KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Vergelijkende studie van raamwerken voor de ontwikkeling van mobiele HTML5-applicaties

Tim Ameye
Sander Van Loock

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdspecialisatie Veilige software en hoofdspecialisatie Gedistribueerde systemen

Promotor:
Prof. dr. ir. E. Duval

Assessoren:
Prof. dr. M. Denecker
F. Van Assche

Begeleider:
Ir. G. Parra

Academiejaar 2012 – 2013

Dankwoord Tim

Ik wil in eerste plaats mijn promotor prof. E. Duval en het bedrijf Capgemini bedanken voor het interessante onderwerp. Daarnaast wil ik G. Parra bedanken voor de wekelijkse meetings en de vele inzichten die hij verschafte doorheen het academiejaar. Ook het nalezen van de gehele thesis tezamen met S. Charleer verdient een woordje van dank. Vervolgens was er ook begeleiding vanuit Capgemini door J. Verhulst en J. Persoons die naast hun job tijd maakten voor het ondersteunen van deze thesis. Ook wil ik de lezers van onze thesisblog bedanken voor de inspirerende commentaren die ze verschaften.

In het bijzonder dank ik mijn thesispartner Sander die samen met mij deze thesis tot een goed einde heeft gebracht. Ik bedank ook Koen voor de vele diepe gesprekken en verlichtende momenten. Niet enkel hen, maar ook vele andere goede vrienden zowel binnen als buiten de richting Computerwetenschappen dienen te worden beloond met een welgemeende dankjewel. Daarnaast wil ik mijn vriendin Marjan bedanken omdat ze er steeds was voor mij. Als laatste maar daarom niet minder belangrijk dank ik mijn ouders, want zonder hen was dit nooit mogelijk geweest.

Dankwoord Sander

Bij het maken van deze thesistekst hebben letterlijk en figuurlijk bloed, zweet en tranen gevloeid. Het resultaat na één jaar hard werken mag dan ook gezien worden. Meerdere mensen hebben hier natuurlijk toe bijgedragen. Eerst zou ik prof. E. Duval willen bedanken voor het interessante onderwerp dat hij in samenwerking met Capgemini beschikbaar stelde. Hierbij wil ik ook Jan en Jannik van Capgemini bedanken om deze thesis mee te ondersteunen. Verder was de hulp van onze begeleider, Gonzalo, onmisbaar. Dear Gonzalo, thanks for all the comments, remarks and opinions that you gave us every week. These definitely brought the quality of this text to a higher level.

De steun van mijn ouders zorgde ervoor dat ik een heel jaar ongestoord kon werken. Ook bedankt voor de steun die jullie mij gaven om mijn diploma te halen. Maaike, ik weet dat je me moest missen als ik tijdens het weekend aan mijn thesis moest werken. Ik vind het super dat je zo geduldig was en hoop alle verloren tijd snel in te halen! Ook is een eervolle vermelding van Gunther Desamblanx op zijn plaats. Samen met Simon de producer zorgde je elke weekdag tussen 22u en 24u voor de meest productieve uren.

Tot slot wil ik mijn thesispartner, Tim, bedanken voor het geweldige jaar dat we samen hebben meegemaakt. Jouw stipheid, nauwkeurigheid en gedrevenheid zijn niet te onderschatten! Ik zal onze nachtelijke discussies nooit vergeten. Je was een fijne collega, ik wens je het allerbeste voor je doctoraat!

Inhoudsopgave

Samenvatting

Ontwikkelaars van mobiele applicaties worden geconfronteerd met een variëteit aan mobiele besturingssystemen die op smartphones en tablets aanwezig zijn. Dit komt doordat een applicatie dient te worden geprogrammeerd aan de hand van een Software Development Kit (SDK) die specifiek is voor het besturingssysteem. De ontwikkeling van mobiele webapplicaties, gebruikmakend van HTML5, is een mogelijke oplossing hiervoor. Om het ontwikkelingsproces van deze mobiele HTML5-applicaties te versnellen worden raamwerken aangeboden. Deze helpen zowel bij het toevoegen van functionaliteit als de elementen voor de gebruikersinterface.

Door de variëteit aan mobiele HTML5-raamwerken dringt een grondige vergelijking zich op. Dit werk vergelijkt Sencha Touch, Kendo UI, jQuery Mobile en Lungo op basis van vijf vergelijkingscriteria: populariteit, productiviteit, gebruik, ondersteuning en performantie. Populariteit kijkt naar de activiteit van de raamwerken op sociale netwerken. Productiviteit wordt opgemeten om aan te tonen hoe lang het duurt om met een raamwerk vertrouwd te raken en er daadwerkelijk iets mee te maken. Het gebruik van de raamwerken bekijkt de elementen die het raamwerk aanbiedt. Ondersteuning test de elementen van het raamwerk op acht verschillende mobiele apparaten. Er wordt een evenwichtige keuze gemaakt tussen Android, iOS, smartphone en tablet. Performantie meet enerzijds de downloadtijd en anderzijds de gebruikerservaring. De laatstgenoemde bepaalt hoe vlot het gaat om door een lange lijst te scrollen.

De vergelijking toont aan dat jQuery Mobile het beste raamwerk is. Daarna volgen Kendo UI, Lungo en Sencha Touch. jQuery Mobile heeft als belangrijkste troef de hoge productiviteit doordat het enerzijds zeer goed gedocumenteerd is en anderzijds geen ontwerppatroon afdwingt. Dit laatste is echter ook een nadeel omdat het hierdoor minder scoort op gebruik. Kendo UI heeft als troef het gebruik doordat het een ontwerppatroon afdwingt. Het scoort echter ondermaats op performantie. Lungo behaalde enkel de maximumscore voor performantie doordat QuoJS, de JavaScript-bibliotheek van Lungo, geoptimaliseerd is voor mobiele apparaten. Sencha Touch is het minst productief en performant. Daarentegen scoort Sencha Touch het best op het vlak van gebruikerservaring. Door het afdwingen van een ontwerppatroon scoort het quasi evengoed als Kendo UI op vlak van gebruik. Alle onderzochte raamwerken scoren zeer goed op ondersteuning op de onderzochte mobiele apparaten.

Abstract

Developers of mobile applications are confronted with a variety of mobile operating systems that are present on smartphones and tablets. This is because an application has to be programmed with a Software Development Kit (SDK) specific for the operating system. Developing mobile web applications using HTML5 can be a solution for this problem. To speed up development, frameworks are presented. These help to add functionality and elements for the user interface.

Because of the variety of mobile HTML5 frameworks, a thorough comparative study is necessary. This thesis compares Sencha Touch, Kendo UI, jQuery Mobile and Lungo based on five comparison criteria: popularity, productivity, usage, support and performance. Popularity looks at the activity of frameworks on social networks. Productivity is determined by measuring how long it takes to get acquainted with the framework and build something useful. Next, usage of the framework is studied by looking which elements are offered by the framework. Support tests the elements of the framework on eight different devices. A good balance is made between Android, iOS, smartphone and tablet. Performance measures the download time and user experience. The latter is determined by the smoothness of scrolling through a long list.

The comparison shows that jQuery Mobile is the best framework followed by Kendo UI, Lungo and Sencha Touch. jQuery Mobile is highly productive, a major advantage because on the one hand it is well-documented and on the other hand it does not impose an architecture. The latter, however, becomes a disadvantage when looking at the usage. Kendo UI has usage as most important advantage because it imposes an architecture. However, it scores inferior on performance. Lungo only achieves a maximum score for performance because QuoJS, the underlying JavaScript library, is optimized for mobile devices. Sencha Touch is the least productive and performant. By contrast, Sencha Touch scores the best in user experience. By enforcing an architecture, it scores almost as good as Kendo UI regarding usage. All frameworks score good for support on the evaluated mobile devices.

Lijst van figuren

Lijst van tabellen

Lijst van afkortingen

AHP	Analytic Hierarchy Process
AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
ASP	Active Server Pages
BS	BesturingsSysteem
CORS	Cross-Origin Resource Sharing
CPU	Central Processing Unit
CRUD	Create Read Update Destroy
CSS(3)	Cascading Style Sheets (3)
DOM	Document Object Model
EDGE	Enhanced Data Rates for GSM Evolution
(G)GI	(Grafische) GebruikersInterface
GNU GPL	GNU's Not Unix General Public License
GPRS	General Packet Radio Service
GPS	Global Positioning System
GPU	Graphics Processing Unit
HSDPA	High-Speed Downlink Packet Access
HTML(5)	HyperText Markup Language (5)
IDE	Integrated Development Environment
ISO	International Organization for Standardization
jQM	jQuery Mobile
JSON(P)	JavaScript Object Notation (with Padding)
JSP	JavaServer Pages
KUI	Kendo UI
LTE	Long Term Evolution
L	Lungo
MIT	Massachusetts Institute of Technology
MVC	Model-View-Controller

MVVM	Model-View-View Model
OEM	Original Equipment Manufacturer
PCAP	Packet CAPture
PDA	Personal Digital Assistant
PDF	Portable Document Format
PHP	PHP: Hypertext Preprocessor
POC	Proof Of Concept
PPI	Pixels Per Inch
QR	Quick Response
RAM	Random Access Memory
RWD	Responsive Web Design
SASS	Syntactically Awesome StyleSheets
SDK	Software Development Kit
SEO	Search Engine Optimization
SMS	Short Message Service
ST	Sencha Touch
USB	Universal Serial Bus
UTMS	Universal Mobile Telecommunications System
W3C	World Web Consortium
WHATWG	Web Hypertext Application Technology Working Group
WORA	Write Once, Run Anywhere
WYSIWYG	What You See Is What You Get
X(HT)ML	eXtensible (HyperText) Markup Language

Hoofdstuk

1 Inleiding

1.1 Achtergrondinformatie

Het gebruik van smartphones en tablets stijgt ontzettend snel in onze samenleving. Voorheen was er de *feature phone* waarop enkel voorgeïnstalleerde applicaties konden worden gebruikt. Nu kunnen smartphones en tablets ook extra applicaties vanuit een winkel downloaden en installeren. Ontwikkelaars van deze mobiele applicaties worden geconfronteerd met de variëteit aan mobiele besturingssystemen die op deze apparaten aanwezig zijn. Dit komt doordat een applicatie dient te worden geprogrammeerd aan de hand van een Software Development Kit (SDK) die specifiek is voor het besturingssysteem. Ontwikkelaars zullen dus eenzelfde applicatie in verschillende programmeertalen dienen te programmeren om een zo groot mogelijk publiek te bereiken. Niet enkel het programmeren, maar ook het onderhoud van de applicaties in verschillende programmeertalen brengt een grote kost met zich mee.

Een oplossing hiervoor is het maken van een mobiele webapplicatie, gebruikmakend van HyperText Markup Language 5 (HTML5). Ten eerste wordt deze rechtstreeks in een webbrowser geopend en dus niet langer vanuit een winkel geïnstalleerd. Dit betekent dus dat ieder mobiel apparaat dat een webbrowser heeft, de webapplicatie kan openen ongeacht zijn mobiel besturingssysteem. Ten tweede wordt de applicatie slechts in één programmeertaal geschreven, wat de kost verlaagd. Om het ontwikkelingsproces van deze mobiele HTML5-applicaties te versnellen, worden raamwerken aangeboden die helpen bij het toevoegen van functionaliteit van de applicatie en de elementen voor de gebruikersinterface (GI).

1.2 Probleembeschrijving

Mobiele HTML5-raamwerken zijn er in overvloed en ook de verschillende versies van eenzelfde raamwerk volgen elkaar in snel tempo op. In de huidige literatuur worden er vaak raamwerken aangehaald en besproken, maar niet vergeleken. Indien deze toch worden vergeleken, gebeurt dit vaak subjectief of worden punten gegeven zonder een gestaafde methode te gebruiken. Ook bestaat er geen literatuur die vergelijkingen van mobiele HTML5-raamwerken aggregeert.

1.3 Doelstellingen

Deze thesistekst bestaat uit twee doelstellingen. Een eerste doel is het definiëren van een methodologie om HTML5-raamwerken met elkaar te vergelijken. Deze methodologie moet alle belangrijke aspecten van de raamwerken tegen het licht houden. Ook moet er geprobeerd worden de werkwijze zo objectief mogelijk te laten verlopen en het resultaat van de studie op een eenvoudige, visuele manier aan de lezer te presenteren. Het tweede doel omvat de effectieve vergelijking van de raamwerken zelf. Door de grote verscheidenheid van HTML5-raamwerken moeten de bestudeerde raamwerken zo worden gekozen dat ze zoveel mogelijk aspecten bevatten. Hier komt ook de afweging tussen het aantal bestudeerde raamwerken en de diepte van de vergelijkende studie, de kop op steken. De raamwerken die worden gekozen moeten vervolgens worden vergeleken met de vooropgestelde methodologie. Het resultaat moet alle positieve en negatieve aspecten van de raamwerken bevatten. Vervolgens moet er gekeken worden of er één raamwerk het beste is of er verschillende raamwerken in bepaalde situaties als beste kunnen worden bestempeld.

1.4 Toepassingsgebied

Mobiele HTML5-raamwerken vergemakkelijken de ontwikkeling van mobiele HTML5-applicaties. Deze applicaties zijn toegankelijk via het web en geoptimaliseerd om op mobiele apparaten te kunnen werken. Het aanspreken van mobiele applicaties via het web heeft zowel voor- als nadelen zeker wanneer er wordt vergeleken met *native* of hybride applicaties. De focus van deze studie ligt echter niet op het onderzoeken van deze vergelijking. Wel zullen de verschillende technologieën besproken worden om uitsluitend mobiele applicaties te maken.

Omdat Capgemini deze thesis mee ondersteunt zullen de onderzochte raamwerken vanuit een bedrijfscontext worden benaderd. Dit zal vooral naar boven komen in de keuze van vergelijkingscriteria en de methode om deze criteria te testen.

1.5 Overzicht

Eerst wordt de basis van dit werk in hoofdstuk 2 uitgelegd. Vervolgens worden de vier gekozen raamwerken in hoofdstuk 3 uitvoerig besproken. Daarna zullen de gekozen vergelijkingscriteria in hoofdstuk 4 aan bod komen en verantwoord worden. Hierna wordt de vergelijking in hoofdstuk 5 uitgevoerd op de gekozen raamwerken aan de hand van de gekozen criteria. Als laatste wordt het besluit in hoofdstuk 6 geformuleerd.

Hoofdstuk

2 Literatuurstudie

Dit hoofdstuk bekijkt welke mobiele apparaten er allemaal bestaan (2.1). Vervolgens wordt er gekeken wat er onder de motorkap van deze apparaten zit, namelijk welke mobiele besturingssystemen (2.2), welke mobiele applicaties (2.4) en welke mobiele webbrowsers (2.3) er bestaan. Daarna komen de drie bouwblokken van het web aan bod (2.5), namelijk HTML, CSS en JavaScript. Hierna wordt ingegaan op bestaande mobiele HTML5-raamwerken (2.6). Ten slotte worden reeds bestaande manieren om software en raamwerken te vergelijken, bekeken (2.7).

2.1 Mobiele apparaten

Mobiele apparaten bestaan in alle soorten en maten, met weinig of veel opties, voor weinig of veel geld zoals wordt geïllustreerd op figuur 2.1. Het verdient daarom de aandacht om deze diversiteit onder de loep te nemen. Eerst wordt ingegaan op twee soorten mobiele apparaten, namelijk de smartphone en tablet [33]. Daarna wordt stilgestaan bij de kenmerken van deze apparaten [94]. Andere mobiele apparaten zoals de *e-reader* en Personal Digital Assistant (PDA) worden buiten beschouwing gelaten.

2.1.1 Soorten

Sinds de voorstelling van de Apple iPhone in 2007 [19], stijgt het gebruik van de smartphone ontzettend snel in onze samenleving. Momenteel zijn er al meer dan één miljard smartphones in gebruik [141] en dit zal tegen 2015 verdubbeld zijn [35]. Foto's of video's nemen, navigeren naar het dichtstbijzijnde restaurant of nog snel het weer voor de komende dagen opzoeken, het is allemaal mogelijk. Hoewel Apple de lat hoog heeft gelegd met het uitbrengen van de iPhone, zijn er ook nog andere spelers op de markt. Zo zijn er bijvoorbeeld de op Google's Android-gebaseerde smartphones zoals de Nexus 4 en de op Windows Phone gebaseerde smartphones zoals de Nokia Lumia 800.

De tweede soort mobiele apparaten zijn tablets. Tegen 2016 zullen er 760 miljoen tablets in gebruik zijn [35]. Ook hier kan terug gedacht worden aan één van Apple's succesvolle producten, namelijk de in 2010 uitgebrachte iPad [7]. Er dient echter wel

Figuur 2.1: Verschillende mobiele apparaten. De bovenste rij zijn tablets met v.l.n.r. iOS en Android als besturingssysteem. De onderste rij zijn smartphones met v.l.n.r. Symbian, Windows Phone, Android, iOS en BlackBerry OS als besturingssysteem [38].

opgemerkt te worden dat Microsoft tien jaar voordien al een tablet had uitgebracht met veel minder succes [78].

2.1.2 Kenmerken

Door de vele verschillende soorten mobiele apparaten, is het nodig om op een hoog niveau te bekijken over welke kenmerken deze allemaal (kunnen) beschikken. Bij deze bespreking wordt ingegaan op de voornaamste kenmerken van smartphones en tablets. De kenmerken en bijhorende uitleg zijn gebaseerd op Dutson [94].

Resolutie en PPI

Een eerste kenmerk waar vooral Apple met haar Retina graag mee uitpakt, is de resolutie. Dit is het aantal pixels getoond op het beeldscherm en wordt uitgedrukt in breedte × hoogte. Hoe kleiner, hoe minder er op het scherm kan worden getoond. Dit is vooral belangrijk wanneer veel informatie op het scherm wordt getoond. Bij een kleine resolutie dient er gescrold te worden om de rest van de informatie te zien.

Indien er naast de resolutie ook nog eens rekening wordt gehouden met de fysieke grootte van het scherm, wordt er gesproken over pixels per inch (PPI). De eerste iPhone had een resolutie van 320×480 en een 3,5” scherm, wat neerkomt op 163 PPI. De iPhone 4 (Retina) daarentegen heeft een resolutie van 640×960 en een 3,5” scherm,

wat neerkomt op 326 PPI. Als er meer pixels op dezelfde fysieke grootte zijn geplaatst, dan zal het beeld scherper zijn.

Aanraakscherm

De populaire soorten schermen zijn resistieve en capacitieve aanraakschermen. De eerstgenoemde soort maakt gebruik van twee lagen die gescheiden worden door een tussenruimte. Door druk ontstaat er contact tussen de twee lagen. Meestal wordt bij deze soort schermen een stilus meegeleverd. De laatstgenoemde soort maakt gebruik van veranderingen in frequentie. Door het scherm aan te raken met een vinger, dat een geleider is, ontstaat er een kleine verandering in frequentie die gedetecteerd wordt.

Camera

Praktisch ieder recent mobiele apparaat is uitgerust met een camera. Sommige bevatten zelfs twee camera's. De camera vooraan is veelal van mindere kwaliteit en wordt gebruikt om videogesprekken te voeren. Achteraan het apparaat zit dan een camera met hogere resolutie om foto's van betere kwaliteit te kunnen maken.

Twee voorbeelden van toepassingen van de camera zijn toegevoegde realiteit (*augmented reality*) en het inscannen van barcodes. Bij het eerstgenoemde kijkt de gebruiker met de camera van zijn mobiel apparaat naar de wereld. Op het scherm wordt het beeld live getoond en wordt hieraan extra informatie toegevoegd. Zo kan bijvoorbeeld een mobiel apparaat dat naar een persoon wordt gericht, ervoor zorgen dat er extra informatie over deze persoon verschijnt.

Het laatstgenoemde wordt bijvoorbeeld gebruikt om de Quick Response Code (QR-code) in te scannen en te zien wat deze betekent. Zo een code kan tekst, een link naar een website, een telefoonnummer, enz. bevatten.

Verbinding

Mobiele apparaten zijn in staat met elkaar en met het Internet verbonden te zijn. Enerzijds kan gebruik worden gemaakt van mobiel Internet en anderzijds van draadloos Internet. Daarnaast kan data ook uitgewisseld worden tussen apparaten, zonder hiervoor het Internet te gebruiken. Een voorbeeld hiervan is Bluetooth dat een draadloze verbinding op korte afstand opzet.

Mobiel Internet Om dit in perspectief te plaatsen wordt eerst even teruggegaan in de tijd van mobiele communicatie. In de jaren '80 was er 1G, de eerste generatie mobiele telefonie, waarbij gesprekken analoog werden verzonden. 2G gebruikte daarentegen een digitaal netwerk, waarbij nieuwe diensten zoals Short Message Service (SMS) beschikbaar werden [74].

Bij de overgang van 2G naar 3G worden General Packet Radio Service (GPRS) en Enhanced Data Rates for GSM Evolution (EDGE) gesitueerd. GPRS is een uitbereiding op 2G waarbij data over het mobiele netwerk kan worden uitgewisseld.

Omdat niet alle telecomproviders hun hardware wilden aanpassen naar 3G, gebruikten ze EGDE dat een uitbreiding is op GPRS en grotere snelheden mogelijk maakt [67].

3G maakt gebruik van een volledig nieuwe technologie, namelijk Universal Mobile Telecommunications System (UTMS). Het is dus niet langer een uitbreiding op het bestaande netwerk zoals dat voor GPRS het geval was. De vraag naar bandbreedte blijft, waardoor High-Speed Downlink Packet Access (HSDPA) werd ontwikkeld. Het is een nieuwe implementatie van UTMS die nog grotere snelheden toelaat [67].

Op dit moment zijn telecomoperatoren volop bezig met de uitrol van 4G. Dit moet nog grotere snelheden mogelijk maken door gebruik te maken van LTE Advanced, een verbetering van Long Term Evolution (LTE).

Draadloos Internet Net zoals laptops draadloos verbinden met Internet via WiFi, kan dit ook op mobiele apparaten. Vaak is de kost van mobiel Internet groter dan die van draadloos Internet, waardoor dit type van verbinding de voorkeur krijgt.

GPS

Met het Global Positioning System (GPS) kan de gebruiker zijn locatie opvragen en doorgeven aan een applicatie om zo bijvoorbeeld het dichtstbijzijnde restaurant te vinden. Doordat het wat kan duren vooraleer de locatie is vastgesteld via GPS, kan het mobiel apparaat ook gebruik maken van mobiele masten of het Internet om zo, hetzij minder nauwkeurig, sneller de locatie te bepalen.

2.2 Mobiele besturingssystemen

Net zoals er een brede waaier bestaat aan besturingssystemen voor computers, geldt dit ook zo voor mobiele apparaten. In deze sectie wordt een overzicht gegeven van mobiele besturingssystemen met een significant marktaandeel [19, 41], namelijk iOS en Android. iOS en Android haalden respectievelijk 14,9% en 75,0% in het derde kwartaal van 2012. Vervolgens wordt ook Windows Phone besproken dat 2% van het marktaandeel binnenhaalt in datzelfde kwartaal [95].

Bij ieder besturingssysteem zal ook de winkel besproken worden waar applicaties kunnen worden gevonden. Net zoals programma's geïnstalleerd worden op een computer, kan dit ook voor mobiele apparaten. Deze aangeboden applicaties worden verzameld in een winkel waarna de gebruiker ze, al dan niet gratis, kan downloaden en installeren. De winkel kan de kwaliteit van deze applicaties hoog houden door regels op te leggen aan de ontwikkelaars (zie 2.4).

Zoals op figuur 2.1 te zien is, verschillen mobiele besturingssystemen ook door hun grafische gebruikersinterface (GGI). Deze worden hieronder niet besproken, maar het is duidelijk dat iOS, Android en Windows Phone er verschillend uitzien. Ook de manier waarop de gebruiker gaat interageren met het mobiel besturingssysteem is verschillend. Beide verschillen worden gebundeld in de term *native look-and-feel* van het mobiel besturingssysteem.

iOS	Marktaandeel (%)
6.1.x	81.2
5.1.x	8.2
6.0.x	8.0
5.0.x	1.2
4.3.x	1.4

Android	Marktaandeel (%)
2.3	38.5
4.0.x	27.5
4.1.x	26.1
2.2	3.7
4.2.x	2.3
2.1	1.7
3.2	0.1
1.6	0.1

Tabel 2.1: Marktaandeel van iOS-versies op 8 mei 2013 en Android-versies op 1 mei 2013. [116, 5].

2.2.1 iOS

Het iPhone-besturingssysteem van Apple is voor het eerst uitgekomen in juni 2007 tezamen met de iPhone. Later werd het iPhone OS en uiteindelijk iOS. Het is duidelijk dat iOS gebonden is aan de hardware van Apple. Verschillende versies volgden elkaar op: iOS 2 (juli 2008), iOS 3 (juni 2009), iOS 4 (juni 2010) en iOS 5 (oktober 2011) [22, 94].

De nieuwste versie, iOS 6, werd uitgegeven in september 2012. Voorbeelden van nieuwigheden zijn de sterkere integratie van sociale media zoals Facebook en Twitter en het gebruik van spraakcommando's [22]. Op tabel 2.1 is te zien dat 90% van de iOS-gebruikers al iOS 6 gebruikt.

Browsen op het web gebeurt met de standaard webbrowser Mobile Safari (zie 2.3). Applicaties worden gedownload van de App Store, de winkel van Apple die sinds iOS 2 aanwezig is [22]. Op deze winkel is er een sterkte controle op de kwaliteit van de aangeboden applicaties door de ontwikkelaars zeer strikte regels op te leggen [6].

2.2.2 Android

Android Inc. werd opgericht in 2003 en werd in 2005 overgekocht door Google Inc [103]. Het is net zoals iOS een mobiel besturingssysteem, maar in tegenstelling tot iOS is het *open-source* [19]. De eerste stabiele versie, Android 1.0, kwam uit in september 2008. Ook hier volgden verschillende versies elkaar op: Android 2.0 (oktober 2009), Android 3.0 (februari 2011) en Android 4.0 (oktober 2011) [103]. Hun nieuwste versie, Android 4.2, werd aangekondigd in oktober 2012 [104].

Op tabel 2.1 is het marktaandeel te zien van de verschillende Android-versies, waargenomen over een periode van 14 dagen. Het is duidelijk dat Android 2.x bijna de helft van het marktaandeel inneemt. Applicaties worden gedownload van Google Play, de winkel van Android. De controle op de kwaliteit van de aangeboden applicaties is minder strikt dan die van Apple. De applicaties moeten aan een aantal algemene regels voldoen, maar het basisconcept is dat van machtigingen [3]. Als een ontwikkelaar in zijn applicatie bijvoorbeeld wil gebruik maken van de SMS-berichten

van de gebruiker, zal hij eerst de machtiging moeten krijgen van de gebruiker. Dit gebeurt net voor de installatie van de applicatie op Google Play. Indien de gebruiker hier niet mee akkoord gaat, kan hij de applicatie niet downloaden [4]. Android bevat de Android browser of Chrome als standaard browser (zie 2.3).

2.2.3 Windows Phone

Windows Phone van Microsoft werd aangekondigd in oktober 2010 als vervanging voor Windows Mobile [106, 68]. Dit is duidelijk te zien als er gekeken wordt naar de versies: de laatste versie was Windows Mobile 6.5.3 en de eerste versie is Windows Phone 7. In 2011 ging Microsoft een partnerovereenkomst aan met Nokia om zo snel de markt te kunnen overwinnen [77]. De nieuwste versie, Windows Phone 8, werd aangekondigd in oktober 2012 [97]. Applicaties worden gedownload van de winkel genaamd Windows Phone Store. Microsoft legt, net zoals Apple, regels op aan de ontwikkelaars voor ze hun applicaties in de winkel kunnen publiceren [75].

2.3 Mobiele webbrowsers

Zoals Hales [41] uitlegt wordt sinds 2008 gesproken van het mobiele web. Vanuit mobiele webbrowsers wordt het web meer en meer aangesproken. Deze mobiele webbrowsers vormen als het ware kleine besturingssystemen, waardoor de browser zelf een platform wordt. Ze geven namelijk toegang tot allerlei kenmerken van het mobiele apparaat zoals camera en GPS. Denk maar aan het heel concreet voorbeeld van Google die het besturingssysteem Chrome OS maakte op basis van de Chrome webbrowser.

Vanuit het standpunt om webapplicaties te maken, is het dan ook zeer belangrijk om deze evolutie op te volgen. Een webbrowser haalt namelijk webpagina's op die geschreven zijn in HTML en andere technologieën. Doordat deze technologieën evolueren (zie 2.5), zullen de webbrowsers zelf ook mee (moeten) evolueren. Niet iedere browser zal dit op dezelfde manier doen, waardoor er verschillen zullen ontstaan. Het is namelijk ongewenst dat een webapplicatie enkel op de webbrowser van iOS werkt als de ontwikkelaar een zo breed mogelijk publiek wenst te bereiken.

In deze sectie worden drie mobiele webbrowsers besproken. Eerst komen de twee meest populaire browsers aan bod, namelijk Mobile Safari en de Android browser. Daarna wordt Internet Explorer Mobile kort besproken. Het marktaandeel van de genoemde browsers wordt getoond in tabel 2.2.

Zowel Mobile Safari als de Android browser zijn beide op de WebKit-browser *engine* gebaseerd [87]. Een *engine* zorgt ervoor dat de code van de opgehaalde webpagina wordt omgezet naar de webpagina die de gebruiker te zien krijgt.

Mobile Safari

Deze webbrowser van Apple zit standaard bij iOS en kan ook enkel op dit besturingssysteem worden gebruikt. Apple probeert om telkens de laatste nieuwe specificaties van HTML5 in zijn webbrowsers te implementeren [41]. Verder biedt Apple vanaf

Mobiele webbrowser	**Marktaandeel (%)**
Mobile Safari	59.42
Android browser	22.89
Chrome	2.63
Internet Explorer Mobile	1.64
Andere	13.42

Tabel 2.2: Marktaandeel mobiele webbrowsers in mei 2013 [83].

2010 geen ondersteuning meer voor Adobe Flash [2] op hun iPods, iPhones en iPads [49].

Android browser

Android biedt de Android browser aan. Implementatie van de HTML5-specificaties hebben wat aangesleept, maar vanaf Android 4.0 gaat dit een stuk beter [41]. Daarnaast is het nu ook mogelijk om de Chrome webbrowser op mobiele apparaten te installeren.

Internet Explorer Mobile

Net zoals bij het Windows besturingssysteem Internet Explorer wordt bijgegeven, geldt dit ook voor hun mobiel besturingssysteem. Bij de nieuwe Windows Phone 8 zal Internet Explorer Mobile 10 worden meegeleverd. Deze gebruikt dezelfde *engine* als Internet Explorer 10. WebSockets, Web Workers, Application Cache en IndexedDB worden hierin ondersteund [41]. Deze kenmerken worden in meer detail uitgelegd in sectie 2.5.

2.4 Mobiele applicaties

Er zijn drie mogelijkheden om mobiele applicaties te maken [1, 41]. Eén aanpak is het maken van een webapplicatie. Deze wordt geprogrammeerd gebruikmakend van een webtechnologie zoals HTML5 (zie 2.5) en wordt geopend vanuit een webbrowser.

Een andere aanpak is een *native* applicatie. Deze wordt geprogrammeerd in een SDK specifiek aan het besturingssysteem. Daarna wordt de applicatie opgeladen naar een winkel (bv. App Store). De ontwikkelaar laadt zijn applicatie op naar de winkel, waarna de winkel deze kan onderwerpen aan een kwaliteitscontrole. Eenmaal gepubliceerd op de winkel kunnen gebruikers de applicatie downloaden en installeren op hun apparaat.

Als laatste kan een mix van de twee gemaakt worden en dat wordt een hybride applicatie genoemd. Dit kan op twee manieren gebeuren. Enerzijds kan de webapplicatie als een *native* applicatie worden ingepakt. Anderzijds kan één programmeertaal worden gebruiken om *native* applicaties te maken voor verschillende besturingssystemen. Bij beide manieren kan de applicatie net zoals een *native* applicatie nog steeds

opgeladen te worden naar een winkel. Hieronder volgen de voor- en nadelen van webapplicaties (2.4.1), *native* applicaties (2.4.2) en hybride applicaties (2.4.3).

2.4.1 Webapplicaties

In het rapport 'The (Not So) Future Web' [93] uit juni 2011 wordt gesteld dat tegen 2015 60% van alle mobiele bedrijfsapplicaties en 40% van alle mobiele consumenten-applicaties, webapplicaties zullen zijn. Er zijn namelijk veel voordelen [1] verbonden aan webapplicaties.

Ten eerste heeft iedereen die een webbrowser heeft op zijn mobiel apparaat, toegang tot de applicatie ongeacht het besturingssysteem. Dit voordeel gaat niet op voor *native* applicaties waar applicaties specifiek voor een besturingssysteem worden geschreven.

Ten tweede, aansluitend bij het bovenstaande voordeel, moet de code slechts eenmaal worden geschreven. Een vaak voorkomende term die dit samenvat is WORA: *write once, run anywhere* [41]. Dit is in tegenstelling tot een *native* applicatie die geprogrammeerd wordt aan de hand van een SDK die specifiek is voor het besturingssysteem.

Ten derde moeten webapplicaties niet worden geverifieerd door een winkel vooraleer ze worden uitgebracht. Dit is wel het geval voor *native* applicaties. Hierdoor kan in een webapplicatie een belangrijke update snel doorgevoerd worden, terwijl de *native* applicatie nogmaals het verificatieproces moet doorlopen.

Een nadeel van een webapplicatie is dat deze, net zoals een *native* applicatie, afhankelijk is van het besturingssysteem. Dit is specifiek het geval wanneer de nieuwste kenmerken van HTML5 (zie 2.5) in webapplicaties worden gebruik.

2.4.2 Native applicaties

Voordelen [1] om een *native* applicatie te schrijven zijn onder meer de snelheidswinst doordat de applicatie rechtstreeks met het besturingssysteem kan werken. Aansluitend bij het vorige kan ook worden geargumenteerd dat een *native* applicatie over het algemeen gemakkelijker de kenmerken van het mobiel apparaat, zoals de camera of GPS, kan aanspreken. Ten derde blijft beveiliging nog altijd een knelpunt bij webapplicaties. Een *native* applicatie heeft hier minder problemen. Zo kunnen *native* applicaties gemakkelijker Central Processing Unit (CPU)-intensieve encryptie uitvoeren en geavanceerde authenticatietechnieken gebruiken zoals gezichtsherkenning. Als laatste kan worden opgemerkt dat het gebruik van een winkel (bv. App Store) voor het aanbieden van een applicatie een voordeel is. Ten eerste maakt een winkel het distribueren van de applicatie gemakkelijker. Ten tweede zorgt de winkel voor de correcte uitbetaling als gebruikers de applicatie downloaden. Als laatste controleert de winkel de applicatie, wat niet het geval is voor webapplicaties.

Zoals opgemerkt bij webapplicaties ligt het nadeel bij *native* applicaties dat deze worden geprogrammeerd met behulp van een SDK specifiek tot het besturingssysteem. Voorbeelden hiervan zijn Objective-C voor iOS en Java voor Android. Om een appli-

catie voor beide te schrijven, dient de code in twee verschillende programmeertalen te worden geschreven én te worden onderhouden.

2.4.3 Hybride applicaties

Het voordeel van een hybride applicatie is dat ze de problemen van webapplicaties oplossen met de voordelen van *native* applicaties [79]. Hierdoor kunnen specifieke kenmerken van het mobiel apparaat benaderd worden die vanuit een pure webapplicatie niet konden worden benaderd. Dit komt omdat hybride raamwerken een Application Programming Interface (API) gebruiken die toegang heeft tot de hardware van het apparaat. Daarentegen is HTML5 vanaf december 2012 in *candidate recommendation* gegaan [45]. Dit impliceert de opkomende ondersteuning van HTML5 in hedendaagse browsers. Hierdoor zullen hybride applicaties waarschijnlijk achterhaald worden naar mate de tijd vordert.

2.5 HTML5, CSS3 en JavaScript

De drie bouwstenen voor webontwikkeling zijn HTML5, CSS3 en JavaScript. HTML5 is verantwoordelijk voor de inhoud, CSS3 voor de presentatie en JavaScript voor de functionaliteit [94]. Hieronder worden deze bouwstenen dan ook toegelicht.

2.5.1 HTML5

HTML5 is een opmaaktaal die gebruikt wordt om webpagina's te maken. De 5 in HTML5 impliceert al dat de taal een lange weg heeft afgelegd. Zoals MacDonald [70] uitlegt, stopte in 1998 het W3C (World Web Consortium) met het werken aan de HTML-standaard en alle energie ging uit naar zijn opvolger: XHTML 1.0 (eXtensible HyperText Markup Language), een verbeterde HTML-versie die XML-gedreven (eXtensible Markup Language) is. XHTML kwam in grote mate overeen met HTML, maar de syntax was veel strikter. In het begin kon het zijn naam waarmaken en webontwerpers helpen om betere resultaten te boeken doordat ze slechte gewoontes moesten opgeven. Jammer genoeg bleven de beloofde voordelen uit. Wat veel erger was voor de nieuwe standaard, was dat geen enkele browser klaagde indien deze strikte syntax niet werd gevolgd.

Als reactie van het W3C werd hierop een nieuwe versie, namelijk XHTML 2, uitgebracht [70]. De manier waarop webpagina's werden geschreven veranderde doordat vele tags waren veranderd of verwijderd. Daarenboven sleepte deze nieuwe standaard maar aan en aan, wat ook niet in hun voordeel was.

In 2004 werd WHATWG (Web Hypertext Application Technology Working Group) door Opera Software, Mozilla Foundation en Apple gevormd. Ze wilden HTML niet vervangen, maar uitbreiden en die manier moest achterwaarts compatibel zijn. Na reflectie geloofde ook het W3C in deze aanpak, weliswaar op hun eigen manier. Zo werd HTML5 geboren, waarbij versie 5 refereert naar waar de vorige versie, HTML 4.01, gestopt was.

HTML5 is op het moment van schrijven nog altijd in ontwerp [70]. Hierdoor kunnen nieuwe kenmerken op ieder moment worden toegevoegd. Er is ook nog steeds onduidelijkheid waar HTML5 zal toe leiden. Het W3C focust op een unieke HTML5-standaard (verwacht rond 2014) terwijl WHATWG de nieuwe opmaaktaal ziet als levende taal waarbij voortdurend nieuwe dingen kunnen worden toegevoegd. Een belangrijke opmerking hierbij is dat het laatste woord altijd bij de webbrowser ligt, net zoals dat het geval was met de strikte syntax in XHTML. Als een kenmerk niet in de browser wordt ondersteund, heeft het ook geen kans op overleven.

Drie basisprincipes

Achter HTML5 zit een filosofie die in drie basisprincipes kan worden samengevat [70]. De eerste is achterwaartse compatibiliteit. De standaard mag geen veranderingen invoeren die oudere pagina's doet breken. Ten tweede moet de standaard geen nieuwe specificaties afdwingen die door de meerderheid op een andere manier worden gedaan. Als laatste moeten de specificaties ook een praktisch nut hebben. Waar veel vraag naar is, weegt het best op om in de specificaties op te nemen.

Acht technologieklassen

HTML5 kan ook bekeken worden volgens acht technologieklassen [138]. Iedere klasse wordt met enkele concrete voorbeelden aangehaald.

- **Multimedia** De nieuwe video- en audiotags maken het mogelijk om video- en geluidsfragmenten toe te voegen zonder gebruik te maken van plug-ins van derden zoals Adobe Flash [2] en Microsoft Silverlight [76].
- **Offline en opslag** Mobiele apparaten zijn onstabiel in hun verbinding met het Internet. HTML5 voorziet het offline werken in de cache, lokale opslag (vroeger kon dit enkel via de zogenaamde *cookies*) en een API om bestanden te manipuleren.
- **Performantie en integratie** Web Workers maken het mogelijk om langdurige JavaScript-taken in de achtergrond uit te voeren zodat webapplicaties dynamisch en snel blijven.
- **Semantiek** Nieuwe tags zorgen voor meer semantiek binnen webpagina's. Waar voorheen de webpagina bestond uit een verzameling `div`-elementen, kan nu veel concreter worden aangegeven wat er precies binnen die tags staat. Dit kan voor Search Engine Optimization (SEO) een grote impact hebben. Daarnaast biedt dit ook mogelijkheden voor schermlezers die nu beter de pagina kunnen analyseren.
- **CSS3** Hand in hand met HTML5 gaat CSS3 (zie 2.5.2). Het laat toe om webpagina's op te maken afhankelijk van het formaat van het mobiele apparaat. Daarnaast kunnen webpagina's ook door middel van CSS3 met effecten worden uitgebreid.

- **3D, grafieken en effecten** De nieuwe `canvas`-tag is in samenwerking met enkele lijnen JavaScript enorm krachtig om eenvoudig tekeningen en animaties te programmeren.
- **Verbinding** Gebeurtenissen aan serverzijde kunnen data naar WebSockets doorsturen. Hierdoor moet de webpagina niet meer voortdurend de server raadplegen, wat veel efficiënter is.
- **Toegang tot het apparaat** Webapplicaties kunnen meer en meer kenmerken zoals camera en GPS aanspreken net zoals *native* applicaties dat kunnen.

Kenmerken detecteren en opvullen

Door enerzijds de levendigheid van HTML5 en anderzijds het verdeelde landschap van browsers en besturingssystemen, worden niet alle kenmerken van HTML5 overal ondersteund. Hierdoor zal ten eerste moeten worden gedetecteerd of het kenmerk al dan niet wordt ondersteund. Ten tweede, indien het kenmerk niet wordt ondersteund, zal het moeten worden opgevuld met een vervanger. Ten derde zal deze aanpak ook vervat worden in de werking van de raamwerken (zie hoofdstuk 3) onder de vorm van *progressive enhancement* en *graceful degradation*. Deze drie ideeën worden hieronder kort toegelicht [70].

KENMERKEN DETECTEREN Er kan zelf worden opgezocht welke kenmerken op welke apparaten werken. Dit kan bijvoorbeeld gecontroleerd worden op websites zoals Can I Use [24] en Mobile HTML5 Compatibility [29].

Wat nog handiger is, is om op het apparaat zelf te detecteren of het gewenste kenmerk beschikbaar is. Een tool die hiervoor kan worden gebruikt, is Modernizr [81]. Het toevoegen van dit JavaScript-bestand creëert een JavaScript-object dat voor elk kenmerk teruggeeft of het al dan niet in de gebruikte browser wordt ondersteund.

KENMERKEN OPVULLEN Wanneer eenmaal gedetecteerd is dat een kenmerk niet aanwezig is, zijn er twee mogelijkheden: ofwel terugvallen op een alternatief of simuleren van dat kenmerk. Een voorbeeld van dit eerste kan gebeuren bij het gebruiken van de `video`-tag. Indien dit niet wordt ondersteund, kan worden teruggevallen op de Adobe Flash plug-in [2]. Voor het emuleren van een kenmerk wordt gebruik gemaakt van *polyfills*. Dit zijn alternatieven op basis van JavaScript waarbij de *native* functionaliteit die normaal aanwezig moet zijn, geëmuleerd wordt [139].

PROGRESSIVE ENHANCEMENT EN GRACEFUL DEGRADATION Er dient een onderscheid te worden gemaakt tussen de begrippen *progressive enhancement* en *graceful degradation* [43]. Bij de eerstgenoemde wordt gestart met de basis HTML-code. Deze code wordt door iedere browser op een goede manier weergegeven. Daarna zullen er iteratief elementen worden toegevoegd tot het moment dat de betreffende browser een bepaald kenmerk niet meer ondersteund.

De tegenhanger hiervan is *graceful degradation*. Hierbij wordt eerst een versie ontwikkeld die enkel in de meest recente browser kan worden getoond. Daarna zullen *fallbacks* geïmplementeerd worden waardoor ook minder recente browsers de applicatie kunnen weergeven.

2.5.2 CSS3

CSS3 zorgt voor de presentatie en is het hart van webdesign. Net zoals HTML5 heeft CSS3 hetzelfde probleem als het aankomt op de ondersteuning bij browsers [70]. Ook hier is er dus een brede waaier aan kenmerken die nog niet overal worden ondersteund. Kenmerken die enkel in een bepaalde browser ondersteund worden, worden voorafgaan door een browserprefix (zoals `-webkit-` voor WebKit gebaseerde browsers).

In deze sectie worden kort de belangrijkste eigenschappen besproken zoals Media Queries, effecten en lettertypes aan de hand van MacDonald [70].

Media Queries

Zoals aangehaald zijn er verschillende apparaten met verschillende schermen en resoluties. Een goede webpagina bestaat erin deze elementen zo goed mogelijk te benutten. Dit kan vanaf nu door gebruik te maken van Media Queries in CSS3. De website kan zich hiermee aanpassen aan het apparaat waarop het wordt getoond, wat Responsive Web Design (RWD) wordt genoemd.

Ook CSS3 volgt het principe van achterwaartse compatibiliteit. Browsers die Media Queries niet ondersteunen, zullen deze negeren en enkel de gewone lay-out toepassen ongeacht het apparaat.

Effecten

Transparantie, afgeronde hoeken, schaduw en kleurenverloop zijn maar enkele van de nieuwe kenmerken in CSS3. Voorheen moest de webdesigner deze dingen vaak met afbeeldingen oplossen, maar nu kan dit allemaal gebeuren met CSS3. Daarnaast zijn er ook effecten als transformaties en transities. Zo is het mogelijk wanneer de cursor over een afbeelding gaat dat deze ingezoomd en geroteerd wordt.

Dit is zeer vooruitstrevend omwille van twee zaken. Enerzijds is het gemakkelijker om dit in CSS3 dan in JavaScript te programmeren. Anderzijds komt er ook meer en meer ondersteuning vanuit de hardware. Zo worden 3D-transformaties in CSS3 versneld door de Graphics Processing Unit (GPU) [41, 63].

