HO
GENT

Faculteit Bedrijf en Organisatie

Kotlin Multiplatform Mobile als alternatief voor native applicaties: een vergelijkende studie en proof-of-concept

Ziggy Moens

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Ludwig Stroobant
Co-promotor:
Kenneth Saey

Instelling: —

Academiejaar: 2020-2021

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Kotlin Multiplatform Mobile als alternatief voor native applicaties: een vergelijkende studie en proof-of-concept

Ziggy Moens

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Ludwig Stroobant
Co-promotor:
Kenneth Saey

Instelling: —

Academiejaar: 2020-2021

Tweede examenperiode

Woord vooraf

Samenvatting

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus.

Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

1. Inleiding

Cross-platform ontwikkeling is een gegeven dat in de laatste jaren aan populariteit wint. Dit vooral omdat er een paar voordelen aan verbonden zijn die voor bedrijven gespecialiseerd in software voor verschillende platformen zeer interessant kunnen zijn. Hierbij treden vooral de snellere ontwikkelingstijd en de lagere ontwikkelingskosten naar de voorgrond. Deze twee zaken spelen dus voor die bedrijven een grote rol binnen hun dag dagelijkse werking. Losstaand van deze voordelen is nog niet elk bedrijf te vinden voor cross-platform ontwikkeling. Het beeld dat deze cross-platform applicaties 'beter' zouden zijn dan de native oplossingen op de markt is nog niet overal aanvaard. Kotlin Multiplatform Mobile (KMM) is de nieuwe cross-platform software development kit van JetBrains zou hier verandering in kunnen brengen. KMM gaat voor een andere aanpak voor hun cross-platform applicaties en zal hier vooral gaan focussen op een gedeelde business logica en een platformafhankelijke en dus native user interface.

1.1 Probleemstelling

Veel bedrijven die zich specialiseren in applicatieontwikkeling voor mobile toestellen zoals Android en iOS toestellen hebben verschillende mogelijkheden om dit aan te pakken. Deze kunnen bijvoorbeeld kiezen voor een native applicatie of een cross-platform-applicatie. Anderzijds zijn er nog mogelijkheden zoals een progressive web application (PWA) geschreven in React native of Angular. De PWA's vallen echter buiten de scope van deze bachelorproef. Er zal dus vooral gefocust worden op bedrijven die momenteel native applicaties ontwikkelen en eventueel willen overschakelen naar cross-platform mochten zij hier grote voordelen uithalen.

1.2 Afbakening van het onderwerp

JetBrains heeft Kotlin Multiplatform (KM) uitgebracht met Kotlin Multiplatform Mobile (KMM) als specialisatie voor de mobile toestellen. KMM is vooral gericht op iPhone en Android in tegenstelling tot KM dat zich ook zal gaan toespitsen op Mac, Windows ... Voor deze bachelorproef zal er dus vooral gefocust worden op KMM aangezien de onderzoeksvraag zich toespitst op de mobile toestellen. KMM biedt ook dezelfde functies die KM zal aanbieden aan de ontwikkelaars aangezien KMM een aftakking is van KM. Voor besturingssystemen van mobile toestellen zal er vooral gekeken worden naar Android en iOS aangezien deze de meest gebruikte besturingssystemen zijn op de markt.

1.3 Onderzoeksvraag

De opzet van deze bachelorproef is na te gaan of native applicatieontwikkeling voor Android en iOS apparaten nog steeds de beste keuze is of KMM een sneller en efficiënter alternatief kan vormen binnen het huidige beeld van applicatieontwikkeling. Om dit te onderzoeken zijn er verschillende testcriteria opgesteld. Deze testcriteria zijn:

- het aantal lijnen code
- de kostprijs
- de ontwikkeltijd
- de compileersnelheid
- de voetafdruk
- de uitbreidingsmogelijkheden van de applicaties

Aan de hand hiervan kan besloten worden of de KMM-applicaties een beter alternatief kunnen vormen voor native applicaties. Daarnaast zal ook de vraag gesteld worden of het huidige beeld van de SWOT-analyse voor cross-platform applicaties ook toepasbaar is voor KMM-applicaties, indien niet zal gekeken worden hoe deze dient aangepast te worden.

1.4 Onderzoeksdoelstelling

Voor deze bachelor zal bekeken worden of Kotlin Multiplatofrm Mobile (KMM) een goed aternatief kan bieden voor native ontwikkeling. Hiervoor zullen de een vergelijkende studie uitvoeren tussen KMM en native applicaties. Voor deze studie zullen meerdere applicaties geschreven worden voor zowel Android als iOS. Eerst zullen er 2 native applicaties geschreven worden. De native iOS applicatie zal geschreven worden in Swift 5.3 of recenter met iOS 14 in gedachten en voor de user interface zal er gebruik gemaakt worden van UIKit. Aan de Android kant van het native verhaal zal er een applicatie geschreven worden met behulp van Kotlin 1.4.20 of een recentere versie. Voor de native Android user interface zal gebruik gemaakt worden van de standaard Android en AndroidX bibliotheek. Naast de native applicaties zal er nog een derde applicatie geschreven worden, namelijk

een Kotlin Multiplatform Mobile (KMM) applicatie. Deze applicatie zal werken op zowel Android als iOS toestellen. Hierbij zal voor het iOS-deel gebruik gemaakt worden van Swift en SwiftUI voor de user interface en voor het Android deel van Kotlin en Jetpack Compose.

Om deze manier van werken correct te beoordelen is het belangrijk dat zowel de native applicaties als KMM dezelfde functionaliteiten zullen aanbieden. Hierbij geldt ook voor de native applicaties voor Android en iOS onderling. De geschreven applicaties zouden dus als het ware niet van elkaar te mogen onderscheiden zijn. Hierbij dient echter rekening gehouden te worden met het feit dat de user interface wel enigszins kan verschillen.

Voor de applicaties zelf zal er gekeken worden naar de voorbeeld applicaties die JetBrains aanbiedt en voorstelt. Dit zijn onder andere de PeopleInSpace applicatie (O'Reilly, 2021) en de SpaceX lauches applicatie (Kotlin & JetBrains, 2020). Deze kunnen gebruikt worden om benchmarks op te testen en als inspiratiebron voor de native applicaties en andere KMM-applicaties.

Om de uiteindelijke de vergelijkende studie te maken tussen de native applicaties en de KMM-applicatie zal er gekeken worden naar verschillende factoren. Deze factoren zijn

- Aantal lijnen code
  - Het totaal aantal lijnen code van een applicatie zal geëvalueerd worden over het gehele project. In het geval van native applicaties zullen deze opgeteld worden met elkaar.
- Kostprijs
  - Hierbij wordt de geschatte kostprijs om een applicatie te laten ontwikkelen door een IT-bedrijf in kaart gebracht. Dit wordt berekend aan de hand van geschatte werkuren en een gemiddelde kostprijs per uur. Daarnaast wordt nagegaan of cross-platform een effect zal hebben op het systeem van vooraf bepaalde totaalprijzen indien bedrijven daarmee werken.
- Ontwikkeltijd
  - Deze tijd beschrijft het aantal werkuren dat een ontwikkelaar nodig heeft om een specifieke applicatie te schrijven. Hierbij kan onder andere gebruik gemaakt worden van platformen zoals GitHub om deze tijd te gaan meten of inschatten.
- Compileersnelheid
  - Dit is de snelheid waarmee de specifieke applicatie zal kunnen compileren en opstarten. Dit kan gemeten worden in de ontwikkelingssoftware voor de desbetreffende taal van de applicatie.
- Voetafdruk
  - Dit impliceert de omvang die de applicatie zal innemen op het platform waarvoor deze ontwikkeld is. Hiervoor kan de applicatie gebruikt worden die de ontwikkelingssoftware aanmaakt.
- Uitbreiding van de applicatie
  - Dit criterium kan geëvalueerd worden door vooraf bepaalde features van de applicatie weg te laten. Eens de applicatie klaar is voor productie kunnen deze

features terug toegevoegd worden. Om de uitbreidbaarheid van de applicatie te gaan staven kan gebruik gemaakt worden van voorgaande testcriteria.

