KU LEUVEN

FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN

CAMPUS GEEL

API ontwikkeling in cross-platform en platform-specifieke frameworks

Een vergelijkende studie

Lennert VAN LOOVEREN

Promotor: Dr. Ing. Peter Karsmakers

Co-promotoren: Koen Swings, Bram Schrijvers

Masterproef ingediend tot het behalen van de graad van master of Science in de industriële wetenschappen: *elektronica-ICT, afstudeerrichting ICT*

Academiejaar 2017-2018

Voorwoord

In de master industriële wetenschappen is de masterproef een essentieel onderdeel van het programma. Hiervoor heb ik adequate hulp gekregen van bepaalde mensen die ik extra moet bedanken.

Ik heb deze masterproef kunnen verwezenlijken dankzij mijn ouders, via hun morele en financiële steun gedurende mijne hele studieperiode. Alsook door mijn promotoren die mij gedurende de hele weg begeleidt hebben en de collega's bij Zappware die voor een leuke sfeer zorgden en mij uit de nood hielpen indien nodig. Met speciale dank aan Bram Schrijvers die dagelijks beschikbaar was om mij te helpen.

Ten slotte wil ik de jury bedanken voor de tijd te nemen om deze masterproef te lezen en te beoordelen.

Samenvatting

De (korte) samenvatting, toegankelijk voor een breed publiek, wordt in het Nederlands geschreven en bevat **maximum 3500 tekens**. Deze samenvatting moet ook verplicht opgeladen worden in KU Loket.

Abstract

Het extended abstract of de wetenschappelijke samenvatting wordt in het Engels geschreven en bevat **500 tot 1.500 woorden**. Dit abstract moet **niet** in KU Loket opgeladen worden (vanwege de beperkte beschikbare ruimte daar).

Keywords: Voeg een vijftal keywords in.

INHOUD

LIJST MET FIGUREN

LIJST MET AFKORTINGEN

API	Application programming interface
(G)UI	(Graphical) User interface
SDK	Software development kit
UWP	Universal Windows Platform
GLUT	OpenGL Utility Toolkit
GLU	OpenGL Utility Library
GLX	OpenGL Extension to the X Window System
AGL	Apple Graphics Library
GNU	GNU Compiler Collection
ICC	Intel C++ Compiler
MSVC	Microsoft Visual C++
DOM	Document Object Model
IDE	Integrated development environment
APK	Android Package File

1 INLEIDING

Zappware

Deze masterproef werd uitgevoerd bij Zappware gestationeerd in Hasselt. Een globaal bedrijf dat zich zowel bezig houdt met het ontwerpen van video UI design als client-software development [1]. Deze masterproef kadert binnen de client-software kant. Hierbij werden er oplossingen gezocht om het proces van individuele API''s niet te moeten gebruiken bij het programmeren van mobiele applicaties. Deze oplossing zou het mogelijk maken om een snel prototype van hun typische UI implementaties te bouwen om deze te kunnen demonstreren, of bij zeer goede resultaten de originele Android- en iOS-ontwikkeling zelfs te vervangen. Zappware voorzag voorbeelden van mogelijke applicaties om te maken en de expertise van hun personeel.

Onderzoeksvraag

Deze masterproef begon met het onderzoek van Qt, een crossplatform framework dat één code zo omzet dat ze zowel op Android als iOS uitgevoerd kan worden. Bij deze studie werd er bekeken of dit platform een effectieve vorm van programmeren opleverde en wat de mogelijkheden waren van Qt. Vroeg in deze studie werd er geconcludeerd dat Qt niet hetgeen opleverde wat Zappware verlangde.

Door deze conclusie werden er naar alternatieven gezocht op Qt, dit werd uiteindelijk de basis van deze masterproef zelf.

Het doel is om een crossplatform framework te vinden dat aan de eisen van performantie, gebruiksvriendelijkheid en ontwikkelingstijd voldoet. De gecreëerde applicaties worden dan met elkaar en met applicaties die wel native gebouwd zijn vergeleken. Uiteindelijk wordt bepaald welke crossplatforms het meest geschikt bevonden zijn voor gebruik of eventueel als een platform om prototypes van hun mobiele applicaties op te bouwen.

2 CROSS-PLATFORM FRAMEWORKS

2.1 Qt

2.1.1 Oorsprong

Qt is ontworpen nadat de co-founders van Trolltech in 1990 samenwerkten aan een database applicatie voor ultrasound-afbeeldingen, geschreven in C++, dat zowel op MAC OS, UNIX en Windows moest draaien. Hiervoor bedachten ze een object-georiënteerd display systeem nodig te hebben. Dat resulteerde een basis voor de object-georiënteerde cross-platform GUI framework dat later gebouwd werd onder de naam Qt, Q-toolkit.

Na 5 jaar schrijven aan de nodige C++ klasses werd in 1995 Qt 0.90 uitgebracht, bruikbaar voor zowel Windows als Unix development en gebruikte voor beide platforms dezelfde API. In 2008 werd TrollTech overgenomen door Nokia en streefden zij om Qt het meest gebruikte development platform te maken voor hun apparaten.

Vanaf 2014 werd Qt beheerd door “The QT Company”. Een dochterbedrijf van Digia, die de rechten van Qt overgekocht heeft van Nokia in 2012. Digia verzorgde de revolutionaire Qt 5.0 waarbij met behulp van JavaScript en QML de performantie en de eenvoud van UI te ontwerpen zwaar werd verbeterd. Alsook werd Qt vanaf toen open-source governance, waardoor ook ontwikkelaars buiten Digia verbeteringen konden voorstellen. [2]

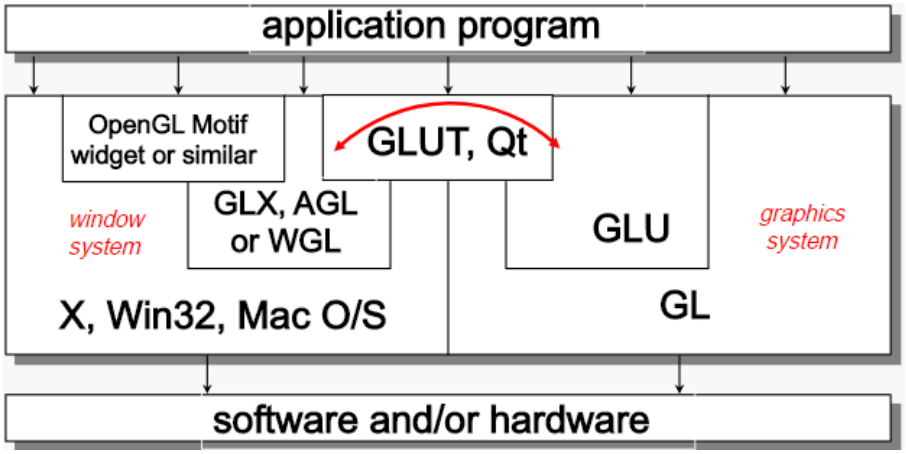
In 2016 werd Digia en “The Qt Company” gesplitst in 2 onafhankelijke bedrijven. [3]

2.1.2 Basis

Qt is een cross-platform applicatie framework dat in C++ geschreven is en Android, iOS en WindowsPhone, … ondersteunt. Het doel van Qt was om zonder codewijzigingen op elk platform zonder performance-verlies te draaien en native te lijken.
Qt ondersteunt elke standaard C++ compiler, zoals GCC, ICC, MSVC en Clang. [4]
De applicatie werd geschreven in een combinatie van C++ en QML, een declaratieve scripting-taal die Javascript gebruikt voor de logica.

Qt dankt zijn cross-platform eigenschap aan het feit dat bij de C preprocessor (*cpp*) condities sommige code chuncks worden enabled/disabled afhankelijk van het platform. Het build proces gaat configuratie scripts runnen om preprocessor flags op te merken, afzetten en aan te passen. [5]

2.1.3 Graphics architectuur



Figuur 2-1: de grafische architectuur van Qt [30]

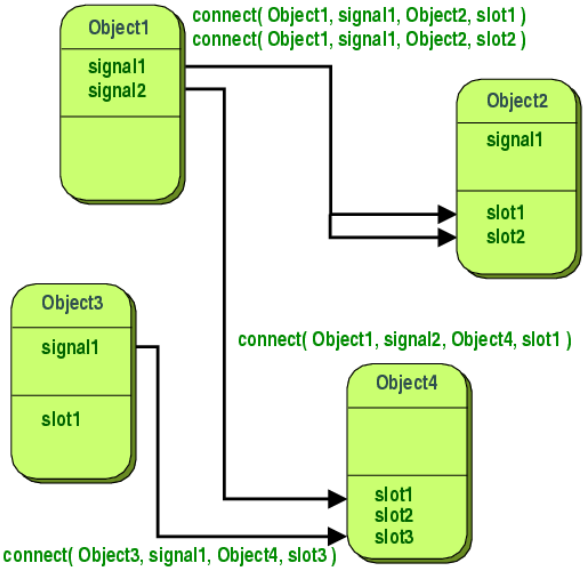
Het grafische systeem beheert de display hardware. Vooral OpenGL-GL, een library die 2D en 3D vectoren rendert, wordt als API gebruikt voor het grafische systeem. [6] Dit terwijl het Windowing systeem de windows genereren, de events, window aanpassingen, … controleert. Het Windowing systeem heeft een eigen API voor programmeren, dat meestal in het OS is geïntegreerd zoals Microsoft Windows en MAC OS.

GLUT is een interface dat de communicatie tussen het window system en het graphic system verzorgt, deze gebruikt GLU en GL voor graphics en controleert operating en windowing systemen door voornamelijk GLX, AGL, … te gebruiken. [7] Dit zijn OpenGL extensies voor specifieke Windowing systemen.

Qt zal op hetzelfde niveau als GLUT zowel het windowing als het graphics systeem kunnen raadplegen. Hierdoor kan het dus zowel Windowing functions als graphic functions oproepen.

