

De interactie van een progressive web application met het besturingssysteem: een vergelijkende studie en proof-of-concept

Onderzoeksvoorstel Bachelorproef 2019-2020

Tijs Martens¹

Samenvatting

Er zal onderzoek gedaan worden naar de huidige staat van progressive web applications (PWA's) en hoe PWA's gebruik kunnen maken van de functies van een besturingssysteem.

In een eerste deel van het onderzoek zal onderzocht worden wat een PWA juist is en hoe dit gemaakt kan worden. Vervolgens zal er een vergelijkende studie uitgevoerd worden tussen PWA's en native applications. Hierbij zal er nagegaan worden welke door het besturingssysteem aangeboden functies kunnen gebruikt worden door een PWA. In het derde deel van het onderzoek zal er gekeken worden hoe de verschillende mobiele besturingssystemen staan ten opzichte van PWA's. Uiteindelijk zal er ook gezocht worden naar eventuele alternatieven voor PWA's.

Dit onderzoek is relevant omdat het ontwikkelen van PWA's een stuk sneller en goedkoper is dan het ontwikkelen van native applications. Door het in kaart brengen van wat mogelijk is en wat niet, kan er beter beslist worden wanneer er een PWA kan gebruikt worden in plaats van een native application. Op deze manier kan er voor sommige bedrijven tijd en budget gespaard worden zonder kwaliteit in te leveren. De verwachting is dat voor veel use-cases een PWA volstaat en er geen ontwikkeling van native applications voor verschillende platformen nodig is.

Simon Floré van experience design studio Bothrs zal me bijstaan in dit onderzoek.

Sleutelwoorden

Webapplicatieontwikkeling. PWA — native apps — besturingssystemen

Co-promotor

Simon Floré² (Bothrs)

Contact: ¹ tijs.martens@student.hogent.be; ² simon@bothrs.com;

Inhoudsopgave

1	Introductie	1
2	State-of-the-art	2
2.1	Wat is een PWA	2
2.2	Welke functies van een besturingssysteem kan een PWA gebruiken	2
2.3	Waarom PWA's	2
2.4	Beveiliging	2
2.5	Native containers	2
2.6	Application shell architecture	2
3	Methodologie	3
3.1	Technologieën	3
4	Verwachte resultaten	3
5	Verwachte conclusies	3
	Referenties	3

1. Introductie

Op een desktop is het gebruikelijk om taken uit te voeren in de browser. Voorbeelden hiervan zijn Gmail en Google

drive. Dit is echter nog niet het geval op mobiele toestellen. Elke digitale toepassing, waarbij mobiele gebruikers het doelpubliek zijn, heeft een native application nodig. Vaak volstaat een traditionele responsive website niet omdat er essentiële functies zoals push notifications ontbreken. Dit heeft als gevolg dat gebruikers minder snel terugkeren naar jouw website. (Hiltunen, 2018)

De ontwikkeling van native applications is echter geen goedkope of snelle oplossing. Er moet gelijkaardige code herschreven worden voor meerdere platformen. Digitale agentschappen, startups en andere software-ontwikkelaars willen vaak een zo snel mogelijke service bieden aan hun klanten. Dit is momenteel moeilijk.

PWA's kunnen voor deze problemen een oplossing bieden. Bij het ontwikkelen van een mobiele applicatie zijn vaak functies, die aangeboden worden door het besturingssysteem, nodig. Voorbeelden hiervan zijn: locatievoorzieningen, offline gebruik, camera, push notifications, ...

In deze thesis zullen volgende onderzoeksvragen uitgewerkt worden:

- Welke stappen zijn nodig om een traditionele website om te vormen tot een PWA?
- Wat zijn de beperkingen van een PWA?

- Kan een PWA alle functionaliteiten gebruiken die beschikbaar zijn voor native applications?
- Hoe staan de verschillende besturingssystemen ten opzichte van PWA's?
- Welke andere technologieën kunnen er gebruikt worden om applicaties te ontwikkelen voor meerdere platformen waarbij er maar één codebase is?