Voor dit onderzoek zullen volgende hardware en software gebruikt worden:
Hardware:

- MacBook Pro 16 inch uit 2019
- iPhone 11 Pro Max uit 2019
- Huawei P9 lite uit 2016

Deze laatste twee toestellen zullen echter minder van belang zijn en zullen enkel ter sprake komen indien er testen gebeuren op fysieke toestellen. Voor het grotere deel van de testen zullen emulators gebruikt worden die ingebouwd zijn in de gekozen ontwikkelingssoftware.

Software:

- Xcode versie 12.3 of recenter
- Android studio versie 4.1.1 of recenter

Voor de KMM-applicaties zal er echter nog de Kotlin Multiplatform Mobile plugin geïnstalleerd moeten worden.

Deze software zal gebruikt worden op de MacBook Pro die hierboven reeds werd beschreven. Voor dit onderzoek geldt de beperking dat er enkel gebruik gemaakt kan worden van toestellen die MacOS gebruiken als besturingssysteem, aangezien dit een vereiste is voor de ontwikkeling van iOS-applicaties.

KMM is een zeer recente technologie die volop in ontwikkeling is. Daardoor wordt verwacht dat de voor deze studie resultaten nog niet het volle potentieel zullen aantonen. Dit wil echter niet zeggen dat de technologie geen meerwaarde kan bieden voor bedrijven die zich momenteel bezighouden met native applicatieontwikkeling voor zowel Android als iOS. Enkele componenten van de SDK staan momenteel nog niet op punt, dit zal waarschijnlijk nog verbeteren naar de toekomst toe. Het onderzoek kan, gezien de prematuriteit van KMM, een tijdelijke referentie bieden omtrent KMM en de mogelijkheid dit als een sneller en efficiënter alternatief te gebruiken voor native applicaties. Dit is vooral interessant voor bedrijven die zich momenteel vooral toespitsen op native ontwikkeling en eventueel naar de toekomst toe de overstap willen maken naar KMM.

1.5 Opzet van deze bachelorproef

Het vervolg van de bachelorproef wordt als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.

In Hoofdstuk 4, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2. Stand van zaken

Binnen de stand van zaken zal er verder ingegaan worden op de theoretische kant van deze bachelorproef. Hierbij zullen volgende thema's aanbod komen: Kotlin, de mogelijke ontwikkelingsvormen voor mobiele applicaties, Kotlin Multiplatform en Kotlin Multiplatform Mobile, de vergelijking met andere alternatieven en de gekozen testcriteria voor deze bachelorproef. De stand van zaken bouwt verder op de informatie uit de inleiding en wordt versterkt aan de hand van de literatuurstudie.

2.1 Kotlin

In het eerste deel van deze stand van zaken zal Kotlin besproken worden. Kotlin vormt de basis van Kotlin Multiplatform en het is dus interessant om dit verder te bekijken. Eerst zal kort de geschiedenis rond Kotlin bespreken en daarna kan de technische kant van de programmeertaal besproken worden.

Kotlin is binnen de informaticawereld nog een vrij recente programmeertaal, de programmeertaal is uitgebracht in juli 2011 door JetBrains.(Jemerov, 2011) JetBrains is een software ontwikkelingsbedrijf dat afkomstig is uit Tsjechisch en is gekend voor hun applicaties zoals IntelliJ IDE, PyCharm... Het bedrijf heeft ondertussen al meer dan 30 producten en meer dan tien miljoen gebruikers.(JetBrains, 2021) In juli 2011 was Kotlin echter al meer dan een jaar in ontwikkeling en is tot de dag van vandaag nog steeds in ontwikkeling. JetBrains heeft dan ook beslist om van Kotlin een open-source project te maken. Dit zorgt ervoor dat iedereen de broncode kan bekijken en eventueel bewerken. Hierdoor hebben ze een zeer grote en actieve community gecreëerd rond Kotlin. In 2017 werd Kotlin door Google als de hoofdtaal gekozen voor de ontwikkeling van Android applicaties.(Shafirov,

2017) Dit alles heeft ertoe geleid dat Kotlin de snelst groeiende programmeertaal was op Github in 2018, zoals te zien was in GitHubs The State of the Octoverse 2018.(GitHub, 2018)

Na de korte geschiedenis rond Kotlin kan er nu overgegaan worden naar de technische zijde van Kotlin. Kotlin is een programmeertaal met een aantal typerende factoren zoals het feit dat Kotlin een algemene programmeertaal is. Daarnaast is Kotlin een statische programmeertaal met type-interferentie. Ook is deze programmeertaal gecreëerd met cross-platform in het achterhoofd.(Oliveira e.a., 2020) Zoals hiervoor vermeld is Kotlin een algemene programmeertaal, hiermee wordt bedoeld dat Kotlin een programmeertaal is die ontwikkeld is voor allerlei verschillende soorten software en dat de programmeertaal gebruikt kan worden binnen verschillende situaties.(Skeen & Greenhalgh, 2018) Daarnaast is gezegd dat Kotlin een statische programmeertaal. Dit gegeven wijst op het feit dat de programmeertaal de types van de objecten zal controleren tijdens het compileren van de code en niet tijdens het uitvoeren. Andere voorbeelden van statische talen met type-interferentie zijn Java en C. Talen die geen statische typering gebruiken zijn dynamische typerende talen zoals Perl, PHP... Daarnaast gebruikt Kotlin type-interferentie daardoor zal Kotlin zelf onderscheid kunnen maken tussen de datatypes van bepaalde expressies.(Meijer & Drayton, 2004) Een laatste punt is het cross-platform gegeven van de programmeertaal, hiermee wordt bedoeld dat de software kan bestaan en werken op verschillende versies. Deze versies kunnen ook draaien op verschillende platformen en is dus niet noodzakelijk gebonden aan één specifiek platform zoals bijvoorbeeld Android.(Bishop & Horspool, 2006)

2.2 Platformen en hun ontwikkelingsvormen

Nadat de basis rond Kotlin is besproken kan bekeken worden wat de mogelijkheden binnen het ontwikkelen van mobiele applicaties. Allereerst zal besproken worden wat gezien word als een platform en daarna wat de ontwikkelingsmogelijkheden zijn voor dat specifieke platform of voor meerdere platformen tegelijk.

2.2.1 Platform

Allereerst zal er bekeken worden wat de term platform inhoudt, daarvoor wordt gebruikt gemaakt van het artikel van Bishop en Horspool (2006). Allereerst moet vermeld worden dat de term platform nog niet strikt gedefinieerd is, echter kunnen er wel enkele zaken gelinkt worden aan de term. Een platform zal meestal een bepaalde programmeertaal, bepaald besturingssysteem of bepaalde hardware beschrijven. Hierbij is belangrijk te vermelden dat het niet enkel over een van deze zaken kan gaan maar ook over een combinatie van meerdere factoren. Hiervan een voorbeeld is bijvoorbeeld een platform dat beschreven word door een bepaald besturingssysteem in combinatie met specifieke hardware. Enkele voorbeelden van platformen in de praktijk:

- Programmeertaal als platform: Hierbij zal een specifieke programmeertaal en even-

tueel bijhorende libraries dienen als het platform waarvoor er bepaalde software voor gemaakt zal worden. Enkele voorbeelden hiervan zijn Java SE 16 [1], AdoptOpenJDK 11[2]... Hiervan zijn nog vele voorbeelden op te sommen, ook de voorgaande versies van deze software worden als andere platformen gezien.

