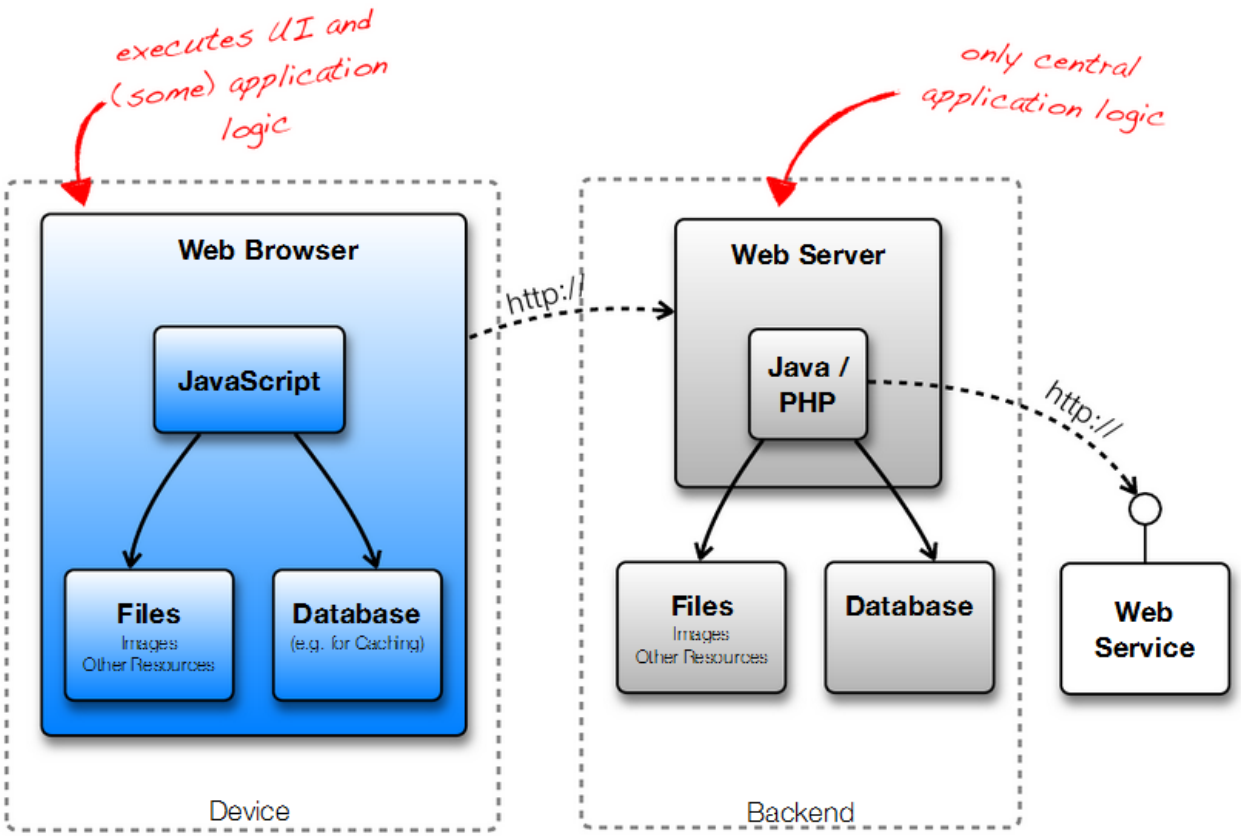
Bij het Mobile Web type wordt een website in twee verschillende varianten gemaakt. De eerste variant is de gebruikelijke desktop versie, zoals wij die ook kennen van het surfen op internet door middel van de computer. De tweede variant is een verkleinde versie van dezelfde website, waarbij tijdens de ontwikkeling rekening wordt gehouden met het kleinere scherm van mobile devices. Hierdoor kan ook besloten worden bepaalde inhoud weg te laten of toe te voegen in de mobiele variant. Zowel de UI als de applicatie logica wordt aan de back-end uitgevoerd, waardoor het apparaat alleen de weergave regelt. Schematisch kan dit als volgt worden weergegeven:



Bron – Cross-platform mobile development presentatie door Peter Friese

Client-Side Web

Bij het Client-Side web type wordt de app, net als bij het Mobile web type, uitgevoerd door middel van een webbrowser. Het enige verschil is dat het in dit geval ook mogelijk is functionele logica uit te voeren op het apparaat. De back-end voert vervolgens alleen centrale logica uit voor bijvoorbeeld de validatie en beveiliging van de app. Schematisch kan dit als volgt worden weergegeven:



Bron – Cross-platform mobile development presentatie door Peter Friese

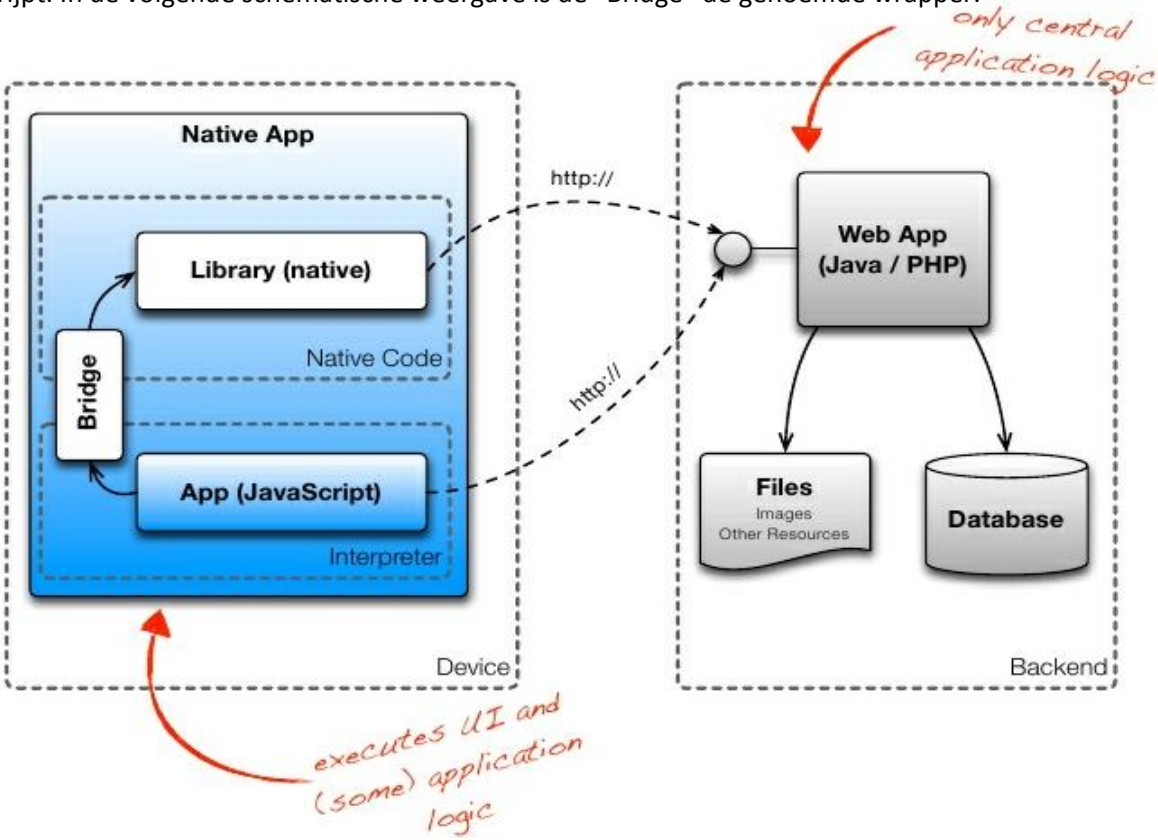
> ***Let op:*** Doordat deze methode vaak in combinatie met die van de Mobile web methode wordt gebruikt en voor een ontwikkelaar vrijwel identiek zullen zijn, valt dit in de rest van dit document onder het type Web app.

4.1.3 Cross-Platform

De derde categorie is de Cross-Platform categorie. Binnen deze categorie zijn drie verschillende app types te definiëren, namelijk de Hybrid apps, de Interpreted apps en Cross-Compiling apps. Deze drie app types zijn allen gericht op het ontwikkelen door middel van een Single-Code-Base, wat inhoud dat zoveel mogelijk code wordt geschreven in een zelfde taal, die dan wel geïnterpreteerd, dan wel omgezet kan worden naar een taal die meerdere apparaten begrijpen. In de volgende sub-paragrafen worden de verschillende app types nader toegelicht.

Hybrid Apps

Bij Hybrid app ontwikkeling wordt de code van de app in een andere taal dan native geschreven (geprobeerd wordt de Native taal zo minimaal mogelijk te gebruiken). Wanneer de code compleet is wordt door middel van encapsuleren de code geëncapsuleerd in een zogenoemde “web view wrapper”. De inhoud van de applicatie draait daardoor in een web omgeving die is gedefinieerd door het besturingssysteem, waardoor het besturingssysteem kan communiceren met de code die wordt gedefinieerd binnen de web view. Deze communicatie lijn wordt ook wel de ‘Bridge’ genoemd. De wrapper vertaalt dus de door de ontwikkelaar gedefinieerde acties naar formaat die het apparaat begrijpt. In de volgende schematische weergave is de “Bridge” de genoemde wrapper.

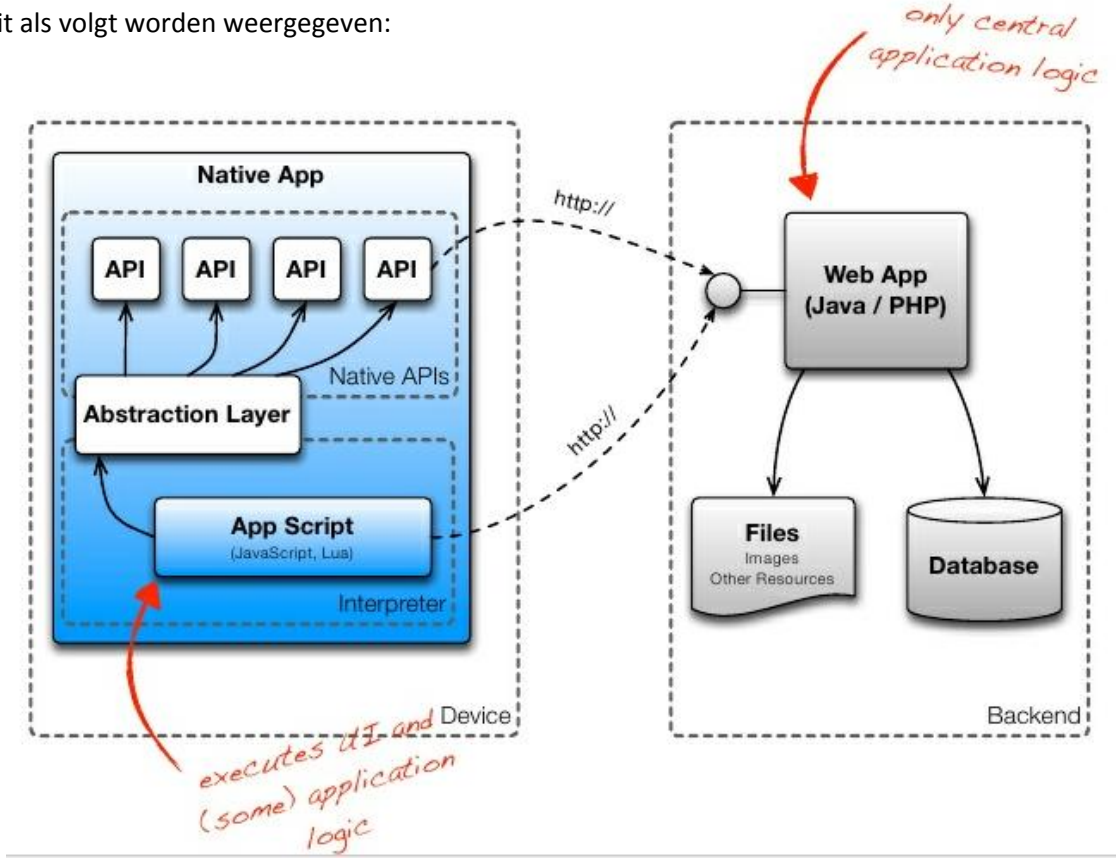


Bron – Cross-platform mobile development presentatie door Peter Friese

Eén van de bekendste frameworks om Hybrid Apps mee te ontwikkelen is PhoneGap. PhoneGap is een open source oplossing voor het ontwikkelen van cross-platform mobiele apps door gebruik te maken van standaard Web technologieën zoals HTML, JavaScript en CSS. Door middel van de eerder beschreven wrapper methode, krijgt de ontwikkelaar toegang tot Native APIs. Het resultaat van deze ontwikkelmethode is een app van het type Hybrid.

Interpreted Apps

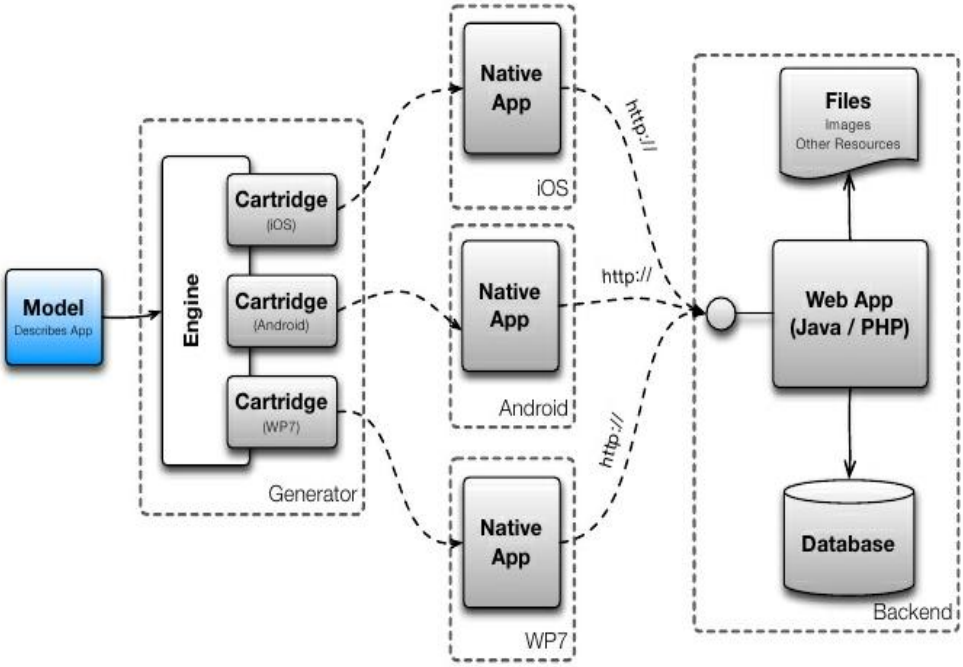
Interpreted apps gebruiken platform-specifieke UI elementen voor de interactie met de gebruiker. De applicatie logica daar in tegen wordt afgehandeld op een platform onafhankelijke manier. Dit laatste wordt bijvoorbeeld gedaan door middel van XML command sets of door een JavaScript-API. Schematisch kan dit als volgt worden weergegeven:



Bron – Cross-platform mobile development presentatie door Peter Friese

Cross Compiling

Bij de cross-compiling app ontwikkeling realiseert de ontwikkelaar een bepaalde input voor een zogenoemde generator. Deze generator zet de specifieke input vervolgens automatisch om in native applicaties. Schematisch kan dit als volgt worden weergegeven:



Bron – Cross-platform mobile development presentatie door Peter Friese

Door middel van een universele taal (of gebruik van een platform specifieke taal) wordt de app gerealiseerd. Vervolgens zorgt de generator ervoor dat de gerealiseerde app, of een deel daarvan wordt gecompileerd naar meerdere talen om gebruik op andere platformen mogelijk te maken. Een bekend platform binnen deze methode van app ontwikkeling is de Corona SDK. Deze methode wordt vaak in combinatie met bijvoorbeeld de Hybrid Apps methode gebruikt.

4.2 Hoe ziet de ontwikkelstraat per app type eruit?

Als tweede onderzoeksvraag is onderzocht hoe de ontwikkelstraat per app type eruit ziet. Hierbij zijn een aantal belangrijke aspecten onderzocht die in overleg met 42 BV zijn bepaald. Deze aspecten zijn binnen het bedrijf belangrijk om een uiteindelijke keuze te maken. De aspecten zijn als volgt:

- *Implementatie*
  Voor de ontwikkeling van specifieke app types kan per ontwikkelmethode worden gekozen voor een specifiek implementatie platform. Dit gaat vaak om een specifieke Software Development Kit (SDK) of framework. Gezien deze per app type verschillend zijn is het voor 42 BV interessant om te weten wat de bekendste per app type zijn, zodat hier in een later stadium rekening mee gehouden kan worden.

- *Ontwikkelomgeving (IDE)*
  Bij het ontwikkelen van software wordt vaak gebruik gemaakt van een Integrated Development Environment, ook wel afgekort als IDE. Het grote voordeel bij gebruik van een dergelijke IDE is dat er vaak debug en test ondersteuning is. Daarnaast kan er gedacht worden aan handige hulpmiddelen als Code Completion, Code Correction en Project Sharing. Tussen de diverse ontwikkelomgevingen zitten vaak verschillen op gebied van programmeertaal, intelligentie en plug-ins. Wanneer het ontwikkelen door middel van een IDE niet mogelijk is, bijvoorbeeld doordat de programmeertaal niet wordt ondersteund, vergt dit een stuk meer kennis van de ontwikkelaar. Daarbij is het werken zonder IDE, door het ontbreken van automatische correctie, fout gevoelig en ten zeerste af te raden.

  Doordat bij 42 BV voornamelijk gebruik wordt gemaakt van Eclipse en IntelliJ worden deze bij elk app type genoemd, met bij de extra informatie of deze IDE wel of niet gebruikt kunnen worden.

- *Programmeertaal*
  Voor het ontwikkelen van apps wordt, afhankelijk van het gekozen app type, gebruik gemaakt van verschillende programmeertalen. Wanneer een platform op een bepaalde programmeertaal is gebaseerd, dan wordt die programmeertaal ook wel de Native taal genoemd.

- *User Interface*
  De User Interface (UI) van een app is de weergave die de gebruiker te zien krijgt, waarbij de UI als interactie medium dient tussen de gebruiker en het systeem. Het is app type afhankelijk wat voor UI gebruikt wordt. Wanneer de Native UI (de UI die bij het specifieke platform hoort) gebruikt kan worden in een eigen applicatie wordt dit als voordeel gezien. Doordat gebruikers aan de look en feel van deze UI gewend zijn, kunnen zij hier vaak gelijk mee overweg. Het wordt dan ook automatisch als nadelig gezien wanneer de app er totaal anders uit ziet dan het platform waarvoor het is bedoeld. Bij sommige platformen zijn er specifieke eisen opgesteld wat UI betreft, dit komt later nog aan bod.

- *Eventafhandeling*
  Bij het gebruik van een smartphone creëert de gebruiker events. Deze events kunnen bijvoorbeeld plaatsvinden wanneer de gebruiker de smartphone draait, waardoor de schermoriëntatie veranderd naar horizontaal. Of de app types dit soort events ondersteunen wordt door dit aspect toegelicht.

- *Distributiewijze*
  Afhankelijk van het gekozen app type wordt de app gedistribueerd door middel van een distributie medium. Dit medium kan een verzamelomgeving zijn voor apps, maar ook een (vrij) toegankelijke plek op het internet, waar gebruikers de app kunnen downloaden. Net als bij de User Interface wordt het over het algemeen als voordeel gezien wanneer een app door middel van de Native distributiewijze kan worden gedistribueerd. Dit komt doordat gebruikers gewend zijn om op de Native manier apps te downloaden en te installeren. Voor 42 BV hoeft dit echter niet een voordeel te betekenen, omdat zij misschien niet bekend zijn met het op deze manier distribueren van apps.

- *Toegang tot Native APIs*
  Om het apparaat waar de app voor is bedoeld zo optimaal mogelijk te kunnen gebruiken is toegang vereist tot de diverse Native APIs. Deze APIs verschaffen toegang tot bijvoorbeeld de GPS functie, netwerk mogelijkheden en camera opties van de smartphone. Niet alle apps vereisen toegang tot deze functionaliteiten, maar het is belangrijk te weten welke app types de Native APIs ondersteunen, doordat het toekomst misschien nodig is.

- *Test platform & apparaten*
  Om een app te testen wordt het gecompileerd voor een bepaald platform of apparaat. Dit aspect geeft aan op welke apparaten en platformen de app getest kan worden.

- *Test framework*
  Aan de hand van test frameworks kan op code niveau een functionaliteit getest worden. Dit is over het algemeen sneller en minder foutgevoelig dan handmatig zelf de applicatie te testen.

- *Opslag mogelijkheden*
  Vrijwel elke app heeft een vorm van data opslag. Vaak gebeurt dit in de vorm van gegevens in een database, maar ook kan gedacht worden aan een lokaal bestandssysteem. De diverse app types hebben verschillende mogelijkheden qua data opslag. Met name op het gebied van caching is het voor 42 BV handig hier meer informatie over te hebben.

- *Portabiliteit*
  De portabiliteit van een app bepaald in welke mate het mogelijk is om met zo min mogelijk werk een app voor meerdere platformen beschikbaar te maken. Het wordt als voordeel gezien een goede portabiliteit te hebben, omdat dit zowel manuren als onderhoudskosten scheelt. De onderhoudskosten komen voort uit het feit dat een app, verschillend voor twee platformen, tweemaal meer onderhoudskosten benodigd dan een app die in één keer voor twee platformen beschikbaar is. Dit komt omdat bij een universele app een single-code base gebruikt kan worden, waar een niet universele app voor elk platform een eigen code base vereist. Met een single-code base kan een app in één keer voor meerdere platformen gewijzigd worden.

- *Build opties*
  De build opties geeft de mogelijkheden weer die het specifieke app type kent om van code uit de IDE als programma op het doelplatform te draaien. Daarbij wordt het als voordeel gezien wanneer het builden via de command-line kan. Dit is een voordeel omdat command-line commands geautomatiseerd kunnen worden voor bijvoorbeeld Continuous Integration.

- *Community*
  Onder het aspect community wordt de gehele development groep om het app type heen bedoeld, met name gericht op de gekozen implementatie. Binnen het community aspect zijn vier sub-aspecten gedefinieerd:

  - *Tools*
    Met tools worden alle hulpmiddelen bedoeld die specifiek voor de implementatie van het bepaalde app type zijn ontwikkeld.

  - *Omvang*
    De grootte van het aantal groepen of mensen die lid zijn van deze community.

  - *Activiteit*
    De mate van activiteit waarbij toevoeging aan de community wordt geleverd.

  - *Cultuur*
    De cultuur van de community met het oog op open of gesloten source code.

- *Maturity*
  Met de maturity wordt per app type en zijn implementaties bepaald hoelang deze al bestaan en in hoeverre zij compleet zijn ontwikkeld. Bij de Native app types wordt dit aspect overgeslagen. Binnen het Maturity aspect wordt specifiek gericht op vier sub-aspecten:

  - *Leeftijd*
    Hoelang de implementatie van het app type bestaat wordt bij dit punt toegelicht.

  - *Versie*
    De huidige versie van de gekozen implementatie.

  - *Issue management*
    Onderzocht wordt hoe de specifieke implementatie omgaat met issues die worden ingeschoten door de community. Hierbij wordt gelet op de reactietijd op gemelde issues en of de gemelde issues ook daadwerkelijk worden opgelost.

  - *Documentatie*
    Gekeken wordt of over de implementaties al veel documentatie bestaat. Dit helpt een ontwikkelaar makkelijker aan de slag te kunnen met een implementatie. Wanneer weinig documentatie beschikbaar is moet de ontwikkelaar meer zelf bedenken, ook wanneer hij of zij tegen problemen aanloopt.

42 BV verwacht dat vrijwel alle apps in de toekomst door middel van een Java back-end worden aangestuurd. Bij deze onderzoeksvraag is dan ook uitgegaan van een Java back-end die in de volgende sub-paragraaf besproken wordt. In de daarop volgende sub-paragrafen wordt per app type de ontwikkelstraat toegelicht.

4.2.1 Back-end

Momenteel worden bij projecten van 42 BV de front en back-end van een systeem zo goed als gescheiden. Dit betekent het één en ander voor de systeemarchitectuur. Doordat hier later in het document nog uitgebreid op terug wordt gekomen, bij het beantwoorden van de tweede deelvraag, volgt nu alleen een overzicht van de hoofdpunten.