Lettertypes

Een laatste kenmerk is de betere ondersteuning van lettertypes. Waar vroeger enkel gewerkt kon worden met veilige lettertypes voor het web, is het nu mogelijk om eigen lettertypes op te laden en te gebruiken op een website.

2.5.3 JavaScript

JavaScript gaat terug tot in 1995, toen LiveScript genoemd [73]. Het is in staat het Document Object Model (DOM) aan te passen [94], dat is een API voor HTML-documenten [42]. Hierdoor kunnen dynamische interfaces gecreëerd worden, kan op

gebeurtenissen - zoals ergens op klikken - onmiddellijk gereageerd worden en is de website dan ook meer bruikbaar geworden door deze directe feedback [73].

Het schrijven van JavaScript is niet gemakkelijk om twee redenen [73]. Ten eerste, vergelijkbaar met HTML5 en CSS3, kunnen browsers JavaScript op verschillende manieren interpreteren. Dit komt doordat JavaScript een scripttaal is en daardoor moet worden geïnterpreteerd. Deze interpretatie is ingebouwd in de webbrowser en doordat er verschillende browsers bestaan, zullen er ook verschillende manieren van interpretatie bestaan. De ontwikkelaar dient dus tijdens het programmeren met deze verschillen rekening te houden. Ten tweede vergt het schrijven van simpele, veel voorkomende taken soms veel code.

Een oplossing voor de bovenstaande pijnpunten is gebruik te maken van een bibliotheek. Een voorbeeld hiervan is de populaire jQuery-bibliotheek [55] . Het is ook mogelijk om deze bibliotheek uit te breiden met verscheidene plug-ins om de functionaliteit te vergroten [73].

2.6 Mobiele HTML5-raamwerken

Om het ontwikkelingsproces van mobiele HTML5-applicaties te versnellen, worden raamwerken aangeboden die helpen bij de functionaliteit van de applicatie en de elementen voor de gebruikersinterface. In deze sectie wordt ingezoomd op bestaande mobiele HTML5-raamwerken die gebruik maken van de laatste nieuwe technologieën zoals HTML5, CSS3 en JavaScript. Doordat deze raamwerken er in overvloed zijn, is het onmogelijk om deze allemaal te bespreken en te vergelijken. Hierdoor werden acht raamwerken geselecteerd waarvan uiteindelijk vier raamwerken gedetailleerd besproken en vergeleken zullen worden.

Raamwerken kunnen worden gecategoriseerd volgens twee aanpakken, namelijk opmaakgedreven en JavaScript-gedreven [87]. Bij een opmaakgedreven aanpak wordt de webapplicatie voornamelijk in HTML-code geschreven. Daarentegen wordt bij een JavaScript-gedreven aanpak hoofdzakelijk in JavaScript geprogrammeerd.

De acht besproken raamwerken zijn op drie manieren onder de aandacht van de auteurs gekomen. Ten eerste werden in samenvattende literatuur over raamwerken jQuery Mobile en Sencha Touch als interessante raamwerken bevonden door hun grote populariteit [30, 41, 87, 19]. Ook The-M-Project werd in de literatuur aangehaald [30]. Lungo en jQT werden in recente literatuur gevonden [30, 41]. Ten tweede werden via het Internet Kendo UI en Moobile gevonden. Als laatste werd door Capgemini DaVinci voorgesteld.

jQuery Mobile

jQuery Mobile is een opmaakgedreven raamwerk dat werd aangekondigd in 2010 en hoofdzakelijk gebruikersinterface-elementen (GI-elementen) aanbiedt [98]. Het raamwerk wordt beheerd door het jQuery Project dat onder andere jQuery beheert waar jQuery Mobile afhankelijk van is [59]. Op het moment van schrijven zit jQuery Mobile aan versie 1.3.1 [92]. Sinds september 2012 is het enkel nog mogelijk om jQuery Mobile onder de Massachusetts Institute of Technology (MIT) licentie te

verkrijgen [25]. Dit betekent dat de code wordt vrijgegeven als *open-source* en dat deze tegelijkertijd kan worden gebruikt in propriëtaire projecten en applicaties [94].

Sencha Touch

Sencha Touch wordt ontwikkeld door Sencha, een bedrijf dat in 2010 is ontstaan als een samensmelting van Ext JS, jQuery Touch en Raphaël. Ext JS kan als de voorganger van Sencha Touch worden beschouwd. Sencha Touch is net als Ext JS een JavaScript-gedreven raamwerk met een Model-View-Controller (MVC) ontwerppatroon. Sencha Touch biedt een commerciële licentie waarbij het bedrijf in kwestie de broncode niet publiek mag maken. Een GNU's Not Unix General Public License v3 (GNU GPLv3)-versie van Sencha Touch laat dit wel toe. Beide licenties zijn gratis. Op het moment van schrijven is Sencha Touch aan versie 2.2 [111].

The-M-Project

The-M-Project is een raamwerk dat het mogelijk maakt om webapplicaties te bouwen die van jQuery Mobile gebruik maken. Initieel werd het in 2012 ontwikkeld door M-Way Solutions maar nu behoort het tot Panacoda, een Duitse ontwikkelaar voor softwaretools en mobiele webapplicaties. Panacoda bezit ook Espresso, een tool om applicaties te bouwen en ontwikkelen met The-M-Project. Het laat ook toe applicaties om te vormen tot *native* applicaties. The-M-Project is *open-source* en wordt vrijgegeven onder een MIT-licentie. Dit raamwerk is volledig JavaScript-gedreven en steunt op het MVC-ontwerppatroon. Ook ondersteunt het HTML5- en CSS3-kenmerken zoals offline beschikbaarheid en lokale opslag. Op het moment van schrijven is The-M-Project aan versie 1.4. Het is belangrijk op te merken dat in de zomer van 2013 versie 2.0 wordt verwacht. De nieuwe versie zal van de grond worden opgebouwd omdat enkel de code aanpassen niet meer voldoende bleek. De voornaamste werkpunten zijn performantie en platformonafhankelijkheid [89, 66].

Lungo

Lungo is een opmaakgedreven raamwerk waarbij versie 1.0 uitkwam in 2011 [122]. Het raamwerk wordt onderhouden door TapQuo dat een Spaans bedrijf is, gespecialiseerd rond mobiele gebruikerservaring [128]. Lungo is afhankelijk van een JavaScript-bibliotheek, namelijk QuoJS. Lungo biedt vooral GI-elementen aan, maar daarnaast zijn er ook *wrappers* voor cache, opslag en SQL beschikbaar [123]. Er wordt geen ontwerppatroon zoals MVC afgedwongen. Het raamwerk wordt onder de GNU GPLv3-licentie vrijgegeven, maar ook een commerciële versie is mogelijk. Op het moment van schrijven zit Lungo aan versie 2.1 [123].

jQT

jQT, voorheen jQTouch, is een plug-in voor jQuery of Zepto. Zepto is een JavaScript-bibliotheek die compatibel is met de API van jQuery, maar minimaler is dan

jQuery [145]. jQT is een opmaakgedreven raamwerk dat wordt vrijgegeven onder de MIT-licentie. Het werd gemaakt door David Kaneda die daarna Sencha heeft opgericht [20]. De start van het raamwerk dateert van 2010 en op het moment van schrijven zit jQT aan versie 1 beta 4rc [50, 20]. De GI-elementen bootsen de *native look-and-feel* van iOS na.

Kendo UI

Kendo UI is een raamwerk van de hand van Telerik en bestaat uit drie luiken: Web, Mobile en DataViz. Eind 2011 werd de eerste stabiele versie vrijgegeven. Het eerste luik is gericht op de ontwikkeling van desktop- en mobiele applicaties, het tweede voegt *native look-and-feel* toe aan mobiele applicaties en het laatste zorgt voor datavisualisatie met HTML5 en JavaScript. Kendo UI is zowel een JavaScript- als opmaakgedreven raamwerk met een MVVM-ontwerppatroon (Model-View-View Model) dat steunt op de jQuery-bibliotheek. Verder heeft de ontwikkelaar ook de mogelijkheid om een *backend* te integreren met de klantzijde. ASP.NET MVC, PHP en JSP zijn momenteel de ondersteunde technologieën voor *backend* integratie. Een licentie voor Kendo UI waarbij één van voornoemde technologieën mogelijk is, kost $999. Zonder *backend* integratie zijn de kosten met $300 verminderd. Op het moment van schrijven is Kendo UI aan versie 2013 Q2 [134].

Moobile

Moobile is een jong raamwerk dat op het moment van schrijven aan versie 0.2.1 zit sinds de eerste *commit* op GitHub in 2011 [23]. Het raamwerk is JavaScript-gedreven, volgt het MVC-ontwerppatroon en wordt onderhouden door de Canadees Jean-Philippe Déry. Zelf biedt Moobile voornamelijk GI-elementen aan, maar deze blijven (voorlopig) beperkt tot zeven componenten en drie schermovergangen [23]. Moobile mist echter ondersteuning voor formulieren. Een bijzonderheid is dat de lay-out de *native look-and-feel* van Android of iOS nabootst. Bij het raamwerk is een simulator en tool om de volledige applicatie te maken, aanwezig. Deze laatstgenoemde zorgt automatisch voor compressie van de verschillende bestanden.

DaVinci

Het Koreaanse raamwerk DaVinci bestaat uit twee tools: DaVinci Studio en DaVinci Animator. De nadruk bij dit raamwerk ligt voornamelijk bij de generatie van code in een WYSIWYG-omgeving (What You See Is What You Get). DaVinci Studio is een plug-in voor Eclipse die HTML-, JavaScript- en CSS-code genereert. De gebruiker kan GI-elementen via *drag-and-drop* aan de applicatie toevoegen. Het binden van data kan door op een visuele manier de mapping tussen GI en data te maken. Het testen van de applicatie kan op een bijgeleverde emulator die de applicatie op verschillende lay-outs kan weergeven. Het raamwerk gebruikt een architectuur dat compatibel is met *open-source* bibliotheken zoals jQuery [55], KnockOut [62] en Backbone [10]. DaVinci Animator kan worden gebruikt om animaties op basis van HTML5 en CSS3

te maken in een grafische omgeving. In Seoul National University Research Park worden beide tools ontwikkeld. Op het moment van schrijven is DaVinci toe aan versie 2.0. Alle documentatie is momenteel nog niet vertaald van het Koreaans naar het Engels [44].

2.7 Vergelijken van raamwerken

Om de verschillende mobiele HTML5-raamwerken te kunnen vergelijken is een consistente manier nodig. HTML5-raamwerken zijn software en om software te vergelijken bestaan er standaarden. Deze worden in sectie 2.7.1 besproken. Daarna worden in sectie 2.7.2 de vergelijkingen van raamwerken geclassificeerd zoals gevonden in de literatuur en blogposts.

2.7.1 Software vergelijken

Hier worden de ISO 25010-standaard en de AHP-methode gepresenteerd. Uit de thesis van Guelinckx [39] die zelf een raamwerk probeert te maken, werd de ISO 25010-standaard aangehaald om software te beoordelen. De paper van Jadhav et al. [46] bespreken evaluatietechnieken voor het selecteren van softwarepakketten. Hierin staat dat de AHP-methode wijd gebruikt is bij de evaluatie van softwarepakketten.

ISO 25010

Onder de ISO 25010-standaard [119] vallen twee modellen: de productkwaliteit en de kwaliteit van het product in gebruik. Beide modellen beschrijven de kwaliteit van software op basis van een aantal categorieën met specifieke kwaliteitseigenschappen. Het beoordelen van de categorieën kan gebeuren op basis van een checklist.

Productkwaliteit De acht categorieën die horen bij dit model worden hieronder geciteerd:

- Functionele geschiktheid is de mate waarin een softwaresysteem functies levert die voldoen aan alle behoeften.
- Prestatie-efficiëntie bepaalt de prestaties in verhouding tot de hoeveelheid gebruikte middelen.
- Uitwisselbaarheid beschrijft de mate waarin een softwaresysteem informatie kan uitwisselen met andere systemen die dezelfde hard- of software-omgeving delen.
- Bruikbaarheid is de mate waarin een softwaresysteem gebruikt kan worden om gespecificeerde doelen te bereiken.
- Betrouwbaarheid is de mate waarin een softwaresysteem gespecificeerde functies uitvoert onder gespecificeerde condities gedurende een gespecificeerde hoeveelheid tijd.

- Beveiligbaarheid bepaalt de mate waarin een product of systeem informatie en gegevens beschermt.
- Onderhoudbaarheid is de mate waarin een softwaresysteem gewijzigd kan worden door de beheerders.
- Overdraagbaarheid is de mate waarin een systeem, product of component overgezet kan worden van één hardware, software of andere operationele gebruiksomgeving naar een andere.

KWALITEIT IN GEBRUIK De vijf categorieën die horen bij dit model worden hieronder geciteerd:

- Effectiviteit is de nauwkeurigheid en volledigheid waarmee gebruikers gespecificeerde doelen behalen.
- Efficiëntie bepaalt de benodigde hulpbronnen die gebruikt zijn waarmee gebruikers gespecificeerde doelen behalen.
- Voldoening bepaalt de tevredenheid van de eindgebruiker wanneer het softwaresysteem gebruikt wordt in een bepaalde context.
- Vrijheid van risico is de mate waarin een softwaresysteem het risico beperkt met betrekking tot economische status, mensenlevens, gezondheid of omgeving.
- Context dekking bepaalt de verschillende contexten waarin een softwaresysteem gebruikt kan worden in combinatie met de vier vorige categorieën.

De kwaliteit in gebruik wordt dus bepaald door zowel de kwaliteit van de software, de hardware en het besturingssysteem samen met de gebruikers, hun taken en de sociale omgeving. Elke categorie kan toegewezen worden aan verschillende activiteiten van belanghebbenden. De belanghebbenden worden opgedeeld in primaire en secundaire gebruikers. De eerste zijn de personen die het systeem actief gebruiken. De laatste zijn diegene die zorgen voor het onderhoud.

AHP-methode

De Analytic Hierarchy Process (AHP) methode werd ontwikkeld door Saaty [100] en kan worden gebruikt voor het vergelijken van softwarepakketten. Ze is gebaseerd op een hiërarchie waarbij de bovenste laag zich bezighoudt met het doel van het selectieproces (bv. kies een softwarepakket). Op de tussenlaag worden de criteria voorgesteld (bv. ondersteuning en performantie). De onderstaande laag bevat de kandidaten (bv. de verschillende softwarepakketten).

Eerst worden prioriteiten toegekend aan de criteria. Daarna zullen de kandidaten per criteria paarsgewijs worden vergeleken. Bij deze vergelijking kunnen scores van één (even belangrijk) tot negen (zeer belangrijk) worden toegekend. Deze twee worden gecombineerd om tot een rangschikking van kandidaten te komen. Hoe hoger de score, hoe meer de kandidaat geschikt is om het doel te beogen.

2.7.2 Classificatie

Op het web en in de literatuur kunnen manieren worden teruggevonden waar raamwerken met elkaar worden vergeleken. Deze werkwijzen verschillen in de gekozen criteria alsook de manier waarop de criteria beoordeeld worden. Er kunnen vier grote werkwijzen worden onderscheiden. Een eerste methode is het iteratief bespreken van gekozen criteria. Een tweede bestaat uit het quoteren van de gekozen criteria op basis van een zelfgekozen puntensysteem. Verder kan er ook van een *proof of concept* (POC) worden uitgegaan om het raamwerk te testen. Ten slotte kunnen de gekozen criteria in tabelvorm worden ondergebracht om tot een overzichtelijke vergelijking te komen.

Bespreking

Een voorbeeld van een blogpost waarbij vergelijkingscriteria iteratief besproken worden is de Mobile framework SMACKDOWN! op Dinosaurs with Laserz [99]. Deze vergelijkt jQT, jQuery Mobile, Sencha Touch, PhoneGap en Titanium. De laatste twee zijn hybride raamwerken. De blogpost verdeelt deze vijf raamwerken onder *web development frameworks* en *custom API frameworks*. De eerste term staat voor raamwerken die uitsluitend in HTML, CSS en JavaScript ontwikkelen. jQT, jQuery Mobile en PhoneGap kunnen hiertoe gerekend worden. *Custom API frameworks* betekent dat de ontwikkelaar uitsluitend gebruik maakt van JavaScript of een eigen API van het raamwerk. Sencha Touch en Titanium behoren tot deze categorie. De criteria die besproken worden zijn:

- *Nativeness*
- Browseronafhankelijkheid
- Performantie
- Gemak bij ontwikkeling
- Toegang tot hardware van het apparaat
- Prijs voor licentie

De blogpost besluit dat geen enkel raamwerk als beste kan worden beschouwd, doordat de keuze varieert van project tot project.

Puntensysteem

Op een blogpost van Codefessions wordt een vergelijking gemaakt tussen jQuery Mobile, Sencha Touch, jQT en Kendo UI [102]. Als referentiesysteem gebruiken ze zeven criteria. De eerste drie zijn de *native look-and-feel*, performantie en platformonafhankelijke capaciteiten. Deze worden gequoteerd met een cijfer van 0 tot 5 waarbij 5 staat voor de maximale score. Kenmerken worden gequoteerd door de raamwerken met elkaar te vergelijken. Het raamwerk met de meeste kenmerken

krijgt een 5, het tweede beste een 4, enzovoort. Op een analoge manier wordt code-efficiëntie en gebruiksgemak gequoteerd. Het raamwerk dat de minste lijnen code vereist voor een simpele applicatie, krijgt de perfecte score. Hierbij moeten wel alle bestanden gerekend worden die het raamwerk nodig heeft om functioneel te zijn. In zekere zin overlapt dit criterium dus met de *proof of concept* werkwijze. Licenties krijgen een score van 0 tot 5 waarbij 0 betekent dat het niet beschikbaar is voor een individuele ontwikkelaar en 5 dat het raamwerk *open-source* en gratis te gebruiken is. Andere factoren zoals omkadering en uitbreidbaarheid worden niet in de vergelijking opgenomen omdat ze afhangen van de interesse van de gebruiker. Ze worden echter wel bekeken.

Het rapport over Cross-platform Developer Tools van Vision Mobile [79] vergelijkt 15 *cross-platform* tools. In deze vergelijkingsmethode worden de tools voorgesteld op basis van negen criteria: kosten, aantal ondersteunde platformen, performantie, leercurve, ervaring met ontwikkeling en debugging, toegang tot hardware API, GI-functionaliteiten, applicatie publicatie en professionele ondersteuning. Elk criterium wordt met een score tussen 1 en 5 gequoteerd. Bij de evaluatie wordt bij elk criterium het gemiddelde van dat criterium voor alle andere *cross-platform* tools geplaatst. Een enquête met meer dan 2400 ontwikkelaars bepaalt de score voor elk criterium. De bespreking van elke tool sluit af door de belangrijkste criteria samen te vatten, meest bekende applicaties te vermelden en een lijst met verbeteringen voor te stellen.

Proof of concept

Het gebruik van een POC of voorbeeldapplicatie probeert raamwerken te vergelijken door ze echt toe te passen. Oehlman et al. [87] vergelijken Jo, jQT, jQuery Mobile en Sencha Touch door een geosociaal spel te ontwikkelen genaamd Moundz. Deze applicatie maakt gebruik van bestaande sociale locatiegebaseerde netwerken als Foursquare en Gowalla. Als benchmark werd deze mobiele applicatie eerst zonder raamwerk gebouwd.

De vier besproken raamwerken worden ingeleid met de meest karakteristieke kenmerken en sterktes. Vervolgens wordt een vereenvoudigde versie van Moundz omgevormd tot een applicatie geïmplementeerd in elk raamwerk. Bij elke transformatie wordt stap voor stap uitgelegd welke veranderingen de originele code moet ondergaan. Hierbij wordt de werking en gebruik van het raamwerk toegelicht.

Een ander voorbeeld van het gebruik van een POC staat in Mobile JavaScript Application Development van Kosmaczewski [64]. Hier wordt een takenapplicatie ontwikkeld in PhoneGap, Sencha Touch en jQuery Mobile. Elk raamwerk wordt ingeleid door eerst de platformen op te lijsten die het ondersteunen en de belangrijkste kenmerken weer te geven. Vervolgens worden de functionaliteiten van het raamwerk uitgelegd en toegepast door stap voor stap de takenapplicatie op te bouwen. Ook worden er tools besproken die de ontwikkeling vergemakkelijken.

Vergelijkingstabellen

Vergelijkingstabellen proberen raamwerken zo objectief mogelijk te vergelijken. Hier worden de raamwerken in de rijen en de vergelijkingscriteria in de kolommen geplaatst of omgekeerd.

Een voorbeeld is een webpagina waar jQuery UI met Kendo UI wordt vergeleken [13]. Deze bestaat uit één grote tabel die specifieke kenmerken tussen beide raamwerken vergelijkt. In de tabel wordt een vergelijking weergegeven van onder andere beschikbare thema's, browsercompatibiliteit, formuliervalidatie en ondersteuning van het product.

Een alternatief is de Mobile Frameworks Comparison Chart van Falk waar 43 raamwerken in een vergelijkingstabel zijn opgenomen [27]. De vergelijkingscriteria worden opgedeeld in compatibiliteit met het besturingssysteem, doel van de applicatie, taal voor ontwikkeling, hardware-interactie, GI, licenties en overige. Deze laatste categorie bevat de criteria of er al-dan-niet een SDK beschikbaar is, encryptie ondersteund wordt en of advertenties worden ondersteund. Handig hierbij is dat de webpagina een stappenplan voorziet waarin per categorie alle vereisten ingevuld kunnen worden. De resultaten zijn dan de raamwerken die compatibel zijn met deze vereisten.

Hoofdstuk

3 Mobiele HTML5-raamwerken

Dit hoofdstuk vergelijkt op een passieve manier de vier gekozen mobiele HTML5-raamwerken, namelijk Sencha Touch, Kendo UI, jQuery Mobile en Lungo. Dit is een vergelijking gebaseerd op informatie die in de literatuur werd gevonden. Omkadering, code en ontwikkeling, functionele kenmerken en niet-functionele kenmerken zullen voor ieder raamwerk worden besproken. Deze criteria worden passieve criteria genoemd.

Het tijdsbudget liet toe om vier van de acht gepresteerde raamwerken (zie sectie 2.6) te vergelijken. In samenspraak met Capgemini werd gekozen om jQuery Mobile, Sencha Touch en Kendo UI te onderzoeken. De twee eerstgenoemde werden gekozen om hun enorme populariteit in de literatuur [30, 41, 87, 19]. Bij interviews met een ontwikkelaar, architect en verkoopmanager van Capgemini kwamen ook deze twee raamwerken naar boven. Kendo UI heeft het voordeel dat het de *native look-and-feel* kan nabootsen, wat ook een handig onderzoekspunt leek voor Capgemini. Doordat The-M-Project een volledige make-over ondergaat in de zomer van 2013 leek het de auteurs niet nuttig om de huidige versie te onderzoeken. Moobile werd ook uitgesloten door de nog zeer jonge versie van het raamwerk tijdens het onderzoek. jQT is slechts een plug-in voor jQuery Mobile en Sencha Touch is hierop verder gebouwd. Hierdoor werd niet voor jQT gekozen. Als laatste werd besloten om DaVinci uit te sluiten door de gebrekkige Engelstalige documentatie van dit Koreaans raamwerk. Het vierde en laatste onderzochte raamwerk werd daarom Lungo.

Zoals besproken in 2.6 kunnen raamwerken worden gecategoriseerd volgens twee aanpakken, namelijk opmaakgedreven en JavaScript-gedreven [87]. Bij een opmaakgedreven aanpak zijn er drie strategieën om webapplicaties te maken [14]. Een eerste is om de volledige applicatie in één webpagina te schrijven. De vele schermen van de webapplicatie zijn dan allemaal samengebracht op eenzelfde webpagina. Het voordeel bij deze aanpak is dat er initieel minder verzoeken zijn naar de server omdat alles in één bestand wordt opgehaald. Dit geldt ook zo voor de geïmporteerde CSS- en JavaScript-bestanden. Een tweede strategie is om voor ieder scherm een aparte webpagina aan te maken. Het voordeel hierbij is dat de eerste pagina waar de gebruiker op terecht komt, sneller wordt gedownload. Bij iedere navigatie naar een ander scherm, moet dit scherm via Asynchronous JavaScript and XML (AJAX) worden opgehaald, waardoor dit vertragend kan werken. Een laatste strategie is om

een mix tussen beide te maken. Men kan bijvoorbeeld alle schermen die de gebruiker vaak nodig heeft op één webpagina plaatsen. De schermen die de gebruiker zelden nodig heeft, worden op aparte webpagina's geplaatst.

Eerst wordt gestart met het bespreken van een volledig JavaScript-gedreven raamwerk, namelijk Sencha Touch (3.1). Daarna volgt Kendo UI (3.2) dat zowel JavaScript- als opmaakgedreven is. De laatste twee besproken raamwerken, jQuery Mobile (3.3) en Lungo (3.4), zijn beide opmaakgedreven. In de laatste sectie (3.5) wordt een overzicht van deze passieve vergelijking weergegeven.

3.1 Sencha Touch

Sencha Touch (ST) wordt ontwikkeld door Sencha. Dit bedrijf is in 2010 ontstaan als een samensmelting van Ext JS, jQuery Touch en Raphaël. Ext JS is een JavaScript-raamwerk voor de ontwikkeling van webapplicaties. jQuery Touch is een jQuery plug-in voor mobiele webontwikkeling. Het steunt op WebKit en voegt *touch events* toe aan jQuery. Raphaël, ten slotte, is een JavaScript-bibliotheek voor vectortekeningen. Op het moment van schrijven is Sencha Touch aan versie 2.2 [111]. De eerste stabiele versie werd in november 2010 voorgesteld.

3.1.1 Omkadering

Programmeertaal Sencha Touch is JavaScript-gedreven, dus alle functionaliteit wordt in JavaScript geïmplementeerd. Alle HTML-code wordt bij het bekijken van de pagina gegenereerd.

Tools Naast Sencha Touch levert Sencha nog producten die het ontwikkelingsproces versnellen. Deze worden hieronder opgelijst [111].

Sencha Architect Dit is een desktopapplicatie die het ontwikkelingsproces vergemakkelijkt met een GGI en *drag-and-drop* [109]. De applicatie kan worden gedownload voor Mac, Windows en Linux distributies. Sencha Architect kan voor 30 dagen uitgeprobeerd worden, daarna moet een licentie worden aangekocht. De prijs voor één ontwikkelaar bedraagt $399 [111].

Sencha Cmd Deze tool vergemakkelijkt de ontwikkeling van Sencha Touch applicaties door middel van commando's die vanuit een *command-line interface* kunnen worden uitgevoerd [110]. Sencha Cmd is beschikbaar voor Mac, Windows en Linux distributies. Het kan initiële applicaties opzetten, bestanden toevoegen en de applicatie bouwen en uitrollen. De applicatie kan ook omgevormd worden tot een *native* applicatie voor iOS en Android. De tool is gratis op de Sencha website te downloaden [111]. Een applicatie, zoals geïnitialiseerd door Sencha Cmd, moet alle benodigde JavaScript-, en CSS-bestanden in een JSON-bestand onderbrengen. Een `microloader` zal de afhankelijke bestanden automatisch laden bij het opstarten van de applicatie.

Het bouwen en uitrollen van een applicatie kan op vier niveaus:

`testing` maakt een testapplicatie om de kwaliteit te testen. JavaScript- en CSS-bestanden worden samengevoegd maar niet verkleind om makkelijk te debuggen.

`package` maakt een zelfstandige applicatie die verspreidbaar is en vanop een bestandensyteem, zonder webserver, kan lopen.

`production` maakt een applicatie die op een webserver beschikbaar wordt gemaakt waarvan de JavaScript- en CSS-bestanden zijn samengevoegd en verkleind. Het maakt de applicatie ook offline beschikbaar door gebruik te maken van de Applicatie Cache van HTML5. Ook is het mogelijk de applicatie op te waarderen naar een nieuwe versie via het Delta Update mechanisme. Dit mechanisme wil voorkomen dat bij een kleine aanpassing in de code, alle bestanden opnieuw moeten worden gedownload.

`native` maakt een *native* applicatie die op het Android- of iOS-besturingssysteem kan lopen.

Sencha Animator Dit is een desktopapplicatie om CSS3-animaties te ontwerpen [108]. De applicatie kan worden gedownload voor Mac, Windows en Linux distributies. Deze animaties worden enkel in WebKit-browsers ondersteund. De prijs voor één ontwikkelaar bedraagt $99 [111].

Documentatie Een zoekfunctie voor objecten, eigenschappen en methoden is in de documentatie aanwezig om snel zaken op te zoeken. De meeste functionaliteiten zijn voorzien van codevoorbeelden samen met het resultaat hoe de browser de code rendert. Alle plug-ins die Sencha aanbiedt, kunnen op de Sencha Market teruggevonden worden [107]. Verder biedt de Sencha website ook hulpmiddelen om Sencha Touch te leren gebruiken zoals handleidingen, introductievideos, enz.

Een ander handig naslagwerk is de Kitchen Sink [113]. Dit is een webapplicatie, geschreven in Sencha Touch, die de belangrijkste functionaliteiten bevat samen met de bijhorende code.

Marktadoptatie Volgens de Sencha website is 50% van de Fortune 100 - een lijst van de grootste Amerikaanse bedrijven gerangschikt op jaaromzet - een Sencha-klant [111]. Enkele van hun grootste klanten zijn CNN, Samsung, Cisco en Visa.

Licenties Sencha Touch biedt een commerciële licentie waarbij het bedrijf in kwestie de broncode niet deelt voor zijn gebruikers. Ook een *open-source* licentie van Sencha Touch bestaat. Beide licenties zijn gratis. De *open-source* licentie is voor applicaties die zelf een GNU GPLv3 opleggen. Dit wil zeggen dat de vrijheid bestaat om aanpassingen aan de broncode te maken en te verspreiden, zolang de code ook maar gratis verspreid wordt voor alle gebruikers.

Voor de ontwikkeling van eigen raamwerken of SDKs wordt een *original equipment manufacturer* (OEM) licentie voorzien. Dit wil zeggen dat bedrijven hun producten gaan verkopen onder hun eigen merk en naam, maar gebruik maken van Sencha Touch. Omdat het gebruik hiervan per gebruiker verschilt, worden OEM-licenties op maat gemaakt [111]. Deze OEM-licentie is niet gratis.

3.1.2 Code en ontwikkeling

Alle code moet in JavaScript worden geschreven. Eén HTML-bestand dient slechts als container om de bestanden in te laden. Sencha Touch valt dus onder JavaScript-gebaseerde raamwerken net zoals Ext JS, waarop Sencha Touch is verdergebouwd.

Samen met SASS (Syntactically Awesome Stylesheets) en Compass kan Sencha Touch lay-outs definiëren per apparaat. SASS breidt CSS uit met variabelen, geneste structuren, *mixins* en overerving [16]. De `Ext.env.Browser`- en `Ext.env.OS`-eigenschappen en `Ext.Viewport.getOrientation`- en `Ext.feature.has`-methoden kunnen de context opvragen en de juiste lay-out kiezen [18].

Debugging De broncode van Sencha Touch kan ingeladen worden met een uitgebreide bibliotheek. Deze versie is niet gecomprimeerd en bevat commentaar en documentatie om makkelijker uit te zoeken welke fout werd gemaakt.

3.1.3 Functionele kenmerken

Sencha Touch bevat alle elementen van de GI als JavaScript-objecten. Net zoals alle objectgerichte programmeertalen maken deze objecten gebruik van een klassensysteem, iets wat slechts vanaf Sencha Touch 2 werd ingevoerd. Op die manier kunnen klassen worden gedefinieerd (`Ext.define`) en aangemaakt (`Ext.create`). Hierbij is ook overerving mogelijk. De basisklasse van alle objecten is `Ext.Component`. Componenten kunnen worden gerenderd, zichzelf tonen of verbergen, centreren op het scherm en zichzelf aan- of uitzetten. Het aanmaken van componenten kan ook door de component als `xtype` te definiëren. Andere componenten kunnen deze `xtype` dan hergebruiken.

Een andere belangrijke component is `Ext.Container`. Containers kunnen subcomponenten bevatten en een lay-out specificeren. Alle componenten krijgen een naam die verwijst naar een *namespace*. Dit is handig om conflicten te vermijden tussen eigen objecten en standaard objecten van het raamwerk. Voor een opsomming van alle componenten wordt verwezen naar de documentatie [112].

Model Data kan intern worden voorgesteld met `Models`. Dit is iets wat hoort bij het MVC-ontwerppatroon (zie sectie 3.1.4). Een model specificeert een lijst van velden waarbij een veld een naam en een type heeft. Optioneel kunnen validaties bij de velden worden toegevoegd om validatieregels op de velden af te dwingen.

Store `Ext.data.Store` is de klasse om instanties van een model op te slaan. Een `Store` wordt voorzien van een `Proxy`. Deze kan data aan de klant- of serverzijde opslaan. Een `Proxy` voor opslag aan klantzijde kan zowel in het RAM-geheugen als in lokale opslag van de browser opslaan. Een `Proxy` voor serveropslag kan data verzenden naar hetzelfde domein via AJAX of tussen verschillende domeinen via JSONP. JSONP staat voor JavaScript Object Notation with Padding en is een methode om data op een server in een ander domein op te vragen. Een `Proxy` kan ook nog voorzien worden van een `Reader` of `Writer` die aangeeft hoe de ontvangen data gelezen of geschreven moet worden.

VIEW `Views` zijn objecten die aan de gebruiker kunnen worden getoond. Een voorbeeld hiervan zijn lijsten, waar instanties van een `Store` kunnen worden weergegeven. Zo een lijst kan gefilterd of gesorteerd worden op basis van velden uit het model. Hiervoor moeten `Filters` of `Sorters` aan de `Store` worden toegevoegd. De lay-out van één lijstitem bepalen kan via een `XTemplate`. Het sjabloon bepaalt de HTML-structuur van elk item. Alle gedefinieerde velden van het model kunnen in het sjabloon worden opgeroepen of gemanipuleerd.

CONTROLLERS De `Controllers` maken de binding tussen `Models` en `Views`. Ze kunnen gebeurtenissen opvangen en bijhorende operaties uitvoeren. Hierdoor is navigatie door de applicatie mogelijk en kunnen alle componenten gemanipuleerd worden. `Controllers` worden ook gebruikt bij de initialisatie van de applicatie.

3.1.4 Niet-functionele kenmerken

PERFORMANTIE In vergelijking met versie 1.1 van Sencha Touch is de performantie gestegen omwille van verschillende factoren. De introductie van het klassensysteem, zoals besproken in de vorige sectie, laat toe objecten dynamisch te laden. Het grote verschil tussen `Ext.define` en `Ext.create` is dat objecten enkel in het geheugen worden geladen na creatie. Het is dus de taak van de programmeur om objecten enkel te construeren wanneer ze nodig zijn.

Verder kwam versie 2.0 met een nieuwe lay-out *engine* die vooral het verwisselen van oriëntatie van het toestel versnelde. Ook een verbetering in performantie op Android-toestellen, voornamelijk bij scrollen en animaties, werd ingevoerd [111].

AANPASBAARHEID Elke component binnen het raamwerk moet overerven van `Ext.Component`, dat een `ui`-attribuut voorziet. De waarde hiervan is een CSS-klasse die bepaald hoe de component er zal uitzien. Sencha Touch heeft al twee CSS-klassen voorzien: `light` en `dark`. Andere componenten kunnen deze lijst uitbreiden. Een knop kan bijvoorbeeld `normal`, `back`, `round`, `small`, `action` of `forward` als `ui`-waarde hebben.

Het is ook mogelijk om eigen waarden voor `ui` te definiëren of de standaarden van Sencha Touch aan te passen. SASS en Compass maken dit mogelijk door eigen CSS-bestanden aan te maken. *Mixins* groeperen enkele CSS-eigenschappen en kunnen worden hergebruikt. Compass is een raamwerk bovenop SASS en CSS. Het compileert SCSS (Sassy CSS) naar CSS-bestanden [26].

Thema's van Sencha Touch bestaan allemaal uit een set van *mixins*. Door zelf *mixins* te creëren of reeds bestaande te manipuleren kunnen eigen thema's gecreëerd worden en ze aan de `ui`-waarde van een component toegekend worden.

PROGRAMMEERBAARHEID Zoals reeds aangehaald ondersteunt Sencha Touch het MVC-ontwerppatroon. Dit ontwerppatroon vermijdt lange JavaScript-bestanden door ze logisch op te delen. Modellen groeperen velden tot een beschrijving van data-objecten, `Views` definiëren de weergave van componenten en `Controllers` verbinden beide op basis van gebeurtenissen.

In theorie zou het verschil tussen mobiele websites en applicaties enkel in de `Views` terug te vinden zijn. Echter, dit wordt nog niet volledig ondersteund en daarom worden aparte projecten voor deze functionaliteit gepromoot.

Ondersteuning browser Sencha Touch steunt op de WebKit-browser *engine* dus moet de browser deze bevatten. Hoewel dit bij de meeste browsers geen probleem meer vormt, vallen toch enkele populaire browsers uit de boot. Sencha Touch is bijvoorbeeld niet compatibel met bijvoorbeeld Internet Explorer Mobile.

Op de website van Sencha zijn voor de belangrijkste browsers en besturingssystemen *scorecards* voorzien om hun compatibiliteit met HTLM5 en Sencha Touch te bespreken [111]. Hoewel geen echte scores worden uitgedeeld, zijn alle belangrijkste kenmerken uitgelicht en uitvoerig besproken.

3.2 Kendo UI

Kendo UI (KUI) is een raamwerk van het Duitse Telerik en in november 2011 gelanceerd. Het vergemakkelijkt de ontwikkeling van webapplicaties en bestaat uit drie luiken. Kendo UI Web is gericht op de ontwikkeling van desktop- en mobiele applicaties, Kendo UI Mobile voegt een *native look-and-feel* toe aan mobiele applicaties en Kendo UI DataViz zorgt voor datavisualisatie met HTML5 en JavaScript. Kendo UI is zowel een JavaScript- als opmaakgedreven raamwerk en heeft een Model-View-View Model (MVVM) ontwerppatroon dat steunt op de jQuery-bibliotheek. Het raamwerk biedt *widgets* aan voor meer geavanceerde functionaliteit, die onderliggend een plug-in in jQuery voorstellen. Verder heeft de ontwikkelaar ook de mogelijkheid om de *backend* te integreren met de klantzijde. ASP.NET MVC, PHP en JSP zijn momenteel de ondersteunde technologieën voor *backend* integratie. Op het moment van schrijven is Kendo UI aan versie 2013 Q2 [134].

3.2.1 Omkadering

Programmeertaal Kendo UI is zowel JavaScript- en opmaakgedreven. Data-attributen kunnen een HTML-element associëren met Kendo UI. Ook kan een jQuery-selector in JavaScript met Kendo UI worden gebonden. Alle GI-elementen van Kendo UI Mobile kunnen met data-attributen worden opgebouwd. *Widgets* van Kendo UI Web kunnen zowel met JavaScript als data-attributen geïnitialiseerd worden.

Het laden van het raamwerk kan door Kendo UI in zijn geheel op te roepen of enkel Kendo UI Mobile met respectievelijk `kendo.all.js` of `kendo.mobile.js`. Elk van de drie luiken - Web, Mobile en DataViz - kan op zichzelf functioneren door hun JavaScript-bestand in te laden. Er kan wel slecht één van de drie scripts gelijktijdig gebruikt worden. Wanneer elementen uit verschillende luiken gebruikt worden, moet `kendo.all.js` worden gebruikt. Een alternatieve oplossing is het inladen van één luik en een JavaScript-bestand met de benodigde componenten. Dit bestand is te genereren met de JavaScript Download Builder die te vinden is op de Kendo UI-website [134]. Hier kunnen de vereiste elementen geselecteerd worden en wordt het

JavaScript-bestand gegenereerd. Op een analoge wijze kan de programmeur kiezen tussen verschillende CSS-bestanden: `kendo.all.css` en `kendo.mobile.css`.

Tools Op de Kendo UI-website staan drie webapplicaties vermeld die Telerik aanbiedt om de programmeur te ondersteunen. De eerste is Kendo UI Dojo [131], een interactieve leeromgeving om met Kendo UI vertrouwd te raken. De gebruiker kan de basis van Kendo UI leren kennen met geleide handleidingen en uitvoerbare voorbeelden. De twee andere webapplicaties zijn een ThemeBuilder voor Web en Mobile die op een grafische manier CSS-bestanden kunnen genereren [132, 133]. Voor Kendo UI Mobile kan een verschillende lay-out bepaald worden voor alle ondersteunde platformen.

Documentatie Twee belangrijke secties van de documentatie zijn de API en Getting Started [130]. Beide kunnen op elkaar gemapt worden omdat alle objecten van Kendo UI die in de API worden aangehaald ook in een pagina onder Getting Started worden besproken. Deze laatste probeert met meer woorden en voorbeelden uit te leggen wat het object juist inhoudt. Verder staan er bij de documentatie nog handleidingen die complexere functionaliteit uit de doeken doen. Ook zijn er demo's die live voorbeelden tonen samen met de code die nodig is om het voorbeeld te maken.

Marktadoptatie Enkele van de populairste klanten van Kendo UI zijn Nikon, Fujifilm en Symantec [134].

Licenties Een licentie voor Kendo UI Complete kost $699 per ontwikkelaar. Voor *backend* ondersteuning in PHP, JSP of ASP.NET MVC moet $300 meer betaald worden. Hierbij zijn één jaar updates mogelijk en wordt professionele ondersteuning aangeboden met een responstijd onder de 48 uur. Een licentie met *backend* integratie garandeert professionele ondersteuning binnen de 24 uur. Voor Kendo UI Web, Mobile en DataViz bestaan ook aparte licenties voor respectievelijk $399, $199 en $399 [134].

3.2.2 Code en ontwikkeling

Kendo UI steunt op de jQuery-bibliotheek en deze moet ingeladen worden voor het Kendo UI-raamwerk zelf wordt aangeroepen. De initialisatie van een applicatie moet onderaan in de HTML-pagina gebeuren. Hier kunnen parameters meegegeven worden die bijvoorbeeld de stijl van één platform vastleggen of het initiële scherm bepalen.

De navigatie naar een scherm gebeurt op basis van de *identifier* van dat scherm. Standaard navigeert Kendo UI naar het eerst gedefinieerde scherm in de pagina. Een ander scherm kan in diezelfde pagina of in een ander bestand staan. Een lokale navigatie wordt herkend door een *hashtag* die voor de *identifier* van het scherm wordt geplaatst als parameter van de `navigate`-methode. Navigatie naar een ander bestand kan door de bestandsnaam als parameter op te geven.

3.2.3 Functionele kenmerken

Kendo UI steunt op het MVVM-ontwerppatroon, wat de functionele kenmerken sterk beïnvloedt. Alle objecten maken deel uit van een klassensysteem.

GI-elementen Formulieren volgen de HTML5-norm. Deze elementen zijn wel enkel functioneel vanaf iOS 5 en Android 4. De stijl van de elementen op andere platformen zal werken, maar is beperkt tot enkel tekstinvoer [134].

Het toevoegen van knoppen kan zowel met de `button`-tag als met standaard hyperlinks. Knoppen kunnen ook samengevoegd worden tot een `ButtonGroup`. Dit maakt het mogelijk om gemeenschappelijke acties aan een groep van knoppen toe te kennen om bijvoorbeeld een menu te maken. Een `TabStrip` is een alternatief waar tabs in de voettekst het scherm kunnen laten variëren.

View Analoog als het MVC-ontwerppatroon, worden schermen voorgesteld met `Views`. Een `View` aanduiden gebeurt door het attribuut `data-role` aan `view` gelijk te stellen. `Views` kunnen met een lay-out worden voorzien met de `data-layout`-tag. Een `Layout` bepaalt de vormgeving van een `View` en kan hergebruikt worden.

Een `ListView` is een specifieke `View` voor lijsten. De `data-template` kan bij lijsten de *identifier* van een sjabloon bevatten die de opmaak van de lijstelementen definieert. Deze sjablonen zijn specifieke Kendo UI-scripts die HTML-tags en JavaScript-code kunnen bevatten. Ook kunnen ze verwijzen naar velden van het model dat aan de lijst is toegekend (zie infra).

Twee andere instanties van `Views` zijn `SplitView` en `ScrollView`. De eerste kan het scherm in twee `Views` splitsen, vaak gebruikt bij applicaties op de tablet. De tweede definieert een verzameling van pagina's die met *swipe* bewegingen gelinkt zijn.

View Model Het View Model behoort tot de kern van Kendo UI en wordt `ObservableObject` genoemd. Dit is een JavaScript-object dat kan gebonden worden aan abonnees. Het ondersteunt het monitoren van wijzigingen en verwittigt elke abonnee wanneer een wijziging zich voordoet. Een `ObservableObject` kan aan een `View` worden toegekend door het in de `data-model`-tag te vermelden.

Er zijn verschillende bindingen mogelijk tussen een `View` en `ObservableObject`. Deze wordt aangegeven in de `data-bind`-tag. Als een gebonden eigenschap wijzigt - door gebruikersinvoer of programmatisch - zal het overeenkomstige veld in het `ObservableObject` ook wijzigen.

Model Het `Model` erft over van `ObservableObject` en breidt het uit met de mogelijkheid om schema's, velden en methoden te definiëren. Velden kunnen van het type `string, number, boolean` of `date` zijn. Ook kunnen de velden verder beschreven worden door bijvoorbeeld een standaard waarde of validatie toe te voegen. De `DataSource` is een Kendo UI-object voor de opslag van lokale of externe data. Deze ondersteunt alle CRUD (*Create, Read, Update en Delete*) operaties en het sorteren, pagineren, filteren, groeperen en aggregeren van data. Het `schema`-attribuut legt de structuur van de data in de `DataSource` vast. Bij externe databronnen bepaalt het hoe binnenkomende data geparset moet worden om aan een opgelegde structuur te voldoen. Een `schema` kan van een `Model` worden voorzien of zelf een `Model` beschrijven. In het eerste geval zullen de waarde van het toegekende `Model` na

CRUD-operaties worden aangepast. In het andere geval zal de `DataSource` instanties van het beschreven `Model` bevatten en manipuleren.

