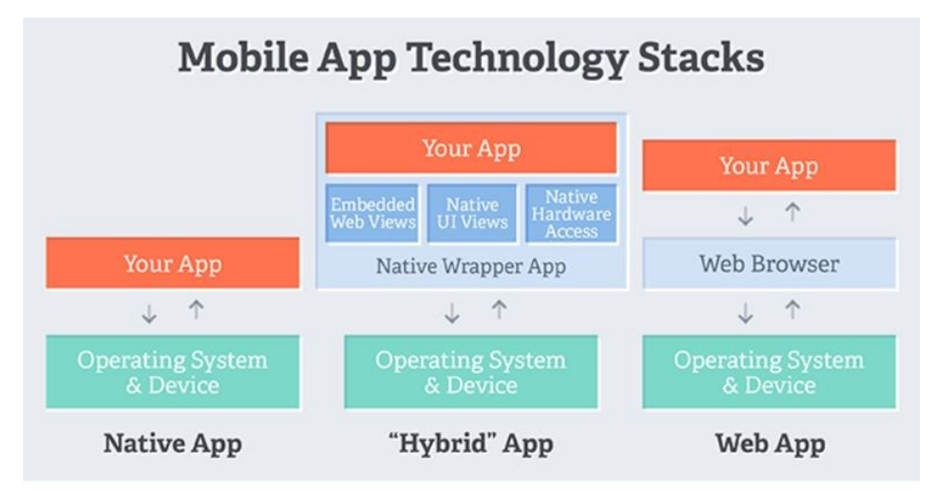
# **Welk ontwikkelraamwerk is geschikt voor het ontwikkelen van de mobiele applicatie rekening houdend met de vooraf gedefinieerde randvoorwaarden?**

Voor het bepalen van een geschikt raamwerk is het belangrijk alle aspecten en eisen mee te nemen in de bepaling hiervan. Hiervoor is eerst een lijst gemaakt met de beschikbare mogelijkheden waarmee een mobiele applicatie kan worden ontwikkeld. Vervolgens moet worden gekeken naar de sterkte en zwaktes van de verschillende raamwerken. Hierna kunnen dan ook aspecten als tooling en de ontwikkelomgeving voor ontwikkelaars. Immers staat de ontwikkelervaring van ieder raamwerk haaks aan het vinden van ontwikkelaars die daadwerkelijk ermee aan de slag willen gaan.

# **Raamwerken overzicht**

Mobiele ontwikkeling is tegenwoordig vaak onmisbaar bij het aantrekken van gebruikers of bij het implementeren van de nieuwste functionaliteiten of mogelijkheden. Onder mobiele ontwikkeling wordt vaak aan mobiele telefoons gedacht, echter gaat mobiele ontwikkeling verder dan mobiele telefoons. Hierbij kunnen ook bijvoorbeeld smartwatches, smart armbanden, tablets, convertables en vouwschermen onder vallen. De scope van dit onderzoek zal zich beperken tot mobiele telefoons die draaien op de twee grootste besturingssystemen Android en iOS.

Voor het ontwikkelen van mobiele applicaties zijn er verschillende raamwerken of technologieën beschikbaar.

Ieder van deze technologieën geeft zijn voor en nadelen. Voor het bepalen van het raamwerk of technologie zijn de mogelijkheden onderverdeeld in 4 categorieën.

### **Native app ontwikkeling**

Ten eerste is het mogelijk om “Native” applicaties te ontwikkelen. Hiermee worden de applicaties bedoeld die zijn ontwikkeld met de software die de makers van de twee populairste besturingssystemen hebben meegeleverd voor de ontwikkeling van de applicaties (Microsoft, sd).

