Multiplatform Frameworks voor mobiele applicaties.

Een vergelijking tussen verschillende frameworks voor multiplatform applicaties zoals Xamarin Forms, NativeScript en Apache Cordova.

Figuur 1: Dijoantonycj, 2015. Gebruikt onder Creative Commons BY-SA

Bachelorscriptie
30 Mei 2017

Naam: Martin Luhrman
Studentnummer: 1638082
Opleiding: HBO-ICT Software Engineering
Onderwijsinstelling: Hogeschool Utrecht
Opdrachtgever: Infoplaza B.V.
Begeleiders:
Dick Pronk (Hogeschool Utrecht)
Sjoerd Huininga (Infoplaza B.V.)
Eerste examinator:
Marco Dumont (Hogeschool Utrecht)

Voorwoord

Voor u ligt de scriptie 'Multiplatform Frameworks voor mobiele applicaties'. Deze scriptie is geschreven in het kader van het behalen van mijn bachelor voor de opleiding Software & Information Engineering bij de Hogeschool Utrecht. Van februari tot en met mei 2017 ben ik bezig geweest met het onderzoek voor mijn scriptie, dat uitgevoerd is in opdracht van Infoplaza BV.

Voor het begin van dit project had ik zeer oppervlakkige kennis over het werken met mobiele platformen. Dit onderzoek sprak mij aan omdat ik graag mijn kennis op dit gebied wilde gaan uitbreiden en mezelf daar mee wou uitdagen.

Samen met mijn bedrijfsbegeleider heb ik mijn onderzoeksvraag voor deze scriptie bedacht. Met de hulp van feedback van mijn leerteam en mijn docentbegeleider is deze onderzoeksvraag verder getoetst zodat de uiteindelijke versie is ontstaan die nu in deze scriptie staat.

Ik wil in de eerste plaats graag mijn bedrijfsbegeleider Sjoerd Huininga bedanken voor de fijne begeleiding. Ik kon goed sparren over complexe vraagstukken en altijd rekenen op goede feedback en ondersteuning waar nodig. Daarnaast wil ik graag mijn collega Robert van Dijk bedanken voor de persoonlijke begeleiding tijdens het afstudeertraject. Dankzij zijn advies en toeziende oog heb ik mij persoonlijk sterk kunnen ontwikkelen.

Ook wil ik graag mijn docentbegeleider Dick Pronk bedanken voor de fijne begeleiding en ondersteuning tijdens het traject. Hierna wil ik ook graag mijn dank uiten naar mijn collega's bij Infoplaza voor de fijne samenwerking en prettige werksfeer. Ze waren altijd makkelijk te benaderen voor vragen en boden mij vaak nieuwe en interessante inzichten.

Als laatste wil ik graag mijn vriendin, ouders, vrienden en familie bedanken voor de morele steun tijdens het proces.

Ik wens u veel leesplezier toe.

Martin Luhrman

Hilversum, 24 mei 2017

Management Samenvatting

Binnen Infoplaza is de wens ontstaan om van de huidige infrastructuur voor hun mobiele applicaties over te stappen naar een cross-platform ontwikkelingsmethode. Dit document onderzoekt de beschikbare frameworks en welke het beste bij Infoplaza past.

Aanbevelingen

- Maak gebruik van het Xamarin Framework voor nieuwe applicaties.
- Maak gebruik van Xamarin University om ontwikkelaars snel Xamarin te leren.
- Maak gebruik van Xamarin Test Cloud, zodat de app snel op meer dan 400 verschillende apparaten getest kan worden.
- Overweeg het gebruik van Azure Mobile Center in de plaats van Crashlytics.

Motivatie

Door middel van interviews en een vergelijkingsonderzoek zijn 13 frameworks met elkaar vergeleken op basis van de doelstellingen en de resultaten van de interviews. Hiervan zijn de top 4 frameworks getest op hun prestaties, waar Xamarin en NativeScript het beste uit de bus kwamen. Uiteindelijk is er gekozen om Xamarin te gaan gebruiken omdat deze het beste aansluit bij de infrastructuur van de organisatie.

Met het Xamarin Framework is daarna de OVplaza app ontwikkeld om te testen of het framework in de praktijk voldoet aan de gestelde doelstellingen. De app voldeed aan het einde van het project aan 6 van de 8 doelstellingen, waarvan de 2 overige doelstellingen niet aantoonbaar waren maar het wel mogelijk is dat deze correct zijn.

Op basis hiervan is geconcludeerd dat het Xamarin Framework inderdaad het meest geschikt is.

Consequenties

Hoewel het Xamarin Framework bewezen voordelen biedt bij het ontwikkelen, is het nog onduidelijk wat de impact zal zijn van het vertalen van de bestaande OVplaza apps. Hiervoor moet op een per-app basis gekeken moeten worden naar de kosten voor het omzetten tegenover de winst die behaald kan worden door de mogelijk verminderde ontwikkeltijd.

Voor het gebruik van Xamarin University moet rekeningen gehouden worden met een prijs van $999,- per jaar per ontwikkelaar. Xamarin Test Cloud kost $99,- tot $799,- per maand, afhankelijk voor welk pakket gekozen wordt.

Als laatste is Azure Mobile Center nog in Preview, waardoor er nog veel functies missen en er nog geen prijzen bekend zijn. Voor een overstap is het daarom aan te raden om af te wachten tot het de preview fase verlaat eind 2017.

Inhoud

Figuren- en tabellenlijst

Figuren

Tabellen

1. Inleiding

Mobiele telefoons zijn een kenmerk van deze tijd. Uit recent onderzoek (Pew Research Center, 2017) blijkt dat 77% van de Amerikanen een smartphone bezitten. Veruit het grootste deel van de smartphones draaien volgens een ander onderzoek (IDC, 2016) Android of iOS. Met zo een grote doelgroep is er voor bedrijven een groot publiek te bereiken via apps voor hun diensten. Om een app te kunnen maken bieden Google en Apple speciale Software Development Kits aan. Voor Android is deze ontwikkeld in Java en voor iOS in Objective-C en Swift.

De apps die gemaakt worden met behulp van deze SDK's staan ook wel bekend als 'Native' apps. Er komt echter al snel het probleem dat wanneer men voor Android én iOS een app wil maken, dat dezelfde logica meerdere keren geschreven wordt vanwege de aparte SDK en talen die bij de platformen horen.

Verschillende partijen hebben oplossingen op de markt gebracht voor dit probleem. De meeste van deze oplossingen maken gebruik van een van de volgende twee strategieën:

1. Er wordt een abstractie boven de API van de platformen gelegd, waardoor er in een enkele codebase een applicatie gemaakt kan worden voor Android en iOS. Deze oplossing produceert een app die na compilatie theoretisch gelijk is aan een native applicatie.
2. De applicatie wordt met HTML, CSS en JavaScript opgebouwd. Via een speciale wrapper app kan deze applicatie op ieder apparaat met dezelfde code getoond worden, omdat het in feite niet veel meer is dan een website.

Binnen Infoplaza wordt er nog native ontwikkeld, maar er is interesse ontstaan voor de mogelijkheden die een cross-platform framework kan bieden voor hun apps.

Dit document geeft toelichting op de opdracht die vanuit Infoplaza aan de afstudeerder is gegeven. Het beschrijft hoe de afstudeerder de opdracht heeft geanalyseerd, op welke manier het onderzoek is uitgevoerd, tegen welke problemen of keuzes de afstudeerder aan is gelopen en hoe de afstudeerder tot zijn conclusies is gekomen.

2. Context en aanleiding

In dit hoofdstuk wordt er voor de context van de opdracht toelichting gegeven over de organisatie en wordt er beschreven wat de aanleiding was voor de opdracht.

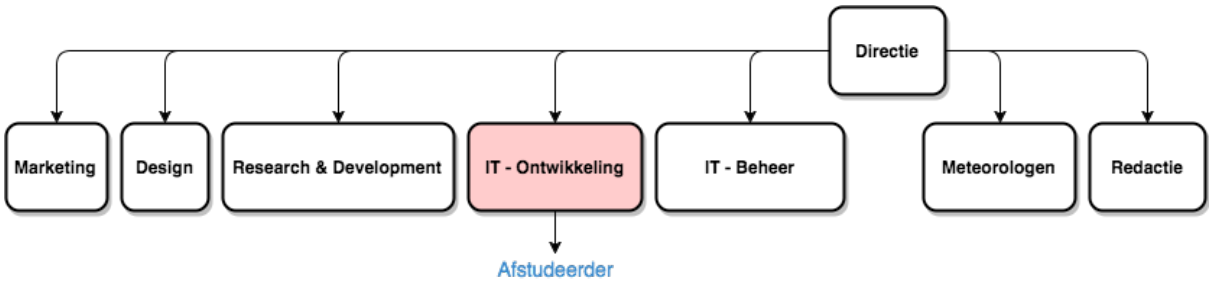
2.1. Organisatorische Context

Infoplaza is een bedrijf dat streeft om weers-, verkeers- en ov-informatie op een duidelijke manier te leveren aan zijn klanten. Deze informatie wordt gedistribueerd aan consumenten via Infoplaza's websites, apps en via de verkoop van data en diensten aan derden zoals de NOS, nu.nl en het merendeel van de kranten in Nederland. Ook biedt Infoplaza zijn informatie en kennis aan bedrijven die afhankelijk zijn van betrouwbare weersinformatie voor het leveren van hun eigen diensten, zoals bijvoorbeeld ProRail.

Er werken op dit moment 21 mensen in fulltime of parttime dienstverband en de organisatie bestaat grofweg uit 3 afdelingen:

- De directie
- De redactie en meteorologen
- Een verzameling van IT-beheer, ontwikkelaars, designers en marketing.

Binnen de organisatie ontfermt de afstudeerder zich over de ov-applicatie van Infoplaza en maakt daarmee deel uit van de ontwikkelaars.



Figuur 2: Organogram

2.2. Aanleiding

De Android, iOS en Windows apps worden in de huidige situatie bij Infoplaza per platform native ontwikkeld. Meestal wordt dit uitbesteed aan externe ontwikkelaars die per platform ingehuurd worden, wat extra complexiteit toevoegt aan het stroomlijnen van het ontwikkelproces. Ook moet de achterliggende logica van de applicaties op ieder platform opnieuw geschreven worden.

Door de tijd die het proces kost en de redundantie die plaatsvindt tijdens het ontwikkelen ziet Infoplaza een kans voor manieren om applicaties cross-platform te kunnen ontwikkelen. Hierdoor kunnen dezelfde ontwikkelaars zich op alle platformen richten, en hoeft logica maar één keer geschreven te worden waarna het makkelijk hergebruikt kan worden.

3. Doelstelling en Opdracht

In dit hoofdstuk wordt aangegeven wat de doelstellingen zijn van de opdrachtgever. Daarna wordt er een beschrijving gegeven van de opdracht van Infoplaza.

3.1. Doelstellingen

Uit de kwestie blijkt dat er binnen de organisatie de wens is om van de huidige infrastructuur voor hun mobiele applicaties over te stappen naar een cross-platform ontwikkelingsmethode. Het vraagstuk dat daaruit ontstaat is of de overstap naar een cross-platform framework de verwachte voordelen kan bieden. Hiervoor wordt een proof of concept applicatie gemaakt waarmee de eisen aan het framework getoetst zullen worden zoals gegeven in Tabel 1.

Tabel 1: Eisen voor het framework

Requirement	Belang
De applicatieontwikkeling loopt gelijk voor Android en iOS.	*Technisch, marketing en financieel*
Het onderhoud aan de applicatie, bijvoorbeeld het toevoegen van een nieuwe functionaliteit of het oplossen van een bug, moet minder tijd kosten dan wanneer dit native zou gebeuren door twee aparte ontwikkelaars.	*Technisch, marketing en financieel*
De applicatie kan op Android en iOS gezamenlijk gelanceerd worden.	*Marketing en financieel*
De interface van de applicatie biedt dezelfde huisstijl op Android en iOS.	*Marketing*
De applicatie maakt gebruik van een gedeelde codebase voor de business logic.	*Technisch en financieel*
De impact van systeemupdates van Android en iOS moet minimaal zijn voor het gebruik van het framework.	*Technisch*
De applicatie behaalt een goede GPU-performance op de verschillende platformen van gemiddeld rond de 60 FPS.	*Technisch en marketing*
De applicatie biedt een minimale impact op de API-infrastructuur van Infoplaza.	*Technisch en financieel*

3.2. Opdracht

De opdracht van Infoplaza is om te onderzoeken welk cross-platform framework gebruikt kan worden voor het ontwikkelen van nieuwe en bestaande apps van Infoplaza. Om dit framework uit te testen is besloten dat de OVplaza app als proof of concept gebruikt zal worden om te kijken of het gekozen framework voldoet aan de wensen van Infoplaza. De bestaande OVplaza app was namelijk al sinds 2013 niet geüpdatet en moest daarom sterk vernieuwd worden.

Voor het onderzoek worden verschillende Frameworks voor het bouwen van cross-platform apps geanalyseerd en vergeleken, zoals bijvoorbeeld het C#.NET Framework Xamarin van Xamarin Inc. en hybride frameworks zoals het Javascript Framework Cordova van The Apache Software Foundation. Uit dit onderzoek moet blijken welk framework het beste binnen Infoplaza kan worden toegepast voor het bouwen van cross-platform mobiele applicaties.

Op basis van de wensen en de huidige situatie van Infoplaza ligt de focus op de mogelijkheden van Xamarin tegenover alternatieve mogelijkheden omdat de ontwikkelaars binnen Infoplaza al veelal bekend zijn met C# en het .NET Framework. Hierdoor moet er ook in de toekomst diepere integratie mogelijk zijn met de websites van Infoplaza omdat deze ook grotendeels in .NET geschreven zijn.

3.3. Onderzoeksvragen

Op basis van de aanleiding en de opdracht zoals geleverd door Infoplaza is de volgende hoofdvraag geformuleerd:

> Op welke manier zou een cross-platform framework voor mobiele applicaties ingezet kunnen worden voor de apps van Infoplaza, zodat er minder code gedupliceerd wordt over de verschillende platformen?

Op basis van deze hoofdvraag zijn drie deelvragen opgesteld:

1. *Hoe worden de huidige mobiele applicaties van Infoplaza gerealiseerd?*
2. *Welke frameworks zijn er beschikbaar die cross-platform capaciteiten bieden voor mobile development?*
3. *Wat zijn de voor- en nadelen van de beschikbare frameworks voor cross-platform applicaties in verhouding tot elkaar?*

Met het beantwoorden van deze drie deelvragen moet er een duidelijk beeld ontstaan van het antwoord op de hoofdvraag.

4. Planning

In dit hoofdstuk worden de verschillende fasen van het onderzoek besproken. Daarna worden de deliverables op een rij gezet en wordt de planning van het project verder toegelicht.

4.1. Fasering

Het verloop van het onderzoek is onder te verdelen in 3 fasen: planning, onderzoek en bouwen van het proof of concept. De onderzoeksfase is op te splitsen in het onderzoek naar de deelvragen en de bouwfase in de 4 sprints die uitgevoerd zijn. De verdeling hiervan is te zien in Tabel 2.

Tabel 2: Onderzoeksfasen

Fase	Sub Fase	Resultaat	Loopduur
Planning	-	Plan van Aanpak	7 feb – 24 mrt
Onderzoek	Onderzoek huidige situatie	Onderdeel onderzoeksrapport	27 feb – 3 mrt
	Onderzoek bestaande frameworks	Onderdeel onderzoeksrapport	7 mrt – 10 mrt
	Onderzoek vergelijking frameworks	Onderdeel onderzoeksrapport	7 mrt – 17 mrt
	Beantwoording hoofdvraag	Onderzoeksrapport	27 feb – 17 mrt
Bouwen Proof of Concept	Voorbereiding	User story map, product backlog	20 mrt – 25 mrt
	Sprint 1	Iteratie op OVplaza app voor Android en iOS, Scrum documentatie van de sprint en iteratie op technisch- en functioneel ontwerprapport	27 mrt – 7 apr
	Sprint 2		10 apr – 21 apr
	Sprint 3		24 apr – 5 mei
	Sprint 4		8 mei – 19 mei

4.2. Deliverables

Aan het einde van dit onderzoek zullen de volgende deliverables opgeleverd worden:

- Het Plan van Aanpak dat beschrijft hoe het onderzoek uitgevoerd gaat worden.
- Het Onderzoeksrapport dat de resultaten van het onderzoek naar de deelvragen en de hoofdvraag beantwoordt.
- De Scriptie die de keuzes die gemaakt zijn tijdens het onderzoek en het bouwen van het proof of concept toelicht.
- Een functioneel ontwerprapport.
- Een technisch ontwerprapport.
- Een applicatie voor Android en iOS.
- Scrum documentatie:
    - Sprint planningen
    - Sprint reviews
    - Sprint retrospectives
    - De product backlog
    - De sprint backlogs
- Een persoonlijke reflectie op het onderzoek.

4.3. Planning

Om een beeld te kunnen geven van de verwachte activiteiten die in dit onderzoek plaats zullen vinden is de onderstaande tabel gemaakt. Hierin is een overzicht gemaakt van de activiteiten die plaats moeten vinden en de verwachte uren die voor deze activiteit nodig zijn.

Tabel 3: Planning

Taak	week																Uren
	Februari			F&M	Maart			M&A	April				Mei				M&J	
	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	Totaal
Onderzoek							DL				DL			DL			DL	284
Plan van aanpak	32	28	16															76
Onderzoek huidige situatie				30														30
Onderzoek beschikbare frameworks					30													30
Onderzoek vergelijken frameworks						30												30
Scriptie							4	8	8	8	8	8	8	10	10	30	16	118
Proof of concept:									OP		OP		OP		OP			269
Voorbereiding				6	6	6	32											50
Sprint 1								37	30									67
Sprint 2										28	20							48
Sprint 3												19	30					49
Sprint 4														28	27			55
Overige Activiteiten																		110
Introductie	8	8																16
Andere taken voor werkgever		4	19	4	4	4	4	2	2	2	2	2	2	2	2	2		57
Vakantiedagen											8	8				8		24
Leerteam			3									3						6
Rockstar IT			2							2	2				1			7
Uren per week	40	40	40	40	40	40	40	47	40	40	40	40	40	40	40	40	16	663
Uren totaal	40	80	120	160	200	240	280	327	367	407	447	487	527	567	607	647	663	

legenda: Deadline Oplevering

De deadlines en de laatste oplevering die aangegeven worden in Tabel 3 staan hieronder nog eens op een rijtje:

24 Maart (week 12), Inleveren afstudeercontract.
18 April (week 16), Inleveren eerste scriptie concept.
8 Mei (week 19), Inleveren tweede scriptie concept.
19 Mei (week 20), Oplevering proof of concept.
30 Mei (week 22), Inleveren eindversie scriptie.

5. Theoretisch kader

In dit hoofdstuk worden de belangrijkste begrippen en afkortingen van dit onderzoek toegelicht.

5.1. Begrippen en Afkortingen

Begrip	Beschrijving
GIT	Git is een versie management system die veelal bij software gebruikt wordt om een historie aan veranderingen bij te houden en om met meerdere ontwikkelaars aan een project te kunnen werken.
Codebase	Codebase is een platform waarop GIT repositories gehost kunnen worden en met behulp van tickets bugs en features bijgehouden kunnen worden van de applicaties waaraan gewerkt worden.
API	Een API is, zoals het woord in dit onderzoek gebruikt wordt, een referentie naar RESTful web API's. Deze wordt door Barre Dijkstra beschreven als volgt: "REST staat voor Representational State Transfer. Het is een manier om webservices te creëren op basis van de bestaande en eenvoudige bouwstenen van het internet. Deze maakt gebruik van URL's voor adressering en de HTTP methodes (GET, POST, DELETE en PUT) voor het aanroepen van de service." (Dijkstra, 2005)
SDK	SDK staat voor Software Development Kit. Een SDK bevat vaak een set aan tooling die gebruikt kan worden om een applicatie te maken die een bepaald framework of platform ondersteund.
Cross-platform	Met cross-platform wordt de mogelijkheid van applicaties die op meerdere platformen werken bedoelt. Dit is mogelijk in de vorm van native of hybride apps.
Framework	Application frameworks zijn sets of tools bovenop een programmeertaal, ontworpen om het werk van een developer te vereenvoudigen. (Code-on, sd)
OVplaza	OVplaza is een platform waarop ov-informatie te vinden is in de vorm van reisplanning, actuele vertrektijden en meer. OVplaza is een onderdeel van Infoplaza.
Weerplaza	Weerplaza is een platform waarop weersinformatie te vinden is in de vorm van weersverwachtingen, een regenradar, nieuwsartikelen en meer. Weerplaza is onderdeel van Infoplaza.
Verkeerplaza	Verkeerplaza is een platform waarop verkeersinformatie te vinden over files zoals de lengte, locatie en vertragingstijd. Verkeerplaza is onderdeel van Infoplaza.
Scrum	Dit is een werkmethode op basis van iteratief werken door middel van verscheidene sprints. Een sprint is een periode van bijvoorbeeld 2 weken waarin besloten wordt een bepaalde set aan functionaliteiten te gaan realiseren. Het doel is om veel contactmomenten met de eigenaar van het eindproduct te hebben waardoor er doelgericht ontwikkeld kan worden en aanpassingen snel gemaakt kunnen worden.
Mocking	Bij mocking wordt er een 'dummy' object gemaakt van een object zodat getest kan worden of bepaalde functies correct aangeroepen worden.
UI	UI is een afkorting voor 'User Interface', waarmee naar het uiterlijk van de app gerefereerd wordt.
Hybride Apps	Hybride apps zijn apps die gebruik maken van web-based technologieën zoals HTML en JavaScript om een app te maken die eenvoudig op meerdere platformen werkt.
Native Apps	Mobiele apparaten bevatten een Software Development Kit (SDK). Android heeft een Java SDK, iOS heeft Swift en Objective-C SDK's en Windows Phone

	heeft een .NET SDK. Apps die gemaakt zijn voor deze apparaten in de bijbehorende taal worden ook wel native apps genoemd.
PropertyCross	PropertyCross is een casestudy van een app voor het kopen van woningen, die op een groot aantal frameworks is geimplementeerd.
FPS	FPS is het aantal frames per seconde dat getoond wordt.

6. Onderzoek

In dit hoofdstuk wordt beschreven welke methoden er gebruikt zijn voor het onderzoek, wat de resultaten waren van de uitvoering van het onderzoek en welke conclusies er getrokken kunnen worden. Aan het einde volgt de aanbeveling die naar Infoplaza is gedaan, en hoe deze de verdere verloop van het bouwen van de proof of concept heeft beïnvloed.

6.1. Methoden

De volgende paragrafen beschrijven de methodes die zijn toegepast bij het beantwoorden van de onderzoeksvragen. In tabel 2 is te zien welke methoden er voor de deelvragen zijn gebruikt.

Tabel 4: Onderzoeksmethoden per deelvraag

#	Deelvraag	Onderzoeksmethoden
1	*Hoe worden de huidige mobiele applicaties van Infoplaza gerealiseerd?*	Veldonderzoek, Interviews
2	*Welke frameworks zijn er beschikbaar die cross-platform capaciteiten bieden voor mobile development?*	Deskresearch
3	*Wat zijn de voor- en nadelen van de beschikbare frameworks voor cross-platform applicaties in verhouding tot elkaar?*	Vergelijkend

6.1.1. Interviews

Voor het onderzoek naar de huidige situatie bij Infoplaza zijn er interviews afgenomen met de interne ontwikkelaars van de mobiele applicaties en de projectmanagers. Hierin is gevraagd naar de duur van projecten volgens de huidige realisatiemethoden en wat zij verwachten dat er voor- of nadelig zal veranderen bij het gebruik van een cross-platform framework.

6.1.2. Veldonderzoek

Voor het onderzoek naar de huidige manier van realiseren is vooral gebruik gemaakt van interviews. Middels deze interviews is in kaart gebracht hoe de huidige structuur voor het realiseren van applicaties in elkaar steekt.

6.1.3. Deskresearch

Om te bepalen welke cross-platform frameworks er beschikbaar zijn is er middels literatuur gezocht naar frameworks. Zodra er een nieuwe naam naar voren kwam is de project pagina van dit framework opgezocht en is er algemene informatie verzameld over de app zoals de laatste release datum, de eigenaar, de programmeertaal en de ondersteunde platformen.

