KU LEUVEN

FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN

CAMPUS GEEL

Development in cross-platform frameworks vergeleken met platform-specifieke APIs

Evaluatie van Qt, Xamarin, React Native en Apache Cordova

Lennert VAN LOOVEREN

Promotor: Prof. P. 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

Ik heb deze masterproef kunnen verwezenlijken dankzij mijn ouders, via hun morele en financiële steun. Alsook door mijn promotoren die mij gedurende de hele weg begeleidt hebben.

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

Symbolenlijst

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

1 INLEIDING

Zappware

Deze masterproef werd uitgevoerd bij Zappware in Hasselt. Een globaal bedrijf dat zich zowel bezig houdt met het ontwerpen van video UI design als client-software development. Deze masterproef kadert binnen de client-software kant. Hierbij werden er oplossingen gezocht om het proces van individuele APIs niet te moeten gebruiken bij het programmeren van mobiele applicaties. Zappware voorzag voorbeelden van mogelijke applicaties om te maken en de expertise van hun personeel.

Onderzoeksvraag

Het doel is om een crossplatform te vinden dat aan de eisen van performantie, useability en development effort voldoet. Deze door crossplatforms gecreëerde applicaties vergelijken we dan met elkaar en met applicaties die wel native zijn gebouwd. Uiteindelijk bepalen we dan welke crossplatforms geschikt bevonden zijn voor gebruik of eventueel voor gebruik als een prototype-platform.

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 runnen. Hiervoor bedachten ze een object-georiënteerd display systeem nodig te hebben. Hieruit resulteerde een basis voor de object-georiënteerde cross-platform GUI framework dat ze later zouden bouwen en de naam Qt, Q-toolkit, zou verkrijgen.

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.

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

2.1.2 Basis

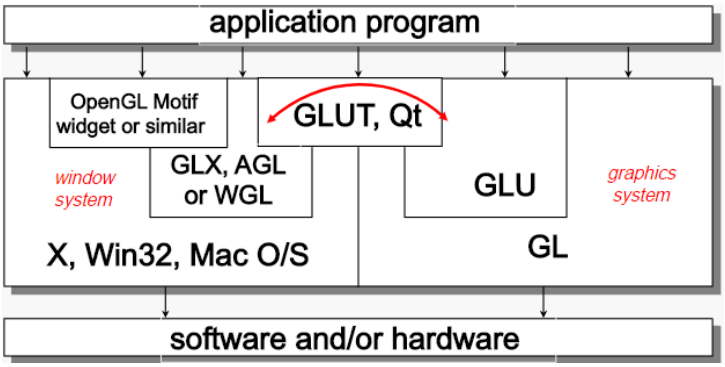
Qt is een cross-platform applicatie framework dat in C++ geschreven is en Android, iOS en WindowsPhone, … ondersteunt. Het doel van Qt was om zonder codewijzigingen op elk platform zonder performance-verlies te draaien en native te lijken. Voorbeelden van programma’s die in Qt Creator, het framework voor Qt, geschreven zijn: Skype, Google Earth, VirtualBox,...

Qt ondersteunt elke standaard C++ compiler, zoals GCC, ICC, MSVC en Clang.
Wij schreven onze applicatie 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.

2.1.3 Graphics architectuur

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. 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.