Voordelen:

* Native applicaties zijn geoptimaliseerd en bieden een beter prestatie en gebruikerservaring (Schmitt, sd)
* Kunnen gebruik maken van alle native API’s en functies van het besturingssysteem
* Geen dataverbinding nodig

Nadelen:

* Het ontwikkelen vergt expertise van verschillende programmeertalen en ontwikkelomgevingen om voor beide platformen te ontwikkelen.
* Onderhouden van meerdere codebases is complex en tijdrovend.

Applicaties die native ontwikkeld zijn:

* Whatsapp
* Spotify
* Pokemon Go
* Waze

### **Cross platform native applicaties**

Cross platform applicaties zijn ontworpen om op verschillende besturingssystemen en apparaten te werken met een enkele codebase. Verder worden voor verschillende frameworks native elementen gebruikt om de user interface op te bouwen en om een zoveel mogelijke native ervaring te geven (Microsoft, sd) (Mosconi, 2022).

Voordelen:

* Cross-platform applicaties kunnen worden ontwikkeld met een enkele codebase.
* Kunnen op verschillende platforms worden ingezet.

Nadelen:

* Prestatie van cross-platform apps komt niet overeen met die van native applicaties.

Applicaties die crossplatform zijn ontwikkeld (Flutter, sd) (Meta, sd):

* Facebook (iOS, Android) - React Native
* Microsoft Teams - React Native
* Outlook - React Native
* Tesla - React Native
* PS app - React Native
* Discord Bloomberg - React Native
* BMW app - Flutter
* Toyota app - Flutter
* Google pay - Flutter
* Nubank - Flutter

### **Hybride mobiele applicaties**

Hybride applicaties zijn zoals crossplatform applicaties geschreven met een code base die werkt op verschillende besturingssystemen en apparaten. Het voornaamste verschil tussen hybride en crossplatform is de opbouw van de gebruikersinterface. Waarbij de gebruikersinterface van hybride apps is geschreven met web technologieën.

Voordelen:

* Hybride apps kunnen op meerdere platforms worden ingezet.
* Native functies en API’s kunnen middels plugins worden benaderd.

Nadelen:

* Prestatie van Hybride apps komt niet overeen met die van native applicaties.

Applicaties die hybride zijn:

* Bobcat - Ionic
* Southwest airlines - Ionic

### **Progressive web apps (PWA)**

“Progressive web apps” zijn webapplicaties ontwikkeld om te optimaal te kunnen worden gebruikt op mobiele apparatuur. De applicatie wordt altijd geopend door een webbrowser zonder dat daarvoor installatie nodig is. Hierbij worden geen afzonderlijke versies voor de verschillende besturingssystemen gebruikt.

Voordelen:

* Kunnen rechtstreeks vanuit het web worden geopend waardoor geen installatie nodig is.
* Kunnen op verschillende platforms worden gebruikt.

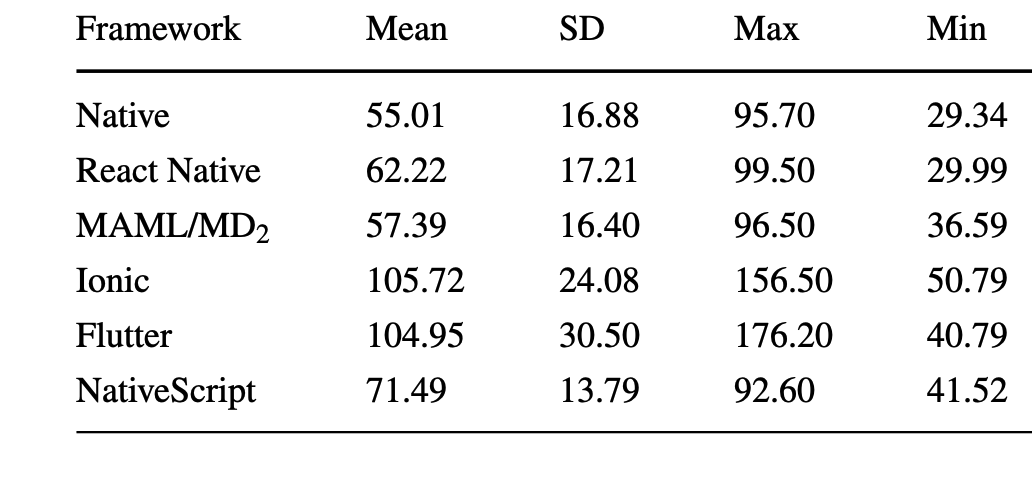
Nadelen:

