

# HoGent

Academiejaar 2012–2013

Geassocieerde faculteit Toegepaste Ingenieurswetenschappen  
Valentin Vaerwyckweg 1 – 9000 Gent

## Ontwikkeling van een cross-platform mobiele applicatie

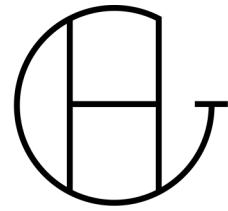
**Masterproef voorgedragen tot het behalen van het diploma van  
Master in de industriële wetenschappen: informatica**

**Benjamin DE CLERCQ**

Promotoren: Geert VAN HOOGENBEMT  
Liesbet LANSSENS (Cerm, 8020 Oostkamp)

Begeleiders: Ann ALGOET (Cerm, 8020 Oostkamp)  
Peter HEYSE (Cerm, 8020 Oostkamp)  
Peter VANVLIERBERGHE (Cerm, 8020 Oostkamp)





# HoGent

Academiejaar 2012–2013

Geassocieerde faculteit Toegepaste Ingenieurswetenschappen  
Valentin Vaerwyckweg 1 – 9000 Gent

## Ontwikkeling van een cross-platform mobiele applicatie

**Masterproef voorgedragen tot het behalen van het diploma van  
Master in de industriële wetenschappen: informatica**

**Benjamin DE CLERCQ**

Promotoren: Geert VAN HOOGENBEMT  
Liesbet LANSSENS (Cerm, 8020 Oostkamp)

Begeleiders: Ann ALGOET (Cerm, 8020 Oostkamp)  
Peter HEYSE (Cerm, 8020 Oostkamp)  
Peter VANVLIERBERGHE (Cerm, 8020 Oostkamp)

# Woord vooraf

*Vooreerst gaat mijn oprechte dank uit naar mijn promotor Geert Van hoogenbemt. Ondanks een drukke agenda door de integratie van de opleiding aan UGent kon ik regelmatig rekenen op zijn raad en advies. Dankzij zijn begeleiding werd tijdig aan alle taken begonnen zodat deadlines geen probleem vormden.*

*Mijn dank gaat ook uit naar alle medewerkers van Cerm, die mij alle gewenste middelen ter beschikking hebben gesteld. Door de aangename werksfeer kwam ik ook steeds met plezier aan mijn masterproef werken op het bedrijf. De interesse van de medewerkers die onder andere bleek uit een demo die ik in een vroeg stadium gegeven heb was zeker een extra motivatie. Voor uitleg, of bij problemen, was er ook altijd iemand waar ik bij terecht kon. Katrien Penneman wil ik hier graag speciaal bij vermelden.*

*Directeur Tom Musschoot wil ik graag uitdrukkelijk bedanken voor de kans om mijn masterproef uit te voeren bij Cerm.*

*Ook aan mijn begeleiders wil ik een speciaal woord van dank richten. Ann Algoet en Peter Heyse volgden het project mee op en wil ik graag bedanken voor hun feedback en enthousiasme over mijn werk. Peter Vanvlierberghe heeft mij goed op weg geholpen in het begin en wil ik hiervoor ook bedanken.*

*Bijzonder veel dank aan mijn promotor Liesbet Lanssens voor de begeleiding. Van haar kreeg ik veel feedback en steun. Ik kon mezelf geen betere promotor wensen. Dankzij haar kan ik heel tevreden zijn over mijn masterproefkeuze.*

*Daarnaast wil ik ook Kathleen Pollefliet en collega's Linsky Aerts en Michiel Gyssels bedanken voor de tips die geholpen hebben bij het schrijven van deze scriptie.*

*Tenslotte wil ik mijn familie en vrienden bedanken voor de morele steun en de nodige ontspanning tijdens het uitwerken van deze masterproef.*

*Benjamin De Clercq  
Gent, juni 2013*

# **Abstract**

Smartphones en tablets zijn populairder dan ooit. Deze mobiele toestellen worden steeds krachtiger en vinden telkens nieuwe toepassingen. Applicaties spelen hierbij een grote rol en zijn ook bepalend voor het succes van het besturingssysteem. Meestal worden deze applicaties ontwikkeld met technologieën en tools die bepaald worden door de keuze van het platform waar men zich op richt. Het grote nadeel hierbij is dat applicaties die ontwikkeld zijn voor een specifiek platform niet kunnen draaien op een ander besturingssysteem. Zeker voor bedrijven is het interessant als het mogelijk zou zijn om met één bepaalde set technologieën en tools de applicatie op zo veel mogelijk platformen ineens beschikbaar te kunnen stellen.

In deze thesis wordt onderzocht wat de mogelijkheden zijn voor cross-platform ontwikkeling voor mobiele applicaties. Welke voor- en nadelen duiken hier bij op? Verschillende mogelijke technologieën, beschikbare ontwikkelingstools en frameworks komen hier dan ook aan bod. In het bijzonder wordt aandacht besteed aan de nieuwe mogelijkheden die de HTML5-technologieën bieden. Uiteindelijk wordt het resultaat van het onderzoek toegepast om een cross-platform mobiele applicatie te ontwikkelen waarmee gegevens uit de Cerm-omgeving geraadpleegd kunnen worden.

**KERNWOORDEN:**

**mobiel, cross-platform, web, hybrid, HTML5, Sencha Touch, iOS, Android, Windows Phone**

# **Abstract**

Smartphones and tablets are more popular than ever. The power of these mobile devices keeps increasing and brings new possibilities to the field. Applications are an important factor in this and can determine the success of a mobile operating system. In most cases the used technologies and development tools are determined by the choice of the targeted operating system. Because of this it isn't possible for an application targeted on one platform to run on different operating systems. Especially for companies, it would be very interesting if there's a possibility to use one set of technologies and tools for application development to target a wider range of mobile operating systems.

This thesis is the result of research focused on the possibilities for cross-platform mobile development. What are the advantages and disadvantages? The research contains different technologies, development tools and frameworks. The main focus goes to the new possibilities that come with the introduction of HTML5. Finally the result of this research is put into practice with the development of a mobile application which can be used to access data from the Cerm environment.

**KEYWORDS:**

**mobile, cross-platform, web, hybrid, HTML5, Sencha Touch, iOS, Android, Windows Phone**

# Inhoudsopgave

<b>Woord vooraf</b>	<b>2</b>
<b>Abstract</b>	<b>3</b>
<b>Inleiding</b>	<b>8</b>
<b>1 Situering en probleemstelling</b>	<b>11</b>
<b>2 Mobiele besturingssystemen en hun native applicaties</b>	<b>14</b>
2.1 iOS . . . . .	15
2.2 Android . . . . .	17
2.3 Windows Phone . . . . .	19
2.4 BlackBerry 10 . . . . .	20
2.5 Sailfish . . . . .	20
2.6 Ubuntu . . . . .	20
2.7 Tizen . . . . .	21
2.8 Firefox OS . . . . .	22
2.9 Conclusie . . . . .	22
<b>3 Webapplicaties</b>	<b>25</b>
3.1 HTML5 . . . . .	26
3.2 Ondersteuning van webapplicaties in iOS . . . . .	28
3.3 Frameworks . . . . .	29
3.3.1 jQuery Mobile . . . . .	29
3.3.2 Sencha Touch . . . . .	31
3.4 Ontwikkelingstools . . . . .	33
3.4.1 Embarcadero HTML5 Builder . . . . .	33
3.4.2 Visual Studio 2012 . . . . .	34
3.4.3 Netbeans 7.3 . . . . .	34
3.4.4 Sencha Architect, Sencha Cmd en Sencha Eclipse Plugin . . . . .	34
<b>4 Hybride applicaties</b>	<b>37</b>
4.1 PhoneGap . . . . .	38
4.2 Xamarin . . . . .	40

4.3	Marmalade . . . . .	41
4.4	Rhomobile . . . . .	42
4.5	Conclusie . . . . .	42
<b>5</b>	<b>Conclusie onderzoek</b>	<b>44</b>
5.1	Algemeen . . . . .	44
5.2	Keuze van de gebruikte technologien . . . . .	46
<b>6</b>	<b>Aanvangen met Sencha Touch</b>	<b>47</b>
6.1	De installatie . . . . .	47
6.2	De MVC-architectuur . . . . .	48
6.3	Het klassensysteem . . . . .	49
6.4	De componenten en het lay-out-systeem . . . . .	51
<b>7</b>	<b>De front-end ontwikkeling van de applicatie</b>	<b>52</b>
7.1	De machinesmodule . . . . .	53
7.1.1	De lijst van machines . . . . .	53
7.1.2	De tellergegevens van een machine weergeven . . . . .	55
7.1.3	De interface optimaliseren voor smartphones én tablets . . . . .	56
7.1.4	De functionaliteiten gebundeld in de toolbar . . . . .	59
7.2	Internationalisering . . . . .	61
7.3	De applicatie voorzien van een login . . . . .	64
7.4	De contactenmodule . . . . .	66
7.4.1	De contactenlijst weergeven . . . . .	66
7.4.2	Contactgegevens weergeven . . . . .	68
7.4.3	Klantgegevens weergeven . . . . .	68
7.4.4	Interactiviteit toevoegen . . . . .	70
7.4.4.1	Functionaliteit voor bellen en mailen . . . . .	70
7.4.4.2	De mapsfunctionaliteit . . . . .	72
7.4.5	Gepaste iconen voorzien . . . . .	74
7.5	Navigatieprobleem oplossen . . . . .	76
7.6	De gebruiker melden hoe de app geïnstalleerd kan worden op een iOS-toestel	76
7.7	Distributie . . . . .	78
<b>8</b>	<b>De backend-ondersteuning voor de applicatie</b>	<b>80</b>
8.1	ServiceCounters . . . . .	80
8.1.1	Keuze voor het JSON-formaat . . . . .	81
8.2	Configuratie van de server . . . . .	83
8.2.1	Cross Origin Resource Sharing . . . . .	84
8.3	ServiceLogin . . . . .	85
8.3.1	Bijkomende serverconfiguratie . . . . .	86
8.4	ServiceContacts . . . . .	87
8.5	Zorgen voor een beveiligde verbinding . . . . .	87

<b>9 Conclusie</b>	<b>89</b>
9.1 Mogelijke uitbreidingen . . . . .	92
<b>Lijst van afkortingen</b>	<b>93</b>
<b>Lijst van afbeeldingen</b>	<b>96</b>
<b>Lijst van tabellen</b>	<b>97</b>
<b>Lijst van codevoorbeelden</b>	<b>98</b>
<b>Literatuurlijst</b>	<b>99</b>

# Inleiding

Tegenwoordig zijn smartphones en tablets al stevig ingeburgerd. Naast reeds gevestigde mobiele besturingssystemen zoals iOS en Android zijn er nu ook Windows Phone en BlackBerry 10. Momenteel staan er nog enkele nieuwe besturingssystemen hun opwachting te maken om de markt te betreden.

In 2009 gebruikte Apple de reclameslogan "*there's an app for that!*" voor de iPhone. Mobiele applicaties vinden verscheidene toepassingen. Door de diversiteit van de verschillende besturingssystemen is een applicatie niet altijd beschikbaar voor het platform dat de gebruiker verkiest. De zogenaamde native applicaties worden namelijk ontwikkeld met technologieën die afhankelijk zijn van het platform. Hierdoor moet vaak een keuze gemaakt worden voor welk besturingssysteem ontwikkeld wordt. Beschikbaarheid op verschillende platformen vereist extra inspanningen. De applicatie moet voor elk specifiek platform opnieuw ontwikkeld worden.

Applicaties ontwikkeld met technologieën die gebruikt kunnen worden voor verschillende besturingssystemen worden cross-platform genoemd. Met een applicatie die cross-platform is kan een groter publiek bereikt worden. In deze masterproef wordt onderzoek gedaan naar de mogelijkheden om een mobiele applicatie te ontwikkelen die platformonafhankelijk is. Het onderzoek leidt hierbij tot de ontwikkeling van een cross-platform mobiele applicatie die gegevens uit het '*Management Information*'-systeem van Cerm kan weergeven.

Hoofdstuk 1 vertelt hiervoor eerst meer over Cerm en het MIS (Management Information System). Daarnaast komt het doel van deze masterproef en de probleemstelling aan bod.

In hoofdstuk 2 worden verschillende mobiele besturingssystemen besproken. Zowel reeds bestaande als toekomstige platformen komen hierbij aan bod. Voor elk platform worden de gebruikte technologieën en eigenschappen beschreven. Dit is belangrijk om weten zodat gebruik gemaakt kan worden van technologieën en eigenschappen die meerdere platformen gemeenschappelijk hebben. Aan het eind van dit hoofdstuk wordt dan ook de conclusie gemaakt.

Iets wat mobiele besturingssystemen zeker gemeen hebben is dat er minstens één internetbrowser beschikbaar is. Webtoepassingen kunnen dus een oplossing bieden. In hoofdstuk 3 wordt HTML5 geïntroduceerd en worden verschillende frameworks en ontwikkelingstools besproken. Omdat iOS van in het begin een goede ondersteuning bood voor webapps komt dit ook in dit hoofdstuk aan bod.

De documentatie van Apple voor het ontwikkelen van dit soort applicaties bleek dan ook een zeer goede informatiebron te zijn.

Een andere oplossing is de applicatie inwikkelen (*wrappen*) in een native applicatie en van daaruit uit te voeren. Hiervoor bestaan verschillende technieken. Zo kan een webapp weergegeven worden met een native *webview*-component en wordt er een brug voorzien naar de API's van het besturingssysteem. Het is ook mogelijk om een interpreter of een runtime te voorzien voor de applicatie. Sommige besturingssystemen voorzien zelf al een extra runtime. De runtime voor de applicatie kan bijvoorbeeld ook met de toepassing gebundeld zijn of vooraf geïnstalleerd worden. Dit soort applicaties worden hybride applicaties genoemd. Enkele frameworks die dit soort hybride oplossingen aanbieden komen in hoofdstuk 4 aan bod.

Uit het onderzoek blijkt dat de ontwikkeling van een webapplicatie een keuze is die met veel argumenten te verdedigen is. Een van de belangrijkste argumenten is het feit dat op die manier eenvoudig twee platformonafhankelijke oplossingen geboden kunnen worden. De webapplicatie kan eenvoudig omgevormd worden tot een hybride applicatie. Beide soorten kunnen dan ook gedistribueerd worden. Wanneer de webapplicatie niet voldoende functionaliteit kan bieden is er nog altijd een uitweg mogelijk door de webtoepassing uit te breiden naar een hybride app. Hierdoor hoeft er geen werk verloren te gaan bij het bereiken van de limieten van een HTML5-app. De conclusie van het volledige onderzoek wordt beschreven in hoofdstuk 5. Hierin wordt ook verteld waarom gekozen werd om te ontwikkelen met het 'Sencha Touch'-framework.

Sencha Touch is een uitgebreid framework. Daarom moet de ontwikkelaar ook enige vertrouwdheid krijgen met de concepten en componenten die eigen zijn aan het framework. In hoofdstuk 6 worden enkele belangrijke concepten zoals de architectuur, het klassensysteem, de componenten en het lay-out-systeem uitgelegd aan de hand van voorbeelden die komen uit de ontwikkelde applicatie.

In hoofdstuk 7 ligt de focus op de front-end ontwikkeling van de applicatie. Hierbij komen alle features van de applicatie aan bod. Doorheen dit hoofdstuk wordt regelmatig verwezen naar externe documentatie. De belangrijkste ontwerpbeslissingen, oplossingen en problemen die opdoken worden meer gedetailleerd besproken.

De webservices waar de applicatie gebruik van maakt worden toegelicht in hoofdstuk 8. Dit hoofdstuk beschrijft ook verschillende serverconfiguraties die gedaan moesten worden. De keuze voor het JSON-formaat wordt besproken, alsook de manier waarop voor een beveiligde communicatie kan gezorgd worden.

In het laatste hoofdstuk wordt de conclusie gevormd over de volledige masterproef. Het resultaat van het onderzoek wordt hier nog eens samengevat. Daarnaast wordt ook nog eens opgesomd wat er allemaal gerealiseerd is en wat er qua performantie en ondersteuning verwacht kan worden. Als afsluiter komen ook nog enkele mogelijke uitbreiden en verbeteringen aan bod.

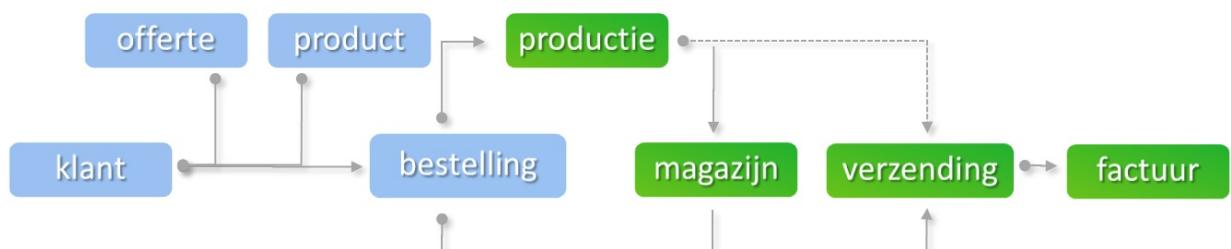
Verwijzingen naar de literatuurlijst worden gedaan door de auteur(s) en het jaartal van publicatie tussen haken te vermelden. In de literatuurlijst kunnen deze referenties op alfabetische volgorde terug gevonden worden.

# Hoofdstuk 1

## Situering en probleemstelling

Cerm is een gespecialiseerd softwarebedrijf. Het bedrijf, dat gevestigd is in Oostkamp, levert een totaaloplossing voor de administratieve automatisering van commerciële en industriële drukkerijen. Dit omvat de voortdurende ontwikkeling (softwareontwikkeling) van een geïntegreerd softwarepakket, de installatie en het onderhoud van de hardware (IT & techniek) en de opleiding en de begeleiding van de klanten (consultancy). De software van Cerm is uitgegroeid tot een gespecialiseerd en gerespecteerd MIS-pakket (Management Information System) voor de grafische industrie. De klanten van Cerm bevinden zich in heel Europa, in de Verenigde Staten en ondertussen ook in Australië.

Het Cerm-systeem wordt volledig geïntegreerd in de workflow van een grafisch bedrijf en wordt gebruikt op elke afdeling. Figuur 1.1 geeft een vereenvoudigde voorstelling van deze workflow. Het gebruik van deze software zorgt voor de automatisering van de administratie. Hierdoor komt er heel wat nuttige informatie ter beschikking. Zo kunnen bijvoorbeeld lopende offertes geraadpleegd worden, of kan de huidige stock opgevraagd worden. Ook verschillende aspecten van de productie kunnen opgevolgd worden. De afvalproductie kan bijvoorbeeld ook in kaart gebracht worden. Dit maakt het voor het management mogelijk om hun volledige workflow te optimaliseren. Het gebruik van de software maakt het ook eenvoudiger om aan de ISO-standaarden te voldoen.



**Figuur 1.1:** Vereenvoudigde voorstelling van de workflow van een grafisch bedrijf.

De sterke troeven van Cerm is de Duitse onderneming Heidelberg niet ontgaan. Heidelberg is een wereldspeler in de productie van drukpersen en opvolging van de productieprocessen in de grafische industrie in het algemeen. Heidelberg huisvest meer dan dertienduizend werknemers wereldwijd. Door de toenemende vraag naar integratie van managementsystemen en productiesystemen ging Heidelberg op zoek naar een goed MIS om deze integratie te realiseren. Hierbij bleek Cerm hun voorkeurspartner en uiteindelijk zijn beide partijen akkoord gegaan. Voor deze verregaande samenwerking is Cerm in 2011 in de Heidelberggroep gestapt. Het doel van de samenwerking is om een volledig geïntegreerd management- en productiesysteem te realiseren om zowel commerciële als productieprocessen op te volgen.

Op dit moment zijn de gegevens uit de software voor de medewerkers van het grafische bedrijf enkel toegankelijk via de Cerm-software op een Windows-systeem binnen het bedrijf. Dat wil zeggen dat een RDP-connectie (Remote Desktop Protocol) nodig is om een remote Windows desktop te laden en zo gebruik te kunnen maken van de Cerm-software om de gegevens te benaderen. Momenteel heeft de gebruiker dus enkel de mogelijkheid om op een smartphone een remote Windows desktop te laden om de Cerm-gegevens te bekijken. Aangezien dit geen gebruiksvriendelijke manier is wordt dit zeer weinig of niet toegepast.

De toenemende populariteit van smartphones en tablets zet zich ook door tot de bedrijfswereld. Om de stijgende vraag naar software voor deze mobiele toestellen te beantwoorden dringt de ontwikkeling van een mobiele applicatie zich op. Een mobiele applicatie zou een veel betere oplossing zijn dan een remote desktop te moeten laden. Hiermee zouden enkele essentiële gegevens op een eenvoudigere manier ter beschikking kunnen gesteld worden aan de medewerkers van het grafisch bedrijf.

Deze mobiele applicatie moet modulair opgebouwd worden zodat nieuwe modules eenvoudig toegevoegd kunnen worden. In een eerste module ligt de focus op de productieomgeving. Bij bedrijven die gebruik maken van de software van Cerm worden er tellers op de drukpersen geplaatst die de activiteit registreren in een database. Het moet mogelijk zijn om via de mobiele app deze gegevens te raadplegen via dynamische grafieken. Een tweede deeltoepassing moet het mogelijk maken om gedetailleerde gegevens van klanten en contactpersonen weer te geven. Deze gegevens moeten ook interactief gebruikt kunnen worden door bijvoorbeeld het e-mailadres of telefoonnummer te selecteren. Om de applicatie te kunnen gebruiken dient de gebruiker zich aan te melden via een login. De applicatie moet ook meerdere talen ondersteunen. De communicatie met de webservices moet op een beveiligde manier gebeuren.

Omwille van de diversiteit in het aanbod van tablets en smartphones moet zeker de nodige aandacht besteed worden aan de schaalbaarheid van de grafische gebruikersinterface alsook aan de mogelijkheden om de applicatie beschikbaar te maken voor de verschillende besturingssystemen (iOS, Android, Windows Phone,...). Ook de verschillen in schermformaten en beeldresoluties zorgen er voor dat het ontwikkelen van een gebruiksvriendelijke grafische interface voor aanraakschermen de nodige aandacht vereist.

Natuurlijk moet er ook rekening gehouden worden met de beperkingen van mobiele toestellen ten opzichte van desktopsystemen. Deze beschikken over minder geheugen en processorkracht. Om gebruik te kunnen maken van de webservices moet de mobiele applicatie over een netwerk- of internetverbinding kunnen beschikken. Dit kan door middel van een WiFi-netwerk, maar het is ook mogelijk om van een mobiel datanetwerk gebruik te maken. Daarom moet men het dataverkeer zoveel mogelijk beperken en eventueel ook bepaalde gegevens lokaal kunnen bewaren. Dit zal ook een positief effect hebben op het batterijverbruik.

## **Hoofdstuk 2**

# **Mobiele besturingssystemen en hun native applicaties**

Smartphones, of vroeger ook PDA's (Personal Digital Assistant) genoemd, werden vroeger vooral gebruikt voor zakelijke toepassingen zoals e-mail, agenda en afspraken. Deze toestellen werkten meestal met een besturingssysteem zoals Windows Mobile, Blackberry OS of Symbian. Er is de voorbije vijf jaar echter heel wat veranderd en smartphones zijn heel populair en wijdverspreid geworden. De smartphone krijgt ook steeds meer nieuwe toepassingen.

De mobiele sector is nog relatief nieuw en evolueert heel snel. De concurrentie tussen fabrikanten, ontwikkelaars en providers onderling is bikkelhard. De ene na de andere rechtszaak wordt uitgevochten en er ontstaan heuse patentenoorlogen die soms leiden tot een verbod tot verkoop van een product.

In dit hoofdstuk wordt besproken hoe de mobiele sector de laatste jaren geëvolueerd is. Om een zicht te krijgen op de mogelijke technologieën en ondersteunde applicaties komen de belangrijkste besturingssystemen aan bod, alsook enkele nieuwe (toekomstige) spelers op de markt die een invloed zouden kunnen hebben in de mobiele sector. Fabrikanten en providers, maar ook consumenten, hebben groot voordeel bij de komst van nieuwe besturingssystemen om de dominantie van Apple en Google af te zwakken en zo terug meer onafhankelijkheid te verkrijgen. Het is dan ook belangrijk hier kennis over te hebben om een technologie te kiezen met het oog op de toekomst. De uiteindelijke technologiekeuze wordt besproken in hoofdstuk 5.

## 2.1 iOS

Begin 2007 introduceerde Steve Jobs de iPhone. De iPhone en later ook de iPad, zorgden voor een revolutie in de wereld van de mobiele toestellen. Die wereld gaat zo snel vooruit dat het besturingssysteem van de iPhone ondertussen reeds een van de oudste mobiele besturingssystemen is die nog actief ontwikkeld worden. Dit wil niet zeggen dat dit OS (Operating System) verouderd is. In tegendeel, Apple maakte van het besturingssysteem een van de populairste en best ondersteunde mobiele platformen. Opmerkelijk is de gelijkenis tussen het OS in 2007 en het OS in zijn huidige vorm. Toch is het aantal features die sindsdien zijn toegevoegd indrukwekkend. Bovendien is Apple er ook in geslaagd dit te doen zonder het OS log en traag te laten worden.

Oorspronkelijk werd het besturingssysteem simpelweg iPhone OS genoemd. Pas sinds de vierde versie die in 2010 gelanceerd werd is het besturingssysteem omgedoopt tot iOS. Er wordt verwacht dat iOS 7 in juni 2013 op de Worldwide Developers Conference van Apple aangekondigd wordt.

Opmerkelijk is het feit dat de iPhone qua specificaties en functies ver achterop liep op de concurrentie. In 2007 waren mobiele besturingssystemen zoals Windows Mobile, Palm OS, Symbian en BlackBerry de gevestigde waarden met een breed arsenaal aan features. Ter vergelijking volgt een opsomming van enkele features die iOS onder andere niet had ten opzichte van de concurrentie:

- 3G-ondersteuning
- multitasking
- copy/paste-functionaliteit
- e-mailbijlagen
- MMS-ondersteuning
- Exchange push-e-mail
- aanpasbaar startscherm
- tethering<sup>1</sup>
- aanpassen van documenten
- spraakgestuurde oproepen
- bestandssysteem dat beschikbaar is voor de gebruikers

---

<sup>1</sup>De functionaliteit om een smartphone of tablet te kunnen gebruiken als mobiele hotspot

Tot slot was het besturingssysteem zo goed als volledig gesloten voor ontwikkelaars.

Toch bleken deze gebreken het succes van iOS niet in de weg te staan. In plaats van de concurrentie aan te gaan op het gebied van de specificaties, zette Apple volop in op de gebruikerservaring. De focus werd gelegd op snelheid, consistentie tussen applicaties en het fel verbeteren van enkele functies in vergelijking met de concurrentie. Het was duidelijk dat iOS op dat gebied een heleboel innovaties bracht.

De grootste vernieuwing was de grafische gebruikersinterface. Tot voor iOS hadden smartphones geen touchscreen of een resistief touchscreen dat meestal bediend werd met een stylus. De iPhone bracht hier verandering in door gebruik te maken van een capacitief touchscreen. Nog belangrijker is dat Apple deze nieuwe hardwarefunctionaliteit koppelde aan een nieuw interactiemodel. Deze vorm van interactie was tegelijkertijd eenvoudiger en krachtiger dan de gebruikersinteractie bij andere systemen. Door het aantal hardwareknoppen te beperken tot vijf werd aanraking de primaire vorm van interactie. Apple introduceerde pinch-to-zoom en scrollen door te vegen om het gebruik van applicaties directer en natuurlijker te laten aanvoelen. (Bohn, 2011)

De belangrijkste feature in het kader van deze masterproef is de Mobile Safari webbrowser, die gebruik maakt van de WebKit-browserengine. Deze browser was, en is nog steeds, een van de krachtigste mobiele browsers. De simpele zoom- en scrollingfunctionaliteit waren op dat moment ongeëvenaard. Deze browser diende ook als platform voor third-party apps. Het was namelijk pas in juli 2008 dat Apple de App Store introduceerde en hierbij ook de iOS SDK (Software Development Kit) beschikbaar stelde voor de ontwikkeling van native apps. Daarvoor promote Apple webapplicaties. Deze toepassingen hebben geen toegang tot de native bibliotheken om de functionaliteit van het systeem en de hardware aan te spreken. Op [www.apple.com/webapps](http://www.apple.com/webapps) zijn nog steeds verschillende web apps terug te vinden. In paragraaf 3.2 wordt verder ingegaan op het belang van deze browser en deze webapplicaties.

Native<sup>2</sup> applicaties voor iOS kunnen enkel via de *App Store* op legitieme manier geïnstalleerd worden. De applicaties worden geschreven in de programmeertaal Objective-C en kunnen enkel gecompileerd worden op een Intel-based Mac. De meest gebruikte IDE(Integrated Development Environment) is XCode en is ook enkel beschikbaar voor Mac OS. Omdat een Mac met Intel-processor noodzakelijk is voor de compilatie van applicaties en deze niet ter beschikking is bij Cerm wordt hier niet verder op ingegaan.

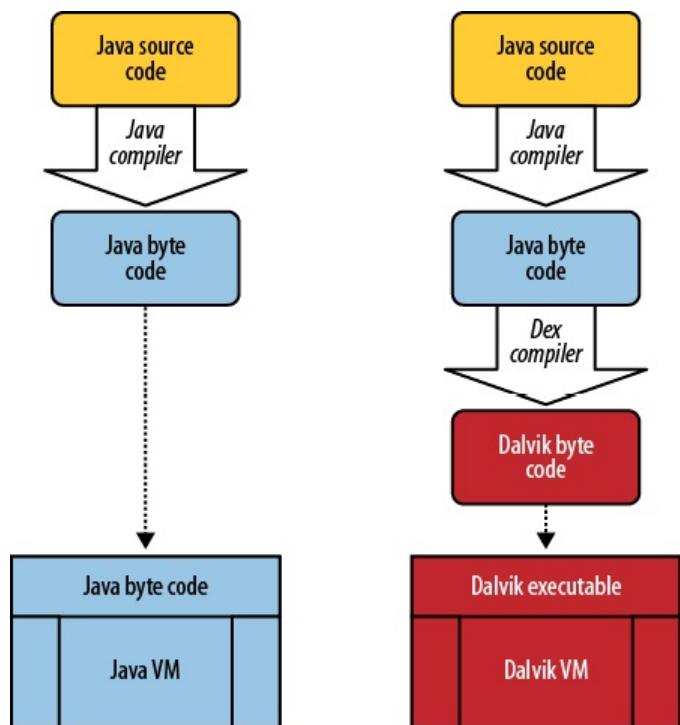
---

<sup>2</sup>Native applicaties worden specifiek voor een bepaald besturingssysteem ontwikkeld. Hiervoor wordt een SDK(Software Development Kit) voorzien.