3.2.4 Niet-functionele kenmerken

Performantie De performantie van een Kendo UI-applicatie wordt deels bepaald door de programmeur. Deze moet er voor zorgen dat de data op het juiste moment geladen wordt. Bij het weergeven van een `View` gaan drie gebeurtenissen vooraf, namelijk `beforeShow, init` en `show`. De eerste wordt uitgevoerd voor een `View` zichtbaar wordt, de tweede na initialisatie en de laatste bij het tonen van een `View`. Het initialiseren van een `View` vindt maar één keer plaats nadat de volledige applicatie geladen is. Bij de ontwikkeling van een `ListView` met data van een externe databron kan de `DataSource` geladen worden bij het initialiseren van de applicatie, de lijst gemaakt worden bij de `init`-gebeurtenis en de lijst ververst worden bij een `show`-gebeurtenis.

Aanpasbaarheid Kendo UI probeert de *native look-and-feel* van verschillende besturingssystemen na te bootsen. Het Kendo UI-pakket bevat ook tien extra thema's die een alternatieve lay-out bepalen. Deze zijn elk nog persoonlijk aan te passen met de Mobile ThemeBuilder zoals beschreven in de sectie 3.2.1.

Programmeerbaarheid Een kennis van zowel HTML als JavaScript is vereist om met dit raamwerk aan de slag te kunnen. Het raamwerk is gebouwd op de jQuery-bibliotheek en maakt dus vaak van jQuery-*selectors* gebruik. Ook kan een AJAX-verzoek met jQuery-syntax geformuleerd worden om externe data voor een `DataSource` op te halen.

Browserondersteuning Ondersteunde systemen zijn iOS, Android, Windows Phone 8 en BlackBerry OS. Alle *widgets*, zoals gebruikt in het raamwerk, ondersteunen *progressive enhancement*. Oudere browsers kunnen zo bestaande inhoud en functionaliteit raadplegen met *native* HTML-types indien bepaalde elementen niet worden ondersteund. Ook de HTML5-formulierelementen worden opgebouwd met *progressive enhancement*.

3.3 jQuery Mobile

jQuery Mobile (jQM) is een opmaakgedreven raamwerk dat hoofdzakelijk GI-elementen aanbiedt. Het werd aangekondigd in 2010 [98] en in november 2011 werd versie 1.0 uitgebracht [90]. Een jaar later werd in oktober versie 1.2 uitgebracht [91] en op het moment van schrijven zit jQuery Mobile aan versie 1.3.1 [92]. Het raamwerk is afhankelijk van jQuery dat een JavaScript-bibliotheek is. Beide worden beheerd door het jQuery Project [59]. jQuery Mobile wordt door onder andere Adobe, BlackBerry en Mozilla gesponsord [56].

3.3.1 Omkadering

Programmeertaal Om met jQuery Mobile aan de slag te kunnen, is er kennis nodig over HTML, CSS en JavaScript. Alle GI-elementen worden geschreven in HTML en aangeduid met data-attributen. Functionele vereisten kunnen worden geschreven in JavaScript.

Tools Codiqa [117] is een tool om snel de GGI van de applicatie op te zetten. Door *drag-and-drop* worden de GI-elementen op het scherm gezet. Codiqua zal automatisch op de achtergrond de HTML- en CSS-code voorzien. Daarnaast biedt jQuery Mobile ook ThemeRoller [61] aan om het standaard kleurenthema aan te passen. Hiermee worden kleuren naar een voorbeeldapplicatie gesleept, waarna het overeenkomstige CSS-bestand kan worden gedownload.

Documentatie Op de documentatiesite van versie 1.2 [57] is een catalogus te vinden van alle mogelijke elementen waarover jQuery Mobile beschikt. Door de broncode van een voorbeeld te bekijken, kan de code worden geschreven om tot dat resultaat te komen.

Naast de GI-elementen is er ook documentatie over de API. Deze gaat over gebeurtenissen en methoden die kunnen worden gebruikt en het configureren van standaardwaarden.

Marktadoptatie Op de website van jQuery Mobile wordt een reeks applicaties getoond die gemaakt zijn met hun raamwerk. Enkele voorbeelden zijn webapplicaties voor Ikea, Disney World, Stanford University en Moulin Rouge [56].

Licenties Sinds september 2012 is het enkel nog mogelijk om jQuery Mobile onder de MIT-licentie te verkrijgen [25]. Dit betekent dat de code wordt vrijgegeven als *open-source* en dat deze tegelijkertijd kan worden gebruikt in propriëtaire projecten en applicaties [94].

3.3.2 Code en ontwikkeling

Zoals werd aangehaald, wordt voornamelijk HTML5-code geschreven voorzien van data-attributen. Daarna zal het raamwerk door middel van *progressive enhancement* allerhande code toevoegen om de beoogde GI-elementen correct te tonen in de browser. Dit wordt verder uitgelegd in de sectie browserondersteuning (zie 3.3.4).

3.3.3 Functionele kenmerken

jQuery Mobile is een raamwerk dat voornamelijk GI-elementen aanbiedt, met name pagina's en dialoogvensters, werkbalken, knoppen, inhoud vormgeven, elementen voor formulieren en lijsten [57]. Deze kenmerken zijn gebaseerd op versie 1.2.

- **Pagina's en dialoogvensters** De basisstructuur van een pagina bestaat uit een koptekst, inhoud en voettekst. Bij het overgaan naar een andere pagina wordt gekozen uit tien overgangseffecten. Voordat deze overgang gebeurt, zal jQuery Mobile altijd eerst die pagina ophalen via AJAX en inladen in het

DOM. Zo kan een soepel overgangseffect worden getoond aan de gebruiker. Daarnaast is het ook mogelijk om gelinkte pagina's op voorhand op te halen. Als laatste biedt jQuery Mobile ook dialoogvensters en pop-ups aan.

- **Werkbalken** Het is mogelijk om zowel knoppen bij de koptekst als bij de voettekst te plaatsen. Bij deze laatste kunnen typisch meer knoppen geplaatst worden, bij de koptekst slechts twee. Daarnaast is het ook mogelijk om navigatiebalken te maken. Aan zowel de werk- als navigatiebalken kunnen iconen worden toegevoegd.

- **Knoppen** Het is mogelijk om knoppen te plaatsen op het scherm. Ook hier is er terug een variëteit aan mogelijkheden: grote of kleine, met iconen of zonder, gegroepeerd of niet.

- **Inhoud vormgeven** De inhoud van de pagina kan worden vormgegeven door gebruik te maken van een rooster. jQuery Mobile laat roosters tot vijf kolommen toe. Daarnaast zijn er ook nog opklapbare blokken ter beschikking. Als laatste kunnen deze blokken ook samengevoegd worden tot een accordeon.

- **Elementen voor formulieren** jQuery Mobile biedt elementen voor formulieren aan zoals tekstinvoer, een dropdownmenu, een zoekveld, een *slider* en een schakelaar. Het raamwerk verplicht om de `label`-tag te gebruiken. Zo wordt de applicatie toegankelijker gemaakt voor bijvoorbeeld mensen met een schermlezer.

- **Lijsten** Een laatste categorie GI-elementen die jQuery Mobile aanbiedt, zijn lijsten. Deze gaan van standaard ongeordende lijsten tot lijsten met alle soorten decoraties als iconen, afbeeldingen, telbubbels en verdelers. Het is ook mogelijk om in deze lijsten te zoeken. Hiervoor dient de ontwikkelaar enkel één data-attribuut toe te voegen, waarna het raamwerk de implementatie voorziet.

3.3.4 Niet-functionele kenmerken

Performantie Zoals gezegd schrijft de ontwikkelaar HTML5-code met specifieke data-attributen en zal het raamwerk daarna de code verder aanvullen. Dit gebeurt enkel op de pagina die de gebruiker op dat moment bekijkt. Dit gaat dus ook op voor een webapplicatie waarbij alle schermen op één webpagina zijn geschreven. Deze webpagina bevat allemaal `div`-verpakkingen voor ieder scherm. jQuery Mobile zal enkel die `div` verder aanvullen die op dat moment getoond wordt aan de gebruiker.

Aanpasbaarheid Er is keuze uit vijf kleurenthema's die kunnen worden toegepast op de gehele applicatie of enkel op bepaalde elementen. Een ontwikkelaar kan ook enkel de structuur downloaden en zelf het thema in CSS schrijven. Doordat dit laatste heel wat inspanning vraagt, kunnen ontwikkelaars hiervoor ThemeRoller [61] gebruiken.

Programmeerbaarheid Bij het programmeren in jQuery Mobile wordt geen ontwerppatroon afgedwongen. De code voor de GI-elementen wordt tenslotte als HTML5-code geschreven. Voor de echte functionaliteit wordt beroep gedaan op JavaScript en meer bepaald op de jQuery-bibliotheek. Ook deze dwingt geen ontwerppatroon af.

Browserondersteuning jQuery Mobile maakt gebruikt van *progressive enhancement* (zie 2.5.1). Hierdoor wordt in principe ieder apparaat ondersteunt door het raamwerk, maar zal de applicatie op een ouder apparaat minder functionaliteit krijgen dan diezelfde applicatie op een nieuwer apparaat. Om dit te verduidelijken deelt jQuery Mobile browsers op in drie verschillende klassen: A, B en C [60]. Hierbij krijgt een klasse A browser de volledige ervaring met AJAX-gebaseerde paginaovergangen. Bij een browser van klasse B wordt geen AJAX ondersteund. Als laatste wordt bij een klasse C browser enkel een basiservaring aangeboden die nog steeds functioneel is.

3.4 Lungo

Lungo (L) is een opmaakgedreven raamwerk onderhouden door TapQuo dat een Spaans bedrijf is gespecialiseerd in mobiele gebruikerservaring [128]. Het raamwerk biedt vooral GI-elementen aan en is afhankelijk van een JavaScript-bibliotheek, namelijk QuoJS [123]. Versie 1.0 kwam uit in 2011 [122] en op het moment van schrijven zit Lungo aan versie 2.1 [123].

3.4.1 Omkadering

Programmeertaal Er is kennis nodig van HTML, CSS en JavaScript. De GI-elementen worden geschreven in HTML en aangeduid aan de hand van zowel CSS-klassen en data-attributen. Functionele vereisten kunnen worden geschreven in JavaScript.

Tools Er worden geen specifieke tools door TapQuo aangeboden om de ontwikkeling te versnellen.

Documentatie Op de documentatiesite [125] wordt eerst getoond hoe het structuur van een Lungo-applicatie er uitziet. Vervolgens zijn er nog acht andere pagina's die de aangeboden GI-elementen en API uitleggen. Op de documentatiepagina's is altijd eerst de broncode te zien. Er kan dan boven de broncode op een knop geklikt worden, waarna een live voorbeeld wordt getoond.

Marktadoptatie Doordat geen informatie te vinden is op de site van Lungo werd een e-mail gestuurd met de vraag welke bedrijven Lungo gebruiken of in welke projecten van TapQuo zelf Lungo werd gebruikt. De auteurs hebben tot op het moment van schrijven geen antwoord ontvangen.

Licenties Het raamwerk wordt onder de GNU GPLv3-licentie vrijgegeven. Daarnaast is een commerciële versie ook mogelijk [124].

3.4.2 Code en ontwikkeling

Zoals gezegd wordt een Lungo-applicatie geprogrammeerd vanuit geannoteerde HTML-code door middel van CSS-klassen en data-attributen. Er wordt geen ontwerppatroon zoals MVC afgedwongen.

Om de verschillende schermen te scheiden wordt gebruik gemaakt van `article`- en `section`-tags die specifiek zijn voor HTML5. Analoog worden ook voor de kop- en voettekst de `header`- en `footer`-tags gebruikt.

Enerzijds kan de volledige applicatie op één webpagina worden geprogrammeerd. Daarnaast biedt Lungo ook de mogelijkheid om de verschillende schermen op afzonderlijke pagina's op te slaan. Hierbij moet enkel de code binnen de `body`-tag worden opgeslagen. Daarna wordt bij de initialisatie van Lungo gedefinieerd welke afzonderlijke pagina's asynchroon dienen te worden opgehaald.

3.4.3 Functionele kenmerken

De documentatiepagina [125] is opgedeeld in negen groepen die hieronder kort worden aangehaald.

- **Prototype** Hierop worden GI-elementen, navigatie, formulieren, scrollen, lijsten en data-attributen aangehaald. Ook is er directe ondersteuning door het raamwerk voor Pull&Refresh. Dit is een lijst die de data automatisch herlaadt wanneer de lijst naar beneden wordt getrokken.

- **Core** Hier worden functies aangehaald die het raamwerk zelf intern gebruikt. Een voorbeeld hiervan is de `Lungo.Core.isMobile`-functie die controleert of de applicatie in een mobiele omgeving wordt uitgevoerd. Een ander voorbeeld is de aanwezigheid van een sorteer- en zoekfunctie voor rijen.

- **Data** Het raamwerk biedt *wrappers* voor cache, opslag en SQL. De ontwikkelaar dient dus zelf geen `if-then-else`-constructies op te stellen of een bepaald kenmerk ondersteund wordt of niet, het raamwerk neemt dat voor zijn rekening. Indien het apparaat het kenmerk niet ondersteunt, voorziet Lungo een *fallback*.

- **DOM** Het manipuleren van de DOM gebeurt door de QuoJS-bibliotheek. Hiervoor wordt gerefereerd naar de documentatie van QuoJS [127]. Zelf biedt Lungo enkel ondersteuning voor gebeurtenissen bij het laden en ontladen van pagina's binnen de applicatie.

- **Element** Toevoegen van een telbubbel, vooruitgangsbalk, Pull&Refresh, laadbalk en een fotocarousel zijn programmeerbaar en manipuleerbaar via de Lungo API vanuit JavaScript.

- **Notification** Er worden standaard verschillende dialoogvensters aangeboden voor fout-, succes- en infomeldingen. Deze dialoogvensters zijn voorzien van een specifieke lay-out. Zo zal een fout- en succesboodschap respectievelijk een rode en groene achtergrondkleur hebben. Een ander type venster dat wordt

aangeboden is een bevestigingsvenster waarbij een vraag wordt gesteld aan de gebruiker waar ja of nee op geantwoord wordt. Het raamwerk voorziet voor beide acties de mogelijkheid om hier een eigen functie aan te koppelen.

- **Routing** Hier wordt ingegaan op de navigatie doorheen de applicatie vanuit JavaScript. Het is mogelijk om binnen eenzelfde artikel tussen verschillende secties te navigeren alsook van artikel te veranderen. Daarnaast is het mogelijk om vanuit JavaScript een zijnavigatie te tonen. Deze navigatie wordt zichtbaar door links bovenaan te klikken. Als laatste voorziet Lungo een terugkeerfunctionaliteit, vergelijkbaar met de functie `history.back` uit JavaScript.
- **Service** Versturen van HTTP-verzoeken kan vanuit Lungo zelf. Zowel HTTP GET als HTTP POST zijn mogelijk. Het raamwerk kan omgaan met tekst, XML, JSON en HTML als antwoord. Er is ook een verkorte notatie indien het antwoord JSON is. Als laatste biedt Lungo ook de mogelijkheid om het antwoord van het HTTP-verzoek te cachen, waarbij de maximale tijd moet worden opgegeven. Indien de oproep nog eens gebeurt en de maximale tijd niet verstreken is, zal Lungo geen HTTP-verzoek versturen, maar het opgeslagen antwoord gebruiken.
- **View** Hier worden JavaScript-methoden aangeboden om het uitzicht binnen een `article` te bepalen. Zo kan de titel worden aangepast of van sectie binnen het artikel worden veranderd. Daarnaast wordt ook ingegaan op het tonen en verbergen van de zijnavigatie.

3.4.4 Niet-functionele kenmerken

Performantie De JavaScript-bibliotheek waarop Lungo steunt is geoptimaliseerd voor mobiel gebruik. Hierdoor bevat het geen methoden voor desktopgebruikers, waardoor het bestand kleiner is dan traditionele JavaScript-bibliotheken.

Aanpasbaarheid TapQuo biedt zelf geen tools aan om de kleuren of het uitzicht van de applicatie te veranderen. Hierdoor zal een ontwikkelaar zelf aangewezen zijn om eigen CSS-code te schrijven.

Programmeerbaarheid Lungo dwingt geen enkel ontwerppatroon af. Voor de echte functionaliteit wordt beroep gedaan op JavaScript. Daarbij is de ontwikkelaar vrij hoe hij te werk gaat en kan hij gebruik maken van de Lungo API. Voor aanpassingen die van toepassing zijn op DOM-manipulatie, is de ontwikkelaar aangewezen op QuoJS. Ook deze dwingt geen ontwerppatroon af.

Ondersteuning browser TapQuo geeft aan op hun website dat ze ondersteuning bieden voor iOS, Android, Blackberry OS en Firefox OS.

3.5 Overzicht

In tabel 3.1 wordt de passieve vergelijking van de raamwerken getoond. In het volgende hoofdstuk worden de vijf actieve vergelijkingscriteria besproken. De resultaten

van de actieve vergelijking kunnen niet worden bekomen door enkel naar de literatuur te kijken, maar moeten actief onderzocht worden. De resultaten hiervan zullen in hoofdstuk 5 besproken worden.

Criterium	**Sencha Touch**	**Kendo UI**	**jQuery Mobile**	**Lungo**
Beheerder	Sencha	Telerik	jQuery Project	TapQuo
Eerste stabiele versie	2010	2011	2011	2011
Huidige stabiele versie	2.2.0	2013 Q2	1.3.1	2.1
Afhankelijke bibliotheken	/	jQuery	jQuery	QuoJS
Type	JavaScript-gedreven	JavaScript- en opmaakgedreven	Opmaakgedreven	Opmaakgedreven
Ontwerppatroon	MVC	MVVM	/	/
Licentie	GNU GPLv3 & commercieel	commercieel	MIT	GNU GPLv3 & commercieel
Licentiekost	gratis	$699	gratis	gratis
Professionele ondersteuning	ja	ja	nee	nee
Kosten ondersteuning	$299	ja (zit in licentiekost)	/	/
Native look-and-feel	nee	ja	nee	nee
Inpakken tot *native* applicatie	ja	nee	nee	nee
Tools voor GI	Sencha Architect	nee	Codiqa	nee
Tools voor thema	nee	Mobile ThemeBuilder	ThemeRoller voor jQuery Mobile	nee
Tools voor uitrollen	Sencha Cmd	nee	nee	nee

Tabel 3.1: Passieve vergelijking.

Hoofdstuk

4 Vergelijkingscriteria

Dit hoofdstuk bekijkt hoe de mobiele HTML5-raamwerken actief zullen worden vergeleken. Hoofdzakelijk zal dit gebeuren aan de hand van een POC die in sectie 4.1 wordt uitgelegd. De gekozen vergelijkingscriteria worden in sectie 4.2 besproken. Deze criteria worden actieve criteria genoemd.

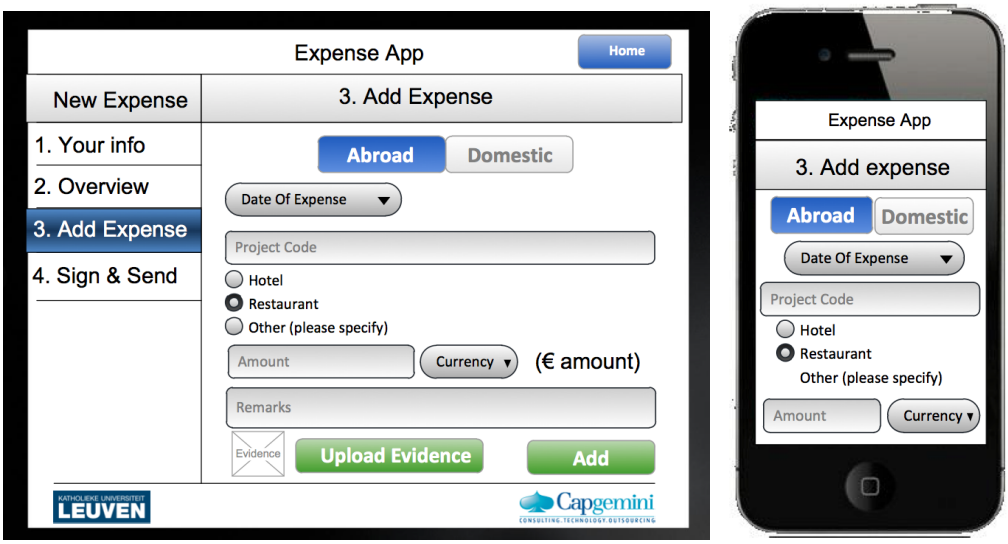
4.1 POC

In samenspraak met Capgemini werd gekozen om een POC op te stellen. Dit is een idee waarmee de uitvoerbaarheid in de verschillende raamwerken kan worden nagegaan. Oehlman et al. [87] gebruikten ook een POC om raamwerken te vergelijken.

Verschillende vergaderingen werden georganiseerd met Capgemini om tot een idee te komen dat vooral in de bedrijfswereld van toepassing is. Het uiteindelijke idee is een applicatie die het mogelijk maakt voor werknemers om hun onkosten via hun mobiel apparaat door te sturen. Dit werd uitgewerkt door Capgemini en geleverd aan de auteurs als *mockup*. Dit is een voorstelling van de applicatie als een reeks schermen zoals deze er zullen uitzien op een apparaat. Een voorbeeld van zo een scherm is te vinden op figuur 4.1. Op de *mockups* staan naast de schermen ook functionele vereisten die op dat scherm van toepassing zijn. Deze vereisten worden hier niet weergegeven, maar zullen gebruikt worden in sectie 4.2.3. De bedoeling van Capgemini is dat deze POC wordt uitgewerkt zowel voor smartphone als tablet, zowel voor Android als iOS, zowel voor staande als liggende apparaten en zowel voor online als offline gebruik.

4.1.1 Aspecten

Een werknemer meldt zich eerst aan op de applicatie en kan daarna ofwel een nieuw onkostenformulier aanmaken of zijn doorgestuurde onkostenformulieren bekijken. De term onkostenformulier is een groepering van meerdere onkosten met bijhorende bewijsstukken en de handtekening van de werknemer. Het aanmaken van een nieuw onkostenformulier verloopt in vier stappen. Indien de werknemer al eerder begonnen was met het aanmaken van een formulier, zal hij worden gevraagd of hij verder wil gaan met dat formulier of met een nieuw formulier wil starten.



Figuur 4.1: POC bij het toevoegen van een nieuwe onkost met aan de linkerkant de weergave op een tablet en aan de rechterkant deze op een smartphone.

1. De eerste stap is het bekijken en/of aanpassen van de persoonlijke informatie van de werknemer. Bij het aanpassen van deze gegevens zullen deze worden gevalideerd. Indien deze validatie faalt, krijgt de werknemer een dialoogvenster te zien met de reden tot falen. Ook worden de foute velden rood gemarkeerd.

2. In de tweede stap kan de werknemer zijn toegevoegde onkosten aan het formulier bekijken. In het begin is deze lijst leeg, tenzij hij eerder een formulier aan het invullen was (zie infra). Indien deze lijst onkosten bevat, is het mogelijk om hierop te klikken en deze te bekijken. Aanpassen is niet mogelijk.

3. In stap drie kan een nieuwe onkost worden toegevoegd. Dit kan ofwel een binnenlandse ofwel buitenlandse onkost zijn. Voor beide dient een datum en projectcode te worden opgegeven. De eerstgenoemde is een *datepicker* die teruggaat tot twee maanden in de tijd. De laatstgenoemde bevat automatische aanvulling, maar de werknemer is niet verplicht om een projectcode uit de aanvulling te selecteren. Daarnaast dient het type en bedrag van de onkost, alsook een bewijsstuk te worden opgegeven. Bij een buitenlandse onkost moet de munteenheid worden opgegeven, waarna de applicatie deze automatisch omvormt naar euro. Het scherm voor het toevoegen van een buitenlandse onkost wordt getoond op figuur 4.1. Net zoals bij stap één geldt ook hier validatie op de formuliervelden.

4. In deze laatste stap dient een handtekening te worden geplaatst waarna het formulier kan worden doorgestuurd. Indien de gebruiker offline werkt, zal deze worden opgeslagen op het toestel. De werknemer kan het formulier opnieuw doorsturen zodra hij terug online is.

Bij het bekijken van de doorgestuurde formulieren is het mogelijk om per formulier de bijhorende PDF te downloaden. Deze bevat een overzicht van de onkosten met bijhorende bewijsstukken, alsook de handtekening van de werknemer.

4.2 Actieve criteria

In deze sectie zullen de actieve criteria toegelicht worden die zullen worden toegepast om de raamwerken te vergelijken. In sectie 2.7 werden reeds technieken besproken die in de literatuur worden toegepast. Elementen van deze technieken zullen terugkomen in de voorgestelde methode om de raamwerken te evalueren. Sectie 3.5 bevatte de passieve vergelijkingscriteria.

Vijf criteria zullen worden gebruikt: populariteit (4.2.1), productiviteit (4.2.2), gebruik (4.2.3), ondersteuning (4.2.4) en performantie (4.2.5). Elk raamwerk krijgt voor elk criterium een score afgeleid uit een formule. De opzet van deze formules is vergelijkbaar met de opzet van het puntensysteem dat werd gebruikt op een blogpost van Codefessions [102]. Deze scores zullen in een spinnenweb worden ondergebracht (zie sectie 4.3). Capgemini moedigde het gebruik van een spinnenweb aan om de vergelijking te visualiseren. Ze verwezen naar een blogpost waarbij een spinnenweb werd gebruikt om *native* en webapplicaties met elkaar te vergelijken [47]. Zoals hierboven vermeld zal een POC gebruikt worden bij de vergelijking. De implementatie van deze POC zal het gebruiks- en ondersteuningscriterium drijven. Dit komt omdat Capgemini de POC zo heeft opgesteld dat het de verschillende functionaliteiten bevat die van een normale applicatie verwacht worden.

De ISO 25010-standaard wordt gekoppeld in alle criteria, behalve bij populariteit. AHP werd niet gebruik omwille van twee redenen. Ten eerste moet, zoals aangehaald door Jadhav et al. [46], alles opnieuw worden geëvalueerd als er nieuwe criteria of kandidaten worden toegevoegd. De methode die hier wordt voorgesteld lijdt niet onder dit probleem. Ten tweede is de paarsgewijze vergelijking relatief ten opzichte van de raamwerken. De methode die hier wordt voorgesteld laat onderzoekers toe raamwerken te evalueren zonder met de andere kandidaten rekening te houden. Dit is niet mogelijk bij de AHP-methode. Een laatste opmerking is dat bij AHP gewichten aan de criteria worden gegeven. De voorgestelde methode stelt het gewicht van ieder criterium gelijk, maar het is echter mogelijk om hier van af te wijken.

4.2.1 Populariteit

De populariteit van een raamwerk is een belangrijke factor want het bepaalt de gemeenschap en levendigheid van het raamwerk. De definitie van gemeenschap op de blogpost van Codefessions [102] zegt ook dat dit een belangrijke factor is omdat het de toekomstige ontwikkeling en de hulp bij het gebruik van het raamwerk aantoont. De populariteit kan in cijfers worden uitgedrukt door gebruik te maken van sociale netwerken. Een tabel zal voorzien worden met in de rijen het aantal volgers op Twitter, sterren en *forkers* van GitHub, vragen op Stack Overflow en aantal vind-ik-leuks van Facebook. Twitter en GitHub worden ook door Hales [41] bekeken wanneer HTML5-raamwerken worden voorgesteld. Stack Overflow vragen

worden als criterium op een blogpost van Monocaffe gebruikt om mobiele raamwerken te vergelijken [9]. Het aantal vind-ik-leuks van Facebook werd zelf geïntroduceerd, omdat dit net zoals Twitter een belangrijk netwerk is.

GitHub kan worden gezien als een sociaal netwerk voor programmeurs [17] en bepaalt dus mee de actieve gemeenschap rond het raamwerk. Raamwerken die niet op GitHub te vinden zijn krijgen nul voor zowel het aantal sterren en *forkers*. Een alternatief hield de interpolatie van de GitHub data van de overige raamwerken in. Omdat deze aanpak het raamwerk onterecht zou bevoordelen, is hier niet voor gekozen.

De som van Twitter volgers (T_r), GitHub sterren (S_r), GitHub *forkers* (F_r), Stack Overflow vragen (SO_r) en Facebook vind-ik-leuks (FB_r) vormt de score voor het populariteitscriterium:

$$\text{Populariteit}_r = T_r + S_r + F_r + SO_r + FB_r \tag{4.1}$$

voor een raamwerk r.

Omdat deze gegevens zeer dynamisch zijn, zullen verschillende metingen in de tijd de evolutie van de data weergeven. Ook zullen de uitkomsten van dit criterium worden vergeleken met data geleverd door Google Trends [37]. Deze webapplicatie toont de evolutie van zoektermen op Google op een schaal van 100, waarbij 100 overeenkomt met de grootste zoekinteresse. Voor elk raamwerk zal het aantal zoekopdrachten op Google in functie van de tijd worden uitgezet.

Er bestaat geen exacte formule om populariteit uit te drukken. De formule die werd gekozen om de score voor dit criterium te quoteren is onderheven aan subjectiviteit. Twee opmerkingen moeten hierbij worden gemaakt. Enerzijds zijn de auteurs zich ervan bewust dat de doorsnede tussen sociale netwerken niet leeg is. Zo kan éénzelfde persoon zowel een volger op Twitter zijn als een vind-ik-leuk op Facebook plaatsen. Verschillende individuen zullen dus dubbel geteld worden in de totale score voor populariteit van het raamwerk. De score zal dus slechts een indicatie geven over de populariteit, het is geen exacte weergave. Ten tweede zijn de auteurs er zicht van bewust dat de inclusie van Stack Overflow op twee manieren kan worden bekeken. Enerzijds kunnen veel vragen duiden op veel onduidelijkheden over het raamwerk. Anderzijds kan dit een maat zijn voor de populariteit van dit onderwerp. De auteurs zijn van mening dat de tweede zienswijze correcter is dan de eerste en het dus valide is Stack Overflow in de formule op te nemen.

4.2.2 Productiviteit

De productiviteit moet weergeven hoe lang het duurt om met het raamwerk vertrouwd te raken en iets nuttig te kunnen bouwen. Dit is belangrijk omdat bedrijven zo min mogelijk tijd willen verliezen om met nieuwe technologieën aan de slag te kunnen. In de ISO 25010-standaard is de categorie van bruikbaarheid en efficiëntie vergelijkbaar met dit criterium.

De auteurs zullen elk de POC in twee verschillende raamwerken maken en daarnaast ook een extra loginapplicatie in twee andere raamwerken. De ene auteur maakt de POC in jQuery Mobile en Lungo en de loginapplicatie in Sencha Touch en

Kendo UI. De andere zal dan de POC in Sencha Touch en Kendo UI maken en de loginapplicatie in jQuery Mobile en Lungo. De tijd die nodig is om de volledige POC te implementeren is een indicatie voor de productiviteit. Er wordt verondersteld dat de auteurs over een gemeenschappelijke technische achtergrond beschikken. Toch kunnen beide onderling verschillen in efficiëntie, waardoor de productiviteit verschilt. Dit probleem is inherent aan dit criterium doordat er geen eenduidige manier is om productiviteit te bepalen. Toch is het belangrijk om een schatting op deze manier te kunnen maken.

Er zijn echter vijf redenen waarom de implementaties van de POC geen goede indicatie zijn voor de productiviteit. Deze werden door de auteurs ervaren wanneer de implementatie in het tweede raamwerk werd uitgevoerd:

1. Betere ervaring met de POC versnelt bij de tweede implementatie het overzicht van functionaliteit dat moeten worden geïmplementeerd.

2. Een verbeterde ervaring met HTML5-raamwerken had een positieve invloed op de verdere implementaties. Dit weerspiegelde zich vooral tussen jQuery Mobile en Lungo. Hoewel ze beide op een verschillende JavaScript-bibliotheek steunen - respectievelijk jQuery en QuoJS - zijn de gelijkenissen tussen deze twee raamwerken groot. Ook leggen ze beide geen ontwerppatroon op.

3. Er kon code, zoals van de implementatie in jQuery Mobile, overgenomen worden bij de implementatie van de POC met Lungo en Kendo UI.

4. Er kwamen bij de eerste implementatie problemen met de *backend* naar boven. Deze waren bij de tweede implementatie reeds opgelost. Door het onnauwkeurig opmeten van de tijd kan er geen schatting worden gemaakt van de tijd die aan de problemen van de *backend* werden besteed.

5. Niet de volledige POC kon met Lungo en Sencha Touch worden ontwikkeld.

De implementatie van de loginapplicatie is een alternatieve test van de productiviteit. Deze applicatie bevat GI-elementen, validaties, *backend* integratie en een lijst. De implementatie van de loginapplicatie kan dus als voldoende steekproef beschouwd worden om ervaring met een raamwerk te testen. Na het aanmelden met deze applicatie zal de gebruiker een lijst van 850 elementen te zien krijgen. Deze lijst is bedoeld als stresstest om de performantie te testen (zie sectie 4.2.5). De elementen in de lijst zullen voorzien worden van een afbeelding en tekst. De lijst kan als een potentiële muziekapplicatie gezien worden waarbij de afbeelding en tekst naar liedjes verwijzen. Het aantal elementen in de lijst - 850 - is een schatting van het maximum aantal liedjes dat ooit is opgenomen door een artiest [146]. Het kan dus als bovengrens voor dit soort applicaties worden beschouwd. De implementatie van deze applicatie zal een indicatie geven hoe snel, zonder al te veel voorkennis van het raamwerk, één eenvoudige applicatie opgeleverd kan worden.

De werkuren van de loginapplicatie bleken niet onderheven aan de vijf zonet opgenoemde redenen:

1. Bij elke implementatie werd met dezelfde achtergrondkennis gestart. De implementatie van de loginapplicatie is triviaal en eenduidig. Er geldt dus voor alle raamwerken dat de vertrouwdheid met de functionaliteit reeds hoog was.
2. Eerst werd de implementatie van de POC gemaakt voordat aan de loginapplicatie werd begonnen. Hierdoor was de algemene ervaring met HTML5-raamwerken reeds groot.
3. Er werd geen code gekopieerd.
4. Er waren geen problemen met de *backend*.
5. Alle functionaliteit van de loginapplicatie kon met alle vier raamwerken worden gebouwd.

Om al deze redenen werd beslist de score voor productiviteit te bepalen door enkel de uren van de loginapplicatie ($t_{r,login}$) te beschouwen. Ook in de vergelijking van Burris werd voor een loginapplicatie gekozen [15]. Deze vergelijkt enkel Sencha Touch en jQuery Mobile. De formule voor productiviteit is dan:

$$\text{Productiviteit}_r = t_{r,login} \tag{4.2}$$

voor een raamwerk r.

De uitkomsten van dit criterium zullen gestaafd worden door het aantal lijnen code te presenteren die nodig waren voor zowel de POC als de loginapplicatie te bouwen. Ook zullen de factoren die de leercurve bepalen, worden bekeken. Dit zijn ten eerste de tools die de programmeur kan gebruiken om eenvoudiger te ontwikkelen. Vervolgens zal de kwaliteit en kwantiteit van de documentatie van elk raamwerk worden bekeken. De mogelijkheden voor debuggen bepalen ook de leercurve en zullen worden onderzocht. Tot slot zal gekeken worden naar de aanwezige literatuur van het raamwerk en waar ontwikkelaars met vragen terecht kunnen.

4.2.3 Gebruik

Dit criterium moet weergeven welke functionaliteit of plug-ins het raamwerk kan bieden. Hoe meer functionaliteiten het raamwerken te bieden heeft, hoe minder de programmeur zelf moet schrijven en hoe bruikbaarder het raamwerk wordt. Ook de ISO 25010-standaard probeert het gebruik met de categorie functionele geschiktheid te testen.

Uit de *mockup* schermen en de bijhorende functionele vereisten werden 13 uitdagingen met in totaal 38 deeluitdagingen geëxtraheerd. Alle functionaliteit die potentieel door een raamwerk kan worden geleverd en in de POC wordt gebruikt, zit in een uitdaging vervat. Echter, een voorbeeld van functionaliteit van de POC die niet in een uitdaging zit, is de omzetting van *identifiers* naar een tekstuele vorm. Dit is geen interessante functionaliteit omdat het eigen is aan de POC zelf. Alle uitdagingen en deeluitdagingen zijn in tabel 4.1 te vinden.

Uitdagingen

U1: Anatomie van pagina
- U1.1 Toon kop-/voettekst en onderkoptekst met titels en knoppen
- U1.2 Tabbar met functionaliteit
- U1.3 Aanpasbaarheid van kleur van knoppen

U2: Toestelspecifieke lay-out
- U2.1 Herkennen van smartphone en tablet
- U2.2 Toon het linker menu op een tablet
- U2.3 Schakel menu in bij smartphone

U3: Laadscherm en dialoogvenster
- U3.1 Toon laadscherm
- U3.2 Toon dialoogvenster

U4: Formulieren
- U4.1: Maak formulieren met *placeholders* zonder labels
- U4.2: Gebruik tekst/nummer/e-mail als formuliertypes
- U4.3: Optieveld
- U4.4: Aangepaste *datepicker* met bereik van data
- U4.5: Aangepaste *datepicker* met enkel maand- en jaarveld
- U4.6: Schakelaar
- U4.7: Wissen van een formulier

U5: Automatisch invullen van formulier
- U5.1 Vul formulierelementen met object
- U5.2 Maak elementen van het formulier *read-only*

U6: Auto-aanvullen
- U6.1 Zoek suggesties gebaseerd op gebruikersinvoer
- U6.2 Toon suggesties in klikbaar dropdownmenu

U7: Toevoegen en verwerken van afbeelding
- U7.1 Kies een afbeelding
- U7.2 Converteer de afbeelding naar Base64
- U7.3 Voorbeeld van de afbeelding

U8: Formuliervalidatie
- U8.1 Validatieregels voor verplichte nummer- en e-mailvelden
- U8.2 Validatieregels voor eigen condities
- U8.3 Foutboodschappen ophalen van ongeldige velden
- U8.4 Rode randen plaatsen rond ongeldige velden

U9: Handtekening
- U9.1 Teken een handtekening met vinger of pen

U10: AJAX: tekst, JSON en XML
- U10.1 Haal tekst zonder opmaak op
- U10.2 Haal JSON-data op en parse
- U10.3 Zend JSON-data
- U10.4 Haal XML-data op en parse

U11: Lijsten

U11.1 Laad data in lijst en stijl de elementen met een sjabloon
U11.2 Klikbare lijstelementen met actie of link naar het record van het element
U11.3 Sorteren van data
U12: Toon PDF
U12.1 Vraag een PDF-bestand op met POST-parameters
U12.2 Toon het PDF-bestand
U13: Offline
U13.1: Bewaar data
U13.2: Maak de applicatie offline beschikbaar

Tabel 4.1: 13 uitdagingen onderverdeeld in 38 deeluitdagingen voor gebruik.

De wijze waarop het raamwerk de uitdaging aangaat zal de score bepalen. Er onderscheiden zich drie gevallen. De hoogste score (2) wordt toegekend wanneer de functionaliteit aangeboden wordt door het raamwerk. Een lagere score (1) betekent dat een plug-in moet worden gezocht. Omdat de raamwerken bouwen op HTML5, zal een kenmerk van HTML5 ook als plug-in beschouwd worden. Voor een oplijsting van de HTML5-kenmerken wordt naar sectie 2.5 verwezen. Wanneer de implementatie zelf moet worden geschreven of een hack noodzakelijk is, zal de laagste score (0) worden toegekend. Ook is het mogelijk dat de uitdaging helemaal niet wordt geïmplementeerd. Dit is mogelijk wanneer het raamwerk de functionaliteit niet ondersteund, geen plug-in werd gevonden en niet aan een eigen implementatie wordt begonnen. Dit zal leiden tot een 0 score. Wanneer CSS-code wordt gebruikt om de uitdaging te implementeren, zal de laagste score worden toegekend. Het gebruik van CSS3 wordt echter als kenmerk van HTML5 gezien en vervolgens met 1 gequoteerd. Tabel 4.2 toont de mogelijke scores $U_{r,i}$ van raamwerk r en voor uitdaging i.

De potentiële score van een uitdaging is discreet en ligt tussen 0 en 2. Er zijn dus slechts 3 scores waaruit gekozen kan worden om de implementatie te beoordelen. De verklaringen bij de scores omvatten alle gevallen op een eenduidige manier. Een alternatief bestaat uit 4 scores waarbij HTML5-kenmerken een lagere score krijgen ten opzichte van plug-ins. Omdat de raamwerken afhankelijk zijn van HTML5 werd hiervoor niet gekozen.

De formule voor gebruik is de volgende:

$$\text{Gebruik}_r = \sum_{i=1}^{38} (U_{r,i}) \tag{4.3}$$

Score	**Verklaring**
$U_{r,i} = 2$	Ondersteund door het raamwerk
$U_{r,i} = 1$	Een plug-in of kenmerk van HTML5 is nodig
$U_{r,i} = 0$	Eigen implementatie of hack of niet geïmplementeerd

Tabel 4.2: Beoordeling van uitdagingen voor gebruik.

voor een raamwerk r en een deeluitdaging i. Omdat er 38 deeluitdagingen zijn, kan een raamwerk voor dit criterium maximaal 76 behalen.

4.2.4 Ondersteuning

Dit criterium moet weergeven hoe goed het raamwerk verschillende toestellen en verschillende besturingssystemen ondersteund. Het is belangrijk dat een zo breed mogelijk publiek met éénzelfde applicatie kan worden bereikt. Net zoals webapplicaties willen *cross-platform* tools zo veel mogelijk platformen bereiken. Dit werd door meer dan 75% van de ondervraagde ontwikkelaars in het Vision Mobile rapport aangehaald als hoofdreden om *cross-platform* tools te gebruiken [79]. Ook de ISO 25010-standaard beschrijft de overdraagbaarheid naar verschillende platformen.

Enkel de standaard browser van het besturingssysteem zal beschouwd worden. Voor Android-toestellen is dit de Android browser of Chrome. Vanaf Android 4.0 wordt Chrome als standaard browser beschouwd [140]. Voor iOS is Mobile Safari de standaard browser.

Een context wordt gedefinieerd als één bepaalde configuratie van toestel, besturingssysteem en browser. In elke context zal de functionaliteit van de POC op ondersteuning worden getest. Uitdagingen die gebruikt zijn om het gebruikscriterium te testen, kunnen hier worden hergebruikt. Aangezien sommige uitdagingen triviaal gelden voor elk apparaat zal er slechts een subset van deze uitdagingen getest worden. De overgebleven uitdagingen zijn:

- U2: Toestelspecifieke lay-out
- U4: Formulieren
- U6: Auto-aanvullen
- U7: Toevoegen en verwerken van afbeelding
- U8: Formuliervalidatie
- U9: Handtekening
- U12: Toon PDF
- U13: Offline

Voor dit criterium worden alle deeluitdagingen verwaarloosd behalve bij U4: Formulieren en U13: Offline. Een uitdaging zal enkel slagen als alle deeluitdagingen ondersteund worden. Zo kan bijvoorbeeld op een apparaat getest worden of auto-aanvullen werkt. Uitdaging U6: Auto-aanvullen bevat als deeluitdagingen het ophalen van suggesties en het tonen van een dropdownmenu. De werking van de uitdaging is een combinatie van beide en zal dus enkel slagen als beide worden ondersteund. De deeluitdagingen van U4: Formulieren en U13: Offline kunnen wel op ondersteuning worden getest. De twee deeluitdagingen van *datepicker* die bij U4: Formulieren

Apparaat	Soort	Lancering	BS	Browser
HTCDesireZ	Smartphone	September 2010	Android 2.3.3	Android browser
GalaxyTab	Tablet	September 2010	Android 2.3.6	Android browser
GalaxyS	Smartphone	Maart 2010	Android 4.1.2	Chrome 26.0.1410.58
Nexus 7	Tablet	Juni 2012	Android 4.2.2	Chrome 18.0.1025.469
iPad1 WiFi	Tablet	April 2010	iOS 5.1.1	Mobile Safari
iPad3 4G WiFi	Tablet	Maart 2012	iOS 6.1.3	Mobile Safari
iPhone 3GS	Smartphone	Juni 2009	iOS 6.0.1	Mobile Safari
iPhone 4S	Smartphone	Oktober 2011	iOS 6.1.3	Mobile Safari

Tabel 4.3: Acht contexten: apparaten met hun soort, lancering, besturingssysteem (BS) en browser.

horen, zullen echter worden samengenomen zodat enkel een *datepicker* op zich en niet een aanpasbare *datepicker* op ondersteuning wordt gecontroleerd.

Het is belangrijk dat het raamwerk en niet een eigen implementatie op ondersteuning wordt getest. Wanneer een uitdaging in het vorige criterium een 0 behaalde, wil dit zeggen dat het raamwerk de uitdaging al niet ondersteunde. In dit geval moet de uitdaging niet worden gecontroleerd. Hierdoor is het aantal uitdagingen of deeluitdagingen die getest worden afhankelijk van het raamwerk.

Voor een score van een uitdaging of deeluitdaging geldt dat $U_{r,c,i} = 1$ als de ondersteuning van die (deel)uitdaging slaagt en $U_{r,c,i} = 0$ als deze niet slaagt. De Cross Platform Capabilities zoals beschreven in op de Codefessions blogpost [102] geven een score aan raamwerken op een gelijkaardige manier. In totaal zullen acht contexten worden gebruikt. Deze worden in tabel 4.3 weergegeven.