2.1.4 Main feature: Signals and slots

Qt behandelt events, zoals *quit(), onkeydown(),…* , niet rechtstreeks. Zogenaamde callback functions hebben een alternatief, namelijk signals and slots. Hierbij wordt een signaal verzonden wanneer een bepaald evenement gebeurt of een verandering van status plaatsvindt. Een slot is een functie die opgeroepen wordt als er een bepaald signaal verzonden wordt. Deze signal-slot relatie zit al in vele Qt-widgets (interface objecten) ingebouwd, maar zelf slots bepalen voor bepaalde signalen is ook veel gebruikt. Dit signal-slot gebruik is zeer veelzijdig, meerdere signalen kunnen aan meerdere slots gekoppeld worden en ook signalen kunnen aaneengeschakeld worden. [8]



Figuur 2-2: Voorbeeld Signals and Slots [8]

```
Int main (int argc, char *argv[])
{ QApplication app(argc, argv); // applicatie object creëren
 QPushButton *button = new QPushButton("Quit");
// Creëren gelabelde knop
 QObject::connect (button, SIGNAL(clicked()), &app, SLOT(quit());
 // als signal "clicked" op object "button" wordt verzonden
, roep functie "quit" op in object "app"
 button->show();
 return app.exec();
}
```

Een voorbeeld voor dit signal-slot gebeuren: een eenvoudige *quit()* functie:

2.1.5 Qt Creator

Qt Creator is het C++,, JavaScript en QML IDE dat deel is van het SDK voor de Qt GUI applicatie development framework. Deze houdt een visuele debugger en een geïntegreerde GUI layout in. In deze IDE zit een eigen editor in die de typische kenmerken bevat zoals syntax highlighting en het automatisch aanvullen van code die zou ontbreken.

In verband met de installatie van Qt Creator is het mogelijk om online een gratis open source versie downloaden die Qt zelf aanbied. Voor commerciële doeleinden heeft Qt een uitgebreider pakket tegen betaling die meer tools, features en support zal bieden.

De installatie zelf is rechtuit: installeren en opstarten. Om de omgeving te laten werken moeten er enkele omgevingsvariabelen aangepast worden aan Windows/MAC en development kits geïnstalleerd worden in Qt zelf om de code te testen op het geprefereerde platform. Hierbij zijn

de mogelijkheden eindeloos: van verschillende versies Android tot desktop applicatie op Windows, analoog op MAC.

2.1.6 Portfolio

Noemenswaardige applicaties die Qt of QML gebruiken: [9]

- Google Earth
- Adobe Photoshop Album
- Adobe Photoshop Elements
- Spotify for Linux
- VLC media player
- Teamviewer

2.2 React Native

2.2.1 Oorsprong

React (soms geschreven als react.js of ReactJS) is een JavaScript library die gemaakt is om UI's te bouwen. Gebouwd door Facebook en Instagram teams in 2013. Het doel van React was om grote schaalbare webapplicaties te maken waarvan de data constant veranderd kon worden zonder de pagina te herladen. [10]

React Native startte als een hackathlon project in de zomer van 2013 [11]. Na een jaar werken aan een prototype, kreeg React Native zijn eerste job: een onafhankelijk werkende iOS app. Het doel was om een volledige React Native aangedreven app te maken die volgens user experience identiek was aan een app die in Objective-C geschreven was. Dit doel werd behaald en er werd beslist om RN cross-platform te maken. Dit begon met een RN Android team die de basic Android Runtime -een applicatie omgeving gebruikt door de Android OS- en componenten schreef. Begin 2015 werd RN publiek tentoongesteld en tegen eind 2015 werd React Native volledig open source. [10]

2.2.2 Basis

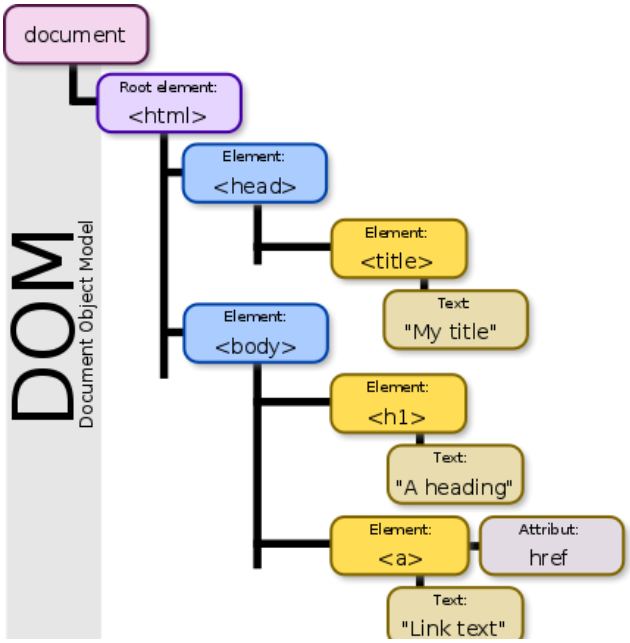
React Native is volledig gebaseerd op React. Enerzijds is het een open-source library die onderhouden wordt door individuen, kleine en grote bedrijven. Anderzijds is het een JavaScript framework om mobiele applicaties voor iOS, Android en UWP native te laten renderen en dit via een JavaScript library die al veel gebruikt werd voor webapplicaties. Dit framework zorgt er voor dat de code die je schrijft voor de 3 mobiele platforms kan gebruikt worden. Deze applicaties zijn geschreven met een mix van JavaScript en JSX, een taal die hard lijkt op XML. React Native gebruikt het platform zijn eigen standaard rendering API zodat de mobiele applicatie dezelfde look en feel zal hebben als een native designed applicatie. [12]

2.2.3 Features

2.2.3.1 Virtual DOM

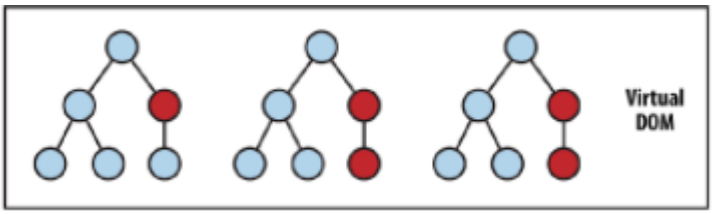
In een web applicatie is één van de meest belastende operaties het veranderen van de DOM, een applicatie programmerende interface dat een HTML, XHTML of XML bestand behandelt als een boomstructuur waarin elke node een object voorstelt als deel van het document. [13]

React onderhoudt een virtuele representatie van deze DOM, Virtual DOM dus. Samen met een 'diffing' algoritme, deze vergelijkt twee trees, kan RN het verschil t.o.v. de oorspronkelijke DOM bepalen en enkel het deel updaten dat veranderd is.
Deze eigenschap is noodzakelijk voor real time applicaties die enige complexiteit bevatten.



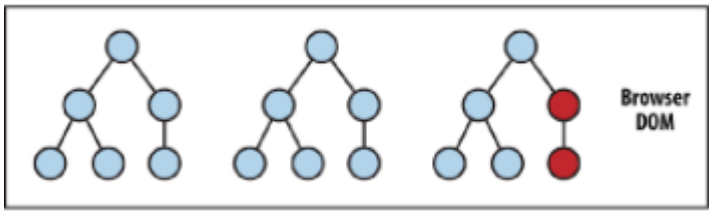
Figuur 2-3: Voorbeeld DOM

In plaats van de veranderingen die gebeuren op een pagina direct te renderen zal React de benodigde veranderingen berekenen in zijn memory en het minimaal mogelijke van de pagina opnieuw renderen. Zo zal niet de gehele pagina moeten worden herladen.



Figuur 2-4: Voorbeeld Virtual DOM [12]

De Virtual DOM heeft dus zijn performance voordelen. Maar het potentieel is veel groter dan enkel performance voordelen. Wat als React een ander doel kon renderen dan de browser z'n DOM?



Figuur 2-5: Voorbeeld Browser DOM [12]

React Native werkt zo. In plaats van de browser z'n DOM te renderen, zal React Native Objective-C APIS gebruiken om iOS componenten te renderen en analoog Java APIs om Android componenten te renderen. Met deze eigenschap onderscheidt RN zich van andere cross-platform development opties, die meestal een web-based view renderen. [12]

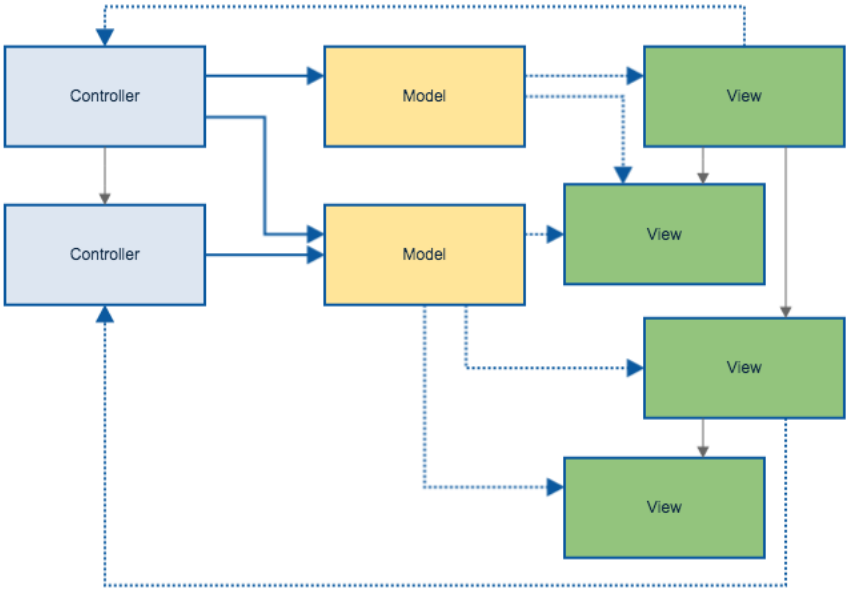
2.2.3.2 One-way data flow: MVC – Flux - Redux

React maakt gebruik van een Flux, een architectuur om data layers te creëren in JavaScript applicaties en een alternatief op het Model-View-Controller-model (MVC) dat in vele Java

applicaties wordt toegepast. MVC heeft als architectuur het nadeel dat als de applicatie complexer en groter wordt dat:

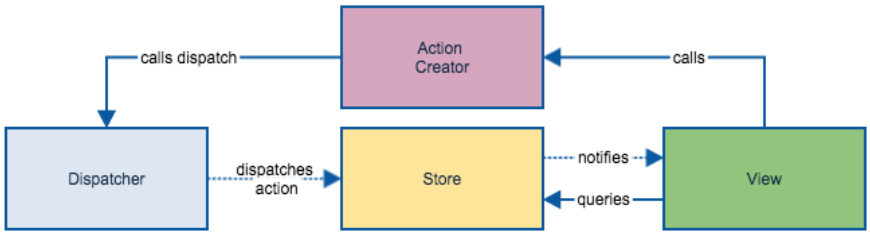
- de relaties tussen View, Models & Controllers te complex worden.
- de code zeer moeilijk te debuggen valt.
- oneindige loops te gemakkelijk geactiveerd worden.