## 2.2 Android

Google bracht in 2008 met Android zijn eigen mobiele besturingssysteem op de markt. De recentste versie is *Jelly Bean* (Android 4.2). Het Android-platform is open-source waardoor iedereen zijn eigen Android-versie kan ontwikkelen (<http://source.android.com/>). Android is gebaseerd op Linux en elke applicatie draait in zijn eigen Linux-proces met toegangswaarden die door het Linux-systeem ingesteld worden.

Androidapplicaties worden geschreven in Java. Deze bronbestanden worden net zoals bij gewone Java-applicaties gecompileerd naar Java-bytecode met dezelfde Java-compiler. Daarna wordt deze code nogmaals gecompileerd naar Dalvik-bytecode die uitgevoerd kan worden door de *Dalvik Virtual Machine*. Dalvik is de Java virtuele machine die specifiek ontworpen werd voor Android met oog op de beperkingen van mobiele toestellen zoals de levensduur van de batterij en processorkracht. De compilatiestappen worden automatisch uitgevoerd door de Eclipse IDE met de Android-plugin. De reden voor deze tussenstap is dat, in tegenstelling tot de programmeertaal Java, de Java-bytecode weinig of geen veranderingen ondergaat. Daarom werd gekozen om Dalvik te baseren op Java-bytecode in plaats van de broncode.



**Figuur 2.1:** Verschil tussen de compilatie van een Java- en een Androidapplicatie.

Hierdoor is het in theorie ook mogelijk om Android apps te schrijven in elke taal die gecompileerd kan worden naar Java-bytecode. In de praktijk is het echter zo dat de nodige bibliotheken die deel uitmaken van de Android SDK hiervoor niet beschikbaar zijn. Dit is iets waar de open source community in de toekomst misschien verandering in kan brengen. De verzameling Java-bibliotheken voor Android leunt het dichts aan bij die van de *Java Standaard Editie*. Het grootste verschil is dat de Java UI-bibliotheken (AWT en SWING) vervangen zijn door Android-specifieke UI-bibliotheken. Android voegt ook nieuwe functie toe terwijl de meeste standaard features van Java ondersteund blijven.

Een applicatie wordt geïnstalleerd met een APK(Application Package) dat bestaat uit drie grote componenten:

- het Dalvik uitvoerbaar bestand die de gecompileerde Dalvik byte code bevat die wordt uitgevoerd
- bronbestanden zoals media, XML-bestanden die layouts beschrijven, taalpakketten,...
- platformspecifieke bibliotheken zoals C/C++-bibliotheken

De APK van een applicatie kan gewoon op het toestel gezet worden of kan gedownload worden via een website of een Android market.

Via de opensource-gemeenschap zijn er vaak C/C++-bibliotheken beschikbaar om de nodige services beschikbaar te stellen aan de Android-applicatielaag (o.a Webkit, SQLite ,OpenGL, OpenSSL,...). Native libraries worden ook gebruikt om hardware-specifiek te ontwikkelen.

De Native Development Kit, of NDK, is een add-on voor de SDK die helpt om native code in een Android-applicatie te integreren. Dit is code in C/C++ die platform-specifieke functies aanspreken via de API's die beschikbaar zijn in C/C++. De NDK laat toe om in de Android-applicatie native code op te roepen en zelf native libraries te includeren.

Vanaf Gingerbread (Android 2.3) wordt deze NDK verder uitgebreid met de introductie van de *NativeActivity*-klasse waarmee nu zelf een volledige *Activity* in C of C++ kan geschreven worden.

De belangrijkste motivatie om delen van een applicatie in native code te ontwikkelen is performantie. De NDK ondersteunt wiskundige en grafische bibliotheken alsook enkele systeemondersteunende bibliotheken. Grafische en rekenkundig intensieve applicatie zijn de beste kandidaten voor de NDK. De NDK is dus heel belangrijk voor mobiele games. De NDK laat toe om delen van een applicatie in C/C++ te schrijven. Voor specifieke types van applicaties kan dit handig zijn zodat bestaande libraries in die talen hergebruikt kunnen worden en voor de mogelijk betere performantie.

De NDK biedt voor de meeste apps echter weinig of geen voordelen. De ontwikkelaar moet zelf de voordelen met de nadelen afgewegen. Het gebruik van native code in Android zal namelijk over het algemeen niet resulteren in een merkbaar betere performantie, maar zal wel altijd zorgen voor een hogere graad van complexiteit van de app. Het wordt aangeraden de NDK enkel te gebruiken indien dit essentieel is voor de applicatie, nooit omdat de voorkeur uitgaat om in die taal te programmeren.(Gargenta, 2011)

Tot slot is het nog belangrijk om te vermelden dat de standaard browser van Android gebruik maakt van dezelfde WebKit-engine als de Safari browser van iOS. Ook in *WebView*-componenten wordt van deze engine gebruik gemaakt. Geïnteresseerden in het ontwikkelen van native apps voor Android kunnen verder terecht op <http://developer.android.com>.

## 2.3 Windows Phone

Microsoft had al voor Apple en Google een mobiel besturingssysteem, Windows Mobile (eerste release in 2000). De snelle veranderingen in de mobiele sector zorgden er echter voor dat dit besturingssysteem plots achterop hinkte op de concurrentie. Daarom moest Microsoft grote veranderingen aanbrengen aan het besturingssysteem. In 2010 werd deze dan geïntroduceerd als Windows Phone (versie 7, als opvolger voor Windows Mobile 6.5).

Windows Phone apps worden ontwikkeld in C# of Visual Basic. Voor de grafische interface wordt XAML (Extensible Application Markup Language, gebaseerd op XML) gebruikt. Direct3D-apps worden ontwikkeld in C++ om volledig gebruik te maken van de capaciteiten van de grafische hardware van het toestel. Visual Studio biedt volledige ondersteuning voor het ontwikkelen van deze applicaties. Meer info is te vinden op <http://dev.windowsphone.com>.

De recentste versie is Windows Phone 8 die opnieuw grote verschillen heeft met Windows Phone 7. De nieuwste versie is gebaseerd op de Windows NT kernel met veel gemeenschappelijke componenten met Windows 8. De aandachtige lezer kan hier misschien uit afleiden dat Microsoft de strategie heeft om de kloof tussen deze twee besturingssystemen te verkleinen om het mogelijk te maken applicaties op beide systemen te draaien. Een ander groot verschil tussen versie 7 en 8 is de browser. Vanaf Windows Phone 8 wordt gebruik gemaakt van de nieuwe Internet Explorer 10 Mobile browser. Deze browser maakt een grote sprong voorwaarts op gebied van HTML5-ondersteuning. Zowel Windows 8 als Windows Phone hebben de Modern UI (eerder gekend als Metro UI, zie figuur 2.2). De zogenaamde native Windows store apps in Windows en Windows Phone 8 kunnen volledig met HTML5 ontwikkeld worden. Deze applicaties die ontwikkeld zijn met de door Microsoft geleverde HTML5-SDK werken op beide besturingssystemen.



**Figuur 2.2:** Microsoft presenteert Windows en Windows Phone 8. Links is de UI van Windows Phone te zien, rechts de UI van Windows 8.

## 2.4 BlackBerry 10

De eerste Blackberry smartphone met het gelijknamige besturingssysteem kwam in 1999 op de markt en werd ontwikkeld door RIM (Research In Motion). BlackBerry's waren heel populair in de zakenwereld. Maar net als Microsoft zag RIM zich genoodzaakt om zijn besturingssysteem grondig onder handen te nemen. Begin 2013 onthulde het bedrijf, dat zich ook volledig omdoopte tot BlackBerry, de eerste smartphones met BlackBerry 10. Terwijl de vorige versies van het besturingssysteem op Java gebaseerd waren, is het nieuwe besturingssysteem gebouwd op het QNX-besturingssysteem, een Unix-achtig besturingssysteem.

BlackBerry 10 is een open platform dat voorziet in een aantal ontwikkelingstalen en runtimes. Zo zijn er SDK's beschikbaar voor C/C++, HTML5 en ActionScript. De BlackBerry-browser biedt momenteel de beste HTML5-ondersteuning van alle mobiele browsers. BlackBerry 10 beschikt zelfs over een Android runtime die het mogelijk maakt om Androidapplicaties te gebruiken. Meer info op <http://developer.blackberry.com>.

## 2.5 Sailfish

Sailfish is een nieuw besturingssysteem dat momenteel nog in ontwikkeling is door het bedrijf Jolla. Het team dat hier aan werkt bestaat vooral uit ingenieurs die bij Nokia aan Maemo en MeeGo werkten. De ontwikkeling van deze twee mobiele besturingssystemen werd stopgezet. Nokia zou op termijn het succesvolle Symbian-besturingssysteem inruilen voor MeeGo maar verlegde uiteindelijk de volledige focus op Windows Phone.

Native applicaties worden ontwikkeld met het Qt<sup>3</sup>-framework in C++. Daarnaast biedt het besturingssysteem ook de mogelijkheid om Android applicaties te draaien in een Android runtime. Momenteel wordt er ook bekeken hoe HTML5-applicaties ondersteund zullen worden. Meer info op [www.sailfishos.org](http://www.sailfishos.org).

## 2.6 Ubuntu

Ubuntu is een populaire Linux-distributie. Canonical, het bedrijf achter Ubuntu, kondigde begin januari 2013 aan dat het besturingssysteem nu ook naar mobiele toestellen komt. Meer nog, Ubuntu wordt nu een platform voor desktops, smartphones, tablets én TV's, zie figuur 2.3. Ubuntu heeft een heel andere aanpak dan de andere mobiele besturingssystemen en zou wel eens een grote invloed kunnen hebben op de concurrentie, zelfs buiten de mobiele sector. Hoewel dit allemaal niet uitgebreid wordt besproken is het zeker de moeite waard om dit eens verder te bekijken op <http://www.ubuntu.com/devices/phone>.

---

<sup>3</sup>Qt is een C++-framework voor cross-platform applicaties en gebruikersinterfaces



**Figuur 2.3:** Ubuntu op tv, desktop, tablet en smartphone.

Applicaties die ontwikkeld worden met de native SDK worden geschreven in C/C++, de grafische interface wordt opgebouwd met QML<sup>4</sup> en JavaScript. Daarnaast worden webapplicaties gebaseerd op HTML5-technologiën net als native applicaties in het besturingssysteem geïntegreerd en biedt Ubuntu grootte ondersteuning met API's voor deze webtoepassingen. Zo kunnen webapps op dezelfde manier notificaties weergeven als native apps, iets wat op andere besturingssystemen tot op heden nog niet mogelijk is voor webapplicaties.

## 2.7 Tizen

Tizen is een open source platform dat voor het eerst in 2012 gelanceerd werd. Het platform kan gebruikt worden voor verschillende toestellen zoals smartphones, tablets, netbooks, boordcomputers in voertuigen en smart-TV's. De belangrijkste componenten van het besturingssysteem zijn de Linux-kernel en de WebKit-runtime. Het OS, dat grotendeels ontwikkeld wordt door Samsung en Intel, zal naar verwachting nog dit jaar voor het eerst op smartphones gebruikt worden.

Tizen voorziet een flexibele en robuuste omgeving voor ontwikkelaars die gebaseerd is op HTML5. De SDK en API van Tizen stellen ontwikkelaars in staat om met HTML5-technologiën applicaties te ontwikkelen die op meerdere toestellen kunnen draaien. De browser in Tizen heeft een score van 492/500 op [www.html5test.com](http://www.html5test.com). Meer info over het besturingssysteem kan bekijken worden op [www.tizen.org](http://www.tizen.org).

<sup>4</sup>Qt Meta Language. Een declaratieve programmeertaal gebaseerd op JavaScript en CSS voor de ontwikkeling van de grafische interfaces.

## 2.8 Firefox OS

Op de MWC<sup>5</sup>-beurs in februari 2013 kondigde browserfabrikant Mozilla zijn eigen mobiel besturingssysteem Firefox OS aan. Mozilla gaat de mobiele sector in met een reden: het wil het mobiele internet uit de greep van de gesloten app stores van Apple, Google en Microsoft halen. Mozilla ziet grote overeenkomsten met het internet ten tijde van Internet Explorer 6, waarbij alternatieve browsers als Firefox en Chrome nodig waren om sitebouwers en Microsoft ervan te overtuigen met webstandaarden te werken waar iedereen mee overweg kan. (Wokke, 2013)

Firefox OS is volledig gebouwd met HTML5 en andere open webstandaarden. Het besturingssysteem kan vergeleken worden met het Chrome OS van Google dat ook volledig binnen een eigen browseromgeving draait. Native applicaties worden met HTML5-technologieën ontwikkeld en hebben volledige toegang tot de hele waaier functies die het besturingssysteem en de hardware bieden. Mozilla levert ook uitstekend werk voor de performantie van zijn JavaScript-engine.

## 2.9 Conclusie

Het is duidelijk dat bij elk mobiel besturingssysteem een welbepaalde set technologieën hoort om native applicaties te ontwikkelen. Toch zijn er daarnaast nog andere mogelijkheden om applicaties te ontwikkelen dankzij onder andere technologieën en eigenschappen die verschillende besturingssystemen gemeenschappelijk hebben. Op basis van de gebruikte technologieën kunnen mobiele applicaties in drie categorieën onderverdeeld worden.

Tot de eerste categorie behoren de native applicaties. Deze toepassingen worden ontwikkeld met een SDK die hoort bij het besturingssysteem en worden dus geschreven in de bijhorende programmeertalen. Native applicaties hebben volledige toegang tot de API's die het besturingssysteem beschikbaar stelt en draaien ook zelfstandig in het besturingssysteem. Deze apps worden ook ontwikkeld volgens paradigma's die eigen zijn aan het besturingssysteem en zijn dus ook volledig platformafhankelijk. Het sterkste punt van dit soort applicaties is natuurlijk performantie en functionaliteit.

Webapplicaties, die verder besproken worden in hoofdstuk 3, vormen de tweede categorie. Deze applicaties worden ontwikkeld met webtechnologieën en draaien binnen een browseromgeving. Elk besturingssysteem beschikt normaal wel over minstens één browser en dus kunnen dit soort applicaties als platformonafhankelijk beschouwd worden. Sommige besturingssystemen zorgen zelf voor een mooie integratie van deze apps zodat het onderscheid met native toepassingen veel kleiner wordt.

Met de komst van HTML5 hebben ontwikkelaars nu ook betere tools om apps te ontwikkelen met webtechnologieën. Toch is het belangrijk dat er rekening gehouden wordt

---

<sup>5</sup>Mobile World Congress

met de verschillende browsers en de technologieën die ze ondersteunen. De HTML5-standaard is niet volledig af en niet alle browsers implementeren hetzelfde van deze standaard. Zo worden bijvoorbeeld nog niet overal dezelfde *device-API's* ondersteund.

De laatste categorie zijn de hybride applicaties, die aan bod komen in hoofdstuk 4. Deze applicaties proberen het beste van twee werelden te combineren: toegang tot de native API's en platformonafhankelijkheid. Bij de ontwikkeling van deze applicaties wordt ook gebruik gemaakt van technologieën die niet gebruikt worden bij de ontwikkeling van native apps. Deze applicaties draaien meestal niet zelfstandig in het besturingssysteem maar maken gebruik van de browserruntime of een andere runtime die bij de applicatie gebundeld wordt.

In het geval van BlackBerry 10 bijvoorbeeld is er zelf een runtime voor Android-applicaties voorzien door het besturingssysteem zelf. In deze runtimes wordt er dan een brug gemaakt naar de native API's.

In tabel 2.1 wordt een vergelijking gemaakt tussen deze verschillende categorieën. Tabel 2.2 geeft een overzicht van deze categorieën, de mogelijke programmeertalen en de besturingssystemen die op die manier ondersteund kunnen worden.

Feature	Native	Hybride	Web
Programmeertaal	enkel native	verschillende	enkel webtalen
Platformonafhankelijkheid	geen	matig tot hoog	zeer hoog
beschikbaarheid API's	zeer hoog	hoog	laag tot matig
Geavanceerde graphics	hoog	matig tot hoog	matig
Upgradeflexibiliteit	laag	matig	hoog
Eenvoud van installatie	hoog (appstore)	hoog (appstore)	matig (via browser)

**Tabel 2.1:** Vergelijking soorten applicaties

<b>Web</b>	HTML5			
<b>Hybride</b>	HTML5(PhoneGap), C/C++(Xamarin), Ruby(Rhomobile),...			
<b>Native</b>	Obj-C	Java	C#,VB.NET,HTML5	C/C++, HTML5, ActionScript
	iOS	Android	Windows Phone	BlackBerry 10

**Tabel 2.2:** Soorten applicaties en programmeertalen

# **Hoofdstuk 3**

## **Webapplicaties**

Webapplicaties worden ontwikkeld aan de hand van webtechnologieën.

Met de komst van HTML5 hebben ontwikkelaars nu meer mogelijkheden om RIA's(Rich Internet Applicaties) te bouwen. In dit hoofdstuk wordt HTML5 geïntroduceerd.

Omdat iOS een belangrijk besturingssysteem is en ook als eerste zorgde voor een mooie integratie van webapps in het besturingssysteem wordt de ondersteuning van webapplicaties in iOS besproken. Tot slot komen verschillende frameworks en ontwikkelingstools aan bod waar ontwikkelaars een beroep op kunnen doen om mobiele webapplicaties te ontwikkelen. In hoofdstuk 5 komen de conclusies hieruit aan bod.

## 3.1 HTML5

*"If HTML were a movie, HTML5 would be its surprise twist." - MacDonald, 2011*



**Figuur 3.1:** HTML5-logo (<http://www.w3.org/html/logo>)

HTML zou sinds 1998 niet meer verder ontwikkeld worden. De officiële organisatie voor webstandaarden, W3C, koos voor XHTML als opvolger van HTML. XHTML heeft grotendeels dezelfde syntaxconventies als HTML, maar hanteerde striktere regels. In 2000 werd XHTML 1.0 een W3C-standaard. Hoewel XHTML een succes leek maakte het voor browsers, die de XHTML-syntax wel verstanden, niets uit of de striktere regels gevuld werden. De oplossing hiervoor moest XHTML 2.0 zijn, die de browsers zou moeten verplichten om niet-wel gevormde XHTML-pagina's te weigeren. Ook werden een aantal aanpassingen doorgevoerd in de conventies die overgeerfd waren van HTML. Theoretisch gezien waren deze aanpassingen logischer en netter. In de praktijk werd iedereen verplicht om zijn manier van webpagina's schrijven te veranderen zonder dat er echt nieuwe functionaliteit toegevoegd was om dit de moeite waard te maken. Bovendien evolueerde deze technologie ijzig langzaam waardoor het enthousiasme van ontwikkelaars geleidelijk aan verdween.

Een groep ontwikkelaars, onafhankelijk van het W3C, begon de toekomst van het web op een andere manier te bekijken. Terwijl het W3C de focus legde op wat er verkeerd was met HTML, legden zij de focus op wat er tekort was in HTML om ontwikkelaars meer

mogelijkheden te bieden. HTML begon als een tool om documenten weer te geven en met de toevoeging van JavaScript veranderde dit in een systeem om webapplicaties te ontwikkelen. Hoewel zo'n webapplicaties reeds veel konden doen was het niet eenvoudig om er een te ontwikkelen en alle onderdelen consistent te laten samenwerken in verschillende browsers. Vooral browserontwikkelaars maakten zich hier zorgen over en wilden het eenvoudiger maken om webapplicaties te ontwikkelen. Hiervoor dienden Opera en Mozilla aanvragen in om in XHTML meer features te introduceren die gericht waren op de ontwikkelaars. Toen zij geen gehoor kregen vormden Opera, Mozilla en Apple in 2004 de WHATWG (Web Hypertext Application Technology Working Group) om nieuwe oplossingen te bedenken.

De WHATWG wilde HTML uitbreiden, in plaats van te vervangen, op een manier die zorgde dat compatibiliteit met de oudere technologieën niet verdween. Hun werk leidde tot HTML5. De naam HTML5 slaat niet alleen op de HTML-syntax maar ook andere technologieën zoals JavaScript, CSS3, SVG (Scalable Vector Graphics), API's, enzovoort. In 2007 besliste het W3C om te stoppen met de ontwikkeling van XHTML 2.0 en samen te werken aan de HTML5-standaard. Als gevolg van deze samenwerking wordt nu gesproken van de W3C HTML5-standaard, die een snapshot is van WHATWG HTML5. De WHATWG blijft hieraan verder werken en noemen dit nu simpelweg HTML, waarbij ze vermelden dat HTML een levende standaard is. Dit wil zeggen dat HTML-pagina's nooit zullen stoppen met werken, nooit een versienummer nodig zullen hebben en dat webontwikkelaars nooit zullen moeten upgraden om compatibiliteit met nieuwere browsers te voorzien. Dit wil ook zeggen dat ontwikkelaars steeds kunnen kiezen om gebruik te maken van de nieuwste features en dat browsers er voor kunnen kiezen om deze te ondersteunen. HTML5 specificeert niet alleen hoe ontwikkelaars moeten werken maar ook hoe browsers moeten werken en fouten afhandelen. (MacDonald, 2011)

De HTML5-syntax voegt een aantal nieuwe tags toe die nieuwe functionaliteit bieden (zoals canvas, audio, video,...) en het document ook semantischer maken (article, section, verschillende soorten inputtypes,...). Door met de HTML5-syntax meer te focussen op semantiek kan een webpagina meer informatie bieden over zijn inhoud en deze informatie ook uitwisselen met bijvoorbeeld zoekmachines of andere programma's. HTML5 heeft ook als doel om plugins zoals Adobe Flash Player overbodig te maken. Denk hierbij maar aan YouTube, waar het eerst niet mogelijk was om video af te spelen zonder eerst de plugin van Adobe Flash Player te installeren. Dit wordt nu mogelijk gemaakt met het video-element. HTML5 focust echt op wat ontwikkelaars nodig hebben om applicaties te ontwikkelen met webtechnologieën. Zo bieden de HTML5-technologieën de mogelijkheid om offline op te slaan, 3D-animaties uit te voeren, te werken met threads en sockets, API's om de hardware aan te spreken, enzovoort.

## 3.2 Ondersteuning van webapplicaties in iOS

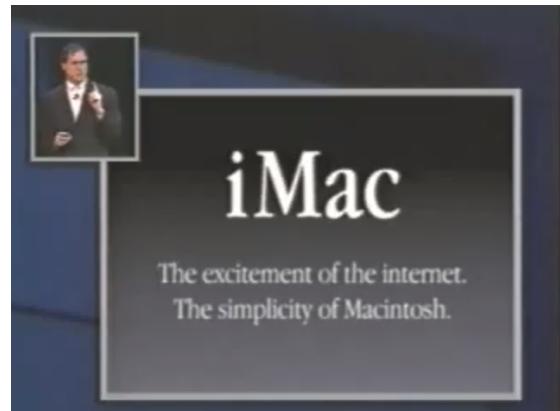
In 1998 presenteerde Apple de iMac.

Het grootste verkoopargument van deze desktop was dat deze snel en eenvoudig in verbinding kon gebracht worden met het internet. De 'i' in iMac stond dan ook in de eerste plaats voor het internet, zie figuur 3.2. Bij de introductie van de iPhone kan gesteld worden dat de 'i' nog steeds staat voor het internet. Ten eerste is het een smartphone die constant in verbinding met het internet kan staan. Ten tweede waren de enige applicaties die extra geïnstalleerd konden worden webapplicaties.

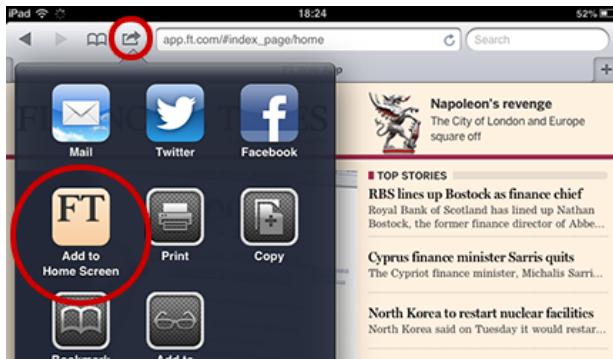
Apple was de eerste die webapplicaties integreerde in het besturingssysteem en had dus ook veel belang bij de oprichting van de WHATWG in 2004.

Apple maakte de integratie van webapps in het besturingssysteem mogelijk door gebruik te maken van een aantal meta-tags waarvan enkele specifiek voor de Safari browser zijn. Deze browser kan op basis van deze meta-tags bepalen hoe de applicatie weergegeven moet worden. Een webapplicatie kan zo geconfigureerd worden om er uit te zien als een native applicatie die op zichzelf staat. Het is mogelijk om een icoontje in te stellen om weer te geven op het startscherm, een afbeelding in te stellen die getoond wordt als de applicatie wordt opgestart, de applicatie weer te geven op het volledige scherm, de weergave van de statusbar aan te passen en tot slot is het ook nog mogelijk om te linken naar andere native applicaties zoals bellen, sms'en, mailen, maps, youtube en iTunes. (Apple, 2009)

De appstore maakte een jaar na de introductie van de iPhone zijn intrede. Hierbij werd ook een SDK beschikbaar gesteld om native applicaties te ontwikkelen. Dit wil evenwel niet zeggen dat er niemand meer kiest om een webapplicatie te ontwikkelen in plaats van een native applicatie. Een goed voorbeeld is de mobiele applicatie van de Financial Times. Deze webapplicatie is momenteel enkel voor iOS geoptimaliseerd, maar er zijn plannen om meerde besturingssystemen te ondersteunen in de toekomst. Naast de applicaties voor Windows 8 en Android is de FT Web App de meeste complete van de drie. De webapp kan geïnstalleerd worden op een iPad of iPhone door met de safari browser te surfen naar <http://app.ft.com>. Vervolgens kan deze app aan het startscherm toegevoegd worden via het menu van de Safari browser. Omdat deze applicatie niet via de appstore verspreid wordt moet er ook geen goedkeuringsprocedure doorlopen worden en komen updates automatisch en veel sneller tot bij de gebruiker.



Figuur 3.2: Presentatie iMac in 1998.



Figuur 3.3: De Financial Times app installeren op iOS.

### 3.3 Frameworks

Voor de ontwikkeling van webapplicaties wordt gebruik gemaakt van verschillende programmeertalen en technologieën. Zeker bij grotere applicaties wordt het dan al een hele uitdaging om al deze technologieën naadloos te laten samenwerken. Gelukkig bestaat er een heel arsenaal aan frameworks en JavaScript-bibliotheken waar ontwikkelaars gebruik van kunnen maken voor hun webapplicatie. Denk hierbij maar aan jQuery, Knockout.js, Modernizer.js, YepNo.js, enzovoort. Zelfs specifiek voor mobiele applicaties bestaan er heel wat frameworks: jQuery Mobile, Sencha Touch, JoApp, KendoUI,...

In de volgende paragrafen worden jQuery Mobile en Sencha Touch, de bekendste en populairste frameworks, besproken.

#### 3.3.1 jQuery Mobile

jQuery Mobile ([www.jquerymobile.com](http://www.jquerymobile.com)) is een framework dat gebouwd is op de JavaScript-bibliotheken jQuery en jQuery UI. Het framework voorziet Javascript- en CSS-code om vooral de grafische kant van de mobiele applicatie te ontwikkelen. Daarnaast biedt het framework ook tools voor het bouwen van een aangepast thema en een aangepaste Javascript- en CSS-build van het framework. De documentatie is in orde en er zijn ook een heel wat boeken over jQuery Mobile beschikbaar. Ook zijn er verschillende plugins en uitbreidingen door derde partijen beschikbaar.

Het jQuery Mobile framework maakt gebruik van de HTML5 'data'-attributen om de opmaak en configuratie van componenten te initialiseren. Bij sommige componenten wordt de opmaak automatisch toegepast, bij andere componenten of wanneer men de opmaak meer wil specificeren moeten enkele 'data'-attributen ingesteld worden. Zo wordt de opmaak bij links in toolbars, buttons en input-buttons automatisch toegepast. Een gewone link kan dan als een knop worden opgemaakt door het 'data-role'-attribuut in te vullen met 'button'. Vervolgens kan men deze componenten nog verder configureren met een hele reeks 'data'-attributen. Het attribuut 'data-theme' geeft bijvoorbeeld aan volgens

welk thema de component opgemaakt moet worden. Met het attribuut 'data-icon' kan dan bijvoorbeeld ook een icoontje worden ingesteld.

Wie een basiskennis van HTML heeft kan dus al snel aan de slag met jQuery Mobile.

Pagina's worden in jQuery Mobile aangeduid door het 'data-role'-attribuut in te stellen op 'page'. Hierdoor kan het framework de nodige opmaak voorzien en de juiste events koppelen. Zo kunnen meerdere pagina's gedefinieerd worden in één HTML-document. Deze aanpak biedt enkele voordelen:

- het framework kan op deze manier zorgen voor vlotte paginaovergangen. Dit zorgt ervoor dat de applicatie de native *look-and-feel* meer kan benaderen;
- jQuery Mobile kan nu ook automatisch navigatie afhandelen door een terugknop en *deep linking*<sup>1</sup> te voorzien;
- omdat alle pagina's in één document zitten hoeft de browser niet voor elke pagina een nieuw document te laden over het netwerk. Dit verbetert de performantie van de applicatie en zal ook minder belastend zijn voor de batterij van het mobiel toestel.

Componenten die geconfigureerd worden als pagina's moeten directe kinderen zijn van het body-element en het is niet mogelijk om pagina's te nesten. Code 3.1 geeft een voorbeeld van hoe een HTML-body er zou kunnen uit zien.

```
<!-- first page -->
<section id="page1" data-role="page">
  <header data-role="header"><h1>jQuery Mobile</h1></header>
  <div data-role="content" class="content">
    <p>First page!</p>
    <p><a href="#page2">Go to Second Page</a></p>
  </div>
  <footer data-role="footer"><h1>Footer</h1></footer>
</section>

<!-- second page -->
<section id="page2" data-role="page">
  <header data-role="header"><h1>jQuery Mobile</h1></header>
  <div data-role="content" class="content">
    <p>Second page!</p>
    <p><a href="#page1">Go back to First Page</a></p>
  </div>
  <footer data-role="footer"><h1>Footer</h1></footer>
</section>
```

**Codefragment 3.1:** Twee 'jQuery Mobile'-pagina's in de body van één HTML-document

---

<sup>1</sup>Deep links zijn links die niet verwijzen naar de startpagina van de applicatie maar naar een bepaald deel van de toepassing

Het is ook mogelijk om de pagina's in aparte documenten te schrijven. Wanneer dan gelinkt wordt naar een externe pagina in plaats van naar een interne pagina met het ID, dan zal jQM asynchroon de pagina ophalen en injecteren in het huidige document om zo toch nog alle navigatiefunctionaliteit te kunnen aanbieden. Hierbij zal jQM de externe pagina ophalen en zoeken naar het eerste element met een 'data-role' die ingesteld is op 'page'. Vervolgens wordt dit element volledig geïnjecteerd in de DOM-structuur van het huidige document. Andere elementen, buiten het eerste element dat gevonden wordt, worden genegeerd. Als er geen pagina kan opgehaald worden of gevonden kan worden zal een foutmelding weergegeven worden. (Reid, 2011)

jQuery Mobile ondersteunt het grootste deel van alle moderne desktop-, smartphone-, tablet- en e-reader-platformen. Daarnaast worden ook feature-gsm's<sup>2</sup> en oudere browsers ondersteund door de progressieve aanpak van jQuery Mobile. Zo zal de applicatie er misschien minder goed uitzien en de native *look-and-feel* niet benaderen, maar de basis HTML-functionaliteit zal er nog altijd zijn.

