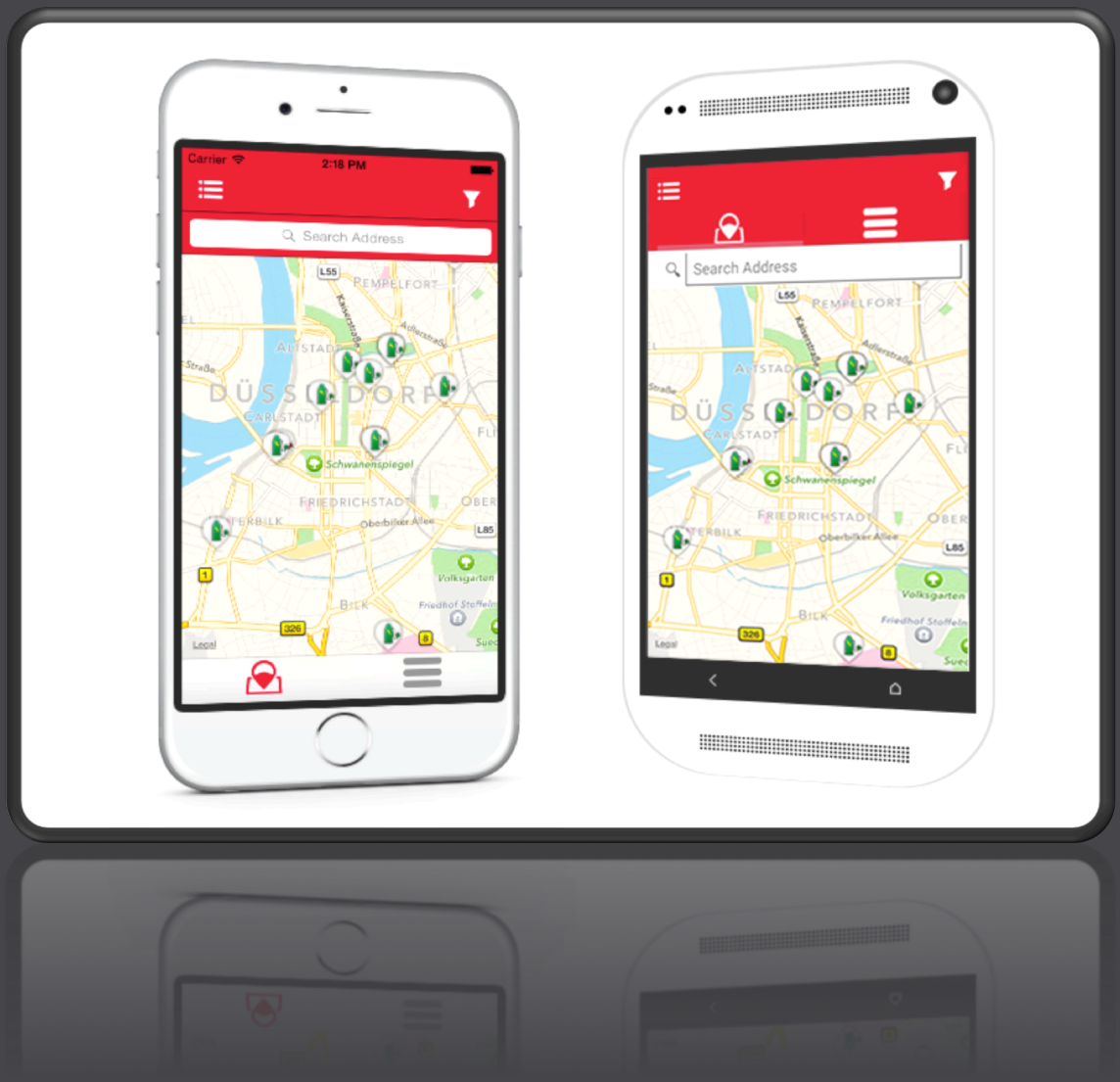
DE HAAGSE HOGESCHOOL

CGI

AFSTUDEERVERSLAG

Ontwikkelen van een mobiele cross-platform applicatie voor het vinden en reserveren van oplaadpunten bij CGI



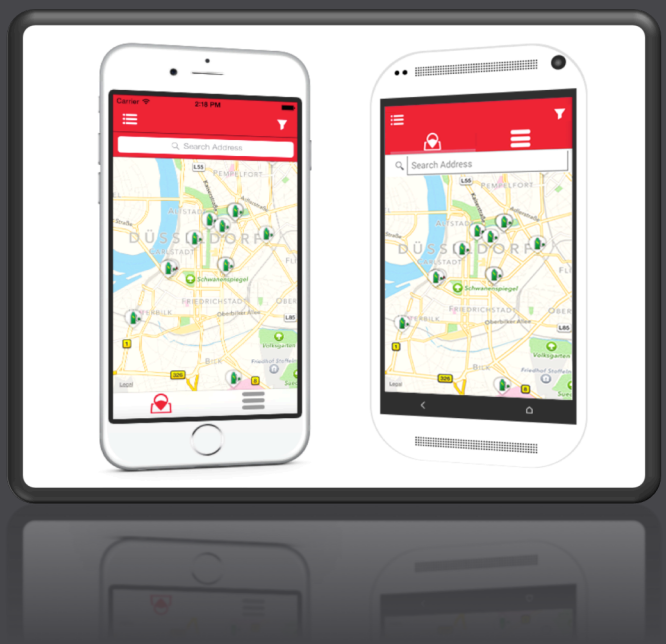
J.M.N. Vrolijk

DE HAAGSE HOGESCHOOL

CGI

AFSTUDEERVERSLAG

Ontwikkelen van een mobiele cross-platform applicatie voor het vinden en reserveren van oplaadpunten bij CGI



12060534
J.M.N. Vrolijk
Student Informatica aan de Haagse Hogeschool

CGI - OU Communications & Utilities
Den Haag – 02-10-2015

Referaat

Vrolijk, Justin

Ontwikkelen van een mobiele cross-platform applicatie voor het vinden en reserveren van oplaadpunten bij CGI.
Rotterdam, CGI, 2015

Afstudeerverslag van Justin Vrolijk, geschreven in het kader van het afstuderen bij de opleiding Informatica aan de academie voor ICT & Media aan de Haagse Hogeschool.

Het verslag behandelt het (ontwikkel)proces dat is doorlopen tijdens de afstudeerperiode van Justin Vrolijk bij CGI te Rotterdam. Deze opdracht stond in het kader van het ontwikkelen van een mobiele cross-platform applicatie voor de bestuurders van een elektrisch voertuig. De afstudeerder moest een geschikt platform selecteren en door middel van een scrumachtige methodiek toewerken naar een demonstreerbare versie.

Descriptoren:

- Afstudeeropdracht
- Mobiele cross-platform applicatie
- iOS
- Android
- Xamarin
- PhoneGap
- C#
- .NET 4.5
- Model View Controller (MVC)
- JSON
- XML
- Scrum
- Webservices

Voorwoord

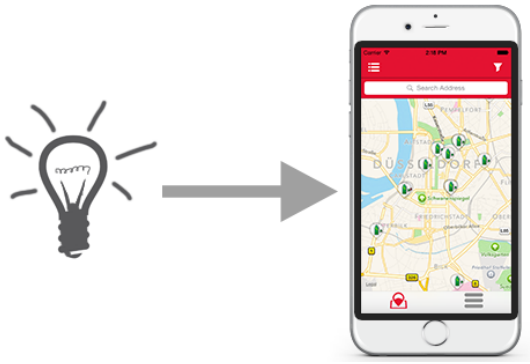
Mijn naam is Justin Vrolijk. Ik volg de opleiding Informatica aan de Haagse Hogeschool te Den Haag. Mijn affiniteit met techniek heeft geresulteerd in deze studiekeuze. Studenten die deze opleiding volgen, ronden hun studie af met een afstudeeropdracht. Mijn vorige stage-opdracht heb ik uitgevoerd binnen een kleine organisatie. Als afsluiting op deze opleiding wou ik graag afstuderen binnen een grote organisatie. Nederland kent een aantal grote IT-gerelateerde organisaties dat werkzaam is in meerdere landen. CGI kende ik via een familielid dat daar in het verleden heeft gewerkt. Daarbij is het kantoor, voor mij, gunstig gevestigd. Ik besloot om contact op te nemen met CGI om de mogelijkheden te bespreken. Met de heer Jos Siemons kwam ik in gesprek en hij bood mij een afstudeeropdracht aan.

Ik had mijn scriptie niet op deze manier af kunnen ronden als ik niet zo goed begeleid was door een aantal personen: E.M. van Doorn en O. Zor van de Haagse Hogeschool en Rémon van den Beemt en Franc Buve van CGI. De heren van Doorn en Zor wil ik bedanken voor de begeleiding door het gehele proces heen, het inzicht, de kennis, de vriendelijkheid en de leuke, opbouwende gesprekken over mijn afstudeerverslag. Rémon en Franc, bedankt voor de goede en leuke begeleiding bij CGI, bedankt voor het feit dat jullie altijd klaar stonden om vragen te beantwoorden, mee te denken en te helpen. Uw scherpheid en uw feedback op mijn stukken. Ook een dankbetuiging naar de heer Jos Siemons. Zonder u had ik niet aan deze opdracht kunnen beginnen. Bedankt dat u mij een afstudeeropdracht hebt aangeboden binnen CGI.

Justin Vrolijk - Den Haag, september 2015

Inhoudsopgave

1 Inleiding



Hoe ontwikkelt een idee zich tot een mobiele app? In dit verslag wordt dat proces beschreven. Ik ga een afstudeeropdracht uitvoeren binnen CGI. Ik ga een mobiele cross-platform applicatie ontwikkelen voor Android en iOS. Met deze app kunnen bestuurders van elektrische auto's gemakkelijk oplaadpunten vinden en reserveren. Met bestaande apps kunnen deze al gelokaliseerd worden, maar doordat een koppeling met het platform voor laadpalen ontbreekt, zijn geavanceerde opties zoals reserveren tot op heden niet mogelijk.

Het verslag is opgedeeld in drie delen. In het eerste deel wordt de context en opdrachtomschrijving besproken. U komt meer te weten over het afstudeerbedrijf CGI en het team. Ook krijgt u verdiepende informatie over het CiMS platform; het platform dat in meerdere landen gebruikt wordt om alle laadpalen op afstand te beheren. Vervolgens wordt de opdracht omschreven zoals het in het afstudeerplan afgesproken is.

Deel twee beschrijft de aanpak en uitvoering. Een goede aanpak is essentieel om een project succesvol af te ronden. In dat hoofdstuk licht ik toe hoe Scrum is toegepast binnen dit project. De software ontwikkelmethodiek moest enigszins aangepast worden omdat het project zelfstandig wordt uitgevoerd. Met deze aanpak wordt vervolgens in iteraties de app ontwikkeld. Het proces vanaf de druk op 'Create new project' knop tot de oplevering van de app, wordt beschreven. Ideeën van stakeholders worden vertaald naar functionaliteiten. Daarbij kunnen problemen ontstaan. Deze problemen worden toegelicht en opgelost. Daarna wordt de app op performance en security getest. Met zelfgeschreven code worden metingen uitgevoerd op het 3G- en 4G netwerk. Vervolgens worden de datapakketen bekeken die over het net worden verstuurd. Met encryptie en decryptie wordt data beveiligd.

In deel drie wordt het resultaat getoond. Verder wordt het doorlopen proces geëvalueerd. Het doel dat geformuleerd is, is behaald. Een app is opgeleverd met meer functionaliteiten dan aanvankelijk was afgesproken. CGI was positief verast over het resultaat. In de evaluatie, gaven ze aan dat ik op sommige punten boven app developer zit die werkzaam zijn in het bedrijfsleven. Ook wordt de evaluatie besproken. Het geeft mijn leerproces weer: wat ging goed en wat ging niet goed? Wat zou ik in het vervolg anders doen?

Deel I. Context en opdrachtomschrijving

2 De organisatie

Hoofdstuk twee gaat over de organisatie. In paragraaf 2.1 wordt de oprichting, activiteiten en organigram weergeven. Hierna wordt hun probleemstelling met betrekking tot de opdracht omschreven. Tot slot wordt het team dat mij begeleidt, geïntroduceerd in paragraaf 2.3.

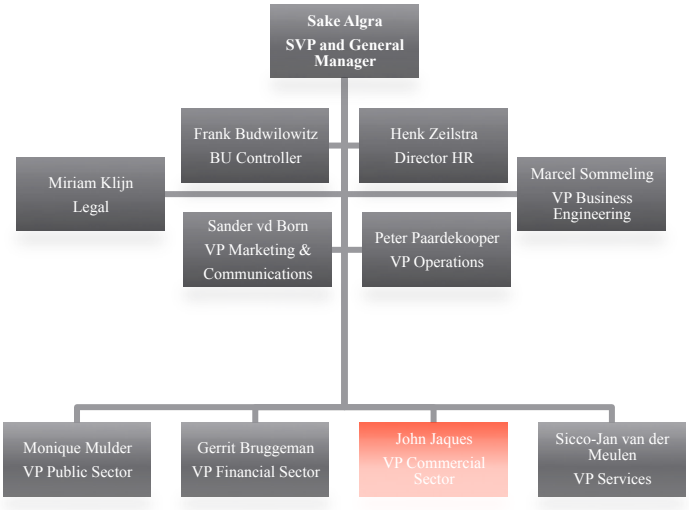
Figuur 2.1 **Hoofdkantoor CGI in Rotterdam**

2.1 CGI

CGI is opgericht in 1976 vanuit de kelder van Serge Godin. Serge heeft dit in samenwerking met André Imbeau gedaan. Dit bedrijf is uitgegroeid tot de op vier na grootste onafhankelijke zakelijke en IT-dienstverlener van de wereld. Ze zijn actief in 40 landen, met ruim 68.000 professionals, gespecialiseerd in business consulting, systeemintegratie en IT-outsourcingdiensten.

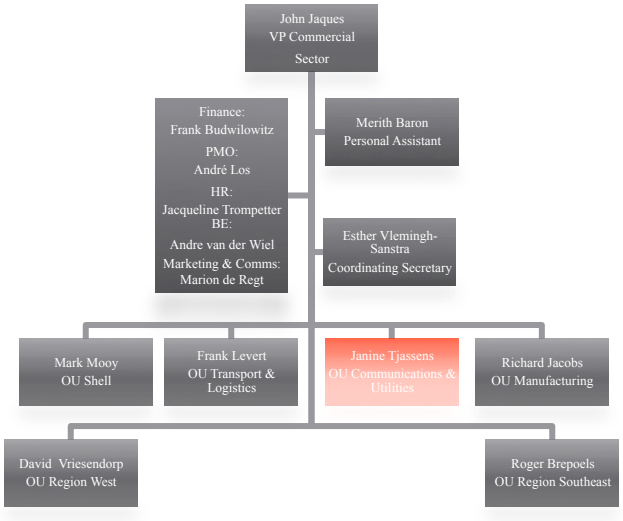
Wereldwijd mogen ze de top 100 bedrijven tot hun klanten rekenen. In Nederland heeft CGI ongeveer 3.500 medewerkers: ze worden members genoemd. Zij zijn actief bij alle grote bedrijven en overheidsinstellingen in Nederland. Het dienstenaanbod van CGI bestaat uit een uitgebreid pakket aan oplossingen. Daardoor zijn ze actief in diverse markten, onder andere: communicatie, energie, financiële dienstverlening, olie en gas, overheid, post en logistiek, productie, zorg en transport.

De bedrijfscultuur is door de omvang van het bedrijf aanzienlijk groot. Het managementteam stuurt meerdere sectoren aan. Het organigram is te zien in Figuur 2.2. Iedere sectorbestuurder is voorzien van zijn eigen team dat op hen beurt ook zogeheten OU's (operatie units) aanstuurt. Elke OU heeft een managementteam met teamleiders. De teamleiders sturen de members aan en zijn hun eerste aanspreekpunt.

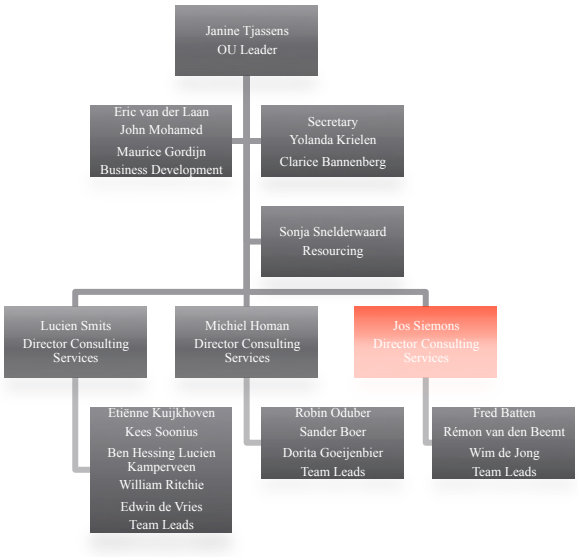


Figuur 2.2 **Organigram CGI Management team Nederland**

De afstudeeropdracht wordt gemaakt voor de heer Jos Siemons. Hij is directeur consulting services binnen Communicatie en Utilities OU. Deze OU wordt bestuurd door mevrouw Jannine Tjansen. Zij neemt deel aan het managementteam in de commerciële sector. De onderstaande twee organigrammen (Figuur 2.3 en 2.4) verduidelijken dit beeld.



Figuur 2.3 **Organigram Commercial Sector**



Figuur 2.4 **Organigram OU Communications & Utilities**

2.2 Probleemstelling

CGI ontwikkelde als eerste een infrastructuur voor de oplaadpunten van elektrische auto's. Zij hebben dit systeem Charge point interactive Management System (CiMS) genoemd. Met dit systeem kunnen laadpunten beheerd en laadtransacties verwerkt en verrekend worden. Doordat het systeem met open standaarden werkt, kunnen verschillende soorten oplaadpunten worden aangesloten. Hierdoor kunnen meerdere netbeheerders gebruik maken van CiMS. Dit platform wordt verder toegelicht in paragraaf 3.1.

In Nederland zijn meer dan 3.000 oplaadpunten verbonden met CiMS. Deze worden beheerd door ElaadNL. Daarnaast wordt dit systeem gebruikt in meerdere landen zoals: Duitsland, België, Luxemburg, Noorwegen en Zweden.

Veel mobiele apps hebben deze oplaadpunten in kaart gebracht. Omdat de communicatie met CiMS ontbreekt, is het alleen mogelijk om ze te lokaliseren. CGI heeft van meerdere klanten die gebruik maken van deze infrastructuur, de vraag gehad of zij een app hebben die niet alleen oplaadpunten kan tonen, maar ook kan communiceren met CiMS. Hierdoor zou het mogelijk zijn om bijvoorbeeld oplaadpunten te reserveren of een transactie via de telefoon te starten en stopen. Tevens zou de huidige status van het oplaadpunt opgevraagd kunnen worden. Daarnaast wil CGI graag aantonen en onderzoeken hoe het systeem inspeelt op de huidige mobiele markt.

2.3 Begeleidingsteam

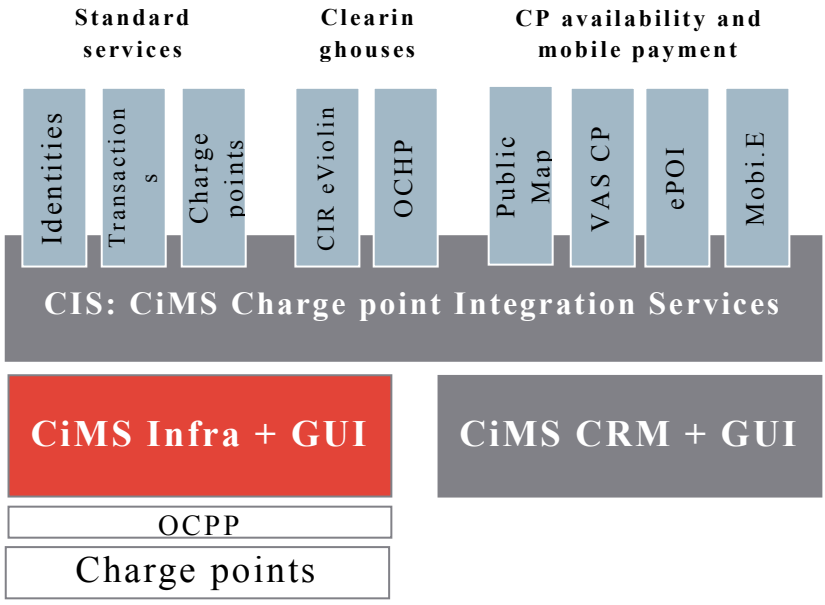
Franc Buve, Software Architect, is het eerste aanspreekpunt voor vragen en beslissingen over de opdracht. Hij bewaakt de voortgang van het project en levert feedback op de resultaten. Rémon van den Beemt , Technical Project Leader, kan benaderd worden met niet-opdracht gerelateerde vragen. Hij begeleidt mij bij aanvang van het project. Nghia Le, Principal IT Architect, en Yard Nijsten, Software Engineer, zijn bereikbaar voor technische problemen. Vragen die betrekking hebben op de gebruikersinterface kunnen gesteld worden aan Chris van Schayk, Software Engineer.

3 Werkomgeving

In paragraaf 3.1 wordt toegelicht wat de CiMS infrastructuur kan en doet. In paragraaf 3.2 staat de systeemontwikkelomgeving bij CGI centraal. Tot slot wordt in paragraaf 3.3 het platform PhoneGap aan bod. Daarin wordt uitgelegd wat PhoneGap is en wat ermee gedaan kan worden.

3.1 CiMS

CiMS is een infrastructuur voor het verbinden en op afstand beheren van oplaadpunten. De verbinding met diverse oplaadpunten is mogelijk door het Open Charge Point Protocol (OCPP). Ook kunnen transacties verwerkt worden en zijn een tal van andere services verbonden met CiMS. Deze zijn verbonden door het CIS platform. CIS is een open interface laag waarmee services verbonden worden met CiMS. Daarnaast is het CRM platform. Dat systeem beheert de kaart- en klantgegevens. Deze communiceert direct met CiMS en stuurt updates naar CIS. In Figuur 3.1 is CiMS schematisch weergegeven.



Figuur 3.1 : **Schematische weergave van CiMS, CIR en CRM (Bron: http://www.cginederland.nl)**

OCPP is een open standaard voor de communicatie tussen oplaadpunten en een centraal systeem. De standaard maakt gebruik van Simple Object Access Protocol (SOAP). Hiermee kunnen berichten tussen twee componenten over het internet verstuurd worden. CiMS gebruikt momenteel versie 1.5 van OCPP.

Deze versie is de facto standaard wereldwijd. Het protocol wordt in meer dan 50 landen gebruikt bij 10.000 plus oplaadpunten. Het is overgenomen door de Open Charge Alliance (http://www.openchargealliance.org/). Hun missie is om de ontwikkeling, goedkeuring, en de naleving van de communicatie protocollen in de elektrische auto laadinfrastructuur te bevorderen.

CIS is een open interface-laag bovenop de CiMS infrastructuur. Het dient als 'roaming hub' voor andere providers. Een roaming hub verbindt meerdere providers met elkaar. Services communiceren met CiMS via deze laag. Daarnaast is het voor derden mogelijk om bijvoorbeeld nieuwe betaaloplossingen erbij te ontwikkelen.

In CRM staan de gegevens van pashouders en hun transacties. Met dit systeem kunnen de pashouders beheert worden. Daarnaast kunnen zij zelf inloggen op dit systeem om hun eigen transactiehistorie te zien en om gegevenswijzigingen door te voeren.

3.2 Systeemontwikkelomgeving

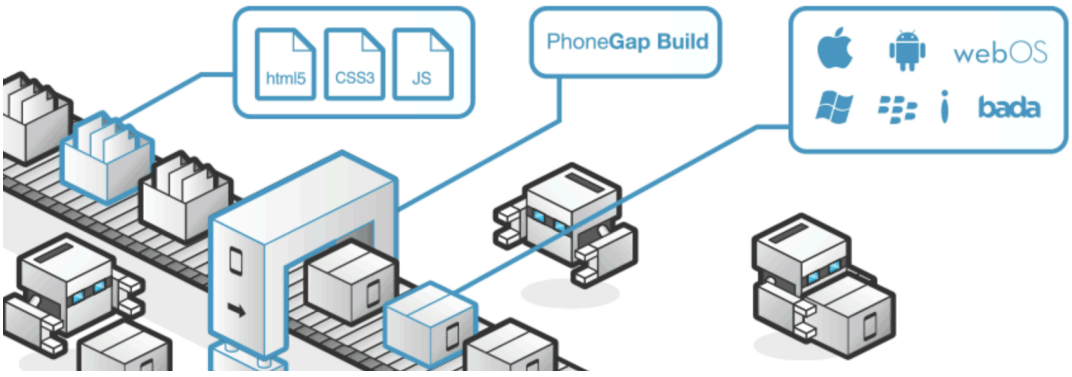
Eén van de software ontwikkelmethodes die CGI gebruikt, is Scrum. Scrum is een iteratieve ontwikkelmethode. In zogeheten sprints worden user story's - vergelijkbaar met requirements - geïmplementeerd. Een sprint duurt twee tot vier weken. Aan het eind van die periode wordt een werkend stuk software opgeleverd. Stakeholders kunnen hierdoor in een vroeg stadium de voortgang bekijken. Zij worden nauw betrokken bij het proces omdat ze sturing kunnen geven bij de vervolg sprints.