Figuur 2-6: Voorbeeld MVC [14]

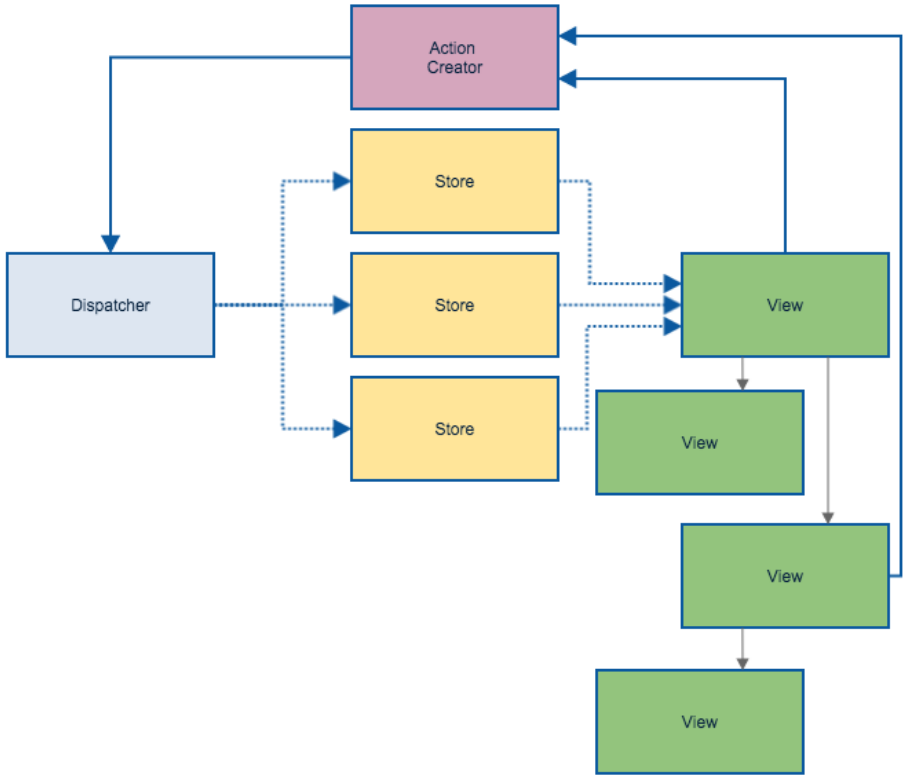


Figuur 2-7: Complexer voorbeeld MVC [14]

De Flux architectuur zorgt ervoor dat elke interactie met de View via één enkel pad naar de dispatcher gaat, deze is dan een centraal punt voor alle acties naar de alle Stores te sturen. De Stores zelf bezitten dan de logica om te bepalen of deze actie dan iets verandert in de eigen data. Elke Store is verantwoordelijk voor een domein van de applicatie. Dit update enkel zichzelf als reactie op de acties die worden doorgezonden vanuit de Dispatcher. Het grootste voordeel van Flux is het feit dat de data maar in één richting gaat, hiermee vermijd men complexiteit, infinite loops, debug problemen, … Eveneens is het veel makkelijker om de flow van data uit te leggen aan iemand die er nog geen kennis van had. [14]



Figuur 2-8: Voorbeeld Flux [14]



Figuur 2-9: Complexer voorbeeld Flux [14]

Tegenwoordig wordt voor de React library veelal de Redux architectuur gehanteerd, dit is een uitbreiding op Flux. Redux heeft als core dezelfde architectuur maar lost nog enkele complexiteit issues op dat Flux bevatte:

- Het gebruikt geen Dispatcher en vertrouwd op pure functies i.p.v. event emitters: pure functies zijn gemakkelijk te maken en hebben geen aparte entiteit nodig die ze managed. [15]
- De callback registration wordt vervangen met een functionele compositie waardoor *reducers* kunnen genest worden i.p.v. een Store die 'vlak' is en helemaal niet flexibel is i.v.m. nesting. In Flux is het moeilijk om de data te onderscheiden voor verschillende requests op de server doordat de Stores onafhankelijke alleenstaande items zijn. Dit lost Redux op door enkel één store te bevatten die gemanaged wordt door *reducers* in een tree-vorm, waarbij de root gereduced wordt, dan de takken die daaruit voortkomen, enzoverder. Deze kan dan zeer makkelijk de data refreshen, verschillende states van de store op slaan, enz. Door dit laatste is Redux dan ook zeer flexibel in termen van redundantie. [16]

2.2.4 Gebruik

Voor React Native is het niet nodig om een IDE downloaden. Projecten worden opgebouwd met behulp van 'Create React Native App' geïnstalleerd op Node.js.. Dit is een platform gebaseerd op de V8 JavaScript engine van Google. Deze genereert machine-code uit

JavaScript code en verzorgt de back-end voor het mogelijk maken van front-end development zonder rekening te moeten houden met I/O events zoals web calls, netwerk communicatie, ...

Nadien wordt er via de command line *npm* gebruikt om *create-react-native-app* te installeren. Nu kan het project worden gecreëerd eveneens via de command line. Daarna kan de app gebruiksvriendelijk aangepast worden in een tekst-editor naar keuze sinds dit enkel pure Javascript code inhoudt.

Om een simulatie van die applicatie te kunnen verkrijgen zijn er enkele mogelijkheden. Eén mogelijkheid is om Expo client applicatie te installeren op een iOS of Android apparaat en deze te connecteren op hetzelfde netwerk als de computer waarop je de app bouwt. Met deze applicatie is het mogelijk om via een QR code ,die de terminal weergeeft bij het runnen van het project, de voorbeeldapp te openen. Als deze voorbeeldapplicatie werkt, kan je de applicatie in de tekst-editor veranderen en zal deze je aanpassingen real-time opnemen en weergeven.
Een andere mogelijkheid is om een emulator te runnen op de computer via Xcode (MAC/iOS) of Android Studio (PC/Android). De command line voorziet een automatische verbinding naar deze emulator als RN aan het runnen is. Het voordeel hierbij is dat de keuze van apparaten waarop de app kan worden uitgevoerd bijna oneindig is. Het is dan ook zeer gemakkelijk dat we via Android Studio bijvoorbeeld. onze app op hetzelfde apparaat simuleren om deze dan te bekijken en vergelijken op performantie, geheugen en snelheid.

2.2.5 Portfolio

Duizenden apps gebruiken React Native, van *Fortune 500* bedrijven tot nieuwe startups. De bekendste voorbeelden: [17]

- Facebook (iOS & Android)
- Instagram (iOS & Android)
- Skype (iOS & Android)
- Discord (iOS)
- Tencent QQ (Android): het grootste messaging platform van China

2.3 Xamarin

2.3.1 Oorsprong

Xamarin is een ontwikkelingsplatform ontwikkeld door het gelijknamige software bedrijf opgericht in 2011 door de ingenieurs die Mono, Mono for Android and MonoTouch hebben gecreëerd, dit zijn cross-platforme implementaties van het Common Language Infrastructure (CLI) en Microsoft .NET. In 2013 werd het development platform Xamarin 2.0 uitgebracht. Deze release had 2 voornamelijke componenten: Xamarin Studio, een open-source IDE , en een integratie met Visual Studio, Microsoft's IDE voor het .NET framework, waardoor Visual Studio gehanteerd kon worden voor het creëren van applicaties voor Android, iOS en Windows. In 2016 werd Xamarin opgekocht door Microsoft en kondigde Microsoft aan dat de

Xamarin SDK open-sourced zal worden en dat deze zal gebundeld worden als een gratis deel binnen Visual Studio's geintegreerde development omgeving. [18]

2.3.2 Basis

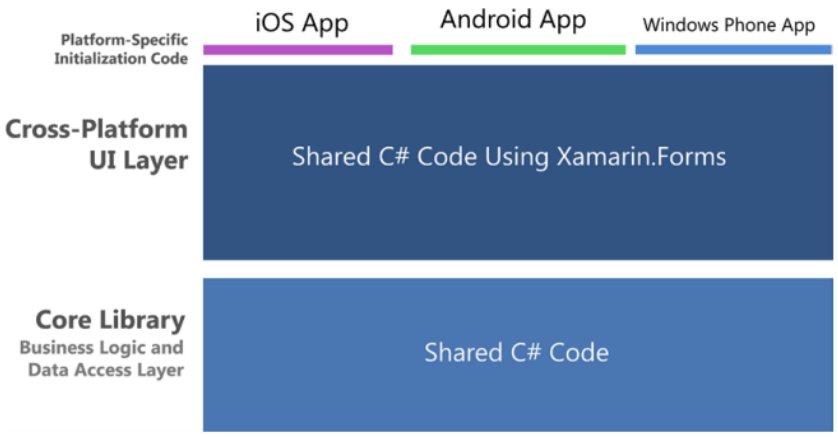
Het Xamarin platform is een .NET omgeving met iOS en Android C# verbonden libraries. Onderliggend Xamarin.Android is Mono for Android, en onder Xamarin.iOS is MonoTouch. Deze zijn de C# verbindingen tot het native Android en iOS API's voor het ontwikkelen op mobiele devices en tablets. Dit geeft het vermogen de Android/iOS UI, graphics, enz. te gebruiken terwijl we enkel C# schrijven. Xamarin.Forms is een laag boven de andere UI verbindingen, die zorgt voor een volledig cross-platform UI library. [19]

Figuur 2-10: Opbouw Xamarin [19]

2.3.3 Features

2.3.3.1 Xamarin.Forms

Xamarin.Forms is een toolkit van cross-platforme UI-klassen gebouwd boven de meer fundamentele platform-spcifieke UI klassen: Xamarin.Android en Xamarin.iOS. Xamarin.Android en Xamarin.iOS voorzien gemapte klassen aan hun respectievelijk native UI SDK's: iOS UIKit en Android SDK. [19] Xamarin.Forms verbind zich ook direct aan de native Windows Phone SDK. Dit voorziet een set van UI-componenten die allemaal in de 3 native operating systemen kunnen gerendered worden en dus allen cross-platform zijn. Deze elementen zijn gebuild met Extensible Application Markup Language (XAML) of zijn gecodeerd in C# via de Page, Layout en View klassen. Hierdoor kunnen we dus native mobiele apps voor verschillende platformen tegelijk creëren. Om een goede architectuur en herbruikbaarheid te bekomen, gebruikt een Xamarin.Forms cross-platform oplossing dikwijls gedeelde C# applicatie code die de business logic en de data access layer bevat. Dit wordt aangeduid als de Core Library. De Xamarin.Forms UI layer is ook C# en de kleine layers is een minimum aan platform-specifieke C# UI code die nodig is voor de applicatie op elke native OS te initialiseren en op te starten.



