Onderzoek Frameworks

Opdracht	Buddy App v3
Organisatie	Superbuddy
Opdrachtgever	Tim van den Heuvel
Schoolbegeleider	Tijn Witsenburg (WGM06)

Auteurs:

Nicander Mohrmann	s1103064
Silvan Otten	s1043664
Bart van Zanten	s1104266

Hogeschool Windesheim
ICT Software Engineering

Versie: 1.0
Datum: 20-04-2018

Inhoudsopgave

1 Inleiding

1.1 Waarom gaan we onderzoeken

Vanuit de opdrachtgever is er de wens om de BuddyApp opnieuw te maken. Hierbij kan het zo zijn dat de huidige inrichting - Ionic 3.2 in combinatie met Cordova - niet meer optimaal is. Voornamelijk hoe Cordova toegang geeft tot de native components wordt als ongebruiksvriendelijk bevonden en het biedt niet 100% plugin support voor ios en android (Max Lynch 2018).

De stabiliteit en de performance van de App moeten toekomstbestendig blijven. Achterblijven op oudere versies is dan niet raadzaam. Ionic heeft al een alpha versie van 4.0 uitgebracht samen met hun eigen cross-platform API en code execution layer genaamd Capacitor.

1.2 Wat gaan we onderzoeken

Het is belangrijk dat de verschillende mogelijkheden in kaart worden gebracht. Hierdoor kan er een advies gegeven worden op basis van de requirements vanuit de opdrachtgever.

De App moet Cross-Platform beschikbaar zijn, daarom wordt gekeken of er andere frameworks zijn die dit mogelijk maken. Daarbij is het van belang dat dit op een intuïtieve manier moet verlopen.

We onderzoeken hoe de native components aangeroepen worden en doen benchmarks om de performance weer te geven.

Het huidige development team is vooral bekend met frameworks die gebruik maken van Angular, daarom vallen sommige frameworks als Xamarin buiten onze scope.

1.3 Hoe gaan we onderzoeken

Om te kijken welke framework het best aansluit op onze opdracht hebben wij ervoor gekozen om exploratief onderzoek te verrichten door in meerdere frameworks een simpele app te maken. De gemaakte app heeft een paar requirements:

- Er moet een foto gemaakt kunnen worden
- De foto moet worden weergegeven in de app na het maken
- De foto is even breed als het scherm
- Verticaal scrollable als de foto te groot is om weer te geven

Bij het ontwikkelen van de app kijken we naar hoe het opzet proces verloopt van de frameworks, denk hierbij aan framework dependencies, ondersteuning van het testen op meerdere platformen (ios, android, web) en de complexiteit van het aanroepen van de native camera functionaliteit binnen de app.

Naast de frameworks zelf wordt er gekeken naar performance, hiervoor zullen we de Android Profiler gebruiken. Wij zullen via de profiler kijken naar de CPU en Memory usage bij het opstarten van de app, openen van de camera en het tonen van de foto.

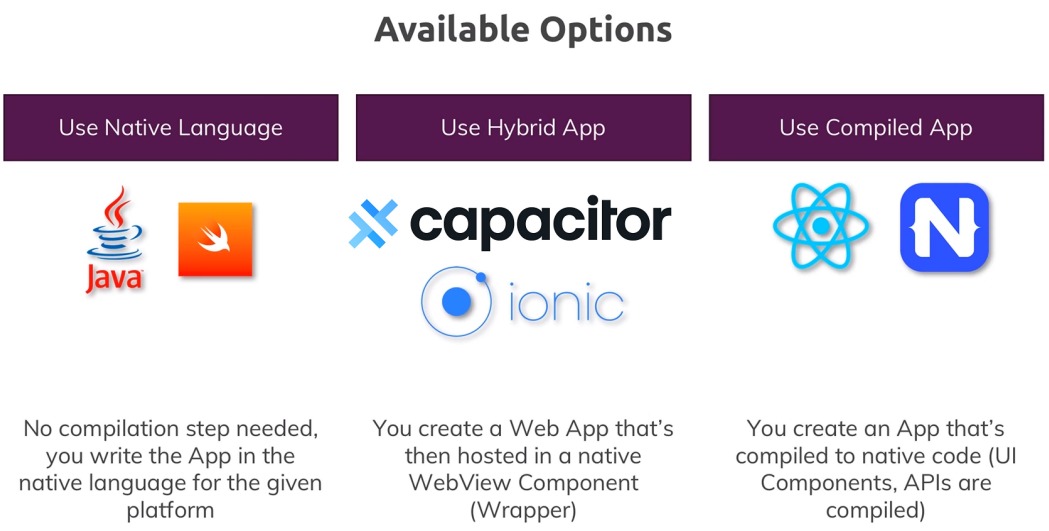
Dit wordt ondersteund door literatuuronderzoek waarbij gekeken wordt naar vergelijkingen van deskundige.