Volgens Schwaber en Sutherland (2010, p. 5) bestaat een Scrum Team uit 'Scrum Master', 'Product Owner' en het team. De Scrum Master bewaakt het scrumproces. Een Product Owner beheert de 'Product Backlog'. Het Product Backlog bestaat uit een lijst met requirements. En tot slot het team dat bestaat uit ontwikkelaars. Zij gebruiken het Product Backlog om story's te selecteren voor de komende sprint.

CGI gebruikt normaliter sprints die twee weken duren. Bij aanvang van elke sprint wordt met het team en de Product Owner een 'grooming' sessie gehouden. De geprioriteerde user story's worden geselecteerd en geanalyseerd. Hierna kan het team bepalen welke story's worden opgenomen in de sprint.
Het scrumproces wordt begeleid door de Scrum Master. Het is zijn taak om ervoor te zorgen dat de scrumprocessen correct worden uitgevoerd. Sprint review en een 'Daily Scrum' zijn voorbeelden van scrumprocessen. Teams houden dagelijks een 'stand up' waar zij drie vragen staand beantwoorden: Wat heb ik gister bereikt? Wat ga ik vandaag doen en welke uitdagingen verwacht ik?
De Product Owner beheert het Product Backlog. Hij is verantwoordelijk voor de opgeleverde producten. Ook is hij degene die contact heeft met de stakeholders. Het is zijn taak om de wensen en eisen in kaart te brengen. Deze neemt hij op in het Product Backlog.

3.3 PhoneGap

Een eis, die beschreven is in hoofdstuk vier, is dat de applicatie gebouwd wordt met PhoneGap. Het is een platform waarmee mobiele cross-platform applicaties ontwikkeld worden met behulp van HTML5, JavaScript en CSS. De ontwikkelaar bouwt in feite een website die gepresenteerd wordt in een native app; ook wel een hybride app genoemd. Deze is voorzien van een browser. PhoneGap bouwt de app met browser. De ontwikkelaar hoeft alleen de 'website' te ontwikkelen.
Daardoor hoeft de ontwikkelaar geen kennis te hebben van de platform specifieke ontwikkeltalen. Een bijkomend voordeel is dat de app eenmalig gemaakt hoeft te worden. Met minieme aanpassingen kan het uitgerold worden naar platformen zoals: iOS, Android en Windows Phone. Het proces van een website (bestaande uit HTML5, CSS en JavaScript) is weergegeven in Figuur 3.2.



Figuur 3.2 **: De werking van PhoneGap (Bron: http://phonegap.com/about)**

PhoneGap blijkt een succesvol platform te zijn om hybride applicaties te ontwikkelen. Zo blijkt uit een artikel op PhoneGap: *"PhoneGap has been downloaded over 1 million times and is being used by over 400,000 developers. Thousands of apps built using PhoneGap are available in mobile app stores and directories."* ("About the Project", z.j.)

4 Opdrachtomschrijving

In hoofdstuk 4 wordt de aanvankelijke opdrachtomschrijving beschreven. Deze komt overeen met het afstudeerplan dat opgenomen is als bijlage (Bijlage I). In paragraaf 4.1 wordt de probleemstelling uit het afstudeerplan beschreven. Paragraaf 3.2 eindigt met de doelstelling.

4.1 Het afstudeerplan

In het afstudeerplan is beschreven waarom CGI dit project wil laten uitvoeren en hoe dit gedaan gaat worden. Belangrijke onderdelen van het plan worden hieronder herhaalt en waar nodig toegelicht.

4.2 Probleemstelling

Derde partijen hebben apps ontwikkeld waar oplaadpunten in kaart zijn gebracht. Het is niet mogelijk om daarmee laadpalen te reserveren of de beschikbaarheid te controleren omdat integratie met CiMS ontbreekt. De klanten willen graag een app die specifiek voor dit systeem is gebouwd met hun eigen branding. Een app die geïntegreerd is met CiMS door een interface, maakt het mogelijk om bijvoorbeeld laadpalen te reserveren of de beschikbaarheid te tonen.

De app moet kunnen communiceren met CiMS om zulke functionaliteiten te realiseren. Het systeem heeft op dit moment nog geen interface die communicatie met mobiele telefoons mogelijk maakt. Een generieke REST API maakt integratie van mobiele telefoons met het systeem mogelijk.

Bovendien moeten de app en interface voldoen aan de volgende punten:

- de communicatie tussen de interface en de app moet beveiligd zijn;
- de app moet gemakkelijk te gebruiken zijn en professioneel ogen;
- responsetijd moet op het 3G en 4G netwerk snel zijn;
- binnen een dag moet klantenbranding toegepast kunnen zijn op de app.

Daarnaast hoopt CGI meer klanten te krijgen door het systeem uit te breiden. Een interface voor communicatie met mobiele telefoons en een White-label app kan daaraan bijdragen. Het doel is om een mobiele cross-platform applicatie te ontwikkelen met behulp van PhoneGap, die aansluit op de wensen van bestuurders van elektrische auto's.

4.3 Doelstelling

De doelstelling is het ontwikkelen van een mobiele cross-platform applicatie met behulp van PhoneGap, die aansluit op de wensen van bestuurders van elektrische auto's. Het zal een overzicht tonen van beschikbare oplaadpalen en een reservering daarvan mogelijk maken. Daarnaast zal de werking van de CiMS interface voor mobiele apps gedemonstreerd kunnen worden.

Tevens kan de app-gebruiker met de app: laadpalen markeren als favoriet, zoeken naar specifieke laadpalen, waarderingen geven en melding maken van onjuiste informatie. Bovendien ligt de nadruk van de app en interface op de volgende kwaliteitseisen: beveiligbaarheid, gebruikersgemak, aantrekkelijkheid, snelheid en wijzigbaarheid.

4.3.1 Het beoogde resultaat

Het resultaat is een White-label mobiele cross-platform applicatie die ontwikkeld is voor iOS en Android. Door middel van een REST API wordt gecommuniceerd met CiMS. Deze API is in samenwerking met een ontwikkelaar van CGI gedefinieerd, maar door CGI gebouwd.

De app kan in deze staat gedemonstreerd worden aan de klant. Met minieme moeite kan binnen een dag klantenbranding toegevoegd worden. De app-gebruiker kan hiermee gemakkelijk en snel oplaadpunten vinden en reserveren. Ook is het volgende mogelijk in de app:

- het markeren van laadpalen als favoriet;
- zoeken naar laadpalen die geschikt zijn voor hun type auto;
- een waardering geven aan een laadpaal;
- het melden van onjuiste informatie.

Tevens is de app gemakkelijk te gebruiken en heeft het een heldere en professionele uitstraling. Op zowel het 3G en 4G netwerk is de responsetijd snel. Ook is de communicatie tussen de app en CiMS is beveiligd zodat ongewilde aanvragen en acties voorkomen worden.

Deel II. Aanpak en uitvoering

5 Aanpak

De gekozen aanpak die in het plan van aanpak staat beschreven, wordt in dit hoofdstuk toegelicht. Het plan van aanpak is opgenomen in bijlage II. In paragraaf 5.1 wordt gekeken of Scrum toepasbaar is binnen dit project. Vervolgens wordt de aanpak met Scrum beschreven. In paragraaf 5.2 komen afspraken die gemaakt zijn op het gebied van: afbakening, risico's en randvoorwaarden aan de orde. In paragraaf 5.3 staat de globale planning voor de uitvoer in hoofdstuk 6.

5.1 Scrum als ontwikkelmethode

De opdrachtgever wenst dat een agile methodiek wordt toegepast. Bij aanvang heeft hij aangegeven dat Scrum gebruikt moet worden. Hij wil gebruik maken van deze methode omdat CGI hiermee bekend is en omdat een online-omgeving klaarstaat. In deze online-omgeving kunnen de scrumprocessen gemakkelijk ingericht worden. Daarnaast beweert hij dat het goed aansluit bij dit project aangezien het een agile aanpak is die weinig overhead oplevert.

Bij mij ontstond de vraag of Scrum wel een bruikbare methode is bij dit project. Ik besluit daarom om te onderzoeken of het wel toepasbaar is. In hoofdstuk vier staan de wensen en eisen die tot zover bekend zijn. Een reeks aan functionele en niet-functionele requirements. Het is duidelijk dat de opdrachtgever een app wil met een aantal functionaliteiten. Het is niet bekend hoe die functionaliteiten, en de app zelf, eruit moet komen te zien. Hij heeft alleen aangegeven dat het professioneel moet ogen en ergonomisch in gebruik moet zijn. Daarnaast ontbreken details aan requirements. Gedurende het project zullen deze waarschijnlijk veranderingen ondergaan. Daaruit kan ik concluderen dat de opdrachtgever nog geen visueel beeld heeft van de app. Een agile methodiek met iteratief karakter is dan aan te raden.

In paragraaf 3.2 staat dat Scrum daaraan voldoet. De Product Owner kan het Product Backlog vullen met de reeds bekende requirements. Hij interviewt de stakeholders om die te vertalen naar user story's en om de prioriteiten te achterhalen. Daar kunnen ook nieuwe story's uit voortvloeien. Per sprint wordt met het team een aantal story's uitgewerkt en geïmplementeerd (vooraf gedetailleerde requirements zijn niet nodig). Hierna ontvangt de opdrachtgever een werkend stuk software. Hij kan dan feedback leveren en gaande weg wordt de applicatie naar zijn wensen en eisen ontwikkeld.

Scrum is in dat opzicht toepasbaar. Het is echter bedoeld voor multidisciplinaire teams. Schwaber en Sutherland (2010, p. 8) schrijft dat scrumteams bestaan uit ongeveer 7 personen. Dat levert een probleem op omdat het project door één persoon wordt uitgevoerd. Een scrumachtige methodiek zou de oplossing zijn. In sub paragraaf 5.1.1 wordt die beschreven.

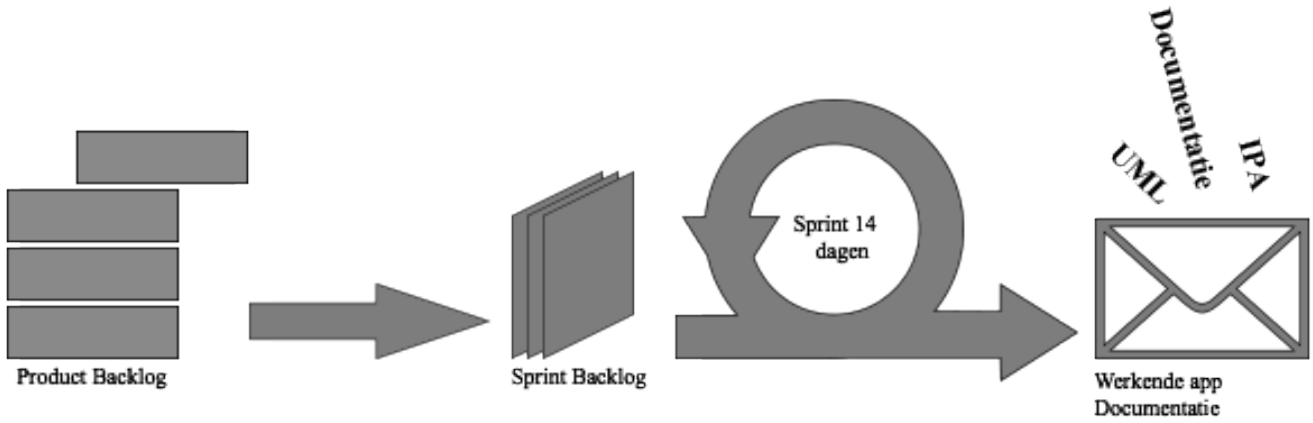
5.1.1 Aanpak voor scrumachtige methodiek

Uit paragraaf 5.1 blijkt dat een scrumachtige methode de oplossing is. Een aantal delen binnen Scrum wordt weggelaten. Ook zal ik meerdere rollen op mij nemen. Het gehele 'team' bestaat uit één persoon. Scrum wordt op de volgende manier aangepast:

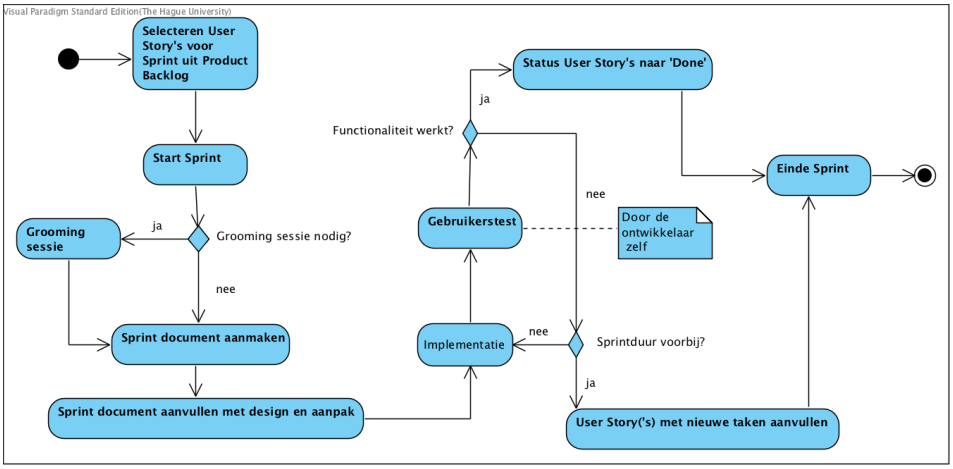
- De Scrum Master komt te vervallen;
- Da Daily Scrum komt te vervallen;
- Het ontwikkelteam bestaat uit één persoon;
- De ontwikkelaar is ook de Product Owner;
- Sprint Review komt te vervallen.

Als Product Owner ga ik eerst het scrumboard inrichten. Hierna zoek ik contact met de stakeholders en ga ik met hun in bespreking. Door die gesprekken kunnen de user story's invulling krijgen. Deze worden vervolgens opgenomen in het Product Backlog. Daarnaast bewaak ik het resultaat en blijf ik met nauwe lijnen in contact met de stakeholders.

In Figuur 5.1 wordt de aanpak geïllustreerd. Het traject wordt opgedeeld in negen sprints van ieder twee weken. Aan het eind van elke sprint wordt van de Product Backlog een aantal geprioriteerde story's geselecteerd. Deze worden opgenomen in de Sprint Backlog en uitgewerkt. Het eind van iedere sprint resulteert in een werkend stuk software. Deze is voorzien van documentatie. Daarin staat het design- en ontwikkelproces beschreven. Scrum is normaliter 'lean': weinig tot geen documentatie. De documentatie schrijf ik om na te denken over oplossingsmogelijkheden. Daarnaast dient het als procesbeschrijving voor de opdrachtgever. Hij kan dan feedback geven op het doorlopen proces indien het nodig is. Het sprintproces dat doorlopen wordt, is weergegeven in Figuur 5.2.



Figuur 5.1 : **Scrum ontwikkelproces voor dit project**



Figuur 5.2 : Sprintproces

5.2 Afspraken

5.2.1 Afbakening

Een project goed afbakenen is noodzakelijk. Vooraf moet afgesproken worden wat wel en niet gedaan wordt. Verwachtingen van beide kanten kunnen namelijk verschillen. In het plan van aanpak is te lezen dat ontwikkeling aan de REST API buiten de scope valt van deze opdracht. Dat geldt ook voor het publiceren van de apps. Dit is de verantwoordelijkheid van CGI.

5.2.2 Randvoorwaarden

Het project wordt zelfstandig uitgevoerd. Vragen worden beantwoord door Jos Siemons, Rémon van den Beemt, Franc Buve en Nghia Le. Zij zijn het aanspreekpunt voor beslissingen die genomen moeten worden. Franc fungeert naast CiMS specialist ook als opdrachtgever.

5.2.3 Risico's

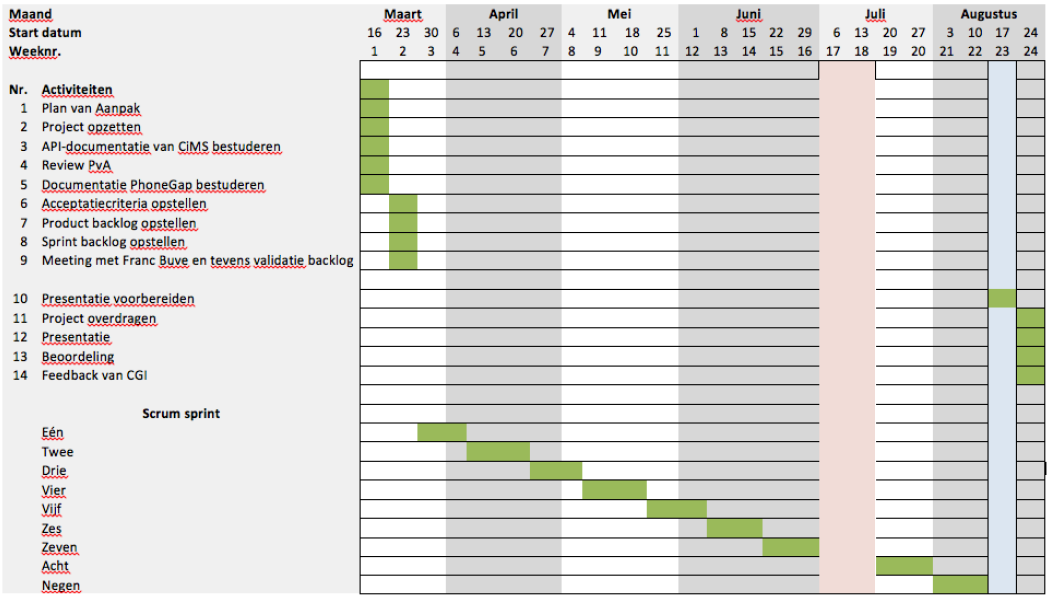
In het plan van aanpak zijn de vooraf gesignaleerde risico's in kaart gebracht. De risico's zijn weergegeven in Tabel 5.1. Deze tabel geeft ook de gevolgen, herkenningspunt, maatregelen en verantwoordelijkheden weer.

Tabel 5.1 **: Risicoanalyse**

#	Risico	Gevolgen	Herkenningspunten	Maatregelen	Verantwoordelijk
1	Een technisch probleem waar ik niet uitkom.	Planning kan uitlopen en onhaalbaar worden.	Na twee uur geen progressie met het probleem.	Contact opnemen met Yard Nijsten of Nghia Le. Aan hun het probleem voorleggen.	Justin Vrolijk
2	Functionaliteit niet mogelijk op iOS en/of Android.	Eindproduct (of bepaald platform) heeft minder functionaliteiten.	Na maatregel uit risico één nog geen oplossing.	Contact opnemen met Franc Buve. Maatregelen met hem afstemmen.	Justin Vrolijk
3	Een niet-functionele eis is niet haalbaar.	Het eindproduct voldoet niet voorledig aan alle wensen.	Na een (performance)test of feedback blijkt de eis niet haalbaar.	Contact opnemen met Rémon van den Beemt en Franc Buve. Met hun kijken of er een oplossing mogelijk is of de eis versoepelen.	Justin Vrolijk
4	Deadline blijkt niet haalbaar.	Deadline aanhouden betekend een eindproduct met minder functionaliteiten en wellicht niet bruikbaar.	Herhaaldelijk zijn risico's: één, twee of drie voortgekomen.	Contact opnemen met Rémon van den Beemt. Planning opnieuw bekijken en eventueel de deadline verschuiven.	Justin Vrolijk
5	Functionaliteiten blijken niet haalbaar met PhoneGap	Minder functionaliteiten of onderzoeken andere mogelijkheden (platformen).	Meerdere functionaliteiten kunnen niet gerealiseerd worden.	Een meeting plannen met Rémon van den Beemt, Franc Buve en Nghia Le.	Justin Vrolijk
6	Ontwikkelaar is niet goed gefaciliteerd.	Eindproduct voldoet niet alle wensen en eisen of eindproduct kan niet gerealiseerd worden.	Ontbreken van licenties of hardware.	Contact opnemen met Rémon van den Beemt of Jos Siemons.	CGI

5.3 Globale planning

In Figuur 5.2 staat de globale planning weergegeven. Op 16 maart 2015 begin ik aan de opdracht. De activiteitennummers één tot en met negen worden in de eerste twee weken uitgevoerd. Deze twee weken staan in kader van kennismaking en het uitvoeren van de activiteiten. Hierna wordt gestart met de eerste sprint. Sprint één tot en met zeven worden achterelkaar doorlopen. Waarna ik twee weken verlof neem. Daarna worden de laatste twee sprints uitgevoerd. De sprint indeling wordt bepaald voor de start van de sprint. Weeknummer 23 staat gereserveerd als uitloop week. In diezelfde week worden ook voorbereidingen verricht voor de eindpresentatie. Deze presentatie is bedoeld voor CGI. In de laatste week, weeknummer 24, wordt het project overgedragen. Daarna volgt ook de eindpresentatie en ontvang ik feedback van CGI.



Figuur 5.2 : **Globale planning**

6 Uitvoering

In hoofdstuk zes staan de uitgevoerd werkzaamheden beschreven. Centraal in de beschrijving staan de vragen:

- Wat heb ik gedaan?
- Hoe heb ik het gedaan?
- Waarom heb ik het zo gedaan en waarom op die manier?
- Wie waren de betrokkenen?
- Welke oplossingsrichtingen waren er?
- Wat waren mijn keuze/beslismomenten?
- Wat was het resultaat?

In paragraaf 6.1 begint met de aanvang van de opdracht. Daarin worden de eerste twee weken van de afstudeeropdracht beschreven. Paragraaf 6.2 begint met de eerste sprint van Scrum. Daarin wordt de softwarearchitectuur bedacht en geïmplementeerd. In sprint twee wordt het hoofdscherm bedacht en kijk ik naar de wijzigbaarheid van de app. Sprint twee wordt beschreven in paragraaf 6.3. Daarna wordt in paragraaf 6.4 de user interface bedacht. Sprint drie bleek een goed moment om de storyboard te ontwerpen. Paragraaf 6.5 beschrijft de koppeling met CiMS en de ondersteuning van meerdere talen. In paragraaf 6.6 komt het filteren van oplaadpunten aanbod. Vervolgens wordt in paragraaf 6.7 een koppeling gemaakt met een nieuw platform. Dat platform heet CRM. Paragraaf 6.8 vervolgt met drie lastige functionaliteiten. Daarin wordt het starten, stoppen en reserveren van een transactie gerealiseerd. In paragraaf 6.9 wordt de user interface nogmaals bekeken en verbeterd. De applicatie wordt in die sprint verstuurd een CGI werknemer. Paragraaf 6.10 beschrijft de laatste sprint. De feedback, die ik heb ontvangen van de CGI werknemer die de app heeft ontvangen in de vorige paragraaf, wordt in deze sprint bekeken. Daarnaast wordt de performance getest en testen we de app op security. Tot slot komt in paragraaf 6.11 de legacy aanbod.

6.1 Aanvang opdracht

De eerste twee weken staan in het kader van oriëntatie, kennismaking en het plan van aanpak. Ook wordt de documentatie van CiMS, PhoneGap en Xamarin bestudeerd en worden de scrumprocessen ingericht.
In het vorige hoofdstuk is het plan van aanpak beschreven. Daarna is er literatuuronderzoek gedaan naar CiMS, PhoneGap en Xamarin. Xamarin is ook een platform waarmee mobiele cross-platform applicaties gebouwd kunnen worden. Paragraaf 6.1.1 gaat over het initiatiedocument. Dat document dient ter voorbereiding op de scrumprocessen. Die komen in paragraaf 6.1.2 aan de orde. Het resultaat van de ondernomen activiteiten staat in paragraaf 6.1.3.

6.1.1 Initiatiedocument op het Product Backlog

Ter voorbereiding op het Product Backlog is een initiatiedocument aangemaakt. Daarin zijn de requirements geanalyseerd, de use cases vastgesteld, de requirements die tegenstijdig zijn opgesomd, alle requirements geprioriteerd, belangrijke use cases zijn beschreven en tot slot is er pakketselectie uitgevoerd tussen Xamarin en PhoneGap.
Dat document is geschreven volgens de richtlijnen van de elaboratiefase uit (R)UP. In de planning was week twee gereserveerd voor het inrichten van het Product Backlog. Dit bleek ruim ingepland te zijn. Ik wou graag de extra tijd gebruiken om meer inzicht te krijgen. De elaboratiefase heeft daarbij in het verleden - bij projecten op school - vaak geholpen. Scrum is echter lean en niks staat vast. Het initiatiedocument dient daarom alleen ter voorbereiding op het project. Met dit document kan invulling worden gegeven aan het Product Backlog, maar het is niet leidend. Veranderingen worden niet in dit document doorgevoerd.

Allereerst zijn de stakeholders in kaart gebracht. Zij staan opgesomd in Tabel 6.1. Met interviews, e-mails en telefonisch contact, worden hun wensen en eisen vastgesteld. Met hun wordt in nauwe

lijnen koppeling gehouden om de voortgang te bewaken. Franc Buve is het eerste aanspreekpunt. Hij neemt de rol van opdrachtgever op zich.