6.1.4. Vergelijkend onderzoek

In dit onderzoek is de data van het zoeken naar beschikbare frameworks verder uitgebreid met details die volgens de interviews uit het onderzoek naar de eerste deelvraag naar voren kwamen. Hierna is er per framework punten toegekend voor alle eigenschappen die het framework bezat die in het eerste deelonderzoek als gewenst werden beschouwd. Deze eigenschappen komen terug in paragraaf 6.2.3.

6.2. Resultaten

In de volgende hoofdstukken wordt uitgelegd welke resultaten naar voren kwamen bij de uitvoering van het onderzoek zoals in hoofdstuk 6.1 beschreven is.

6.2.1. Onderzoek huidige situatie

Op basis van de afgenomen interviews is er een beeld opgebouwd over de huidige situatie voor het maken van mobiele applicaties binnen Infoplaza. Hiermee wordt een antwoord gegeven op de vraag "*Hoe worden de huidige mobiele applicaties van Infoplaza gerealiseerd?*".

Interviews

De iOS en Android apps voor de apps van Weerplaza en Verkeerplaza worden ontwikkeld door twee ZZP'ers die allebei voor een apart platform ontwikkelen. Infoplaza bezit ook als zusterorganisatie Moop Mobility, wiens apps uitbesteed worden aan een externe partij genaamd Appnormal.

In beide gevallen wordt er samengewerkt met de designers van Infoplaza om een universele look te behalen op Android en iOS die past bij de huisstijl van Infoplaza.

Een van de eerste resultaten uit het interview die opviel was dat er intern vooral iOS kennis in huis aanwezig was. De eerste apps die binnen de organisatie ontwikkeld waren, waren alleen op iOS beschikbaar. Toen Android eenmaal in beeld kwam werd dit uitbesteed, waardoor de kennis onder de vaste werknemers over Android lager is ten opzichte van iOS.

Hoewel er duidelijke communicatie is over de functionaliteiten komt het weleens voor dat de apps afwijkende eigenschappen ontwikkelen, bijvoorbeeld omdat de ontwikkelaar zijn eigen beeld heeft bij de opgegeven functionaliteit of dat deze niet op dezelfde manier haalbaar is door limitaties op het platform.

Wat wel als cruciaal wordt gezien is dat de apps veel zware grafische functionaliteiten gebruiken, vooral voor de weer apps. Deze laten vaak kaarten zien waar informatie bovenop getoond wordt, iets wat volgens Sjoerd erg zwaar was, wat hij in zijn interview ook noemde: "Dat zijn echt dingen waar we zeker, 2 jaar geleden toen we begonnen, tegen aan liepen: de limieten van de toestellen."

Ook wordt er gesproken over de uitdaging van het behouden van een huisstijl. Door de verschillende design standaarden van iOS en Android is het lastig een uniforme stijl aan te houden.

De volledige interviews zijn terug te vinden in het onderzoeksrapport. (Bijlage 11.2)

6.2.2. Onderzoek bestaande frameworks

Voor het tweede onderzoek is er uitgezocht welke frameworks er aangeboden worden op de markt. Door gebruik te maken van de framework vergelijkingssite PropertyCross (Eberahrdt & Price, 2016) is er een lijst samengesteld van beschikbare frameworks. De informatie over de frameworks zijn al aardig verouderd en sommige frameworks bestaan niet meer, maar ondanks dat bevat deze site de meest complete lijst aan frameworks die gevonden kon worden. Ook zijn nog enkele los gevonden frameworks toegevoegd die voor het ontdekken van PropertyCross gevonden waren. De frameworks die niet meer bestonden of overduidelijk gestopt zijn met ontwikkelen zijn weggelaten uit het onderzoek, omdat deze geen perspectieven bieden om mee te kunnen gaan ontwikkelen.

Daarnaast kwam uit de interviews voor het onderzoek van de eerste deelvraag regelmatig naar voren dat mensen bij hybride cross-platform frameworks last hadden van of gehoord hadden over slechte performance. Omdat de performance een belangrijke eis is aan het framework, zijn veel hybride frameworks hierdoor weggelaten uit verdere analyse in dit onderzoek. Als uitzondering is Cordova wel nog opgenomen in de vergelijking ter verifiëring van deze hypothese tijdens het benchmarken in het vergelijkeng onderzoek.

Antwoord deelvraag

Het resultaat van het filteren van de frameworks gevonden op PropertyCross is de volgende lijst met frameworks:

Tabel 5: Frameworks

Framework	Programmeertaal	Type (Native of Hybride)
Xamarin	C#	Native, Hybride
Adobe AIR	ActionScript, Flash	Native
Crosslight	C#	Native
Delphi	Object Pascal	Native
NativeScript	JavaScript, TypeScript	Native
Qt	C++	Native
RhoMobile	JavaScript, Ruby	Native
Titanium	JavaScript	Native
React Native	JavaScript	Native
NeoMAD	Java	Native
Codename One	Java	Native
Kivy	Python	Native
Cordova/Phonegap	JavaScript	Hybride

6.2.3. Onderzoek vergelijking frameworks

Als eerste deel van dit onderzoek zijn de resultaten van de vorige deelvraag verder uitgebreid met details over de frameworks. Deze frameworks zijn hierna vergeleken met elkaar en de resultaten hiervan zijn te zien in Tabel 7.De puntenverdeling heeft plaatsgevonden op basis van de beoordelingscriteria in Tabel 6.

Tabel 6: Beoordelingscriteria

Eigenschap	Reden
Datum laatste release (levensduur)	De datum wordt gebruikt in het onderzoek om te bepalen hoe actief het project is. Als deze datum in 2017 viel werd er een punt bijgerekend.
Eigenaar (levensduur)	De reputatie van de eigenaar kan een belangrijke factor zijn in het maken van een framework keuze voor een organisatie. Als de organisatie een grote marktspeler is, is daar een tweede punt voor uitgereikt.
Hybride of native	Het verschil in achterliggende techniek tussen native en hybride. Er is een punt gegeven als het framework native is.
Programmeertaal	De programmeertaal is belangrijk in verband met de huidige kennis in de organisatie. Er is een punt gegeven als de programmeertaal bekend is binnen de organisatie.
Tooling	De beschikbaarheid van tooling als een IDE-omgeving of een CLI kan groot effect hebben op het gebruiksgemak van het framework. Als het framework meer biedt dan alleen een CLR of een plugin voor een IDE wordt er een punt toegekend.
Prijs	De prijs is een belangrijke factor voor het management van de organisatie. Frameworks die gratis zijn hebben een volledig punt gekregen. Frameworks die geld kosten maar wel gratis pakketten aanbieden hebben een half punt gekregen.
Support	De manier waarop support geleverd wordt heeft grote invloed op hoe snel problemen tijdens het ontwikkelen opgelost kunnen worden. Als er professionele support geleverd wordt is er een punt toegekend.

Tabel 7: Beoordelingsmatrix

Naam	Totaal	Levensduur	Native	Taal	Prijs	Tooling	Support
Xamarin	6,5	++	+	+	~	+	+
NativeScript	6	++	+	+	+	-	+
React Native	5	++	+	+	+	-	-
Cordova / Phonegap	5	++	-	+	+	+	-
Titanium	5	+	+	+	-	+	+
Qt	4,5	+	+	-	~	+	+
RhoMobile	4	-	+	+	+	+	-
AIR	4	+	+	-	-	+	+
NeoMAD	3,5	-	+	+	~	-	+
Codename One	3,5	+	+	-	~	-	+
Crosslight	3	-	+	+	-	-	+
Kivy	3	-	+	+	+	-	-
Delphi	3	-	+	-	-	+	+

+ = 1 punt, - = geen punt, ~ = 0.5 punt.

Benchmark

Uit deze lijst met frameworks is een selectie gemaakt van 4 frameworks die in een benchmark getest zullen worden:

- Xamarin
- NativeScript
- React Native
- Cordova

Xamarin en NativeScript zijn geselecteerd op basis van hun puntenscore. Deze hebben de hoogste en een na hoogste score behaald, en zijn daardoor als waarschijnlijk geschikte kandidaten opgenomen.

Daarnaast is React Native opgenomen. Hoewel React een derde plaats deelde met Titanium, is er wegens naambekendheid gekozen voor React. De naamsbekendheid van het framework en de achterliggende organisatie, Facebook, heeft als voordeel dat er meer gebruikers zijn die al veel mogelijke problemen online besproken hebben en dat er voor de ondersteuning van het platform meer resources zijn dan bijvoorbeeld bij een kleinere organisatie.

Als laatste is Cordova ook onderdeel geworden van de selectie om een hybride framework te kunnen vergelijken met zijn Native tegenhangers.

Omdat er bij de afstudeerder nog geen kennis was voor het uitvoeren van een benchmark is er gebruik gemaakt van een beschikbare benchmark van NativeScript. Hierdoor bestaat de mogelijkheid dat deze voordeel biedt voor hun framework. Het project is echter open-source, waardoor ervan uit is gegaan dat deze code geen onregelmatigheden bevat die een oneerlijk voordeel kunnen bieden voor een van de geteste frameworks.

De resultaten van de benchmarks zijn te zien in Tabel 8 en Tabel 9. Hierin zijn de gemiddelden van alle tests opgenomen.

Tabel 8: Gemiddelde - Simulator iPhone 7, iOS 12.2.1 op een MacBook Pro 13" (2015)

X	Native	Xamarin	React Native	NativeScript	Cordova
Primitives	9ms	341ms	33374ms	538ms	18765ms
Strings	30ms	231ms	37237ms	124ms	47226ms
Big Data Marshalling	271ms	901ms	34617ms	661ms	53822ms

Tabel 9: Gemiddelde - iPhone 5s, iOS 12.2.1

X	Native	Xamarin	React Native	NativeScript	Cordova*
Primitives	5ms	31ms	43651ms	1252ms	0
Strings	39ms	241ms	49152ms	471ms	0
Big Data Marshalling	314ms	2059ms	46615ms	1613ms	0

* Cordova crashte iedere test met de melding: "terminated due to memory issue"

6.3. Conclusie

In de vorige paragraaf zijn de resultaten gegeven van het onderzoek. Hiervan bieden de resultaten van de eerste twee deelvragen informatie die samenkomt in het onderzoek naar de derde deelvraag. Uit de resultaten van het onderzoek naar de frameworks kwam Xamarin het beste uit de bus bij de vergelijking tussen eigenschappen van het framework. Uit de benchmark die daar op volgde bleek dat de prestaties van Xamarin en NativeScript het beste waren ten opzicht van Native code. Deze platformen bieden goede prestaties, een bekende programmeertaal en premium support.

De hoofdvraag “Op welke manier zou een cross-platform framework voor mobiele applicaties ingezet kunnen worden voor de apps van Infoplaza, zodat er minder code gedupliceerd wordt over de verschillende platformen?” kan nu beantwoord worden met:

Door het inzetten van het cross-platform framework Xamarin of NativeScript. Deze frameworks bieden capaciteiten om gedeelde business logica te benutten in de applicaties en bieden allebei de mogelijkheid om ook gedeelde User Interface code te gebruiken.

6.4. Advies

Op basis van de conclusie kwam de keuze voor de aanbeveling te liggen tussen Xamarin en NativeScript. Er is hier uiteindelijk gekozen voor Xamarin op basis van voordelen boven NativeScript voor de organisatie:

- Xamarin is inbegrepen bij Visual Studio, een tool die binnen de organisatie al beschikbaar is middels een Microsoft Developer Network licentie.
- Xamarin maakt gebruik van C#, een taal die het meest gebruikt wordt binnen de infrastructuur van de organisatie.
- Xamarin Test Cloud, deze toolset die geleverd wordt door Xamarin biedt de mogelijkheid om snel en eenvoudig de UI en prestaties van een applicatie op een groot scala aan apparaten te testen. Dit platform is echter ook te gebruiken voor frameworks die los staan van Xamarin.
- Xamarin biedt Xamarin University aan, een leerplatform voor hun framework. Xamarin University biedt een groot pakket aan cursussen en seminars aan. Ook is het mogelijk om een certificering als ‘certified mobile developer’ te behalen binnen dit platform.

Er is bij het gebruik van Xamarin gekozen voor Xamarin.Forms voor het schrijven van de interface. Xamarin.Forms is een extensie van het Xamarin framework die het mogelijk maakt om middels gedeelde code een interface te schrijven die native is voor Android en iOS.

Dit advies is middels het onderzoeksrapport aan de bedrijfsbegeleider overgedragen, die zijn akkoord heeft gegeven om bij het bouwen van de proof of concept applicatie Xamarin te gebruiken. Op basis van deze keuze is er in de week voorafgaande aan de eerste sprint een trial cursus op Xamarin University gevolgd om de afstudeerder voor te bereiden op de proof of concept.

7. Proof of Concept

In de volgende paragrafen wordt toegelicht welke methoden er zijn gebruikt voor het bouwen van de proof of concept en wat voor resultaten de proof of concept opgeleverd heeft.

7.1. Methode

In deze paragraaf wordt beschreven welke ontwikkelmethodes toe zijn gepast bij het bouwen van de proof of concept applicatie.

7.1.1. Scrum

Voor de ontwikkelfase van dit project is er gebruik gemaakt van de Scrum methodiek volgens de scrumguide (Schwaber & Sutherland, 2016). De sprints en backlogs worden bijgehouden in Codebase, waar ook de code wordt bijgehouden. Er was gekozen voor sprints van 2 weken zodat er snel en iteratief functionaliteit opgeleverd kon worden.

7.1.2. Unit testen

Om de code kwaliteit van de app te kunnen garanderen zullen er unit tests geschreven worden. Deze tests zorgen ervoor dat als er grote wijzigingen in de toekomst gedaan worden deze niet onopgemerkte fouten kunnen bevatten. Hiervoor is gebruik gemaakt van NUnit voor de tests en Moq voor mocking.

Daarnaast is een unit test in zeker mate ook een vorm van documentatie van de geschreven code. De unit test beschrijft namelijk hoe een functionaliteit zou moeten werken.

7.1.3. UI testen

Ter uitbreiding van de Unit code is er ook gekozen om UI tests te schrijven. Normale unit tests worden vaak gebruikt om iedere functie in je code los te testen. Middels een UI test loop je de lifecycle van de applicatie door en controleer je of alle pagina's de inhoud tonen die je verwacht wanneer deze doorlopen wordt.

7.2. Resultaten

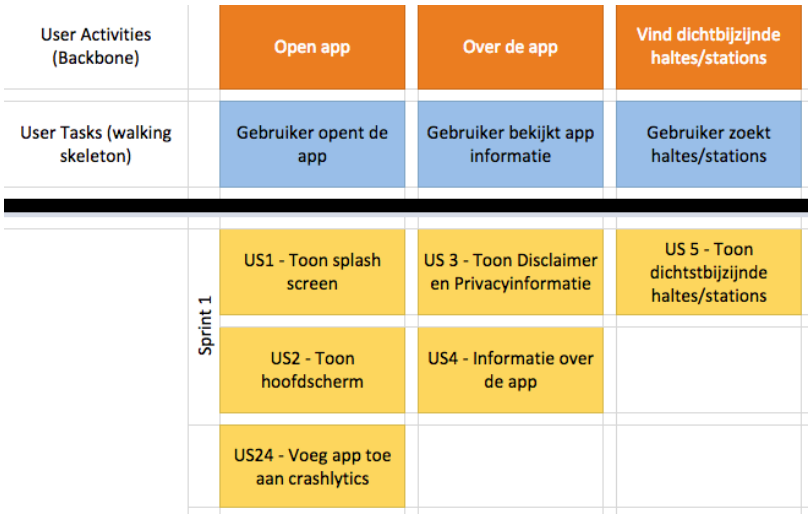
In de volgende paragrafen wordt de loop van het bouwen van de proof of concept applicatie beschreven. Per sprint wordt er toegelicht welke taken er volbracht zijn en welke keuzes er gemaakt zijn. Er is overwogen om hierbij ook de sprint burndown charts te laten zien maar deze zijn niet toegevoegd omdat deze een incorrect beeld tonen van het verloop van de sprint. De burndown charts werden automatisch gemaakt op basis van de tickets in codebase, maar deze hield geen rekening met weekenden waardoor soms vlakke lijnen ontstonden. Ook werden onderdelen als de prioriteit en de punten schatting van de taken niet meegenomen in de burndown chart.

7.2.1. Voorbereiding

In de voorbereiding van het bouwen is er een trial cursus gevolgd op Xamarin University. Dit is een trainingsplatform van Xamarin met verschillende lessen om alle aspecten van Xamarin te leren. De trial gaf toegang tot 11 lessen die de basis behandelde van Xamarin.Android, Xamarin.iOS, cross-platform development en Xamarin.Forms. Deze cursus is gevolgd gedurende een periode van 1 week.

In dezelfde week is er een story board opgesteld voor de applicatie. Deze is gemaakt op basis van gesprekken met de Product Owner. Deze user stories zijn opgedeeld in backlog items om te kunnen indelen in de sprints.

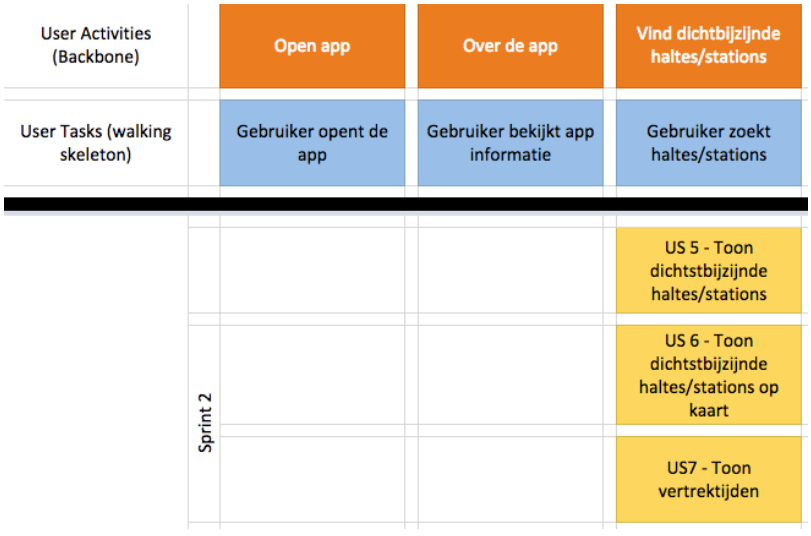
7.2.2. Sprint 1



Figuur 3: Storyboard sprint 1

In de eerste sprint moest de eerste versie van de app neergezet worden. Dit is terug te zien in de user stories op het storyboard in Figuur 3. Hiervoor is er eerst een schil ontwikkeld waarin de basis UI-elementen zoals navigatie, splash screens, 'about' pagina en de homepagina geïmplementeerd waren. Ook is Fabric's Crashlytics op verzoek van de opdrachtgever toegevoegd aan de applicatie waardoor er crash rapporten gestuurd kunnen worden, gebruiksdata bekeken kan worden en er bèta releases naar een selecte groep gebruikers gedistribueerd kunnen worden. In deze sprint was het doel "Controleren of Xamarin de mogelijkheden biedt die nodig zijn". Dit doel was achteraf gezien nog niet met deze gekozen functionaliteit te behalen, maar er was wel middels de gebouwde app aan te tonen dat Xamarin de mogelijkheid biedt om Android en iOS tegelijk de distribueren.

7.2.3. Sprint 2



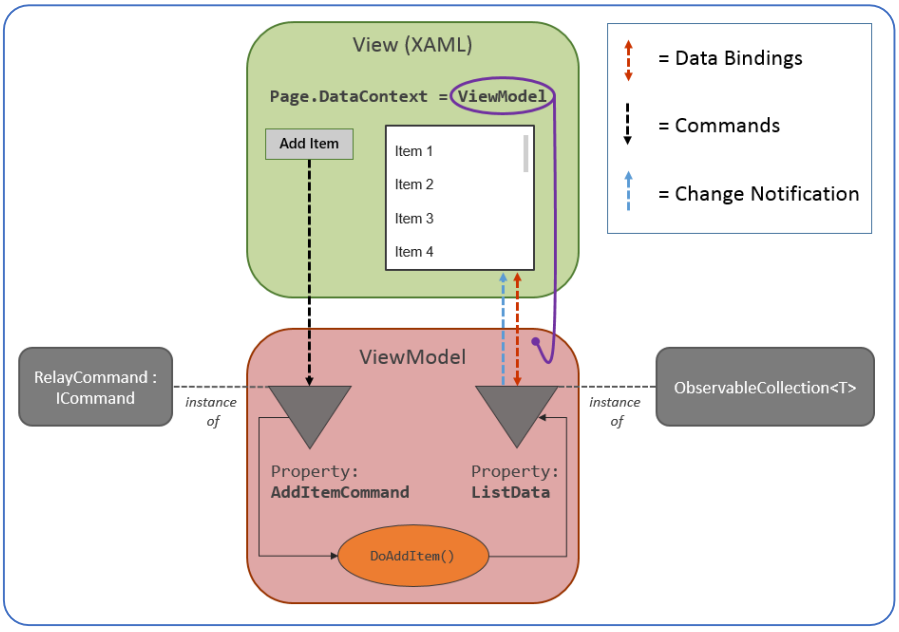
Figuur 4: Storyboard sprint 2

In deze sprint is er veel tijd besteed aan het correct implementeren van een lijst met nabije haltes. Deze lijst werd eerst gemaakt op basis van een ListView waarvan de bron een list was met objecten die properties van de header onderdelen én item onderdelen bevatte. Deze aanpak leverde echter een complexe implementatie waarbij de lijst gevuld werd via de code-behind middels een grote for loop constructie met verschillende if-statements.

Daarom werd er een andere aanpak geprobeerd. Hierbij werd er gebruik gemaakt van de grouping functionaliteit van de Xamarin ListView. Daarbij is de bron een lijst met objecten die extensies zijn op een lijst. Deze objecten kunnen daardoor data voor de grouping headers bevatten, waarna de inhoud van de list die ze erfden onder de grouping gesorteerd wordt. Deze aanpak leverde visueel een goed resultaat op, maar leverde problemen op omdat de group header niet klikbaar was wat wel nodig was voor de lijst. Ook werd de lijst nog steeds gevuld via de code-behind wat een grote hoeveelheid code opleverde.

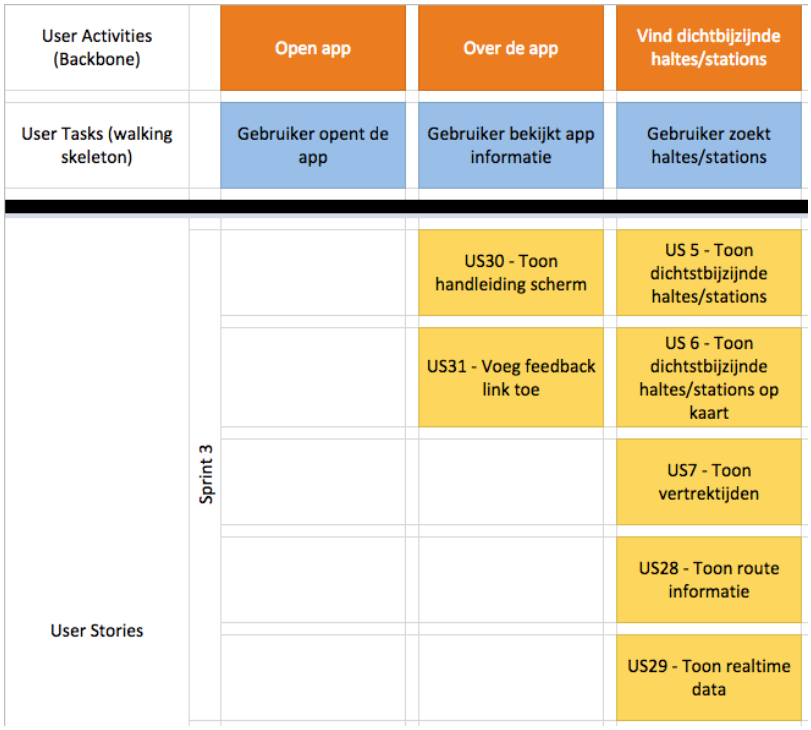
Omdat dit ook geen goeie aanpak bleek, is er op advies van de begeleider overgestapt op het gebruik van een TableView. Deze werkt vrij gelijk aan een ListView, maar wordt niet gevuld via een ItemSource maar deze werkt volledig handmatig via de control view tree. Visueel leverde dit weer een goed resultaat, maar de code om deze lijst te vullen bracht significante performanceproblemen met zich mee, waardoor de iOS app door het systeem na korte tijd afgesloten werd.