- Besturingssysteem als platform: Hierbij zal een bepaald besturingssysteem gebruikt worden als platform. Hierbij kunnen al 3 grote platformen worden opgehaald namelijk Windows, MacOs en Linux. Echter zijn deze te globaal en zullen hierbij bijvoorbeeld Windows 10[3], MacOs Big Sur[4] en Ubuntu 20.04[5] gekozen worden als een platform voor ontwikkeling. Afhankelijk van de gekozen versie van het besturingssysteem als platform zal de ontwikkelde software bepaalde apparaten al dan niet ondersteunen.
- Hardware als platform: Hierbij zal een specifiek deel van de hardware binnenin bepaalde apparaten fungeren als platform. Dit kan echter zeer ruim bekeken worden. Een bepaalde processor of CPU (central processing unit), een grafische processor of GPU (graphics processing unit)... Dit zijn allemaal voorbeelden van hardware die gekozen kunnen worden als platform. Enkele praktijkvoorbeelden zijn ondere andere een bepaalde CPU van Intel[6] of AMD[7] zoals de Intel Core i9 11900K[8] of de AMD Ryzen Threadripper 3970X[9].
- Combinatie van programmeertaal en/of besturingssysteem en/of hardware: Een laatste mogelijkheid om een platform te beschrijven is door een combinatie van bovenstaande punten te gebruiken. Zo kan er specifiek op bepaalde platformen gericht worden die , dit kan voor bepaalde situaties zeer handig zijn. Een voorbeeld hiervan is een computer die uitgerust met een MacOs Big Sur besturingssysteem en een Intel processor. Indien bepaalde niche software enkel door zo een type toestel gebruikt zal worden kan het handig zijn om het platform zo gedetailleerd te beschrijven.

2.2.2 Ontwikkelingsvormen

Native ontwikkeling

Cross-platform ontwikkeling

Eens de term platform duidelijker is, kan er makkelijker omschreven worden wat precies een cross-platform applicatie is en wat een native applicatie is. Een cross-platform applicatie is zoals de naam het al zegt een applicatie die op meerdere platformen zal werken. De software zal dus licht verschillende versies hebben die op hun beurt op allerlei verschil-

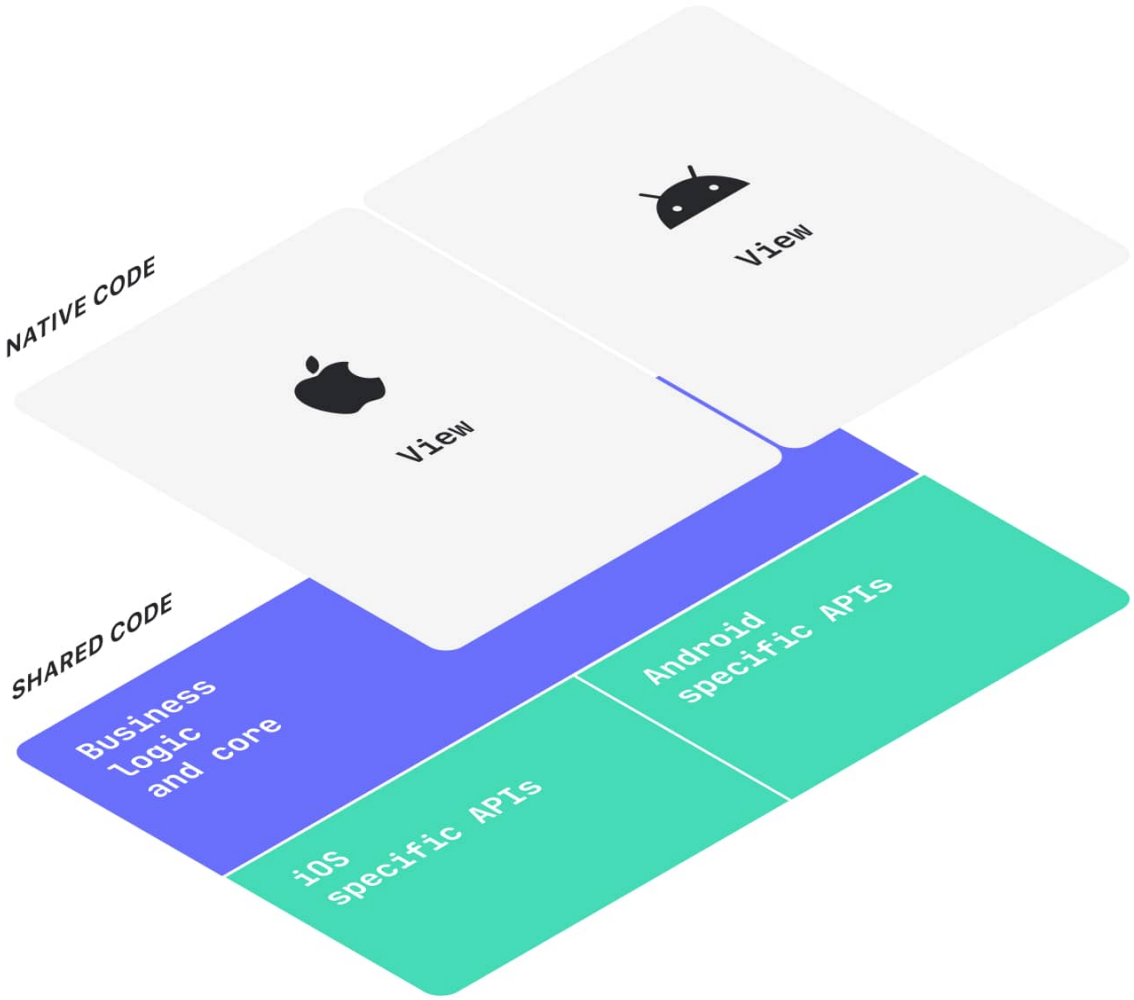
[1] https://www.oracle.com/java/technologies/javase-downloads.html
[2] https://adoptopenjdk.net/index.html
[3] https://www.microsoft.com/nl-be/windows
[4] https://www.apple.com/benl/macos/big-sur/
[5] https://ubuntu.com
[6] https://www.intel.com
[7] https://www.amd.com/
[8] https://www.intel.com/content/www/us/en/products/sku/212325/intel-core-i911900k-processor-16m-cache-up-to-5-30-ghz/specifications.html
[9] https://www.amd.com/en/products/cpu/amd-ryzen-threadripper-3970x

lende platformen zullen werken. Native applicaties daarentegen zullen zich toespitsen op één bepaald platform.
ă
Verder wordt er SWOT-analyse bekeken voor cross-platform applicaties. Deze sterkte-zwakteanalyse zal bekijken wat de sterktes en zwaktes zijn maar ook een beeld schetsen van de kansen en bedreigingen. Hiermee kunnen we een beeld schetsen wat de huidige positie is van cross-platform applicaties binnen de huidige markt. Uit onderzoek van Tommi Nivanaho naar cross-platform applicaties met React Native zijn een aantal factoren naar voor gekomen.(Nivanaho, 2019) Hier moet wel rekening gehouden worden met het feit dat sommige factoren niet toepasbaar zijn op alle cross-platform SDKs en toolkits. Hieronder een overzicht van enkele punten uit de studie toepasbaar zijn op cross-platform applicaties in het algemeen.

- Sterktes of Strengths
  - Sneller te ontwikkelen
  - Kostenbesparend
  - De applicatie ondersteunt meerdere platformen tegelijkertijd
- Zwaktes of Weaknesses
  - Nog steeds nood aan native code per platform
  - Upgrades kunnen omslachtiger zijn
- Kansen of Opportunities
  - Snelle ontwikkeling voor meerdere platformen tegelijkertijd
  - Native applicaties kunnen relatief vlot omgevormd worden naar cross-platform applicaties
- Bedreigingen of Threats
  - Updates aan het gekozen cross-platform systeem kunnen de reeds geschreven applicaties onbruikbaar maken

2.3 Multiplatform

Kotlin Multiplatform is de nieuwe cross platform software development kit (SDK) van JetBrains. Ook werd een SDK uitgebracht specifiek gericht op de mobiele toestellen namelijk Kotlin Multiplatform Mobile (KMM). Kotlin Multiplatform werd voor het eerst verwerkt in Kotlin 1.2 in november 2017 als experimentele functie.(Jemerov, 2017) Ondertussen heeft de ontwikkeling van KMM niet stil gestaan en is in augustus 2020 de alpha versie van deze SDK uitgebracht voor het grote publiek.(Petrova, 2020) In november 2020 werd de 0.2.0 versie uitgebracht, deze is verwerkt in Kotlin 1.4.20.(JetBrains, 2020) Enkele delen van deze SDK en bijhorende componenten zijn nog steeds in de experimentele fase van ontwikkeling, maar JetBrains geeft aan dat het nu al een ideaal moment is om deze software te testen.(Petrova, 2020) De community achter KMM en het open source verhaal spelen hier een grote rol en zullen ook zorgen voor een snellere ontwikkeling van deze software. Bedrijven kunnen dus momenteel al experimenteel aan de slag met deze nieuwe software en voorbeelden van enkele bedrijven die de stap al hebben gezet naar KMM zijn onder andere Netflix, VMware en Autodesk.(Kotlin, 2020)



Figuur 2.1: Grafische voorstelling Kotlin Multiplatform Mobile (Kotlin & JetBrains, g.d.)