Tabel 6.1 **: Stakeholders zonder contactgegevens**

ID	Naam	Organisatie en functie	Info
SA1	Buve, Franc	CGI - Software Architect	Software architect voor ElaadNL en CiMS
DCS1	Siemons, Jos	CGI - Director Consulting Services	Marketing Manager
IT1	Le, Nghia	CGi - Principal IT Architect	
PL1	Beemt, Rémon van den	CGI - Technical Project Leader	
EV1	Siemons, Jos	-	Wensen voor de app als EV bestuurder
EV2	Petzer, Guido	-	

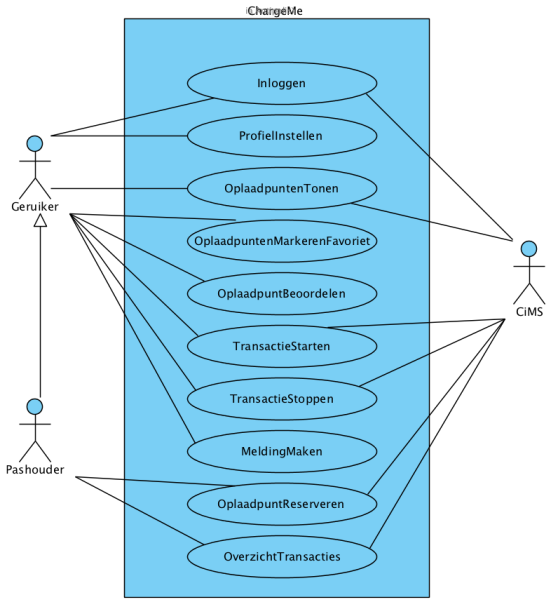
Vervolgens zijn de requirements opgesomd. Allereerst zijn de wensen en eisen uit de opdrachtomschrijving (afstudeerplan) vertaald. Een onderscheid is gemaakt tussen gebruikers wensen, project-, design- en ontwerpbeslissingen. Deze zijn te herkennen aan de eerste twee letters van het ID. Gebruikerswensen beginnen met UR, projectgrenzen met PC, etc.

Tabel 6.2 **: Requirements uit opdrachtomschrijving**

ID	Omschrijving	Stakeholder	Bron	Versie	Datum
UR01	De eindgebruikers kunnen laadpalen zoeken.	SA1	Afstudeerplan	1.0	16-03-2015
UR02	Eindgebruikers kunnen laadpalen reserveren.	SA1	Afstudeerplan	1.0	16-03-2015
UR03	Eindgebruikers kunnen laadpalen markeren als favoriet.	SA1	Afstudeerplan	1.0	16-03-2015
UR04	Eindgebruikers kunnen laadpalen zoeken die geschikt zijn voor hun type oplaadstekker.	SA1	Afstudeerplan	1.0	16-03-2015
UR05	Eindgebruikers kunnen een waardering geven over het oplaadpunt.	SA1	Afstudeerplan	1.0	16-03-2015
UR06	De eindgebruikers kunnen een melding maken van onjuiste informatie.	SA1	Afstudeerplan	1.0	16-03-2015
UR07	De eindgebruikers kunnen zien welke laadpalen beschikbaar zijn.	SA1	Afstudeerplan	1.0	16-03-2015
PC01	Het project moet een Agile methodiek hanteren. Scrum wordt geprefereerd.	SA1	Afstudeerplan	1.0	16-03-2015
PC02	Het project moet eind augustus demonstreerbaar zijn.	SA1	Afstudeerplan	1.0	16-03-2015
DC01	Het project moet met PhoneGap ontwikkeld worden.	SA1	Afstudeerplan	1.0	16-03-2015
DC02	Het project moet met Xamarin ontwikkeld worden.	IT1	Gesprek	1.0	18-03-2015
O01	De laadpalen worden getoond in een geografische map.	PL1	Gesprek	1.0	18-03-2015

Daarna ben ik interviews gaan inplannen. Daarmee wou ik de resterende wensen en eisen achterhalen. Ook draagt een persoonlijk gesprek bij aan een gezamenlijk beeld. Ter voorbereiding op die gesprekken ben ik eerst gaan onderzoeken hoe soortgelijke apps werken. Een aantal stakeholders bezit namelijk een elektrische auto. Zij zullen gebruik maken van bestaande apps en zij zullen vermoedelijk de minheden benoemen. De requirements uit die interviews zijn opgenomen in een aparte tabel in het initiatiedocument.
Nu een aardige waslijst aan requirements bekend is, moeten ze geprioriteerd worden. Hiervoor maakte ik gebruik van de MoSCoW methode. Met die methode worden requirements verdeeld in één van de volgende categorieën: must have, should have, could have en wont have. Deze prioritering kan toegepast worden in het Product Backlog.

Vervolgens is een use case diagram gemaakt. Requirements die ingedeeld zijn in de must have categorie, zijn opgenomen als use cases. Vervolgens zijn de actoren bepaald. Naast de gebruikersactor is er ook een pashouderactor toegevoegd, opdat de functionaliteiten niet voor iedereen beschikbaar zijn. CiMS is ook als actor opgenomen omdat het systeem daarmee data uitwisselt. Belangrijke use cases zijn uitgewerkt in use case beschrijvingen.



Figuur 6.1 : **Use case diagram**

6.1.1.1 Xamarin

De term Xamarin is eerder in dit hoofdstuk al eens benoemd. Xamarin is een alternatief platform op PhoneGap. Net als PhoneGap kan met dit platform mobiele cross-platform applicaties ontwikkeld worden. Twee platformen die hetzelfde einddoel hebben, maar wezenlijk verschillen van elkaar.
Dit platform wordt bestudeerd omdat twee requirements elkaar tegenspreken. Franc Buve wil graag dat PhoneGap wordt gebruikt als platform en Nghia Le prefereert Xamarin. Teamleider Rémon heeft aangegeven dat het platform gebruikt moet worden dat het beste aansluit op de wensen en eisen. Ik besloot daarom om de documentatie van Xamarin te bestuderen, zodat vervolgens pakketselectie uitgevoerd kan worden.

Uit een artikel op Xamarin blijkt dat met dit platform native applicaties geheel in de programmeertaal C# gebouwd kunnen worden ("Create native iOS, Android, Mac and Windows apps in C#", z.j.). Android en Apple ondersteunen beide verschillende programmeertalen voor hun apparaten. Android applicaties worden geschreven in Java en die van Apple in Objective-C (of Swift). Dit heeft tot op heden altijd een knelpunt gevormd. Ontwikkelaars die beide ontwikkeltalen eigen zijn, zijn schaars. Ook moet een app twee keer gebouwd worden. Dat is kostbaar en tijdsintensief. Het alternatief was een PhoneGap app. Dat zijn echter geen native applicaties, maar hybride apps.

Met Xamarin kunnen dus apps geheel in C# ontwikkeld worden. De native UI en API's kunnen aangeroepen worden. Hierdoor ervaart de gebruiker de native performance. Kennis van beide ontwikkelmethodes blijft echter noodzakelijk. De programmeertaal is dan wel gelijk, beide hebben hun eigen ontwikkelmethodes. Android gebruikt bijvoorbeeld XAML en Activity's (schermen). iOS pakt het heel anders aan. Zij maken gebruik van delegates en het MVC principe. Daarnaast verschillen hun codebibliotheken drastisch van elkaar.

Om die beperking te minimaliseren, is Xamarin Forms ontwikkeld. Forms is als het ware een extra platform bovenop het huidige. Het heeft een eigen codebibliotheek die door de compiler vertaald wordt naar iOS, Android en Windows Phone code. In Figuur 6.2 is te zien hoe dat in zijn werk gaat. Aan de linkerzijde wordt een pagina aangemaakt met behulp van objecten uit de Forms bibliotheek. De compiler handelt het verder intern af. Het resultaat zijn drie apps die gebouwd zijn met de native schermelementen. Vandaar het merkbare verschil in uiterlijk.

Figuur 6.2 : **Xamarin Forms voorbeeld (Bron: https://xamarin.com/forms)**

6.1.1.2 Pakketselectie tussen Xamarin en PhoneGap

Ten slotte is er pakketselectie uitgevoerd tussen Xamarin en PhoneGap. Hiermee kan pragmatisch de beste kandidaat gekozen worden. Van Leeuwen (2007) heeft een selectieproces in kaart gebracht. Deze bestaat uit: bedrijfsanalyse, analyse huidige situatie, analyse gewenste situatie, vaststellen mogelijke oplossingen en selecteren definitieve oplossing. Door middel van deze methode kan een product gekozen worden dat aansluit op de wensen en eisen. Ook wordt rekening gehouden met belangrijke aspecten die over het hoofd gezien kunnen worden.
Het doorlopen van het gehele selectieproces is bij dit project overbodig omdat de keuze gemaakt moet worden tussen twee, voor CGI bekende (en gebruikte), producten. Alleen de huidige en gewenste situatie zijn geanalyseerd. Vervolgens is een definitieve keuze - definitief voor deze selectie - gemaakt.

Uit de analyse van de huidige situatie bleek dat CGI bekend was met beide producten. Zowel Xamarin als PhoneGap is in eerdere projecten gebruikt. Er wordt onderzocht of de gewenste functionaliteiten gerealiseerd kunnen worden op beide platformen. Daarnaast wordt ook gekeken in welke mate de platformen bijdragen aan de volgende aspecten: performance, security, bruikbaarheid en overdraagbaarheid.

In de gewenste situatie wordt het platform gebruikt dat het meest aan de gestelde eisen en aspecten voldoet. Een belangrijk punt is de overdraagbaarheid naar toekomstige klanten. Zij moeten met minieme moeite de app kunnen branden.

Bij het selecteren van de definitieve oplossing zijn de wensen en eisen in kaart gebracht. Per requirement is gekeken naar de mogelijkheden. Deze worden tegen elkaar afgewogen en voorzien van een beoordeling. Ten slotte worden de beoordelingen bij elkaar opgeteld wat resulteert in twee totaal scores. Deze worden met elkaar vergeleken voor de eindkeuze. Het resultaat is te zien in Figuur 6.3.

	Cordova	Xamarin	Note
Geografische map	2	2	Google Maps is vrijblijvend tot 25 000 map loads per dag.
Geografische map manipuleren	3	3	Toevoegen markers, etc..
Betaling verrichten met creditcard	2	2	PayPal Checkout SDK
Integratie third-party hardware	2	3	
Flexibiliteit UI	2	3	Xamarin: Een XML bestand waarin de UI instellingen staan en dit in runtime parsen. Cordova: XML voor settings en CSS voor design
Performance	1	3	Cordova: Website met HTML,CSS en Javascript. Xamarin: Native app met shared code in C#.
Beveiliging	2	3	Cordova is gevoelig voor XSS.
Overdraagbaarheid	2	1	Voor beide heb je een Mac en Windows computer nodig.
Benodigde betaalde licenties	2 (A, An)	3 (A,An,X en V (V =optioneel)	Apple dev licence Android licentie Visual Studio (optioneel) Xamarin
Benodigde hardware	0	0	Apple en Windows
Klantenbranding	2	3	
Kennis binnen CGI over het product	3	3	
Totaal	21	26	
Kosten			
Apple developers licentie	99 $	99$	Per jaar
Android licentie	25 $	25 $	Eenmalig
Xamarin licentie	/	25 $	Vanaf 25 dollar per maand
Cordova licentie	0 $	/	
0 = onvoldoende 1 = voldoende 2 = goed 3 = zeer goed			

Figuur 6.3 : Pakketselectie tussen Xamarin en PhoneGap

De conclusie is als volgt. Op veel punten scoort Xamarin hoger. Een belangrijke afweging is de overdraagbaarheid naar toekomstige klanten. Voor Xamarin moet één extra licentie worden aangeschaft. De Indie license is voldoende om wijzigingen aan te brengen aan de app. Die licentie kost maandelijks 25 dollar. Voor beide producten moet een Apple en Android developers licentie aangeschaft worden. Dit kost de klant 99 dollar per jaar voor Apple en eenmalig 25 dollar voor Android. Tevens moet men in het bezit zijn van Windows en Apple pc om op te testen en compileren. Hieruit is besloten om Xamarin te adviseren. Dit raamwerk scoort op vele punten beter dan PhoneGap en een licentie van 25 dollar per maand vormt waarschijnlijk geen drempel voor bedrijven.

6.1.2 Scrumprocessen inrichten

Bij het inrichten van de scrumprocessen heb ik de Product Backlog aangevuld met user story's, acceptatiecriteria geschreven en een selectie gemaakt voor de eerste sprint.

Ten eerste is het Product Backlog aangemaakt. CGI had een account ter beschikking gesteld voor Visual Studio Online. Op die website staat een omgeving klaar waarin de scrumprocessen ingericht kunnen worden. De requirements en use cases uit het initiatiedocument heb ik gebruikt bij het aanmaken van de user story's. Ook kwam de prioritering die ik gedaan had van pas. Elke story is daarvan voorzien. Story's met een hoge prioriteit worden automatisch bovenaan geplaatst.
Het initiatiedocument heb ik gebruik voor de story's omdat daarin de requirements en use cases al staan opgesteld. Deze zijn reeds al besproken en overlegt met de stakeholders. Alleen een vertaalslag was nodig omdat user story's anders beschreven worden. Deze worden normaliter op de volgende manier geschreven: als {persoon} wil ik dat {actie/wens}.
De opdrachtgever Franc Buve heeft toegang tot deze omgeving. Hij kon de user story's bekijken en valideren.

Ten tweede zijn de acceptatiecriteria vastgesteld. Acceptatiecriteria geven zowel aan mij als aan de opdrachtgever duidelijkheid wanneer de app, sprint en of taak af is. De app is afgerond wanneer de volgende punten afgestreept zijn:

- Alle sprints met bijhorende taken zijn doorlopen;

- De user story's met de hoogste prioriteit zijn afgerond;
- Het voldoet aan de niet-functionele eisen;
    - Communicatie tussen CiMS en de app is beveiligd;
    - De laadtijd voor het ophalen van laadpalen bedraagt maximaal 3 seconden;
- Code is voorzien van commentaar;
- Eindpresentatie is gegeven;
- Gehele project is overgedragen aan Franc Buve.

Iedere sprint is ook voorzien van acceptatiecriteria. Deze zijn bij elke sprint hetzelfde. Het gaat met name om de user story's en taken. Elke story kan pas afgerond worden wanneer alle taken doorlopen zijn, de code voorzien is van commentaar, de functionaliteit uitgebreid is getest door eigen gebruikerstesten en nadat het is bekeken/goed gekeurd door Franc Buve. Verder moet elke sprint voorzien zijn van documentatie waarin het proces beschreven staat.
De acceptatiecriteria moeten echter wel haalbaar zijn. Tot zover is er een globale planning gemaakt die niet tot in detail is voorzien van activiteiten. Dit is ook niet mogelijk bij Scrum. Desondanks moet de haalbaarheid bekeken worden. Op dit moment zijn er negen user story's met een hoge prioriteit: de 'must haves' uit het initiatiedocument. Deze zouden onderverdeeld kunnen worden in negen sprints. Dat zou betekenen dat voor het behalen van de minimale eisen grofweg één story per sprint wordt gerealiseerd. Dat is in mijn ogen haalbaar.

Ten slotte is er een selectie gemaakt voor de eerste sprint. Nu het platform bekend is, moet eerst de nodige software geïnstalleerd worden. Vervolgens moet het project aangemaakt worden en het nodige ingesteld worden. Ook lijkt het mij verstandig om stil te staan bij de softwarearchitectuur. Dit heeft er toe geleid dat het Product Backlog is aangevuld met twee nieuwe user story's. Deze krijgen een hoge prioriteit zodat ze direct opgepakt kunnen worden.
Het Product Backlog is een levend 'document'. Gedurende het project zullen er story's gewijzigd en aangevuld worden. Bij aanvullingen moet ik kritisch blijven kijken of het project haalbaar blijft. Het installeren en instellen van de benodigde tools is noodzakelijk. Een softwarearchitectuur vooraf bedenken is ook verstandig. Hierdoor kunnen functionaliteiten sneller geïmplementeerd worden. Ook voorkomt het spaghetticode waardoor het project overzichtelijk blijft. Wijzingen kunnen sneller worden doorgevoerd.

6.1.3 Resultaat

Deze twee weken hebben meerdere resultaten opgeleverd. Allereerst is het initiatiedocument geschreven. Deze is opgenomen in bijlage III. Hierna heb ik de literatuur bestudeerd van CiMS, PhoneGap en Xamarin. Kennis die ik daaruit heb opgedaan is beschreven in dit verslag. Vervolgens heb ik de scrumprocessen ingericht. Hierbij zijn geen fysieke documenten opgeleverd omdat het in een online omgeving staat. Tabel 6.3 toont de user story's die zijn opgenomen in het Product Backlog.

Tabel 6.3 **: User story's uit het Product Backlog**

User story's	
• Als opdrachtgever wil ik dat de app op het 3G en 4G netwerk snel werkt • Als opdrachtgever wil ik dat de interface een heldere en professionele uitstraling heeft. • Als gebruiker wil ik een transactie kunnen starten vanuit de app. • Als gebruiker wil ik een overzicht van mijn transacties kunnen inzien. • Als gebruiker van de app wil ik in kunnen loggen als ik een pashouder ben. • Als gebruiker wil ik een melding kunnen maken van onjuiste informatie. • Als gebruiker wil ik kunnen kiezen tussen het tonen van enkel beschikbare oplaadpalen en ook niet beschikbare	• Als opdrachtgever en teamleider wil ik dat de communicatie tussen CiMS en de APP beveiligd is, zodat misbruik voorkomen wordt. • Als gebruiker wil ik een transactie kunnen stoppen vanuit de app. • Als gebruiker met een betaalpas, wil ik een oplaadpunt kunnen reserveren. • Als gebruiker wil een profielpagina hebben die ik kan instellen • Als gebruiker wil ik oplaadpunten kunnen markeren als favoriet. • Als gebruiker wil ik oplaadpunten kunnen beoordelen. • Als opdrachtgever wil ik dat de app gemakkelijk te wijzigen is, zodat klanten die de app overnemen eenvoudig branding kunnen toepassen.

• Als developer wil ik een koppeling maken met CiMS, zodat data uit dat systeem gebruikt kan worden in de app. • Als opdrachtgever wil ik dat het hoofdscherm wordt voorzien van een geografische map • Als developer wil ik eerst het project opzetten.	• Als developer wil ik de softwarearchitectuur van de app bedenken • Als developer wil ik eerst een navigatie map ontwerpen.

6.2 Sprint één - De softwarearchitectuur

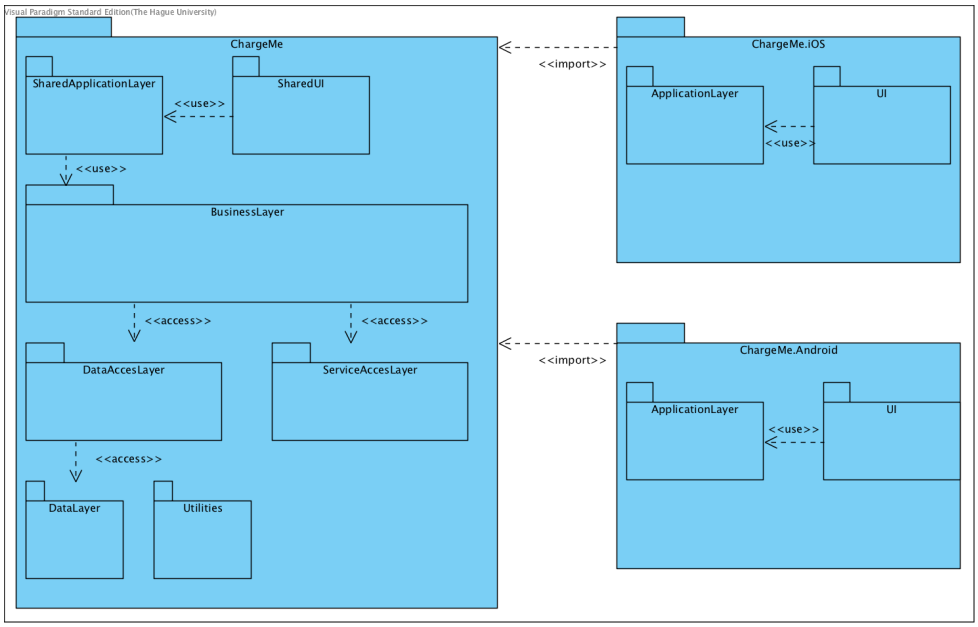
Het uitgangspunt van sprint één is het bedenken van de softwarearchitectuur voor de app. Daarnaast staat op de planning om de benodigde tools en software te downloaden en installeren. In Tabel 6.4 staan de story's en bijhorende taken voor deze sprint.
Allereerst worden de benodigde applicaties geïnstalleerd en geconfigureerd. Vervolgens wordt de softwarearchitectuur ontworpen. Tot slot wordt het resultaat getoond in de vorm van een test.

Tabel 6.4 **: User story's voor sprint één**

User story's	Taken
Als developer wil ik de softwarearchitectuur van de app bedenken	• Analyse packagediagram ontwerpen. • Softwarearchitectuur testen door middel van een test implementatie.
Als developer wil ik eerst het project opzetten om.	• Project aanmaken in Xamarin. • GIT instellen.
Als developer wil ik eerst een navigatie map ontwerpen.	• Navigatie map creëren met UML. • Navigatie map opnemen in initiatie document.

Ten eerste is het project aangemaakt. De integrated development environment (IDE) van Xamarin wordt gebruikt als ontwikkelomgeving. Een naam moet opgegeven worden bij de aanmaak van het project. Ik heb besloten om de app ChargeMe te noemen. Verder in dit verslag zal deze naam gehanteerd worden. Verder is er een lokale GIT repository aangemaakt. Deze wordt gesynchroniseerd met de TFS repository van Visual Studio Online.

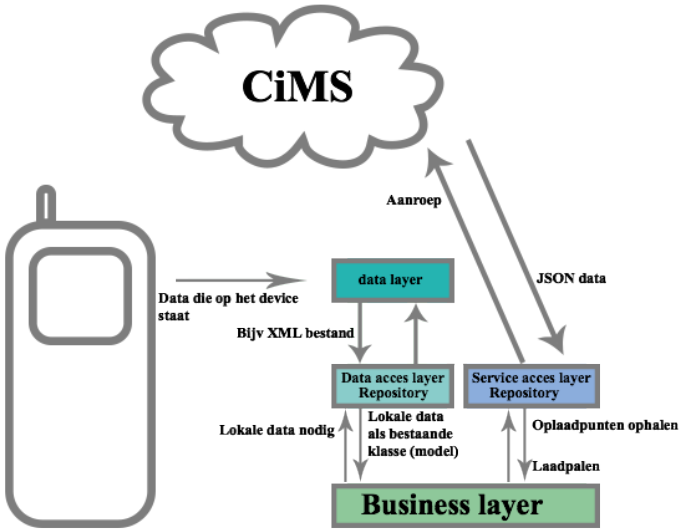
Vervolgens is het analyse packagediagram ontworpen. Hiermee kan de softwarearchitectuur gerepresenteerd worden. ChargeMe is opgedeeld in drie losse projecten. Een project voor iOS, Android en een gedeelde codebibliotheek. De platform specifieke projecten bevatten hun platform gebonden klassen. Bijvoorbeeld de opstartklasse en de platform specifieke schermen en code.
Het gedeelde codebibliotheek heet ChargeMe. ChargeMe.iOS en ChargeMe.Android hebben een referentie naar dit project. Naast de verschillende lagen, heeft deze package ook een gedeelde UI- en applicatie-laag. Deze zijn bedoeld voor de schermen en controllers van het Forms platform. Ik ga gebruik maken van het MVC design patroon: views komen in de UI-laag, controller in de applicatie-laag en modellen in de business laag.
De verschillende lagen hebben alle hun eigen verantwoordelijkheid. De BusinessLayer bestaat uit de modellen en managerklassen. De managers zijn het aanspreekpunt voor de UI- en applicatie-laag. De andere lagen kunnen zij niet direct benaderen. Alleen de BusinessLayer kan dat, opdat spaghetticode wordt voorkomen. De Service- en AccesLayer zijn bedoeld voor het ophalen en versturen van data naar externe en interne bronnen. Figuur 6.4 illustreert het analyse packagediagram.



Figuur 6.4 : **Analyse packagediagram ChargeMe**

Een softwarearchitectuur bedenken, wordt vaak niet gezien als agile. Een stuk vrijheid gaat verloren. Het is toch verstandig om stil te staan bij softwarearchitectuur. Dat blijkt ook uit een artikel op CGI "Agilisten kunnen succesvoller worden als ze de projectcontext meenemen in hun oordeel over het nut van architectuur" (Poort, 2013). De softwarearchitectuur wordt vanaf een andere invalshoek bekeken. Deze biedt een goede basis om mee te starten, maar staat niet vast. Gedurende het project kan het aangepast worden. Ik heb besloten om gebruik te maken van een analyse packagediagram. Analyse diagrammen worden niet tot in detail ontworpen. Dat is gunstig voor Scrum.