2. State-of-the-art

2.1 Wat is een PWA

Een PWA is een website gebouwd met web technologieën die zich gedraagt als een native application maar waarbij er niet door het installatieproces moet gegaan worden zoals bij native applications. (Tandel & Jamader, 2018)

Een PWA is een 'enhanced website', dit is een website met een paar extra bestanden die er voor zorgen dat de site extra functionaliteiten heeft. Volgende bestanden zijn nodig om van een website een PWA te maken:

- Manifest.json – Dit is een bestand waar je enkele eigenschappen van de applicatie instelt zoals: app icoon, startpagina, kleurschema, ...
- Service worker – Dit is een bestand waarbij je zelf caching kan doen. Hierdoor kan, eens een website geladen is, de site offline gebruikt worden.

(Harris, 2017)

2.2 Welke functies van een besturingssysteem kan een PWA gebruiken

Niet alle besturingssystemen stellen evenveel functies beschikbaar die gebruikt kunnen worden vanuit een PWA: IOS and Android stellen volgende functies ter beschikking

- Notificaties op het toestel
- Locatievoorziening
- Camera
- Gebruik van de sensoren van het toestel
- Audio-output
- Betalingssystemen (Android pay voor Android, Apple pay voor IOS)
- Spraakinput
- Bluetooth (enkel Android)

Andere functies waar native applications wel toegang tot hebben, zijn niet beschikbaar voor PWA's:

- Toegang tot contacten
- Toegang tot de kalender
- Toegang tot alarmen
- Toegang tot telefoniedata
- Berichten
- Belfunctie
- Toegang tot low level hardware sensoren
- Camera (video's)
- Maximum opslag van 50Mb op IOS
- Geen widgets

(Ivano, 2016) (Destrebecq, 2019)

2.3 Waarom PWA's

De verwachtingen die een gebruiker heeft van een website of digitale toepassing zijn hoger dan ooit. Als je website niet binnen 3 seconden geladen is, zal je al 53% van de

gebruikers verliezen. Dit kan voorkomen worden door progressive web applications. Als de website éénmaal geladen is, kan de site opgeslagen worden in het cachegeheugen. Dit zorgt voor een snellere ervaring. (Google & awwards, 2017)

Gebruikers raken gefrustreerd als het gedrag van een mobiele applicatie anders is op de verschillende platformen. Dit verschil komt er omdat dit totaal verschillende codebases zijn, die vaak door verschillende teams ontwikkeld en onderhouden worden. Met PWA's wordt er één codebase geschreven die op alle platformen gebruikt wordt. Hierdoor zal het systeem op elk platform gelijkaardig werken. (Google e.a., 2019)

Volgens Google zijn er drie grote redenen om over te schakelen naar PWA's:

- Betrouwbaarheid – Door het cachegeheugen zal een gebruiker nooit op een pagina terechtkomen met een melding dat er geen internetconnectie is.
- Snelheid – Een PWA kan sneller zijn dan een gewone website door het gebruik van cachegeheugen. Een PWA kan sneller zijn dan een native application doordat het een veel kleiner bestand is.
- Aantrekkelijkheid – een PWA kan aanvoelen als een native application.

(Google, 2019)

2.4 Beveiliging

Een PWA moet altijd een HTTPS-verbinding hebben. Dit wil zeggen dat de data die tussen de client en de server verstuurd wordt, versleuteld is. (Zakir e.a., 2013) Dit is een stap in de juiste richting maar het zorgt er dus niet voor dat elke PWA een veilige toepassing is. PWA's worden vandaag gebruikt om gebruikers op te lichten. Een vaak gebruikte techniek is dat een bestaande native application wordt nagemaakt. Op deze manier proberen ze wachtwoorden en betalingsdetails te verkrijgen. (Lee e.a., 2018)