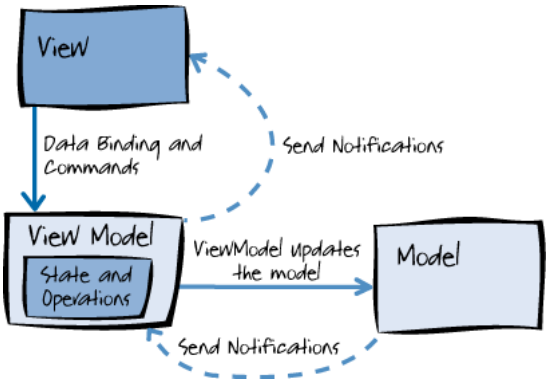
Figuur 2-11: Xamarin.Forms' oplossing architectuur [19]

De trade-off is dus de veelzijdigheid van Xamarin.Forms tegenover de volledige features en functionaliteit van de platform-specifieke UI's.

2.3.3.2 MVVM (Model-View-Viewmodel)

De Xamarin.Forms ontwikkelaar creëert zijn UI typisch in XAML, en voegt daarna zijn back-end code toe dat op de user interface werkt. Als applicaties groter en complexer worden kunnen onderhoudsproblemen toedreden.

Het Model-View-Viewmodel helpt met het scheiden van de ontwikkeling van de GUI en van de back-end logica. Deze gescheiden houden zorgt ervoor om problemen te voorkomen en de applicatie makkelijker testbaar en onderhoudbaar te maken. Het View Model van de MVVM is de tussenstap die de data objecten van het model verandert zodat deze gemakkelijk zijn te beheren en te presenteren door het View. [20]



Figuur 2-12: De relaties tussen de 3 kerncomponenten van MVVM [21]

Het Model stelt een deel klassen voor die de business logica en data voor. Het definieert ook de regels voor deze data hoe deze data kan veranderd en gemanipuleerd worden.

Het View stelt de UI componenten voor. Deze is enkel verantwoordelijk voor de data weer te geven die het verkrijgt van de controller, het ViewModel dus.

Het View Model is verantwoordelijk voor het onderhouden van de methodes, commands en andere eigenschappen die de state van het View moeten onderhouden, het model moet

manipuleren als resultaat van de acties op het view, en events in het view zelf te triggeren. [22]

De voordelen die het MVVM patroon aanbiedt:

- Het model, de back-end basis, aanpassen qua logica kan riskant of moeilijk zijn. In dit scenario kan het viewmodel bescherming bieden aan deze model klassen en voorkomt dat er schadelijke veranderingen zouden gebeuren in de model-code.
- Ontwikkelaars kunnen testen zonder de view te moeten gebruiken. Deze tests kunnen exact hetzelfde functioneren alsof ze gebruikt worden door een view.
- De UI van de applicatie kan compleet opnieuw ontworpen worden, als deze helemaal in XAML geïmplementeerd is, zonder ook maar één lijn back-end code aan te passen.
- Deze separatie van view en de andere componenten zorgt er ook voor dat designers en ontwikkelaars volledig apart kunnen werken aan hun eigen componenten. [23]

2.3.4 Gebruik

Xamarin gebruikt de Visual Studio IDE van Microsoft als ontwikkelingsomgeving. Xamarin kan geïnstalleerd worden als deel van een nieuwe Visual Studio installatie, of kan achteraf erbij geïnstalleerd worden. Nu kan de code voor applicaties geschreven worden in de editor van Visual Studio.

Met een installatie van Xamarin.Forms zal men enkel 1 code moeten schrijven, en deze wordt automatisch overgebracht naar Xamarin.iOS en Xamarin.Android. Een simulatie kan dan bekeken worden met de Xamarin.Forms Previewer die rechtstreeks in de IDE is geintegreerd.

2.3.5 Portfolio

Lijst van applicaties developped met Xamarin voor bekende bedrijven: [24]

- EasyJet’s app (Android/iOS)
- Snap Attack door Microsoft (Android/iOS)
- World Bank’s Survey Solutions (Android)
- Pepsi’s augmented reality app (Android/iOS)
- MixRadio’s muziek app (Android/iOS)
- Pinterest (Android/iOS)

2.4 Apache Cordova/PhoneGap

2.4.1 Oorsprong

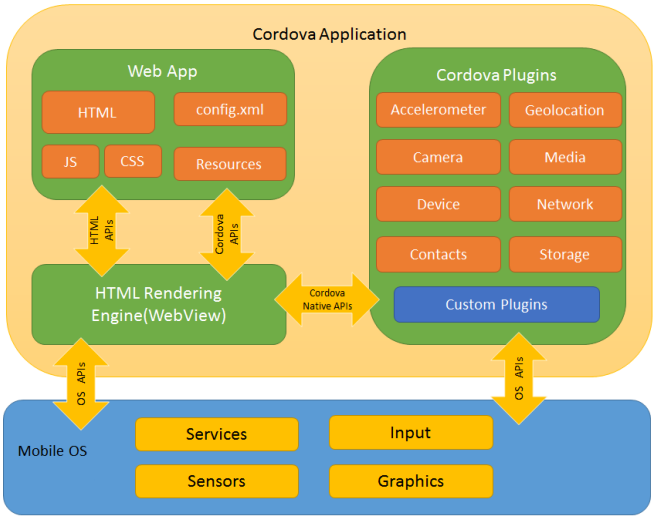
Apache Cordova is een ontwikkelingsframework voor mobiele applicaties gemaakt door het Canadese bedrijf Nitobi. Eerst noemde het PhoneGap, onder die naam werd het gecreëerd op een iPhoneDevCamp evenement in San Francisco in 2009 door het Canadese bedrijf Nitobi.

Adobe kocht in 2011 Nitobi over. [25] Nadat beslist werd door Adobe dat de PhoneGap codebase gedoneerd ging worden aan de Apache Software Foundation kreeg het de naam Apache Cordova, die dan ook open source werd. Dit terwijl PhoneGap de naam Adobe PhoneGap kreeg. [26] Wat is nu het verschil tussen PhoneGap en Apache Cordova? PhoneGap is Cordova plus Adobe services en extensies die de capaciteiten verder verbeteren. Cordova kan nog steeds apart gebruikt worden. [27]

2.4.2 Basis

Apache Cordova laat toe om standaard web technologieën -HTML5, CSS3 en JavaScript- te gebruiken voor cross-platform ontwikkeling. Applicaties worden uitgevoerd binnenin native applicatie wrappers die voor elk platform specifiek gebouwd worden door het framework. Hierop komt dan een WebView die toegang krijgt tot device-level APIs waardoor het gebruik kan maken van sensoren, data, netwerk status, … Deze WebView voorziet de hele UI voor de applicatie, voor sommige platformen kan dit ook een component zijn binnnenin een grote, hybride applicatie die de WebView mixt met native applicatie componenten. [28]

2.4.3 Architectuur



Figuur 2-13: Architectuur van een Cordova applicatie [28]

Als de architectuur bekeken wordt, valt op dat de Web App het deel is waar de geschreven applicatie code zit. De applicatie zelf is geïmplementeerd als een web pagina die CSS, JavaScript en de recourses die het nodig heeft refereert. De Cordova Plugins zijn zeer belangrijk, zij voorzien een interface voor Cordova en native componenten om met elkaar te communiceren en voorzien verbindingen naar standaard device APIs zoals de camera, contacten, … [28]

2.5 Kwaliteitscontrole van de applicaties

2.5.1 Theoretisch

Om uit de platformen van elkaar te onderscheiden wordt er bekeken in welke eigenschappen de platformen verschillen. Hieruit kunnen er al een conclusies trekken om in te schatten dewelke het meest geschikt is om een native Android/iOS implementatie te vervangen.

Platform	**Programmeertaal**	**WebView gebaseerd?**	**UI Rendering?**	**Open source?**
React Native	JavaScript en JSX	Nee	Native UI controllers	Ja
Xamarin	C# en XAML	Mogelijk als component maar Nee.	Native UI controllers	Ja, maar heeft een betaalde en uitgebreidere versie voor commercieel gebruik.
Qt	C++ en QML	Mogelijk als component maar Nee.	Native en OpenGL	Ja, maar heeft een betaalde en uitgebreidere versie voor commercieel gebruik.
Apache Cordova	HTML5, CSS3 en JavaScript	Ja	HTML.CSS	Ja, maar heeft platform (PhoneGap, Ionic, …) hierop gebouwd die wel bepaalde kosten zullen vragen voor bepaalde features te kunnen gebruiken.

Uit de programmeertalen wordt niet veel uit gehaald. Hierbij is het belangrijk met welke talen iemand het meest ervaren is en is het eerder een keuze uit comfort en kennis. Wel is het boeiend te bespreken dat JSX, XAML en QML alle drie talen zijn die afgeleid zijn van een bekend alternatief (JSX beetje uit HTML, XAML uit XML en QML lijkt vrij hard op CSS) maar toch vrij specifiek gebouwd zijn voor hun respectievelijke platformen. Hierdoor wordt geconcludeerd dat de programmeertalen puur eigen keuze zijn, maar dat het enige platform zonder leercurve voor ervaren programmeurs Apache Cordova zal zijn.