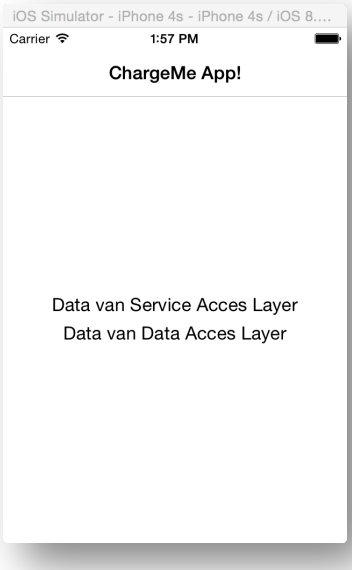
Deze softwarearchitectuur heb ik toegepast omdat het de cohesie maximaliseert en koppelingen minimaliseert. Hierdoor wordt het systeem robuuster. Ook komt het ten goede van de beheerbaarheid. Arlow en Neustadt (2011, p. 235) en Rozanski en Eoin (2008, p. 294) concluderen dat ook.
De softwarearchitectuur is gebaseerd op die van ASP.NET MVC. Xamarin stelt een gelijke architectuur voor in hun guides. De Data- en ServiceAccesLayer zijn gebruikt omdat de app data ophaalt en verstuurt naar interne en externe bronnen. Deze lagen bieden een scheidingslaag door middel van 'repository's'. Een repository krijgt in allerlei vormen data binnen. Het is zijn verantwoordelijkheid om die te converteren naar klassen binnen het systeem. Figuur 6.5 toont een voorbeeld van de werking van deze lagen.



Figuur 6.5 : **Werking service- en data acces layer**

Ten slotte is een test geïmplementeerd zodat de werking van de softwarearchitectuur getest kan worden. Data wordt opgehaald uit de data- en serviceAccesLayer. Via de UI wordt de manager in de

businessLayer aangeroepen. Die roept op zijn beurt de repository aan in de DataAccesLayer. Deze spreekt vervolgens de DataLayer aan. Hetzelfde principe geldt voor de service acceslaag. In Figuur 6.6 is het resultaat te zien.



Figuur 6.6 : **Softwarearchitectuur test**

6.3 Sprint twee - Hoofscherm en wijzigbaarheid

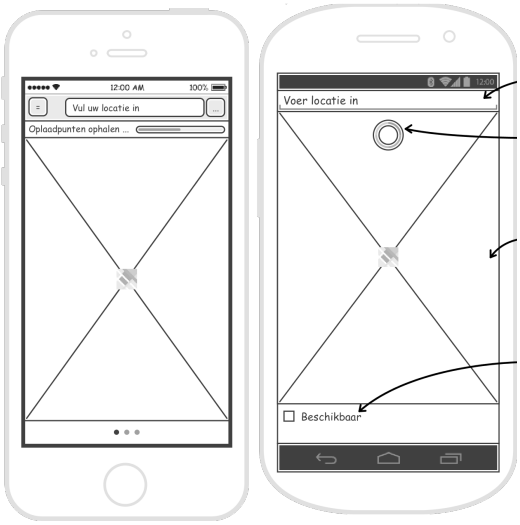
In sprint twee komen de volgende twee story's aan de orde: bedenken en implementeren van het hoofdscherm respectievelijk een systeem bedenken die de wijzigbaarheid ten goede komt. Het hoofdonderdeel en tevens startpunt van de app, is het hoofdscherm. Dit scherm is nodig om verder te kunnen werken aan de vervolg schermen. Ook wordt in een vroeg stadium de mogelijkheden van de geografische map onderzocht. Daarna wordt een systeem bedacht die de wijzigbaarheid bevordert. De story's zijn weergegeven in Tabel 6.5.

Tabel 6.5 **: User story's van sprint twee**

Story's	Taken
Als opdrachtgever wil ik dat het hoofdscherm wordt voorzien van een geografische map.	• UI van hoofdscherm ontwerpen. • Hoofdscherm implementeren. • Laadpalen tonen op de map (test data). • Filter mogelijkheid maken tussen actieve en niet actieve laadpalen. • Zoek functionaliteit implementeren in de map, zodat naar een locatie wordt genavigeerd d.m.v. een adres. • Zelf een gebruikerstest uitvoeren.
Als opdrachtgever wil ik dat de app gemakkelijk te wijzigen is, zodat klanten die de app overnemen eenvoudig branding kunnen toepassen.	• Nadenken over een mogelijk systeem (design). • Design implementeren. • Zelf een gebruikerstest uitvoeren.

6.3.1 De hoofdschermen

Allereerst heb ik de UI voor de hoofschermen ontworpen. Hoewel het UI niet vast staat, is het bedenken van een eerste opzet verstandig. Er moet rekening gehouden worden met het uiterlijk dat Android en iOS hanteert. De opdrachtgever wenst dat de app ergonomisch is in het gebruik. Ook een heldere en professionele uitstraling is vereist. Een interface die clean is en gebruik maakt van platform afhankelijke componenten, draagt bij aan de wens van de opdrachtgever. In Figuur 6.7 zijn de verschillen tussen iOS en Android merkbaar. Het behouden van de native elementen is wenselijk voor de ergonomie. Daarnaast draagt het bij aan de wens voor een professionele en heldere interface.



Figuur 6.7 **: UI hoofdschermen**

Vervolgens is met het design het hoofdscherm geïmplementeerd. Met Xamarin Forms heb ik een hoofdpagina aangemaakt. Een geografische map heb ik daaraan toevoegd. De ServiceAccesLayer heb ik gevuld met test data. De data wordt getoond in de geografische map. Ook heb ik een switch gemaakt. Daarmee kunnen laadpalen gefilterd worden op beschikbaarheid. Verder is de map voorzien van een zoekvenster. De gebruiker kan daarin een adres typen. Een lijst met adressen wordt vervolgens getoond. De map navigeert naar het geselecteerde adres.
In Figuur 6.8 is het klassendiagram te zien. Het is volgens de softwarearchitectuur in paragraaf 6.2 gebouwd. De service laag is uitgebreid met een CiMS-repository-klasse. Deze klasse spreekt de klasse aan die uiteindelijk een koppeling heeft met de REST API van CiMS. Voor nu stuurt die alleen test data. De repository vertaalt die data naar een lijst van Chargepoints. Deze lijst wordt bijgehouden in de managerklasse. Ik heb een abstracte generieke managerklasse geschreven omdat er meerdere managerklassen in vervolg sprints komen.

De handige mapping van Xamarin Forms brengt ook nadelen met zich mee. Zo blijkt bij het implementeren van de hoofdpagina's. De speld-knoppen - de spelden waarmee een punt op de map gemarkeerd wordt - in de geografische map bijvoorbeeld. Het uiterlijk kan niet zomaar gewijzigd worden; Forms heeft nog een gelimiteerde interface. Dit blijkt ook bij veel andere elementen zo te zijn. Kort samengevat: de gehele interface van Android en iOS zijn nog niet volledig vertaald in Forms.

De bruikbaarheid van Forms wordt nog een keer onderzocht. Hoewel het uitgangspunt is om het uiterlijk van de native elementen niet te veel te veranderen, vormt het wel een knelpunt. Twee opties blijken mogelijk: gebruik maken van zelf geschreven renderers en de Dependency Service of geen gebruik maken van Forms. Het laatste, Forms niet gebruiken, betekent dat beide platformen apart ontwikkeld worden.

Eerst kijk ik naar de zogeheten 'custom renderers' en 'Dependency Services' van Forms. Met dat systeem kan de native functionaliteit aangeroepen worden. Hierdoor is het mogelijk om, wanneer het nodig is, platform specifieke implementatie bij te bouwen. Deze code komt te staan in de platform specifieke projecten. De ontwikkelaar moet de 'uitbreiding' wel voor elk ondersteund platform apart ontwikkelen.
Het alternatief is de schermen per platform zelf ontwikkelen. Forms wordt dan niet gebruikt. Het project ChargeMe bestaat dan alleen uit de business rules en data- en service laag. De UI en applicatie-laag staat in de platform specifieke projecten. Per platform worden de schermen naar wens ontwikkeld. Uitgebreide kennis van beide platform is noodzakelijk, want je moet bekend zijn met hun ontwikkelmethodes.

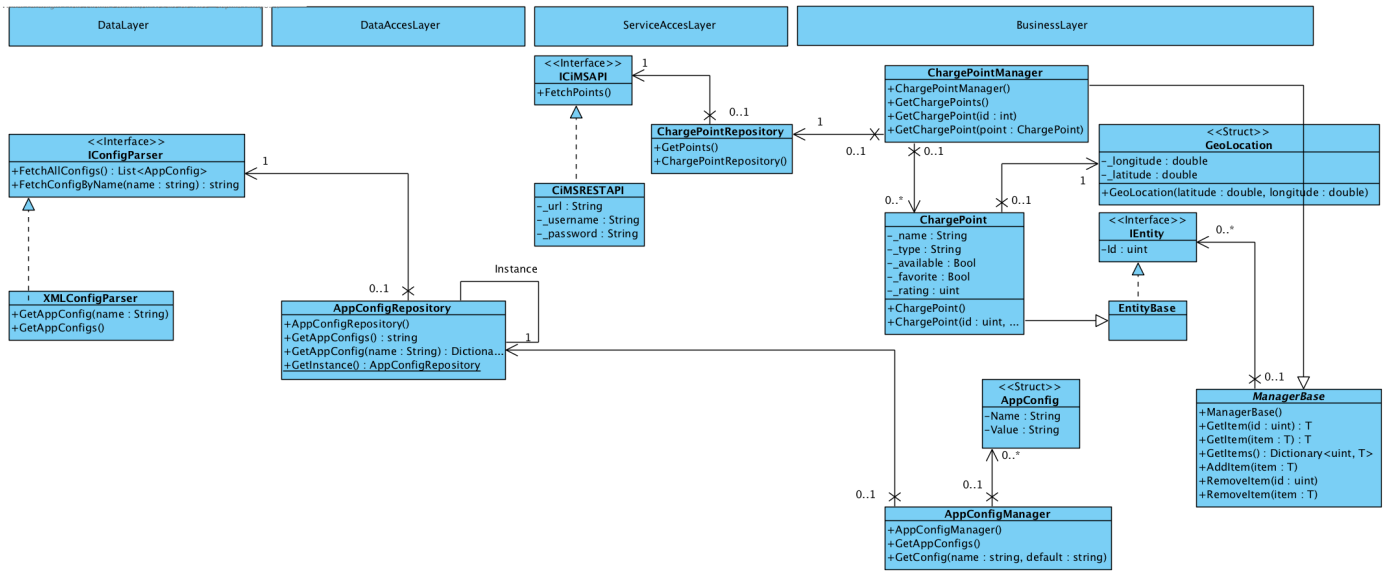
Beide mogelijkheden heb ik bekeken. Per platform de schermen opbouwen geeft de meeste mogelijkheden. Echter gaat dat gebaard met veel meer ontwikkeltijd: twee apps worden afzonderlijk

ontwikkeld. Met Forms zijn ook platform specifieke ontwikkelingen nodig, maar alleen bij elementen waarbij Forms niet toereikend genoeg is. Aangezien de eerste optie, per platform ontwikkelen, te veel tijd kost en niet haalbaar is met de planning, heb ik gekozen om Forms te blijven gebruiken.

6.3.2 Wijzigbaarheid van de app

Ik heb een systeem bedacht waarmee alle configuraties in een centraal XML-bestand staan. Het systeem leest dit bestand uit en past het toe. Ook dit systeem is gebouwd volgens de softwarearchitectuur richtlijnen. Het XML-bestand wordt lokaal opgeslagen op het device. De parser en repository worden daarom in de DataAccesLayer geplaatst; een interne bron. De parser erft over van een interface. Door de interface kunnen later andere typen parsers gemakkelijk bijgebouwd worden. Het singleton ontwerppatroon wordt toegepast op de configuratielader repository. De configuraties worden dan maar één keer ingeladen; het wordt namelijk op meerdere locaties gebruikt. De manager in businesslaag is het aanspreekpunt voor andere klassen, zoals bijvoorbeeld de controllers. Het erft over van de eerder gemaakte base-manager-klasse. Het klassendiagram is weergegeven in Figuur 6.8.

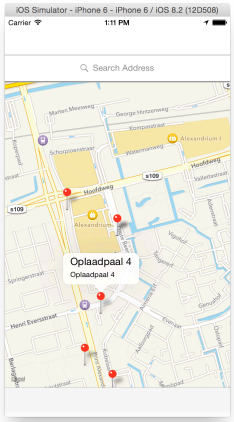
Een belangrijke eis is de overdraagbaarheid van ChargeMe. De app wordt overgenomen door klanten van CGI. Zij zullen de app zelf branden. Een centraal bestand waar alle configuraties instaan, draagt bij aan de overdraagbaarheid. Ik heb gekozen voor XML omdat het makkelijker te lezen is dan bijvoorbeeld JSON. Ook heb ik besloten om van de AppConfig een struct te maken. Dit model is alleen bedoeld als datacontainer. Het hoeft niet voorzien te worden van methodes. Een struct levert betere performance omdat deze een lichtgewicht klasse is. Zo blijkt ook uit een C# boek: “you can think of a structure as a ‘lightweight class,’ given that structures provide a way to define a type that supports encapsulation” (Troelsen, 2012, p. 146).



Figuur 6.8 : **Klassendiagram sprint één**

Tijdens het implementeren van dit systeem bleek dat Forms moeite heeft met het inladen van bestanden. Forms kan niet het pad naar de locatie van de app achterhalen. Android en iOS hebben beide een andere mappenstructuur. Er zijn twee mogelijkheden: gebruik maken van de Dependency Service of tijdens runtime de locatie opvragen van een klasse.
Via de Dependency Service kunnen methodes per platform aangeroepen worden die het pad teruggeven. De andere mogelijkheid is: de locatie van een klasse opvragen door gebruik te maken van de klasse-assembly.
Ik heb gekozen voor het laatste. Dit scheelt overbodige code en houdt het overzichtelijker. De assembly heeft een eigenschap waarin de locatie van het project staat. Daarmee kan vervolgens het pad bepaald worden.

6.3.3 Resultaat



Figuur 6.9

Het resultaat van sprint twee is een hoofdscherm met een geografische map. Daarin wordt een aantal laadpalen getoond. Deze kunnen gefilterd worden op beschikbaarheid. Via het zoekvenster kan naar een adres worden genavigeerd. Daarnaast worden configuraties geladen uit een XML-bestand.

6.4 Sprint drie - De user interface

In de 'grooming' sessie aan het eind van sprint twee, sprak ik met Franc Buve. Hij stelde voor dat ik na ga denken over het uiterlijk van de gehele app. Welke schermen komen in de app en hoe zien die eruit? Het leek hem verstandig om dat zo vroeg mogelijk te doen. Ik deelde zijn mening omdat het helderheid geeft over het gehele plaatje. De vervolg functionaliteiten hoeven dan alleen gerealiseerd te worden in de schermen. Het uiterlijk hoeft niet telkens bedacht te worden.
Dit betekent wel dat er nieuwe user story's worden toegevoegd. Dat is geen probleem met Scrum, maar ik moet wel de haalbaarheid blijven bewaken. Op dit moment zijn er twee 'must haves' afgerond. Dat betekent dat er nog zeven user story's over zijn met een hoge prioriteit. Verder zijn er nog 6 sprints over. Aangezien ik ongeveer twee a drie story's per sprint afrond, zal het toevoegen van een nieuwe story geen gevaar vormen voor de haalbaarheid.

In sprint drie staat de user interface centraal. In eerdere sprints is al een aanzet gedaan voor het hoofdscherm en de navigatie-map. Deze worden gebruikt bij het ontworpen van een storyboard. Het Product Backlog wordt aangevuld met twee nieuwe user story's. Deze zijn te zien in Tabel 6.6. In paragraaf 6.4.1 wordt eerst het storyboard ontworpen. In paragraaf 6.4.2 worden de schermen geïmplementeerd. Ze worden vooralsnog alleen voorzien van navigatiemogelijkheden.

Tabel 6.6 : **User story's van sprint drie**

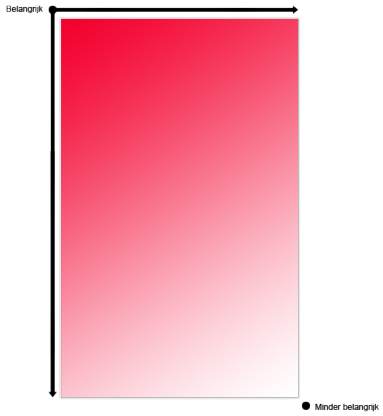
Story's	Taken
Als teamleider wil ik een storyboard die alle schermen en de flow van de app tonen, zodat goed nagedacht wordt over de te maken schermen.	• iOS guidelines lezen. • Android guidelines lezen. • Storyboard ontwerpen.
Als developer wil ik de schermen van het storyboard kunnen doorlopen zonder verdere interactie mogelijkheden in de schermen, zodat de opdrachtgever de flow van app kan testen en daar op kan inhaken wanneer nodig.	• Schermen uit het storyboard aanmaken in het project. • Navigatie functionaliteit toevoegen.

6.4.1 Het storyboard ontwerpen

Allereerst heb ik besloten om de richtlijnen van Apple en Android te bestuderen. Met de opgedane kennis heb ik het storyboard ontworpen. De navigatie map in het initiatiedocument (bijlage III) is gebruikt bij het bedenken van de schermen. Daarnaast heb ik gekeken welke functionaliteiten in welk scherm komen te staan. Ook heb ik bestaande applicaties bekeken zoals Fastned, The New Motion, Laadpunten en PlugSurfing. Die apps heb ik als referentie gebruikt. Het heeft geholpen bij het bedenken van de benodigde schermen en bij het bedenken van de navigatiestructuur.

Het design wordt conform de richtlijnen van iOS en Android ontworpen. Belangrijke elementen staan linksboven en mindere rechtsonder, zoals te zien is in Figuur 6.10. Klantenbranding wordt toegepast door: bedrijfskleuren terug te laten komen in de tekstkleur en knoppen, de achtergrondkleur in de bedrijfskleur(en) en het bedrijfslogo wordt niet op elk scherm getoond. Het

laatste levert een positieve bijdrage aan de gebruikerservaring. Eindgebruikers hebben hierdoor het gevoel dat zij een native applicatie gebruiken. Het scherm wordt zo optimaal gebruikt.

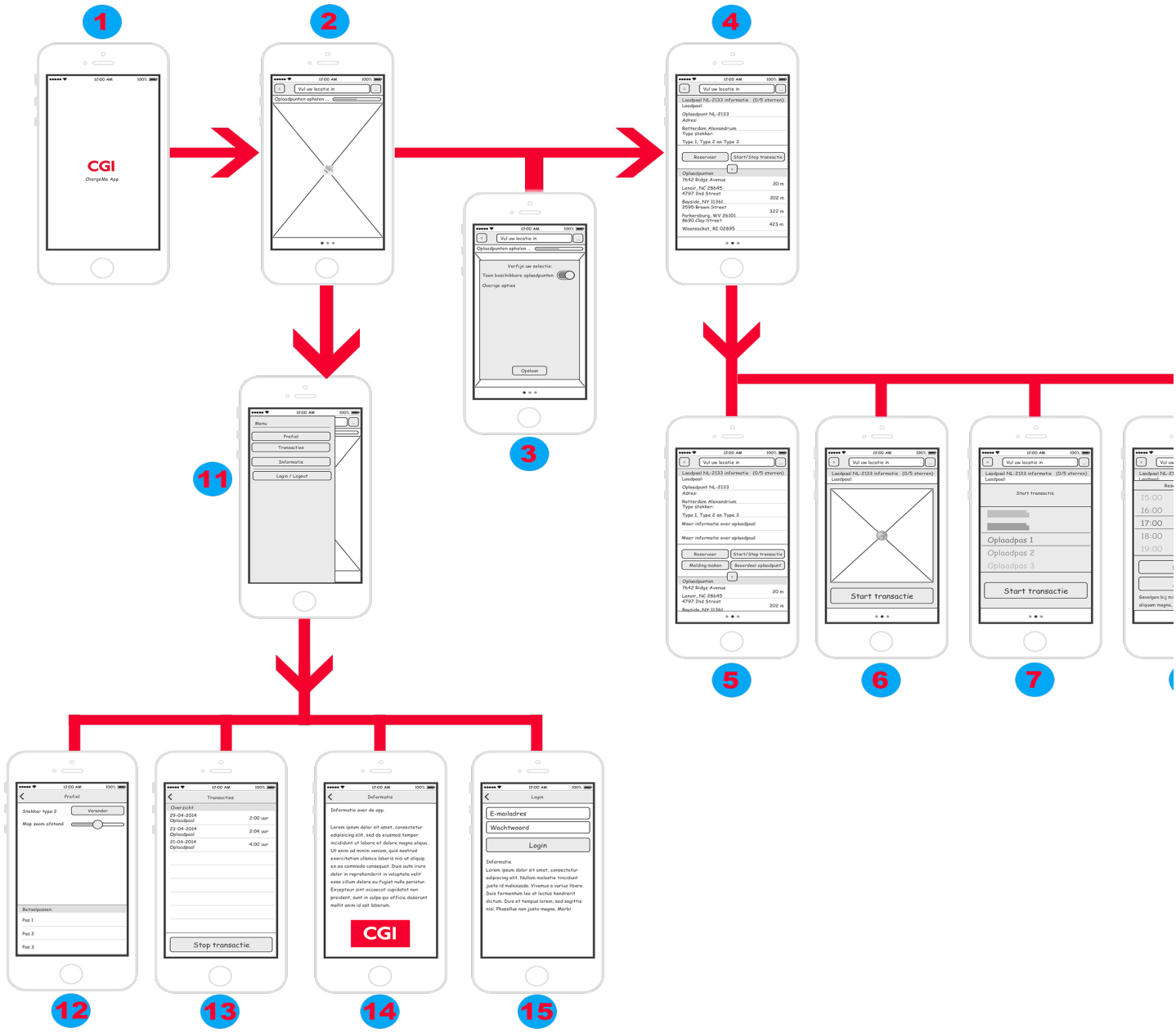


Figuur 6.10 : **Beste plaatsing van elementen op het scherm**

De opdrachtgever vindt het belangrijk dat ChargeMe ergonomisch in gebruik is en gebruiksvriendelijk. Ook moet de app een heldere en professionele uitstraling hebben. Android en iOS hebben beide een andere uitstraling. Daarom heb ik besloten om de richtlijnen van beide platformen te bestuderen. Deze kennis kan ik vervolgens toepassen in het ontwerp. Dit draagt bij aan de eerder benoemde eigenschappen waar de app aan moet voldoen. Android (z.j.) en Apple (z.j.) bevelen onder ander het volgende aan:

- plaats belangrijke elementen links boven en mindere rechtsonder;
- laat het bedrijfslogo niet op elk scherm zien;
- wijzig het uiterlijk van elementen niet drastisch;
- plaats niet teveel knoppen/iconen achter elkaar;
- voeg klantenbranding onoverzichtelijk toe.

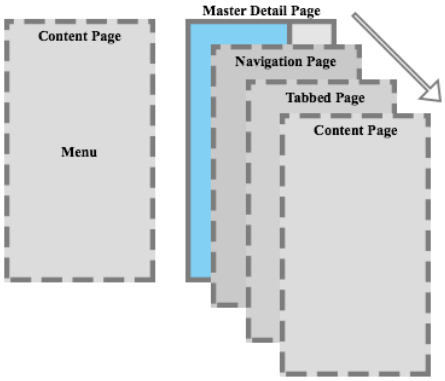
Het storyboard is goedgekeurd door Franc Buve. Hij was positief over het resultaat. Hierna kan er begonnen worden aan het implementeren. Het resultaat is te zien in Figuur 6.11. De schermen worden ook apart getoond en omschreven in het Sprint Backlog document. Dit is opgenomen in bijlage VI.



Figuur 6.11 : Storyboard van ChargMe app

6.4.2 Storyboard implementeren

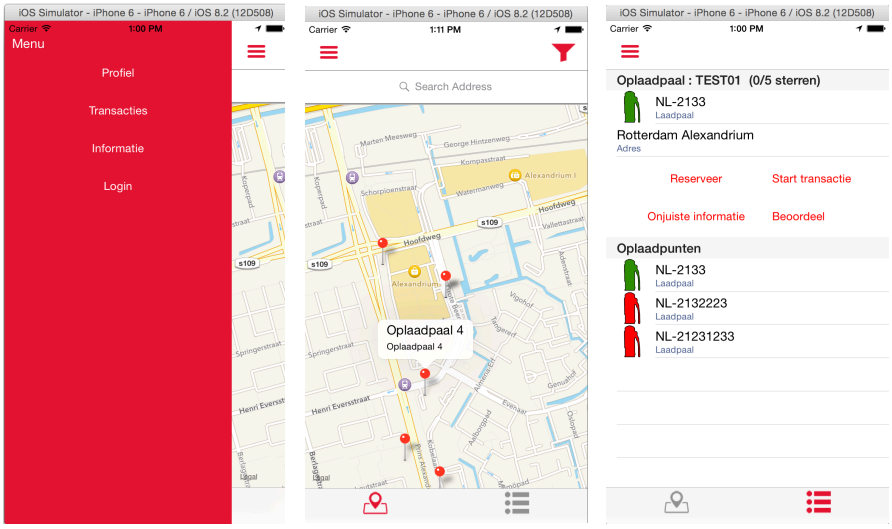
Met het storyboard worden de schermen geïmplementeerd. Verschillende type pagina's worden gebruikt. De 'root' pagina is een 'Master Detail Pagina'. Dit is een paginavariant die Forms onder ander aanbiedt. Deze pagina heeft twee kinderpagina's. Een content pagina voor het menu en een navigatiepagina voor het detailscherm. De navigatiepagina geeft de basis functionaliteit voor onderliggende pagina's. Deze hebben een knop waarmee het menu geopend kan worden. Ook kan de gebruikers naar rechts slepen op scherm om het menu te tonen. Het daar onderliggende scherm is een 'Tabbed Page'. Het dient als basis voor de tab-pagina's. Het hoofd- en detailscherm staan daarop. Figuur 6.12 weergeeft de hiërarchie van de pagina's.