De keuze van de acht contexten waarop ondersteuning wordt getest, is voornamelijk bepaald door de beschikbaarheid van de apparaten op het Departement Computerwetenschappen van de KU Leuven. Er werd een evenwichtige keuze gemaakt tussen besturingssysteem, browser en type apparaat. Er werd gekozen voor vier Android- en vier iOS-apparaten. Bij de vier Android-apparaten zijn er twee met een Android browser en twee met Chrome browser. Ook werd er gekozen voor vier smartphones en vier tablets.

De som van de scores van de verschillende contexten bepaalt de score van het ondersteuningscriterium:

$$\text{Ondersteuning}_r = \sum_{c=1}^{8} \left(\sum_{i=1}^{N_r} U_{r,c,i} \right) \tag{4.4}$$

voor een raamwerk r, een context c, N_r het maximum aantal geïmplementeerde deeluitdagingen voor een raamwerk r en een uitdaging i.

Indien het raamwerk een implementatie bevat voor alle uitdagingen en deeluitdagingen kan er per context maximaal 13 gescoord worden. Indien de acht contexten alle uitdagingen en deeluitdagingen correct weergeven zal de maximale score van 104 behaald worden.

4.2.5 Performantie

Performantie wordt opgesplitst in twee verschillende factoren: downloadtijd en gebruikerservaring. Het is noodzakelijk dat een applicatie zowel snel wordt gedownload als vlot is in gebruikerservaring. De ISO 25010-standaard gebruikt voor de eerstgenoemde de categorie prestatie-efficiëntie om de snelheid en gebruikte middelen te beoordelen.

De downloadtijd meet hoelang het duurt om de webapplicatie te downloaden. De gebruikerservaring bepaalt hoe vlot het gaat om door een lange lijst van 850 elementen te scrollen. Zoals in sectie 4.2.2 werd verteld, wordt de loginapplicatie als stresstest gebruikt om de performantie te testen. De volledige POC kon niet in ieder raamwerk worden geïmplementeerd omdat de raamwerken niet alle functionaliteit konden aanbieden. Dit is in tegenstelling tot de loginapplicatie die wel in de vier raamwerken kan worden geïmplementeerd. Hierdoor zal de POC niet worden gebruikt in dit criterium. De downloadtijden en gebruikerservaring zullen op acht verschillende apparaten worden opgemeten. Dit zijn dezelfde apparaten als bij het ondersteuningscriterium (zie tabel 4.3).

Gemiddelde downloadtijd

Bij de downloadtijden onderscheiden zich twee gevallen die samen de totale downloadtijd bepalen. Eerst zal de downloadtijd van de loginapplicatie worden bekeken ($\widehat{l}_{r,c,login}$). Vervolgens zal de tijd worden opgemeten om de loginapplicatie uit het cachegeheugen te downloaden ($\widehat{l}_{r,c,login_{cache}}$). Deze downloadtijden zullen voldoende keren per apparaat moeten worden uitgevoerd om een betrouwbare meting te bekomen.

Het opmeten van de downloadtijden zal met TCPdump [129] gebeuren, zoals werd voorgesteld door Thair [135]. Hiervoor wordt een laptop als hotspot ingesteld en zullen de acht apparaten op deze hotspot connecteren via WiFi. Wanneer de meting wordt gestart, zal op het apparaat naar de applicatie gesurft worden. Nadat alle bestanden zijn ingeladen wordt de meting beëindigd. De uitvoer van TCPdump is een PCAP-bestand die de HTTP-trafiek bevat. Deze zal via PCAP Web Performance Analyzer [69] worden omgezet naar een HAR-file, waarna een HTTP-waterval zal worden getoond. Hieruit kan de totale downloadtijd worden gehaald.

De gemiddelde downloadtijd voor een raamwerk wordt bepaald door de som van de gemiddelde downloadtijden per apparaat:

$$\text{Gemiddelde downloadtijd}_r = \frac{\sum_{c=1}^{8} \left(\widehat{l}_{r,c,login} + \widehat{l}_{r,c,login_{cache}} \right)}{8} \tag{4.5}$$

voor een raamwerk r en context c. Dit is een perceptie en komt niet overeen met de werkelijk gemiddelde downloadtijd van de applicatie, want het downloaden kan ofwel van de server ofwel uit het cachegeheugen. De downloadtijd verschilt sterk bij deze twee manieren. Hierdoor is het dus nodig om beide in rekening te brengen.

Gebruikerservaring

Eerst werd voorgesteld om de rendertijd ($\widehat{l}_{r,c,lijst}$) te bepalen in plaats van de gebruikerservaring. De rendertijd is de tijd die het raamwerk nodig heeft om de GI-elementen te renderen. Hiervoor wordt een lijst van 850 elementen gebruikt die getoond wordt na aanmelden op de loginapplicatie. De tijd die het raamwerk nodig heeft om de lijst de renderen kan gemeten worden met JavaScript [135]. Net zoals de downloadtijden zullen deze voldoende keren per apparaat moeten worden uitgevoerd.

De gemiddelde rendertijd is:

$$\text{Gemiddelde rendertijd}_r = \frac{\sum_{c=1}^{8} \left(\widehat{l}_{r,c,lijst}\right)}{8} \tag{4.6}$$

voor een raamwerk r en context c.

Enkel bij jQuery Mobile en Kendo UI kon de rendertijd via JavaScript worden opgemeten. Bij de twee andere raamwerken werden er geen gebeurtenissen gevonden om correct de tijd op te meten. Doordat er maar data van twee raamwerken voor handen was, werd de rendertijd vervangen door de gebruikerservaring van een lijst. Deze bestaat eruit de vlotheid van het scrollen door de lijst van 850 lijstelementen voor de vier raamwerken op de acht apparaten te vergelijken. Per apparaat wordt een score ($\text{ervaring}_{r,c}$) van 1, 2, 3 of 4 uitgedeeld aan de raamwerken. Hierbij is 4 de beste score wat overeenkomt met het vlotste scrollen door de lijst relatief ten opzichte van de drie andere raamwerken. Deze test werd uitgevoerd door twee personen.

Om de score voor gebruikerservaring van een raamwerk te bepalen worden de scores voor dat raamwerk op ieder apparaat opgeteld. De formule voor gebruikerservaring voor een raamwerk r wordt:

$$\text{Gebruikerservaring}_r = \sum_{c=1}^{8} \text{ervaring}_{r,c} \tag{4.7}$$

In het bekomen eindklassement komt de hoogste totaalscore overeen met het raamwerk dat de vlotste gebruikerservaring aanbiedt op alle acht apparaten.

Totaal

De performantie wordt bepaald door de gemiddelde downloadtijd en de gebruikerservaring. De opzet van de formule is om een raamwerk dat slecht scoort op de gemiddelde downloadtijd, maar sterk scoort op gebruikerservaring, een middelmatige score te geven. Beide factoren hebben een andere eenheid waardoor deze niet kunnen worden opgeteld. Daardoor werd gekozen om gebruikerservaring als gewicht voor gemiddelde downloadtijd te zien. Voor de gemiddelde downloadtijd geldt hoe kleiner, hoe beter. Dit geldt niet voor de gebruikerservaring waardoor de geïnverteerd gebruikerservaring als gewicht wordt gebruikt. De formule voor de score voor de performantie wordt:

$$\text{Performantie}_r = \frac{\text{Gemiddelde downloadtijd}_r}{\text{Gebruikerservaring}_r} \tag{4.8}$$

van een raamwerk r.

De maximale responstijd is wanneer de loginapplicatie niet uit cache wordt geladen:

$$\text{Maximale responstijd}_r = \frac{\sum_{c=1}^{8} \left(\widehat{l}_{r,c,login} + \widehat{l}_{r,c,lijst} \right)}{8} \tag{4.9}$$

Deze maximale responstijd kan gecategoriseerd worden met limieten uitgedrukt in seconden zoals opgelegd door Jakob Nielsen [85]:

- Maximale responstijd$_r$ < 0.1 s: de gebruiker heeft het gevoel dat het systeem direct reageert.
- Maximale responstijd$_r$ < 1 s: de gedachtengang van de gebruiker zal niet worden onderbroken, maar hij zal toch een vertraging waarnemen.
- Maximale responstijd$_r$ < 10 s: de limiet om de aandacht van de gebruiker te behouden.

De maximale responstijd kan echter niet worden berekend omdat er geen rendertijden bij Sencha Touch en Lungo kunnen worden opgemeten. Indien de rendertijd uit de formule wordt weggelaten, blijft de maximale responstijd een schatting voor de limiet van Nielsen.

Om de scores van het performantiecriterium te duiden zal de gemiddelde downloadtijd van de POC en de loginapplicatie met elkaar worden vergeleken. Ook zullen de resultaten gecontroleerd worden met Google Page Speed [82]. Deze tool kan de code van een webpagina analyseren en de performantie testen specifiek voor mobiele apparaten. Het resultaat is een score op 100 en een lijst van werkpunten om de performantie van de applicatie te verbeteren. Een hoge score duidt op weinig plaats voor verbetering, een lagere score duidt op meer plaats voor verbetering. Google Page Speed meet niet de tijd om een pagina te laden.

4.3 Overzicht

Om de scores van de vijf criteria samen te vatten zal een spinnenweb worden gebruikt. Hierdoor moet elke score op dezelfde schaal worden gebracht om duidelijk de verschillen te kunnen waarnemen. De Matlab-extensie om spinnenwebben te genereren, vereist dit ook [71]. Er werd gekozen om alle scores te relativeren. Hiervoor moet elke score van een criterium gedeeld worden door het maximaal behaalde resultaat van dat criterium. Alle scores zullen vervolgens tussen 0 en 1 liggen. Deze methode zal ervoor zorgen dat het raamwerk met de beste score een 1 behaalt.

Om verwarring te voorkomen, moeten ook de scores voor het productiviteitscriterium en performantiecriterium geïnverteerd worden. Dit komt omdat voor deze criteria geldt: hoe lager de score, hoe beter het raamwerk.

De formules om de relatieve scores te bereken worden hieronder weergegeven. De relatieve scores zullen gebruikt worden om het spinnenweb op te stellen.

$$\text{Populariteit}_r^{\pentagon} = \frac{\text{Populariteit}_r}{\max\limits_m\{\text{Populariteit}_m\}} \tag{4.10}$$

$$\text{Productiviteit}_r^{\pentagon} = \frac{\text{Productiviteit}_r^{-1}}{\max\limits_m\{\text{Productiviteit}_m^{-1}\}} \tag{4.11}$$

$$\text{Gebruik}_r^{\pentagon} = \frac{\text{Gebruik}_r}{\max\limits_m\{\text{Gebruik}_m\}} \tag{4.12}$$

$$\text{Ondersteuning}_r^{\pentagon} = \frac{\text{Ondersteuning}_r}{\max\limits_m\{\text{Ondersteuning}_m\}} \tag{4.13}$$

$$\text{Performantie}_r^{\pentagon} = \frac{\text{Performantie}_r^{-1}}{\max\limits_m\{\text{Performantie}_m^{-1}\}} \tag{4.14}$$

De totale score voor een raamwerk r wordt dan:

$$\begin{aligned}\text{Score}_r = \frac{1}{5}\Big(&\text{Populariteit}_r^{\pentagon} + \text{Productiviteit}_r^{\pentagon} + \text{Gebruik}_r^{\pentagon} \\ &+ \text{Ondersteuning}_r^{\pentagon} + \text{Performantie}_r^{\pentagon}\Big)\end{aligned} \tag{4.15}$$

Hoofdstuk

5 Evaluatie

In dit hoofdstuk wordt de vergelijking uitgevoerd op basis van de vijf actieve vergelijkingscriteria uit hoofdstuk 4, namelijk populariteit (5.1), productiviteit (5.2), gebruik (5.3), ondersteuning (5.4) en performantie (5.5). Daarna zullen deze vijf vergelijkingscriteria in sectie 5.6 worden samengevat in een spinnenweb.

5.1 Populariteit

De populariteit van de vier raamwerken op 8 mei 2013 wordt weergegeven in tabel 5.1. Voor de score van populariteit op basis van sociale netwerken wordt naar formule 4.1 verwezen.

Kendo UI neemt de eerste plaats voor zich dankzij het zeer groot aantal vind-ik-leuks op Facebook. jQuery Mobile en Sencha Touch slepen respectievelijk een tweede en derde plaats in de wacht, ondanks het feit dat ze in de literatuur de meest aangehaalde raamwerken zijn [19, 30, 41, 87]. Als laatste eindigt Lungo met een opmerkelijk lage populariteit op Stack Overflow en Facebook. Bij het kijken naar de totaalscore kunnen twee groepen worden waargenomen, enerzijds de groep bestaande uit Kendo UI en jQuery Mobile en anderzijds de groep bestaande uit Sencha Touch en Lungo.

Op Twitter heeft jQuery Mobile de meeste volgers, gevolgd door Kendo UI. Op de voorlaatste plaats komt Lungo, maar als het aantal *tweets* wordt uitgezet ten

Populariteit	**ST**	**KUI**	**jQM**	**L**
Twitter volgers	1197	9532	12952	1665
GitHub sterren			8184	1361
GitHub *forkers*			1886	316
Stack Overflow vragen	5621	2257	20630	19
Facebook vind-ik-leuks	748	54224	2760	28
Totaal	7566	66013	46412	3389

Tabel 5.1: Overzicht van populariteit op 8 mei 2013.

opzichte van het aantal volgers, kan er gesteld worden dat Lungo het meest actief is. jQuery Mobile en Kendo UI hebben een vergelijkbare activiteit bij het sturen van *tweets*. Sencha Touch heeft het minst aantal volgers en het aantal verstuurde *tweets* is slechts één. Deze *tweet* verwijst naar de Twitter-account van Sencha zelf. Sencha heeft een kleine 3.000 *tweets* en een kleine 20.000 volgers.

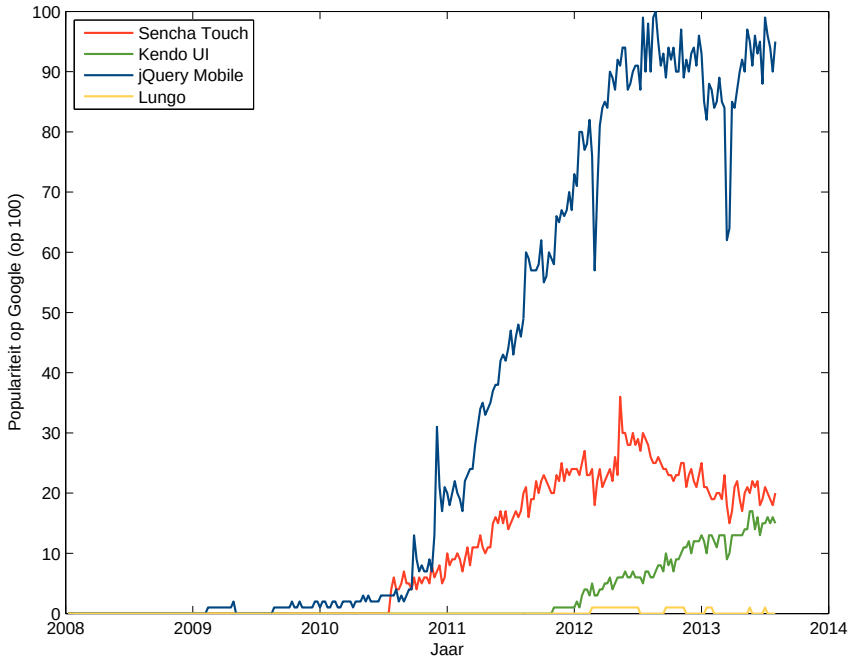
In tegenstelling tot jQuery Mobile en Lungo bevinden Kendo UI en Sencha Touch zich niet op GitHub. Zelfs indien GitHub wordt weggelaten, blijft de rangschikking ongewijzigd. De opvatting van Capgemini dat enkel *open-source* raamwerken populair zijn, wordt ontkracht door deze cijfers.

Kendo UI verwijst op zijn website voor ondersteuning rechtstreeks naar de fora op Stack Overflow. Toch blijft de populariteit van Kendo UI op Stack Overflow lager dan die van jQuery Mobile. Sencha Touch behaalt de voorlaatste plaats, maar opvallend is Lungo die slechts een dertigtal vragen op Stack Overflow heeft en dus op de laatste plaats eindigt.

Kendo UI en jQuery Mobile hebben beide een fanpagina op Facebook opgericht in respectievelijk november 2011 en augustus 2010. De fanpagina van Kendo UI heeft dus in een kortere tijd veel meer vind-ik-leuks opgeleverd dat de eerder opgerichte fanpagina van jQuery Mobile. De verschillende producten van Kendo UI worden geaggregeerd op één fanpagina. Sencha Touch en Lungo hebben enkel een interessepagina op Facebook. Dit verklaart het grote verschil in vind-ik-leuks op Facebook. Het aantal vind-ik-leuks bepaalt de populariteit in grote mate. Indien deze vind-ik-leuks worden weggelaten, wisselen jQuery Mobile en Kendo UI van plaats. De auteurs zijn van mening dat Facebook een belangrijk sociaal netwerk is. Daarenboven kost het ook geen geld om een fanpagina aan te maken. Hierdoor kan Facebook niet uit het populariteitscriterium worden weggelaten.

Deze populariteit werd ook iedere week bijgehouden over een periode van zeven weken. Enkel Kendo UI kende in deze korte periode een opmerkelijke groei. Dit komt grotendeels door het enorm stijgend aantal vind-ik-leuks op Facebook. De andere drie raamwerken stijgen gestaag.

Als laatste wordt de populariteit aan de hand van Google Trends bekeken op figuur 5.1. Duidelijk is dat hier jQuery Mobile de grote winnaar is. Sinds 2011 maakt het raamwerk een grote opmars door het uitbrengen van de eerste stabiele versie 1.0. Tussen versie 1.2 (uitgekomen in oktober 2012) en versie 1.3 (uitgekomen in februari 2013) daalt de interesse met 40%. Dit valt te verklaren doordat de nieuwe interesse in versie 1.2 daalt, maar weer opwakkerd bij de aankondiging van versie 1.3. Een gelijkaardige trend kan ook bij Sencha Touch opgemerkt worden. Sencha Touch kende een piek in maart 2012 bij het uitbrengen van Sencha Touch 2.0. Sinds begin 2012 maakt Kendo UI een opmars en als de trend zich verder zet, zal het Sencha Touch inhalen. Dit komt overeen met de waargenomen opmars van Kendo UI tijdens de periode van 7 weken. Lungo is nauwelijks op de grafiek waarneembaar.



Figuur 5.1: Populariteit op Google Trends waargenomen van januari 2008 tot heden waarbij een resultaat van 100 overeenkomt met de grootste zoekinteresse [37].

Productiviteit	**ST**	**KUI**	**jQM**	**L**
Loginapplicatie (u)	7.3	5.0	2.1	2.9

Tabel 5.2: Overzicht van productiviteit.

5.2 Productiviteit

In deze sectie zal de productiviteit van de vier raamwerken worden onderzocht. Voor de score van productiviteit wordt naar formule 4.2 verwezen waarbij de uren gewerkt aan de loginapplicatie in rekening worden gebracht. Tabel 5.2 bevat een overzicht van de uren gespendeerd aan de loginapplicatie.

De uren voor de implementatie van de loginapplicatie tonen dat jQuery Mobile productiever is dan Lungo. Hierna volgt Kendo UI en Sencha Touch. Er zijn twee conclusies: het type raamwerk (JavaScript-gedreven of opmaakgedreven) en het al dan niet afdwingen van een ontwerppatroon bepalen de productiviteit. Raamwerken zonder ontwerppatroon (jQuery Mobile en Lungo) zijn productiever dan raamwerken die een ontwerppatroon afdwingen (Sencha Touch en Kendo UI). Indien het raamwerk een ontwerppatroon afdwingt, dan is een puur JavaScript-gedreven raamwerk (in dit geval Sencha Touch) minder productief dan een raamwerk dat zowel JavaScript- als opmaakgedreven is. De productiviteit van Kendo UI is 31.8% groter ten opzichte van Sencha Touch.

Code	ST		KUI		jQM		L	
	POC	Login	POC	Login	POC	Login	POC	Login
HTML	0	0	302	54	653	48	464	74
JavaScript	1927	177	762	51	1174	62	640	48
CSS	14	6	62	32	300	7	58	3
Totaal	1941	183	1126	137	2127	117	1162	125

Tabel 5.3: Aantal lijnen effectief geschreven code.

5.2.1 Duiding

In wat volgt zullen factoren worden besproken die de verschillen in productiviteit verklaren.

Lijnen code In tabel 5.3 staan het aantal lijnen effectief geschreven code die nodig waren om de POC en loginapplicatie te maken. Zowel HTML-, JavaScript- als CSS-code werden geteld. De conclusies voor productiviteit worden bevestigd met het aantal lijnen code: de meest productieve raamwerken vereisen het minst aantal lijnen code. Hetzelfde geldt niet als naar het aantal lijnen geschreven code bij de implementatie van de POC wordt gekeken. Kendo UI vraagt minder lijnen code in vergelijking met de andere raamwerken. Er kan worden geconcludeerd dat bij een grotere applicatie het MVVM-ontwerppatroon van Kendo UI het aantal lijnen code reduceert. Door de grotere complexiteit zal het schrijven van deze code echter meer tijd vergen in vergelijking met opmaakgedreven raamwerken. Een opmerking over de data van Sencha Touch en Lungo moet worden gemaakt: niet alle functionaliteit van de POC kon worden geïmplementeerd dus zijn de bijhorende lijnen code onvolledig.

Tools Enkel bij de ontwikkeling met Sencha Touch kon beroep worden gedaan op een grafische tool om het ontwikkelingsproces te vergemakkelijken: Sencha Architect [109]. Dit is een desktopapplicatie die het ontwikkelingsproces vergemakkelijkt met een GGI en *drag-and-drop*. Bij de ontwikkeling van de POC werd Sencha Architect versie 2.1 gebruikt. Het grootste voordeel van Sencha Architect kon bij de ontwikkeling van `Views` worden gevonden. De `Views` kunnen specifiek voor mobiele schermen worden geoptimaliseerd, dit zowel voor staande als liggende apparaten. Na enkele dagen werd er overgeschakeld op PhpStorm [48] als Integrated Development Environment (IDE). De voornaamste reden hiervoor was de beperkte bruikbaarheid van Sencha Architect voor de ontwikkeling van meer geavanceerde functionaliteit. PhpStorm werd ook gebruikt bij de ontwikkeling met andere raamwerken en biedt ondersteuning voor zowel Windows, Mac als Linux. Een voordeel bij het gebruik van deze IDE is onder andere de automatische code-aanvulling. Daarnaast toont de IDE hints voor het optimaliseren van JavaScript-code die specifiek gebruik maakt van de jQuery-bibliotheek.

Als laatste werd Yeoman [143] gebruikt bij Kendo UI, jQuery Mobile en Lungo om applicaties te bouwen. Kendo UI en Lungo konden gebruik maken van Twitter

Bower [136] om de bestanden van de GitHub-*repository* toe te voegen. Bij Sencha Touch werd Sencha Cmd gebruikt om de applicatie te bouwen.

Boilerplate code Sencha Touch biedt een tool voor het initialiseren van een nieuwe applicatie, Sencha Cmd [110]. Deze tool kan de initiële applicaties opzetten, bestanden toevoegen en de applicatie bouwen en uitrollen. Na de geautomatiseerde initialisatie van een project zijn de gemaakte folders en bestanden echter niet geheel duidelijk. Een nieuwe `Controller`, `Store`, `Model` of `View` genereren kan met of zonder Sencha Cmd. Zonder Sencha Cmd moeten het nieuwe JavaScript-bestand in de juiste folder worden ondergebracht. Sencha Touch legt een strenge structuur van folders op die ongewijzigd moet blijven opdat de applicatie zou werken. Dit leidde bij aanvang van de implementatie tot veel verwarring.

jQuery Mobile en Kendo UI beschrijven *boilerplate* code in hun documentatie [57, 130]. Ook is bij de documentatie van beide raamwerken een expliciete sectie aanwezig die de programmeur helpt om een applicatie op te zetten. Bij Lungo is dit niet het geval. Om een Lungo applicatie op te zetten, moest naar de broncode van de voorbeelden op de documentatie gekeken worden.

Documentatie De Sencha Touch documentatie is het grootst in vergelijking met de andere raamwerken. De grootte van de documentatie maakt het moeilijk om zelf gericht naar onduidelijkheden te zoeken. De zoekfunctie met auto-aanvulling is noodzakelijk om de juiste documentatie terug te vinden. De documentatie van Kendo UI is overzichtelijker en volledig. De combinatie van de secties API en Getting Started boden de voornaamste hulp. Ook worden alle kenmerken die het raamwerk aanbiedt met demo's en codevoorbeelden getoond. De zoekfunctie van de documentatie is niet optimaal en hierdoor werd deze maar weinig gebruikt. De documentatie van jQuery Mobile 1.2 bevat geen zoekfunctie en codevoorbeelden. Hierdoor moet naar de broncode worden gekeken om de code van een kenmerk te begrijpen. Een voordeel van de jQuery Mobile documentatie is dat deze zelf met jQuery Mobile is gebouwd. Een belangrijke opmerking is dat jQuery Mobile 1.3 wel een zoekfunctie en codevoorbeelden in zijn documentatie heeft opgenomen. De documentatie van Lungo is zeer beknopt en daarenboven bestaat deze van QuoJS uit slechts één pagina. Alle kenmerken van Lungo worden met codevoorbeelden verduidelijkt. Sommige voorbeelden zijn echter incorrect.

Debugging De applicatie werd lokaal uitgerold en in de Web Inspector van Chrome gedebugd. Debuggen op de apparaten kon door deze te connecteren met de computer via Universal Serial Bus (USB). Enkel Android-toestellen met een Chrome browser kunnen worden gedebugd met de Web Inspector van het apparaat. Debuggen op iOS-toestellen kan op een Mac met dezelfde Web Inspector maar in de Safari browser. Dit werkt echter niet op iPad1 WiFi.

Literatuur Zowel jQuery Mobile als Sencha Touch werden vaak in de literatuur aangehaald. Op Safari Books Online kunnen 13 boeken teruggevonden worden die uitsluitend jQuery Mobile behandelen [88]. Het boek van Dutson in de Sam Teach Yourself serie werd gebruikt om met jQuery Mobile vertrouwd te geraken [94]. Ook Pro jQuery Mobile van Broulik [14] werd gelezen maar dit is nagenoeg een kopie

van de documentatie en bevat bijgevolg niet veel extra informatie. Vier boeken over Sencha Touch kunnen op Safari Books Online worden teruggevonden. Het boek van Clark et al. werd gebruikt [18]. Kendo UI komt slechts in één boek voor [86], Lungo helemaal niet. Het boek rond Kendo UI is slechts een proefdruk en wordt pas in augustus 2013 officieel gepubliceerd. Dit werk werd niet gebruikt.

Fora Wanneer de programmeur met problemen van het raamwerk werd geconfronteerd, kan op het web naar oplossingen gezocht worden. In sectie 5.1 werden reeds het aantal vragen van het raamwerk op Stack Overflow bekeken. Hoe groter dit aantal, hoe groter de kans dat een probleem reeds is aangehaald. Bij Lungo is dit aantal minimaal en was de programmeur vaak aangewezen om zelf oplossingen te zoeken. Er werden twee vragen gesteld op Google Groups en LungoJS Community, maar hier is tot op moment van schrijven nooit antwoord op gekomen. Voor Sencha Touch en Kendo UI geldt dat ook het forum voor professionele ondersteuning hulpvaardig was. Het plaatsen van vragen bij Sencha Touch is onmogelijk zonder betaalde ondersteuning, Kendo UI laat het stellen van maximaal vijf vragen toe zonder de aankoop van een licentie. De meest voorkomende problemen zijn op de fora al aangehaald en gedetailleerd besproken. De antwoorden kunnen zonder licentie bekeken worden.

5.3 Gebruik

Het gebruik van de vier raamwerken wordt samengevat voor de 13 uitdagingen in tabel 5.4. Voor de score van gebruik wordt naar formule 4.3 verwezen waarbij de scores van alle uitdagingen worden opgeteld. Voor Sencha Touch werd gebruik gemaakt van versie 2.1.1, voor Kendo UI van versie 2013 Q1, voor jQuery Mobile van versie 1.3.0 en voor Lungo van versie 2.1. Per sectie zal iedere uitdaging per raamwerk worden besproken.

Het raamwerk dat het beste scoort bij gebruik is Kendo UI, kort gevolgd door Sencha Touch. Dit kan grotendeels door de aanwezigheid van een ontwerppatroon, respectievelijk MVVM en MVC, worden verklaard. Het ontbreken van een ontwerppatroon bij de andere twee raamwerken resulteert in een omslachtige aanpak, waardoor punten worden verloren. Opmerkelijk is de perfecte score van Kendo UI voor formulieren en de mindere ondersteuning voor offline in vergelijking met Sencha Touch. jQuery Mobile behaalt de helft met wat overschot. Op het vlak om data automatisch in te laden in velden of lijsten, scoort het nul door het ontbreken van een ontwerppatroon. Als laatste komt Lungo dat een onvoldoende behaalt. Dezelfde pijnpunten van jQuery Mobile zijn ook geldig voor Lungo. Daarenboven is er bij Lungo een totaal gebrek aan formuliervalidatie en aan de meer geavanceerdere formulierelementen. Voor deze laatste moet er een plug-in worden gebruikt of zal de functionaliteit zelf moeten worden geïmplementeerd.

Een algemene trend bij ieder raamwerk is de volle ondersteuning van AJAX-verzoeken, behalve dan bij één geval voor Lungo. Ook laadschermen en dialoogvensters zijn bij ieder raamwerk volledig aanwezig.

Uitdaging	**Max**	**ST**	**KUI**	**jQM**	**L**
U1: Anatomie van pagina	6	6	2	4	2
U2: Toestelspecifieke lay-out	6	4	6	3	3
U3: Laadscherm en dialoogvenster	4	4	4	4	4
U4: Formulieren	14	8	14	9	6
U5: Automatisch invullen van formulier	4	4	4	0	0
U6: Auto-aanvullen	4	1	4	4	2
U7: Toevoegen en verwerken van afbeelding	6	4	4	4	4
U8: Formuliervalidatie	8	6	6	3	0
U9: Handtekening	2	1	1	1	0
U10: AJAX: tekst, JSON en XML	8	8	8	8	6
U11: Lijsten	6	6	4	0	2
U12: Toon PDF	4	2	1	1	1
U13: Offline	4	4	2	2	3
Totaal	76	58	60	43	33

Tabel 5.4: Overzicht van gebruik.

Uitdaging	**Max**	**ST**	**KUI**	**jQM**	**L**
U1.1 Toon kop-/voettekst en onderkoptekst met titels en knoppen	2	2	0	2	0
U1.2 Tabbar met functionaliteit	2	2	2	2	2
U1.3 Aanpasbaarheid van kleur van knoppen	2	2	0	0	0
Totaal	6	6	2	4	2

Tabel 5.5: Gebruik van U1: Anatomie van pagina.

5.3.1 U1: Anatomie van pagina

In tabel 5.5 worden de resultaten getoond van de drie deeluitdagingen. Hieronder wordt per raamwerk verklaard waarom dat resultaat werd behaald.

Sencha Touch Het opbouwen van een `View` in Sencha Touch gebeurt hiërarchisch met containers. Een component die aan een container kan worden toegevoegd is de `Toolbar`. Een `Toolbar` kan zowel bovenaan als onderaan een container worden vastgezet. Dit kan dan dienst doen als kop- en voettekst. Een cascade van kopteksten is door de hiërarchische opbouw van containers mogelijk en laat zo onderkopteksten toe. Een `Toolbar` kan voorzien worden van een titel door de `title`-eigenschap in te vullen. Knoppen aan een `Toolbar` toevoegen kan door een lijst van `buttons` aan de `items`-eigenschap toe te voegen. Het toevoegen van een tabbar verloopt analoog. De `items`-eigenschap bevat dan een lijst van `Views` met een titel waarbij de `View` zichtbaar wordt als op de titel wordt gedrukt. Knoppen van kleur veranderen kan door de `style`-eigenschap van een knop te zetten.

Kendo UI Het skelet van een `View` kan binnen een HTML-tag worden geschreven. Kop- en/of voetteksten toevoegen kan door in het skelet de bijhorende HTML-tags te gebruiken. `Header`-tags kunnen niet genest worden, dus zijn onderkopteksten niet mogelijk. Een tabbar moet met een `Buttongroup` worden gemaakt, dit is een lijst van knoppen. Na deze lijst moeten de bijhorende `Views` worden geschreven. De `Buttongroup` voorziet een gebeurtenis bij het selecteren van een knop. Hier moet de overeenkomstige `View` zichtbaar worden gemaakt met behulp van CSS-manipulaties. De kleur van een knop wijzigen kan door een nieuwe CSS-klasse aan de knop toe te kennen.

jQuery Mobile Het toevoegen van een kop- en voettekst gebeurt door gebruik te maken van HTML5-tags. Wel moest dezelfde code op ieder scherm worden herhaald. Dit kan worden vermeden door gebruik te maken van eenzelfde data-attribuut. Daarnaast werd de voettekst gefixeerd aan de onderkant van het scherm en de bijhorende logo's links en rechts uitgelijnd. Voor dit laatste werd gebruik gemaakt van de zogenaamde *grid* die jQuery Mobile 1.3 zelf aanbiedt. De voettekst wordt niet getoond op een smartphone, wat wordt bekomen door gebruik te maken van de CSS3 Media Queries.

De onderkoptekst werd eerst geïmplementeerd met een lijst met enkel één lijstdeler. Dit schoof echter de inhoud van de pagina niet mee naar onder. De uiteindelijke oplossing kwam vanuit de documentatie om dit met behulp van de CSS-klasse `ui-bar` te implementeren [54]. Deze extra titel wordt ook gebruikt om naar het menu voor de smartphone te gaan (zie 5.3.2).

Standaard is er een tabbalk aanwezig in jQuery Mobile, maar de POC impliceerde een tabbalk die niet de volledige breedte innam. Daarom werd gekozen voor een formulierveld met twee opties. ThemeRoller [61] werd gebruikt om de knoppen groen te maken. Door daarna de knop te annoteren met de bekomen CSS-klasse, wordt het betreffende thema geactiveerd. Om de knop blauw te maken was er geen nood aan een aanpassing. Blauw is al één van de standaard thema's en kon dus direct worden gebruikt.

Lungo Het tonen van kop- en voettekst gebeurt door de verschillende schermen van de applicatie te omvatten met HTML5-tags. Dit is in tegenstelling tot jQuery Mobile waarbij wordt gebruik gemaakt van data-attributen. In de voettekst kunnen door CSS-regels de twee logo's links en rechts uitgelijnd worden. De onderkoptekst werd met een omweg bekomen door een lijst te maken met slechts één lijstitem. Het maken van een tabbar is standaard aanwezig in Lungo. De tabbar wordt getoond over de volledige breedte en komt onmiddellijk onder de koptekst en dus boven de onderkoptekst. Dit in tegenstelling tot het gevraagde in de POC waar de tabbar onder de onderkoptekst diende te komen. Het veranderen van de kleur van knoppen gebeurt in CSS waarbij de achtergrondkleur van de knop kan worden aangepast.

5.3.2 U2: Toestelspecifieke lay-out

In tabel 5.6 worden de resultaten getoond van de drie deeluitdagingen. Hieronder wordt per raamwerk verklaard waarom dat resultaat werd behaald.

Uitdaging	Max	ST	KUI	jQM	L
U2.1 Herkennen van smartphone en tablet	2	2	2	1	1
U2.2 Toon het linker menu op een tablet	2	2	2	1	1
U2.3 Schakel menu in bij smartphone	2	0	2	1	1
Totaal	6	4	6	3	3

Tabel 5.6: Gebruik van U2: Toestelspecifieke lay-out.

Sencha Touch Sencha Touch ondersteunt het herkennen van de context waarin de applicatie wordt gebruikt. Het kan zowel besturingssysteem, browser als ondersteunde (HTML5-)kenmerken opvragen en herkennen. De creatie van de tablet lay-out steunt op de `HBox`-lay-out die componenten horizontaal naast elkaar plaatst. De creatie van de smartphone lay-out maakt het menu in de linkse component van de lay-out onzichtbaar. Sencha Touch ondersteunt geen klikbare kopteksten om naar het smartphonemenu terug te keren. Hiervoor werd een extra knop in de koptekst toegevoegd.

Kendo UI Ook Kendo UI kan de context waarin de applicatie wordt uitgevoerd, opvragen. De lay-out van de tablet is mogelijk met een `Splitview`. Deze `View` is specifiek voor tablets en kan het scherm horizontaal of verticaal opdelen. De tablet lay-out wordt standaard gebruikt. Om de smartphone lay-out te verkrijgen moet de `Splitview` door standaard `Views` worden vervangen met DOM-manipulaties.

jQuery Mobile Het raamwerk biedt zelf geen functies aan om te herkennen of het toestel een smartphone of tablet is. In jQuery Mobile is er ook geen functionaliteit aanwezig om een menu in te schakelen op tablets. Drie plug-ins [96, 142, 31] werden gevonden via een blogpost [21] en getest. Elk hadden ze hun tekorten. Zo was de eerste destructief ten opzichte van het raamwerk. Dit betekent dat de bestanden van het raamwerk zelf werden aangepast, wat het moeilijker maakt als er moet worden geüpdatet naar een nieuwe versie. De tweede plug-in werkte enkel tot versie 1.0.1. De laatste plug-in paste zich niet aan aan de veranderende afmetingen van de browser. Uiteindelijk werden CSS3 Media Queries gebruikt [40, 53]. Bij CSS3 Media Queries moet zelf een breekpunt (uitgedrukt in pixels) worden opgegeven wanneer dient geschakeld te worden tussen smartphone lay-out of tablet lay-out. Het smartphonemenu is altijd geactiveerd en kan ook worden gebruikt als de applicatie op een tablet wordt getoond.

Lungo Het raamwerk biedt enkel een functie aan om te weten of het huidige apparaat mobiel is. QuoJS biedt daarenboven de mogelijkheid om de breedte en hoogte van het scherm terug te geven. Dit betekent dat de ontwikkelaar nog altijd zelf instaat voor de bepaling of het een smartphone of tablet is. Als oplossing werd gebruik gemaakt van CSS3 Media Queries, met dezelfde aanpak als jQuery Mobile. Het smartphonemenu is altijd geactiveerd en kan ook worden gebruikt als de applicatie op een tablet wordt getoond.

Uitdaging	Max	ST	KUI	jQM	L
U3.1 Toon laadscherm	2	2	2	2	2
U3.2 Toon dialoogvenster	2	2	2	2	2
Totaal	4	4	4	4	4

Tabel 5.7: Gebruik van U3: Laadscherm en dialoogvenster.

5.3.3 U3: Laadscherm en dialoogvenster

In tabel 5.7 worden de resultaten getoond van de twee deeluitdagingen. Hieronder wordt per raamwerk verklaard waarom dat resultaat werd behaald.

Sencha Touch Een laadscherm tonen kan door een masker op de huidige `View` te plaatsen. Dit masker kan een bericht bevatten dat de laadtekst voorstelt. Sencha Touch biedt drie standaarden van dialoogvensters aan: `alert`, `prompt` en `confirm`. De eerste laat de gebruiker een bericht zien, de tweede vraagt de gebruiker om invoer en de laatste vraagt bevestiging aan de gebruiker. De dialoogvensters kunnen met gelijknamige functies opgeroepen worden, waarbij parameters de knoppen, titel en tekst bepalen.

Kendo UI Het raamwerk biedt methoden om het laadscherm te tonen of te verbergen. De laadtekst voor alle laadschermen wordt opgegeven bij de initialisatie van de applicatie. Een dialoogvenster wordt in Kendo UI `ModelView` genoemd. Dit is niet hetzelfde als de View Model component van het MVVM-ontwerppatroon. Het is mogelijk een `ModelView` in JavaScript te selecteren en bijhorende methoden op te roepen. In HTML kan er gelinkt worden naar een dialoogvenster zoals er naar traditionele `Views` wordt gelinkt. Een attribuut moet dan wel weergeven dat de link naar een `ModelView` gaat.

jQuery Mobile Het standaard laadscherm is enkel een laadindicator die ronddraait, die niet opvallend aanwezig is en zonder tekst eronder. Door de opties in de API te gebruiken, komt de laadindicator duidelijk naar voor door een zwarte achtergrond en kan er ook tekst worden ondergezet. Eerst werd een plug-in [101] gebruikt om een dialoogvenster te tonen, maar deze was slecht aanpasbaar. Uiteindelijk werden de dialoogvensters van jQuery Mobile gebruikt zodat de lay-out gemakkelijker kon worden aangepast.

Lungo Een laadscherm of dialoogvenster tonen, gebeurt met dezelfde functie. Indien er geen parameters worden meegegeven, zal een laadscherm getoond worden. De parameters bevatten de titel, omschrijving, tijd op het scherm en de functie die wordt opgeroepen bij het sluiten van het venster. Daarnaast worden er ook specifieke dialoogvensters aangeboden om een succes- of foutmelding te tonen. Deze zullen respectievelijk een groene en rode kleur hebben. Als eerst een laadscherm wordt getoond, daarna wordt verborgen en daarna een dialoogvenster wordt getoond, verschijnt het dialoogvenster niet. Een oplossing hiervoor is om het laadscherm niet te verbergen, waardoor enkel het dialoogvenster zal worden getoond.

Uitdaging	**Max**	**ST**	**KUI**	**jQM**	**L**
U4.1: Maak formulieren met *placeholders* zonder labels	2	2	2	2	2
U4.2: Gebruik tekst/nummer/e-mail als formuliertypes	2	2	2	2	2
U4.3: Optieveld	2	2	2	2	0
U4.4: Aangepaste *datepicker* met bereik van data	2	0	2	1	0
U4.5: Aangepaste *datepicker* met enkel maand- en jaarveld	2	0	2	0	0
U4.6: Schakelaar	2	2	2	2	2
U4.7: Wissen van een formulier	2	0	2	0	0
Totaal	14	8	14	9	6

Tabel 5.8: Gebruik van U4: Formulieren.

5.3.4 U4: Formulieren

In tabel 5.8 worden de resultaten getoond van de zeven deeluitdagingen. Hieronder wordt per raamwerk verklaard waarom dat resultaat werd behaald.

Sencha Touch Een formulier kan in Sencha Touch aan een `Formpanel` worden toegevoegd. Dit object kan op zijn beurt voorzien worden van onder andere tekst-, e-mail- en nummervelden. Ook optievelden of schakelaars kunnen hier aan het formulier worden toegevoegd. Bij het renderen worden deze tot HTML5-invoertypes omgevormd. Een *placeholder* toevoegen kan door een veld te voorzien met de eigenschap `placeholder`.

Sencha Touch voorziet een *datepicker* maar deze is niet aanpasbaar volgens de vereisten van de POC. Wel kan een begin- en eindjaar van de *datepicker* worden geconfigureerd. Een *datepicker* maken waarbij het bereik kleiner is dan een jaar, is niet mogelijk. Ook is het onmogelijk om enkel een maand- en jaarveld te tonen. Het leegmaken van een formulier gebeurt niet automatisch wanneer het verzonden wordt. Hiervoor moet de `reset`-methode op het bijhorende `Formpanel` worden opgeroepen.

Kendo UI Formulierelementen definiëren kan via attributen door gebruik te maken van de opmaakgedreven aanpak van Kendo UI. Volgende HTML5-invoertypes worden onder andere door Kendo UI ondersteund: `text`, `email`, `number` en `radio`. HTML5 voorziet het invoertype `range` voor een veld met beperkte invoer maar Kendo UI heeft een `Switch` om een schakelaar te maken.

Datepickers worden als *widget* aangeboden in Kendo UI Web. Het bereik van de selectie kan worden ingeperkt door de minimum en maximum eigenschap te zetten. Enkel maand- en jaarvelden tonen kan door de diepte van de *datepicker* in te stellen. Deze eigenschappen worden bij initialisatie van het object meegegeven.

Het wissen van formulieren steunt op het MVVM-ontwerppatroon. Een formulier kan worden gebonden aan een (View) Model. Wanneer een uitgave wordt toegevoegd, zal de huidige waarde van het (View) Model worden gereset. Door de dubbele binding

tussen het formulier en het (View) Model zal ook de inhoud van de formulierelementen worden gewist.

jQuery Mobile Voor het toevoegen van *placeholders* in de formuliervelden werd beroep gedaan op het `placeholder`-attribuut van HTML5. Labels zijn verplicht in jQuery Mobile, maar kunnen onzichtbaar worden gemaakt met de CSS-klasse `ui-hide-label` [58]. Als er wordt teruggekeerd naar het formulier nadat het verzonden is, bevat het nog alle waarden. Na het versturen van het formulier, wordt het leeggemaakt met behulp van de `reset`-functie in JavaScript.

Voor de types van de formuliervelden werd beroep gedaan op de HTML-invoertypes, gelijkaardig met Kendo UI. Het type voor een datum werd echter niet gebruikt omwille van twee redenen. Ten eerste was hiervoor een slechte ondersteuning naar mobiele browsers toe [24]. Android 2.3 ondersteunt dit niet en de *placeholder* tekst in het veld ontbrak op iOS 6 en Android 4.2. Hierdoor weet de gebruiker in eerste instantie niet wat hij hier moet invullen. Zelf een *placeholder* instellen is onmogelijk [11]. Een tweede probleem was het opleggen van het bereik van datums, wat ook onmogelijk is. Beide problemen werden opgelost door gebruik te maken van de Date & Time Picker van Mobiscroll [80] die ook aangepaste lay-out heeft conform met die van jQuery Mobile. Het veld heeft dan wel tekst als invoertype. Het is dus in principe mogelijk om iets anders dan een datum in te geven. Dit wordt belet door ook nog eens een datumvalidatie (zie 5.3.8) te doen op dit tekstveld mocht de plug-in het niet hebben afgedwongen.