2.5 Native containers

Niet alle problemen kunnen opgelost worden via het web de dag van vandaag. Voor sommige taken zijn nog steeds native applications nodig. Als er een PWA moet gemaakt worden met de functies van een native application, kan deze PWA ontsloten worden door een native wrapper. Een nadeel hiervan is dat de grootte van een PWA drastisch verhoogt. Een ander nadeel is dat er twee of meer codebases onderhouden moeten worden, één van de site en minstens één van de wrappers. Deze websites zijn dan niet meer beschikbaar via de browser en moeten geïnstalleerd worden via een app-store. Dit wordt ook wel 'web based hybrid mobile app' genoemd. (Richard, 2019) (Ivano, 2016)

2.6 Application shell architecture

Om een snelle ervaring te bieden aan de eindgebruiker moet een applicatie opgedeeld worden in twee delen: de inhoud en de 'application shell'. De application shell is het deel van de interface die op elke pagina terugkomt. Dit kan bestaan uit achtergronden, navigatie, ... Deze application shell moet volledig lokaal opgeslagen worden zodat deze niet steeds opnieuw gedownload moet worden. (Hiltunen, 2018)

3. Methodologie

In een eerste fase van het onderzoek wordt bekeken wat er gedaan moet worden om een bestaande traditionele webapplicatie om te vormen tot een PWA. Dit houdt in dat de gebruiker deze website op zijn toestel kan installeren.

Bij het tweede luik van het onderzoek zal er een vergelijkende studie uitgevoerd worden tussen PWA's en native applications. Er zal een businesscase uitgewerkt worden als PWA en als native application, deze businesscase zal bepaald worden in samenspraak met Bothrs. Tijdens deze ontwikkeling zal er gekeken worden welke functies van het besturingssysteem gebruikt kunnen worden door PWA's en welke niet.

Na het voeren van deze twee praktische onderzoeken zullen er ook theoretische onderzoeken gevoerd worden. Er zal bekeken worden welke besturingssystemen de meeste functionaliteiten bieden waar PWA's gebruik kunnen van maken. Er zal ook in kaart gebracht worden wat de zwaktes en limiterende factoren zijn van een PWA ten opzichte van een native application. In een laatste deel van het onderzoek zal er gezocht worden naar andere technologieën waarbij applicaties kunnen ontwikkeld worden voor meerdere platformen met één codebase.

3.1 Technologieën

PWA's kunnen gemaakt worden met de tools en technologieën die gebruikt worden om traditionele webapplicaties te bouwen. Voor de onderzoeken zal voor de eerste fase HTML, CSS en JavaScript gebruikt worden. Eventueel kan er ook gebruik gemaakt worden van een javascript library zoals React. Er zijn verschillende tools beschikbaar voor het creëren van PWA's. Eén van deze tools is Ionic. Voor het maken van een native shell zal er waarschijnlijk gebruik gemaakt worden van react native of flutter. Dit zijn frameworks waarbij één codebase gebruikt wordt om apps te ontwikkelen voor IOS en Android.

4. Verwachte resultaten

Een website maken die voldoet aan de normen van een PWA zal niet veel tijd in beslag nemen. Er moeten slechts twee bestanden toegevoegd worden: het app-manifest bestand en een serviceworker. Het app-manifest bestand bepaalt de look en feel van de applicatie eens deze geïnstalleerd is. De serviceworker zorgt voor het cachen van data afkomstig van een API. Echter, om aan de normen van een PWA te voldoen, moet deze file gewoon aanwezig zijn. Als dit het geval is, kan de website geïnstalleerd worden op het toestel van de eindgebruiker en voldoet de website dus aan de normen van een PWA.