	Back-end systeem
Ontwikkeling framework	Spring
Programmeertaal	Java
Server	TomCat
Database	PostgreSQL
Connectie front en back-end	REST

Over het algemeen wordt PostgreSQL als database management systeem gebruikt. Toch spreek ik bij het behandelen van de specifieke app types ook over lokale dataopslag mogelijkheden. Dit heeft te maken met caching van bepaalde data. Wanneer er geen internet verbinding beschikbaar is kan besloten worden (mits mogelijk) de data te cachen in een lokale opslag, zodat wanneer een internetverbinding beschikbaar komt, de data met de server kan worden gesynchroniseerd.

4.2.2 Android

Het eerste app type dat wordt toegelicht is de Android app, waarbij specifiek wordt ingezoomd op Android als besturingssysteem. Naast dat Android een besturingssysteem is, is het ook een collectie van voor geïnstalleerde applicaties en een applicatie framework[21]. Dit framework uit zich in de virtual machine Dalvik (Dalvik VM), welke een grootte ondersteuning van diverse tools kent[22]. Doordat Android vandaag de dag door steeds meer hardware producenten wordt gebruikt als besturingssysteem voor hun apparatuur is dit het snelst groeiende besturingssysteem van het moment. Naast het gebruik in smartphones wordt Android ook gebruikt in onder andere tablets, media spelers en auto systemen.

Eén van de meest besproken onderwerpen bij het ontwikkelen voor Android is de versnippering over meerdere apparaten. Er bestaan zoveel verschillende apparaten dat ontwikkelaars zich afvragen of hun applicatie wel op al deze apparaten werken. Om deze versnippering tegen te gaan zijn alle apps, die gemaakt zijn voor eerdere versies van Android, forward compatible. Dit betekent dat een app geschreven voor Android 2.1 ook werkt op Android 4.0[23]. Wel kan het nog steeds apparaat afhankelijk zijn of bepaalde features van de smartphone wel of niet werken. Denk hierbij aan bijvoorbeeld Camera en flitser, GPS en de netwerkconnectie. Dit heeft te maken met de grote diversiteit aan fabrikanten die Android apparaten produceren, Android kan onmogelijk individuele ondersteuning voor al deze apparaten bieden. Het is dan vaak ook aan de fabrikant te taak hun producten compatible te maken het platform waar zij het apparaat voor maken. De meeste smartphone fabrikanten houden dit dan ook in het achterhoofd. Dat smartphone features niet werken komt dan ook in principe alleen voor wanneer de gebruiker zelf Android op een in dit geval niet Android smartphone zet. In de praktijk gebeurd dit overigens niet vaak en de gebruikers die zich hiermee bezig houden zijn op de hoogte van de risico's.

Android apps kunnen door middel van de Play Store worden gedistribueerd, de officiële distributiewijze van Google's platform. Daarnaast is het mogelijk de app zelf ter download aan te bieden op bijvoorbeeld een eigen server of website. Dit laatste houdt overigens wel in dat wanneer de gebruiker deze app wilt installeren hij wel een beveiligingsinstelling op de smartphone moet wijzigen.

Beperkingen

De Native programmeertaal van Android is Java. Niet alle Java libraries worden ondersteund door Android en er zijn veel platform specifieke APIs. Wanneer een ontwikkelaar wil beginnen met ontwikkelen van een Android app, moet de Android SDK worden gedownload. Deze SDK is voor zowel Windows, Mac OS X en Linux beschikbaar en bestaat uit diverse onderdelen. Zo beschikt het over de tools om de ontwikkelde app te builden, maar ook om deze te testen, te debuggen en te analyseren. Verder beschikt de SDK over Android NDK, een toolset die het mogelijk maakt in een andere Native taal (bijvoorbeeld C en C++) code te implementeren[24].

[21] Google-6
[22] Google-1
[23] Google-2
[24] Google-3

Bij het ontwikkelen wordt vaak een IDE (Integrated Development Environment) gebruikt. Voor Android zijn meerdere opties mogelijk, maar de meest gebruikte omgeving is Eclipse, die vrij toegankelijk is voor iedereen. Ontwikkelaars die op zoek zijn naar een completer pakket kunnen bijvoorbeeld voor het commerciële IntelliJ kiezen[25]. Deze IDE heeft een Android plug-in en kent een hogere mate van intelligentie op het gebied van code completion, code sharing en het delen van resources tussen projecten. Voor de Eclipse IDE zijn diverse Android Developer Tools beschikbaar, die bijvoorbeeld helpen bij het optimaliseren van de ontwikkelde apps. Een voorbeeld van een soortgelijk optimalisatie developer tool is Lint[26]. Deze developer tool helpt de app te optimaliseren door veel gemaakte fouten in de code aan te wijzen, welke vervolgens een kleine ingreep benodigen om geoptimaliseerd te worden.

Werking van het systeem[27]

Wanneer een app wordt geïnstalleerd en gestart op het Android systeem krijgt deze een eigen security sandbox toegewezen.

- Het Android besturingssysteem is gebaseerd op een multi-user Linux systeem, waarin elke applicatie een eigen user is.
- Standaard wijst het systeem een uniek Linux User ID aan een applicatie toe. Daarbij worden beperkingen met betrekking tot toegang aan alle bestanden behorende tot de applicatie opgelegd, zodat alleen de user ID van de applicatie de bestanden kan benaderen. De applicatie kent zijn eigen user ID niet.
- Elk proces op het Android systeem draait in zijn eigen Virtual Machine (VM). Een applicatie is dus geïsoleerd van andere applicaties.

Elke applicatie heeft standaard alleen toegang tot zijn eigen componenten. Dit creëert een veilige omgeving waarbij applicaties geen toegang hebben tot onderdelen van het systeem zelf. Daarbij stopt Android automatisch processen wanneer geheugen moet worden vrijgemaakt voor andere applicaties.

Ondanks dat het standaard dus niet mogelijk schijnt te zijn, zijn er wel manieren om data tussen applicaties te delen. Het is mogelijk één user ID aan twee applicaties te koppelen. Hierdoor kunnen gedeelde bestanden van applicaties éénmalig worden opgeslagen. Daarbij worden de applicaties door het delen van hetzelfde user ID in één VM en Linux proces gedraaid.

Applicaties verkrijgen toegang tot data als contacten, SMS en de SD kaart door middel van rechten (permissies) die de gebruiker aan de applicatie toekent. De aanvraag voor deze rechten gebeurd tijdens de installatie. Wanneer de rechten worden geweigerd, kan de applicatie niet worden geïnstalleerd.

[25] JetBrains-1
[26] Google-7
[27] Google-1

Opbouw van de applicatie

Applicaties binnen Android worden opgebouwd uit meerdere componenten, waarbij vier verschillende component types zijn te definiëren[28].

- ***Activities***
  Een Activity is een scherm met een user interface (UI). Als voorbeeld in de Android documentatie wordt gegeven dat een email applicatie een Activity voor het weergeven van nieuwe e-mails kan hebben, een Activity om een nieuwe email te maken en een derde Activity voor het lezen van een email. Hierbij zijn de Activities onafhankelijk van elkaar, zodat deze onafhankelijk van elkaar gestart kunnen worden.

- ***Services***
  Een Service is een component dat op de achtergrond draait. Dit component is verantwoordelijk voor de processen die voor langere tijd draaien, of voor het aanroepen van externe processen. Daarbij heeft de service component geen user interface. Het voorbeeld dat wordt gegeven is het afspelen van muziek op de achtergrond, terwijl de gebruiker een nieuwe email schrijft. Hierbij is het afspelen van muziek het werk van een Service, terwijl het schrijven van een nieuwe email een Activity betreft.

- ***Content Providers***
  De Content Provider is verantwoordelijk voor het delen van de applicatie data. Mogelijkheden van data opslag die standaard worden ondersteund zijn het bestandssysteem, een SQLite Database en natuurlijk het web (externe opslag). De Content Provider is verantwoordelijk voor het wijzigen, aanmaken, updaten en verwijderen van data. Ook is het mogelijk dat andere applicaties (mits toegestaan) door middel van dezelfde Content Provider deze acties uitvoeren. Het is mogelijk een applicatie alleen toegang te geven tot een specifiek deel van de Content Provider.

- ***Broadcast Receivers***
  De Broadcast Receivers reageren op broadcast meldingen van het systeem. Voorbeelden van een broadcast kunnen zijn dat het scherm uit is gegaan of dat de batterij bijna leeg is. De applicatie kan hier op reageren met een eigen gedefinieerde actie. Ook kan een applicatie zijn eigen broadcast melding genereren. Ondanks dat de Broadcast Receiver geen UI heeft, kan deze een status-bar notificatie weergeven.

[28] Google-4

Het grootte voordeel van deze aanpak, volgens de makers van Android, is dat op deze manier van werken een applicatie in principe altijd een component van een andere applicatie kan starten (mits deze rechten worden toegekend). Hierdoor is het mogelijk dat wanneer een applicatie bijvoorbeeld een foto moet kunnen nemen, de applicatie hiervoor het desbetreffende component van de camera applicatie aanroept, waarbij de foto wordt teruggegeven aan de applicatie. Hierdoor hoeft de ontwikkelaar zelf geen code voor het aansturen van de camera te schrijven. Voor de user ziet het eruit alsof het nemen van de foto onderdeel is van de applicatie.

Activeren van componenten

Het activeren van Activities, Services en Broadcast Receivers gebeurt door middel van asynchrone berichten, welke “Intents” worden genoemd. Doordat processen altijd in hun eigen secure VM draaien, kunnen componenten elkaar niet direct aanroepen. Deze Intents zijn dan ook bedoeld voor het Android systeem, welke de individuele componenten runtime aan elkaar koppelt.

Voor Activities en Services geldt dat de Intent een actie definieert welke uitgevoerd dient te worden en kan daarbij optioneel de data waarmee gewerkt dient te worden meegeven. Daarbij is het ook mogelijk een Intent te gebruiken om een Activity the starten waarbij een resultaat moet worden teruggestuurd (zoals bij de Intent om een foto te maken waarbij de foto bedoeld is voor gebruik in de eigen applicatie). De Broadcast Receivers daar in tegen geven alleen de melding die gebroadcast moet worden.

Het vierde component type, Content Provider, wordt niet door middel van Intents aangeroepen, maar wordt geactiveerd op aanvraag van de Content Resolver. De content resolver zorgt voor alle directe transacties met de Content Provider wat inhoud dat het component die de transacties uitvoert verder geen verantwoordelijkheid heeft over de afhandeling.

Het aanroepen van acties door middel van Intents kan worden gedaan door de componenten expliciet te noemen, maar het mooiste is natuurlijk als dit ook impliciet kan worden gedaan. Dit is mogelijk door de Intent Action te definiëren, waarin de ontwikkelaar aangeeft wat voor type actie uitgevoerd dient te worden, optioneel met data toegevoegd. Daarna gaat het systeem zelf op zoek naar een component op het apparaat die de actie uit kan voeren. Wanneer meerdere componenten dezelfde actie uit kunnen voeren krijgt de gebruiker de keuze welk hij of zij wil gebruiken. Het systeem zoekt naar andere componenten aan de hand van Manifests, welke nader worden toegelicht in de volgende paragraaf.

Manifest

Aan de basis van een Android applicatie staat het Manifest. Het Manifest, altijd genoemd AndroidManifest.xml, is het bestand waarin de ontwikkelaar alle componenten die deel zijn van de applicatie definieert. Deze component definitie wordt zoals eerder gezegd gebruikt tijdens het zoeken naar welke applicaties een bepaalde Intent uit kunnen voeren. Om dit geheel te optimaliseren kunnen aan deze component definities Intent-Filters worden toegevoegd, die aangeven wat een bepaald component wel en niet kan. Het Manifest staat altijd in de root folder van de applicatie.

Naast het definiëren van de componenten heeft het Manifest nog een aantal andere taken:

- Het definiëren van alle rechten die de applicatie nodig heeft.
- Het definiëren van de minimale versie van de APIs (de API Levels).
- Het definiëren van de hardware en software features waar de applicatie toegang toe moet hebben.
- Het definiëren van andere APIs die de applicatie nodig heeft, buiten de Android Framework APIs om.

Het is belangrijk dat de ontwikkelaar aangeeft aan welke eisen het apparaat moet voldoen voordat een applicatie geïnstalleerd wordt. Wanneer een applicatie gebruik maakt van de camera APIs (die zijn geïmplementeerd in Android 2.1), moet dit worden gedefinieerd in het Manifest. Wanneer dit wordt gedaan kan een apparaat zonder camera of een Android versie lager dan 2.1 de applicatie niet installeren. Wanneer het gebruik van de camera optioneel is kan tijdens runtime hier een check op worden uitgevoerd, zodat features die de camera vereisen worden uitgeschakeld.

Andere apparaat specifieke zaken waar rekening mee gehouden moet worden zijn scherm grootte, Apparaat features als BlueTooth en camera, Input configuraties (keyboard) en platform versies. Door het definiëren van deze zaken wordt automatisch bepaald voor welke apparaten de applicatie beschikbaar is in de Play Store.

Data storage

Er zijn binnen Android diverse manieren om data op te slaan. Zo is er volledige ondersteuning voor SQLite databases[29]. Elke klasse binnen de applicatie waarvoor de SQLite database is gedefinieerd heeft toegang tot deze database.

Daarnaast kunnen bestanden direct op het interne of externe geheugen geplaats worden. Bestanden die naar het interne geheugen worden geschreven zijn standaard van het type private en kunnen alleen door de specifieke app worden aangeroepen. Bij het deïnstalleren van de applicatie worden deze bestanden verwijderd. Bestanden op het externe geheugen (bijvoorbeeld SD kaart) zijn toegankelijk voor alle applicaties en de gebruiker (door middel van een USB aansluiting) en kunnen dus buiten de applicatie om gewijzigd of verwijderd worden.

Als laatste opslag methode zijn er de Shared Preferences. Dit framework kan alleen worden gebruikt voor het opslaan van primitieve datatypes (booleans, floats, ints, longs en strings) die in combinatie met elkaar een bepaalde instelling voorstellen. Naast de genoemde methodes is het uiteraard ook mogelijk data extern (niet op het apparaat) op te slaan. Dit gebeurt dan door middel van de netwerkconnectie.

Portabiliteit

Bij dit app type wordt gebruik gemaakt van de Native programmeertaal. Hierdoor gaat de portabiliteit van de app sterk achteruit. Een Android app wordt geschreven in de Java programmeertaal en het omzetten naar iOS kan hierdoor als heel vervelend worden ervaren. Dit komt doordat de app in principe in zijn geheel herschreven dient te worden. Dit wordt echter een ander verhaal wanneer de Android App voor een groot deel in de programmeertaal C wordt geschreven, wat wordt mogelijk gemaakt door de Android NDK. Gezien de iOS programmeertaal Objective-C een superset van de programmeertaal C is, werkt de code geschreven in C ook op iOS. Hierdoor wordt het omzetten van een Android app (geschreven in C) naar een iOS app al een stuk minder werk, omdat een groot deel van de code hergebruikt kan worden. Wel blijft het zo dat het programmeren van een Android app anders verloopt dan bij iOS. Zo zijn de behandelde componenten (Activities, Services, Content Provider en Broadcast Receiver) van toepassing bij Android, maar niet bij iOS. De manier van werken binnen iOS wordt later toegelicht.

[29] Google-6

Overzicht ontwikkelstraat

	Android	*Extra informatie*
Implementatie	Android SDK	De Android SDK is beschikbaar voor Windows, Mac OS X en Linux.
Ontwikkelomgeving	Eclipse	Eclipse is de meest gebruikte omgeving voor Android development, mede doordat deze ontwikkelomgeving vrij toegankelijk is.
	IntelliJ	Voor een completer pakket dan Eclipse kan gekozen worden voor IntelliJ, een commerciële IDE ontwikkeld door JetBrains. Deze IDE beschikt over meer functionaliteit dan Eclipse standaard te bieden heeft.
Programmeertaal	Java	Java is het Native programmeertaal van Android. Dit betekend echter niet dat alle Java libraries zullen werken op Android. Op de Android site is een complete lijst van packages te vinden[30].
	C/C++	Door middel van de Android NDK is het mogelijk code te implementeren die is geschreven in bijvoorbeeld C en C++.
Type User Interface	Native	Doordat het Android type app in de Native taal van het Android besturingssysteem wordt geschreven, heeft de ontwikkelaar toegang tot alle Native APIs, waaronder ook de User Interface API.
Event afhandeling	Sensor Manager	De Sensor Manager is binnen Android verantwoordelijk voor het doorgeven van User Events. Hierbij moet expliciet worden gelet op dat alle sensors die niet worden gebruikt door de applicatie uitgeschakeld zijn, doordat anders de batterij binnen enkele uren leeg is.
Distributiewijze	Play Store	De Play Store is de officiële Android store voor apps.
	Alternatieven	Naast de Play Store is het mogelijk de app door middel van een third-party medium te distribueren. Om deze app vervolgens te kunnen installeren moet de gebruiker wel een beveiligingsoptie uitschakelen. Het gebruik van de Play Store geeft de eindgebruiker makkelijk toegang tot de apps. Alternatieven zijn onder andere de Amazon App Store, Opera Mobile App Store en GetJar.

[30] Google-11

Toegang tot Native APIs		Onbeperkt	Doordat de app in de Native taal wordt geschreven is er onbeperkte toegang tot Native APIs. Hierdoor is het mogelijk alle apparaat sensoren en hardware functionaliteiten die het besturingssysteem en smartphone te bieden hebben optimaal te gebruiken.
Test platform & apparaten		Android smartphone	-
		Android Emulator	Door gebruik te maken van de Android Emulator, welke is bijgevoegd in de Android SDK, is het mogelijk een app te testen zonder gebruik te maken van een fysiek apparaat.
Test framework		Android test framework	De Android test suites zijn gebaseerd op JUnit. Klassen die niet een Android API aanroepen kunnen door middel van standaard JUnit worden getest. Andere klassen worden getest door middel van de Android JUnit extension.
		MonkeyRunner	Voor geautomatiseerd testen heeft Android ondersteuning voor MonkeyRunner. Deze API biedt de functionaliteit een programma te creëren die een Android apparaat of Emulator bestuurd, buiten de source code om.
Data opslag mogelijkheden		SQLite	-
		File System	Zowel voor intern als extern geheugen (SD kaart)
		Shared References	Deze opslag is alleen geschikt voor primitieve dataparen.
Portabiliteit		Slecht	Doordat Android apps in een Native taal worden geschreven dient de app herschreven te worden voor elk platform. Hierdoor is de portabiliteit samen met de iOS type apps het slechts van de app types.
Build opties		Ant	Standaard heeft Android ondersteuning voor Ant.
		Maven	Door middel van een plug-in kan build automation tool Maven worden gebruikt.
Community	*Tools*	Veel	-
	Omvang	Groot	-
	Activiteit	Veel	-
	Cultuur	Open	Android betreft een open source project, die onder de Apache Licentie valt. Hierdoor is het een open cultuur.
Maturity	*Leeftijd*	2007, november	Op 5 november 2007 onthulde de Open

			Handset Alliance hun eerste product, Android.
	Versie	4.1	-
	Issue management	-	-
	Documentatie	Veel	Doordat Android vandaag de dag het grootste marktaandeel van mobiele besturingssystemen in handen heeft, is de interesse in het product groot. Om aan deze interesse te voldoen is er door de jaren heen veel documentatie beschikbaar gemaakt.

4.2.3 iOS

Het tweede app type dat wordt toegelicht is iOS, waarbij specifiek wordt ingezoomd op iOS als besturingssysteem[31]. iOS is een besturingssysteem ontwikkeld door Apple voor gebruik op de iPhone, iPod Touch en iPad. Met de 200 miljoen apparaten wereldwijd is het is één van de populairste platformen vandaag de dag om apps voor te ontwikkelen. In tegenstelling tot het Android besturingssysteem heeft iOS maar een beperkt aantal apparaten waar het systeem op draait, waardoor er weinig sprake is van versnippering. Hierdoor is het ontwikkelen van apps voor meerdere (iOS) apparaten een stuk makkelijker dan in vergelijking met andere besturingssystemen.

Beperkingen

De Native programmeertaal voor het ontwikkelen voor iOS is Objective-C. Hiervoor dient gebruik te worden gemaakt van de iOS SDK. De toegang tot deze SDK is beperkt tot het ontwikkelen van een app. Voor het testen van de app op een fysiek apparaat en de daarbij behoren distributie via de App Store, dient de ontwikkelaar zich bij Apple te registreren als ontwikkelaar. Aan deze registratie zitten kosten verbonden, namelijk $99 per jaar. Door deze aanmelding krijgt de ontwikkelaar toegang tot de App Store en de mogelijkheid van testen op fysieke apparaten. Wanneer laatst genoemde niet gewenst is kan de ontwikkelaar ervoor kiezen zich niet aan te melden.

De iOS SDK bestaat uit diverse onderdelen, namelijk de Xcode IDE, de Interface Builder, Instruments (een tool om het uitvoeren van de app te monitoren) en een iOS simulator. De iOS SDK is alleen beschikbaar voor Mac OS X, het computer besturingssysteem van Apple. In de vroegere jaren van iOS was het gebruik van de SDK uitsluitend mogelijk door middel van de Xcode IDE, maar na versoepeling van de regels met betrekking tot de App Store is het ontwikkelen in third-party IDEs ook toegestaan. Binnen iOS is dit te zien in de game ontwikkeling, waarbij hedendaags veel gebruik wordt gemaakt van de Unreal Development Kit[32]. JetBrains (de maker van de bij Android app genoemde IntelliJ) heeft ook een IDE ontwikkeld speciaal voor iOS ontwikkeling, namelijk AppCode[33]. Deze IDE zit nog wel in een Beta fase en valt buiten een eventuele IntelliJ licentie.

[31] Apple-3,4,5, Enough-1, Encyclopedie-3
[32] Epic games-1
[33] Jetbrains-2

Implementatie

De iOS SDK bestaat uit een groot aantal aan APIs, onderverdeeld in frameworks en libraries. De belangrijkste zijn als volgt:

- *Cocoa Touch*
  Deze verzameling bestaat uit de User Interface (UI) libraries en de input mogelijkheden zoals multi-touch en de accelerometer.

- *Media Frameworks*
  Een groot aantal aan audio en video libraries, met daarnaast visualisering tools als OpenGL ES, Quartz en Core Animation.

- *Core Services*
  Services als netwerk, SQLite, camera en andere APIs.

De meeste apps worden gemaakt door middel van de Cocoa Touch APIs. Door het gebruik hiervan wordt de code in Objective-C geschreven. De UI wordt daarbij gemaakt door middel van de Interface Builder (die standaard toegankelijk is in Xcode).

Tijdens iOS ontwikkeling wordt gebruik gemaakt van View Controllers. De View Controllers zijn verantwoordelijk voor de inhoud die, door middel van de UI, aan de gebruiker wordt getoond. Standaard zijn er een groot aantal aan view controllers voor gedefinieerd, zoals die van de navigatiebar.

De View Controllers zijn te herkennen uit het Model-View-Controller principe. De base-class is de View controller die veel algemene functionaliteit mogelijk maakt. Voor sommige functionaliteiten wordt een deel van de oplossing geleverd door de base-class, maar wordt verder geïmplementeerd in een eigen View Controller. Een voorbeeld wat in de iOS documentatie wordt gegeven is als volgt:

Wanneer de gebruiker het apparaat roteert, probeert de standaard actie (gedefinieerd in de base-class) de UI te roteren. De View Controller echter kan bepalen of de UI wordt geroteerd en zo ja, hoe de configuratie van de Views in dat geval moet wijzigen.

Over het algemeen is de View Controller verantwoordelijk voor een deel van de user interface. De View Controller wordt daardoor verantwoordelijk voor interactie tussen meerdere views en data objecten. De user interface is opgebouwd uit meerdere View Controllers.

Testing

Binnen de iOS SDK is pas vanaf versie 2.2 een test framework geïntegreerd. Het gaat hier om OCUnit, een test framework gebaseerd op het SUnit framework[34]. Daarnaast zijn er een tal van frameworks ontwikkeld door derde partijen. Twee bekende hiervan zijn Keep it Functional (KIF) en TestFlight.