Figuur 6.12 : **Hiërarchie van de paginastructuur**

Deze structuur heb ik gebruikt zodat alle onderliggende pagina's altijd het menu kunnen bereiken. De tab-pagina heb ik gebruikt omdat het hoofd- en detailscherm nauw met elkaar verbonden zijn. Voor de gebruiker is het dan duidelijk dat ze bij elkaar horen. Ik heb eerst een carrouselpagina gebruikt. Dit werkte echter niet goed omdat de gebruiker maar een klein oppervlakte heeft om naar rechts of links te vegen. Elementen zoals de geografische map verhinderen de veegherkenning van het apparaat. Tevens zorgde het voor verwarring omdat naar rechts slepen het menu opent.

Het resultaat is te zien in Figuur 6.13. Alleen de navigatie tussen de schermen werkt. De actie- en menuknoppen doen nog niks. Het hoofdscherm werkt wel omdat dit in een eerdere sprint is gerealiseerd.



Figuur 6.13 : **Resultaat sprint drie**

6.5 Sprint vier - Talen ondersteuning en CiMS integratie

Sprint vier beschrijft de integratie met CiMS en talenondersteuning. In een gesprek met Franc Buve besproken we de overdraagbaarheid. De app wordt verkocht aan buitenlandse klanten. Dat houdt in dat de teksten in Charge*Me* vaak van taal veranderen. Het gemakkelijk kunnen wijzigen van de taal

draagt bij aan de overdraagbaarheid. Ik heb besloten om een nieuwe story met hoge prioriteit toe te voegen om dit te behandelen.
Allereerst heb ik gecontroleerd of het geen problemen oplevert met de deadline. Een nieuwe story met hoge prioriteit levert geen problemen. Ik geef dit voorrang op de andere story's omdat het gebruikt gaat worden in de hele app. Achteraf kost deze story implementeren meer tijd. Alle bestaande schermen waar tekst instaat, moet dan worden aangepast.

In subparagraaf 6.4.3 komt de themaklasse aanbod. Met deze klasse kan de styling voor de app gecentraliseerd worden. Vervolgens wordt in subparagraaf 6.4.4 ondersteuning voor meerdere talen ontworpen en geïmplementeerd. Tot slot wordt de koppeling met CiMS gelegd in subparagraaf 6.4.5.

Tabel 6.7 : **Sprint vier user story's**

Story's	Taken
Als developer wil ik een theme class ontwikkelen waarin de basis kleuren van elementen worden gedefinieerd, zodat niet ieder element apart geconfigureerd hoeft te worden.	• Basis klasse schrijven die de gebruikte elementen een basis styling geeft. • Configuraties aanmaken in het XML-bestand. • Configuraties uitladen en toepassen in de basis klasse.
Als opdrachtgever wil ik dat de app multi-language ondersteund, zodat klanten die de app overnemen gemakkelijk van taal kunnen veranderen.	• Config loader uitbreiden. • App Config file voorzien van een config waar de taal geselecteerd kan worden + file name. • Language file maken die gebruikt wordt bij inladen van teksten.
Als developer wil ik een koppeling maken met CiMS, zodat data uit dat systeem gebruikt kan worden in de app.	• Sequence diagram tekenen. • Service layer uitbreiden met CiMS koppeling.

6.5.1 Standaard themaklasse

Allereerst heb ik een basisklasse geschreven waarin elementen vooraf voorzien worden van een styling. Deze configuraties worden uit het configuratiebestand geladen. Dit heb ik gedaan door gebruik te maken van Style. Style is een object van Forms. Hiermee kunnen door middel van databinding de eigenschappen van visuele elementen ingesteld worden. Met databinding wordt een connectie gelegd tussen een property en een object. De kleurcodes en andere instellingen worden geplaatst in het XML-bestand. Deze worden uitgelezen en onder ander toegepast in deze klasse.

De klasse heb ik geschreven met de bedoeling om styling te centraliseren. Het voorkomt ook dubbele code. Wat uiteindelijk weer tijd scheelt. Elementen nemen standaard de kleurcodes en instellingen over van het device. Elk element zou, zonder deze klasse, opnieuw van styling voorzien moeten worden. Ook hoeft niet elke pagina aangepast te worden als er een wijziging in het uiterlijk is.

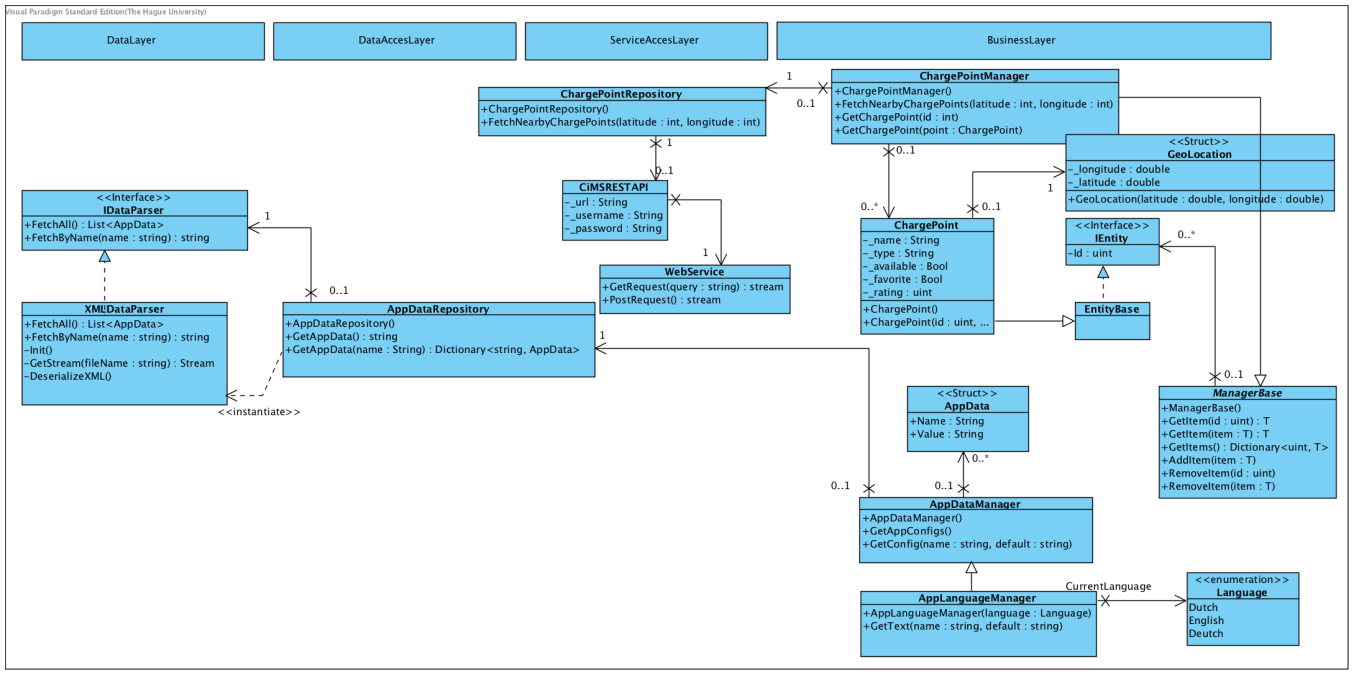
De meeste elementen kunnen geconfigureerd worden. Ik liep echter hier ook tegen de limitaties van Forms op. Niet alle elementen kunnen voorzien worden van een styling. Ook niet alle bindings werken juist. Het probleem heb ik allereerst voorgelegd op het developerforum van Xamarin. Daar ontving ik geen reacties op en binnen CGI kon de vraag ook niet beantwoord worden. Ik besloot vervolgens om een custom renderer te gebruiken bij die elementen. Met de custom renderer maakte ik een implementatie per platform. Daarin paste ik de styling vervolgens toe.

Het resultaat is een klasse die ingeladen wordt in de mainmethode. Hierdoor hoef ik elementen niet meer te configureren op kleur, grote, etc. Het toevoegen van nieuwe pagina's en elementen gaat nu vele malen sneller.

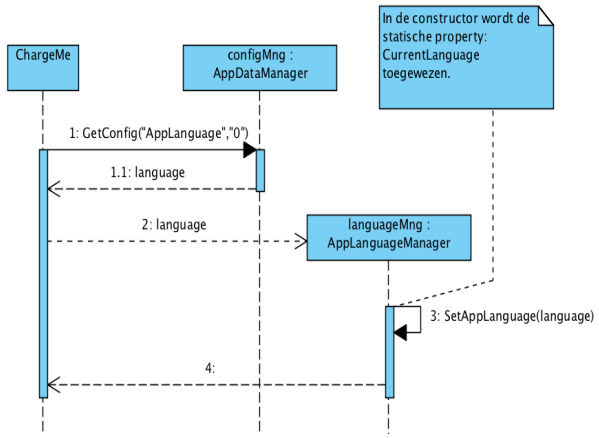
6.5.2 Meerdere talen ondersteuning

Vervolgens heb ik functionaliteit geprogrammeerd waarmee de app van taal kan veranderen. De app is aangevuld met, voor nu, drie extra XML-bestanden. Deze worden gevuld met taalconfiguraties voor Engels, Nederlands en Duits. De bestaande functionaliteit waarmee configuraties ingeladen worden, heb ik gewijzigd. Zo is onder ander de naamgeving veranderd. iConfigParser en

XMLConfigParser veranderen van naam zodat zij generiek zijn toe te passen. Nu impliceert de naam dat alleen configuratiebestanden uitgelezen kunnen worden.
Ook wordt de singleton verwijderd uit de configuratie-repository. Verder wordt een nieuwe manager aangemaakt: de taalmanager. Deze erft over van de configuratiemanager. Dit is te zien in Figuur 6.14. Als laatst wordt een enumeratie toegevoegd. De numerieke waarde van deze enumeratie worden gebruikt in het configuratiebestand. Hiermee wordt de taal geselecteerd.



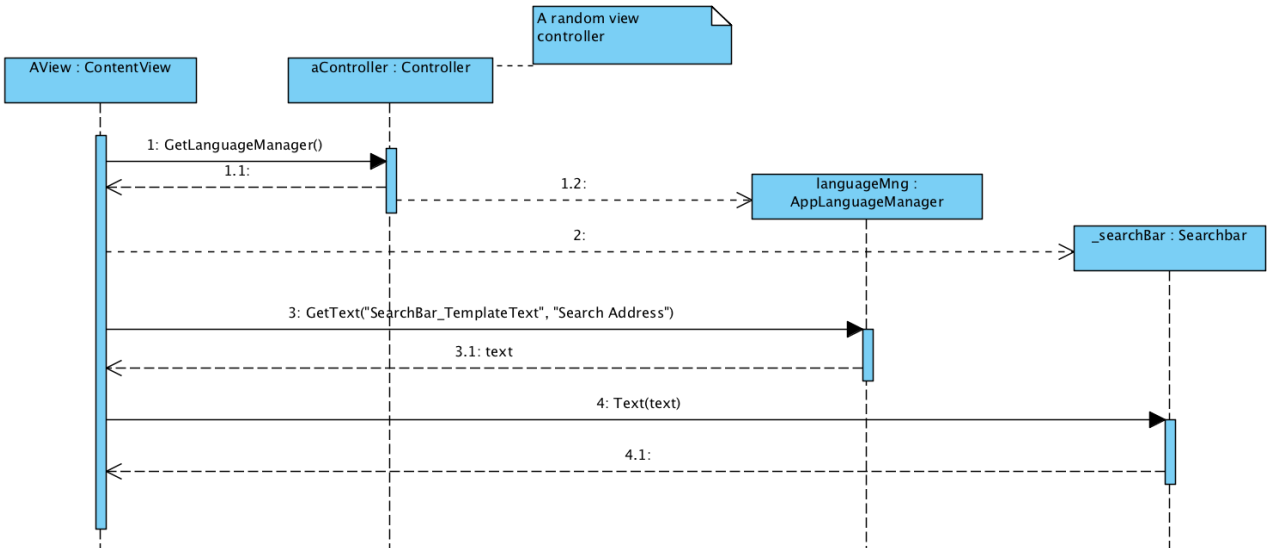
Figuur 6.14 **: Klassendiagram sprint drie**



Figuur 6.15

Naast het ontwerpen van het klassendiagram, is ook het sequentiediagram gemodelleerd. Het initialisatieproces heb ik gemodelleerd omdat het afhankelijk is van de gekozen taal. De gekozen taal wordt in de mainmethode opgevraagd. Dat gebeurt door de AppDataManager aan te roepen. De aanroep stuurt een nummer terug dat vertaald wordt naar de enumeratie met talen. Hierna wordt de taalmanager aangemaakt. Deze bezit een statische eigenschap. Daarin wordt de gekozen taal aangegeven. Latere instanties van deze klasse hoeven dan niet opnieuw de taal op te vragen.

In Figuur 6.16 is ook het opvragen van een regel tekst te zien. Vanuit een willekeurige content pagina kan via een controller de LanguageManager aangeroepen worden; ik heb een base-controller-klasse aangemaakt die standaard een language- en configuratiemanager initialiseren. Vanuit de content pagina kan de methode ‘GetText’ aangeroepen worden. Als parameter wordt de code meegegeven die overeenkomt met die uit het taalbestand. Ook wordt een standaard tekst opgegeven. Deze wordt teruggestuurd als tijdens het uitlezen van het taalbestand een fout optreedt. Vervolgens kan de tekst die teruggestuurd is, gebruikt worden om bijvoorbeeld de tekst van het zoekvenster te vullen.



Figuur 6.16 : **Sequentiediagram ophalen tekst per taal**

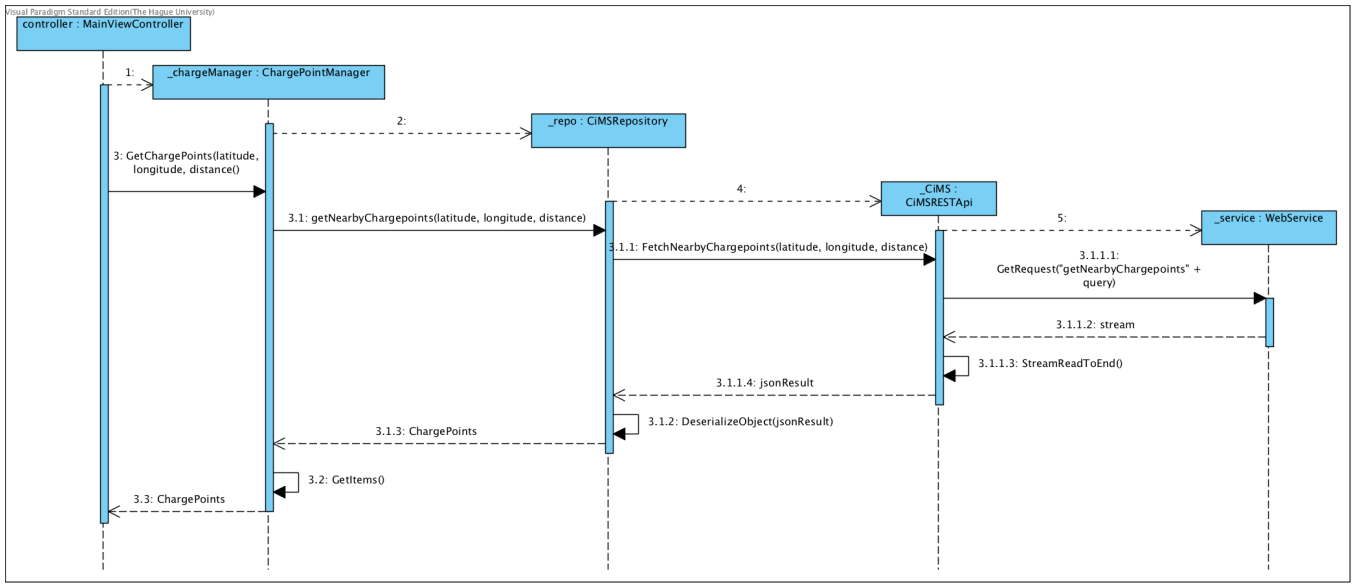
Tijdens de implementatie heb ik geen problemen ondervonden. Het resulteerde in een systeem waarmee gemakkelijk teksten opgehaald kunnen worden. Het project is aangevuld met drie XML-bestanden. Hierin worden de teksten opgenomen met ieder hun eigen id. Deze zijn in alle taalbestanden gelijk.

6.5.3 CiMS koppeling

De laatst uitgevoerde user story is een koppeling maken met CiMS. Met die koppeling heb ik de eerste aanvraag toegevoegd: het ophalen van laadpalen. Daarvoor moet de locatie van de gebruiker bekend zijn. De aanvraag heeft als parameters de lengte- en hoogtegraad nodig. Deze vraag ik op door de GPS-voorziening van het apparaat te gebruiken.

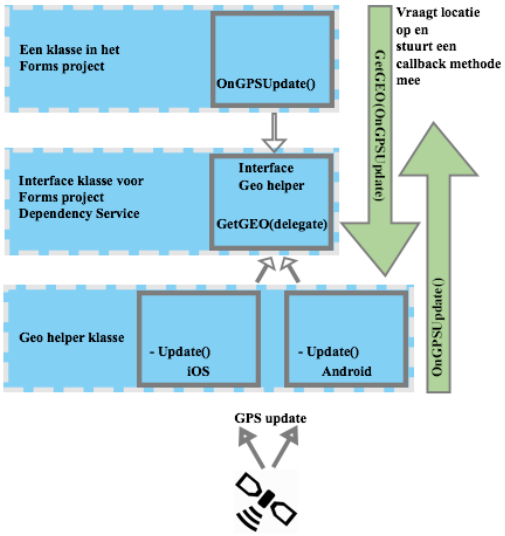
Ten eerste heb ik het klassendiagram aangepast. De interface CiMSAPI heb ik weggehaald omdat een ander type koppeling met CiMS onwaarschijnlijk is. De klasse die daarvan overerfde, is losgekoppeld. Deze is hernoemt naar: CiMSRESTAPI. Ook is er een nieuwe WebService klasse aangemaakt. Die handelt de web gerelateerde aanvragen af. Dit is te zien in Figuur 6.14 in het vorige paragraaf. De repository blijft hetzelfde. Die vertaalt de data, die in JSON formaat wordt ontvangen, naar een lijst van oplaadpunten.

Daarnaast heb ik een sequentiediagram gemaakt. In Figuur 6.17 is te zien hoe de koppeling met CiMS wordt gebruik om laadpalen op te halen. De oplaadpuntenmanager vraagt aan de repository de oplaadpunten. Die spreekt op zijn beurt de CiMSRESTAPI klasse aan. De generieke GET-aanvraag wordt gebruikt om de laadpalen op te halen. Als parameter wordt de action meegestuurd en de query. De WebSerivce klasse stuurt altijd een stream terug zodat later ook andere data formaten gebruikt kunnen worden. CiMSRESTAPI klasse leest de stream en stuurt het in tekstformaat terug. Het is de taak van de repository om de tekst te vertalen. Hiervoor maak ik gebruik van de ‘NuGet Package’ JSON.NET. NuGet is een platform voor .NET. Daarmee kunnen gemakkelijk pakketten toevoegd worden. JSON.NET kan onder andere tekst vertalen naar een object van het type dat wordt aangegeven. Een lijst van de objecten, van het type ChargePoint, wordt teruggestuurd. Het hoofdscherm beschikt al over de implementatie om deze lijst te tonen.



Figuur 6.17 **: Sequentiediagram voor het ophalen van oplaadpunten.**

Ten slotte heb ik de locatie van het apparaat achterhaald. Dat doe ik met de GPS-voorziening. De lengte- en breedtegraat moet achterhaalt worden omdat het een vereiste parameter is bij het ophalen van de laadpalen. Deze aanvraag levert de lijst met laadpalen op basis van coördinaten en een radius. Voor Android en iOS heb ik een GPS-locatie helper klasse gemaakt. Door middel van de Depency Service (een interface) kan ik die aanspreken in het gedeelde project. Het is voorzien van een delegate om de locatiewijzingen door te geven aan de aanvrager. Figuur 6.18 geeft dit proces weer.



Figuur 6.18 **: GPS locatie opvragen en update callback**

De webservice klasse heb ik zelf geschreven omdat geen bibliotheken beschikbaar zijn voor de Xamarin versie die ik gebruik. Ik had twee oplossingsmogelijkheden: een oudere versie gebruiken of zelf een klasse schrijven die de aanvragen verstuurt en afhandelt. Teruggaan naar een oudere versie lijkt mij niet verstandig. Er ontstaat risico dat andere packages of codes niet meer werken. Ik besloot om zelf de klasse te schrijven omdat ik over de benodigde kennis beschik van Transmission Control Protocol (TCP) en de Hypertext Transfer Protocol (HTTP) methodes GET, POST, etc.

Het resultaat is een CiMS koppeling met een generieke webservice-klasse die bruikbaar is voor eventueel andere koppelingen. Via deze koppeling worden laadpalen opgevraagd op basis van de huidige positie van het apparaat. Het systeem wordt op de hoogte gesteld wanneer het apparaat van positie verandert. Dit wordt gedaan door de delegate. Hierdoor wordt een nieuwe lijst met laadpalen opgehaald die gebaseerd is op zijn nieuwe locatie.

6.6 Sprint vijf - Oplaadpunten filteren

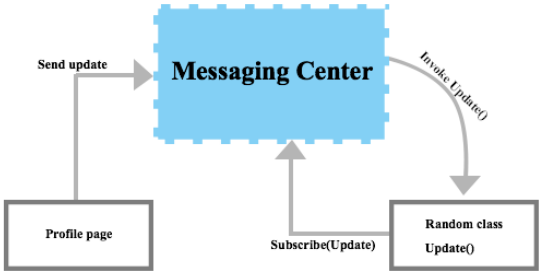
In sprint vijf ligt de focus op het filteren van oplaadpunten. Als eerst wordt de bestaande filter- en profielscherm uitgebreid met instellingmogelijkheden. Hierna wordt gekeken hoe deze instellingen opgeslagen kunnen worden. Vervolgens wordt een ontwerp bedacht voor het opslaan, inladen en toewijzen van gemarkeerde en beoordeelde laadpalen. Het hoofdstuk wordt afgesloten met de implementatie voor het melden van onjuiste informatie.

Tabel 6.8 : **User story's sprint vijf**

Story's	Taken
Als gebruiker wil ik kunnen kiezen tussen het tonen van enkel beschikbare oplaadpalen en ook niet beschikbare	• Instellingen scherm uitbreiden met aan-uit switch voor het tonen van beschikbare oplaadpunten die alleen beschikbaar zijn. • Map en lijst filteren op beschikbare oplaadpunten. • Testen of alleen beschikbare oplaadpunten worden getoond. • Instellingen lokaal opslaan en weer inladen.
Als gebruiker wil ik oplaadpunten kunnen markeren als favoriet.	• Na denken over het design en dit modeleren. Hoe wordt de lokale data opgeslagen? • Ontwerp implementeren. • Testen.
Als gebruiker wil ik oplaadpunten kunnen beoordelen.	• Detail scherm uitbreiden met beoordeling. • Beoordeling lokaal opslaan. • Beoordelingen inladen en synchroniseren met de oplaadpunten die opgehaald worden vanuit de API. • Lijst met oplaadpunten rangschikken op basis van afstand en beoordeling.
Als gebruiker wil ik een melding kunnen maken van onjuiste informatie.	• Een scherm aanmaken waarin de gebruiker een bericht kan schrijven.

In deze sprint heb ik de profielpagina gemaakt. In de profielpagina kan de gebruiker instellingen configureren. Voor deze sprint is het een switch-knop waarmee laadpalen getoond kunnen worden op hun beschikbaarheid. Ook kunnen laadpalen gemarkeerd worden als favoriet of voorzien worden van een ranking. Deze data wordt lokaal op het device opgeslagen. Het opslaan kan in XML of JSON. De data wordt bij het opstarten weer ingeladen en gesynchroniseerd.

Als eerst heb ik de instellingenpagina gemaakt. Dit is een content pagina die te openen is vanuit het menu. Een switch-knop is daaraan toegevoegd. Deze knop is gekoppeld aan de 'Messaging Service'. Deze service versimpelt het gebruik van delegates. Hiermee kunnen callbacks geregistreerd worden voor een event met lage koppeling. Zo concludeert ook Xamarin: *"The MessagingCenter is a simple way to reduce coupling, especially between view models. It can be used to send and receive simple messages or pass an argument between classes"* Xamarin (z.j.). In figuur 6.19 wordt dat geïllustreerd.