Na al deze pogingen is er een nieuw ontwerp gemaakt voor de code middels het MVVM pattern (zie Figuur 5) van Microsoft, die ook gepromoot wordt in het gebruik van Xamarin Forms. Na een halve dag werk was er een ontwerp gerealiseerd van 3 viewmodellen die de logica opsplitsten tussen de UI van de nearby pagina, de halte cel en de vertrektijden cel. Het overzicht was weer een listview geworden die nu met een collectie van haltes werd gevuld. Deze haltes bevatte per stuk weer een collectie vertrektijden die onder de haltes werden geplaatst. Het eindresultaat was een UI en een model die 'loosely coupled' zijn, dynamisch data kunnen verversen en makkelijk uit te breiden zijn.



Figuur 5: XAML MVVM

7.2.4. Sprint 3

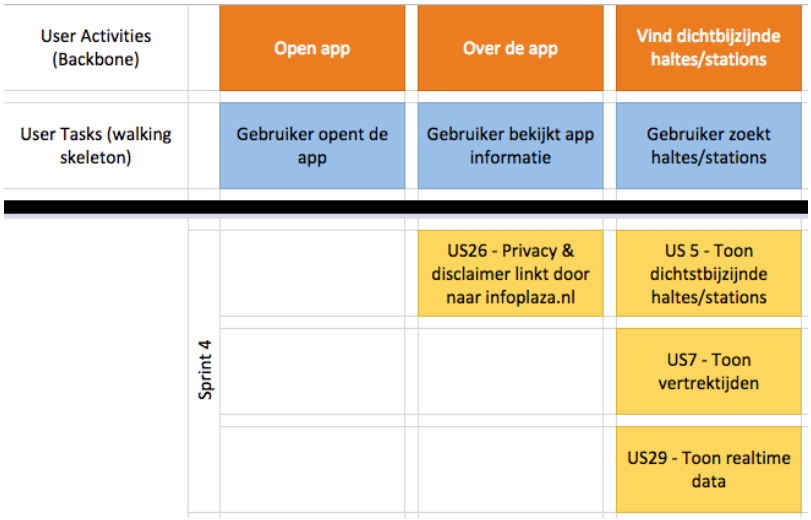


Figuur 6: Storyboard sprint 3

Tijdens sprint 3 is er vooral ontwikkeld aan de lijst met nabije haltes en de kaart die deze locaties toont. Er waren ook andere taken toegevoegd aan de backlog zoals te zien in Figuur 6, maar deze zijn grotendeels doorgeschoven naar de volgende sprints. Een probleem waar tijdens deze sprint tegen aan gelopen werd was dat de user stories opgesplitst waren in te grote taken. Hierdoor was de voortgang van de sprint heel lastig te meten omdat er gedurende de sprint amper taken afgerond konden worden. Uiteindelijk is er na opsplitsing van de bestaande taken in kleinere taken een lijst ontstaan van 24 taken, waarvan er 4 volledig afgerond zijn. Deze taken hadden voor een merendeel 8 scrumpunten toegewezen gekregen, wat aangaf dat dit complexe taken waren. Na de opsplitsing waren er dus duidelijk te veel taken ingepland in de sprint.

De afgeronde taken aan het einde van de sprint betreffen het toevoegen van de kaart aan de homepagina, het uitbreiden en visueel verbeteren van de lijst met nabije haltes en de toevoeging van een halte pagina, waarop alle vertrektijden van 1 halte getoond worden.

7.2.5. Sprint 4



Figuur 7: Storyboard sprint 4

In de 4e en daarmee laatste sprint voor de proof of concept applicatie is er gefocust op het afronden van gemaakte functies en het opschonen van de UI zodat er geen visueel onafgeronde componenten meer in de app voorkomen.

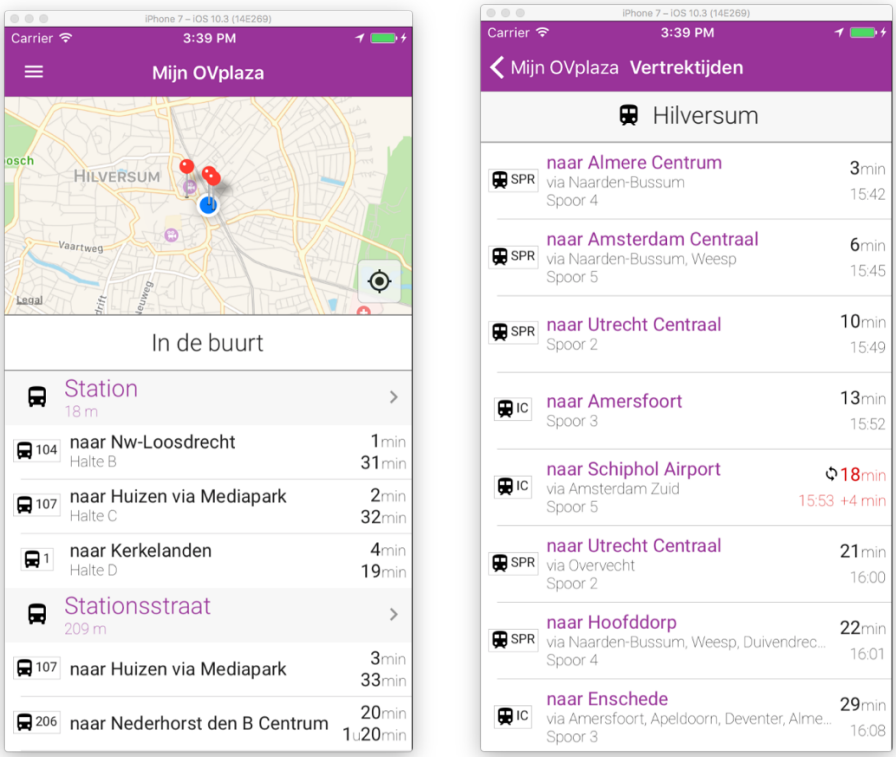
Hiervoor was een van de complexere vraagstukken het toevoegen van geclusterde vertrektijden onder elkaar voor dezelfde lijn bij een halte op het hoofdscherm van de app. Xamarin Forms heeft namelijk maar één control die een collectie van objecten als bron kan ontvangen en dat is de listview. Om deze functie toe te voegen moest er een component gemaakt worden die op basis van een collectie aan objecten een view kan herhalen. Uiteindelijk is deze constructie gelukt door het uitbreiden van een StackLayout. Deze bevat 3 bindable properties: een ItemSource voor de collectie met data vanuit een viewmodel, een DataTemplate voor de objecten uit de collectie die geclusterde tijden heeft en een template voor de objecten die niet-geclusterde vertrektijden bevatte. Per object in de collectie werd gekeken of deze geclusterd was of niet, waarna het correcte datatemplate werd gebruikt om een nieuwe child aan de StackLayout hiërarchie toe te voegen. Met behulp van deze custom StackLayout zijn uiteindelijk de geclusterde departures gerealiseerd.

Een ander belangrijk onderdeel wat toegevoegd is tijdens deze sprint zijn unit- en UI-tests. De unit tests zijn geschreven voor de business logic code die de domein objecten beheert en de API-calls afhandelt. Deze code staat ook in een apart project omdat deze code bedoeld is om ook in andere projecten bruikbaar te zijn. De UI Tests zijn geschreven voor de doorloop naar iedere pagina die beschikbaar is in de applicatie.

De unit tests zijn alleen geschreven voor de business logic omdat na overleg met collega's aangegeven werd dat er geen behoefte was aan het onderhoud dat nodig zou zijn om de unit tests up-to-date te houden. Daarnaast zou het schrijven van meer tests ook hinderen in het afronden van de functionele eisen aan de app.

8. Afsluiting

Met behulp van het framework dat uit het onderzoek naar voren is gekomen, Xamarin, is de OVplaza app gemaakt zoals te zien is in Figuur 8.



Figuur 8: Hoofdscherm en vertrektijden scherm OVplaza app

In hoofdstuk 6.3 is een antwoord gegeven op de hoofdvraag. Met behulp van de Proof of concept applicatie die gemaakt is op basis van het antwoord van de hoofdvraag kan bevestigd worden of dit antwoord en de daar op volgende aanbeveling correct zijn. Door te kijken of de in hoofdstuk 3.1 gestelde doelstellingen gehaald zijn kan geconcludeerd worden of het Xamarin Framework inderdaad voldoet aan de wensen van Infoplaza.

8.1. Doelstellingen

De applicatieontwikkeling loopt gelijk voor Android en iOS.

De applicatie kon iedere sprint in praktisch gelijke staat voor Android en iOS opgeleverd worden. Er zijn wel enkele problemen die platform specifiek waren, zoals een bug op Android waarbij de labels verdwijnen wanneer de telefoon vergrendeld werd met de app open en daarna weer ontgrendeld wordt. Ondanks deze bugs kon er echter iedere sprint een gelijke app getoond worden op ieder platform.

Het onderhoud aan de applicatie, bijvoorbeeld het toevoegen van een nieuwe functionaliteit of het oplossen van een bug, moet minder tijd kosten dan wanneer dit native zou gebeuren door twee aparte ontwikkelaars.

Toen deze Doelstelling neergezet werd was eigenlijk al duidelijk dat deze lastig te bewijzen zou zijn. Ondanks dat is deze vraag behouden vanwege de relevantie en de impact van het vraagstuk. Er spelen echter veel factoren mee die de vergelijking moeilijk dan wel onmogelijk maken, zoals het verschil in ervaring tussen de afstudeerder en de huidige ontwikkelaars van de mobiele applicaties

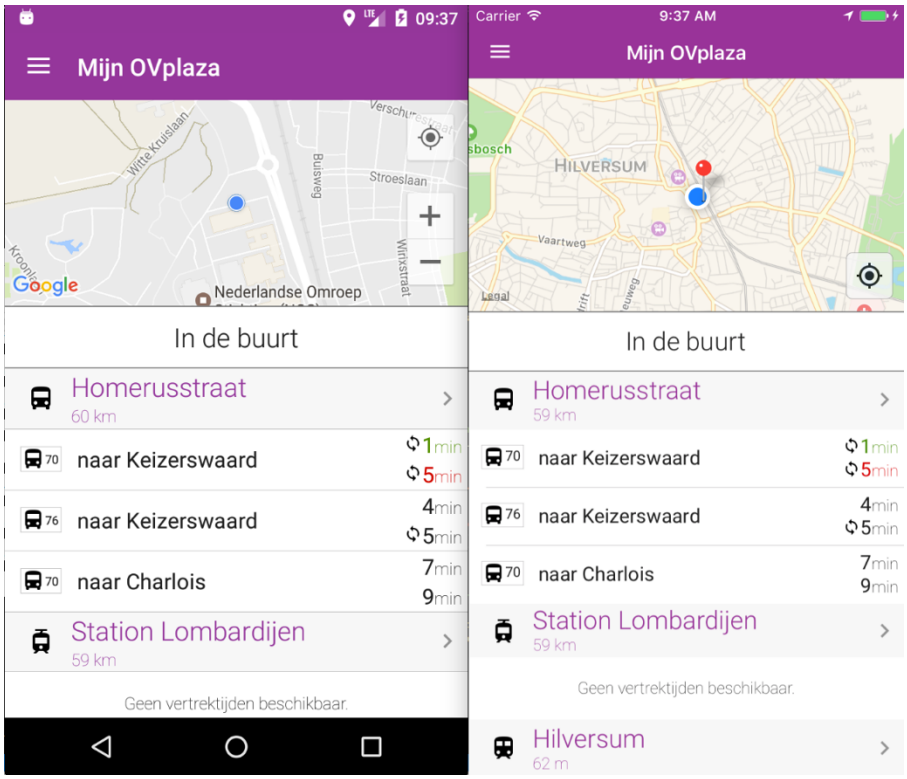
bij Infoplaza. Er is tijdens het eerste deelonderzoek gevraagd hoe lang de huidige ontwikkelaars zouden doen over het realiseren van een lijst met nieuwsitems en een scherm zodra een item geklikt wordt. Dit voorbeeld is echter niet te vergelijken met de ontwikkeling van OVplaza vanwege de componenten die er bij zijn komen kijken om bijvoorbeeld een lijst te maken met departures. Er kan daardoor geen conclusie getrokken worden of deze doelstelling is gehaald of niet.

De applicatie kan op Android en iOS gezamenlijk gelanceerd worden.

De applicatie is gedurende de loop van het ontwikkelen regelmatig naar testers gedistribueerd. De stap om dan een publiekelijke lancering te houden is erg klein. Daarnaast koppelt deze vraag terug naar de eerste doelstelling omdat de parallelle ontwikkeling de gezamenlijke distributie mogelijk maakt.

De interface van de applicatie biedt dezelfde huisstijl op Android en iOS.

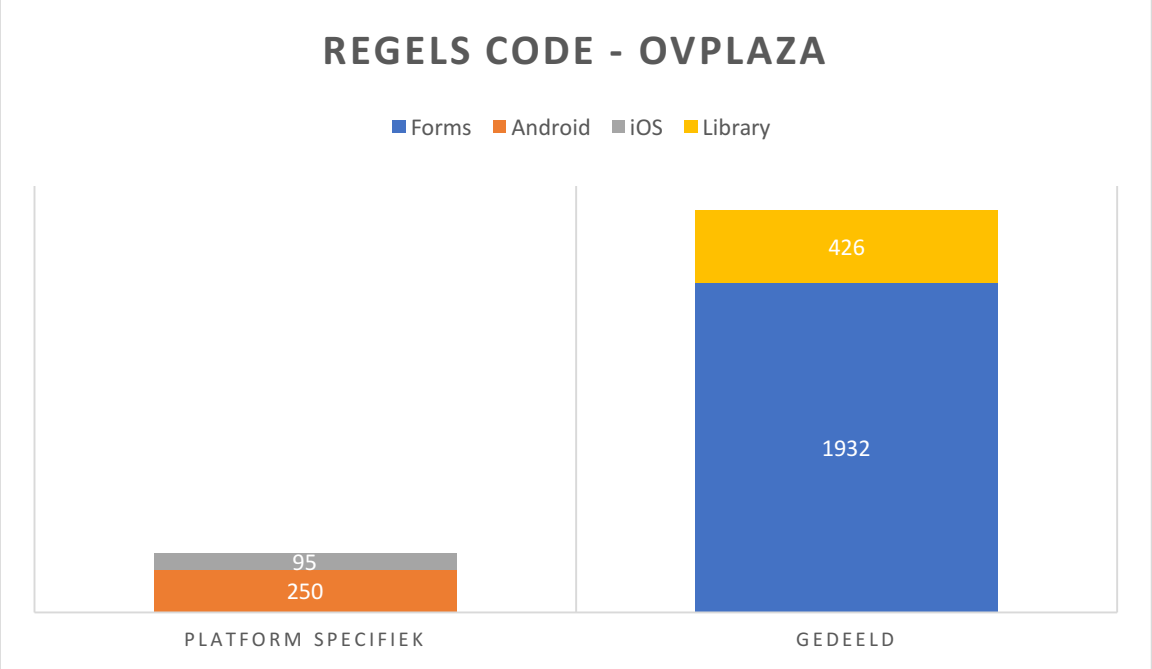
Zoals in Figuur 9 te zien is, dragen beide platformen zeker dezelfde huisstijl. Visueel lijken de apps ook heel veel op elkaar, maar platform specifieke verschillen blijven behouden zoals het scrollen, de visuele eigenschappen van knoppen en de animaties waardoor de apps native aanvoelen op zowel iOS als Android.



Figuur 9: Android en iOS naast elkaar

De applicatie maakt gebruik van een gedeelde codebase voor de business logic.

De applicatie maakt gebruik van een library die los van de OVplaza app projecten is geschreven, die het mogelijk maakt de business logic niet alleen onder de apps maar ook hierbuiten te hergebruiken. Deze business logic betreft de modellen voor de vertrektijden en de API-interactie. Naast de logic is ook de UI grotendeels gedeeld over de verschillende platformen, zoals te zien is in Figuur 10.



Figuur 10: Regels code van OVplaza

De impact van systeemupdates van Android en iOS moet minimaal zijn voor het gebruik van het framework.

Deze vraag is wat minder concreet te beantwoorden. Xamarin heeft een goede reputatie voor het updaten van zijn framework bij Android en iOS systeemupdates. Gemiddeld was er binnen enkele dagen of zelfs op de dag zelf een nieuwe release die ondersteuning biedt voor de nieuw beschikbare API's. Hiermee is er duidelijk een minimale impact van systeemupdates.

De applicatie behaalt een goede GPU-performance op de verschillende platformen van gemiddeld rond de 60 FPS.

Om dit te kunnen bevestigen is er een benchmark uitgevoerd van de app op drie apparaten: een Oneplus One, een Galaxy S7 Edge en een iPhone 5s. Hierbij is de app 3 minuten gebruikt waarbij de framerate is gemeten. De resultaten zijn als volgt:

Tabel 10: FPS Benchmark

Telefoon	OS Versie	Gemiddelde framerate	FPS stabiliteit
iPhone 5s	iOS 10.2.1	59 FPS	91,21%
Galaxy S7 Edge	Android 7.0	59 FPS	85,44%
Oneplus One	Android 6.0.1	52 FPS	70,47%

Op iOS is er duidelijk de hoogste stabiliteit, zelfs voor een iPhone uit 2013. Bij Android is er een duidelijker verschil tussen de nieuwere Galaxy S7 edge uit 2016 en de Oneplus One uit 2014. De S7 behaalt iets lagere stabiliteit dan de iPhone maar heeft wel gemiddeld 59FPS. De Oneplus One

scoort het laagste in het geheel, maar met 52 FPS en een stabiliteit van 70% voelt de app nog steeds vloeiend aan. Deze doelstelling is hiermee gehaald.

De applicatie biedt een minimale impact op de API-infrastructuur van Infoplaza.

Deze vraag is nog lastig te beantwoorden omdat de app nog niet live gebruik wordt. Pas zodra de gebruikers in grotere getalen gebruik gaan maken van de applicatie zal de impact gemeten kunnen worden. Uiteraard is er wel al rekening gehouden met de infrastructuur door de API-calls minimaal te houden. Zo zijn er nooit meer dan 2 calls nodig om een scherm met data te tonen. Deze doelstelling is echter niet aantoonbaar gehaald.

8.2. Conclusie

In totaal zijn 6 van de 8 doelstellingen van Infoplaza behaald, waarbij de twee andere doelstellingen uiteindelijk niet meetbaar waren. Voor de hoofdvraag “Op welke manier zou een cross-platform framework voor mobiele applicaties ingezet kunnen worden voor de apps van Infoplaza, zodat er minder code gedupliceerd wordt over de verschillende platformen?” kan daarom geconcludeerd worden dat het Xamarin Framework geschikt is om ingezet te worden voor het bouwen van cross-platform applicaties. Infoplaza is ook zeer tevreden met de OVplaza app, die ook na dit onderzoek doorontwikkeld zal worden.

8.3. Aanbevelingen

Op basis van de resultaten van dit onderzoek en de eigen bevindingen van de afstudeerder zijn er de volgende aanbevelingen richting Infoplaza:

Gebruik Xamarin voor nieuwe applicaties

Xamarin heeft zich duidelijk geschikt getoond voor het ontwikkelen van nieuwe applicaties. Hierbij kan er van de grond af aan direct geleverd worden op meerdere platformen, waardoor de applicatie snel naar de stores te brengen is en deze eenvoudig te onderhouden is. Voor de huidige applicaties moet nog op een per-applicatie basis gekeken worden of deze voordeel halen uit de vertaling naar Xamarin. In de eerste instantie zal naar de kosten gekeken moeten worden om een ontwikkelaar de bestaande applicatie om te laten zetten tegenover de kosten die bespaard kunnen worden met het onderhoud. Ook zijn grafisch zeer complexe apps mogelijk niet geschikt voor Xamarin, omdat Xamarin een minimale maar toch wel aanwezige performance impact heeft kan dit net het verschil maken.

Maak gebruik van Xamarin Test Cloud.

Xamarin Test Cloud is een dienst van Xamarin waar een applicatie een geautomatiseerde UI test draait op meer dan 400 verschillende telefoons. Hiermee kan de garantie geleverd worden dat de UI correct werkt op alle platformen. De Test Cloud kan afgenomen worden in pakketten verschillende van $99 tot $799 per maand. Een gratis trial van 30 dagen is mogelijk voor alle pakketten.

Onderzoek een overstap van Crashlytics naar Azure Mobile Center.

Crashlytics is een dienst die zeer sterke analytics levert voor apps. Ook ondersteund deze de release van beta versie naar een selecte test groep. Hoewel de functionaliteiten ruim zijn, is er geen officiële ondersteuning voor het Xamarin Framework. Op dit moment is deze ondersteund middels een mapping van de officiële libraries. Deze mist echter de generatie van een build ID bij de release van Android die normaal via de Android Studio plugin verlopen. Deze moet daarom bij iedere release gegenereerd worden in Android Studio en handmatig gekopieerd worden naar het visual studio project.

Azure Mobile Center biedt veel voordelen boven Crashlytics.

1. Er is een officiële library voor Azure Mobile Center ondersteuning voor Xamarin, Java en Objective-C.
2. Er is ondersteuning voor continous integration waardoor code direct vanuit de git repository gecompileerd en verspreid kan worden onder de testers. Dit verhoogt het tempo van distributie aan testers exponentieel.
3. Er is ondersteuning voor iOS, Android, Xamarin, React Native en Unified Windows Platform. Met minimale aanpassingen zouden ook de bestaande apps overgezet kunnen worden naar Azure Mobile Center zodat alles in een enkel platform blijft.
4. Er is ondersteuning voor meerdere distributie groepen die ieder aparte releases kunnen ontvangen. Dit geeft meer controle over wie je welke versie wilt laten testen.
5. Er is ook directe integratie met de Xamarin Test Cloud, waardoor ook de app automatisch getest kan worden los van en tijdens de continuous integration cyclus. Dit levert een kleinere kans op dat er een distributie wordt gedaan die fouten bevat.

Een overstap naar Azure Mobile Center is waarschijnlijk wel pas op de langere termijn wenselijk, omdat deze op het moment van schrijven nog in preview is. De publiekelijke release staat op de roadmap voor eind 2017.

Zo is er op dit moment geen ondersteuning voor codebase voor continuous integration en de prijzen zullen niet bekend zijn tot na de release, al is er wel bevestigd dat er gratis tiers beschikbaar zullen zijn.

Maak gebruik van Xamarin University.

Xamarin biedt een trainingscursus aan genaamd Xamarin University. Deze geeft toegang tot live online lessen, 1 op 1 sessies met Xamarin Experts, zelfstudie lessen, gast lezingen en meer. Via dit platform is het ook mogelijk om certificering te behalen als Xamarin Certified Developer. Dit platform zou goed benut kunnen worden om nieuwe ontwikkelaars voor Xamarin apps snel up-to-speed te brengen. De kosten hiervan bedragen $83,25 per maand per gebruiker.

9. Evaluatie

In dit hoofdstuk wordt de procesgang van het project geëvalueerd. Dit is onderverdeeld in de drie fasen van het project.

9.1. Planning

De planning van het project verliep vrij moeizaam. De afstudeerder heeft veel moeite met het schrijven van verslagen, waardoor er veel tijd en iteraties nodig waren voor schrijven van het plan van aanpak. Hierdoor is er ongeveer twee weken langer dan gepland nodig geweest om het plan van aanpak af te ronden. Hierdoor is wel veel iteratie mogelijk geweest over het plan van aanpak, waardoor deze uiteindelijk wel zeer solide opgeleverd kon worden.

9.2. Onderzoek

Het onderzoek naar de eerste deelvraag omvatte 6 interviews, waarbij de afstudeerder vrij passief was met het inplannen waardoor deze over een grotere tijdspan verspreidde dan gewenst was. Ook koste het transcriberen meer tijd dan verwacht was omdat de afstudeerder deze te netjes probeerde uit te werken. Zo kostte een interview van 10 minuten ruim 4 uur om uit te werken. De interviews leverden wel meer informatie op dan verwacht. Er werden veelal uitgebreide antwoorden gegeven op de vragen, waarmee een zeer duidelijk beeld van de huidige situatie gevormd kon worden.