2.3.1 Kotlin Multiplatform

2.3.2 Kotlin Multiplatform Mobile

2.4 Kotlin Multiplatform versus alternatieven

Kotlin Multiplatform (KM) zal het concept van cross-platform anders aanpakken dan andere alternatieven op de markt vandaag. Hierbij zit het verschil vooral in welke code van de applicatie zal gedeeld worden tussen de verschillende platformen. In de Kotlin documentatie staat beschreven hoe bepaalde delen van de code correct kunnen gedeeld worden tussen verschillende platformen. KM zal anders dan alternatieven de business logica delen tussen de verschillende platformen.(Kotlin & JetBrains, 2021) Hierbij wordt ook vermeld dat men enkel de business logica moet delen die gebruikt kan worden op alle platformen.

Figuur A.1 geeft een goed algemeen beeld van de structuur van Kotlin Multiplatform Mobile. De shared code in de figuur A.1 verwijst hier naar Common Kotlin of CommonMain,

dit is het gedeelde binnenin KM dat de gedeelde logica zal bevatten. Daarnaast zal er nog per platform een aparte main zijn die de code zal implementeren. Hier in de figuur A.1 zal het project ook nog een iosMain en een kotlinMain, deze kunnen later ook nog uitgebreid worden met bijvoorbeeld een macosX64Main. De specifieke situatie in figuur A.1 is een applicatie waar Kotlin Multiplatform Mobile gebruikt wordt. Deze zal zich speciaal richten op applicaties voor iOS en Android. Voor de communicatie tussen het common deel en de platform specifieke delen zal Kotlin gebruik maken van het expected/actual systeem. Hierbij worden binnenin de CommonMain elementen gedeclareerd met expect en de platform specifieke delen zullen dezelfde elementen declareren met actual. Deze structuur kan gebruikt worden voor functies, klassen, interfaces, enumeraties, properties en annotaties.
ă
Nu er een beter beeld is geschetst van hoe KM, werkt zal er gekeken worden naar een mogelijk alternatief met een andere aanpak voor de gedeelde code, bijvoorbeeld Flutter. Flutter is een user interface toolkit ontwikkeld door Google en zit momenteel aan versie 1.22 sinds oktober 2020.(Sells, 2020) De toolkit richt zich op mobile, web en desktop applicaties en zal de code native compileren. Zoals reeds beschreven is Flutter een user interface toolkit en zal dus de user interface delen over de verschillende platformen. Hiervoor zal Flutter widgets gebruiken die geïnspireerd zijn door React.(Flutter, g.d.) Dit impliceert dus dat hierbij de business logica zal moeten verwerkt worden per platform.

De developer kan dus kiezen om de user interface te delen onder de platformen en een toolkit te gebruiken zoals Flutter. Anderzijds kan er gekozen worden voor het delen van de business logica onder de platformen en dan kan KM gebruikt worden. Het delen van de user interface zal als voordeel hebben dat alle applicaties op de verschillende platformen dezelfde look en feel zullen hebben, echter is een nadeel dat de business logica per platform verwerkt zal moeten worden. Aan de kant van de gedeelde business logica is er het voordeel dat deze gedeeld is dus dat alle applicaties dezelfde logica hebben en implementeren, alsook kan er voor de user interface gebruik gemaakt worden van de platform specifieke frameworks. Er is echter ook een nadeel, hierbij kan gedacht worden aan het feit dat de user interface niet overal exact dezelfde zal zijn.

2.5 Testcriteria

Tijdens deze bachelorproef zullen van verschillende applicaties testcriteria geëvalueerd worden. Deze zullen ons vertellen in welke mate de cross-platform applicaties beter zijn dan de native. Volgende testcriteria werden gekozen voor deze vergelijkende studie.

- Aantal lijnen code
  - Het totaal aantal lijnen code van een applicatie zal geëvalueerd worden over het gehele project. In het geval van native applicaties zullen deze opgeteld worden met elkaar.
- Kostprijs
  - Hierbij wordt de geschatte kostprijs om een applicatie te laten ontwikkelen door een IT-bedrijf in kaart gebracht. Dit wordt berekend aan de hand van

geschatte werkuren en een gemiddelde kostprijs per uur. Daarnaast wordt nagegaan of cross-platform een effect zal hebben op het systeem van vooraf bepaalde totaalprijzen indien bedrijven daarmee werken.

- Ontwikkeltijd
  - Deze tijd beschrijft het aantal werkuren dat een ontwikkelaar nodig heeft om een specifieke applicatie te schrijven. Hierbij kan onder andere gebruik gemaakt worden van platformen zoals GitHub om deze tijd te gaan meten of inschatten.
- Compileersnelheid
  - Dit is de snelheid waarmee de specifieke applicatie zal kunnen compileren en opstarten. Dit kan gemeten worden in de ontwikkelingssoftware voor de desbetreffende programmeertaal van de applicatie.
- Voetafdruk
  - Dit impliceert de omvang die de applicatie zal innemen op het platform waarvoor deze ontwikkeld is. Hiervoor kan de applicatie gebruikt worden die de ontwikkelingssoftware aanmaakt.
- Uitbreiding van de applicatie
  - Dit criterium kan geëvalueerd worden door vooraf bepaalde features van de applicatie weg te laten. Eens de applicatie klaar is voor productie kunnen deze features terug toegevoegd worden. Om de uitbreidbaarheid van de applicatie te gaan staven kan gebruik gemaakt worden van voorgaande testcriteria.

3. Methodologie

Etiam pede massa, dapibus vitae, rhoncus in, placerat posuere, odio. Vestibulum luctus commodo lacus. Morbi lacus dui, tempor sed, euismod eget, condimentum at, tortor. Phasellus aliquet odio ac lacus tempor faucibus. Praesent sed sem. Praesent iaculis. Cras rhoncus tellus sed justo ullamcorper sagittis. Donec quis orci. Sed ut tortor quis tellus euismod tincidunt. Suspendisse congue nisl eu elit. Aliquam tortor diam, tempus id, tristique eget, sodales vel, nulla. Praesent tellus mi, condimentum sed, viverra at, consectetuer quis, lectus. In auctor vehicula orci. Sed pede sapien, euismod in, suscipit in, pharetra placerat, metus. Vivamus commodo dui non odio. Donec et felis.

Etiam suscipit aliquam arcu. Aliquam sit amet est ac purus bibendum congue. Sed in eros. Morbi non orci. Pellentesque mattis lacinia elit. Fusce molestie velit in ligula. Nullam et orci vitae nibh vulputate auctor. Aliquam eget purus. Nulla auctor wisi sed ipsum. Morbi porttitor tellus ac enim. Fusce ornare. Proin ipsum enim, tincidunt in, ornare venenatis, molestie a, augue. Donec vel pede in lacus sagittis porta. Sed hendrerit ipsum quis nisl. Suspendisse quis massa ac nibh pretium cursus. Sed sodales. Nam eu neque quis pede dignissim ornare. Maecenas eu purus ac urna tincidunt congue.

Donec et nisl id sapien blandit mattis. Aenean dictum odio sit amet risus. Morbi purus. Nulla a est sit amet purus venenatis iaculis. Vivamus viverra purus vel magna. Donec in justo sed odio malesuada dapibus. Nunc ultrices aliquam nunc. Vivamus facilisis pellentesque velit. Nulla nunc velit, vulputate dapibus, vulputate id, mattis ac, justo. Nam mattis elit dapibus purus. Quisque enim risus, congue non, elementum ut, mattis quis, sem. Quisque elit.