Gaat het platform een applicatie starten om er een WebView op te zetten om dan daarop de UI te tonen? Dit is enkel het geval voor Apache Cordova. Voor alle andere platformen zal er effectief Android/iOS componenten aan te pas komen. Dit is een voordeel bij zowel de opstarttijd en de *look and feel* van de applicatie sinds de applicaties ook eerder Android & iOS zullen aanvoelen.

Als er dan ook bekeken wordt hoe de platformen hun UI rendert, dus hoe het zijn componenten gebruikt om een UI op het scherm te krijgen dan wordt opgemerkt dat zowel Xamarin als React Native echt native UI controllers hanteren. Dit betekent dat bij deze platformen de applicatie niet enkel Android/iOS aanvoelt, maar effectief is. Voor deze reden is er de voorkeur gegeven om deze twee platformen uit te diepen.

Het feit dat alles open source te vinden is, is een voordeel aan al deze cross-platform frameworks. Maar het feit dat React Native al zijn features gratis aanbiedt, is één van de redenen dat deze het hoogste staat gerangschikt op onze theoretische lijst aangezien dit zeer interessant zal zijn voor een bedrijf als Zappware, die ook aan de financiële kant moet denken.

Als we al deze feiten opsommen dan wordt bevestigd dat theoretisch gezien **React Native** het boeiendste concept is van alle platformen. Met een combinatie van free-to-use, native UI controllers en een veelgebruikte bekende taal ziet hierin het meeste potentieel als de platformen vanuit een theoretische ooghoek bekeken worden.

Eén van de belangrijkste onderdelen van de criteria is de *look and feel,* aangezien **Xamarin** eveneens de native UI controllers gebruikt komt deze als 2de meest geschikte crossplatform framework qua potentieel.

Dan blijft Qt en Apache Cordova over. Aangezien **Qt** geen WebView gebruikt om zijn UI op te renderen, beschouwen we deze toch theoretisch beter dan **Apache Cordova**. Door tijdsgebrek zal niet alles geïmplementeerd kunnen worden, hierdoor zullen deze laatste twee applicaties niet effectief gebouwd worden. Deels omdat er al ervaring was opgedaan met Qt in het vorige jaar en daaruit is gebleken dat dit platform toch geen ideaal cross-platform bleek.

2.5.2 Hoe wordt dit praktisch gecontroleerd?

Eerst worden er besproken hoe dit in het algemeen wordt aangepakt. Later wordt besproken hoe de applicaties het stuk voor stuk deden.

2.5.2.1 Development effort

Om de ontwikkelingstijd van de applicatie te meten, worden de stappen besproken die nodig zijn om:

- De omgeving in te stellen.
- De code en emulator/real device op punt te stellen.
- De applicatie uit te rollen naar iOS en Android.

Hieruit moeten de volgende vragen kunnen beantwoord worden:

- Hoe groot is de overgangstap naar het benoemde platform qua setup?
- Hoe moeilijk is de taal, vanuit een beginnerstandpunt?
- Hoe lang duurt het implementeren van een prototype applicatie?
- Hoe gedocumenteerd is de taal?
- Wat zijn de mogelijkheden van het platform qua schaalbaarheid, bij grotere complexere applicaties?
- Is de mogelijkheid om de applicatie te publiceren reëel?
- Is het platform eerder geschikt om prototypes mee te bouwen of kan het als effectieve vervanger van de native Android en iOS projecten dienen?

Bij dit hoofdstuk wordt eerst onze persoonlijke ervaring met het proces van de ontwikkeling op een chronologische wijze besproken, daarna wordt we veralgemeend en conclusies getrokken.

2.5.2.2 Snelheid

Android

Om de snelheid van onze applicaties te testen wordt er gebruik gemaakt van *appachhi.com*, een test-API die via een cloud verzekert dat de Android applicaties die er ontwikkeld zijn, vlot werken op een bereik van verschillende devices. Voor die tests worden er op 5 verschillende devices de applicaties uitgeprobeerd. Hieronder is ook de Motorola Moto E (2nd generation) gratis te gebruiken en die leunt het dichtst aan het eigen apparaat. De belangrijkste specificaties van deze GSM zijn:

OS: Android 6.0

API level: 23

CPU: ARM | quad-core | 1200 MHz

RAM: 1024 MB

Maar aangezien wij onze applicaties vergelijken met hetzelfde apparaat zal dit in onze vergelijkende studie niet uitmaken.

Bij snelheid gaat verstaan worden: de installatiesnelheid, de tijd die dit apparaat nodig heeft om de applicatie te installeren, en de opstarttijd, de tijd die het nodig heeft om de geïnstalleerde app op te starten zonder dat er iets in het cachegeheugen gestoken is.

iOS

2.5.2.3 Geheugengebruik

Android

Het geheugengebruik wordt getest op een eigen echt apparaat, geen simulatie dus. Dit gebeurd op een Motorola Moto C Plus met als belangrijkste specificaties:

OS: Android 7.0

API level: 23

CPU: Quad-core 1.3 GHz Cortex-A53

RAM: 2028 MB

Hier wordt het cache-geheugen, het RAM-geheugen en het interne geheugen met elkaar vergeleken. Hoe minder geheugen er gebruikt wordt, hoe beter.

iOS

2.5.2.4 User experience

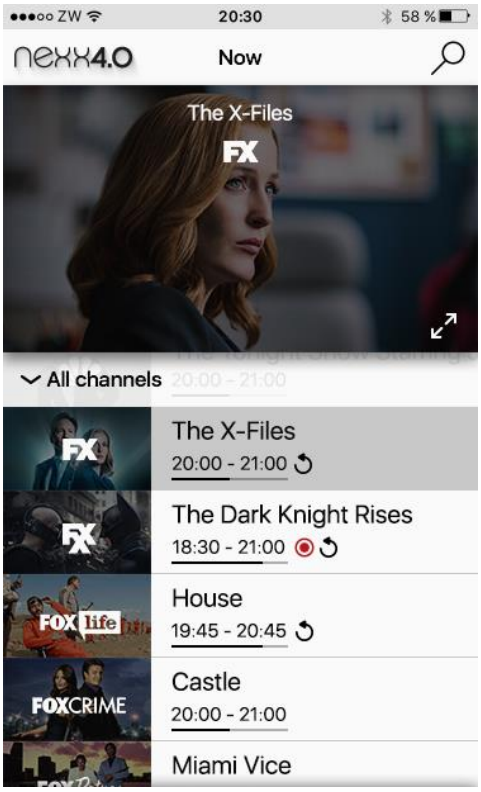
3 APPLICATIE

De applicatie die gebouwd gaat worden via de crossplatform frameworks gaat een vrij basic tv-gids applicatie zijn. Deze blijven vrij basic zodat de mogelijkheid bestaat om deze te kunnen vergelijken op basis van verschillende criteria zoals development effort, snelheid, geheugen, … De layout zal op de platformen verschillen van elkaar afhankelijk van welke componenten beschikbaar zijn sinds er in andere talen gewerkt wordt. Het algemene beeld van de applicatie gaat er als volgt uitzien:

Vanboven zijn er drie componenten: één logo zonder functies, één zone die het uur/moment/dag aanduid, één zoek-optie met een zoekfunctie waarbij er bepaalde shows kunnen gezocht worden.

Hieronder is de video-component die een stream van het huidige onderdeel op het aangeduide kanaal moet kunnen weergeven.

Beneden wordt er per kanaal laten zien wat er op het aangeduide moment bezig is, hoelang dit al bezig is, of dit opgenomen wordt, … En hierbij moet er ook één kanaal kunnen aangeduid worden om te laten zien wat het programma is van dit kanaal op die dag. Deze optie is er achteraf uitgelaten sinds dit enkel meer codewerk zou zijn en niet essentieel is. Dit word gedaan om de tijd beter te kunnen beheren.



Figuur 3-1: Einddoel applicatie

Om een voorbeeld applicatie te bouwen is er een JSON bestand geüpload op een webserver met behulp van www.myjson.com zodat deze kan aangesproken worden op elk apparaat via het internet. Dit JSON-bestand bevat alle gegevens over de programmatie van de kanalen, alle metadata van de kanalen, het feit dat het programma momenteel aan het opnemen is, … en is gemakkelijk aan te spreken in alle programmeertalen zodat dit consequent kan gebeuren bij de verschillende platformen. Eveneens is dit een kleine database met 23 kanalen zodat eventuele performantie-problemen niet onnodig groot zijn en nog op dezelfde manier kunnen beoordeeld worden.

Het doel is om op elk platform deze voorbeeldapp zo goed mogelijk na te bouwen en zo de limieten en mogelijkheden van het platform te vinden.

4 VERGELIJKEN PERFORMANCE

4.1 Development effort

4.1.1 React Native

4.1.1.1 Het proces

Voor te beginnen aan een React Native applicatie, heeft men een Javascript-library en een React-framework nodig. Via Node.js, een softwareplatform voor Javascript-toepassingen die we gewoonweg gebruiken met behulp van de command prompt van de PC, installeren we “Create React Native App”. In dit framework zit alles in dat we nodig hebben, zodra dit geinstalleerd is kunnen we een React Native project opzetten. Dit project kunnen we via een tekst-editor aanpassen en uitbreiden. Expo is een applicatie die we dan op onze GSM kunnen installeren om een voorbeeld van onze applicatie te projecteren. Dit framework opzetten in zijn geheel duurde 1,5u in totaal. Men kan ook een emulator van Android Studio of Xcode gebruiken voor je voorbeeld.

Documentatie voor RN is vrij uitgebreid te vinden op het internet. Voor een complexvrije applicatie zoals de onze is er meer als genoeg informatie te vinden op de officiële pagina van RN. Wat opviel bij het gebruiken van de RN-framework was dat bij errors het debuggen niet vlot verliep. Errors gaven problemen aan op plaatsen die niet het probleem waren, problemen met de emulator, problemen bij het installeren van bepaalde componenten voor je applicatie, … Op het internet staan wel oplossingen maar het was een serieuze zoektocht. Daardoor is er veel tijd kwijtgeraakt aan kleine problemen. De algemene indruk was duidelijk dat het een vrij nieuwe taal is, die nog volop wordt ontwikkeld. Hierdoor zijn er nog heel wat complicaties mogelijk en niet altijd gemakkelijk op te lossen.