Bij het onderzoek naar deelvraag twee waren er weinig bronnen te vinden die een duidelijk beeld gaven van het landschap aan frameworks, mede omdat deze constant veranderd. Uiteindelijk is gekozen om de lijst op propertycross (Eberahrdt & Price, 2016) te volgen in combinatie met de frameworks die hiervoor gevonden waren. Er is gekozen om het bij deze lijst te laten om de scope van het onderzoek beperkt te houden. Hierdoor kon er gedetailleerd gekeken worden naar de inhoud van deze frameworks.

Het onderzoek naar de derde deelvraag begon met het maken van een vergelijkingsmatrix. Hierin werden de belangrijke eigenschappen van de frameworks uitgezet op basis van wat volgens de resultaten van de interviews belangrijke eigenschappen zijn. Er is gekozen om deze lijst vervolgens naar een viertal frameworks terug te dringen middels een puntentelling om de scope van het onderzoek te beperken. Er was simpelweg geen tijd om ieder framework tot in de diepgang te onderzoeken, dus is er gefocust op de meest relevante frameworks voor Infoplaza.

Het onderzoek vervolgde met deze vier frameworks door deze te benchmarken middels een testapplicatie voor ieder framework. Hiervoor is er gekozen gebruik te maken van een bestaande benchmark die ontwikkeld was door NativeScript. Er is hier gekozen om niet een eigen benchmark te creëren, omdat hiervan te weinig kennis aanwezig was bij de afstudeerder. Dit bespaarde een hoop tijd die anders toegewijd zou zijn aan het ontwikkelen hiervan en leverde krachtige data op voor het onderzoek. Bij cross platform frameworks is een van de eerste factoren die ter discussie komt toch wel de performance van de apps.

9.3. Ontwikkeling

Het ontwikkelproces begon met het maken van een storyboard. Hierin waren alle user stories opgenomen die gewenst waren voor de OVplaza app. Deze werd zeer goed ontvangen door de opdrachtgever, die ook voor het eerst gebruik maakte van de Scrum ontwikkelmethode.

In de eerste sprint waren vooral de basis stories opgenomen, die resulteerden in een kale app met navigatie en een “over de app” pagina. Deze sprint was vooral een kennismaking met Xamarin, waarin een skelet voor de rest van de app wordt opgezet. Hierbij heeft de afstudeerder gefocust op een sterke grondlegging voor de app.

Tijdens de tweede sprint ontstond er een complex vraagstuk bij de lijst met halte op het hoofdscherm. Omdat de kennis van Xamarin nog beperkt was werd deze initieel via for loops in de code gevuld. Deze code werkte te veel op de UI-thread waardoor de prestaties van de app sterk terugliepen.

Na enkele pogingen om de code te optimaliseren is er een andere invalshoek onderzocht middels XAML. Deze bleek na onderzoek ondersteuning te bieden voor property binding waarmee de lijst dynamisch gevuld kon worden. Deze oplossing presteerde beter en leverde een betere scheiding tussen UI en Logica op, omdat de logica nu geïsoleerd was in een viewmodel. Hoewel dit een week aan tijd koste leverde het een stevigere basis voor de app op waardoor de toekomstige ontwikkelingen sneller uitgevoerd konden worden.

Op dit punt werd, vanwege de complexiteit van de nieuwe kennis die opgedaan moest worden tijdens het ontwikkelen, minder gedocumenteerd dan was gehoopt. Deze documenten zijn later volledig bijgewerkt, maar hadden achteraf gezien meer in het iteratieve proces meegenomen moeten worden.

In sprint 3 werd het heel snel na de planning duidelijk dat er te veel taken waren opgenomen in de sprint backlog en dat deze taken te groot waren, zoals ook uitgelegd staat in hoofdstuk 7.2.4. Deze taken werden daarom opgesplitst in aparte, kleinere taken. Dit zorgde ervoor dat er een duidelijker beeld ontstond wat er gerealiseerd moest worden en er een betere inschatting gedaan kon worden hoe lang dit zou duren.

In sprint 4 kon er vanwege de verbeteringen aan de taken in sprint 3 een betere inschatting gedaan worden wat er gerealiseerd kon worden in de sprint. Deze sprint is er veel focus geweest op kwaliteitsgarantie. Door middel van vooral Unit tests kon de kwaliteit van de library gegarandeerd worden, zodat deze veilig aangeboden kan worden voor andere applicaties. Daarnaast wist de afstudeerder in deze sprint ook een control te realiseren waarmee het mogelijk is ook kleinere collecties te laten zien in een view middels een RepeaterView, die een collectie als bron gebruikt om een of meerdere views te laten zien. Deze RepeaterView is onder andere benut om de meerdere vertrektijden voor een enkele lijn te kunnen laten zien.

10. Literatuurlijst

Adobe Systems Inc. (2016). *Adobe PhoneGap*. Opgeroepen op Maart 17, 2017, van http://phonegap.com/

Adobe Systems Software Ireland Ltd. (2017). *Adobe AIR*. Opgeroepen op Maart 17, 2017, van http://www.adobe.com/nl/products/air.html

Axway. (2017). *Appcellerator*. Opgeroepen op Maart 17, 2017, van https://www.appcelerator.com/

Code-on. (sd). *Waarom je best een framework gebruikt.* Opgeroepen op Mei 26, 2017, van code-on: http://code-on.be/onze-webdeveloper-blog/wat-is-een-framework/

Dijkstra, B. (2005, Maart). *Webservices via REST*. Opgehaald van Whitehorses: http://www.whitehorses.nl/whitebooks/2005/webservices-rest

Dijoantonycj. (2015, Mei 2). *8-UX-Pitfalls-To-Avoid-In-Mobile-App-Design.jpg*. Opgeroepen op April 2, 2017, van Wikimedia: https://commons.wikimedia.org/wiki/File:8-UX-Pitfalls-To-Avoid-In-Mobile-App-Design.jpg

Eberahrdt, C., & Price, C. (2016, October 28). *PropertyCross*. Opgeroepen op Maart 3, 2017, van http://propertycross.com/

Embarcadero Technologies, Inc. (2017). *Delphi*. Opgeroepen op Maart 17, 2017, van https://www.embarcadero.com/products/delphi

Facebook Inc. (2017). *React Native*. Opgeroepen op Maart 17, 2017, van https://facebook.github.io/react-native/

Holland, B. (2016, Juli 14). *NativeScript And Xamarin*. Opgehaald van Nativescript blog: https://www.nativescript.org/blog/nativescript-and-xamarin

IDC. (2016). *Smartphone OS Market Share, 2016 Q3*. Opgeroepen op Maart 3, 2017, van IDC: http://www.idc.com/promo/smartphone-market-share/os

Intersoft Solutions. (2017). *Crosslight*. Opgeroepen op Maart 17, 2017, van https://www.intersoftsolutions.com/Crosslight

LTD, C. O. (2012). *Codename One*. Opgeroepen op Maart 17, 2017, van https://www.codenameone.com/

Neomades. (2017). *NeoMAD*. Opgeroepen op Maart 17, 2017, van http://neomades.com/en/crossplatform-mobile-development/neomad4

Pew Research Center. (2017, Januari 12). *Mobile fact sheet*. Opgeroepen op Maart 3, 2017, van Pewinternet: http://www.pewinternet.org/fact-sheet/mobile/

Progress Software Corporation. (2017). *NativeScript*. Opgeroepen op Maart 17, 2017, van https://www.nativescript.org/

Schwaber, K., & Sutherland, J. (2016). *The Scrum Guide.* Opgeroepen op Maart 3, 2017, van Scrumguides.org: http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf

Scribbr. (sd). *Overzicht van onderzoekssoorten*. Opgeroepen op April 7, 2017, van Scribbr: https://www.scribbr.nl/category/onderzoeksmethoden/

TAU Technologies. (2016, October 22). *Rhodes*. Opgeroepen op Maart 17, 2017, van https://github.com/rhomobile/rhodes

The Apache Software Foundation. (2017). *Apache Cordova*. Opgeroepen op Maart 17, 2017, van http://cordova.apache.org/

The Kivy Organization. (sd). *Kivy*. Opgeroepen op Maart 17, 2017, van https://kivy.org/

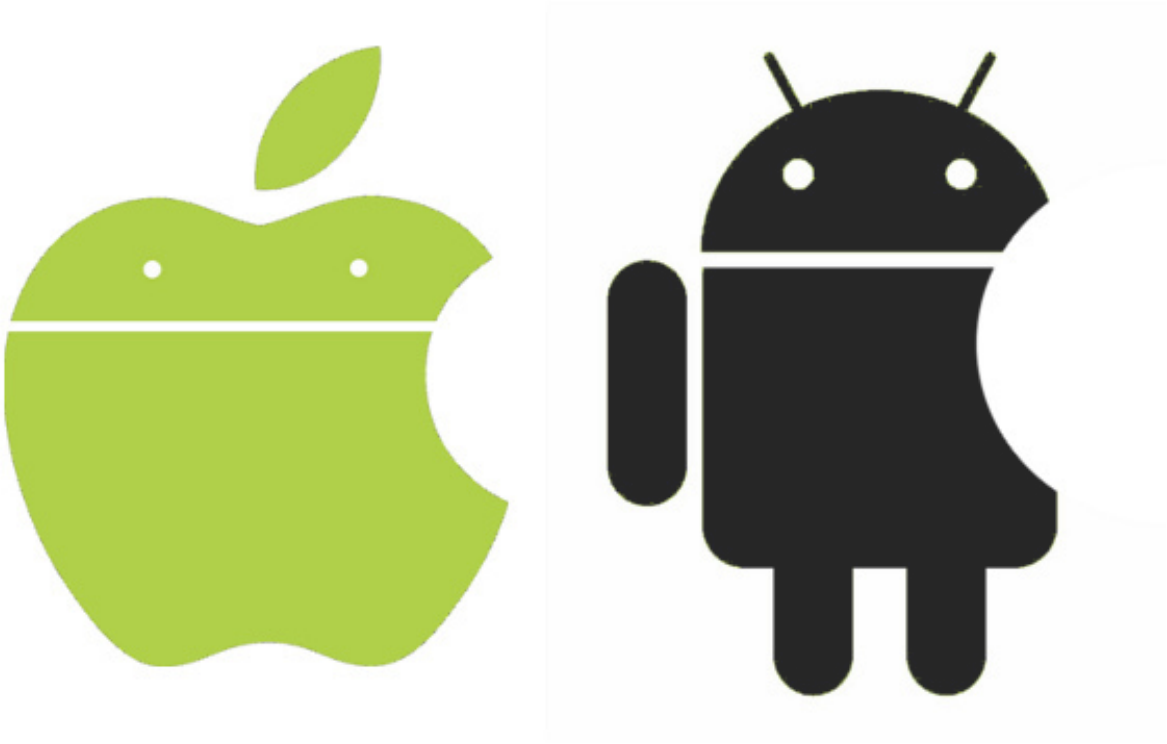
The Qt Company. (2017). *Qt*. Opgeroepen op Maart 17, 2017, van https://www.qt.io/

Xamarin Inc. (2017). *Xamarin*. Opgeroepen op Maart 17, 2017, van https://www.xamarin.com/

11. Bijlagen

11.1. Plan van Aanpak

Vanaf de volgende pagina is het plan van aanpak toegevoegd.



PLAN VAN AANPAK

Infoplaza BV & Hogeschool Utrecht

CROSS-PLATFORM FRAMEWORKS

Martin Luhrman - 1638082

Inhoudsopgave

1. Inleiding

Mobiele telefoons zijn een kenmerk van deze tijd. Uit recent onderzoek (Pew Research Center, 2017) blijkt dat 77% van de Amerikanen een smartphone bezitten. Veruit het grootste deel van de smartphones draaien Android of iOS volgens ander onderzoek (IDC, z.j.). Met zo'n grote doelgroep is het interessant voor bedrijven om met het maken van apps voor hun diensten een groter publiek te bereiken. Om een app te kunnen maken bieden Google en Apple speciale Software Development Kits aan voor Android in Java en iOS in Objective-C en Swift.

De apps die gemaakt worden met behulp van deze SDK's staan ook wel bekend als 'Native' apps. Er komt echter al snel het probleem dat wanneer je voor Android en iOS een app wilt maken, je vanwege de aparte SDK en talen die hierbij horen je de apps vaak praktisch dubbel aan het ontwikkelen bent.

Om dit probleem op te lossen zijn er in grote lijnen twee oplossingen ontstaan:

1. Er wordt een abstractie boven de API van de platformen gelegd, waardoor er in een enkele codebase een applicatie gemaakt kan worden voor Android en iOS. Deze oplossing produceert een app die na compilatie theoretisch gelijk is aan een native applicatie.
2. De applicatie wordt met HTML, CSS en JavaScript opgebouwd. Via een speciale wrapper app kan deze applicatie op ieder apparaat met dezelfde code getoond worden, omdat het in feite niet veel meer is dan een website.

Binnen Infoplaza wordt er nog native ontwikkeld, maar er is interesse ontstaan voor de mogelijkheden die een cross-platform framework kan bieden voor hun apps.

Dit document geeft toelichting op de opdracht die vanuit Infoplaza aan de afstudeerder is gegeven. Het beschrijft hoe hij van plan is om het onderzoek te verrichten voor de opdracht, welke methodieken er bij de uitvoering van de opdracht gebruikt zullen worden en hoe de globale planning voor de opdracht eruitziet.

2. Aanleiding en context

2.1. Organisatorische Context

Infoplaza is een bedrijf die streeft om weers-, verkeers- en ov-informatie op een duidelijke manier te leveren aan zijn klanten. Deze informatie wordt gedistribueerd aan consumenten via hun websites, apps en via de verkoop aan derden zoals de NOS, nu.nl en het merendeel van de kranten in Nederland. Ook biedt Infoplaza zijn informatie en kennis aan bedrijven die afhankelijk zijn van betrouwbare weerinformatie voor het leveren van hun eigen diensten, bijvoorbeeld ProRail.

Er werken op dit moment 21 mensen in fulltime of parttime dienstverband en de organisatie bestaat grofweg uit 3 onderdelen:

- De directie
- Daarna de redactie en meteorologen
- Als laatste een verzameling van IT-beheer, ontwikkelaars, designers en marketing.

Binnen de organisatie zal de afstudeerder als App ontwikkelaar zich ontfermen over de ov-applicaties van Infoplaza en daarmee deel uitmaken van de ontwikkelaars.

2.2. Aanleiding

De Android, iOS en Windows apps worden in de huidige situatie bij Infoplaza per platform native ontwikkelt. Meestal wordt dit uitbesteed aan externe ontwikkelaars die per platform ingehuurd worden, wat extra complexiteit toevoegt aan het stroomlijnen van de verschillende platformen. Ook moet de achterliggende logica van de applicaties op ieder platform opnieuw geschreven worden.

Door de tijd die het proces kost en de redundantie die plaatsvindt tijdens het ontwikkelen ziet Infoplaza een kans voor manieren om applicaties cross-platform te kunnen ontwikkelen. Hierdoor zouden dezelfde ontwikkelaars zich op alle platformen kunnen richten, en hoeft logica maar één keer geschreven te worden waarna het makkelijk hergebruikt kan worden. De mobiele applicaties van Infoplaza kunnen daardoor hopelijk sneller ontwikkeld worden.

3. Opdracht en Doelstelling

3.1. Beschrijving

De opdracht vanuit Infoplaza is het maken van een cross-plaform applicatie voor iOS en Android voor hun ov-dienst OVplaza. Deze applicatie bestaat in de huidige vorm als een iOS app die al enkele jaren geen updates meer heeft ontvangen. Omdat deze app een volledig nieuwe versie vereist, is het de perfecte kandidaat om deze cross-platform te ontwikkelen omdat er geen rekening gehouden hoeft te worden met oudere versies van de applicatie.

Als onderzoek voor de opdracht zal gekeken worden naar verschillende Frameworks voor het bouwen van cross-platform apps, zoals bijvoorbeeld het C#.Net Framework Xamarin van Xamarin Inc. (z.j.) en hybride frameworks zoals het Javascript Framework Cordova van The Apache Software Foundation (z.j.). Uit dit onderzoek zal moeten blijken welk framework het beste binnen Infoplaza kan worden toegepast voor het bouwen van cross-platform mobiele applicaties.

Op basis van de wensen en de huidige situatie van Infoplaza zal de focus liggen op de mogelijkheden van Xamarin tegenover alternatieve mogelijkheden omdat de ontwikkelaars binnen Infoplaza al veelal bekend zijn met C# en het .Net Framework. Hierdoor zou er ook in de toekomst diepere integratie mogelijk kunnen zijn met de websites van Infoplaza omdat deze ook veelal op .NET gebaseerd zijn.

Uit het onderzoek zal een framework naar voren komen waarmee na goedkeuring van Infoplaza een proof of concept gemaakt gaat worden van de OVplaza app. op basis daarvan te kan geconcludeerd worden of het framework daadwerkelijk aan de wensen en verwachtingen van Infoplaza voldoet.

3.2. Doelstellingen

Uit de kwestie blijkt dat er binnen de organisatie wens is om van de huidige infrastructuur voor hun mobiele applicaties over te stappen naar cross-platform ontwikkelingsmogelijkheden. Het vraagstuk dat daaruit ontstaat is of de overstap naar een cross-platform framework de verwachte voordelen kan bieden. Hiervoor zal een proof of concept applicatie gemaakt worden waarmee de eisen aan het framework getoetst zullen worden zoals gegeven in Tabel 1.

Tabel 1: Eisen voor het framework

Requirement	Belang
De applicatieontwikkeling loopt gelijk voor Android en iOS.	*Technisch, marketing en financieel*
Het onderhoud aan de applicatie, bijvoorbeeld het toevoegen van een nieuwe functionaliteit of het oplossen van een bug, moet minder tijd kosten dan wanneer dit native zou gebeuren door twee aparte ontwikkelaars.	*Technisch, marketing en financieel*
De applicatie kan op Android en iOS gezamenlijk gelanceerd worden.	*Marketing en financieel*
De interface van de applicatie biedt dezelfde huisstijl op Android en iOS.	*Marketing*
De applicatie maakt gebruik van een gedeelde codebase voor de business logic.	*Technisch en financieel*
De impact van systeemupdates van Android en iOS moet minimaal zijn voor het gebruik van het framework.	*Technisch*
De applicatie behaald een goede GPU-performance op de verschillende platformen van gemiddeld 60 FPS.	*Technisch en marketing*
De applicatie biedt een minimale impact op de API-infrastructuur van Infoplaza.	*Technisch en financieel*

3.3. Onderzoeksvragen

Op basis van de aanleiding en de opdracht zoals geleverd door Infoplaza is er de volgende hoofdvraag ontstaan:

Op welke manier zou een cross-platform framework voor mobiele applicaties ingezet kunnen worden voor de apps van Infoplaza, zodat er minder code geduplicеerd wordt over de verschillende platformen?

Om deze hoofdvraag beter te kunnen beantwoorden zal er onderzoek gedaan worden naar de verschillende onderdelen die van noodzaak zijn voor de hoofdvraag:

1. *Hoe worden de huidige mobiele applicaties van Infoplaza gerealiseerd?*
2. *Welke frameworks zijn er beschikbaar die cross-platform capaciteiten bieden voor mobile development?*
3. *Wat zijn de voor- en nadelen van de beschikbare frameworks voor cross-platform applicaties in verhouding tot elkaar?*

4. Resultaten

4.1. Deelresultaten

Uit het onderzoek van de eerste deelvraag zal een beschrijving komen van de huidige situatie van Infoplaza. Deze informatie biedt een duidelijke context om te vergelijken met een potentiele nieuwe situatie zoals beschreven in de hoofdvraag.

Uit het onderzoek van de tweede deelvraag zal een overzicht ontstaan van de mogelijkheden op het gebied van cross-platform frameworks. Op basis van deze resultaten kan hierna verder onderzocht worden welke voor- en nadelen deze frameworks met zich meebrengen. Op basis van het overzicht aan voor- en nadelen dat hieruit ontstaat kan er een aanbeveling gedaan worden naar de opdrachtgever toe voor een keuze van een framework. Na goedkeuring van deze keuze zal er een proof of concept gemaakt worden op basis van het resulterende framework.

4.2. Eindresultaten

Er zijn verschillende producten die opgeleverd moeten zijn aan het einde van dit onderzoek:

- Een scriptie met de verdediging voor de keuzes die gemaakt zijn tijdens het onderzoek.
- Een onderzoeksrapport waarin de genomen stappen en de resultaten van het onderzoek worden beschreven.
- Een Android en een iOS Applicatie.
- Functioneel ontwerprapport voor de proof of concept.
- Technisch ontwerprapport voor de proof of concept.
- Documentatie voor het scrumproces dat beschreven is in The Scrum Guide van Schwaber en Sutherland (2016):
    - Sprint evenementen:
        - Sprint Planning
        - Sprint Review
        - Sprint Retrospective
    - De Product backlog
    - De Sprint backlogs
- Persoonlijke reflectie.

5. Aanpak

5.1. Methoden

Om de onderzoeksvragen te kunnen beantwoorden en een proof of concept te kunnen maken zal er gebruik gemaakt gaan worden van verschillende onderzoeks- en ontwerpmethoden.

Tabel 2: Onderzoeksfasen

Fase	Methode	Toelichting
Onderzoek deelvraag 1	Interviews, Deskresearch	Middels een Veldonderzoek zoals beschreven door Scribbr (z.j.) bestaande uit vooral interviews en deskresearch zal de huidige stand van zaken in kaart gebracht worden omtrent de ontwikkeling van mobiele applicaties. Hierdoor kan beter bepaald worden welk framework het beste bij Infoplaza zal passen.
Onderzoek deelvraag 2	Deskresearch	Middels een Inventarisatieonderzoek zoals beschreven door Scribbr (z.j.) zal het landschap van cross-platform frameworks in kaart gebracht worden. Zodra duidelijk in beeld is wat de opties zijn, kan er gekeken gaan worden naar deelvraag 3.
Onderzoek deelvraag 3	Deskresearch	Middels een Vergelijkend onderzoek zoals beschreven door Scribbr (z.j.) zullen de resultaten van deelvraag 2 tegen elkaar op worden gezet en zullen de voor- en nadelen van ieder framework onderzocht worden.
Bouwen Proof of Concept	Proof of Concept	Het Proof of Concept wordt gemaakt volgens Agile Scrum zoals beschreven in de scrum guide van Schwaber en Sutherland (2016). Omdat er echter geen sprake is van een team maar van een enkele ontwikkelaar zullen de elementen van scrum waar een team voor nodig is niet uitgevoerd kunnen worden. Verder zal er geen daily scrum uitgevoerd worden, omdat dit ook een groepsactiviteit is. Om de backlog te bepalen wordt er gebruik gemaakt van user story mapping. Deze worden met een digitaal Story Board bijgehouden. Verder wordt er voor de kwaliteitsgarantie van de proof of concept app gebruik gemaakt van Unit testing en UI testing. De exacte implementatie voor deze tests hangt af van het framework dat uit het onderzoek naar voren zal komen.

5.2. Globale planning

Om een beeld te kunnen geven van de verwachte activiteiten die in dit onderzoek plaats zullen vinden is de onderstaande tabel gemaakt. Hierin is een overzicht gemaakt van de activiteiten die plaats moeten vinden en de verwachte uren die voor deze activiteit nodig zijn. Onder aan het overzicht staan nog de exacte opleverdata genoemd.

Tabel 3: Planning

Taak	week																Uren
	Februari			F&M	Maart			M&A	April				Mei				M&J	
	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	Totaal
Onderzoek							DL				DL			DL			DL	284
Plan van aanpak	32	28	16															76
Onderzoek huidige situatie				30														30
Onderzoek beschikbare frameworks					30													30
Onderzoek vergelijken frameworks						30												30
Scriptie							4	8	8	8	8	8	8	10	10	30	16	118
Proof of concept:									OP		OP		OP		OP			269
Voorbereiding				6	6	6	32											50
Sprint 1								37	30									67
Sprint 2										28	20							48
Sprint 3												19	30					49
Sprint 4														28	27			55
Overige Activiteiten																		110
Introductie	8	8																16
Andere taken voor werkgever		4	19	4	4	4	4	2	2	2	2	2	2	2	2	2		57
Vakantiedagen											8	8				8		24
Leerteam			3									3						6
Rockstar IT			2							2	2				1			7
Uren per week	40	40	40	40	40	40	40	47	40	40	40	40	40	40	40	40	16	
Uren totaal	40	80	120	160	200	240	280	327	367	407	447	487	527	567	607	647	663	663

legenda: Deadline Oplevering

Opleverdata:

24 Maart (week 12), Inleveren afstudeercontract.
18 April (week 16), Inleveren eerste scriptie concept.
8 Mei (week 19), Inleveren tweede scriptie concept.
19 Mei (week 20), Oplevering proof of concept.
30 Mei (week 22), Inleveren eindversie scriptie.