Maecenas non massa. Vestibulum pharetra nulla at lorem. Duis quis quam id lacus dapibus interdum. Nulla lorem. Donec ut ante quis dolor bibendum condimentum. Etiam

egestas tortor vitae lacus. Praesent cursus. Mauris bibendum pede at elit. Morbi et felis a lectus interdum facilisis. Sed suscipit gravida turpis. Nulla at lectus. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Praesent nonummy luctus nibh. Proin turpis nunc, congue eu, egestas ut, fringilla at, tellus. In hac habitasse platea dictumst.

Vivamus eu tellus sed tellus consequat suscipit. Nam orci orci, malesuada id, gravida nec, ultricies vitae, erat. Donec risus turpis, luctus sit amet, interdum quis, porta sed, ipsum. Suspendisse condimentum, tortor at egestas posuere, neque metus tempor orci, et tincidunt urna nunc a purus. Sed facilisis blandit tellus. Nunc risus sem, suscipit nec, eleifend quis, cursus quis, libero. Curabitur et dolor. Sed vitae sem. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Maecenas ante. Duis ullamcorper enim. Donec tristique enim eu leo. Nullam molestie elit eu dolor. Nullam bibendum, turpis vitae tristique gravida, quam sapien tempor lectus, quis pretium tellus purus ac quam. Nulla facilisi.

4. Conclusie

Curabitur nunc magna, posuere eget, venenatis eu, vehicula ac, velit. Aenean ornare, massa a accumsan pulvinar, quam lorem laoreet purus, eu sodales magna risus molestie lorem. Nunc erat velit, hendrerit quis, malesuada ut, aliquam vitae, wisi. Sed posuere. Suspendisse ipsum arcu, scelerisque nec, aliquam eu, molestie tincidunt, justo. Phasellus iaculis. Sed posuere lorem non ipsum. Pellentesque dapibus. Suspendisse quam libero, laoreet a, tincidunt eget, consequat at, est. Nullam ut lectus non enim consequat facilisis. Mauris leo. Quisque pede ligula, auctor vel, pellentesque vel, posuere id, turpis. Cras ipsum sem, cursus et, facilisis ut, tempus euismod, quam. Suspendisse tristique dolor eu orci. Mauris mattis. Aenean semper. Vivamus tortor magna, facilisis id, varius mattis, hendrerit in, justo. Integer purus.

Vivamus adipiscing. Curabitur imperdiet tempus turpis. Vivamus sapien dolor, congue venenatis, euismod eget, porta rhoncus, magna. Proin condimentum pretium enim. Fusce fringilla, libero et venenatis facilisis, eros enim cursus arcu, vitae facilisis odio augue vitae orci. Aliquam varius nibh ut odio. Sed condimentum condimentum nunc. Pellentesque eget massa. Pellentesque quis mauris. Donec ut ligula ac pede pulvinar lobortis. Pellentesque euismod. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent elit. Ut laoreet ornare est. Phasellus gravida vulputate nulla. Donec sit amet arcu ut sem tempor malesuada. Praesent hendrerit augue in urna. Proin enim ante, ornare vel, consequat ut, blandit in, justo. Donec felis elit, dignissim sed, sagittis ut, ullamcorper a, nulla. Aenean pharetra vulputate odio.

Quisque enim. Proin velit neque, tristique eu, eleifend eget, vestibulum nec, lacus. Vivamus odio. Duis odio urna, vehicula in, elementum aliquam, aliquet laoreet, tellus. Sed velit. Sed vel mi ac elit aliquet interdum. Etiam sapien neque, convallis et, aliquet vel, auctor non, arcu. Aliquam suscipit aliquam lectus. Proin tincidunt magna sed wisi. Integer

blandit lacus ut lorem. Sed luctus justo sed enim.

Morbi malesuada hendrerit dui. Nunc mauris leo, dapibus sit amet, vestibulum et, commodo id, est. Pellentesque purus. Pellentesque tristique, nunc ac pulvinar adipiscing, justo eros consequat lectus, sit amet posuere lectus neque vel augue. Cras consectetuer libero ac eros. Ut eget massa. Fusce sit amet enim eleifend sem dictum auctor. In eget risus luctus wisi convallis pulvinar. Vivamus sapien risus, tempor in, viverra in, aliquet pellentesque, eros. Aliquam euismod libero a sem.

Nunc velit augue, scelerisque dignissim, lobortis et, aliquam in, risus. In eu eros. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Curabitur vulputate elit viverra augue. Mauris fringilla, tortor sit amet malesuada mollis, sapien mi dapibus odio, ac imperdiet ligula enim eget nisl. Quisque vitae pede a pede aliquet suscipit. Phasellus tellus pede, viverra vestibulum, gravida id, laoreet in, justo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer commodo luctus lectus. Mauris justo. Duis varius eros. Sed quam. Cras lacus eros, rutrum eget, varius quis, convallis iaculis, velit. Mauris imperdiet, metus at tristique venenatis, purus neque pellentesque mauris, a ultrices elit lacus nec tortor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent malesuada. Nam lacus lectus, auctor sit amet, malesuada vel, elementum eget, metus. Duis neque pede, facilisis eget, egestas elementum, nonummy id, neque.

A. Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

A.1 Introductie

De keuze voor cross platform applicaties wordt de dag van vandaag steeds aantrekkelijker. Dit kan onder andere verklaard worden door de financiële voordelen ten opzichte van native applicaties, meer specifiek op vlak van de impact op het ontwikkelingsbudget van bedrijven. Cross platform applicaties worden al heel lang gepromoot als zijnde 'beter' dan native applicaties. Deze cross platform applicaties zijn sneller te ontwikkelen en vereisen minder lijnen code. De slogan Write once, run anywhere van Sun Microsystems (Oracle, g.d.) wordt vaak gebruikt om het grote voordeel van cross platform applicaties aan te duiden, dit duidt op het idee dat grote delen van de code kunnen gebruikt worden voor verschillende besturingssystemen. De duurdere native applicaties hebben echter nog steeds een aantal voordelen ten opzichte van de cross platform applicaties, aangezien deze specifiek geschreven zijn voor het gekozen besturingssysteem zoals bijvoorbeeld voor Android of iOS. De meeste reeds bestaande cross-platform frameworks, zoals Flutter, delen de user interface van de applicatie over de verschillende besturingssystemen. Dit zorgt voor een applicatie die voor geen enkel van de gekozen besturingssysteem perfect zal zijn, aangezien hierbij dan geen gebruik kan gemaakt worden van de specifieke frameworks voor dat specifieke besturingssysteem. Kotlin Multiplatform Mobile (KMM) gooit het concept van cross platform over een andere boeg en werkt met een gedeelde businesslogica in plaats van een gedeelde user interface. De applicaties worden dus elk afgewerkt met een native user interface. De opzet van deze bachelorproef is na te gaan of native applicatieontwik-

keling voor Android en iOS apparaten nog steeds de beste keuze is of KMM een sneller en efficiënter alternatief kan vormen binnen het huidige beeld van applicatieontwikkeling. Om dit te onderzoeken zijn er bepaalde testcriteria opgesteld. Deze testcriteria zijn: het aantal lijnen code, de kostprijs, de ontwikkeltijd, de compileersnelheid, de voetafdruk en de uitbreidingsmogelijkheden van de applicaties. Aan de hand hiervan kan besloten worden of de KMM-applicaties een beter alternatief kunnen vormen voor native applicaties. Daarnaast zal ook de vraag gesteld worden of het huidige beeld van de SWOT-analyse voor cross-platform applicaties ook toepasbaar is voor KMM-applicaties, indien niet zal gekeken worden hoe deze dient aangepast te worden.