### 3.3.2 Sencha Touch

Sencha Touch is een framework voor mobiele applicaties dat is ontwikkeld door Sencha. Het framework is voortgekomen uit de ideeën en gedachten achter het succesvolle Sencha Ext JS<sup>3</sup> framework. Gebruik makend van gestandaardiseerde HTML5-technologieën biedt het framework ontwikkelaars de mogelijkheid om platformonafhankelijke toepassingen te ontwikkelen die de *look-and-feel* van native applicaties zoveel mogelijk benaderen.

Deze mobiele applicaties kunnen beschikbaar gesteld worden via een webserver of een hybride applicatie door gebruik te maken van de Sencha native packager of andere tools zoals bijvoorbeeld PhoneGap (zie 4.1).

Net zoals Ext JS, brengt Sencha Touch een native gebruikerservaring door een ingenieuze mix van HTML5, CSS3 en JavaScript die geoptimaliseerd wordt voor de best mogelijke mobiele ervaring. Hierbij wordt rekening gehouden met de beperkingen van mobiele toestellen zoals de beperkte CPU-kracht en beschikbaarheid van geheugen voor applicaties. Sencha Touch is ook zoals Ext JS ontworpen met het oog op flexibiliteit en uitbreidbaarheid.

Er zijn veel gelijkenissen tussen Ext JS en Sencha Touch. Deze twee frameworks zijn echter niet hetzelfde aangezien er grote verschillen zijn tussen desktopapplicaties en mobiele applicaties. Sencha Touch is geen extensie van Ext JS maar is van de grond af opgebouwd met hier en daar hergebruik van code uit het Ext JS framework.

Typisch aan mobiele applicaties is de interactie op basis van aanraking. Sencha Touch beschikt over een *gesture*-bibliotheek waardoor eenvoudig op deze op aanraking gebaseerde

<sup>2</sup>Feature-gsm's bieden meer functionaliteit, zoals bijvoorbeeld een browser, dan een gewone gsm maar zijn geen smartphones.

<sup>3</sup>Ext JS is het JavaScript-framework van Sencha dat gericht is op desktopapplicaties.

gebeurtenissen, zoals tikken, knijpen en slepen, kan worden ingehaakt. Een manier waarop Sencha Touch de native ervaring benadert is door het gebruik van een aangepaste *Scroller*-klasse die gebaseerd is op fysicawetten. Deze klasse maakt gebruik van de hardwareversnelde CSS3-transities en belangrijke variabelen zoals frictie- en springeffecten om het scroll-effect op een natuurlijke manier weer te geven.  
(Clark & Johnson, 2012)(Garcia, De Moss & Simoens, 2013)

Het Sencha Touch framework is heel erg uitgebreid en biedt ontwikkelaars heel wat mogelijkheden en tools (zie paragraaf 3.4.4). Naast het opvangen van de verschillen en tekortkomingen bij de HTML5-ondersteuning in de verschillende browsers biedt het framework ook enkele verbeteringen en uitbreidingen op de HTML5-technologie. Zo werkt het framework bijvoorbeeld met een eigen delta-updatesysteem, dat gebouwd is bovenop de offline-technologie van HTML5, waardoor enkel de verschillen moeten gedownload worden en niet het hele document waarin aanpassingen zijn gebeurd. Bovendien biedt het framework ook de mogelijkheid om verschillende soorten grafieken weer te geven, waar er anders beroep zou moeten gedaan worden op een andere JavaScript-bibliotheek.

Ontwikkeling van een applicatie met het Sencha Touch framework gebeurt hoofdzakelijk in JavaScript. Enkel voor templates en custom componenten zal de ontwikkelaar HTML moeten schrijven. Voor de opmaak wordt natuurlijk CSS gebruikt, maar ook hier voorziet het framework een uitbreiding, namelijk SASS (Syntactically Awesome Style Sheet, [www.sass-lang.com](http://www.sass-lang.com)). SASS is een uitbreiding op CSS3 die het mogelijk maakt om gebruik te maken van variabelen, nesting, functies en overerving.

Het Sencha Touch framework zet deze SASS dan automatisch om in CSS3 met behulp van de Compass-command-line-tool (<http://compass-style.org>). Dankzij het klassensysteem is hergebruik van code en overerving mogelijk. Ook mede door de layoutengine is het ontwikkelen van een Sencha Touch app gelijkaardig aan het ontwikkelen van een interface in Java Swing. Het framework injecteert de HTML-componenten in de DOM-structuur van het HTML-document dat gebruikt wordt om de applicatie op te starten.

Hoewel het Sencha Touch framework een steilere leercurve heeft, biedt het wel grote voordelen. Dankzij het klassensysteem en de MVC-architectuur (Model-View-Controller) blijft de code makkelijker te onderhouden. Het framework evolueert ook vlot waarbij er bijvoorbeeld functionaliteit wordt toegevoegd of verbeteringen zijn in performantie. De documentatie is heel uitgebreid maar mist soms wat diepte. Soms is de documentatie nog niet helemaal up-to-date. In dit geval kan er echter wel gerekend worden op de grote actieve community. Op het Sencha-forum helpen ontwikkelaars elkaar met vragen, ook medewerkers zijn actief op deze fora. Sencha heeft op zijn website ([www.sencha.com](http://www.sencha.com)) een uitgebreid aanbod aan voorbeelden. Op [www.try.sencha.com](http://www.try.sencha.com) en [www.senchafiddle.com](http://www.senchafiddle.com) is het mogelijk om ook zelf eens snel met het framework te experimenteren.

## 3.4 Ontwikkelingstools

In principe kunnen webapplicaties volledig ontwikkeld worden met een doodgewone tekstbewerker. Toch is een goede ontwikkelingstool wenselijk. In volgende paragrafen worden vier verschillende ontwikkelomgevingen besproken.

### 3.4.1 Embarcadero HTML5 Builder

Op de ontwikkelingsafdeling van Cerm wordt reeds gebruik gemaakt van de producten van Embarcadero ([www.embarcadero.com](http://www.embarcadero.com)). Bij de nieuwste bundel zou ook HTML5 Builder er gratis bij komen. Daarom was het interessant om dit eens te bekijken.

HTML5 Builder integreert het jQuery Mobile JavaScript framework voor de ontwikkeling van mobiele applicaties. Daarnaast wordt ook het PhoneGap framework, dat verder besproken wordt in 4.1, geïntegreerd om hybride applicaties te bouwen met behulp van webtechnologie. Hierdoor wordt bij installatie op Windows meteen ook alles wat nodig is om te ontwikkelen voor het Android platform geïnstalleerd. Zo is het dan ook mogelijk om de applicatie meteen als een hybride app te bouwen voor één van de ondersteunde platformen vanuit de HTML5 Builder. Inclusief bij HTML5 Builder zit een commerciële licentie voor het gebruik van de TeeChart JavaScript-bibliotheek. TeeChart is een bibliotheek voor het weergeven van kaarten en grafieken op het HTML5 canvas-element. Een enkele licentie voor TeeChart kost apart \$129 ([www.steema.com](http://www.steema.com)).

Naast HTML, JavaScript en CSS ondersteunt HTML5 Builder ook PHP. Bij de FAQ's over HTML5 Builder is dan ook te lezen dat dit eigenlijk de vervanger is voor Embarcadero RadPHP. Pagina's zijn dan ook eigenlijk PHP-scripts die de webpagina's genereren. Voor de ontwikkeling van deze pagina's biedt HTML5 Builder een grafische drag-and-drop-tool. Hiermee kunnen componenten op een canvas geslept worden dat ingesteld kan worden op verschillende vormfactoren zoals bijvoorbeeld een iPhone of iPad. Deze componenten kunnen dan ook op een grafische manier geconfigureerd worden. De ontwikkelaar wordt dus zoveel mogelijk van de code afgescheiden. De gegenereerde code kan bekijken worden in de tag editor maar kan niet aangepast worden. Bij deployment kan er voor gekozen om alle client-side code te genereren.

De eigen ervaring met de HTML5 Builder was niet zo positief. Er kwamen heel wat tekortkomingen naar voor. De interface van de HTML5 Builder zelf werkt niet goed. Om dit op te lossen moesten er aanpassingen gedaan worden aan de scherminstellingen, terwijl het wel de standaardwaarde was die was ingesteld. Ook de *tag editor*, die de gegeneerde code zou moeten weergeven, is zeer verwarring en doet niet wat er van verwacht zou worden. In tegenstelling tot wat de naam doet vermoeden, kan er helemaal niets aangepast worden. De *tag editor* geeft ook enkel iets weer als er op een component geklikt wordt. De output doet op het eerste zicht vermoeden dat de output voor de volledige pagina getoond wordt maar het is enkel de gegenereerde code voor de aangeklikte component die wordt weergegeven in een HTML-document.

Wanneer in de grafische designer gewisseld wordt tussen vormfactoren van het canvas, bijvoorbeeld van iPhone naar iPad, is het meteen duidelijk dat de interface zich niet aanpast. Alle componenten krijgen standaard vaste afmetingen en een percentage instellen voor de breedte bleek niet mogelijk te zijn.

Voor navigatie moet tussen verschillende pagina's verwezen worden naar de overeenkomstige PHP-bestanden. Bij het testen van de applicatie treedt er bij elke navigatie naar een andere pagina een vertraging op doordat de HTML-code gegenereerd moeten worden.

Volgens de documentatie zou bij het deployen voor een client (alle client-side code éénmaal genereren) alle links juist omgezet worden, dit bleek echter niet het geval te zijn. Wanneer dan links gebruikt worden naar de HTML-documenten kan de applicatie niet getest worden zonder eerst te deployen voor een client zodat alle HTML-code gegenereerd is.

Het contacteren van de klantendienst leverde ook geen bevredigende antwoorden op. Op het forum werden er veel gelijkaardige negatieve ervaringen gedeeld. Het gebruik van de HTML5 Builder voelde eerder aan als een beperking en beroving op flexibel ontwerp.

### 3.4.2 Visual Studio 2012

Microsoft Visual Studio 2012 biedt zeer goede ondersteuning voor de HTML5-technologieën. Deze ontwikkelingsomgeving biedt IntelliSense-ondersteuning voor HTML5, CSS3, JavaScript en zelfs voor de jQuery JavaScript-bibliotheek.

Visual Studio 2012 is dan ook de tool om Windows 8 en Windows Phone 8 apps te ontwikkelen met HTML5-technologieën aangezien ook de WinJS JavaScript-bibliotheek volledig geïntegreerd is.

Ook voor debugging zijn er tools aanwezig binnen de ontwikkelomgeving. De eigen ervaring met deze ontwikkelingstool was dan ook zeer positief. Het is ook zeer eenvoudig om een bestaande webapplicatie met Visual Studio te openen en hierin verder te werken. ([www.microsoft.com/visualstudio](http://www.microsoft.com/visualstudio))

### 3.4.3 Netbeans 7.3

Netbeans biedt sinds versie 7.3 nieuwe ondersteuning voor HTML5-technologieën.

Ook hier is de IntelliSense-ondersteuning zeer goed. Deze tool heeft een ingebouwd Webkit-browser en met de Netbeans-plugin voor Chrome krijgt de ontwikkelaar een fantastische interactie tussen Chrome en de ontwikkelomgeving. Aanpassingen worden zo live zichtbaar en ook bij het debuggen is deze samenwerking uitstekend. Ontwikkelen met Netbeans wordt zo echt een plezier. Bovendien is deze IDE volledig gratis. Dankj video's en tutorials is het ook eenvoudig om hiermee aan de slag te gaan. ([www.netbeans.org](http://www.netbeans.org))

### 3.4.4 Sencha Architect, Sencha Cmd en Sencha Eclipse Plugin

Sencha biedt voor de ontwikkeling met hun frameworks (zie 3.3.2) ook enkele tools aan. Hierbij is Sencha Cmd de enige tool die gratis verkrijgbaar is. Deze command-line-tool kan voor verschillende taken gebruikt worden: het genereren van een nieuwe applicatie om mee

te beginnen ontwikkelen, het genereren van nieuwe klassen, het builden van de applicatie, enzovoort. Daarnaast is Sencha Architect apart beschikbaar met licentie of in een van de Sencha bundels waar ook de Sencha Eclipse Plugin bij zit. Zowel Sencha Architect als Sencha Cmd zijn ontwikkeld in JavaScript.

Sencha Architect is een visuele geïntegreerde ontwikkelomgeving die gebruikt kan worden voor het design, het ontwikkelen en het deployen van applicaties. Architect helpt op verschillende manieren om sneller en beter applicaties voor desktop of mobiele toestellen te bouwen. In deze paragraaf wordt Architect vooral in functie van het Sencha Touch framework besproken. Sinds de tweede versie, met de ondersteuning van Sencha Touch, is de naam veranderd van Ext Designer naar Sencha Architect.

Met behulp van de drag-and-drop-tool is het eenvoudig om het grafische gedeelte van de applicatie op te bouwen. In de toolbox zijn er verschillende componenten beschikbaar die versleept kunnen worden naar een canvas. De layout van dat canvas kan op verschillende vormfactoren ingesteld worden. Zo is het mogelijk om te wisselen tussen de voorop ingestelde afmetingen van een iPhone of een Nexus tablet of kan een aangepaste vormgeving toegevoegd worden. Bij het wisselen tussen deze instellingen is het duidelijk dat de layoutengine zijn werk doet om de interface dynamisch aan de vormfactoren aan te passen. Dit is één van de vele punten waarop Sencha Touch en Architect veel beter zijn dan de HTML5 Builder van Embarcadero en de daarbij gebruikte frameworks.

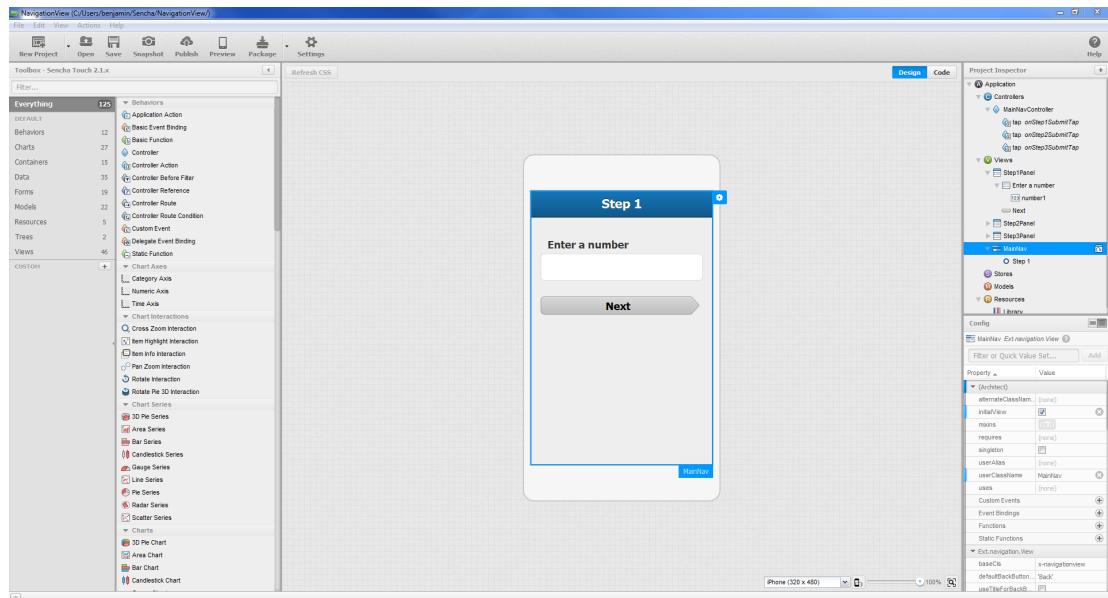
Achter de schermen wordt code gegenereerd wanneer gebruik gemaakt wordt van de visuele drag-and-drop-functionaliteit. De ontwikkelaar hoeft bij deze visuele aanpak echter niet in te boeten aan controle over de code en flexibiliteit van het ontwerp. Componenten kunnen aangepast en uitgebreid worden met behulp van de projectinspector en het configuratievenster. Op elk moment kan de code bekijken worden. Het toevoegen van code gebeurt altijd op een gecontroleerde begeleide manier. De code wordt door Architect in afzonderlijke delen aangeboden en van elkaar afgeschermd. Startend vanuit de projectinspector of het configuratievenster kunnen stukken code toegevoegd of aangepast worden of kan overgeschakeld worden naar de code editor. Een actie toevoegen aan een controller gebeurt bijvoorbeeld door de controller te selecteren en via het configuratievenster een actie toe te voegen. Door via de projectinspector deze actie te selecteren kan in de code editor de code voor deze actie geschreven worden. Deze code wordt dan automatisch ingevoerd in de controllerklasse. Deze manier van werken is even wennen maar biedt zeker een aantal voordelen. Deze aanpak zorgt er voor dat de code zoveel mogelijk geschreven wordt volgens de 'best practices'. De code wordt overzichtelijk gehouden en de ontwikkelaar wordt als het ware meer begeleid bij het schrijven van een klasse. Architect combineert de kracht van een visuele ontwikkelingstool met de flexibiliteit van een code editor.

Vanuit Architect kunnen applicaties ook geïmplementeerd worden als native applicatie voor iOS of Android. Sencha Touch beschikt hiervoor ook over een eigen device API. Deze API biedt echter nog niet alle functionaliteit en ondersteunt enkel iOS en Android.

Gelukkig werkt Sencha Touch volgens de webstandaarden en is het dus geen probleem om hier alternatieven zoals PhoneGap (zie 4.1) voor te gebruiken.

Sencha Architect is een ideaal middel om wat vertrouwd te geraken met het Sencha Touch framework en zal ook het opzoekwerk tijdens het ontwikkelen verminderen. Architect is beschikbaar voor Mac, Windows en Linux. Sencha biedt een gratis trialversie aan die dertig dagen geldig is. Eén enkele licentie kost 399 dollar of is inbegrepen in het Sencha Complete en Sencha Touch pakket. Meer info of een videointroductie is terug te vinden op [www.sencha.com/products/architect](http://www.sencha.com/products/architect). In de documentatie zijn ook verschillende gidsen en videotutorials terug te vinden om aan de slag te gaan met Architect.

Figuur 3.4 geeft een beeld van de interface van Architect. Links bevindt zich de toolbox met alle componenten. Deze componenten kunnen geslept worden op het designcanvas in het midden dat aangepast kan worden aan verschillende afmetingen zoals bijvoorbeeld die van een iPhone. Er kan hier ook gewisseld worden naar de code editor. Rechts bevindt zich de projectinspector die gestructureerd is volgens het MVC-patroon om het project overzichtelijk, gemakkelijk in onderhoud en schaalbaar te houden. Onder de projectinspector zit het configuratievenster waarmee de configuratie van de geselecteerde component mee bekijken of gewijzigd kan worden.



**Figuur 3.4:** De grafische interface van Sencha Architect.

Tot slot is er ook de Sencha Eclipse plugin. Dit is een plugin voor de Eclipse IDE die IntelliSense voor het Sencha framework toevoegt aan de Eclipse ontwikkelomgeving. Deze plugin is enkel verkrijgbaar in één van de Sencha-bundels.

# Hoofdstuk 4

## Hybride applicaties

Hybride applicaties proberen de volledige functionaliteit van native applicaties aan te bieden met de voordelen van platformonafhankelijke code. Deze applicaties worden ontwikkeld met technologieën en programmeertalen waar niet allemaal een SDK voor bestaat.

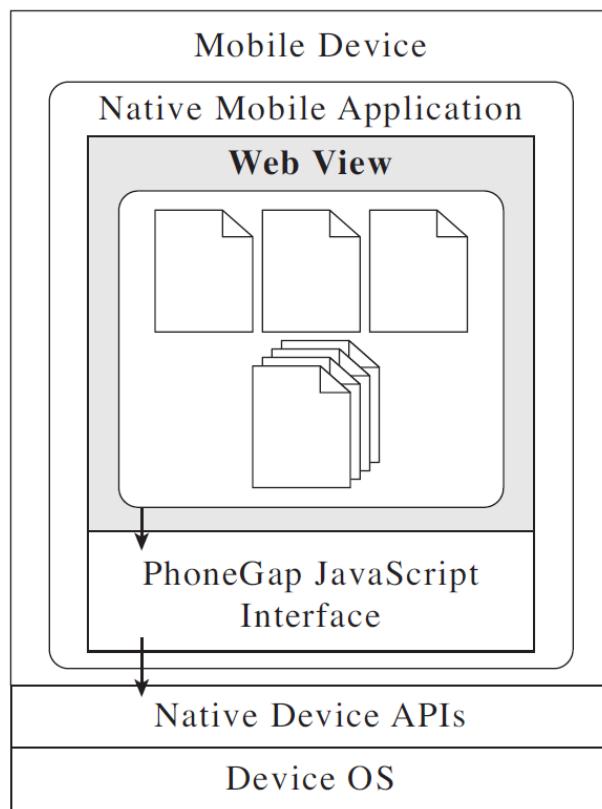
Het simpelste voorbeeld van een hybride applicatie is een applicatie waarbij delen gewoon draaien in een native web-component. Zo is het mogelijk om een volledige webapplicatie als het ware te wrappen in een native applicatie. Wanneer er de noodzaak is om de native API's aan te spreken kan bijvoorbeeld gebruik gemaakt worden van het PhoneGap-framework dat een brug voorziet tussen JavaScript en de native API's. Het is natuurlijk ook mogelijk om deze brug zelf te voorzien.

Naast de aanpak om de applicatie te draaien binnen de runtime van de browser zijn er nog andere frameworks beschikbaar die een eigen runtime voorzien. Voorbeelden hiervan zijn Xamarin en Rhomobile. Met Xamarin is het bijvoorbeeld mogelijk om delen van de applicatie te schrijven in C# en uit te voeren op de verschillende platformen dankzij de Mono Runtime die bij de applicatie wordt gebundeld. In volgende paragrafen worden enkele frameworks voor hybride applicaties besproken.

## 4.1 PhoneGap

PhoneGap ([www.phonegap.com](http://www.phonegap.com)) is een opensource framework om cross-platform hybride mobiele applicaties te ontwikkelen met HTML, CSS en JavaScript. PhoneGap volgt hierbij de W3C-webstandaarden. IBM en Adobe ondersteunen het Apache Cordova project waarvan PhoneGap de commerciële naam is. PhoneGap heeft een actieve gemeenschap van ontwikkelaars en er zijn ook verschillende plugins beschikbaar voor extra functionaliteit. Dit framework kan niet gebruikt worden om de UI mee op te bouwen maar voorziet enkel een brug in JavaScript naar de native API's.

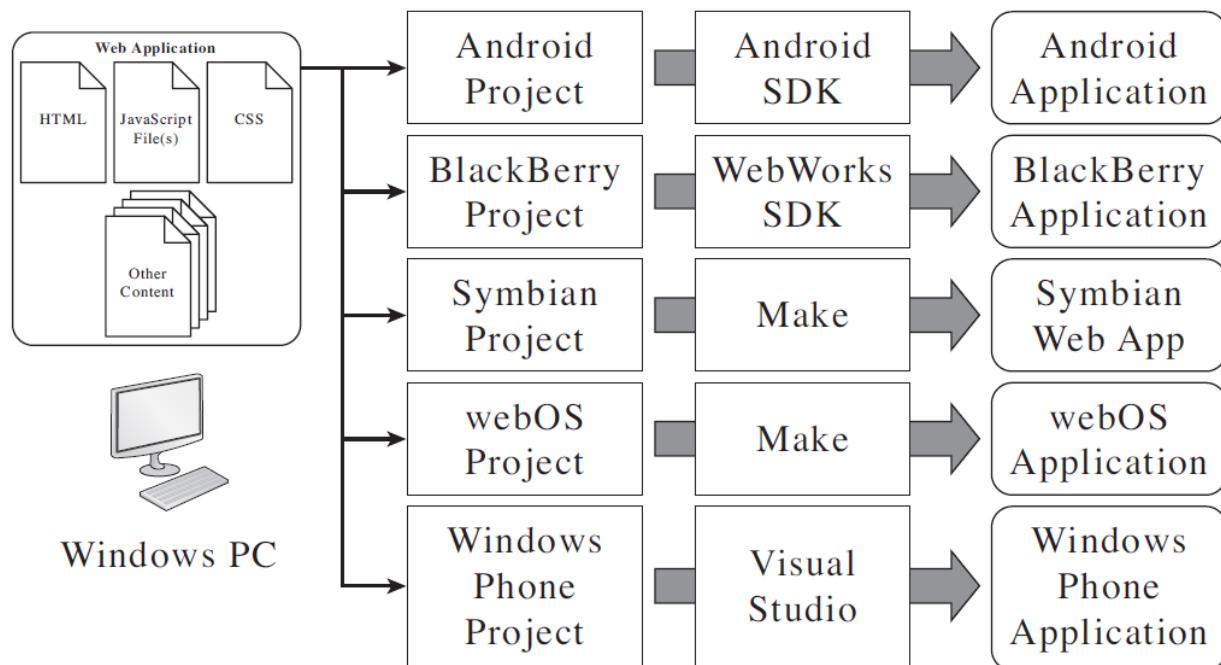
Via het navigator-object worden de PhoneGap API's aangeboden om zo de native API's te kunnen aanspreken met JavaScript. De webapplicatie, geschreven in HTML, CSS en JavaScript, wordt dan weergegeven in een webview. Figuur 4.1 geeft een beeld van de architectuur van een PhoneGap-applicatie.



**Figuur 4.1:** De architectuur van een PhoneGap-applicatie (Wargo, 2012)

Belangrijk is wel dat deze hybride applicaties nog steeds voor elk platform gecompileerd moeten worden en dat de juiste bruggen naar de native API's ingepast moeten zijn. Dit wil zeggen dat voor elk platform de SDK geïnstalleerd moet zijn. Vervolgens moet voor elke platform met zijn specifieke SDK een project aangemaakt worden. In dit project moet

de webapplicatie dan ingepast worden met een webview. Tenslotte wordt de applicatie gecompileerd en kan deze gedistribueerd worden voor het betreffende platform. Figuur 4.2 geeft een overzicht van dit proces op een Windows PC.



**Figuur 4.2:** Een PhoneGap-applicatie voor meerdere platformen compileren (Wargo, 2012)

Zoals de aandachtige lezer misschien al heeft opgemerkt, staat er op figuur 4.2 geen verwijzing naar iOS. Dit komt omdat het niet mogelijk is om iOS-applicaties te compileren op een Windows PC. Daarnaast is het ook duidelijk dat er initieel heel wat SDK's geïnstalleerd moeten worden en dat de applicatie compileren voor elk platform heel wat werk en tijd in beslag kan nemen. Gelukkig biedt PhoneGap hiervoor een oplossing. Deze oplossing heet PhoneGap Build (<https://build.phonegap.com>) en is een online service waar ontwikkelaars hun webapplicatie kunnen uploaden naar servers van PhoneGap die deze applicatie dan voor elk ondersteund platform compileren.

De hybride aanpak van PhoneGap is niet de enige op de markt. Hoewel PhoneGap deze trend gestart heeft, zijn er nu verschillende gelijkaardige frameworks op de markt. Interessant om te vermelden is Titanium van Appcelerator ([www.appcelerator.com](http://www.appcelerator.com)). Een Titanium-applicatie gebruikt een native applicatie als container om JavaScript uit te voeren. De applicatie zelf wordt volledig in JavaScript geschreven, zowel de interface als de logica. Via JavaScript worden dan ook native UI-elementen aangemaakt. Hierbij kunnen wel problemen opduiken qua platformonafhankelijkheid omwille van de platformspecifieke UI-paradigma's. Hierdoor is het mogelijk dat niet alle code over de platformen heen gedeeld kan worden. (Wargo, 2012)

## 4.2 Xamarin

Xamarin ([www.xamarin.com](http://www.xamarin.com)), vroeger bekend als Mono, is een softwareplatform dat het voor ontwikkelaars mogelijk maakt om gemakkelijk cross-platform applicaties te maken. Het is een open-source implementatie van Microsoft's .NET-framework gebaseerd op de ECMA (European Computer Manufacturers Association) voor C# en de Common Language Runtime.

Tijdens het onderzoek is de naam veranderd van Mono naar Xamarin. Vroeger was er hierbij ook MonoTouch en Mono for Android, dit is nu respectievelijk Xamarin.iOS en Xamarin.Android. Dit zijn commerciële SDK's die implementaties zijn van Mono voor deze mobiele besturingssystemen. Beide SDK's zijn ook goed gedocumenteerd met tutorials en een volledige API reference.

Deze SDK's maken de native API's voor iOS en Android aanspreekbaar met C#. Zo wordt het mogelijk om gemiddeld 75% van de code te delen tussen deze platformen. Hoewel de UI ook volledig in C# geschreven kan worden is het niet mogelijk om die allemaal te delen over de verschillende platformen omdat van de verschillende paradigma's. Daarom is het belangrijk om voor een goede scheiding tussen logica en UI te zorgen (zie figuur 4.3). Windows Phone apps kunnen ook native in C# geschreven worden dus ook hier kan code hergebruikt worden. Xamarin houdt deze API's goed up-to-date en soms zijn er zelfs verbeteringen geïmplementeerd bovenop de native API's. De compiler van Xamarin zorgt voor de juiste output voor elk platform. Voor iOS doet deze een volledige AOT-compilatie (Ahead-of-Time) in de Mono runtime om een ARM binair uitvoerbaar bestand te maken dat klaar is voor de iOS app store. Voor Android wordt de Mono runtime bij de applicatie gebundeld die zorgt voor een JIT-compilatie (Just-in-Time) op het Android toestel zelf. Hierdoor is een native UI en performance mogelijk met een verwaarloosbare opstarttijd. De compilatie voor iOS vereist wel een Mac.



**Figuur 4.3:** Scheiding UI en gemeenschappelijke code ([www.xamarin.com](http://www.xamarin.com))

Het is mogelijk om alle bestaande .NET-bibliotheken te importeren en te gebruiken in een mobiele applicatie. Bij Xamarin.iOS is het ook mogelijk om gebruik te maken bestaande C- en Objective-C-bibliotheken net zoals het bij Xamarin.Android mogelijk is om gebruik te maken van bestaande Java-bibliotheken. Xamarin.iOS en Xamarin.Android worden volledig ondersteund in de Xamarin Studio IDE. Er is ook ondersteuning voor Xamarin in Xcode en in Visual Studio. De huidige versie op het moment van schrijven is Xamarin 2.0.

## 4.3 Marmalade

Marmalade is een vrij compleet cross-platform framework, maar ook heel complex. Het framework voorziet SDK's voor ontwikkeling in C++, HTML5-technologieën en Obj-C.