2 Het onderzoek

Wat zijn de verschillende mogelijkheden om een Web App te bouwen binnen de gestelde kaders vanuit het huidige development team?
Om die vraag te kunnen beantwoorden hebben wij twee deelvragen opgesteld en gaan we een experiment uitvoeren. De kaders worden verder beschreven in welke frameworks er zijn.

2.1 Welke frameworks zijn er?

Er zijn meerdere verschillende manieren om een app te ontwikkelen (figuur 1).
Van deze mogelijkheden vallen native languages meteen af omdat er simpelweg geen tijd is binnen de organisatie om twee talen te leren en de app in die beide talen te schrijven. Wel is het interessant om overige frameworks hiermee te vergelijken om te zien in hoeverre deze een vergelijkbare ervaring opleveren. Verdere uitleg over de beschikbare opties is te lezen in Bijlage A.



Figuur 1: Mogelijkheden om een app te maken

Aangezien het huidige development team af wilt van Cordova kijken wij naar andere opties. De frameworks die uiteindelijk het meest belovend overblijven zijn: Ionic 4 met Capacitor, React Native en NativeScript.

2.2 Wat zijn de verschillen?

Op basis van de meest belangrijke aspecten volgen in Figuur 2 individuele spectra waarop te zien is waar de verschillende frameworks zich bevinden.

2.2.1 Java / Swift

De twee native talen hebben voornamelijk als nadelen dat ze niet herbruikbaar zijn voor andere platformen. Je moet dus én meerdere talen in je dev team beheersen én meerdere keren code schrijven.

2.2.2 Ionic 4 / Capacitor

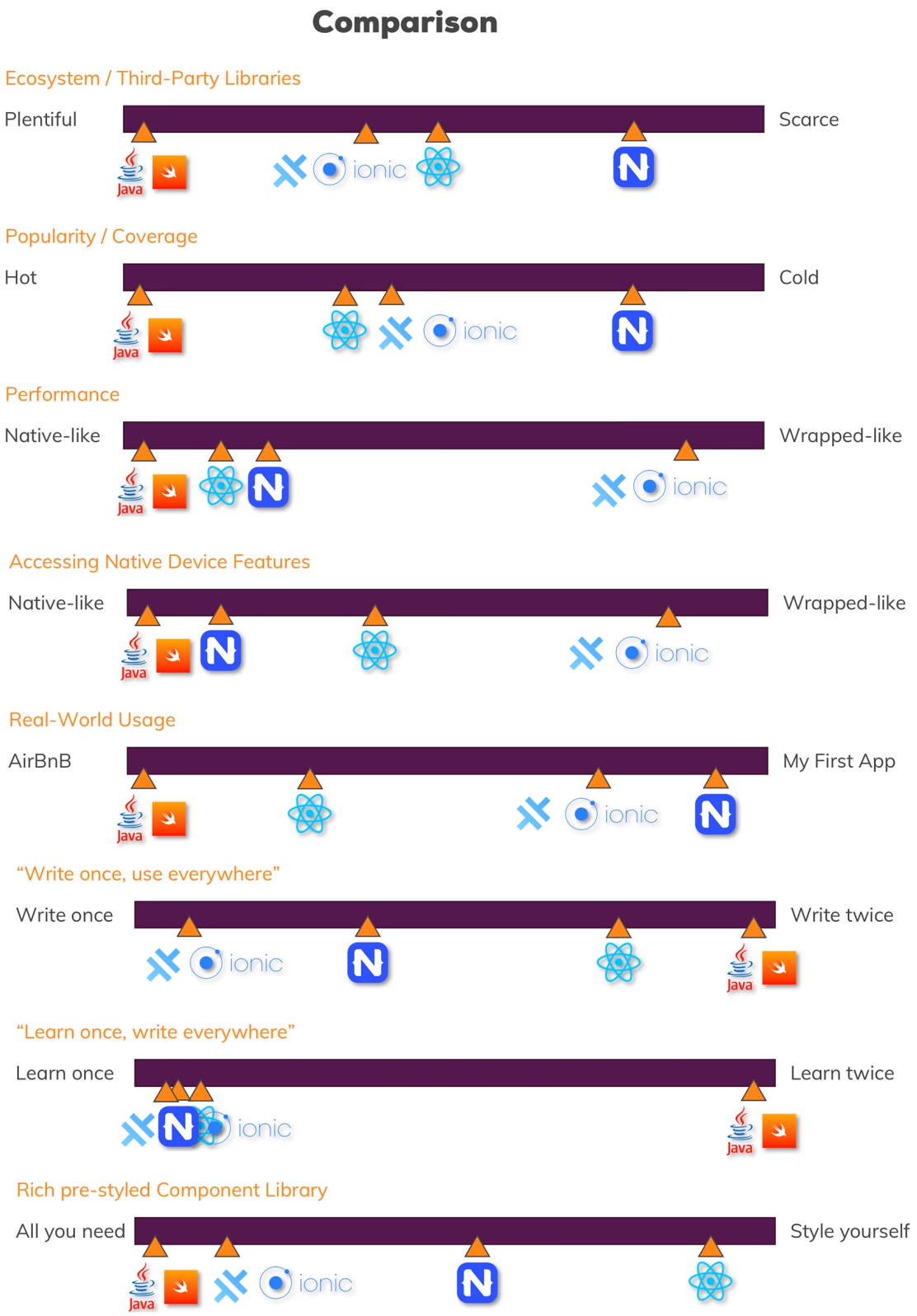
Deze combinatie geeft je de mogelijkheid om de vruchten van webtech te plukken, maar tegelijkertijd native components te gebruiken zonder teveel performance verlies. Libraries zijn goed onderhouden en redelijk beschikbaar, zeker omdat je ook gebruik kan maken van JS libraries die dom bewerkingen doen. Om het native-like te maken moet er meer aandacht aan gegeven worden en het is zeker mogelijk, maar deze geven standaard al redelijk binnen een native-feel. Daarbij komt dat er wel voldoende libraries zijn om uit te putten. Met capacitor moet het aanroepen van native components nog makkelijker worden.