Keep it Functional (KIF) is een framework dat zich richt op automated testing. Hierbij wordt gebruik gemaakt van gesimuleerde gebruikers interactie en lijkt in dat opzicht erg op bijvoorbeeld Selenium. KIF is speciaal bedoeld voor iOS en wordt dan ook geschreven in Objective-C[35].

TestFlight is een framework wat het testen door echte gebruikers mogelijk maakt[36]. Voorafgaand aan het test proces kiest de ontwikkelaar een select aantal testers, waarna hij de TestFlight SDK implementeert in de app. Wanneer klaar voor testen, upload de ontwikkelaar de bèta app naar TestFlight, waarna TestFlight deze distribueert naar de geselecteerde testers. Vervolgens kan de tester door middel van voor gedefinieerde checkpoints in de app feedback geven. Zodra de tester een dergelijk checkpoint bereikt wordt hem of haar gevraagd feedback te geven (aan de hand van een door de ontwikkelaar vooraf gedefinieerd formulier). Ook worden zaken als crashes en performance gemonitord door TestFlight. Alle gegevens zijn vervolgens via de site of de desktop applicatie van TestFlight inzichtelijk.

[34] Sente
[35] Square
[36] TestFlight

Overzicht ontwikkelstraat

	iOS	*Extra informatie*
Implementatie	iOS SDK	De iOS SDK is alleen beschikbaar voor Mac OS X. Een computer met daarop Mac OS X is daarom ook vereist om voor iOS apps te kunnen ontwikkelen.
Ontwikkelomgeving	Xcode	De iOS SDK komt met een eigen IDE, namelijk Xcode. Deze IDE is vrij toegankelijk voor Mac OS X gebruikers en beperkt zich tot dit platform.
	AppCode/IntelliJ	Binnen de IntelliJ IDE is geen mogelijkheid tot ontwikkelen van Object-C projecten. JetBrains (de makers van IntelliJ), hebben wel een andere IDE ontwikkeld, speciaal bedoeld voor Objective-C, namelijk AppCode.
Programmeertaal	Objective-C	iOS apps worden geschreven in de Objective-C programmeertaal (de Native taal van iOS). Andere programmeertalen zijn tot dusver niet mogelijk, hoewel er wel geruchten zijn van een Java Virtual Machine.
Type User Interface	Native	Doordat het iOS type app in de Native taal van het iOS besturingssysteem wordt geschreven, heeft de ontwikkelaar toegang tot alle Native APIs, waaronder ook de User Interface API. Binnen iOS heet deze API Cocoa Touch.
Event afhandeling	Core Motion	Het Core Motion systeem framework is verantwoordelijk voor het waarnemen van motion data, om deze vervolgens door te sturen naar de applicaties.
Distributiewijze	App Store	iOS apps kunnen alleen via de App Store worden gedistribueerd. Apps verkregen buiten de App Store kunnen niet worden geïnstalleerd. Voor het distribueren via de App Store is een $99 kostende developer licentie vereist. Het gebruik van de App Store geeft de eindgebruiker makkelijk toegang tot de apps.
Toegang tot Native APIs	Onbeperkt	Doordat de app in de Native taal wordt geschreven is er onbeperkte toegang tot Native APIs. Hierdoor is het mogelijk alle apparaat sensoren en hardware functionaliteiten die het besturingssysteem en smartphone te bieden hebben optimaal te gebruiken.
Test platform & apparaten	iOS apparaat (iPhone, iPad, iPod Touch)	-

		iOS Simulator	De iOS SDK komt met een iOS Simulator.
Test framework		OCUnit	Het standaard SUnit gebaseerde testframework, geïntegreerd in Xcode.
		KIF	Keep it Functional is een automated test framework welke veel weg heeft van Selenium.
		TestFlight	TestFlight stelt de ontwikkelaar in staat op een gemakkelijke manier zijn app te laten testen door vooraf geselecteerde testers.
Lokale opslag mogelijkheden		Core Data	Core Data is een dataframework gebaseerd op het MVC design pattern. Standaard zitten in dit framework een datamodel designer (grafisch) en standaard CRUD functionaliteit geïntegreerd. Hierbij wordt gebruik gemaakt van een Build-in SQLite database.
		File System	-
		NSUserDefaults	De NSUserDefaults is de klasse waarin de standaard configuratie voor bepaalde applicaties of user Data wordt opgeslagen, dit wordt ook wel de Defaults System genoemd. Deze is vanuit de hele applicatie te benaderen.
Portabiliteit		Slecht	Doordat iOS apps in een Native taal worden geschreven dient de app herschreven te worden voor elk platform. Hierdoor is de portabiliteit samen met het Android type het slechtst van de app types.
Build opties		Xcode Build	iOS apps worden gecompileerd door middel van de Xcode Build tool. Dit gebeurd via de command-line.
Community	*Tools*	Veel	-
	Omvang	Groot	-
	Activiteit	Veel	-
	Cultuur	Gesloten	-
Maturity	*Leeftijd*	2007, juni	-
	Versie	6.0	Versie 6.0 van iOS verkeert momenteel in een Beta fase, maar betreedt rond september of november officieel de markt.
	Issue management	-	-
	Documentatie	Veel	Doordat de interesse naar het iOS besturingssysteem vanaf de release datum al groot is, is er door de jaren heen veel documentatie bij gekomen.

4.2.4 Web

In de web categorie zijn twee app types te vinden. Voor de ontwikkelaar zijn deze vrijwel identiek aan elkaar en worden in de praktijk vaak gecombineerd. Zoals eerder gezegd zullen deze twee types dan ook als één type worden behandeld, namelijk het Web type.

Een implementatie die voor dit app type gebruikt kan worden is Sencha. Sencha is een collectie van verschillende tools en frameworks, waaronder Sencha Architect, Sencha Touch en Sencha GXT. Sencha Architect is een HTML5 app builder die zowel de ontwikkelaar als de designer van een app de mogelijkheid geeft in één ontwikkelomgeving samen te werken. De Graphical User Interface (GUI) kan worden gemaakt door middel van een GUI-Builder. Daarnaast kan de ontwikkelaar zelf code implementeren om de afhandeling van gebruikers interactie in goede banen te leiden. Door middel van het Sencha GXT framework is het mogelijk in Java een app te schrijven, die vervolgens door gebruik te maken van de Google Web Toolkit wordt omgezet in HTML5 en JavaScript. Sencha Touch levert de ontwikkelaar een high-performance mobile HTML5 application framework.

Naast Sencha kan ook worden gekeken naar Kendo UI. Kendo UI is een framework wat HTML5 apps op Native apps probeert te laten lijken. Hierbij ligt de nadruk op probeert. Bij het Android platform zijn zoals eerder gezegd diverse smartphone fabrikanten actief. Deze smartphone fabrikanten produceren vaak een eigen User Interface om de smartphone ‘persoonlijker’ te maken. Kendo UI houdt echter alleen rekening met de Native Android UI, die UI die Google introduceert.

Distributiewijze

Web apps worden gedistribueerd via een webbrowser. Dit gebeurd op dezelfde manier als dat men dat doet via een browser op de computer. Dit betekend overigens ook dat de app alleen bereikbaar is wanneer een internet connectie beschikbaar is.

Native APIs

De Web app type heeft een zeer beperkte toegang tot Native APIs, maar door de komst van HTML5 wordt de toegang wel steeds minder beperkt. Door de komst van HTML5 is het ook mogelijk een site offline te gebruiken, maar zonder deze technologie is dit niet mogelijk. Daarbij laat ook de Performance het afweten ten opzichte van de Native apps. Bij het gebruik van populaire internet technologieën als HTML en CSS is ontzettend veel documentatie beschikbaar, waardoor de ontwikkeling sneller kan verlopen.

Portabiliteit

Dit app type is qua portabiliteit het beste in vergelijking met de overige app types. Doordat zo goed als elke browser de standaard web technologieën als HTML en CSS ondersteund (versie nummer achterwege latend) en vrijwel alle smartphones browser ondersteuning hebben, wordt een zo groot mogelijk publiek bereikt. Daarnaast zijn het niet alleen de smartphone gebruikers die op deze manier bereikt worden. Wanneer namelijk wordt gekeken naar andere media apparaten als mediaspelers en moderne tv’s, is steeds vaker te zien dat deze een internet aansluiting hebben. In combinatie met deze internet aansluiting is het voor de gebruiker mogelijk door middel van een webbrowser het internet te benaderen. Hierbij moet echter wel sterk rekening worden gehouden met de technologie ondersteuning

van de grote diversiteit aan browsers. Zo ondersteund Internet Explorer(IE), welke in mei 2012 nog 18.1% van het browser aandeel in handen heeft, pas sinds IE versie 10 een volledige ondersteuning voor alle HTML5 functies[37]. Dat terwijl bijna de volledige user-base (99.45%) van IE nog een lagere versie gebruikt. Dit komt overigens mede doordat IE versie 10 nog in de bèta fase zit. Dat IE versie 10 nog in de bèta fase zit betekend niet dat bij de officiële release iedereen gelijk overstapt. Van de eerder gestelde 18.1% gebruik namelijk 11.5% nog steeds een IE versie lager dan 9 (IE versie 9 is de laatst officieel uitgegeven versie)[38].

[37] HTML5Readiness
[38] W3Schools

Overzicht ontwikkelstraat

	Web	*Extra informatie*
Implementatie	Sencha	Een collectie van tools en frameworks om HTML5 development te vergemakkelijken.
	Kendo UI	Framework welke de Native look en feel nabootst.
	Vele andere	Er zijn ontzettend veel verschillende implementaties van het web mogelijk, teveel om op te noemen.
	Spring MVC en Tools	Binnen 42 BV wordt bij web ontwikkeling veel gebruik gemaakt van Spring MVC en bijbehorende tools. Hier wordt later nog op terug verwezen.
Ontwikkelomgeving	Favoriete editor	Het ontwikkelen van een Mobile website is niet gebonden aan een bepaalde editor, waarvan er tientallen zijn te vinden. Doordat de editor niet bindend is kan de ontwikkelaar zijn favoriet gebruiken.
	Eclipse	Door middel van plug-ins is ondersteuning voor JavaScript, HTML en CSS mogelijk binnen Eclipse.
	IntelliJ	IntelliJ heeft standaard ondersteuning voor JavaScript, HTML en CSS. Dit is ook in de vorm van bijvoorbeeld Code Completion.
	Sencha Architect	Sencha Architect is de IDE die onderdeel uitmaakt van het Sencha platform.
Programmeertaal	HTML/CSS/JavaScript	-
	Java	Door middel van de Google Web Toolkit kan een web app in Java worden geschreven, waarna hij naar JavaScript wordt gecompileerd.
Type User Interface	Web interface	Door het open karakter van web interfaces kan de designer/ontwikkelaar zijn eigen design implementeren.
	Native	Door middel van frameworks als Kendo UI is de Native look en feel mogelijk, maar alsnog beperkt. Gezien de hoeveelheid werk die dit met zich mee brengt is dit af te raden.
Event afhandeling	iOS: JavaScript API	Via de in iOS 4.2 geïntroduceerde JavaScript API is het mogelijk de data van de accelerometer en gyroscoop uit te lezen[39].
Distributiewijze	Browser	De Web app wordt gedistribueerd via het web en is daardoor alleen benaderbaar via een internet browser.

[39] Friese-3

Toegang tot Native APIs		Geen	Toegang tot Native APIs is bij de Web type app is niet mogelijk. Wel zijn er steeds meer mogelijkheden door de komt van HTML5. Hierbij kan onder andere gedacht worden aan geolocatie.
Test platform & apparaten		Apparaten met webbrowser	-
Test framework		Selenium	Selenium is een automated test framework. Dit framework automatiseert de browser door gebruikers interactie te simuleren.
		Jasmine	Een framework om JavaScript functioneel te testen.
Lokale opslag mogelijkheden		HTML5	Alleen door gebruik te maken van HTML5 is lokale opslag van grotere bestanden dan Cookies mogelijk.
Portabiliteit		Super goed	Doordat een groot aantal aan apparaten (tv's, smartphones, mediaspelers, ect cetera) tegenwoordig ondersteuning biedt voor web functionaliteiten is de portabiliteit van de Web app meer dan goed. Echter, wanneer gebruik wordt gemaakt van web technologieën als HTML5 en CSS3 moet wel rekening worden gehouden met de beperkte ondersteuning van de diverse browsers.
Build opties		Maven	De apps van het type Web kunnen gebuild worden door middel van build automation tool Maven. Hierbij is het ook mogelijk JavaScript compressie uit te voeren door middel van minify.
Community	Tools	Veel	Voor het web zijn de afgelopen jaren heel veel nieuwe tools uitgebracht. Deze variëren van nieuwe editors, tot ontwikkel frameworks en test tools.
	Omvang	Heel groot	Het web heeft zonder twijfel van alle app types de grootste community. Dit komt met name doordat het web al veel langer bestaat en (online) één van de meest gebruikte middelen is om informatie te delen in de wereld.
	Activiteit	Veel	De door ontwikkeling van het web gaat alsmaar door en door de komst van het nieuwe HTML5 en CSS3 verwachten veel mensen dat het door blijft groeien.
	Cultuur	Open	De community cultuur is open. De algehele web community is opgedeeld in kleinere communities. Hier zullen uiteraard ook

			gesloten communities tussen zitten.
Maturity	Leeftijd	1990	In 1990 was de ontwikkeling van het http protocol, de html taal, de eerste webbrowser, http server software en web server klaar.
	Versie	-	
	Issue Management	-	
	Documentatie	Veel	Over het web is heel veel documentatie beschikbaar. Ook over de programmeertalen voor ontwikkelmogelijkheden is veel te vinden.

4.2.5 Hybrid

De apps van het type Hybrid vallen qua ontwikkelstijl precies tussen de Native en Web app types. Zoals eerder beschreven wordt bij de ontwikkeling van het Hybrid app type zo minimaal mogelijk gebruik gemaakt van de Native taal. Hierbij wordt vaak gebruik gemaakt van web technologieën als HTML, CSS en JavaScript. Dit houdt in dat het omzetten van een applicatie naar andere platformen een stuk minder werk is, doordat vrijwel alle platformen deze web technologieën ondersteunen.

Eén van de bekendste platformen om Hybrid apps mee te maken is PhoneGap[40]. Bij gebruik van dit platform wordt de app ontwikkeld door middel van HTML en JavaScript[41]. De aangeraden ontwikkelomgeving is Adobe Dreamweaver, waarbinnen de gemaakte app gebuild kan worden. Het hoofddoel van PhoneGap is het de ontwikkelaar in staat stellen dat een HTML-gebaseerde app geïnstalleerd kan worden als Native app. Daarnaast geeft PhoneGap toegang tot diverse Native APIs die normaal gesproken niet beschikbaar zijn in een web applicatie.

Dit doel wordt behaald als volgt. Wanneer de ontwikkelaar zijn project build (de opties hiertoe komen later aan bod) resulteert dit in een aantal Native apps. Deze Native apps creëren een web view, waar vervolgens de HTML, JavaScript en CSS mee ingeladen kan worden. Doordat de Native app deze web view creëert is hij ook in staat met de JavaScript code in de web view te communiceren. In de context van de PhoneGap architectuur wordt dit de “Bridge” genoemd, welke verschillend per platform wordt geïmplementeerd. PhoneGap maakt gebruik van de communicatie mogelijkheid tussen de Native code en de web view door een JavaScript API te implementeren die het mogelijk maakt berichten terug te sturen naar de code.

Wanneer de ontwikkelaar zelf uitbreidingen aan PhoneGap wil maken moeten een aantal stappen worden ondernomen.

1. Het ontwikkelen van een JavaScript interface die gebruik maakt van de PhoneGap API om berichten te sturen naar de Native code.
2. De uitbreiding moet worden geregistreerd in het Native project (verschillend per platform).
3. Het ontwikkelen van de native code die de vraag van de JavaScript interface afhandelt.

[40] PhoneGap-1
[41] Whinnery

Build

Voor het build proces kan gebruik worden gemaakt van PhoneGap Build. Hiervoor is internet verbinding vereist, PhoneGap Build is namelijk gelokaliseerd in de Cloud. De ontwikkelaar upload zijn project naar PhoneGap build, welke vervolgens het project compileert naar zeven Native apps. Vervolgens krijgt de ontwikkelaar een aantal links toegestuurd. Door middel van deze links kunnen de apps worden gedownload voor verdere verwerking. Bij verdere verwerking kan onder andere gedacht worden aan de distributie naar de diverse app stores.

Doordat het build proces bij de PhoneGap Build methode geheel automatisch verloopt, is de hoeveelheid werk die vereist is om de app cross-platform te maken minimaal. Het nadeel van deze benadering is dat het uploaden en downloaden van het resultaat een handmatig proces betreft. Hierdoor is Continuous Integration op deze manier tot nog toe niet mogelijk, maar het staat wel als toekomstige feature op de planning[42].

Naast de PhoneGap Build methode is het ook mogelijk via de command line te builden. Dit gebeurd door handmatig shell scripts uit te voeren. Hierbij worden de verschillende shell scripts per doel platform aangeroepen[43]. Een medewerker van 42 BV heeft uit ervaring gezegd dat dit soort dingen over het algemeen zijn te automatiseren met Maven, waardoor Continuous Integration mogelijk is. Bij gebruik van deze methode dienen ook alle officiële SDK geïnstalleerd te worden op het systeem.

Performance

Een veel genoemd nadeel is dat dit app type op het Performance aspect vaak minder scoort dan het Native type app. Dit verschil is met name te merken bij het gebruik van Rich Media (interactieve media). Margus Pala heeft in januari een kleine test applicatie gemaakt waarmee hij het performance verschil tussen Native Android en een PhoneGap applicatie probeert te achterhalen[44]. Aan de resultaten is te zien dat in de test-case die Margus beschrijft, PhoneGap ongeveer drie keer zo langzaam is dan Native code. Het gaat hier echter wel alleen om het uitvoeren van een simpele for-loop. Wanneer verder gezocht wordt naar user experiences komt wel naar voren dat PhoneGap iets langzamer werkt, maar dat het niet als hinderlijk wordt bevonden.

Dit heeft vaak te maken met de kwaliteit van de web view en rendering engine van het doelplatform. Gezien elk platform verschillend is, verschilt de kwaliteit en performance van PhoneGap ook per platform. Zo stelt Kevin Whinnery dat iOS qua web view en rendering engine de beste kwaliteit weet te behalen[45]. Ook verwijst hij door naar een lijst met limieten waar de Android web view momenteel aan vast zit[46]. Daarbij is het simpel gezegd momenteel nog niet mogelijk met een web based app dezelfde performance en User Interface kwaliteit te behalen als dat kan met een Native app. Het is dan ook (nog) niet mogelijk met PhoneGap een dynamische Native UI te creëren.

---

[42] PhoneGap-2
[43] PhoneGap-3
[44] Pala
[45] Whinnery
[46] Donald

Supported Features

Het Hybrid app type is dus een goede keuze wanneer de ontwikkelaar op alle fronten een klein voordeel wil hebben. De portabiliteit van de Hybrid App is in vergelijking met de Web methodes minder, omdat er nog werk zit in het builden en distribueren van de apps. Daarbij moeten per platform waarvoor de app beschikbaar gesteld moet worden bepaalde plug-ins worden geconfigureerd. Deze methode komt, net als de overige Cross-Platform methodes, wel erg dicht in de buurt van de Native Experience omdat het gebruik van Native APIs mogelijk is. De volgende platform specifieke functionaliteiten zijn beschikbaar bij gebruik van PhoneGap:

	iOS iPhone / iPhone 3G	iOS iPhone 3GS and newer	Android	OS 5.x	OS 6.0+	WebOS	WP7	Symbian	Bada
ACCELEROMETER	✓	✓	✓	✓	✓	✓	✓	✓	✓
CAMERA	✓	✓	✓	✓	✓	✓	✓	✓	✓
COMPASS	✗	✓	✓	✗	✗	✗	✓	✗	✓
CONTACTS	✓	✓	✓	✓	✓	✗	✓	✓	✓
FILE	✓	✓	✓	✓	✓	✗	✓	✗	✗
GEOLOCATION	✓	✓	✓	✓	✓	✓	✓	✓	✓
MEDIA	✓	✓	✓	✗	✗	✗	✓	✗	✗
NETWORK	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (ALERT)	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (SOUND)	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (VIBRATION)	✓	✓	✓	✓	✓	✓	✓	✓	✓
STORAGE	✓	✓	✓	✓	✓	✓	✓	✓	✗

Bron – PhoneGap website

Overzicht ontwikkelstraat

	Hybrid	*Extra informatie*
Implementatie	PhoneGap	Platform gebaseerd op HTML5 en JavaScript. Na compilen is de app beschikbaar voor 7 verschillende platformen, waaronder iOS, Android, Windows Phone 7 en BlackBerry.
	Adobe Flash Builder (alternatief)	Software gebaseerd op de Eclipse ontwikkelomgeving waarbinnen door middel van ActionScript en het Flex framework software gemaakt kan worden.
Ontwikkelomgeving	Adobe Dreamweaver	Dreamweaver CS6 heeft sinds kort ondersteuning voor PhoneGap Build en is daardoor de aangeraden IDE om te gebruiken.
	Favoriete IDE	Doordat ontzettend veel ontwikkelomgevingen HTML en JavaScript ondersteunen kan de ontwikkelaar in veel van de gevallen zijn favoriete editor kiezen.
	Eclipse/IntelliJ	Beide IDEs kunnen worden gebruikt voor PhoneGap ontwikkeling.
Programmeertaal	HTML/JavaScript ActionScript	Afhankelijk van de gekozen implementatie wordt in de meeste gevallen gebruik gemaakt van bekende web technologieën als HTML en JavaScript.
Type User Interface	Web interface	Bij het maken van een Hybrid type app kan gekozen worden een eigen web interface te implementeren, zodat alle eigenschappen volledig naar eigen inzicht zijn aan te passen.
	Native	Een platform als PhoneGap beschikt niet over Native UI APIs voor bijvoorbeeld Android en iOS, waardoor het veel werk is om de Native feel te krijgen. Wel zijn er tools als jQueryMobile beschikbaar om toch een goede interface te kunnen maken.
	Kendo UI	De ontwikkelaar kan gebruik maken van het bij de Web app beschreven Kendo UI framework om de Native UI na te bootsen.
Event afhandeling	PhoneGap: Diverse APIs beschikbaar	Voor het uitlezen van de User Data zijn verschillende APIs beschikbaar, waaronder de Accelerometer API en Geolocatie API.
Distributiewijze	App Store / Play Store	Doordat Hybrid apps worden gewrapped in Native code kunnen zij worden gedistribueerd via de officiële app stores.
Toegang tot Native APIs	Afhankelijk van gebruikte API	Bij dit type app is de toegang tot Native APIs beperkt en altijd afhankelijk van het functioneel zwakste platform. Wel is er meer toegang te verkrijgen door het gebruik van community

			libraries.
Test platform & apparaten		Native Devices en beschikbare Simulators	Doordat het testen alleen mogelijk is na het compilen van de code, wat resulteert in een Native app type, kan dit alleen worden getest op een Native device of een eventueel beschikbare simulator (afhankelijk per Native platform).
Test framework		Selenium	Selenium is een automated test framework. Dit framework automatiseert de browser door gebruikers interactie te simuleren.
		Jasmine	Een framework om JavaScript functioneel te testen.
Lokale opslag mogelijkheden		SQLite	Binnen het PhoneGap framework wordt gebruik gemaakt van een W3C Web SQL Database, die op zijn beurt weer SQLite gebruikt.
Portabiliteit		Goed	Doordat bij het compilen de app beschikbaar wordt gesteld voor zeven verschillende platformen is de portabiliteit goed.
Build opties		PhoneGap Build	Wanneer de app is ontwikkeld, upload de ontwikkelaar deze naar PhoneGap Build, die vervolgens het geheel compiled. Na enkele minuten krijgt de ontwikkelaar een download link toegestuurd naar de diverse platformen.
		Command Line build	Ook is het mogelijk via de command line te builden. De ontwikkelaar moet dan echter wel handmatig voor de diverse platformen de build draaien. Dit is te automatiseren in Maven.
Community	Tools	Veel	Op de website van PhoneGap worden tientallen tools aangeraden[47]. Deze tools variëren van een JavaScript framework, Apple push notifications door middel van PHP & MySQL tot jQTouch. Daarnaast kan de community zelf plug-ins maken voor PhoneGap. Deze plug-ins moeten echter wel meer malen voor de diverse platformen gemaakt worden, wat ertoe leidt dat sommige plug-ins alleen voor bijvoorbeeld iOS of Android beschikbaar zijn.
	Omvang	Gemiddeld	De PhoneGap community is redelijk groot, maar in vergelijking met bijvoorbeeld Android en iOS is het een stuk minder groot.
	Activiteit		Naast dat er op internet veel over het ontwikkelen door middel van PhoneGap is te

[47] PhoneGap-4

			vinden, worden er ook regelmatig PhoneGap cursussen gegeven[48]. Hieruit blijkt dat PhoneGap toch een populair medium is om apps mee te maken. Ook komen er dagelijks steeds meer PhoneGap apps in de diverse stores terecht, doordat de beperkingen van het platform steeds kleiner worden.
	Cultuur	Open	Doordat PhoneGap een open source framework betreft is de cultuur erg open. Ontwikkelaars kunnen hun hierdoor eigen plug-ins ontwikkelen.
Maturity	Leeftijd	2008, augustus	Het idee voor PhoneGap ontstond op de iPhoneDevCamp in 2008. Na twee dagen lag er een concept versie klaar van de software.
	Versie	1.9	
	Issue Management	Apache Cordova	PhoneGap is sinds kort een distributie van het Apache Cordova project. Bug reports betreffende PhoneGap worden daardoor in de issue repository geplaats van het Cordova project[49]. Ondanks dat er nog issues van november 2011 openstaan, heb ik het idee dat deze bewust worden genegeerd of niet naar gekeken. Zeker gezien er op bijvoorbeeld 11 en 12 juli 2012 al meer dan 20 issues zijn gesloten.
	Documentatie	Voldoende	Op de website van PhoneGap is ook de PhoneGap documentatie te vinden. Naast dat door middel van code voorbeelden alle APIs worden uitgelegd, zijn er ook 'Getting started guides' te vinden, om de ontwikkelaar die nieuw is met PhoneGap op weg te helpen.