De focus ligt echter eerder op de ontwikkeling van games. Hierbij worden verschillende manieren aangeboden om grafische output op het scherm weer te geven. Er kan bijvoorbeeld gebruik gemaakt worden van de 'OpenGL ES'-bibliotheek waarvoor aan de hand van de *lwGL* API in een wrapper voorzien wordt. Er is ook een API beschikbaar die op een nog lager niveau werkt waarbij men rechtstreeks pixels kan besturen. Daarnaast is er ook een API voor de weergaven van 2D grafische componenten, de *lw2D* API genoemd. Tot slot is er *lwGx* API die de meeste functionaliteit aanbiedt.

Een game heeft meestal ook wel bepaalde UI-elementen nodig zoals bijvoorbeeld een simpele knop. Hiervoor zijn twee API's beschikbaar. De eerste is de *lwUI* API waar ontwikkelaars interfaces mee kunnen opbouwen die bestaan uit knoppen, labels en dergelijke. Deze API bevat veel functionaliteit en biedt dus ook de mogelijkheid om interfaces te bouwen voor andere applicaties dan games. De tweede API is *lwNUI*, waarbij de 'N' staat voor native. Met deze API is het mogelijk om interfaces te ontwikkelen die gebruik maken van de standaard UI-controls van het mobiele platform waar de applicatie op draait. Dit wordt momenteel echter niet voor alle platformen ondersteund. Platformen die niet ondersteund worden maken dan gebruik van de default stijl met de *lwUI* API. (Scapplehorn, 2012)

Met Marmalade is het mogelijk om een applicatie met één compilatie beschikbaar te maken voor alle ondersteunde platformen. Dit is mogelijk door de *single-binary* architectuur die door Marmalade gepatenteerd is. Het proces van ontwikkeling, testen en deployen is volledig onafhankelijk van de SDK van het OS. Dit wil zeggen dat het niet nodig is om de code voor de verschillende platformen te compileren. Het is dus ook niet nodig om van IDE's te wisselen of van Windows naar Mac bijvoorbeeld.

Vroeger was er voor het ontwikkelen van dit soort interfaces de Marmalade Studio UI Builder. Tijdens het onderzoek bleek deze tool niet meer beschikbaar, dit wordt ook vermeld in het boek van Scapplehorn. Op het moment van schrijven blijkt er echter terug een tool beschikbaar, Marmalade Quick 1.1. Meer informatie over het Marmalade-framework is te vinden op [www.madewithmarmalade.com](http://www.madewithmarmalade.com).

## 4.4 Rhomobile

Rhomobile ([www.rhomobile.com](http://www.rhomobile.com)) is een commercieel ondersteund open source framework voor cross-platform ontwikkeling en ondersteunt de belangrijkste platformen. Rhomobile is vooral gericht op bedrijfsapplicaties. Deze applicaties worden geschreven in Ruby. Het is echter ook mogelijk om de UI met behulp van HTML, CSS en Javascript te ontwikkelen of om gebruik te maken van frameworks zoals Sencha Touch of jQuery Mobile. Applicaties worden gecompileerd tot Ruby 1.9 bytecode. Om deze code uit te voeren wordt dan een Ruby virtuele machine voorzien. (Allen, 2010)

De Ruby virtuele machine voor Android is geschreven in C++ met de Android NDK<sup>1</sup>. De RubyVM wordt normaal gebundeld bij de applicatie. Het is echter ook mogelijk om een applicatie te ontwikkelen in 'shared mode'. Hierdoor wordt de VM niet bij de applicatie gebundeld maar geïnstalleerd op een voorgedefinieerde locatie op het toestel zodat die voor elke Rhomobile applicatie beschikbaar is. Rhomobile voorziet ook API's om onder andere barcodes te scannen, om handtekeningen te plaatsen en voor dataencryptie. De device API's worden beschikbaar gesteld voor Ruby, JavaScript en via HTML-meta-tags.

## 4.5 Conclusie

Voor de ontwikkeling van hybride applicaties bestaan er verschillende frameworks. De hybride aanpak tussen deze frameworks kan heel verschillend zijn. Zo kunnen applicaties bijvoorbeeld volgens een compleet andere architectuur opgebouwd zijn. Er zijn meestal ook verschillen in het aantal ondersteunde platformen. Het ene framework biedt al meer ondersteuning dan het andere en zal waarschijnlijk ook sneller ondersteuning bieden voor nieuwe platformen. Meestal gaat dit wel gepaard met een mindere performantie. Dit verschilt ook van framework tot framework.

Naast het verschil in applicatiearchitectuur is er ook een belangrijk onderscheid op UI-gebied. Sommige frameworks gaan gebruik maken van native UI-elementen. Soms worden deze UI's '*on the fly*' gevormd. Elk platform heeft qua gebruikersinterface zijn eigen model of zienswijze. Zo kunnen bepaalde componenten op verschillende platformen heel anders weergegeven worden of gewoonweg niet bestaan. Hierdoor kan het noodzakelijk zijn dat de UI herschreven wordt voor een bepaald platform.

Xamarin werd in de onderzoeksfase als eerste uitgeprobeerd. Hiervoor werd eerst bekijken hoe native Android applicaties ontwikkeld worden. Zo kon ontwikkeling met Xamarin hier ook mee vergeleken worden. Om de applicatie voor iOS te kunnen compileren is een Mac nodig. Aangezien er geen Mac ter beschikking was werd ook besloten hier niet mee verder te werken. Daarnaast ondersteund dit framework ook minder platformen.

PhoneGap werd tijdens deze masterproef enkel gebruikt als *wrapper* voor webapplicaties.

---

<sup>1</sup>Native Development Kit

De PhoneGap API's werden niet uitgeprobeerd aangezien er eerst gepoogd wordt zoveel mogelijk functionaliteit toe te voegen door te steunen op de HTML5-technologieën. Marmalade werd ook uitgetest maar al snel weer aan de kant geschoven. Dit framework is vooral gericht op games en is bijgevolg te complex voor het doel van deze masterproef. Rhomobile werd niet geprobeerd omdat ook hierbij geen compilatie voor iOS gedaan kan worden zonder een Mac.

Bij de keuze van een hybride framework moet duidelijk zijn wat het doel en de eisen zijn. Enkel op basis hiervan kan er een goede beslissing genomen worden. Hierbij moet het belang die gegeven wordt aan performantie en platformonafhankelijkheid tegen elkaar afgewogen worden.

# **Hoofdstuk 5**

## **Conclusie onderzoek**

### **5.1 Algemeen**

In hoofdstuk 2 werd aandacht besteed aan verschillende mobiele besturingssystemen en hun applicaties. Sommige zijn gevestigde waarden, andere zijn nieuwe spelers die hun marktaandeel proberen te vergroten. Nog andere zijn besturingssystemen die in de nabije toekomst op de markt zullen verschijnen. In 2013 kan gesteld worden dat Android en iOS de gevestigde waarden zijn. Windows Phone en BlackBerry zijn de uitdagers. In de verdere loop van het jaar zullen enkele nieuwe besturingssystemen hun intrede doen op de markt en zal het aanbod aan mobiele besturingssystemen verdubbelen. Deze nieuwe besturingssystemen worden met groot enthousiasme verwelkomd door hardwarefabrikanten en telecomproviders omwille van de huidige hegemonie van Android en iOS. Ook voor de consumenten kan dit positief uitdraaien omwille van de concurrentie. Concurrentie zal leiden tot scherpere prijzen en snellere vernieuwing. Er is echter ook een keerzijde aan de medaille. Het grote aanbod zal de keuze voor een toestel met een mobiel besturingssysteem moeilijker maken. Applicaties spelen in deze keuze zeker een hoofdrol.

Gevestigde waarden zoals Android en iOS hebben reeds een uitgebreid aanbod aan apps. De nieuwste populairste apps worden ook meestal gelanceerd voor deze besturingssystemen. Windows Phone en BlackBerry proberen hun aanbod aan apps te vergroten met behulp van cross-platforme oplossingen. Microsoft zorgde voor een HTML5 SDK waarmee apps ontwikkeld kunnen worden die zowel op Windows Phone 8 als op Windows 8 zullen werken. BlackBerry zorgt voor een Android runtime, waardoor mits een kleine aanpassing Android apps kunnen draaien in het BlackBerry besturingssysteem. Daarnaast biedt BlackBerry ook een zeer goede ondersteuning voor HTML5. Ook de nieuwe spelers zullen dergelijke strategie voeren. Ofwel wordt een Android virtual machine voorzien, wat dan beschouwd zou kunnen worden als een soort mobiele variant op de Java virtual machine, ofwel gaat het besturingssysteem diepe ondersteuning bieden voor web apps. Vooral de ondersteuning voor webtoepassingen lijkt een populaire keuze te zijn. Besturingssystemen die een goede integratie bieden voor platformonafhankelijke apps zullen in de strijd om het grootste en beste aanbod aan apps één front kunnen vormen. Bovendien wordt de hardware aan een

hoog tempo beter en zal platformonafhankelijkheid alleen maar belangrijker worden.

Hoofdstuk 3 handelde over webapplicaties. De komst van HTML5 biedt ontwikkelaars meer mogelijkheden om web apps te maken die rijk zijn aan functionaliteit en een goede gebruikerservaring bieden. Apple richtte de WHATWG mee op en biedt op iOS een goede ondersteuning en integratie van web apps. Voor de ontwikkeling van web apps zijn er ook verschillende frameworks vorhanden. jQuery Mobile bouwt de interface op met behulp van enkele HTML-attributen en bijhorende CSS. Hierbij wordt ook een JavaScript-bestand geleverd dat meer zorgt voor de interactiviteit en transities van pagina's.

Sencha Touch is een framework dat een pak uitgebreider is en ook heel wat meer functionaliteit biedt. Applicaties worden geschreven in JavaScript en worden opgebouwd volgens het MVC design-pattern. Bovendien biedt het framework een eigen klassensysteem. Daarnaast gaven testen met applicaties van beide frameworks het gevoel dat applicaties van Sencha Touch toch een iets betere performantie en gebruikerservaring bieden. Visual Studio 2012 en Netbeans 7.3 bleken goede ontwikkelingstools voor HTML5-webapplicatie. Daarnaast biedt het Sencha-framework ook zijn eigen ontwikkelingstools aan. De visuele interface van Sencha Architect biedt hierbij een grote meerwaarde.

Webapplicaties draaien in een browser en zijn daardoor platformonafhankelijk. Browsers kunnen echter ook anders geïmplementeerd zijn, daardoor is er wel een bepaald mate van browserafhankelijkheid. Hoewel de HTML5-standaard hier al veel verbetering in gebracht heeft ondersteunen de verschillende browsers deze standaard nog niet volledig. Sencha ondersteunt momenteel alle WebKit-browsers (zoals Safari, Chrome, BlackBerry, ...) en Internet Explorer 10. Voor Firefox wordt aan ondersteuning gewerkt. Sencha werkt hierbij samen met de browserfabrikanten. Hoewel er dus naar de standaard toe gewerkt wordt, moet de ontwikkelaar browserafhankelijkheid in gedachten houden. Ook al vangt Sencha en HTML5 bepaalde browserafhankelijkheden op, toch blijft het aangewezen hier rekening mee te blijven houden.

In hoofdstuk 4 werden hybride applicaties besproken. Verschillende aanpakken zijn mogelijk om hybride applicaties te bouwen. Een webapplicatie kan weergegeven worden met een native webview en men kan een brug voorzien om via JavaScript de native API's aan te spreken. Hier bestaan verschillende frameworks voor. Een andere mogelijkheid is een eigen runtime voorzien voor de applicatie. Daarnaast kan ook het onderscheid gemaakt worden tussen hybride applicaties die gebruik maken van native UI-componenten en deze die dit niet doen, zoals bijvoorbeeld een gewrapte webapplicatie.

Hybride oplossingen die gebruik maken van native UI-componenten hebben het probleem dat elk besturingssysteem zijn eigen paradigma's heeft betreffend de grafische interface. Hierdoor is het niet opportuun of zelf niet mogelijk om 100% van alle code te hergebruiken voor de verschillende platformen. Xamarin bijvoorbeeld geeft aan dat met zijn framework gemiddeld 75% van de code hergebruikt kan worden tussen de ondersteunde platformen. Uit getuigenissen op de site blijkt dat dit kan gaan van 60% tot 95%.

Grafische interfaces van webapplicaties schalen zich ook makkelijker dan interfaces die opgebouwd zijn met native UI componenten. Een mooi voorbeeld hiervan zijn de iPhone

applicaties die werden weergegeven met zwarte balken boven en onder de applicatie op de iPhone 5 die een groter scherm kreeg. Op de iPad dan weer worden iPhone applicaties weergegeven binnen een digitale iPhone die wordt weergegeven op het scherm.

Het internationale onderzoeks- en adviesbureau (voor technologie) Gartner zegt dat tegen 2016 meer dan 50% van de applicaties hybride applicaties zullen zijn. (Gartner, 2013)

*"Our advice would be to assume the enterprise will have to manage a large and diverse set of mobile applications that will span all major architectures. Enterprises should consider how applications can be enriched or improved by the addition of native device capabilities and evaluate development frameworks that offer the ability to develop native, hybrid and Web applications using the same code base. Where possible, development activities should be consolidated via cross-platform frameworks."* - Van Baker, research vice president Gartner.  
(Gartner, 2013)

## 5.2 Keuze van de gebruikte technologieën

Naast het onderzoek was de opdracht de ontwikkeling van een applicatie. Er werd gekozen om hiervoor een webapplicatie te ontwikkelen. De webapplicatie zou op elk platform moeten werken in een browser. Bovendien biedt iOS een goede ondersteuning en integratie van deze webapplicaties en zullen nieuwe besturingssystemen zoals Ubuntu Mobile en Firefox OS hierin volgen. Een webapplicatie zal zich ook eenvoudig kunnen schalen naar verschillende schermformaten. Daarnaast kan met de HTML5-technologieën al heel wat functionaliteit aangeboden worden. Een ander voordeel bij deze aanpak is dat een webapplicatie eenvoudig omgebouwd kan worden tot een hybride applicatie. Op deze manier kan de applicatie dan gedistribueerd worden via de *app stores* en kan extra functionaliteit toegevoegd worden als de HTML5-technologieën nog niet voldoen.

Voor de ontwikkeling van de webapplicatie werd gekozen om gebruik te maken van het Sencha Touch framework. Ondanks de steilere leercurve biedt dit framework vele voordelen. Zo biedt het framework de mogelijkheid om grafieken weer te geven en moet hiervoor niet meer op een andere bibliotheek (zoals HighChart.js bijvoorbeeld) beroep gedaan worden. Datakoppeling zit onder andere ook in het framework, hiervoor zou men anders bijvoorbeeld beroep kunnen doen op het Knockout.js-framework. Daarnaast biedt het framework ook een aantal handige tools.

Na een testperiode met het Sencha Touch framework en de ontwikkelingstools van Sencha werd besloten om de Sencha Touch bundel aan te kopen. Deze bundel bevat een licentie voor de Sencha Architect ontwikkelomgeving, de Sencha Eclipse plugin, de commerciële versie van het Sencha Touch framework, de Sencha Touch grafieken zonder markering, de Sencha Mobile Packaging voor het bouwen van hybride applicaties en een Sencha Support pakket. Meer info op <http://www.sencha.com/products/touch-bundle/>.

# Hoofdstuk 6

## Aanvangen met Sencha Touch

Het Sencha Touch framework is geschreven in JavaScript. Kennis van deze scripttaal is dus zeker een voordeel. Eerder werd al vermeld dat dit framework een steilere leercurve heeft dan jQuery Mobile. Dit komt omdat dit framework echt heel uitgebreid is. In dit hoofdstuk zullen dan ook maar enkele aspecten aan bod komen. Meer informatie is natuurlijk altijd terug te vinden in de documentatie van Sencha Touch of de DZone Refcard over Sencha Touch (gratis download via <http://refcardz.dzone.com/refcardz/sencha-touch>) is zeker ook aangewezen om kennis te maken met het framework.

Ook de Sencha Architect ontwikkelomgeving bleek een goede manier om kennis te maken met het framework. Alle componenten zijn aanwezig in een toolbox, eigenschappen zijn terug te vinden in het configuratievenster, er wordt gelinkt naar de relevante documentatie en Architect maakt zelf gebruik van de Sencha Cmd tool. Daarom is het zeker aangewezen om gebruik te maken van de gratis trial-versie om wat vertrouwd te geraken met het framework.

### 6.1 De installatie

Voor de installatie van de SDK kan de gids gebruikt worden die terug te vinden is in de documentatie: *Getting Started with Sencha Touch*. Er doken echter problemen op bij het builden van een applicatie. Dit bleek te liggen aan de incompatibiliteit tussen Compass en de nieuwste versie van Ruby. Compass is een tool die geschreven is in Ruby.

Deze tool wordt gebruikt om het SASS-bestand te compileren naar CSS3. SASS (Syntactically Awesome StyleSheet) is een uitbreiding op CSS3 en maakt onder andere overerving mogelijk. De installatie van Compass en Ruby gebeurt automatisch bij de installatie van Sencha Cmd. Om het probleem te verhelpen moest Ruby 2.0.0 verwijderd worden en Ruby 1.9.3 geïnstalleerd worden. Via <http://rubyinstaller.org> kan dit eenvoudig gedaan worden op Windows. Vervolgens wordt Compass geïnstalleerd met het commando **gem install compass**. Ondertussen is dit probleem al aangepakt door Sencha en wordt Ruby 1.9.3 geleverd bij de installatie van Sencha Cmd.

## 6.2 De MVC-architectuur

Sencha Touch voorziet een applicatiearchitectuur volgens de principes van het MVC-design-pattern. Deze aanpak zorgt er voor dat de code overzichtelijk gehouden kan worden en ook goed onderhoudbaar is.

Figuur 6.1 toont de volledige mapstructuur van de applicatie. De belangrijkste map is de *app*-map waar de verschillende mappen terug te vinden zijn voor de models (M), de views (V) en de controllers(C). Daarnaast is er ook nog de *store*-map waar alle *store*-klassen in terecht komen. Deze *stores* worden gebruikt voor de datakoppeling. Een vijfde map, *profile* genaamd, dient voor de opslag van profielklassen. Deze profielen bepalen welke delen van de applicatie geladen moeten worden afhankelijk van het profiel dat actief is. Profielen komen later nog aan bod.

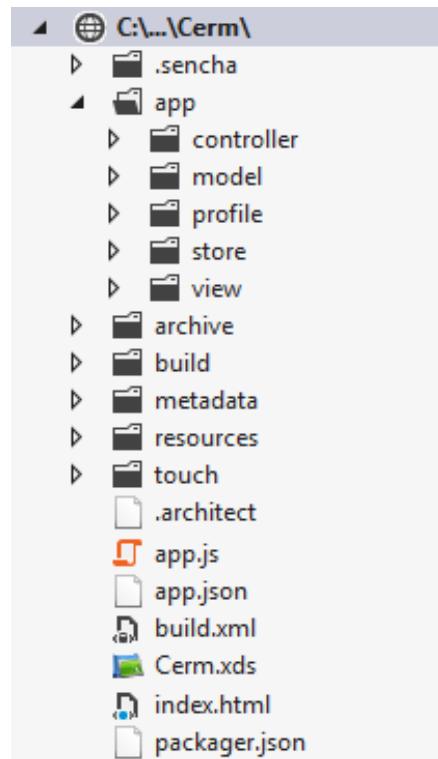
In *touch* zitten alle bestanden van het framework. Resources bevat zaken zoals CSS, afbeelding, ...

In *build* komen de build-versies van de applicatie terecht en de andere mappen zijn vooral voor metadata en configuratie. De belangrijkste bestanden zichtbaar in figuur 6.1 zijn app.js, app.json en index.html. In app.js wordt de applicatie beschreven; hierbij wordt aangegeven wat de afhankelijkheden zijn. Zo wordt bijvoorbeeld aangegeven wat de controllers, views en models zijn van de applicatie. Ook de start-iconen en afbeeldingen worden hier onder andere aangegeven.

In app.json worden verschillende paden aangegeven, zoals bijvoorbeeld waar de map met CSS zich bevindt. Ook andere configuraties zoals de naam of de configuraties voor de appcache worden hierin beschreven. Het index.html-bestand wordt gebruikt om de applicatie op te starten. Dit document bevat enkel een CSS-animatie die getoond wordt bij het opstarten en een JavaScript waarmee de applicatie geladen wordt. Verder wordt alle HTML door de applicatie geïnjecteerd in de DOM-structuur van dit document. Tot slot zijn er nog enkele bestanden terug te vinden die gebruikt worden door Sencha Architect en Sencha Cmd.

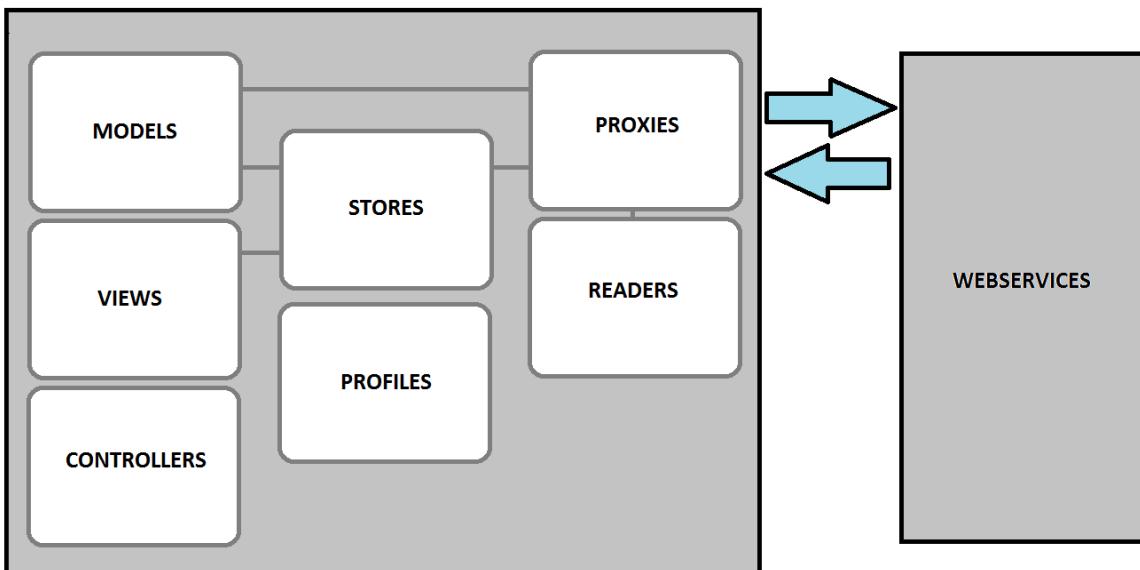
Meer informatie over hoe applicaties gebouwd met Sencha Touch in elkaar zitten is te lezen in volgende gidsen uit de documentatie:

- *Intro to Applications with Sencha Touch*  
[http://docs.sencha.com/touch/2.2.0/#!/guide/apps\\_intro](http://docs.sencha.com/touch/2.2.0/#!/guide/apps_intro)
- *Managing Dependencies with MVC*  
[http://docs.sencha.com/touch/2.2.0/#!/guide/mvc\\_dependencies](http://docs.sencha.com/touch/2.2.0/#!/guide/mvc_dependencies)



**Figuur 6.1:** De mapstructuur van de applicatie.

Figuur 6.2 geeft een beeld van hoe de applicatie wordt opgebouwd. De MVC-structuur is duidelijk aanwezig. De *stores* spelen een belangrijke rol bij de datakoppeling. Aan een *store* wordt een model gekoppeld. Deze *store* wordt dan gekoppeld aan een *view*. Zo zal een *ListView* de *records* weergeven die zich in de store bevinden door middel van de geconfigureerde *template* van de *ListView*. Wanneer de gegevens van een externe bron (zoals bijvoorbeeld van een webservice) moet komen, kan gebruik gemaakt worden van een proxy. Deze proxy kan gekoppeld worden aan een model of aan een *store*. De proxy zal communiceren met de webservice en met behulp van een *reader* worden objecten van de modelklasse gebouwd op basis van de verkregen data.



**Figuur 6.2:** De algemene architectuur van de applicatie.

### 6.3 Het klassensysteem

De hoeksteen van het Sencha Touch framework is het klassensysteem. Dankzij dit klassensysteem is het mogelijk om gebruik te maken van overerving. In puur JavaScript is dit niet mogelijk. Het is wel mogelijk om een prototype te definiëren en hier dan vervolgens een object mee te maken. Dat object kan dan verder uitgebreid worden. Het is echter niet mogelijk om een prototype te definiëren dat overerft van een andere prototype. Het klassensysteem maakt het dus mogelijk om toch gemeenschappelijke code samen te houden.

Dit klassensysteem wordt gedemonstreerd aan de hand van een voorbeeld uit de applicatie. Enkel de dummyfunctie werd hiervoor aan deze klasse toegevoegd. Codefragment 6.1 toont hoe een klasse wordt gedefinieerd met behulp van de methode **Ext.define**. Dit is de modelklasse voor een machine. Deze machine is eigenlijk een drukpers uit de drukkerij en

heeft een bepaalde naam (of niet), een referentie en een aanduiding 'active'. Omdat dit een modelklasse is wordt deze afgeleid van de klasse `Ext.data.Model`. De `Machine`-klasse behoort ook tot de namespace `Cerm.model`. Via het 'config'-object wordt gedefinieerd welke eigenschappen de klasse heeft. Het framework zorgt automatisch voor getters en setters. Zo is bijvoorbeeld automatisch de functie `isActive` beschikbaar. De `mapping`-variabelen dienen voor de omzetting van een JSON-object naar het modelobject. Het is natuurlijk ook mogelijk om functies voor deze klasse te schrijven.

```
Ext.define('Cerm.model.Machine', {
    extend: 'Ext.data.Model',

    config: {
        fields: [
            {
                name: 'active',
                mapping: 'Active'
            },
            {
                name: 'name',
                mapping: 'Name'
            },
            {
                name: 'reference',
                mapping: 'Reference'
            }
        ]
    },
    dummyFunction: function(){}
});
```

**Codefragment 6.1:** Definieren van de modelklasse Machine

Code 6.2 zorgt voor de creatie van een `Machine`-object. De functie `Ext.create` vraagt twee argumenten: de naam van de klasse en een optioneel config-object waarmee de eigenschappen worden ingevuld.

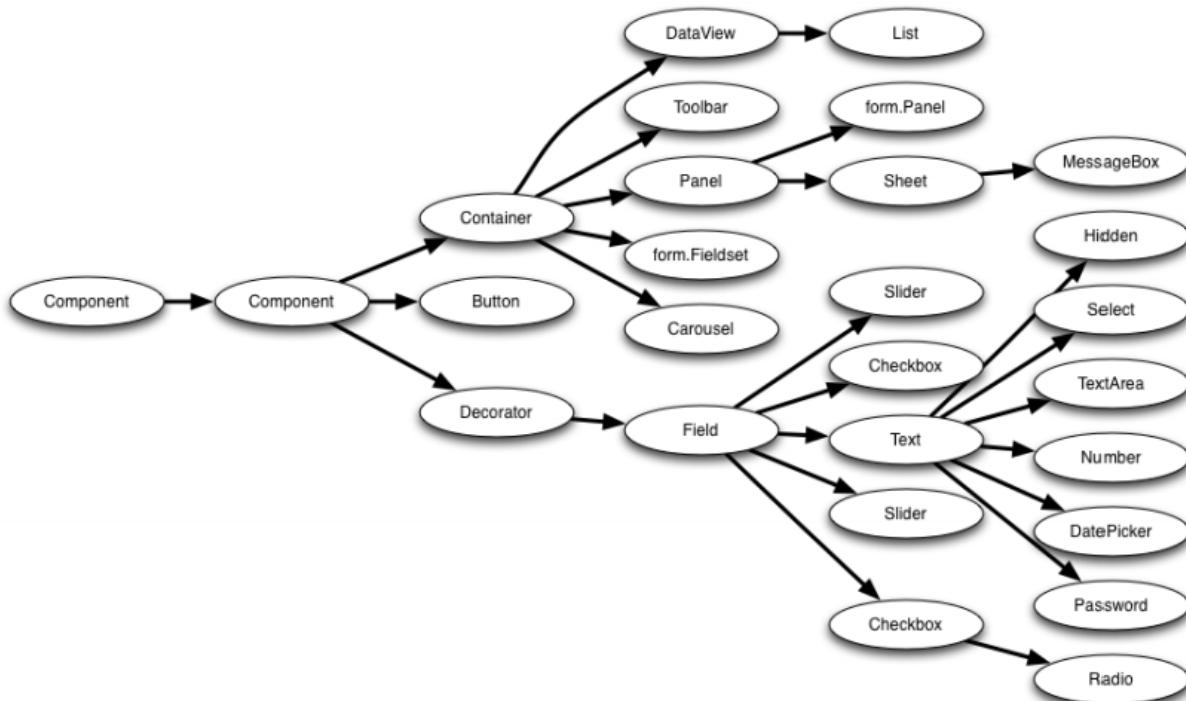
```
var machine1 = Ext.create('CermMobile.model.Machine', {
    active: 1,
    name : 'voorbeeldMachine',
    reference: 'ref00001'
});
```

**Codefragment 6.2:** Instantiëren van een Machine-object

## 6.4 De componenten en het lay-out-systeem

Sencha voorziet heel wat standaardcomponenten voor het opbouwen van de grafische interface. Dit componentensysteem maakt gebruik van het klassensysteem, zie figuur 6.3. Elke klasse wordt afgeleid van de *Component*-klasse. Componenten kunnen gedefinieerd worden met een *xtype*. Dit is een verkorte benaming voor de klasse en laat *lazy instantiation* toe. Zo wordt de component pas gecreëerd als hij echt nodig is.

Meerdere componenten kunnen toegevoegd worden aan een container. Het framework voorziet hierbij een lay-out-systeem, zo kan de container met een bepaalde lay-out geconfigureerd worden. Mogelijkheden zijn *vbox*, *hbox*, *card* of *fit*. De *card*-lay-out plaatst de componenten bovenop elkaar. *Vbox* en *hbox* plaatsen de componenten respectievelijk verticaal of horizontaal ten opzicht van elkaar. De *fit*-lay-out zorgt er voor dat de eerste component de volledige ruimte van de container zal innemen. (Bershadskiy, 2013)



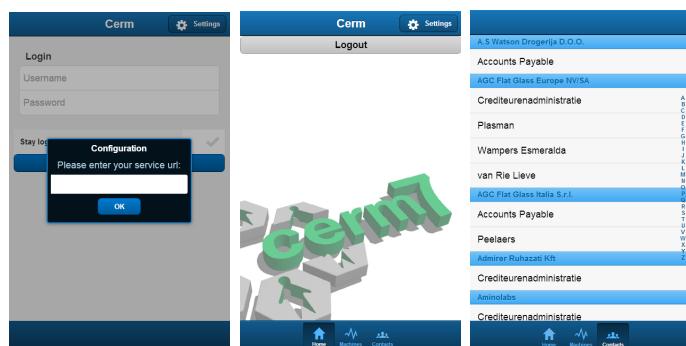
**Figuur 6.3:** Het componentensysteem waarin alle componentklassen afgeleid worden van de hoofdklasse *Component*. Een pijl van A naar B wil zeggen dat B afgeleid is van A. (Bershadskiy, 2013).