2.2.3 NativeScript

NativeScript heeft vooral minder libraries en minder populariteit waar een valkuil in zit dat het niet volledige documentatie over hoe je dit kan gebruiken beschikbaar is. Daarbovenop komt nog dat Angular en TypeScript beiden gebruikt worden in de NS community waardoor er verwatering van ondersteuning is. Het is wel sterk in het gebruiken van native components en geeft het een native feel.

2.2.4 React Native

React Native wordt onderhouden door Facebook’s product infracture engineering’s team. Je ziet dat zij hierdoor op verschillende spectra telkens een andere lijn trekken. Het heeft goede toegang tot native components en voelt redelijk native, maar om verdere styling toe te passen is veel extra werk nodig aangezien er weinig in libraries te vinden is.
Je ziet ook dat native components op verschillende manieren aangeroepen kunnen, welke niet altijd even goed samenwerken. Hierdoor zit er een redelijke learning curve in.



Figuur 2: Aangepast overgenomen uit React Native vs Ionic vs NativeScript vs Android/ iOS Native Apps, van Academind (2018)

2.3 Experimenteel onderzoek

2.3.1 Ionic 4 / Capacitor

Opzetten van Ionic 4 met Capacitor was goed te doen. Ionic 4 geeft de mogelijkheid om direct een app te builden met een keuze uit een aantal bestaande templates. Na het opzetten van de basis app moest ik zelf nog Capacitor toevoegen aan het project. Hierbij moet je wel opletten dat je bij het opzetten van Ionic 4 aangegeven hebt dat je geen cordova wil gebruiken in het project.

Om de camera functionaliteit toe te voegen moet je een node package toevoegen aan het project die camera functionaliteiten kan aanroepen, in mijn geval was dit cordova-camera-plugin. Verder was het noodzakelijk om Android Studio op te zetten zodat je android kan draaien op je telefoon, dit houd in het installeren van Android Studio en het installeren van de benodigde android sdk modules.

Ionic 4 werkt fijn, het is erg handig dat het ontwikkelen en het builden van de app gescheiden van elkaar is en je meer zeggenschap hebt over hoe je je app wil builden aangezien Ionic dit niet voor jou doet.