[48] Eduvision, Kassenaar
[49] Apache

4.2.6 Interpreted

Het Interpreted app type maakt gebruik van platform specifieke APIs, waaronder de User Interface elementen voor de interactie met de gebruiker. Hierdoor dient de ontwikkelaar enige kennis van de platformen te hebben waarvoor hij deze ontwikkeld. De overige platform specifieke zaken worden door middel van een cross-platform JavaScript-API afgehandeld. Een bezwaar wat hierbij door gebruikers wordt genoemd, is dat het ontwikkelen door middel van deze methode wel snel gaat, maar dat het verschil in werking van de gebruikte API op de diverse platformen zorgt voor instabiliteit en geheugenproblemen.

Een veel gebruikt platform binnen deze categorie is Appcelerator Titanium, een web technologie gebaseerd framework[50]. Tijdens de ontwikkeling van apps door middel van Appcelerator Titanium wordt gebruik gemaakt van JavaScript, waarbij het noodzakelijk is dat de ontwikkelaar de Titanium SDK leert. De Titanium SDK heeft veel weg van jQuery, maar de inhoud verschilt op bepaalde punten. Alternatieven aan Appcelerator Titanium zijn onder andere het Ruby gebaseerde RhoMobile en JMango. JMango heeft een eigen script taal. De apps worden ontwikkeld in de webbrowser, via de JMango Webportal. Deze ontwikkeling kan zowel grafisch, als door gebruik te maken van de eigen taal.

Appcelerator Titanium

Het doel van de Titanium SDK is op een hoog niveau een cross-platform JavaScript runtime API beschikbaar te maken, die ondersteuning biedt voor diverse Android en iOS APIs. Andere platformen (Windows en BlackBerry) staan op de planning om binnenkort ook ondersteund te worden. De Titanium SDK beschikt alleen over de meest gebruikte Native APIs, die ook nog eens te normaliseren zijn over de genoemde platformen. Wanneer een gebruiker een andere API wil gebruiken moet hier Native code voor ontwikkeld worden. Echter blijkt, in de praktijk hoeft de ontwikkelaar vrijwel geen Native code te schrijven.

De Titanium SDK zorgt dus voor een platform onafhankelijke manier voor het beschikbaar stellen van Native APIs. Tijdens runtime bestaat een Titanium app in principe uit drie delen:

- De eigen JavaScript source code.
- De platform specifieke implementatie van de Titanium API in de Native taal van dat platform.
- Een JavaScript interpreter, die de code tijdens runtime evalueert.

Bij de ontwikkeling wordt gebruik gemaakt van Titanium Studio, de Titanium IDE specifiek bedoeld voor de Titanium API. Deze IDE is gebaseerd op Eclipse.

Bij het starten van de applicatie wordt er in de Native code een JavaScript execution environment gecreëerd. Vervolgens wordt een “Proxy” object aangemaakt in de JavaScript omgeving, die vervolgens een paired object vormt met een proxy object in de Native code. Hierdoor ontstaan twee parallelle omgevingen waarin het proxy object aan de Native kant de eigenschappen weergeeft aan het proxy object aan de JavaScript kant.

[50] Appcelerator-1

In dit opzicht verschilt Titanium dan ook van bijvoorbeeld PhoneGap. Waar PhoneGap bij het compilen de JavaScript evalueert, doet Titanium dat tijdens runtime. Er wordt dan ook geen cross-compile gedaan naar de diverse Native talen[51].

Build

Binnen de Titanium Studio IDE is een wizard interface aanwezig waarmee de ontwikkelaar het project kan builden. Officieel gezien is het alleen mogelijk deze visuele interface te gebruiken, waardoor Continuous Integration niet mogelijk is. Wel is het mogelijk door middel van een modificatie via de command-line te builden. Dit wordt echter niet ondersteund door Appcelerator Titanium zelf[52].

Tijdens het build proces van een Titanium app wordt er een pre-compilatie gedaan op de JavaScript code[53]. Hieruit komt een symbool hiërarchie voort waaruit vervolgens kan worden afgeleid welke APIs de ontwikkelde app nodig heeft. Hierdoor wordt er runtime een parallelle omgeving gemaakt naar de Native taal. Op de achtergrond blijft altijd een Interpreter meedraaien, omdat anders de mogelijkheid tot dynamische objecten wegvalt.

Appcelerator Titanium is zoals gezegd alleen geschikt voor iOS en Android development[54]. Daarbij biedt Titanium nooit de volledige ondersteuning van alle platform APIs, doordat de APIs niet op beide platformen bestaan of doordat deze van elkaar verschillen.

[51] Whinnery
[52] Chapiewski
[53] Brogdon
[54] Appcelerator-2

Overzicht ontwikkelstraat

	Interpreted	*Extra informatie*
Implementatie	Appcelerator Titanium	Dit platform biedt een manier om door middel van web technologieën (met name Javascript) een app te ontwikkelen.
	RhoMobile	Een alternatief platform voor Appcelerator, die gebaseerd is op Ruby.
	JMango	Een mobile app framework waarbij de app wordt ontwikkeld door middel van de JMango web portal.
Ontwikkelomgeving	Titanium Studio	Bij het gebruik van Appcelerator wordt gebruik gemaakt van de Titanium Studio IDE welke specifiek is ingericht op de gebruikte Titanium API. Deze IDE is Eclipse gebaseerd.
	IntelliJ/Eclipse	Appcelerator's eigen IDE is officieel gezien de enig mogelijke IDE waarmee ontwikkeld kan worden, waardoor IntelliJ en Eclipse worden uitgesloten. In IntelliJ is door middel van een aantal stappen wel code completion voor de Titanium SDK te verkrijgen[55]. Daarbij is Titanium Studio Eclipse gebaseerd en is er een plug-in voor Eclipse beschikbaar.
Programmeertaal	JavaScript	-
	Ruby	Bij gebruik van RhoMobile wordt gebruik gemaakt van de Native taal Ruby.
Type User Interface	Native	Door gebruik te maken van de specifieke platform API (Appcelerator Titanium) worden Native APIs op een platform onafhankelijke manier benaderd.
Event afhandeling	Titanium APIs	Voor Appcelerator zijn er diverse Titanium APIs die gegevens van de sensoren op kunnen vangen en doorsturen.
Distributiewijze	App Store / Play Store	Doordat het toepassen van deze techniek resulteert in apps van het type Native, kunnen de apps worden gedistribueerd in de officiële stores.
Toegang tot Native APIs	Veel toegang	Appcelerator Titanium heeft toegang tot meer dan 5000 Native APIs.
Test platform & apparaten	Native devices	-
	Simulator	-
Test framework	Jasmine	Na een aantal kleine aanpassingen te maken, is het mogelijk door middel van Jasmine de Appcelerator app te testen[56].

[55] Peiris
[56] Morandi-1

Lokale opslag mogelijkheden		App Properties	Lijkt op de Shared Preferences binnen het Android platform. Een combinatie van een Eigenschap-Waarde worden door middel van App Properties opgeslagen.
		File System	-
		SQLite	-
Portabiliteit		Goed	Appcelerator Titanium biedt op het moment ondersteuning voor Android en iOS. Dit is momenteel voldoende voor 42 BV, maar kan in de toekomst nadelig zijn als zij zich meer willen gaan richten op andere mobiele platformen.
Build opties		Titanium Build	Titanium Build is de enige build methode die officieel wordt ondersteund. Mogelijkheden tot gebruik van de command line zijn er alleen in de vorm van modificaties en hacks.
Community	Tools	Beperkt	Voor de Appcelerator Titanium zijn maar een beperkt aantal plug-ins beschikbaar.
	Omvang	Gemiddeld	Een gemiddeld actieve community waarbij veel draait om een aantal personen.
	Activiteit		Appcelerator biedt zelf trainingen voor hun SDK aan. Deze trainingen vinden voornamelijk plaats in Amerika. In Nederland zijn een stuk minder, tot geen, trainingen beschikbaar.
	Cultuur	Open	Doordat Appcelerator Titanium een open source project betreft, heeft de SDK met bijbehorende community een open cultuur.
Maturity	Leeftijd	December 2008	Bij de release in 2008 was Appcelerator met name gericht op het ontwikkelen van desktop applicaties. In juni 2009 kreeg Appcelerator ondersteuning voor het ontwikkelen van iOS en Android based apps.
	Versie	2.1	-
	Issue Management	Appcelerator Jira	Ondanks dat er bij Appcelerator een stuk minder bug reports zijn dan bijvoorbeeld bij PhoneGap, wordt er heel actief aan de issues gewerkt. Zo dateert het oudste, openstaande issue, van 13 mei dit jaar. Daarbij moet wel gezegd worden dat er veel bug reports als “invalid” of “duplicate” worden afgehandeld. Een issue wordt als Invalid gezet als bijvoorbeeld geen reactie van de reporter komt.
	Documentatie	Voldoende	Op de Appcelerator site is ook de documentatie van de SDK te vinden. Naast deze API documentatie zijn er quick start guides die variëren van het installeren van de SDK tot voorbeeld applicaties.

4.2.7 Cross-Compiling

Het Cross-Compiling app type komt door het gebruik van apparaat en platform specifieke APIs het dichtst in de buurt van de Native Experience. Bij het Cross-Compile type wordt code in een universele taal geschreven, waarna dit in een zo genoemde generator wordt omgezet naar native code.

Het generator principe leg ik uit aan de hand van een bekend voorbeeld binnen deze app categorie, de Corona SDK. Vervolgens worden ook nog twee alternatieven aan de Corona SDK gegeven.

Corona SDK

De Corona SDK stelt de ontwikkelaar in staat door middel van een single-code base (gebaseerd op de Lua script taal) Native applicaties te genereren[57]. De ondersteuning is momenteel beperkt tot iOS, Android en eBooks (Kindle Fire). Door middel van 500 APIs krijgt de ontwikkelaar toegang tot apparaat mogelijkheden als camera, toetsenbord, GPS en de accelerometer. De SDK beschikt over de Corona Simulator waarmee kan worden gesimuleerd hoe de apps op diverse apparaten eruit komt te zien. Om toegang te krijgen tot de Corona SDK moet een licentie worden afgesloten welke $99/jaar kost.

Distributiewijze

Het resultaat van ontwikkeling door middel van een cross-compiler zijn een aantal apps van het type Native. Deze apps worden dan ook hetzelfde gedistribueerd als de apps van het type Native, namelijk via de App Store (iOS) en de Play Store (Android).

Native APIs

Verschillende platformen betekend over het algemeen verschillende APIs en functionaliteiten. Dat niet dezelfde APIs op alle platformen bestaan zorgt er voor dat het onmogelijk is een volledige ondersteuning te hebben van alle APIs op alle platformen. Wel komt steeds meer ondersteuning voor dit soort platformen, omdat momenteel de hele mobiele wereld tegen het probleem van multi-platform programmeren aanloopt. Door het ontwikkelen met behulp van een cross-compiler is de app portable, maar betekend dus wel beperktere toegang tot Native APIs. Doordat het hier een vrij nieuwe manier van ontwikkelen betreft, is er (nog) weinig documentatie beschikbaar.

Build

Via de Corona Simulator heeft de ontwikkelaar toegang tot de build opties van de SDK. Het betreft hier een handmatig proces van het kiezen van een build platform (iOS of Android), om vervolgens zaken als applicatie naam, versie en pad settings in te vullen. Gezien het build proces op de servers van Corona gebeurd is dit niet te bewerkstelligen door middel van de command-line[58].

[57] Corona
[58] Bochinski

MonoTouch

Naast cross-compilers als de Corona SDK zijn platformen beschikbaar die zich specifiek op bijvoorbeeld iOS development richten. Een voorbeeld hiervan is MonoTouch[59]. Dit platform geeft de ontwikkelaar de mogelijkheid om in de programmeertaal C# en .NET te ontwikkelen, wat het compileren naar bijvoorbeeld een Windows 7 Phone App vergemakkelijkt. Het is met dit platform nog steeds mogelijk gebruik te maken van de iOS APIs. Daarbij stellen de makers van MonoTouch dat door gebruik te maken van hun platform, veel code kan worden hergebruikt bij bijvoorbeeld Android ontwikkeling.

Applause

Ondanks dat de ontwikkelaars van Applause stellen dat hun framework niet binnen de categorie cross-compiler vallen, ben ik van mening dat deze daar zeker thuis hoort. Door middel van een eigen Domain Specific Language (DSL) wordt een app ontwikkeld die vervolgens wordt omgezet naar onder andere Objective-C, Java en C#[60].

[59] Xamarin
[60] Applause

Overzicht ontwikkelstraat

	Cross-Compiling	*Extra informatie*
Implementatie	Corona SDK	Een omgeving gebaseerd op de Lua scripttaal, waarmee de ontwikkelaar toegang heeft tot iets meer dan 500 Native APIs.
	Applause	Een framework gebaseerd op een custom DSL, waarmee code voor data-driven apps omgezet kan worden naar onder andere Java en Objective-C.
	MonoTouch	Een C# en .NET gebaseerde omgeving, met name gericht op iOS development. Echter, door gebruik te maken van dit platform stelt het bedrijf dat omzetten naar onder andere Android en Windows Phone een stuk makkelijker is.
Ontwikkelomgeving	Corona SDK	Voor de Corona SDK wordt gebruik gemaakt van een willekeurige tekst editor. Daarnaast is het mogelijk een Lua plug-in voor Eclipse te downloaden.
	Eclipse	Applause maakt hoofdzakelijk gebruik van Eclipse.
	Xcode	Ontwikkelen door middel van MonoTouch kan door middel van veel IDEs. Een veel gebruikte is de iOS IDE, Xcode.
	IntelliJ	Voor IntelliJ is een Lua plug-in beschikbaar[61].
Programmeertaal	Lua	De Corona SDK maakt gebruik van de Lua scripttaal.
	Custom DSL	Applause maakt gebruik van een custom DSL.
	C#/.NET	MonoTouch maakt gebruik van C# en .NET.
Type User Interface	Beperkt Native	Er is binnen de Corona SDK beperkte toegang tot Native User Interface componenten.
Event afhandeling	Event API	Binnen de Corona SDK worden door middel van de Event API de sensoren uitgelezen.
Distributiewijze	App Store / Play Store	De apps die door de compiler worden gegenereerd zijn van het type Native en dus te distribueren via de Native kanalen.
Toegang tot Native APIs	Beperkter	Door het cross-compile principe kunnen alleen Native functionaliteiten gebruikt worden die voor alle ondersteunde platformen beschikbaar zijn. Hierdoor is de toegang tot de volledige Native API base beperkt.
Test platform & apparaten	Native Devices	Doordat de ontwikkeling door middel van een cross-compiler resulteert in een Native app, kunnen deze apps worden getest op de Native apparaten.

[61] Sylvanaar

		Corona Simulator	Door middel van de Corona Simulator kan worden gekeken hoe de uiteindelijke applicatie eruit ziet op de diverse ondersteunde apparaten (alleen mogelijk in combinatie met de Corona SDK).
Test Framework		Lua-TestMore	Alhoewel niet standaard ondersteund binnen de Corona SDK, is het gebruikers gelukt Lua-TestMore voor de Corona SDK werkende te krijgen.
Lokale opslag mogelijkheden		SQLite	Binnen de Corona SDK kan gebruik worden gemaakt van de SQLite mogelijkheden van zowel iOS als van Android.
		File System	Bestanden kunnen lokaal worden opgeslagen in een Sandbox omgeving, wat betekend dat deze alleen toegankelijk zijn voor de applicatie. Hierbij kunnen geen mappen worden gemaakt.
Portabiliteit		Goed	De Corona SDK biedt op het moment ondersteuning voor Android, iOS en eReaders. Dit is momenteel voldoende voor 42 BV, maar kan in de toekomst nadelig zijn als zij zich meer willen gaan richten op andere mobiele platformen.
Build opties		Corona Build	De Corona SDK komt met build opties welke toegankelijk zijn via de Simulator. Dit betreft een handmatig proces en is niet te automatiseren.
Community	Tools	Voldoende	Vanuit de Corona SDK worden zestien third-party tools ondersteund. Deze tools variëren van een geavanceerdere IDE (Corona Cider), project managers (Corona Project Manager) tot Kwik, een PhotoShop plug-in waarmee de ontwikkelaar een interactief boek kan maken.
	Omvang	Klein	De Corona Labs heeft een kleine community met in totaal ongeveer 150.000 developers.
	Activiteit		Er worden Corona SDK cursussen aangeboden door Eduvision.
	Cultuur	Gesloten	Doordat toegang tot de Corona SDK een abonnement vereist, waarbij de source code beschermd blijft, is de cultuur van deze community gesloten. Er zijn dan wel een aantal third-party tools ontwikkeld, maar dit is volgens mij in afspraak met Corona Labs. Hier heb ik echter geen bewijs voor gevonden en kan daardoor ook als onjuist worden gezien.
Maturity	Leeftijd	2009, juli	-
	Versie	2012.840	-

	Issue Management		Bugs in de Corona SDK kunnen wel worden gemeld, maar een lijst van eerder gemelde bugs is niet te vinden.
	Documentatie	Voldoende	Naast dat de Corona website alle API documentatie bevat, zijn er ook genoeg andere documentatie sites te vinden over de Corona SDK. Een voorbeeld hier van is de Learning Corona website, onderhouden door David Papandrew[62]. Deze site staat vol met tutorials die een ontwikkelaar kan volgen om de Corona SDK in gebruik te nemen.

[62] Papandrew

4.3 Tegen welke overige problemen lopen ontwikkelaars aan?

Een aantal problemen waar andere ontwikkelaars tegenaan lopen zijn al naar voren gekomen tijdens het beantwoorden van de vorige twee onderzoeksvragen. Zo is het grootste obstakel, het ontwikkelen van een app voor meerdere platformen, al ruim aan bod gekomen. Om de huidige onderzoeksvraag te beantwoorden ben ik op zoek gegaan naar problemen, die nog niet duidelijk naar voren zijn gekomen. Dit bleek nog niet zo gemakkelijk als ik vooraf had gedacht. Ontwikkelaars lopen tegen genoeg problemen aan, de fora staan er vol van, maar alle problemen hebben ook een oplossing. De ontwikkelaars lopen tegen problemen aan doordat zij de kennis niet hebben om verder te komen, wat er dus toe leidt dat zij de vraag op fora stellen. Dit zorgt voor individuele vragen, die beantwoord worden door andere ontwikkelaars die wel de desbetreffende kennis hebben. Dit worden in een andere term ook wel 'Incidenten' genoemd. Hierdoor zijn vaak vrij simpele oplossingen mogelijk en was het vinden van een echt probleem lastig.

Een wel veel voorkomend probleem, wat bij het beschrijven van de Native Experience al naar voren is gekomen, is dat de app die ontwikkelaars ontwikkelen soms wordt geweigerd in bepaalde stores. Dit gebeurd met name in Apple's App Store, waar behoorlijk wat beperkingen aan zijn verbonden. Dit kan als voordeel of nadeel worden gezien. Wat op internet veel genoemd wordt is dat iOS apps er vaak beter eruit zien en van betere kwaliteit zijn. Dit heb ik vaak langs zien komen, zo ook in een blog post op Appstorm, geschreven door Connor Turnbull[63]. Zo zou de user interface bij iOS applicaties van betere kwaliteit zijn, door onder andere betere grafische kwaliteit. Een ander punt wat wordt aangehaald door Connor is het feit dat iOS apps naar zijn idee vaak in een eerder stadium worden gemaakt. Regelmatig kom je op het internet langs winkels of websites die een app voor iOS aanbieden, maar nog niet voor Android. Hier zit een kern van waarheid in, die naar mijn mening logisch is te verklaren. Apple stelt bepaalde eisen aan de apps die op hun platform worden uitgebracht, waaronder eisen aan de kwaliteit van de user interface. Omdat het Android platform een stuk minder van deze beperkingen heeft zullen ontwikkelaars bewust, of onbewust, vaker met de regels van het iOS systeem in hun achterhoofd een applicatie ontwikkelen. Doordat de app dus eigenlijk gespecifieerd wordt voor het iOS systeem en daar dus het meeste werk in zit, is de kans groot de app beter werkt op dit platform en misschien ook eerder uitkomt op dat platform. Dit is een groot voordeel voor iOS gebruikers, maar een nadeel voor de ontwikkelaars. Doordat ontwikkelaars vast zitten aan strenge eisen die iOS aan apps stelt, zijn er veel verhalen te vinden over apps die in eerste instantie worden geweigerd in de app store.

Een ander probleem wat veel wordt genoemd, met name bij de cross-platform opties, is dat er na mate de ontwikkelde app complexer wordt, steeds vreemdere bugs opduiken. Zo bevestigd ook Andrea Dallera op zijn blog[64]. De bugs die worden vermeld variëren van kleine bugs als "De titel in de navigatiebar centreert niet meer" en "de focus actie op het scherm wordt ontzettend vaak aangeroepen" tot grote bugs als "Bij het starten van de app, crasht het op willekeurige momenten gelijk". In deze blog, die gebaseerd is op de bij interpreted apps methode genoemde Appcelerator, wordt de schuld gegeven aan het ingebouwde geheugen management. Ondanks dat deze

[63] Connor-1
[64] Dallera

geheugenmanager meestal goed functioneert zijn er meldingen van geheugen lekken uit kleine, Appcelerator gecreëerde apps. Hierbij heeft de gebruiker in eerste instantie niet door dat het vollopen van het geheugen door dit framework komt. In reactie op deze blog (die dateert uit 2011) heeft een ontwikkelaar van Appcelerator gezegd dat dit geheugen management probleem is verholpen in de 1.7 versie van het programma. Echter, meerdere reacties daarop ontkrachten de oplossing en stellen dat dit probleem nog steeds bestaat.

4.4 Conclusie

De conclusie van deze deelvraag komt in de vorm van een overzicht van de verschillende app types. De tabel op de volgende pagina geeft het overzicht van de in voorgaande paragrafen besproken onderwerpen weer. Hierbij komen geen user stories aan bod.