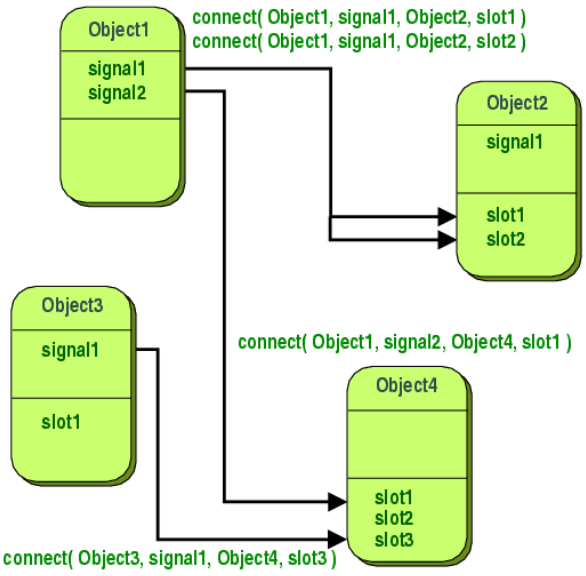


GLUT is een interface dat de communicatie tussen het window system en het graphic system verzorgt, deze gebruikt GLU en GL voor graphics en controleert operating en windowing systemen door voornamelijk GLX, AGL, … te gebruiken. Dit zijn OpenGL extensies voor specifieke windowing systemen.

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 Signals and slots

Qt behandelt events, zoals *quit(), onkeydown(),…* , niet rechtstreeks. Er wordt een alternatief gebruikt op zogenaamde callback functions, 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.

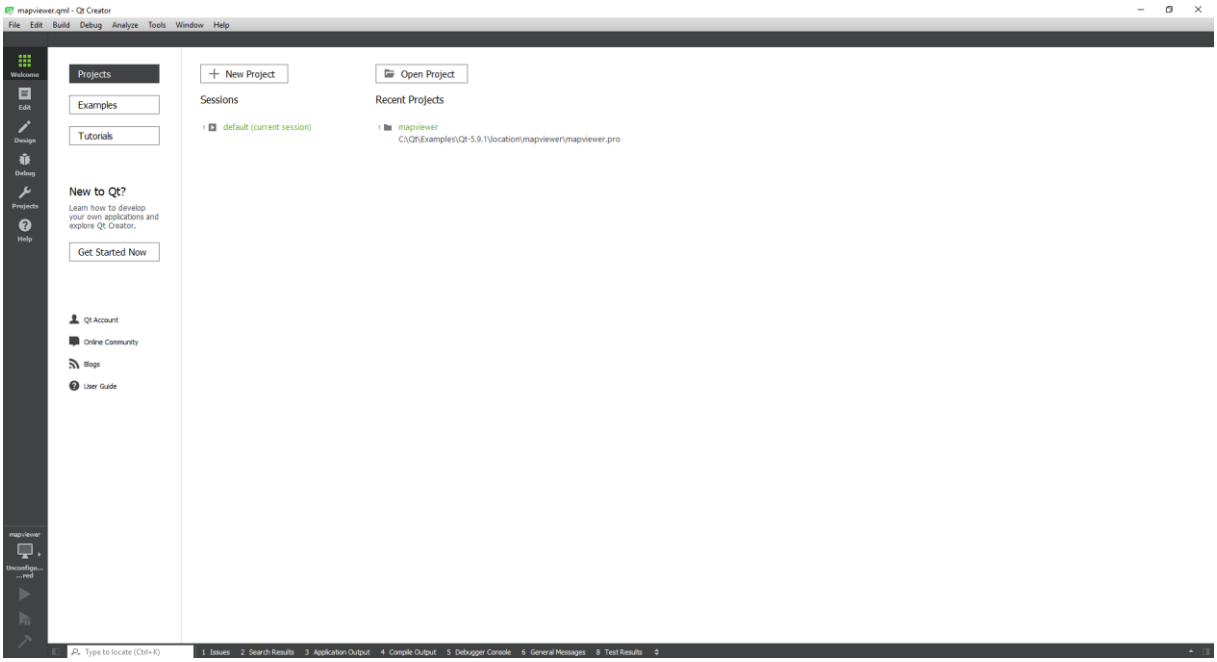


Een voorbeeld voor dit signal-slot gebeuren: een eenvoudige *quit()* functie:

```
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();
}
```

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 kan je 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.

<insert print screen van code>

2.1.6 Portfolio

Noemenswaardige applicaties die Qt of QML gebruiken:

- 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.

React Native startte als een hackathlon project in de zomer van 2013. 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.