* Hebben beperkte toegang tot native API’s en functies.
* Prestatie komt niet overeen met zowel native, hybride als crossplatform applicaties.

## **Memory gebruik**

Het memorygebruik van een applicatie kan een relevante factor zijn bij het bepalen van de prestatie van een framework. Door het gebruik van niet native oplossingen in de verschillende frameworks kan het RAM gebruik anders uitvallen.

In het onderstaande overzicht is het memorygebruik van een applicatie die op ieder onderzocht framework op de zelfde wijze is nagebouwd onderzocht (Biorn-Hansen, Rieger, Gronli, Majchrzak, & Ghinea, 2020). Hierbij is voor het maken van de native applicatie een Android native applicatie gebouwd. Hierbij is een onderscheid gemaakt tussen RAM-gebruik tijdens de “IDLE-state” van de applicatie en tijdens intensief gebruik.



Tabel 2 RAM-gebruik ‘IDLE-state’ (Biorn-Hansen, Rieger, Gronli, Majchrzak, & Ghinea, 2020)

Tabel RAM-gebruik 'IDLE-state'

Figuur RAM-gebruik intensief gebruik

Tabel 1 RAM-gebruik intensief gebruik (Biorn-Hansen, Rieger, Gronli, Majchrzak, & Ghinea, 2020)

Uit het overzicht is te zien dat de Native oplossing onder de applicaties met vergelijkbare functionaliteit en prestatie wat betreft de UI consistentie het laagste RAM-gebruik ondervindt. Hierbij zijn een aantal grote uitblinkers zoals Flutter en Ionic te onderscheiden die beduidend meer RAM-gebruik vertoond. Hoger ram-gebruik zou kunnen leiden tot vertraging op mobiele appraten die over weinig ram beschikken. Ook leidt hoger RAM-gebruik tot meer energieverbruik.