Voor bepaalde componenten te gebruiken kon men Expo ook niet meer gebruiken omdat sommigen niet ondersteund werden, waardoor je je project moest losmaken van Expo. Dit betekende dan ook dat een heel deel van de infrastructuur wegviel waardoor alles via Android Studio moesten opzetten en alles niet meer vlot verliep zoals bij het Expo-build proces.

Als de app af was, moest deze app nog zowel in Android als iOS gebouwd worden. Bij Android gebruiken we onze Windows command line om een *release* versie van onze applicatie te bouwen. Om een applicatie te kunnen runnen in een echte Android omgeving moet deze gesigneerd worden met een certificaat voordat ze kunnen worden geïnstalleerd en dus moeten we een key genereren waarmee we een APK kunnen genereren die gesigneerd is.Deze key is te maken via een tool die Java aanbied: *keytool.* Met deze tool krijgen we een file*: my-release-key.keystore* die we in de map van onze applicatie steken en via onze *gradle* code van Android aanspreken met het juiste wachtwoord dat we hebben ingesteld waarop onze applicatie in Androidomgevingen kan geïnstalleerd worden. Dit instellen, bouwen en uitzoeken duurde 1 uur in totaal.

De installatie op iOS volgt een analoog concept. Aangezien dit na het Android gegeven verliep, moesten we enkel de omgeving errond regelen en de benodigde pakketten voor onze applicatie installeren.

Het deployment van deze iOS applicatie loopt eveneens analoog. In XCode wordt de React Packager ook gebruikt om de applicatie te bouwen naar een .ipa vorm. Deze moet dan gesigneerd worden door een geverifieerd Apple ID. De opgave ging niet vanzelf, maar na enkele aanpassingen – die allemaal online te vinden waren – en enkele uren was het uiteindelijk gelukt een werkende .ipa file te verkrijgen.

Door enkele problemen werd ook een simpel project tijdrovend om te creëren en heeft het uiteindelijk zo'n 80 uren geduurd om het te maken van begin tot einde. Hierbij moet er rekening gehouden worden met het gebrek aan ervaring en know-how waarmee ik dit project begon, aangezien het de eerste applicatie was die gemaakt werd. Hierdoor is dit getal niet volledig representatief.

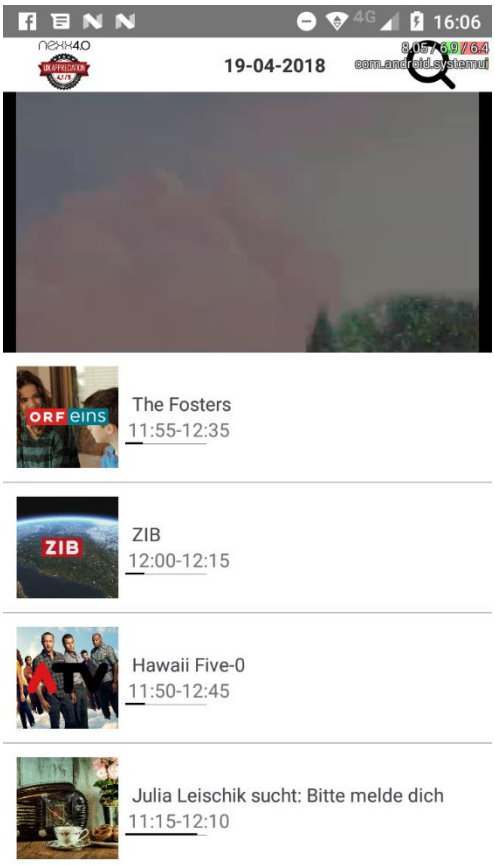
4.1.1.2 Details

Beginnend aan zo een applicatie, wordt eerst de structuur van de app nagemaakt en in componenten opgedeeld:

Volgens de Javascript componenten (<>):

- Vanboven: één <View> opgedeeld in 3 stukken rij, daarin kunnen we 3 componenten steken. In dit geval:
    - Links: <Image>
    - Midden: <DatePicker>: Een component die ervoor zal zorgen dat je een datum kunt kiezen, zowel in Android als iOS
    - Rechts: <Image> met functionaliteit onPress: een zoekactie ondernemen.
- Midden: een <VideoPlayer> component die enkel een url als bron nodig heeft om te werken in zowel Android als iOS.

- Vanonder: een <Flatlist> waarmee het JSON-bestand kan uitlezen worden en via <ListItem> deze één voor één getoond wordt. In component ListItem wordt dan bepaald welke data gekozen wordt via verschillende props die worden ingesteld.



Figuur 4-1: React Native applicatie

4.1.2 Xamarin

4.1.2.1 Het proces

Voor de Xamarin applicatie hebben is de Visual Studio IDE nodig. Deze installatie duurt lang en neemt veel geheugen in. Na 3 uur is de installatie compleet en kunnen we beginnen aan een lege applicatie. Bij deze lege applicatie waren er al direct problemen: vele instellingen stonden standaard verkeerd en het debuggen via een emulator had allerlei kinderziektes, van niet opstarten tot errors aanduiden zonder een reden te geven. Hiervoor was op het internet wel oplossingen te vinden maar doordat de problemen bleven opstapelen duurde het nog 3 uur voordat er effectief begonnen kon worden aan de applicatie met de juiste omgeving. Voor een snelle preview bestaat er in VS het handig venster Xamarin.forms previewer, hier krijg je al te zien hoe je applicatie eruit ziet op alle verschillende platformen.

Om de applicatie effectief te testen gebruiken we een real device, de Moto C Plus, via USB. Dat zou ook gaan met Xamarin Live, dit is een applicatie in de Play Store of App Store waarbij de gsm verbonden wordt met Visual Studio en in real-time een voorbeeldapplicatie op je apparaat kan vrijgeven worden. Maar dit doen we niet sinds hierop geen debugging kan gebeuren van ons code, wat cruciaal is. Er zijn ook mogelijkheden in VS verwerkt via emulators waarvan de installatie omslachtig is en veel geheugen in gebruik neemt, daarom gebruiken we een real device.

Tijdens het schrijven van de code viel direct op hoe sommige componenten niet beschikbaar waren. Een videoplayer-component vinden die native videospelers aanspreekt was -in het begin- enkel te vinden tegen betaling. Terwijl deze voor React Native wel direct meerdere gratis versies voor te vinden was. Uiteindelijk is het gelukt een werkende, gratis Xamarin video player te verkrijgen door het toch wel uitgebreide gamma aan open source pakketten te doorzoeken. Het installeren van deze uitbreidingen ging vlot maar het implementeren ging moeizamer. Dit kwam door errors die lagen aan het .NET framework waarop dat gebaseerd was terwijl mijn VS een heel ander .NET framework had getarget, na het evalueren van onze geïnstalleerde componenten konden we concluderen dat we nog een paar Visual Studio installatie onderdelen tekort kwamen. Er moet dus rekening gehouden worden met het feit dat de standaardinstallatie van VS niet genoeg is om alle componenten te kunnen gebruiken.

Na 40 uren werken aan de Xamarin implementatie was de applicatie nog niet af. Door tijdsgebrek werd het project gepauzeerd. Ondanks dat was onze structuur intact en moesten er enkel nog details afgewerkt worden, maar deze konden potentieel nog vrij tijdrovend zijn.

Het uitrollen van de applicatie op Android was duidelijk dankzij de interne documentatie van Xamarin. Maar bij het uitrollen waren er toch nog wat kinderziektes die VS maar niet kan vermijden. Deze waren vrijwel direct opgelost, de APK-file was direct bruikbaar.

//Op iOS was het uitrollen vrijwel analoog.

4.1.2.2 Details

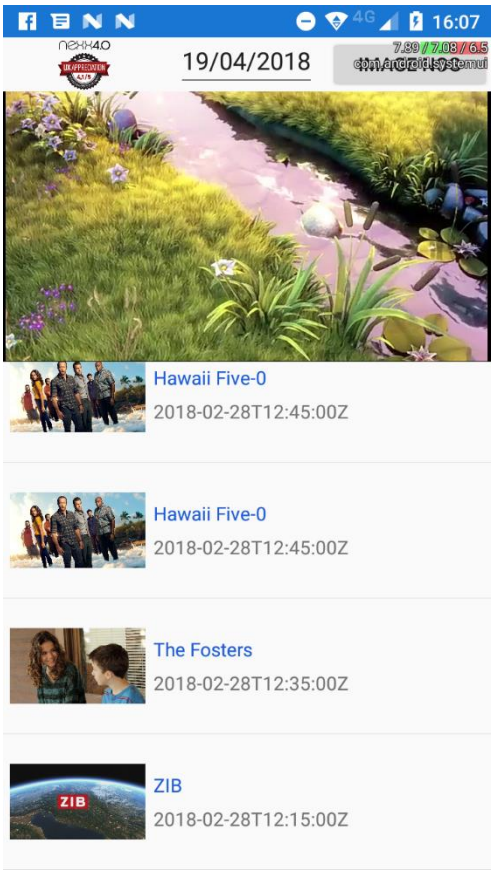
De Xamarin implementatie volgt volgende structuur:

Volgens XAML componenten (<>):

- : één <Grid> opgedeeld in 3 stukken kolom en 1 rij, daarin kunnen we 3 componenten steken. In dit geval:
    - Links: <Image>
    - Midden: <DatePicker>:
    - Rechts: <Button> met een standaard eigenschap onClicked die via een C# functie te verbinden is met de gegeven functionaliteit (niet geimplementeerd).
- Midden: een <VideoPlayer> component die enkel een url als bron nodig heeft om te werken in zowel Android als iOs.
- Vanonder: een <Listview> die statisch geregistreerde data-onderdelen weergeeft. Bij de <ListView> zou een JSON uitgelezen moeten worden. Om de structuur van onze applicatie te behouden hebben we onze JSON onderdelen dan omgezet naar statische eigenschappen van een klasse <List>, de data manueel toegevoegd en deze klasse gebruikt als databron voor onze <ListView>.