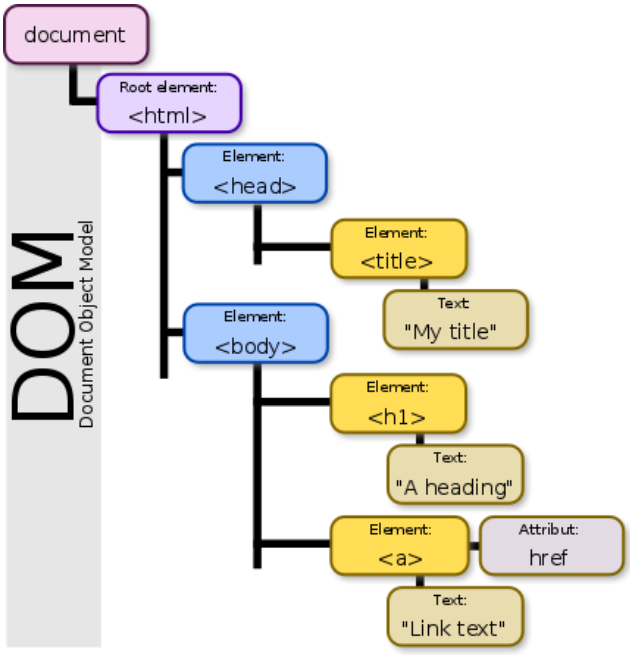
2.2.2 Basis

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.

2.2.3 Features

2.2.3.1 Virtual DOM

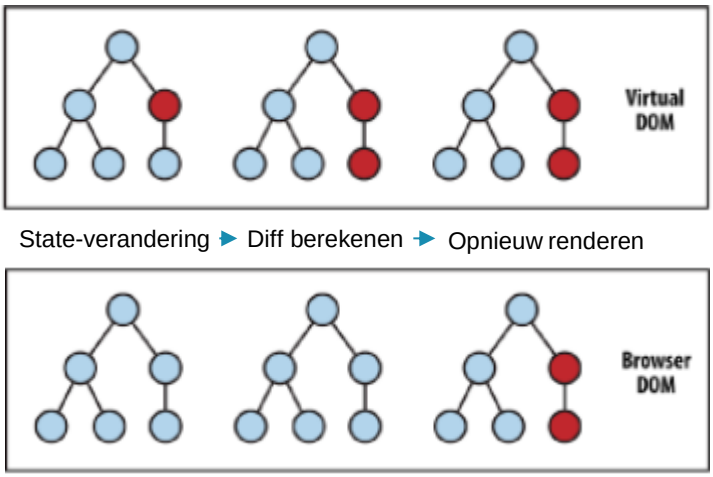
In een web applicatie is één van de meest belastende operaties het veranderen van de DOM, een applicatie programmerende interface dat een HTML, XHTML of XML bestand behandelt als een boomstructuur waarin elke node een object voorstelt als deel van het document.



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.

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.



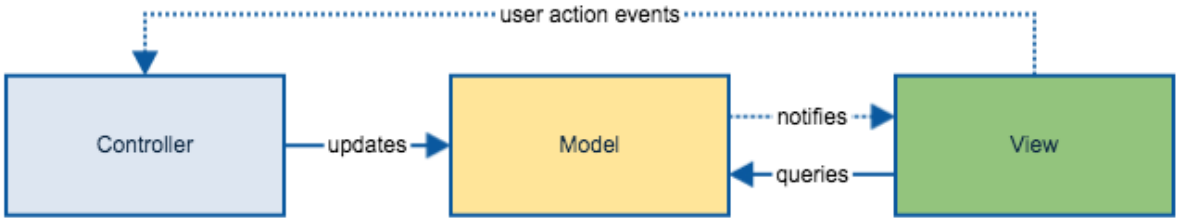
De Virtual DOM heeft dus zijn performance voordelen. Maar het potentieel is veel groter dan enkel performance voordelen. Wat als React een ander doel kon renderen dan de browser z'n DOM?