# Hoofdstuk 7

## De front-end ontwikkeling van de applicatie

Dit hoofdstuk beschrijft de belangrijkste lijnen van de ontwikkeling van de applicatie. Een eerste vereiste was dat de applicatie modular opgebouwd werd. Dankzij het klassensysteem en de MVC-structuur kan dit eenvoudig gerealiseerd worden. Om deze modules beschikbaar te maken via de applicatie werd gekozen voor een **tabpanel (Ext.tab.Panel)**. Elke module krijgt een tab. Het is mogelijk om meer tabs toe te voegen dan zichtbaar kunnen zijn op het scherm omdat er door de bar met tabs gescroold kan worden.

De mobiele toepassing zal ook moeten communiceren met webservices. Daarom wordt de url waar deze services gevonden kunnen worden gevraagd wanneer de applicatie voor het eerst opgestart wordt of wanneer er nog geen url is opgegeven. Deze url wordt opgeslagen in de *localStorage*. Deze webservices, de configuratie van de server en dergelijke worden besproken in hoofdstuk 8. Op figuur 7.1 is links te zien hoe deze url aan de gebruiker gevraagd wordt met een prompt. In het midden zijn onderaan de verschillende tabs van de modules te zien. Op de rechtse figuur is als voorbeeld de contactenmodule geselecteerd. De tabs die getoond worden, dus de modules die beschikbaar gesteld worden, zijn afhankelijk van de rechten die de ingelogde gebruiker heeft.



**Figuur 7.1:** De algemene interface van de applicatie.

## 7.1 De machinesmodule

De eerste module is de machinesmodule. Deze moet het mogelijk maken om de productie op te volgen. Hiervoor zijn er tellers geïnstalleerd op de drukpersen die de activiteit registreren. Deze registraties komen terecht in het Cermssysteem en kunnen opgevraagd worden via de webservice.

### 7.1.1 De lijst van machines

Als eerste wordt een *listview* (*Ext.dataview.List*) gedefinieerd die een lijst van drukpersen zal weergeven. Deze klasse wordt *Cerm.view.MachineList*, of kortweg *MachineList*, genoemd. Aan deze listview wordt een *store* gekoppeld die de objecten van de drukpersen zal bevatten. Dit zijn objecten van de klasse *Machine* zoals in 6.3 beschreven. De *store* wordt ook zo geconfigureerd dat de records gesorteerd worden op de 'active'-eigenschap, en vervolgens op de 'name'-eigenschap. De lijst wordt zo alfabetisch weergegeven met de actieve personen bovenaan.

Met een template wordt bepaald hoe elk record in de lijst weergegeven moet worden. Deze template wordt geconfigureerd met de 'itemTpl'-eigenschap en kan eenvoudig geschreven worden met HTML en eventueel wat inline CSS. Omdat hier echter een meer dynamische oplossing nodig is wordt gebruik gemaakt van een *XTemplate*. Op deze manier kunnen actieve personen met een groene cirkel aangeduid worden terwijl niet actieve personen een rode cirkel krijgen. Hoe de 'itemTpl'-eigenschap geconfigureerd wordt is te zien in codefragment 7.1. Meer info over XTemplates is ook te vinden in de documentatie via <http://docs.sencha.com/touch/2.2.1/#!/api/Ext.XTemplate>.

Een vaak voorkomende functionaliteit bij lijsten is de zogenaamde *pull-to-refresh*. Door de lijst naar beneden te trekken kan de recentste data opgehaald worden. De standaard listview ondersteunt deze functionaliteit door middel van een plugin te configureren, zie codefragment 7.1. Deze plugin is van het type 'pullrefresh'. Belangrijk is dat de store die aan de listview gekoppeld is een 'idProperty' voor de records geconfigureerd heeft zodat er geen dubbels weergegeven worden wanneer de nieuwe data is opgehaald. Omdat het mogelijk is dat een pers beginnen drukken is nadat de lijst een eerste keer is opgehaald moet de store de records opnieuw sorteren wanneer een pull-to-refresh gedaan wordt. Ook moet de listview zelf opnieuw getekend worden om de juist gekleurde cirkel weer te geven. Om dit te realiseren kan ingehaakt worden op het 'latestfetched'-event van de plugin. Dit event werd pas recent toegevoegd aan het eventsysteem van het framework en bevatte in het begin zowel in de code als in de documentatie nog een typfout. Het is ook nog niet mogelijk om naar dit event te luisteren in een controller. Daarom moet er voorlopig rechtstreeks aan de plugin een luisterraar gekoppeld worden die de lijst juist update als de lijst vernieuwd is.

Om de data te verkrijgen via de webservice wordt een proxy van het type 'rest' geconfigureerd voor de store. Om een antwoord in JSON-formaat te krijgen wordt een

'accept'-header aan het request toegevoegd die ingesteld is op 'application/json'. Om deze data om te zetten in objecten wordt een JSON-reader aan de proxy gekoppeld. Deze reader kan de objecten aanmaken aan de hand van de geconfigureerde mappings van de klasse. De webservice wordt verder besproken in hoofdstuk 8. Meer info over proxies is te vinden op <http://docs.sencha.com/touch/2.2.1/#!/api/Ext.data.proxy.Proxy>.

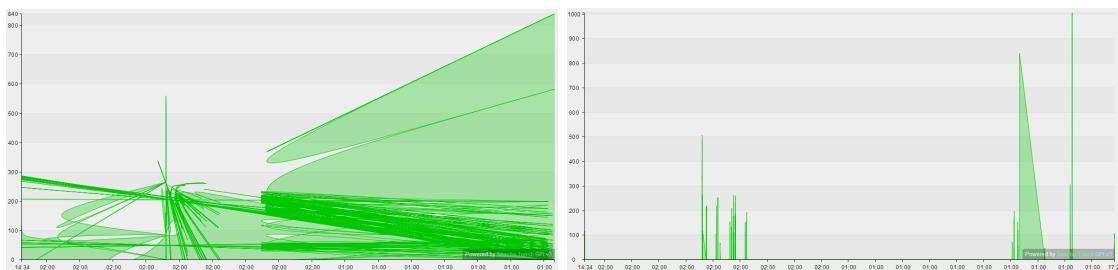
```
Ext.define('Cerm.view.MachineList', {
    extend: 'Ext.dataview.List',
    alias: 'widget.MachineList',
    requires: [
        'Ext.XTemplate'
    ],
    config: {
        store: 'machineStore',
        onItemDisclosure: false,
        itemTpl: Ext.create('Ext.XTemplate',
            '<tpl for=".">' +
            '    <tpl if="this.isActive(active)">' +
            '        ' +
            '    </tpl>',
            '    <tpl if="!this.isActive(active)">' +
            '        ' +
            '    </tpl>',
            '    <div style="overflow: hidden">{name}</div>',
            '</tpl>',
            '    ',
            '{ isActive: function(active) {return active==1;} }'
        ),
        plugins: [{
            listeners: {
                latestfetched: function(){
                    this.getParent().getStore().sort();
                    this.getParent().refresh();
                }
            },
            pluginId: 'pullRefreshPlugin',
            type: 'pullrefresh'
        }]
    }
});
```

**Codefragment 7.1:** Definitie van de MachineList

### 7.1.2 De tellergegevens van een machine weergeven

De gegevens die geregistreerd worden door de teller worden via een webservice beschikbaar gemaakt voor de applicatie. Net zoals in de vorige paragraaf worden deze gegevens opgehaald door een proxy, omgezet door een reader naar objecten van een klasse en bewaard in een store waaraan de proxy gekoppeld is. De klasse van deze objecten heeft als eigenschappen een tellerwaarde, een snelheidsmeting, een waarde die weergeeft of de machine gestopt is en het tijdstip van registratie. De controller, die verantwoordelijk is voor het aansturen van de interface en alle logica die te maken heeft met de machines, stelt bij het selecteren van een machine de juiste url in voor de proxy. De store die aangemaakt wordt om deze gegevens bij te houden krijgt als 'storeId' de referentie van de machine. Zo krijgt elke machine die tijdens de volledige levenscyclus van de applicatie eens geselecteerd werd een store waar de tellerregistraties in terecht komen.

Om deze gegevens weer te geven werd gekozen om net als in de Cermssoftware de snelheid in functie van de tijd weer te geven in een grafiek. Sencha Touch bevat hiervoor verschillende soorten grafieken. Normaal zouden deze gegevens in een *LineChart* goed weergegeven moeten worden. Figuur 7.2 toont echter dat dit mis loopt. Hoewel de grafiek links in de figuur dezelfde data bevat als de grafiek rechts in de figuur blijkt het toch helemaal mis te lopen. De grafiek rechts in de figuur is een *AreaChart* en geeft wel een gewenst resultaat. De oorzaak van dit probleem is niet duidelijk. De *LineChart* zou normaal een lineaire grafiek moeten weergeven terwijl het resultaat daar ver van ligt. Waarschijnlijk is dit een bug in het framework. De *AreaChart* geeft wel het gewenste resultaat dus werd hier verder mee gewerkt.



**Figuur 7.2:** Een LineChart (links) en een AreaChart (rechts) met dezelfde data.

In de applicatie wordt gebruik gemaakt van één grafiek. Bij de selectie van een machine wordt de store met het storeId overeenkomstig met de referentie van de machine aan de grafiek gekoppeld. Een andere mogelijkheid is om voor elke machine een store en een grafiek te voorzien en deze bovenop elkaar te leggen in een container met een *card*-layout. De geselecteerde machine komt dan bovenaan te liggen en is dan zichtbaar. Deze aanpak werd niet gekozen omdat dit zou zorgen voor een grotere DOM-structuur. Bovendien wordt de grafiek geanimeerd wanneer de gekoppelde store geupdate of veranderd wordt.

Interactie met de grafieken is ook mogelijk. Zo kan de gebruiker op de grafiek in- of uitzoomen met het gekende 'pinch-to-zoom'-gebaar. Vervolgens is het ook mogelijk om door een ingezoomde grafiek te scrollen. Bij tests bleek dit uitstekend te werken op toestellen met iOS. Bij een test op een Samsung Galaxy Tab 2.0 (Android) werkte dit minder vlot. De reden hiervoor blijkt de minder goede overgang van CPU naar GPU op Android-toestellen om deze animatie uit te voeren. Dit viel ook al op bij de animaties wanneer er van tab veranderd wordt. Bij tests op een Android-toestel die een quad-core processor heeft in plaats van een dual-core processor bleek dit echter wel dezelfde performantie en gebruikerservaring als op de iOS-toestellen te benaderen. Om de gebruikerservaring te verbeteren voor Android-toestellen in het algemeen werd de interactie met de grafiek zo geconfigureerd dat het enkel mogelijk is om de grafiek uit te vergroten in de richting van de x-as. Dit is minder werk voor de processors en zorgt voor een betere algemene gebruikerservaring.

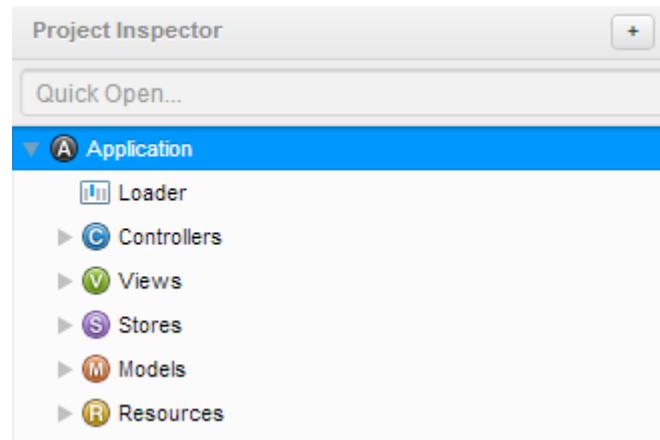
### 7.1.3 De interface optimaliseren voor smartphones én tablets

Het opmerkelijkste verschil tussen smartphones en tablets is natuurlijk de grootte. Tablets hebben een groter scherm en kunnen dus meer weergeven in één scherm. Of anders bekeken, smartphones hebben een kleiner scherm en kunnen dus niet zoveel ineens weergeven als een tablet. De eenvoudigste oplossing is natuurlijk om op een tablet hetzelfde te tonen als op smartphone. Maar op deze manier wordt het grotere scherm van de tablet niet optimaal benut. Werken in de omgekeerde richting kan echter voor problemen zorgen. Om alles op het scherm te krijgen moet alles verkleind worden. Hierdoor kan de interface heel ongebruiksvriendelijk worden of zelfs onbruikbaar. Een beter oplossing past de interface anders aan en maakt van het grotere scherm optimaal gebruik. Ook hiervoor voorziet Sencha Touch een mechanisme, *profiles*.

Profielen bestaan binnen de context van een applicatie. Om de toepassing te optimaliseren voor grote en kleine schermen zijn twee profielen nodig. In dit geval wordt een profiel voorzien voor smartphones en een profiel voor tablets. Deze profiles moeten net als de models, views en controllers ook aangegeven worden in het 'app.js'-bestand.

Profielen worden helaas nog niet ondersteund in de Architect-omgeving. Het is ook niet mogelijk om het 'app.js'-bestand aan te passen buiten Architect. Wanneer het project opgeslagen wordt in Architect wordt namelijk alle code opnieuw gegenereerd en zullen aanpassingen die buiten Architect gemaakt werden overschreven worden.

Gelukkig biedt Sencha Architect de nodige flexibiliteit en kunnen 'custom properties' toegevoegd worden. Op deze manier kan toch aangegeven worden dat er een 'Phone'- en een 'Tablet'-profiel is. Om de profielen zelf aan te maken in Architect is er echter geen omweg. Dit moet volledig buiten de Architect-omgeving gebeuren. Profielen worden namelijk verwacht in de 'app/profile'-map. Er is geen mogelijkheid om in Architect om deze map of zelfs profiles toe te voegen. Profielen worden gewoonweg nog niet ondersteund binnen de ontwikkelomgeving, zie figuur 7.3.



**Figuur 7.3:** Profiles kunnen niet toegevoegd worden in de project inspector van Architect.

Om profielen te gebruiken moet dus buiten de Architect-omgeving gewerkt worden. Hoewel Sencha een plugin voorziet voor Eclipse werd gewerkt met Visual Studio 2012. Binnen de *app-map* moet een nieuwe map voor de profielen gemaakt worden, zie figuur 6.1. In deze map komen de JavaScript-bestanden voor de profielen. In de nieuwe profile-klassen wordt het profiel beschreven. Hier worden nu ook net als in *app.js* klassen aangegeven die dan wel specifiek zijn voor het profiel. Belangrijk is de *isActive*-functie die *true* weergeeft als het profiel actief is. Als deze functie een *true*-waarde geeft wordt de *launch*-functie van dit profiel uitgevoerd bij het opstarten van de applicatie. Meer details over profiles zijn te vinden op <http://docs.sencha.com/touch/2.2.1/#!/guide/profiles>.

Codefragment 7.2 toont hoe de profielklasse voor tablets er uit ziet, enkel de implementatie van de *launch*-functie is hier nu weggelaten. De *launch*-functie maakt een *tabpanel* aan waarbij aan de machinestab een view gekoppeld wordt die specifiek is voor dit profiel.

```
Ext.define('Cerm.profile.Tablet',{
    extend: 'Ext.app.Profile',
    config: {
        name: 'Tablet',
        views: ['MachineView'],
        controllers: ['MachineController']
    },
    isActive: function(){ return Ext.os.is.Tablet || Ext.os.is.
        Desktop;},
    launch: function(){
        //De UI opbouwen om de applicatie mee te starten
    }
});
```

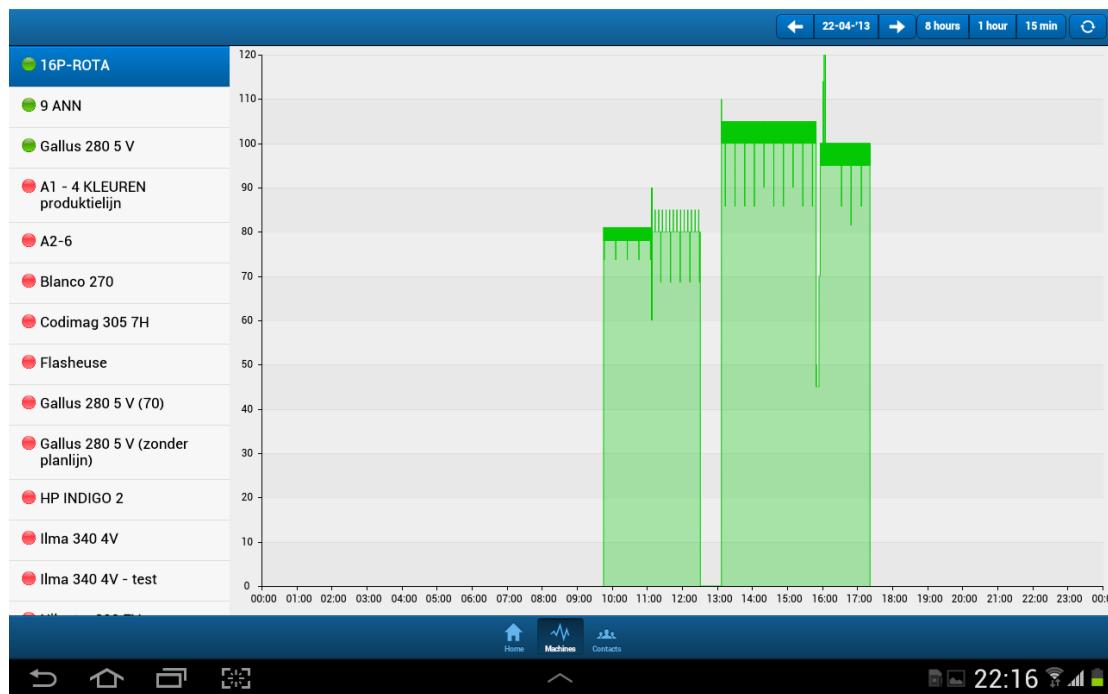
**Codefragment 7.2:** De profielklasse voor tablets

In codefragment 7.2 is ook te zien dat er een view en een controller aangegeven wordt.

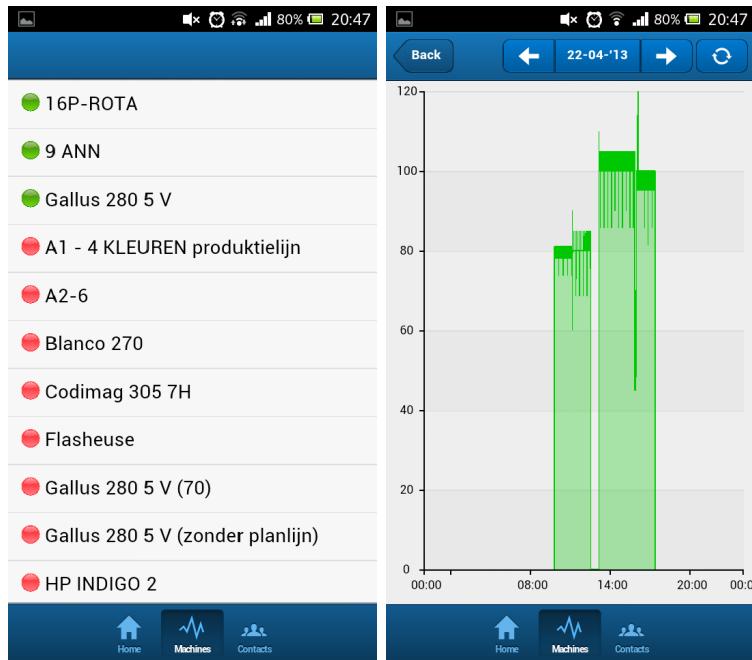
De naam van deze elementen is relatief ten opzichte van het profiel. De naam van de view die geladen zal worden door dit profiel is voluit `Cerm.view.phone.MachineView` en niet `Cerm.view.MachineView` zoals verwacht zou worden bij de afhankelijkheden in het 'app.js'-bestand. Is de klassenaam niet relatief ten opzicht van het profiel dan moet deze voluit aangegeven worden. Zoals de volledige naam van de klasse al doet vermoeden moet er een nieuwe submap gemaakt worden in de `view-map` voor de views die bij dit profiel horen. Voor profielspecifieke controllers moet dit ook gedaan worden in de `controller-map`.

De *MachineView* voor tablets is een container met een *hbox*-layout. Dit wil zeggen dat het zijn kindelementen horizontaal naast elkaar zal weergeven. De elementen die in deze container naast elkaar worden weergegeven zijn de lijst van machines en de grafiek. Door het 'flex'-attribuut van deze twee elementen in te stellen worden de componenten met de corresponderende verhoudingen weergegeven in de container, zie figuur 7.4.

Voor smartphones is het niet ideaal om deze twee elementen naast elkaar weer te geven. Daarom wordt in dit geval gebruik gemaakt van een container met een *card*-layout. Hierbij worden de kindelementen bovenop elkaar gelegd, waarbij de gebruiker de bovenste natuurlijk te zien krijgt. De gebruiker krijgt op zijn smartphone eerst de lijst van drukpersen te zien. Wanneer een machine geselecteerd wordt komt de grafiek bovenaan te liggen die de data van deze pers zal weergeven. Met een terugknop wordt de machinelijst terug naar boven gebracht, zie figuur 7.5.



**Figuur 7.4:** De MachineView op een Samsung Galaxy Tab 2.0.



**Figuur 7.5:** De MachineView op een Sony Xperia P.

Omdat het aansturen van de interface verschilt tussen tablets en smartphones zal een controller geladen moeten worden die dit op de juist manier doet. De controller voor smartphones bijvoorbeeld zal het juiste element naar voor moeten brengen als een machine geselecteerd wordt of wanneer de terugknop gebruikt wordt.

Omdat de controllers voor smartphones en tablets veel gemeenschappelijke code zullen hebben werd een hoofdklasse geschreven waar de andere twee controllers dan van worden afgeleid. Zo komt alle gemeenschappelijke functionaliteit, zoals bijvoorbeeld het laden van de stores, in *Cerm.controller.MachineController*. De controllers voor smartphones en tablets worden dan respectievelijk *Cerm.controller.phone.MachineController* en *Cerm.controller.tablet.MachineController*. Voor deze profiel-specifiche controllers werd terug gewerkt met Visual Studio 2012.

#### 7.1.4 De functionaliteiten gebundeld in de toolbar

Zoals op figuur 7.4 en figuur 7.5 te zien is zijn er bovenaan extra knoppen in de toolbar. Deze knoppen zijn gegroepeerd in *segmented buttons*. De meest linkse groep knoppen zijn de knoppen om een datum te kiezen. De middelste van die groep toont de geselecteerde datum en toont een *datepicker* wanneer op deze knop getikt wordt. De knoppen links en rechts hiervan selecteren een dag eerder of later dan de huidig geselecteerde datum. Een datum selecteren zorgt er altijd voor dat de data voor heel die dag zichtbaar is.

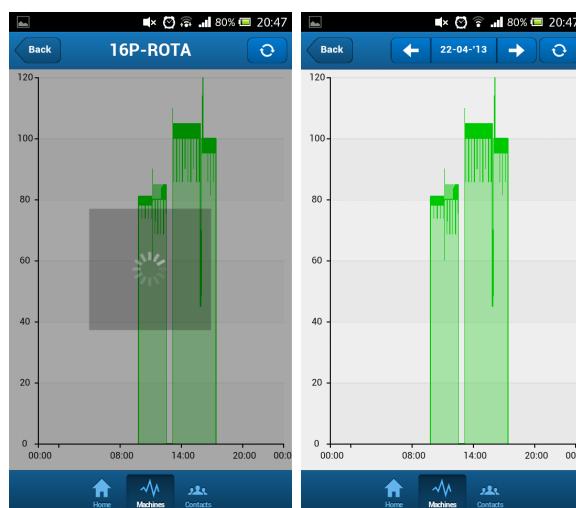
Op figuur 7.4 is nog een tweede groep knoppen te zien. Deze knoppen dienen om een *zoomvenster* in te stellen. Zo kan er bijvoorbeeld ingezoomd worden naar de laatste voorbijge

acht uur, het voorbije uur of het voorbije kwartier. Eens een *zoomvenster* geselecteerd kan dit verschoven worden over de grafiek. Op een andere zoomknop tikken zorgt voor een corresponderende uitvergrotting. Opnieuw op de knop tikken die al ingedrukt is verwijderd het *zoomvenster* zodat terug de hele dag te zien is. Wisselen tussen machines behoudt de geselecteerde datum en het *zoomvenster*. Op deze manier kan de gebruiker bij wijze van spreken snel door hetzelfde venster naar verschillende drukpersen kijken.

De meest rechtse knop, die niet gegroepeerd is, dient om de grafiek te vernieuwen. Een tik op de knop zorgt er voor dat de data opnieuw wordt opgehaald. Lang drukken op deze knop zorgt er voor dat de data automatisch vernieuwd wordt om een bepaalde tijd. Om de gebruiker hierop te wijzen kleurt deze knop groen. Het automatisch vernieuwen wordt gedaan met behulp van de *setInterval*-functie van het globale *window*-object. Opnieuw eenmaal tikken verwijdert het interval en kleurt de knop terug blauw.

Al de logica hierachter is gelijk voor smartphones en tablets en wordt dus samengebracht in de hoofdklasse *Cerm.controller.MachineController*.

Op de tablet is er ruimte genoeg om al deze knoppen meteen weer te geven. Dit is een probleem op het kleinere scherm van smartphones. Als oplossing wordt, zoals te zien in figuur 7.5, er voor gezorgd dat in portretmodus de knoppen om een *zoomvenster* in te stellen niet getoond worden. Omdat op een smartphone de lijst niet meer zichtbaar is na het selecteren van een machine wordt er eerst nog een seconde lang de naam van de drukpers getoond alvorens de knoppen om een datum te selecteren tevoorschijn komen, zie figuur 7.6. Hiervoor wordt gebruik gemaakt van de *setTimeout*-functie van het *window*-object. Om de knoppen voor de *zoomvensters* te kunnen gebruiken moet de gebruiker zijn smartphone in landschapmodus houden, zie figuur 7.7. Hierop is ook te zien dat de automatische vernieuwing ingeschakeld is.



**Figuur 7.6:** De dynamische toolbar op een smartphone in portretmodus.  
De knoppen om een zoomvenster in te stellen zijn niet zichtbaar.



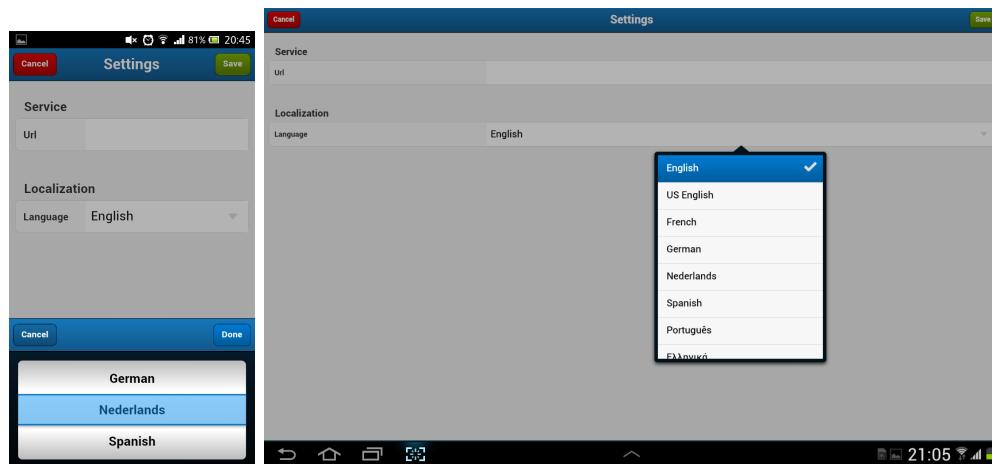
**Figuur 7.7:** De dynamische toolbar op een smartphone in landschapmodus.  
De knoppen om een zoomvenster in te stellen zijn zichtbaar.

Het *longpress*-event waarop de automatische vernieuwing ingeschakeld wordt is geen standaard event van een knop. De controller blijkt dit ook niet op te vangen. Daarom wordt aan de vernieuwknop een luisterraar gekoppeld die bij deze gebeurtenis zelf nog eens expliciet een *longpress*-event afvuurt voor deze knop. Op deze manier lukt het wel om deze gebeurtenis op te vangen in de controller.

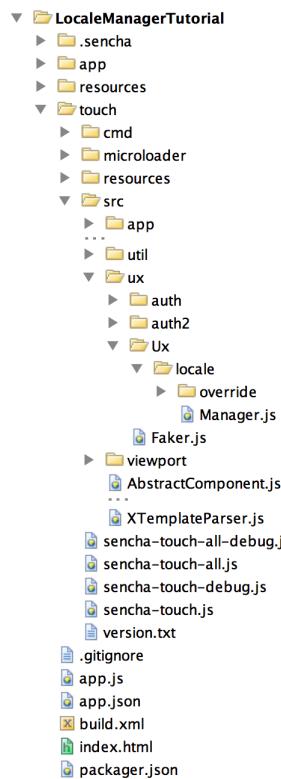
## 7.2 Internationalisering

De applicatie moet in meerdere talen beschikbaar zijn. Standaard wordt de applicatie opgestart in het Engels. Om de taal in te stellen wordt een instellingenmenu voorzien dat toegankelijk is vanaf het startscherm. Op figuur 7.8 is te zien dat de interface zich hier ook automatisch aanpast aan een tablet of smartphone. Dit gebeurt automatisch door de component uit het framework. Het is hier ook mogelijk om de service-url in te vullen of aan te passen. De instellingen worden opgeslagen in de *localStorage*.

Er is geen HTML5-standaard om webapplicaties te internationaliseren. Meestal wordt dit bij webtoepassingen gedaan met behulp van server-side-technologie. De applicatie moet echter volledig zelfstandig aan de clientzijde werken. Er wordt enkel met de server gecommuniceerd om data op te halen. Het Sencha Touch framework biedt hier ook geen ingebouwde oplossing voor. Mitchell Simoens, die bij Sencha werkt sinds maart 2011, heeft hiervoor wel een oplossing geschreven. Om de applicatie te internationaliseren wordt een *Ux.locale.Manager*-klasse voorzien. Deze klasse zal *gelocaliseerde* componenten updaten als de *locale* veranderd wordt. Om dit te realiseren moeten componenten ook wel uitgebreid worden om dit te ondersteunen. Ook hier heeft Mitchel Simoens al voor de uitbreiding van enkele populaire componenten gezorgd. De code is opensource, mag vrij gebruikt worden en is terug te vinden op <https://github.com/mitchellsimoens/Ux.locale.Manager>. Een voorbeeld is ook beschikbaar.