Meer niet-geïmplementeerde onderdelen wegens tijdsgebrek:

- Het beperken van de officiële tijdsformat tot een leesbare vorm.
- Een tijdsbalk die aangeeft hoever het programma gevorderd is.
- <Image> in <Button> van het zoekgedeelte plaatsen.



Figuur 4-2: Xamarin applicatie

4.1.3 Conclusies

Het gebruik van Xamarin en React Native is op zich weinig verschillend. Het feit dat React Native geen IDE gebruikt is een groot voordeel voor de setuptijd van het framework, dit kost maar een uur, terwijl de installatie van Visual Studio zelf al 3 uren duurde. Hierbij komt dan wel kijken dat er veel tools voor het debuggen ontbreekt, die Visual Studio wel uitgebreid aanbiedt. Er kan enkel uitgegaan worden van de error-berichten die de applicatie aangeeft. Maar hierop zijn er alternatieven, zoals React Native Devtools, die open-source te vinden zijn en die meer info bieden over details van de applicatie.

De talen waarin geschreven werd voelen vertrouwd aan. In zowel C# als Javascript werd er inmiddels al ervaring opgedaan. Maar het feit dat React Native voornamelijk JS was voelde toch als een gemakkelijkere omgeving dan de C# van Xamarin, waarbij de structuur toch onduidelijker was en waarbij structurele fouten niet zo makkelijk werden ingezien als bij JS. Javascript voelt directer en eenvoudiger aan. Zeker bij een simpele applicatie als de deze.

Dit wordt ook vereenvoudigd door het feit dat de documentatie rond Xamarin vrij onduidelijk en moeilijk te vinden is. Goede voorbeelden zijn te vinden na lang zoeken, maar hun eigen

documentatie brengt meestal niet meer op dan weten wat een bepaalde functie doet, zonder context. De gemeenschap rond React Native lijkt veel actiever en behulpzamer. Voor elk soort probleem was er al een antwoord. Bij Xamarin daarentegen moest er begonnen worden bij een vraag rond een probleem en zo een oplossing proberen te verzinnen, dit kost tijd en vergt inzicht. Zeker voor een onervaren persoon zal dit een doorslag geven. Bij beiden was het proces van de setup , allerlei installaties en de essentiële basis goed gedocumenteerd.

De structuur van beide frameworks is bruikbaar voor grotere complexe applicaties. Dit dankzij hun MVVM- en Flux-architectuur. Bij Xamarin zorgt MVVM ervoor dat de back-end en de front-end mooi gescheiden blijft, en werkt met een tussenstation voor het versturen van de data. Terwijl voor RN Flux de data slechts naar één kant stuurt waarbij één dispatcher alle *callbacks* verzend naar de entiteiten waarvoor deze bestemd is.

// deployment Android & iOS

Uit deze conclusie kan er een tabel gemaakt worden die kort weergeeft wat de voor- en nadelen zijn van onze twee cross-platform frameworks vergeleken met elkaar.

	React Native	**Xamarin**
Setup	Kort (1u) en palmt weinig geheugen in.	Relatief lang (3u) en palmt veel geheugen in.
Debugging	Mogelijkheden beschikbaar.	Uitgebreide geïntegreerde mogelijkheden.
Programmeertalen	Vertrouwd, weinig complexite.it	Vertrouwd, matige complexiteit.
Documentatie	Uitgebreid qua eigen documentatie. Zeer veel vragen/antwoorden op het internet.	Matig uitgebreid qua eigen documentatie. Matig veel vragen/antwoorden op het internet.
Scalability	Zeer schaalbaar	Zeer schaalbaar
Deployment		

Bij deze vergelijking kan er geconcludeerd worden dat RN het betere cross-platform framework kan beschouwd worden wat de ontwikkeling zelf betreft. Dit baseren we op

4.2 Snelheid

4.2.1 Vergelijking

Bij deze studie wordt de installatiesnelheid en de opstartsnelheid van de applicaties in React Native en Xamarin vergeleken met degene die in Android en iOS native gebouwd zijn. Vooral de opstartsnelheid is uitermate belangrijk aangezien deze actie een cruciaal onderdeel is van een mobiele applicatie.

Android

Voor Android bekijken we via *Appachhi.com* de snelheden van de applicatie op de Motorola Moto E (2nd generation) en vergelijken we deze met elkaar.

	Native	React Native	Xamarin
Installatietijd		$9\ seconden$	$10.75\ seconden$
Opstarttijd		$0.4\ seconden$	$5.92\ seconden$

Bij deze is het ook belangrijk te realiseren dat de grootte en uitgebreidheid van de drie applicaties verschillen. De complexiteit van de native app is groter dan die van RN, en de RN app is volledi ger dan die van Xamarin. Hierdoor is het toch opvallend dat de RN applicatie veel beter scoort op snelheid dan degene gebouwd met Xamarin.

iOS

4.2.2 Conclusie

4.3 Geheugen

4.3.1 Vergelijking

Android

iOS

4.3.2 Conclusie

4.4 User experience

4.5 Conclusie

5 BESLUIT

6 VERWIJZINGEN

[1] Zappware NV, „Info over Zappware NV,” [Online]. Available: https://zappware.com/.

[2] „Qt (software),” [Online]. Available: https://en.wikipedia.org/wiki/Qt_(software). [Geopend Maart 2018].

[3] Digia , „Digia and Qt have demerged into two companies – Digia’s new strategy’s main themes revealed,” [Online]. Available: http://digia.com/en/actual/news/2016/digia-and-qt-have-demerged-into-two-companies--digias-new-strategys-main-themes-revealed/. [Geopend Maart 2018].

[4] N. Mehrotra, „Qt: What’s best about the cross-platform development toolkit,” [Online]. Available: http://opensourceforu.com/2017/06/qt-cross-platform-development-toolkit/. [Geopend Maart 2018].

[5] B. Starynkevitch, „How is QT cross-platform,” [Online]. Available: https://www.quora.com/How-is-QT-cross-platform. [Geopend Maart 2018].

[6] „OpenGL,” [Online]. Available: https://en.wikipedia.org/wiki/OpenGL. [Geopend Maart 2018].

[7] „GLUT - The OpenGL Utility Toolkit,” [Online]. Available: https://www.opengl.org/resources/libraries/glut/. [Geopend Maart 2018].

[8] Qt, „Signals & slots,” [Online]. Available: http://doc.qt.io/archives/qt-4.8/signalsandslots.html. [Geopend Maart 2018].

[9] „Applications using Qt,” [Online]. Available: https://en.wikipedia.org/wiki/Qt_(software)#Applications_using_Qt. [Geopend Maart 2018].

[10] „React (JavaScript library),” [Online]. Available: https://en.wikipedia.org/wiki/React_(JavaScript_library). [Geopend Maart 2018].

[11] M. Konicek, „React Native: A year in review,” [Online]. Available: https://code.facebook.com/posts/597378980427792/react-native-a-year-in-review/. [Geopend Maart 2018].

[12] B. Eisenman, Learning React Native, O'Reilly Media, Incc., 2016.

[13] „Document Object Model,” [Online]. Available: https://en.wikipedia.org/wiki/Document_Object_Model. [Geopend Maart 2018].

[14] M. Tilley, „What is Flux?,” [Online]. Available: http://fluxxor.com/what-is-flux.html. [Geopend Maart 2018].

[15] Redux, [Online]. Available: https://redux.js.org/introduction/prior-art. [Geopend Maart 2018].

[16] D. Abramov, „Why use Redux over Facebook Flux?,” [Online]. Available: https://stackoverflow.com/questions/32461229/why-use-redux-over-facebook-flux. [Geopend Maart 2018].

[17] React Native, „Who's using React Native?,” [Online]. Available: http://facebook.github.io/react-native/showcase.html. [Geopend Maart 2018].

[18] „Xamarin,” [Online]. Available: https://en.wikipedia.org/wiki/Xamarin. [Geopend Maart 2018].

[19] D. Hermes, Xamarin Mobile Application Development - Cross-platform C# and Xamarin.Forms Fundamentals, Apress, 2015.

[20] „Model-View-Viewmodel,” [Online]. Available: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel. [Geopend April 2018].

[21] Microsoft, „The MVVM Pattern,” [Online]. Available: https://msdn.microsoft.com/en-us/library/hh848246.aspx. [Geopend April 2018].

[22] „Understanding MVC, MVP and MVVM Design Patterns,” [Online]. Available: http://www.dotnettricks.com/learn/designpatterns/understanding-mvc-mvp-and-mvvm-design-patterns. [Geopend April 2018].

[23] Microsoft, „MVVM,” [Online]. Available: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm. [Geopend April 2018].

[24] Xamarin, [Online]. Available: https://www.xamarin.com/customers. [Geopend Maart 2018].

[25] „Apache Cordova,” [Online]. Available: https://en.wikipedia.org/wiki/Apache_Cordova. [Geopend April 2018].

[26] B. LeRoux, „PhoneGap, Cordova, and what's in a name?,” [Online]. Available: https://phonegap.com/blog/2012/03/19/phonegap-cordova-and-whate28099s-in-a-name/. [Geopend April 2018].

[27] S. A. Wilkins Fernandez, Beginning App Development with Parse and PhoneGap, Apress, 2015.

[28] Apache Cordova, „Overview Apache Cordova,” [Online]. Available: https://cordova.apache.org/docs/en/latest/guide/overview/. [Geopend April 2018].

[29] M. S. Jasmin Blanchette, C++ GUI Programming with Qt 4, Trolltech Press, 2006.

[30] B. C. Daniels, „QT – Introduction C++ GUI Programming with Qt 4,” [Online]. Available: http://slideplayer.com/slide/7747920/. [Geopend Maart 2018].