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 meestel een web-based view renderen.

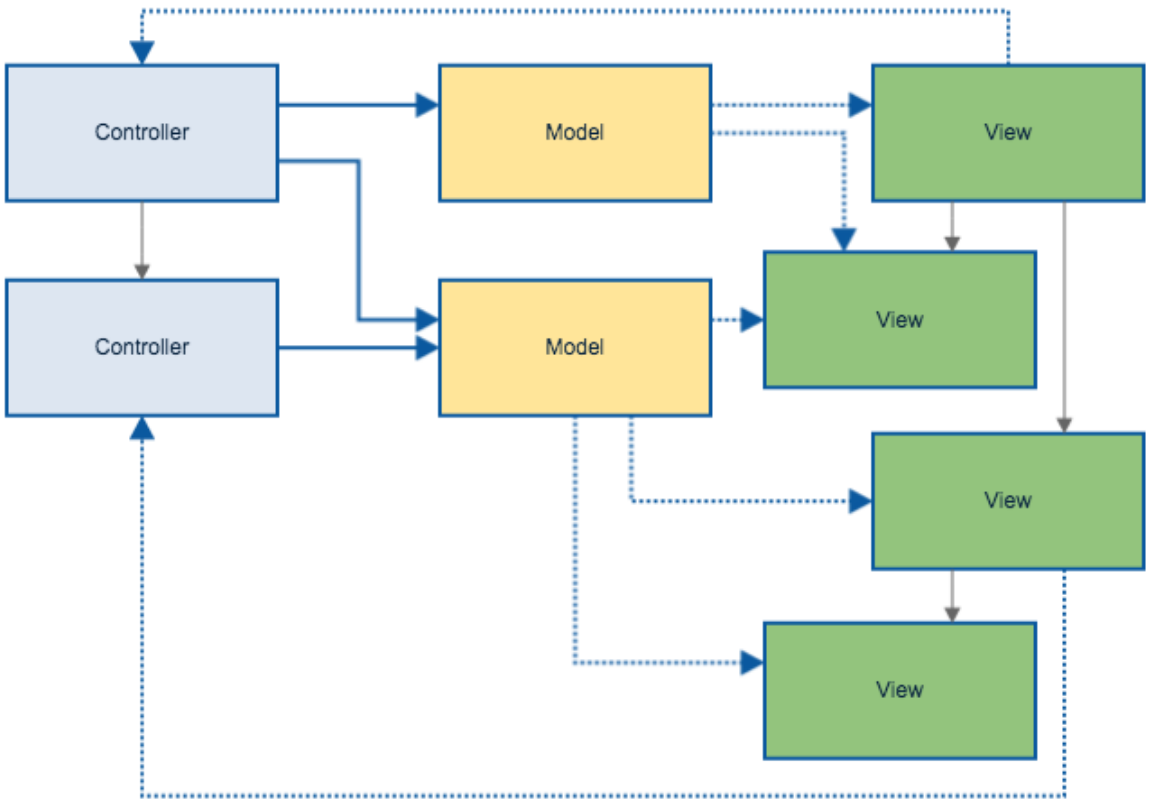
2.2.3.2 One-way data flow: MVC – Flux - Redux

React gebruikt 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 gebruikt. 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 getriggered worden.