Het was ook nodig om enkel de maand en het jaar in te geven als datum, dus zonder dag. Ook hier kon niet het `date`-type gebruikt worden, omdat daar ook een dag voor nodig is. Daardoor werden de maanden handmatig geprogrammeerd als vaste lijstitems. De jaren zijn dynamisch en zijn telkens dit jaar, het volgende en het vorige jaar. Deze functionaliteit kon ook met de plug-in van Mobiscroll worden verwezenlijkt. Als laatste werd zowel het optieveld als de schakelaar door jQuery Mobile zelf aangeboden en konden direct gebruikt worden.

Lungo Het toevoegen van *placeholders* in de formuliervelden gebeurt met het HTML5-atttribuut `placeholder`. In Lungo zijn labels niet verplicht. Indien deze niet gewenst zijn, kunnen deze gewoon uit de HMTL5-code weggelaten worden.

Voor de types van de formuliervelden werd beroep gedaan op de HTML-invoertypes, gelijkaardig met Kendo UI. Door de twee aangehaalde problemen bij jQuery Mobile werd een plug-in gebruikt om de functionaliteit met datums op te lossen. De `date-picker` werd gebruikt van de plug-in pagina van Lungo zelf [126]. Bij deze plug-in is al voorbeeldcode aanwezig die nodig is om automatisch een *datepicker* te openen en de aangeklikte datum in het formulierveld te zetten. De plug-in laat echter niet toe om een bereik op te geven. De datum met enkel een maand en jaar diende handmatig geprogrammeerd te worden omdat de aangeboden plug-in hiervoor geen ondersteuning bood. De jaren zijn dynamisch en zijn telkens dit jaar, het volgende en het vorige jaar.

Een optieveld werd niet aangeboden door Lungo en werd vervangen door een dropdownmenu. Een schakelaar daarentegen werd dan weer wel aangeboden. Het

Uitdaging	**Max**	**ST**	**KUI**	**jQM**	**L**
U5.1 Vul formulierelementen met object	2	2	2	0	0
U5.2 Maak elementen van het formulier *read-only*	2	2	2	0	0
Totaal	4	4	4	0	0

Tabel 5.9: Gebruik van U5: Automatisch invullen van formulier.

legen van een formulier gebeurt door de `reset`-functie in JavaScript op te roepen op dat formulier.

5.3.5 U5: Automatisch invullen van formulier

In tabel 5.9 worden de resultaten getoond van de twee deeluitdagingen. Hieronder wordt per raamwerk verklaard waarom dat resultaat werd behaald.

Sencha Touch Het invullen van een formulier wordt ondersteund door het MVC-ontwerppatroon. Twee verschillende methoden worden in de POC gebruikt. De eerste methode zal een formulier vullen door er een instantie van een model aan toe te kennen. Sencha Touch zal automatisch de velden invullen waarbij de naam gelijk is aan de eigenschap van het model. Zo kan tekst op tekstvelden worden gemapt, nummers op numerieke velden en `booleans` op schakelaars. Er bestaat echter geen exacte mapping tussen een model en `radio`-velden. Deze moeten apart worden ingevuld door een extra methode op te roepen. De tweede methode voor het invullen van formulieren maakt gebruik van een `Navigationview` en wordt besproken in 5.3.11. De `readOnly`-eigenschap bepaalt de aanpasbaarheid van formulierelementen. Bij `radio`-velden heet deze eigenschap `disabled`.

Kendo UI Het invullen van een formulier steunt op het MVVM-ontwerppatroon. De dubbele binding tussen een formulier en (View) Model wordt met een HTML-tag aangegeven. Een View Model wordt in Kendo UI `ObservableObject` genoemd. Een `Model` breidt een `ObservableObject` uit met de mogelijkheid om schema's, velden en methoden te definiëren. Om een formulier met data te vullen is het de taak van de programmeur de eigenschappen van het `ObservableObject` van de gewenste waarden te voorzien. De gebonden formulierelementen zullen vervolgens automatisch worden ingevuld. *Read-only* velden moeten als attribuut in het formulierelement worden gespecificeerd.

jQuery Mobile Het vullen van formuliervelden wordt niet door jQuery Mobile geautomatiseerd. Hierdoor moet ieder formulierveld worden gezocht om daarna zijn waarde in te stellen. Bij een dropdownmenu en `radio`-velden kan deze methode niet worden gebruikt. Voor de eerstgenoemde moet de gewenste optie worden gezocht en daaraan het `selected`- of `checked`-attribuut moeten worden toegevoegd. Het *read-only* maken van velden gebeurt via het HTML-attribuut `readonly`. Dit geldt voor alle types van velden, behalve voor dropdownmenu's en `radio`-velden waar

Uitdaging	Max	ST	KUI	jQM	L
U6.1 Zoek suggesties gebaseerd op gebruikersinvoer	2	1	2	2	1
U6.2 Toon suggesties in klikbaar dropdownmenu	2	0	2	2	1
Totaal	4	1	4	4	2

Tabel 5.10: Gebruik van U6: Auto-aanvullen.

`disabled` moeten worden gebruikt. De andere niet-benodigde items worden uit de lijst verwijderd.

Lungo Velden vullen met data dient handmatig te gebeuren door eerst het formulierveld op te zoeken en daarna de waarde te zetten. Deze waarde moet tekst zijn, waardoor bijvoorbeeld getallen eerst moeten worden omgevormd. Geoptimaliseerde mobiele lay-out voor `radio`-velden is niet aanwezig in Lungo. Het *read-only* maken van velden gebeurt via het HTML-attribuut `readonly`. Dit gaat voor alle types van velden, behalve voor dropdownmenu's. Daar worden de niet-benodigde opties uit de selectie verwijderd.

5.3.6 U6: Auto-aanvullen

In tabel 5.10 worden de resultaten getoond van de twee deeluitdagingen. Hieronder wordt per raamwerk verklaard waarom dat resultaat werd behaald.

Sencha Touch Het automatisch aanvullen van een formulierelement steunt op een plug-in van Tajur [121]. Deze plug-in is niet op de Sencha Market terug te vinden. Het toevoegen van de plug-in zal een invoerelement beschikbaar maken dat suggesties met een dropdownmenu weergeeft. Dit element kan een `Proxy` definiëren die de *backend* aanspreekt om suggesties asynchroon op te halen. De *backend* van de POC geeft bij een bepaalde invoer suggesties in een JSON-rij terug. De rij is voorzien van een sleutel, maar alle elementen van de rij hebben er geen. Geen van de beschikbare methoden was in staat de rij met suggesties te parsen van rij-element naar modelinstantie. Hierdoor kon geen klikbaar dropdownmenu worden getoond.

Kendo UI Het automatisch aanvullen van een formulierelement wordt als *widget* door Kendo UI Web aangeboden. Suggesties van het element kunnen zowel door een lokale als externe bron worden aangeleverd. Externe suggesties moeten via een `DataSource` worden ingeladen. Ook het minimale aantal suggesties en een filter kunnen worden opgegeven. De filter bepaalt de methode om suggesties op te halen en kan `startswith` zijn.

jQuery Mobile Indien er wordt gebruik gemaakt van versie 1.2 is een plug-in nodig om auto-aanvulling te bekomen. Hiervoor kan de plug-in van Andy Matthews worden gebruikt die zowel met lokale data als externe data kan werken [72]. Sinds versie 1.3 voorziet jQuery Mobile deze functionaliteit zelf [51]. De filterfunctie moet zelf worden geschreven, maar code op de site kon als voorbeeld worden overgenomen.

Uitdaging	Max	ST	KUI	jQM	L
U7.1 Kies een afbeelding	2	1	2	2	2
U7.2 Converteer de afbeelding naar Base64	2	1	1	1	1
U7.3 Voorbeeld van de afbeelding	2	2	1	1	1
Totaal	6	4	4	4	4

Tabel 5.11: Gebruik van U7: Toevoegen en verwerken van afbeelding.

Lungo Standaard biedt Lungo geen auto-aanvullig aan, maar wel op zijn site van plug-ins [126]. Daar werd de plug-in AutoComplete gebruikt. De voorbeeldcode maakt het gemakkelijk om onmiddellijk een werkend voorbeeld van de plug-in te hebben. Veel code kon dus worden overgenomen om een werkende auto-aanvulling te bekomen. Er diende nog één extra CSS-regel te worden toegevoegd om het symbool voor de suggesties te verwijderen.

5.3.7 U7: Toevoegen en verwerken van afbeelding

In tabel 5.11 worden de resultaten getoond van de drie deeluitdagingen. Hieronder wordt per raamwerk verklaard waarom dat resultaat werd behaald.

Sencha Touch Het opladen van een afbeelding steunt op een plug-in van Smirnov [115] en kan in de Sencha Market gevonden worden. De plug-in is generiek voor het opladen van elk type bestand. De plug-in voorziet twee modes voor het opladen van bestanden: lokaal of extern. De eerste laat toe afbeeldingen lokaal of in het DOM als Base64 op te laden. De tweede mode zal bestanden naar de externe server versturen. De POC vereist de eerste aanpak. Nadat een bestand is opgeladen zal een gebeurtenis het slagen of falen van de operatie bepalen. Het is de taak van een `Controller` om deze gebeurtenissen op te vangen en te delegeren naar een bijhorende methode. De succesfunctie krijgt de Base64-tekst mee en kan een voorbeeld van de afbeelding weergeven.

Kendo UI Kendo UI Web biedt een *widget* aan die het opladen van bestanden toelaat. Deze *widget* kan in twee modi worden gebruikt: synchroon of asynchroon. In synchrone modus wordt het formulier samen met de afbeelding verzonden als een uitgave wordt toegevoegd. In asynchrone modus gebeurt het opladen meteen na het selecteren van de afbeelding. Deze methode steunt op de HTML5 FileReaderAPI. Omdat een voorbeeld van de afbeelding werd gevraagd, is de asynchrone oplossing gekozen. Voor ASP.NET MVC, JSP en PHP is een implementatie beschikbaar om het opladen van bestanden aan serverzijde af te handelen. Er werd gekozen om de PHP-implementatie te gebruiken omdat deze technologie reeds gekend was. Wanneer een afbeelding succesvol is opgeladen, wordt het bestand met een `FileReaderAPI` gelezen, aan een `canvas` toegevoegd en naar Base64 omgezet met de `toDataURL`-methode.

jQuery Mobile Het toevoegen van een afbeelding gebeurt door het bestandstype als invoertype aan het formulierveld toe te voegen. In versie 1.2 wordt dit veld

Uitdaging	Max	ST	KUI	jQM	L
U8.1 Validatieregels voor verplichte nummer- en e-mailvelden	2	2	2	1	0
U8.2 Validatieregels voor eigen condities	2	2	2	1	0
U8.3 Foutboodschappen ophalen van ongeldige velden	2	2	2	1	0
U8.4 Rode randen plaatsen rond ongeldige velden	2	0	0	0	0
Totaal	8	6	6	3	0

Tabel 5.12: Gebruik van U8: Formuliervalidatie.

niet opgemaakt met lay-out, maar dit gebeurt wel in versie 1.3 [52]. Het omvormen van de afbeelding naar Base64 is analoog aan Kendo UI. Het voorvertonen van de geüploade afbeelding hangt af van het mobiele besturingssysteem. Zo wordt op iOS 6 een miniatuurafbeelding getoond, terwijl op Android de bestandsnaam wordt getoond in het invoerveld. Het is natuurlijk ook mogelijk om de voorvertoning na conversie zelf te tonen op het scherm. Bij iOS zouden er dan twee voorvertoningen te zien zijn op hetzelfde scherm.

Lungo Een afbeelding kiezen gebeurt door aan het formulierveld het bestandstype toe te voegen. De methode om een afbeelding om te vormen naar Base64 is analoog met Kendo UI. Het voorvertonen van de geüploade afbeelding gebruikt dezelfde aanpak als jQuery Mobile.

5.3.8 U8: Formuliervalidatie

In tabel 5.12 worden de resultaten getoond van de vier deeluitdagingen. Hieronder wordt per raamwerk verklaard waarom dat resultaat werd behaald.

Sencha Touch Een model kan worden voorzien van validatieregels. Deze regels worden in een rij aan de `validations`-eigenschap van een model toegekend. Verplichte velden moeten worden aangeduid met `presence`. De controle op geldige e-mailadressen kan door `email` als validatieregel aan het invoerelement toe te kennen. Controleren op een nummer kan met `format` en de `\d+` reguliere expressie. Om een bepaalde modelinstantie te valideren moet de `validate`-methode op de instantie worden opgeroepen. Om eigen validatieregels toe te laten moet de implementatie van deze methode worden overschreven [65]. Deze functionaliteit zit dus niet standaard in Sencha Touch. Met de nieuwe `validate`-methode kan een `validator` aan een validatieregel worden toegevoegd. Dit is een functie die de programmeur zelf bepaalt en `true` of `false` teruggeeft bij het al dan niet slagen van een conditie.

Het opbouwen van een foutboodschap kan door te itereren over de fouten die na validatie worden teruggegeven. Een specifieke foutboodschap kan aan elke validatieregel worden toegekend. Incorrecte formulierelementen aanduiden met een rode rand

wordt niet door Sencha Touch ondersteund. Foutief ingevulde formulierelementen moeten na validatie met een eigen CSS-klasse worden aangevuld.

KENDO UI Het Kendo UI-raamwerk ondersteunt de validaties zoals aangeboden binnen HTML5. Het `required`-attribuut maakt de velden verplicht `email` en `number` als invoertype zullen de opgegeven waarden controleren op geldigheid. De Kendo UI `validator` gebruikt attributen van HTML5-validaties. HTML5-validatie wordt echter niet ondersteund op mobiele browsers [24]. Kendo UI zal de HTML5-attributen omvormen tot een validatie in JavaScript. Het definiëren van een eigen validatieregel kan door een JavaScript-functie aan de `validator` toe te voegen. Bij het controleren van invoerelementen worden altijd eerst de standaard validatieregels gecontroleerd, daarna de eigen validatieregels. De volgorde waarin de controles worden uitgevoerd, ligt vast en zal stoppen zodra één controle mislukt. Validatieberichten kunnen voor standaard validaties door het raamwerk zelf worden opgebouwd. Deze kunnen worden overschreven door zelf een bericht aan het invoerelement toe te kennen. Bij eigen validatieregels kan een bericht per validatieregel worden gespecificeerd.

Incorrecte formulierelementen aanduiden met een rode rand wordt niet door Kendo UI ondersteund. De standaard implementatie voorziet een `tooltip` per incorrect veld. Deze zal het validatiebericht naast het invalide formulierelement plaatsen. Een venster tonen waarbij alle foutboodschappen zijn samengevat is ook niet standaard aanwezig. De volledige foutboodschap moet worden geconstrueerd door foutboodschappen, zoals teruggegeven door de `validator`, te concateneren. Hoe een dialoogvenster gemaakt wordt, werd in sectie 5.3.3 besproken.

JQUERY MOBILE Validatie is niet standaard aanwezig in jQuery Mobile. Eerst werd geprobeerd om de verplichte velden te voorzien van het `required`-attribuut in HTML5. Doordat er geen ondersteuning is voor mobiele browsers, zoals opgemerkt bij Kendo UI, werd de plug-in van Jörn Zaefferer gebruikt [144]. Deze kan op twee manieren gebruikt worden: enerzijds annoteren van de formuliervelden met speciale CSS-klassen in de HTML-code en anderzijds door programmatie met JavaScript. Beide aanpakken werden gebruikt doorheen de implementatie. De plug-in bevat alle gevraagde validatieregels. Daarnaast was het nodig dat een veld verplicht was enkel indien een bepaalde optie was aangevinkt. Deze afhankelijkheidsrelatie is standaard aanwezig in de plug-in.

De plug-in toont standaard een foutboodschap onder het foute formulierveld. Door de uitvoerig documentatie van de plug-in, konden alle foutboodschappen samen in één dialoogvenster worden weergegeven. De plug-in annoteert de incorrecte velden met een CSS-klasse. Hierdoor kon de rode rand in CSS worden geprogrammeerd. Voor `radio`-velden en dropdownmenu's gaf dit problemen door de extra code die jQuery Mobile genereert rond deze velden. Als oplossing moest de omvattende code worden geannoteerd.

LUNGO Validatie is niet aanwezig in Lungo en er kon ook geen plug-in voor QuoJS gevonden worden. Er werd geprobeerd om bestaande plug-ins voor andere JavaScript-bibliotheken om te vormen en deze te laten werken met QuoJS. Aangezien deze manier niet direct een oplossing bracht en er geen beroep kon worden gedaan op HTML5-validatie (zie Kendo UI), moest alle validatie manueel geprogrammeerd worden.

Uitdaging	Max	ST	KUI	jQM	L
U9.1 Teken een handtekening met vinger of pen	2	1	1	1	0
Totaal	2	1	1	1	0

Tabel 5.13: Gebruik van U9: Handtekening.

Bij wijze van voorbeeld werd enkel validatie op het loginscherm geprogrammeerd. Validatie op de rest van de formulieren doorheen de POC is volgens dezelfde werkwijze mogelijk, maar werd niet geïmplementeerd. Het tonen van foutboodschappen alsook het tonen van een rode rand rond de foute velden werden ook zelf geprogrammeerd. Bij een incorrect veld werd een CSS-klasse toegevoegd die zorgde voor een rode rand.

5.3.9 U9: Handtekening

In tabel 5.13 worden de resultaten getoond van de deeluitdaging. Hieronder wordt per raamwerk verklaard waarom dat resultaat werd behaald.

Sencha Touch Het tekenen van een handtekening steunt op een plug-in van SimFla [114] en is in de Sencha Market te vinden. De plug-in maakt een nieuw invoerelement beschikbaar dat in een formulier kan worden gebruikt. Dit element maakt gebruik van het HTML5 `canvas` en retourneert de handtekening als Base64-tekst.

Kendo UI Aangezien Kendo UI steunt op de jQuery bibliotheek is Kendo UI ook compatibel met jQuery plug-ins. De jSignature handtekening plug-in van Willow Systems [120] werd geïmplementeerd. De plug-in maakt gebruik van het HTML5 `canvas`-element en de `toDataURL`-methode. Hierdoor kan de Base64-string bekomen worden die naar de *backend* moet worden gestuurd.

jQuery Mobile Er werd gezocht naar een plug-in om deze functionaliteit te bekomen, doordat jQuery Mobile dit niet standaard aanbiedt. De eerst gevonden plug-in [12] werd niet gebruikt wegens problemen met het aanpassen van de lay-out. Uiteindelijk werd dezelfde plug-in als bij Kendo UI gebruikt. Deze gaf ook het voordeel dat het gebied van de handtekening de volledige schermbreedte innam. Dezelfde aanpak zoals bij Kendo UI werd gevolgd voor het omvormen naar Base64.

Lungo Het maken van een handtekening is niet standaard aanwezig en daarenboven kon ook geen plug-in worden gevonden.

5.3.10 U10: AJAX: tekst, JSON en XML

In tabel 5.14 worden de resultaten getoond van de vier deeluitdagingen. Hieronder wordt per raamwerk verklaard waarom dat resultaat werd behaald.

Sencha Touch AJAX-verzoeken kunnen zowel expliciet via een directe oproep met `Ext.Ajax.request` als impliciet via `Stores` worden uitgevoerd. De expliciete

Uitdaging	Max	ST	KUI	jQM	L
U10.1 Haal tekst zonder opmaak op	2	2	2	2	2
U10.2 Haal JSON-data op en parse	2	2	2	2	2
U10.3 Zend JSON-data	2	2	2	2	0
U10.4 Haal XML-data op en parse	2	2	2	2	2
Totaal	8	8	8	8	6

Tabel 5.14: Gebruik van U10: AJAX: tekst, JSON en XML.

oproep is gelijkaardig aan de `$.ajax`-methode van de jQuery bibliotheek. De enige uitzondering is te vinden bij AJAX-verzoeken naar een ander domein. Om aan de CORS-standaarden (Cross-Origin Resource Sharing) te voldoen moet de eigenschap `useDefaultXhrHeader` op `false` worden gezet.

De tweede impliciete methode voor AJAX-verzoeken is via `Stores`. Een `Store` wordt voorzien van een `Proxy`. Een `Proxy` kan data verzenden via AJAX. Hierbij kunnen `Readers` geconfigureerd worden om data van de server te lezen. Sencha Touch voorziet drie methoden om de resultaten van een `Proxy` te parsen naar modelinstanties:

`JsonReader` parst JSON-sleutels naar velden van een model.

`XmlReader` parst XML-tags naar velden van een model.

`ArrayReader` mapt elementen van een rij op velden van een model.

Het verzenden van een JSON-*payload* moet met een expliciet AJAX-verzoek gebeuren. Na het encoderen kan de JSON-data aan het verzoek worden gekoppeld.

Kendo UI Om asynchrone verzoeken naar de *backend* te implementeren, moet een `DataSource` worden gebruikt. Deze bevat een `transport`-eigenschap die data kan creëren (`create`-eigenschap), lezen (`read`-eigenschap), verwijderen (`destroy`-eigenschap) en op te waarderen (`update`-eigenschap). Deze vier eigenschappen moeten geconfigureerd worden zoals de `$.ajax`-methode van de jQuery-bibliotheek.

Hoe data moet worden geparset, staat gedefinieerd in de `schema`-eigenschap van de `DataSource`. Zowel JSON als XML worden ondersteund. Aan een `schema` kan een (View) Model worden toegekend. Er onderscheiden zich twee gevallen: één bestaand `ObservableObject` kan met data worden geladen of nieuwe instanties van een `Model` kunnen worden aangemaakt. Het eerste geval zal de velden van één `ObservableObject` wijzigen als CRUD-operaties worden uitgevoerd. De eigenschap moet dan aan het `ObservableObject` worden gelijkgesteld. Het tweede geval zal de instanties van een `Model` wijzigen als CRUD-operaties worden uitgevoerd. Hiervoor moet het model in de `schema`-eigenschap worden gedefinieerd. Een `OservableObject` kan met de `toJSON`-methode als JSON-object worden verkregen. Het serialiseren kan via `JSON.stringify` en deze kan met een AJAX-verzoek worden verstuurd.

Uitdaging	Max	ST	KUI	jQM	L
U11.1 Laad data in lijst en stijl de elementen met een sjabloon	2	2	2	0	0
U11.2 Klikbare lijstelementen met actie of link naar het record van het element	2	2	0	0	0
U11.3 Sorteren van data	2	2	2	0	2
Totaal	6	6	4	0	2

Tabel 5.15: Gebruik van U11: Lijsten.

jQuery Mobile Het maken van oproepen via AJAX gebeurt door `$.ajax` in jQuery. Bij de oproep wordt ingesteld wat het te verwachten antwoord is (tekst, JSON of XML). Met tekst en JSON kan onmiddellijk worden omgegaan. Gegevens uit XML halen vraagt meer werk doordat *selectors* moet worden gebruikt.

Het versturen van JSON is gelijkaardig met het versturen van andere data. Eerst moet de JSON-data worden omgezet naar tekst, wat gebeurt door `JSON.stringify`. Daarna moet in het AJAX-verzoek worden aangegeven dat de inhoud JSON is, wat gebeurt door `contentType:"application/json"`.

Lungo Standaard biedt Lungo functies aan voor het ophalen en versturen van data via AJAX. Deze functies zullen intern de functies van QuoJS oproepen. Tekst en JSON kunnen onmiddellijk worden gebruikt. Voor XML dient er gebruik te worden gemaakt van de *selector* in QuoJS om de gevraagde data op te zoeken.

Bij het versturen van JSON-data konden de functies van Lungo zelf niet worden gebruikt. Deze hadden te weinig opties om aan te geven dat de verstuurde data JSON was. Hierdoor werden de functies van QuoJS gebruikt, die meer opties hadden. Toch bleef er een probleem bij het versturen van JSON-*payload*. QuoJS wil namelijk altijd de parameters serialiseren. Dit is uiteraard niet nodig als ruwe data, zoals JSON, wordt meegegeven. Aangezien dit een fout was in de bibliotheek werd het JavaScript-bestand zelf aangepast.

5.3.11 U11: Lijsten

In tabel 5.15 worden de resultaten getoond van de drie deeluitdagingen. Hieronder wordt per raamwerk verklaard waarom dat resultaat werd behaald.

Sencha Touch Een lijst kan voorzien worden van een sjabloon. Hier kunnen HTML-tags de lay-out van de lijstelementen vastleggen of kan er JavaScript-code worden uitvoeren. Zoals reeds besproken in sectie 5.3.5, kan een formulier worden ingevuld met een `Navigationview`. Deze `View` heeft een `push`- en `pop`-methode om een `View` op een stapel te plaatsen of af te halen. Om een `View` te tonen die hoort bij een lijstelement, wordt gebruik gemaakt van deze `Navigationview` en de `push`-methode.

Een `Controller` merkt het aanraken van een lijstelement en zal de `push`-methode op de `Navigationview` oproepen. De methode kan geparameteriseerd worden met de modelinstantie die hoort bij het lijstelement. Hierdoor zal de nieuwe `View` worden gevuld met waarden van de modelinstantie. Automatisch zal er ook een *back* knop worden gegenereerd die na aanklikken de `pop`-methode zal oproepen.

Een `Store` kan aan een lijst worden gekoppeld zodat alle modelinstanties van de `Store` in de lijst worden weergegeven. Het sorteren van een lijst kan automatisch door de `Store` te voorzien van een `Sorter`. Deze kan modelinstanties van een `Store` sorteren op basis van eigenschappen van het bijhorende model. Ook kan de richting van sorteren worden geconfigureerd.

KENDO UI Lijsten worden met een `Listview` gemaakt en kunnen worden gebonden met een `DataSource`. Hierdoor zullen alle instanties van de `DataSource` als element in de lijst verschijnen. Ook kan een JavaScript-rij aan een `Listview` worden gebonden op basis van de `source`-eigenschap. Deze methodiek is uitvoerig in de documentatie beschreven. De opmaak van lijsten kan met `Templates` worden uitgedrukt. Dit zijn sjablonen die met specifieke scripts in een HTML-bestand worden geschreven. De sjablonen hebben toegang tot de velden van de modelinstanties en kunnen JavaScript-functies uitvoeren.

De link van elk lijstelement moet in het sjabloon worden gedefinieerd. Om de elementen uit het uitgavenoverzicht te linken, werd gebruik gemaakt van een geparameteriseerde `View`. Dit laat toe om parameters in de link naar de `View` op te geven, analoog aan HTTP GET-verzoeken. De functie zal een `ObservableObject` laden met data op basis van de meegegeven parameter. Dit `ObservableObject` is gekoppeld aan een formulier dat automatisch zal worden ingevuld zodra het object wordt geïnitialiseerd. Het sorteren van een lijst kan door de `DataSource` die aan de lijst is gekoppeld van een sorteereigenschap te voorzien.

JQUERY MOBILE Het laden van data in een lijst dient zelf geprogrammeerd te worden. Via jQuery wordt één voor één een item toegevoegd aan de lijst. Na alle elementen te hebben toegevoegd, moet de lijst ververst worden zodat jQuery Mobile de correcte lay-out toepast op de volledige lijst Het klikbaar maken van de gegenereerde lijstitems gebeurt bij het toevoegen van de items zelf. Ieder lijstitem krijgt een unieke *identifier* waarmee de uit te voeren actie wordt bepaald. Er komt geen functionaliteit van het raamwerk om data te sorteren. Eerst moet zelf een vergelijkingsfunctie in JavaScript worden geschreven, waarna die aan de sorteerfunctie van JavaScript wordt meegegeven.

LUNGO Het laden van data in een lijst dient zelf geprogrammeerd te worden. Items worden één voor één toegevoegd aan de lijst met behulp van QuoJS. Het is daarentegen niet nodig om de lijst te verversen. Om de gegenereerde lijstitems klikbaar te maken, wordt de werkwijze van jQuery Mobile gebruikt. Dit gebeurt bij het genereren van de items zelf en op basis van een unieke *identifier* die de uit te voeren actie bepaald. Het sorteren van data gebeurt door de aangeboden functies van het raamwerk zelf. Deze maken het mogelijk om te sorteren volgens een bepaalde eigenschap, zowel oplopend als aflopend.

Uitdaging	Max	ST	KUI	jQM	L
U12.1 Vraag een PDF-bestand op met POST-parameters	2	1	0	0	0
U12.2 Toon het PDF-bestand	2	1	1	1	1
Totaal	4	2	1	1	1

Tabel 5.16: Gebruik van U12: Toon PDF.

5.3.12 U12: Toon PDF

In tabel 5.16 worden de resultaten getoond van de twee deeluitdagingen. Hieronder wordt per raamwerk verklaard waarom dat resultaat werd behaald.

Sencha Touch Het tonen van een PDF steunt op een plug-in van Fiedler [28] en kan op de Sencha Market gevonden worden. De PDF zal in een paneel met een koptekst worden getoond. Het paneel toont één pagina van het PDF-bestand, de koptekst bevat de navigatie naar andere pagina's. Om de plug-in in de POC te gebruiken, waren echter twee aanpassingen noodzakelijk. Het PDF-bestand moet via een POST-verzoek worden opgehaald waarbij parameters het exacte PDF-bestand aanduiden. Ook moest er een terugknop in de koptekst van het paneel worden aangebracht om terug naar het overzicht van doorgestuurde formulieren te gaan.

Kendo UI AJAX is bedoeld om tekst op te halen, maar geen ruwe data zoals een PDF [105]. Hierdoor werd gebruik gemaakt van een verborgen formulier met de nodige parameters die de PDF ophaalt bij de *backend*. Na het klikken op een lijstitem in het overzicht, wordt dit verborgen formulier opgestuurd naar de *backend* die dan een PDF teruggeeft in de browser. Het weergeven van de PDF wordt overgelaten aan het mobiel apparaat dat de correcte applicatie hiervoor opstart.

jQuery Mobile De implementatie voor het tonen van het PDF-bestand is analoog als de implementatie met Kendo UI.

Lungo De implementatie voor het tonen van het PDF-bestand is analoog als de implementatie met Kendo UI.

5.3.13 U13: Offline

In tabel 5.17 worden de resultaten getoond van de twee deeluitdagingen. Hieronder wordt per raamwerk verklaard waarom dat resultaat werd behaald.

Sencha Touch Zoals besproken in sectie 5.3.10 kan een `Store` voorzien worden van een `Proxy` die data lokaal opslaat aan klantzijde. Deze `Store` maakt gebruik van de HTML5 `localStorage`. Om gegevens van een gebruiker en onverzonden onkosten lokaal te bewaren moeten twee `Stores` met deze `Proxy` worden gedefinieerd. Een belangrijke opmerking is dat geen twee `Proxies` aan dezelfde `Store` kunnen worden toegevoegd. Gegevens van een gebruiker moeten van de server worden opgehaald - met een AJAX-`Proxy` - en lokaal worden opgeslagen - met een `LocalStorageProxy`.

Uitdaging	Max	ST	KUI	jQM	L
U13.1: Bewaar data	2	2	1	1	2
U13.2: Maak de applicatie offline beschikbaar	2	2	1	1	1
Totaal	4	4	2	2	3

Tabel 5.17: Gebruik van U13: Offline.

Hiervoor zijn twee verschillende `Store` instanties nodig die gesynchronizeerd moeten worden. Bij het laden van de applicatie zal de `Store` die gebruikers lokaal opslaat, op data worden gecontroleerd. Indien er data wordt gevonden, was de gebruiker reeds ingelogd. De applicatie zal dan meteen naar het startscherm navigeren.

Om onverzonden uitgaven lokaal op te slaan, moeten de uitgaven ook aan een `Store` met lokale `Proxy` worden toegevoegd. De uitgaven worden dan automatisch naar de `localStorage` weggeschreven. De controle op onverzonden uitgaven wordt herleid tot het controleren van data in de `Store`. Het verwijderen van onverzonden lokale uitgaven kan door de data in de `Store` te wissen.

De applicatie offline beschikbaar maken wordt ondersteund door Sencha Cmd [110]. Hiervoor moet de applicatie gebouwd worden voor productie (zie sectie 3.1.1). De tool zal automatisch een `manifest`-bestand aanmaken die alle vereiste bestanden bevat die offline nodig zijn.

Kendo UI Vanuit het raamwerk is er geen ondersteuning om gegevens offline te bewaren. Hiervoor werd gebruik gemaakt van `localStorage`, wat gespecificeerd is in HTML5. Het controleren of `localStorage` al dan niet wordt ondersteund, gebeurd door Modernizr [81].

Na het aanmelden zullen alle gegevens van de werknemer geserialiseerd worden opgeslagen. Wanneer het inlogscherm wordt getoond, zal gecontroleerd worden of er gegevens van een werknemer lokaal beschikbaar zijn. Indien dit het geval is, wordt automatisch naar het startscherm genavigeerd en de `DataSource` met informatie van de gebruiker ingevuld. Een uitgave toevoegen moet zowel aan een `DataSource` als in de `localStorage` gebeuren. Nadat een uitgaveformulier is verzonden, worden de uitgaven uit beide plaatsen verwijderd.

Het offline beschikbaar maken van de applicatie wordt vanuit Kendo UI zelf niet ondersteund. Dit kan opgelost worden met HTML5 Application Cache. Om het proces te vergemakkelijken werd het `manifest`-bestand gegenereerd aan de hand van Yeoman [143].

jQuery Mobile Vanuit het raamwerk komt er standaard geen ondersteuning om gegevens offline te bewaren of de applicatie offline beschikbaar te maken. Net zoals Kendo UI werden lokale opslag en HTML5 Application Cache gebruikt. Het startscherm werd als eerste scherm gekozen, omdat jQuery Mobile altijd het eerste scherm in de HTML-code inlaadt. Indien gemerkt wordt dat de gebruiker niet aangemeld was, dan wordt hij doorverwezen naar het inlogscherm. Hierdoor zal,

Uitdaging	ST		KUI		jQM		L	
	Score	Max	Score	Max	Score	Max	Score	Max
U2: Toestelspecifieke lay-out	7	8	8	8	7	8	7	8
U4: Formulieren	33	40	34	40	38	40	22	32
U6: Auto-aanvullen			8	8	8	8	7	8
U7: Toevoegen en verwerken van afbeelding	5	8	5	8	5	8	5	8
U8: Formuliervalidatie	8	8	8	8	8	8		
U9: Handtekening	8	8	8	8	8	8		
U12: Toon PDF	6	8	5	8	5	8	5	8
U13: Offline	16	16	16	16	16	16	16	16
Totaal	83	96	92	104	95	104	62	80

Tabel 5.18: Ondersteuning per uitdaging.

Apparaat	ST		KUI		jQM		L	
	Score	Max	Score	Max	Score	Max	Score	Max
HTCDesireZ	8	12	9	13	10	13	5	10
GalaxyTab	7	12	9	13	9	13	5	10
GalaxyS	11	12	11	13	12	13	8	10
Nexus 7	12	12	12	13	13	13	9	10
iPad1 WiFi	11	12	12	13	12	13	8	10
iPad3 4G WiFi	12	12	13	13	13	13	9	10
iPhone 3GS	11	12	13	13	13	13	9	10
iPhone 4S	11	12	13	13	13	13	9	10
Totaal	83	96	92	104	95	104	62	80

Tabel 5.19: Ondersteuning per apparaat.

als de applicatie offline is, de gebruiker kunnen navigeren doorheen de applicatie doordat zijn gegevens werden bewaard.

Lungo Het raamwerk biedt zelf functies aan om data lokaal op te slaan. Dit kan zowel permanent als voor de huidige browsersessie zijn. Net zoals bij Kendo UI werd HTML5 Application Cache gebruikt.

5.4 Ondersteuning

In deze sectie zal de ondersteuning van de raamwerken op mobiele apparaten worden onderzocht. In de secties 5.4.1 tot 5.4.8 zal iedere uitdaging per raamwerk uitvoerig worden besproken. Voor de score van ondersteuning wordt naar formule 4.4 verwezen waar de som van de ondersteuning op alle acht apparaten wordt gemaakt. De scores van de vier raamwerken wordt samengevat per uitdaging in tabel 5.18. Een ander zicht op diezelfde data wordt bekomen door de ondersteuning van de vier raamwerken samen te vatten per apparaat. Deze weergave wordt getoond in tabel 5.19. Voor een gedetailleerd overzicht van de ondersteuning per apparaat wordt verwezen naar appendix C.

Zoals besproken in sectie 4.2.4 wordt enkel het raamwerk en niet een eigen implementatie op ondersteuning getest. Daarom verschilt de maximale score voor ondersteuning per raamwerk. Welke uitdagingen door het raamwerk konden worden geïmplementeerd werd in het vorige criterium geduid. De maximale scores voor Sencha Touch, Kendo UI, jQuery Mobile en Lungo zijn respectievelijk 96, 104, 104 en 80. De score voor ondersteuning is respectievelijk 83, 92, 95 en 62. Wanneer de relatieve score per raamwerk bekeken wordt, heeft Sencha Touch een score van 86%, Kendo UI 88%, jQuery Mobile 91% en Lungo 78%. Hoewel de verschillen klein zijn kan gezegd worden dat jQuery Mobile de beste ondersteuning biedt, gevolgd door een Kendo UI en Sencha Touch. Er kan geconcludeerd worden dat de raamwerken buiten Lungo behoorlijk hoog scoren. Dit wil zeggen dat de keuze voor één van de drie raamwerken onafhankelijk is van de ondersteuning. De verschillende raamwerken ondersteunen dezelfde apparaten. Dit komt doordat de raamwerken allemaal steunen op dezelfde onderliggende technologie, namelijk HTLM5.

Alle raamwerken buiten Lungo ondersteunen gemiddeld 88% van de geïmplementeerde uitdagingen. Lungo ondersteunt slechts 78% van de geïmplementeerde uitdagingen. De grootste problemen werden bij formulieren waargenomen. Ook kon er niet genavigeerd worden doorheen de volledige POC op de HTCDesireZ en GalaxyTab. Dit komt doordat de *tap* gebeurtenis werd gebruikt om te navigeren. Op de HTCDesireZ en GalaxyTab werden deze echter niet afgevuurd. Op de andere toestellen had dit een dubbelklikeffect. Hierdoor gebeurde de eerste klik op het huidig scherm en de tweede klik op het volgende. Zo kan het bijvoorbeeld gebeuren dat een menu op het volgende scherm al openklapt, zonder dat de gebruiker dit verwachtte.

Toestellen met Android 2.3 - HTCDesireZ en GalaxyTab - hebben het minste ondersteuning. Android 4 en iOS-toestellen scoren gemiddeld 95%. De gemiddelde score bij alle Android-toestellen is 79% in tegenstelling tot 95% ondersteuning op iOS-toestellen. Er moet opgemerkt worden dat enkel versie 5 en 6 van iOS werd getest (zie tabel 4.3). Van alle acht apparaten is de iPhone 3GS het oudst (officiële voorstelling in juni 2009) gevolgd door de GalaxyS (officiële voorstelling in maart 2010) [118, 34]. Hoewel beide apparaten opgewaardeerd zijn naar een recenter besturingssysteem, is de trend dat Android-toestellen dit minder vaak aanbieden ten opzichte van iOS-toestellen. De reden hiervoor is de grotere heterogeniteit tussen Android-toestellen. iOS-apparaten zullen dus sneller de nieuwste HTML5-kenmerken ondersteunen.

Een opmerkelijk resultaat is de maximale ondersteuning van formuliervalidatie en handtekening voor alle raamwerken behalve Lungo. Als laatste ondersteunen alle raamwerken offline opslag op alle beschikbare apparaten.

5.4.1 U2: Toestelspecifieke lay-out

Sencha Touch en Kendo UI voorzien methoden om de context waarin de applicatie wordt uitgevoerd, op te vragen. De implementatie van jQuery Mobile en Lungo steunt op CSS3. Het soort apparaat (smartphone of tablet) werd op alle toestellen behalve de GalaxyTab correct herkend. Door de lage resolutie werd deze als smartphone gecategoriseerd op Sencha Touch, jQuery Mobile en Lungo. Bij jQuery Mobile en

Uitdaging	ST		KUI		jQM		L	
	Score	Max	Score	Max	Score	Max	Score	Max
U4.1: Maak formulieren met *placeholders* zonder labels	8	8	4	8	8	8	8	8
U4.2: Gebruik tekst/nummer/e-mail als formuliertypes	6	8	6	8	6	8	6	8
U4.3: Optieveld	3	8	8	8	8	8		
U4.4: *Datepicker*	8	8	8	8	8	8	8	8
U4.6: Schakelaar	8	8	8	8	8	8	0	8
Totaal	33	40	34	40	38	40	22	32

Tabel 5.20: Ondersteuning van U4: Formulieren.

Lungo geldt dit niet wanneer de applicatie liggend wordt opgestart. Daarentegen herkende Kendo UI de GalaxyTab als tablet zowel in staande als liggende modus.

5.4.2 U4: Formulieren

In tabel 5.20 worden de resultaten getoond van de vijf deeluitdagingen. Hieronder wordt per raamwerk verklaard waarom dat resultaat werd behaald. De tweede deeluitdaging faalt bij ieder raamwerk op de HTCDesireZ en GalaxyTab aangezien geen e-mailtoetsenbord getoond wordt.

Sencha Touch Alle smartphones en de GalaxyTab vertoonden een probleem bij de labels van de `option`-knoppen. Deze zijn nodig om het type van uitgave te selecteren bij het aanmaken van een nieuwe uitgave. Alle labels buiten *Hotel* en *Restaurant* zijn te lang om in de voorziene ruimte weer te geven. Sencha Touch kort deze af met ellipsen waardoor niet de volledige naam leesbaar is. Dit kan tot verwarring leiden. In liggende mode worden sommige labels wel zichtbaar. De *datepicker* en schakelaar die Sencha Touch voorziet, werkten op elk toestel.

Kendo UI Alle Android-toestellen toonden een wit kader bij *read-only* invoervelden waarbij de waarde uit een *widgets* kwam. Deze zijn te zien wanneer het overzicht van een ingevoerde uitgave wordt opgevraagd. Het raamwerk moet de waarden van de *widgets* in de *read-only* invoerelementen invullen. Deze functionaliteit werkte wel op iOS-toestellen. Het openen van *datepicker* bij Kendo UI vindt plaats als de gebruiker op het icoontje in het invoerveld klikt. Indien op de datum wordt geklikt zal de gebruiker de datum tekstueel aanpassen. Hiervoor moet het correcte formaat worden gebruikt. Ook werd op alle toestellen een virtueel toetsenbord weergegeven ondanks de invoer uit de *datepicker* moet worden gekozen. De schakelaar en optievelden werkten op elk apparaat.

jQuery Mobile Op iedere deeluitdaging behaalt jQuery Mobile de maximumscore. Het scrollen in de *datepicker* werkte echter niet vloeiend op de HTCDesireZ en GalaxyTab, maar de functionaliteit werkte wel. Op de iPad1 WiFi werd opgemerkt dat door de lokale instellingen, een scheidingsteken voor de duizendtallen werd gebruikt voor het getal in het nummerveld. Als `dateString` werd het patroon `jaar/maand/dag` uiteindelijk gebruikt, want het patroon `jaar-maand-dag` gaf op de HTCDesireZ en GalaxyTab een ongeldige datum. De schakelaar en optievelden werkten op elk apparaat.

Lungo De slechtste ondersteuning was op de HTCDesireZ en GalaxyTab. Met deze toestellen kon zeer moeilijk op de velden worden geklikt. Soms lukte dit, soms lukte dit niet. Dit wordt niet in de punten gereflecteerd. De aangepaste *datepicker* werkte op ieder toestel, wat in schril contrast staat met de schakelaar die op geen enkel toestel correct werkte. Deze laatste deed verschillende acties naargelang geklikt of ge*swipe*d werd op de schakelaar.

5.4.3 U6: Auto-aanvullen

Sencha Touch kon deze uitdaging niet implementeren en dit werd bijgevolg niet op ondersteuning getest. Kendo UI en jQuery Mobile behaalde beide de maximum score. Op de HTCDesireZ werden bij Lungo de suggesties dubbel op het scherm aan de gebruiker getoond.

5.4.4 U7: Toevoegen en verwerken van afbeelding

De scores zijn gelijk voor alle toestellen. Kendo UI, jQuery Mobile en Lungo gebruiken dezelfde aanpak om deze uitdaging te implementeren. Deze aanpak steunt op de FileReaderAPI en wordt enkel vanaf Android 3 en iOS 6 ondersteund [24]. Ook de plug-in die Sencha Touch gebruikt voor deze uitdaging maakt gebruik van deze API. Hierdoor zullen de HTCDesireZ, GalaxyTab en iPad1 WiFi deze uitdaging in de vier raamwerken niet ondersteunen.

Er dient opgemerkt te worden dat de hoge resolutie camera aan de achterkant van de iPad3 4G WiFi, iPhone 3GS en iPhone 4S niet kon worden gebruikt. Dit is een gekende limiet van iOS waarbij apparaten met minder dan 256 MB RAM maximaal drie megapixels foto's kunnen importeren op het `canvas`. Apparaten met 256 MB RAM of meer kunnen foto's tot vijf megapixels importeren op het `canvas` [8]. Een gelijkaardig probleem gaat ook op voor Android-apparaten als de foto wordt opgeslagen in de lokale opslag. De exacte limieten voor `localStorage` en `sessionStorage` zijn browser- en versiespecifiek [36]. Hierdoor moet de camera aan de voorkant worden gebruikt of een afbeelding met lagere resolutie worden gekozen. De iPad1 WiFi biedt geen ondersteuning voor het formulierveld van het type `file` en zal het opladen van een afbeelding dus ook niet ondersteunen [137].