6. Risicoanalyse en benodigde middelen

6.1. Risico's

Tabel 1: Risico analyse

Mogelijk risico	Maatregel(en)
De stagebegeleider en opdrachtgever zijn niet altijd op locatie aanwezig.	Ze zijn meestal via Slack, Email of telefonisch bereikbaar voor vragen als het nodig is.
De student heeft voorkeur om geen Javascript framework te gebruiken.	Op basis van goede bronnen en beargumentering een keuze maken, persoonlijke voorkeur opzijzetten.
De student heeft veel moeite met verslaglegging.	Regelmatig feedback vragen aan begeleiders.
Vanwege drukte of vakantie is er tijdelijk geen aanspreekpunt binnen de organisatie bereikbaar.	Zelfstandig doorwerken, feedback vragen aan docentbegeleider of leerteam.
Android mist functionaliteiten die wel beschikbaar zijn in iOS of vice versa.	Als het mogelijk is zoeken of er een omweg is om deze functionaliteit alsnog naar het andere platform te krijgen. Zo niet overleggen of de functionaliteit essentieel is of dat deze weggelaten kan worden.
De student is snel afgeleid door gesprekken van andere collegas.	De student kan zichzelf afscheiden door in een aparte ruimte te gaan zitten.

6.2. Middelen

De middelen zijn lastig te bepalen omdat de programmeertaal en bijbehorende tooling nog niet bepaald zijn. De middelen die gebruikt gaan worden die wel vast staan zijn:

- Een MacBook Pro.
- Parallels Desktop om windows te kunnen draaien op de MacBook.
- XCode voor iOS compilatie en simulatie.
- Android SDK voor Android compilatie en emulatie.
- CodebaseHQ voor code- en projectbeheer.
- Git voor codebeheer
- Visual Studio voor het schrijven van code.
- Fysieke Android en iOS Telefoon om te testen op verschillende apparatuur.

7. Communicatie

7.1. Bedrijfs-/persoonsgegevens

Er zijn verschillende personen betrokken bij dit project.

Tabel 2: Contactgegevens

Rol	Naam	Email adres	Telefoonnummer
Opdrachtgever	René Westening	*	*
Bedrijfsbegeleider	Sjoerd Huininga	*	*
Docentbegeleider	Dick Pronk	*	*
Stagiair	Martin Luhrman	*	*

* Deze informatie is vanwege de vertrouwelijke aard verwijderd.

Contact met de opdrachtgever en de bedrijfsbegeleider verloopt grotendeels op locatie en verder eventueel telefonisch, over de email of over Slack. Iedere week wordt er verantwoording gedaan naar de bedrijfsbegeleider van de taken die zullen plaatsvinden.

Contact met de docentbegeleider gebeurt primair over de email en daarnaast op locatie bij Infoplaza. De docentbegeleider heeft een maximumbudget van 20 uur om de student te begeleiden. Als het project goed loopt zal veel van deze tijd pas aan het einde gebruikt worden voor feedback op de scriptie, maar mochten er problemen voordoen die niet door het stagebedrijf opgelost kunnen worden zal de docent eerder in het traject hierbij betrokken worden.

8. Begrippenlijst

Apache Cordova

Apache Cordova is een hybride applicatie framework die gebruik maakt van CSS, HTML en Javascript om cross-platform applicaties te kunnen maken. Apache Cordova is ontstaan als open source versie van PhoneGap, een framework dat in 2011 door Adobe Systems overkocht werd van Nitobi.

Cross-platform

Met cross-platform wordt de mogelijkheid van applicaties die op meerdere platformen werken bedoelt. Dit is mogelijk in de vorm van 'native' of hybride apps, waarbij respectievelijk de native API gewrapped wordt of een veelal HTML en Javascript

Framework

Een verzameling aan voorgeschreven functionaliteiten die ondersteuning vormen voor een ontwikkelaar om mee te kunnen werken.

Native apps

Mobiele apparaten bevatten allemaal wel een Software Development Kit (SDK). Android heeft een Java SDK, iOS heeft Swift en Objective-C SDK's en Windows Phone heeft een .NET SDK. Apps die gemaakt zijn voor deze apparaten in de bijbehorende taal worden ook wel native apps genoemd.

Ov-Plaza

Ov-Plaza is de naam van het platform waar Infoplaza zijn ov-informatie aan de consument aanbiedt.

Scrum

Agile werkmethode op basis van iteratief werken door middel van verscheidene sprints. Het doel is om veel contactmomenten met de eigenaar van het eindproduct te hebben waardoor er doelgericht ontwikkeld kan worden en aanpassingen snel gemaakt kunnen worden.

Xamarin

Xamarin is de naam voor een Framework waarmee in .NET mobiele applicaties geschreven kunnen worden. Oorspronkelijk was het een apart bedrijf, maar deze is begin 2016 overgenomen door Microsoft. Dit framework biedt een abstractie boven de native API van iOS en Android.

9. Voorlopige Bronnenlijst

Apple Inc. (z.j.). [Apple API Reference]. Geraadpleegd op 14 februari, 2017, van https://developer.apple.com/reference

Apple Inc. (z.j.). [Apple Design]. Geraadpleegd op 15 februari, 2017, van https://developer.apple.com/design/

Holland, B. (2016, 14 juli). NativeScript And Xamarin [Blogpost]. Geraadpleegd van https://www.nativescript.org/blog/nativescript-and-xamarin

IDC. (z.j.). Smartphone OS Market Share, 2016 Q3. Geraadpleegd op 3 maart, 2017, van http://www.idc.com/promo/smartphone-market-share/os

Pew Research Center. (2017, 12 januari). Mobile Fact Sheet. Geraadpleegd van http://www.pewinternet.org/fact-sheet/mobile/

Schwaber, K., & Sutherland, J. (2016). *The Scrum Guide*. Geraadpleegd van http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf

Scribbr. (z.j.). Overzicht van onderzoekssoorten. Geraadpleegd op 9 februari, 2017, van https://www.scribbr.nl/category/onderzoeksmethoden/

The Apache Software Foundation. (z.j.). [Cordova Documentation]. Geraadpleegd op 14 februari, 2017, van https://cordova.apache.org/docs/en/latest/

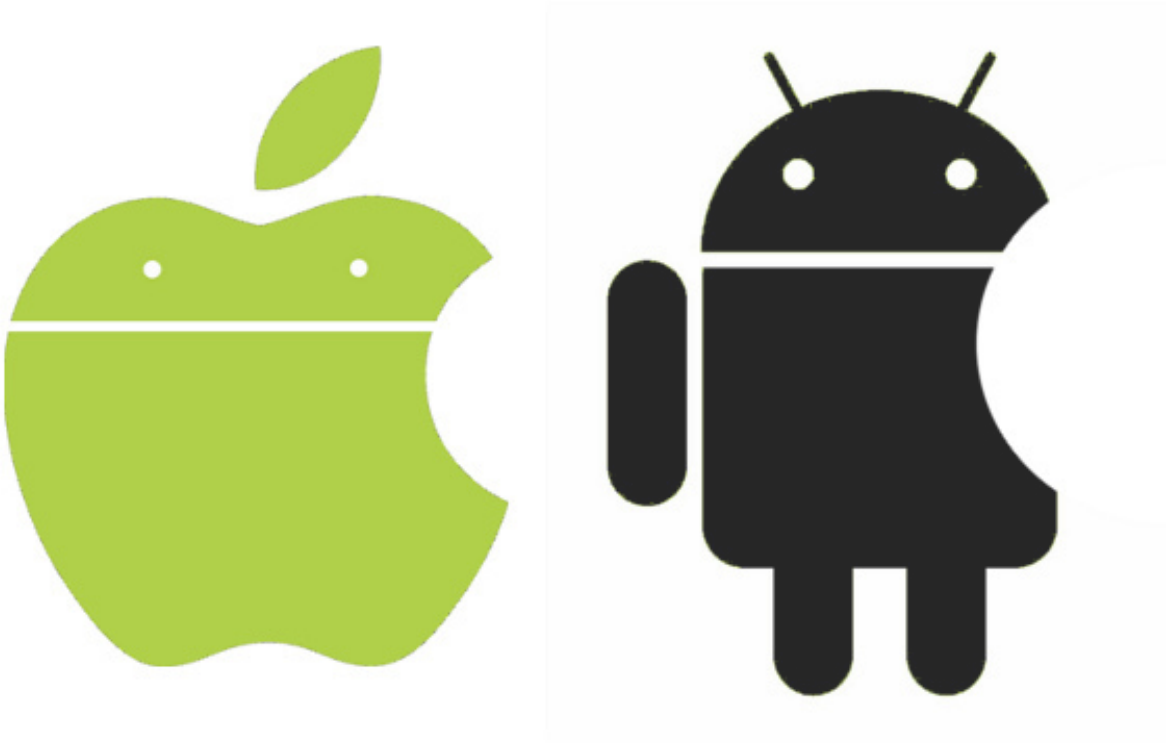
Xamarin Inc. (z.j.). [Xamarin University]. Geraadpleegd op 7 februari, 2017, van https://university.xamarin.com/self-guided

Xamarin Inc. (z.j.). [Xamarin API Documentation]. Geraadpleegd op 7 februari, 2017, van https://developer.xamarin.com/api/

Xamarin Inc. (z.j.). [Xamarin Guides]. Geraadpleegd op 7 februari, 2017, van https://developer.xamarin.com/guides/

11.2. Onderzoeksrapport

Vanaf de volgende pagina is het onderzoeksrapport toegevoegd.



ONDERZOEKSRAPPORT

Infoplaza BV & Hogeschool Utrecht

CROSS-PLATFORM FRAMEWORKS

Martin Luhrman - 1638082

Inhoudsopgave

1. Inleiding

Mobiele telefoons zijn een kenmerk van deze tijd. Uit recent onderzoek (Pew Research Center, 2017) blijkt dat 77% van de Amerikanen een smartphone bezitten. Veruit het grootste deel van de smartphones draaien Android of iOS volgens ander onderzoek (IDC, z.j.). Met zo'n grote doelgroep is het interessant voor bedrijven om met het maken van apps voor hun diensten een groter publiek te bereiken. Om een app te kunnen maken bieden Google en Apple speciale Software Development Kits aan voor Android in Java en iOS in Objective-C en Swift.

De apps die gemaakt worden met behulp van deze SDK's staan ook wel bekend als 'Native' apps. Er komt echter al snel het probleem dat wanneer je voor Android en iOS een app wilt maken, je vanwege de aparte SDK en talen die hierbij horen je de apps vaak praktisch dubbel aan het ontwikkelen bent.

Om dit probleem op te lossen zijn er in grote lijnen twee oplossingen ontstaan:

1. Er wordt een abstractie boven de API van de platformen gelegd, waardoor er in een enkele codebase een applicatie gemaakt kan worden voor Android en iOS. Deze oplossing produceert een app die na compilatie theoretisch gelijk is aan een native applicatie.
2. De applicatie wordt met HTML, CSS en JavaScript opgebouwd. Via een speciale wrapper app kan deze applicatie op ieder apparaat met dezelfde code getoond worden, omdat het in feite niet veel meer is dan een website.

Binnen Infoplaza wordt er nog native ontwikkeld, maar er is interesse ontstaan voor de mogelijkheden die een cross-platform framework kan bieden voor hun apps.

In dit rapport staat beschreven welk onderzoek hiervoor is verricht en de resultaten die hieruit naar voren zijn gekomen.

2. Opdracht

In dit hoofdstuk wordt de achterliggende opdracht voor het onderzoek toegelicht en worden de vragen beschreven die zijn ontstaan uit de analyse van deze opdracht.

2.1. Beschrijving

De opdracht vanuit Infoplaza is het maken van een cross-plaform applicatie voor iOS en Android voor hun ov-dienst OVplaza. Deze applicatie bestaat in de huidige vorm als een iOS app die al enkele jaren geen updates meer heeft ontvangen. Omdat deze app een volledig nieuwe versie vereist, is het de perfecte kandidaat om deze cross-platform te ontwikkelen omdat er geen rekening gehouden hoeft te worden met oudere versies van de applicatie.

Als onderzoek voor de opdracht zal gekeken worden naar verschillende Frameworks voor het bouwen van cross-platform apps, zoals bijvoorbeeld het C#.NET Framework Xamarin van Xamarin Inc. (z.j.) en hybride frameworks zoals het Javascript Framework Cordova van The Apache Software Foundation (z.j.). Uit dit onderzoek zal moeten blijken welk framework het beste binnen Infoplaza kan worden toegepast voor het bouwen van cross-platform mobiele applicaties.

Op basis van de wensen en de huidige situatie van Infoplaza zal de focus liggen op de mogelijkheden van Xamarin tegenover alternatieve mogelijkheden omdat de ontwikkelaars binnen Infoplaza al veelal bekend zijn met C# en het .NET Framework. Hierdoor zou er ook in de toekomst diepere integratie mogelijk kunnen zijn met de websites van Infoplaza omdat deze ook veelal op .NET gebaseerd zijn.

Uit de resultaten van het onderzoek zal er een framework voorgedragen worden die de beste mogelijkheden biedt voor het vervangen van de mobiele infrastructuur van Infoplaza.

2.2. Onderzoeksvragen

Op basis van de geformuleerde opdracht is de volgende hoofdvraag ontstaan:

Op welke manier zou een cross-platform framework voor mobiele applicaties ingezet kunnen worden voor de apps van Infoplaza, zodat er minder code gedupliceerd wordt over de verschillende platformen?

Om deze hoofdvraag beter te kunnen beantwoorden zal er onderzoek gedaan worden naar de verschillende onderdelen die van noodzaak zijn voor de hoofdvraag:

1. *Hoe worden de huidige mobiele applicaties van Infoplaza gerealiseerd?*
2. *Welke frameworks zijn er beschikbaar die cross-platform capaciteiten bieden voor mobile development?*
3. *Wat zijn de voor- en nadelen van de beschikbare frameworks voor cross-platform applicaties in verhouding tot elkaar?*

3. Methoden

Om de onderzoeksvragen te beantwoorden is er gebruik gemaakt van verscheidene onderzoeksmethoden, zoals beschreven staan in tabel 2.

Tabel 1: Onderzoeksmethoden

Fase	Methode	Toelichting
Onderzoek deelvraag 1	Interviews, Deskresearch	Middels een veldonderzoek zoals beschreven door Scribbr (z.j.) bestaande uit vooral interviews en deskresearch is de huidige stand van zaken in kaart gebracht omtrent de ontwikkeling van mobiele applicaties. Hierdoor kon er beter bepaald worden welk framework het beste bij Infoplaza zou passen.
Onderzoek deelvraag 2	Deskresearch	Middels een inventarisatieonderzoek zoals beschreven door Scribbr (z.j.) is het landschap van cross-platform frameworks in kaart gebracht. Met deze informatie kon er verder onderzoek gedaan worden naar deelvraag 3.
Onderzoek deelvraag 3	Deskresearch	Middels een vergelijkend onderzoek zoals beschreven door Scribbr (z.j.) zijn de resultaten van deelvraag 2 met elkaar vergeleken en zijn de voor- en nadelen van ieder framework onderzocht.

4. Resultaten

In dit hoofdstuk zullen de resultaten worden opgenomen die naar voren zijn gekomen uit de interviews en het inventariseren en vergelijken van de verschillende frameworks.

4.1. Huidige situatie

Voor het onderzoek zijn 6 stakeholders geïnterviewd over het huidige proces binnen Infoplaza, elke met verschillende rollen: projectmanager, iOS ontwikkelaar, Android ontwikkelaar en (user experience) Designer. Een van de eerste resultaten die opviel was dat er intern vooral iOS kennis in huis aanwezig was. De eerste apps die binnen de organisatie ontwikkeld waren, waren alleen op iOS beschikbaar. Toen Android eenmaal in beeld kwam werd dit uitbesteed, waardoor de kennis onder de vaste werknemers over Android lager is ten opzichte van iOS.

Hoewel er duidelijke communicatie is over de functionaliteiten komt het weleens voor dat de apps afwijkende eigenschappen ontwikkelen, zij het omdat de ontwikkelaar zijn eigen beeld heeft bij de opgegeven functionaliteit of deze niet op dezelfde manier haalbaar is door limitaties op het platform.

Wat wel als cruciaal wordt gezien is dat de apps veel zware grafische functionaliteiten gebruiken, vooral voor de weer apps. Deze laten vaak kaarten zien waar informatie bovenop getoond wordt, iets wat volgens Sjoerd erg zwaar was, wat hij in zijn interview ook noemde: “Dat zijn echt dingen waarvan we zeker 2 jaar geleden toen we begonnen tegen aan liepen, de limieten van de toestellen.”

Ook wordt er gesproken over de uitdaging van het behouden van een huisstijl. Door de verschillende design standaarden van iOS en Android is het lastig een uniforme stijl aan te houden.

4.2. Beschikbare frameworks

Voor dit onderzoek is er onderzocht welke frameworks er beschikbaar zijn. Deze lijst is gebaseerd op de website PropertyCross (Eberhardt & Price, sd). De frameworks van deze website zijn allereerst gefilterd op activiteit. Projecten die sinds voor 2016 geen updates hebben gehad zijn uitgesloten omdat niet aan te tonen is dat deze projecten nog actief zijn. Daarnaast is er gekeken of, wat in het geval van hybride frameworks veel het geval was, het framework een zelfstandig framework was of alleen UI-componenten biedt bovenop een van de frameworks in de lijst.

- Xamarin - Native & Hybride, C#
- AIR - Native, ActionScript/Flash
- Crosslight - Native, C#
- Delphi - Native, Object Pascal
- NativeScript - Native, JS(JavaScript)/TypeScript
- Qt - Native, C++
- RhoMobile - Native, JS/Ruby
- Titanium - Native, JS
- React Native - Native, JS
- NeoMAD - Native, Java
- Codename One - Native, Java
- Kivy - Native, Python
- Cordova/Phonegap - Hybride, JS

4.3. Vergelijking frameworks

Als eerste deel van dit onderzoek zijn de resultaten van het vorige hoofdstuk verder uitgebreid met details. Daarnaast kwam uit de interviews voor het onderzoek van de eerste deelvraag veel naar voren dat mensen bij hybride cross-platform frameworks last hadden of gehoord hadden over slechte performance. Omdat de performance een belangrijke eis is aan het framework, zijn veel hybride frameworks hierdoor weggelaten uit verdere analyse in dit onderzoek. Als uitzondering is Cordova wel nog opgenomen in de vergelijking ter verifiëring van deze hypothese tijdens het benchmarken.

Alle frameworks hebben een puntenscore gekregen op basis van enkele aspecten van het framework:

- Om over te stappen naar een framework wil je er zeker van zijn dat deze een lange levensduur heeft en goed ondersteund wordt. Een van de makkelijkste manieren om dat te toetsen is door te kijken naar de datum van de laatste release van het framework en voor een extra punt het formaat van de achterliggende organisatie. Voor dit onderzoek heb ik een punt toegekend als de laatste update in 2017 heeft plaatsgevonden.
- Er is een punt gegeven als het framework naar native compileert, vanwege de vereisten aan de performance van de applicatie.
- Er is een punt toegekend als de programmeertaal die gebruikt wordt door het framework bekend is binnen Infoplaza. De meeste kennis ligt veruit bij C#, maar er is ook veel ervaring met JavaScript en Python binnen de organisatie.
- Er is een punt toegekend als de prijs voor het gebruiken van het framework (nagenoeg) gratis is. Hoewel dit geen eis was, kunnen er veel kosten bespaard worden als er geen licentiekosten nodig zijn.
- Er is een punt toegekend als er enterprise IDE tooling beschikbaar is.
- Er is een punt toegekend als er support voor het framework geleverd wordt door de organisatie die het framework levert. Omdat het framework op grote schaal toegepast moet kunnen worden is premium support heel belangrijk om bij problemen goede ondersteuning te kunnen krijgen. Ook is het van belang dat deze support snel geleverd kan worden zodat de mogelijke vertraging minimaal is.

Tabel 2: framework vergelijking

Naam	Totaal	Levensduur	Native	Taal	Gratis	Tooling	Support
Xamarin	6,5	++	+	+	~	+	+
NativeScript	6	++	+	+	+	-	+
React Native	5	++	+	+	+	-	-
Cordova / Phonegap	5	++	-	+	+	+	-
Titanium	5	+	+	+	-	+	+
Qt	4,5	+	+	-	~	+	+
RhoMobile	4	-	+	+	+	+	-
AIR	4	+	+	-	-	+	+
NeoMAD	3,5	-	+	+	~	-	+
Codename One	3,5	+	+	-	~	-	+
Crosslight	3	-	+	+	-	-	+
Kivy	3	-	+	+	+	-	-
Delphi	3	-	+	-	-	+	+

Uit deze lijst met frameworks is daarna een selectie gemaakt van 4 frameworks:

- Xamarin
- NativeScript
- React Native
- Cordova

Xamarin en NativeScript zijn geselecteerd op basis van hun puntenscore. Deze hebben de hoogste en een na hoogste score behaald, en zijn daardoor als waarschijnlijk geschikte kandidaten opgenomen. Daarnaast is React Native opgenomen. Hoewel React een derde plaats deelde met Titanium, is er wegens naambekendheid gekozen voor React. De naamsbekendheid van het framework en de achterliggende organisatie, Facebook, heeft als voordeel dat er meer gebruikers zijn die al veel mogelijke problemen online besproken hebben en dat er voor de ondersteuning van het platform meer resources zijn dan bijvoorbeeld bij een kleinere organisatie. Als laatste is Cordova ook onderdeel geworden van de selectie om een hybride framework te kunnen vergelijken met zijn Native tegenhangers.

Deze selectie van 4 is gebenchmarked met behulp van code die beschikbaar is op de github pagina van NativeScript: https://github.com/NativeScript/sample-iOS-Profiling/tree/performance-tests

Hoewel de test code gelijk is voor ieder platform, bestaat er wel de mogelijkheid dat de tests zo gekozen zijn dat NativeScript een voordeel heeft boven de rest. De resultaten zijn echter wel naar verwachting.

Deze code test de bovengenoemde frameworks op snelheid op 3 vlakken:

- Primitieve getallen marshalling: Hiermee worden er een miljoen integers opgeslagen en getest hoe lang het framework daar over doet.
- String marshalling: Hiermee worden 100.000 strings aangemaakt en weer de looptijd getest.
- Big data marshalling: Hierbij worden ~65.000 objecten in 200 arrays geplaatst en weer de looptijd getest.

Deze tests zijn uitgevoerd op een simulator van een iPhone 7 en op een fysieke iPhone 5s, de resultaten hiervan zijn te zien in tabel 3.

Tabel 3: Gemiddelde - Simulator iPhone 7, iOS 12.2.1 op een MacBook Pro 13" (2015)

X	Native	Xamarin	React Native	NativeScript	Cordova
Primitives	9ms	341ms	33374ms	538ms	18765ms
Strings	30ms	231ms	37237ms	124ms	47226ms
Big Data Marshalling	271ms	901ms	34617ms	661ms	53822ms

Tabel 4: Gemiddelde - iPhone 5s, iOS 12.2.1

X	Native	Xamarin	React Native	NativeScript	Cordova*
Primitives	5ms	31ms	43651ms	1252ms	0
Strings	39ms	241ms	49152ms	471ms	0
Big Data Marshalling	314ms	2059ms	46615ms	1613ms	0

* Cordova crashte iedere test met de melding: “terminated due to memory issue”

Uit deze resultaten blijkt duidelijk dat Cordova en React Native aanzienlijke trager zijn dan bijvoorbeeld Xamarin en NativeScript.