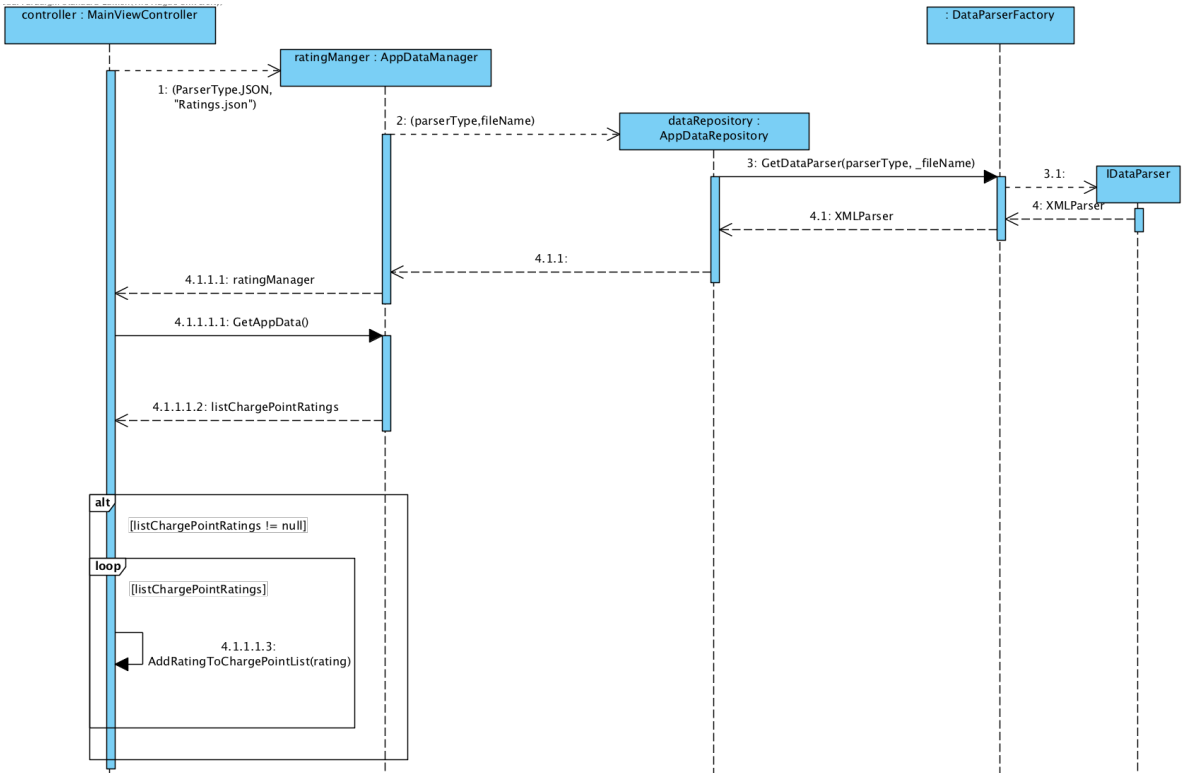


Figuur 6.19 : **Messaging Center werking**

Vervolgens is het beoordelen van een laadpaal en het markeren als favoriet geïmplementeerd. Voordat ik aan de implementatie begon, heb ik eerst een sequentiediagram gemaakt. In Figuur 6.20 is dat diagram te zien. Ik heb hiervoor de bestaande implementatie gebruik van de datamanager. Deze moet wel uitgebreid worden omdat het alleen bestanden kan uitlezen. Het moet nu ook data

gaan opslaan. In het sequentiediagram is te zien hoe dit gaat verlopen. Het gebruik van de datamanager gaat op dezelfde manier als eerder is beschreven in paragraaf 6.4.4. De manager wordt wel aangepast.

Naast XML wil ik ook de mogelijkheid hebben om JSON-bestanden te lezen en opslaan. Door middel van het factory designpatroon wil ik dit bewerkstelligen. Bij de aanmaak van de manager wordt een parser type - een enumeratie - meegestuurd. De DataParserFactory ontvangt het type. De factory maakt vervolgens de correcte parser aan en stuurt het terug. Eenzelfde implementatie is gebruikt voor de favorieten.



Figuur 6.20 **: Sequentiediagram beoordelen oplaadpunt**

Ten slotte moet de lijst gesynchroniseerd worden en gerangschikt. Het synchroniseren gebeurt bij het opstarten en na een update. Via de favoriete- en ratingmanager wordt eerst de data ingeladen. Deze wordt vervolgens gesynchroniseerd met de lijst van laadpalen. Hierna wordt de lijst gefilterd. Ook het veranderen van de instellingen roept opnieuw de filtermethode aan. Bij het ophalen worden de oplaadpunten gefilterd. Het filteren gebeurt met LINQ. Hiermee kunnen gemakkelijk query's worden afgevuurd op een lijst. Het implementatievoorbeeld is te zien in Figuur 6.21. Als eerst wordt er gekeken of de gebruiker alleen beschikbare laadpalen wil tonen. Vervolgens worden de laadpalen, waarvan de status niet gelijk is aan 'Free', uit de lijst verwijderd. Filterinstellingen waarmee potentieel de lijst kleiner wordt, worden eerst uitgevoerd. Het geheel van de resterende lijst moet namelijk altijd gerangschikt worden. Een kleinere lijst rangschikken is sneller. Als laatst wordt de lijst geordend op afstand, beoordeling en favorieten.

```
/// Filters the charge points...
private void FilterChargePoints(ref Dictionary<string, ChargeStation> chargeStations)
{
    if (Settings.AvailableSwitch)
        chargeStations = chargeStations.Where(p => p.Value.ChargePoint.Any(c => c.Status == "Free"))
            .AsEnumerable().ToDictionary(k => k.Value.StationID, k => k.Value);

    if(Settings.ChargePlugType != Settings.ChargePlugDefault)
        chargeStations = chargeStations.Where(p => p.Value.ChargePoint.Any(c => c.PlugType == Settings.ChargePlugType))
            .AsEnumerable().ToDictionary(k => k.Value.StationID, k => k.Value);

    chargeStations = chargeStations.OrderBy(x => x.Value.Distance).OrderByDescending(
        d => d.Value.Rating
    ).OrderByDescending(
        fav => fav.Value.Favorite
    ).AsEnumerable().ToDictionary(k => k.Value.StationID, k => k.Value);
}
```

Figuur 6.21 **: Codevoorbeeld filteren chargepoints**

Sprint vijf heeft geresulteerd in een aantal nieuwe functionaliteiten. De app is voorzien van een profielpagina. Daarin kan ingesteld worden om alleen beschikbare laadpalen te tonen. Verder kunnen laadpalen gemarkeerd worden als favoriet of voorzien worden van een beoordeling. Deze data wordt op het apparaat opgeslagen in JSON-formaat. De configuratiemanager kan gemakkelijk wijzigen tussen XML en JSON. Dit heb ik gedaan door gebruik te maken van het factory designpatroon. Verder worden de laadpalen gefilterd en gesynchroniseerd. Bij het filteren wordt de lijst gefilterd op beschikbaarheid. Vervolgens wordt het gerangschikt op afstand van het device, rating en favorieten. Deze lijst wordt getoond in het detailscherm in die volgorde.

In deze sprint stond ook de story gepland voor het melden van onjuiste informatie. Deze heb ik niet gehaald omdat mij gevraagd was een presentatie te geven. De presentatie ging over de app en vond plaats tijdens de teammeeting. Het is goed verlopen. De presentatie gaf ik voor een groep van 40 á 50 werknemers. Van hen heb ik veel positieve reacties ontvangen. De user story voor het melden van onjuiste informatie, is verplaatst naar de volgende sprint.

6.7 Sprint zes - Koppeling met een nieuw platform genaamd CRM

In sprint zes komt een nieuwe webservice. Deze koppelt met het systeem waar onder ander de gegevens en transactiehistorie van pashouders instaan. Dat platform heet CRM. Maar eerst wordt de user story uit de vorige sprint ontworpen en geïmplementeerd. Hierna wordt de profielpagina beschreven. Vervolgens wordt de koppeling gemaakt met CRM. Hierdoor kan de in- en uitlogfunctionaliteit gerealiseerd worden. Ten slotte wordt het klassendiagram opnieuw bekeken en waar nodig aangepast. Daar komen een paar wijzingen uit die de code verbeterd.

Tabel 6.9 **: User story's sprint zes**

Story's	Taken
Als gebruiker wil ik een melding kunnen maken van onjuiste informatie.	• Een scherm aanmaken waarin de gebruiker een bericht kan schrijven.
Als gebruiker wil een profielpagina hebben die ik kan instellen	• Profielpagina voorzien van elementen. • Instellingen opslaan en inladen. • Betaalpassen ophalen en tonen in een lijst.
Als gebruiker van de app wil ik kunnen aanmelden als ik een pashouder ben.	• CRM koppeling ontwerpen. • Service layer uitbreiden met CRM implementatie. • Model voor gebruiker bedenken en implementeren. • Login scherm implementeren. • Uitlog scherm implementeren.

Het melden van onjuiste informatie was in de vorige sprint niet gelukt. Dat wordt als eerste opgepakt. Gebruikers kunnen bij elk geselecteerde laadpaal een melding versturen. Het mailprogramma van het device wordt geopend en is vooraf gevuld met een template. De gebruiker kan direct een e-mail versturen of een bericht bijvoegen. De interne mailprogramma's van het device kunnen geopend worden vanuit de standaard bibliotheek van het device. Bij het aanroepen van de opstartmethode, kunnen parameters meegeven worden. Hiermee vul ik e-mailbericht vooraf met de benodigde informatie (het template).

Er waren drie mogelijkheden: de gebruiker doorsturen naar een web formulier, CiMS API uitbreiden met een call zodat in de app een formulier ingevuld kan worden of het gebruik van de interne mailprogramma. De drie mogelijkheden heb ik voorgelegd aan de opdrachtgever. CGI had geen programmeurs beschikbaar die de CiMS API kon uitbreiden. De gebruiker doorsturen naar een webpagina was ook geen optie. Een bericht versturen via het mailprogramma had de voorkeur.

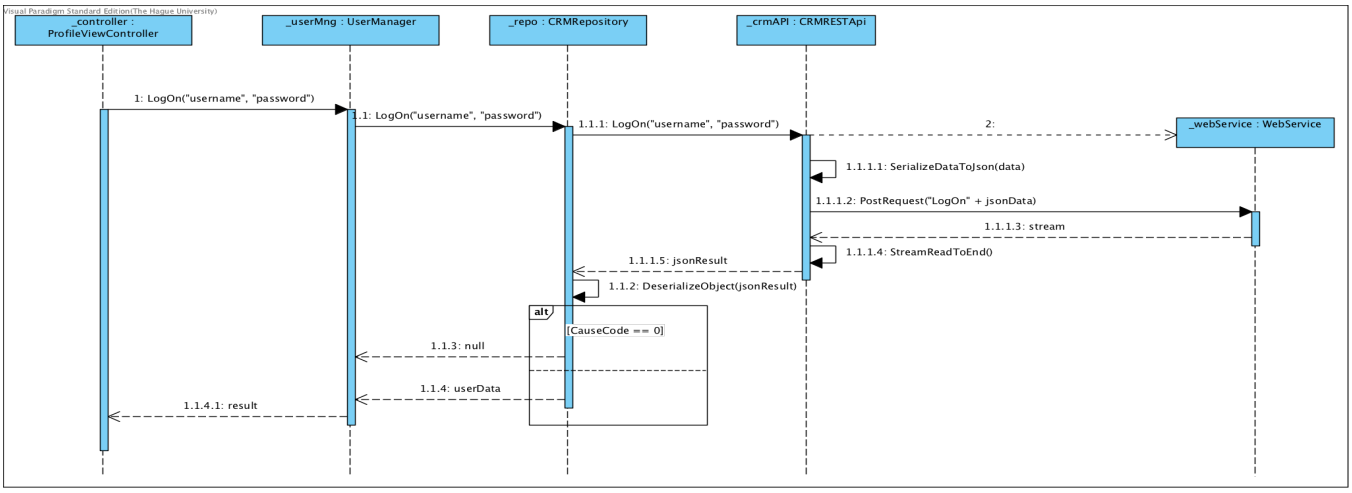
Hierna heb ik de profielpagina verder uitgebreid. De pagina is voorzien van een filter waarmee het type oplaadstekker geselecteerd kan worden. Ook worden de betaalpassen van de gebruiker getoond. De betaalpassen worden alleen getoond als de gebruiker is ingelogd. Dit wordt gerealiseerd nadat de CRM koppeling is ontwikkeld. Het ontwerpproces en implementatie komen overeen met de aanpak en uitvoer van sprint vijf.

Vervolgens is er een koppeling gemaakt met het CRM platform. Door de koppeling is het mogelijk voor de gebruiker om aan- en af te melden. Ook heb ik het klassendiagram aangepast en opgeschoond.

CRM koppeling heb toevoegt aan de ServiceAccesLaag. Die erft over van de nieuwe RESTAPI base-klasse. Deze heb ik toegevoegd voor een lage koppeling met webservice klasse. CRM werkt met POST-aanvragen. De POST-methode had ik al eerder aangemaakt maar implementatie was nog niet nodig. Dat heb ik deze sprint toevoegt.

Nu de CRM koppeling is gelegd, kan het inlogproces ontworpen worden (Figuur 6.22). Via de CRM koppeling wordt een POST-aanvraag naar de REST API gestuurd. Het proces verloop voor een groot deel hetzelfde als die van de GET-aanvraag. De webservice stuurt een fout code met het nummer nul terug als de inloggegeven onjuist zijn of bij een foutieve aanvraag. Hier controleer ik op. De gebruikersdata wordt terug gestuurd als er geen fout code is. De gebruikersdata wordt bijgehouden in een nieuwe manager.

De UserManager bewaard die data en dient als koppeling met UI en applicatie laag. Er moesten ook twee nieuwe model klassen aangemaakt worden, namelijk User en CPCard (Figuur 6.25). Deze klasse heb ik gemaakt nadat ik de ontvangen gebruikersdata heb geanalyseerd.

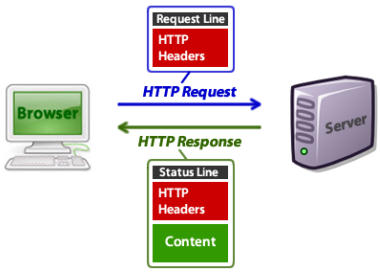


Figuur 6.22 **: Sequentiediagram aanmelden gebruiker**

De CRM koppeling is verder uitgebreid met de volgende aanvragen:

- afmelden
- ophalen gebruikers betaalhistorie
- ophalen betaalpassen

Deze aanvragen werken alleen als de cookie, die wordt teruggestuurd in de loginaanvraag, wordt meegestuurd. Er bleek echter een probleem te zijn met het .NET HttpWebResponse object. Deze zou de cookie object moeten ontvangen na het inloggen. De cookie-property was echter leeg. Ik heb vervolgens alle data onderzocht die ik terug kreeg. De cookie bleek terug gestuurd te worden in de header. De header van een request is voorzien van allerlei informatie over de browser, aangevraagde pagina, de server, etc. In Figuur 6.23 is te zien dat de header een belangrijk onderdeel is van http pakketten.



Figuur 6.23 : **HTTP headers (Bron: http://code.tutsplus.com/tutorials/http-headers-for-dummies--net-8039)**

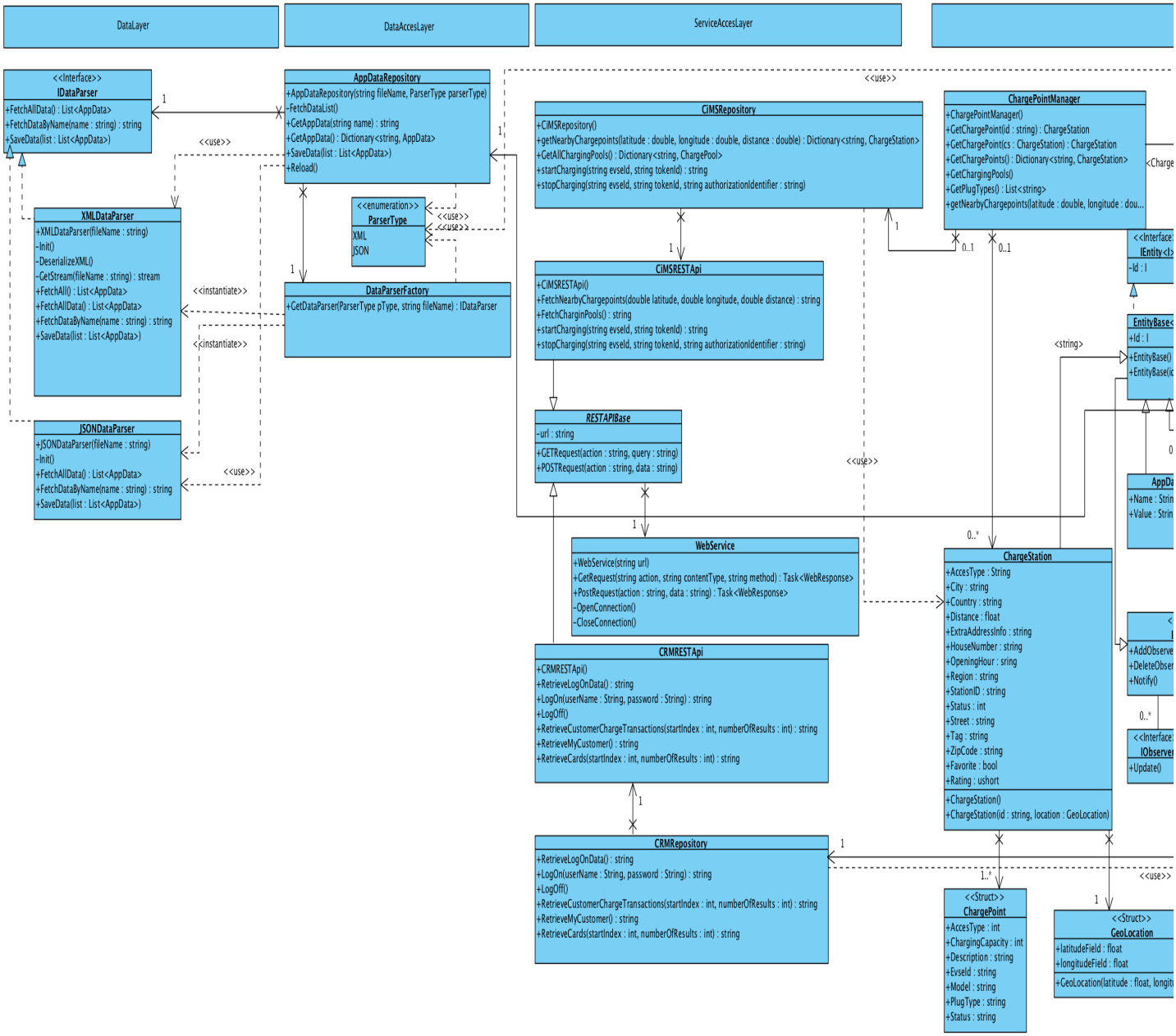
De cookie moet ik uit de header vertalen. In Figuur 6.24 is te zien hoe ik dat gedaan heb. Een cookie is te herkennen aan het sleutelwoord: 'Set-Cookie'. Dat woord wordt gezocht in de header. De data van de cookie wordt opgeslagen in een aparte string nadat een match is gevonden. De data is echter een lange brei aan tekst. Deze split ik naar een array. Vervolgens maak ik gebruik LINQ om het verder te splitsen naar de sleutel- en datawaarde. De eerste sleutel is altijd het unieke nummer van de cookie. Hiermee maak ik zelf een cookie-object aan. Vervolgens haal ik het pad en verloopdatum uit de cookiedata en vul die in het zojuist aangemaakte cookie-object.

```
/// Tries the get cookie from headers...
private Cookie TryGetCookie(HttpWebResponse response)
{
    Cookie myCookie = null;

    string[] headers = response.Headers.AllKeys;

    if(headers.Contains("Set-Cookie"))
    {
        string cookieData = response.Headers["Set-Cookie"];
        string[] cookieDataSplitted = cookieData.Split(';');

        var cookieDict = cookieDataSplitted.Select(s => s.Split('='))
            .ToDictionary(key => key[0].Trim(), value => value[1].Trim());

        myCookie = new Cookie(cookieDict.First().Key, cookieDict.First().Value);
        string dateTime;
        string path;

        cookieDict.TryGetValue("path", out path);
        cookieDict.TryGetValue("expires", out dateTime);

        myCookie.Path = path;
        myCookie.Expires = Convert.ToDateTime(dateTime);
    }

    return myCookie;
}
```

Figuur 6.24 : **Codevoorbeeld cookie uit header halen**

Ten slotte heb ik het klassendiagram bijgewerkt. Onder ander de AppDataManager is aangepast. Deze erft over van de base-manager omdat deze de zelfde functionaliteit heeft als alle andere managerklassen.

Het resultaat van deze sprint zijn een aantal nieuwe functionaliteiten. Gebruikers kunnen melding maken van onjuiste informatie. Hiervoor wordt het interne mailprogramma gebruikt. Ook heeft de gebruiker een uitgebreider profielpagina gekregen. Daarin kan het type oplaadstekker van zijn elektrische auto geselecteerd worden. Hierna worden alleen de laadpalen getoond die dat type stekker bezitten. Ook worden zijn betaalpassen getoond nadat er is ingelogd. Met de koppeling naar CRM kan de gebruiker inloggen en uitloggen. Deze koppeling wordt ook gebruikt om de betaalhistorie en betaalpassen op te halen. Verder is het klassendiagram onder de loep genomen. Ik heb gekeken welke verbeteringen doorgevoerd konden worden in het diagram en in de code.



Figuur 6.23 : **Klassendiagram sprint zes**

6.8 Sprint zeven - Reservatie en transacties starten of stoppen

In sprint zeven staat het starten en stoppen van een transactie centraal. Ook wordt de reservatie van een oplaadpunt mogelijk. Verder kan de gebruiker zijn betaalhistorie inzien in een nieuwe pagina. Als eerst wordt de API-documentatie geschreven omdat het als hulpmiddel wordt gebruikt bij het starten en stoppen van een transactie. Dat wordt gedaan in paragraaf 6.7.1. Daarna wordt het lastige vraagstuk voor het starten en stoppen van een transactie beschreven in paragraaf 6.7.2. Met die kennis kan vervolgens een systeem bedacht worden voor de reservatie. Dat wordt gedaan in paragraaf 6.7.3. In paragraaf 6.7.4 wordt kort gesproken over de transactiehistorie pagina en het resultaat van dit alles staat in paragraaf 6.7.5.

Tabel 6.10 : **User story's sprint zeven**

Story's	Taken
Als gebruiker met een betaalpas, wil ik een oplaadpunt kunnen reserveren.	• Reservering mogelijkheid onderzoeken en ontwerpen. • Reservering implementeren. • Reservering testen.
Als gebruiker wil ik een transactie kunnen starten vanuit de app.	• Transactie starten ontwerpen voor pashouders. • Transactie starten ontwerpen voor niet-pashouders. • Transactie starten implementeren (pashouders). • Transactie starten implementeren (niet-pashouder). • Testen.
Als gebruiker wil ik een transactie kunnen stoppen vanuit de app.	• Functionaliteit implementeren. • Testen.
Als gebruiker wil ik een overzicht van mijn transacties kunnen inzien.	• Transactiepagina vullen met het transactieoverzicht van pashouder. • Melding weergeven voor gebruikers die niet aangemeld zijn. • Testen.

6.8.1 API documentatie

In het afstudeerplan was er afgesproken dat ik samen met een member van CGI de API documentatie zou definiëren. Bij aanvang van de opdracht bleek echter dat een collega in Eindhoven al een mobiele interface had ontwikkeld. Deze moest ik gaan gebruiken voor dit project. Tekortkomingen in die interface kon ik definiëren. Deze zouden door een member ontwikkeld worden. In de vorige sprint is gebleken dat uitbreidingen niet mogelijk zijn omdat er geen programmeurs beschikbaar zijn. Uitbreindingen aan die interface vallen buiten de scope van deze opdracht.

Tot op heden heb ik de sourcecode bestanden gebruikt van die interfaces. Daaruit achterhaalde ik de beschikbare methodes en hun parameters. Het opleveren van een gedefinieerde API is onderdeel van deze opdracht. Dit had echter een lagere prioriteit. Aangezien als volgens planning verloopt en het beschrijven van de API mij helpt bij de vervolg story's, heb ik besloten om het eerst te definiëren. In Figuur 6.24 is te zien hoe de aanvragen zijn omschreven. De API-documentatie is opgenomen in bijlage XI en XII.