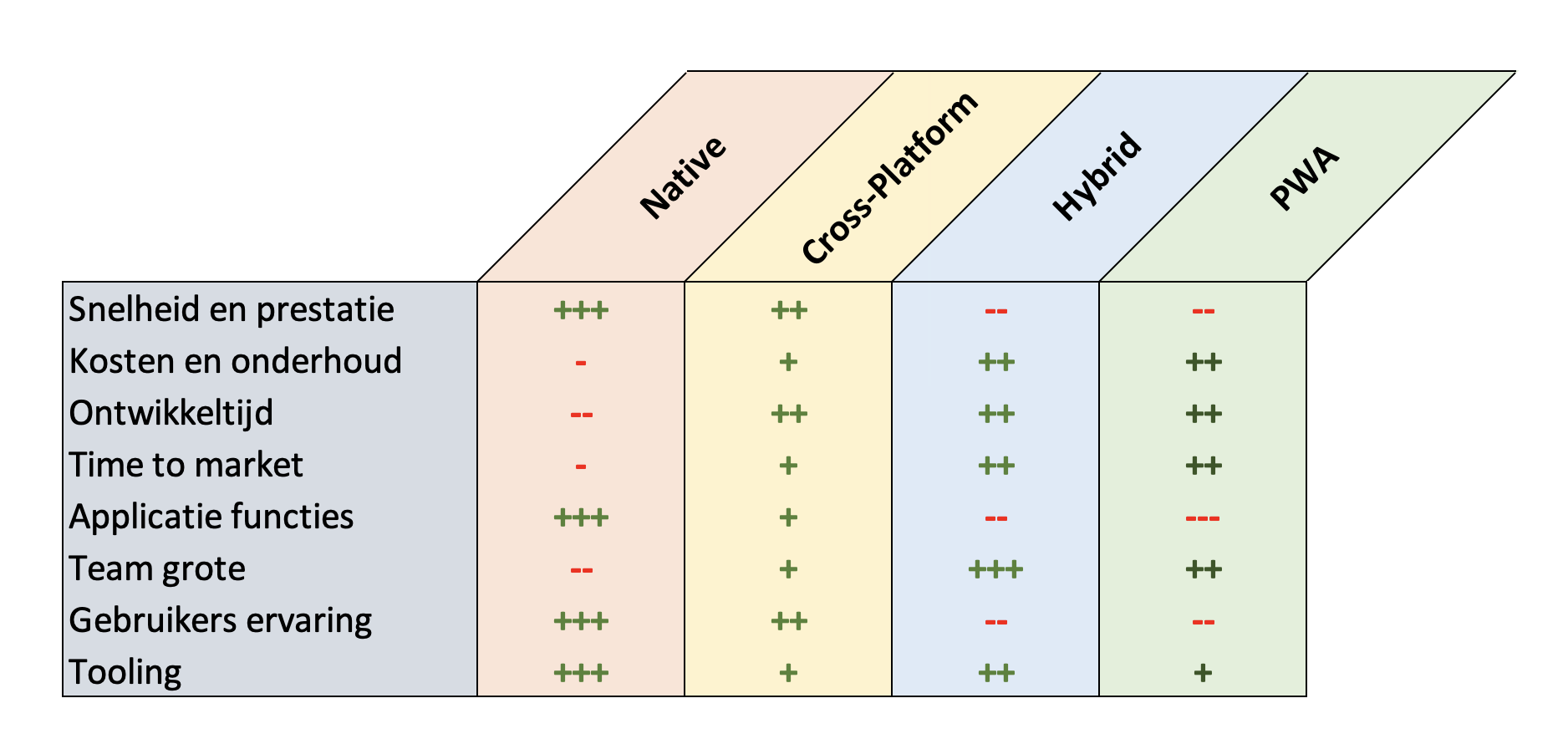
## **Overwegingen Homecooked**

Voor het bepalen van een geschikt raamwerk voor het ontwikkelen van een mobiele applicatie zijn er een aantal factoren die moeten worden overwogen. Onderstaand overzicht geeft de lijst aan aspecten waarop de verschillende raamwerken worden beoordeeld. Gezien de beperkte beschikbaarheid van tijd en middelen wordt aan de hand van deze aspecten een keuze gemaakt voor twee raamwerken waarin de Proof of Concepts nader worden uitgewerkt.

* Snelheid en prestatie: Hiermee wordt de prestatie van de applicatie bedoeld, zoals de snelheid van openen van de applicaties, het inzien van data en de framesnelheid die de “smootheness” bepaald.
* Kosten en onderhoud: De kosten voor het ontwikkelen en onderhouden van een applicatie spelen een belangrijk rol in de keuze van het raamwerk. Hierbij wordt een inschatting gegeven van de totale kosten voor de ontwikkeling en onderhoud.
* Ontwikkeltijd: De ontwikkeltijd wordt gemeten in manuren die nodig zijn om de applicatie als MVP ten ontwikkelen en uit te brengen.
* Time to market: De time to market geeft de tijd van ontwerp tot deployment van de applicatie weer op de twee ondersteunende platforms.
* Applicatie functies: De gewenste functionaliteit speelt een belangrijke rol bij de keuze van het ontwikkelplatform. Bepaalde functies zijn alleen beschikbaar bij een native oplossing waarbij andere vormen afvallen.
* Team grote: De grote van het team heeft een sleutelrol in het bepalen van bijvoorbeeld de time to market of de ontwikkelkosten. Er is immers aan schaarste aan ontwikkelaars.
* Gebruikerservaring: De gebruikerservaring speelt een zeer belangrijke rol in de bepaling van een raamwerk. Deze ervaring staat ook in verband met de snelheid en prestatie van de applicatie. Tevens kunnen raamwerken specifieke functies bevatten die deze ervaring verbeteren. De gebruikerservaring is tevens een belangrijke factor voor het succes van een applicatie. Native applicaties kunnen de beste gebruikerservaring bieden, terwijl native en hybride applicaties wellicht concessies moeten maken op verschillende aspecten.
* Tooling: Met tooling wordt de ontwikkelomgeving en alle support rondom het ontwikkelen van de applicatie bedoeld. De ontwikkelaarservaring speelt hierbij een grote rol. Tevens is de beschikbaarheid van libraries en raamwerken voor het testen en deployen van applicaties een belangrijk onderdeel hiervan.