A.2 State-of-the-art

A.2.1 Kotlin

In juli 2011 kwam JetBrains, de maker van onder andere IntelliJ IDEA, naar buiten met een nieuw project genaamd Kotlin.(Jemerov, 2011) Kotlin is een algemene, cross-platform en statische taal met type-inferentie. (Oliveira e.a., 2020) Hierbij verwijst de term algemene naar het feit dat Kotlin is ontwikkeld is voor allerlei verschillende types van software en in verschillende situaties kan gebruikt worden. Cross-platform gaat over het gegeven dat software kan bestaan in verschillende versies op meer dan één platform. Ook is er vernoemd dat Kotlin statische typering gebruikt als taal, dit komt neer op het feit dat Kotlin de types van de objecten zal controleren tijdens het compileren van de code en niet tijdens het uitvoeren. Andere talen die dezelfde strategie volgen zijn Java en C. Alsook zal Kotlin gebruik maken van type-inferentie, hiermee zal de taal zelf onderscheid kunnen maken tussen datatypes van bepaalde expressies.

A.2.2 Cross-platform en native

Allereerst moet de omschrijving van een platform besproken worden, een platform zal meestal een bepaalde programmeertaal, besturingssysteem of bepaalde hardware beschrijven. Dit kan ook een combinatie van meerdere zijn.(Bishop & Horspool, 2006) Hiervan zijn enkele voorbeelden Java SE 15, hierbij wordt de taal bedoeld als platform. Een voorbeeld van een platform als besturingssysteem is bijvoorbeeld Windows 10 of MacOS Big Sur. Een voorbeeld hardware als platform is bijvoorbeeld een Intel of AMD processor. Uiteindelijk kan er ook gezegd worden dat een computer met als besturingssysteem Windows 10 en een laatste generatie Intel processor ook een specifiek platform is.

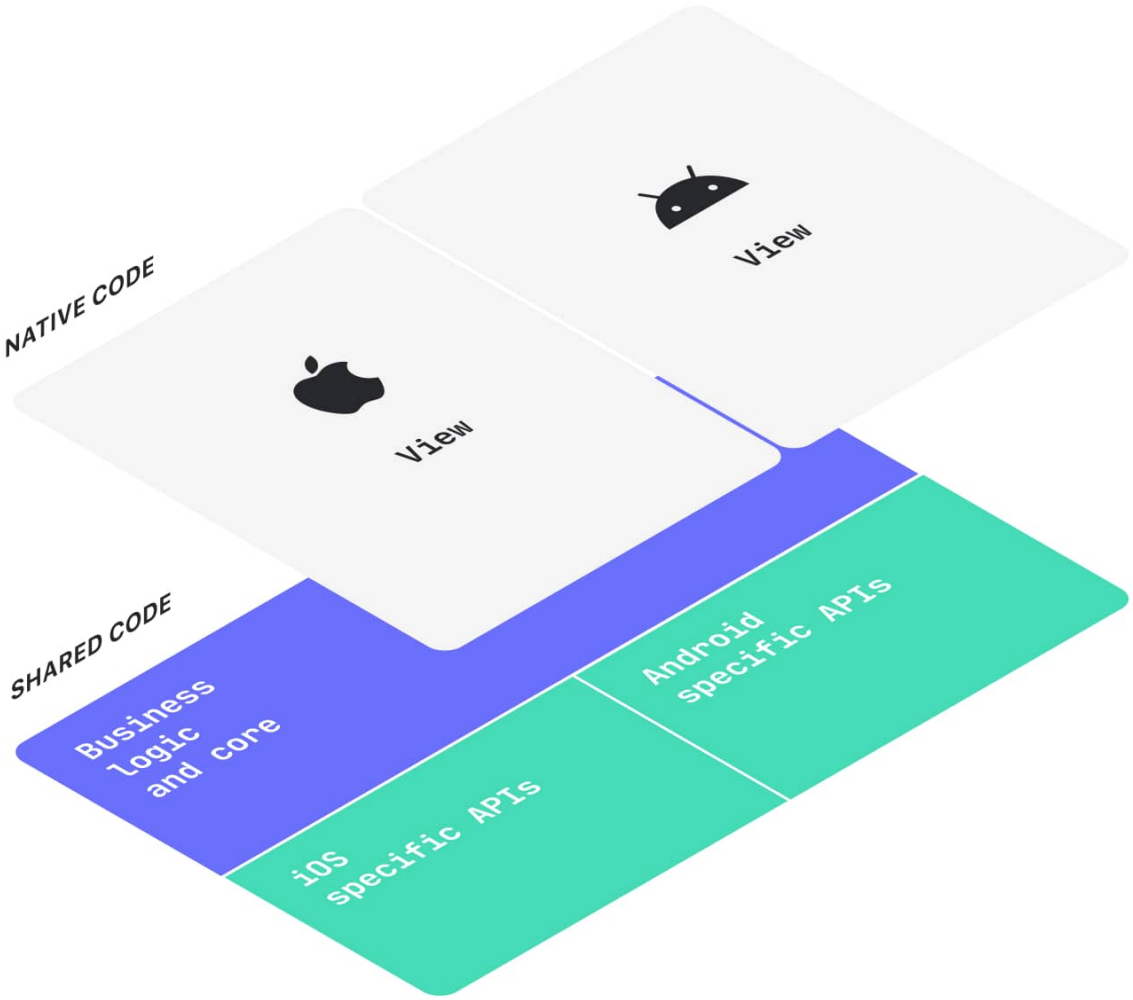
Eens de term platform duidelijker is, kan er makkelijker omschreven worden wat precies een cross-platform applicatie is en wat een native applicatie is. Een cross-platform applicatie is zoals de naam het al zegt een applicatie die op meerdere platformen zal werken. De software zal dus licht verschillende versies hebben die op hun beurt op allerlei verschillende platformen zullen werken. Native applicaties daarentegen zullen zich toespitsen op één bepaald platform.
ă

Verder wordt er SWOT-analyse bekeken voor cross-platform applicaties. Deze sterkte-zwakteanalyse zal bekijken wat de sterktes en zwaktes zijn maar ook een beeld schetsen van de kansen en bedreigingen. Hiermee kunnen we een beeld schetsen wat de huidige positie is van cross-platform applicaties binnen de huidige markt. Uit onderzoek van Tommi Nivanaho naar cross-platform applicaties met React Native zijn een aantal factoren naar voor gekomen.(Nivanaho, 2019) Hier moet wel rekening gehouden worden met het feit dat sommige factoren niet toepasbaar zijn op alle cross-platform SDKs en toolkits. Hieronder een overzicht van enkele punten uit de studie toepasbaar zijn op cross-platform applicaties in het algemeen.

- Sterktes of Strengths
  - Sneller te ontwikkelen
  - Kostenbesparend
  - De applicatie ondersteunt meerdere platformen tegelijkertijd
- Zwaktes of Weaknesses
  - Nog steeds nood aan native code per platform
  - Upgrades kunnen omslachtiger zijn
- Kansen of Opportunities
  - Snelle ontwikkeling voor meerdere platformen tegelijkertijd
  - Native applicaties kunnen relatief vlot omgevormd worden naar cross-platform applicaties
- Bedreigingen of Threats
  - Updates aan het gekozen cross-platform systeem kunnen de reeds geschreven applicaties onbruikbaar maken

A.2.3 Kotlin Multiplatform

Kotlin Multiplatform is de nieuwe cross platform software development kit (SDK) van JetBrains. Ook werd een SDK uitgebracht specifiek gericht op de mobiele toestellen namelijk Kotlin Multiplatform Mobile (KMM). Kotlin Multiplatform werd voor het eerst verwerkt in Kotlin 1.2 in november 2017 als experimentele functie.(Jemerov, 2017) Ondertussen heeft de ontwikkeling van KMM niet stil gestaan en is in augustus 2020 de alpha versie van deze SDK uitgebracht voor het grote publiek.(Petrova, 2020) In november 2020 werd de 0.2.0 versie uitgebracht, deze is verwerkt in Kotlin 1.4.20.(JetBrains, 2020) Enkele delen van deze SDK en bijhorende componenten zijn nog steeds in de experimentele fase van ontwikkeling, maar JetBrains geeft aan dat het nu al een ideaal moment is om deze software te testen.(Petrova, 2020) De community achter KMM en het open source verhaal spelen hier een grote rol en zullen ook zorgen voor een snellere ontwikkeling van deze software. Bedrijven kunnen dus momenteel al experimenteel aan de slag met deze nieuwe software en voorbeelden van enkele bedrijven die de stap al hebben gezet naar KMM zijn onder andere Netflix, VMware en Autodesk.(Kotlin, 2020)



Figuur A.1: Grafische voorstelling Kotlin Multiplatform Mobile (Kotlin & JetBrains, g.d.)