**Figuur 7.8:** De instellingen op smartphone en tablet.



**Figuur 7.9:** De locatie voor de extensie.

Om deze extensie in te voegen in het project kan de code als zip gedownload worden via bovenstaande link. Deze zip wordt uitgepakt in *touch/src/ux*, zie figuur 7.9. Opdat de applicatie zou weten waar te zoeken naar klassen binnen de *Ux*-namespace, wordt het pad naar deze map toegevoegd aan de configuratie van de loader van de applicatie. De sleutel van dit pad wordt *Ux*, de waarde *touch/src/ux/Ux*. Vervolgens worden alle uitgebreide klassen en de manager zelf toegevoegd aan de afhankelijkheden van de applicatie. Zo worden deze klassen geladen als de applicatie gestart wordt en kan gebruik gemaakt worden van de extra functionaliteit.

Als de applicatie opgestart wordt moet de *Ux.local.Manager* geconfigureerd en geïnitialiseerd worden. De *SettingsController* is verantwoordelijk voor alles wat met de instellingen te maken heeft. De initialisatie en configuratie gebeurd in de *launch*-functie van deze controller. De controller kijkt of er al een taal opgeslagen is in de *localStorage*. Als er geen taal opgeslagen is wordt Engels gekozen om de manager te configureren. Tijdens de configuratie wordt ook aangegeven waar de bronbestanden voor de vertalingen zich bevinden. Er werd gekozen om deze bronbestanden te plaatsen in *resources/locales*. (Ashworth, 2013)

Eens de configuratie compleet is kan de *init*-functie van *Ux.locale.Manager* opgeroepen worden en kan deze zijn werk doen. Hiervoor zal de manager afhankelijk van de geselecteerde *locale* het juiste bronbestand ophalen uit *resources/locales* en de tekst van de ondersteunde componenten vertalen. Zo'n bronbestanden zijn JSON-bestanden. Codefragment 7.3 geeft een voorbeeld. Cerm heeft een eigen systeem waarmee vertalingen beheerd worden. Hierbij is het Nederlands de brontaal. Het JSON-bestand beschrijft een object met verschillende sleutel-waarde-paren. De sleutel is altijd de Nederlandse term, de waarde is de term in de taal voor de *locale* die opgegeven is als bestandsnaam. Bijvoorbeeld het bestand met de vertalingen voor Amerikaans Engels heeft 'en\_US.json'.

```
{
  "Annuleren" : "Cancel",
  "Contacten" : "Contacts",
  "Home" : "Home",
  "Instellingen" : "Settings",
  "Machines" : "Machines",
  "OK" : "OK",
  "Opslaan" : "Save",
  "Taal" : "Language",
  "Terug" : "Back"
}
```

**Codefragment 7.3:** Een voorbeeld JSON-bestand voor Engelse vertaling

Om bijvoorbeeld een knop te internationaliseren moet te werk gegaan worden zoals in code 7.4. Aan het *config*-object van de knop wordt een *locales*-object toegevoegd. In dit *locales*-object worden de Nederlandse termen gekoppeld aan de eigenschappen van de knop. Zo wordt in het voorbeeld de tekst van de knop ingesteld op de native string 'Terug'. Op deze manier weet de manager dat de tekst voor deze knop de vertaling voor 'Terug' moet zijn. De manager gaat hiervoor afhankelijk van de ingestelde *locale* in het juiste bestand op zoek naar de waarde voor de sleutel 'Terug'. Belangrijk is dat er een uitbreiding beschikbaar is op de *button*-klasse in *touch/src/ux/Ux/locale/override*. Zoals eerder vermeld zijn deze reeds beschikbaar voor een aantal veelgebruikte componenten. Wanneer dit niet het geval is moet deze uitbreiding zelf geschreven worden om de vertalingen te ondersteunen.

```
Ext.define('Cerm.view.BackButton', {
  extend: 'Ext.Button',
  alias: 'widget.BackButton',
  config:
  {
    locales: {text: 'Terug'}
  }
});
```

**Codefragment 7.4:** Een terugknop internationaliseren

De ondersteunde talen worden beschreven in *supported.json* dat zich op dezelfde locatie bevindt als de bestanden voor de vertaling. Dit bestand bevat voor elke ondersteunde taal een object met een naam die ingesteld is op de taal en waarde die ingesteld wordt met de bijhorende locale. Via een store worden deze gegevens gekoppeld aan de *listinput* op het instellingenschermscherm. Op deze manier kan de gebruiker kiezen uit de lijst ondersteunde talen. Bij selectie van een taal wordt de *updateLocale*-functie van de manager opgeroepen met als argument de corresponderende locale van de geselecteerde taal. Onmiddelijk worden alle beschikbare vertalingen uitgevoerd.

Alles werkte perfect, zowel in Safari (iOS) als Chrome (Android). Behalve bij Internet Explorer 10 (Windows Phone 8). De applicatie wou na toevoeging van de internationalisatie maar niet opstarten. Dit kwam omdat er een fout optrad bij het laden van *Ux.locale.Manager*. In de code van deze klasse werd gebruik gemaakt van de *language*-eigenschap van het globale *navigator*-object. Op deze eigenschap wordt dan een *split*-functie uitgevoerd. Het bleek echter dat Internet Explorer hierbij de standaard niet volgt en het *navigator*-object geen *language*-eigenschap heeft, zie [http://msdn.microsoft.com/en-us/library/ie/ms535867\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/ie/ms535867(v=vs.85).aspx). Wanneer de *split*-functie op een niet bestaand object uitgevoerd wordt geeft dit een fout. Om dit op te lossen werd de manager zo aangepast dat *navigator.userLanguage* (wat niet tot de standaard behoort, maar wel gebruikt wordt door Microsoft) gebruikt wordt wanneer *navigator.language* niet bestaat.

### 7.3 De applicatie voorzien van een login

Een login voor de applicatie is ook een van de vereisten. Hiervoor krijgt de gebruiker een loginscherm te zien als hij de applicatie opstart. De gebruiker logt in door zijn gebruikersnaam en wachtwoord in te vullen. Hierbij krijgt de gebruiker ook de keuze of de applicatie de login moet onthouden. Wanneer de gebruiker ingelogd is komt het startscherm tevoorschijn met een knop om terug uit te loggen. Op de tabbalk worden de verschillende modules beschikbaar gesteld waar de gebruiker rechten voor heeft. Figuur 7.10 toont het scherm voor en nadat de gebruiker is ingelogd.

Achter de schermen gaat de *AuthenticationController* een JSON-object posten naar de loginservice, op voorwaarde dat alle velden zijn ingevuld. Deze service controleert de gebruikersnaam en het wachtwoord en stuurt een JSON-object terug met de gegevens van de gebruiker. Aan de hand van deze data wordt een *User*-object aangemaakt. Dit object bevat de naam van de gebruiker, een referentie en de rechten die de gebruiker heeft. Op basis van deze rechten zorgt de *AuthenticationController* dat de juiste modules beschikbaar zijn. Wanneer de gebruiker aangeeft dat de login onthouden moet worden wordt het verkregen *User*-object bewaard in een store die zijn records bewaart in de *localStorage*. Bij het opstarten van de applicatie wordt gecontroleerd of er een *User*-object opgeslagen is. Is dit het geval dan wordt de gebruiker meteen naar het startscherm gebracht. In het andere geval wordt gevraagd om in te loggen. Wanneer de gebruiker uitlogt wordt dit *User*-object verwijderd en zal de gebruiker dus niet onthouden worden.



Figuur 7.10: Inloggen in de applicatie.

Deze keer werd geen proxy gebruikt om te communiceren met de webservice. De post naar de webservice werd in puur JavaScript geschreven in de controller met behulp van een *XmlHttpRequest*-object. Het *User*-object wordt dus nu ook zonder *reader* aangemaakt. Belangrijk is dat opnieuw een *accept*-header toegevoegd wordt aan het *request*. Deze keer moet er ook een *content-type*-header toegevoegd worden met de waarde *application/json* omdat er een JSON-object gepost wordt.

Nadat de gebruiker ingelogd is moet het loginscherm vernietigd worden. Anders bevindt dit scherm zich nog in de DOM-structuur. Wanneer de gebruiker uitlogt en een andere gebruiker wil zich aanmelden zal opnieuw de vorige gebruiker aangemeld worden. Dit komt omdat de vorige gegevens die ingevuld werden zich voor de nieuwe waarden in de DOM-structuur zullen bevinden en deze dus gebruikt zullen worden om in te loggen. Daarom wordt bij het inloggen het loginscherm vernield en bij het uitloggen een nieuw loginscherm getoond. Anders zou het zelf mogelijk zijn om in te loggen zonder iets in te vullen nadat een gebruiker zich heeft uitgelogd en de applicatie niet opnieuw opgestart is. Opdat het wachtwoord van de gebruiker niet zou kunnen onderschept worden tijdens de communicatie met de server moet van de webservice gebruik gemaakt worden via het https-protocol. Hiervoor moet de service-url ingesteld worden met HTTPS als protocol. In hoofdstuk 8 wordt meer besproken over de *LoginService* en het configureren van de server om communicatie met het HTTPS-protocol mogelijk te maken.

## 7.4 De contactenmodule

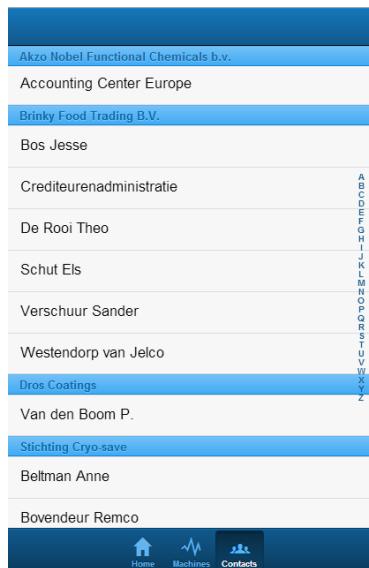
De tweede module die tot de opdracht behoorde is een module waarin de gebruiker contactgegevens kan raadplegen en interactief gebruiken. Zo moet het bijvoorbeeld mogelijk zijn om een contact direct te bellen of een e-mail te sturen.

### 7.4.1 De contactenlijst weergeven

Contacten worden op dezelfde manier weergegeven zoals de lijst van machines. Klanten kunnen één of meerdere contactpersonen hebben. Een *CustomerContact* heeft dus een klantennaam en -referentie, een contactreferentie en een voor- en achternaam. Een proxy haalt de contacten op van de ingelogde gebruiker en plaats objecten met deze gegevens in een store. Deze store is zo geconfigureerd dat records gegroepeerd worden op klantennaam. Binnen deze groepering worden contacten gesorteerd volgens achternaam. De *CustomerContactList* krijgt een eigenschap 'indexBar' die op 'true' wordt ingesteld. Hierdoor wordt rechts van de lijst een control voorzien waarmee door de contacten gebladerd kan worden op niveau van de klantennamen in plaats van op het niveau van de contacten zelf te moeten scrollen. Deze control komt ook voor op iOS-toestellen en is vooral handig voor lange lijsten. Sencha Touch heeft ook een zoekveld. Maar omdat de datastructuur van de stores is het niet evident om deze functionaliteit op een efficiënte, gebruiksvriendelijke en responsieve manier te aan te bieden. Een goede oplossing zou de contacten lokaal kunnen bewaren in een database waarop efficiëntere queries kunnen uitgevoerd worden. Een andere oplossing zou zijn dit aan de serverzijde uit te voeren. Figuur 7.11 heeft een beeld van hoe de lijst contacten wordt weergegeven. Op die figuur is te zien dat de contactpersonen gegroepeerd zijn per klant en dat de klant een blauwe achtergrond heeft. Bij het bladeren door de contacten blijft de klantennaam altijd bovenaan hangen tot er een andere klantennaam bovenaan komt en zijn plaats inneemt. Hoe meer contactpersonen een klant heeft hoe langer deze klantennaam bovenaan zal blijven hangen wanneer er verder gescrolld wordt.

Contacten worden niet lokaal opgeslagen. Nochtans zou dit wel mogelijk gemaakt kunnen worden. Hier zijn twee mogelijkheden voor. Een eerste mogelijkheid is dat de webapp omgebouwd wordt tot een hybride app zodat er gebruik gemaakt kan worden van de native API's om de contacten lokaal te op te slaan. Bij een tweede mogelijkheid wordt er gebruik gemaakt van de IndexedDB-API die tot de HTML5-standaard behoort. Deze API wordt helaas nog niet door alle mobiele browsers ondersteund. Belangrijke browsers die deze API nog niet ondersteunen zijn Safari op iOS en de standaard Androidbrowser. Internet Explorer 10 ondersteunt deze API wel. Ook de BlackBerry browser biedt hiervoor ondersteuning maar vereist wel nog een webkit-prefix hiervoor, wat toch nog niet helemaal volgens de standaard is. Safari, BlackBerry en Android ondersteunen wel de WebSQL-API maar deze hoort niet bij de standaard. Sowieso zou dus een beroep gedaan moeten worden op beide API's om alle besturingssystemen te ondersteunen. Sencha Touch heeft standaard een component om met de WebSQL-API te werken en gebruikers van het framework hebben

zelf ook al een component geschreven voor de IndexedDB-API. Toch werd beslist hiervoor een afwachtende houding aan te nemen en geen prioriteit te geven aan het lokaal bewaren van contacten. Nagaan welke browsers een bepaalde HTML5-feature ondersteunen kan eenvoudig via <http://caniuse.com>.



**Figuur 7.11:** De contactenlijst.

Contacten worden opgehaald wanneer de gebruiker voor het eerst naar de contactenmodule gaat als de applicatie opgestart is. Eenmaal geladen blijven deze contacten gedurende de volledige levenscyclus van de applicatie beschikbaar. Tijdens het laden wordt een draaiend wiel getoond om de gebruiker er op te wijzen dat de data geladen wordt. Om een idee te geven om hoeveel data het hier gaat wordt een overzicht gegeven van testdata in tabel 7.1.

Contacten	kB
600	94
1800	254
12500	1800
32000	3200

**Tabel 7.1:** Schatting van het aantal contacten en hun grootte in kilobytes.

Bij testen konden meer dan dertigduizend contacten geladen worden, zowel op Android als op iOS 6. Bij iOS 5.1 blijkt een maximum te liggen op iets meer dan tienduizend, bij meer dan twaalfduizend crashte de browser. Op het Sencha-forum zijn er nog gevallen van Safari op iOS 5.1 die crasht, dit werd dan ook opgenomen als een gekende bug bij Sencha.

### 7.4.2 Contactgegevens weergeven

Na het selecteren van een contact worden de contactgegevens weergegeven. In dit geval werd een proxy rechtstreeks aan het *ContactDetails*-model gekoppeld aangezien er toch maar één contact geselecteerd kan zijn op een bepaald moment.

De contactgegevens bestaan uit een naam, voornaam, referentie, telefoonnummer, e-mailadres, stad, land, provincie, wijk, straat en postcode. Om deze gegevens weer te geven wordt gebruik gemaakt van een *Ext.form.Panel*. Door de *name* van de velden op het formulier in te stellen op de overkomstige eigenschap van het model wordt de datakoppeling gerealiseerd door het opgehaalde object in te stellen als record van het formulier.

Figuur 7.12 toont hoe de contactgegevens weergegeven worden. Net boven de tabbalk zijn drie knoppen te zien. Deze dienen om de contactpersoon te mailen, te bellen of om de wegbeschrijving naar het adres op te vragen. Op een tablet zal de knop voor te bellen niet aanwezig zijn aangezien dit weinig nut heeft. Dit wordt verder besproken in 7.4.4.

Voor de contactenmodule wordt gebruik gemaakt van de standaardcomponent *Ext.navigation.View*. Deze component is een container met een card-lay-out die zelf een terugknop voorziet wanneer een component naar deze container *gepushed* wordt.

Bij het testen dook er een probleem op wanneer er te snel achter elkaar op een contact in de lijst werd getikt. Het gevolg van deze actie was dat er twee componenten naar de container *gepushed* werden in plaats van één. Hierdoor kwam de navigatie in de war en was het niet mogelijk om volledig terug te keren naar de contactenlijst. Dit probleem wordt verder geadresseerd in paragraaf 7.5.

Contact	
First Name	[REDACTED]
Name	[REDACTED]
Email	[REDACTED]@[REDACTED]
Phone	0032-[REDACTED]

Address	
City	HASSELT/KURINGEN
Country	BE
County	
District	[REDACTED]
Street	[REDACTED]

Figuur 7.12: Contactgegevens

### 7.4.3 Klantgegevens weergeven

Op figuur 7.12 is te zien dat bovenaan de naam van de klant staat. Als op deze klantnaam getikt wordt *pusht* de *ContactController* een nieuw formulier op de container om de klantgegevens weer te geven.

Het weergeven van klantgegevens gebeurt op dezelfde manier als bij de contactgegevens. Klantgegevens bestaan uit een naam, telefoonnummer, referentie en een adres zoals bij de contactgegevens. Om de klantennaam weer te geven bij de contactgegevens en het mogelijk te maken op deze klantennaam te tikken moet dit geprogrammeerd worden in de functie die wordt uitgevoerd wanneer op een contact in de lijst getikt wordt, zie codefragment 7.5. Deze code werd ingekort en er werd commentaar geplaatst bij de belangrijkste delen. Het gewenste resultaat wordt verkregen door bij het aanmaken van de view die de contactgegevens weergeeft de titel in te stellen. De titel wordt automatisch getoond in de titlebar van de *navigationview* wanneer deze view bovenaan komt. Vervolgens moet er nog voor gezorgd worden dat er een event afgevuurd wordt wanneer er op deze titel getikt wordt. Op die manier kan de controller hierop reageren door naar dit event te luisteren. De reden waarom dit zo gedaan wordt is omdat er gebruik gemaakt wordt van de standaard *navigationview* en er weinig controle mogelijk is over wat er in de titelbalk bovenaan terecht komt. Voor de navigatie bij de machines op een smartphone werd geen gebruik gemaakt van een *navigationview* en werd dit allemaal zelf gecontroleerd. Gewoon een container gebruiken met een card-layout en zelf zorgen voor een terugknop enzovoort biedt meer flexibiliteit. Op deze manier zou bijvoorbeeld gewoon een knop geplaatst kunnen worden in de titelbar.

```
onCustomerContactListItemSelected: function(..., record, ...){  
    var me = this;  
    var settings = me.getApplication()  
        .getController('SettingsController');  
    var contact = Ext.create('Cerm.model.ContactDetails');  
    //...  
    //View aanmaken en title instellen  
    var contactView = Ext.create('Cerm.view.ContactDetailView',  
    {title:  
        '<h1 class="...">  
        <span class="...">i</span>' + record.get('customerName') +'  
        </h1>');  
    //...  
    //Event afvuren wanneer op klantennaam getikt wordt  
    me.getContactsView().getNavigationBar().element.on({  
        delegate: 'h1',  
        tap: function(e){  
            me.getContactsView().fireEvent('bartap', me.getContactsView());  
        }  
    });  
    me.getContactsView().push(contactView);  
    me.getContactsView().unmask();  
}
```

Codefragment 7.5: Implementatie van onCustomerContactListItemSelected

## 7.4.4 Interactiviteit toevoegen

### 7.4.4.1 Functionaliteit voor bellen en mailen

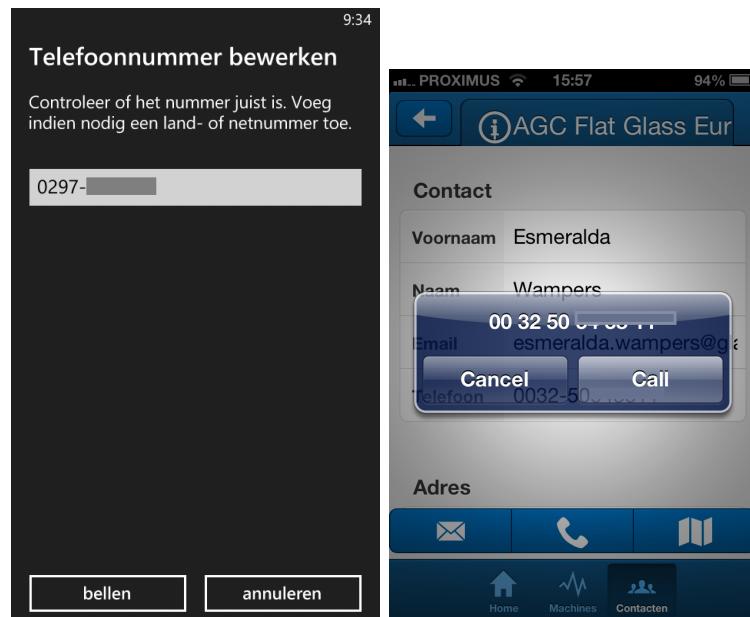
Bij figuur 7.12 werd al gesproken over de knoppen om te mailen, te bellen en een kaart te tonen. Deze knoppen komen ook voor bij de klantgegevens. De controller controleert hiervoor of deze knoppen wel nuttig zijn. Zo wordt bijvoorbeeld een mailknop enkel getoond als er een e-mailadres beschikbaar is, anders wordt deze verborgen. Bellen is meestal ook niet mogelijk met een tablet, daarom wordt de belknop ook niet getoond op tablets.

De mailknop zal de mailapplicatie starten om een email te verzenden met het emailadres van de contactpersoon als geadresseerde. De belknop opent de telefoonfunctionaliteit met het telefoonnummer van de contactpersoon om vervolgens te bellen, zie figuur 7.13. Deze functionaliteit is natuurlijk eigen aan het besturingssysteem en kan dus anders aangeboden worden. Op Android bijvoorbeeld zal het besturingssysteem vragen welke mailapplicatie, als er meerdere geïnstalleerd zijn, gebruikt moet worden wanneer er op de mailknop gedrukt wordt. Er kan dan ook gekozen worden om een bepaalde applicatie als standaard in te stellen zodat dit niet telkens gevraagd moet worden, zie figuur 7.14.

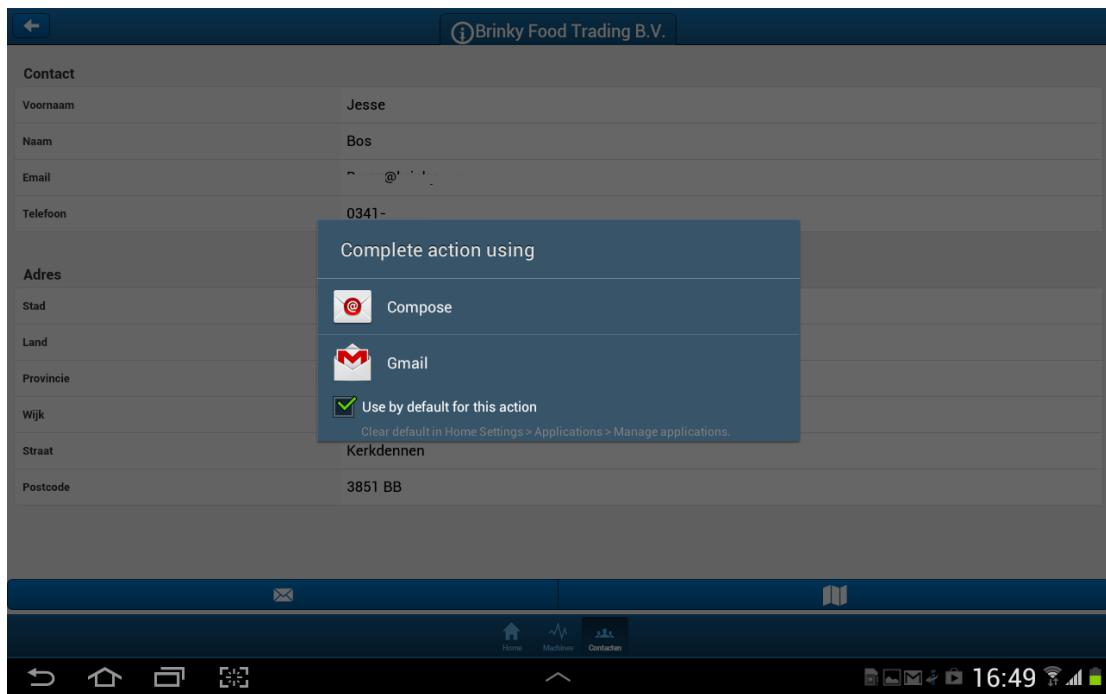
In paragraaf 3.2 bleek dat Apple als een pioneer beschouwd kan worden op het gebied van ondersteuning voor webapps in mobiele besturingssystemen. Er zijn dan ook mechanismen voorzien om andere applicaties op te starten vanuit een webtoepassing. Hiervoor bestaan er verschillende schema's voor links. De meeste worden ook ondersteund op andere platformen. Codefragment 7.6 toont een HTML-voorbeeld van deze links voor mailen en bellen. In de applicatie worden deze links door de controller samengesteld en geopend met behulp van de *open*-functie van het globale *window*-object. De knoppen voor deze acties zijn eerder al gezegd respectievelijk enkel zichtbaar wanneer er een e-mailadres of een telefoon nummer beschikbaar is. (Apple, 2012)

```
<!-- Bellen -->
<a href="tel:1-408-555-5555">1-408-555-5555</a>
<!-- Mailen -->
<a href="mailto:frank@wwdcdemo.example.com">Mail John Frank</a>
```

**Codefragment 7.6:** Links om te mailen en te bellen



**Figuur 7.13:** Het scherm na het gebruiken van de belknop op een Windows Phone en iPhone.



**Figuur 7.14:** Het scherm na het gebruiken van de mailknop op een Android-toestel.

#### 7.4.4.2 De mapsfunctionaliteit

Tikken op de kaartknop zorgt ervoor dat de controller een nieuwe component naar de container *push*t. Dit is een *Ext.Map*-component die gebruik maakt van de maps-API van Google. In Sencha Architect moet hiervoor de Google Maps API toegevoegd worden aan de resources. Dit voegt een object toe aan de *js*-rij in het *app.json*-bestand met de *path* ingesteld op de url die verwijst naar de API en *remote* op *true*.

De controller gaat als eerste met behulp van de Google Maps API het adres vertalen naar coördinaten op basis van de straat en de postcode. Is er geen straat en postcode beschikbaar dan wordt de mapknop niet getoond. Wanneer de locatie niet gevonden kan worden, bijvoorbeeld wanneer het adres een postbus is, wordt dit aan de gebruiker gemeld en keert de applicatie terug naar het vorige scherm. Is de locatie wel gevonden dan wordt de kaart geladen en weergegeven met een markering op de locatie.

Hierna wordt de eigen locatie bepaald. Wanneer nog geen toestemming of weigering gegeven is om de locatie te gebruiken wordt hier om gevraagd. Nadat de locatie bepaald is wordt de route opgehaald via Google Maps en wordt deze weergegeven op de kaart, zie figuur 7.15. (Drucker & Perry, 2013)



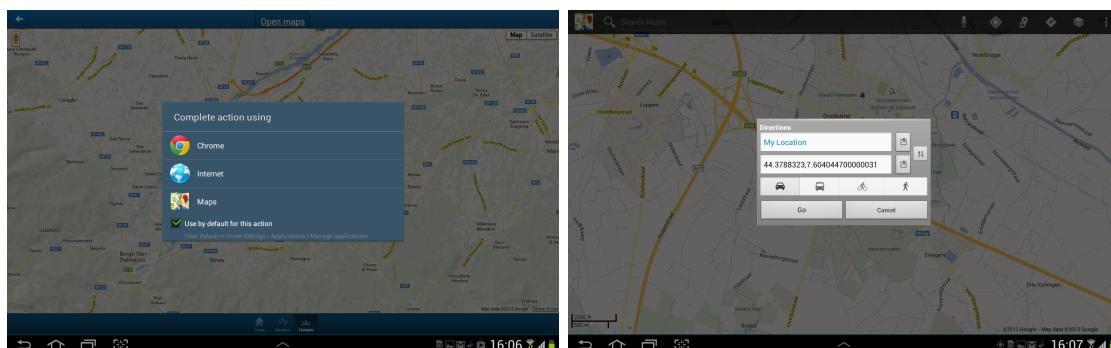
Figuur 7.15: De mapview van de applicatie.

Boven de kaart is er een knop geplaatst, net zoals de klantennaam bij de contactgegevens, om een navigatie-applicatie te openen als deze beschikbaar is. Op Android is dit standaard natuurlijk de native app voor Google Maps maar andere navigatietoepassingen of browsers kunnen hierbij ook geopend worden, zie figuur 7.16. Windows Phone heeft geen standaard navigatieapplicatie en opent de map-link gewoon in de browser met Google Maps.

Op iOS wordt gebruik gemaakt van Apple Maps. Hiervoor wordt de gebruikte link programatisch aangepast in de controller, zie codefragment 7.7. De navigatieapplicaties worden opgestart met de coördinaten van de bestemming zodat de navigatie gewoon starten de volgende stap is. In het codefragment is te zien dat voor iOS gebruik gemaakt wordt van het *maps*-schema in de link. Voor andere besturingssystemen wordt gewoon de link naar Google Maps gebruikt. Een andere mogelijkheid voor alle besturingssystemen is gebruik maken van het *geo*-schema. Hiermee werd echter niet het gewenste resultaat bereikt. Ofwel werd Google Earth opgestart op iOS ofwel kon de navigatie niet meteen hierna gestart worden op Android.

```
var openmaps = 'http://maps.google.com/?daddr=';
if(Ext.os.is.iOS) openmaps = 'maps:q=';
me.getContactsView().getNavigationBar().setTitle('<h2 class="...><a style="color:white" target="_blank" href="'+openmaps
+lat+', '+lng+'>Open maps</a></h2>');
```

**Codefragment 7.7:** De link juist instellen om de navigatieapplicatie te starten



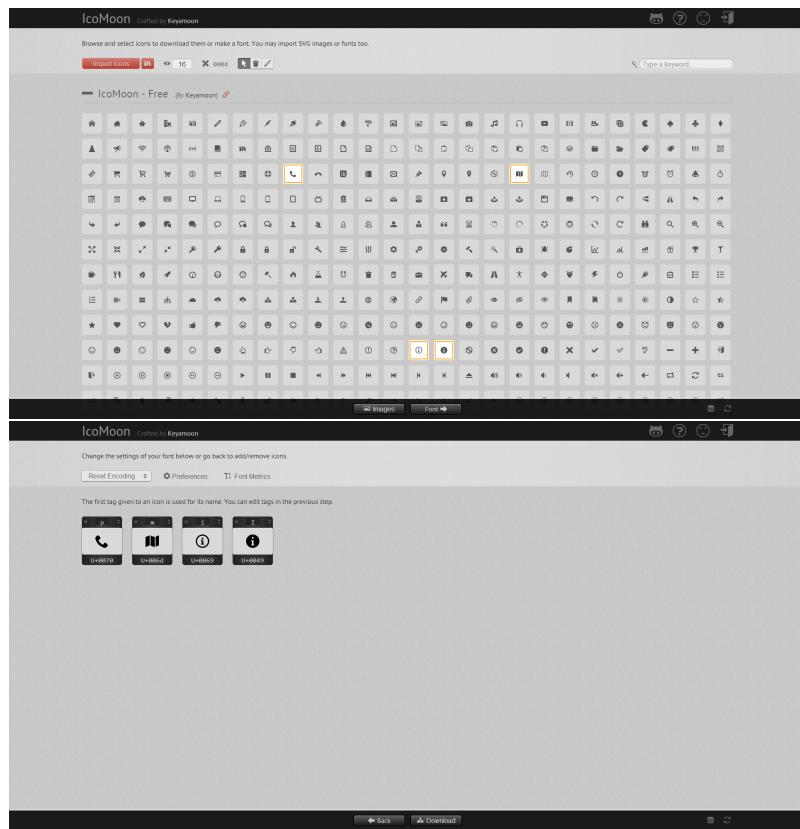
**Figuur 7.16:** De Google Maps app gestart vanuit de applicatie op Android.