2.3.2 NativeScript

Het opzetten van de omgeving voor NativeScript ging redelijk soepel. Het enige waar problemen ontstonden was met de tool die gebruikt wordt om de app te builden, deze tool, NativeScript Sidekick, werkte vaak niet en liep geregeld aan vast. Zonder gebruik te maken van deze tool, door de app te builden via de commandline, werkte het verder prima en was het goed te gebruiken.
Er was genoeg informatie te vinden op het internet om een basis-app te bouwen met een camera functie. Buiten de externe site om waar mensen informatie aanbieden, is er een makkelijk te doorlopen tutorial op de site van NativeScript zelf.

2.3.3 React-native

Om met React-native je eerste App draaiend te krijgen op je telefoon kun je redelijk gemakkelijk guides volgen die je op de juiste weg sturen. Het builden zelf en het draaien van de App ging gemakkelijk met behulp van Expo, een open source toolchain. De error afhandeling werkte niet geheel lekker en was niet altijd even duidelijk of het een runtime noch buildtime error omvatte.
Om daadwerkelijk de camera een foto te laten maken waren behoorlijk veel omslachtige handelingen nodig, van het toevoegen van verschillende node modules tot het instellen van je app config. Hierdoor voelde het niet heel vanzelfsprekend hoe je ermee om kon gaan en de online ondersteuning verschilt telkens vanuit welke library je een component probeert aan te roepen. Hierdoor liep het uiteindelijk redelijk in de soep en werkte het niet behoorlijk.

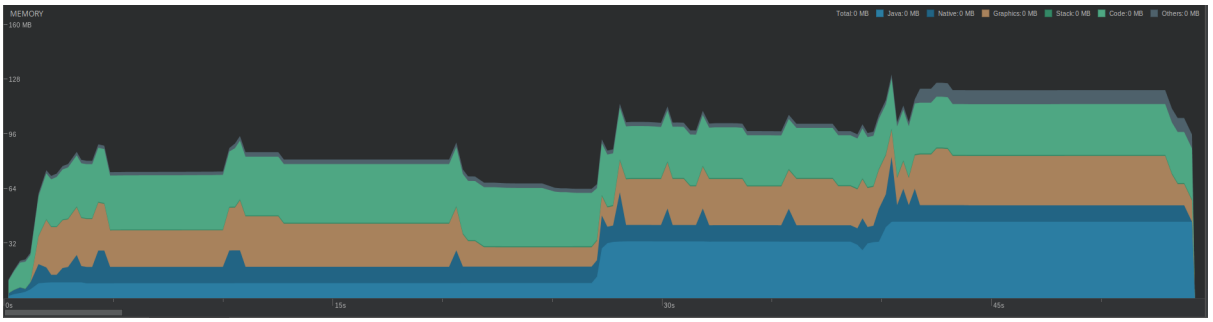
2.4 Performance

Voor het testen van de performance is dezelfde telefoon gebruikt voor beide aps. Dit betreft een Motorola Moto G (2nd gen) waar android 6.0 op staat. De telefoon bevat 1GB RAM en heeft een display formaat van 1280 bij 729.

2.4.1 Ionic 4 / Capacitor

2.4.1.1 Memory usage

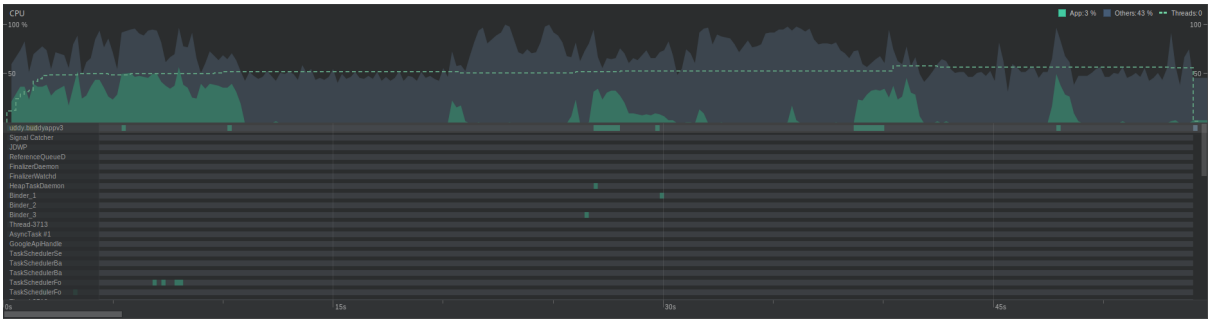
Aan het begin in de tijdlijn van de memory usage gaat het gebruik ervan langzaam omhoog, met een piek van 90 MB. Na zo'n 5 seconden is de app volledig opgestart en is de memory usage gedaald naar 74 MB. Daarna stijgt de memory usage naar 94MB en dit heeft te maken met het inladen van de view. Daarna blijft de memory op 81 MB. Na zo'n 20 seconden wordt de camera opgestart, op dit moment gaat de memory weer omhoog naar 88MB. Na het openen van de camera ontstaat er aantal spikes in memory usage dit komt omdat de app twee keer wisselt tussen de app en de camera zelf. Tijdens het maken van een foto is er een spike van 130 MB, na deze memory spike is het process van een foto maken afgerond, Hierna blijft de memory usage op 121 MB. Dit komt omdat in de app de foto weergegeven wordt.