5.4.5 U8: Formuliervalidatie

HTML5-formuliervalidatie wordt op geen enkel beschikbaar mobiel besturingssysteem ondersteund [24]. Elke implementatie van deze uitdaging werd door alle toestellen ondersteund. Een specifiek mobiel probleem bij jQuery Mobile was dat bij het tonen van het dialoogvenster, de plug-in op de achtergrond de cursor op het eerste veld zette. Hierdoor verscheen het toetsenbord op het scherm van het mobiele apparaat wanneer het dialoogvenster tevoorschijn kwam, wat niet de bedoeling is. Dit werd opgelost door `focusInvalid:false` in te stellen.

Uitdaging	ST		KUI		jQM		L	
	Score	Max	Score	Max	Score	Max	Score	Max
U13.1: Bewaar data	8	8	8	8	8	8	8	8
U13.2: Maak de applicatie offline beschikbaar	8	8	8	8	8	8	8	8
Totaal	16	16	16	16	16	16	16	16

Tabel 5.21: Ondersteuning van U13: Offline.

5.4.6 U9: Handtekening

Kendo UI en jQuery Mobile hanteren dezelfde plug-in om deze uitdaging te implementeren. Onderliggend steunt deze op het `canvas`-kenmerk van HTML5. De plug-in die Sencha Touch gebruikt is ook afhankelijk van dit kenmerk. Alle onderzochte toestellen ondersteunen de `canvas` [24] en de drie raamwerken behalen vervolgens de maximumscore. Omdat bij Lungo geen plug-in werd gevonden voor deze uitdaging, kon ook de ondersteuning niet worden onderzocht.

5.4.7 U12: Toon PDF

Kendo UI, jQuery Mobile en Lungo hergebruiken dezelfde code voor deze uitdaging. Drie van de acht apparaten hadden problemen met het tonen van de PDF. Op de HTCDesireZ werd een wit scherm getoond, op de GalaxyTab lukt het niet en op de GalaxyS gaf het besturingssysteem de boodschap 'Download unsuccessful' terug. Bij de vier andere apparaten werd de PDF-lezer geopend en kon het bestand worden bekeken. De score voor deze drie raamwerken is dus 5/8.

Sencha Touch maakt gebruik van een plug-in voor het tonen van PDF-bestanden. Deze plug-in maakt gebruik van PDF.JS, een PDF-renderer van Mozilla [32]. PDF.JS is geïmplementeerd met JavaScript en HTML5. Het `canvas`-kenmerk en ondersteuning voor de SVG (Scalable Vector Graphics) API is noodzakelijk. Slechts vanaf Android 3 wordt dit laatste ondersteund. Alle gecontroleerde versies van iOS ondersteunen dit wel [24]. Het tonen van een PDF werkt dus niet op de HTCDesireZ en GalaxyTab. Een belangrijke opmerking is dat de ondersteuning van deze uitdaging werd getest op een implementatie die niet gebouwd was met Sencha Cmd. Dit was omdat de gebouwde versie een foutboodschap gaf wanneer de plug-in werd aangesproken. Sencha Touch haalt een score van 6/8 omdat de GalaxyS deze uitdaging wel ondersteunt in tegenstelling tot de drie andere raamwerken.

5.4.8 U13: Offline

In tabel 5.21 worden de resultaten getoond van de twee deeluitdagingen. Offline en opslag is één element van de acht technologieklassen van HTLM5 (zie sectie 2.5). Om de POC offline beschikbaar te maken is zowel ondersteuning voor `manifest` en `localStorage` noodzakelijk. Beide kenmerken worden door alle beschikbare mobiele besturingssystemen ondersteund [24].

Performantie	**ST**	**KUI**	**jQM**	**L**
Gemiddelde downloadtijd (s)	6.82	4.78	4.26	2.19
Gebruikerservaring (score op 32)	32	8	22	14
Score	0.21	0.60	0.19	0.16

Tabel 5.22: Overzicht van performantie.

Downloadtijden	**ST**	**KUI**	**jQM**	**L**
Gemiddelde downloadtijd (s)	4.48	4.23	3.95	1.79
Gemiddelde downloadtijd uit cache (s)	2.34	0.55	0.30	0.40
Totaal	6.82	4.78	4.26	2.19

Tabel 5.23: Gemiddelde downloadtijd van de loginapplicatie.

5.5 Performantie

De bekomen performantie voor de vier raamwerken wordt weergegeven in tabel 5.22. De performantie wordt bepaald volgens formule 4.8 die de gemiddelde downloadtijd en de gebruikerservaring bevat. Voor de score van performantie geldt hoe kleiner, hoe beter.

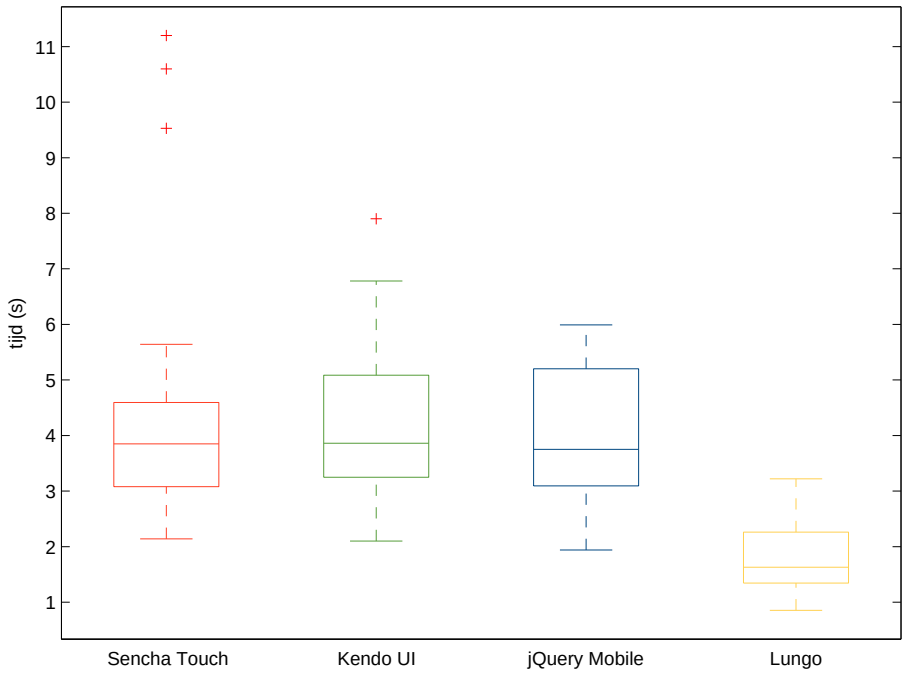
Op vlak van performantie scoort Lungo het beste, kort gevolgd door jQuery Mobile en Sencha Touch. Kendo UI heeft een beduidend lage score en is hierdoor laatst. Dit wordt verklaard doordat de eerste twee raamwerken geen ontwerppatroon afdwingen waardoor er ook een kleinere JavaScript-code dient te worden gedownload ten opzichte van raamwerken die wel een ontwerppatroon afdwingen (zie ook tabel 3.1). Sencha Touch scoort het slechtst op gemiddelde downloadtijd, maar behaalt de beste score op gebruikerservaring.

Eerst zal in sectie 5.5.1 de gemiddelde downloadtijd gedetailleerd worden besproken. Hieropvolgend zal in sectie 5.5.2 de gebruikerservaring worden besproken. Als laatste wordt in sectie 5.5.3 de performantie getoetst aan andere metrieken.

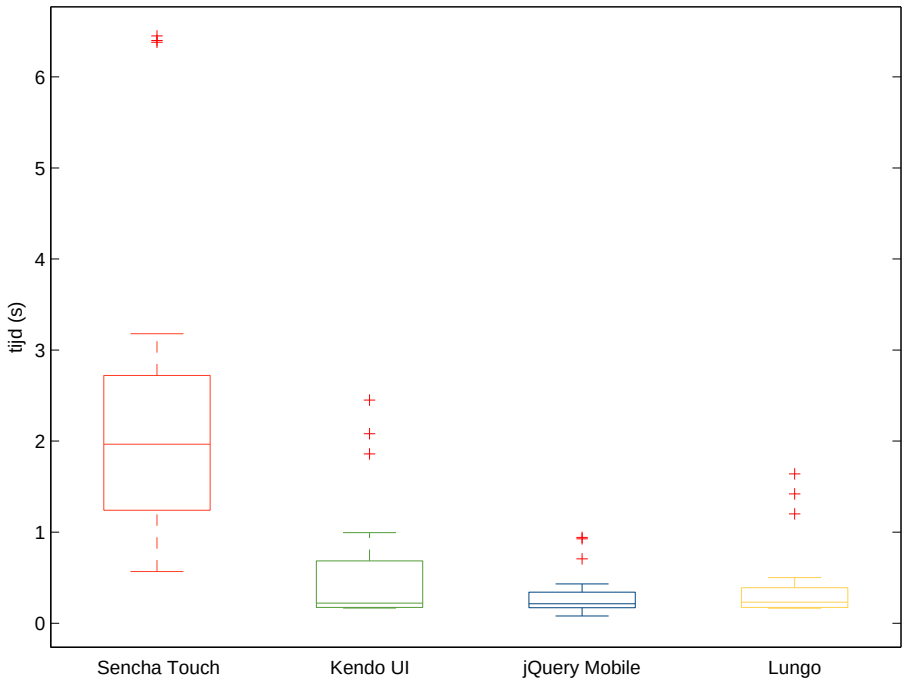
5.5.1 Gemiddelde downloadtijd

De gemiddelde downloadtijd wordt bepaald volgens formule 4.5 die zowel de loginapplicatie als de loginapplicatie uit cache in rekening brengt. De uitkomsten hiervan worden weergegeven in tabel 5.23. Om de spreiding van deze downloadtijd op de acht apparaten beter te kunnen bekijken, wordt op figuur 5.2 de downloadtijd uitgezet in een boxplot. Voor de bespreking van uitschieters wordt verwezen naar appendix D. Eerst zal deze van de loginapplicatie worden besproken en vervolgens van de loginapplicatie uit cache.

Loginapplicatie Lungo behaalt de snelste downloadtijd (zie figuur 5.2a). jQuery Mobile en Kendo UI behalen respectievelijk een tweede en derde snelste downloadtijd.



(a) Loginapplicatie



(b) Loginapplicatie uit cache

Figuur 5.2: Downloadtijden van de loginapplicatie.

Afhankelijkheden	**ST**	**KUI**	**jQM**	**L**
Grootte JavaScript-bibliotheek (kB)	506.1	280.2	277.5	47.8
Grootte JavaScript-plug-ins (kB)	0	0	38.1	0
Grootte CSS (kB)	214.1	199.1	136	115.8
Totaal	720.2	479.3	451.6	163.6

Tabel 5.24: Grootte van de afhankelijkheden van de raamwerken voor het implementeren van de loginapplicatie.

Lungo is meer dan de helft sneller dan jQuery Mobile, Kendo UI of Sencha Touch. Deze drie raamwerken behalen quasi dezelfde downloadtijd. Sencha Touch is het traagst. De verwachting is dat de downloadtijd evenredig is met de downloadgrootte van de applicatie. De afhankelijkheden van het raamwerk nemen het merendeel in van de totale downloadgrootte. Dit komt omdat er voor de loginapplicatie relatief weinig code moest worden geschreven. Tabel 5.24 toont voor alle raamwerken de grootte van de afhankelijkheden die nodig zijn om de loginapplicatie op te zetten. De implementatie van de loginapplicatie in jQuery Mobile gebruikt een plug-in voor validaties, vandaar dat ook de grootte van deze plug-in is opgenomen. Deze gegevens omvatten drie conclusies. Ten eerste kan gezien worden dat QuoJS, de JavaScript-bibliotheek van Lungo, geoptimaliseerd is voor mobiele apparaten. Verder is het al dan niet aanwezig zijn van een ontwerppatroon niet uit de grootte van de JavaScript-bestanden af te leiden. Kendo UI en jQuery Mobile hebben namelijk beide afhankelijkheden die ongeveer even groot zijn. Ten slotte kan geconcludeerd worden dat Sencha Touch de meeste JavaScript-code gebruikt in zijn afhankelijkheden. Dit komt omdat Sencha Touch een JavaScript-gedreven raamwerk is waarbij alle HTML-code ter plaatse moet worden gegenereerd.

De verwachte evenredigheid tussen downloadtijd (zie figuur 5.2) en downloadgrootte (zie tabel 5.24) wordt bevestigd door Kendo UI, jQuery Mobile en Lungo. Sencha Touch gebruikt uitgesteld parsen van JavaScript-code (zie sectie 5.5.3). Hierdoor zal de browser niet moeten wachten op het parsen en dus sneller alle afhankelijkheden downloaden.

Loginapplicatie uit cache Als naar de versie uit cache wordt gekeken, scoren Kendo UI, jQuery Mobile en Lungo hetzelfde (zie figuur 5.2b). Daarentegen behaalt Sencha Touch telkens een veel tragere tijd. Enerzijds komt dit doordat de drie eerstgenoemde raamwerken enkel gebruik maken van HTML5 Application Cache. Sencha Touch voegt het Delta Update mechanisme toe aan HTML5 Application Cache. Dit mechanisme wil voorkomen dat bij een kleine aanpassing in de code, alle bestanden opnieuw moeten worden opgehaald die in het `manifest`-bestand staan opgelijst. De `microloader` is verantwoordelijk voor het asynchroon ophalen van alle benodigde JavaScript- en CSS-bestanden. Na het bouwen van een applicatie met Sencha Cmd, zullen de gewijzigde bestanden gearchiveerd worden en worden de veranderingen tussen elke versie opgeslagen. Na het laden van de applicatie, zal de

Apparaat	ST	KUI	jQM	L
HTCDesireZ	4	3	2	1
GalaxyTab	4	2	3	1
GalaxyS	4	2	3	1
Nexus 7	4	1	3	2
iPad1 WiFi	4		2	3
iPad3 4G WiFi	4		3	2
iPhone 3GS	4		3	2
iPhone 4S	4		3	2
Totaal	32	8	22	14

Tabel 5.25: Gebruikerservaring van het scrollen door een lange lijst.

`microloader` met een GET-verzoek controleren op wijzigingen. Dit GET-verzoek zal de grootste tijd voor zijn rekening nemen bij de downloadtijden bij applicaties uit de cache. Sencha Touch heeft er dus voor gekozen om aan performantie in te boeten ten voordele van het update mechanisme [84]. Kendo UI, jQuery Mobile en Lungo gebruiken Yeoman om de applicatie te bouwen. De webapplicaties gemaakt in Sencha Touch gebruiken daarentegen Sencha Cmd.

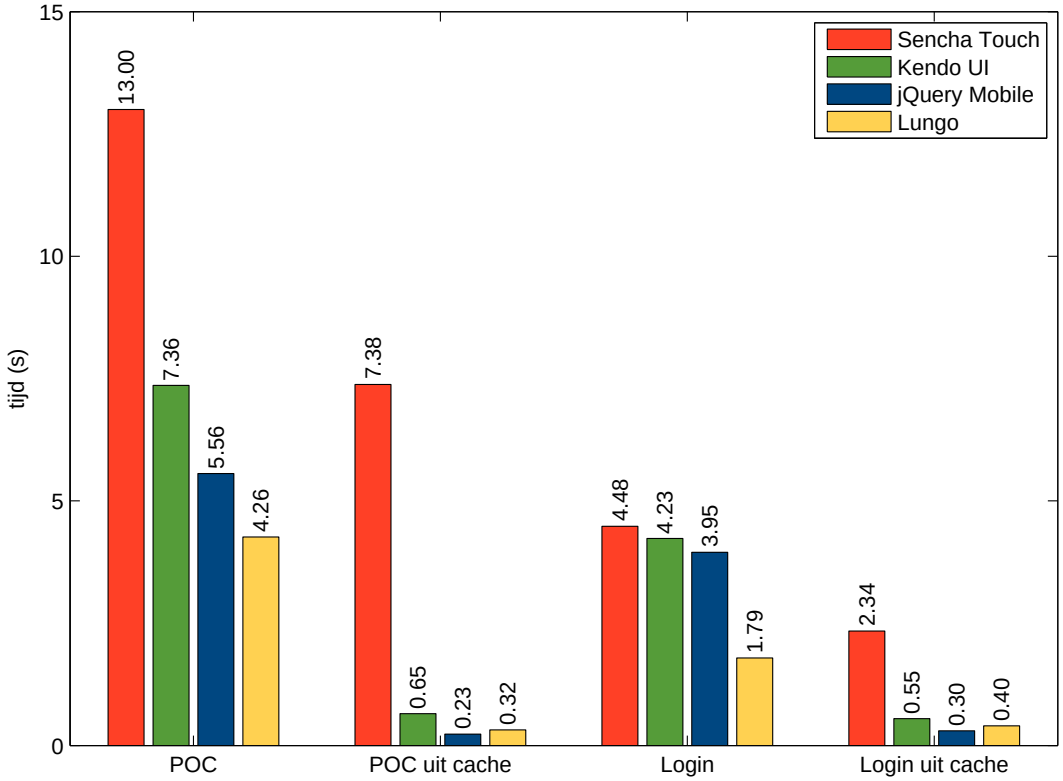
5.5.2 Gebruikerservaring

In tabel 5.25 wordt de totaalscore voor de gebruikerservaring getoond. Sencha Touch behaalt de maximale score. Dit wil zeggen dat op alle toestellen het scrollen door de lijst van Sencha Touch het vlotst werd ervaren. jQuery Mobile werd zes keer als tweede beste beoordeeld. Op de HTCDesireZ liep Kendo UI vlotter, op de iPad1 WiFi was Lungo nummer twee. De lijst genereren met Kendo UI op iOS-toestellen was onmogelijk omdat de applicatie de browser liet crashen. De reden waarom alsook de grens van het aantal lijstelementen wanneer Kendo UI crasht op iOS-toestellen werd door tijdsbudget niet gecontroleerd. Een mogelijke denkpiste is dat Kendo UI een overhead genereert die het maximale toegelaten geheugen voor het iOS-besturingssysteem overschrijdt. Op Android-toestellen kon de lijst echter wel worden getoond. De score van Kendo UI is dus slechts voor vier apparaten.

5.5.3 Duiding

In wat volgt zullen metrieken worden besproken die de score van de performantie zullen duiden. Enerzijds wordt de gemiddelde downloadtijd getoond voor zowel de POC als loginapplicatie en anderzijds wordt naar de score van Google PageSpeed gekeken.

Downloadtijd POC versus loginapplicatie Op figuur 5.3 wordt de gemiddelde downloadtijd van de POC en loginapplicatie, zowel gewoon als uit cache, voor



Figuur 5.3: Gemiddelde downloadtijd van POC, POC uit cache, loginapplicatie en loginapplicatie uit cache voor elk raamwerk.

de vier raamwerken getoond. Voor de gemiddelde downloadtijd per apparaat en per raamwerk wordt verwezen naar appendix D.

De tragere downloadtijd van Sencha Touch, zoals geconcludeerd in sectie 5.5.1, wordt hier bevestigd. De verschillen tussen Sencha Touch en de andere raamwerken zijn groter bij de POC dan bij de loginapplicatie. Dit komt omdat bij de loginapplicatie een afhankelijkheid met het strikt minimale (`sencha-touch.js`) werd gebruikt. Het volledige raamwerk (`sencha-touch-all.js`) werd daarentegen bij de POC gebruikt. Het Delta Update mechanisme voor het updaten van de cache zal Sencha Touch nog meer vertragen bij de POC.

Indien Sencha Touch buiten beschouwing wordt gelaten, duurt de eerste keer laden van de POC gemiddeld $5,73\,\mathrm{s}$. Het laden van de POC uit cache duurt gemiddeld $400\,\mathrm{ms}$. De eerste keer laden van de loginapplicatie duurt gemiddeld $3,32\,\mathrm{s}$. Indien deze uit cache komt, duurt dit gemiddeld nog $420\,\mathrm{ms}$.

Indien enkel de downloadtijd in rekening wordt gebracht en Sencha Touch buiten beschouwing wordt gelaten, kan er gezien worden dat de downloadtijden $< 10\,\mathrm{s}$, dit zijn aanvaardbare tijden in het domein van gebruiksvriendelijkheid [85]. De gebruiker zal de vertraging waarnemen maar de aandacht niet verliezen. Het initieel laden van de POC implementatie met Sencha Touch moet van een laadscherm worden voorzien

omdat de downloadtijd $> 10\,\mathrm{s}$.

Google Page Speed De score op 100 die Google Page Speed [82] aan de loginapplicatie toekent is 96 voor Sencha Touch, gevolgd door Lungo (82), Kendo UI (73) en jQuery Mobile (71). Er kan geconcludeerd worden dat Sencha Cmd de applicatie optimaal weet te bouwen, er is het minst plaats voor verbetering. Hoewel Sencha Touch de meeste tijd vraagt om te laden, zal het hierna sneller werken. Dit wordt bevestigd in de voorbije testen.

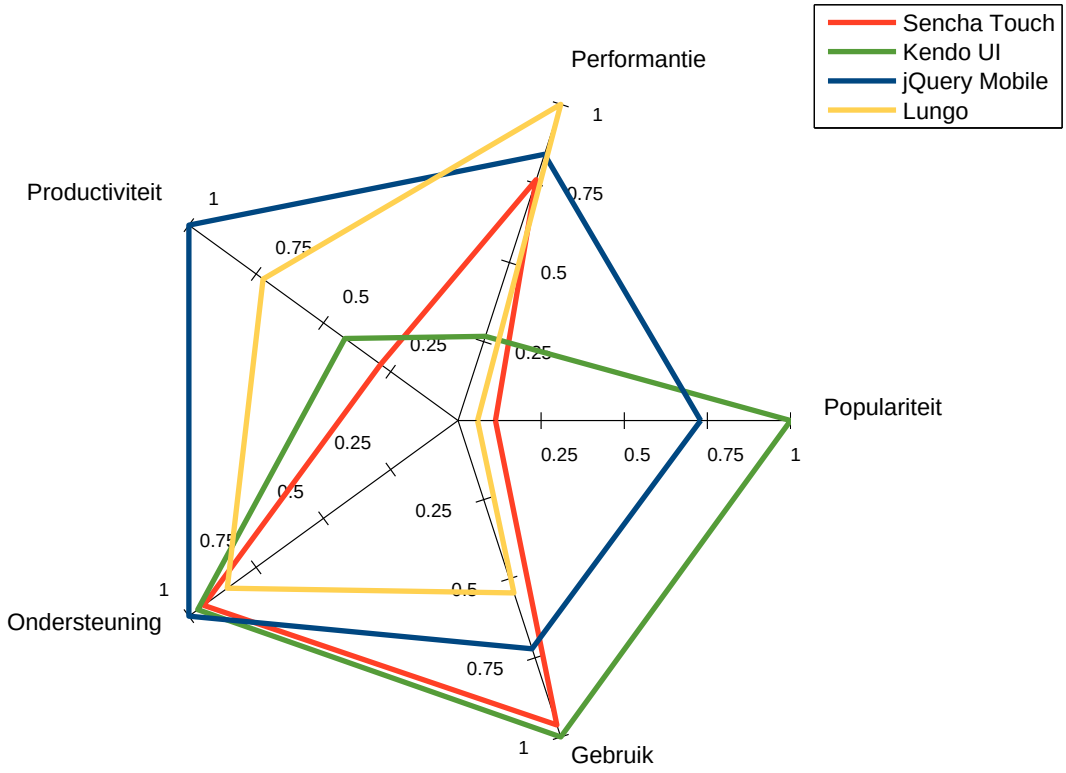
Alle implementaties worden door Google Page Speed aangeraden een tekenset te specificeren en gebruik te maken van het cachegeheugen van de browser. Dit laatste is een suggestie om de maximale duur van documenten in de cache een week in de toekomst te zetten. De huidige implementaties hebben een vervaldatum van slechts tien minuten. Beide werkpunten hebben volgens Google Page Speed een lage prioriteit. Alle raamwerken buiten Sencha Touch worden aangeraden om de JavaScript-code uitgesteld te parsen. Dit is inherent aan de raamwerken en niet aan de tool die de applicatie bouwt. Bij Kendo UI en jQuery Mobile heeft dit werkpunt een hogere prioriteit in tegenstelling tot Lungo. Dit komt omdat Lungo maar een kleiner aantal bytes moet verwerken. De implementatie van Sencha Touch wordt ook aangeraden om vraagtekens uit URLs te verwijderen. Dit zou de oorzaak kunnen zijn voor het niet-cachen van bestanden.

5.6 Overzicht

Figuur 5.4 toont een overzicht van de scores van de vier raamwerken op de vijf criteria in de vorm van een spinnenweb. De formules van de vergelijkingscriteria voor het plotten van het spinnenweb kunnen teruggevonden worden in sectie 4.3. Daar werden de gerelativeerde formules voorgesteld om een score tussen 0 en 1 te bekomen. Ook werden de formules voor productiviteit en performantie geïnverteerd omdat voor deze criteria geldt: hoe lager de score, hoe beter het raamwerk. In wat volgt zullen de raamwerken besproken worden volgens de totale score voor alle criteria. Hiervoor werd formule 4.15 gebruikt waar de scores van de criteria werden opgeteld.

jQuery Mobile heeft de beste score (86%) en is de winnaar. Een van de belangrijkste factoren die jQuery Mobile tot winnaar maakt is het feit dat er geen ontwerppatroon wordt afgedwongen. Dit maakt dat de leercurve veel lager ligt ten voordele van de productiviteit. Ook een goede documentatie en omkadering maakt jQuery Mobile productiever. De grotere populariteit in vergelijking met Lungo maakt jQuery Mobile aantrekkelijker als er geen ontwerppatroon noodzakelijk is. De afwezigheid van een ontwerppatroon brengt echter twee nadelen met zich mee. Ten eerst zal het raamwerk minder functionaliteit kunnen aanbieden. Door plug-ins en HTML5-kenmerken zal dit gebrek aan functionaliteit toch beperkt blijven. Een tweede nadeel is de code die moet geschreven worden. jQuery Mobile is opmaakgedreven waarbij veel HTML-code moet worden geschreven met grote verbositeit. Dit maakt het linken van HTML- met JavaScript-code gevoelig voor fouten.

Kendo UI bekleedt de tweede plaats met een score van 73%. In tegenstelling tot jQuery Mobile heeft het wel een ontwerppatroon en is het dus meer bruikbaar.



Figuur 5.4: Overzicht met de vijf vergelijkingscriteria.

De score van de productiviteit bevindt zich onder de helft maar in vergelijking met Sencha Touch scoort het beter. Het grote nadeel van Kendo UI, wat niet in het spinnenweb kan worden gezien, is de hoge kost voor een licentie. Het feit dat Kendo UI niet *open-source* is, reflecteert zich echter niet in de populariteit. Van alle vier raamwerken is Kendo UI het minst performant. De downloadtijden waren sneller in vergelijking met Sencha Touch maar de crashes op iOS-toestellen zorgden voor een lage gebruikerservaring. Het feit dat Kendo UI een *native look-and-feel* aanbiedt is een belangrijk kenmerk maar dit kan ook niet op het spinnenweb worden teruggevonden. De iOS-lay-out is sterk gelijkend op de echte iOS-lay-out. Dit was bij Android minder geslaagd.

Lungo heeft de derde score (64%). Op drie van de vijf criteria scoort Lungo het slechtst. Er is geen ontwerppatroon opgelegd en weer kan dit voor- en nadeel zowel in productiviteit en gebruik opgemerkt worden. De voordelen bij productiviteit zijn echter minder dan bij jQuery Mobile omdat de ondersteuning en documentatie minimaal is. De nadelen bij gebruik zijn dan weer uitvergroot omdat er niet zoveel functionaliteit en plug-ins als bij jQuery Mobile konden worden teruggevonden. Positief is dan weer dat Lungo de beste downloadtijden behaalde omdat QuoJS,

de JavaScript-bibliotheek van Lungo, geoptimaliseerd is voor mobiele apparaten. De testen op gebruikerservaring zwakte de performantie van Lungo af. Lungo is veruit het minst populair. Dit kan zowel met de literatuur als met sociale netwerken bevestigd worden.

Sencha Touch is volgens de gekozen criteria het slechtste raamwerk (61%). De combinatie van het MVC-ontwerppatroon en JavaScript-gedreven opmaak maken Sencha Touch zowel het minst productief als minst performant in vergelijking met de andere raamwerken. Alle HTML-code wordt door het raamwerk zelf aangemaakt waardoor de JavaScript-bestanden zeer groot zijn. De downloadtijden waren ook opmerkelijk trager door het Delta Update mechanisme. De gebruikerservaring gaf echter het tegenovergestelde resultaat want na de lange downloadtijd reageerde Sencha Touch het beste. De tools die Sencha aanbiedt om ontwikkelaars te helpen (Sencha Architect en Sencha Cmd) blijken niet voldoende om het verschil in productiviteit met andere raamwerken te verkleinen. Met de tools leren werken, is een leerproces op zich. Hoewel de ondersteuning van Sencha Touch hoog is, moet er opgemerkt worden dat het afhankelijk is van de WebKit *engine*. Toestellen met een standaard browser die deze *engine* niet bevatten, zoals Windows Phone, kunnen Sencha Touch niet gebruiken.

Hoofdstuk

6 Besluit

6.1 Conclusie

Deze thesistekst bestaat uit twee doelstellingen. Een eerste doel is het definiëren van een methodologie om HTML5-raamwerken met elkaar te vergelijken. Het tweede doel omvat de effectieve vergelijking van de raamwerken zelf.

De uitgelichte raamwerken zijn Sencha Touch, Kendo UI, jQuery Mobile en Lungo. Sencha Touch bouwt op het MVC-ontwerppatroon en is JavaScript-gedreven. Het raamwerk is gratis en heeft zowel een commerciële als *open-source* licentie. Kendo UI dwingt het MVVM-ontwerppatroon af en is zowel JavaScript- als opmaakgedreven. Een licentie voor het gebruik van Kendo UI kost $699. jQuery Mobile en Lungo hebben geen ontwerppatroon en zijn beide opmaakgedreven. Beide raamwerken zijn *open-source*.

Vijf criteria werden gekozen om de vergelijkende studie uit te voeren: populariteit, productiviteit, gebruik, ondersteuning en performantie. Elk criterium werd voorzien van een formule om een score te berekenen. In samenspraak met Capgemini werd een POC opgesteld die werknemers toelaat onkosten toe te voegen. Deze werd gebruikt om het gebruik en de ondersteuning te drijven. Om populariteit te meten werd naar de activiteit van de raamwerken op sociale netwerken gekeken. De tijd om een loginapplicatie te ontwikkelen, bepaalde de productiviteit. Deze applicatie is een subset van de POC dat is aangevuld met een lange lijst. De POC werd onderverdeeld in 13 uitdagingen met in totaal 38 deeluitdagingen om de functionaliteit van het raamwerk te testen en het gebruik te quoteren. Vervolgens werd een subset van de uitdagingen getest op acht verschillende mobiele toestellen om de ondersteuning te controleren. Er werd een evenwichtige keuze gemaakt tussen Android, iOS, smartphone en tablet. Ten slotte bepaalden de downloadtijd en de gebruikerservaring van de loginapplicatie de performantie. De gebruikerservaring bepaalt hoe vlot het gaat om door een lange lijst te scrollen. De scores van de vijf criteria voor de vier raamwerken werden in één spinnenweb ondergebracht.

Na evaluatie is jQuery Mobile het beste raamwerk op basis van de gekozen criteria, gevolgd door Kendo UI, Lungo en Sencha Touch. Deze volgorde werd bepaald door de scores van alle criteria op te tellen. jQuery Mobile heeft als belangrijkste troef de hoge productiviteit doordat het enerzijds goed gedocumenteerd is en anderzijds

geen ontwerppatroon afdwingt. Dit laatste is echter een nadeel waardoor het minder scoort op gebruik. Kendo UI heeft als belangrijkste troef het gebruik doordat het een ontwerppatroon afdwingt. Het scoort echter ondermaats op performantie door het crashen van lange lijsten op iOS. Lungo behaalt de snelste downloadtijd doordat QuoJS, de JavaScript-bibliotheek van Lungo, geoptimaliseerd is voor mobiele apparaten.Sencha Touch is het minst productief en minst performant in vergelijking met de andere raamwerken. Daarentegen scoort Sencha Touch het best op het vlak van gebruikerservaring. Door het afdwingen van een ontwerppatroon scoort het quasi evengoed als Kendo UI op vlak van gebruik. Alle onderzochte raamwerken scoren goed op ondersteuning op de onderzochte apparaten.

6.2 Geleerde lessen

Bij het uitvoeren van een vergelijkende studie is de keuze van de vergelijkingscriteria bepalend voor het resultaat van het onderzoek. Daarbij is het belangrijk om reeds bestaande literatuur grondig te controleren. Zo werd er in het begin te specifiek naar papers gezocht omtrent het vergelijken van HTML5-raamwerken. Meer algemene methodologieën om software te vergelijken moesten worden gezocht waaruit criteria konden worden hergebruikt.

Bij het opzetten van criteria is het van groot belang dat de manier waarop deze criteria zullen worden getoetst, zeer gedetailleerd worden neergeschreven. De exacte formules van de criteria werden pas tijdens de evaluatie vastgelegd. Pas dan werd de nood van een formele notatie duidelijk. Ook is het belangrijk om op voorhand de elementen van een vergelijkende studie uit te testen alvorens de vergelijkingscriteria vast te leggen. Hierdoor kunnen iteraties over de criteria vermeden worden als blijkt dat ze niet toepasbaar zijn. Een initiële vertrouwdheid met de raamwerken had er ook voor gezorgd dat de criteria meer kenmerken van de raamwerken zouden bevatten. De raamwerken die steunen op een ontwerppatroon konden op die manier meer bevoordeeld worden door bijvoorbeeld uitbreidbaarheid te introduceren, zoals dat ook in de ISO-25010 wordt gebruikt.

Een andere geleerde les is dat er in het uitvoeren van de evaluatie veel werk kruipt. Echter, het interpreteren en begrijpen van de resultaten duurde zowaar nog langer. Ook het opsporen en verbeteren van eigen fouten is zeer tijdrovend. Door op een consistente en gestructureerde methode de evaluatie te voltooien, moet het aantal fouten tot een minimum worden beperkt.

Er werden tot slot nog twee praktische zaken geleerd. Het opmeten van tijd en het gebruik van logboeken vereist een zekere vorm van discipline. Het eerste was noodzakelijk om de productiviteit van het raamwerk te toetsen. De tijdbudgetten die nodig waren moesten gemakkelijk kunnen worden gereconstrueerd. De logboeken werden gebruikt bij de implementatie van de POC en moesten bij de evaluatie het gebruikscriterium drijven. Een tweede praktisch punt gaat over de samenwerking tussen beide auteurs. Het is geen gemakkelijke opgave om als twee individuen continu op gelijke hoogte te zitten. Dit vraagt enorm veel onderlinge gesprekken met duidelijke communicatie. Discipline, sociale netwerken en technologieën zoals Google

Drive en GitHub hielpen de samenwerking te verbeteren.

6.3 Verder onderzoek

Er kan verder gezocht worden naar de oorzaak waarom bepaalde zaken zeer opmerkelijk waren of waarom ze niet lukten tijdens de vergelijking. Zo was er enerzijds de crash van de 850 lijstitems van Kendo UI op iOS. Hier kan gezocht worden naar de oorzaak van de crash, maar tevens kan ook gezocht worden naar de grens van het aantal lijstitems waarbij dat het wel lukt. Anderzijds was er een opmerkelijke waarneming bij de performantie van de applicaties uit cache. Zo ligt bijvoorbeeld de gemiddelde downloadtijd van de loginapplicatie uit cache hoger dan deze van de POC voor jQuery Mobile en Lungo.

Ook kunnen nieuwe raamwerken worden toegevoegd aan de vergelijking. Hierdoor vergroot ten eerste de grootte van de vergelijking, maar kan ten tweede ook de methode telkens opnieuw worden getoetst met deze nieuwe raamwerken. Daarnaast komen van de reeds vergeleken raamwerken geregeld nieuwe versies uit. Zo is het ook mogelijk om de evolutie in de rangschikking van de vier vergeleken raamwerken over de tijd te bekijken. Mogelijk kan de rangschikking veranderen bij het uitbrengen van nieuwe versies of plug-ins.

De huidige methode omvat vijf vergelijkingscriteria die worden gedreven door de POC. Verder onderzoek kan deze POC uitbreiden met extra kenmerken zoals Pull&Refresh. Dit loopt in de lijn om ook andere gebeurtenissen te gebruiken dan alleen maar de *tap* gebeurtenis. Andere gebeurtenissen zijn bijvoorbeeld *double tap*, *swipe*, *hold*, maar ook gebeurtenissen waar meerdere vingers voor nodig zijn zoals *rotate*. Daarnaast is ook de integratie van HTML5-kenmerken zoals GPS, *push events*, *drag-and-drop*, video en audio in de raamwerken zeker het onderzoeken waard.

Naast het toevoegen van extra kenmerken aan de POC, kunnen criteria ook op andere manieren gecontroleerd worden. Nu worden bij ondersteuning enkel apparaten met een Android- of iOS-besturingssysteem gebruikt. Dit kan worden vervangen of uitgebreid naar andere besturingssystemen zoals Windows Phone en BlackBerry OS. Een andere voorbeeld is dat voor de downloadtijd bij performantie WiFi werd gebruikt voor de verbinding. Andere verbindingsmogelijkheden zoals 3G kunnen worden gebruikt en hierdoor kunnen andere resultaten bekomen worden. Een derde voorbeeld zijn de rendertijden die niet konden worden gebruikt bij performantie. Deze konden niet worden opgemeten bij Sencha Touch en Lungo. Aanpassingen aan de broncode zouden het opmeten van rendertijden wel kunnen toelaten. Daarnaast kan het gebruikte alternatief dat bepaalt hoe vlot het gaat om door een lange lijst te scrollen, ook verbeterd worden. Als laatste kunnen ook de lijnen effectief geschreven code worden gebruikt bij productiviteit.

Een andere onderzoekspiste is om nieuwe criteria toe te voegen. Zo kan bijvoorbeeld het criterium uitbreidbaarheid worden onderzocht. Dit criterium omvat hoe gemakkelijk het gaat om de bestaande applicatie geïmplementeerd in een bepaald raamwerk uit te breiden. Een te onderzoeken hypothese hierbij is dat raamwerken die een ontwerppatroon afdwingen beter zullen scoren dan raamwerken zonder ontwerp-

patroon. Een bijkomende hypothese is dat de totale score van raamwerken die een ontwerppatroon afdwingen zal stijgen en deze van raamwerken zonder ontwerppatroon zal dalen. Dit komt doordat de ene worden afgestraft op productiviteit en de andere op uitbreidbaarheid. Mogelijk kan de rangschikking van de vier onderzochte raamwerken veranderen. Anderzijds kan ook een criterium worden toegevoegd die kijkt naar het finale resultaat van het raamwerk. Zo kunnen bepaalde raamwerken de *native look-and-feel* van mobiele besturingssystemen nabootsen, andere raamwerken bieden dan weer standaard een frisse hedendaagse lay-out.

Andere onderzoeksvragen kunnen een stap terugnemen door bijvoorbeeld af te vragen of het batterijverbruik door webapplicaties een probleem vormt. De bekomen data kan worden vergeleken met *native* en hybride applicaties. Deze laatste vergelijking kan zelfs veralgemeend worden waardoor een vergelijking tussen web-, *native* en hybride applicaties zich opdringt.

Bijlagen

Bijlage

A Poster

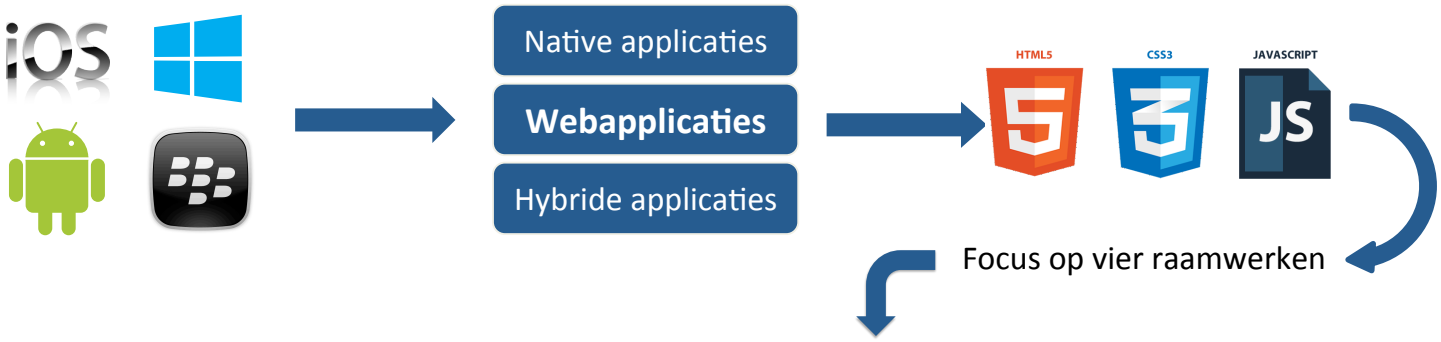
KU LEUVEN

Tim Ameye en Sander Van Loock

Promotor: Prof. dr. ir. E. Duval – Begeleiders: Ir. G. Parra, J. Verhulst en J. Persoons – Academiejaar: 2012-2013

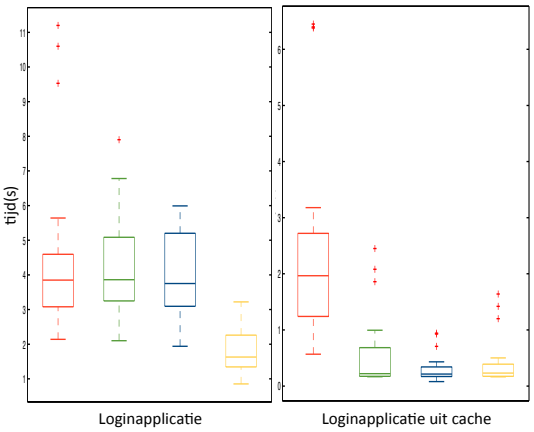
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Vergelijkende studie van raamwerken voor de ontwikkeling van mobiele HTML5-applicaties

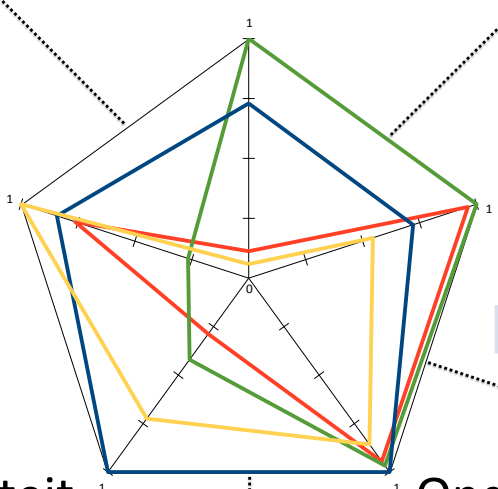


	Sencha	Kendo UI	jQuery mobile	Ionic
Type	JavaScript-gedreven	JavaScript- en opmaakgedreven	opmaakgedreven	opmaakgedreven
Ontwerppatroon	MVC	MVVM	geen	geen
Licentie	GPLv3, commercieel	$ 699	MIT	GPLv3, commercieel

Performantie



Populariteit



Gebruik

Score	58/76	**60**/76	43/76	33/76

Productiviteit

Werkuren				
Proof of concept	159	68*	117	**30***
Loginapplicatie	7	5	**2**	3

*tweede implementatie

Geschreven lijnen code

	Proof of concept				Loginapplicatie			
HTML	0	302	653	464	0	54	48	74
JS	1927	762	1174	640	177	51	62	48
CSS	14	62	300	58	6	32	7	3

Ondersteuning

Score*	83/96	92/104	**95**/104	62/80

*uitdagingen getest die geïmplementeerd konden worden

Bijlage

B Wetenschappelijk artikel

Comparative study of frameworks for the development of mobile HTML5 applications

Tim Ameye
tim.ameye@student.kuleuven.be
Departement Computerwetenschappen
KU Leuven

Sander Van Loock
sander.vanloock1@student.kuleuven.be
Departement Computerwetenschappen
KU Leuven

Abstract

The need to make mobile applications that run on every device is ever-growing. One solution is to build mobile web applications using HTML5. To speed up development, frameworks are presented. These help to add functionality and elements for the user interface. Because of the variety of existing frameworks, a thorough comparative study is needed to know which framework suits the job best. This paper compares Sencha Touch, jQuery Mobile, Kendo UI and Lungo by using popularity, productivity, usage, support and performance as comparison criteria. The conclusion is that jQuery Mobile is the best framework with great productivity. Kendo UI gets the second place and its main asset is usage. Lungo is third and has the fastest download performance. Sencha Touch ends last but gets the best user experience. Furthermore all four frameworks have good device support.

1 Introduction

Since the release of the Apple iPhone in 2007 [8], the demand for smartphones is ever since increasing. Today, over one billion smartphones are in use globally [58]. This will double by 2015 [16]. Also tablets conquer market share as mobile devices. 760 million tablets will be in use globally by 2016 [16]. Besides Apple, other companies like Google and Microsoft are on the track too. All those mobile devices come in different shapes and sizes. They also include different features like GPS, camera and connectivity like Bluetooth, WiFi and 3G.

On these mobile devices run mobile operating systems (OS). The two mobile OSs with the greatest market share are iOS and Android with 14.9% and 75.0% respectively in Q3 of 2012 [35]. These mobile OSs run both on smartphones and tablets. Furthermore, they differ in graphical user interface (GUI) and the way users interact with the OS. This is called native look and feel.