@ GETNEARBYCHARGEPOINTS

Title	*Get chargepoints nearby given latitude, longitude and distance*
URL	/getNearbyChargepoints?latitude=:latitude&longitude=:longitude&radius=:radius
Method	GET
URL Params	Optional: latitude=[float] example: latitude=0.2 longitude=[float] example: longitude=0.5 radius=[float] example: radius=100

Success Response	Code: 200
	Content:
	[
	{
	"AccesType": "Private",
	"ChargePoint": [
	{
	"AccesType": 1,
	"ChargingCapacity": 9,
	"Description": "",
	"EvseId": "DE*SWD*MENDUS002:1",
	"Model": "Mennekes-ACUV3",
	"PlugType": "Iec_62196_2_Type_2",
	"Status": "Free"
	},
	}
	]

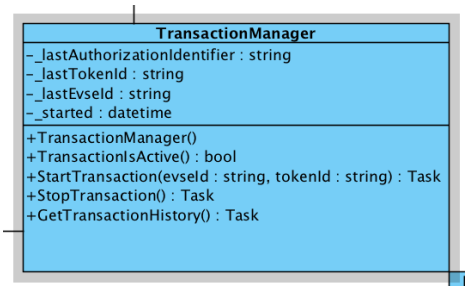
Error Response	None
Notes	*All chargingpoints are returned when no parameters are used.*

Figuur 6.24 : **API documentatie**

6.8.2 Transactie starten en stoppen

Als eerst heb ik de functionaliteit gemaakt voor het starten en stoppen van een transactie. Een onderscheid is gemaakt tussen pashouders (aangemelde gebruikers) en niet-pashouders. Pashouders kunnen gemakkelijk een transactie starten nadat zij een laadpaal en een betaalpas hebben geselecteerd. Niet-pashouders doorlopen een ander proces. Zij worden eerst doorgestuurd naar de betaalomgeving van Adyen. Daar wordt bedrag x gereserveerd. Aan het eind van de transactie wordt het verschuldigde bedrag afgeboekt. Beide type gebruikers kunnen met één klik de transactie stoppen.

Figuur 6.25



Allereerst heb ik bedacht hoe pashouders een transactie kunnen starten. Uit de API-documentatie blijkt dat drie variabelen nodig zijn: kaartnummer, oplaadpuntnummer en autorisatiecode. Een transactiemanager wordt aangemaakt voor die data en voor de aanroepen.

Het starten van een transactie voor een pashouder is niet lastig. De gebruiker kan een oplaadpunt selecteren en op 'start transactie' klikken. In een vervolg scherm vraag ik welke zijde van het oplaadpunt gebruikt wordt en met welke betaalpas betaald wordt. Hierna zijn de benodigde nummers om de transactie te starten bekend. De aanvraag geeft een tokennummer terug die nodig is om de transactie weer te stoppen. Deze wordt daarom in de manager vastgehouden.

Vervolgens kijk ik naar het starten voor niet-pashouders. De start-transactie methode heeft een pas- en paalnummer nodig. Niet-pashouders hebben geen pas en dus ook geen pasnummer. Franc vertelde mij dat in plaats van het kaartnummer ook een ander nummer meegestuurd kan worden.

Door middel van een creditcard betaling zouden deze gebruikers ook een transactie kunnen starten. Via een online betaalomgeving zou de gebruiker kunnen betalen en het verificatienummer daarvan kan in plaats van het pasnummer verstuurd worden. Pasnummer zijn via een bepaald formaat opgemaakt waardoor transacties via een creditcard betaling herkend kunnen worden.

De opdrachtgever wou dat ik gebruik maak van Adyen. Dat is een online betalingsdienst. Op hun website heb ik een account aangemaakt. In de app wordt de gebruiker via een internetframe genavigeerd naar de betaalomgeving. Hier doorlopen zij het betaalproces. Een bedrag, dat ingesteld wordt op de website van Adyen, wordt vervolgens gereserveerd. Adyen stuurt een code terug als de betaling is voldaan. Met dit nummer kan ik vervolgens de transactie starten.

Een aantal problemen ontstonden. De betaalomgeving kan niet zomaar benaderd worden. Hiervoor moest ik eerst een signatuur generen. Deze moet meegestuurd worden bij de benadering. Deze handtekening wordt gemaakt met een 'hmac' sleutel en 'signing' string. De sleutel bestaat uit cijfers. Deze sleutel staat op de profielpagina van Adyen. De 'signing' string is een lange lap met tekst waar de volgende variabelen respectievelijk achter elkaar worden geplakt: betaling, valuta, verstuur datum, sessie, e-mailadres van winkel, winkel referentie, toegestane methodes, geblokkeerde methodes en de offset. Met de HMAC sleutel en signing string kan ik vervolgens de benodigde SHA-1 signatuur genereren.

Ook kan de website geen event afvuren in de app als de betaling is voldaan. URL-shemes bleek de oplossing te zijn (Moschetti, 2014). Adyen kan wel een speciale URL terugsturen naar de browser. Deze URL onderschept het device. Per device moet de methode die zulke URL's afhandelt, gewijzigd worden. In Figuur 6.26 is te zien hoe dit bij iOS wordt gedaan. De host wordt vergeleken met een uniek nummer. De query uitgelezen als dit overeenkomt. Met het berichtencentrum van Xamarin wordt vervolgens een event afgevuurd.

```
//custom url sheme.
public override bool OpenUrl (UIApplication application, NSUrl url, string sourceApplication, NSObject annotation)
{
    // custom stuff here using different properties of the url passed in
    if(url.Host.Equals("com.chargeme.payment"))
    {
        string query = url.Query;
        Dictionary<string,string> dicQueryString =
            query.Split('&')
                .ToDictionary(c => c.Split('=')[0],
                    c => Uri.UnescapeDataString(c.Split('=')[1]));

        var transaction = new Transaction();
        transaction.authResult = dicQueryString["authResult"];
        transaction.merchantReference = dicQueryString["merchantReference"];
        transaction.skinCode = dicQueryString["skinCode"];
        transaction.shopperLocale = dicQueryString["shopperLocale"];
        string psp = "";
        dicQueryString.TryGetValue("pspReference", out psp);
        transaction.pspReference = psp;
        transaction.merchantSig = dicQueryString["merchantSig"];

        MessagingCenter.Send<Transaction> (transaction, "transaction_adyen");
    }

    return true;
}
```

Figuur 6.26 : Codevoorbeeld URL-scheme

Het laatste probleem was het tokennummer. Dat verdwijnt wanneer de app opnieuw wordt opgestart; het staat opgeslagen in memory. Hierdoor kan een transactie niet meer gestopt worden. Om dit op te lossen wordt de data van de transactiemanager op het device opgeslagen. De manager leest deze data in tijdens het opstarten.

Ten slotte is het stoppen van een transactie geïmplementeerd. Het ontwerp- en implementatieproces is gelijk aan die van het starten.

6.8.3 Oplaadpunt reserveren

Nu bekend is hoe een transactie gestart wordt. Kan de reserveerfunctie bedacht worden. Het reserveren heeft veel overeenkomsten met het starten van een transactie. Beide zullen het kaart- en oplaadpuntnummer nodig hebben. Het onderscheid dat gemaakt moet worden, is dat de transactie nog geen kosten in rekening brengt. Het CiMS platform moet hiervoor uitgebreid worden met een webservice aanroep. Ook zou log en beheer mogelijkheden nodig zijn in CiMS.

Het reserveren van een oplaadpunt is momenteel niet realiseerbaar. Het CiMS platform beschikt niet over de nodige functionaliteit om een oplaadpunt te reserveren. Vanwege de kosten en lastige vraagstukken die bij een reserveersysteem komen kijken, staat dit momenteel niet op de planning als uitbreiding aan het CiMS platform.

Het platform uitbreiden valt buiten de scope van deze afstudeeropdracht. Aangezien het reserveren van een oplaadpunt wel een kenmerkend onderdeel is van de afstudeeropdracht, heb ik het reserveringssysteem wel geïmplementeerd in de app. De functionaliteit wordt gebouwd alsof de CiMS webservice beschikt over een call voor een reservatie. De data wordt echter lokaal opgeslagen. Hierdoor kan de reservatiefunctionaliteit wel getoond worden in de demonsteerbare versie. De benodigde methode heb ik ook uitgewerkt in bijlage XI.

6.8.4 Overzicht transacties

De CRM koppeling was al voorzien van de methode om de transactiehistorie op te vragen. Deze wordt gebruikt bij het tonen van de lijst in een nieuwe pagina. Een pagina waarin de gehele historie staat van de gebruiker. Het design en ontwerpproces is het zelfde verlopen als in sprint zes.

6.8.5 Resultaat

Sprint zeven heeft veel nieuwe en lastige functionaliteiten opgeleverd. Gebruikers

hebben een transactiepagina erbij gekregen waarin zij hun transactiehistorie kunnen inzien. Daarnaast kunnen ze vanuit de app een transactie starten voor een oplaadpunt. Zowel pas- als niet-pashouders kunnen dit doen. Deze kunnen zij ook weer stoppen. De gebruiker hoeft alleen nog maar de stekker te verbinden met de auto. Verder kunnen ze een oplaadpaal reserveren voor een aantal minuten. Deze functionaliteit werkt echter alleen in de demoversie omdat CGI niet over het budget beschikt om dit te realiseren.

6.9 Sprint acht - De interface

In sprint acht ligt de focus op de interface van de app. Het uiterlijk wordt kritisch bekeken en waar nodig aangepast. Als eerst ga ik op bugs testen omdat ik de licentie van Xamarin heb ontvangen. Hierna worde de iconen vervangen. Vervolgens wordt het uiterlijk van de app verbeterd. Ook heb ik wijzingen doorgevoerd die de gebruiksvriendelijkheid verbeterd. Daarna heb ik twee bugs opgelost van de vorige sprint. Tot slot wordt de app verstuurd naar Chris Schayk. Hij gaat de UI beoordelen en levert feedback. Ook wordt de app verstuurd naar andere members om de gebruikerservaring te testen.

Tabel 6.11 : **User story's sprint acht**

Story's	Taken
Bug testen op iOs en testen op Android device	• Testen op Android. • Testen op iOS.
De geo positie staat op nul als de locatie van de gebruiker niet opgehaald kan worden.	• Startpositie aanmaken die configureerbaar is via de app config
Op iOs toestemming vragen om gebruik te maken van de gebruikerspositie	• Bij het opstarten vragen om toestemming, opdat de huidige locatie opgevraagd kan worden.
Als opdrachtgever wil ik dat de app zelfsturend in het gebruik ervan is.	• App laten testen door anderen om te kijken of hieraan voldaan is
Als opdrachtgever wil ik dat de interface een heldere en professionele uitstraling heeft.	• Laad icoon tonen wanneer de app aan het laden is • Huidige interface laten beoordelen door Chris • Iconen maken voor de app

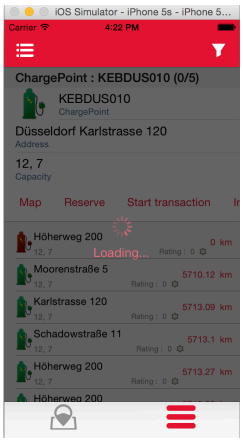
Voorheen heb ik de app alleen kunnen testen in de simulator. In sprint acht heb ik een licentie bemachtigd van Xamarin. Hierdoor kon ik eindelijk beginnen met het testen op een apparaat. Ik beschikte echter niet over een toestel met Android. Testen hierop verliep voornamelijk via de simulator. Dit resulteerde in een behoorlijke lijst met bugs. In bijlage XIII staan alle bugs opgesomd. Een groot gedeelde van die bugs heb ik kunnen oplossen. De overige heb ik verplaatst naar de uitloop week omdat de resterende story's een hogere prioriteit hebben.

Vervolgens is de interface kritisch bekeken. Iconen heb ik zelf gecreëerd. Ook heb ik een laadscherm gemaakt die gemakkelijk is toe te passen in de gehele app. Tot slot heb ik de schermen aangepast zodat het professioneler en duidelijker oogt.

De iconen heb ik gemaakt met behulp van Photoshop. Vanwege de grote variatie in scherm groottes, zijn alle iconen in drie verschillende formaten gemaakt. Iconen worden getoond op basis van de schermresolutie van het apparaat. In Figuur 6.27 zijn de iconen te zien.

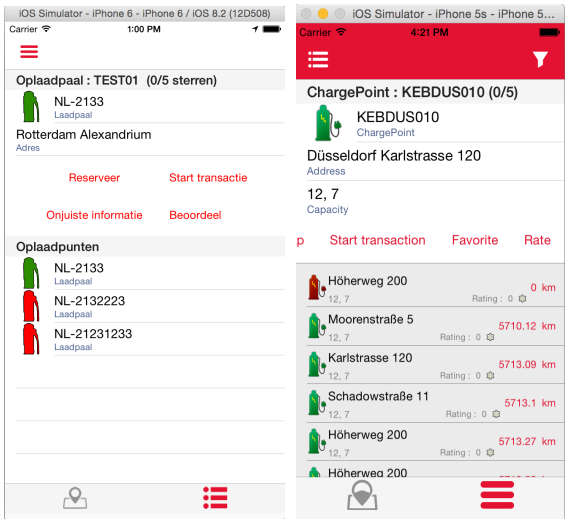
Figuur 6.27 : **Iconen in de app**

Voor het tonen van een laadscherm wou ik een systeem maken dat gemakkelijk te gebruiken is. De root page wordt voorzien van een overlay view. Door middel van twee methoden kan de overlay getoond en verborgen worden. Hierdoor kan bij elke gewenste functionaliteit de overlay getoond worden. In Figuur 6.28 is te zien hoe de overlay eruit ziet.



Figuur 6.28 : **Loading overlay**

De app krijgt ook de nodige wijzingen. Zowel in uiterlijk als toevoeging van extra functionaliteiten. Op de detailpagina, met de lijst van alternatieve oplaadpunten, zijn meerdere wijzigingen doorgevoerd. Het onderscheid tussen de details met bijhorende actieknoppen en de oplaadpuntenlijst, is verduidelijkt. Dat heb ik gedaan door de achtergrond wat donkerder te maken. Ook is een scheidingslijn toegevoegd. Tevens is de indeling van de cells en afbeeldingen aangepast. Oplaadpunten die geen beschikbare oplaadpunten hebben zijn rood. Als één daarvan gereserveerd is en de andere niet beschikbaar dan is de afbeelding geel. Een groen plaatje wordt getoond als één van de stekkers beschikbaar zijn.
De plaatsing van de knoppen wordt ook veranderd. Ze worden nu horizontaal geplaatst en kunnen versleept worden naar links of recht. Knoppen die enkel voor ingelogde gebruikers beschikbaar zijn, worden verborgen. Deze worden getoond zodra de gebruiker inlogt. In Figuur 6.29 zijn een deel van de veranderingen te zien. Het linker scherm weergeeft de oude interface en het rechter de nieuwe.



Figuur 6.29 : **Interface wijzingen**

Verder heb ik in deze sprint twee bugs opgelost van de vorige sprint. De locatie van de gebruiker werd niet altijd correct opgehaald. Dit komt omdat de gebruiker hier eerst toestemming voor moet geven. Bij het opstarten van de app - alleen de eerste keer - krijgt de gebruiker een melding. Die melding vraagt de gebruiker of ChargeMe de GPS-functionaliteit mag gebruiken. Ook heb ik het configuratiebestand aangevuld met een standaard lengte- en breedtegraad. Deze wordt gebruikt als de gebruiker de vorige melding heeft afgewezen of wanneer het ophalen van de GPS-positie mislukt. In de demoversie wordt dan de map van Nederland getoond; normaliter wordt positie 0 gebruikt. Dan wordt een oceaan weergegeven.

Ten slotte heb ik de app verstuurd naar Chris Schayk. Hij gaat de UI van de app beoordelen nu alle schermen zijn geïmplementeerd. De feedback die ik van hem krijg, wordt in de volgende sprint

verwerkt. Daarnaast is de app naar een aantal members binnen CGI gestuurd om de gebruikerservaring te toetsen.

6.10 Sprint negen - Performance en security

De performance en beveiliging kan nu gecontroleerd worden omdat alle functionaliteiten gerealiseerd zijn. Hiermee heb ik tot het eind gewacht omdat ik dan de staat waarin de app nu verkeerd, test. In paragraaf 6.91 wordt eerst de feedback van Chris verwerkt. Hem had ik gevraagd de UI de beoordelen en feedback te leveren in sprint acht. Vervolgens wordt in paragraaf 6.9.2 de performance getest. Paragraaf 6.9.3 gaat over de beveiliging.

Tabel 6.12 **: User story's sprint negen**

Story's	Taken
Als opdrachtgever en teamleider wil ik dat de communicatie tussen CiMS en de APP beveiligd is, zodat misbruik voorkomen wordt.	• Veiligheid testen communicatie CiMS en app. • App zelf testen op veiligheid
Als opdrachtgever wil ik dat de app op het 3G en 4G netwerk snel werkt	• Testen hoelang het duurt om data op verschillende schermen in te laden. • Wanneer nodig wijzigingen aanbrengen die de performance verbeteren.
Als developer wil ik de feedback van Chris verwerken	• Tutorial pagina aanmaken waar de gebruiker met informatie wordt voorzien over de app. • Tutorial pagina openen bij eerste opstarten van de app.

6.10.1 Feedback Chris

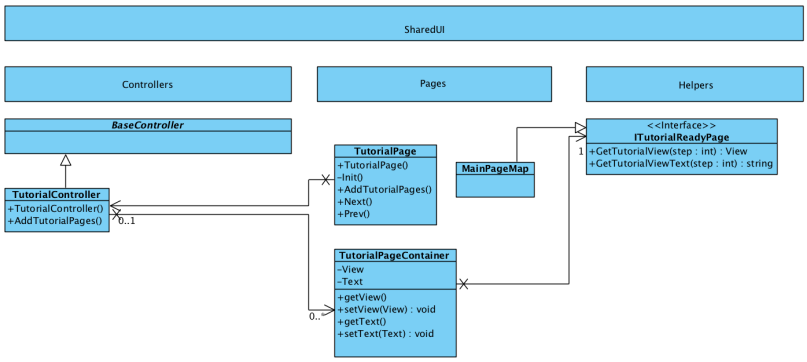
De feedback heb ik bij aanvang van sprint negen, de laatste sprint, ontvangen. Chris Schayk heeft de UI beoordeeld en feedback geleverd. Hij was positief over het resultaat. Zijn reactie is hieronder te lezen:

> *"Hallo Justin,*
> *Vandaag toch nog wat tijd gevonden. Om te beginnen, voor een app ziet ie er goed uit. Animaties zijn vloeiend en de afbeeldingen zijn scherp (ook voor retina).*
> *Mij is echter niet duidelijk wat de app precies doet en wat er van mij als gebruiker verwacht wordt.*
> *De navigatie is duidelijk, zoals Apple het gedocumenteerd heeft. Dus goed bezig! Hier en daar nog wat afbeeldingen die niet lekker staan, qua uitlijning. Het is basic, hier zou je nog wat extra aandacht aan kunnen spenderen, maar zoals je al aangaf gaat het hier om een demo voor een klant, dus geen zorgen.*
> *Conclusie; ziet er netjes en verzorgd uit, alleen er mag wat uitleg bij voor de gebruiker (misschien gaat het om een specifieke doelgroep, die de kennis al bezit welke nodig is).*
> *Met vriendelijke groet,*
> *Ing. Chris van Schayk | Software Engineer".*

In zijn bericht is te lezen dat hij positief is over de UI. De app ziet er goed uit en is naar de maatstaven van Apple gebouwd. Overgangen verlopen vloeiend en afbeeldingen zijn scherp. Hij had een opmerking over de uitlijning, maar dat is voor een demo versie niet erg. Het was hem echter niet direct duidelijk waarvoor de app dient. Zoals hij zelf ook al opmerkte, 'misschien gaat het om een specifieke doelgroep', waarmee hij doelt op de EV-rijder. Zij hebben meer verstand van het onderwerp en termen als 'chargepoint' kennen zij. Desalniettemin min lijkt het mij verstandig om wat met de feedback te doen.
Ik heb drie dagen gereserveerd voor het verwerken van de feedback. Ik heb besloten om een tutorialsysteem te maken omdat wijzingen aan UI niet nodig zijn. De app wordt aangevuld met een tutorialsysteem waar gemakkelijk pagina's aangemaakt kunnen worden om de gebruiker door de app te lijden en informeren.

De tutorial wordt getoond in een aparte pagina. De tutorial krijgt een eigen pagina en controller. De tutorialpagina kan schermen presenteren in een kleinere view. De controller houdt een lijst bij met containers. Hierin staat een pagina die de interface ´ITutorialReadyPage´ heeft geïmplementeerd. Ook staat daarin de tekst die getoond wordt. Alle schermen, ook bestaande, kunnen getoond worden

in het tutorialscherm. Ze moeten alleen de bovengenoemde interface implementeren. In Figuur 6.30 is het klassendiagram te zien.

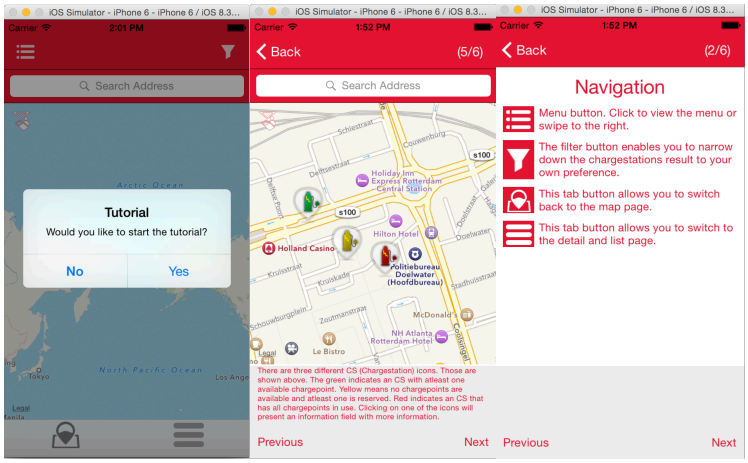


Figuur 6.30 : **Klassendiagram tutorial systeem**

Door deze methode kan een statisch scherm getoond worden, maar ook een interactieve. De paramater 'stap' is toegevoegd zodat een scherm meerdere stappen kan tonen. Het initiatie proces van het scherm, kan per stap verschillen.

Een tutorialsysteem is op veel verschillende manieren mogelijk. Er kan bijvoorbeeld een navigeerbare pagina getoond worden met plaatjes en tekst. Ook kan een tutorialscherm over de app zelf heen getekend worden. Stapsgewijs wordt elk scherm doorlopen. Het laatste is in mijn optiek, het mooist. Echter vergt zo´n systeem veel ontwikkeltijd. Een pagina dat plaatjes en tekst weergeeft, is sneller te implementeren, maar brengt een knelpunt met zich mee.
Klanten willen namelijk branding toevoegen. Daardoor veranderd het uiterlijk van de app. Een tutorialpagina met plaatjes ziet er dan anders uit, tenzij de klant elk plaatje vervangt met het nieuwe uiterlijk. Een tijdrovende klus die de overdraagbaarheid niet ten goede komt. Daarom heb ik besloten om boven genoemde systeem te realiseren. Het is een middenweg tussen beide genoemde opties.

Het resultaat is een tutorialsysteem die gemakkelijk uit te breiden is. De mogelijkheden zijn ook aanzienlijk omdat het initiatieproces per scherm bepaald kan worden. Het is aan de ontwikkelaar of hij een statisch scherm toont of een interactieve. Figuur 6.31 toont het resultaat.



Figuur 6.31 : **Tutorial pagina**

6.10.2 Performance

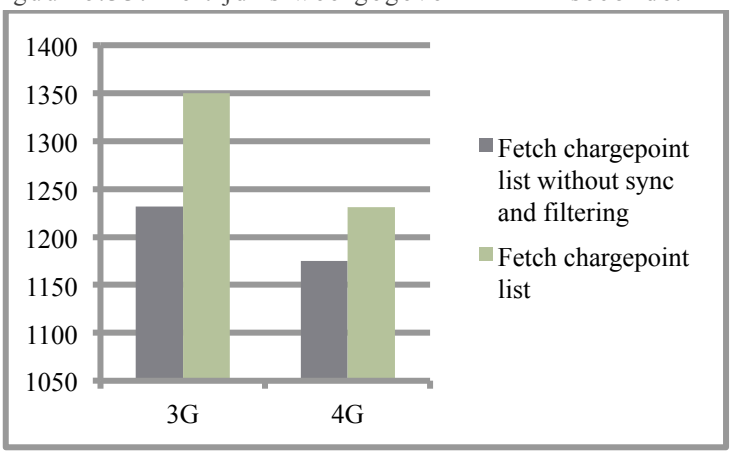
Een eis van de opdrachtgever is dat de app snel is. In de acceptatiecriteria, paragraaf 6.1.2, is afgesproken dat de laadpalen in maximaal drie seconden getoond worden op het 3G- en 4G netwerk. Om dit te kunnen testen heb ik eerst een methode geschreven waarmee de performance getest kan

worden. In Figuur 6.32 is de code te zien. Allereerst wordt de methode, die getest wordt, eenmalig uitgevoerd voordat de meting begint. Dit wordt gedaan voor een precieze meting. De eerste keer dat de methode wordt aangeroepen kan langer duren door bijvoorbeeld initiaties. Dat heeft effect op de duur.

```
public static void Start(string name, int reps, Action action)
    {
      _results = new double[reps];
      //warmup
      action();

      for (int i = 0; i < reps; ++i) {
        Stopwatch sw = Stopwatch.StartNew ();
        action();
        _results [i] = sw.Elapsed.TotalMilliseconds;
      }

      _min = _results.Min ();
      _max = _results.Max ();
      _avg = _results.Average ();
      _name = name;
    }
```

Figuur 6.32 : **Performance testmethode**

Vervolgens ben ik naar buiten gegaan en met de zelfde signaalsterkte de testen uitgevoerd op een mobielapparaat. Met een signaalsterkte van ongeveer 80 procent zijn 100 keer achterelkaar de laadpalen opgehaald. Deze test is vier keer gedaan. Tweemaal op 3G en twee keer op 4G. Een onderscheid is gemaakt tussen ophalen en ophalen met synchronisatie. Het gemiddelde van het resultaat is te zien in Figuur 6.33. De tijd is weergegeven in milliseconde.