	Android	*iOS*	*Web*	*Hybrid*	*Interpreted*	*Cross-Compiling*
Implementatie	Android SDK	iOS SDK	Sencha Kendo UI Nog veel meer	PhoneGap	Appcelerator	Corona SDK
Ontwikkelomgeving	Eclipse IntelliJ	Xcode AppCode	Favoriete Editor Eclipse IntelliJ Sencha Architect	Favoriete IDE Dreamweaver Eclipse IntelliJ	Titanium Studio Eclipse IntelliJ	Corona SDK
Programmeertaal	Java C, C++ (beperkt)	Objective-C	HTML CSS JavaScript Java (GWT)	HTML JavaScript	JavaScript	Lua
User Interface	Native	Native	Web Interface Native (lastig)	Web-Interface Kendo UI	Native	Native
Event afhandeling	Sensor Manager	Core Motion	JavaScript API (alleen iOS)	Event APIs	Titanium Event APIs	Corona Event APIs
Distributiewijze	Play Store Andere mediums	App Store	Browser	App Store Play Store	App Store Play Store	App Store Play Store
Toegang tot Native APIs	Onbeperkt	Onbeperkt	Geen	Veel toegang	Veel toegang	Beperkt
Test platform & Apparaten	Android smartphone Android Emulator	iOS apparaat iOS Simulator	Apparaten met een webbrowser	Native devices Beschikbare Simulators	Native devices Beschikbare Simulators	Native devices Beschikbare Simulatoren
Test Framework	Android test framework MonkeyRunner	OCUnit KIF TestFlight	Selenium Jasmine	Selenium Jasmine	Jasmine	In Simulator
Data Opslag	SQLite File System Shared Preferences	Core Data File System NSUserDefaults	HTML5	SQLite	App Properties File System SQLite	SQLite File System
Portabiliteit	Slecht	Slecht	Super Goed	Goed	Goed	Goed
Build	Ant	Xcode Build	Maven	PhoneGap Build Command-line	Titanium Build	Corona Build

5 Deelvraag twee – Ontwikkelstraat

De tweede deelvraag betreft het in kaart brengen van de huidige ontwikkelsituatie, met het oog op de verschillende app types. De deelvraag is als volgt geformuleerd:

> Hoe ziet de huidige ontwikkelstraat van 42 BV eruit en welke beoordeling kan, in vergelijking daarmee, per app type worden gegeven.

Ook bij deze deelvraag zijn een aantal onderzoeksvragen gespecificeerd. Deze onderzoeksvragen worden individueel behandeld in de volgende sub-paragrafen.

5.1 Hoe ziet de huidige ontwikkelsituatie van 42 BV eruit?

Binnen 42 BV worden een aantal frameworks, applicaties en tools gebruikt om de ontwikkelstraat in te richten. Onderstaand in de tabel zijn de tools te vinden die het meest worden gebruikt.

Functie	*Gebruikte tool*
Programmeertaal	Java, JavaScript, HTML en CSS
Ontwikkelomgeving	Eclipse, IntelliJ
Database Management	PostgreSQL, Oracle DB, HSQL
Test frameworks	JUnit, Mockito, JMockit, Selenium, Jasmine
Kwaliteitswaarborging	Sonar
Automated build tool	Maven
Continuous Integration	Bamboo
Samenwerkingsomgeving	Confluence

Deze tools worden regelmatig toegepast binnen de ontwikkelstraat van 42 BV. De standaard ontwikkeltaal voor projecten van 42 BV is Java, maar in sommige situaties wordt hiervan afgeweken. Een voorbeeld van zo een afwijking is bij het project Goudvink. Bij dit project heeft de te maken applicatie naast een Java back-end systeem een web applicatie die wordt gebruikt voor interactie met de gebruiker. Bij dit project is dan ook gebruik gemaakt van een aantal web technologieën, te weten JavaScript, HTML en CSS. Ook bij andere projecten binnen 42 BV worden deze webtechnologieën regelmatig toegepast.

Dergelijke projecten bestaan vaak uit een front-end en back-end systeem. Om deze twee delen van het systeem aan elkaar te koppelen wordt gebruik gemaakt van de Representational State Transfer (REST) architectuur stijl, welke zit geïntegreerd in Spring MVC Framework. Dit framework, wat vaak wordt gerefereerd aan web services, is gebaseerd op het Model-View-Control design pattern[65]. REST focust op het adresseren en gebruiken van applicatie resources, net als bijvoorbeeld het bekendere SOAP doet. Hierbij sluit het dicht aan op het http-protocol, maar in tegenstelling tot SOAP creëert REST geen extra

[65] Spring

tussenlaag[66]. Daarbij maakt REST gebruik van JSON (JavaScript Object Notation), in plaats van XML die wordt gebruikt bij SOAP. Het voordeel hiervan is dat JSON kleiner van omvang is (snelheid) en daarbij bedoeld is voor gebruik binnen JavaScript, een web technologie waar het bedrijf gebruik van maakt (compatibiliteit).

Vrijwel alle applicaties hebben te maken met data verwerking, waarbij de data vaak wordt opgeslagen in een Database Management Systeem. Bij 42 BV wordt gebruik gemaakt van PostgreSQL en Oracle DB. Tijdens de ontwikkeling van een applicatie wordt veelal gebruik gemaakt van HSQL, wat een in-memory database betreft. Om te testen is dit sneller dan een losse database. Om de data vanuit de applicatie in de database te verwerken wordt gebruik gemaakt van een connectie tussen deze twee systemen. Veelal wordt deze connectie geregeld door middel van de Java Persistence API (JPA), met als specifieke implementatie Hibernate. Dit framework wordt gebruikt om data uit de relationele database te managen binnen Java platformen[67]. Doordat JPA het managen van dit soort data sterk vereenvoudigd wordt het veel gebruikt binnen 42 BV.

42 BV staat in voor de kwaliteit van de software die zij ontwikkelen. Hierdoor is testen van de ontwikkelde software uitermate belangrijk. Voor dit testen worden diverse methoden en technieken gebruikt. De drie meest gebruikte zijn JUnit, Mockito en JMockit. JUnit is een unit test framework welke officieel gezien voor de Java programmeertaal werd ontwikkeld. Echter zijn er steeds meer zo genoemde Ports gekomen, waarbij JUnit voor andere programmeertalen (waaronder C#, C++ en JavaScript) is omgezet. In aanvulling op het JUnit Framework wordt gebruik gemaakt van Mockito, een mock framework wat veel wordt gebruikt bij het testen. Bij Mockito wordt gebruik gemaakt van mock objecten, welke het gedrag van een echt object simuleren[68]. Een mock wordt gecreëerd wanneer een ander object getest wordt. Hierdoor zijn er binnen het test gedeelte geen afhankelijkheden tussen de verschillende objecten. Wel wordt de koppeling tussen de test-code en de applicatie code hierdoor een stuk sterker. Aansluitend hierop wordt soms gebruik gemaakt van JMockit, een mocking API die de ontwikkelaar in staat stelt met minimale test code, gestructureerde en leesbare tests te schrijven[69]. Doordat het testen van deze code alleen aan de back-end kant van het systeem gebeurt, wordt voor de front-end een ander platform gebruikt, te weten Selenium. Deze browser-automation tool kan automatisch door een web applicatie heen ‘klikken’ om op die manier fouten op te sporen[70].

Naast de kwaliteitswaarborging door middel van tests wordt ook het Sonar platform gebruikt voor verbetering van de code. Dit wordt gedaan door middel van een code drill-down, waarbij wordt gecontroleerd of programmeer regels op de juiste manier worden toegepast. Dit is ook mogelijk met een zelf ingevoerde Code-Conventie. Uiteindelijk krijgt de ontwikkelaar een overzicht van statistieken te zien over specifiek zijn project, of meerdere projecten.

[66] Bakker, Holthe, IBM
[67] Oracle
[68] Mockito
[69] JMockito
[70] Selenium

Het ontwikkelen van software zonder gebruik te maken van Integrated Development Environments (IDEs) is tegenwoordig bijna ondenkbaar. De IDEs maken het ontwikkelen makkelijker door functionaliteiten als code completion, build automation tools en debug opties te bieden. Binnen 42 BV wordt gebruik gemaakt van Eclipse (met als specifieke variant Eclipse STS, welke specifiek bedoeld is voor het Spring framework) en IntelliJ. Door middel van plug-ins kunnen gebruikte tools als Jira, Spring en Maven voor een groot deel worden geïntegreerd in de Eclipse ontwikkelomgeving. In de Eclipse STS (Spring Source Tool Suit) is de Spring plug-in standaard aanwezig. IntelliJ is een commerciële IDE en ondersteund deze soortgelijke functionaliteiten direct bij installatie[71].

Een andere plug-in wat is genoemd bij de Eclipse IDE is Maven. Maven is een build automation tool, die onder andere het build proces van applicaties automatiseert. Aansluitend daarop wordt ook gebruik gemaakt van Bamboo, een Continuous Integration programma. Dit programma bestaat uit functionaliteiten die zich focussen op het proces wat wordt doorlopen bij een release naar de productieomgeving[72]. Om dit mogelijk te maken wordt gebruik gemaakt van onder andere Maven (voor het build en test proces) en TomCat (voor het deploy proces). Standaard is Bamboo voorzien van een aantal opties. Daarnaast wordt Bamboo ook gebruikt in combinatie met de eerder beschreven Sonar, zodat de ontwikkelaar na het maken van een wijziging gelijk aan de slag kan met het optimaliseren van de code. Ook kan Bamboo aan de productieomgeving worden gekoppeld waardoor de ontwikkelaar vanuit Bamboo een release kan doen.

Om de documentatie, die wordt gemaakt tijdens de werkzaamheden, bij elkaar te houden maakt 42 BV gebruik van nog een ander Atlassian product, te weten Confluence. In deze omgeving kan iedere werknemer nieuwe pagina's aanmaken met informatie die hij of zij heeft verkregen (een soort van Wiki pagina). Naast project documentatie wordt deze omgeving ook gebruikt voor bijvoorbeeld het plannen en kenbaar maken van events.

Als laatste veel besproken onderwerp binnen 42 BV noem ik security. Twee belangrijke tools die worden gebruikt ter verbetering van de security zijn Spring Security en Atlassian's Crowd. Spring Security is een authenticatie framework en is onderdeel van Spring[73]. Crowd betreft een gecentraliseerd inlog systeem, wat al binnen diverse interne projecten van 42 BV wordt gebruikt (waaronder de Atlassian producten en Goudvink).

[71] JetBrains-1
[72] Atlassian-2
[73] Spring-1

5.2 In welke mate zijn aspecten en gebruikte technologieën uit de huidige ontwikkelstraat terug te vinden bij de diverse app types?

Aan de hand van deze onderzoeksvraag wordt bepaald welke ontwikkelstraten van de app types lijken op de huidige ontwikkelstraat van 42 BV.

Bij de tweede onderzoeksvraag van de eerste deelvraag zijn een aantal aspecten genoemd die voor 42 BV belangrijk zijn om meer over te weten. Vervolgens is in de vorige onderzoeksvraag beantwoord hoe de huidige ontwikkelsituatie van 42 BV eruit ziet. Bij beantwoording van deze onderzoeksvraag wordt een waardering gegeven aan de diverse aspecten uit zowel de huidige ontwikkelstraat als die van de app types. Daarbij kan per aspect een aantal punten behaald worden (de waardering), waarbij per app type in totaal 100 punten gehaald kunnen worden. Het per aspect te behalen punten zijn in samenspraak met 42 BV vastgesteld en zijn gebaseerd op hoe belangrijk dat aspect voor het bedrijf is. Hoe hoger de totale score van een specifiek app type is, hoe meer waarde en gelijkenissen dit app type voor 42 BV heeft. De beoordeelde aspecten en bijbehorende punten zijn als volgt:

	Beoordeelde aspecten	*Aantal te behalen punten*
Ontwikkelstraat (50)	Ontwikkelomgeving	10
	Programmeertaal	10
	Testmogelijkheden	10
	Distributiewijze	4
	Automated Build Tool	10
	Sonar	6
Community (8)	Tools	4
	Omvang	2
	Cultuur	2
Maturity (12)	Leeftijd/versie	6
	Issue management	3
	Documentatie	3
Overige baten (30)	Native APIs	9
	User Interface	9
	Offline mogelijkheden	3
	Portabiliteit	9
Totaal		100

5.2.1 Puntverdeling

De punten worden per aspect bepaald aan de hand van een aantal criteria. Aan deze criteria wordt een percentage gekoppeld, welke de mate weergeeft waarin het besproken aspect kan worden gereflecteerd op de huidige ontwikkelstraat van 42 BV. Doordat niet elk aspect is te reflecteren op de huidige ontwikkelstraat wijkt de volgorde van aspecten af ten opzichte van voorgaande paragrafen. Per aspect zijn de criteria, met bijbehorend percentage, als volgt:

- *Ontwikkelomgeving*

Score	*Criteria*
0%	Geen ontwikkelomgeving beschikbaar. Ontwikkelen vind plaats door gebruik te maken van een normale teksteditor.
30%	Een ontwikkelomgeving is beschikbaar, maar deze is nog (vrijwel)onbekend voor 42 BV (Xcode, Titanium Studio, Adobe Dreamweaver, Corona SDK).
60%	Bij ontwikkeling van iOS kan gebruik worden gemaakt van de AppCode ontwikkelomgeving. Dit betreft een ontwikkelomgeving die is ontwikkeld door dezelfde makers als van IntelliJ. Ik verwacht dat deze ontwikkelomgevingen veel gelijkenissen hebben.
80%	Ofwel Eclipse, ofwel IntelliJ, is beschikbaar voor het ontwikkelen van dit app type.
100%	Zowel IntelliJ als de Eclipse ontwikkelomgeving zijn beschikbaar voor ontwikkeling van het app type.

- *Programmeertaal*

Score	*Criteria*
0%	Binnen het bedrijf is helemaal geen ervaring met de programmeertaal (ActionScript, Lua, .NET).
30%	Enkele medewerkers binnen het bedrijf hebben ervaring met de programmeertaal (Objective C, C#, C++, Ruby).
70%	De programmeertaal is al vaker gebruikt voor projecten van 42 BV (HTML, CSS, JavaScript).
100%	De programmeertaal wordt in de huidige situatie altijd toegepast (Java).

- *Testmogelijkheden*

Score	*Criteria*
0%	Voor de gebruikte programmeertaal zijn geen geautomatiseerde test frameworks beschikbaar.
30%	Voor de gebruikte programmeertaal zijn wel testmogelijkheden beschikbaar, maar deze zijn binnen 42 BV niet bekend.
70%	Voor de programmeertaal zijn testmogelijkheden beschikbaar en 42 BV heeft hier (beperkte) ervaring mee.
100%	Om de code te testen kan gebruik worden gemaakt van een voor 42 BV bekend test framework (JUnit, Mockito, JMockit).

- *Distributiewijze*

Score	*Criteria*
0%	De distributiewijze is volledig onbekend.
40%	De app wordt gedistribueerd via de officiële Native kanalen. 42 BV is niet bekend met deze distributiewijze, maar de aanwezigheid van genoeg documentatie vergemakkelijkt de distributie.
100%	De app kan door middel van een .war bestand worden uitgerold als web applicatie. Dit is de meest gebruikte methode binnen 42 BV.

- *Automated build tool*

Score	*Criteria*
0%	Het build proces verloopt via een grafische interface.
30%	Het build proces verloopt officieel gezien via een grafische interface, maar door middel van een modificatie in de software is het mogelijk dit via de command-line te doen.
70%	Automatisch builden is mogelijk, maar gebeurt door middel van een voor 42 BV onbekende methode.
100%	Automatisch builden is onbeperkt mogelijk en wordt officieel ondersteund.

- *Sonar*

Score	*Criteria*
0%	De gebruikte programmeertaal kent geen ondersteuning binnen Sonar.
50%	Een aantal van de gebruikte talen, of een deel daarvan, wordt ondersteund door Sonar (HTML, CSS, JavaScript).
100%	De gebruikte programmeertaal kent ondersteuning in Sonar.

- *Community*

Het bepalen van een community score verloopt anders dan dat van voorgaande aspecten. Dit komt mede doordat het verkrijgen van informatie over bijvoorbeeld de omvang van een community een erg onnauwkeurig werk betreft. Dit geldt ook voor het aantal tools die beschikbaar zijn voor een bepaald app type ontwikkeling. Door het ontbreken van een overzicht van (officiële) plug-ins en de fragmentatie op het internet, kan er nooit met zekerheid een feitelijk aantal worden vastgesteld. Dit komt met name doordat een community niet altijd op een enkele plek samen komt, waardoor de verdeling over (in dit geval) het internet groot is. De community score is dan ook vooral gebaseerd op een ervaring, die ik tijdens het onderzoek naar de diverse app types heb opgedaan.

Net als bij de beantwoording van de tweede deelvraag wordt het aspect community opgedeeld in sub-aspecten. Activiteit achterwegenlatent zullen drie sub-aspecten worden gedefinieerd, de beschikbaarheid van tools, de omvang van de community en de cultuur.

Sub-aspect	*Percentage*	*Score*	*Criteria*
Tools	40%	0%	Er zijn geen tools beschikbaar.
		60%	Er is een beperkt aantal tools beschikbaar.
		100%	Er zijn voldoende tot veel tools beschikbaar.
Omvang	30%	0%	Geen actieve community.
		40%	Kleine actieve community, waarbij veel draait om een aantal personen.
		70%	Gemiddeld grote community met veel actieve leden.
		100%	Grote en actieve community.
Cultuur	30%	0%	Het betreft een community met een gesloten cultuur.
		100%	Het betreft een community met een open cultuur.

- *Maturity*

Net als bij de community score komt er ook bij de maturity score meer kijken dan alleen een op feiten gebaseerde gegevens. De maturity van een app type wordt namelijk ook bepaald door de omvang van de community en de cultuur daarvan. Wanneer een app type een open cultuur kent en de community groot is, hebben hoogstwaarschijnlijk meer ontwikkelaars bijgedragen aan de ontwikkeling van dat app type. Hierdoor is de maturity van zo een app type hoger. Doordat hier geen feitelijke gegevens over zijn onderzocht, wordt ook de maturity score deels worden bepaald aan de hand van een ervaringsscore. Er worden drie sub-aspecten gedefinieerd, leeftijd, issue management en documentatie.

Sub-aspect	*Percentage*	*Score*	*Criteria*
Leeftijd	33%	0%	Het app type bestaat nog geen jaar en er zijn weinig release cycles geweest.
		30%	Het app type bestaat al een tijd, maar er wordt in minder snelle mate aan de doorontwikkeling gewerkt.
		60%	Het app type bestaat al enige tijd, maar is nog steeds in een vroeg stadium van ontwikkeling.
		80%	Het app type bestaat al langere tijd en er wordt nog steeds veel aan doorontwikkeld.
		100%	Er zijn veel release cycles geweest gedurende de gehele periode van bestaan.
Issue management	33%	0%	Er is geen sprake van issue management.
		30%	Issues worden wel behandeld, maar vaak niet opgelost.
		70%	In plaats van issue management zijn er meerdere (actieve) fora beschikbaar waar de ontwikkelaar met zijn vraag of issue terecht kan.
		100%	Issues worden in relatief snelle tijd opgelost.
Documentatie	33%	0%	Er is geen documentatie beschikbaar.
		30%	Er is maar een beperkte hoeveelheid documentatie beschikbaar. Bij vragen is de ontwikkelaar door ontbreken van voldoende experts, vooral op zichzelf aangewezen.
		70%	Er is voldoende documentatie over het app type beschikbaar. Vragen worden redelijk snel opgelost en ook zijn er diverse tutorials te vinden.
		100%	Er is veel documentatie over het app type beschikbaar. Met vragen kan de ontwikkelaar terecht op diverse fora waar hij relatief snel wordt geholpen. Daarnaast zijn er genoeg tutorials te vinden die de ontwikkelaar kunnen helpen bekend te raken met het app type.

- *Native APIs*

Score	*Criteria*
0%	Er is geen toegang tot Native functionaliteit.
40%	Er is toegang tot Native functionaliteit, maar deze is beperkt.
70%	De meest gebruikte Native functionaliteit wordt ondersteund.
100%	Er is onbeperkte toegang tot Native functionaliteit.

- *User Interface (UI)*

Score	*Criteria*
0%	Er is geen Native UI beschikbaar.
40%	De Native UI kan worden nagemaakt.
80%	Voor de User Interface kan gebruik worden gemaakt van beperkte Native UI componenten.
100%	Voor de User Interface kan gebruik worden gemaakt van alle Native UI componenten.

- *Offline mogelijkheden*

Score	*Criteria*
0%	Offline kan de app niet worden gebruikt.
30%	De app kan alleen offline worden gebruikt nadat deze één maal met internet is verbonden (HTML5).
100%	De app kan volledig offline worden gebruikt. Toegevoegde data wordt op een later tijdstip met de server gesynchroniseerd.

- *Portabiliteit*

Score	*Criteria*
0%	Na ontwikkeling is de app voor één besturingssysteem beschikbaar.
20%	Na ontwikkeling is de app voor één besturingssysteem beschikbaar, maar grote delen van de code kunnen worden hergebruikt voor een ander besturingssysteem.
70%	De app is na ontwikkeling beschikbaar voor een beperkt aantal smartphone besturingssystemen, waaronder de vooralsnog voor 42 BV meest belangrijke (Android en iOS).
90%	De app is na ontwikkeling beschikbaar voor de meest gebruikte smartphone besturingssystemen.
100%	De app is volledig portable en kan op vrijwel alle apparaten worden gebruikt.

5.2.2 Punten overzicht

Het score overzicht van de diverse app types wordt gegeven aan de hand van een tabel. Aan de linker kant van deze tabel is het te beoordelen aspect te vinden waarna in de overige kolommen de aspecten per app type worden beoordeeld. Doordat niet alle aspecten direct op de ontwikkelstraat van 42 BV zijn te reflecteren, zijn deze (vier) aspecten onderverdeelt in de categorie ‘overige baten’. Na het overzicht volgt een nadere toelichting op de gegeven beoordeling per app type. Hierbij wordt soms informatie die eerder in dit document aan bod is gekomen herhaald.

Te beoordelen aspect	*Telt mee voor (%)*	*Punten*	*Android*		*iOS*		*Web*		*PhoneGap*		*Titanium SDK*		*Corona SDK*	
Ontwikkelomgeving		10	100%	10,00	60%	6,00	100%	10	100%	10	80%	8,00	80%	8
Programmeertaal		10	100%	10,00	30%	3,00	70%	7	70%	7,00	70%	7,00	0%	0
Test mogelijkheden		10	100%	10,00	70%	7,00	70%	7	70%	7,00	50%	5	0%	0
Distributiewijze		4	40%	1,60	40%	1,60	100%	4	40%	1,60	40%	1,60	40%	1,60
Automated Build Tool		10	100%	10,00	70%	7,00	100%	10	100%	10	30%	3,00	0%	0
Sonar		6	100%	6,00	0%	0	50%	3	50%	3	50%	3	0%	0
Sub-totaal		*50*		*47,60*		*25*		*41,00*		*38,60*		*27,60*	*1,20*	*9,60*
Community		8												
Tools	50,00%		100%	4,00	100%	4,00	100%	4,00	100%	4,00	60%	2,40	100%	4,00
Omvang	25,00%		100%	2,00	100%	2,00	100%	2,00	70%	1,40	40%	0,80	40%	0,8
Cultuur	25,00%		100%	2,00	0%	0	100%	2,00	100%	2,00	100%	2,00	0%	0,0
Maturity		12												
Leeftijd	50,00%		100%	6,00	100%	6,00	100%	6,00	80%	4,80	70%	4,20	30%	1,80
Issue management	25,00%		100%	3,00	100%	3,00	70%	2,10	100%	3,00	100%	3,00	30%	0,90
Documentatie	25,00%		100%	3,00	100%	3,00	100%	3,00	70%	2,10	70%	2,10	70%	2,10
Sub-totaal		*70*		*67,60*		*42,60*		*60,10*		*55,90*		*42,10*		*19,20*
Overige baten														
Native APIs		9	100%	9,00	100%	9,00	0%	0	70%	6,30	70%	6,30	40%	3,60
User Interface		9	100%	9,00	100%	9,00	40%	3,6	0%	0	80%	7,20	80%	7,20
Offline mogelijkheden		3	100%	3,00	100%	3,00	30%	0,9	100%	3	100%	3	100%	3
Portabiliteit		9	0%	0,00	0%	0,00	100%	9	90%	8,10	70%	6,30	70%	6,30
Totaal		100,00		88,60		63,60		73,60		73,30		64,90		39,30

5.2.3 Android

In deze paragraaf wordt de beoordeling voor het app type Android uit voorgaande tabel toegelicht.

- *Ontwikkelomgeving*
  Na het installeren van een plug-in ondersteund de, binnen 42 BV veel gebruikte, Eclipse IDE, ontwikkeling door middel van de Android SDK. IntelliJ heeft standaard een Android module beschikbaar. Naast deze twee ontwikkelomgevingen zijn er meerdere ontwikkelomgevingen beschikbaar, waardoor Android 100% van de punten krijgt toegewezen op dit aspect.