Merk op dat dit bedoeld is voor webapps. Bij het testen van de applicatie als hybride app op een Android-tablet werd gewoon de browser geopend en bleek het niet mogelijk te kiezen om Google Maps te openen. Hiervoor zal dan gebruik gemaakt moeten worden van de voorziene JavaScript API's die een brug maken naar de native API's.

### 7.4.5 Gepaste iconen voorzien

Sencha Touch beschikt standaard over een verzameling iconen. Vroeger was deze bijgeleverd set zeer uitgebreid, alle iconen van Pictos ([www.pictos.cc](http://www.pictos.cc)) waren beschikbaar. Om een betere browseronafhankelijke ondersteuning te bieden werd de techniek die voor iconen gebruikt werd veranderd. Met de lancering van Sencha Touch 2.2, dat ondersteuning bracht voor Internet Explorer 10, werd overgeschakeld naar lettertypes om iconen weer te geven. Hierbij worden symbolen uit het lettertype weergegeven als iconen. Als gevolg hiervan werd de bijgeleverd iconenset beperkt tot het pictosfont waarmee zo nog vierennegentig iconen beschikbaar zijn.

Deze standaard iconenset bevatte geen passend icoontje voor de belknop. Gelukkig is het eenvoudig om zelf iconen toe te voegen, zie <http://docs.sencha.com/touch/2.2.1/#!/guide/theming-section-IconFonts>. Om de nodige iconen te voorzien werd gebruikt gemaakt van webapplicatie die te vinden is op <http://icomoon.io/app>. Met deze toepassing is het mogelijk de gewenste iconen te selecteren en deze te koppelen aan een karakter. Vervolgens maakt de applicatie hier een lettertype van waarvoor de verschillende font-bestanden dan beschikbaar zijn om te downloaden, zie figuur 7.17.



**Figuur 7.17:** Iconen selecteren, binden aan een karakter en downloaden als lettertype.

Om gebruik te kunnen maken van deze iconen moeten ze geïncludeerd worden in het SASS-bestand zodat ze mee gecompileerd worden in het CSS-bestand. Codefragment 7.8 toont hoe dit gedaan wordt. Een *icon* wordt geïncludeerd door een naam voor de *iconClass*, het karakter waaraan het icoon gekoppeld is en het lettertype waaruit dit icoon gehaald moet worden op te geven. Door een knop bijvoorbeeld een bepaalde *iconClass* te geven wordt het gekoppelde icoon gebruikt op de knop. Om gebruik te maken van het zelfgemaakte lettertype moet dit lettertype eerst geïncludeerd worden. Vervolgens kunnen de iconen uit dit lettertype ook gekoppeld worden aan een *iconClass*.

```
//Pictos
@include icon('refresh','O','Pictos');
@include icon('machines','Y','Pictos');
@include icon('mail','M','Pictos');
@include icon('home','H','Pictos');
@include icon('team','g','Pictos');
@include icon('settings','y','Pictos');
@include icon('arrow_left','[','Pictos');
@include icon('arrow_right',']','Pictos');
@include icon('user','U','Pictos');
@include icon('info','i','Pictos');

//Icomoon
@include icon-font('IcoMoon', inline-font-files('icomoon/icomoon.
woff', woff, 'icomoon/icomoon.ttf', truetype, 'icomoon/icomoon
.svg', svg));
@include icon('call','p','IcoMoon');
@include icon('maps','m','IcoMoon');
@include icon('info_white','i','IcoMoon');
@include icon('info_black','I','IcoMoon');
```

Codefragment 7.8: Iconen includeren in SASS.

Het SASS-bestand kan rechtstreeks gecompileerd worden naar CSS met de Compass-tool door het commando **compass compile /pad/naar/sass** uit te voeren in een command prompt. Wanneer de app *gebuilt* wordt gebeurt dit ook automatisch. *Builden* kan vanuit Architect of vanuit een command prompt door het commando **sencha app build** uit te voeren vanuit de locatie van de toepassing. Dit bouwproces wordt verder besproken in 7.7.

## 7.5 Navigatieprobleem oplossen

In 7.4.2 werd vermeld dat er een probleem opdook wanneer er twee keer na elkaar getikt werd op een contact in de contactenlijst. Hierdoor werd er een component teveel *gepushed* naar de container en liep de navigatie bij het terugkeren verkeerd. Dit werd eenvoudig opgelost door te luisteren naar het *itemsingletap*-event in plaats van het *itemtap*-event van de lijst. Hierna werd de applicatie uitvoerig getest om scenario's op te sporen waarin dit probleem zich nog voordeed. Hieruit bleek dat het probleem ook opdook wanneer een tik op een knop een view moet pushen. Helaas heeft een knop enkel een *tap*-event en geen *singletap*-event. Het eventsysteem van Sencha is hier dus voor verbetering vatbaar.

Bij knoppen kon dit opgelost worden door de knop voor een kort interval op *disabled* te zetten wanneer op de knop getikt wordt. Op een knop die uit staat kan niet getikt worden. Met behulp van de *setTimeout*-functie wordt de knop na een korte tijd, in de applicatie een halve seconde, terug aan gezet.

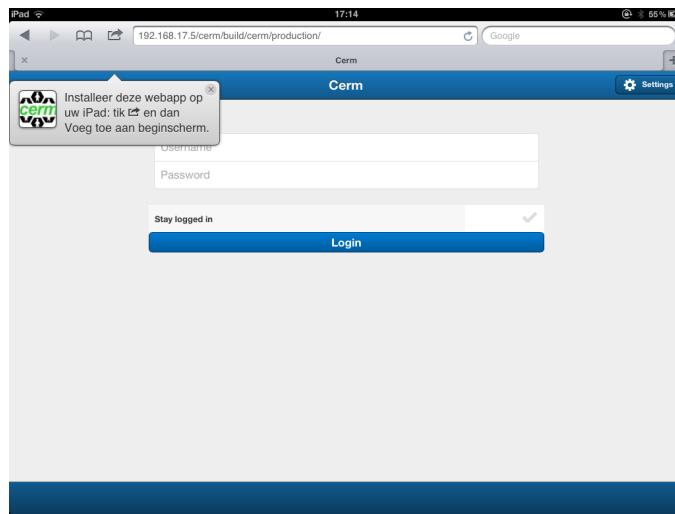
Hetzelfde probleem kwam ook voor bij de klantennaam waarbij een tik de klantgegevens naar voor brengt. Hiervoor wordt echter geen knop gebruikt en dus kunnen we die component ook niet uit zetten. De controller houdt hiervoor een variabele bij om een gelijkaardig oplossing te bieden. Dezelfde methode zou ook gebruikt kunnen worden bij de knoppen. Hierbij doet de controller gewoon een extra controle om de navigatie goed te laten verlopen. Met een *if*-structuur wordt gecontroleerd of de variabele op *false* staat. Als dit het geval is wordt in deze *if*-structuur de variabele op *true* gezet en wordt de view die *gepushed* moet worden aangemaakt. Hierna wordt de view *gepushed* naar de container en wordt de variabele opnieuw na een kort interval op *false* gezet.

## 7.6 De gebruiker melden hoe de app geïnstalleerd kan worden op een iOS-toestel

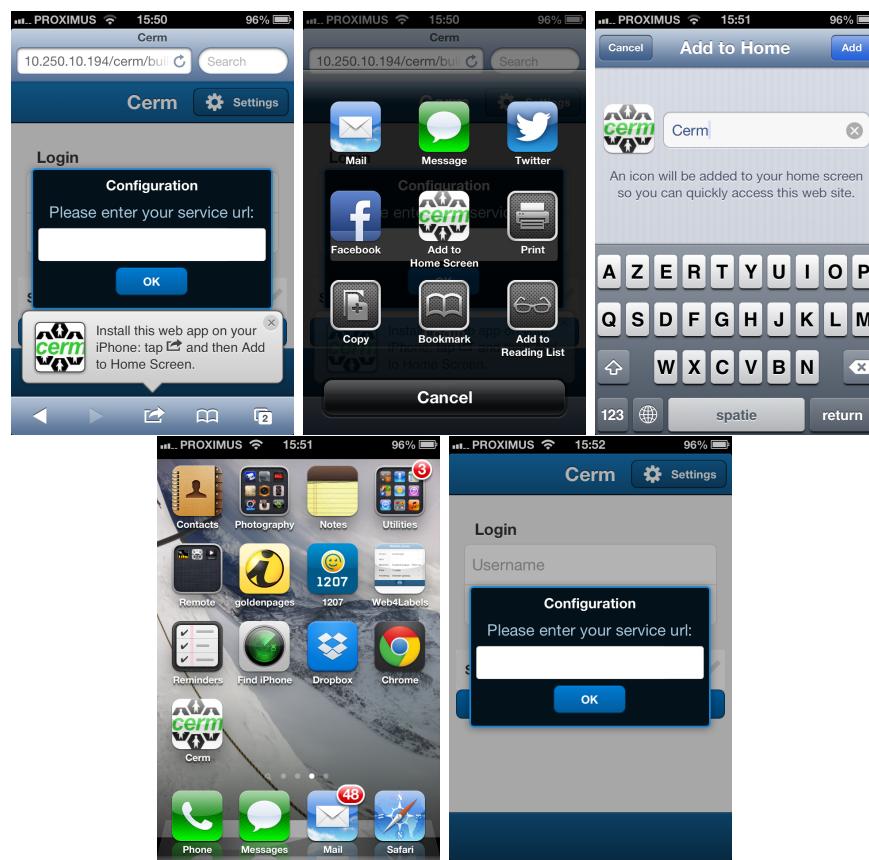
Webapps installeren op een iOS-toestel is heel eenvoudig, zie 3.2. Veel gebruikers weten dit echter niet en zeker voor webapplicaties is dit zeer handig. Daarom zou het mooi zijn om iOS-gebruikers hier op te wijzen wanneer de applicatie gestart wordt vanuit Safari.

Matteo Spinelli heeft hier een mooi script voor geschreven, Add2Home.js, dat vrij te gebruiken is onder een MIT-licentie. Op <http://cubiq.org/add-to-home-screen> staat uitgelegd hoe dit te gebruiken.

Om het JavaScript- en CSS-bestand mee te laten verwerken in het bouwproces moeten deze bestanden toegevoegd worden aan het app.json-bestand. Hoewel het niet door Sencha aangewezen wordt om aanpassingen te doen aan index.html blijkt dit voorlopig toch de enige oplossing. Als de manier gevuld wordt die Sencha prefereert gaat de functionaliteit verloren na het bouwproces. Het is niet duidelijk waarom het hier dan misloopt. Het is ook niet echt de moeite om dit verder te onderzoeken aangezien dit ook zeker geen slechte oplossing is. Het resultaat is te zien op figuur 7.18 en 7.19.



Figuur 7.18: Add2Home-melding op een iPad.



Figuur 7.19: De app installeren op een iPhone.

## 7.7 Distributie

Sencha voorziet met Sencha Cmd de mogelijkheid om een webapplicatie te optimaliseren voor productie. Het is natuurlijk perfect mogelijk om de applicatie beschikbaar te maken zoals ze ontwikkeld wordt. Toch ondergaat deze om verschillende redenen best eerst het bouwproces. Dit proces zorgt voor:

- minimalisatie van de *payload* die het netwerk opgestuurd moet worden;
- minimalisatie van het aantal HTTP-aanvragen dat vereist is;
- schnellere weergave van de interface wanneer de applicatie voor het eerst geladen wordt;
- *caching* van bronnen die niet frequent veranderen;
- minimalisatie van het netwerkverkeer bij het updaten naar een nieuwe versie.

Dit wordt gedaan door:

- een minimale lijst op te maken van de afhankelijkheden voor de applicatie zodat niet de volledig bibliotheek geïncludeerd moet worden;
- minimaliseren en samenvoegen van aparte JavaScript/CSS-bestanden in enkele bundels;
- gebruik maken van asynchrone laadmechanismen;
- lokaal bewaren van alle JavaScript/CSS-bestanden en hiervan gebruik maken bij de volgende keren dat de applicatie gestart wordt zodat geen netwerkconnectie nodig is;
- *delta's* genereren tussen alle versies van de applicatie en de cliënt automatisch de opdracht geven om deze *delta's* te downloaden en te patchen op de lokale kopieën.

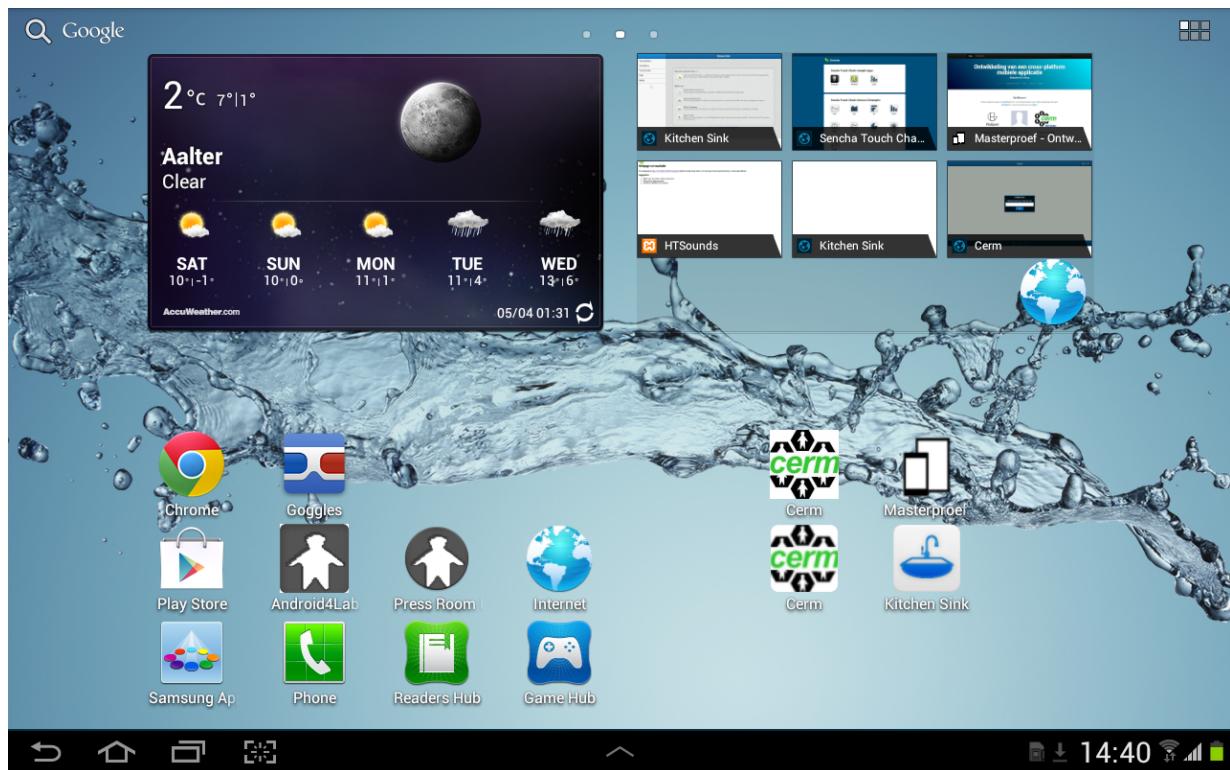
Dit wordt verder in detail besproken op een blogpost van Sencha die te lezen is op [www.sencha.com/blog/behind-sencha-command-and-the-build-process](http://www.sencha.com/blog/behind-sencha-command-and-the-build-process).

Het bouwproces kan gestart worden vanuit de Architect-omgeving of vanuit een command prompt door het commando **sencha app build** uit te voeren in de map van de applicatie. Het resultaat komt terecht in de map *build/AppNaam/Modus*. De standaard bouwmodus is *production*. De bouwmodus kan meegegeven worden als argument voor het bouwcommando. Met de bestanden die hier te vinden zijn kan de webapplicatie beschikbaar gemaakt worden op een webserver van Cerm of lokaal bij de drukkerij.

Het is ook mogelijk om deze bestanden te zippen en te uploaden naar de build-service van PhoneGap (<http://build.phonegap.com>) om hiermee een hybride applicatie te maken. Ook Architect biedt de mogelijkheid om een hybride applicatie te bouwen.

Vervolgens kan de applicatie beschikbaar gemaakt worden via de stores.

Voor Android kan de APK ook zelf verspreid worden. Voor iOS is daarvoor een inschrijving op het Enterprise Program van Apple nodig. Figuur 7.20 toont de webapp en de hybride app die toegevoegd zijn aan het startscherm van een Android-toestel.



**Figuur 7.20:** De webapp (onderaan) en de hybride app (bovenaan) toegevoegd aan het startscherm van een Android-tablet.

# Hoofdstuk 8

## De backend-ondersteuning voor de applicatie

In het vorige hoofdstuk werd de ontwikkeling van de applicatie besproken. Dit hoofdstuk handelt over de server en de daarop geïnstalleerde webservices waarop de applicatie steunt. Alle data in het Cermsysteem wordt op deze server verzameld in een databank. In volgende paragrafen komen de webservices en de configuratie van de server aan bod. Er zal ook uitgelegd worden waarom voor het JSON-formaat gekozen werd en hoe er gezorgd kan worden voor een veilige communicatie tussen client en server.

### 8.1 ServiceCounters

De webservice om gegevens van machines en tellers op te halen was reeds ontwikkeld om te consumeren met een native Android-applicatie. Deze webservice werd ontwikkeld met de WCF-technologie (Windows Communication Foundation) volgens de REST-architectuur. Ook voor de andere webservices werd voor deze technologie gekozen. Data wordt standaard getransfereerd in XML-formaat. Om de data in JSON-formaat te krijgen moet de headeroptie 'accept' toegevoegd worden aan het HTTP-request. Deze optie moet ingesteld worden op 'application/json'.

Bij het overzetten van deze webservice naar de IIS-server op de laptop waarop ontwikkeld werd was er een probleem bij het *deployen*. Het probleem was dat de service wordt opgestart met een *init*-bestand en dit bestand niet in de *build*-map zat. Hierdoor werd dit bestand niet mee gekopieerd naar de IIS-server. Door in de broncode te kijken werd dit probleem geïdentificeerd en werd de locatie van dit *init*-bestand gevonden. Dit stond niet gedocumenteerd. Het *init*-bestand bevat onder andere de connectiestring en de authenticatiegegevens voor de database.

Met deze webservice kan een lijst van de machines opgehaald worden. Vervolgens is het mogelijk om met een referentie van een machine alle tellergegevens op te vragen of enkel deze binnen een opgegeven interval. Het is ook mogelijk om meer gedetailleerde informatie op te vragen zoals bijvoorbeeld welke werknemer de machine bedient of voor welke order er gedrukt wordt, maar hier wordt in de applicatie nog geen gebruik van gemaakt. Tabel 8.1 geeft een overzicht van de mogelijke operaties die deze webservice aanbiedt. Deze kunnen met de HTTP-GET-methode aangesproken worden op volgende manier: <protocol>://<server\_name>:<port>/CermRestServices/ServiceCounters/<operatie>.

OPERATIE	BESCHRIJVING
Machines	Geeft een lijst van alle machines terug.
MachineCounterAll/{MachineReference}	Geeft alle tellerregistraties op een machine terug.
MachineCounter/{MachineReference}/ {Start}/{End}	Geeft alle tellerregistraties in het interval [Start-End] op een machine terug.
MachineCounterDetailsAll/{MachineReference}	Geeft alle detailinformatie van een teller terug.
MachineCounterDetails/{MachineReference}/ {Start}/{End}	Geeft alle detailinformatie van een teller in het interval [Start-End] terug.

**Tabel 8.1:** Operaties ServiceCounters.

### 8.1.1 Keuze voor het JSON-formaat

Zoals reeds vermeld biedt de webservice de keuze om de data in XML- of JSON-formaat te verkrijgen. Code 8.1 toont een voorbeeld van een lijst van machines die werd opgevraagd via de operatie 'Machines' in JSON-formaat. Code 8.2 toont dit in het XML-formaat. Beide voorbeelden zijn nu gestructureerd voor de leesbaarheid maar het is duidelijk dat het JSON-formaat veel minder tekens bevat. De grootte van een antwoord in JSON-formaat is kleiner dan hetzelfde antwoord in XML. Gebruik maken van het JSON-formaat biedt dus het voordeel dat er minder dataverkeer nodig zal zijn. Minder dataverkeer betekent lagere kosten, een sneller antwoord en ook een lager batterijverbruik voor het mobiele toestel.

```
[
{
  "Active": -1,
  "Name": "Flasheuse",
  "Reference": "1234"
},
{
  "Active": 1,
  "Name": "A1 - 4 KLEUREN produktielijn",
  "Reference": "5140"
},
{
  "Active": 0,
  "Name": "A2-6",
  "Reference": "5260"
}
]
```

**Codefragment 8.1:** Lijst van machines in JSON-formaat.

```
<ArrayOfCermMachine xmlns="http://schemas.datacontract.org
/2004/07/CermRestMachines.Business" xmlns:i="http://www.w3.org
/2001/XMLSchema-instance">
<CermMachine>
<Active>-1</Active>
<Name>Flasheuse</Name>
<Reference>1234</Reference>
</CermMachine>
<CermMachine>
<Active>1</Active>
<Name>A1 - 4 KLEUREN produktielijn</Name>
<Reference>5140</Reference>
<CermMachine>
<Active>0</Active>
<Name>A2-6</Name>
<Reference>5260</Reference>
</CermMachine>
</ArrayOfCermMachine >
```

**Codefragment 8.2:** Lijst van machines in XML-formaat.

JSON staat voor JavaScript Object Notation en is een syntax om informatie op te slaan of uit te wisselen in tekstvorm. JSON is kleiner dan XML, sneller en makkelijk te parsen. JSON is taalonafhankelijk ook al wordt de JavaScript syntax gebruikt. JSON-parsers en -bibliotheeken bestaan voor verschillende programmeertalen.

Het JSON-tekstformaat is syntactisch identiek aan de code voor het creëeren van een JavaScript-object. Omwille van deze overeenkomstigheid is het niet nodig om een parser te gebruiken in JavaScript. Hier kan gewoon de ingebouwde *eval*-functie gebruikt worden om de JSON-data uit te voeren om JavaScript-objecten aan te maken. Dit zal dus sneller gebeuren dan in de applicatie XML te moeten parsen.

Daarnaast is het feit het JSON-formaat minder data beslaat ook een groot voordeel.

Ter illustratie geeft tabel 8.2 een vergelijking tussen de hoeveelheid data die nodig is voor de contacten in JSON- en XML-formaat. Hieruit is op te maken dat dankzij het JSON-formaat ongeveer 40% minder data nodig is bij het versturen van de contactenlijst.

Meer informatie over JSON is te vinden op <http://www.w3schools.com/json/>.

Contacten	JSON(kB)	XML(kB)
600	94	160
12500	1800	3100
32000	3200	5300

**Tabel 8.2:** Vergelijking JSON versus XML.

## 8.2 Configuratie van de server

Om de applicatie goed te ondersteunen zijn een aantal serverconfiguraties noodzakelijk. Tijdens deze masterproef werd gebruik gemaakt van een IIS-server.

Voor eerst moet de *identity* van de *application pool* waartoe de webservices behoren ingesteld zijn op een account die ook rechten heeft op de database.

Ten tweede moeten ook de juiste MIME-types toegevoegd worden. Om de applicatie offline te kunnen gebruiken worden delen lokaal opgeslagen. Welke bestanden lokaal bewaard moeten worden staan opgegeven in het *manifest*. Hiervoor moet het MIME-type *text/cache-manifest* toegevoegd worden met de *appcache*-extensie. Dit manifest wordt automatisch door Sencha Touch aangemaakt. Meer info over offline web-apps en het manifest op [www.w3.org/TR/offline-webapps](http://www.w3.org/TR/offline-webapps).

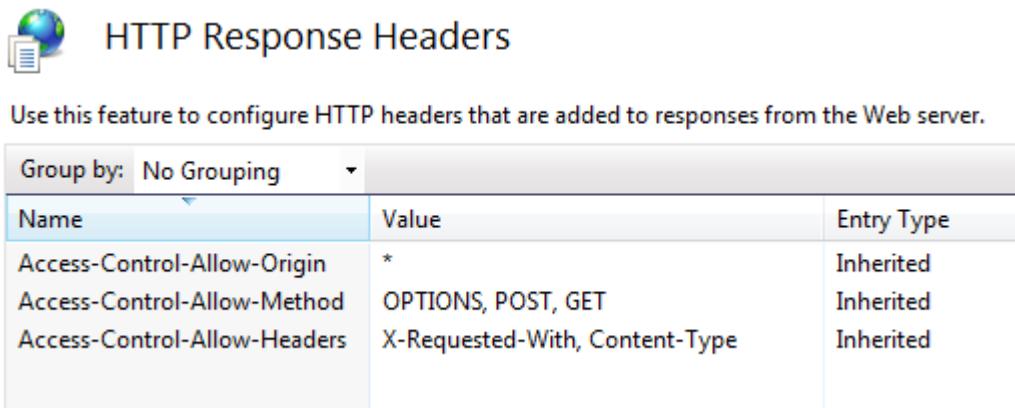
Omdat data getransfereerd wordt in het JSON-formaat moet ook het MIME-type *application/json* toegevoegd zijn met de *json*-extensie.

### 8.2.1 Cross Origin Resource Sharing

Omdat de applicatie geschreven wordt in JavaScript moet er rekening gehouden worden met de *Same-origin policy*. Dit beperkt de manier waarop documenten of scripts die geladen zijn van een bepaalde oorsprong kunnen interageren met een bron van een andere oorsprong.

De oorsprong is gelijk als het protocol, de poort en de host hetzelfde zijn voor verschillende bronnen. Om deze reden zal communicatie met de webservices wel lukken als de applicatie en de webservices op dezelfde server staan en via dezelfde poort en hetzelfde protocol beschikbaar gemaakt worden. In alle andere gevallen, wanneer de applicatie bijvoorbeeld lokaal op een mobiel toestel staat, zal dit niet werken. (Ruderman, 2013)

Om deze beperking te omzeilen bestaat er een *work-around* die JSON-P heet. Deze methode is echter half-duplex en kan enkel gebruikt worden om data op te halen. Op <http://www.json-p.org> is meer terug te vinden over deze methode. Het aanbevolen mechanisme voor dit probleem is CORS (Cross Origin Resource Sharing, <http://www.w3.org/TR/cors>). Om gebruik te maken van dit mechanisme moeten enkele HTTP-response headers toegevoegd worden voor de webservices, zie figuur 8.1.



The screenshot shows a configuration interface for 'HTTP Response Headers'. At the top, there's a globe icon and the title 'HTTP Response Headers'. Below that is a sub-instruction: 'Use this feature to configure HTTP headers that are added to responses from the Web server.' A dropdown menu 'Group by:' is set to 'No Grouping'. The main area is a table with three columns: 'Name', 'Value', and 'Entry Type'. The table contains the following rows:

Name	Value	Entry Type
Access-Control-Allow-Origin	*	Inherited
Access-Control-Allow-Methods	OPTIONS, POST, GET	Inherited
Access-Control-Allow-Headers	X-Requested-With, Content-Type	Inherited

**Figuur 8.1:** Toegevoegde HTTP-response headers.

De *Access-Control-Allow-Origin*-header wordt ingesteld op een sterretje om alle oorspronken toe te laten. Met de header *Access-Control-Allow-Method* wordt aangegeven dat de OPTIONS-, POST-, en GET-methode toegelaten worden. Tot slot worden de *X-Requested-With*- en *Content-Type*-headers toegelaten door deze in te stellen in *Access-Control-Allow-Headers*. De *X-Requested-With*-header wordt door de meeste JavaScript-frameworks ingesteld op *XmlHttpRequest* en de *Content-Type*-header wordt gebruikt om aan te geven dat er JSON-data verzonden wordt.

## 8.3 ServiceLogin

Om de applicatie te kunnen gebruiken moet de gebruiker eerst inloggen door zijn gebruikersnaam en wachtwoord op te geven. Deze gegevens worden als een JSON-object naar de webservice gepost die vervolgens de nodige controles uitvoert. Codefragment 8.3 toont een loginrequest met een verkeerd wachtwoord. De service stuurt altijd een JSON-object terug met de gebruikersnaam en -referentie die overeenkomen met de logingegevens. Bestaat de gebruiker niet of is het wachtwoord verkeerd dan zijn die velden lege strings, zie code 8.4. Het teruggestuurde object bevat ook waarden die de rechten beschrijven die de gebruiker heeft, zie codefragmenten 8.4 en 8.5.

```
Request URL:http://localhost//CermRestServices/ServiceLogin/Login
Request Method:POST
Status Code:200 OK
Request Headersview source
Accept:application/json
Accept-Encoding:gzip, deflate, sdch
Accept-Language:en-US,en;q=0.8
Connection:keep-alive
Content-Length:51
Content-Type:application/json
Host:localhost
Origin:http://localhost
Referer:http://localhost/Cerm/?deviceType=Phone
User-Agent:Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/27.0.1453.94 Safari/537.36
Request Payload:{Name:benjamin, Password:verkeerdwachtwoord}
```

**Codefragment 8.3:** HTTP-POST naar de loginservice.

```
{
  UserName: "",
  UserReference: "",
  ViewCounters: false,
  ViewCustomers: false
}
```

**Codefragment 8.4:** Resultaat verkeerde logingegevens.

```
{
  UserName: "benjamin",
  UserReference: "100154",
  ViewCounters: true,
  ViewCustomers: true
}
```

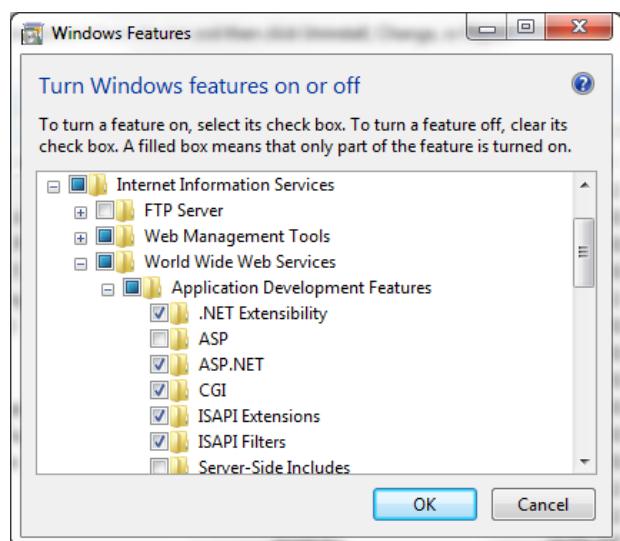
**Codefragment 8.5:** Resultaat juiste logingegevens.