5. Conclusies

In dit rapport is verslag gedaan over het onderzoek naar de volgende hoofdvraag:

Op welke manier zou een cross-platform framework voor mobiele applicaties ingezet kunnen worden voor de apps van Infoplaza, zodat er minder code gedupliceerd wordt over de verschillende platformen?

Om deze vraag te kunnen beantwoorden zijn er verschillende onderzoeksvragen gesteld om hier een beter beeld op te krijgen. In het vorige hoofdstuk staan de resultaten hiervan uitgewerkt.

Uit dat onderzoek kwam allereerst naar voren dat hybride slecht te naam staat. Na het uitvoeren van benchmarks bleek ook in de praktijk dat hybride frameworks als Cordova zich niet staande houden tegenover frameworks die naar native compileren.

Van de native frameworks is er een selectie gemaakt op basis van verschillende eisen die aan het framework gesteld werden door de organisatie en op basis van de bestaande situatie van de organsisatie. Uiteindelijk blijven er maar twee grote kandidaten over aan het einde van dat onderzoek. Xamarin en NativeScript. Deze platformen bieden goede prestaties, een bekende programmeertaal en premium support.

6. Aanbevelingen

Op basis van de conclusie kan er de volgende aanbeveling gedaan worden:

Gebruik Xamarin voor het bouwen van cross platform applicaties.

Waar NativeScript een zeer interessant alternatief is, biedt Xamarin veel voordelen boven NativeScript voor de organisatie:

- Xamarin is inbegrepen bij Visual Studio, een tool die binnen de organisatie al beschikbaar is middels een msdn licentie.
- Xamarin maakt gebruik van C#, een taal die het meest gebruikt wordt binnen de infrastructuur van de organisatie.
- Xamarin Test Cloud, deze toolset die geleverd wordt door Xamarin biedt de mogelijkheid om snel en eenvoudig de UI en prestaties van een applicatie op een groot scala aan apparaten te testen. Dit platform is echter ook te gebruiken voor frameworks die los staan van Xamarin.
- Xamarin biedt Xamarin University aan, een leerplatform voor hun framework. Xamarin University biedt een groot pakket aan cursussen en seminars aan. Ook is het mogelijk om een certificering als ‘certified mobile developer’ te behalen binnen dit platform.

7. Literatuur

Adobe Systems Inc. (2016). *Adobe PhoneGap*. Opgeroepen op Maart 17, 2017, van http://phonegap.com/

Adobe Systems Software Ireland Ltd. (2017). *Adobe AIR*. Opgeroepen op Maart 17, 2017, van http://www.adobe.com/nl/products/air.html

Axway. (2017). *Appcellerator*. Opgeroepen op Maart 17, 2017, van https://www.appcelerator.com/

Eberhardt, C., & Price, C. (sd). *PropertyCross*. Opgeroepen op Maart 17, 2017, van http://propertycross.com

Embarcadero Technologies, Inc. (2017). *Delphi*. Opgeroepen op Maart 17, 2017, van https://www.embarcadero.com/products/delphi

Facebook Inc. (2017). *React Native*. Opgeroepen op Maart 17, 2017, van https://facebook.github.io/react-native/

Holland, B. (2016, Juli 14). *NativeScript And Xamarin*. Opgehaald van Nativescript blog: https://www.nativescript.org/blog/nativescript-and-xamarin

Intersoft Solutions. (2017). *Crosslight*. Opgeroepen op Maart 17, 2017, van https://www.intersoftsolutions.com/Crosslight

LTD, C. O. (2012). *Codename One*. Opgeroepen op Maart 17, 2017, van https://www.codenameone.com/

Neomades. (2017). *NeoMAD*. Opgeroepen op Maart 17, 2017, van http://neomades.com/en/crossplatform-mobile-development/neomad4

Progress Software Corporation. (2017). *NativeScript*. Opgeroepen op Maart 17, 2017, van https://www.nativescript.org/

TAU Technologies. (2016, October 22). *Rhodes*. Opgeroepen op Maart 17, 2017, van https://github.com/rhomobile/rhodes

The Apache Software Foundation. (2017). *Apache Cordova*. Opgeroepen op Maart 17, 2017, van http://cordova.apache.org/

The Kivy Organization. (sd). *Kivy*. Opgeroepen op Maart 17, 2017, van https://kivy.org/

The Qt Company. (2017). *Qt*. Opgeroepen op Maart 17, 2017, van https://www.qt.io/

Xamarin Inc. (2017). *Xamarin*. Opgeroepen op Maart 17, 2017, van https://www.xamarin.com/

8. Bijlagen

8.1. Bijlage 1: Geanalyseerde frameworks

Naam	punten	Independent or 'Extension'	Laatste release versie	Laatste release datum	Eigenaar	Hybride/Native	Taal/Talen	Platformen	Tooling	Prijzen (Eenmalig/per maand)	In-house Support	Websites	framework support voor OS updates (Voorbeeld: Android 7.0 en iOS 10)	Android version support	iOS version support
Xamarin	6,5	Independent	Cycle 9*	22/02/2017	Microsoft	Native, Handmatig hybride	C#	iOS, Android, OS X, Windows and Windows Phone	Visual Studio Extensie, Xamarin Studio, Xamarin Test Cloud, Xamarin Insights, Xamarin.Forms	Gratis met visual studio Community. Betaald professional /enterprise licentie. Test cloud start vanaf $99 p/m	Ja	https://www.xamarin.com/	Android: 15 dagen, iOS: -5 dagen	1.6+ (Forms: 4.0.3+)	6.0+ (Forms: 6.1+)
NativeScript	6	Independent	2.5.0	01/02/2017	Telerik	Native, Hybride	JavaScript	Android, iOS (Windows on roadmap)	Visual studio code plugin, NativeScript CLI, Telerik platform	Gratis	Ja, betaald	https://www.nativescript.org/	0 dagen (dynamic API mapping)	4.2+	7+
React Native	5	Independent	0.42.0	01/03/2017	Facebook	Native	JavaScript	Android, iOS	Visual studio code plugin, CLI	Gratis	Nee	https://facebook.github.io/react-native/	Android: onbekend, iOS: -4 dagen	4.1+	8+
Cordova/Phonegap	5	Independent	6.5.0 / 6.4.5	23/01/2017	Apache Software Foundation / Adobe	Hybride	Javascript	Android, iOS, Ubuntu, Windows phone 8 en 8.1, Windows 8.1 en 10, OS X	Cordova CLI / Phonegap CLI, Phonegap desktop app, phonegap developer app, phonegap build	Gratis, Phonegap build vanaf $9,99 p/m	Nee	http://cordova.apache.org/ - http://phonegap.com/	Android: 63 dagen, iOS: 41 dagen	4.1+	7.0+
Titanium	5	Independent	6.0.2.GA	27/02/2017	Appcelerator	Native	JavaScript	iOS, Android, Windows	Appcelerator IDE, Titanium CLI	vanaf $36 p/m	Ja, vanaf enterprise pakketten	https://www.appcelerator.com/			
Qt	4,5	Independent	5.8	23/01/2017	Qt Company	Native	C++	Windows Mobile, Linux, MacOS, Android, iOS, Windows 10	Qt Creator	Gratis, andere pakketten vanaf $295 p/m	Ja, betaald	https://www.qt.io/			
RhoMobile	4	Independent	5.5	14/10/2016	Zebra Technologies	Native, Hybride	JavaScript, Ruby	Windows Mobile, Android, iOS, Windows	RhoStudio, RhoMobile	Gratis	Nee	https://github.com/rhomobile/rhodes			
AIR	4	Independent	24	13/12/2016	Adobe	Native	Flash, ActionScript	Windows, OS X, Android, iOS, BlackBerry	Adobe flash builder	Adobe Creative cloud vanaf €60,49 p/m	Ja	http://www.adobe.com/nl/products/air.html			
NeoMAD	3,5	Independent	4.2.0	27/01/2017	NeoMAD	Native	Java	Android, iOS. Premium: Windows Phone, Windows 8	Eclipse IDE Plugin	Gratis, andere pakketten vanaf €999 p/m	Ja, vanaf €4999 per jaar per dev pakker	http://neomades.com/en/crossplatform-mobile-development/neomad4			
Codename One	3,5	Independent	3.6	07/01/2017	Codename One	Native	Java	Android, iOS	Eclipse/Netbeans/IntelliJ IDE Plugin	Gratis, andere pakketten vanaf €19 p/m	Ja, vanaf $79 p/m	https://www.codenameone.com/			
Crosslight	3	Independent	4.0.5000.327	26/11/2015	Intersoft Solutions	Native	C#	Windows Mobile, Android, iOS, Windows	Geen	vanaf $999	Ja	https://www.intersoftsolutions.com/Crosslight			
Kivy	3	Independent	1.9.1	01/01/2016	Kivy Organization	Native	Python	OS X, Linux, Windows, iOS, Android	Geen	Gratis	Nee	https://kivy.org/			
Delphi	3	Independent	Delphi 10.1 Berlin	20/04/2016	Embarcadero	Native	Object Pascal	Windows, OS X, Android, iOS	Delphi IDE	vanaf €1955,36	Ja, betaald	https://www.embarcadero.com/products/delphi			

= top selectie
= buiten selectie
= punt
= half punt

* https://releases.xamarin.com/stable-release-cycle-9/

8.2. Bijlage 2: Interview Boy

Martin:
Om dan te beginnen met de eerste vraag, welke apps binnen Infoplaza heb jij allemaal aan gewerkt?

Boy:
Aan Buienalarm en Onweeralarm. Nu begin ik aan Meteoplaza en Weerplaza. En toen ik viel onder Moop mobility heb ik ook gewerkt aan een prototype voor reisbewaking. Dus dat je een melding krijgt wanneer je moet uitchecken of overstappen.

Martin:
Nou, je beantwoordde net al (voor het interview): is dat vooral iOS of Android geweest.

Boy:
Ja, alleen Android.

Martin:
Hoe lang doe je gemiddeld over het realiseren van een nieuwe functionaliteit. Dan als voorbeeld, een app activity met dan een lijst erin van nieuws artikelen die je dan kan openklappen met titelbeschrijving en auteur en dergelijken.

Boy:
Dus ook het detailscherm erbij?

Martin:
Ja

Boy:
Dus een lijst, je drukt erop en je krijgt tweede activity eigenlijk.

Martin:
Ja

Boy:
Even kijken, want ik had eerst gedacht "lijst met nieuwsartikelen". Het licht eraan hoeveel informatie er in zit, hoe complex de layout is en of er animaties moeten zijn of zo. Inclusief detail zou ik zeggen een dag, misschien anderhalf om dat toe te voegen als de API gewoon beschikbaar is. Denk ik, maar ja dat verschilt per keer. Soms loop je tegen iets aan en dan ben je drie dagen bezig, een andere keer ben je in 3 uurtjes klaar. Dus gemiddeld zou ik zeggen anderhalve dag.

Martin:
Welke mogelijke nadelen ervaar je van het apart ontwikkelen in Android en iOS van de apps?

Boy:
Dat het gemeenschappelijke gedeelte wat je bij cross platform samen krijgt, dat mis je. Dat je van uit één codebase de API kunt aanspreken. Verder waarschijnlijk dat je ook twee verschillende soorten bugs krijgt links en rechts. Maar, zelf geen ervaring mee. Ik heb een beetje met Appcelerator zitten werken, dat is Javascript en Javascript dat kende ik toen nog minder dan nu natuurlijk.

Martin:
Ja klopt, Appcelerator die ben ik ook tijdens het onderzoek langs gekomen. Die schijnt ook vrij populair te zijn naar mijn weten. Volgens mij hadden ze het Titanium framework, zo heette dat.

Boy:
Ja. Andere nadelen verder. Ik weet het niet zeker, maar misschien duurt het iets langer als je ze allebei apart wilt doen. Maar dat weet ik niet zeker, omdat ik ook heb begrepen, afhankelijk van welk platform je pakt, dat je twee keer de UI apart moet bouwen. Allebei sowieso. Dus dat doe je sowieso. Dus daar zie ik geen voordeel in.

Martin:
Wat ik wel begreep van anderen interviews tot nu toe is dat het wel vaak het geval was dat, omdat je in de eerste instantie al tegen problemen aan loopt met zo'n applicatie en je al bepaalde keuzes hebt gemaakt voor de API bijvoorbeeld. Dat kan je wel weer meenemen uiteindelijk naar de andere platformen zodra dat staat. Op die manier wordt er vaak gezegd dat het minder tijd kost, maar het is nu beetje lastig te zeggen omdat jij toch vooral met je eigen deel bezig bent geweest volgens mij.

Boy:
Maar je zegt dat die problemen waar je tegenaan loopt bij de API dat dat eigenlijk alleen maar is als je ze na elkaar ontwikkelt, maar niet als je ze tegelijkertijd doet.

Martin:
Dat is wel waar ja. Nu is het zo dat ze hier veel na elkaar ontwikkeld hebben. Al is nu wel bij Buienradar en Onweeralarm dat die veel naast elkaar ontwikkeld worden begreep ik.

Boy:
Ik weet niet hoe dat begonnen is want ik heb het over genomen. Dus ik weet niet of ze eentje eerst hebben gedaan en dan de ander. Dat lijkt me wel. Zo begint iedereen eigenlijk.

Martin:
Ja, klopt. Dan naar de volgende vraag. Wat zijn de voordelen van apart ontwikkelen in Android en iOS?

Boy:
De nieuwste SDK-mogelijkheden heb je gelijk tot je beschikking. Ik heb een beetje hier en daar zitten kijken, maar cross-platform dingen. Volgens mij doen er een aantal ook wel dat je mixed kunt ontwikkelen, dus dat je ook gewoon native en de cross-platform dingen door elkaar kunt gebruiken. Misschien heb je dan die features ook tegelijkertijd.

Martin:
Ik weet dat bijvoorbeeld NativeScript, dat is misschien nog wel populairder dan Titanium, day zero ondersteuning heeft voor SDK-releases omdat die een soort dynamische mapping doen over de API.

Boy:
Oh, okee. Een paar jaar geleden volgens mij niet, bij Appcelerator.

Martin:
Nee klopt, maar goed. Die frameworks die veranderen ook iedere maand zo'n beetje al significant. Het is wel handig om onderscheid ook te maken met bijvoorbeeld Cordova. Dingen als React native, Titanium, Xamarin, dat zijn cross platform frameworks die naar native code toe vertalen. Dus die compileren daadwerkelijk, in zeker mate, daarnaartoe volgens mij. Sommige draaien nog wel een speciale runtime daar bovenop, maar dat roept wel dan gewoon native Java componenten aan uiteindelijk.

Boy:
Dus dan heb je eigenlijk niet echt problemen met snelheid of wat dan ook.

Martin:
Nou, in ieder geval aanzienlijk minder. Behalve in het geval van React, die doet het nog verassend traag. Ik heb daar wel leuk benchmarks van, ik kan je die zo wel laten zien als je wilt.

Boy:
Ja, graag.

Martin:
Nou, om dan nu verder een beetje te speculeren richting de cross-platform frameworks, Welke voordelen verwacht je te kunnen halen uit het gebruik van een cross platform framework?

Boy:
Bij Cordova was het gewoon een webpagina vroeger. Ik weet niet hoe het op dit moment is.

Martin:
Nog steeds.

Boy:
Volgens mij zal responsiveness best wel tegenvallen. Het is wel heel makkelijk te updaten volgens mij. Ik weet niet of je dat ook online zou kunnen doen, dus dat je geen app update nodig hebt voor dat soort dingen.

Martin:
Klopt, dat heb ik weleens vaker gezien ja.

Boy:
Dus dat is natuurlijk interessant want die site zet je online en dan is deze voor iedereen beschikbaar. Een app moet je nog altijd updaten, dat is vervelend. En de andere versie die gewoon naar native compileert die zal een stuk sneller zijn. Sneller nieuwe features van de SDK ondersteunen. Moeilijker te worden onderscheiden van native gebouwde apps, dat lijkt me een voordeel daarin. Vandaag, als ik ergens mee zou beginnen dan zou ik inderdaad iets van React Native of Ionic kiezen, al weet ik niet precies hoe Ionic werkt?

Martin:
Ionic is op basis van Cordova.

Boy:
Ok, dan zou ik daar wegblijven. Ik kende een stagair die zei: "Oh, daar heb ik mee gewerkt. Dat was wel leuk." Maar als dat op web basis is dan doe ik dat niet.

Martin:
Ik weet wel dat die heel populair is inderdaad. Het is wel een overweging die je moet maken. Die web frameworks staan er om bekend dat ze traag zijn, maar tegelijkertijd bieden ze wel hele snelle ontwikkeling en als je iets simpels hebt als een lijstje met nieuwsartikelen, dan kan je dat heel snel in zo'n hybride applicatie doen.

Boy:
Dat klopt, maar waarom dan geen website vraag ik me af in zo'n situatie. Als je bijvoorbeeld kijkt naar Tweakers, hoe snel dat is. Ik ben het helemaal mee eens dat die app die ze toen gemaakt hadden, dat ze daarmee zijn ze mee gestopt. Ze hebben nu alles gewoon op de website, dat is ideaal.

Martin:
Klopt, maar goed dan is het toch de fancyheid van het hebben van een app die je kan downloaden.

Boy:
Ja dat houdt misschien een beetje op dat gedeelte. Ik denk dat je dat echt moet gaan gebruiken voor wanneer het echt nodig is. En bij een Buienalarm en een Onweeralarm werkt het nou eenmaal beter met native componenten.

Martin:
Dan heb je de GPU-aandrijving nodig, die heb je niet echt veel bij web based.

Boy:
En ook dat met inzoomen, dat gaat allemaal net wat lekkerder met native.

Martin:
Om ook weer de andere kant te belichten, welke nadelen voorzie je bij het gebruik van een cross platform framework?

Boy:
Nou, dat je soms niet de laatste features hebt. Je moet alsnog allebei de UI's gaan ontwikkelen om te zorgen dat het er voor Android en iOS native uitziet voor de gebruiker, wat ze gewend zijn. Of je moet ervoor kiezen dat het er compleet anders uit ziet. Maar daar ben ik nooit zo'n voorstander van.
Volgens mij is het vooral de look en feel en de beschikbaarheid van de SDK. Ik vind het heel moeilijk om te bepalen, omdat ik super weinig ervaring heb met cross platform. Ik zou het graag een keer willen doen. Ik ken ook een ex-collega, die is leraar geweest op de hogeschool en die gaf les in Xamarin. Hij was zelf echter compleet tegen, hij vindt dat helemaal niks.

Martin:
Het is ook lastig omdat er zó veel frameworks zijn waar je uit kan kiezen, het is echt absurd.

Boy:
Ja, je zou echt niet meer weten wat je moet kiezen. React zou ik misschien niet kiezen omdat het van Facebook is.

Martin:
Voor mijn onderzoek moet ik wel alles openhouden, dus ik probeer zo objectief mogelijk ernaar te kijken.
Maar we raken echter een beetje off topic voor het interview volgens mij, die op dit punt in principe afgerond is. Dus, hartelijk bedankt daarvoor.

Boy:
Ja, graag gedaan.

8.3. Bijlage 3: Interview Esmeralda

Martin:
Welke app projecten zijn er op dit moment actief?

Esmeralda:
Nou, we hebben eigenlijk altijd wel verschillende projecten rond apps actief. Momenteel werken we aan een grote update van de weerapps. Zowel onweeralarm als buienalert en Weerplaza staan binnenkort op de planning. Voor Buienalert hebben we net versie 1.5.0 afgerond en in de play store gezet. 1.5.1 moet er waarschijnlijk nog wel een keer achteraankomen in april, maar we gaan eerst eventjes de andere apps updaten. iOS loopt een beetje achter op schema, maar die kan verwacht ik toch ook wel volgende week een keer richting de store. En daarna volgt onweer alarm. En daarna dus een wat groter project rond Weerplaza. We hebben nu twee apps: Meteoplaza en Weerplaza. Meteoplaza is de meertalige versie eigenlijk van Weerplaza maar die is apart neergezet en dat gaan we één app maken en dat is best wel groot project. Dus dat is iets wat loopt. Daarnaast is Verkeerplaza eigenlijk net afgerond, daar was Hans voor verantwoordelijk. De ov-apps, dan heb ik het over de klanten van Moop: GVB, RET, Transdev, is eigenlijk een continu proces. Daar werken we in verschillende sprints aan en hebben we weer allerlei afspraken staan.

Martin:
Wie werken er aan deze mobiele applicaties en wat zijn hun rollen daarbij.

Esmeralda:
Ja, wie werken eraan. Aan de apps zelf natuurlijk de developers. Voor Android hebben wij extern Boy, die werkt aan de weerapps. Voor iOS is dat Paul. Daarnaast zijn er natuurlijk niet alleen de apps waar je aan moet werken maar ook API's moeten klaarstaan en dat soort dingen dat ligt toch grotendeels bij Sjoerd. Soms wordt Noud er ook weleens bij betrokken als het om radar dingen gaat. Verder voor de Moop apps werken we in sprints met Appnormal, een extern bureau. Daar doet vooral Sven de datakant. Daar zit ik een beetje als projectmanager bij. Hans hebben we natuurlijk ook nog, die heeft ook zijn projectmanagers rol en die doet ook nog weleens wat voor design. Voor design hebben we ook Leo en die zal zich vooral bezighouden met weerplaza, met de nieuwe app.

Martin:
Wat is de loopduur van deze projecten? En dan bedoel ik vanaf het begin tot nu. Of ben je daar niet bekend mee?

Esmeralda:
Je bedoelt echt van wanneer buienalert gelanceerd is voor het eerst? Uit mijn hoofd is dat afgelopen zomer of die zomer daarvoor. Die zomer daarvoor dacht ik, dat moet ik eigenlijk even nakijken wanneer precies. Ik ben in ieder geval erna in dienst gekomen en toen bestonden alle drie de apps al. Weerplaza is sowieso de oudste app, daarna buienalert en daarna onweeralarm. De reden dat die in de zomer zijn gelanceerd is omdat onweer natuurlijk heel relevant is dan, veel relevanter dan in de winter dus dat is wel zo gepland. En ja, Weerplaza is gewoon veel completer en de app is echt al enkele jaren in ontwikkeling, al zou ik ook even precies moeten nakijken wat daar de startdatum van is. Maar meestal werk je dus qua loopduur voor een project echt in versies. Dat

zijn vaak gewoon blokken van enkele weken of enkele maanden, doorgaans enkele weken maar het loopt nog weleens uit.

Martin:
Om daar mee verder te gaan, wat is de gemiddelde duur van projecten waarin een nieuwe app wordt gerealiseerd tot aan de eerste 'release'?

Esmeralda:
Een volledig nieuwe app?

Martin:
Ja

Esmeralda:
Ja, dat vind ik heel lastig eigenlijk in te schatten. Volgens mij is het heel erg verschillend, afhankelijk van hoe complex een app is. Destijds werkten we toen met beacons en moest alles daarop afgestemd worden. Daar zat een deel van de reisplanner in, daar konden we ook wel weer een en ander kopiëren van andere apps en ik denk niet dat het ene project met het andere te vergelijken is. Zeker een app als Onweeralarm is aanzienlijk kleiner dan Weerplaza. Ik denk dat je het niet zo kan zeggen. Als ik kijk naar de projecten die nu lopen is het inderdaad zo dat het ene project enkele weken duurt, het andere project duurt echt wel een paar maanden. Dus het zit een beetje daartussenin.

Martin:
De Android en de iOS apps, worden die vaak gezamenlijk gerealiseerd of na elkaar. Dat als eerste bijvoorbeeld een iOS versie werd en dan later pas een Android versie of loopt dat meestal wel aardig gelijk.

Esmeralda:
Ook dat is verschillend, We hebben bijvoorbeeld laatst Verkeerplaza helemaal vanaf het begin af aan eerst helemaal gemaakt voor iOS. Daarna is Android er achteraangegaan terwijl we bijvoorbeeld bij de weerapps doorgaans tegelijk aan de apps werken. Dan kan het wel zo zijn, zoals nu ook met Buienalart is gebeurd, dat Android eerder klaar is en dan gaat hij naar de store en dat we bij iOS tegen wat hickups aanlopen en dan duurt dat net iets langer. Dat gebeurt dan ook weleens andersom, het is een beetje afhankelijk van het type project. Een bestaande app, zoals nu met de weer dingen, daar werken we doorgaans wel tegelijk aan zodat je ook qua versiebeheer een beetje hetzelfde blijft. Maar als je iets echt helemaal opnieuw gaan neerzetten dan gebeurt dat ook weleens na elkaar.

