In bepaalde situaties zal een PWA niet alle functionaliteiten kunnen ondersteunen om alle requirements van een project te voldoen. Een intuïtieve beslissing zou hier zijn om over te schakelen naar native applicaties en PWA’s links te laten liggen. Dit zou wel als gevolg hebben dat er veel potentiële gebruikers verloren gaan omdat de applicatie dan enkel te vinden is in de app stores en niet via de browser.

In deze sectie van het onderzoek zal er gekeken worden als er een hybride oplossing mogelijk is. Hierbij worden zoveel mogelijk functionaliteiten van de applicatie geïmplementeerd als PWA.

Vervolgens kan er een native applicatie gemaakt worden die de ontbrekende functionaliteiten implementeert. Deze native applicatie kan de functionaliteiten die al ontwikkeld zijn in de PWA implementeren als een webview. Op deze manier moet er geen redundante code geschreven worden

Data toont aan dat de meeste tijd van een gebruiker op een smartphone in native applicaties is (87%) maar van deze 87% wordt 80% van de tijd in slechts 3 verschillende applicaties gespendeerd. Terwijl er veel tijd besteed wordt in native applicaties, is het toch heel moeilijk om tijd te krijgen van de gebruiker om jouw applicatie te gebruiken.

Het is ook niet gemakkelijk om gebruikers te introduceren tot jouw applicatie, de gemiddelde gebruiker download 0 nieuwe applicaties per maand.

Echter op het web, waar slechts 17% van de tijd besteed wordt, worden er maandelijks ongeveer 100 websites bezocht. Hier is er voor bedrijven dus een kans om een gebruiker jouw platform te laten ontdekken.

<https://www.youtube.com/watch?time_continue=15&v=1QILz1lAzWY>

Hier wordt het grote voordeel van deze benadering duidelijk. Gebruikers kunnen de applicatie ontdekken en uittesten via de PWA. Als ze da applicatie interessant vinden kan een volledigere versie gedownload worden via de app-store

Deze benadering zal een aantal uitdagingen met zicht meebrengen maar biedt ook een aantal voordelen.

De grootste uitdaging zal data uitwisseling van data tussen de native app en de web app zijn. De state gesynchroniseerd houden tussen beide projecten zal niet eenvoudig zijn. Er zullen meer data requests van de backend nodig zijn om deze state gesynchroniseerd te houden.   
Het is ook belangrijk dat een applicatie een ‘native feeling’ heeft. Er zullen code optimalisaties nodig zijn om de webviews snel en responsive te laten werken.

De technologie die hiervoor gebruik zal worden is expo. Expo is een framework dat gebruikt maakt van react native, het voordeel van expo is dat de ontwikkelaar enkel javascript schrijft om een native applicatie te ontwikkelen. Bij de ‘traditionele’ react-native moet er vaak platform specifieke code geschreven worden in Java of Objective-c.

Verder in deze thesis zal er een voorbeeld uitgewerkt worden die deze methode demonstreert.

Apple maakt echter wel duidelijk dat het niet de bedoeling is om een website te maken en deze zonder toegevoegde native functionaliteiten in de app-store te plaatsen. De regelse van de appstore stellen het volgende:

*“Your app should include features, content, and UI that elevate it beyond a repackaged website. If your app is not particularly useful, unique, or “app-like,” it doesn’t belong on the App Store.”*

<https://developer.apple.com/app-store/review/guidelines/>

De google play store heeft hier geen specifieke regels over. <https://play.google.com/about/developer-content-policy-print/>

<https://developers.google.com/web/fundamentals/app-install-banners/native>

**INTEGREREN!!!!!!!!!!!!!!**