A.2.4 Kotlin Multiplatform versus alternatieven

Kotlin Multiplatform (KM) zal het concept van cross-platform anders aanpakken dan andere alternatieven op de markt vandaag. Hierbij zit het verschil vooral in welke code van de applicatie zal gedeeld worden tussen de verschillende platformen. In de Kotlin documentatie staat beschreven hoe bepaalde delen van de code correct kunnen gedeeld worden tussen verschillende platformen. KM zal anders dan alternatieven de business logica delen tussen de verschillende platformen.(Kotlin & JetBrains, 2021) Hierbij wordt ook vermeld dat men enkel de business logica moet delen die gebruikt kan worden op alle platformen.

Figuur A.1 geeft een goed algemeen beeld van de structuur van Kotlin Multiplatform Mobile. De shared code in de figuur A.1 verwijst hier naar Common Kotlin of CommonMain, dit is het gedeelde binnenin KM dat de gedeelde logica zal bevatten. Daarnaast zal er nog per platform een aparte main zijn die de code zal implementeren. Hier in de figuur A.1 zal het project ook nog een iosMain en een kotlinMain, deze kunnen later ook nog uitgebreid worden met bijvoorbeeld een macosX64Main. De specifieke situatie in figuur A.1 is

een applicatie waar Kotlin Multiplatform Mobile gebruikt wordt. Deze zal zich speciaal richten op applicaties voor iOS en Android. Voor de communicatie tussen het common deel en de platform specifieke delen zal Kotlin gebruik maken van het expected/actual systeem. Hierbij worden binnenin de CommonMain elementen gedeclareerd met expect en de platform specifieke delen zullen dezelfde elementen declareren met actual. Deze structuur kan gebruikt worden voor functies, klassen, interfaces, enumeraties, properties en annotaties.
ă
Nu er een beter beeld is geschetst van hoe KM, werkt zal er gekeken worden naar een mogelijk alternatief met een andere aanpak voor de gedeelde code, bijvoorbeeld Flutter. Flutter is een user interface toolkit ontwikkeld door Google en zit momenteel aan versie 1.22 sinds oktober 2020.(Sells, 2020) De toolkit richt zich op mobile, web en desktop applicaties en zal de code native compileren. Zoals reeds beschreven is Flutter een user interface toolkit en zal dus de user interface delen over de verschillende platformen. Hiervoor zal Flutter widgets gebruiken die geïnspireerd zijn door React.(Flutter, g.d.) Dit impliceert dus dat hierbij de business logica zal moeten verwerkt worden per platform.
ă
De developer kan dus kiezen om de user interface te delen onder de platformen en een toolkit te gebruiken zoals Flutter. Anderzijds kan er gekozen worden voor het delen van de business logica onder de platformen en dan kan KM gebruikt worden. Het delen van de user interface zal als voordeel hebben dat alle applicaties op de verschillende platformen dezelfde look en feel zullen hebben, echter is een nadeel dat de business logica per platform verwerkt zal moeten worden. Aan de kant van de gedeelde business logica is er het voordeel dat deze gedeeld is dus dat alle applicaties dezelfde logica hebben en implementeren, alsook kan er voor de user interface gebruik gemaakt worden van de platform specifieke frameworks. Er is echter ook een nadeel, hierbij kan gedacht worden aan het feit dat de user interface niet overal exact dezelfde zal zijn.

A.2.5 Testcriteria

Tijdens deze bachelorproef zullen van verschillende applicaties testcriteria geëvalueerd worden. Deze zullen ons vertellen in welke mate de cross-platform applicaties beter zijn dan de native. Volgende testcriteria werden gekozen voor deze vergelijkende studie.

- Aantal lijnen code
  - Het totaal aantal lijnen code van een applicatie zal geëvalueerd worden over het gehele project. In het geval van native applicaties zullen deze opgeteld worden met elkaar.
- Kostprijs
  - Hierbij wordt de geschatte kostprijs om een applicatie te laten ontwikkelen door een IT-bedrijf in kaart gebracht. Dit wordt berekend aan de hand van geschatte werkuren en een gemiddelde kostprijs per uur. Daarnaast wordt nagegaan of cross-platform een effect zal hebben op het systeem van vooraf bepaalde totaalprijzen indien bedrijven daarmee werken.
- Ontwikkeltijd
  - Deze tijd beschrijft het aantal werkuren dat een ontwikkelaar nodig heeft om

een specifieke applicatie te schrijven. Hierbij kan onder andere gebruik gemaakt worden van platformen zoals GitHub om deze tijd te gaan meten of inschatten.

- Compileersnelheid
  - Dit is de snelheid waarmee de specifieke applicatie zal kunnen compileren en opstarten. Dit kan gemeten worden in de ontwikkelingssoftware voor de desbetreffende taal van de applicatie.
- Voetafdruk
  - Dit impliceert de omvang die de applicatie zal innemen op het platform waarvoor deze ontwikkeld is. Hiervoor kan de applicatie gebruikt worden die de ontwikkelingssoftware aanmaakt.
- Uitbreiding van de applicatie
  - Dit criterium kan geëvalueerd worden door vooraf bepaalde features van de applicatie weg te laten. Eens de applicatie klaar is voor productie kunnen deze features terug toegevoegd worden. Om de uitbreidbaarheid van de applicatie te gaan staven kan gebruik gemaakt worden van voorgaande testcriteria.

A.3 Methodologie

Voor deze bachelorproef zullen enkele applicaties geschreven worden voor zowel Android als iOS. Eerst zullen er 2 native applicaties geschreven worden. De native iOS applicatie zal geschreven worden in Swift 5.3 of recenter met iOS 14 in gedachten en voor de user interface zal er gebruik gemaakt worden van UIKit. Aan de Android kant van het native verhaal zal er een applicatie geschreven worden met behulp van Kotlin 1.4.20 of een recentere versie. Voor de native Android user interface zal gebruik gemaakt worden van de standaard Android en AndroidX bibliotheek. Naast de native applicaties zal er nog een derde applicatie geschreven worden, namelijk een Kotlin Multiplatform Mobile (KMM) applicatie. Deze applicatie zal werken op zowel Android als iOS toestellen. Hierbij zal voor het iOS-deel gebruik gemaakt worden van Swift en SwiftUI voor de user interface en voor het Android deel van Kotlin en Jetpack Compose.

Bij deze manier van werken is het belangrijk dat zowel de native applicaties als KMM dezelfde functionaliteiten hebben. Hierbij geldt ook dat Android en iOS dezelfde functionaliteiten zullen aanbieden aan de gebruikers. De geschreven applicaties zouden dus als het ware niet van elkaar te mogen onderscheiden zijn. Hierbij dient echter rekening gehouden te worden met het feit dat de user interface wel enigszins kan verschillen.

Voor de applicaties zelf zal er gekeken worden naar de sample applicaties die JetBrains aanbiedt en voorstelt. Dit zijn onder andere de PeopleInSpace applicatie (O'Reilly, 2021) en de SpaceX lauches applicatie (Kotlin & JetBrains, 2020). Deze kunnen gebruikt worden om benchmarks op te testen en als inspiratiebron voor de native applicaties en andere KMM-applicaties.

Om de uiteindelijke vergelijking te maken tussen de native applicaties en de KMM-applicatie zal er gekeken worden naar verschillende factoren. Deze staan reeds beschreven in A.2.

State-of-the-art.

Voor dit onderzoek zullen volgende hardware en software gebruikt worden:
Hardware:

- MacBook Pro 16 inch uit 2019
- iPhone 11 Pro Max uit 2019
- Huawei P9 lite uit 2016

Deze laatste twee toestellen zullen echter minder van belang zijn en zullen enkel ter sprake komen indien er testen gebeuren op fysieke toestellen. Voor het grotere deel van de testen zullen emulators gebruikt worden die ingebouwd zijn in de gekozen ontwikkelingssoftware.