Bijlagen

Bijlage A Javascript-code van de React Native applicatie

Bijlage B XAML-code van de Xamarin applicatie

Bijlage C C#-code van de Xamarin applicatie

A.

```
"use strict";
import React, {Component} from 'react';
import {StyleSheet, Image, View, ScrollView, ListView, Text, TouchableHighlight, AppRegistry, Modal, FlatList} from 'react-native';
import VideoPlayer from 'react-native-video-controls';
import DatePicker from 'react-native-datepicker';
import SearchBar from 'react-native-searchbar';
import Dimensions from 'Dimensions';
import {List, ListItem} from 'react-native-elements';
import Hr from 'react-native-hr-component';




var elements = [];
          function parseJsonDate(jsonDateString){
    return Date.parse(jsonDateString);

}

     function calculatePercentage(start, end){
          var starttijd = parseJsonDate(start);
          var eindtijd = parseJsonDate(end);
          var nu = parseJsonDate('2018-02-28T12:03:30Z');
          var calc1 = eindtijd - starttijd;
```

```
            var calc2 = eindtijd - nu;
            var calc3 = calc1 - calc2;
            var calcpercent = calc3/calc1;
            return calcpercent;
      }
      function calculatePercentageAnder(start,end){
            var starttijd = parseJsonDate(start);
            var eindtijd = parseJsonDate(end);
            var nu = parseJsonDate('2018-02-28T12:03:30Z');
            var calc1 = eindtijd - starttijd;
            var calc2 = eindtijd - nu;
            var calc3 = calc1 - calc2;
            var calcpercent = 1 - calc3/calc1;
            return calcpercent;
      }

export default class App extends React.Component {
      constructor(){
            super()

            var today = new Date(),
            today = today.getDate() + '-' + (today.getMonth() + 1) + '-'
+today.getFullYear();
            global.vandaag = today;



            this.state = {
            items,
         results: [],
            date: today,
            count: 0,
            modalVisible: false,
            data: []

            };
             this._handleResults = this._handleResults.bind(this);

      }
```

```
      componentWillMount() {
  this.fetchData();
}

fetchData = async () => {
  const response = await fetch("https://api.myjson.com/bins/87f6f");
  const json = await response.json();
  this.setState({ data: json.data.channelList.channels.edges });

};



   renderdate(){
         if (this.state.date == vandaag){
               return (
               this.state.date
               );
         }
         else{
               return (
         this.state.date
         );
         }}
         _handleResults(results) {
        this.setState({ results });
      }




render() {

      let logo = {
      uri: 'http://zappware.com/wp-content/uploads/2017/04/1.png'}
      let zoek = {
```

```
        uri:        'https://cdn4.iconfinder.com/data/icons/pictype-free-vector-
icons/16/search-128.png'}
        ;
        var { width, height } = Dimensions.get('window')
    return (



     <View style={styles.overallcontainer}>
     <Modal      visible={this.state.modalVisible}      animationType={'slide'}
onRequestClose={() =>this.closeModal()}>
          <View style={{ marginTop: 110 }}>


                {
                          this.state.results.map((result, i) => {
                            return (
                              <Text key={i}>
                                {typeof   result   ===   'object'   &&   !(result
instanceof Array) ? 'gold object!' : result.toString()}
                              </Text>
                            );
                          })
                        }
        </View>
    </Modal>
           <View style={styles.container}>
                <Image      source={logo}      style={{flex:1,      resizeMode:
Image.resizeMode.contain}} />
                <View style={{flex: 2, backgroundColor: 'white'}} >
                      <DatePicker
                           style={{width: 200}}
                           date={this.renderdate() }
                           mode="date"
                           placeholder="select date"
                           format="DD-MM-YYYY"
                           minDate="01-06-2016"
                           maxDate="02-07-2019"
                           confirmBtnText="Confirm"
                           cancelBtnText="Cancel"
                           showIcon={false}
```

```
                            hideText={false}
                            customStyles={{
                              dateInput: {
                                  borderWidth: 0,
                                  marginLeft: 20,
                              },
                              dateText:
                              {
                                  fontWeight: 'bold',
                              }
                            }}
                            onDateChange={(date)     =>     {this.setState({date:
date})}}
                          />
                </View>
                <TouchableHighlight   underlayColor="white"   onPress={()   =>
this.doSearchopen() } style={{flex: 1, backgroundColor: 'white', marginBottom:
2, marginTop: 2}} >
                      <Image    source={zoek}    style={{flex:1,    resizeMode:
Image.resizeMode.contain,}} />


                </TouchableHighlight>
           </View>
           <View style={styles.video} >
                <VideoPlayer                    source={{                    uri:
'http://d23dyxeqlo5psv.cloudfront.net/big_buck_bunny.mp4'     }}     navigator={
this.props.navigator }/>
           </View>
           <View style={styles.channel} >



                <FlatList
          data={this.state.data}
          keyExtractor={(x, i) => i}
          renderItem={({ item }) =>
            (<ListItem
                hideChevron={true}
                avatar ={
                      <View  style={{height:  75,  width:  75  ,justifyContent:
'center', alignItems: 'center'}}>
```

```
                    <Image                                          source={{uri:
item.node.eventsAt.TVitem.smallImage.url}}      style={{height:75,   width:75,
justifyContent:    'center',    alignItems:    'center',    backgroundColor:
'rgba(0,0,0,0.5)' }}   >
                    <Image         source={{uri:         item.node.logo.url}}
style={{resizeMode: 'center', height: 60, width: 60, alignItems: 'center', }}
/>
                    </Image>
                    </View>
               }
               title={item.node.eventsAt.TVitem.title}
               subtitle={
                    <View style={styles.totalbox}>
                         <Text            style={styles.textbox}            >
{item.node.eventsAt.TVitem.start.substring(11,16)}-
{item.node.eventsAt.TVitem.end.substring(11,16)}</Text>
                    <View style={styles.viewboxoverhead}>
                         <View                                    style={{flex:
calculatePercentage(item.node.eventsAt.TVitem.start,
item.node.eventsAt.TVitem.end), borderBottomColor: 'black', borderBottomWidth:
1.5,marginLeft: 5}} />
                              <View                              style={{flex:
calculatePercentageAnder(item.node.eventsAt.TVitem.start,
item.node.eventsAt.TVitem.end), borderBottomColor: 'grey',  borderBottomWidth:
0.3, marginRight: 200}} />
                         </View>
                    </View>
               }
               />
          )}
       />
          </View>
          <SearchBar
            ref={(ref) => this.searchBar = ref}
            data={items}
            handleResults={this._handleResults}
          />
     </View>
```

```
 );}

 doSearchopen(){
 this.searchBar.show();
 this.setState({modelVisible:true});
 }
 doSearchclose(){
 this.searchBar.hide();
 this.setState({modelVisible:false});
 }
}
const items = [

];


const styles = StyleSheet.create({
  container:
  {
     height: 40,
    backgroundColor: "white",
      flexDirection: 'row',
  },


  video:
  {
        flex: 1,
        backgroundColor: 'red',
  },
   channel:
  {
        flex: 2,
        backgroundColor: 'white',
        flexDirection: 'column',
  },
```

```
  overallcontainer:
  {
        flex: 1,
        backgroundColor: 'blue',
  },
  scrollview:
  {
        flex: 1,


  }
  ,
  totalbox:{
        flexDirection: 'column'
  },
  textbox:
  {
        flex:1,
  },
  viewboxoverhead:
  {
        flexDirection: 'row',
        flex:1,
  },

});
```

B

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:local="clr-namespace:XamarinProjectWerkNuIs"
             xmlns:xamarians="clr-
namespace:Xamarians.MediaPlayer;assembly=Xamarians.MediaPlayer"
             x:Class="XamarinProjectWerkNuIs.MainPage">
    <ActivityIndicator VerticalOptions="Center" HorizontalOptions="Center"
x:Name="activity_indicator" Color="#4D7EE1" />
    <StackLayout Spacing="0" x:Name="layout">
        <Grid HorizontalOptions="FillAndExpand" VerticalOptions="Start" ColumnSpacing="0" >
            <Grid.RowDefinitions>
                <RowDefinition Height="40" />

            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*"/>
                <ColumnDefinition Width="*"/>
                <ColumnDefinition Width="*"/>
            </Grid.ColumnDefinitions>
            <Image Source="http://zappware.com/wp-content/uploads/2017/04/1.png"
Grid.Column="0" VerticalOptions="FillAndExpand" HorizontalOptions="FillAndExpand"/>
            <DatePicker Grid.Column="1" VerticalOptions="End" HorizontalOptions="Center"/>
            <Button Text="Image nog aanpassen"  Grid.Column="2" VerticalOptions="End"
HorizontalOptions="End" />
        </Grid>


        <xamarians:VideoPlayer Source="http://d23dyxeqlo5psv.cloudfront.net/big_buck_bunny.mp4"
AutoPlay="True" HeightRequest="200" VerticalOptions="Fill"/>



        <ListView x:Name="myList" HasUnevenRows="true" ItemsSource="{Binding Items}" >
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <!-- Here we make a Horizontal orientation with the help of
StackLayout-->
                        <StackLayout Orientation="Horizontal" Margin="5" HeightRequest="90">
                            <Image Source="{Binding smallimageURL}" WidthRequest="100"
HeightRequest="200" Aspect="AspectFit" >
                                <!--<Image Source="{Binding imageURL}" WidthRequest="20"
HeightRequest="40" Aspect="AspectFit" VerticalOptions="Center"/>-->
                            </Image>
                            <StackLayout VerticalOptions="Center">
                                <Label Text="{Binding title}" TextColor="#1C5AD8" />
                                <StackLayout Orientation="Horizontal" Padding="0">
                                    <Label Text="{Binding end}"  />

                                </StackLayout>

                            </StackLayout>
                        </StackLayout>

                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
```

```
    </StackLayout>
</ContentPage>
```

C

```
using Newtonsoft.Json;
using Plugin.Connectivity;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Diagnostics;
using System.Linq;
using System.Net.Http;
using Xamarin.Forms;
using XamarinProjectWerkNuIs.ViewModels;

namespace XamarinProjectWerkNuIs
{
    public partial class MainPage : ContentPage
    {
        public int Count = 0;
        public short Counter = 0;
        public int SlidePosition = 0;
        int heightRowsList = 90;

        private const string Url = "https://api.myjson.com/bins/e7jqf";

        // This handles the Web data request
        private HttpClient _client = new HttpClient();

        public MainPage()
        {
            InitializeComponent();

            BindingContext = new ItemListViewModel();
        }


            }
        }
```

FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN
CAMPUS GEEL
Kleinhoefstraat 4
2440 GEEL, België
tel. + 32 14 80 22 40
iiw.geel@kuleuven.be
www.iiw.kuleuven.be