Het effectief gebruik maken van de functionaliteiten die een serviceworker kan bieden, kent een steilere leercurve. De ontwikkelaar moet verschillende, complexe concepten begrijpen en kunnen toepassen. Een voorbeeld hiervan is de levenscyclus die een serviceworker doorloopt. Dit is belangrijk voor het updaten van de inhoud van de website op basis van het cachegeheugen of de API. Om een goede offline ervaring te bieden zal het ontwerp van de webapplicatie ook herzien moeten worden. Er moet een application shell gedefinieerd worden. (Gaunt, 2019) (Osmani, 2016)

Er wordt verwacht dat er bij de besturingssystemen een duidelijk verschil zal zijn. Google, de ontwikkelaar van Android, heeft de revolutie die PWA's wel zijn, gestart. Hierdoor wordt er verwacht dat Android meer functies van het besturingssysteem open zal stellen voor PWA's. Het is daarentegen bekend dat Apple toegang tot het besturingssysteem zoveel mogelijk wil beperken. Een duidelijk verschil tussen deze twee besturingssystemen wordt verwacht. (Biorn-Hansen e.a., 2017)

5. Verwachte conclusies

Native applications zullen waarschijnlijk niet snel vervangen worden. Ze bieden nog steeds de meeste flexibiliteit. Maar native applications bouwen is tijdrovend en duur. Voor projecten met een kleiner budget of een scherpe deadline zullen PWA's de markt domineren. PWA's bieden vandaag al de snelste ervaring maar er ontbreken nog functionaliteiten. Er wordt verwacht dat dit de komende jaren zal verbeteren.

Referenties

- Biorn-Hansen, A., Majchrzak, T. A. & Gronli, T.-M. (2017). Progressive Web Apps: The Possible Web-native Unifier for Mobile Development. *SCITEPRESS*. <https://doi.org/10.5220/0006353703440351>
- Destrebecq, O. (2019, februari 12). *For your MVP: PWA or mobile application?* <https://medium.com/swlh/pwa-or-mobile-application-d26e904e814e>
- Gaunt, M. (2019). *Service workers: an introduction*. <https://developers.google.com/web/fundamentals/primers/service-workers>
- Google. (2019). *Progressive Web Apps*. <https://developers.google.com/web/progressive-web-apps>
- Google & awwards. (2017). *speed matters*. <https://www.awwwards.com/brainfood-mobile-performance-vol3.pdf>
- Google, Microsoft & Awwards. (2019). *Progressive Web Apps: The future of the mobile web*. <https://www.awwwards.com/PWA-ebook/en>
- Harris, M. (2017, oktober 10). *PWAs vs Native (aka There's A Progressive Web App For That)*. <https://www.youtube.com/watch?v=vhg01MI-8pI>
- Hiltunen, M. (2018). Creating multiplatform experiences with progressive web apps. https://www.theseus.fi/bitstream/handle/10024/157245/Hiltunen_Mira.pdf?sequence=1
- Ivano, M. (2016). Beyond Native Apps: Web Technologies To The Rescue. *ResearchGate*. <https://research.vu.nl/en/publications/beyond-native-apps-web-technologies-to-the-rescue-keynote>
- Lee, J., Kim, H. & Park, J. (2018). Pride and Prejudice in Progressive Web Apps : Abusing Native App-like Features in Web Applications. *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. <https://cps.kaist.ac.kr/papers/18CCS-PPP.pdf>

- Osmani, A. (2016, oktober 5). *Progressive Web Apps with React.js: Part 3 — Offline support and network resilience*. <https://medium.com/@addyosmani/progressive-web-apps-with-react-js-part-3-offline-support-and-network-resilience-c84db889162c>
- Richard, S. (2019, november 16). *Bridging the native app gap (Chrome Dev Summit 2019)*. <https://www.youtube.com/watch?v=JKVZMqpiY7w&feature=youtu.be>
- Tandel, S. & Jamader, A. (2018). Impact of Progressive Web Apps on Web App Development. *Interanational journal of innovative research in Science, Engineering and Technology*, 7(9), 9439–9444. <https://doi.org/10.15680/IJIRSET.2018.0709021>
- Zakir, D., Kasten, J. & Halderman, A. j. (2013). Analysis of the HTTPS Certificate Ecosystem. *IMC '13 Proceedings of the 2013 conference on Internet measurement conference*. <https://doi.org/10.1145/2504730.2504755>