- *Programmeertaal*
  Zoals eerder gezegd is Java de Native programmeertaal van Android[74], maar dit betekent niet automatisch dat alle Java APIs worden ondersteund op het platform. Naast de Native programmeertaal Java is er ondersteuning voor meerdere talen[75]. Doordat binnen 42 BV hoofdzakelijk Java wordt gebruikt, scoort Android 100% van de punten op dit aspect.

- *Testmogelijkheden*
  Android heeft JUnit geïntegreerd in de SDK, waardoor geautomatiseerd testen standaard ondersteund wordt[76]. Daarnaast is Mockito, een ander veel gebruikt framework binnen 42 BV, beschikbaar gemaakt voor Android door Jesse Wilson[77]. Doordat hierdoor de twee belangrijkste test frameworks voor 42 BV worden ondersteund, scoort dit aspect 100% van de punten.

- *Distributiewijze*
  De distributiewijze loopt via de officiële Android kanalen (de Play Store). Daarnaast is het ook mogelijk apps buiten de Play Store om te distribueren. Doordat binnen het bedrijf geen ervaring is met deze distributiewijze, maar hier wel veel documentatie over beschikbaar is, scoort dit aspect 40% van de punten.

- *Automated Build Tools*
  Het builden van een Android project kan onder andere gedaan worden door Build Automation tool Ant. Doordat deze build tool ook een plug-in binnen Bamboo heeft, is Continuous Integration mogelijk. Daarnaast is ook een Maven plug-in beschikbaar, welke veel door 42 BV wordt gebruikt. Door deze uitgebreide mogelijkheden scoort Android 100% van de punten op dit aspect.

- *Sonar*
  Doordat Android gebruik maakt van de Java programmeertaal, kan deze ontwikkelstraat gebruik maken van Sonar.

[74] Google-1
[75] Google-3
[76] Google-9
[77] Wilson

- *Community*
  Eerder is al aan bod gekomen dat Android tot de grootste besturingssystemen behoort. Het zal dan ook geen verassing zijn dat de interesse van developers voor dit besturingssysteem groot is. Naast dat de makers van Android al diverse tools op de markt hebben gezet voor het eigen besturingssysteem, zijn er ook vele freelance-developers die dit hebben gedaan. Daarbij zijn er geen harde onderzoeksgegevens vereist om met zekerheid te kunnen zeggen dat de community, met de opkomst van de smartphone, de afgelopen jaren sterk is gegroeid en steeds actiever is geworden. Als laatste aspect wordt genoemd dat Android een open source project betreft.

- *Maturity*
  Ondanks dat Android relatief gezien nog vrij jong is (5 jaar), is het besturingssysteem ontzettend snel doorontwikkeld de afgelopen paar jaar. Daarbij betreft het een open source project, waardoor andere ontwikkelaars ook hun steentje hebben kunnen bijdragen aan de ontwikkeling.

- *Native APIs*
  Doordat alle Native APIs gebruikt kunnen worden binnen Android, scoort dit aspect 100% van de punten.

- *User Interface*
  Doordat alle Native APIs (waaronder ook de User Interface) gebruikt kunnen worden binnen Android, scoort dit aspect 100% van de punten.

- *Offline mogelijkheden*
  Als Native besturingssysteem heeft Android toegang tot alle bestaande opslag APIs voor het platform. Hierbij kan gedacht worden aan de SQLite database, maar ook aan het lokale systeem opslag, waardoor caching mogelijk wordt gemaakt. Daarnaast vereist een Android applicatie (afhankelijk van de inhoud) geen internet verbinding om te werken. Doordat bij dit app type de beperking tot offline mogelijkheden minimaal zijn, scoort dit aspect 100% van de punten.

- *Portabiliteit*
  De reden waarom Native apps niet portable zijn is voorafgaand in dit document al meerdere malen aan bod gekomen. De reden dat Android 0% van de punten op dit aspect scoort is dan ook duidelijk.

5.2.4 iOS

In deze paragraaf wordt de beoordeling voor het app type iOS uit voorgaande tabel toegelicht.

- *Ontwikkelomgeving*

  Voor Eclipse is geen iOS plug-in beschikbaar. Tot voor kort was er wel een project gaande om voor Eclipse een plug-in te maken, maar door een onpraktische code-base is hier niet aan verder gewerkt[78]. Wel wordt getracht dit project bij de nieuwe Juno release van Eclipse (versie 4.2) voort te zetten. Verder is er binnen IntelliJ geen mogelijkheid tot Objective-C ontwikkeling. Wel heeft hetzelfde bedrijf (JetBrains) een andere IDE voor Objective-C ontwikkeld, namelijk AppCode[79]. Deze valt echter niet binnen de IntelliJ licentie en moet los worden aangeschaft.

  Doordat het bedrijf enige herkenning heeft met de AppCode IDE van JetBrains, maar verder alles nieuw is, scoort iOS 60% van de punten op dit aspect.

- *Programmeertaal*

  De Native programmeertaal van iOS is Objective-C en officieel zijn hier geen alternatieven voor. Echter in 2008 dook een bericht op, afkomstig van een medewerker van toenmalig Sun Microsystems, waarin duidelijk gesteld wordt dat Sun Microsystems bezig is met de ontwikkeling van een Java Virtual Machine voor iOS[80]. Op de JavaOne conferentie in 2011 heeft Oracle een demo gegeven van Java applicaties die onder andere op Windows Phone en een iPad draaien[81]. Dit is opvallend, gezien bij ontwikkeling voor de iOS SDK strikte regels gelden. Zo ook de volgende regel:

  > An Application may not itself install or launch other executable code by any means, including without limitation through the use of a plug-in architecture, calling other frameworks, other APIs or otherwise. No interpreted code may be downloaded or used in an Application except for code that is interpreted and run by Apple's Documented APIs and built-in interpreter(s).

  Om de applicatie van Oracle toch werkende te krijgen is gebruik gemaakt van een linked Java Virtual Machine[82]. Omdat Oracle nog niet heeft geprobeerd deze app in de App Store te krijgen is het nog onduidelijk of dit door Apple wordt geaccepteerd of niet. Een alternatief voor soortgelijke functionaliteiten is iPhoneJava[83]. Deze manier van werken wordt echter niet door Apple ondersteund, wat het hoogst waarschijnlijk maakt dat deze app niet wordt toegelaten. Doordat dit niet officieel wordt ondersteund door Apple en maar enkele medewerkers van 42 BV ervaring hebben met Objective C, scoort iOS 30% van de punten op dit aspect.

[78] Objectiveclipse
[79] JetBrains-2
[80] Krill
[81] Bruno
[82] McKenzie
[83] iPhoneJava

- *Testmogelijkheden*
  Voor iOS bestaat niet een specifiek JUnit test framework, maar wel een test framework genaamd OCUnit. Dit framework is gebaseerd op het SUnit test framework (waar JUnit ook op gebaseerd is) en is standaard geïntegreerd in Xcode[84]. Doordat dit waarschijnlijk enige herkenning betekend voor de medewerkers van 42 BV scoort iOS 70% van de punten op dit aspect. Naast OCUnit zijn ook externe test frameworks als KIF en TestFlight beschikbaar (zie deelvraag één, onderzoeksvraag twee).

- *Distributiewijze*
  De distributiewijze loopt via het officiële iOS kanaal (de App Store). Doordat binnen het bedrijf geen ervaring is met deze distributiewijze, maar hier wel veel documentatie over beschikbaar is, scoort dit aspect 40% van de punten.

- *Automated Build Tools*
  Door middel van een (zelf te maken) Build Script is het mogelijk het build proces van een iOS app te integreren met Bamboo. Doordat het hier een onbekend, handmatig en onofficieel proces betreft scoort dit aspect 30% van de punten.

- *Sonar*
  Sonar kent geen ondersteuning voor de Objective-C programmeertaal.

- *Community*
  iOS behoort, net als Android, tot de top 3 van grootste mobiele besturingssystemen. Hierdoor is het ook bij iOS niet verassend dat de interesse vanuit de bedrijven en developers in het besturingssysteem. Markttechnisch gezien is het voor bedrijven erg interessant om in iOS development te investeren, doordat Apple, het bedrijf achter iOS, het besturingssysteem dermate limiteert dat illegale software downloads minimaal zijn. Door deze gesloten instelling scoort iOS wel 0% op het cultuur aspect.

- *Maturity*
  De ontwikkeling van iOS is net als bij Android in een razend tempo geweest. Binnen enkele jaren is het besturingssysteem met een kleine groep gebruikers, uitgegroeid tot één van de meest succesvolle mobiele besturingssystemen hedendaags. Wel betreft iOS een closed source project, waardoor er vanuit de community een stuk minder input is op het gebied van doorontwikkeling.

- *Native APIs*
  Doordat alle Native APIs en bijbehorende telefoon functionaliteiten gebruikt kunnen worden binnen iOS, scoort dit aspect 100% van de punten.

[84] Sente

- *User Interface*
  Doordat alle Native APIs (waaronder ook de User Interface) gebruikt kunnen worden binnen iOS, scoort dit aspect 100% van de punten.

- *Offline mogelijkheden*
  Als Native besturingssysteem heeft iOS toegang tot alle bestaande opslag APIs voor het platform. Hierbij kan gedacht worden aan de SQLite database, maar ook aan het lokale systeem opslag, waardoor caching mogelijk wordt gemaakt. Daarnaast vereist een iOS applicatie (afhankelijk van de inhoud) geen internet verbinding om te werken. Doordat bij dit app type de beperking tot offline mogelijkheden minimaal zijn, scoort dit aspect 100% van de punten.

- *Portabiliteit*
  De reden waarom Native apps niet portable zijn is voorafgaand in dit document al meerdere malen aan bod gekomen. De reden dat iOS 0% van de punten op dit aspect scoort is dan ook duidelijk.

5.2.5 Web

In deze paragraaf wordt de beoordeling voor het app type Web uit voorgaande tabel toegelicht.

- *Ontwikkelomgeving*
  Bij het ontwikkelen van het app type Web kan van vrijwel alle ontwikkelomgevingen gebruik gemaakt worden, waaronder ook IntelliJ en Eclipse. Hierdoor scoort dit aspect 100% van de punten.

- *Programmeertaal*
  De programmeertalen die worden gebruikt bij de ontwikkeling van een Web app (HTML, CSS, JavaScript) zijn vaker gebruikt voor projecten van het bedrijf, maar behoren niet tot de hoofdprogrammeertalen van het bedrijf. Hierdoor scoort dit app type 70% van de punten op dit aspect.

- *Testmogelijkheden*
  Voor de gebruikte programmeertalen zijn testmogelijkheden beschikbaar. Echter, doordat 42 BV hier maar beperkte ervaring mee heeft scoort dit app type 70% van de punten op dit aspect.

- *Distributiewijze*
  Het app type kan door middel van een .war bestand worden uitgerold als web applicatie. Doordat dit de meest gebruikte methode binnen 42 BV betreft scoort dit app type 100% van de punten op dit aspect.

- *Automated Build Tools*
  Voor het app type Web is de automatisch builden onbeperkt. Hierdoor scoort dit app type 100% van de punten op dit aspect.

- *Sonar*
  HTML, CSS en JavaScript kennen beperkte ondersteuning binnen Sonar, waardoor dit app type 50% van de punten scoort op dit aspect.

- *Community*
  Binnen het community aspect scoort het app type Web uitzonderlijk goed. Er zijn ontzettend veel tools beschikbaar en de community is qua omvang de grootste van alle onderzochte communities behorende bij andere app types. Daarbij is de cultuur open, waardoor dit aspect 100% van de punten scoort op dit aspect.

- *Maturity*
  Dit app type is het verste doorontwikkeld van alle app types. Dit komt met name doordat het web al veel langer bestaat, en daarbij ook het meest gebruikt is op het internet. Het app type scoort 100% van de punten op twee van de sub-aspecten. Doordat er geen officiële omgeving is waar de gebruiker terecht kan met web issues, maar wel heel veel fora beschikbaar zijn, scoort het app type 70% op het issue management aspect.

- *Native APIs*
  Dit app type heeft geen toegang tot Native APIs, waardoor 0% van de punten wordt gescoord.

- *User Interface*
  Dit app type kan geen gebruik maken van de officiële User Interface APIs. Wel zijn diverse frameworks beschikbaar die de user interface nabootsen, waardoor dit app type 40% van de punten scoort.

- *Offline mogelijkheden*
  Door de komst van HTML 5 is het offline gebruiken van een web app mogelijk. Hier moet wel eenmalig internet connectie voor zijn, waardoor het app type 30% van de punten scoort op dit aspect.

- *Portabiliteit*
  Door de maximale portabiliteit die met Web apps mogelijk is scoort dit app type 100% van de punten op dit aspect.

5.2.6 PhoneGap

In deze paragraaf wordt de beoordeling voor het app type Hybrid uit voorgaande tabel toegelicht. Deze beoordeling is gebaseerd op het gebruik van de PhoneGap implementatie.

- *Ontwikkelomgeving*
  De aangeraden IDE voor PhoneGap is Adobe's Dreamweaver. Dit komt omdat deze IDE officieel ondersteuning biedt voor PhoneGap Build. Dit is echter niet de enige IDE die gebruikt kan worden. Zo is het ook mogelijk de binnen 42 BV veel gebruikte Eclipse en IntelliJ te gebruiken. Naast bovenstaand genoemde zijn er nog een aantal andere ontwikkelomgevingen die de gebruikte web technologieën ondersteunen. Door deze uitgebreide keuze scoort dit aspect 100% van de punten.

- *Programmeertaal*
  Ontwikkelen met daarbij gebruik makend van PhoneGap gebeurd door middel van HTML, CSS en JavaScript, alle drie zeer bekende web technologieën. Een groot deel van de medewerkers van 42 BV zijn bekend met deze web technologieën, waardoor dit app type 70% scoort op dit aspect.

- *Testmogelijkheden*
  Doordat gebruik wordt gemaakt van, voor 42 BV, bekende programmeertalen, zijn ook de testmogelijkheden bekend. Gebruik kan worden gemaakt van Selenium (automated browser testing) en Jasmine (het functioneel testen van JavaScript). Doordat met beide methoden al enige ervaring is binnen 42 BV, scoort dit app type 70% van de punten op dit aspect.

- *Distributiewijze*
  Het distribueren van apps van dit type gebeurd door middel van de, voor 42 BV onbekende, officiële kanalen. Doordat hier uitgebreide documentatie over beschikbaar is, scoort dit app type 40% van de punten op dit aspect.

- *Automated Build Tools*
  Met PhoneGap is het mogelijk door middel van de command-line te builden. Uit ervaring zeggen medewerkers van 42 BV dat het uitvoeren van commands door middel van de command-line is te automatiseren door Maven. Hierdoor kent dit app type op dit aspect een goede integratie op de huidige release cycle van 42 BV. Het app type scoort dan ook 100% op dit aspect.

- *Sonar*
  Doordat PhoneGap JavaScript gebaseerd is en Sonar hier deels ondersteuning voor biedt, scoort dit aspect 50% van de punten op dit aspect.

- *Community*
  Doordat PhoneGap een open cultuur kent, zijn er vanuit de community een groot aantal aan tools en plug-ins beschikbaar voor het te gebruiken framework. Daarbij kan de ontwikkelaar zelf

ook plug-ins creëren, iets wat PhoneGap ook probeert te stimuleren. Naast dat de community dus een actieve bijdrage aan het framework levert, is de omvang van de community naar mijn gevoel ook nog eens behoorlijk groot. Wanneer op internet wordt gezocht naar bepaalde problemen zijn er behoorlijk wat fora die vol staan met PhoneGap specifieke vragen. Deze vragen worden ook nog eens actief beantwoord door de community. Toch is de community in vergelijking met de Native communities een stuk kleiner, waardoor dit app type 70% op dit aspect scoort.

- *Maturity*

  Ondanks dat PhoneGap qua leeftijd al vrij oud is (2008), scoort dit aspect toch minder. Dit komt met name door de release cycles. Ik heb wel het gevoel dat het framework continu wordt doorontwikkeld, maar dat er zoveel problemen zijn om rekening mee te houden dat het nog ver van compleet is. Dit komt ook met name door de continu door ontwikkelde Native besturingssystemen. Zolang als hier nieuwe functionaliteiten aan toe worden gevoegd, zal PhoneGap nooit volledig zijn. Ondanks dat zijn de ontwikkelaars wel al een heel eind op weg, waardoor dit aspect 80% van de punten behaalt. Ondanks dat er vrij veel documentatie over PhoneGap is te vinden, is het toch nog vaak lastig om specifieke informatie over het framework te vinden. Als voorbeeld geef ik hoe de achterliggen build functionaliteit nu precies werkt. Hierover is lastig informatie te verkrijgen, waarbij een combinatie van een aantal bronnen nodig is om uiteindelijk zelf tot een conclusie te kunnen komen.

- *Native APIs*

  Dat PhoneGap toegang heeft tot Native APIs is al ruim aan bod gekomen. Ondanks dat de vooralsnog voor 42 BV belangrijke platformen (iOS en Android) een goede ondersteuning kennen, zijn er nog steeds APIs die in vergelijking met de Native app niet worden ondersteund. Door deze beperktere toegang scoort dit app type 70% van de punten op dit aspect.

- *User Interface*

  De User Interface is bij gebruik van PhoneGap over het algemeen niet Native. De Native look en feel na bootsen vereist heel veel werk. Daarbij resulteren pogingen tot nabootsen in een ‘net-niet’ situatie. Een goed voorbeeld wat hierbij gegeven kan worden is de diverse hardware fabrikanten voor Android. Veel van deze fabrikanten produceren hun eigen User Interface, om op die manier de smartphone meer fabrikant gebonden te maken. Wanneer de Native UI van Android echter wordt nagemaakt, dan is dit de standaard UI van Android en niet die van de smartphone fabrikant, waardoor de gebruiker alsnog met een verschillende UI moet werken.

  De Native UI namaken is dus niet aan te raden, maar dat is dan ook helemaal niet nodig. Er zijn genoeg alternatieven zoals jQueryMobile. Doordat de Native look en feel niet (goed) mogelijk is, scoort dit app type 0% op dit aspect.

- *Offline mogelijkheden*
  Door de toegang tot de Native APIs, wordt ook offline storage mogelijk gemaakt. Deze offline storage komt in de vorm van het bestandssysteem en toegang tot een SQLite Database. Doordat offline werken mogelijk is bij dit app type, wordt 100% gescoord op dit aspect.

- *Portabiliteit*
  Door middel van PhoneGap kan de app voor meerdere platformen beschikbaar worden gemaakt. Hierdoor is de portabiliteit hoog en scoort dit app type 90% op dit aspect.

5.2.7 Titanium SDK

In deze paragraaf wordt de beoordeling voor het app type Interpreted uit voorgaande tabel toegelicht. Deze beoordeling is gebaseerd op de Appcelerator implementatie.

- *Ontwikkelomgeving*
  Appcelerator heeft in principe zijn eigen ontwikkelomgeving, Titanium Studio. Officieel gezien is het dan ook niet mogelijk andere ontwikkelomgevingen te gebruiken. Echter, door een aantal stappen te nemen, kan er wel code completion te verkrijgen in IntelliJ. Daarnaast is de Titanium Studio Eclipse gebaseerd, waardoor er een plug-in voor Eclipse beschikbaar is. Hierdoor scoort dit app type 80% van de punten op dit aspect.

- *Programmeertaal*
  Als programmeertaal wordt gebruik gemaakt van JavaScript. Naast JavaScript wordt gebruik gemaakt van de Titanium SDK, waar een klein leertraject aan vooraf schijnt te gaan. Doordat de medewerkers wel bekend zijn met JavaScript en eigenlijk elk app type een kleine leer fase bevat, scoort dit app type 70% van de punten op dit aspect.

- *Testmogelijkheden*
  Als test framework kan gebruik worden gemaakt van Jasmine. Doordat dit wel een aantal stappen vereist om dit werkend te krijgen scoort dit app type minder dan de gebruikelijke 70% van de punten op dit aspect, namelijk 50%.

- *Distributiewijze*
  De distributiewijze verloopt hetzelfde als bij de apps van het type Native. Hierdoor scoort dit app type 40% op dit aspect.

- *Automated Build Tools*
  Het geautomatiseerd builden door middel van de command-line is alleen mogelijk door middel van een eerder genoemde modificatie[85]. Het officiële build proces verloopt via een grafische interface. Doordat command-line builden niet standaard wordt ondersteund in de Titanium SDK en de enige andere optie is het handmatig te doen, scoort het app type 30% op dit aspect.

- *Sonar*
  Doordat de Titanium SDK JavaScript gebaseerd is en Sonar hier deels ondersteuning voor biedt, scoort dit aspect 50% van de punten op dit aspect.

- *Community*
  Wat betreft Appcelerator's Titanium schijnt het toch een iets minder bekend platform te zijn. Hierdoor heb ik het idee dat de cultuur ook een stuk kleiner is dan bijvoorbeeld die van

[85] Chapiewski

PhoneGap. Bijkomend heeft deze kleinere cultuur tot gevolg dat, ondanks dat het een open source project betreft, in mindere mate tools beschikbaar zijn. Daarnaast heb ik het idee dat de community draait op een aantal personen die veel verstand hebben van de SDK. Een aantal namen zie je dan ook op verschillende fora terugkomen met antwoorden op vragen van gebruikers. Hierdoor scoort dit app type zowel op beschikbaarheid van tools als de omvang van de community een lage score, namelijk 60% en 40% van de punten.

- *Maturity*
  Ook qua maturity blijft deze manier van ontwikkelen achter. Zowel op documentatie vlak als doorontwikkeling heb ik het idee dat de SDK verder achterloopt dan bij zijn voornaamste concurrent PhoneGap het geval is. Desondanks scoort dit app type op beide vlakken wel 70% van de punten.

- *Native APIs*
  Appcelerator Titanium heeft toegang tot meer dan 5000 platform APIs. Ondanks dat dit een hele grote ondersteuning is, zijn er nog steeds andere APIs die niet worden ondersteund. Op dit aspect scoort het app type dan ook 70% van de punten.

- *User Interface*
  Gebruik kan worden gemaakt van platform specifieke User Interface componenten. Wanneer een bepaald Native component niet wordt ondersteund binnen de Titanium API kan de gebruiker deze zelf implementeren. Door deze mogelijkheid, in combinatie met de grote set standaard beschikbare UI componenten, scoort het app type hoog op dit aspect.

- *Offline mogelijkheden*
  De ontwikkelaar kan door de Titanium API toegang verkrijgen tot diverse lokale opslag mogelijkheden. Doordat offline werken hierdoor mogelijk is, scoort het app type hoog op dit aspect.

- *Portabiliteit*
  Na het build proces is de app beschikbaar voor zowel iOS en Android. Het nadeel is dat andere platformen zoals Windows Phone of BlackBerry tot nog toe niet mogelijk zijn. Mede doordat vooralsnog alleen iOS en Android belangrijk zijn voor 42 BV, scoort het app type toch hoog op dit aspect.

5.2.8 Corona SDK

In deze paragraaf wordt de beoordeling voor het app type Cross-Compiling uit voorgaande tabel toegelicht. Deze beoordeling is gebaseerd op de Corona SDK implementatie.

- *Ontwikkelomgeving*
  Bij het ontwikkelen van een Corona SDK app wordt gebruik gemaakt van een willekeurige tekst editor. Daarnaast is er voor Eclipse een plug-in voor de Lua scripttaal (de taal voor de Corona SDK) beschikbaar. Doordat Eclipse bekend terrein is en daarnaast vrijwel alle editors gebruikt kunnen worden, scoort het app type 80% van de punten op dit aspect.

- *Programmeertaal*
  Corona apps worden geschreven door middel van de Lua scripttaal. Gezien dit onbekend terrein is voor 42 BV medewerkers en hier geen alternatieven voor zijn, scoort het app type geen punten op dit aspect.

- *Testmogelijkheden*
  Testen gebeurd met name in de Corona Simulator. Daarnaast is het mogelijk om Lua-TestMore werkende te krijgen voor de SDK, maar dit wordt niet standaard ondersteund. Hierdoor scoort dit app type geen punten op dit aspect.

- *Distributiewijze*
  De uiteindelijke apps die worden gegenereerd zijn van het app type Native en kunnen daartoe via de officiële kanalen worden gedistribueerd. Doordat deze distributiewijze binnen 42 BV niet bekend is, maar hier wel voldoende documentatie over beschikbaar is, scoort dit app type 40% van de punten op dit aspect.