The mobile OS from Apple, called iOS, first appeared in 2007. Since then, a new version was released every year with the most recent version being iOS 6.1 which was released in January 2013 [9; 34; 3]. The mobile browser from Apple shipped with iOS is called Mobile Safari. Apple did make sure to catch up with the latest HTML5 specifications in its browser [17]. They also dropped support for Adobe Flash on their mobile devices [21].

iOS	Share (%)
6.1.x	81.2
5.1.x	8.2
6.0.x	8.0
5.0.x	1.2
4.3.x	1.4

Android	Share (%)
2.3	38.5
4.0.x	27.5
4.1.x	26.1
2.2	3.7
4.2.x	2.3
2.1	1.7
3.2	0.1
1.6	0.1

Table 1: Usage of iOS versions on May 8, 2013 and Android versions on May 1, 2013 [47; 2].

In 2005, Google bought Android Inc. and released its first stable mobile OS, called Android, in 2008 [40]. Like with iOS, new versions were released on a yearly basis. Android 4.2 is the latest version and was released in October 2012 [41]. Android comes with the Android browser. The implementation of HTML5 specifications has dragged, but as of Android 4.0, support is increasing [17]. It is now also possible to install the mobile version of the Chrome desktop browser on devices with Android 4.0 and later.

It is important to look at the usage of the different OS versions, which is shown in table 1. This is because not every user has (updated to) the latest version on his mobile device. The table shows that almost 90% of iOS users already has version iOS 6.x. In contrast, about a half of the Android users is still using Android 2.x while the latest major version is Android 4.x. Those users cannot benefit of the latest bells and whistles, particularly the new features of HMTL5, on their devices.

There are three possible ways to make a mobile application [1; 17]. The first one is a *web application* using HTML5 which runs entirely in the browser. The second type of application is a *native application* which is installed on the device and programmed with a Software Development Kit (SDK) specific for the mobile OS.

Lastly, a mix of both can be made which is called a *hybrid application*. One approach is to pack the web application as a native application. The other approach is to use one pro-

gramming language to build native applications for different OSs.

Advantages of web applications are the presence of a browser on mobile devices, so everyone can use it directly without installing the app first and regardless of their mobile OS [1]. Secondly, the code needs only to be written once and can be run anywhere (WORA) [17]. This is in contrast to native, were a codebase has to be maintained for every mobile OS. Lastly, web applications do not need to be published to a store (e.g. App Store for iOS and Google Play for Android) prior to release. Such a store is a place where users can download applications to their mobile device. A disadvantage of web applications is that they are dependent on the mobile OS just like native applications. This is especially the case when using new features of HTML5 in web applications.

In contrast [1], native applications provide better performance. Secondly, it is easier to use features like GPS and camera of the mobile device with a native application. Thirdly there is a security issue were web applications lack behind. Native applications can do CPU intensive custom encryption and use more advanced authentication techniques like face recognition. Lastly, publishing applications through a store has two advantages. On the one hand, it makes distributing the application easier. On the other hand, stores can also supervise the applications, which is not the case for web applications.

The advantage of hybrid applications is that they patch the issues of web applications with the benefits of native applications. However, as of December 2012, HTML5 went into candidate recommendation [18]. This implies the upcoming support of HTML5 in contemporary browsers and that hybrid applications will probably become obsolete over time.

2 Related Work

In the literature, many frameworks are presented, but not often compared. If they do get compared, none of them are scientifically published or use a proof of concept (POC) to validate their comparison. The idea of using a POC is not new. Oehlman [29] and Kosmaczewski [26], for example, use a small geosocial game and a todo list application respectively to present mobile HTML5 frameworks.

Some blog posts [39; 4; 37] all have their own criteria and methodologies to assess different mobile frameworks. The overall application of the criteria changes from case to case. Rozynski [37] presents the chosen criteria and discusses them for each framework. Ayuso [4] presents 17 criteria but all of them are discussed at once per framework. Thereafter, advantages and disadvantages are proposed to the reader. Finally, Sarrafi [39] presents their chosen criteria together with a scorecard and an explanation of scores per criterion. Each framework gets evaluated based on the scores for each criterion. All these blog posts compare mobile and hybrid frameworks. Some websites [5; 7] only focus on two mobile HTML5 frameworks in contrast to the Mobile Frameworks Comparison Chart from Falk [12] that tries to compare as much as frameworks as possible in a large tabular form.

The report about Cross-Platform Developer Tools from Vision Mobile [27] compares 15 cross-platform tools. The evaluation is based on a questionnaire with more than 2400 developers worldwide. Their main focus lies on hybrid applications because they state that hybrid applications try to combine the best of both web and native applications.

The ISO 25010 standard evaluates the product quality and quality in use of software [49]. The assessment of the categories is done by a checklist. Jadhav et al. [19] discusses evaluation techniques to select software. A widely used technique is the use of the analytic hierarchy process (AHP) developed by Saaty [38]. It uses a hierarchy where the top layer represents the goal, the intermediate layer represents the criteria and the bottom layer represents the candidates. Priorities are given to the criteria and candidates are compared pairwise.

3 Comparison criteria

The comparison study of frameworks for mobile HTML5 applications will be driven by a POC, like Oehlman et al. [29]. This POC encompasses an application that allows employees to attach, sign and submit expenses to their employer. Afterwards, the employee can view the uploaded expenses and download them as PDF.

Five important criteria will be used: popularity, productivity, usage, support and performance. The POC will be used for usage and support. A login application, that is a small subset of the POC, will be used for productivity and performance.

The ISO 25010 standard is represented in all criteria but popularity. AHP was not used because of two reasons. Firstly when new criteria or candidates are added, a re-evaluation of all the candidates is needed [19]. Our method does not suffer from this problem. Secondly, AHP uses pairwise comparison. Our method lets researchers evaluate frameworks without taking the other frameworks into account.

The proposed methodology that results in a score for each criterion will be described below. Per criterion, a table or graph will be provided to give an overview of the total scores. At the final stage, these scores will be plotted in a spider graph. This representation is similar to a blogpost that compared native and web applications [20].

Popularity Social networks provide a good indication to determine the popularity that goes with the framework. It is also an indication how much the framework will be developed in the future [39]. Twitter followers, GitHub stars, GitHub forkers, Stack Overflow questions and Facebook likes will be recorded. Hales [17] used Twitter and GitHub and Monocaffe [4] used Stack Overflow. The summation of these five numbers give the score for the framework's popularity.

Productivity Productivity reflects how long it takes to get used to a framework and to implement something useful. This is because companies do not want to spend a lot of time in using new technologies. Furthermore, the ISO 25010 standard uses usability to the check the efficiency of reaching goals with software.

Both authors implement the login application and carefully record the time to implement it. The login application contains the POC's login and loads a long list afterwards. The time to implement the login application gives the score for the framework's productivity.

Score	Assessment criteria
$C_{f,i} = 2$	Supported by the framework
$C_{f,i} = 1$	A plugin or HTML5 feature is needed
$C_{f,i} = 0$	No, custom or hacky implementation

Table 2: Assessment criteria for implementation of challenges.

Both authors also implement the full POC in two different frameworks. However, this time was influenced by many factors. The second implementations will have an advantage because of the gained experience after the first implementation. Also, code reuse and possible incomplete implementations because of the limited functionality of the framework, affect this time in a negative way. The implementation of the login application does not suffer from these factors.

Usage Usage reflects which features are supported by the framework. The more features a framework has, the less code the developer needs to write himself. Furthermore, the ISO 25010 standard also tries the same with functional suitability.

The POC will be subdivided in 13 challenges with a total of 38 sub challenges. The score of this criterion will depend on the completion of these challenges by the framework. The assessment of the challenges can be found in table 2. The total of all challenges gives the score for the usage of framework f:

$$\text{Usage}_f = \sum_{i=1}^{38} (C_{f,i}) \tag{1}$$

Support This criterion reflects the support on different devices and mobile OSs. It is important that the framework has the widest possible support. Furthermore, the ISO 25010 standard describes portability to different platforms.

A context is defined as a combination of a particular mobile OS, device and browser. Both Android and iOS, smartphones and tablets will be checked. The used browser will be the native browser of the mobile OS (see section 1). In total, eight contexts will be checked for support. In each context, a subset of the challenges from the previous criterion will be checked. Only the challenges where the framework actually provided an implementation (N_f) are checked for support. If the challenge works as expected then $C_{f,c,i} = 1$. If some part of the challenge is infeasible then $C_{f,c,i} = 0$. The summation of scores for each context will give the score for the framework's support.

$$\text{Support}_f = \sum_{c=1}^{8} \left(\sum_{i=1}^{N_f} C_{f,c,i} \right) \tag{2}$$

Performance The login application will be used as a stress test for the POC. It is important that a web application is both fast in downloading as it is smooth in user experience. Furthermore, the ISO 25010 uses the former in performance efficiency to asses the speed and used resources.

On the one hand, the average download time of both the non-cached ($l_{\widehat{f,c,login}}$) and cached version ($l_{\widehat{f,c,login_{cache}}}$) of the login application will be taken into account. The average download time for a framework f is:

$$\text{Average download time}_f = \frac{\sum_{c=1}^{8} \left(\widehat{l}_{f,c,login} + \widehat{l}_{f,c,login_{cache}} \right)}{8}$$

On the other hand, the user experience by scrolling through a long list of 850 items will be tested. The user experience will be measured on each context with two user tests (User experience_f). The result will be a total order of the frameworks per context that indicate the smoothness. The performance for a framework f is defined by:

$$\text{Performance}_f = \frac{\text{Average download time}_f}{\text{User experience}_f} \tag{3}$$

Initially, render times would be measured by the time to render the 850 list items. This would be added to the download time to get the total performance. This was not possible because ST and Lungo do not have an event after the elements are rendered.

4 Frameworks

This section introduces the compared mobile HTML5 frameworks: Sencha Touch, Kendo UI, jQuery Mobile and Lungo. Frameworks can be categorised in two approaches: markup driven and JavaScript driven [29].

There are three ways of writing a HTML5 application for markup driven frameworks [6]. The first one is to write the full application on one single web page. The advantage is that there are initially less requests to the server. The second option is to write a web page for each screen. The advantage here is that the first viewed screen is downloaded more quickly. However, with each transition, the next screen has to be fetched which can delay navigation. Lastly, one can mix the two above to find an optimum by putting the most likely viewed screens on one web page and the less likely viewed on separated pages.

4.1 Sencha Touch

Sencha Touch (ST) is a framework developed by Sencha, a company founded in 2010 as a composition of Ext JS, jQuery Touch and Raphaël. The first stable version of ST was officially released in November 2010. Ext JS is a JavaScript framework for the development of desktop applications. jQuery Touch is a jQuery plugin for mobile development that adds touch events to jQuery and depends on the WebKit engine. Finally, Raphaël is a JavaScript library for vector drawings. Parts of the first two technologies can be found in the implementation of the ST framework. As at the time of writing, ST is at version 2.2 [43].

Documentation The documentation describes all objects and features, some with code examples and live previews [44]. The key concepts of ST are explained in tutorials: some are texts, some are videos.

A useful tool to discover the ST features is the Kitchen Sink [46]. This is a HTML5 application that lines up all possibilities of the framework combined with the corresponding code.

License ST is free within a commercial context in which the developer does not share the code with its users. There is also the option to use an open-source version. This comes with a GNU GPLv3 license which implies a free code redistribution as most important property. Other licenses can be found at the Sencha website [45].

Code and development ST is written on top of Ext JS and is a JavaScript driven framework. All code needs to be written in JavaScript and loaded by one HTML container. Another important aspect of ST is that it imposes the Model-View-Controller (MVC) architecture. Models group fields to data objects, views define how the content is presented to the user and controllers connect both based on events.

ST contains all UI elements as JavaScript objects. Just like object-oriented programming, those objects are part of a class system. Classes can be defined and/or created. Single inheritance and overriding is also possible. To enhance performance, it is the programmer's task to create components before they are used. In this way, the programmer partially determines the performance of the application.

Browser support Just like jQuery Touch, ST is based upon the WebKit browser engine. This forms the major requirement for browser support. Although most mobile browsers contain this engine, some do not like Internet Explorer Mobile.

The framework offers the programmer methods to ask for the current context. Also, it is possible to query specific features, comparable with Modernizr [28].

4.2 Kendo UI

Kendo UI (KUI) is a framework from Telerik, launched in November 2011 and consists of three layers: Web, Mobile and DataViz. The first focusses on the development of web applications, the second makes them mobile and the last facilitates data representation. KUI is a hybrid form of JavaScript and markup driven frameworks. It is build upon the (Model-View-View Model) MVVM architecture and requires the jQuery library. At the time of writing, KUI is at version 2013 Q2 [56].

Documentation Two important sections from the documentation are the API and Getting Started [55]. Both sections follow the same structure and each feature from the API is explained in detail in the Getting Started section. Also, every feature can be viewed in a demo with the corresponding code.

License A license for KUI Complete costs $699 per developer. Each layer can be bought separately for a different license fee. KUI also provides server-side wrappers that can automatically create client-side JavaScript code. At the time of writing, PHP, JSP and ASP.NET MVC are supported as server-side technologies. A license for a server-side wrapper costs $300 [56].

Code and development Both JavaScript and HTML code needs to be written because Kendo UI is both JavaScript and markup driven. Kendo UI is dependent on the jQuery library. The MVVM architecture is imposed by KUI. The Views and Models are similar like the Views and Models of the MVC architecture from ST. The view model is an object that binds views with models in both directions and is called `ObservableObject` in KUI. The binding is annotated in HTML with data attributes. KUI supports bindings based on different properties from JavaScript objects. Examples of properties are `checked`, `value` or `visible`. If a view is bound to a model, the model is changed when a user changes the content of a view or the view is changed when the model is programmatically updated.

Browser support One of the most important aspects of KUI is the mimicking of the native look and feel of the current operating system. Supported platforms are iOS, Android, BlackBerry OS and Windows Phone 8.

All widgets from Kendo UI Web support progressive enhancement. Also, the features of Kendo UI Mobile that depend on HTML5 support progressive enhancement. The styling of forms, for example, will still work on older platforms, but the functionality will be limited to text input only.

4.3 jQuery Mobile

jQuery Mobile (jQM) is a mobile HTML5 UI framework that was announced in 2010 [36]. In November 2011 version 1.0 was released [31] and one year later in October, version 1.2 was released [32]. As at the time of writing, jQM released version 1.3.1 [33]. The framework is controlled by the jQuery Project that also manages jQuery. The latter is a JavaScript library where jQM is depending on [24]. jQM is among other things sponsored by Adobe, Nokia, BlackBerry and Mozilla [22].

License As of September 2012 it is only possible to use jQM under the Massachusetts Institute of Technology (MIT) license [11]. This means that the code is released as open source and can also be used in proprietary projects [34].

Documentation The documentation site of version 1.2 [23] is a catalog of all possible elements that jQM offers. It contains an overview of all possible UI components. By checking the source code, one can find out what code to write to get the same result. Furthermore it explains the API on how to configure settings, use events, methods, utilities, data attributes and theme the framework [23].

Code and development jQM is a markup driven UI framework and thus provides mainly UI components. jQM provides six categories of components: pages and dialogs, toolbars, buttons, content formatting, form elements and listviews [23]. One can obtain these components by writing HTML5 with jQM specific data attributes. When running the application, jQM will add the extra necessary code to correctly show these components by doing progressive enhancement. Switching pages in a multi-page application is done with AJAX by default.

Browser support jQM divides browsers into three grades: A, B and C. In an A-graded browser, the application is fully enhanced with AJAX-based animated page transitions. In a B-graded browser, the application has an enhanced experience, but there are no AJAX navigation features. Lastly in a C-graded browser, the application has a basic, non-enhanced

Popularity	ST	KUI	jQM	L
Twitter followers	1197	9532	12952	1665
GitHub stars			8184	1361
GitHub forkers			1886	316
Stack Overflow questions	5621	2257	20630	19
Facebook likes	748	54224	2760	28
Total	7566	66013	46412	3389

Table 3: Overview of popularity on May 8, 2013.

HTML experience, but is still functional [25].

4.4 Lungo

Lungo (L) is a markup driven framework of which version 1.0 appeared in 2011 [50]. The framework is maintained by the Spanish company TapQuo that is specialised in mobile user experience [54]. Lungo depends on QuoJS which is a JavaScript library optimised for mobile. As of the time of writing, Lungo is at version 2.1 [51].

Documentation The documentation site [53] first shows how a typical Lungo application looks like. There are eight other pages explaining briefly the various UI components and API functionality. On these pages the source code is shown for each example. One can also view a live preview of each example.

License The framework is released by the GNU GPLv3 license. A commercial license is also available [52].

Code and development Programming a Lungo application is done by annotating the HTML code with CSS classes and data attributes. There is no architecture like MVC enforced by the framework. To manipulate the DOM, the developer has to use QuoJS. This JavaScript library that is optimised for mobile does not contain methods for desktop users therefore making it smaller than traditional JavaScript libraries like jQuery. The library does not enforce an architecture.

One can write a single page web application or create multiple pages for each screen. The `article` and `section` HTML5 tags are used to separate the different screens of the application. Lungo supports a multipage web application by providing an asynchronous loader when initializing the application. Only the code between the `body` tags of the screens need to be saved on different web pages.

Browser support The framework indicates on their website to have support for iOS, Android, Blackberry OS and FirefoxOS.

5 Evaluation

This section evaluates the four mobile HTML5 frameworks by five criteria, namely popularity (5.1), productivity (5.2), usage (5.3), support (5.4) and performance (5.5).

5.1 Popularity

The scores for popularity can be found in table 3. KUI takes the first place due to the large amount of Facebook likes. jQM and ST take respectively second and third place despite the fact that these two frameworks are the most popular in literature [8; 13; 17; 29]. The last place goes to Lungo with a remarkable low popularity on Stack Overflow and Facebook. When looking at the total, there are two groups of frameworks. On the one hand there is KUI together with jQM and on the other hand there is ST together with Lungo.

jQM has the most followers on Twitter, followed by KUI. Lungo is the penultimate, but the ratio tweets to followers indicate that it is the most active one. jQM and KUI have a similar ratio. This is in contrast to ST which has send only one tweet and also has the least number of followers. This tweet references to the Twitter account of Sencha that has about 3,000 tweets and 20,000 followers.

Only jQM and Lungo are on GitHub. If the GitHub stars and GitHub forkers are left out of popularity, the ranking stays the same.

KUI refers in the support menu on their website directly to Stack Overflow. However the popularity of KUI is lower than jQM. ST is the penultimate, but more surprisingly is that Lungo only has about 30 questions on Stack Overflow and therefore takes the last place.

KUI and jQM both have created a fan page on Facebook in respectively November 2011 and August 2010. The fan page of KUI thus has gained more Facebook likes in a shorter period of time than the earlier created fan page of jQM. An explanation is that the different layers of Kendo UI are aggregated on one fan page. ST and Lungo only have a interest page on Facebook. This explains the great difference in Facebook likes. If the likes are left out of popularity, KUI and jQM switch places.

Productivity	ST	KUI	jQM	L
Login application (h)	7.3	5.0	2.1	2.9

Table 4: Overview of productivity.

5.2 Productivity

This section will investigate the productivity of the different frameworks. Table 4 contains the hours to implement the login application for each framework. This determines the score for this criterion. The data shows that jQM is the winner, followed by Lungo and KUI. ST is the least productive.

The imposed architecture of ST and KUI increases the learning curve. Also, the fact that a framework is JavaScript driven complicates the implementation. This is why ST is the least productive.

There are also other factors that influence the productivity. Firstly, tools can speed up the production process. Sencha provides Sencha Architect, a graphical environment to visually build applications. Version 2.1 of Sencha Architect was used to build the views but was not of much use to build more sophisticated functionality. Secondly, boilerplate code should be used to initialize a new project. All frameworks besides Lungo describe such code in their documentation. ST even offers Sencha Cmd to automate this process. Also, the quality and quantity of the documentation influence the productivity. A well-engineered search functionality, real-time code

examples and a structured classification is crucial for an optimal documentation. No framework has a documentation that excels in all three components. An other determinant factor is the available literature. Safari Books Online contains 13 books that specifically discuss jQM, Lungo has no books at all [30]. The final factor contains the virtual places where a developer can ask questions. Stack Overflow or fora are examples of such places. As presented in the previous criterion, jQM excels in the amount of Stack Overflow questions. ST and KUI provide professional support on fora from their website. Although no questions can be asked without a license, frequently asked questions are already discussed and explained on the fora.

5.3 Usage

In this section the usage is evaluated by evaluating 13 challenges. The total score per challenge can be found in table 5.

The best framework in usage is KUI, followed closely by ST. This can be explained because they both enforce an architecture, MVVM and MVC respectively. The lack of enforcing an architecture by the two other frameworks result in a laborious approach, where points are lost due to the lack of features by the framework. C8: Form validation and C5: Automatic form filling are an example of this. Remarkable is the maximum score of KUI for C4: Forms and the lower score for C13: Offline in comparison with ST.

jQM has a bit more than half of the maximum score. It gets zero points on C5: Automatic form filling and C11: Lists. This is because of the lack of enforcing an architecture, the developer has to do those things manually.

The last place goes to Lungo that fails the usage criterion. The same problems of jQM are also applicable for Lungo. Furthermore Lungo gets a zero for C8: Form validation and also has problems with the more advanced form elements in C4: Forms. One has to have the luck of finding a plugin or implement the functionality by hand.

It can be generally stated that each of the four frameworks have great support for C3: Load screen and dialog. C10: AJAX: text, JSON and XML is also fully supported by each framework, except for one particular case with Lungo. QuoJS did not correctly process a request with JSON as payload. To fetch the PDF however in C12: Show PDF, most frameworks have trouble if this is done via an AJAX request because AJAX is not the preferred way for fetching raw data. Hacky implementations like submitting a hidden form in jQM, KUI and Lungo solve the problem. In ST, a plugin was used but it issued a GET request to retrieve the PDF. A parameterized POST request was required instead.

Challenge	Max	ST	KUI	jQM	L
C1: Anatomy of page	6	6	2	4	2
C2: Device-specific layout	6	4	6	3	3
C3: Load screen and dialog	4	4	4	4	4
C4: Forms	14	8	14	9	6
C5: Automatic form filling	4	4	4	0	0
C6: Autocomplete	4	1	4	4	2
C7: Attach and process image	6	4	4	4	4
C8: Form validation	8	6	6	3	0
C9: Signature	2	1	1	1	0
C10: AJAX: text, JSON and XML	8	8	8	8	6
C11: Lists	6	6	4	0	2
C12: Show PDF	4	2	1	1	1
C13: Offline	4	4	2	2	3
Total	76	58	60	43	33

Table 5: Overview of usage.

The following five challenges are not always supported by each framework. Firstly, if C8: Form validation is supported, the frameworks or plugins do not have support to make a red border around the invalid fields. Secondly, C6: Autocomplete is out-of-the-box provided by KUI and jQM, plugins had to be used for Lungo and ST. The latter did not work in cooperation with the backend. Thirdly, only ST fully provides C13: Offline. In the other frameworks one has to work directly with HTML5 `localStorage` and create the `manifest` file for HTML5 Application Cache. Fourthly, both ST and KUI have support for C2: Device-specific layout. jQM and Lungo are using CSS3 Media Queries to accomplish this. Lastly, frameworks do not always support advanced form elements for C4: Forms. Only KUI accomplishes both date and month pickers with a specified range. Lungo lacks support for an option field.

No framework has support for C9: Signature. A plugin was found for every framework except for Lungo. C7: Attach and process image also suffers from the same problem, especially when the image has to be converted to Base64. In ST a plugin could be used, but with the other three frameworks, this has to be programmed manually using the HTML5 FileReaderAPI and `canvas`.

5.4 Support

This section will evaluate the support of the frameworks on eight mobile devices. The scores for the four frameworks are summarized in table 6. As mentioned in section 3, only the functionality of the framework is tested so custom or hacky implementations are not checked for support. This is the reason why the maximum score for each framework is different. The challenges that could be implemented using the framework are described in the previous criterion. The maximum scores for ST, KUI, jQM and Lungo are respectively 96, 104, 104 and 80. Their result for support is respectively 83, 92, 95 and 62. Looking at the relative scores, ST supports 86%, KUI 88%, jQM 91% and Lungo 78%. A first conclusion is that the first tree frameworks have a remarkable high score. The reason why the scores are more or less identical is that the frameworks all depend on the same underlying technology, HTML5. The major problems for Lungo can be found in C4: Forms. The navigation through the POC did not work on Android 2.3 devices. The tap event was used to implement the navigation, but did not fire when tapping. On the other devices, the tap event resulted in a double tap. Therefor each tap also resulted in a tap on the next screen which can result in an undesirable action.

In general, Android 2.3 devices had the least support. All other devices scored an average of 95%. The differences in operating system were remarkable: Android has 79% support while iOS has 95%. Only version 2.3 and 4 of Android and version 5 and 6 of iOS were tested. The two oldest devices were the iPhone 3GS and GalaxyS respectively released

Challenge	ST		KUI		jQM		L	
	Score	Max	Score	Max	Score	Max	Score	Max
C2: Device-specific layout	7	8	8	8	7	8	7	8
C4: Forms	33	40	34	40	38	40	22	32
C6: Autocomplete			8	8	8	8	7	8
C7: Attach and process image	5	8	5	8	5	8	5	8
C8: Form validation	8	8	8	8	8	8		
C9: Signature	8	8	8	8	8	8		
C12: Show PDF	6	8	5	8	5	8	5	8
C13: Offline	16	16	16	16	16	16	16	16
Total	83	96	92	104	95	104	62	80

Table 6: Overview of support.

Performance	ST	KUI	jQM	L
Average download time	4.48	4.23	3.95	1.79
Average download time from cache	2.34	0.55	0.30	0.40
User experience	32	8	22	14
Score	0.21	0.60	0.19	0.16

Table 7: Overview of performance.

in June 2009 and March 2010 [48; 15]. Although both devices are upgraded to a later operating system, Android devices offer updates less frequently. This can be explained with the large heterogeneity between Android devices. Hence, the newest HTML5 features will be adopted more quickly by iOS devices.

Only ST and KUI provide methods to query the context. jQM and Lungo implemented this feature with CSS3 Media Queries. This implementations worked on all devices except for the GalaxyTab. However, KUI was able to recognize this device as a tablet and achieved the maximum score. Regarding forms, each framework had some specific issues. The labels from ST were abbreviated with ellipses on low resolution devices. Read-only field were white in the KUI implementation. The only problem with jQM was the absence of the virtual email keyboard for email fields, but this applied to all framework. Lungo lagged on Android 2.3 devices and the switch did not work at all. Uploading an image requires support for both the `localStorage` and FileReaderAPI. Also the `file` input type is necessary to indicate the upload component. The four implementations of this challenge all fail for on iOS 5 and Android 2.3 because the lack of support for one or more of these dependencies [10; 57]. HTML5 provides features for form validation but none of the eight devices support these features. However, the implementation of form validation in all frameworks but Lungo do work. All four implementations for the signature depend on the `canvas` feature of HTML5. This is supported on all devices [10]. KUI, jQM and Lungo reuse the code to implement C12: Show PDF. ST has a plugin available that depend on the PDF.JS, a PDF renderer from Mozilla [14]. The first implementation failed to work on the GalaxyS while latter implementation showed the PDF successfully. Finally, offline capabilities and HTML5 Application Cache are supported on all eight devices [10] so each framework achieved the maximum score.

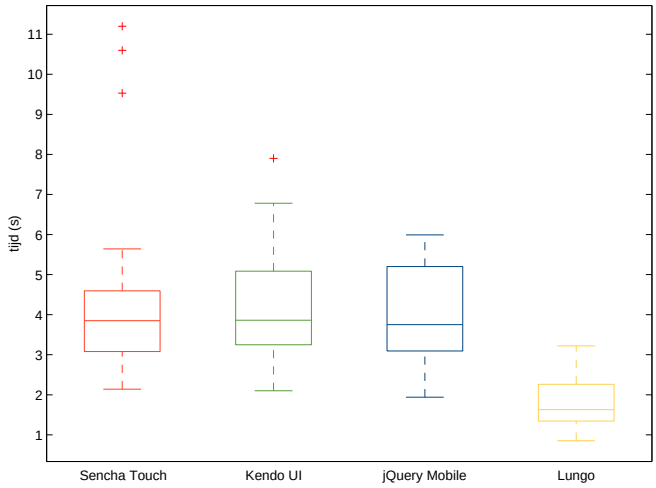
5.5 Performance

The result of both the average download time and user experience of each framework can be found in table 7. Lungo takes first place, shortly followed by jQM and ST. KUI takes the last place. This is because Lungo and jQM do not enforce an architecture, which means that less JavaScript code has to be downloaded. ST has the worst download time, but has the best user experience.

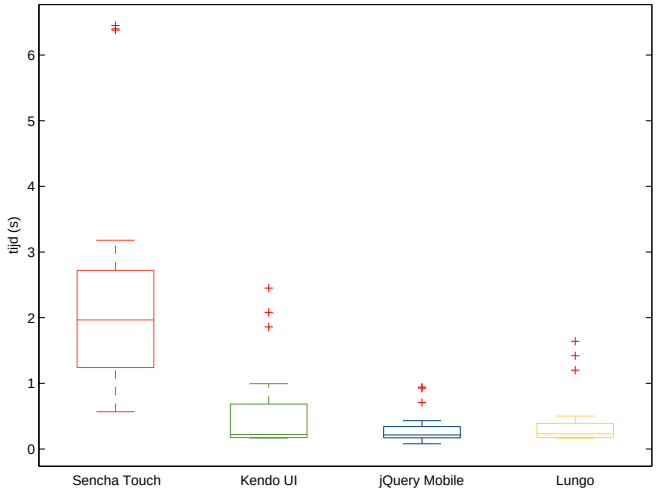
Average download time Figure 1 shows the download time of the login application and the login application from cache for the four frameworks.

Lungo has the fastest download time. jQuery Mobile, KUI en ST have respectively the second, third and fourth fastest download time. Lungo is about half of the time faster than the other three frameworks. The download times of KUI, jQM and Lungo are proportional to the download size of the framework's resources. QuoJS, the underlying JavaScript library of Lungo, is optimized for mobile devices and needs the least resources when looking at code. The JavaScript dependencies of KUI and jQM are more or less equal in size although the former imposes an architecture and the latter does not. Finally, ST uses the most JavaScript code because it is JavaScript driven. However, ST uses postponed parsing of JavaScript so the browser can download all JavaScript resources faster.

When looking at the login application from cache, KUI, jQM and Lungo have about the same download time. However ST has a much higher download time when using the cached version. The first reason is that the first three frameworks are only using HTML5 Application Cache. ST also uses the Delta Update mechanism in addition to HTML5 Application Cache. When the application is updated, this mechanism will only download the updated code instead of all the files that are declared in the `manifest` file. The second reason is that the applications in the first three frameworks are compiled using Yeoman [59] and Sencha Touch with Sencha Cmd [42].



(a) Login application



(b) Login application from cache

Figure 1: Download time of the login application.

User experience Table 8 shows the user experience. ST takes first place with the maximal score. This means that the scroll experience was best on all the devices. The generation of the 850 list items was impossible for KUI on all the iOS devices. The browser crashed on these devices and returned to the home screen. In contrast, list generation did work on all Android devices.

5.6 Comparison overview

Figure 2 shows the spider graph with the scores for the five criteria for the four frameworks.

jQM has the best overall score (86%) and is the winner following our criteria. One of the major factors that makes jQM successful is that it does not enforces an architecture. This decreases the learning curve in favour of productivity. Also, the extensive documentation and large community make jQM more productive. Lungo also forces no architecture but is extremely unpopular. The absence of an architecture also has two disadvantages. Firstly, the framework provides less functionality but many plugins and HTML5 features act as substitutes. Secondly, the HTML code that needs to be written is verbose. This complicates the linking of HTML with JavaScript code.

Device	ST	KUI	jQM	L
HTCDesireZ	4	3	2	1
GalaxyTab	4	2	3	1
GalaxyS	4	2	3	1
Nexus 7	4	1	3	2
iPad1 WiFi	4		2	3
iPad3 4G WiFi	4		3	2
iPhone 3GS	4		3	2
iPhone 4S	4		3	2
Total	32	8	22	14

Table 8: User experience of scrolling through a long list.

KUI is second with a total score of 73%. The MVVM architecture makes KUI more usable but less productive in comparison with jQM. However, the MVVM architecture seems more intuitive than the MVC architecture of ST. The main drawback of KUI, which can not be seen on the figure, is the high license cost. The expectation that open-source frameworks are more popular is not reflected in the popularity scores. KUI is also the least performant. The crashes on iOS devices caused a decrease in the user experience tests and hence a drop in performance. The native look and feel and corresponding benefits are not been studied although this feature is stressed by the KUI company.

Lungo has a score of 64% and claims the third place. In three out of five criteria, Lungo is last. Again, no architecture is imposed which can be seen in productivity and usage. However, the advantage in productivity is smaller in comparison with jQM because of the limited documentation. Also, the disadvantage in usage is larger because not as much functionality or plugins as with jQM could be found. A positive aspect is that Lungo achieved the best download times. This is because QuoJS, the JavaScript library of Lungo, is optimised for mobile devices. The user experience tests weakened the score for performance

ST has a final score of 61% and is the worst framework following these criteria. The combination of the MVC architecture and the fact that it is JavaScript driven makes ST both the least productive and least performant in comparison to the other frameworks. All HTML code is generated by the framework and therefor the JavaScript files are very large. Also, ST uses the Delta Update mechanism to update files from the cache. As a result, the download times were remarkably larger. The user experience tests gave an opposite result because after the long download time, ST had the best user experience. The tools provided by Sencha to support the developer are not able to close the gap in productivity. The tools require a learning process on their own. As a last remark, ST is depending on the WebKit engine. Devices with a default browser that do not contain this engine, can not be used with ST.

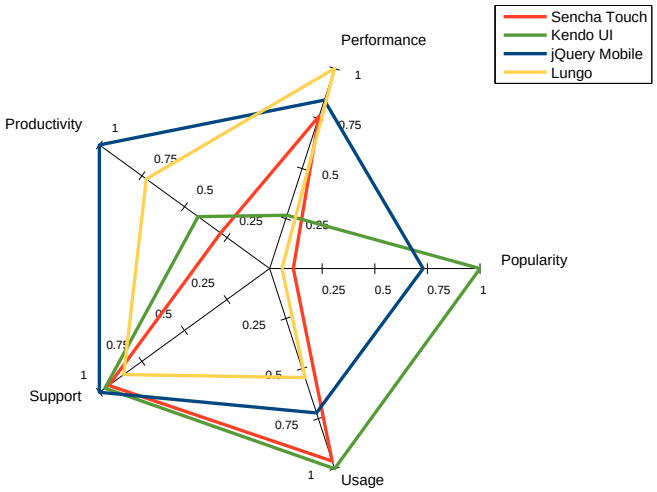


Figure 2: Overview with the five comparison criteria.

6 Future work

Firstly, one can look for deeper understanding why certain results are remarkable and why other things failed on particular devices. An example for the latter is to search why the 850 list items of KUI crashed on all iOS devices. Furthermore research can be done to know the limit of list items when it starts crashing.

Secondly, one can add new frameworks to the comparison. This will enlarge the comparison itself, but will also recheck the comparison method. Furthermore, new versions of the frameworks will be published. It may also be worth evaluating the comparison over time. This is because the ranking can change due to new features in framework or new plugins.

Thirdly, the five criteria are driven by the POC. This POC can be further extended with extra features like pull-to-refresh list. Similar to this action, one can also use other events than the tap event. Examples of these are double tap, swipe, hold, and also events that require multiple fingers like a rotate event. Furthermore, the integration of HTML5 features in frameworks likes GPS, push events, drag and drop, video and audio can be investigated.

Fourthly, new criteria can be added to the comparison. It is off interest to incorporate extensibility which states how easy it is to extend an existing application, similar as in the ISO 25010 standard. A hypothesis is that frameworks that enforce an architecture will have better extensibility than others. This new criterion could change the current ranking of the four frameworks. Another possible new criterion is to look at the final result of the application created by the framework. Some frameworks make applications that imitate the nativeness of the current OS. Other frameworks make applications with an out-of-the-box modern layout.

Lastly, one can take a step back and investigate how web applications cope with battery usage. This data can also be compared with native and hybrid applications. This comparison can be more generalised by comparing web, native and hybrid applications.

7 Conclusion

This paper described a comparative study between ST, KUI, jQM and Lungo. ST is built with the MVC architecture and is JavaScript driven. KUI enforces the MVVM architecture and is both JavaScript as markup driven. jQM and Lungo do not have an architecture and are both markup driven.

Five criteria were chosen to execute the comparative study: popularity, productivity, usage, support and performance. Each criterion was provided with a formula to calculate a score. A POC was formalised to drive the usage and support criterion. The activity of the framework on social networks determined the popularity. The productivity was measured by time tracking the implementation of a login application. This application is a subset of the POC extended with a long list. The POC was subdivided in 13 challenges with a total of 38 sub challenges to test the usage of the framework. Next, a subset of these challenges were tested on 8 different devices to check the support. Finally, the download time and user experience of the login application determined the performance criterion. All the scores were plotted in a spider graph.

The comparison shows that jQM is the best framework. Then, KUI, Lungo and ST follow on respectively second, third and forth place. jQM is highly productive because on the one hand it is well-documented and on the other hand it does not impose an architecture. The latter, however, is a disadvantage when looking at the usage. KUI has the MVVM architecture as most important advantage. However, it scores below average on performance. Lungo only achieved a maximum score for performance because QuoJS, the underlying JavaScript library of Lungo, is optimized for mobile devices. ST is the least productive and performant in comparison with other frameworks. By contrast, ST scores the best in user experience. By enforcing an architecture, it scores almost as good as KUI regarding usage. All frameworks provide good support on the devices.

References

[1] Accenture. HTML5: The Path to Cross-Platform Mobile Development. 2012.

[2] Android. Platform Versions. `http://developer.android.com/about/dashboards/index.html`. [Online; accessed 12/05/2013].

[3] Apple. Apple Updates iOS to 6.1. `http://www.apple.com/pr/library/2013/01/28Apple-Updates-iOS-to-6-1.html`. [Online; accessed 28/02/2013].

[4] A. Ayuso. Mobile Frameworks Comparison. `http://monocaffe.blogspot.be/2012/06/mobile-frameworks-comparison.html`. [Online; accessed 19/02/2013].

[5] J. Bristowe. jQuery UI vs Kendo UI. `http://jqueryuivskendoui.com/`. [Online; accessed 19/02/2013].

[6] B. Broulik. *Pro jQuery Mobile*. Apress, 2012.

[7] C. Burris. Sencha Touch vs. JQuery Mobile Cage-Fight REMATCH ! `http://stvjqm.mobivant.`

com/presentations/svj/. [Online; accessed 19/02/2013].

[8] M. David. *HTML5 Mobile Websites: Turbocharging HTML5 with jQuery Mobile, Sencha Touch, and Other Frameworks.* Focal Press, 2011.

[9] P. Deitel, H. Deitel, A. Deitel, and E. Kern. *iOS 6 for Programmers: An App-Driven Approach, Second Edition.* Prentice Hall, 2012.

[10] A. Deveria. Can I Use. http://caniuse.com/. [Online; accessed 14/05/2013].

[11] Dmethvin. jQuery Licensing Changes. http://blog.jquery.com/2012/09/10/jquery-licensing-changes/. [Online; accessed 28/02/2013].

[12] M. Falk. Mobile Framework Comparison Chart. http://www.markus-falk.com/mobile-frameworks-comparison-chart/. [Online; accessed 28/01/2013].

[13] M. Firtman. *Programming the Mobile Web, 2nd Edition.* O'Reilly Media, Inc., 2013.

[14] A. Gal. PDF.JS. http://mozilla.github.io/pdf.js/, 2010.

[15] T. Gideon. Samsung Galaxy Tablet Coming in September. http://www.pcmag.com/article2/0,2817,2368214,00.asp. [Online; accessed 15/05/2013].

[16] F. Gillett. Why Tablets Will Become Our Primary Computing Device. http://blogs.forrester.com/frank_gillett/12-04-23-why_tablets_will_become_our_primary_computing_device. [Online; accessed 27/02/2013].

[17] W. Hales. *HTML5 and JavaScript Web Apps.* O'Reilly Media, Inc., 2012.

[18] I. Jacobs. HTML5 Definition Complete, W3C Moves to Interoperability Testing and Performance. http://www.w3.org/2012/12/html5-cr. [Online; accessed 27/02/2013].

[19] A. S. Jadhav and R. M. Sonar. Evaluating and selecting software packages: A review. *Information and Software Technology*, 51(3):555–563, Mar. 2009.

[20] T. Jeroen. Html5 vs Native for Mobile Application Development. http://www.icapps.be/html5-vs-native/. [Online; accessed 30/05/2013].

[21] S. Jobs. Thoughts on Flash. http://www.apple.com/hotnews/thoughts-on-flash/. [Online; accessed 30/12/2012].

[22] JQuery. jQuery Mobile. http://www.jquerymobile.com. [Online; accessed 22/02/2013].

[23] JQuery. jQuery Mobile: Demos and Documentation. http://jquerymobile.com/demos/1.2.0/. [Online; accessed 22/02/2013].

[24] JQuery. jQuery Project. http://www.jquery.org. [Online; accessed 29/05/2013].

[25] JQuery. Mobile Graded Browser Support. http://jquerymobile.com/gbs/. [Online; accessed 28/02/2013].

[26] A. Kosmaczewski. *Mobile JavaScript Application Development.* O'Reilly Media, 2012.

[27] V. Mobile. Cross-platform developer tools 2012. Technical report, London, 2012.

[28] Modernizr. Modernizr: the feature detection library for HTML5/CSS3. http://www.modernizr.com. [Online; accessed 28/02/2013].

[29] D. Oehlman and S. Blanc. *Pro Android Web Apps: Develop for Android using HTML5, CSS3 & JavaScript.* Apress, 2011.

[30] T. O'Reilly. Safari Books Online. http://www.safaribooksonline.com/. [Online; accessed 30/05/2013].

[31] T. Parker. Announcing jQuery Mobile 1.0. http://jquerymobile.com/blog/2011/11/16/announcing-jquery-mobile-1-0/. [Online; accessed 28/02/2013].

[32] T. Parker. Announcing jQuery Mobile 1.2.0 Final. http://jquerymobile.com/blog/2012/10/02/announcing-jquery-mobile-1-2-0-final/. [Online; accessed 28/02/2013].

[33] T. Parker. Announcing jQuery Mobile 1.3.1. http://jquerymobile.com/blog/2013/04/10/announcing-jquery-mobile-1-3-1/. [Online; accessed 15/04/2013].

[34] Phil Dutson. *Sams Teach Yourself jQuery Mobile in 24 Hours.* Sams, 2012.

[35] E. Protalinski. Android grabs 75.0% market share in Q3, followed by 14.9% for iOS and 4.3% for BlackBerry. http://thenextweb.com/mobile/2012/11/01/android-grabs-75-0-market-share-in-q3-followed-by-14-9-for-ios-and-4-3-for-blackberry/. [Online; accessed 28/02/2013].

[36] J. Resig. Announcing the jQuery Mobile Project. http://jquerymobile.com/blog/2010/08/11/announcing-the-jquery-mobile-project/. [Online; accessed 28/02/2013].

[37] C. Rozynski. Mobile framework SMACKDOWN! http://dinosaurswithlaserz.com/2011/03/28/mobile-framework-throwdown/. [Online; accessed 28/02/2013].

[38] T. L. Saaty. *The Analytic Hierarchy Process, Planning, Piority Setting, Resource Allocation.* McGraw-Hill, 1980.

[39] A. Sarrafi. Mobile JavaScript frameworks, Evaluation and Comparison. http://www.codefessions.com/2012/04/mobile-javascript-

frameworks-evaluation.html. [Online; accessed 19/02/2013].

[40] Satyesh. Android Versions History. http://androidtrickz.com/android-tips/android-versions-history/489/. [Online; accessed 30/12/2012].

[41] P. Sawers. Google announces Android 4.2, a "new flavor of Jelly Bean" with gesture typing and slick photo sphere camera. http://thenextweb.com/google/2012/10/29/google-announces-android-4-2-for-nexus-a-new-flavor-of-jelly-bean-with-gesture-typing-multiple-users/. [Online; accessed 30/12/2012].

[42] Sencha. Sencha Cmd. http://www.sencha.com/products/sencha-cmd/download, 2012.

[43] Sencha Inc. Sencha. http://www.sencha.com/. [Online; accessed 28/01/2013].

[44] Sencha Inc. Sencha Touch Documentatie. http://docs.sencha.com/touch/2-1/. [Online; accessed 28/01/2013].

[45] Sencha Inc. Sencha Touch Licenses. http://www.sencha.com/products/touch/license/. [Online; accessed 18/02/2013].

[46] Sencha Inc. Sencha Touch Kitchen Sink. http://dev.sencha.com/deploy/touch/examples/production/kitchensink/, 2013.

[47] D. Smith. iOS Version Stats. http://david-smith.org/iosversionstats/. [Online; accessed 12/05/2013].

[48] C. N. Staff. Live blog: iPhone 3G S launch day. http://news.cnet.com/8301-13579_3-10268140-37.html. [Online; accessed 15/05/2013].

[49] I. Standard. ISO/IEC 25010. Technical report, 2010.

[50] TapQuo. Initial commit v.1.0.0. https://github.com/TapQuo/Lungo.js/commit/de5f4b86b02a719f9cf1049a1ec895f76c00cf0d. [Online; accessed 15/04/2013].

[51] TapQuo. Lungo. http://lungo.tapquo.com/. [Online; accessed 15/04/2013].

[52] TapQuo. Lungo licence. https://github.com/TapQuo/Lungo.js/blob/master/LICENSE.md. [Online; accessed 29/05/2013].

[53] TapQuo. Lungo Reference. http://lungo.tapquo.com/howto/prototype/. [Online; accessed 14/05/2013].

[54] TapQuo. TapQuo. http://www.tapquo.com/. [Online; accessed 15/04/2013].

[55] Telerik. Kendo UI Documentatie. http://docs.kendoui.com/. [Online; accessed 12/04/2013].

[56] Telerik. Kendo UI Website. http://www.kendoui.com. [Online; accessed 13/04/2013].