## **Keuze POC’s**

In het onderstaande overzicht is een uiteenzetting van de verschillende raamwerken gemaakt. Waarbij er een puntensysteem is gehanteerd voor de verschillende onderdelen. Echter hebben de onderdelen onderling niet dezelfde weging. Zo zijn een aantal onderdelen zoals de gebruikers- en ontwikkelaarservaring zeer belangrijk voor de keuze van een raamwerk.



Op basis van bovenstaande lijst is de keuze voor het ontwikkelen van de POC’s gevallen op een Native ontwikkelde en Cross-platform ontwikkelde applicatie. De keuze hiervoor is voor een vooral toe te wijzen aan de gebruiksvriendelijkheid, ontwikkeltijd en de tooling van de twee raamwerken.

De applicatie heeft eis dat de gebruiksvriendelijkheid bovenaan staat. Immers is de eerste indruk die je van een applicatie krijgt zeer belangrijk voor het verkrijgen van klanten. Zo is een van de top 5 grootste valkuilen van een slecht presterende app de snelheid hiervan (Top 5 User Experience Issues In Mobile Apps, 2021).

# Bibliografie

Biorn-Hansen, A., Rieger, C., Gronli, T.-M., Majchrzak, T., & Ghinea, G. (2020). An empirical investigation of performance overhead in cross-platform mobile development frameworks. *Springer*.

Bohara, M. (sd). Comparison between React native and native application devcelopment platforms.

Flutter. (sd). *Flutter Showcases*. Opgehaald van Flutter: Https://flutter.dev/showcase

Majchrzak, T., & Heitkotter, H. (sd). *Status quo and best practices of app development in regional compa- ¨ nies. I*. Opgehaald van Springer.

Meta. (sd). *React native showcases* . Opgehaald van React native : https://reactnative.dev/showcase

Microsoft. (sd). *Native, hybrid, or cross-platform apps?* Opgehaald van Microsoft: https://powerapps.microsoft.com/en-us/native-vs-cross-platform-apps/

Mosconi, N. (2022, November). *Native vs Cross-platform Development: Pros and Cons*. Opgehaald van Devlane: https://www.devlane.com/blog/cross-platform-apps-vs-native-apps-pros-cons

Schmitt, J. (sd). *Cicleci*. Opgehaald van Circleci: https://circleci.com/blog/native-vs-cross-platform-mobile-dev/?psafe\_param=1&utm\_source=google&utm\_medium=sem&utm\_campaign=sem-google-dg--emea-en-dsa-tROAS-auth-nb&utm\_term=g\_-\_c\_\_dsa\_&utm\_content=&gclid=CjwKCAjwgqejBhBAEiwAuWHioMuAtyYOjAy8mgV5LCd\_x4k7oBa

*Top 5 User Experience Issues In Mobile Apps*. (2021). Opgehaald van Idealogic: https://idealogic.dev/5-top-user-experience-issues-in-mobile-apps/