Figuur 6.33 : **Resultaat performance test**

Het resultaat toont aan dat de oplaadpunten, met ongeveer 80 procent signaalsterkte, ruim binnen de drie seconden wordt opgehaald. Daarbij is het ook het vermelden waard dat deze test data verkregen is van de test server. Op de live omgeving zal de ophaaltijd nog lager zijn.

6.10.3 Security

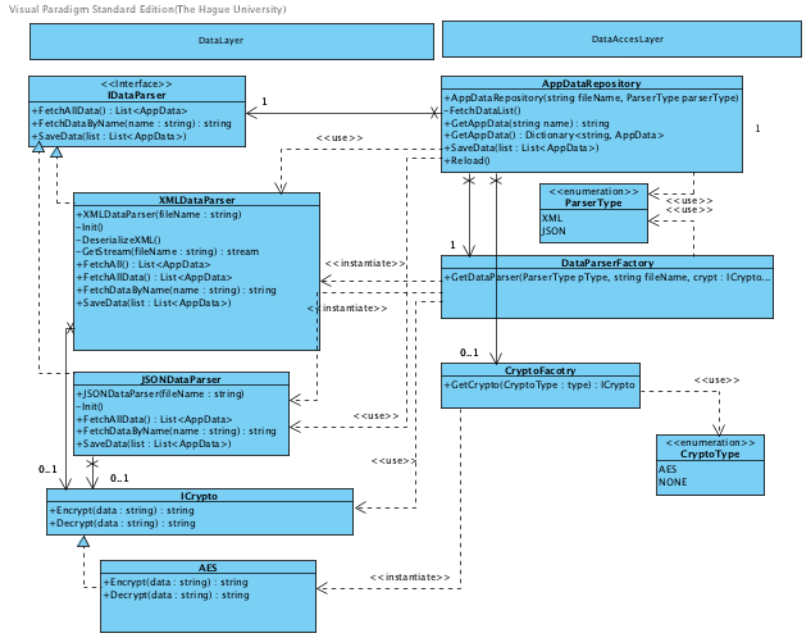
Communicatie tussen CiMS en de app moet beveiligd zijn. Communicatie met CiMS verloopt via het HTTPS protocol. Dit protocol versleuteld de data voordat het verstuurd wordt van of naar de webserver. Hierdoor is de communicatie tussen CiMS en de app veilig; het breken van SSL encryptie in HTTPS is vrijwel onmogelijk.
Een voorbeeld. Een pashouder kan zich aanmelden met zijn accountgegevens. Die gegevens worden verstuurt naar de server. De data wordt zonder SSL leesbaar verzonden over het net. Iedere computer die verbonden is aan dat netwerk kan de data uitlezen. In het voorbeeld hieronder is aan de linkerzijde te zien hoe niet-geencrypte data wordt verstuurd. Het rechter kolom toont de geencryptedata. Alleen de gene met de meestersleutel kan deze wirwar van tekens en cijfers ontcijferen.

Tabel 6.13 : **Encryptie voorbeeld**

Login:Justin *Password:Secret!1*	*MprWnARzn0PPFj21u8edxqsCXw51YTkZ4ZYYpSMoD1TOUl/eUx0VehdVRE+tqy/ibgah+lfuuWZ62Vy/q18fHfJ8vk00swOf36ORIJH2ocusnPtgaXgv4H+LWNK5/PfNsXJ89ct0eFPU38ffPntqPQ==*

Daaruit kunnen we concluderen dat de communicatie beveiligd is tegen ongewenste acties. De app moet ook beveiligd zijn. Functies die alleen beschikbaar zijn voor pashouders zijn al afgeschermd. Deze zijn niet te bereiken zolang de gebruiker niet aanmeld. De app slaat ook data op de telefoon op. Deze data kan wel uitgelezen en bewerkt worden. De opgeslagen data vormt momenteel geen beveiligingsrisico, omdat het voornamelijk over persoonlijke data gaat die niet verzonden wordt naar de server. Om wel de mogelijkheid te bieden om data veilig op te slaan op het device, wordt encryptiemogelijkheid opgenomen in de architectuur.

Figuur 6.34 toont een selectie uit het klassendiagram dat aangevuld is met cryptografie. Ik heb gebruik gemaakt van het 'factory' patroon, wat uitbreiding met andere crypto-grafische methodes vergemakkelijkt. Voor de test versie is alleen AES geïmplementeerd.



Figuur 6.34 : Klassendiagram encryptie

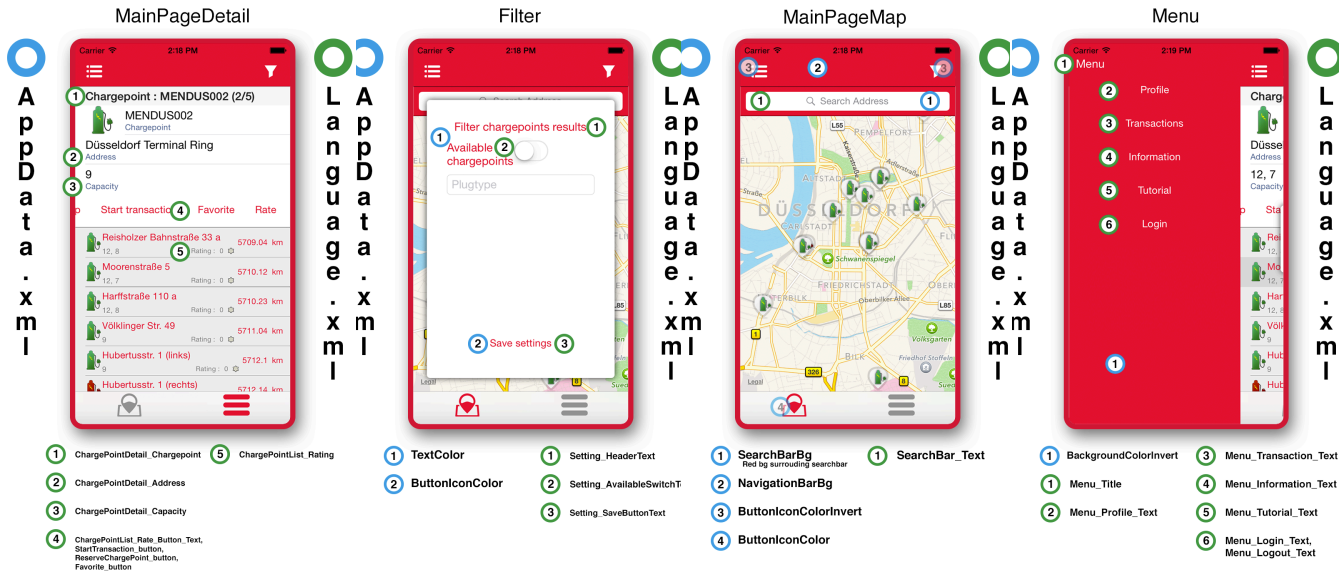
De verschillende dataverwerkers (parsers) kunnen de ICrypto implementeren om data eerst te encrypten en decrypten. Als de dataverwerker de cryptografie mogelijkheden heeft geïmplementeerd, kan de ontwikkelaar nog zelf bepalen om data te versleutelen of niet. Bij het declareren van een nieuwe AppDataManager moet de constructor voorzien worden van een aantal parameters. Pas wanneer een encryptietype wordt meegegeven en deze niet van het type 'NONE' is, dan wordt encryptie toegepast. Een voorbeeld:

var ratingManager = new AppDataManager(ParserType.JSON, "Ratings.json", CryptoType.AES);

6.11 Legacy

Met de tijd die ik nog over had, heb ik overdraagbaarheid geprobeerd te verbeteren. Alle code heb ik voorzien van commentaar, waar dit nog niet gebeurd was. Daarnaast heb ik ook voor elk scherm een afbeelding gemaakt. De afbeeldingen heb ik bewerkt zodat aangegeven wordt welke configuraties en tekstcodes gebruikt zijn. Hiermee kan CGI en klanten die het overnemen, gemakkelijk inzien welke

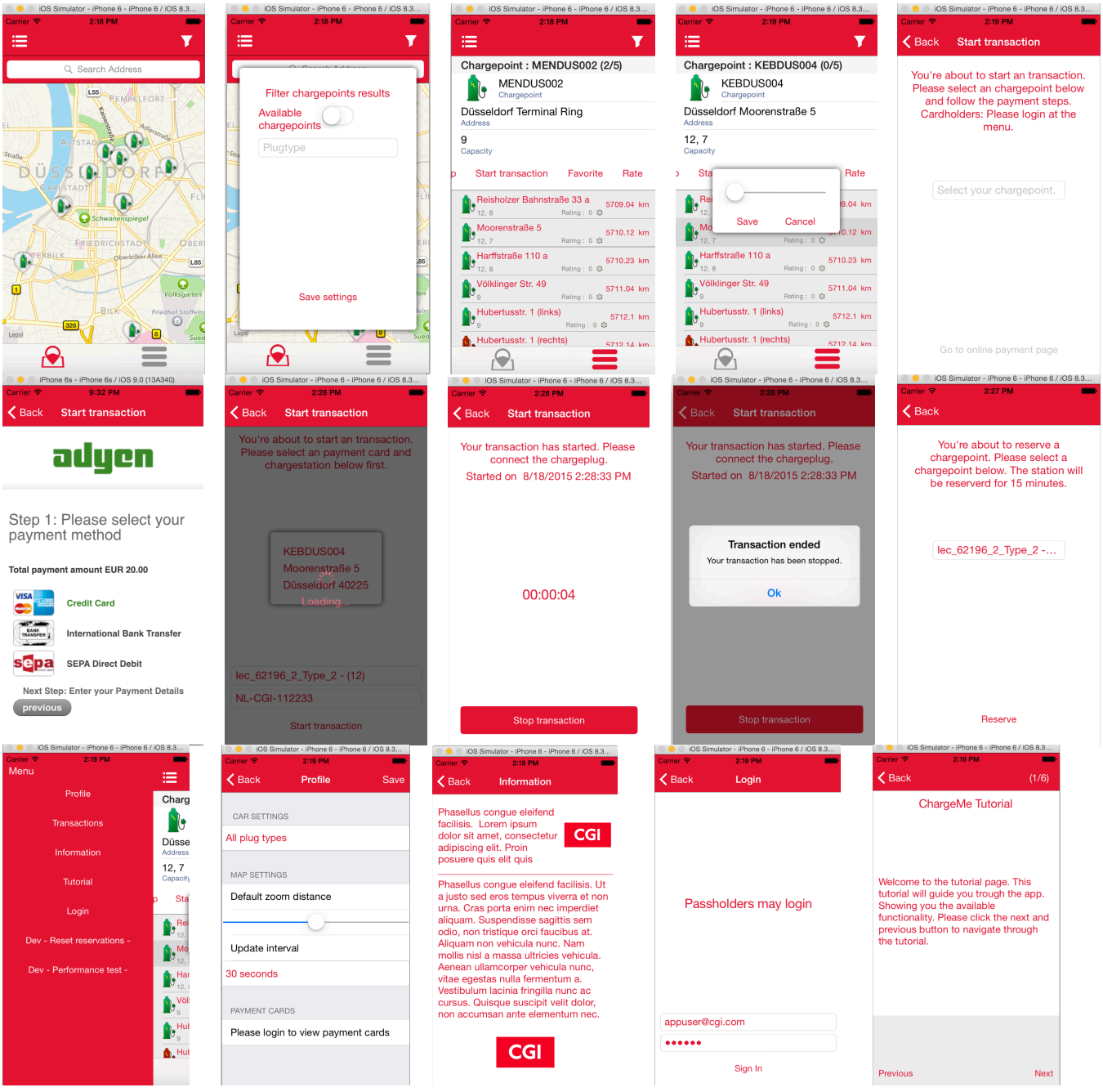
configuratiecodes waar gebruikt wordt en visa versa. Een klein deel van de schermen is te zien in Figuur 6.35.



Figuur 6.35 : Legacy

7 Resultaat

Om de uitvoering enigszins te ontdoen van plaatjes, heb ik besloten om de resultaten op te nemen in en eigen hoofdstuk. In dit hoofdstuk presenteer ik de eindresultaten van de app. Onder het moto ‘een foto zegt meer dan duizend woorden’ presenteer ik hieronder de resultaten. Het volledige klassendiagram heb ik opgenomen als bijlage (zie Bijlage XV).



8 Evaluatie

8.1 Productevaluatie

Met het eindresultaat ben ik zeer tevreden. De applicatie bevat meer functionaliteiten dan vooraf afgesproken. Ook voldoet de app aan alle niet-functionele wensen en eisen. Ik ben van mening dat de applicatie demo-klaar is. Mijn bedrijfsmentor gaf aan dat het de demoversie overstijgt. Hij zei dat een paar aanvullingen nodig zijn voordat het op de markt wordt gebracht. Terwijl de afstudeeropdracht een demoversie beloofde. De ontvangen feedback op reviews ondersteunen mijn mening: voornamelijk positieve feedback en weinig tot geen correcties.
Daarnaast heb ik de applicatie tweemaal gedemonstreerd in een presentatie. De eerste presentatie hield ik halverwege mijn opdracht. Dat was tijdens de teammeeting voor een groep van 40 á 50 man. Zij gaven allen positieve reacties en waren zeer geïnteresseerd. Tot slot hield ik een eindpresentatie voor een kleinere groep. Aan hun demonstreerde ik ook de applicatie nadat ik het doorlopen proces heb toegelicht. Zij waren positief verast met het eind resultaat. De opdrachtgever bood mij direct een baan aan. Bovendien waren ze zeer positief over mijn prestaties. Dat is terug te vinden in de stage-evaluatie: "*Op sommige punten zit hij nog boven sommige app developers die in het bedrijfsleven werkzaam zijn.*"

Het plan van aanpak heeft de opdrachtgever en begeleider inzicht gegeven in mijn aanpak en planning. De planning geeft een duidelijk overzicht wanneer tussenresultaten verwacht konden worden. Hierdoor kon hij vooraf tijd reserveren om resultaten te reviewen en feedback te leveren. Deze momenten kon ik gebruiken om tussenresultaten af te stemmen. Het reviewen en afstemmen verliep effectief en volgens plan.

Het initiatiedocument dat ik heb geschreven, is niet gebruikelijk met Scrum. Het document heb ik geschreven om meer inzicht te krijgen. Methodes die - met (R)UP - effectief zijn om de wensen en eisen in kaart te brengen, hebben mij geholpen bij het aanvullen van het Product Backlog. Daarnaast kon de opdrachtgever met het document verifiëren dat het verwachte resultaat bij mij en hem overeenkomt.

Tijdens het definiëren van de opdracht was ik nog niet op de hoogte gesteld van de online-omgeving voor het Product Backlog. Het backlog is daarom niet opgeleverd als document. De omgeving heeft ervoor gezorgd dat ik effectief de scrumprocessen kon inrichten. Het geeft, naar mijn mening, een beter overzicht dan een fysieke scrumboard of document. Daarnaast kon mijn begeleider en opdrachtgever op elk gewenst moment het Backlog inzien. Hierdoor waren zij goed op de hoogte van de voortgang. Het backlog heeft mij geholpen met het overzicht te houden en sprints te plannen met hoog geprioriteerde functionaliteiten.

De Sprint Backlog kon ik in die zelfde omgeving inrichten. Tijdens de grooming sessies kon ik de sprints doeltreffend inplannen. Ik kon ze gemakkelijk voorzien van taken. De taken konden aangevuld worden met een verwachte tijdsindicatie. Hierdoor kon ik gericht de hoeveelheid story's inplannen. Dit heeft goed gewerkt omdat het zelden is voorgekomen dat de planning uitliep. Naast het online-inplannen, heb ik ook een sprintdocument gemaakt bij aanvang van elke sprint. In dat document beschreef ik de user story's en de aanpak. Met onder ander UML-diagrammen beschreef ik de aanpak. Dit document heeft mij geholpen met stilstaan bij problemen en hun mogelijke oplossingen. Daarbij gaf het de opdrachtgever inzicht in het doorlopen proces.

Aanvankelijk zou ik de API voor CiMS met een member van CGI definiëren. Achteraf bleek dat een member al een interface heeft gebouwd. Daarvoor was geen documentatie beschikbaar. Het schrijven van de documentatie had geen hoge prioriteit. Dit heb ik uitgesteld tot de start van de functionaliteit voor het starten en stoppen van transacties. API-documentatie heeft mij meer inzicht gegeven in de benodigde calls. Daarmee kon ik starten aan de implementatie. Ook heb ik de documentatie aangevuld met één call voor het reserveren. Deze kan gebruikt worden wanneer CGI de functionaliteit gaat inbouwen in CiMS. De integratie van de reserveerfunctionaliteit zal hierdoor direct werken.

Tot slot heb ik een document afgeleverd met afbeeldingen. Van elk scherm in de app heb ik een schermkopie gemaakt. Deze heb ik bewerkt en aangevuld met de codes die gebruikt zijn in de taal- en configuratie bestanden. Hierdoor kan CGI en toekomstige klanten gemakkelijk de app branden.

8.2 Procesevaluatie

Over het hele proces ben ik geen problemen tegengekomen waar ik niet uit kwam. Voor het grootste gedeelte ben ik tevreden over het hele proces.

Na de tweede sprint kwam ik erachter dat user story's aangevuld kunnen worden met een tijdsindicatie. Hierdoor kon ik sprints beter inplannen. In vergelijkbare volgende projecten begin ik daar in de eerste sprint mee. Achteraf heeft dat geen gevolgen gehad voor de planning. Alles is eigenlijk volgens planning verlopen. Alleen in sprint vijf heb ik de user story voor het melden van onjuiste informatie niet gehaald. Dit kwam door de presentatie die ik had gegeven.

De licentie voor Xamarin kreeg ik vrij laat. Daardoor kon ik niet doeltreffend testen op mobiele apparaten. Zonder licentie is het alleen mogelijk om in de simulator te testen. Ik had de licentie veel eerder willen ontvangen zodat ik direct kon beginnen met testen op mobiele telefoons. Ik vind dat ik voldoende heb gedaan om de licentie te krijgen. Iedere week vroeg ik naar de status. Ook vertelde ik de noodzaak ervan. Dat ik zonder licentie niet een werkbare applicatie kan garanderen. Ik heb het gemeld bij mijn begeleiders en opdrachtgever. Zij konden het proces niet versnellen omdat goedkeuring nodig was van het management.

Het Xamarin raamwerk blijkt een fijn platform om applicaties mee te ontwikkelen. Ik zou Xamarin aanraden in vervolg projecten. Met Xamarin Forms kunnen apps sneller ontwikkeld worden en kennis van iOS en Android is minder noodzakelijk. Forms bleek wel gelimiteerd te zijn. Ik moest veel functionaliteiten op beide platformen ontwikkelen omdat Forms nog een gelimiteerde interface heeft.

Achteraf ben ik tevreden met de gekozen aanpak. Scrum paste goed bij dit project. Ik had geen ervaring met deze software-ontwikkelmethode, maar ik was wel bekend met de theorie. Scrum bleek makkelijk toepasbaar en leverde geen problemen op. Door het iteratief karakter kon ik de opdrachtgever goed op de hoogte houden. Daardoor werd een app ontwikkeld die aan zijn verwachtingen voldoet.
Het enige wat ik miste met Scrum was een testmethodiek. Ik testte de app wel uitvoerig waardoor er weinig tot geen bugs in het resultaat zit. In vervolg projecten zou ik unittesten toevoegen.

8.3 Evaluatie beroepstaken

1.1 Selecteren methoden, technieken en tools
Deze competentie heb ik gedeeltelijk uitgevoerd op niveau twee. Bij aanvang van het project zou PhoneGap gebruikt worden als ontwikkeltool. Andere stakeholders raadde Xamarin aan. Ik had besloten om pakketselectie uit te voeren tussen beide producten. Daardoor kon ik een advies geven over de ontwikkeltool die gebruikt gaat worden. In paragraaf 6.1 heb ik de pakketselectie beschreven.

1.4 Uitvoeren analyse door definitie van requirements
Op niveau vier kan ik deze competentie demonstreren. Het project bestaat uit een groot aantal requirements met tegengestelde wensen en eisen. De requirements zijn traceerbaar, geprioriteerd en voorzien van een versie. In bijlage III staan de requirements.

3.1 Ontwerpen softwarearchitectuur
Deze competentie heb ik gedeeltelijk uitgevoerd op niveau vier. De softwarearchitectuur is ontworpen voor een Android en iOS applicatie. Zij gebruiken een gedeelde code-base. In paragraaf 6.2 heb ik de softwarearchitectuur ontworpen.

3.2 Ontwerpen systeemdeel

Ook deze competentie heb ik uitgevoerd op niveau 4. In het ontwerpproces heb ik rekening gehouden met niet-functionele kwaliteitsattributen. De app heeft een ergonomische interface en oogt professioneel. Op het 3G en 4G netwerk voldoet de app aan de performance eisen. Verder is de app en API beveiligd. Ik heb gebruik gemaakt van meerdere design patronen: MVC, façade, factory, singleton en repository.

In paragraaf 6.2 heb ik de softwarearchitectuur bedacht. Daarin maak ik gebruik van het MVC, façade en repository patroon. Het factory patroon pas ik toe in paragraaf 6.6. De user interface komt aan in paragraaf 6.3 aan de orde. De performance en beveiliging worden beschreven in paragraaf 6.9 en 6.10.

3.3 Bouwen applicatie

Het bouwen van de applicatie heb ik op niveau 3 uitgevoerd. Het betreft een object georiënteerd systeem. Vanwege de scrumachtige aanpak heb ik rekening gehouden met veel wijzigende code. Ook heb ik veel rekening gehouden met de overdraagbaarheid. De app is gemakkelijk te wijzigen en branden.

Literatuurlijst

About the Project. (z.j.). Geraadpleegd van http://phonegap.com/about/
Create native iOS, Android, Mac and Windows apps in C#.. (z.j.). Geraadpleegd van https://xamarin.com/platform

Android. (z.j.). User Interface. Geraadpleegd op 27 april, 2015, van https://developer.android.com/guide/topics/ui/index.html

Apple. (z.j.). Designing for iOS. Geraadpleegd op 27 april, 2015, van https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/

Arlow, J., & Neustadt, I. (2011). UML 2 and the unified process (8e ed.). New Jersey, US: Pearson Education, Inc.

Guzel, B. (2009, 2 december). HTTP Headers for Dummies. Geraadpleegd van http://code.tutsplus.com/tutorials/http-headers-for-dummies--net-8039

Leeuwen, S. J. O. R. S. van. (2007). LEVERANCIER- EN SOFTWARESELECTIE. Geraadpleegd van http://www.indora.nl/wp-content/uploads/2013/06/BSR-Selecteren-van-een-CRM-Systeem.pdf

Moschetti, R. (2014, 3 augustus). Opening a mobile app from a link: the Xamarin way (URL-Schemes). Geraadpleegd van http://riccardo-moschetti.org/2014/10/03/opening-a-mobile-app-from-a-link-the-xamarin-way-url-schemas/

Poort, E. (2013, juni). Architectuur kan ook agile zijn. Geraadpleegd van http://www.cginederland.nl/artikelen/architectuur-kan-ook-agile-zijn

Rozanski, N., & Eoin, E. (2008). Software Systems Architecture. New Jersey, US: Pearson Education, Inc.

Schwaber, K., & Sutherland, J. (2010). Scrum: Developed and sustained. Geraadpleegd van www.scrum.org

Troelsen, A. (2012). Pro C# and the .NET 4.5 Framework (6e ed.). New York, USA: Springer Science.

Xamarin. (z.j.). Publish and Subscribe with MessagingCenter. Geraadpleegd op 25 mei, 2015, van https://developer.xamarin.com/guides/cross-platform/xamarin-forms/messaging-center/

Verklarende woordenlijst

Woord	Uitleg
.NET	Applicatieframework van Microsoft.
CiMS	Platform voor het op afstand beheren van laadpalen. Ontwikkeld door CGI.
iOS	Besturingssysteem van Apple dat draait op al hun apparaten.
PhoneGap	Platform voor het ontwikkelen van mobiele cross-platform applicaties met HTML, JavaScript en CSS.
Visual Paradigm	Ontwerptool waarmee UML-diagrammen ontworpen kunnen owrden.
Xamarin	Platform voor het ontwikkelen van mobiele cross-platform applicaties met C#.

Bijlage

I. Afstudeerplan
II. Plan van aanpak
III. Product initiatie document
IV. Sprint één
V. Sprint twee
VI. Sprint drie
VII. Sprint vier
VIII. Sprint vijf
IX. Sprint zes
X. Sprint zeven
XI. CiMS API documentatie
XII. CRM API documentatie
XIII. Sprint acht
XIV. Sprint negen
XV. Klassendiagram
XVI. Evaluatie CGI
XVII. Tussentijds assessment