Figuur 3: Memory usage Ionic 4

2.4.1.2 CPU usage

Evenals de memory usage stijgt de CPU usage langzaam tijdens het opstarten van de app. met een piek van 56% CPU usage. Na het afronden van het volledig opstarten van de app en het weergeven van de view daalt de CPU usage naar 0%. De CPU usage stijgt weer wanneer de camera geopend wordt. Tijdens het weergeven van de foto gaat de CPU usage weer omhoog en daalt deze direct als de foto weergegeven is.

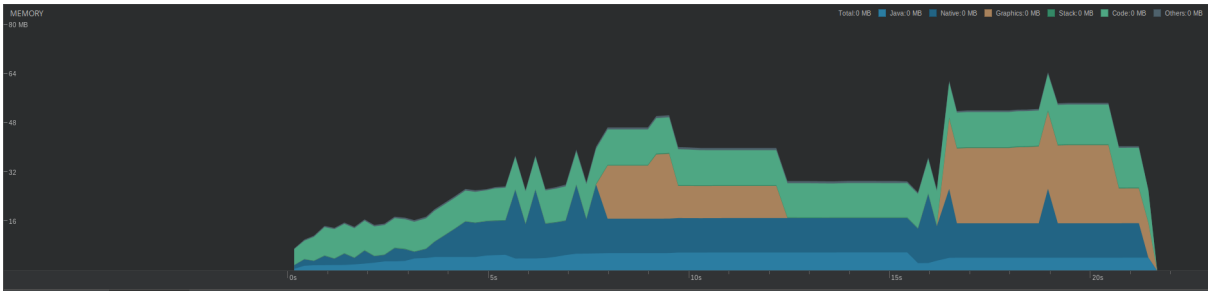


Figuur 4: CPU usage Ionic 4

2.4.2 NativeScript

2.4.2.1 Memory usage

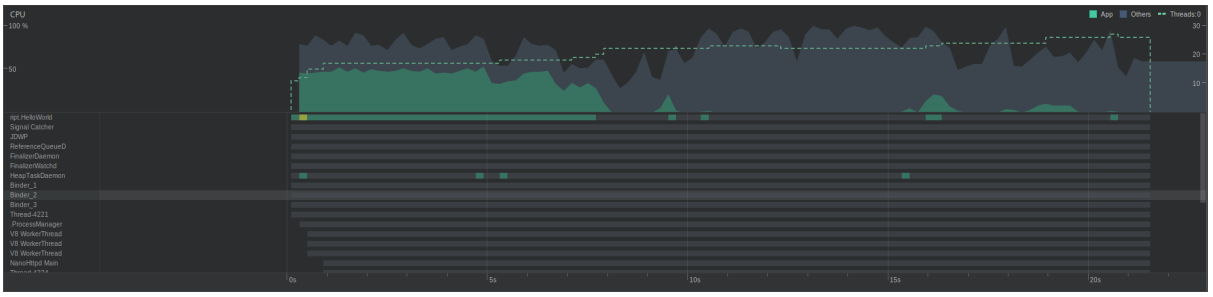
Bij het opstarten van de app stijgt de memory usage langzaam en eindigt op een piek van 50 MB. Op dit moment is de app volledig geladen en heeft het 9 seconden geduurd. Hierna daalt de memory usage naar een stabiele 40 MB memory usage gevolgd door nog een daling van memory usage naar 30 MB. Dit heeft te maken met het openen van de camera. Na het maken van een foto gaat de memory usage omhoog naar 60 MB. Deze daalt dan weer naar een stabiele 52 MB, op dit moment is de afbeelding weergegeven in de app.



Figuur 5: Memory usage NativeScript

2.4.2.2 CPU usage

De CPU usage bij het opstarten van de app ligt rond de 50%, deze daalt dan vervolgens naar 0% en stijgt weer naar 20%. Op het moment dat de CPU usage stijgt naar 20% is de app volledig opgestart. Hierna zien we weinig cpu gebruik met als 20% als hoogste gebruik, op dat punt is de foto weergegeven.