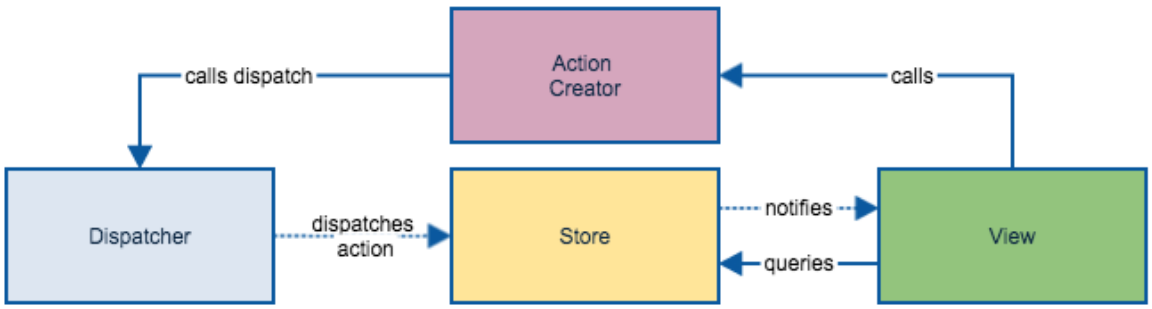


MVC

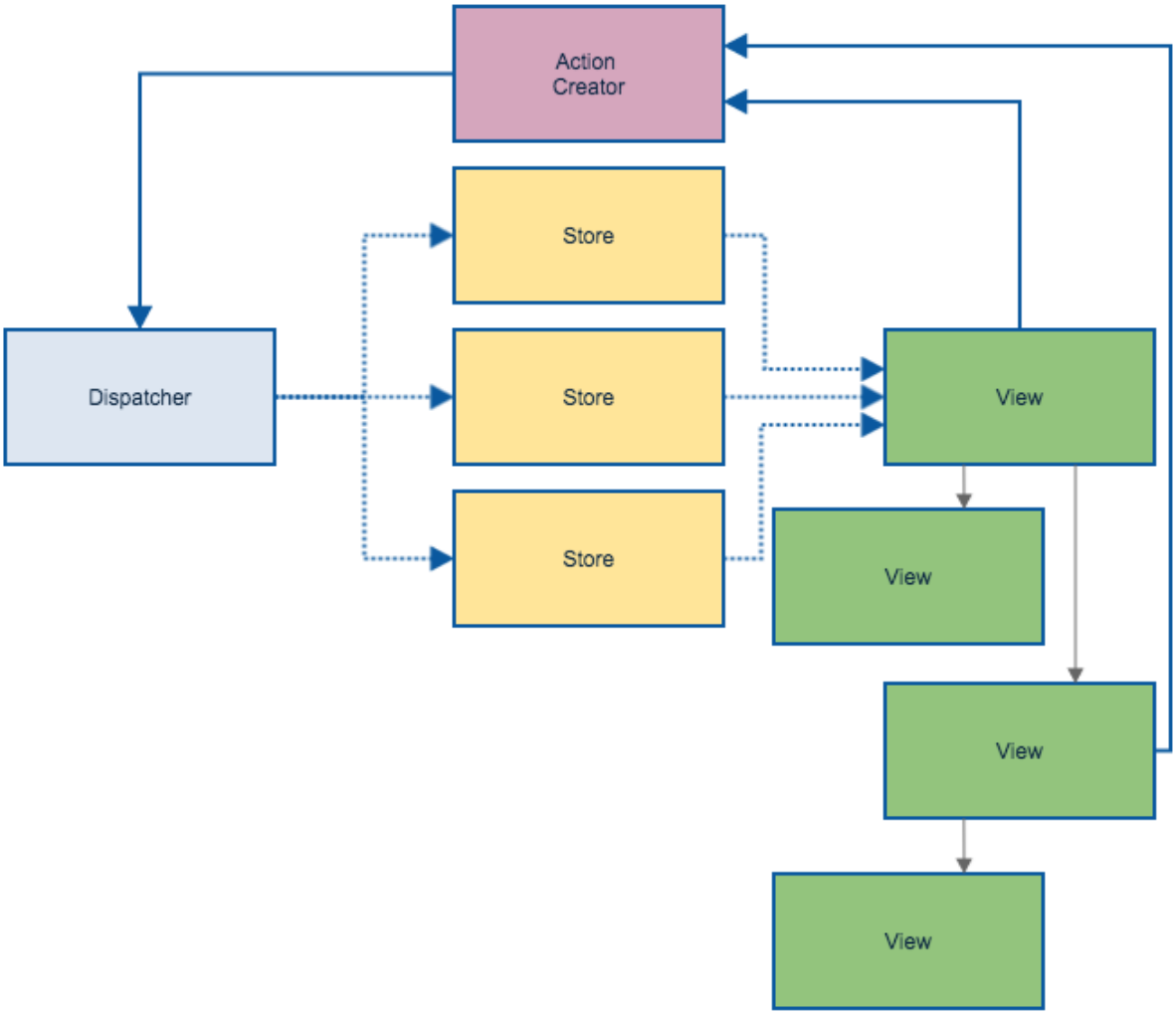


MVC: complexer

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 en 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 je 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.



Flux



Flux: complexer

Tegenwoordig wordt voor de React library veelal de Redux architectuur gebruikt, dit is een uitbreiding op Flux. Redux heeft als core dezelfde architectuur maar lost nog enkele complexiteit issues op dat Flux bevatte: de callback registration 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 te refreshen, verschillende states van de store op te slaan, … Door dit laatste is Redux dan ook zeer flexibel in termen van redundantie.