Martin:
In het geval dat het inderdaad na elkaar wordt ontwikkeld, stel dat de iOS app eerst wordt gemaakt en daarna de Android app, kost dit dan relatief aan de iOS app kost het Android bouwen dan langer of korter meestal wel gelijk?

Esmeralda:
Die is dan wel korter.

Martin:
Okee, dus Android wordt meestal sneller gerealiseerd?

Esmeralda:
Naja niet per se Android, maar de eerste app duurt langer dan de tweede.

Martin:
Ah, op die manier.

Esmeralda:
Bij het vorige project, Verkeerplaza, was het inderdaad eerst de iOS en daarna Android, maar als het andersom is zal het andersom ook zijn. Dan zal de iOS app minder tijd kosten.

Martin:
Waardoor denk jij dat de tweede app sneller ontwikkeld kan worden?

Esmeralda:
Nou, eigenlijk omdat je bij de eerste app vaak tegen bepaalde problemen aan loopt. Bijvoorbeeld met de data kant, hoe dat communiceert met elkaar. Of er wordt dan toch nog een API aangepast, daar moet de app ontwikkelaar dan op wachten. Of je komt daar zelf achter dus dan vertraagd dat het proces bij de eerste app. Dat is bij de tweede app niet aan de orde, daar is eigenlijk het hele verhaal uitgeschreven. Die kan aan de slag met de definitieve API, en het definitieve design. Bepaalde problemen waar de eerste ontwikkelaar tegenaan is gelopen en die daar bijvoorbeeld een slimme oplossing voor heeft gevonden, die communiceert dan wel met de andere ontwikkelaar om te zeggen van: "hey ja, kijk. Ik heb het zo opgelost. Dus dat zou je zo kunnen implementeren" en ja, dat gaat gewoon aanzienlijk sneller.

Martin:
Zijn er specifieke problemen die je herkent in het ontwikkelen van apps voor Android en iOS, dus dat je inderdaad voor allebei moet ontwikkelen.

Esmeralda:
Wat bedoel je precies?

Martin:
Omdat je toch steeds met twee projecten bezig bent, de problemen waar je daarmee tegenaan loopt.

Esmeralda:
Bedoel je dan over echt specifieke Android problemen en specifieke iOS problemen of meer het proces?

Martin:
Meer het proces ja.

Esmeralda:
Ik vind het op zich persoonlijk niet echt een probleem om te moeten schakelen tussen de verschillende projecten. Je communiceert voor het ene gewoon met developer A en het andere met developer B. We hebben bijvoorbeeld weleens in het verleden een developer gehad voor een bepaald platform die wat minder, laat ik het zo zeggen, communicatief was en dat vermoeilijkt dan het proces met dat platform wel. Dan werkt bijvoorbeeld iOS op dat moment een stuk soepeler ook qua proces. De ene developer is gewoon de andere ook niet. De een is misschien wel gewoon iets beter dan de ander. Dus dat zijn wel dingen waar je dan tegen aan loopt. Doorgaans vind ik het eigenlijk wel soepel gaan.

Martin:
Ja. Dan als laatste nog: welke effecten verwacht je van het gebruik van een cross platform framework bij de ontwikkeling van apps.

Esmeralda:
Ik denk dat je bedoelt dan dat we alles gewoon op één platform kunnen?

Martin:
Ja, maar dus ook dan het proces omdat je dan in theorie dan maar een developer aanstuurt, of 1 groep developers.

Esmeralda:
Nou, ik denk dat bij het maken van een nieuwe app dat je op zich net zo veel tijd kwijt zou zijn als dat je normaal dus bezig bent met iOS, of met één app. Dus dat zou het proces om vervolgens meerdere apps in een keer neer te zetten best wel kunnen versnellen denk ik. Volgens mij is het ook een stuk makkelijker om een platform als Windows mee te gaan nemen, wat we nu eigenlijk helemaal niet meer doen. Daar hebben we gewoon geen developer voor in huis en daar lopen we ook gewoon tegen andere problemen aan die je dan niet hebt. Dan is het gewoon opslaan als, toch? Dat heb ik begrepen. Opslaan als Windows. Dus dat is wel winst die je met het nieuwe framework dan zou kunnen behalen. En het proces zelf, ik denk dat je dus veel makkelijker meerdere apps naast elkaar neer kan zetten voor verschillende platformen. Maar dat je uiteindelijk tegen dezelfde problemen aan loopt met API's en algemeen ontwikkelingen. Dus ik denk dat het neerzetten van een nieuwe app niet per se sneller zal gaan dan het neerzetten van een nieuwe app op iOS bijvoorbeeld. Dat denk ik. Maar ja, de tijd zal dat moeten leren.

Martin:
Okee. Heb je verder misschien nog iets toe te voegen?

Esmeralda:
Ik zou het zo niet weten nee.

Martin:
Okee. Nou, hartstikke bedankt.

Esmeralda:
Ja, graag gedaan.

8.4. Bijlage 4: Interview Paul

Martin:
De allereerste vraag is: aan welke apps heb je gewerkt bij Infoplaza?

Paul:
Nou, dat is Ouienalert, Onweeralarm, Verkeerplaza en de credit for miles app. dat zijn denk ik de vier belangrijkste apps waar ik hieraan heb gewerkt.

Martin:
En heb je hierbij vooral met iOS of ook wel met Android gewerkt.

Paul:
Volledig iOS, geen Android

Martin:
Dan vallen er hier een paar vragen weg, die zijn wat specifieker voor als er met allebei gewerkt is. Hoe lang doe je gemiddeld over het realiseren van een nieuwe functionaliteit. Hierbij het ik dan een kleine beschrijving staan om even een idee te vormen: stel, je zou een overzicht maken met nieuwsartikelen waarin je zou kunnen openklappen om daarna specifieke informatie voor de artikelen te tonen. Hoe lang je met zoiets bezig zijn?

Paul:
Ja, dat is altijd lastig. Dat is altijd lastig te zeggen. Als je iets simpels hebt met nieuwsartikelen en die moeten openen is dat in een paar dagen klaar. Maar als daar advertenties bij moeten komen en mensen moeten kunnen reageren op die nieuwsartikelen en kunnen sorteren, en dan ...

Martin:
Laten we het dan op de simpele versie houden, zeg gewoon: lijstje, openen en text.

Paul:
Als dat goed aangeleverd wordt kost dat max. 2 dagen.

Martin:
Welke mogelijke nadelen herken je aan het apart ontwikkelen in Android en iOS?

Paul:
Dat je niet even snel ontwikkeld. Dus soms is het zo dat de iOS versie eerder klaar is of de Android versie is eerder klaar. Wat ook nog zo kan zijn is dat functionaliteit uiteenloopt. Dat niet dezelfde functionaliteit op iOS als op Android heeft. Gedrag kan er net iets anders uitzien tussen iOS en Android. Dat kan anders reageren, dat soort zaken. Dat proberen we natuurlijk te ondervangen door goede specs neer te leggen en dan te ontwikkelen volgens deze specs, toch zie je dat het dan wel uit elkaar kán lopen. Soms is dat niet erg, omdat dingen nou eenmaal anders gaan op Android als op iOS. Soms is het wel vervelend want dan wil je gewoon dezelfde functionaliteit op beide platformen. Dus dat is een van de nadelen. Een ander nadeel is dat je alles drie keer moet maken. Je moet je code maken voor iOS en Android. Dus je hebt 2 verschillende codebases en beide moeten onderhouden. En als je een functionaliteit erbij wilt maken, dan zul je dat in beide codebases moeten doen. Je moet ook apart testen. Het kan zijn dat bugs op Android niet in iOS zitten. Want ja, dat is een heel andere codebase en

het kan ook andersom. Dat dingen die in iOS fout gaan of bugs in zitten, die zitten niet in Android. Je moet ook alles dubbel testen. Misschien ook wel als je het tegelijk zou doen, maar dat zijn dan wel de nadelen. Het kost meer tijd.

Martin:
Welke voordelen dingen merk je wel aan het apart ontwikkelen in Android en iOS.

Paul:
Nou, het zijn twee aparte 'dingen'. Je ontwikkelt specifiek voor dat platform en vaak zie je dat als je specifiek voor een platform ontwikkelt dat die app gewoon vlotter werkt, sneller werkt, zo werkt dat als de gebruiker verwacht omdat het hetzelfde werkt als op andere apps op datzelfde platform. Dus dat zijn de voordelen, dat het vaak soepeler werkt en het ziet er ook vaak uit zoals je verwacht op dat platform. Oh ja, en een voordeel is dat je maar van een platform kennis hoeft te hebben. Ik hoef niet te weten, omdat ik alleen voor iOS ontwikkel, wat de beperkingen van Android zijn of de mogelijkheden zijn. Dat is andersom ook, degene die de Android versie ontwikkeld, die hoeft niks van iOS te weten. Die hoeft niet te weten hoe je daar platform-specifiek dingen kunt doen. Dat zijn ook nog wel voordelen.

Martin:
Zeker. Dan om door te gaan omtrent cross-platform frameworks: welke voordelen verwacht je te kunnen halen uit het gebruik van een cross-platform framework?

Paul:
Nou, dat is eigenlijk het tegenovergestelde van de nadelen. Het voordeel is dat je 1 codebase hebt zeg maar. Dus als jij een wijziging wilt doen, hoef je die in theorie maar op 1 plek te doen en dan zou die voor beide platformen na het compileren opgelost zijn. Het voordeel is ook dat het gedrag hetzelfde is omdat je je achterliggende datamodel maar één model is, dus de hele werking van de app die zal op Android en iOS grotendeels hetzelfde zijn. Een bug die er eventueel in zou zitten die zit en in Android, en in iOS. De kans dat je een bug vindt is groter omdat je kan hem sneller vinden want hij zit op beide platformen. Maar ik denk voornamelijk dat je maar een codebase hebt, dat dat het grootste voordeel is want dat hoef je ook maar een keer te onderhouden. Dus in plaats van dat je nu 2 ontwikkelaars aan het werk hebt, heb je dan maar een ontwikkelaar aan het werk. De vraag is, en dan komen we denk ik gelijk bij punt 10, welke nadelen ondervind je met cross-platform, ja dat duurt wel langer voordat het klaar is. Je kan niet parallel ontwikkelen want het duurt gewoon langer, omdat het lastig is om iets te maken wat op die beide platformen werkt. Dat klinkt in theorie heel leuk, maar in de praktijk werkt dat eigenlijk niet en dan moet je een soort work-around gaan bedenken van: oh ja, dit moet op alletwee die platformen werken. Hoe krijg ik dat voor elkaar? Het is wat complexer want je moet van beide platformen moet je ook weten wat de mogelijkheden zijn. Hoe je dingen oplost. En vaak zie je dat je een beetje universele oplossing krijgt en zou je een oplossingen kunnen krijgen die die de gebruiker niet gewend is op dat platform. Vaak zie je ook dat universele apps ook wat trager zijn omdat er nogal een laag tussen zit. Je kan vaak de native dingen niet aanspreken van dat toestel. Je loopt ook vaak wat achter. Stel je voor, er komt een nieuwe versie uit van Android met nieuwe mogelijkheden dan kan je die pas gebruiken op het moment dat die algemene tool is aangepast. Sterker nog, je kan het slechtste van beide werelden maar gebruiken. Stel je voor Android heeft bepaalde functionaliteit niet of dat kan gewoon niet, dan kan je die ook niet op iOS gebruiken want je hebt een universele tool. Dus je werkt altijd met de subset van het slechtste wat er is. Een beperkte subset. En dat is ook een nadeel.

Ik denk dat je het verschil een beetje kunt zien in bijvoorbeeld automerken, bijvoorbeeld BMW. Ik heb een jaar een BMW gereden, en als ik dan wat had met die auto dan ging ik ook naar de BMW-dealer, want die had daar kennis van. Die had geen kennis van nadere merken ... "dat is goed.", "BMW, daar hebben wij kennis van.", "oh is dat het probleem?", "Dan los ik dat zo op.". Maar ik had net zo goed met die BMW naar universele garage om de hoek kunnen gaan. Hadden ze dat misschien ook opgelost, misschien niet helemaal. Misschien eerst nog een keertje verkeerd opgelost maar op een gegeven moment hadden ze het wel opgelapt en dan had het wel weer gewerkt, maar de meeste mensen die een bepaald merk auto hebben die gaan ook naar een dealer van dat merk of naar een garage van dat merk puur vanwege dat ze daar de kennis hebben. Dat is denk ik met deze platformen ook zo. Ja natuurlijk je kan een universele garage pakken, een universele app maken maar dat zal nooit zo goed zijn als dat je dat los van elkaar houdt. Dus dat is het verhaal wat mij betreft. Dus ja het kan wel, nee het zal niet zo goed zijn als dat je het los houdt.

Martin:
Ja, duidelijk. Hartstikke bedankt.

8.5. Bijlage 5: Interview Hans

Martin:
Welke projecten binnen Infoplaza werk je aan?

Hans:
De Verkeerplaza app, Android en iOS. De iOS versie is live en Android moet nog 1 of 2 versies omhoog. Dus dat is het project wat ik doe. Dan heb ik een stukje vormgeving gedaan, of user experience, omdat toen ik hier startte in juli ongeveer, die app al min of meer lanceerbaar was. Maar daar was eigenlijk niet goed over nagedacht. Niet dat hij nu perfect is, maar hij is in ieder geval leefbaar.
Verder heb ik veel gedaan hier voor de ontwikkeling van de website van Weerplaza, met name aan de voorkant. De strategie, wat zet je waar, waarom, enzovoorts. Alles wat ik doe is in samenwerking, dus ik doe vaak suggesties of geef advies en dan is het maar de vraag of het dan ook echt opgepikt zal worden. Je moet ook rekening houden met techniek en al die andere dingen.
Ik ben met René bezig met Vialis, met een project dat heet Schwung. Kort gezegd, dat is voor fietsers voor gemeenten om een soort van app te ontwikkelen waar fietsers sneller mee door groen kunnen. Dan krijgen fietsers prioriteit boven auto's. Voor gemeentes is dat interessant. En daar werk ik samen met iemand van Vialis aan het concept Hoe het zou moeten werken, wat we willen, wat we ermee kunnen, wat het voor de gebruikers oplevert maar ook voor de gemeente.
Verder, voor Infoplaza zelf denk ik mee aan sommige dingen. Stukjes strategie maar ook ook uitwerken van dashboards voor Prorail heb ik gedaan.

Martin:
Ok. Ik weet dat Esmeralda hier ook wel wat moeite mee had maar: wat is een beetje de gemiddelde loopduur van de verschillende projecten waaraan je hebt gewerkt.

Hans:
Gemiddelde? Nou als je kijkt naar Verkeerplaza waren ze er eigenlijk al mee bezig toen ik juni/juli hierzo kwam. Dan heb je natuurlijk zomervakantie. Voordat de iOS app af was, was het ongeveer november dus een maand of 5. Android iets sneller, maar die is eigenlijk ook nog niet helemaal gelijkgetrokken dus ik denk 4 maanden. Dat is best wel lang.

Martin:
Ja, klopt.

Hans:
Jij bent natuurlijk specifiek geïnteresseerd in app ontwikkeling.

Martin:
Ja.

Hans:
Ik denk toch wel 5 maanden. Echt ontwikkeling dan hé, dus daar gaat nog wel een hele tijd voor zitten en nazorg en dergelijken.

Martin:
Ja klopt, alle requirements opstellen en dat soort zaken.

Hans:
Als we echt gaan ontwikkelen, dus echt het programmeren dan denk ik toch wel 4-5 maanden.

Martin:
Worden de Android en iOS apps vaak gezamenlijk gerealiseerd of na elkaar?

Hans:
Wat ik nu zie is allemaal na elkaar. Meestal iOS eerst omdat Paul toch het meest dichtbij de organisatie zit.

Martin:
Intern is het ook heel veel iOS begreep ik. Sjoerd bijvoorbeeld heeft veel gewerkt aan de iOS apps.

Hans:
Ja, de kennis ligt het meest dicht bij huis wat betreft iOS. Meestal, maar dat zal wel gaan veranderen, werd er gedacht: 'Oh, als je iOS hebt dan kun je een Android applicatie ook wel verder uitrollen.' Dat scheelt misschien wel iets van tijd, maar uiteindelijke wordt het een na het ander gerealiseerd. Buienalert en Onweeralarm lopen de updates nu wel min of meer synchroon. Maar het heeft dus met interne kennis te maken en ook vaak met beschikbaarheid van capaciteiten. Dat is ook toch wel een probleem.

Martin:
Het bouwen van deze apps na elkaar, kost dat over het algemeen minder tijd, meer tijd of enigszins evenveel tijd om de tweede te maken.

Hans:
Van iOS naar Android ofzo?

Martin:
Ja

Hans:
Wat ik al zei, volgens mij gaat Android op basis van de kennis van iOS iets sneller, omdat je dan toch weet van: let hier eens op, let daar eens op. Uiteindelijk is het toch vaak een specifiek dingetje wat je moet oplossen. Bijvoorbeeld een HD-radar binnenhalen of, in het geval van verkeerplaza maken we gebruik van de standaard kaarten die aanwezig zijn op een smartphone. En dan blijkt dan toch dat de Apple kaarten net even iets handiger werkt dan die van Android. Met renders die net anders gaan enzovoorts. Dus ik denk dat het na elkaar ontwikkeling je iets van efficiëntie oplevert, maar je uiteindelijk toch tegen specifieke dingen aan loopt die bij iOS of Android horen.

Martin:
Je hebt nu inderdaad de redenen waarom al uitgelegd op zich. Zijn er verder specifieke problemen die je herkent? Dus met het apart ontwikkelen van Android en iOS?

Hans:
Nou, je maakt natuurlijk niet gebruik van het hoofd van één iemand. Dus voor iOS hebben we dan met name Paul, Sjoerd kan daar ondersteunen. Voor Android hebben we verschillende mensen. Soms moet je toch een bepaalde aanpak hebben om iets te kunnen oplossen. En dan zie je niet een persoon gebruikt voor Android en iOS. Omdat je verschillende personen gebruikt kan de aanpak net iets anders zijn, dus daar kan je ook op verliezen als het ware. Voor specifieke oplossingen die je dan maakt, bijvoorbeeld voor een app in een van de twee dingen, kan het zijn dat een andere daar heel veel moeite mee heeft omdat die dat net niet snapt ofzo. Dat kan best. Misschien zou het in sommige gevallen efficiënter zijn om met Android te beginnen en dan iOS, maar je weet het niet. Maar het zou kunnen. Bij buienalert bijvoorbeeld is Boy, die doet dat Android, vrij snel met een oplossing gekomen om voor de Benelux de HD-radar in te kunnen laden en als je dan gaat uitzoomen krijg je de wereldradar. Daar liep Paul erg op vast, dat kan. Dat is een specifiek iets of het kan ook aan de ontwikkelaar liggen. Ja, dat is niet om iemand de schuld of iets te geven, maar dat is omdat als je kennis niet hebt je zelf soms moeilijk kan inschatten: is het nou moeilijker op dit device, of dat. Of hoeveel uur kost iets. Ik weet het niet.

Martin:
Welke effecten verwacht jij uit het gebruiken van een cross-platform framework?

Hans:
Het meest voor de hand liggende is denk ik efficiëntie. Ik denk ook wel dat je weer nieuwe inzichten krijgt, misschien kun je dit wel verder trekken dan alleen te denken aan apps. Misschien komen er interessante dingen naar boven die we nu nog niet zien. Dat verwacht ik wel, dat er meer uitkomt dan alleen efficiëntie.

Martin:
Op zich is dat wat René inderdaad ook noemde als mogelijke wens als we specifiek voor Xamarin gaan. Dan kan je gewoon C# libraries schrijven die je misschien ook in de ASP kant kan gooien, dus dat het in de website en in de applicaties één backend gebruikt in feite.

Hans:
Dus, dat je van tevoren eigenlijk niet helemaal had verwacht maar dat gaandeweg wel iets kan zijn. Dus ik denk efficiëntie en daarmee tijdwinst, kostenbesparing. Dat zijn voor de hand liggende dingen.

Martin:
Kan je daar ook misschien nog nadelige effecten bij voorzien?

Hans:
Die zijn er uiteraard. Ik denk dat je, omdat je cross platform gaat werken, misschien iets wat bijvoorbeeld heel goed werkt in Android of in iOS daardoor niet meer gebruik van kan maken. Dus dat je misschien dingen gaat laten liggen puur omdat je cross wil werken, dus dat kan best zijn. Dat iets wat heel goed werkt op Android dan straks wat minder goed werkt.

Martin:
Bijvoorbeeld iOS specifieke functionaliteiten zoals jij al in het design liet zien, dat je op de lockscreen zo'n venstertje krijgt met verschillende knoppen erin. Dat is iets dat echt iOS specifiek is, dat is 'not done' in Android eigenlijk. Dat soort dingen misschien?

Hans:
Ja, misschien kan dat wel een reden zijn om dan te zeggen: Ja, dan moeten we misschien toch niet cross. Dat kan een uitkomst zijn. Dus ja, je zult dingen moet laten liggen omdat je cross gaat werken denk ik. Dat is het nadeel.

Martin:
Ja dat zijn allemaal dingen die je wel mee moet nemen. En in theorie kan ook uit het onderzoek komen: cross-platform moet je niet gaan doen. Dat gaat niet helpen.

Hans:
Ja, of pas als dit, dat of zus of zo mogelijk is op cross-platform.

Martin:
Ja klopt, ik weet dat bijvoorbeeld Xamarin mogelijkheden heeft om daadwerkelijk specifiek voor bepaalde systemen ook nog onderdelen bij te maken. Dus dat je inderdaad juist weer wel die notificaties bijvoorbeeld in iOS wel zou kunnen doen en in Android weg kan laten.

Hans:
Ja. Ik denk dat het enorm leerzaam is gewoon. Niet alleen voor jou maar voor ons ook. En voor anderen denk ik ook.

Martin:
Je had gezegd dat je ook veel met user experience bezig bent. Nu heb ik voor de OVplaza app ook al aardig wat van gezien ondertussen. Daar heb ik ook wat specifieke vragen voor opgesteld. In de eerste instantie heb ik hier staan: welke apps binnen Infoplaza heb je aan gewerkt? maar goed, dat hebben we al besproken. Laten we dan zeggen inderdaad, specifiek user experience design.

Hans:
Buienalert komt ook voor een deel van mijn kant nog. Maar dat is ook weer volledig in samenwerking. Je kijkt natuurlijk elke keer van: wat is er mogelijk? Dus ze willen die regen dan op een bepaalde manier laten zien, maar je wilt ook de kaart laten zien. Ergens moet je een keer een keuze maken. Toen hebben we bedacht van nou, dat moet dan er overheen kunnen worden getrokken en weer terug. Dat was toen een goed idee, we staan er nu niet meer zo achter. Verkeerplaza dan en Schwung.

Martin:
Bij het ontwerpen hiervan, heb je dan ook echt wel het specifieke platform altijd in gedachten of probeer je dat een beetje universeel over na te denken.

Hans:
Nee, ik denk niet. Laat ik het zo zeggen, technisch gezien weet ik heel weinig van iOS en Android. En ik wil dat graag zo houden. Ik kijk niet specifiek naar: kunnen we dit op iOS of Android. Helemaal niet eigenlijk. Dan laat ik me liever terugfluiten door een ontwikkelaar of iemand anders dan dat ik me te veel daar mee bezig hou. Ik vind het ook een beperking.

Martin:
Android en iOS hebben vanuit Apple en google specifieke design patterns die ze proberen aan te bevelen om te gebruiken. Maar probeer je die ook wel terug te laten komen in je ontwerpen?

Hans:
Ja, maar dat zijn ook eigenlijk meestal wel bekende use cases. Je weet ook wel, de meeste liggen er voor de hand en uiteraard volg je die en dan lees je er ook artikelen over en dergelijke. Dus ja, dat wel. Maar we hadden laatst een instelling menu die je aan of uit kon zetten en dan wordt er zo'n standaard checkbox van Apple ingeduwd. Dan is het feitelijk een hele goeie, duidelijke checkbox maar aan de andere kant begin je toch te twijfelen want hoe weet de gebruiker nou of die aan of uit staat? Dat zie je eigenlijk pas op dat moment als je hem aanraakt. Dat kan heel situationeel zijn, dat is het ook altijd. Dan ga je toch twijfelen, is dit nou handig? Wat je ook steeds vaker ziet is dat je veel meer rekening moet houden met mensen met beperkingen, met visuele beperkingen of noem maar op. Dus je moet ook knoppen uitleggen eigenlijk, steeds meer. Dat is waar ik ook veel over lees, of wat ik interessant vind.