- *Automated Build Tools*
  Voor het build proces van apps gemaakt door middel van de Corona SDK, kan alleen gebruik worden gemaakt van Corona Build. Dit betreft een handmatig proces, welke herhaalt dient te worden per besturingssysteem. Doordat dit niet te automatiseren valt, scoort het app type 0% op dit aspect.

- *Sonar*
  Doordat de Corona SDK Lua gebaseerd is en Sonar daar geen ondersteuning voor kent, scoort dit app type geen punten op dit aspect.

- *Community*
  Ondanks dat de Corona SDK zowel een kleine community als een gesloten cultuur heeft, zijn er toch nog een redelijk aantal aan tools beschikbaar voor de SDK. Dit is opvallend, doordat ik bij andere app types met name opmerkte dat wanneer de cultuur open is, meer tools beschikbaar zouden zijn. Dit is dus niet het geval bij de Corona SDK, waardoor dit app type 100% van de

punten op het tools aspect scoort. Wel is de community erg klein, waardoor hier maar 40% wordt behaald.

- *Maturity*
  Ik heb het idee dat de Corona SDK, wanneer wordt gekeken naar leeftijd en release cycles, nog erg jong is. Het heeft naar mijn mening nog niet de mate van doorontwikkeling gehad als de overige app types. Helaas kan ik weinig zeggen over het issuemanagement van het platform, doordat een eerdere lijst van bug reports niet is te vinden. Hierdoor kan niet worden gekeken naar de snelheid waarop problemen worden behandeld. Van de documentatie heb ik wel het idee dat er genoeg te vinden is, wat er uiteindelijk toe leidt dat dit app type een boven gemiddelde score haalt op dit aspect.

- *Native APIs*
  Vanuit de Corona SDK is toegang tot ongeveer 500 Native APIs, waardoor de meest gebruikte Native functionaliteiten binnen de Corona SDK gebruikt kunnen worden. Doordat deze toegang tot Native APIs in vergelijking met bijvoorbeeld PhoneGap en Titanium erg beperkt is, scoort het app type 40% van de punten op dit aspect.

- *User Interface*
  Doordat de Corona SDK maar beperkte toegang tot Native User Interface componenten biedt, scoort het app type 80% op dit aspect.

- *Offline mogelijkheden*
  De ontwikkelaar heeft toegang tot lokale opslag mogelijkheden waardoor de gemaakte app ook offline kan werken. Door deze mogelijkheid scoort het app type 100% van de punten op dit aspect.

- *Portabiliteit*
  Door gebruik te maken van de Corona Build is de app beschikbaar te maken voor zowel iOS en Android (beide moeten handmatig worden gebuild). Het nadeel is dat andere platformen zoals Windows Phone of BlackBerry tot nog toe niet mogelijk zijn. Mede doordat vooralsnog alleen iOS en Android belangrijk zijn voor 42 BV, scoort het app type toch hoog op dit aspect.

5.3 Conclusie app-types

In de conclusie van dit hoofdstuk worden de sterke en zwakke punten van de diverse app types kort samengevat.

Android-app

Dit app type is het sterkst op de aansluiting die het vind op de huidige ontwikkelsituatie van 42 BV. Zo maakt het gebruik van Java, kan de ontwikkeling zowel in een Eclipse als IntelliJ ontwikkelomgeving plaatsvinden, en kan gebruik worden gemaakt van voor 42 BV bekende testmethoden.

Naast de bijna perfecte aansluiting op de ontwikkelsituatie van 42 BV geven Android apps de gebruiker de best mogelijke Native Experience, omdat gebruik kan worden gemaakt van de Native User Interface API, hetgeen waar het begrip Native Experience op is gebaseerd. Daarnaast kan door het gebruik van de Android SDK tijdens de ontwikkeling alle native functionaliteit worden gebruikt die benodigd is voor de app.

De een reden om echter niet voor Android te kiezen ligt in een heel andere hoek, namelijk die van portabiliteit. Doordat Android apps specifiek voor één platform worden geschreven is de portabiliteit tussen verschillende platformen minimaal. Dit maakt Android in veel mindere mate geschikt om binnen 42 BV te gebruiken, omdat kostenbesparend werken van belang is. Door de slechte portabiliteit dient voor elk overig gewenst platform een nieuwe app te worden ontwikkeld, wat dubbele kosten tot gevolg heeft.

iOS-app

De iOS app heeft zijn sterke punten op de toegang tot de native functionaliteit en experience. Doordat de app specifiek voor het native besturingssysteem iOS wordt geschreven kan gebruik worden gemaakt van de Native APIs, waardoor alle smartphone functionaliteit benut kan worden. Tevens wordt hierdoor de Native User Interface gebruikt, wat een Native Experience voor de gebruiker tot gevolg heeft.

Ondanks de voordelen van dit app type wegen de nadelen zwaar op. De ontwikkelstraat van het iOS app type sluit op vrijwel geen enkel aspect aan op de ontwikkelsituatie van 42 BV, wat een lang leertraject tot gevolg heeft voor de developers. Hier gaat veel tijd inzitten, wat niet altijd mogelijk is. Daarbij komt ook dat de iOS apps geheel niet portable zijn.

Web-app

Ook de Web-app heeft een aantal sterke punten. Door gebruik te maken van het web wordt door dit app type de maximale portabiliteit behaald. Daarbij sluit de ontwikkelstraat van het app type aan op de huidige ontwikkelsituatie van 42 BV. Het bedrijf heeft al vaker te maken gehad met het ontwikkelen van web applicaties, waardoor het ontwikkelen van web apps voor mobiel gebruik een minimaal leerproces bevat.

De maximale portabiliteit die wordt behaald heeft echter wel een hoge prijs, het app type Web levert namelijk alles in op het gebied van native functionaliteit. Doordat de Native APIs niet gebruikt kunnen

worden valt ook de Native Experience tegen. Dit laatste kan alleen (deels) behaald worden door de Native Experience na te bootsen.

PhoneGap- app

Een PhoneGap-app vindt een middenweg tussen de bedrijfssituatie, portabiliteit en native functionaliteit. Naast dat PhoneGap gebruik kan maken van de Native APIs is het platform ook portable. Momenteel kent PhoneGap ondersteuning voor de meest gangbare smartphone besturingssystemen, waarvan volledige functionaliteit ondersteuning voor iOS en Android. De ontwikkelsituatie van PhoneGap sluit half aan op de ontwikkelsituatie van 42 BV. Zo kan gebruik worden gemaakt van Eclipse en IntelliJ en wordt de app door middel van bekende web talen geschreven.

Doordat PhoneGap gewrapped wordt in een web view functionaliteit van de diverse platformen is gebruik van de Native User Interface niet mogelijk.

Titanium-app

Ook de Titanium-app weet een goede middenweg te vinden in de belangrijkste punten die zijn onderzocht. Een Titanium-app heeft toegang tot de grote meerderheid van de Native functionaliteiten en kan daarbij door zijn platform onafhankelijke manier van werken gebruik maken van de Native User Interface van de besturingssystemen. Dit wordt mogelijk gemaakt doordat run-time de code wordt geïnterpreteerd door middel van de Titanium SDK. Naast dat de Titanium-app goede ondersteuning kent voor deze twee aspecten is het platform tevens portable genoeg om in het begin stadium van app ontwikkeling binnen 42 BV te voldoen. De SDK kent namelijk ondersteuning voor zowel iOS als Android, de twee besturingssystemen waar 42 BV vooralsnog mee aan de slag wil.

De ontwikkelsituatie sluit echter niet aan op de ontwikkelstraat van 42 BV. Voor veel aspecten moeten eerst een aantal modificaties worden gedaan wil het mogelijk zijn de ontwikkelstraat van 42 BV te gebruiken in combinatie met de Titanium SDK.

Corona-app

De Corona-app is in vergelijking met de overige app types het minst geschikt voor gebruik binnen 42 BV. Er worden wel een aantal Native functionaliteiten beschikbaar gesteld, waaronder de Native User Interface, maar de hoeveelheid functionaliteiten die worden ondersteund zijn beperkt. Daarbij sluit de ontwikkelsituatie in bijna geen enkel opzicht aan op de huidige ontwikkelstraat van 42 BV.

6 Deelvraag 3 – Commercieel aspect

Door ontwikkelingen en nieuwe inzichten is besloten de onderzoeksvragen van de derde deelvraag niet nader te onderzoeken. 42 BV heeft aangegeven zelf al een duidelijk beeld te hebben bij wat de klanten voor mobiele applicaties willen, waardoor een onderzoek hier naar momenteel geen meerwaarde heeft. In de conclusie van het rapport wordt het commerciële aspect van het mobiele markt niet meegewogen, waardoor het uiteindelijk resultaat op technische aspecten wordt gebaseerd.

Conclusie

Uit het onderzoek is gebleken dat er voor het ontwikkelen van mobiele apps een vijftal aan opties bestaan om met de ontwikkeling van dit soort apps aan de slag te gaan. Elke optie, resulterend in een app type, heeft zijn eigen voor en nadelen en hier moet het bedrijf een afweging in maken.

Ideaalsituatie

In de ideaalsituatie bestaat een app type die aan de volgende eisen voldoet:

1. De gebruiker heeft het gevoel dat hij met een Native applicatie te maken heeft (Native Experience).
2. Alle functionaliteiten van de smartphone kunnen worden benut (Native APIs).
3. De applicatie kan na ontwikkeling direct op meerdere platformen draaien (Portabiliteit).

Dit zijn echter niet de enige factoren waar rekening mee gehouden moet worden, want uiteindelijk moet 42 BV met de app ontwikkeling aan de slag kunnen, zonder dat daar een heel leerproces voor de medewerkers aan vooraf gaat. Hierdoor wordt een vierde eis aan de ideaalsituatie gesteld.

4. De ontwikkelstraat van het app type moet aansluiten op de ontwikkelstraat die momenteel wordt gehanteerd binnen 42 BV (Ontwikkelsituatie).

Deze vier eisen aan de ideaalsituatie houden geen rekening met het commerciële aspect van de mobiele markt. Het commerciële aspect wordt in veel mindere mate meegewogen in het uiteindelijke advies, maar komt bij de Native app types wel aan bod.

Reflectie op de ideaalsituatie

Het onderzoek wijst uit dat er app types bestaan die aan maximaal drie van de vier gestelde eisen voldoen, maar nooit allemaal. Kortom, het ideale app type bestaat (nog) niet. Wat er dan wel bestaat is voor 42 BV van belang om te weten, zodat een verantwoorde keuze kan worden gemaakt. Om het overzicht van de onderzochte app types, gereflecteerd op de ideaalsituatie, is als volgt:

	Native Experience	*Native APIs*	*Portabiliteit*	*Ontwikkelsituatie*
Android	++	++	--	++
iOS	++	++	--	--
Web	--	--	++	+
PhoneGap	--	+	++	+-
Titanium	+	+	+	-
Corona	+	-	+	--

++ Aansluiting op het aspect is optimaal.
+ Aansluiting op het aspect is goed.
- Aansluiting op het aspect is slecht.
-- Aansluiting op het aspect is minimaal.

Advies

Het is duidelijk dat welke app ook gekozen wordt, er altijd een consequentie aan vast zit waar het bedrijf mee om moet leren gaan. In eerste instantie is het van essentieel belang dat 42 BV bepaald welke aspecten uit de ideaalsituatie naar bedrijfsmening de hoogste prioriteit hebben. Dit kan eventueel in overleg met de klant worden gedaan waarvoor de app bedoeld is. Het is daarbij belangrijk dat duidelijk wordt uitgelegd aan de klant wat het probleemdomein is en waarom een app niet aan alle punten van de ideaalsituatie kan voldoen.

Mijn definitieve advies is dan ook in grote mate afhankelijk van de gestelde prioriteit. In het onderstaande overzicht geef ik de keuzes die overwogen kunnen worden met het grootst geleden nadeel:

App type	*Nadeel*
Android/iOS	Hoge ontwikkelkosten, lang leertraject iOS.
PhoneGap	Geen native experience.
Web	Geen native functionaliteit.
Titanium	Mindere aansluiting op de ontwikkelomgeving met een leertraject tot gevolg.

Indien alleen wordt gekeken naar de cijfers die voorafgaand in dit onderzoek naar voren zijn gekomen (zie ***pagina 79***), zou de keuze tot ontwikkeling vallen op een Android app. Wanneer de keus echter op Android valt, moet er hoogstwaarschijnlijk door toedoen van de klant ook voor iOS een app worden ontwikkeld. Qua aansluiting op ontwikkelstraat van 42 BV is dit besturingssysteem bijna het tegenovergestelde van Android. Doordat hierdoor twee apps ontwikkeld moeten worden, stijgen de ontwikkeling en onderhoudskosten. Daarbij is een langer leertraject benodigd om met iOS aan de slag te gaan. Dit is ook nadelig voor de klant, omdat zij dieper in de buidel moeten tasten daardoor. Met name hierdoor is het belangrijk dat ook de klant, of een gerepresenteerde, wordt betrokken bij het maken van een keuze tussen de verschillende app types en hun nadelen.

Als laatste wil ik een aantal persoonlijke ideeën delen over het PhoneGap en Titanium framework.

PhoneGap

PhoneGap is het framework waarvan ik het idee heb dat deze het beste aansluit bij de wensen van zowel 42 BV als de klant. Het enige nadeel aan dit framework is het ontbreken van een Native Experience. Doordat de meningen of dit nu wel of niet nadelig is, is per klant nogal verschillend. Hierdoor is het raadzaam dit bespreekbaar te maken met de klant. Het is van belang te realiseren dat er diverse User Interface frameworks beschikbaar zijn voor gebruik met PhoneGap, die speciaal zijn geoptimaliseerd voor mobiel gebruik. Vooraf kan hierover worden gezegd dat deze niet de volledige Native Experience na kunnen bootsen, maar de kwaliteit en gebruikersvriendelijkheid is wel degelijk aan de hoge kant. Voorbeelden van soortgelijke frameworks zijn Sencha Touch en jQuery Mobile. Het kan voordelig zijn een demo applicatie te maken die gebruik maakt van een dergelijk UI-framework, waarmee de klant overtuigd kan worden dat een app niet per definitie aan de Native Experience hoeft te voldoen om kwalitatief hoogwaardig te zijn. Aan de hand van een soortgelijke demo kan worden

achterhaald of de klant het inderdaad als een probleem ziet dat er geen native user interface wordt gebruikt.

Titanium

Als tweede raad ik aan verder te oriënteren op de Titanium SDK. In het onderzoek is naar voren gekomen dat dit app type niet goed aansluit op de huidige ontwikkelsituatie. Echter is ook naar voren gekomen dat door middel van een aantal modificaties de ontwikkelsituatie meer geschikt is te maken voor gebruik binnen 42 BV. Om de mate waarin de ontwikkelsituatie na het doorvoeren van modificaties aansluit op die van het bedrijf te bepalen, moet echter een praktisch onderzoek worden verricht. In dit onderzoek moet worden gekeken naar welke modificaties allemaal mogelijk zijn en of dit ook daadwerkelijk een verandering brengt in de aansluiting op de ontwikkelsituatie. Dit is een eenmalige investering in dit framework, wat mits de modificaties verschil maken, nog best wel eens zijn vruchten kan afwerpen. In het geval dit verschil maakt, voldoet de Titanium SDK namelijk aan alle vier de punten van de ideaalsituatie. Echter moet wel rekening gehouden worden met het feit dat de Titanium SDK momenteel alleen ondersteuning biedt voor Android en iOS. Dit kan in de toekomst nog enige problemen opleveren wanneer het bedrijf besluit meerdere platformen te ondersteunen.

Complexiteit app

Eerder in het onderzoek is naar voren gekomen dat naarmate apps complexer worden, de ontwikkelaars tegen steeds vreemdere bugs aanlopen. Naarmate de ontwikkelaar tegen de limieten van een framework aan ontwikkeld, zal steeds duidelijker worden waar een framework wel en niet toe in staat is. Gezien dit niet binnen het bereik van dit onderzoek valt, maar mijn verwachting is dat dit bij de diverse frameworks een rol gaat spelen in de toekomst, is het aan te raden hier verder onderzoek naar te doen indien voor een specifiek framework wordt gekozen.

Bronvermelding

[Adobe]

Publicatie	Internet (webpagina)
Datum geraadpleegd	03-07-2012
Auteur	Adobe
Titel	Flash Builder
Bron	http://www.adobe.com/products/flash-builder.html

[AMC-UVA]

Publicatie	Internet
Datum geraadpleegd	11-06-2012
Auteur	AMC - UvA
Titel	Richtlijnen Kwalitatief Onderzoek
Bron	http://www.emgo.nl/kc/preparation/research%20design/Richtlijnen%20Kwalitatief%20Onderzoek_AmCOGG.pdf

[Apache-1]

Publicatie	Internet (webpagina)
Datum geraadpleegd	25-06-2012
Auteur	Apache
Titel	What is Maven
Bron	http://maven.apache.org/what-is-maven.html

[Apache-2]

Publicatie	Internet (webpagina)
Datum geraadpleegd	16-07-2012
Auteur	Apache
Titel	Apache Cordova
Bron	https://issues.apache.org/jira/browse/CB#selectedTab=com.atlassian.jira.plugin.system.project%3Aissues-panel

[Appcelerator-1]

Publicatie	Internet
Datum geraadpleegd	19-06-2012
Auteur	N.N.
Titel	Appcelerator Titanium
Bron	http://en.wikipedia.org/wiki/Appcelerator_Titanium

[Appcelerator-2]
Publicatie Internet (webpagina)
Datum geraadpleegd 05-07-2012
Auteur Appcelerator
Titel Platform
Bron http://www.appcelerator.com/platform

[Applause]
Publicatie Internet (webpagina)
Datum geraadpleegd 10-07-2012
Auteur N.N.
Titel Applause
Bron https://github.com/applause/applause

[Apple-1]
Publicatie Internet (nieuwsartikel)
Datum geraadpleegd 1-05-2012
Auteur Apple
Titel iPhone App Store downloads top 10 Million in first weekend
Bron http://www.apple.com/pr/library/2008/07/14iPhone-App-Store-Downloads-Top-10-Million-in-First-Weekend.html

[Apple-2]
Publicatie Internet (nieuwsartikel)
Datum geraadpleegd 1-05-2012
Auteur Apple
Titel Press Info
Bron http://www.apple.com/pr/

[Apple-3]
Publicatie Internet
Datum geraadpleegd 18-06-2012
Auteur Apple
Titel Start Developing iOS Apps
Bron https://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMapiOS/GetToolsInstall/GetToolsandInstall.html

[Apple-4]
Publicatie Internet
Datum geraadpleegd 18-06-2012
Auteur Apple
Titel iOS
Bron https://developer.apple.com/programs/ios/

[Apple-5]
Publicatie Internet
Datum geraadpleegd 18-06-2012
Auteur Apple
Titel View Controller Basics
Bron http://developer.apple.com/library/ios/#featuredarticles/ViewControllerPGforiPhoneOS/Introduction/Introduction.html

[Atlassian-1]
Publicatie Internet (webpagina)
Datum geraadpleegd 25-06-2012
Auteur Atlassian
Titel Jira
Bron http://www.atlassian.com/software/jira/overview

[Atlassian-2]
Publicatie Internet (webpagina)
Datum geraadpleegd 26-06-2012
Auteur Atlassian
Titel Bamboo Overview
Bron http://www.atlassian.com/software/bamboo/overview

[Bakker]
Publicatie Java Magazine
Datum geraadpleegd 25-06-2012
Auteur Paul Bakker
Titel RESTful Webservices
Bron http://www.infosupport.com/RESTful_Webservices_Paul_Bakker

[Bleijendaal]
Publicatie Internet
Datum geraadpleegd 01-05-2012
Auteur Remy Bleijendaal
Titel App Bouwen? Begin met Android
Bron http://www.frankwatching.com/archive/2011/10/03/app-bouwen-begin-met-android/

[Bochinski]
Publicatie Internet (webpagina)
Datum geraadpleegd 10--07-2012
Auteur Arek Bochinski
Titel Corona SDK Build Automation
Bron http://arekzb.wordpress.com/2011/10/03/corona-sdk-build-automation/

[Brogdon]
Publicatie Internet (webpagina)
Datum geraadpleegd 05-07-2012
Auteur Darrell Brogdon
Titel How does Appcelerator Titanium Mobile work
Bron http://stackoverflow.com/questions/2444001/how-does-appcelerator-titanium-mobile-work/2471774#2471774

[Bruno]
Publicatie Internet (webpagina)
Datum geraadpleegd 27-06-2012
Auteur Eric Bruno
Titel JavaOne: JavaFX 2, Java on iOS
Bron http://www.drdobbs.com/blogs/jvm/231900029

[Business 2 Community]
Publicatie Internet (artikel)
Datum geraadpleegd 1-05-2012
Auteur Business 2 Community
Titel A Look Back in Time at the First Smartphone Ever
Bron http://www.business2community.com/mobile-apps/a-look-back-in-time-at-the-first-smartphone-ever-040906

[Chapiewski]
Publicatie Internet (webpagina)
Datum geraadpleegd 03-07-2012
Auteur Guilherme Chapiewski
Titel Titanium Mobile Hack
Bron http://guilherme.pro/2011/04/06/titanium-mobile-hack-execute-your-projects-from-the-command-line-using-make/

[Connor-1]
Publicatie Internet (blog)
Datum geraadpleegd 22-06-2012
Auteur Connor Turnbull
Titel
Bron http://android.appstorm.net/general/opinion/android-vs-ios-its-about-the-apps/

[Consumenten Bond-1]
Publicatie Internet
Datum geraadpleegd 18-06-2012
Auteur Consumenten Bond
Titel Mobiele besturingssystemen
Bron http://www.consumentenbond.nl/test/elektronica-communicatie/telefonie/mobiele-telefoons/extra/mobiele-besturingssystemen/

[Corona]
Publicatie Internet (webpagina)
Datum geraadpleegd 10-07-2012
Auteur CoronaLabs
Titel Corona SDK
Bron http://www.coronalabs.com/products/corona-sdk/

[Dallera-1]
Publicatie Internet(blog)
Datum geraadpleegd 21-06-2012
Auteur Andrea Dallera
Titel Why you should stay away from Appcelerator's Titanium
Bron http://usingimho.wordpress.com/2011/06/14/why-you-should-stay-away-from-appcelerators-titanium/

[Donald]
Publicatie Internet (webpagina)
Datum geraadpleegd 18-07-2012
Auteur Simon Mac Donald
Titel Android issues all PhoneGap developers should start.
Bron http://simonmacdonald.blogspot.nl/2012/02/android-issues-all-phonegap-developers.html

[Eduvision]
Publicatie Internet (webpagina)
Datum geraadpleegd 16-07-2012
Auteur Eduvision
Titel Cursus PhoneGap
Bron http://www.eduvision.nl/cursus/cursus_phonegap.html

[Encyclopedie-1]
Publicatie Internet (Encyclopedie)
Datum geraadpleegd 1-05-2012
Auteur N.N.
Titel Smartphone History
Bron http://en.wikipedia.org/wiki/Smartphone#History

[Encyclopedie-2]
Publicatie Internet (Encyclopedie)
Datum geraadpleegd 1-05-2012
Auteur N.N.
Titel App Store
Bron http://en.wikipedia.org/wiki/App_Store_(iOS)

[Encyclopedie-3]
Publicatie Internet
Datum geraadpleegd 1-05-2012
Auteur N.N.
Titel List of iOS devices
Bron http://en.wikipedia.org/wiki/List_of_iOS_devices

[Enough-1]
Publicatie Internet
Datum geraadpleegd 15-06-2012
Auteur Enough
Titel Mobile Developers Guide
Bron http://www.enough.de/products/mobile-developers-guide/

[Epic Games-1]
Publicatie Internet
Datum geraadpleegd 18-06-2012
Auteur Epic Games
Titel Unreal Development Kit
Bron http://www.udk.com/

[FD]
Publicatie Internet
Datum geraadpleegd 06-06-2012
Auteur FD
Titel Prijsstijging voor mobiel internet onvermijdelijk
Bron http://fd.nl/Archief/2011/02/26/prijsstijging-voor-mobiel-internet-onvermijdelijk

[Friese-1]
Publicatie Presentatie
Datum geraadpleegd 13-06-2012
Auteur Peter Friese
Titel Cross-platform mobile Development
Bron http://www.slideshare.net/peterfriese/cross-platform-mobile-development-11239246