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 handig dat we via Android Studio bv. 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:

- 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 development platform gemaakt 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 gebruikt 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.

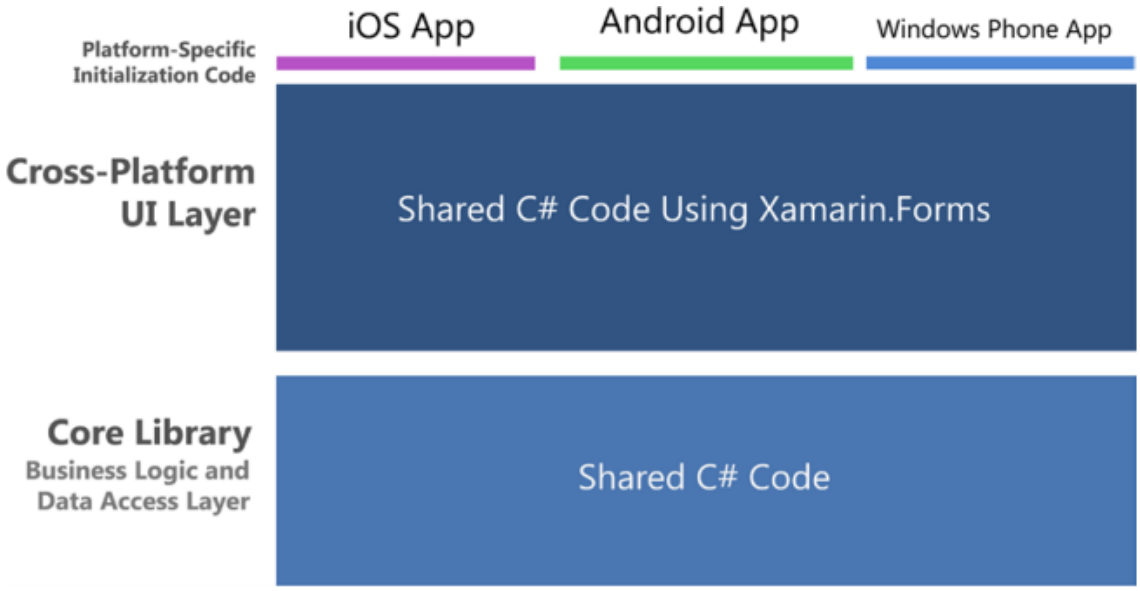
2.3.2 Basis

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 en iOS UI, graphics, … 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.

2.3.3 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. 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.