Software:

- Xcode versie 12.3 of recenter
- Android studio versie 4.1.1 of recenter

Voor de KMM-applicaties zal er echter nog de Kotlin Multiplatform Mobile plugin geïnstalleerd moeten worden.

Deze software zal gebruikt worden op de MacBook Pro die hierboven reeds werd beschreven. Voor dit onderzoek geldt de beperking dat er enkel gebruik gemaakt kan worden van toestellen die MacOS gebruiken als besturingssysteem, aangezien dit een vereiste is voor de ontwikkeling van iOS-applicaties.

A.4 Verwachte resultaten

Indien het onderzoek wordt uitgevoerd zoals beschreven in A.3.Methodologie, kunnen volgende resultaten voor de testcriteria uit A.2. State-of-the-art worden verwacht.

- Aantal lijnen code
  - Kotlin Multiplatform Mobile (KMM) zal hier een voordeel hebben in het aantal lijnen code gezien de business logica maar eenmaal moet geschreven worden. Dit is niet het geval bij de native applicaties, daar zal de business logica uitgewerkt worden voor zowel Android als iOS.
- Ontwikkeltijd
  - Aangezien de KMM-applicatie de gehele domein laag zal kunnen delen, wordt er verwacht dat deze ook sneller te ontwikkelen zal zijn dan de native varianten.
- Kostprijs
  - Verdergaand op de ontwikkeltijd kan hier ook gesteld worden dat KMM goedkoper zal zijn om te ontwikkelen. Aangezien er minder werkuren nodig zullen zijn kan ook het systeem van de totaalprijzen herzien worden. Hierdoor zullen applicaties met KMM goedkoper zijn voor de klant.

- Compileersnelheid
  - Gezien de complexe hiërarchie wordt verwacht dat de KMM-applicatie trager zal zijn dan de native applicaties. De native applicaties zijn echter ook beter geoptimaliseerd voor deze compilers en apparaten.
- Voetafdruk
  - Zoals reeds besproken zal ook de complexere hiërarchie hier ook een rol spelen. Er wordt verwacht dat de KMM-applicaties een grotere voetafdruk zullen hebben op de toestellen in kwestie, het verschil zal echter verwaarloosbaar klein zijn.
- Uitbreiding van de applicatie
  - De KMM-applicaties zullen hier ook een groter voordeel hebben omdat de nieuw geïmplementeerde features direct voor alle platformen zullen geïmplementeerd zijn. Aan de kant van de native applicaties kunnen bepaalde features vergeten of anders geïmplementeerd worden. Dit zorgt voor een verschil tussen de twee native applicaties.

A.5 Verwachte conclusies

Kotlin Multiplatform Mobile (KMM) is een zeer recente technologie die volop in ontwikkeling is. Daardoor wordt verwacht dat de resultaten op dit moment nog niet de volle kracht van deze techniek zullen aantonen. Dit wil echter niet zeggen dat de technologie geen grote meerwaarde kan bieden voor bedrijven die zich momenteel bezighouden met native applicatieontwikkeling voor zowel Android als iOS. Enkele componenten van de SDK staan nog niet op punt, dit zal waarschijnlijk nog verbeteren naar de toekomst toe. Het onderzoek kan, gezien de prematuriteit van KMM, een tijdelijke referentie bieden omtrent KMM en de mogelijkheid dit als een sneller en efficiënter alternatief te gebruiken voor native applicaties. Dit is vooral interessant voor bedrijven die zich momenteel vooral toespitsen op native ontwikkeling en eventueel naar de toekomst toe de overstap willen maken naar KMM.

Bibliografie

Bishop, J. & Horspool, N. (2006). Cross-Platform Development: Software that Lasts. *Computer*, *39*(10), 26–35. https://doi.org/10.1109/MC.2006.337

Flutter. (g.d.). *Introduction to widgets* (Flutter, Red.). Verkregen 11 februari 2021, van https://flutter.dev/docs/development/ui/widgets-intro

GitHub. (2018). *The State of the Octoverse* (GitHub, Red.). Verkregen 11 februari 2021, van https://octoverse.github.com/2018

Jemerov, D. (2011, juli 19). *Hello World* (D. Jemerov, Red.). Verkregen 11 februari 2021, van https://blog.jetbrains.com/kotlin/2011/07/hello-world-2/

Jemerov, D. (2017, november 28). *Kotlin 1.2 Released: Sharing Code between Platforms* (D. Jemerov, Red.). Verkregen 11 februari 2021, van https://blog.jetbrains.com/kotlin/2017/11/kotlin-1-2-released/

JetBrains. (2020, november 23). *KMM plugin releases* (JetBrains, Red.). Verkregen 11 februari 2021, van https://kotlinlang.org/docs/mobile/kmm-plugin-releases.html#update-to-the-new-release

JetBrains. (2021). *Corporate overview* (JetBrains, Red.). Verkregen 27 maart 2021, van https://resources.jetbrains.com/storage/products/jetbrains/docs/corporate-overview/en-us/jetbrains_corporate_overview.pdf

Kotlin. (2020). *Case Studies* (Kotlin, Red.). Verkregen 11 februari 2021, van https://kotlinlang.org/lp/mobile/case-studies/

Kotlin & JetBrains. (g.d.). *Kotlin Multiplatform Mobile* (Kotlin & JetBrains, Red.). Verkregen 11 februari 2021, van https://kotlinlang.org/lp/mobile/

Kotlin & JetBrains. (2020, november 18). *Kotlin Multiplatform Hands-on: Networking and Data Storage* (Kotlin & JetBrains, Red.). Verkregen 11 februari 2021, van https://github.com/kotlin-hands-on/kmm-networking-and-data-storage/tree/final

Kotlin & JetBrains. (2021, februari 3). *Kotlin Multiplatform: Share code on platforms* (Kotlin & JetBrains, Red.). Verkregen 11 februari 2021, van https://kotlinlang.org/docs/reference/mpp-share-on-platforms.html#share-code-on-similar-platforms

Meijer, E. & Drayton, P. (2004). Static typing where possible, dynamic typing when needed: The end of the cold war between programming languages, In *Intended for submission to the Revival of Dynamic Languages*. Citeseer. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.394.3818&rep=rep1&type=pdf

Nivanaho, T. (2019, mei 27). *Developing a cross-platform mobile application with React Native* (masterscriptie). Lappeenranta-Lahti University of Technology LUT. http://lutpub.lut.fi/handle/10024/159507

Oliveira, V., Teixeira, L. & Ebert, F. (2020, februari 18). On the Adoption of Kotlin on Android Development: A Triangulation Study, In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, London, ON, Canada. https://doi.org/10.1109/SANER48275.2020.9054859

Oracle. (g.d.). *Moved by Java Timeline* (Oracle, Red.). Verkregen 11 februari 2021, van https://www.oracle.com/java/moved-by-java/timeline/

O'Reilly, J. (2021, februari 5). *PeopleInSpace* (J. O'Reilly, Red.). Verkregen 11 februari 2021, van https://github.com/joreilly/PeopleInSpace

Petrova, E. (2020, augustus 31). *Kotlin Multiplatform Mobile Goes Alpha* (E. Petrova, Red.). Verkregen 11 februari 2021, van https://blog.jetbrains.com/kotlin/2020/08/kotlin-multiplatform-mobile-goes-alpha/

Sells, C. (2020, oktober 1). *Announcing Flutter 1.22*. Verkregen 11 februari 2021, van https://medium.com/flutter/announcing-flutter-1-22-44f146009e5f

Shafirov, M. (2017, mei 17). *Kotlin on Android. Now official* (M. Shafirov, Red.). Verkregen 11 februari 2021, van https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/

Skeen, J. & Greenhalgh, D. (2018). *Kotlin Programming: The Big Nerd Ranch Guide*. Pearson Technology Group. https://books.google.be/books?hl=nl&lr=&id=Ol9qDwAAQBAJ&oi=fnd&pg=PT14&dq=General-purpose%20programming%20language%20kotlin&ots=iivTPTKZ50&sig=Der5JeIYFyHVC2JD6xCIx-gR0Mc&redir_esc=y#v=onepage&q=General-purpose%20programming%20language%20kotlin&f=false