Figuur 6: CPU usage NativeScript

2.4.3 Vergelijking

Voor het vergelijken hebben we vooral gekeken naar de memory / cpu usage en de opstart tijd. De opstart tijden tussen de Ionic 4 app en de NativeScript app verschillen van elkaar met zo'n ongeveer 4 seconden. De Ionic 4 app is de snellere van de twee met een opstarttijd van ongeveer 5 seconden. Echter is de algemene memory usage van de Ionic 4 app aanzienlijk hoger vergeleken de NativeScript app. De hoogste piek van de Ionic 4 app is 130 MB en die van de NativeScript app 60 MB. NativeScript gebruikt dus aanzienlijk minder memory gedurende de hele test. Wat betreft CPU usage is het voor beide apps grotendeels gelijk. Zowel de Ionic 4 app als de NativeScript app gebruiken maximaal 50% cpu tijdens het opstarten van de app. Gedurende de test is de CPU usage niet hoger gekomen dan 20%.

3 Conclusie

Welke frameworks zijn geschikt voor dit project
React native, Ionic + Capacitor en NativeScript zijn allemaal valide opties om mee te werken voor dit project. Native valt af vanwege de wensen vanuit het bedrijf.

Wat zijn de verschillen tussen de frameworks
De grootste verschillen de frameworks is de manier waarop het build process verloopt. React native en Ionic zullen na het build process een app maken die onder water gewoon een browser is. Hieronder valt een tussen laat die communicatie regelt tussen de front end en native functionaliteiten (zoals camera). NativeScript aan de andere kant bevat richtlijnen tijdens het developing process zodat ze jou geprogrammeerde code om kunnen zetten naar native components. Dit heeft voornamelijk betrekking tot frontend onderdelen. Denk hierbij aan dat je een bepaalde tag moet gebruiken voor een button of label zodat deze direct omgezet kan worden naar de native variant hiervan.

Ionic + Capacitor en NativeScript kunnen beiden werken met Angular waar react native met react werkt. Angular heeft vanuit het bedrijf een voorkeur omdat Angular ook wordt gebruikt voor alle andere projecten.

Opzet frameworks
Alle drie de frameworks kunnen opgezet worden door middel van node packaging module. Ionic + Capacitor was het makkelijkst om op te zetten en te draaien aangezien de framework de frontend in een browser kan laten tonen. Voor React Native en NativeScript heb je android sdk (of android studio) en xcode nodig om de app te kunnen draaien. Dus de initiële opzet van React Native en NativeScript is wat lastiger.

Ontwikkelen test app
Ionic + Capacitor en NativeScript kunnen met een fresh install van de frameworks al direct de camera aanspreken, deze benut de native functionaliteiten van de camera en biedt het gene wat je kan verwachten van een camera functie in een app.

Performance frameworks
Performance wise heeft NativeScript de beste performance, de gemaakte app verbruikt zowel minder memory als cpu.

Aanbeveling
Uitgaand van de bovenstaande punten is NativeScript het best voor dit project. Ondanks het opzet process iets lastiger is dan Ionic + Capacitor biedt NativeScript een betere performance. Het bouwen van de test app ging net zo gemakkelijk als Ionic + Capacitor. Naast dat zit NativeScript niet in een alpha / beta fase zoals Ionic + Capacitor wel is.

4 Bronnen

Academind (2018)
React Native vs Ionic vs NativeScript vs Android/ iOS Native Apps
Geraadpleegd op 11 april 2018, van
https://www.youtube.com/watch?v=rb8smP_xTTY

Max Lynch (27 februari 2018)
Announcing Capacitor 1.0.0 Alpha
Geraadpleegd op 12 april 2018, van
https://blog.ionicframework.com/announcing-capacitor-1-0-0-alpha/

5 Bijlage

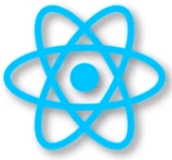
Bijlage A: uitleg over verschillende oplossingen

Java is the native language used for Android devices

Swift is the native language used for iOS devices based on Object-C.

Capacitor is a cross-platform API and code execution layer.

Ionic is a hybrid mobile SDK built on Angular implementing web technologies.



React-Native lets you build mobile apps using only JavaScript.

NativeScript is an open source framework for building truly native mobile apps with Angular, TypeScript or Javascript.