### 8.3.1 Bijkomende serverconfiguratie

De service die de login verzorgt maakt gebruik van externe DLL's die mee geïnstalleerd worden met de Cermsofware. Hierdoor zijn er extra serverconfiguraties nodig om deze loginservice te laten werken.

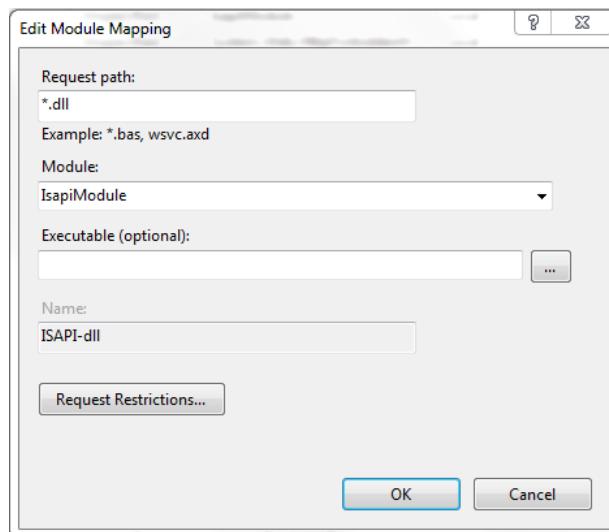
Vooreerst moet in de instellingen van de *application pool* van de webservice het gebruik van 32-bit applicaties aangezet worden.

Er moeten ook enkele Windows features toegevoegd worden, zie figuur 8.2. De nodige features zijn *CGI* (Common Gateway Interface), *ISAPI Extensions* en *ISAPI Filters* (Internet Server Application Programming Interface).



**Figuur 8.2:** Toegevoegde Windows features.

Vervolgens moet er voor de server ook een *handler mapping* toegevoegd worden voor DLL's met de IsapiModule (figuur 8.3). Tenslotte moeten de *ISAPI and CGI restrictions* op *allowed* ingesteld worden.



**Figuur 8.3:** Handler mapping instellen.

## 8.4 ServiceContacts

Ook de contactenmodule steunt op een eigen webservice. Deze webservice kan aangesproken worden om alle contacten van een gebruiker op te halen. Er zijn ook operaties om detailgegevens op te halen van een klant of een contactpersoon, zie tabel 8.3. Alle operaties kunnen met de HTTP-GET-methode aangesproken worden op volgende manier: <protocol>://<server\_name>:<port>/CermRestServices/ServiceContacts/<operatie>.

OPERATIE	BESCHRIJVING
CustomerContacts/{UserReference}	Geeft een lijst van alle contacten van de gebruiker met de opgegeven referentie.
Contact/{CustomerReference}/{ContactReference}	Geeft alle gegevens van een contactpersoon terug.
Customer/{Reference}	Geeft alle gegevens van een klant terug

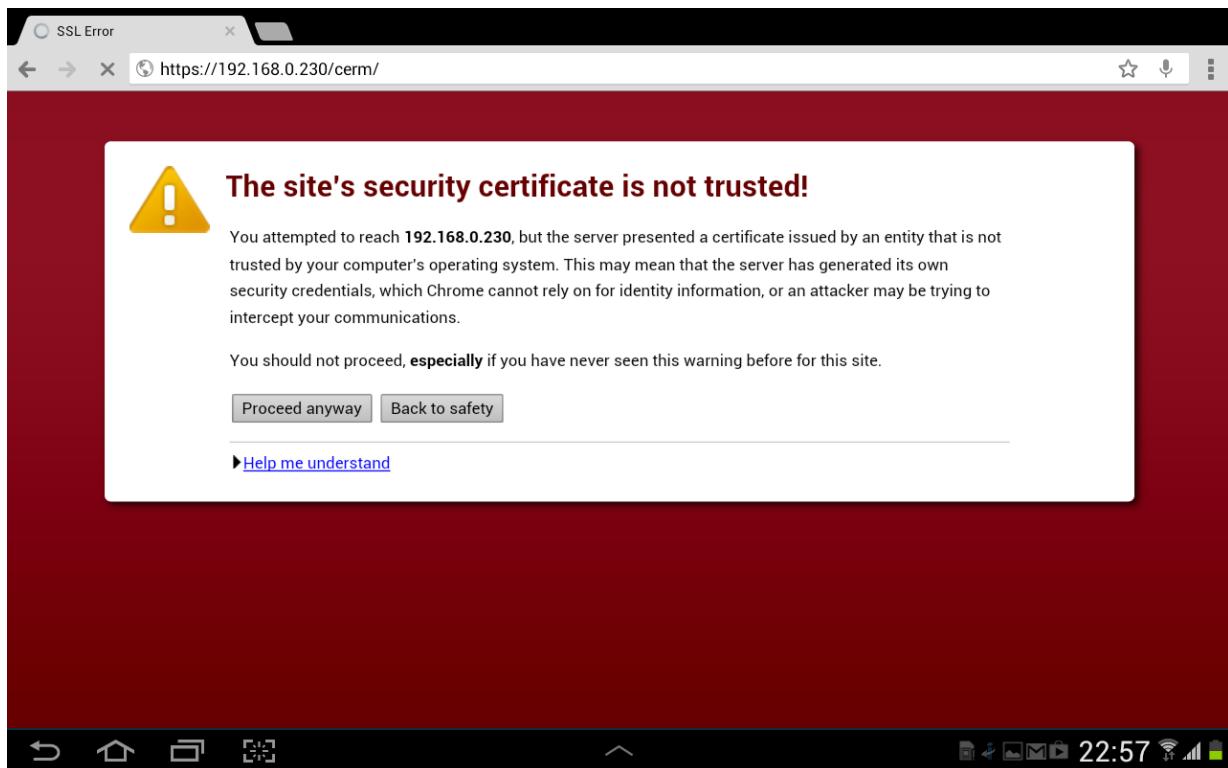
Tabel 8.3: Operaties ServiceContacts.

## 8.5 Zorgen voor een beveiligde verbinding

In 7.3 en 8.3 werd respectievelijk de front- en de back-end van de login besproken. De logingegevens worden hierbij gepost naar de loginservice. Als dit bericht onderschept wordt, kunnen de logingegevens hier eenvoudig uit afgelezen worden. Om dit te voorkomen moet het dataverkeer via een beveiligd communicatiekanaal lopen. Hiervoor wordt gebruik gemaakt van SSL (Secure Sockets Layer). Daarom moet de SSL/HTTPS-service geconfigureerd worden op de server. Hiervoor is een certificaat nodig voor het coderen en decoderen van de gegevens die via het netwerk verzonden worden. Om de HTTPS-service in te stellen kan het artikel op <http://support.microsoft.com/kb/324069/nl> gevolgd worden.

Belangrijk is dat het certificaat getekend is door een certificeringsinstantie die vertrouwd wordt door het besturingssysteem. Is dit niet het geval dan moet deze CA (Certificate Authority) toegevoegd worden aan de vertrouwde instanties van het besturingssysteem.

Android is hier blijkbaar minder strikt in dan iOS. Wanneer bijvoorbeeld gebruik gemaakt wordt van een zelfgetekend certificaat en de gebruiker surft in de browser via HTTPS naar de applicatie, dan wordt er zowel op Android als op iOS gemeld dat dit certificaat niet vertrouwd wordt, zie figuur 8.4. Hierop kan de gebruiker er voor kiezen om toch door te gaan en zal het invullen van de service-url met het HTTPS-protocol ook werken voor de communicatie met de webservices.



Figuur 8.4: Melding van een certificaat dat niet vertrouwd wordt op Android.

Als de applicatie gestart wordt buiten een browseromgeving dan zijn er verschillen tussen Android en iOS. Bij het gebruiken van de hybride applicatie op Android wordt er geen melding gedaan dat het certificaat niet vertrouwd wordt, toch lukt het om de service-url in te stellen met het HTTPS-protocol en hier gebruik van te maken. Op iOS lukt dit echter niet wanneer de applicatie gestart wordt via het starticoontje. Het iOS-besturingssysteem laat geen zelfgetekende certificaten toe om *man-in-the-middle*-aanvallen te vermijden. Het is mogelijk om certificaten te installeren, maar als dit certificaat niet getekend is door een vertrouwde CA, is dit niets waard. Voor iOS is het dus noodzakelijk dat een vertrouwde certificeringsinstantie het certificaat tekent. Om zelf vertrouwde CA's toe te voegen kan gebruik gemaakt worden van de *iPhone Configuration Tool*, beschikbaar voor Mac en Windows, en moet het iOS-toestel verbonden zijn met de computer.  
(Arnott, 2013)(Guyton, 2012)(Felker, 2011)

# Hoofdstuk 9

## Conclusie

In deze masterproef werd onderzoek gedaan naar de mogelijkheden om platformonafhankelijke mobiele applicaties te ontwikkelen. Als eerste kwamen de verschillende bestaande en toekomstige besturingssystemen aan bod. Hierbij werden de ondersteunde technologieën, die vooral gebruikt worden voor native applicaties, vermeld. Opmerkelijk was de trend naar ondersteuning van platformonafhankelijke oplossingen voor de ontwikkeling van apps. HTML5 of een beschikbare Android runtime zijn hierbij de oplossingen die door de ontwikkelaars van besturingssystemen aangeboden worden.

Vervolgens werden webapplicaties besproken. De HTML5-standaard biedt nieuwe mogelijkheden voor het ontwikkelen van functionele en gebruiksvriendelijke webtoepassingen. Extra aandacht werd besteed aan de ondersteuning van webapps in iOS. Dit besturingssysteem zorgt voor een mooie integratie van deze webtoepassingen en enkele toekomstige besturingssystemen zullen hierin volgen. Voor de ontwikkeling van mobiele webapplicaties werden verschillende frameworks vermeld. Hiervan werden jQuery Mobile en Sencha Touch verder besproken. Daarnaast kwamen ook enkele ontwikkelingstools aan bod.

Na webapplicaties werd aandacht besteed aan hybride applicaties. Hiervoor bestaan verschillende oplossingen. De voor- en nadelen zijn dan ook afhankelijk van welke soort hybride applicatie gekozen wordt. Ook de functionaliteit, gebruiksvriendelijkheid, platformonafhankelijkheid en performantie kan afhankelijk van de methodiek zeer uiteenlopend zijn. Verschillende frameworks voor de ontwikkeling van hybride apps werden summier besproken.

Voor de realisatie van deze masterproef werd gekozen om gebruik te maken van Sencha Touch. Met dit framework werd een webtoepassing ontwikkeld die steunt op client-side technologie. De applicatie zelf kan dus lokaal op een mobiel toestel geïnstalleerd worden en kan in principe volledig offline werken. Enkel voor data moet de applicatie communiceren met een server. Hoewel een gratis commerciële licentie beschikbaar is werd de Sencha Touch bundel aangeschaft omwille van de ontwikkelingstools en de commerciële licentie voor de grafieken. Hierdoor moet geen beroep gedaan worden op een externe bibliotheek voor grafieken (zoals bijvoorbeeld HighChart of ZingChart) en is de integratie eenvoudig. De gemaakte keuze biedt verder nog vele voordelen.

Het eerste voordeel is dat er veel platformen ineens ondersteund kunnen worden. Ook de toekomstige besturingssystemen zullen daar bij horen. De applicatie wordt ontwikkeld volgens de HTML5-standaarden. Hierdoor kan de applicatie *future proof* genoemd worden. Testen kan in de eerste fase gewoonweg met een desktopbrowser. Het is ook niet nodig om de applicatie voor elk platform afzonderlijk te compileren.

Een tweede voordeel is dat het relatief eenvoudig is om de webapplicatie om te vormen naar een hybride applicatie. Eventuele functionele tekortkomingen kunnen hiermee opgevangen worden. Dit biedt ook verschillende mogelijkheden om de applicatie te distribueren. Hybride apps kunnen via de app stores verspreid worden, webapps kunnen zonder de stores verspreid worden en moeten dus ook geen goedkeuringssprocedures doorlopen. Voor elk platform kan hierbij de beste keuze gemaakt worden.

Een derde voordeel is dat de HTML5-technologie verder evolueert en ook meer en beter ondersteund zal worden. HTML5 is *hot* en de vooruitgang die werd waargenomen tijdens de relatief korte tijdspanne van deze masterproef is veelbelovend. Browsers en JavaScript-engines worden steeds krachtiger. Ook de ontwikkelingsframeworks bieden steeds meer mogelijkheden en performantieverbeteringen. Niet onbelangrijk zijn ook de tools die beschikbaar zijn voor ontwikkelaars.

Tot slot biedt het Sencha Touch framework zelf ook heel wat voordelen voor de ontwikkeling en het gebruik van de webapplicatie. Door de beschikbaarheid van verschillende grafische componenten die geoptimaliseerd zijn voor aanraakschermen bijvoorbeeld kan de interface sneller opgebouwd worden en moet er minder aandacht besteed worden aan het design. De MVC-architectuur en het klassensysteem zorgen ervoor dat de code overzichtelijk en onderhoudbaar gehouden kunnen worden en het framework voorziet ook verschillende mechanismen voor performantieverbeteringen. Hierbij kan alvast het delta-updatesysteem, het build-mechanisme en het laadsysteem genoemd worden.

De applicatie zelf bevat een login, een module voor de productie en een contactenmodule. Inloggen kan met behulp van het HTTPS-protocol op een beveiligde manier gebeuren. De machinesmodule stelt de gebruiker in staat om te zien welke drukpersen actief zijn. Voor elke machine is het mogelijk om de tellergegevens te raadplegen, zowel voor de huidige productie als die in het verleden. De grafieken kunnen uitvergroot worden en met een paar presetknoppen is het mogelijk om snel op de vast ingestelde tijdsspannen in te zoomen. De applicatie past zijn interface aan voor tablets en smartphones. Alle beschikbare functionaliteit op een tablet is ook beschikbaar op een smartphone dankzij een dynamische interface.

De contactenmodule kan gebruikt worden om klant- en contactgegevens te raadplegen. Met een speciaal bedieningselement, dat gekend is vanop het iOS-platform, kan snel door een grote lijst contacten gebladerd worden. De contacten moeten telkens wanneer de applicatie opgestart wordt éénmalig geladen worden. Het offline bewaren van contacten wordt nog niet ondersteund. De applicatie biedt de mogelijkheid om gegevens interactief te gebruiken. Zo kan e-mail- en navigatiefunctionaliteit vanuit de applicatie gestart worden. Bellen behoort ook tot de mogelijkheden. Deze functionaliteiten worden enkel aangeboden als ze mogelijk zijn.

Aan de gebruikerservaring werd ook gedacht door bedieningen zoals *pull-to-refresh* en *pinch-to-zoom*, die typisch zijn voor aanraakschermen, te gebruiken. Wanneer het nodig is worden laadschermen getoond en de gebruiker kan aangeven dat de applicatie zijn gegevens mag onthouden zodat inloggen niet steeds nodig is. Op iOS wordt de gebruiker er in de browser op gewezen hoe de applicatie geïnstalleerd kan worden. Er is ook een mechanisme voorzien om verschillende talen te ondersteunen.

Voor de uitwisseling van data met de webservices werd gekozen voor het JSON-formaat. Hierdoor is er minder dataverkeer nodig en kunnen de gegevens sneller omgezet worden in objecten wat voordelig is voor de performantie en het batterijverbruik.

Zowel de webapp als de hybride app werden uitvoerig getest. De hybride app werd enkel getest op Android omdat deze het eenvoudigst zonder tussenkomst van een store geïnstalleerd kan worden. De webapp werd getest op Android(verschillende smartphones en tablets), iOS (iPhone en iPad), Windows Phone 8 (Nokia Lumia 620) en verschillende desktopbrowsers (Internet Explorer 10, Safari en Chrome). Op de mobiele toestellen bleek de performantie en gebruikerservaring op iPad en iPhone uitstekend, deze benadert mede dankzij de mooie integratie de gebruikerservaring van native applicaties.

Op Android is er een grotere fragmentatie op gebied van hardware en software.

Performantie en gebruikerservaring zijn hierbij afhankelijk van de hardware, de versie van het besturingssysteem en de gebruikte browser. Een belangrijke factor hierbij is een vlotte overschakeling naar de GPU voor het grafische gedeelte van de app. Dit is natuurlijk zowel van de hardware, het besturingssysteem, als van de browser afhankelijk. Android is met de ondersteuning van grafische hardwareacceleratie begonnen sinds Android 3.0, de versie die ook gekend staat als Honeycomb en enkel voor tablets bedoeld was. Het verschil tussen browsers werd duidelijk bij het testen op een Samsung Galaxy Tab 2.0. Deze tablet heeft een 1Ghz dual-core processor en 1Gb RAM-geheugen. De standaardbrowser biedt daar een minder goede performantie en gebruikerservaring dan Chrome. De native webview die gebruikt wordt voor de hybride applicatie maakt van dezelfde engine gebruik als de standaardbrowser. Vanaf Android 4.2 wordt Chrome de standaardbrowser voor het besturingssysteem. Grote verbeteringen zijn merkbaar op toestellen met een quad-core processor en Android 4.1+. Toestellen die getest werden zijn de Sony Xperia P, de Samsung Galaxy SIII en SIV en de Nexus 7 van Asus. Op deze toestellen komt de performantie en gebruikerservaring overeen met die van op iOS.

Volgens de onderzoeksresultaten die terug te vinden zijn op de blog van Sencha mag ook een goede performantie en gebruikerservaring verwacht worden op BlackBerry 10. Uit de cijfers op <http://html5test.com/results/mobile.html> blijkt ook dat BlackBerry 10 de beste HTML5-ondersteuning biedt op dit moment. Sencha werkt aan ondersteuning voor Firefox en kondige ondersteuning voor het Tizen-besturingssysteem aan. De Tizen-browser scoort op de HTML5-test een 492/500 en zal dus de BlackBerry-browser voorbijsteken bij lancering. Ubuntu mobile zal hoogstwaarschijnlijk ook meteen ondersteund kunnen worden. Tot slot kan gesteld worden dat alle besturingssystemen die een Android runtime hebben ook ondersteund kunnen worden.

## 9.1 Mogelijke uitbreidingen

Tijdens de ontwikkelingsfase kwamen er al snel verschillende ideeën op om uit te breiden of te verbeteren. Sommige, zoals bijvoorbeeld de mapsfunctionaliteit, werden meteen uitgevoerd. De lijst met mogelijke uitbreidingen en verbeteringen is natuurlijk zeer lang en zal met de verdere evolutie van de HTML5-technologie alleen maar langer worden.

In de productiemodule kan er bijvoorbeeld voor gezorgd worden dat de grafiek rood kleurt wanneer de drukpers afval produceert, groen bij goede producten en blauw wanneer hier geen aanduiding voor is. Wanneer WebSockets meer ondersteund wordt kan de automatische vernieuwing hiermee ook verbeterd worden. Zo zou het mogelijk zijn om data naar de app te pushen in plaats van gebruik te maken van een *polling*-mechanisme. Statistieken weergeven over de productie zou ook een interessante feature zijn.

De contactenmodule kan uitgebreid worden door offline opslag van contacten te ondersteunen. Hierbij zou de gebruiker dan zelf kunnen kiezen wanneer de contacten gesynchroniseerd moeten worden. Hoe dit gedaan wordt, is vooral afhankelijk van het feit of Safari IndexedDB zal gaan ondersteunen. Indien dit niet het geval zou zijn dan moet zowel IndexedDB als WebSQL ondersteund worden in de applicatie. Een andere mogelijkheid is om als hybride applicatie gebruik te maken van de native API's.

Zoekfunctionaliteit toevoegen is ook een mogelijke uitbreiding, hiervoor moet wel eerst onderzocht worden hoe dit het efficiëntst, web of hybrid, gerealiseerd kan worden. Een server-side oplossing is desnoods misschien ook een mogelijkheid.

Een algemene verbetering zou het ondersteunen van een terugknop zijn door middel van *history support*. Nu is het enkel mogelijk om terug te keren in de applicatie door een terugknop die voorzien wordt door de applicatie zelf. Het is ook mogelijk om de staat van de applicatie te bewaren voor wanneer de applicatie een volgende keer opgestart wordt.

Tenslotte kunnen natuurlijk nog altijd modules toegevoegd worden. Hier moet wel bij opgelet worden dat de applicatie niet te groot wordt en dat dit niet ten koste gaat van de performantie. Dankzij het modulair ontwerp is het echter eenvoudig om van een module een alleenstaande applicatie te maken.

# Lijst van afkortingen

**AJAX** Asynchronous JavaScript And XML

**API** Application Programming Interface

**APK** Application Package

**CA** Certificate Authority

**CGI** Common Gateway Interface

**CMS** Content Management System

**CORS** Cross-Origin Resource Sharing

**DLL** Dynamic Link Library

**DOM** Document Object Model

**HTML** HyperText Markup Language

**HTTPS** HyperText Transfer Protocol Secure

**IDE** Integrated Development Environment

**IIS** Internet Information Service

**ISAPI** Internet Service Application Programming Interface

**JS** JavaScript

**JSON** JavaScript Object Notation

**MIS** Management Information System

**MVC** Model-View-Controller

**OS** Operation System

**PDA** Personal Digital Assistant

**PHP** Hypertext Preprocessor

**RAD** Rapid Application Development

**RDP** Remote Desktop Protocol

**REST** Representational State Transfer

**RIA** Rich Internet Application

**RPC** Remote Procedure Call

**SASS** Syntactically Awesome StyleSheet

**SDK** Software Development Kit

**SSL** Secure Socket Layer

**UI** User Interface

**URL** Uniform Resource Locator

**VM** Virtual Machine

**W3C** World Wide Web Consortium

**WCF** Windows Communication Foundation

**XAML** Extensible Application Markup Language

**XML** Extensible Markup Language

**XHR** XMLHttpRequest

# Lijst van figuren

1.1	Vereenvoudigde voorstelling van de workflow van een grafisch bedrijf.	11
2.1	Verschil tussen de compilatie van een Java- en een Androidapplicatie.	17
2.2	Microsoft presenteert Windows en Windows Phone 8. Links is de UI van Windows Phone te zien, rechts de UI van Windows 8.	19
2.3	Ubuntu op tv, desktop, tablet en smartphone.	21
3.1	HTML5-logo ( <a href="http://www.w3.org/html/logo">http://www.w3.org/html/logo</a> )	26
3.2	Presentatie iMac in 1998.	28
3.3	De Financial Times app installeren op iOS.	29
3.4	De grafische interface van Sencha Architect.	36
4.1	De architectuur van een PhoneGap-applicatie (Wargo, 2012)	38
4.2	Een PhoneGap-applicatie voor meerdere platformen compileren (Wargo, 2012)	39
4.3	Scheiding UI en gemeenschappelijke code ( <a href="http://www.xamarin.com">www.xamarin.com</a> )	40
6.1	De mapstructuur van de applicatie.	48
6.2	De algemene architectuur van de applicatie.	49
6.3	Het componentensysteem waarin alle componentklassen afgeleid worden van de hoofdklasse <i>Component</i> . Een pijl van A naar B wil zeggen dat B afgeleid is van A. (Bershadskiy,2013)	51
7.1	De algemene interface van de applicatie.	52
7.2	Een LineChart (links) en een AreaChart (rechts) met dezelfde data.	55
7.3	Profiles kunnen niet toegevoegd worden in de project inspector van Architect.	57
7.4	De MachineView op een Samsung Galaxy Tab 2.0.	58
7.5	De MachineView op een Sony Xperia P.	59
7.6	De dynamische toolbar op een smartphone in portretmodus. De knoppen om een zoomvenster in te stellen zijn niet zichtbaar.	60
7.7	De dynamische toolbar op een smartphone in landschapmodus. De knoppen om een zoomvenster in te stellen zijn zichtbaar.	61
7.8	De instellingen op smartphone en tablet.	62
7.9	De locatie voor de extensie.	62
7.10	Inloggen in de applicatie.	65
7.11	De contactenlijst.	67

7.12 Contactgegevens . . . . .	68
7.13 Het scherm na het gebruiken van de belknop op een Windows Phone en iPhone. . . . .	71
7.14 Het scherm na het gebruiken van de mailknop op een Android-toestel. . . . .	71
7.15 De mapview van de applicatie. . . . .	72
7.16 De Google Maps app gestart vanuit de applicatie op Android. . . . .	73
7.17 Iconen selecteren, binden aan een karakter en downloaden als lettertype. . . . .	74
7.18 Add2Home-melding op een iPad. . . . .	77
7.19 De app installeren op een iPhone. . . . .	77
7.20 De webapp (onderaan) en de hybride app (bovenaan) toegevoegd aan het startscherm van een Android-tablet. . . . .	79
8.1 Toegevoegde HTTP-response headers. . . . .	84
8.2 Toegevoegde Windows features. . . . .	86
8.3 Handler mapping instellen. . . . .	86
8.4 Melding van een certificaat dat niet vertrouwd wordt op Android. . . . .	88

# Lijst van tabellen

2.1	Vergelijking soorten applicaties . . . . .	23
2.2	Soorten applicaties en programmeertalen . . . . .	24
7.1	Schatting van het aantal contacten en hun grootte in kilobytes. . . . .	67
8.1	Operaties ServiceCounters. . . . .	81
8.2	Vergelijking JSON versus XML. . . . .	83
8.3	Operaties ServiceContacts. . . . .	87

# Lijst van codefragmenten

3.1	Twee 'jQuery Mobile'-pagina's in de body van één HTML-document . . . . .	30
6.1	Definieren van de modelklasse Machine . . . . .	50
6.2	Instantierien van een Machine-object . . . . .	50
7.1	Definitie van de MachineList . . . . .	54
7.2	De profielklasse voor tablets . . . . .	57
7.3	Een voorbeeld JSON-bestand voor Engelse vertaling . . . . .	63
7.4	Een terugknop internationaliseren . . . . .	63
7.5	Implementatie van onCustomerContactListItemSingletap . . . . .	69
7.6	Links om te mailen en te bellen . . . . .	70
7.7	De link juist instellen om de navigatieapplicatie te starten . . . . .	73
7.8	Iconen includeren in SASS. . . . .	75
8.1	Lijst van machines in JSON-formaat. . . . .	82
8.2	Lijst van machines in XML-formaat. . . . .	82
8.3	HTTP-POST naar de loginservice. . . . .	85
8.4	Resultaat verkeerde logingegevens. . . . .	85
8.5	Resultaat juiste logingegevens. . . . .	85

# Literatuurlijst

Allen S., Graupera V. & Lundrigan L. (2010).

*Pro Smartphone Cross-platform Development:*

*iPhone, Blackberry, Windows Mobile and Android Development and Distribution.*

Apress.

Arnott, N. (2013).

*IPhone Apps Accepting Self-Signed SSL Certificates*

Laatst geraadpleegd op 16 mei 2013

via <http://www.neglectedpotential.com/2013/01/sslol>

Apple (2009).

*Getting Started With iOS Web Apps.*

Safari Developer Library.

Apple Inc.

Laatst geraadpleegd op 1 mei 2013

via [https://developer.apple.com/library/safari/#referencelibrary/GettingStarted/GS\\_iPhoneWebApp/\\_index.html](https://developer.apple.com/library/safari/#referencelibrary/GettingStarted/GS_iPhoneWebApp/_index.html)

Apple (2012)

*Apple URL Scheme Reference*

iOS Developer Library

Apple Inc.

Laatst geraadpleegd op 16 mei 2013

via [http://developer.apple.com/library/ios/#featuredarticles/iPhoneURLScheme\\_Reference/Introduction/Introduction.html#/apple\\_ref/doc/uid/TP40007891-SW1](http://developer.apple.com/library/ios/#featuredarticles/iPhoneURLScheme_Reference/Introduction/Introduction.html#/apple_ref/doc/uid/TP40007891-SW1)

Ashworth, S. (2013).

*Localising Sencha Touch and Ext JS applications with Ux.locale.Manager:*

*Using the Ux.locale.Manager extension*

Laatst geraadpleegd op 15 april 2013

via <http://www.swarmonline.com/2013/02>

- Bershadskiy, S. (2013).  
*Sencha Touch:*  
*Open-Source HTML5 Mobile Web Development Made Easy.*  
DZone Inc.  
Laat geraadpleegd op 9 april 2013  
via <http://refcardz.dzone.com/refcardz/sencha-touch>
- Bohn, D. (2011).  
*iOS: a visual history.*  
The Verge.  
Laatst geraadpleegd op 11 maart 2013  
via <http://www.theverge.com/2011/12/13/2612736/ios-history-iphone-ipad>.
- Clark, J.E. & Johnson, B.P. (2012).  
*Sencha Touch Mobile JavaScript Framework:*  
*Build web applications for Apple iOS and Google Android touchscreen devices with this first HTML5 mobile framework.*  
Birmingham: Packt Publishing.
- Drucker, S. & Perry, M. (2013)  
*Teach Yourself Sencha Complete.*  
Fig Leaf.  
Laatst geraadpleegd op 14 mei 2013  
via <http://training.figleaf.com/tutorials/senchacomplete>
- Felker, M. (2011).  
*Secure iPad Deployment*  
Laatst geraadpleegd op 16 mei 2013  
via [http://www.tomsitpro.com/articles/ipad\\_iphone\\_security-iphone\\_configuration\\_unility\\_2-179.html](http://www.tomsitpro.com/articles/ipad_iphone_security-iphone_configuration_unility_2-179.html)
- Garcia, J., De Moss, A. & Simoens, M. (2013, negende versie).  
*Sencha Touch in Action.*  
Manning Publications.
- Gargenta, M (2011, eerste editie).  
*Learning Android.*  
Sebastopol: O'Reilly.
- Gartner. (2013).  
*Gartner Says by 2016, More Than 50 Percent of Mobile Apps Deployed Will be Hybrid.*  
Persbericht 4 februari 2013  
via <http://www.gartner.com/newsroom/id/2324917>

Guyton, N. (2012).  
*Adding trusted root certificate authorities to iOS*  
Laatst geraadpleegd op 16 mei 2013  
via <http://nat.guyton.net/2012/01/20>

Kosmaczewski, A. (2012, eerste editie).  
*Mobile JavaScript Application Development: Bringing Web Programming to Mobile Devices.*  
Sebastopol: O'Reilly.

Reid, J. (2011, eerste editie).  
*jQuery Mobile: Build Cross-platform Mobile Applications.*  
Sebastopol: O'Reilly.

Ruderman, J. (2013).  
*Same-origin policy*  
Mozilla Developer Network.  
Laatst geraadpleegd op 9 april 2013  
via [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Same\\_origin\\_policy\\_for\\_JavaScript](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Same_origin_policy_for_JavaScript)

Scapplehorn, S. (2012).  
*Marmalade SDK Mobile Game Development Essentials.*  
Birmingham: Packt Publishing.

Villa, C. & Gonzalez, A. (2013).  
*Learning Ext JS 4: Sencha Ext JS for a beginner.*  
Birmingham: Packt Publishing.

Wargo, M. J. (2012).  
*PhoneGap Essentials: Building Cross-Platform Mobile Apps.*  
Addison-Wesley.

Wokke, A. (2013).  
*Tablet- en telefoonbeurs MWC: de aanval op Android.*  
Tweakers.  
Laatst geraadpleegd op 24 maart 2013  
via <http://tweakers.net/reviews/2953/2>.