Martin:
Je noemde het net inderdaad ook al een beetje, zijn er beperkingen van het platform waar je weleens tegenaan loopt?

Hans:
Ja, wat ik net noemde. Die checkbox bijvoorbeeld. Wat bij ons veel gebeurd is natuurlijk dat we veel werken met kaarten. We hebben een hele tijd gedacht om een soort cross-platform achtige map in te zetten. Die wordt ook nog wel hier en daar gebruikt. Maar dat kost enorm veel geld dus daar loop je op leeg. Dus hebben we onder andere voor Verkeerplaza gekozen om de smartphone-eigen map te gebruiken. Dat doe je dus enerzijds vanwege de kosten, anderzijds is zo'n kaart ook heel efficiënt en snel want dat staat er al allemaal op. Maar dan heb je wel te maken met beperkingen. Sommige kleuren passen gewoon net niet of er zit te veel detail in wat je eigenlijk helemaal niet nodig hebt voor bijvoorbeeld verkeer. Die beperkingen neem je dan voor lief, dat is heel vaak een afweging. Waar je ook altijd rekening mee moet houden is met updates die verschillende platforms zelfs hebben. Zoals bij iOS, een update naar versie 10.1 ofzo. En dan bleek dus dat als je ging zoomen, met name op die kaart, dat lijnen die we zelf erop hadden getekend dat die gesmudged werden of zoiets. Die schaalden niet mee in de eerste instantie. Dat vind ik ook een beperking. Je bent ook afhankelijk daardoor van de update frequentie van de leveranciers van Android of iOS omdat je gebruik maakt van hun tools.

Martin:
Goed, verder is er nog een laatste vraag: Welke effecten denk je dat het gebruik van een cross platform framework zou hebben op jouw designs voor zo een applicatie?

Hans:
Goeie vraag.

Martin:
Zou je het daadwerkelijk anders aanpakken denk je?

Hans:
Nou ja, dan wil je toch iets meer weten over of het specifiek beperkingen of nieuwe kansen met zich mee brengt. Dat is nog een beetje mistig. Maar ik denk, als je me nu vraagt zo, dat ik wel voordelen vooral zie. Ik zie meer voordelen dan beren op de weg, laat ik het zo zeggen. Een voorbeeld: ik heb vorige week de buienalert app, die klaar staat om gelanceerd te worden of inmiddels is gelanceerd, om die nog even te bekijken op een aantal punten. En dan zie je dus dat twee ontwikkelaars door specifieke platform beperkingen of eisen bepaalde onderdelen toch net iets anders in elkaar zetten dan je van tevoren had bedacht. Gewoon omdat het niet anders kan. En op zich is dat prima, maar je hebt dingen van tevoren bedacht met een bepaalde reden. En als je dus heel specifiek moet aanpassen voor een van die platformen, dan moet je daar van afwijken en ik hoop dat je dat kan voorkomen.

Martin:
Dat je toch de twee designs echt gestroomlijnd met elkaar houdt.

Hans:
Ja, je maakt bewust een keuze om een button een kleur of een grote of wat dan ook te geven en als je dan beperkt wordt door een platform, dan is dat op zich niet positief. Misschien kun je dus wat meer eenduidig werken. Je krijgt uiteindelijk maar een lijn dan. In die zin zie ik dat wel als voordeel ja.

Martin:
En dan is het nog de vraag natuurlijk, zeker als je native (cross platform) frameworks hebt, die gebruiken vaak native componenten. Als je dan een knop neemt in Android en iOS, die zijn visueel daadwerkelijk significant anders. Dat zal waarschijnlijk nog wel kans hebben om te blijven. je kan waarschijnlijk wel alsnog die knoppen ook wel handmatig anders maken.

Hans:
Een eigen stijlsheet bij wijze van spreken?

Martin:
Dus op die manier kan er wel een oplossing geboden worden voor die situatie ja. Maar dat is ook weer een afweging tussen: wil je dit component echt gestroomlijnd hebben tussen de 2, dat het er hetzelfde uitziet of wil je dat het duidelijk herkenbaar is als Android en iOS.

Hans:
Nou ja, dat is ook wel een uitdaging dat mensen die echt Android fan zijn, verwachten ook wel bepaalde interacties zeg maar. Bijvoorbeeld swipes en zo, die zijn heel anders. Dus dat kan ook een nieuwe uitdaging zijn om te kijken: hoe gaan we dat oplossen dan, als je alles op een lijn zet. Maar jouw vraag was het effect. Nou ja, ik zie dat niet als iets negatiefs hoor eerlijk gezegd. En ik vind het ook wel goed dat er twee of meer platformen zijn want dat geeft ook vaak wel weer nieuwe inzichten.

Martin:
Als je Windows er bijvoorbeeld bij gaat betrekken dan wordt het al helemaal leuk, die heeft ook weer zijn eigen stijl met Metro interfaces enzovoorts.

Hans:
Ja, volgens mij gaat het ons heel veel brengen gewoon. Daar ben ik wel benieuwd naar.

Martin:
Dat was alles wat ik zo'n beetje heb. Hartstikke bedankt.

Hans:
Ja, geen dank.

8.6. Bijlage 6: Interview Sjoerd

Martin:
Welke apps binnen Infoplaza heb je eigenlijk allemaal allemaal aan gewerkt als ontwikkelaar?

Sjoerd:
Dat is voornamelijk de Weerplaza iPhone app en dan verschillende iteraties, vanaf de eerste versie. We zitten nu op versie 6.
Ik heb een iPad app gemaakt die niet langer in de store staat, daarmee zijn we gaan pionieren eigenlijk. Toen iOS 3 er net was, dat was de versie na de versie waarbij appstore kwam. Toen zijn we begonnen met … zodra de iPad uitkwam hebben we ? geimporteerd en zijn we begonnen met ontwikkelen en daar zijn we gaan kijken wat er mogelijk was in het platform. Die is niet langer in de store te vinden, maar die heb ik aan gewerkt. Ik heb de Verkeerplaza app gemaakt en ik heb de OVplaza app gemaakt. Dus eigenlijk onze hele scala aan apps 3-4 jaar geleden. Dat kwam bij mij vandaan. Verkeerplaza is onderhand door Paul opgepakt, die heeft iets nieuws gemaakt daarvoor. Weerplaza die staat er nog zoals het was. Daar bovenop zijn de apps Meteoalert, Buienalert en Onweerradar gekomen. Daar heb ik zijdelings mee te maken gehad en heb ik een aantal componenten voor geschreven, maar niet langer de hele app helemaal zelf. Dus in principe bij alle apps ben ik wel betrokken in meer of mindere maten en sommige heb ik gewoon letterlijk elke regel code in geschreven.

Martin:
Je noemde vooral iOS, maar heb je ook aan android versies gewerkt van deze apps?

Sjoerd:
Nee ik ben geen productie apps geweest. Ik heb er wel eens aan gewerkt maar, nee. het is echt alleen puur iOS.

Martin:
Betreft iOS, hoe lang doe je gemiddeld over het realiseren van een nieuwe functionaliteit, bijvoorbeeld een pagina met nieuws artikelen erin die bekeken kunnen worden.

Sjoerd:
Dat is een goede. Om daar antwoord op te geven, we hebben natuurlijk eigenlijk twee dingen die we daar voor nodig hebben, voor zo'n pagina, als we het native bouwen. Wat in onze huidige app trouwens niet het geval is. De nieuwspagina daar is niet native, dat is een hybride pagina.
Daar heb ik natuurlijk die API voor nodig die de data oplevert. Ik ben degene die ook de API's maakt, dus die heb ik eerst gemaakt. Zeg, het implementeren van een overzichtspagina waar je doorheen kan scrollen, waar je 15 tot 20 artikelen hebt staan met een plaatje, tekstomschrijving, auteur enzovoorts. Dan denk ik dat ik dat in een dag of 2 a 3 realiseer, inclusief het testen daarvan. Op zich is dat een redelijke doorlooptijd en Objective-C vind ik in ieder geval zelf een taal die veel mogelijkheid geeft om relatief snel te ontwikkelen. Het is geen C# want daar zou je het sneller in maken maar ik heb wel het idee dat het taal is waar je redelijk snel mee kan ontwikkelen.

Martin:
Die apps zijn in Android ondertussen overgenomen, welke mogelijke nadelen heb je ervaren wel van dat apart in Android en iOS ontwikkelt moest worden.

Sjoerd:
Ja, wat we vooral merken is een huisstijl. Ik weet niet of je hans al hebt gesproken?

Martin:
Ja, ik heb hem wel gesproken maar dat was nog vooral over de user experience van OVplaza.

Sjoerd:
Ja precies maar je hebt nog niet geïnterviewd om het zo te zeggen.

Martin:
Nee.

Sjoerd:
Ja wat ik vooral merk is dat we een huisstijl ontwikkelen en zeker in het begin was onze huisstijl een combinatie tussen wat we op de website hebben en componenten die op dat moment in iOS waren. Je moet je voorstellen, we waren in het begin van mobiele apps want alles wat ervoor kwam was van die brakke Java spullen allemaal en Java applets enzo en toen kwam iOS opeens met echte apps. Je had daar hele specifieke componenten en de visie van apple dat eigenlijk elke app er hetzelfde uit hoort te zien zodat de gebruiker weet hoe de functionaliteit werkt. Dus dat is best wel een uitdaging in de eerste instantie te zorgen dat onze eigen kleurstijl onze fonts enzovoort daar op beschikbaar werden en dat dat tot een stijl werd gebakken waar we tevreden mee waren en waarbij we aansluiten bij de specificaties van het platform, dus wat de gebruiker verwacht van zo'n app. We merkten zeker dat op het moment dat dat er stond, zijn we pas naar de android app gegaan. en dat was een hele lastige slag om te maken want we zaten heel erg in de mindset dat dit onze mobiele huisstijl was terwijl het eigenlijk een fusie was tussen onze huisstijl en iOS componenten. Dus we hebben geprobeert een hele iOS stijl neer te zetten op Android. En dat heeft even geduurd voordat we daarachter waren dat dat niet de juiste manier is van werken. Dus ja, ik zat gelukkig aan de goeie kant om dat zo te zeggen. Ik kon schakelen met de designers we gaan dit en dit doen. Dus ik heb het niet echt als een nadeel ervaren maar ik was wel zeker weten bij het project betrokken waarbij we op een of andere manier die iOS stijl naar Android stijl moesten zien te krijgen. En daar heb ik wel ervaren dat daar heel erg veel tijd in heeft gezeten. En dat is namelijk omdat we gewoon iOS first shop waren. Dat is iets wat we dan hopelijk niet meer zijn.

Martin:
Om verder ook beide kanten van de zaak te belichten, welke voordelen heb je wel expliciet gemerkt van het bouwen in android en iOS.

Sjoerd:
Dat is lastig om een baseline neer te zetten. Om die app op Android te hebben moet je hem bouwen voor Android. Dus dat maakt het lastig want het voordeel was in dat opzicht dat we een app op android hebben.

Martin:
En als je bijvoorbeeld, ik weet niet in hoeverre je een beetje bekend bent met cross platform mogelijkheden en welke kennis die je daarover hebt, dan native ontwikkelen vergelijkt met wat je weet van cross-platform ontwikkelen welke voordelen heeft het daartegenover.

Sjoerd:
Ja ik kan wel wat theoretische antwoorden daarop geven inderdaad omdat we wat geprototyped hebben en wat spullen hebben gedaan en we hebben natuurlijk ook allemaal andere apps gezien. Cordova apps zie je veel inderdaad. Wat mijn mening daarvan is, of mijn beeld of gedachte bij een app zoals cordova of gebaseerd op cordova eigenlijk. Ik heb het idee dat dat voor de standaard CRUD-dingetjes dus create, update, delete prima werkt, maar als bijvoorbeeld naar onze buienalert app kijkt waarbij we eigenlijk iets doen wat we niet heel veel zien, dat is namelijk op een google of iOS kaart animatielayers afspelen. Dat is iets wat ontzettend zwaar is om native te doen en waarbij je echt al moet optimaliseren, zorgen dat je code op de GPU wordt uitgevoerd bijvoorbeeld, anders is je telefoon binnen no time leeg. Dat zijn echt dingen waarvan we zeker 2 jaar geleden toen we begonnen tegen aan liepen, de limieten van de toestellen. Ik kan me niet voorstellen dat met een platform die hybride werkt, dus niet naar native compileert, mogelijk is om ook maar de helft van de performance te halen. Dus in dat opzicht denk ik dat het wel heel erg fijn is dat we native mogelijkheden hebben gehad. Maar tegelijkertijd is dat een puur theoretisch exercitie, voor hetzelfde geld kan je met cordova dat heel makkelijk doen. Ik weet dat niet, ik het me alleen niet voorstellen want we liepen echt tegen de beperkingen van de hardware aan. We hebben best low-level dingen moeten doen om te zorgen dat het werkte. Daar zie ik wel een uitdaging: hoe gaan we dat voor elkaar krijgen bij een hybride oplossing? Ik kan me ook voorstellen dat een oplossing die dus naar native code compileert daar voordeel zou hebben.

Martin:
Welke voordelen verwacht je te kunnen halen uit het gebruik van een cross-platform framework en dan is het misschien ook wel handig om ook wel deze vraag apart te beantwoorden over hybride inderdaad en bijvoorbeeld cross-platform frameworks die vertalen naar native zoals Xamarin.
Met Cordova heb je het hybride eigenlijk al wel genoemd, dat het juist niet echt een voordeel was volgens mij.

Sjoerd:
Nee, ik denk als we het vanuit design oogpunt bekijken of van wat ik eerder aangaf dat we een iOS-first shop waren. Op het moment dat we een hybrid-first shop worden kan ik het me wel heel goed voorstellen dat dat het design te goede komt. Dat we betere mogelijkheden hebben om onze eigen huisstijl echt te definiëren die los staan van de componenten specifiek aan iOS maar dat je vervolgens ook gedwongen wordt om te kijken: ok, we moeten wel zorgen dat het op allebei de platformen draait. En volgens mij, bij hybride oplossingen als Cordova, kan je bij wijze van spreken een derde huisstijl introduceren die niks met iOS te maken heeft en volgens mij ook niet zoveel met Android. Zo'n Xamarin, dat compileert naar native code, gebruikt ook native componenten. Volgens mij pak je daar al een deel een voordeel omdat je wel je kleurencombinaties mee kan pakken, maar dat je omdat je ook moet focussen op die twee stukken native dat je daardoor ook gewoon platform specifieke aanpassingen kan maken. Dus dat je navigatie kan aanpassen op basis van hoe het werkt op het ene en op het andere platform. Ik

verwacht dat we daar wel echt grote voordelen gaan behalen dat we een consistentere stijl krijgen dat je binnen de platformen het duidelijk herkent van: "ok, dit zijn weerplaza apps". Maar dat je daar bovenop ook duidelijk ziet van ok, maar het is wel echt een android app, en dit is echt een iOS app en het is geen verbasterde versie die het allebij net niet is ofzo. Dat is de gedachte of hoop eigenlijk, hoop is het op dit moment, die we gaan tegenkomen bij het cross-platform ontwikkelen. Zo zie ik dat voor me en dat zou heel erg veel schelen. Daarbovenop hebben we natuurlijk het technische aspect waarvan het delen van de business logic een hele goeie zou kunnen zijn. We hebben in het verleden gezien dat de Weerplaza apps op Android en iOS toch informatie anders wisten te presenteren. Het is iets heel sufs, maar wat we wel zagen is dat afrondingsregels anders waren. Het zijn verschillende ontwikkelaars dus dat kan, maar ik ga ervan uit dat op het moment dat die business logic gewoon gedeelde code is, dat het niet voorkomt dat de ene 4,7 afrondt naar 5 en dat die ander gewoon 4 ervan maakt. Dat is natuurlijk heel lastig om aan je klanten uit te leggen van hee op mijn android telefoon is het een graad warmer. Daarbovenop natuurlijk alle bugs die je dubbel of driedubbel of vierdubbel aan het fixen bent. Dus daar verwacht ik ook heel veel voordeel van. Het hele aspect eigenlijk, van voor naar achter, verwacht ik heel veel van.

Martin:
Duidelijk. Hartstikke bedankt.

Sjoerd:
Ja, natuurlijk.

8.7. Bijlage 7: Interview Bart

1. Welke apps van Infoplaza heb je aan gewerkt?

Ik heb gewerkt aan de apps van Breng, Connexxion, GVB, Hermes, RET Real Time en Veolia.

2. Heb je vooral aan iOS of aan Android gewerkt?

Alleen aan iOS.

3. Welke grote verschillen ervaarde je tussen deze twee platformen?

-

4. Was het lastig om voor het andere platform te ontwikkelen?

-

5. Hoe lang doe je gemiddeld over het realiseren van een nieuwe functionaliteit, bijvoorbeeld een scherm met een lijst van nieuws artikelen?

Het is erg lastig om een realistische schatting te geven wat betreft het ontwikkelen van een 'nieuwe functionaliteit', aangezien het afhankelijk is van een aantal factoren, zoals:

- Om welke functionaliteit gaat het precies? Hoe omvangrijk is het?
- Is een design beschikbaar?
- Heeft de functionaliteit te maken met een API?
- Zijn alle requirements bekend, zoals (offline) opslag van objecten?

Een scherm met een lijst van nieuwsartikelen klinkt vrij eenvoudig. Mits een design beschikbaar is en het een eenvoudige lijst betreft, met bijvoorbeeld een titel, beschrijving en afbeelding per nieuwsartikel, zou de lijst al in enkele uren gebouwd kunnen worden. Toch kunnen verschillende factoren / requirements de ontwikkeltijd aanzienlijk beïnvloeden, zoals:

- Komen de nieuwsartikelen van een API? Zo ja, is de benodigde API call daarvoor beschikbaar? Zo ja, in welke structuur gebeurt dit?
- Is de app al geschikt voor communicatie met een API?
- Dienen de nieuwsartikelen opgeslagen te worden?
- Beschikt de lijst over 'paging', oftewel, bij het naar onderen scrollen worden automatisch meer nieuwsartikelen geladen?

Afhankelijk van de antwoorden op bovenstaande vragen, kan 'enkele uren' al snel veranderen in 1 of zelfs 2 dagen. In het algemeen geldt: hoe meer duidelijkheid beschikbaar is wat betreft requirements, des te sneller kan de functionaliteit ontwikkeld worden.

Inmiddels heb ik veel ervaring opgedaan met het creëren van lijsten, inclusief lokale opslag, API-communicatie, paging-functionaliteiten en het weergeven van online afbeeldingen, dus zou 2 tot 4 uur een realistische schatting kunnen zijn.

6. Hoe lang duurt dit daarna op het andere platform?

-

7. Welke mogelijke nadelen ervaar je met het apart ontwikkelen in Android en iOS?

Ik ervaar geen nadelen met het apart ontwikkelen voor beide platformen. Uiteraard dienen functionaliteiten in zekere zin dubbel gemaakt te worden, maar beide platformen doen dit dusdanig op hun eigen, geoptimaliseerde manier, dat ik dit niet als nadeel kan zien. De platformen hebben eigen, specifieke oplossingen voor bijvoorbeeld API-communicatie, lokale opslag, het bouwen van de user interface, het navigeren naar andere schermen en push notifications. Door aparte ontwikkeling te handhaven, kunnen beide platformen optimaal gebruik maken van hun native eigenschappen.

Business logica, zoals het data(base)model, zal op iOS altijd grotendeels dan wel geheel overeenkomen met business logica op Android. Dit betreft echter vaak een relatief klein deel van de app. Bij grotere, complexere business logica komen bovendien weer platform-eigen functies naar voren.

8. Welke voordelen ervaar je van het apart ontwikkelen in Android en iOS?

Zoals hierboven al genoemd, heeft het apart ontwikkelen als voordeel dat de platform-eigen oftewel native functionaliteiten en oplossingen gebruikt kunnen worden. Deze zijn toegespitst op het betreffende platform en halen het maximale uit het OS en het device.

Ten slotte kan de user interface altijd conform de platformrichtlijnen geïmplementeerd worden, zodat zowel een Android-app als een iOS-app altijd native aanvoelt. Het zijn immers native apps.

Beide platformen brengen hun eigen ontwikkelprocessen met zich mee, wat goed past bij het apart ontwikkelen voor de platformen.

9. Welke voordelen verwacht je te kunnen halen uit het gebruik van een cross-platform framework.

Het enige voordeel, in mijn ogen, van het gebruik van een cross-platform-framework is het kunnen delen van business logica. Data(base)modellen hoeven op deze manier niet dubbel gecreëerd en gehandhaafd te worden. Een wijziging in de business logica heeft direct invloed op beide platformen.

Aan de andere kant heb ik hier ook direct mijn bedenkingen bij. iOS heeft een andere manier van data-afhandeling dan Android. Ik vraag me dan ook af hoe het delen van business logica de data-afhandeling op beide platform beïnvloedt. Vanwege mijn tekort aan ervaring op dit gebied kan ik hier geen antwoord op geven.
Ik weet dat het delen van business logica mogelijk is op cross-platform-frameworks, maar toch ben ik erg voorzichtig in het bestempelen als ‘voordelig’.

10. Welke nadelen zie je in het ontwikkelen met een cross-platform framework?
In mijn ogen komen de volgende zaken al snel naar voren:

- Het hanteren van een andere programmeertaal dan de platform-specifieke programmeertaal. Men programmeert voor beide platformen, maar heeft (te) weinig kennis van de native programmeertalen.
- De invloed van de vertaalslag van cross-platform-code naar iOS- en Android-apps. Is dit nadelig voor de performance van de apps? Mijn ervaring tot nu toe met native apps vs. cross-platform-apps heeft laten zien dat native apps beter presteren.
- De ondersteuning voor native UI en UX. Indien de user interface niet op een native manier gebouwd kan worden, is de kans groot dat dit zichtbaar terugkomt in de app. De app voelt (misschien) niet native aan.
- Het creëren van een 'native feel' kan erg veel tijd kosten.
- Indien een iOS-app niet native genoeg aanvoelt, is de kans aanwezig dat de App Store de app afkeurt.
- De ondersteuning voor nieuwe platformfunctionaliteiten. Nieuw geïntroduceerde classes, protocollen, views en andere functionaliteiten zijn (mogelijk) niet direct beschikbaar in cross-platform-frameworks, waardoor men altijd achter zal lopen.
- Op internet is veel meer informatie te vinden over platform-specifieke onderwerpen en problemen dan over cross-platform-gerelateerde onderwerpen en problemen. Het oplossen van een probleem in een native app zal dan ook minder tijd in beslag nemen in vergelijking met een probleem in een cross-platform-app.
- Met name Xcode zorgt ervoor dat het ontwikkelen snel verloopt. Het uitvoeren van de gemaakte app op een iOS-device of iOS-simulator gebeurt in de regel erg snel. Op cross-platform-frameworks verloopt dit aanzienlijk langzamer, wat de belevenis van ontwikkelaars negatief beïnvloedt.
- Het ontwikkelen van een zeer complexe app kan al snel veel problemen opleveren of zelfs onmogelijk worden voor cross-platform-platforms.

11.3. Technisch ontwerprapport

Vanaf de volgende pagina is het Technisch Ontwerp toegevoegd

Deze bijlage is weggelaten in verband met de vertrouwelijke inhoud hiervan.

11.4. Functioneel ontwerprapport

Vanaf de volgende pagina is het functioneel ontwerprapport toegevoegd.

Deze bijlage is weggelaten in verband met de vertrouwelijke inhoud hiervan.

11.5. Scrum documenten

De scrum documenten zijn terug te vinden als losse bijlages in het archiefbestand “scrum documenten.zip”.

Deze bijlage is weggelaten in verband met de vertrouwelijke inhoud hiervan.

11.5. Zelfreflectie

De zelfreflectie is terug te vinden als losse bijlage in het bestand “zelfreflectie.pdf”.

Deze bijlage is weggelaten in verband met de vertrouwelijke inhoud hiervan.