[57] Viljami Salminen. File Upload Support on Mobile. http://viljamis.com/blog/2012/file-upload-support-on-mobile/. [Online; accessed 20/05/2013].

[58] J. Yang. Smartphones in Use Surpass 1 Billion, Will Double by 2015. http://www.bloomberg.com/news/2012-10-17/smartphones-in-use-surpass-1-billion-will-double-by-2015.html. [Online; accessed 01/02/2013].

[59] Yeoman. Yeoman. http://yeoman.io/. [Online; accessed 4/05/2013].

Bijlage

C Ondersteuning

In deze appendix wordt een gedetailleerd overzicht gegeven van de ondersteuning van de uitdagingen op de acht apparaten: HTCDesireZ (zie tabel C.1), GalaxyTab (zie tabel C.2), GalaxyS (zie tabel C.3), Nexus 7 (zie tabel C.4), iPad1 WiFi (zie tabel C.5), iPad3 4G WiFi (zie tabel C.6), iPhone 3GS (zie tabel C.7), iPhone 4S (zie tabel C.8).

Uitdaging	**ST**	**KUI**	**jQM**	**L**
U2: Toestelspecifieke lay-out	1	1	1	1
U4.1: Maak formulieren met *placeholders* zonder labels	1	0	1	1
U4.2: Gebruik tekst/nummer/e-mail als formuliertypes	0	0	0	0
U4.3: Optieveld	0	1	1	
U4.4: *Datepicker*	1	1	1	1
U4.6: Schakelaar	1	1	1	0
U6: Auto-aanvullen		1	1	0
U7: Toevoegen en verwerken van afbeelding	0	0	0	0
U8: Formuliervalidatie	1	1	1	
U9: Handtekening	1	1	1	
U12: Toon PDF	0	0	0	0
U13.1: Bewaar data	1	1	1	1
U13.2: Maak de applicatie offline beschikbaar	1	1	1	1
Totaal	8	9	10	5

Tabel C.1: Ondersteuning op HTCDesireZ.

Uitdaging	**ST**	**KUI**	**jQM**	**L**
U2: Toestelspecifieke lay-out	0	1	0	0
U4.1: Maak formulieren met *placeholders* zonder labels	1	0	1	1
U4.2: Gebruik tekst/nummer/e-mail als formuliertypes	0	0	0	0
U4.3: Optieveld	0	1	1	
U4.4: *Datepicker*	1	1	1	1
U4.6: Schakelaar	1	1	1	0
U6: Auto-aanvullen		1	1	1
U7: Toevoegen en verwerken van afbeelding	0	0	0	0
U8: Formuliervalidatie	1	1	1	
U9: Handtekening	1	1	1	
U12: Toon PDF	0	0	0	0
U13.1: Bewaar data	1	1	1	1
U13.2: Maak de applicatie offline beschikbaar	1	1	1	1
Totaal	7	9	9	5

Tabel C.2: Ondersteuning op GalaxyTab.

Uitdaging	**ST**	**KUI**	**jQM**	**L**
U2: Toestelspecifieke lay-out	1	1	1	1
U4.1: Maak formulieren met *placeholders* zonder labels	1	0	1	1
U4.2: Gebruik tekst/nummer/e-mail als formuliertypes	1	1	1	1
U4.3: Optieveld	0	1	1	
U4.4: *Datepicker*	1	1	1	1
U4.6: Schakelaar	1	1	1	0
U6: Auto-aanvullen		1	1	1
U7: Toevoegen en verwerken van afbeelding	1	1	1	1
U8: Formuliervalidatie	1	1	1	
U9: Handtekening	1	1	1	
U12: Toon PDF	1	0	0	0
U13.1: Bewaar data	1	1	1	1
U13.2: Maak de applicatie offline beschikbaar	1	1	1	1
Totaal	11	11	12	8

Tabel C.3: Ondersteuning op GalaxyS.

Uitdaging	**ST**	**KUI**	**jQM**	**L**
U2: Toestelspecifieke lay-out	1	1	1	1
U4.1: Maak formulieren met *placeholders* zonder labels	1	0	1	1
U4.2: Gebruik tekst/nummer/e-mail als formuliertypes	1	1	1	1
U4.3: Optieveld	1	1	1	
U4.4: *Datepicker*	1	1	1	1
U4.6: Schakelaar	1	1	1	0
U6: Auto-aanvullen		1	1	1
U7: Toevoegen en verwerken van afbeelding	1	1	1	1
U8: Formuliervalidatie	1	1	1	
U9: Handtekening	1	1	1	
U12: Toon PDF	1	1	1	1
U13.1: Bewaar data	1	1	1	1
U13.2: Maak de applicatie offline beschikbaar	1	1	1	1
Totaal	12	12	13	9

Tabel C.4: Ondersteuning op Nexus 7.

Uitdaging	**ST**	**KUI**	**jQM**	**L**
U2: Toestelspecifieke lay-out	1	1	1	1
U4.1: Maak formulieren met *placeholders* zonder labels	1	1	1	1
U4.2: Gebruik tekst/nummer/e-mail als formuliertypes	1	1	1	1
U4.3: Optieveld	1	1	1	
U4.4: *Datepicker*	1	1	1	1
U4.6: Schakelaar	1	1	1	0
U6: Auto-aanvullen		1	1	1
U7: Toevoegen en verwerken van afbeelding	0	0	0	0
U8: Formuliervalidatie	1	1	1	
U9: Handtekening	1	1	1	
U12: Toon PDF	1	1	1	1
U13.1: Bewaar data	1	1	1	1
U13.2: Maak de applicatie offline beschikbaar	1	1	1	1
Totaal	11	12	12	8

Tabel C.5: Ondersteuning op iPad1 WiFi.

Uitdaging	**ST**	**KUI**	**jQM**	**L**
U2: Toestelspecifieke lay-out	1	1	1	1
U4.1: Maak formulieren met *placeholders* zonder labels	1	1	1	1
U4.2: Gebruik tekst/nummer/e-mail als formuliertypes	1	1	1	1
U4.3: Optieveld	1	1	1	
U4.4: *Datepicker*	1	1	1	1
U4.6: Schakelaar	1	1	1	0
U6: Auto-aanvullen		1	1	1
U7: Toevoegen en verwerken van afbeelding	1	1	1	1
U8: Formuliervalidatie	1	1	1	
U9: Handtekening	1	1	1	
U12: Toon PDF	1	1	1	1
U13.1: Bewaar data	1	1	1	1
U13.2: Maak de applicatie offline beschikbaar	1	1	1	1
Totaal	12	13	13	9

Tabel C.6: Ondersteuning op iPad3 4G WiFi.

Uitdaging	**ST**	**KUI**	**jQM**	**L**
U2: Toestelspecifieke lay-out	1	1	1	1
U4.1: Maak formulieren met *placeholders* zonder labels	1	1	1	1
U4.2: Gebruik tekst/nummer/e-mail als formuliertypes	1	1	1	1
U4.3: Optieveld	0	1	1	
U4.4: *Datepicker*	1	1	1	1
U4.6: Schakelaar	1	1	1	0
U6: Auto-aanvullen		1	1	1
U7: Toevoegen en verwerken van afbeelding	1	1	1	1
U8: Formuliervalidatie	1	1	1	
U9: Handtekening	1	1	1	
U12: Toon PDF	1	1	1	1
U13.1: Bewaar data	1	1	1	1
U13.2: Maak de applicatie offline beschikbaar	1	1	1	1
Totaal	11	13	13	9

Tabel C.7: Ondersteuning op iPhone 3GS.

Uitdaging	**ST**	**KUI**	**jQM**	**L**
U2: Toestelspecifieke lay-out	1	1	1	1
U4.1: Maak formulieren met *placeholders* zonder labels	1	1	1	1
U4.2: Gebruik tekst/nummer/e-mail als formuliertypes	1	1	1	1
U4.3: Optieveld	0	1	1	
U4.4: *Datepicker*	1	1	1	1
U4.6: Schakelaar	1	1	1	0
U6: Auto-aanvullen		1	1	1
U7: Toevoegen en verwerken van afbeelding	1	1	1	1
U8: Formuliervalidatie	1	1	1	
U9: Handtekening	1	1	1	
U12: Toon PDF	1	1	1	1
U13.1: Bewaar data	1	1	1	1
U13.2: Maak de applicatie offline beschikbaar	1	1	1	1
Totaal	11	13	13	9

Tabel C.8: Ondersteuning op iPhone 4S.

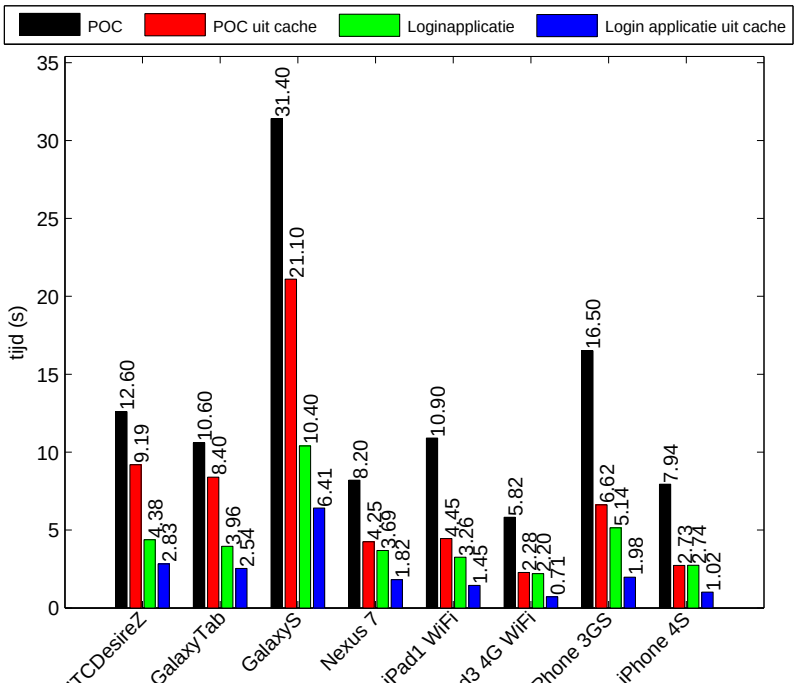
Bijlage

D Performantie

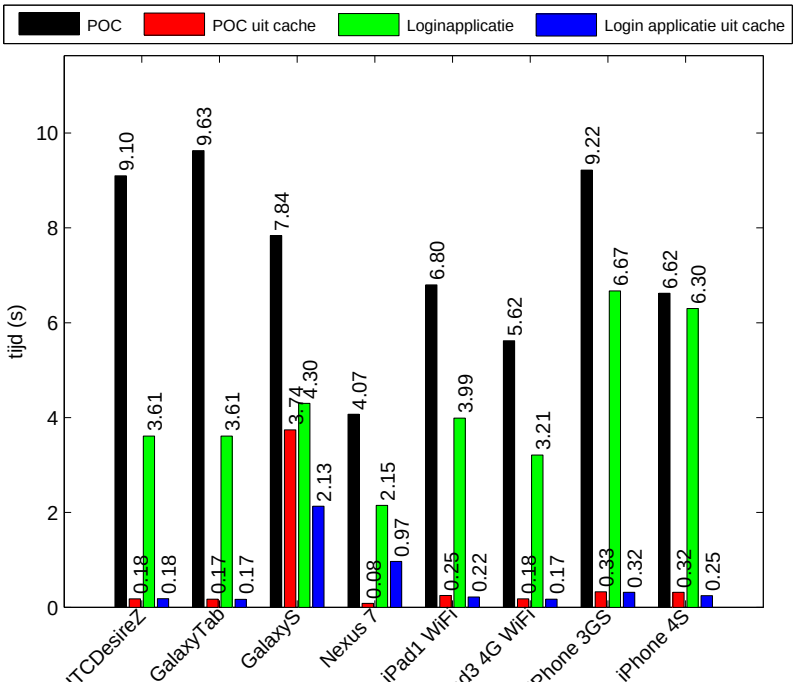
In deze appendix wordt een gedetailleerd overzicht gegeven van de performantie voor de vier raamwerken op de acht apparaten: Sencha Touch (zie D), Kendo UI (zie D), jQuery Mobile (zie D), Lungo (zie D).

Sencha Touch

Op figuur D.1 wordt de gemiddelde downloadtijd van Sencha Touch getoond op elk apparaat. Voor de POC is een dalende downloadtijd waarneembaar wanneer het



Figuur D.1: Gemiddelde downloadtijden van Sencha Touch voor POC, POC uit cache, loginapplicatie en loginapplicatie uit cache voor elk apparaat.



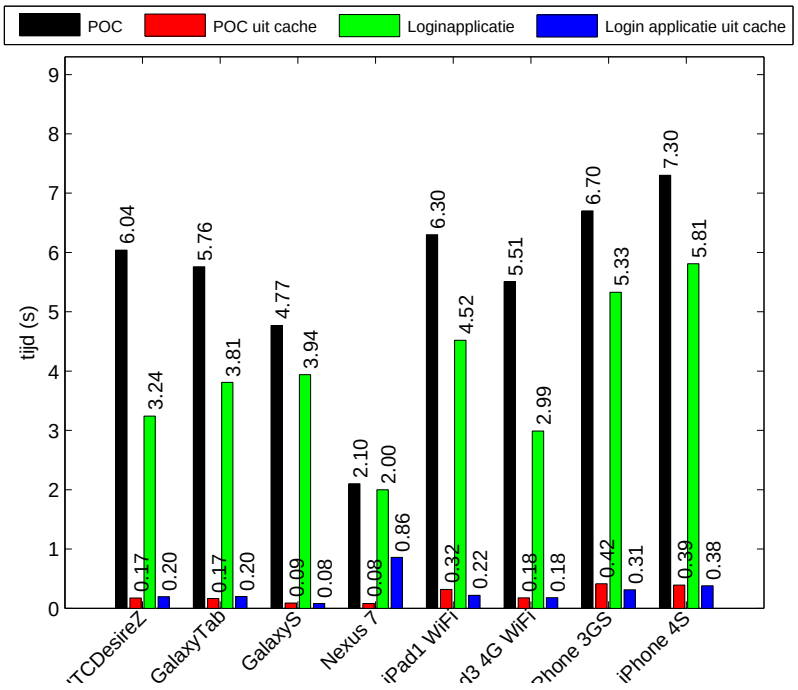
Figuur D.2: Gemiddelde downloadtijden van Kendo UI voor POC, POC uit cache, loginapplicatie en loginapplicatie uit cache voor elk apparaat.

Android-apparaat recenter wordt. De downloadtijd van de POC op de GalaxyS duurde gemiddeld 31.43 s. Gemiddeld moeten Android-toestellen 5 s langer laden in vergelijking met iOS-toestellen. Dit gemiddelde wordt sterk beïnvloed door de trage downloadtijd van de GalaxyS.

Een opmerking die bij Sencha Touch moet worden gemaakt, is dat AJAX-verzoeken van een `Proxy` naar een ander domein altijd vooraf worden gegaan met een OPTIONS-verzoek. Dit is een verzoek om informatie over de beschikbare opties van het communicatiekanaal op te vragen. Standaard zet Sencha Touch de `X-Requested-With` op XMLHttpRequest en hierdoor zal de browser een OPTIONS-verzoek als *preflight* sturen.

Kendo UI

Op figuur D.2 worden de gemiddelde downloadtijd van Kendo UI getoond op elk apparaat. De GalaxyTab vertoont de hoogste downloadtijd, gevolgd door de iPhone 3GS en HTCDesireZ. Opmerkelijk is dat de loginapplicatie uit cache op de Nexus 7 tien keer trager laadt dan de POC uit cache. Het ophalen van een applicatie uit cache werkt bij de GalaxyS het traagst. Bij Kendo UI is er geen opmerkelijk verschil waarneembaar tussen Android- en iOS-toestellen. Android is gemiddeld 60 ms trager.



Figuur D.3: Gemiddelde downloadtijd van jQuery Mobile voor POC, POC uit cache, loginapplicatie en loginapplicatie uit cache voor elk apparaat.

jQuery Mobile

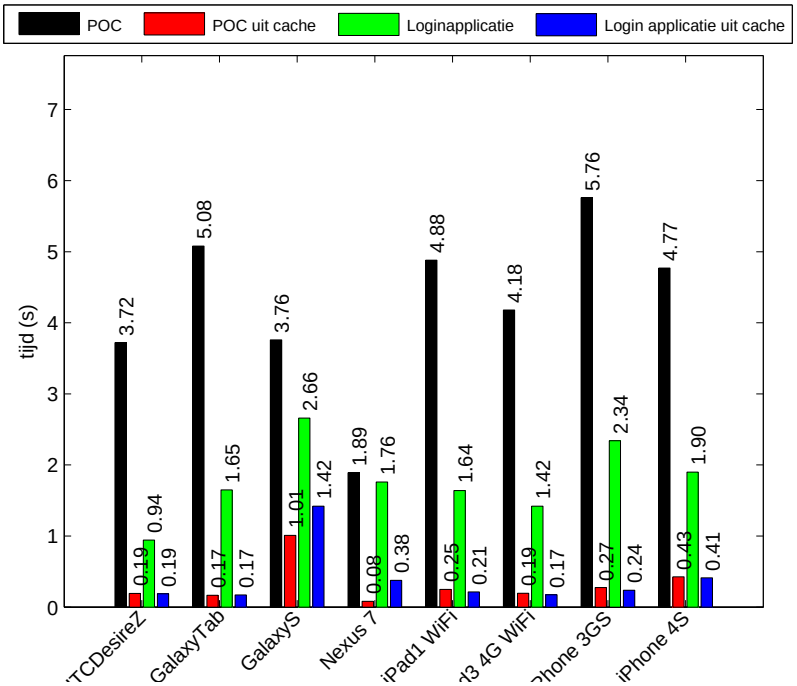
Op figuur D.3 wordt de gemiddelde downloadtijd van jQuery Mobile getoond op elk apparaat. Voor de POC is een dalende downloadtijd waarneembaar wanneer de Android-versies op het apparaat recenter worden. iPads dalen in downloadtijd als het apparaat recenter wordt, daarentegen stijgt de downloadtijd bij iPhones.

Als de loginapplicatie wordt bekeken, wordt hetzelfde waargenomen als voor de POC. Enkel bij de Android-apparaten wordt de downloadtijd trager, naarmate de Android-versie recenter wordt, wat in tegenstelling is tot de POC. Enkel de Nexus 7 volgt deze trend niet. Op de Nexus 7 wordt de loginapplicatie ongeveer even snel gedownload als de POC. Een opmerkelijke waarneming is dat het op de Nexus 7 langer duurt om de loginapplicatie uit cache te laden dan de volledige POC uit cache.

Lungo

Op figuur D.4 wordt de gemiddelde downloadtijd van Lungo getoond op elk apparaat. Er is geen opmerkelijke trend waarneembaar voor Android. Wel is er een opmerkelijke waarneming op de GalaxyS, waarbij de downloadtijden uit cache langer duren dan 1 s, terwijl op alle apparaten deze tijden rond de $0,25$ s liggen. Bij iOS daalt de downloadtijd als het toestel recenter is. Zo is de downloadtijd op iPad3 4G WiFi sneller dan iPad1 WiFi. Het geldt ook zo dat de iPhone 4S sneller downloadt dan de

iPhone 3GS.



Figuur D.4: Gemiddelde downloadtijd van Lungo voor POC, POC uit cache, loginapplicatie en loginapplicatie uit cache voor elk apparaat.

Bibliografie

[1] Accenture. HTML5: The Path to Cross-Platform Mobile Development. 2012.

[2] Adobe. Adobe Flash Player 11. http://www.adobe.com/be_nl/products/flashplayer.html. [Online; geraadpleegd op 14/05/2013].

[3] Android. Launch Checklist. http://developer.android.com/distribute/googleplay/publish/preparing.html. [Online; geraadpleegd op 16/05/2013].

[4] Android. Permissions. http://developer.android.com/guide/topics/security/permissions.html. [Online; geraadpleegd op 16/05/2013].

[5] Android. Platform Versions. http://developer.android.com/about/dashboards/index.html. [Online; geraadpleegd op 12/05/2013].

[6] Apple. App Store Review Guidelines. http://stadium.weblogsinc.com/engadget/files/app-store-guidelines.pdf. [Online; geraadpleegd op 16/05/2013].

[7] Apple. Apple Launches iPad. http://www.apple.com/pr/library/2010/01/27Apple-Launches-iPad.html. [Online; geraadpleegd op 27/02/2013].

[8] Apple. Know iOS Resource Limits. http://developer.apple.com/library/safari/#documentation/AppleApplications/Reference/SafariWebContent/CreatingContentforSafarioniPhone/CreatingContentforSafarioniPhone.html#//apple_ref/doc/uid/TP40006482-SW15. [Online; geraadpleegd op 26/02/2013].

[9] A. Ayuso. Mobile Frameworks Comparison. http://monocaffe.blogspot.be/2012/06/mobile-frameworks-comparison.html. [Online; geraadpleegd op 19/02/2013].

[10] Backbone. Backbone. http://backbonejs.org/. [Online; geraadpleegd op 19/05/2013].

[11] R. Berjon, T. Leithead, E. D. Navara, E. O'Connor, and S. Pfeiffer. HTML 5.1: Forms. http://www.w3.org/html/wg/drafts/html/FPWD51/forms.html#forms. [Online; geraadpleegd op 25/02/2013].

[12] T. Bradley. Signature Pad. http://thomasjbradley.ca/lab/signature-pad/. [Online; geraadpleegd op 22/02/2013].

[13] J. Bristowe. jQuery UI vs Kendo UI. http://jqueryuivskendoui.com/. [Online; geraadpleegd op 19/02/2013].

[14] B. Broulik. *Pro jQuery Mobile*. Apress, 2012.

[15] C. Burris. Sencha Touch vs. JQuery Mobile Cage-Fight REMATCH ! http://stvjqm.mobivant.com/presentations/svj/. [Online; geraadpleegd op 19/02/2013].

[16] H. Catlin, N. Weizenbaum, and C. Eppstein. SASS. http://sass-lang.com/. [Online; geraadpleegd op 28/01/2013].

[17] J. Catone. GitHub: A Social Network for Programmers. http://readwrite.com/2008/04/11/github_a_social_network_for_programmers. [Online; geraadpleegd op 14/05/2013].

[18] J. E. Clark and B. P. Johnson. *Sencha Touch Mobile JavaScript Framework*. Packt Publishing, 2012.

[19] M. David. *HTML5 Mobile Websites: Turbocharging HTML5 with jQuery Mobile, Sencha Touch, and Other Frameworks*. Focal Press, 2011.

[20] David Kaneda. jQT. http://jqtjs.com/. [Online; geraadpleegd op 15/04/2013].

[21] S. Deering. 12+ jQuery Mobile Layout Plugins and Examples. http://www.jquery4u.com/page-layout/12-jquery-mobile-layout-plugins-examples/. [Online; geraadpleegd op 22/02/2013].

[22] P. Deitel, H. Deitel, A. Deitel, and E. Kern. *iOS 6 for Programmers: An App-Driven Approach, Second Edition*. Prentice Hall, 2012.

[23] J.-P. Déry. Moobile. http://moobilejs.com/. [Online; geraadpleegd op 15/04/2013].

[24] A. Deveria. Can I Use. http://caniuse.com/. [Online; geraadpleegd op 14/05/2013].

[25] Dmethvin. jQuery Licensing Changes. http://blog.jquery.com/2012/09/10/jquery-licensing-changes/. [Online; geraadpleegd op 28/02/2013].

[26] C. M. Eppstein. Compass. http://compass-style.org/. [Online; geraadpleegd op 28/01/2013].

[27] M. Falk. Mobile Framework Comparison Chart. http://www.markus-falk.com/mobile-frameworks-comparison-chart/. [Online; geraadpleegd op 28/01/2013].

[28] A. Fiedler. Ext.ux.panel.PDF. https://github.com/SunboX/st2_pdf_panel, 2012.

[29] M. Firtman. Mobile HTML5 - compatibility on iPhone, Android, Windows Phone, BlackBerry, Symbian and other mobile and tablet devices. http://mobilehtml5.org/. [Online; geraadpleegd op 14/05/2013].

[30] M. Firtman. *Programming the Mobile Web, 2nd Edition*. O'Reilly Media, Inc., 2013.

[31] S. Franck. jQuery Mobile Multiview Plugin. https://github.com/frequent/multiview/. [Online; geraadpleegd op 22/02/2013].

[32] A. Gal. PDF.JS. http://mozilla.github.io/pdf.js/, 2010.

[33] GCF. What is a Mobile Device? http://www.gcflearnfree.org/computerbasics/9/print. [Online; geraadpleegd op 27/02/2013].

[34] T. Gideon. Samsung Galaxy Tablet Coming in September. http://www.pcmag.com/article2/0,2817,2368214,00.asp. [Online; geraadpleegd op 15/05/2013].

[35] F. Gillett. Why Tablets Will Become Our Primary Computing Device. http://blogs.forrester.com/frank_gillett/12-04-23-why_tablets_will_become_our_primary_computing_device. [Online; geraadpleegd op 27/02/2013].

[36] S. González. Web Storage Support Test. http://dev-test.nemikor.com/web-storage/support-test/. [Online; geraadpleegd op 26/02/2013].

[37] Google. Google Trends. https://www.google.com/trends/, 2012.

[38] A. Grady. Introduction to Mobile Device Management. http://www.bubblews.com/news/396878-introduction-to-mobile-device-management. [Online; geraadpleegd op 16/05/2013].

[39] F. Guelinckx. *Onderzoek naar samenwerking op een tabletop en mobiele apparaten*. KU Leuven, 2012.

[40] K. Hadlock. Implement responsive design with jQuery Mobile and CSS3. pages 1–10, 2012.

[41] W. Hales. *HTML5 and JavaScript Web Apps*. O'Reilly Media, Inc., 2012.

[42] P. L. Hégaret, L. Wood, and J. Robie. What is the Document Object Model? http://www.w3.org/TR/DOM-Level-3-Core/introduction.html. [Online; geraadpleegd op 28/02/2013].

[43] J. Hens. Progressive Enhancement versus Graceful Degradation. http://www.goodbytes.be/blog/article/progressive-enhancement-versus-graceful-degradation. [Online; geraadpleegd op 28/02/2013].

[44] Incross. DaVinci. http://www.davincisdk.com/. [Online; geraadpleegd op 8/04/2013].

[45] I. Jacobs. HTML5 Definition Complete, W3C Moves to Interoperability Testing and Performance. http://www.w3.org/2012/12/html5-cr. [Online; geraadpleegd op 27/02/2013].

[46] A. S. Jadhav and R. M. Sonar. Evaluating and selecting software packages: A review. *Information and Software Technology*, 51(3):555–563, Mar. 2009.

[47] T. Jeroen. Html5 vs Native for Mobile Application Development. http://www.icapps.be/html5-vs-native/. [Online; geraadpleegd op 30/05/2013].

[48] JetBrains. PhpStorm. http://www.jetbrains.com/phpstorm/. [Online; geraadpleegd op 17/05/2013].

[49] S. Jobs. Thoughts on Flash. http://www.apple.com/hotnews/thoughts-on-flash/. [Online; geraadpleegd op 30/12/2012].

[50] JQT. First commit. https://github.com/senchalabs/jQTouch/commits/master?page=14. [Online; geraadpleegd op 15/04/2013].

[51] JQuery. Autocomplete. http://view.jquerymobile.com/1.3.0/docs/widgets/autocomplete/. [Online; geraadpleegd op 23/02/2013].

[52] JQuery. Form reference. http://view.jquerymobile.com/1.3.0/docs/widgets/forms/. [Online; geraadpleegd op 23/02/2013].

[53] JQuery. Going Responsive. http://view.jquerymobile.com/1.3.0/docs/intro/rwd.php. [Online; geraadpleegd op 25/02/2013].

[54] JQuery. Header structure. http://jquerymobile.com/demos/1.2.0/docs/toolbars/docs-headers.html. [Online; geraadpleegd op 23/02/2013].

[55] JQuery. jQuery. http://jquery.com/. [Online; geraadpleegd op 14/05/2013].

[56] JQuery. jQuery Mobile. http://www.jquerymobile.com. [Online; geraadpleegd op 22/02/2013].

[57] JQuery. jQuery Mobile: Demos and Documentation. http://jquerymobile.com/demos/1.2.0/. [Online; geraadpleegd op 22/02/2013].

[58] JQuery. jQuery Mobile Docs - Forms. http://jquerymobile.com/demos/1.2.0/docs/forms/docs-forms.html. [Online; geraadpleegd op 22/02/2013].

[59] JQuery. jQuery Project. http://www.jquery.org. [Online; geraadpleegd op 29/05/2013].

[60] JQuery. Mobile Graded Browser Support. http://jquerymobile.com/gbs/. [Online; geraadpleegd op 28/02/2013].

[61] JQuery. ThemeRoller for jQuery Mobile. http://jquerymobile.com/themeroller/. [Online; geraadpleegd op 22/02/2013].

[62] KnockOut. KnockOut. http://knockoutjs.com/. [Online; geraadpleegd op 19/05/2013].

[63] M. Kool. Let's Play With Hardware-Accelerated CSS. http://mobile.smashingmagazine.com/2012/06/21/play-with-hardware-accelerated-css/. [Online; geraadpleegd op 28/02/2013].

[64] A. Kosmaczewski. *Mobile JavaScript Application Development*. O'Reilly Media, 2012.

[65] Krause. Conditional fields validations. http://www.sencha.com/forum/showthread.php?122680-Conditional-fields-validations. [Online; geraadpleegd op 17/05/2013].

[66] D. Laubach. The-M-Project 2013: What's next? http://blog.the-m-project.org/2013/01/28/the-m-project-2013-whats-next/. [Online; geraadpleegd op 8/04/2013].

[67] Y. Lauwers. Mobiel breedband nu en in de toekomst. http://tweakers.net/reviews/662/4/mobiel-breedband-nu-en-in-de-toekomst-hsdpa.html. [Online; geraadpleegd op 14/05/2013].

[68] D. Lieberman. Microsoft's Windows Phone 7 to replace Windows Mobile. http://usatoday30.usatoday.com/tech/wireless/2010-10-11-windows-phone-7_N.htm. [Online; geraadpleegd op 30/12/2012].

[69] M. A. lsong, bmcquade. PCAP Web Performance Analyzer. http://pcapperf.appspot.com/, 2010.

[70] M. MacDonald. *HTML5: The Missing Manual*. O'Reilly Media, Inc., 2011.

[71] Martti. Spider plot. http://www.mathworks.com/matlabcentral/fileexchange/14875-spider-plot, 2007.

[72] A. Matthews. AutoComplete. https://github.com/commadelimited/autoComplete.js. [Online; geraadpleegd op 22/02/2013].

[73] D. S. McFarland. *JavaScript & jQuery: The Missing Manual, Second Edition*. O'Reilly Media, Inc., 2011.

[74] U. Miami. Mobile phones: 1G 2G 3G 4G. http://it.med.miami.edu/x1645.xml. [Online; geraadpleegd op 14/05/2013].

[75] Microsoft. App certification requirements for Windows Phone. http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh184843(v=vs.105).aspx. [Online; geraadpleegd op 16/05/2013].

[76] Microsoft. Microsoft Silverlight. http://www.microsoft.com/silverlight/. [Online; geraadpleegd op 14/05/2013].

[77] Microsoft. Nokia and Microsoft Announce Plans for a Broad Strategic Partnership to Build a New Global Mobile Ecosystem. https://www.microsoft.com/en-us/news/press/2011/feb11/02-11partnership.aspx. [Online; geraadpleegd op 30/12/2012].

[78] Microsoft. Tablet PC Brings the Simplicity of Pen and Paper to Computing. http://www.microsoft.com/en-us/news/features/2000/nov00/11-13tabletpc.aspx. [Online; geraadpleegd op 27/02/2013].

[79] V. Mobile. Cross-platform developer tools 2012. Technical report, London, 2012.

[80] Mobiscroll. Date & Time Scroller. http://mobiscroll.com/component/datetime. [Online; geraadpleegd op 23/02/2013].

[81] Modernizr. Modernizr: the feature detection library for HTML5/CSS3. http://www.modernizr.com. [Online; geraadpleegd op 28/02/2013].

[82] J. Morgan. How to Measure Website Performance with Three Free Tools. http://usabilityetc.com/2011/08/measure-website-performance-with-free-tools/. [Online; geraadpleegd op 19/02/2013].

[83] NetApplications. Mobile/Tablet Browser Market Share. http://www.netmarketshare.com/browser-market-share.aspx?qprid=0&qpcustomd=1. [Online; geraadpleegd op 14/05/2013].

[84] J. Nguyen. Behind the Sencha Command Utility and the Build Process. http://www.sencha.com/blog/behind-sencha-command-and-the-build-process. [Online; geraadpleegd op 21/05/2013].

[85] J. Nielsen. Response Times: The 3 Important Limits. http://www.nngroup.com/articles/response-times-3-important-limits/. [Online; geraadpleegd op 11/05/2013].

[86] N. Nishanth and R. K. Bhandari. *Building Mobile Applications with Kendo UI Mobile*. Packt Publishing, 2013.

[87] D. Oehlman and S. Blanc. *Pro Android Web Apps: Develop for Android using HTML5, CSS3 & JavaScript*. Apress, 2011.

[88] T. O'Reilly. Safari Books Online. http://www.safaribooksonline.com/. [Online; geraadpleegd op 30/05/2013].

[89] Panacoda. The M-Project. http://www.the-m-project.org/. [Online; geraadpleegd op 8/04/2013].

[90] T. Parker. Announcing jQuery Mobile 1.0. http://jquerymobile.com/blog/2011/11/16/announcing-jquery-mobile-1-0/. [Online; geraadpleegd op 28/02/2013].

[91] T. Parker. Announcing jQuery Mobile 1.2.0 Final. http://jquerymobile.com/blog/2012/10/02/announcing-jquery-mobile-1-2-0-final/. [Online; geraadpleegd op 28/02/2013].

[92] T. Parker. Announcing jQuery Mobile 1.3.1. http://jquerymobile.com/blog/2013/04/10/announcing-jquery-mobile-1-3-1/. [Online; geraadpleegd op 15/04/2013].

[93] G. Phifer and D. M. Smith. The (Not So) Future Web. Technical report, Gartner, 2011.

[94] Phil Dutson. *Sams Teach Yourself jQuery Mobile in 24 Hours*. Sams, 2012.

[95] E. Protalinski. Android grabs 75.0% market share in Q3, followed by 14.9% for iOS and 4.3% for BlackBerry. http://thenextweb.com/mobile/2012/11/01/android-grabs-75-0-market-share-in-q3-followed-by-14-9-for-ios-and-4-3-for-blackberry/. [Online; geraadpleegd op 28/02/2013].

[96] A. A. Rahman. jQuery Mobile Splitview plugin. https://github.com/asyraf9/jquerymobile-splitview. [Online; geraadpleegd op 22/02/2013].

[97] B. Reed. Microsoft show off new key Windows Phone 8 features. http://bgr.com/2012/10/29/microsoft-windows-phone-8-announcement/. [Online; geraadpleegd op 30/12/2012].

[98] J. Resig. Announcing the jQuery Mobile Project. http://jquerymobile.com/blog/2010/08/11/announcing-the-jquery-mobile-project/. [Online; geraadpleegd op 28/02/2013].

[99] C. Rozynski. Mobile framework SMACKDOWN! http://dinosaurswithlaserz.com/2011/03/28/mobile-framework-throwdown/. [Online; geraadpleegd op 28/02/2013].

[100] T. L. Saaty. *The Analytic Hierarchy Process, Planning, Piority Setting, Resource Allocation*. McGraw-Hill, 1980.

[101] J. Sage. jQuery Mobile DateBox. https://github.com/jtsage/jquery-mobile-datebox. [Online; geraadpleegd op 25/02/2013].

[102] A. Sarrafi. Mobile JavaScript frameworks, Evaluation and Comparison. http://www.codefessions.com/2012/04/mobile-javascript-frameworks-evaluation.html. [Online; geraadpleegd op 19/02/2013].

[103] Satyesh. Android Versions History. http://androidtrickz.com/android-tips/android-versions-history/489/. [Online; geraadpleegd op 30/12/2012].

[104] P. Sawers. Google announces Android 4.2, a "new flavor of Jelly Bean" with gesture typing and slick photo sphere camera. http://thenextweb.com/google/2012/10/29/google-announces-android-4-2-for-nexus-a-new-flavor-of-jelly-bean-with-gesture-typing-multiple-users/. [Online; geraadpleegd op 30/12/2012].

[105] Scott. jQuery Plugin for Requesting Ajax-like File Downloads. http://www.filamentgroup.com/lab/jquery_plugin_for_requesting_ajax_like_file_downloads/. [Online; geraadpleegd op 4/05/2013].

[106] B. Seitz. Windows Phone 7 Global Portfolio Unveiled in New York. http://blogs.windows.com/windows_phone/b/windowsphone/archive/2010/10/11/windows-phone-7-global-portfolio-unveiled-in-new-york.aspx. [Online; geraadpleegd op 30/12/2012].

[107] Sencha. Sencha Market. https://market.sencha.com/. [Online; geraadpleegd op 21/05/2013].

[108] Sencha. Sencha Animator. http://www.sencha.com/products/animator/, 2012.

[109] Sencha. Sencha Architect. http://www.sencha.com/products/architect/, 2012.

[110] Sencha. Sencha Cmd. http://www.sencha.com/products/sencha-cmd/download, 2012.

[111] Sencha Inc. Sencha. http://www.sencha.com/. [Online; geraadpleegd op 28/01/2013].

[112] Sencha Inc. Sencha Touch Documentatie. http://docs.sencha.com/touch/2-1/. [Online; geraadpleegd op 28/01/2013].

[113] Sencha Inc. Sencha Touch Kitchen Sink. http://dev.sencha.com/deploy/touch/examples/production/kitchensink/, 2013.

[114] SimFla. SimFla-signaturePad. https://github.com/SimFla/SimFla-signaturePad, 2011.

[115] C. V. Smirnov. File-uploading-component-for-Sencha-Touch. https://github.com/kostysh/File-uploading-component-for-Sencha-Touch, 2012.

[116] D. Smith. iOS Version Stats. http://david-smith.org/iosversionstats/. [Online; geraadpleegd op 12/05/2013].

[117] B. Sperry and M. Lynch. Codiqa. https://www.codiqa.com. [Online; geraadpleegd op 28/02/2013].

[118] C. N. Staff. Live blog: iPhone 3G S launch day. http://news.cnet.com/8301-13579_3-10268140-37.html. [Online; geraadpleegd op 15/05/2013].

[119] I. Standard. ISO/IEC 25010. Technical report, 2010.

[120] W. Systems. jSignature. http://willowsystems.github.com/jSignature. [Online; geraadpleegd op 22/02/2013].

[121] M. Tajur. Sencha Touch 2 Autocomplete Textfield. https://github.com/martintajur/sencha-touch-2-autocomplete-textfield, 2012.

[122] TapQuo. Initial commit v.1.0.0. https://github.com/TapQuo/Lungo.js/commit/de5f4b86b02a719f9cf1049a1ec895f76c00cf0d. [Online; geraadpleegd op 15/04/2013].

[123] TapQuo. Lungo. http://lungo.tapquo.com/. [Online; geraadpleegd op 15/04/2013].

[124] TapQuo. Lungo licence. https://github.com/TapQuo/Lungo.js/blob/master/LICENSE.md. [Online; geraadpleegd op 29/05/2013].

[125] TapQuo. Lungo Reference. http://lungo.tapquo.com/howto/prototype/. [Online; geraadpleegd op 14/05/2013].

[126] TapQuo. Lungo thirdparties. github.com/TapQuo/lungo.thirdparties. [Online; geraadpleegd op 4/05/2013].

[127] TapQuo. QuoJS. http://quojs.tapquo.com/. [Online; geraadpleegd op 14/05/2013].

[128] TapQuo. TapQuo. http://www.tapquo.com/. [Online; geraadpleegd op 15/04/2013].

[129] Tcpdump. Tcpdump. http://www.tcpdump.org/. [Online; geraadpleegd op 17/05/2013].

[130] Telerik. Kendo UI Documentatie. http://docs.kendoui.com/. [Online; geraadpleegd op 12/04/2013].

[131] Telerik. Kendo UI Dojo. http://try.kendoui.com.

[132] Telerik. Kendo UI Mobile Theme Builder. http://demos.kendoui.com/mobilethemebuilder/index.htm.

[133] Telerik. Kendo UI Web Theme Builder. http://demos.kendoui.com/themebuilder/web.htm.

[134] Telerik. Kendo UI Website. http://www.kendoui.com. [Online; geraadpleegd op 13/04/2013].

[135] S. Thair. Measuring mobile performance. http://www.slideshare.net/sthair/measuring-mobile-web-performance-v2. [Online; geraadpleegd op 17/05/2013].

[136] Twitter. Twitter Bower. https://github.com/twitter/bower. [Online; geraadpleegd op 15/04/2013].

[137] Viljami Salminen. File Upload Support on Mobile. http://viljamis.com/blog/2012/file-upload-support-on-mobile/. [Online; geraadpleegd op 20/05/2013].

[138] W3C. HTML5 Logo. http://www.w3.org/html/logo/. [Online; geraadpleegd op 28/02/2013].

[139] E. Weyl, L. Lazaris, and A. Goldstein. *HTML5 & CSS3 For The Real World*. SitePoint, 2011.

[140] T. Wimberly. Goodbye old Browser, Chrome to become the standard browser on Android 4.0 and above. http://androidandme.com/2012/02/applications/goodbye-old-browser-chrome-to-become-the-standard-browser-on-android-4-0-and-above/. [Online; geraadpleegd op 14/05/2013].

[141] J. Yang. Smartphones in Use Surpass 1 Billion, Will Double by 2015. http://www.bloomberg.com/news/2012-10-17/smartphones-in-use-surpass-1-billion-will-double-by-2015.html. [Online; geraadpleegd op 01/02/2013].

[142] P. Yared and M. Raiser. SimpleSplitView. http://simplesplitview.sourceforge.net/. [Online; geraadpleegd op 22/02/2013].

[143] Yeoman. Yeoman. http://yeoman.io/. [Online; geraadpleegd op 4/05/2013].

[144] J. Zaefferer. jQuery plugin: Validation. http://bassistance.de/jquery-plugins/jquery-plugin-validation/. [Online; geraadpleegd op 22/02/2013].

[145] Zepto. Zepto. http://www.zeptojs.com/. [Online; geraadpleegd op 15/04/2013].

[146] Zimmy. Which band/artist has the most songs? http://answers.yahoo.com/question/index?qid=20110315195858AAAsqHA. [Online; geraadpleegd op 14/05/2013].

Fiche masterproef

Studenten: Tim Ameye
Sander Van Loock

Titel: Vergelijkende studie van raamwerken voor de ontwikkeling van mobiele HTML5-applicaties

Engelse titel: Comparative study of frameworks for the development of mobile HTML5 applications

UDC: 681.3

Korte inhoud:

Ontwikkelaars van mobiele applicaties worden geconfronteerd met een variëteit aan mobiele besturingssystemen die op smartphones en tablets aanwezig zijn. Dit komt doordat een applicatie dient te worden geprogrammeerd aan de hand van een Software Development Kit (SDK) die specifiek is voor het besturingssysteem. De ontwikkeling van mobiele webapplicaties, gebruikmakend van HTML5, is een mogelijke oplossing hiervoor. Om het ontwikkelingsproces van deze mobiele HTML5-applicaties te versnellen worden raamwerken aangeboden. Deze helpen zowel bij het toevoegen van functionaliteit als de elementen voor de gebruikersinterface. Door de variëteit aan mobiele HTML5-raamwerken dringt een grondige vergelijking zich op. Dit werk vergelijkt Sencha Touch, Kendo UI, jQuery Mobile en Lungo op basis van vijf vergelijkingscriteria: populariteit, productiviteit, gebruik, ondersteuning en performantie. Populariteit kijkt naar de activiteit van de raamwerken op sociale netwerken. Productiviteit wordt opgemeten om aan te tonen hoe lang het duurt om met een raamwerk vertrouwd te raken en er daadwerkelijk iets mee te maken. Het gebruik van de raamwerken bekijkt de elementen die het raamwerk aanbiedt. Ondersteuning test de elementen van het raamwerk op acht verschillende mobiele apparaten. Er wordt een evenwichtige keuze gemaakt tussen Android, iOS, smartphone en tablet. Performantie meet enerzijds de downloadtijd en anderzijds de gebruikerservaring. De laatstgenoemde bepaalt hoe vlot het gaat om door een lange lijst te scrollen. De vergelijking toont aan dat jQuery Mobile het beste raamwerk is. Daarna volgen Kendo UI, Lungo en Sencha Touch. jQuery Mobile heeft als belangrijkste troef de hoge productiviteit doordat het enerzijds zeer goed gedocumenteerd is en anderzijds geen ontwerppatroon afdwingt. Dit laatste is echter ook een nadeel omdat het hierdoor minder scoort op gebruik. Kendo UI heeft als troef het gebruik doordat het een ontwerppatroon afdwingt. Het scoort echter ondermaats op performantie. Lungo behaalde enkel de maximumscore voor performantie doordat QuoJS, de JavaScript-bibliotheek van Lungo, geoptimaliseerd is voor mobiele apparaten. Sencha Touch is het minst productief en performant. Daarentegen scoort Sencha Touch het best op het vlak van gebruikerservaring. Door het afdwingen van een ontwerppatroon scoort het quasi evengoed als Kendo UI op vlak van gebruik. Alle onderzochte raamwerken scoren zeer goed op ondersteuning op de onderzochte mobiele apparaten.

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdspecialisatie Veilige software en hoofdspecialisatie Gedistribueerde systemen

Promotor: Prof. dr. ir. E. Duval

Assessoren: Prof. dr. M. Denecker
F. Van Assche

Begeleider: Ir. G. Parra