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

2.3.4 Gebruik

2.3.5 Portfolio

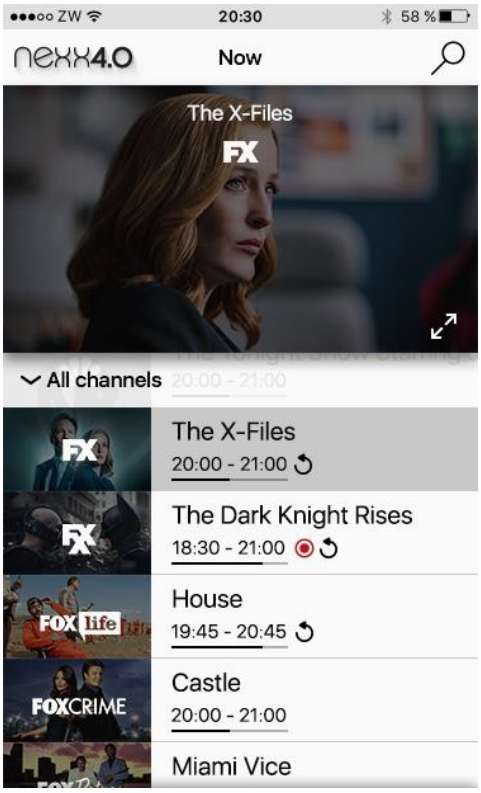
Lijst van applicaties developped met Xamarin voor bekende bedrijven:

- 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)

2.4 Apache Cordova

3 APPLICATIE

De applicatie die wij gaan bouwen via onze crossplatforms gaat een vrij basic tv-gids applicatie zijn. We houden het vrij basic omdat we graag een paar applicaties op verschillende platformen willen bouwen om een soortgelijke applicatie 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 we in andere talen bezig zijn. Het algemene beeld van onze applicatie gaat er als volgt uitzien:



Vanboven hebben we 3 componenten: 1 logo zonder functies, 1 zone die het uur/moment/dag aanduid, 1 zoek-optie met een zoekfunctie waarbij we bepaalde shows moeten kunnen zoeken.

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

Beneden laten we per kanaal zien wat er op de aangeduide moment bezig is, hoelang dit al bezig is, of dit opgenomen wordt, … En hierbij moeten we ook 1 kanaal kunnen aanduiden om te laten zien wat het programma is van dit kanaal op die dag.

Om een voorbeeld applicatie te bouwen is er een JSON bestand geüpload op een webserver met behulp van www.myjson.com zodat we deze kunnen aanspreken 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 we consequent kunnen zijn bij de verschillende plaformen. 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

Voor te beginnen aan een React Native applicatie, heb je 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 we dit geinstalleerd hebben 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 zo’n 1u in totaal. Je 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 werd soms toch een serieuze zoektocht waardoor ik veel tijd ben kwijtgeraakt aan kleine problemen.

Voor bepaalde componenten te gebruiken kon je 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 je infrastructuur wegviel waardoor we alles via Android Studio moesten opzetten en alles niet meer vlot verliep zoals bij het Expo-build proces.

Door deze problemen werd ook een simpel project tijdrovend om te creëren en heeft het uiteindelijk zo’n 100 uren geduurd om het te maken van begin tot einde.

4.2 Snelheid en geheugen

4.3 User experience

4.4 Conclusie

5 BESLUIT

Referenties

Hier komt de volledige referentielijst in de gekozen stijl APA of IEEE.

https://en.wikipedia.org/wiki/Qt_(software)

https://zappware.com/

https://nl.wikipedia.org/wiki/Qt-toolkit

https://en.wikipedia.org/wiki/React_(JavaScript_library)

Bijlagen

Bijlagen worden bij voorkeur enkel elektronisch ter beschikking gesteld. Indien essentieel kunnen in overleg met de promotor bijlagen in de scriptie opgenomen worden of als apart boekdeel voorzien worden.

Er wordt wel steeds een lijst met vermelding van alle bijlagen opgenomen in de scriptie. Bijlagen worden genummerd het een drukletter A, B, C, …

Bijlage A Detailtekeningen van de proefopstelling

Bijlage B Meetgegevens (op US

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