[Friese-2]
Publicatie Internet
Datum geraadpleegd 13-06-2012
Auteur Peter Friese
Titel Peter Friese
Bron http://www.slideshare.net/peterfriese

[Friese-3]
Publicatie Internet (webpagina)
Datum geraadpleegd 11-07-2012
Auteur Peter Friese
Titel How to use the gyroscope of your iPhone in a mobile web app
Bron http://www.peterfriese.de/how-to-use-the-gyroscope-of-your-iphone-in-a-mobile-web-app/

[Gartner-1]
Publicatie Internet (nieuwsartikel)
Datum geraadpleegd 1-05-2012
Auteur Gartner
Titel Worldwide mobile phone sales
Bron http://www.gartner.com/it/page.jsp?id=1466313

[Gartner-2]
Publicatie Internet (nieuwsartikel)
Datum geraadpleegd 1-05-2012
Auteur Gartner
Titel Worldwide smartphone sales
Bron http://www.gartner.com/it/page.jsp?id=910112

[Gartner-3]
Publicatie Internet (nieuwsartikel)
Datum geraadpleegd 1-05-2012
Auteur Gartner
Titel Sales of mobile devices
Bron http://www.gartner.com/it/page.jsp?id=1848514

[Google-1]
Publicatie Internet
Datum geraadpleegd 18-06-2012
Auteur Google
Titel Fundamentals
Bron http://developer.android.com/guide/topics/fundamentals.html

[Google-2]
Publicatie Internet
Datum geraadpleegd 18-06-2012
Auteur Google
Titel Platform Versions
Bron http://developer.android.com/resources/dashboard/platform-versions.html

[Google-3]
Publicatie Internet
Datum geraadpleegd 18-06-2012
Auteur Google
Titel Overview
Bron http://developer.android.com/sdk/ndk/overview.html

[Google-4]
Publicatie Internet
Datum geraadpleegd 15-06-2012
Auteur Google
Titel Guide
Bron http://developer.android.com/guide

[Google-5]
Publicatie Internet
Datum geraadpleegd 15-06-2012
Auteur Google
Titel Reference
Bron http://developer.android.com/reference

[Google-6]
Publicatie Internet
Datum geraadpleegd 14-06-2012
Auteur Google
Titel SDK
Bron http://developer.android.com/sdk

[Google-7]
Publicatie Internet
Datum geraadpleegd 14-06-2012
Auteur Google
Titel Lint
Bron http://tools.android.com/recent/lint

[Google-9]
Publicatie Internet (webpagina)
Datum geraadpleegd 25-06-2012
Auteur Google
Titel Testing Android
Bron http://developer.android.com/tools/testing/testing_android.html

[Google-10]
Publicatie Internet (webpagina)
Datum geraadpleegd 27-06-2012
Auteur Google
Titel Android Testing
Bron https://sites.google.com/site/androiddevtesting/

[Google-11]
Publicatie Internet (webpagina)
Datum geraadpleegd 25-06-2012
Auteur Google
Titel Package Index
Bron http://developer.android.com/reference/packages.html

[GoogleGroup]
Publicatie Internet
Datum geraadpleegd 21-06-2012
Auteur Smukamuka (Google Group gebruiker)
Titel App Rejected two times
Bron https://groups.google.com/forum/?fromgroups#!topic/phonegap/Yj84MvdUUVc

[Holthe]
Publicatie Internet
Datum geraadpleegd 25-06-2012
Auteur Oscar Westra van Holthe - Kind
Titel REST-Webservice
Bron https://confluence.42.nl/display/projects/REST-webservice

[Hoogers-1]
Publicatie Internet
Datum geraadpleegd 11-06-2012
Auteur Rebecca Hoogers
Titel Kwalitatief Onderzoek
Bron http://wetenschap.infonu.nl/onderzoek/88702-kwalitatief-onderzoek.html

[Hoogers-2]
Publicatie Internet
Datum geraadpleegd 11-06-2012
Auteur Rebecca Hoogers
Titel Kwantitatief Onderzoek
Bron http://wetenschap.infonu.nl/onderzoek/88701-kwantitatief-onderzoek.html

[HSQL]
Publicatie Internet (webpagina)
Datum geraadpleegd 25-06-2012
Auteur SourceForge
Titel HyperSQL Database Engine
Bron http://sourceforge.net/projects/hsqldb/forums/forum/73674/topic/4538757

[HTML5Readiness]
Publicatie Internet
Datum geraadpleegd 19-06-2012
Auteur N.N.
Titel HTML5 & CSS3 Readiness
Bron http://www.html5readiness.com/

[IBM]
Publicatie Internet (artikel)
Datum geraadpleegd 25-06-2012
Auteur Yi Ming Huang, Dong Fei Wu (werknemers IBM)
Titel Build RESTful web services using Spring 3
Bron http://www.ibm.com/developerworks/webservices/library/wa-spring3webserv/index.html

[iPhoneJava]
Publicatie Internet (webpagina)
Datum geraadpleegd 12-07-2012
Auteur N.N.
Titel iPhone Java
Bron http://www.iphonejava.org/

[Jensen]
Publicatie Internet (webpagina)
Datum geraadpleegd 25-06-2012
Auteur Eric Jensen
Titel Connecting to an Oracle DB via an Android App
Bron http://stackoverflow.com/questions/5727857/connecting-to-an-oracledb-via-an-android-app

[JetBrains-1]
Publicatie Internet (webpagina)
Datum geraadpleegd 25-06-2012
Auteur JetBrains
Titel IntelliJ IDEA
Bron http://www.jetbrains.com/idea/

[JetBrains-2]
Publicatie Internet (webpagina)
Datum geraadpleegd 27-06-2012
Auteur JetBrains
Titel Objective-C IDE that makes a difference
Bron http://www.jetbrains.com/objc/index.html

[JMockit]
Publicatie Internet(artikel)
Datum geraadpleegd 25-06-2012
Auteur N.N
Titel JMockit
Bron http://code.google.com/p/jmockit/

[JUnit]
Publicatie Internet(Webpagina)
Datum geraadpleegd 25-06-2012
Auteur N.N.
Titel JUnit
Bron http://www.junit.org/

[Kassenaar]
Publicatie Internet (webpagina)
Datum geraadpleegd 16-07-2012
Auteur Peter Kassenaar
Titel Cursus mobiele apps ontwikkelen met Dreamweaver, jQuery Mobile en PhoneGap.
Bron http://www.kassenaar.com/blog/post/2012/02/15/Cursus-Mobiele-apps-ontwikkelen-met-Dreamweaver-jQuery-Mobile-en-PhoneGap.aspx

[Krill]
Publicatie Internet (webpagina)
Datum geraadpleegd 27-06-2012
Auteur Paul Krill
Titel Sun: We'll put Java on the iPhone
Bron http://www.infoworld.com/d/developer-world/sun-well-put-java-iphone-042

[McKenzie]
Publicatie Internet (webpagina)
Datum geraadpleegd 25-06-2012
Auteur Cameron McKenzie
Titel Java on the iPhone at JavaOne
Bron http://www.theserverside.com/news/thread.tss?thread_id=63072

[Mediawijzer]
Publicatie Internet (nieuwsartikel)
Datum geraadpleegd 1-05-2012
Auteur Mediawijzer
Titel Forse stijging aantal smartphonebezitters
Bron http://www.mediawijzer.net/professionals/nieuws/forse-stijging-aantal-smartphonebezitters

[Media Optima Forma]
Publicatie Internet (artikel)
Datum geraadpleegd 1-05-2012
Auteur Media Optima Forma
Titel Wat zijn apps?
Bron http://mediaoptimaforma.nl/modern-media/wat-zijn-apps/

[Mockito]
Publicatie Internet(artikel)
Datum geraadpleegd 25-06-2012
Auteur N.N.
Titel Mockito
Bron http://code.google.com/p/mockito/

[Morandi-1]
Publicatie Internet (webpagina)
Datum geraadpleegd 05-07-2012
Auteur Olivier Morandi
Titel Testing Titanium Mobile Applications With Jasmine and Sinon.
Bron http://titaniumninja.com/testing-titanium-mobile-applications-with-jasmine-and-sinon-part-i/

[Niezink]
Publicatie Internet (Onderzoeksrapport)
Datum geraadpleegd 1-05-2012
Auteur Tine Niezink (Telecompaper)
Titel Dutch Smartphone User - Q2 2011
Bron http://www.telecompaper.com/research/dutch-smartphone-user-q2-2011

[Norman]
Publicatie Internet (artikel)
Datum geraadpleegd 1-05-2012
Auteur Jeremy Norman
Titel The first access to the Mobile Web (1996)
Bron http://www.historyofinformation.com/index.php?id=2421

[Objectiveclipse]
Publicatie Internet (webpagina)
Datum geraadpleegd 27-06-2012
Auteur N.N.
Titel Objectiveclipse
Bron http://code.google.com/p/objectiveclipse/

[OCMockito]
Publicatie Internet (webpagina)
Datum geraadpleegd 27-06-2012
Auteur N.N.
Titel OCMockito
Bron https://github.com/jonreid/OCMockito

[Oracle]
Publicatie Internet
Datum geraadpleegd 25-06-2012
Auteur Oracle
Titel Java EE 6 tutorial
Bron http://docs.oracle.com/javaee/6/tutorial/doc/bnbpz.html

[ORMLite]
Publicatie Internet (webpagina)
Datum geraadpleegd 25-06-2012
Auteur ORMLite
Titel SQLite Java Android ORM
Bron http://ormlite.com/sqlite_java_android_orm.shtml

[Pala]
Publicatie Internet (webpagina)
Datum geraadpleegd 03-07-2012
Auteur Margus Pala
Titel PhoneGap performance measurement results
Bron http://marguspala.com/phonegap-performance-measurement-results/

[Papandrew]
Publicatie Internet (webpagina)
Datum geraadpleegd 17-07-2012
Auteur David Papandrew
Titel Learning Corona
Bron http://www.learningcorona.com/

[Peiris]
Publicatie Internet (webpagina)
Datum geraadpleegd 11-07-2012
Auteur Navin Peiris
Titel Appcelerator Titanium JavaScript code completion in IntelliJ
Bron http://navinpeiris.com/2011/10/17/appcelerator-titanium-javascript-code-completion-in-intellij/

[PGSQL]
Publicatie Internet (webpagina)
Datum geraadpleegd 25-06-2012
Auteur N.N.
Titel Getting started with PGSQL
Bron http://www.postgresqlformac.com/data_access_2/getting-started-pgsqlkit-on.html

[PhoneGap-1]
Publicatie Internet (webpagina)
Datum geraadpleegd 03-07-2012
Auteur Nitobi Software Inc.
Titel PhoneGap
Bron https://www.phonegap.com/

[PhoneGap-2]
Publicatie Internet (webpagina)
Datum geraadpleegd 03-07-2012
Auteur Nitobi Software Inc.
Titel FAQ
Bron https://build.phonegap.com/faq

[PhoneGap-3]
Publicatie Internet (webpagina)
Datum geraadpleegd 04-07-2012
Auteur PhoneGap
Titel Command Line Usage
Bron http://docs.phonegap.com/en/1.9.0/guide_command-line_index.md.html#Command-Line%20Usage

[PhoneGap-4]
Publicatie Internet (webpagina)
Datum geraadpleegd 16-07-2012
Auteur PhoneGap
Titel Tools
Bron http://phonegap.com/tools

[Selenium]
Publicatie Internet (Webpagina)
Datum geraadpleegd 25-06-2012
Auteur N.N.
Titel Selenium
Bron http://seleniumhq.org/

[Sente]
Publicatie Internet (webpagina)
Datum geraadpleegd 25-06-2012
Auteur Sente
Titel OCUnit
Bron http://www.sente.ch/software/ocunit/

[Spring-1]
Publicatie Internet (webpagina)
Datum geraadpleegd 27-06-2012
Auteur N.N.
Titel Spring
Bron http://www.springsource.org/

[Spring-2]
Publicatie Internet (webpagina)
Datum geraadpleegd 27-06-2012
Auteur Spring
Titel Spring for Android
Bron http://www.springsource.org/spring-android/

[Square]
Publicatie Internet (webpagina)
Datum geraadpleegd 03-07-2012
Auteur Square Up
Titel iOS Integration Testing
Bron http://corner.squareup.com/2011/07/ios-integration-testing.html

[Sylvanaar]
Publicatie Internet (webpagina)
Datum geraadpleegd 11-07-2012
Auteur Sylvanaar
Titel Lua
Bron http://plugins.intellij.net/plugin/?id=5055

[Taft]
Publicatie Internet (webpagina)
Datum geraadpleegd 16-07-2012
Auteur Darryl K. Taft
Titel PhoneGap Simplifies iPhone, Android, BlackBerry Development
Bron http://www.eweek.com/c/a/Application-Development/PhoneGap-Simplifies-iPhone-Android-BlackBerry-Development-788189/

[TestFlight]
Publicatie Internet (webpagina)
Datum geraadpleegd 25-06-2012
Auteur N.N.
Titel TestFlight
Bron https://www.testflightapp.com/

[Verkuil]
Publicatie Internet (artikel)
Datum geraadpleegd 1-05-2012
Auteur Martijn Verkuil
Titel Apple minder innovatief dan vaak gedacht
Bron http://www.computeridee.nl/nieuws/apple-minder-innovatief-dan-vaak-gedacht

[Von Tetzchner]
Publicatie Internet (nieuwsartikel)
Datum geraadpleegd 1-05-2012
Auteur Jon S. von Tetzchner (Opera Software)
Titel State of the Mobile Web, May 2008
Bron http://www.opera.com/smw/2008/05/

[W3Schools]
Publicatie Internet
Datum geraadpleegd 19-06-2012
Auteur W3Schools
Titel The Internet Explorer browser
Bron http://www.w3schools.com/browsers/browsers_explorer.asp

[Whinnery]
Publicatie Internet (webpagina)
Datum geraadpleegd 18-07-2012
Auteur Kevin Whinnery
Titel Comparing Titanium and PhoneGap
Bron http://developer.appcelerator.com/blog/2012/05/comparing-titanium-and-phonegap.html

[Wilson]
Publicatie Internet
Datum geraadpleegd 27-06-2012
Auteur Jesse Wilson
Titel Mockito
Bron http://code.google.com/r/jessewilson-mockito/source/detail?r=805e98fd4190acbbdaab87172c31b8dc0b78236e

[Xamarin]
Publicatie Internet (webpagina)
Datum geraadpleegd 10-07-2012
Auteur Xamarin
Titel MonoTouch
Bron http://xamarin.com/monotouch/

Bijlage G – Tussentijdse beoordeling concept

Student: Wilco Stuyfersant **Studentnummer:** 08024197

Datum: 13 juli 2012

Tijdens de bespreking is het volgende geconstateerd:		ja	nee
a	*Het voortgangsverslag is ontvangen*		X
b	*Het afstudeerdossier is digitaal beschikbaar*	X	
c	*Het afstudeerdossier is opgebouwd conform de richtlijnen*	X	
d	*Het goedgekeurde afstudeerplan is aanwezig*	X	
e	*Het plan van aanpak is aanwezig*	X	
f	*Reeds geleverd commentaar is aanwezig*		X
g	*Het afstudeerdossier geeft voldoende inzicht in de stand van zaken*	X	
h	*De afstudeeropdracht is tot nu toe naar behoren uitgevoerd*	X	X

Verbeterpunten:

Er is achterstand ontstaan in de op te leveren producten door nieuwe inzichten van de opdrachtgever

Opmerkingen:

Er wordt gedurende de zomer doorgewerkt waardoor waarschijnlijk de achterstand ingelopen kan worden.

Naam begeleidend examinator: Nanny Jacobs

Datum:13-7-2012

Dit formulier wordt door de begeleidend examinator digitaal ingevuld en per email naar de student verstuurd met een cc naar de coördinator van ICT & Media @ Work (A.M.Schipper@hhs.nl). Het formulier dient door de student te worden opgenomen in het afstudeerdossier.

Bijlage H – Tussentijdse beoordeling assessment

Student: Wilco Stuyfersant **Studentnummer:** 08024197

Datum: 31 augustus 2012 **eerste / tweede TTA:** eerste

Tijdens het tussentijds assessment is het volgende geconstateerd:		**ja**	**nee**
a	*Het voortgangsverslag is ontvangen*	x	
b	*Het afstudeerdossier is digitaal beschikbaar*	x	
c	*Het afstudeerdossier is opgebouwd conform de richtlijnen*	x	
d	*Het goedgekeurde afstudeerplan is aanwezig*	x	
e	*Het plan van aanpak is aanwezig*	x	
f	*Reeds geleverd commentaar is aanwezig*		
g	*Het afstudeerdossier geeft voldoende inzicht in de stand van zaken*		x
h	*De afstudeeropdracht is tot nu toe naar behoren uitgevoerd*	x	

Aanpak	**O**	**T**	**V**	**G**
Passend		x		
Theoretisch verantwoord		x		
Samenhang uitvoering beroepstaken		x		

Beroepstaken op afgesproken niveau uitgevoerd?		**O**	**T**	**V**	**G**
1	Uitvoeren analyse door definitie van requirements		x		
2	Ontwerpen softwarearchitectuur		x		
3	Bouwen applicatie		x		
4	Uitvoeren van en rapporteren over het testproces		x		
5					
6					
7					

Producten	O	T	V	G
Tussenproducten		x		
Eindproducten	x			

Effectief communiceren	O	T	V	G
Binnen afstudeerbedrijf			x	
Afstudeerdossier		x		

Reflectie	O	T	V	G
Inzicht in eigen functioneren		x		
Inzicht in eigen leerproces		x		

Toelichting per beoordelingscriterium

Aanpak
De aanpak wordt in het verslag niet voldoende beschreven. Het verslag geeft geen inzicht of de gemaakte keuzes goed zijn.

Beroepstaken op afgesproken niveau uitgevoerd?
De beroepstaken komen niet voldoende aan bod in het verslag om hier een voldoende voor te geven. Het verslag geeft met name een beschrijving van de subjectieve ervaringen van de student in zijn omgeving, maar laat nog onvoldoende de aankomende professionele Software Engineer zien.

Producten
De producten in de bijlage geven aan dat de omvang van het werk voldoende is. Ook wordt zichtbaar dat er een zekere mate van complexiteit zichtbaar.

Effectief communiceren
Het lijkt erop dat er wel voldoende is gecommuniceerd met het bedrijf. In het afstudeer dossier is de aansturing vanuit de organisatie en het bijbehorende validatieproces deels zichtbaar. Het is nog onduidelijk of dit ook efficiënt is geweest

Reflectie
Wilco heeft tijdens het gesprek wel voldoende inzicht gegeven in zijn eigen functioneren, in de gemaakte keuzen en onderbouwing. In het verslag zelf komt het nog niet aan bod.

Advies

X	Positief **(bindend advies)**
	Verlengen **(vrijblijvend advies)**
	Negatief **(vrijblijvend advies)**

Besluit student

Aankruisen welke beslissing de student heeft genomen (alleen na vrijblijvend advies)

x	**Afstudeerdossier wordt op afgesproken datum ingeleverd**	Inleverdatum: 24-09-2012
	Afstudeerperiode wordt verlengd	Inleverdatum:
	Student stopt met afstudeeropdracht	

Naam begeleidend examinator: Nanny Jacobs

Naam tweede examinator: Arno Nederend

Datum: 3 september 2012

Dit formulier wordt door de tweede examinator digitaal ingevuld, waarna de begeleidend examinator het per email verstuurt naar de student met een cc naar de coördinator van ICT & Media @ Work (A.M.Schipper@hhs.nl). Het formulier dient door de student te worden opgenomen in het afstudeerdossier.

Bijlage I – Evaluatie bedrijfsmentor

DE HAAGSE HOGESCHOOL

Academie voor ICT & Media

Den Haag
Delft
Zoetermeer

Evaluatieformulier afstuderen

In te vullen door opdrachtgever c.q. bedrijfsmentor(en)

Student: Wilco Stuyfersant

Periode: mei tot en met september 2012

Bedrijf c.q. instelling: 42 B.V.

Bedrijfsmentor: Robert Bor

Plaats: Zoetermeer

Datum: 23 september 2012

1. **Heeft de student zich zelf snel en goed ingewerkt in het bedrijf en de uit te voeren afstudeeropdracht?**

Wilco heeft zich goed ingewerkt in de materie. Hij heeft zich een beeld gevormd van alle soorten app typen en deze op duidelijke wijze inzichtelijk gemaakt.

2. **Hoe beoordeelt u de communicatieve vaardigheden van de student (in de samenwerking met collega's, in contacten met de opdrachtgever, bij mondelinge presentaties, schriftelijke rapportages)?**

Wilco is goed in staat zichzelf uit te drukken en te communiceren, zowel mondeling als verbaal. Als teamleider doet hij het prima, zeker gegeven zijn leeftijd. Hij werd in deze rol soepel geaccepteerd door zijn collega's.

3. Hoe heeft de student tijdens het uitvoeren van de opdracht gefunctioneerd?

- **Qua verantwoordelijkheid** ~~goed / voldoende~~ / **matig** / ~~onvoldoende~~
- **Qua zelfstandigheid** ~~goed~~ / **voldoende** / ~~matig / onvoldoende~~
- **Qua planmatig werken** **goed** / ~~voldoende / matig / onvoldoende~~
- **Qua creativiteit** ~~goed / voldoende~~ / **matig** / ~~onvoldoende~~
- **Qua productiviteit** ~~goed~~ / **voldoende** / ~~matig / onvoldoende~~
- **Qua samenwerken met collega's** **goed** / ~~voldoende / matig / onvoldoende~~
- **Qua draagvlakontwikkeling** ~~goed~~ / **voldoende** / ~~matig / onvoldoende~~
- **Qua inspelen op bedrijfscultuur** ~~goed~~ / **voldoende** / ~~matig / onvoldoende~~
- **Qua rekening houden met de specifieke context van het bedrijf** ~~goed~~ / **voldoende** /~~ matig / onvoldoende~~
- **Qua het op gang brengen van de nodige veranderingen** ~~goed~~ / **voldoende** / ~~matig / onvoldoende~~

4. Hoe beoordeelt u de kennis en kunde van de student in verhouding tot wat u verwacht van een bijna afgestudeerde?

Wilco is sterk in zijn planmatige kant en zijn communiceren. Qua rol is WIlco beter voor te stellen als manager, projectleider of consultant dan als ontwikkelaar, een rol waar hij ogenschijnlijk weinig affiniteit mee lijkt te hebben.

5. Hoe beoordeelt u de kwaliteit van de opgeleverde (tussen)producten?

Het onderzoeksrapport voldoet aan onze verwachtingen. Ook de bijdrage die Wilco heeft geleverd aan het Goudvink team was ruim voldoende, vooral in de inter-persoonlijke sfeer. Helaas is geen werkend prototype van Phonegap opgeleverd.

6. Bent u tevreden over het opgeleverde (eind)product?

- **In hoeverre heeft u gekregen wat is afgesproken?**

Ik ben tevreden over het rapport en de bijdrage van Wilco aan het Goudvink team. Het niet opleveren van Phonegap applicatie (vanwege het gedoe met het eindverslag) vind ik zeer betreurenswaardig.

- **In hoeverre voldoet het (eind)product aan uw verwachtingen?**

Het rapport was duidelijk en hierin is goed rekening gehouden met onze wensen en de specifieke achtergrond/cultuur van 42.

- **Wat is de bruikbaarheid en onderhoudbaarheid hiervan?**

Prima, in combinatie met een nog uit te voeren marktonderzoek en het ontwikkelen van een aantal prototypes om een hands-on gevoel voor de technologie te krijgen.

- **Wat gebeurt er met het opgeleverde (eind)product?**

De conclusie van het rapport wordt gebruikt voor de volgende fase, het marktonderzoek. Op basis hiervan gaan we bepalen welke kant we opgaan met mobile.

- **Kunt u direct met het opgeleverde product aan de slag?**

Nee, zie boven.

7. Zijn er nog aspecten voor u van belang die nog niet aan de orde zijn geweest?

In het begin waren er wat probleempjes tussen Wilco en een aantal mensen in de organisatie. We hebben deze zaken besproken en voor mijn gevoel heeft Wilco echt wat gedaan met deze feedback. Mijn complimenten hiervoor.

8. Bent u bereid een volgende keer weer uw medewerking te verlenen aan het beschikbaar stellen van een afstudeerplaats (graag met toelichting)**?**

Jazeker.